Kaj je Blazor Hybrid in kako nam lahko pomaga pri nadgradnji programske opreme?

Matjaž Prtenjak,
Endava, Ljubljana, Slovenija
matjaz.prtenjak@endava.com

Blazor je Microsoftova tehnologija za razvoj spletnih strani v okolju .NET. Kot taka se seveda uporablja za razvoj klasičnih spletnih strani, ki tečejo v brskalniku in katerih koda se lahko izvaja na strežniku ali odjemalcu. Blazor Hybrid pa je Microsoftova komponenta, s pomočjo katere lahko spletne strani izvajamo znotraj obrazca v grafičnem uporabniškem vmesniku. S pomočjo tehnologije MAUI (Multi-platform Application UI) lahko potem takšna koda teče v obrazcih na različnih okoljih (Windows, iOS, Android ali Mac). V tem prispevku pa se bomo fokusirali izključno na Windows okolja in nadgradnjo obstoječih WinForms aplikacij v novejša spletna ogrodja.

Ključne besede:
Blazor Hybrid
.NET
WinForms
spletne aplikacije
namizne aplikacije
C#

Univerzitetna založba Univerze v Mariboru
ISBN 978-961-286-772-0
DOI https://doi.org/10.18690/um.feri.8.2023.11

1 Uvod

Serijo prispevkov na OTS konferenci sem začel v letu 2019 s prikazom, kako lahko nek starejši razvijalec/razvijalka, ki pozna jezike kot so C, C++, C# in podobne, začne razvijati spletne aplikacije. **Predstavil sem programsko okolje VUE.**

Po koronskem času, v letu 2022, se je situacija spremenila in osebam iz prvega odstavka se ni bilo potrebno več učiti JavaScript-a in VUE, temveč so lahko s pomočjo tehnologije Blazor razvijali spletne aplikacije v znanem okolju .NET/C#.

V tem prispevku imamo pred očmi še vedno iste osebe, ki so skozi desetletja razvile obstoječe produkte v WinForm-s tehnologiji, zdaj pa bi želeli to tehnologijo, vezano samo in izključno na Okna, vsaj malce posodobiti s spletnimi tehnologijami in **predstavil bom Blazor Hybrid**.

V tej nadgradnji lahko takšni stari, vendar delujoči aplikaciji, vdihnemo novo preobleko in začnemo uporabljati spletne tehnologije kot so HTML, CSS in JavaScript. Nakar jo lahko kdaj kasneje dvignemo še višje in tako počasi iz WinForm-s aplikacije preidemo v spletno aplikacijo.

V najavi prispevka sem omenil tudi Android aplikacijo, ki bi bila razvita z uporabo MAUI ogrodja. Vendar pa ima MAUI svoj lasten prispevek izpod peresa Alena Grande (*Neprekinjena integracija in postavitev MAUI mobilnih aplikacij*), zato se bom tukaj osredotočil na WinForm-s aplikacije in kako jih lahko s pomočjo Blazor Hybrid tehnologije nadgradimo. Je pa sicer zadeva v Androidu oz. okolju MAUI enakovredna.

1.1 Zakaj ravno WinForms

Blazor Hybrid nikakor ni vezan na WinForm-s aplikacije, je pa v njih vsekakor uporaben. Kaj je torej tako posebnega na WinForm-s aplikacijah, da jih je potrebno posebno omeniti?

WinForm-s je ena izmed najbolj priljubljenih tehnologij in je tudi tehnologija, ki je povzdignila Visual Basic ter razmah Windows aplikacij. Windows aplikacije so resničen razcvet doživele od leta 1993 dalje, ko se je pojavil Visual Basic 3.0, s katerim se je bilo preprosto povezati s podatkovno bazo in zgraditi grafični vmesnik.

WinForm-s tehnologija se je potem transparentno prenesla v .NET Framework in je tam ostala najpomembnejša. Skozi čas je Microsoft v .NET Framework sicer uvedel tudi druge tehnologije, kot so Microsoft Silverlite ali WPF (Windows Presentation Foundation), vendar nobena izmed njih ni v resnici ogrozila primata WinForm-s aplikacij.

Situacija je torej takšna, da v svetu obstaja mnogo programov, ki uporabljajo WinForm-s tehnologijo in nerealno je pričakovati, da se vse to vrže stran ter začne pisati stvari »na novo«. Uporabnejše in boljše je torej, če lahko te aplikacije nekako nadgradimo.

2 Kaj je Blazor Hybrid?

Preden opišem Blazor Hybrid, moram opisati nekaj osnov in sicer zmedo okoli poimenovanj .NET ogrodij, Blazor, MAUI in šele nato Blazor Hybrid

2.1 .NET Framework ali .NET Core ali .NET

Na žalost so v podjetju Microsoft kratico .NET uporabili tolikokrat, da se je zelo težko znajti in razumeti, kaj trenutno .NET sploh označuje. Tukaj vam bom torej zelo na hitro in poenostavljeno poskušal razčistiti zmedo s kratico .NET.

2.1.1 .NET Framework

.NET Framework je ogrodje, ki ga je podjetje MS lansiralo leta 2002, z verzijo 1.0. To je bil Microsoftov odgovor na programski jezik Java in z Javo si deli mnogo skupnih točk. Oba uporabljata vmesni jezik in programe prevajata v vmesni jezik, ki jih nato *prevajalnik ob izvajanju* (JIT – just-in-time compiler) prevede v strojno kodo računalnika.

Seveda si z Javo deli tudi praktično vse druge značilnosti, se pa razlikujeta v dveh pomembnih lastnostih:

1. Java teče na mnogih operacijskih sistemih in vedno uporablja programski jezik Java
2. .NET Framework teče samo na sistemih Windows in lahko uporablja različne programske jezike (v praksi pa predvsem C# in VB)

Čeravno je bil .NET Framework zamišljen kot ogrodje, ki bi lahko teklo v različnih operacijskih sistemih, je vedno teklo samo in izključno v Oknih. Kot zanimivost lahko omenim, da je bila tehnologija WinForms ravno ena izmed teh težav, ki so .NET Framework vezala na Okna.

2.1.2 .NET Core

.NET Framework ima nekaj odprtokodnih elementov, sicer pa je to zaprta tehnologija podjetja Microsoft. Z .NET Frameworkom se je hkrati, vendar neodvisno, razvijala tudi odprtokodna različica Mono, ki je lahko tekla na različnih operacijskih sistemih.

Na podlagi .NET Framework in Mono projekta so začeli v podjetju Microsoft odpirati tehnologijo in nekje v letu 2017 se je pojavila odprtokodna različica .NET, ki je tekla na različnih operacijskih sistemih in se je imenovala .NET Core

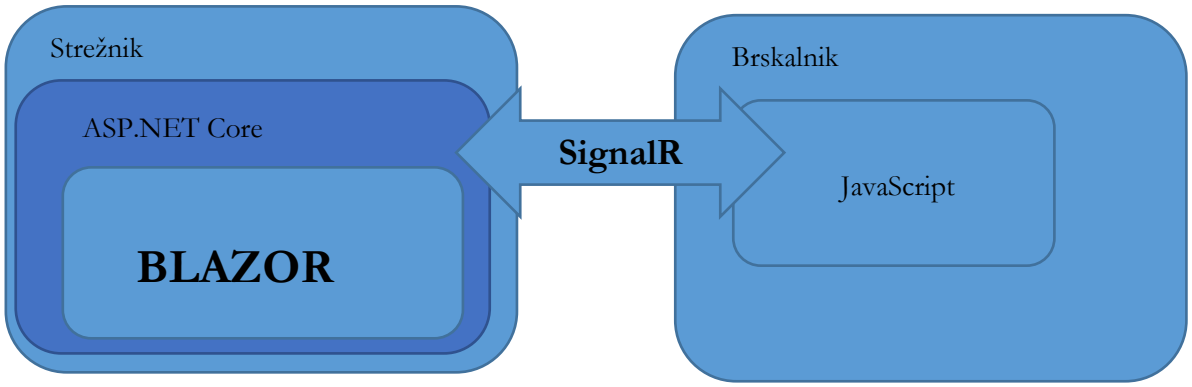
2.1.3 .NET

In da je zmeda popolna, se je potem poimenovanje spet spremenilo in iz različice .NET Core 3.1 smo prešli v različico .NET 5.0

Danes se torej uporablja samo kratica .NET in to pomeni odprtokodno različico ogrodja, ki teče na več operacijskih sistemih. Med razvijalci pa je še vedno prisotno staro ime .NET Core, kar povzroča določeno zmedo.

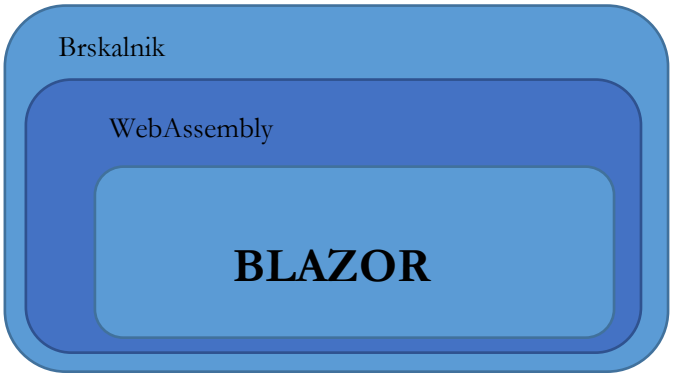
2.2 Blazor

O tehnologiji Blazor sem govoril na lanski konferenci OTS, zato tukaj ponovim samo nekaj osnov. Blazor je Microsoftova odprtokodna spletne tehnologija, kjer lahko spletne aplikacije razvijamo z uporabo C# programskega jezika. Blazor lahko teče na strežniku in se z odjemalci pogovarja preko SignalR knjižnice v realnem času, lahko pa v celoti teče tudi na uporabnikovem brskalniku.



Slika 1: Blazor Server Side.

V slednjem primeru, ki sem ga podrobneje opisoval lani, se celoten program prenese v brskalnik, kjer potem teče v WASM oz. WebAssembly-u. V tem primeru imamo torej v brskalniku programski jezik (WebAssembly), v katerega se prevede naš Blazor program in se nato izvaja v brskalniku.



Slika 2: Blazor WebAssembly.

2.2.1 Razor

Beseda Razor je zelo podobna besedi Blazor in to namenoma. Razor je namreč MS programski jezik za »nadzor« HTML kode. Blazor za obdelavo HTML uporablja Razor. Od tod v resnici tudi izhaja njegovo ime. Razor je namreč MS že imel in potem si je nad njim pač »izmislil« Blazor.

Primer Razor kode:

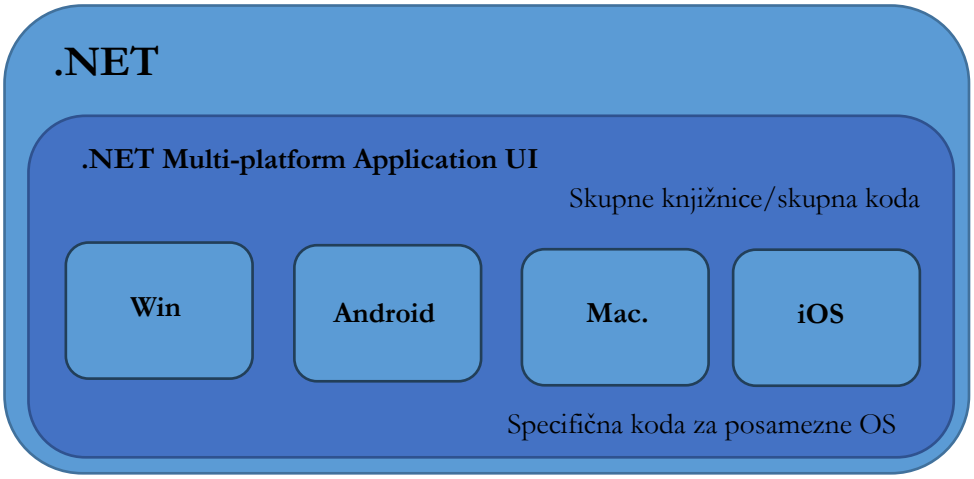
```
<div @onclick="ToogleMessage">
  @if (seen)
  {
    <span>Now you see me</span>
  }
  else
  {
    <span>Click Me!</span>
  }
</div>
```

2.3 MAUI

MAUI ali Multi-platform Application UI je Microsoftova tehnologija, s pomočjo katere lahko razvijamo programe, ki se potem izvajajo v Oknih, na Androidu, na Macintosh-u ali iOS.

Tukaj lahko takoj pogrešamo Linux, razlog pa je dokaj preprost. Pri tehnologiji MAUI govorimo o grafičnih uporabniških vmesnikih in pri Linux sistemih slednji niso standardizirani, zato jih MAUI (zaenkrat) ne podpira.

Na kratko je torej MAUI tehnologija, s pomočjo katere lahko v .NET-u razvijamo grafične uporabniške vmesnike za različne operacijske sisteme:



Slika 3: MAUI.

2.4 Blazor Hybrid

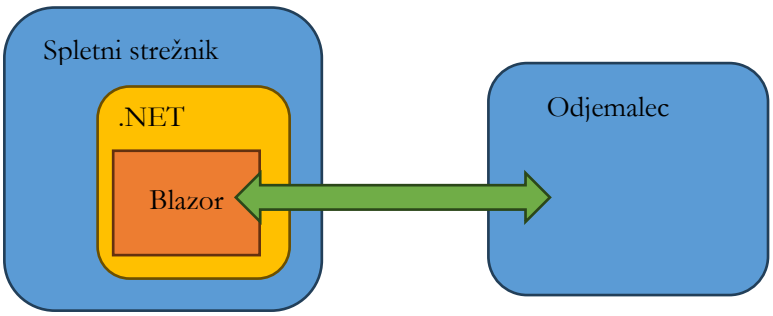
In sedaj, ko smo spoznali posamezne gradnike, se lahko povrnemo k originalnemu vprašanju: »Kaj je Blazor Hybrid?«

Microsoft na spletnih straneh pravi: »*Uporabite Blazor Hybrid za razvoj namiznih in mobilnih aplikacij z .NET in Blazor.*« To nam ne pove kaj dosti. Če na isti spletni strani beremo dalje, lahko preberemo sledeče (poskusil sem izbrati čimbolj dobeseden prevod):

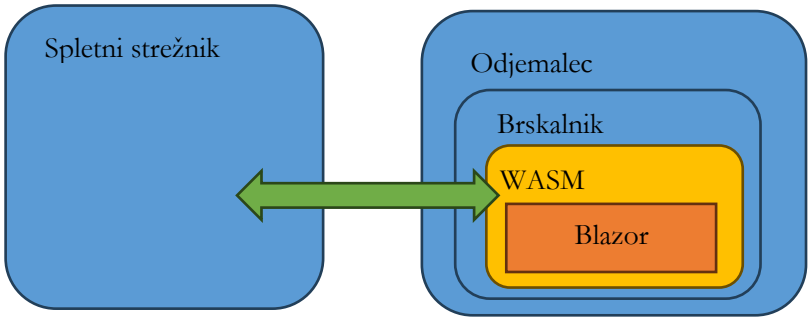
V Blazor hibridni aplikaciji se Razor komponente izvajajo na napravi sami. Komponente se prikažejo v vgrajenem spletnem pogledu (WebView) prek lokalnega vmesnega kanala. Komponente ne delujejo v brskalniku in ne vključujejo WebAssembly-ja. Razor komponente se hitro naložijo in izvedejo kodo ter imajo popoln dostop do naravnih zmogljivosti naprave prek .NET platforme. Slogi komponent, ki se prikažejo v spletnem pogledu, so odvisni od platforme in lahko zahtevajo prilagoditve zaradi razlik v prikazu med platformami z uporabo prilagojenih slogovnih datotek.

Pravzaprav je tudi iz tega kar težko razbrati, o čem je govora, zato je lažje, če se spet povrnemo na razlago o Blazor tehnologiji in se spomnimo sledečega:

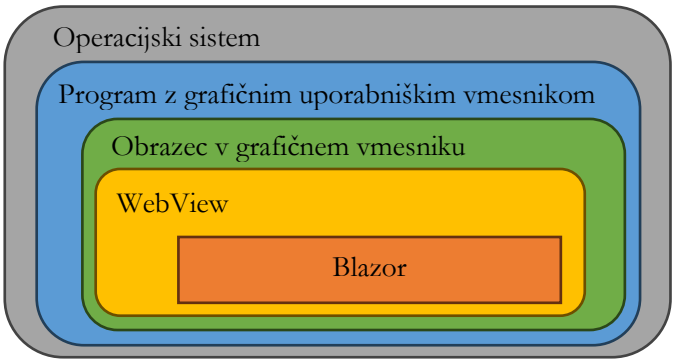
1. **Blazor Server Side** teče na spletnem strežniku in tam programsko kodo Blazor komponent izvaja .NET ogrodje, ki je instalirano na spletnem strežniku
2. V primeru **Blazor WASM**, se celoten program prenese na odjemalca in na odjemalcu WebAssembly izvaja programsko kodo Blazor komponent
3. Zdaj pa imamo še tretjo različico **Blazor Hybrid**, kjer pa Blazor izvaja posebna komponenta na vašem uporabniškem vmesniku.



Slika 4: Blazor Server Side; Blazor teče v .NET okolju, ki teče na strežniku in preko njega ima dostop do vsega na strežniku, do česar ime dostop .NET.



Slika 5: Blazor WAMS; Blazor teče v WASM (WebAssembly) in ima dostop do vsega, do česar lahko znotraj brskalnika dostopa WASM.



Slika 9: Blazor Hybrid; Ni spletnega strežnika! Blazor kodo izvaja komponenta WebView in preko nje ima Blazor dostop do vsega, do česar ima pravico dostopati program, ki teče na operacijskem sistemu.

Torej je razlika v tem, **kje teče Blazor koda in kdo jo izvaja ter posledično tudi, do česa ima dostop.**

Blazor Hybrid programska koda teče v nekem oknu vašega programa (bodisi v Windows, na Androidu, na iOS-u…) in tam posebna komponenta skrbi zanjo.

Na okno dodamo WebView komponento in potem v tej komponenti teče naša Blazor koda, ki ima tako dostop tudi do elementov obrazca, na katerem teče in preko obrazca tudi do ostalih delov sistema.

2.4.1 Prenosljivost Blazor kode

Ena izmed prednosti, ki jo avtomatično pridobimo, pa je zmožnost prenosa Blazor kode. V kolikor ne uporabljate specifičnih elementov operacijskega sistema, lahko kodo, ki ste jo razvili kot Blazor Hybrid prenesete tudi v Blazor WASM in stvar bo delovala.

3 WinForms in Blazor Hybrid

Sedaj, ko smo obdelali celoten nabor tehnologij, pa lahko preidemo na praktično uporabo Blazor Hybrid tehnologije za nadgradnjo obstoječe programske opreme, pisane v tehnologiji WinForms.

3.1 Obstoječe WinForms Aplikacije

Pri obstoječih WinForms aplikacijah lahko pričakujemo:

- da so kompleksne,
- jih je razvijalo več razvijalcev,
- ne uporabljajo priporočenih programerskih vzorcev,
- nimajo testov.

Vse to so povsem realna pričakovanja, saj tukaj govorimo o večjih aplikacijah, ki so se razvijale skozi daljše obdobje in bi jih želeli še uporabljati.

V kolikor gre za manjše aplikacije, potem je zadeva še toliko bolj preprosta, saj jih lahko pač napišemo ponovno.

3.2 Ali moramo WinForms aplikacije nadgraditi

To je povsem legitimno vprašanje in odgovor nanj je NE. Aplikacij ne potrebujemo nadgrajevati zaradi kakšnih posebnih zunanjih vzrokov. Microsoft še vedno razvija .NET Framework, knjižnice so dosegljive…

Pri nadgradnji nas torej mora voditi neka večja želja, saj nadgradnja ne bo trivialna in bo zahtevala prilagajanje in določeno mero truda.

Zakaj bi torej sploh želeli nadgraditi aplikacije? Prilagam nekaj mogočih vzrokov:

- Razvijalci imajo rajši spletne tehnologije in tudi uporabniki so jih bolj navajeni.
- Del aplikacije je lahko moderen, svež, srž aplikacije pa še zmeraj ostaja in ga ne želimo spreminjati.
- V aplikaciji si želimo več avtomatičnih testov in modernejše programerske vzorce.

3.3 Kako se lotiti nadgradnje?

Tu pa je odgovorov seveda mnogo in noben ni nujno (povsem) pravilen, kot tudi noben zagotovo ni povsem napačen. Kot vse v svetu razvoja programske opreme, je tudi to odvisno od vaših konkretnih želja in potreb.

Pot, ki jo predstavljam v tem prispevku, je pot skozi uporabo Blazor Hybrid tehnologije, saj boste na ta način dobili aplikacijo, ki bo imela (vsaj del) kode pisane v .NET Blazor tehnologiji in ko imate kodo pisano v tej tehnologiji, jo lahko lažje prenesete tudi na splet – v kolikor to seveda želite.

3.4 Nadgradnja

Bistveno je, da se zavedate omejitve, ki sem jo postavil na prvo mesto predlagane nadgradnje **in to je čim manj spremembe obstoječe kode.**

Razlog za čim manj sprememb je jasen in leži v kompleksnosti kode in pomanjkanju testov.

3.4.1 Kaj dobimo po nadgradnji

Še vedno imamo WinForms aplikacijo, ki še vedno teče v Windows okolju, le da aplikacija teče v .NET (in ne več v .NET Framework), kar pomeni, da je (vsaj v teoriji) prenosljiva.

3.5 Postopek nadgradnje

Postopek nadgradnje sestoji iz treh korakov, med katerimi prosto prehajate:

1. Priprava aplikacije na nadgradnjo
2. Nadgradnja aplikacije s pomočjo orodja za nadgradnjo
3. Popravek aplikacije po avtomatični nadgradnji

3.5.1 Priprava aplikacije za nadgradnjo

Ta korak je zelo neotipljiv in je pravzaprav odvisen od naslednjega koraka. Oba sta namreč prepletena in iz drugega koraka se boste vračali na prvega.

V drugem koraku bomo namreč uporabili orodje, ki avtomatično nadgradi WinForms Framework aplikacijo v .NET 7.0 WinForms aplikacijo. In kot vsa druga avtomatična orodja tudi slednje ni brez težav in seveda na zmore kar brez težav prenesti vse kode v .NET 7.0 (ali novejši).

Najbolj se zatakne pri nadgradnji knjižnic. V kolikor namreč uporabljate knjižnice, ki v .NET 7.0 ne obstajajo, jih boste morali zamenjati ali pa preprogramirati kodo, da jih boste zamenjali s katero drugo knjižnico.

To je tudi najtežji korak nadgradnje, ki nima magične paličice. Ta korak je poseben za vsako aplikacijo. Microsoft je pripravil oporne točke, ki jih najdete na spletni strani (Overview of porting from .NET Framework to .NET)

3.5.2 Uporaba avtomatičnega orodja za nadgradnjo

Korak 2 poteka avtomatično s pomočjo orodja in če/ko se zatakne, se povrnete v korak 1 ter spremenite zadeve ali pa spreminjate kodo v koraku 3.

V Visual Studio naložite dodatek **.NET Upgrade Assistant**, s pomočjo katerega lahko avtomatično nadgradite vaš projekt.

Kot že omenjeno v prvem koraku, vam to po vsej verjetnosti ne bo uspelo v prvem poskusu. V takšnem primeru se vrnete na korak 1, popravite/odpravite težavo in se zopet vrnite na korak 2.

Tu bi še enkrat želel poudariti, da magična paličica ne obstaja.

3.5.3 Popravek aplikacije po avtomatični nadgradnji

Tudi ta korak je podobno kot korak 1 zelo neotipljiv in je povsem odvisen od konkretne aplikacije, zato je sprehod med koraki 1, 2 in 3 poljuben ter ponovljiv.

3.6 Uporaba Blazor Hybrid tehnologije

Po uspešni nadgradnji imamo torej aplikacijo, ki namesto v .NET Framework-u teče v .NET in v kolikor je bila to edina želja, se lahko tukaj ustavimo.

Blazor Hybrid tehnologija vam pomaga samo v toliko, da lahko začnete vašo WinForms aplikacijo pisati s spletnimi tehnologijami.

Vi imate torej v tem trenutku še vedno pred sabo vašo staro aplikacijo v vsem njenem sijaju, torej delujoč program, ki ga uporabniki poznajo, toda kode se bojite, ker je stara, nima testov in bojite se jo dotikati.

3.6.1 WebView

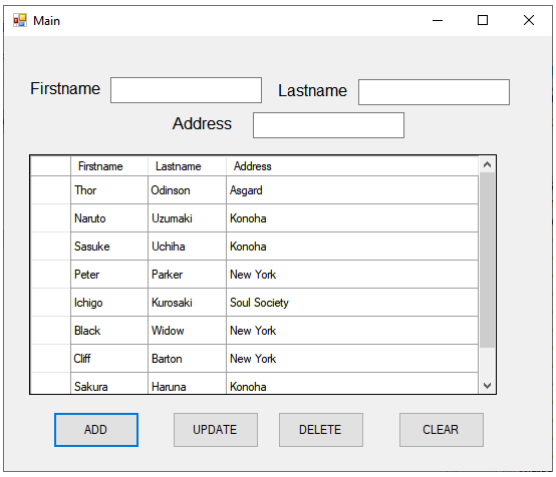
Podobno kot pri nadgradnji, tudi tukaj zadeva ni problematična s tehnološkega vidika. Najprej se morate namreč vi, kot razvijalci/uporabniki odločiti, zakaj bi želeli to uvesti in ko poznate razlog, potem je tudi lažje določiti, kam bi to uvedli.

Blazor Hybrid uporabite tako, da na obrazec dodate komponento WebView in potem v tej komponenti gostite spletne strani, ki so pisane v Blazor tehnologiji in imajo dostop do vsega, do česar lahko dostopa vaša aplikacija.

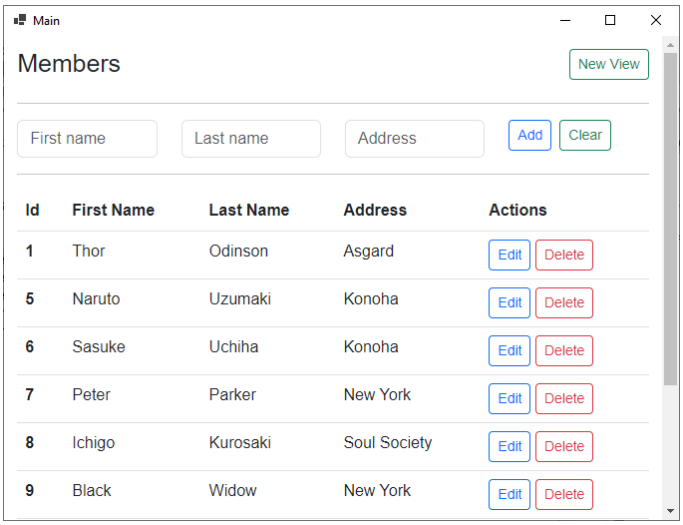
To slednje je najpomembnejše, spletna stran, ki teče v WebView komponenti ima torej dostop do vsega, do česar ima dostop uporabnik konkretnega programa na konkretnem operacijskem sistemu.

4 Praktični primer

Za potrebe povsem praktičnega prikaza nadgradnje obstoječe WinForms aplikacije v modernejšo obliko sem pripravil podrobno vadnico, kjer v 20-korakih nadgradim obstoječo aplikacijo, ki izgleda takole:



V aplikacijo, ki izgleda takole:



Celoten postopek, vključno s podrobnim opisom korakov, sprememb in podobnega najdete na GitHub spletni strani: https://github.com/MPrtenjak/WinForms_2_BlazorHybrid

Seveda pa je ob tej tranziciji pomembno, da:

- Se obstoječe kode nisem dotikal.
- Nove kode je čim manj.
- Vse akcije se še vedno izvajajo v stari kode, za katero predvidevamo, da deluje pravilno.
- S spremembo torej nismo povzročili nedelovanja ali kakšnih posebnih novih hroščev, saj tudi nova aplikacija še vedno uporablja kodo stare aplikacije.

Literatura

[1] WinForms_2_BlazorHybrid, Nadgradnja obstoječe aplikacije, https://github.com/MPrtenjak/WinForms_2_BlazorHybrid, obiskano 10.07.2023

[2] ASP.NET Core Blazor Hybrid, https://learn.microsoft.com/en-us/aspnet/core/blazor/hybrid/?view=aspnetcore-7.0, obiskano 10.07.2023

[3] Overview of porting from .NET Framework to .NET, https://learn.microsoft.com/en-us/dotnet/core/porting/, obiskano 10.07.2023

[4] Prerequisites to porting code, https://learn.microsoft.com/en-us/dotnet/core/porting/premigration-needed-changes, obiskano 10.07.2023

[5] Bring your apps to the latest .NET, https://dotnet.microsoft.com/en-us/platform/upgrade-assistant, obiskano 10.07.2023

[6] How to upgrade a Windows Forms desktop app to .NET 7, https://learn.microsoft.com/en-us/dotnet/desktop/winforms/migration/?view=netdesktop-7.0&WT.mc_id=dotnet-35129-website, obiskano 10.07.2023

[7] Install the .NET Upgrade Assistant, https://learn.microsoft.com/en-us/dotnet/core/porting/upgrade-assistant-install#install-the-visual-studio-extension, obiskano 10.07.2023