.Net Maui - framework za razvoj više platformskih hibridnih aplikacija

Kustura, Ivica

Sveučilište u Zadru

Odjel za informacijske znanosti

Sveučilišni prijediplomski studij

Informacijske tehnologije

Ivica Kustura

.Net Maui - framework za razvoj više platformskih hibridnih aplikacija

Završni rad

Zadar, 2022

Sveučilište u Zadru
Odjel za informacijske znanosti
Stručni prijediplomski studij
Informacijske tehnologije

.Net Maui - framework za razvoj više platformskih hibridnih aplikacija

Završni rad

Student/ica:
Ivica Kustura

Mentor/ica:
Marko Buterin mag. ing. inf. et comm. techn.

Zadar, 2022.

Izjava o akademskoj čestitosti

Ja, **Ivica Kustura,** ovime izjavljujem da je moj **završni** rad pod naslovom **.Net Maui – Framework za razvoj više platformskih hibridnih aplikacija** rezultat mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na izvore i radove navedene u bilješkama i popisu literature. Ni jedan dio mojega rada nije napisan na nedopušten način, odnosno nije prepisan iz necitiranih radova i ne krši bilo čija autorska prava.

Izjavljujem da ni jedan dio ovoga rada nije iskorišten u kojem drugom radu pri bilo kojoj drugoj visokoškolskoj, znanstvenoj, obrazovnoj ili inoj ustanovi.

Sadržaj mojega rada u potpunosti odgovara sadržaju obranjenoga i nakon obrane uređenoga rada.

Zadar, 12. svibnja 2022.

SAŽETAK

U ovom radu će se opisati .NET MAUI Blazor framework za kreiranje hibridnih cross-platform aplikacija s jednim izvornim kodom, koristeći C# programski jezik i XAML ili Blazor, razvijen od strane Microsoft korporacije. Bit će spomenute hibridne aplikacije i dosadašnja primjena za razvoj aplikacija na Windows platformi, Blazor kao framework za izradu web aplikacija te njegova integracija u .NET MAUI framework. Objasnit će se kako Blazor hibridne aplikacije funkcioniraju te će biti prikazan postupak razvoja cross-platform aplikacije koristeći Visual Studio razvojni alat i Blazor MAUI framework s detaljnim koracima za stvaranje takve aplikacije. Na kraju će biti istaknute specifičnosti Blazor hibridnog pristupa i mogućnost programera da razvijaju desktop i mobilne aplikacije koristeći HTML i CSS.

Ključne riječi

.Net Maui, .NET Maui Blazor, Blazor, Microsoft Hybrid Applications, Cross-Platform Applications, C#, XAML, XAMARIN Forms

Riječnih stranih riječi

Framework – Razvojna okolina ili struktura na kojoj se gradi software. Služi kao polazna točka i obično je asociran s nekim od programskih jezika.

BLC – Base Class Library – temeljni dio frameworka koji omogućuje ispravan rad Common Language Runtimea

CLR – Common Language Runtime – .NET okolina za izvršavanje koja pokreće kōd i pruža usluge koje olakšavaju proces razvoja

Library – kolekcija datoteka, programa, rutina, skripti, ili funkcija koje se mogu referencirati u programskom kodu

Runtime – Vrijeme izvođenja, je faza životnog ciklusa programiranja. To je vrijeme u kojem se program izvodi zajedno sa svim vanjskim uputama potrebnim za pravilno izvršenje. Neke od ovih vanjskih instrukcija nazivaju se *runtime* sustavi (*runtime systems*) ili *runtime* okoline (*runtime environments*) i dolaze kao sastavni dijelovi programskog jezika.

UI – User Interface – korisničko sučelje koje se iscrtava na ekranu uređaja i omogućuje korisničku interakciju s aplikacijom

Service Worker – servisni radnik je vrsta web radnika, koja je u biti JavaScript datoteka koja se pokreće odvojeno od glavne niti (*thread*) preglednika, presreće mrežne zahtjeve, predmemorira ili dohvaća resurse iz predmemorije i isporučuje push poruke.

Solution – kontejner koji Visual Studio koristi za organizaciju jednog ili više povezanih projekata.

Plugin – software-ski dodatak koji je instaliran u program u svrhu poboljšavanja njegovih karakteristika

Software – skup uputa, podataka ili programa koji se koriste za rad računala i izvršavanje određenih zadataka.

Host – fizički uređaj koji ima mogućnost dopuštanja pristupa mreži putem korisničkog sučelja, specijaliziranih programa, mrežne adrese, stoga (*stack*) protokola ili bilo kojeg drugog načina.

Sandbox – izolirana testna okolina koja omogućuje korisnicima pokretanje programa ili otvaranje datoteka bez utjecaja na aplikaciju, sustav ili platformu na kojoj rade.

Thread – kratak set instrukcija dizajniranih da se zakažu i izvrže ne ovisno o nadležnom procesu.

Stack – struktura podataka koja djeluje smo na posljednjoj dodanoj stavci, također poznatoj kao LIFO (*last-in, first-out)*
LIFO (last-in, first-out) – princip strukture podataka korištene u računalnoj znanosti. Kada se element dodaje u strukturu onda se pokazivač pomiče i pokazuje na zadnji dodani element na vrhu, a kada se dohvaća element iz strukture onda se zadnji dodani element gura s vrha i miče iz strukture. Tada se pokazivač pomiče na novi element na vrhu.

Dictionary of foreign words

Framework – Development environment or structure on which software is built. It serves as a starting point and is usually associated with a programming language.
BLC – Base Class Library – fundamental part of a framework that enables the proper functioning of the Common Language Runtime.
CLR – Common Language Runtime – .NET execution environment that runs code and provides services that facilitate the development process.
Library – Collection of files, programs, routines, scripts, or functions that can be referenced in program code.
Runtime – Execution time, a phase in the programming lifecycle. It's the time during which a program runs along with all the external instructions necessary for proper execution.
UI – User Interface – User Interface - the graphical interface displayed on a device's screen, enabling user interaction with an application.
Service Worker – A type of web worker, essentially a JavaScript file that runs separately from the browser's main thread. It intercepts network requests, caches or fetches resources from the cache, and delivers push messages.
Solution – A container used by Visual Studio to organize one or more related projects.
Plugin – A software add-on installed in a program to enhance its features.
Software – A set of instructions, data, or programs used for operating a computer and performing specific tasks.
Host – A physical device capable of allowing network access through a user interface, specialized programs, network addresses, or any other means.
Sandbox – An isolated test environment that allows users to run programs or open files without impacting the application, system, or platform they're running on.

Thread – A short set of instructions designed to be scheduled and executed independently of the owning process.

Stack – A data structure that operates only on the last-added item, also known as Last-In, First-Out (LIFO).

LIFO (last-in, first-out) – A principle of a data structure used in computer science where the last added element is the first one to be removed from the structure.

Sadržaj

1. UVOD

.NET MAUI je višeplatformski *Framework* za izradu nativnih mobilnih i desktop aplikacija koristeći C# programski jezik i XAML. U ovom radu opisuje se razvoj hibridnih aplikacija, njihove karakteristike i način rada, zajedno s ključnim značajkama .NET MAUI *frameworka*. Objašnjava se kako on radi i kako napraviti nativnu mobilnu i desktop aplikaciju, koje se mogu pokrenuti na Androidu, iOS-u, macOS, i Windowsu, s jednom kodnom bazom. Isto tako demonstrirat će se kako se koristi *Blazor WebView*, dijeljenje komponenta između web i nativnih aplikacija te proces razvoja aplikacija s pomoću .NET MAUI *frameworka* s naglaskom na izradi korisničkog sučelja s pomoću HTML I CSS *markup* jezika koji omogućuje .NET MAUI Blazor.

2. ŠTO JE .NET MAUI

.NET MAUI ujedinjuje Android, iOS, macOS i Windows API u jedan jedinstveni API koji omogućuje „write-once run-anywhere[1]“ razvojno iskustvo, sto znači da se jednom napisani kod može izvršiti na bilo kojoj platformi, dok ujedno i omogućuje pristup svim nativnim funkcijama platforme.

.NET 6 pruža niz platformski specifičnih *frameworka* za izradu aplikacija: .NET za Android, iOS, macOS, i WinUI 3 *library*. Svi navedeni *framework*-ovi imaju pristup istoj .NET 6 *Base Class Library* (BLC). Ovaj *library* apstrahira detalje osnovne platforme programerskog kōda. BLC ovisi o .NET vremenu izvođenja za pružanje okruženja za izvršavanje kōda. Android, iOS i macOS okruženje implementirana su s pomoću MONO-a. Mono je software-ska platforma otvorenog kōda dizajnirana da bi razvojnim timovima omogućila stvaranje višeplatformskih aplikacija i dio je .NET platforme. Za Windows Win32 omogućuje okruženje izvršavanja.

Dok je BLC zadužen da se aplikacije pokreću na različitim platformama sa zajedničkom logikom, različite platforme imaju različite načine definiranja korisničkog sučelja aplikacije i pružaju različite modele za određivanje elemenata korisničkog sučelja i njihovu međusobnu komunikaciju. UI se može izgraditi koristeći specifičan *framework* za svaku platformu, ali to iziskuje održavanje kōda pojedine platforme posebno za svaki uređaj. .NET MAUI pruža jedan *framework* za izgradnju UI elemenata za mobilne i desktop aplikacije.

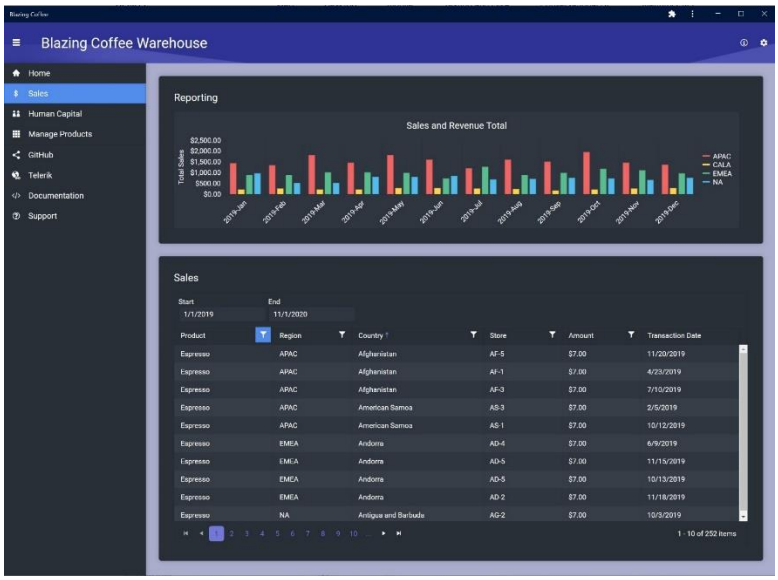
Blazor na desktopu nije nova ideja, način na koji .Net Maui pristupa razvoju višeplatformskih aplikacija je ono što ga odvaja od drugih rješenja. Postoje dva široko prihvaćena rješenja pokretanja Blazor aplikacija na desktopu uređaja. Koristeći Blazor aplikacije kao progresivne web aplikacije (*Progressive Web Applications* - PWA) ili koristeći ih unutar Elektron ljuske.

[1] https://www.codemag.com/Article/2111092/Blazor-Hybrid-Web-Apps-with-.NET-MAUI

2.1.Progresivna web aplikacija

Progresivna web aplikacija (PWA) je tip web aplikacije koja se može instalirati na operativni sustav bez potrebe za dodatnim sustavima distribucije i povezivanja. Objavljivanje u trgovini aplikacija, poput Microsoft Store, Google Play ili Apple Play Store je opcionalno i može zahtijevati dodatno povezivanje. PWA su izgrađene sa standardnim web tehnologijama koje uključuju HTML i CSS, JavaScript i *WebAssembly*, koji rade na svim standardnim web preglednicima. PWA značajke podržane su u različitim stupnjevima i na desktop i mobilnim uređajima, a Apple znatno zaostaje u usvajanju[2].

Jednom kada je PWA instaliran na uređaju, adresna traka i gumbi na web pregledniku nestaju. Kao i kod izvorne aplikacije, PWA ima ikonu za pokretanje i izvornu interakciju s programskom trakom sustava Windows. PWA dobiva bitnu značajku koja dodatno pospješuje korisničko iskustvo a to je *Service workers. Service workers* su JavaScript kōd koji se izvodi odvojeno od glavne niti preglednika koji može pružiti i *offline* mod (presretanje mrežnih zahtjeva, pred memoriranje ili dohvaćanje resursa iz pred memorije[3]) i isporučuje *push*[4] poruke. Iako PWA ne može pristupiti API-jima na razini operativnog sustava, uvelike pružaju osjećaj korištenja izvorne aplikacije više od tradicionalnih web aplikacija tako da oponašaju njihovo ponašanje.



TELERIK BLAZING COFFEE DEMO PWA APLIKACIJA 2-1[5]

[2] https://www.techrepublic.com/article/apple-could-lose-billions-on-progressive-web-apps-but-it-has-no-choice/
[3] eng. cache
[4] Odnosi se na server push metodu internet komunikacije gdje se razmjena informacije inicira iz servera i šalje na klijent
[5] https://demos.telerik.com/blazor-coffee/

Blazor *WebAssembly* aplikacije mogu iskoristiti prednosti PWA značajki jednostavnim ispunjavanjem instalacijskih kriterija PWA. PWA opcija je već dostupna za Blazor pri pokretanju novog projekta iz predloška. Iako se Blazor PWA aplikacije mogu lako izraditi, postoje kompromisi. Budući da ne postoji podrška za .NET API za *service workers,* sve funkcionalnosti moraju biti izvedene u JavaScriptu. S obzirom na to da je jedna od Blazor-ovih atrakcija C#, to odvraća neke developera da se previše upuste u *service workers*.

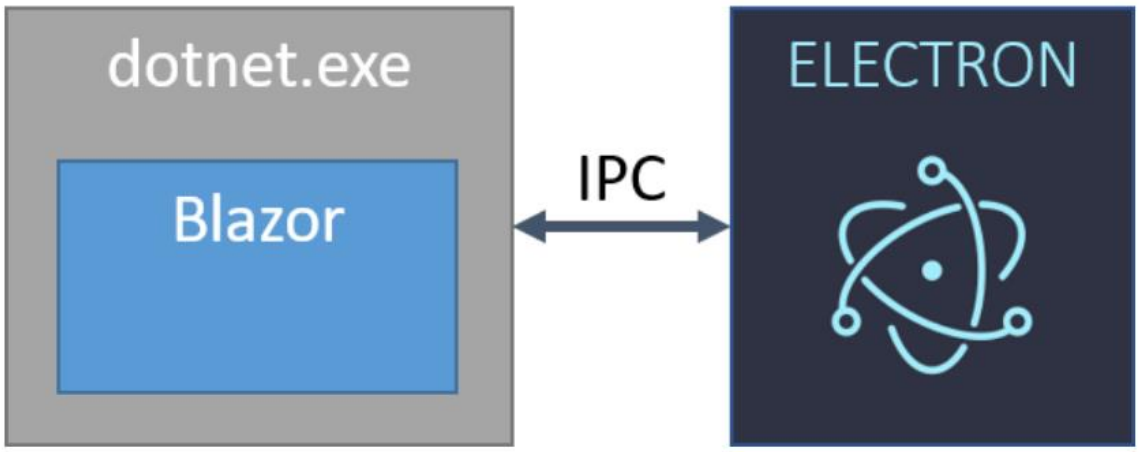
2.2.Elektron

Elektron je *open-source framework* za razvoj izvornih desktop aplikacija koristeći web tehnologije poput JavaScript, HTML I CSS. Elektron koristi ugrađeni *Chromium* omot[6] koji se pokreće s pomoću Node.js-a. Elektron omogućuje stvaranje kros-platformskih aplikacija s jednom JavaScript kodnom bazom koje rade na Windows, macOS i Linux platformama. Mnoge popularne desktop aplikacije su u biti Elektrom web aplikacije poput *Visual Studio Code, Microsoft Teams, Slack i Figma*[7].

Elektron je više od samog web omota jer također pruža prilagođen set JavaScript API-ja za interakciju s operativnim sustavom domaćina. Ovi moduli kontroliraju nativne desktop funkcionalnosti, poput menija, dijaloga i ikona na traci. Iako API pruža neke nativne funkcionalnosti, to nije sirovi pristup punoj platformi, umjesto toga, to je odabrani skup zajedničkih značajki između platforma. API-ji uključuju pristup sustavu datoteka, pokretanju procesa izvan testnog okruženja web preglednika, kiosk način rada, snimanje zaslona, podršku za minimizirane aplikacije i još mnogo toga.

Budući da Elektron radi korištenjem standardnih web tehnologija putem Chromea, podržava *WebAssembly*. To znači da se Blazor *WebAssembly* aplikacija može ugraditi u Electron ljusku i transformirati u desktop aplikaciju. Takve aplikacije koriste .NET *runtime* u kontekstu Blazor *WebAssembly* koji radi u interpretiranom modu, koji je manje učinkovit od svog desktop ekvivalenta. Korištenjem Electron-ovih API-ja s pomoću .NET-a vrši se putem Electron.NET, omotač oko JavaScript bazirane Elektron aplikacije s ugrađenom ASP.NET Core aplikacijom. Preko Electron:NET IPC mosta, Electron API-ji se pozivaju iz Blazor aplikacije.

[6] Framework otvorenog koda koji omogućuje ugrađivanje Chromium preglednika unutar druge aplikacije https://en.wikipedia.org/wiki/Chromium_Embedded_Framework
[7] https://www.codemag.com/Article/2111092/Blazor-Hybrid-Web-Apps-with-.NET-MAUI



DIJAGRAM BLAZOR ELEKTRON APLIKACIJE 2-2

Iako Blazor aplikacije mogu uspješno koristiti Electron i Electron.NET, Elektron je samo za desktop aplikacije, ne i za mobilne. Blazor aplikacije se oslanjaju na više *frameworka* od različitih autora i zajednica, a performanse nisu one koje pruža .NET koji radi na izvornom operativnom sustavu. Osim toga, pristup API-ju je ograničen na ono što se pruža unutar okvira Elektron i Elektron.NET.

2.3.Blazor hibridna arhitektura

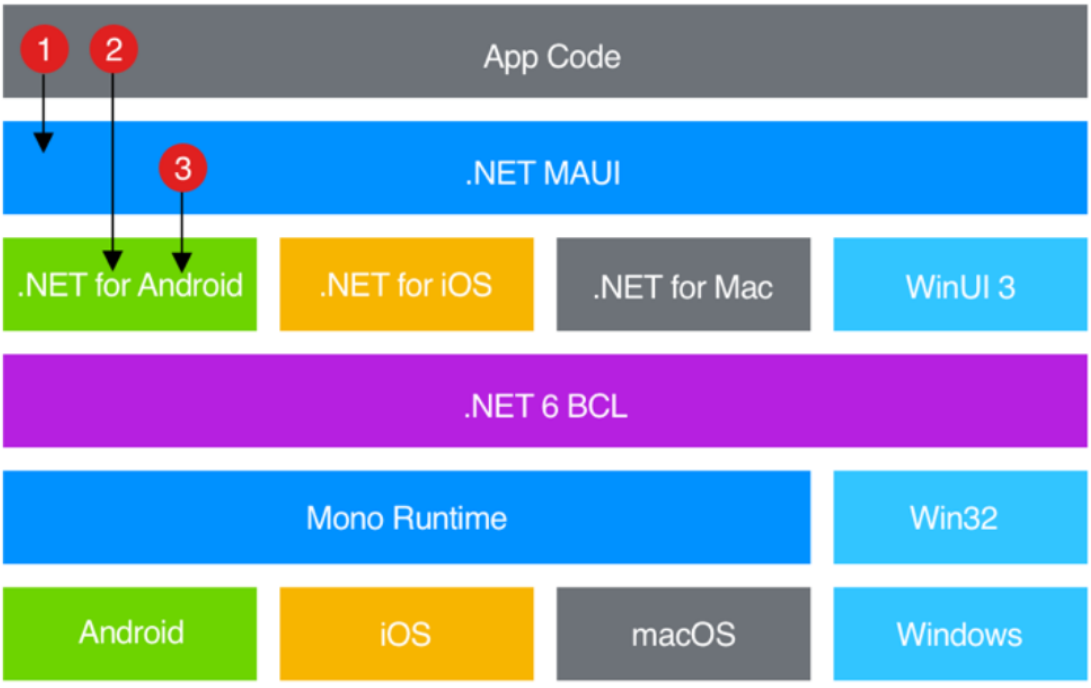
MAUI je evolucija Xamarin.Formsa, koji inicijalno cilja iOS i Android, a s MAUI-jem proširuje se i na desktop. Koristeći MAUI mogu se razvijati kros-platformske aplikacije u jednom *solution*-u s opcijom pisanja specifičnog koda za platformu po potrebi. S obzirom na to da je MAUI u potpunosti .NET, dijeljenje kōda, logike, testova, i alata moguće je kroz cijeli *solution*. Blazor Hybrid uzorak izgrađen je na MAUI-ju i implementiran kroz *Blazor WebView*, MAUI komponentu koja se koristi za renderiranje ugrađenog Blazor Web prikaza koristeći WebView2 *runtime.*

.NET MAUI koristi jedan API koji ujedinjuje Android, iOS, macOS, i Windows API-je u „write-once run anywhere[8]“ developersko iskustvo. MAUI aplikacije pružaju dubok pristup u svaku nativnu platformu. .NET 6 uvodi niz *frameworka* specifičnih za platformu: .NET za Android, .NET za iOS, .NET za macOS, i Windows UI (WinUI) Library. .NET 6

[8]Referira se na činjenicu da se jedan kod može izvršiti na bilo kojoj platformi https://www.codemag.com/Article/2111092/Blazor-Hybrid-Web-Apps-with-.NET-MAUI

Base Class Library je dijeljena među svim platformama dok se iz koda apstrahiraju pojedinačne karakteristike svake platforme. .NET *runtime* se koristi za okruženje izvršavanja za MAUI aplikacije, iako temeljne implementacije vremena izvođenja mogu biti različite, ovisno o *host*-u. Za Android, iOS, i macOS okruženje je implementirao s Mono *runtime*-om, a za Windows s pomoću WinRT.

MAUI je više od apstraktnog BCL[9]-a za dijeljenje zajedničke logike između različitih platformi, ono također ujedinjuje i razvoj korisničkog sučelja (UI). .NET MAUI pruža jedan *framework* za razvoj UI-a za mobilne i desktop aplikacije. S obzirom na to da svaka platforma ima svoj model i elemente za prikazivanje UI-a, koristiti individualan *framework* za pojedinu platformu za izgradnju UI-a bilo bi teško za održavanje. Upravo zbog toga MAUI pruža zajednički multi-platformski *framework* za izgradnju korisničkog sučelja, dok zadržava sposobnost da cilja fleksibilnosti pojedine specifičnosti platforme po potrebi. Uz izvorne UI *framework*-e, MAUI također predstavlja *Blazor WebView*. Koristeći *Blazor WebView* komponentu, MAUI aplikacije mogu koristiti Blazor Web *framework* stvarajući .NET MAUI Blazor aplikaciju.



PRIKAZ ARHITEKTURE .NET MAUI APLIKACIJE 2-3

[9] *Base Class Library*

U .NET MAUI aplikaciji kōd koji se piše primarno komunicira sa .NET MAUI API-jem (1). .NET MAUI zatim direktno konzumira API od izvorne platforme (3). Osim toga, kōd aplikacije može izravno koristiti API-je platforme (2), ako je potrebno. .NET MAUI aplikacije može se napisati na PC-u ili Mac-u i kompilirati u izvorne pakete aplikacije:

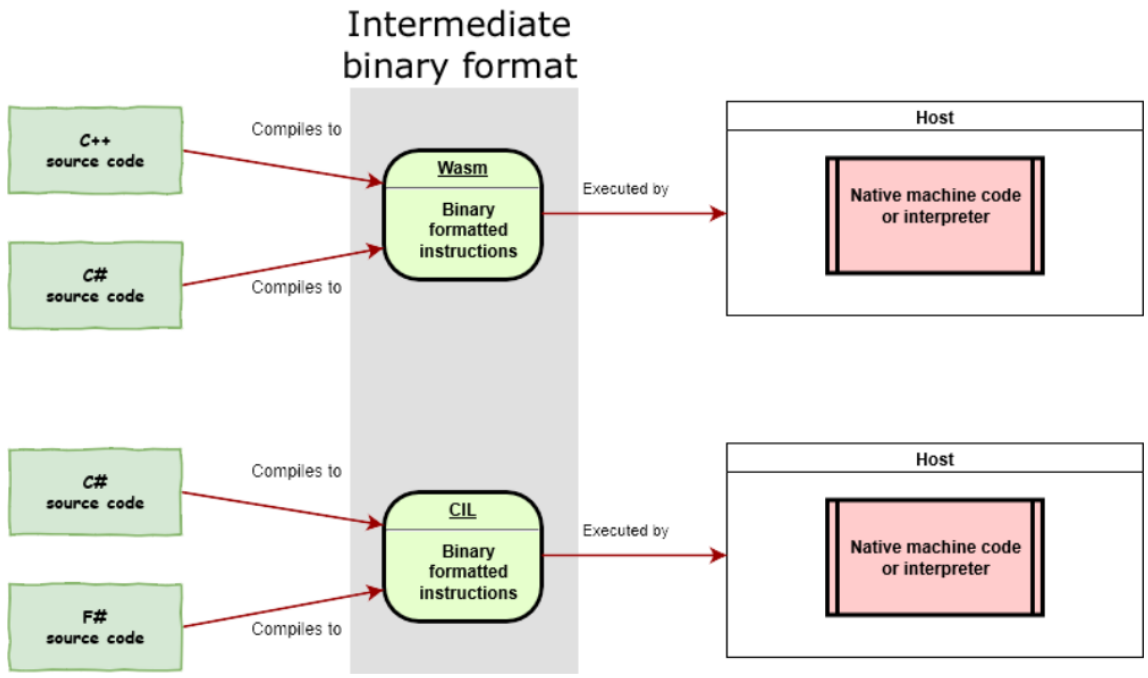
- Android aplikacije kompiliraju se iz C# u među jezik (*intermediate language* - IL) koji se tada na vrijeme (*just in time* – JIT) kompiliraju u izvorni sklop kada se aplikacija pokrene.
- iOS aplikacije su u cijelosti ispred vremena (*ahead of time* – AOT) kompilirane iz C# u izvorni ARM kōd.
- macOS aplikacije koriste MAC Catalyst, rješenje Applea koje donosi iOS aplikaciju izgrađenu s UI Kitom na radnu površinu i po potrebi je nadograđuje dodatnim App Kitom i API-jima platforme.
- Windows aplikacije koriste biblioteku Windows UI 3 za izradu izvornih aplikacija koje ciljaju na radnu površinu sustava Windows.

3. BLAZOR FRAMEWORK

Blazor je razvojna okolina za izgradnju SPA (*Single page application*) web aplikacija. Blazor nije kao prethodni Microsoftov *framework* Silverlight koji zahtijeva instalaciju *plugin*-a u pregledniku za pokretanje, što je bila prepreka da se aplikacija pokrene na iOS uređaju.

Blazor ne zahtijeva ikakav dodatak instaliran na klijentu da bi se mogao izvršiti unutar preglednika. Blazor se može pokretati na serveru, u čijem slučaju aplikacija se izvršava na serveru a web preglednik služi samo kao terminal za prikazivanje, ili se može pokretati u samom pregledniku kod klijenta koristeći *WebAssembly*.

WebAssembly (skraćeno „Wasm“) je skup instrukcija napravljenih da se izvršavaju na bilo kojem *host*-u sposobnom da interpretira takav skup instrukcija. Formatiran je u specifičan binarni format. Svaki *host* (hardverski ili softverski) koji odgovara ovim specifikacijama je sposoban čitati i izvršavati takav binarni kōd, interpretiran ili kompiliran direktno u strojni jezik specifičan za taj uređaj. Wasm je sličan uobičajenom skupu instrukcija (*Common Intermediate Language*) na koji se .NET izvorni kōd prevodi. Baš kao .NET, Wasam se može generirati iz viših jezika kao što je C#.



BLAZOR NE ZAHTIJEVA .NET NA KLIJENTU DA BI SE MOGAO POKRENUTI KROZ WEBASSEMBLY 3-1

3.1.Blazor WebAssembly

Blazor ne zahtijeva .NET instaliran na klijentu da bi se izvršavao preko *WebAssembly*-a. *BlazorWebAssembly* je SPA (*single-page application*) *framework* za izradu interaktivnih web aplikacija u .NET-u i podržan je u svim modernim desktop i mobilnim web preglednicima. *WebAssembly* kōd ima potpun pristup funkcionalnosti web preglednika preko JavaScript jezika, zvanom JavaScript interoperabilnost, skraćeno JS interop. .NET kōd izvršen preko *WebAssembly*-a u web pregledniku pokreće se unutar JavaScript *sandbox*-a sa svim zaštitama protiv zlonamjernih radnji na klijentu koje *sandbox* pruža.

Blazor je *open-source* u vlasništvu ne profitne organizacije .NET Foundation, napravljene sa ciljem da podrži *open-source* projekte bazirane oko .NET *frameworka*.

Blazor aplikacije bazirane su na komponentama. Komponenta u Blazoru predstavlja UI element poput stranice, dijaloga ili forme za unos podataka. Komponenta je uobičajeno napisana u obliku Razor *markup* stranice i ima .razor ekstenziju. Razor je sintaksa koja kombinira HTML *markup* i C# jezik i dopušta developerima da slobodno kombiniraju i HTML i C# u jednoj datoteci. Blazor koristi standardne HTML elemente za slaganje UI-a.

razor

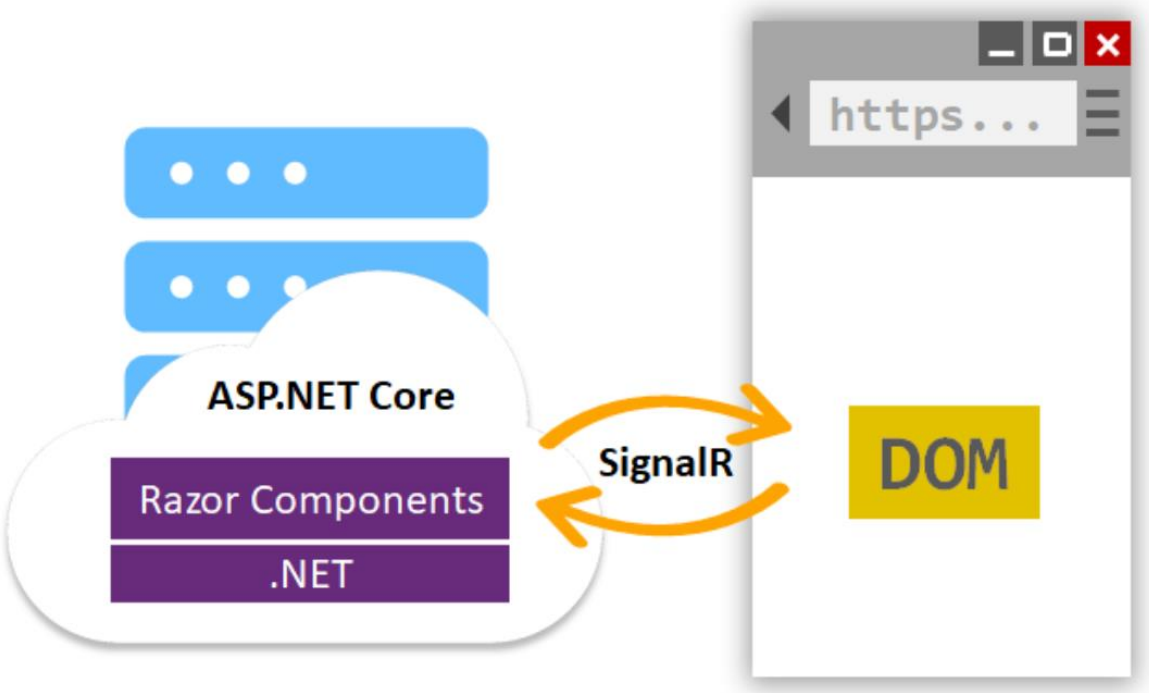
```razor
<div class="card" style="width:22rem">
    <div class="card-body">
        <h3 class="card-title">@Title</h3>
        <p class="card-text">@ChildContent</p>
        <button @onclick="OnYes">Yes!</button>
    </div>
</div>

@code {
    [Parameter]
    public RenderFragment? ChildContent { get; set; }

    [Parameter]
    public string? Title { get; set; }

    private void OnYes()
    {
        Console.WriteLine("Write to the console in C#! 'Yes' button selected.");
    }
}
```

RAZOR MARKUP, KLIKOM KORISNIKA NA GUMB PRIKAZUJE SE DIJALOG 3-2

U primjeru *OnYes* je C# metoda koja se okine na *onclick* event od gumba.

3.2.Blazor server

Blazor Server pruža mogućnost *hostinga* Razor komponenata na poslužitelju u ASP.NET Core aplikaciji. Ažuriranja korisničkog sučelja obrađuju se putem *SignalR* veze. Konekcija koja služi za komunikaciju Blazor Servera i web preglednika također se koristi za rukovanje JS interop poziva.



PRIKAZ KOMUNIKACIJE BLAZORA S DOM-OM PREKO SIGNALR KONEKCIJE 3-3

Blazor Server aplikacije renderiraju[10] sadržaj drugačije od tradicionalnog modela renderiranja UI u ASP.NET Core aplikacijama koje koriste Razor preglede ili Razor stranice. Oba modela koriste Razor jezik za opis HTML sadržaja koji će se prikazivati u aplikaciji, ali način renderiranja je uvelike različit.

Kada se Razor strana ili pregled renderira, svaka linija Razor kōda emitira HTML u obliku teksta, zatim nakon renderiranja poslužitelj očisti stranicu ili instancu prikaza, uključujući bilo koje stanje koje je proizvedeno. Kada se pojavi drugi zahtjev za prikaz stranice, cijela stranica se ponovno generira u HTML i šalje klijentu.
Blazor Server proizvede graf komponenata za prikazivanje sličan HTML-u ili XML *Document Object Model* (DOM). Graf komponenata uključuje stanje sadržano u svojstvima i poljima. Blazor procjenjuje graf komponente kako bi proizveo binarni prikaz oznake, koji se

[10] Proces korišten kod web razvoja koji pretvara kōd u interaktivne stranice koje korisnik vidi na ekranu

šalje klijentu na renderiranje. Nakon što se uspostavi veza između klijenta i poslužitelja, statički se unaprijed prikazani elementi komponente zamjenjuju sa interaktivnim elementima. Pred renderiranje sadržaja na poslužitelju stvara kod aplikacije osjećaj veće responzivnosti.[11] Nakon što komponente postanu interaktivne na klijentu, UI ažuriranja okidaju se interaktivnošću korisnika i *evenata* u aplikaciji. Kada se dogodi promjena, graf komponenata se ponovno renderira i provjerava se razlika između UI elemenata na klijentu. Razlika se zatim šalje klijentu u obliku binarnog zapisa i primjenjuje se u pregledniku.

3.3.Blazor hibrid

Hibridne aplikacije koriste mješavinu izvornih i web tehnologija. Blazor Hibridna aplikacija koristi Blazor u izvornoj klijentskoj aplikaciji. Razor komponente pokreću se izvorno u .NET procesu i renderira web UI u *embedanoj Web View* kontroli koristeći lokalni interop kanal. *WebAssembly* se ne koristi u hibridnim aplikacijama. Hibridne aplikacije obuhvaćaju tehnologije poput .NET Multi-platform App IU (.NET Maui): *cross-platform framework* za izradu nativnih mobilnih i desktop aplikacija koristeći C# i XAML, *Windows Presentation Foundation* (WPF): UI *framework* koji je neovisan o rezoluciji i koristi stroj za renderiranje baziran na vektoru, napravljen da iskoristi prednosti modernog grafičkog hardvera, *Windows Forms*: UI *framework* koji stvara bogatu klijentsku desktop aplikaciju koja podržava bogati spektar razvojnih značajki aplikacije poput, kontrola, grafike, vezivanje podataka i korisnički unos.[12]

[11] https://www.codemag.com/Article/2111092/Blazor-Hybrid-Web-Apps-with-.NET-MAUI
[12] https://www.codemag.com/Article/2111092/Blazor-Hybrid-Web-Apps-with-.NET-MAUI

4. KAKO BLAZOR HIBRIDNE APLIKACIJE RADE ?

.NET MAUI se isporučuje s praktičnom kontrolom zvanom *Blazor WebView*. Ovaj naziv može biti malo zavaravajući, jer poručuje da se Blazor aplikacija i dalje pokreće kao web aplikacija, nekako isključena od izvornog hardvera uređaja na kojem je pokrenuta.

U stvarnosti, unatoč tome što *Blazor WebView* uistinu omogućuje *hosting* Blazor web aplikacije u .NET MAUI aplikaciji, aplikacija se ne izvodi na *WebAssembly*-u ili nekoj drugoj tehnologiji baziranoj na mrežnom pregledniku. Nije potreban mrežni server ili neki drugi način za *hosting* aplikacije.

.NET Blazor hibridna aplikacija izvodit će se u potpunosti izvorno, na uređaju, ne putem HTTP-a i neće se nalaziti u testnom okruženju preglednika, što se inače dogodi kada se pokreće web aplikacija.[13]

Jedini web dio aplikacije je UI, koji je građen koristeći HTML i CSS i renderirano u kontrolu web prikaza. Sav kōd, aplikacija i poslovna logika se izvodi lokalno na uređaju.

Kao rezultat toga, Blazor sa .NET MAUI nudi prikladan način izgradnje aplikacije, koristeći opće poznate paradigme, alate i iskustvo tijekom dizajna, dok donosi prednosti pokretanja izvorne aplikacije uključujući pristup izvornim API-jima kao što su GPS i API-ji za akcelerometar za izvorni uređaj.

[13] https://www.techtarget.com/searchsoftwarequality/definition/hybrid-application-hybrid-app

5. KAKO SE KORISTI BLAZOR WEBVIEW

Blazor kod hibridnog načina rada koristi komponentu *Blazor WebView* koja omogućuje Blazor unutar MAUI aplikacije, stvarajući .NET MAUI Blazor aplikaciju. .NET MAUI Blazor omogućuje izvorno i web korisničko sučelje unutar jedne aplikacije gdje mogu koegzistirati u jednom pregledu. Uz .NET MAUI Blazor, aplikacije mogu iskoristiti Blazorov model komponenti (Razor komponente), koji koristi HTML i CSS i Razor sintaksu. Blazor dio aplikacije može ponovno koristiti komponente, izglede i stilove koji se koriste u postojećoj redovnoj web aplikaciji. *Blazor WebView* može se sastaviti uz izvorne elemente, osim toga, koriste se značajke platforme i dijele stanje s preostalim nativnim elementima

.

U .NET MAUI Blazor aplikacijama, sav kōd, kako za izvorne dijelove korisničkog sučelja tako i za dijelove web korisničkog sučelja, izvodi se kao .NET kōd u *runtime*-u platforme koristeći jedan proces. Nema lokalnog web servera i nema *WebAssembly*-a (WASM) u Blazor Hibridnom načinu rada. Izvorne UI komponente pokreću se kao standardne komponente korisničkog sučelja uređaja (gumb, oznaka itd.), a komponente web korisničkog sučelja smještene su u web prikazu. Komponente mogu dijeliti stanje s pomoću standardnih .NET uzoraka, poput *event handlers, dependency injection*, ili bilo što drugo što se danas koristi u aplikacijama.[14]

5.1.Stvaranje .NET MAUI Blazor aplikacije

Za stvaranje .NET MAUI Blazor aplikacije potrebno je napraviti novi projekt s pomoću Visual Studija.

Tada je moguće deklarirati instancu *Blazor WebView* kontrole, koja pokazuje na bilo koju Razor komponentu koju se želi renderirati.

[14] https://www.codemag.com/Article/2111092/Blazor-Hybrid-Web-Apps-with-.NET-MAUI

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:b="clr-namespace:Microsoft.AspNetCore.Components.WebView.Maui;assembly=Microsoft.AspNetCore.Components.WebView.Maui"
             xmlns:local="clr-namespace:TodoAppMauiBlazor"
             x:Class="TodoAppMauiBlazor.MainPage"
             BackgroundColor="{DynamicResource PageBackgroundColor}">

    <b:BlazorWebView HostPage="wwwroot/index.html">
        <b:BlazorWebView.RootComponents>
            <b:RootComponent Selector="#app" ComponentType="{x:Type local:Main}" />
        </b:BlazorWebView.RootComponents>
    </b:BlazorWebView>

</ContentPage>
```

MAINPAGE.XAML 5-1

U gore navedenom primjeru prikazana je *Main* komponenta (koja se nalazi unutar MAUI klijentske aplikacije). Ovaj dio koda nalazi se u *MainPage.xaml* datoteci i prikazuje embedanu kontrolu preglednika koja će renderirati zadane Blazor komponente koja je u ovom slučaju definirana unutar *Main.razor* datoteke.

```
<Router AppAssembly="@typeof(Main).Assembly">
    <Found Context="routeData">
        <RouteView RouteData="@routeData" DefaultLayout="@typeof(MainLayout)" />
        <FocusOnNavigate RouteData="@routeData" Selector="h1" />
    </Found>
    <NotFound>
        <LayoutView Layout="@typeof(MainLayout)">
            <p role="alert">Sorry, there's nothing at this address.</p>
        </LayoutView>
    </NotFound>
</Router>
```

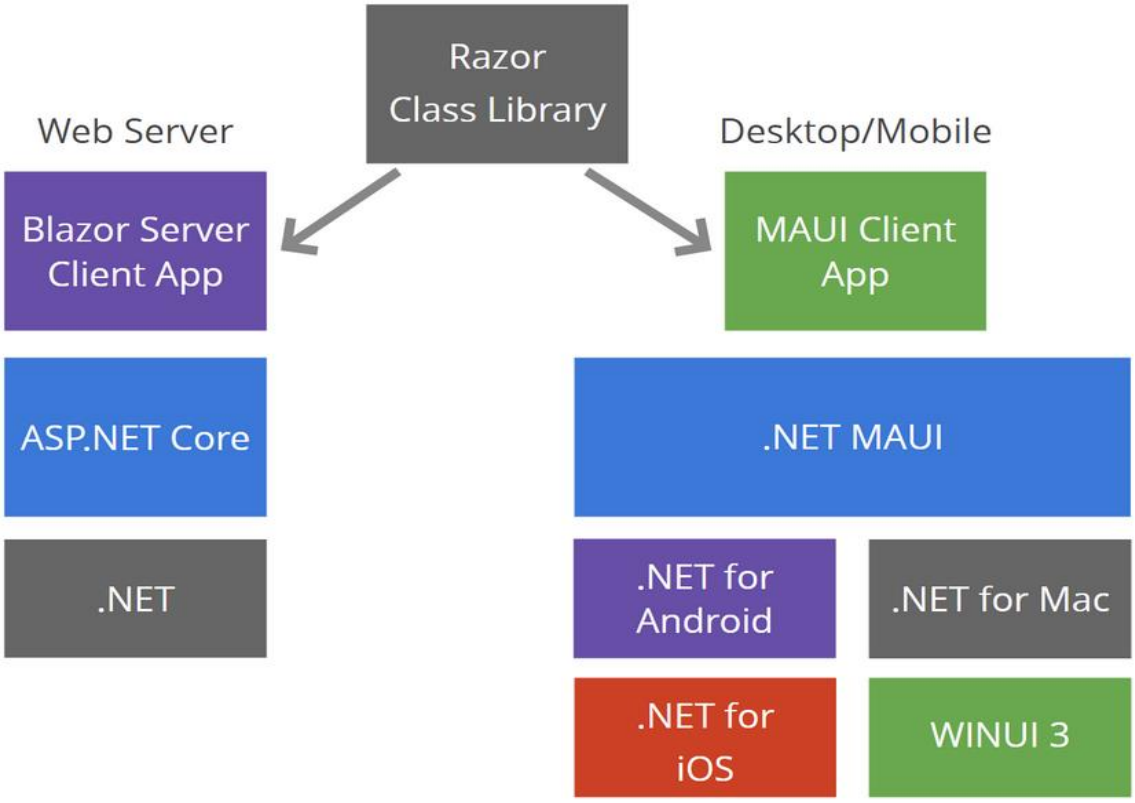
MAIN RAZOR KOMPONENTA 5-2

Main.razor komponenta je običan Blazor *router* koji će pokazivati na daljnje blazor komponente-stranice koje se nalaze u mapi aplikacije pod nazivom *Pages*. Takve komponente imaju na vrhu anotaciju @page "/" koja označava rutu.

Moguće je imati više od jedne *Blazor WebView* komponente, kojom se otvara mogućnost izgradnje .NET MAUI aplikacije koja koristi Blazor za neke ali ne i sve UI komponente (potencijalno koristeći više *Blazor WebView* kontrola za renderiranje različitih komponenata na više mjesta u .NET MAUI aplikaciji).

6. DIJELJENJE KOMPONENATA IZMEĐU WEB I NATIVNOG KLIJENTA

Postoji opcija renderiranja komponenata koje su definirane unutar Razor klasne biblioteke. S tim, moguće je stvoriti dijeljene komponente koje su definirane u Razor klasnoj biblioteci a mogu se koristiti u oba slučaja, Blazor server/WASM i .NET MAUI klijetima.

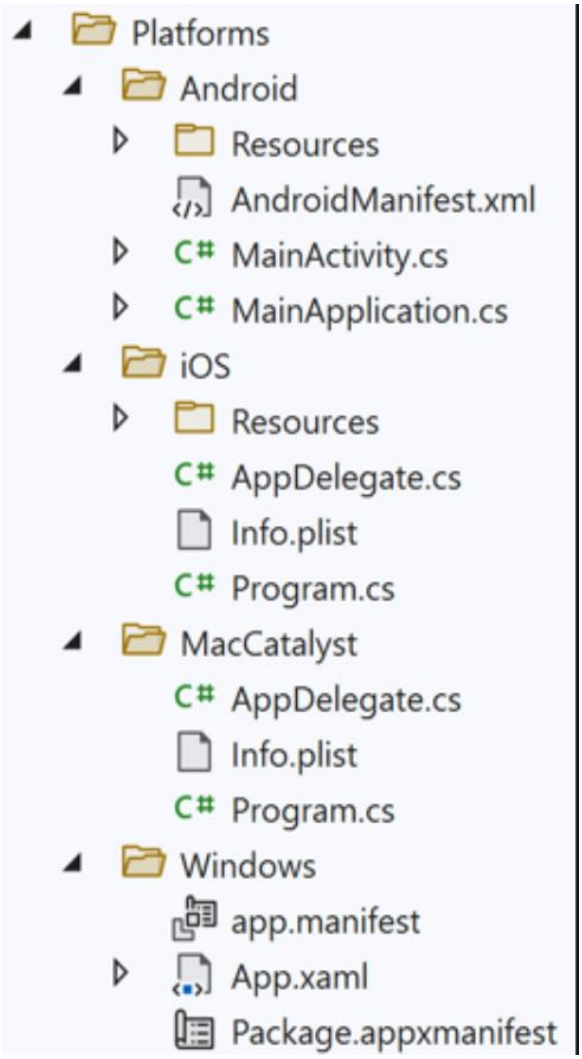


PRIKAZ KLASNE BIBLIOTEKE DIJELJENE MEĐU DVA RAZLIČITA KLIJENTA 6-1

U ovom primjeru, prikazana je Blazor Server aplikacija (koja se oslanja na ASP.NET i radi putem .NET-a na web poslužitelju) i .NET MAUI aplikacija (koja se izvršava na specifičnoj platformskoj verziji .NET *frameworka*). Oboje mogu renderirati komponente iz dijeljene Razor klasne biblioteke.

7. RUKOVANJE RAZLIKAMA IZMEĐU RAZLIČITIH PLATFORMI

.NET MAUI projekt sadrži mapu zvanu *Platforms* koji u sebi sadrži podmape specifične za pojedinu platformu. Svaki od tih mapa sadrži resurse specifične za platformu zajedno sa kōdom koji pokreće aplikaciju na pojedinoj platformi. Kod vremena izgradnje (eng. *build time*) sistem uključuje samo kōd od pojedine platforme. Tako npr. kada se *build*-a aplikacija za Android, samo će se kōd iz mape *Platforms/Android* uključiti u paket aplikacije, ali ostale mape neće. Ovaj pristup koristi *multi-targeting* za ciljanje više platformi koristeći jedan projekt. *Multi-targeting* u kombinaciji s parcijalnim klasama i metodama omogućuje pozivanje izvornih funkcionalnosti pojedine platforme.

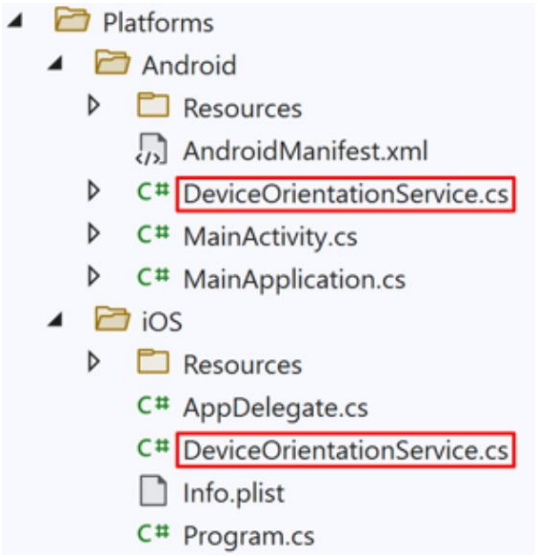


.NET MAUI STRUKTURA MAPA 7-1

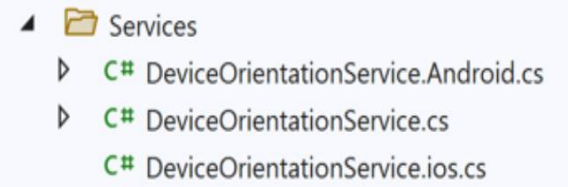
```
namespace InvokePlatformCodeDemos.Services
{
    public enum DeviceOrientation
    {
        Undefined,
        Landscape,
        Portrait
    }

    public partial class DeviceOrientationService
    {
        public partial DeviceOrientation GetOrientation();
    }
}
```

PARCIJALNA KLASA DEVICEORIENTATIONSERVICE 7-2

Parcijalna klasa nazvana je *DeviceOrientationService*, koja uključuje parcijalnu metodu nazvanu *GetOrientation.* Nakon definiranja *cross-platform* API-ja, zadana klasa treba biti implementirana na potrebnim platformama, što znači postavljanje parcijalne klase u odgovarajući *Platform* pod folder da bi se osiguralo da se zadani kōd uključi u projekt samo kod izgradnje aplikacije za specifičnu platformu. Slika desno pokazuje primjer uključivanja parcijalne klase za platformu Android i iOS. Alternativno *multi-targeting* se može izvesti na temelju vlastitog naziva datoteke i kriterija mape, umjesto korištenja mapa *Platforms*. Ovaj način omogućuje strukturiranje .NET MAUI projekta na način da kōd specifičan za pojedinu platformu nije potrebno postaviti u pojedine mape unutar *Platforms* mape.



PRIKAZ POSTAVLJANJA PARCIJALNE KLASE U FOLDER SPECIFIČAN ZA PLATFORMU 7-3



FILENAME-BASED MULTITARGETING 7-5

FOLDER-BASED MULTITARGETING 7-4

Na slikama je primjer *multi-targeting* paterna koji prikazuje *filename-based multi-targeting* (lijevo) i *folder-based multi-targeting* (desno).

Slijedeći primjer pokazat će korištenje kōda specifičnog za platformu implementiranog direktno u Razor komponenti lociranoj u .NET MAUI Blazor projektu koja provjerava na kojem je trenutno OS-u (operativnom sustavu) pokrenuta aplikacija i na temelju toga odabire koji kōd se treba izvršiti. Recimo da želimo prikazati gumb koji omogućava korisniku snimanje slika, koristeći kameru na iOS-u ili Androidu, ali ne na Windows ili MacOS-u.

```
bool showCameraButton = OperatingSystem.IsAndroid() || OperatingSystem.IsIOS();
```

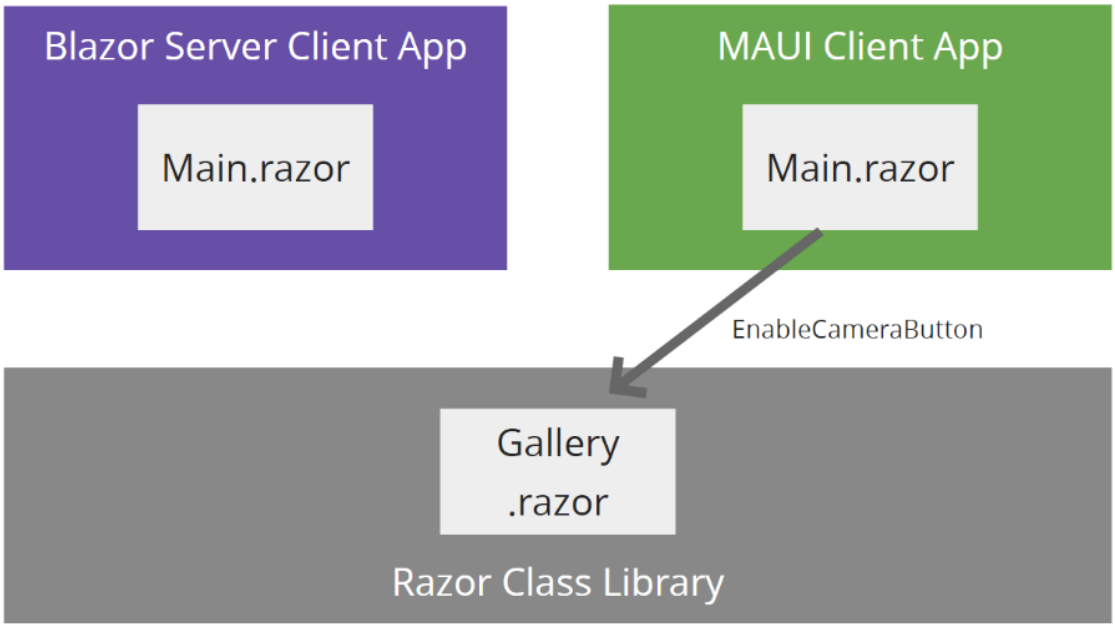
VARIJABLA SHOWCAMERABUTTON 7-6

S pomoću varijable koja provjerava na kojem je trenutno sustavu pokrenuta aplikacija na ekranu se iscrtava UI element ili ne.

```
@if(showCameraButton){
    // camera button here!
}
```

KŌD KOJI PROVJERAVA OS I ISCRTAVA GUMB 7-7

S obzirom na to da je Blazor ujedno i *framework* koji se koristi za izradu web aplikacija, ako postoji potreba zadanu komponentu iskoristiti i za web aplikaciju potrebno je jasnije definirati koji dio kōda se želi podijeliti a koji ne. Tako će gore navedeni kondicional za prikazivanje gumba za pokretanje kamere moći zaživjeti samo u .NET MAUI klijentskoj aplikaciji, jer samo tamo je moguće detektirati operativni sustav. Da bi se komponenta mogla podijeliti između različitih aplikacija potrebno je uzdignuti logiku za provjeru OS-a nivo iznad u komponentu koja se samo pokreće u .NET MAUI aplikaciji, i dozvoliti joj da proslijedi parametar dalje u dijeljenu komponentu, koja će zatim na temelju toga zaključiti svoje ponašanje.



PREMJEŠTANJE ENABLECAMERABUTTON FUNKCIONALNOSTI U RAZOR CLASS LIBRARY 7-8

Main.razor (MAUI client)

```
<Gallery EnableCameraButton="@enableCamera" />
```

```
@code {
     private bool enableCamera = // platform checks to decide;
}
```

Gallery.razor (Razor Class Library)

```
@if (EnableCameraButton)
{
    <button>Take a Picture</button>
}
```

```
@code {

    [Parameter]
    public bool EnableCameraButton { get; set; }

}
```

MAIN.RAZOR I GALLERY.RAZOR 7-9

Na slikama je prikazan primjer *Main.razor* komponente koja se nalazi u .NET MAUI aplikaciji i koja provjerava platformu na kojoj je pokrenuta, i zatim tu informaciju prosljeđuje kao parametar u dijeljenu komponentu *Gallery.razor* koja se nalazi u dijeljenom projektu *Razor Class Library.* Da bi se vratila informacija o tome je li gumb kliknut ili ne natrag u *Main.razor* komponentu pobrinuti će se novi parametar tipa *EventCallback* pod nazivom *CameraButtonClicked* koji omogućuje slanje povratne informacije kada se okine event *onClick* na html <button></button> elementu.

Gallery.razor (Razor Class Library)

```
@if (EnableCameraButton)
{
    <button @onclick="CameraButtonClicked">Take a Picture</button>
}
```

```
@code {

    [Parameter]
    public bool EnableCameraButton { get; set; }

    [Parameter]
    public EventCallback CameraButtonClicked { get; set; }

}
```

Main.razor (MAUI client)

```
<Gallery EnableCameraButton="@enableCamera" CameraButtonClicked="@OpenCamera" />
```

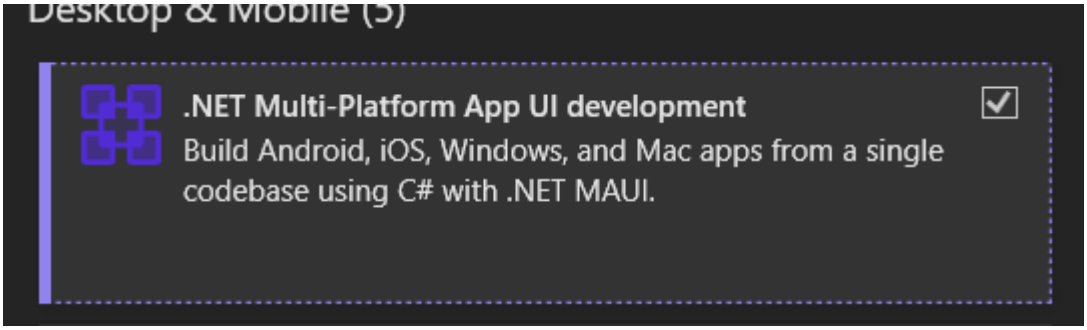
```
@code {
    private bool enableCamera = true;

    void OpenCamera()
    {
        // call platform specific code to open the camera
    }
}
```

MAIN.RAZOR I GALLERY.RAZOR 7-10

Nakon toga moguće je u *Main.razor* komponenti reagirati na event i izvršiti metodu koja je zaslužna za specifičan kōd pojedine platforme.

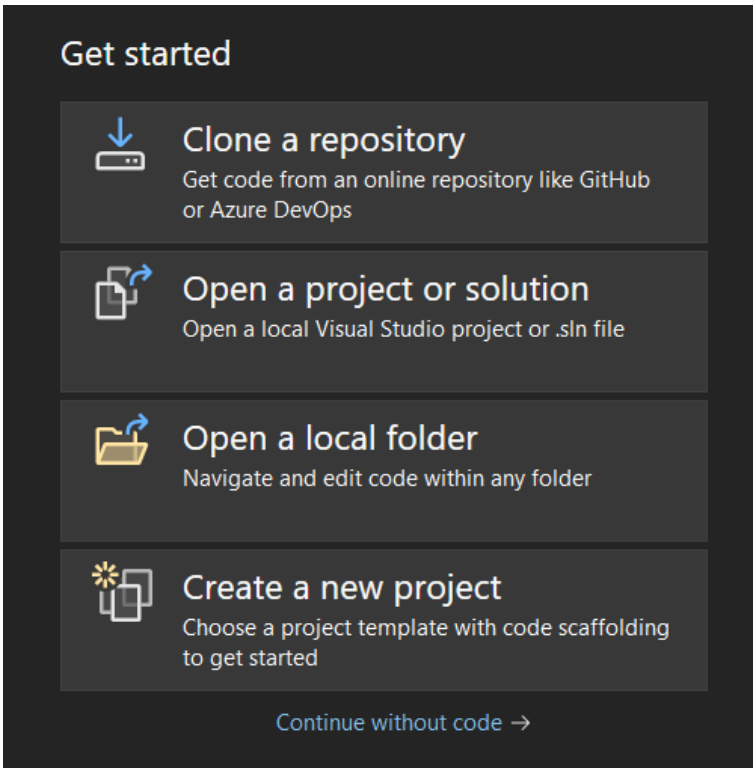
8. RAZVOJ APLIKACIJE KORISTEĆI .NET MAUI BLAZOR

Preduvjet za novi .NET MAUI projekt je instalacija Visual Studio 2022 Preview skupa s paketom .NET Multi-Platform App UI development koji sadrži .NET 6 SDK, MAUI SDK i MAUI predložak.



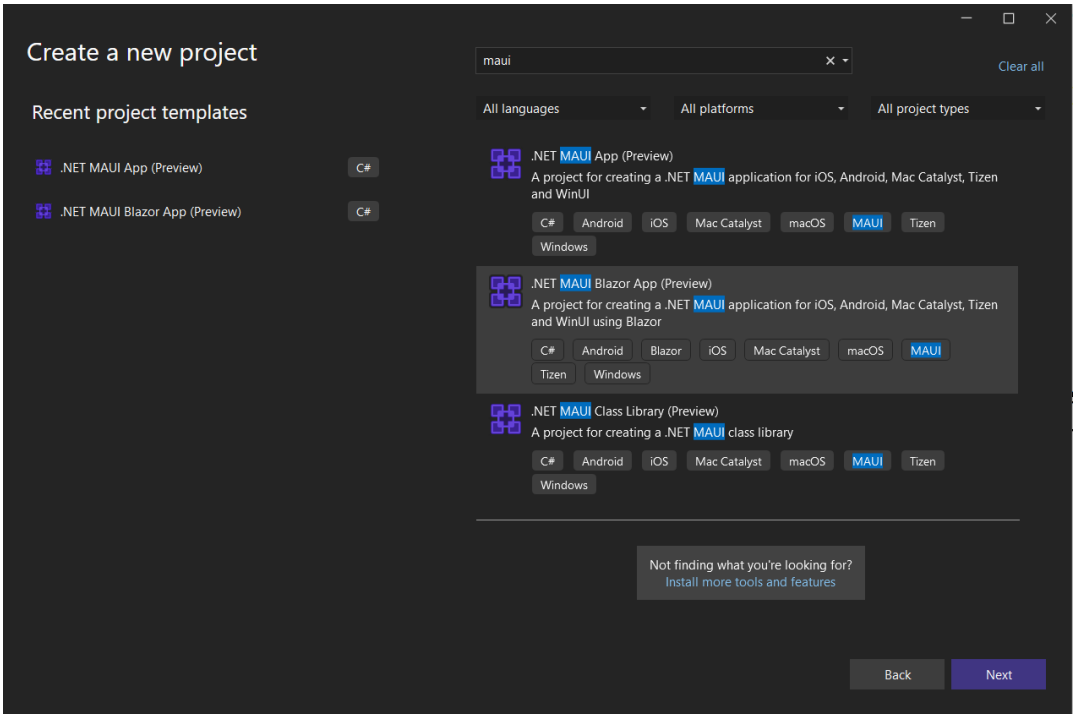
.NET MULTI-PLATFORM APP UI DEVELOPMENT INSTALACIJSKI PAKET 8-1

Novi .NET MAUI projekt može se napraviti unutar Visual Studija jednostavnim odabirom **File** > **New** ili klikom na **Create a new project** prilikom pokretanja Visual Studija.



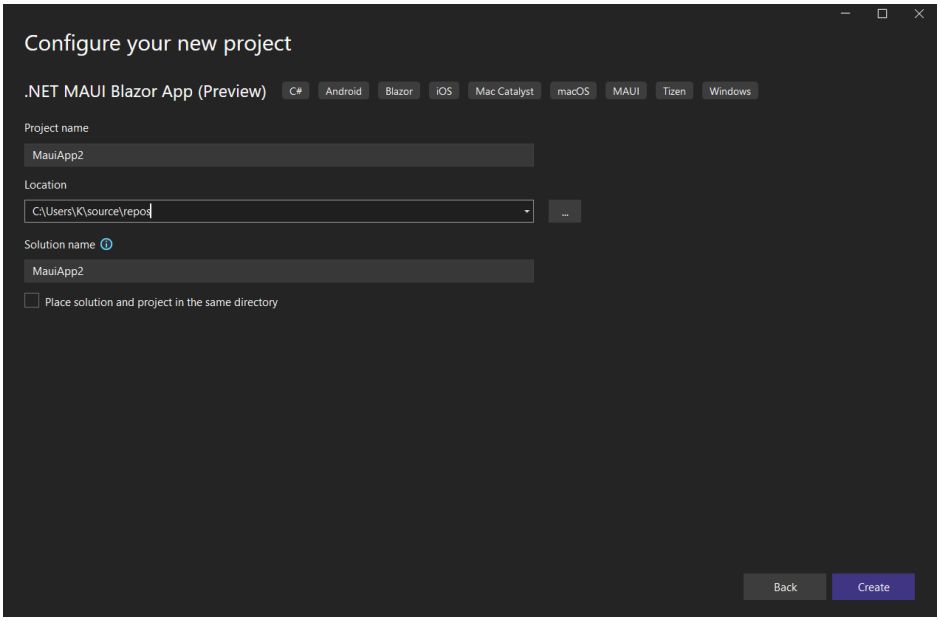
GET STARTED PROZOR 8-2

Potrebno je odabrati **.NET MAUI Blazor App** predložak i kliknuti dugme **Next**.



.NET MAUI PREDLOŽAK 8-3

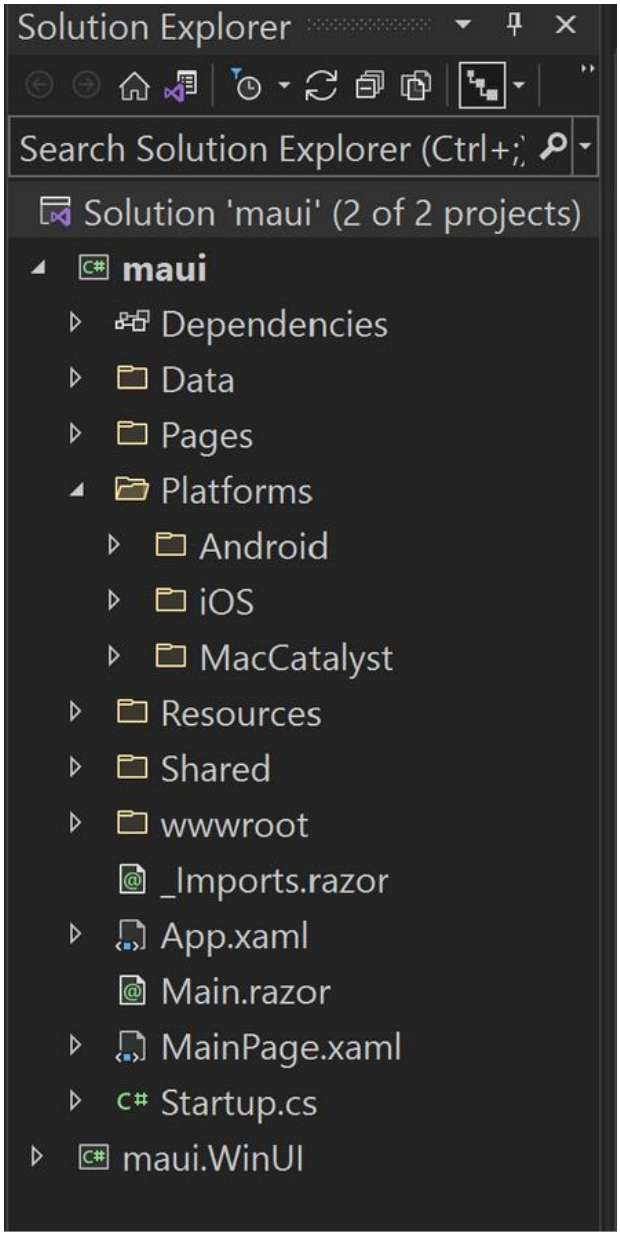
U dijalogu za konfiguriranje projekta postavlja se ime i lokaciju na disku i klikom na **Create** kreirat će se novi .NET MAUI Blazor projekt.



DIJALOG ZA KREIRANJE PROJEKTA 8-4

Nakon sto template napravi novi projekt, može se vidjeti kako se .NET MAUI Blazor aplikacija strukturirana.

.NET MAUI Blazor aplikacija dijeli sličnosti s tradicionalnom Blazor aplikacijom uz dodatak MAUI značajki, kao sto su mapa sa značajkama specifičnim za platformu i XAML datoteke



NOVOIZGRAĐENA .NET MAUI BLAZOR APLIKACIJA NAPRAVLJENA IZ PREDLOŠKA 8-5

.NET MAUI Blazor aplikacija koristi obrazac koji se može pronaći u mnogim .NET aplikacijama. Neke ključne razlike su dualnost hibridnog scenarija gdje postoji preklapanje između koncepata *root-level* pregleda komponenti i *routing*. Mape u projektu podijeljene su u nekoliko kategorija. *Data* sadrži klase koje su zadužene za dohvaćanje i obradu podataka, *Pages* sadrži stranice koje imaju direktivu na vrhu datoteke *@page "/"* (tekst pod navodnicima pokazuje na rutu na kojoj se nalazi stranica), *Platforms* je mapa specifična

za .NET MAUI projekt koja sadrži kōd specifičan za pojedinu platformu, *Resources* sadrži resurse aplikacije poput fonta, slika i sl., *Shared* mapa sadržava dijeljene komponente u projektu. w*wwroot* mapa je bazna mapa projekta u kojoj se nalazi *index.html* datoteka i *css* mapa s css datotekama, također može sadržavati i dodatne mape poput *js* mape u kojoj je sadržan JavaScript kōd.

8.1.MauiProgram.cs

Kao kod većine .NET aplikacija *Startup.cs* je ulazna točka aplikacije, kod MAUI projekta *Startup.cs* nazvan je *MauiProgram.cs* . Kako se pokretanje inicijalizira, aplikacija se konstruira i servisi se registriraju, aplikacija se konfigurira i *dependency injection* se

```
4 references
public static class MauiProgram
{
    4 references
    public static MauiApp CreateMauiApp()
    {
        var builder = MauiApp.CreateBuilder();
        builder
            .UseMauiApp<App>()
            .ConfigureFonts(fonts =>
            {
                fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");
            });

        builder.Services.AddMauiBlazorWebView();
#if DEBUG
        builder.Services.AddBlazorWebViewDeveloperTools();
#endif

        builder.Services.AddSingleton<WeatherForecastService>();

        return builder.Build();
    }
}
```

MAUIPROGRAM.CS 8-6

razlučuje.

U primjeru iznad *builder* se koristi za dodavanje značajki u aplikaciju. *UserMauiApp<App>* metoda inicijalizira *App roop* aplikacije. *App* klasa specificira *MainPage* aplikacije, koji se inicijalizira unutar *App.xaml.cs* klase. Ostatak *builder* registrira i konfigurira servise koji se koriste s pomoću *dependency injection* (DI)[15] kroz cijelu aplikaciju.

[15] Dizajn pattern u kojem objekt ili funkcija prima druge objekte ili funkcije o kojima ovisi, što dovodi do „loosely coupled“ programa.

8.2.MainPage.xaml(.cs)

U *MainPage.xaml* datoteci može se vidjeti prvo predstavljanje Blazora kao *Blazor WebView*. *MainPage* omotava *Blazor WebView* izravno unutar *ContentPage*, stvarajući Blazor prikaz cijele stranice unutar ljuske korisničkog sučelja aplikacije.

```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:local="clr-namespace:MauiApp2"
             x:Class="MauiApp2.MainPage"
             BackgroundColor="{DynamicResource PageBackgroundColor}">

    <BlazorWebView HostPage="wwwroot/index.html">
        <BlazorWebView.RootComponents>
            <RootComponent Selector="#app" ComponentType="{x:Type local:Main}" />
        </BlazorWebView.RootComponents>
    </BlazorWebView>

</ContentPage>
```

PRIKAZ BLAZOR WEBVIEW KOMPONENTE UNUTAR MAINPAGE.XAML 8-7

Blazor WebView koristi *HostPage* parametar za identifikaciju HTML stranice, koja će pokrenuti aplikaciju Blazor. U *index.html* se nalazi *root* dokument koji hosta Blazor aplikaciju u pregledu.

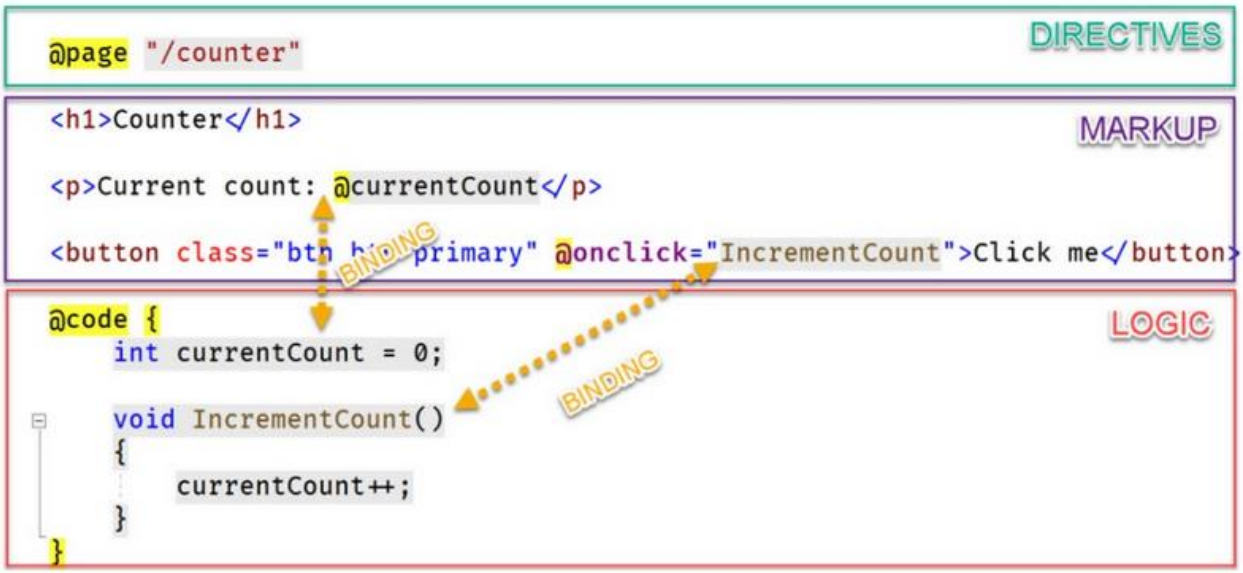
```
<body>

    <div class="status-bar-safe-area"></div>

    <div id="app">Loading...</div>

    <div id="blazor-error-ui">
        An unhandled error has occurred.
        <a href="" class="reload">Reload</a>
        <a class="dismiss">✕</a>
    </div>

    <script src="_framework/blazor.webview.js" autostart="false"></script>

</body>
```

BLAZOR.WEBBIEW.JS UNUTAR INDEX.HTML 8-8 DATOTEKE

Za razliku od Blazor WebAssembly, ova html datoteka inicijalizira Blazor koristeći *blazor.webview.js* umjesto *blazor.webassembly.js*. Ovdje je razlika u tome što Blazor ne koristi WebAssembly već .NET *runtime* za izvođenje glavne aplikacije.

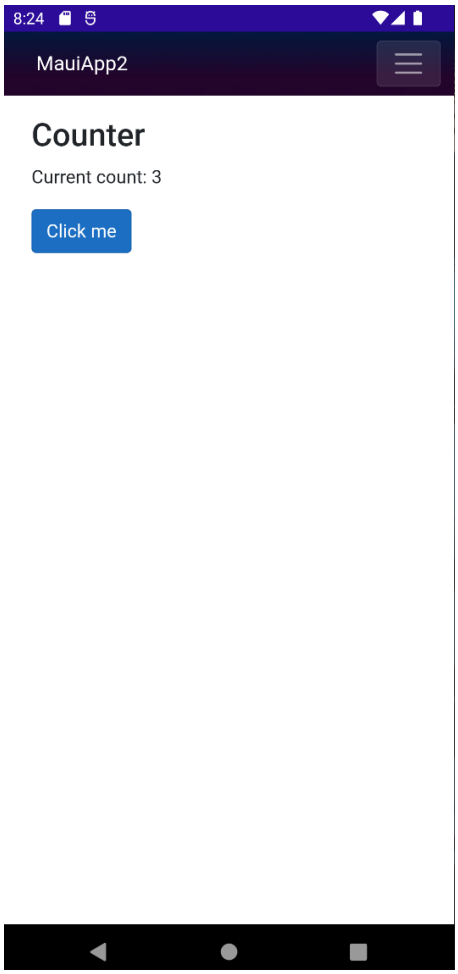
8.3.Counter.razor

Counter page je primjer jednostavne razor komponente dekorirane s *@page* direktivom na vrhu. Ova komponenta demonstrira osnovnu strukturu Razor komponente koja uključuje *routing*, *data binding* i *event binding*/*handling* i prikazuje na ekranu jednostavan brojač koji se inkrementira klikom na dugme u stvarnom vremenu.



PRIMJER STRUKTURE BLAZOR KOMPONENTE 8-9

Counter komponenta koristi osnovne HTML elemente poput <h1> za naslov, <p> za paragraf i <button> za inkrementiranje brojača koji je prikazan unutar <p> taga. Ažuriranja u Blazor WebView DOM-u se obavljaju s pomoću Blazor frameworka koristeći *data binding*.



PRIKAZ COUNTER KOMPONENTE RENDERIRANE U .NET MAUI BLAZOR ANDROID EMULATORU 8-10

8.4.FetchData.razor

U .NET MAUI Blazor projektu *FetchData.razor* je komponenta koja dohvaća podatke preko servisa. *FetchData.razor* komponenta demonstrira *dependency injection* i osnovne koncepte Razor predloška.

```
@page "/fetchdata"

@using MauiApp2.Data
@inject WeatherForecastService ForecastService

<h1>Weather forecast</h1>

<p>This component demonstrates fetching data from a service.</p>

@if (forecasts == null)
{
    <p><em>Loading...</em></p>
}
else
{
    <table class="table">
        <thead>
            <tr>
                <th>Date</th>
                <th>Temp. (C)</th>
                <th>Temp. (F)</th>
                <th>Summary</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var forecast in forecasts)
            {
                <tr>
                    <td>@forecast.Date.ToShortDateString()</td>
                    <td>@forecast.TemperatureC</td>
                    <td>@forecast.TemperatureF</td>
                    <td>@forecast.Summary</td>
                </tr>
            }
        </tbody>
    </table>
}

@code {
    private WeatherForecast[] forecasts;

    protected override async Task OnInitializedAsync()
    {
        forecasts = await ForecastService.GetForecastAsync(DateTime.Now);
    }
}

```

FETCHDATA.RAZOR 8-11

8.5.Pokretanje .NET MAUI BLAZORA

Prilikom pokretanja aplikacije u Visual Studiju, lako je uočljiva *cross-platform*-ska priroda MAUI-a. Odabirom na "run" u Visual Studiju nudi se mogućnost odabira emulatora i/ili fizičkog uređaja koji su spojeni na računalo.

SELEKCIJA ZA POKRETANJE APLIKACIJE U VISUAL STUDIJU 8-12

Stvaranje konekcije s emulatorima i uređajima je olakšano putem Visual Studija, to je značajka koja je evoluirala iz dugogodišnjeg podržavanja Xamarina. Kada se aplikacija pokrene, vidljiva je Blazor aplikacija koja radi u izvornoj ljusci, a sve bez potrebe za web preglednikom ili plug-inom. U zadanom primjeru Blazor Web UI koristi se za svu navigaciju, *routing*, i poglede u aplikaciji. Uključeni su primjeri brojača i dohvaćanja podataka, koji su standardni primjeri za Blazor Web projekte.

Predložak je samo uvid u to kako je Blazor aplikacija integrirana s MAUI-jem kako bi se formirala .NET MAUI Blazor aplikacija. Blazor Hybrid ima mnogo više od same komponente Blazor WebView.

9. BLAZOR HYBRID POSEBNOSTI

Kad je Blazor prvi put uveden u web, jedan od primarnih ciljeva bio je omogućiti programerima da izgrade korisničko sučelje web aplikacija koristeći .NET. Blazor omogućuje rad postojećeg .NET kōda korištenjem .NET runtimea putem WebAssembly-a. Uz Blazor Hybrid, primarni se cilj neznatno pomaknuo proširenjem mogućnosti .NET programera izvan weba na desktop i mobilni prostor uz ponovnu upotrebu HTML i CSS vještina uz .NET. Blazor Hybrid uzorak s .NET MAUI Blazor nudi neke jedinstvene sposobnosti koje nisu dostupne kod razvoja usmjerenog na web.

9.1.Smanjivanje kompromisa pokretanjem izvornog .NET-a

Blazor WebAssembly i Blazor Server dolaze s kompromisima. Dobije se mogućnost .NET-a umjesto JavaScript-a po cijenu apstrakcije. Kod korištenja WebAssembly-ja .NET *runtime* radi u interpretiranom načinu rada i nije tako učinkovit kao .NET koji se izvorno izvodi. Iako napredak ove tehnologije dolazi u .NET 6 verziji s *Ahead of Time Compilation* (AOT)[16], ostaje kompromis pri odabiru WebAssembly-a. Slično tome, Blazor Server također ima kompromise. Iako Blazor Server dobiva mogućnost pokretanja .NET-a izvorno na serveru, zahtijeva konstantnu konekciju prema klijentu i performanse ovise o klijentskoj latenciji.

.NET MAUI Blazor aplikacija ima jedinstvenu poziciju u Blazor ekosustavu gdje može eliminirati kompromise tako sto se pokreće u .NET runtime-u koji isporučuje izvorna platforma. Za razliku od Blazor WebAssembly-ija, .NET MAUI Blazor ne koristi interpretirani način rada i radi kao izvorna aplikacija uređaja. Budući da se .NET MAUI Blazor obrađuje lokalno, kompromisi Blazor Servera također se zaobilaze. Iako postoje nijanse u izvršavanju .NET runtimea, Blazorovo izvršavanje je „*Closer to the metal*“[17] nego WebAssembly.

Uz performanse, .NET MAUI Blazor također ima najveći potencijal za dijeljenje jedne baze koda, dok istovremeno cilja razvoj na različitim platformama. To uključuje mogućnost objavljivanja aplikacija u svim glavnim trgovinama aplikacija s dodatnom mogućnošću primanja ikone mobilnog početnog zaslona ili ikone na radnoj površini.

[16] https://www.codemag.com/Article/2111092/Blazor-Hybrid-Web-Apps-with-.NET-MAUI
[17] https://www.codemag.com/Article/2111092/Blazor-Hybrid-Web-Apps-with-.NET-MAUI

9.2.Korištenje API-ja

.NET MAUI Blazor aplikacije nisu ograničene u isti Web *sandbox* kao Blazor *WebAssembly*. .NET MAUI Blazor aplikacije koriste .NET 6 *Base Class Library* (BLC) koja je implementirana na svim platformama. .NET MAUI objedinjuje API-je za više platformi u jedan API koji omogućuje razvojnom programeru jednokratno pokretanje bilo gdje, dok dodatno pruža dubok pristup svakom aspektu svake izvorne platforme. .NET MAUI također nudi MAUI Essentials, više-platformski API za izvorne značajke uređaja.

Primjer funkcionalnosti koje pružaju .NET MAUI uključuju:

- Pristup senzorima, kao što su akcelerometar, kompas i žiroskop na uređajima
- Mogućnost provjere stanja mrežne povezanosti uređaja i otkrivanja promjena
- Dohvaćanje informacije o uređaju na kojem se aplikacija izvodi
- Kopiranje i lijepljenje teksta u među spremnik na kojem se aplikacija izvodi
- Odabir jedne ili više datoteka s uređaja
- Sigurno pohranjivanje podataka u obliku ključ-vrijednost
- Korištenje ugrađenog mehanizma za pretvaranje teksta u govor za čitanje teksta s uređaja
- Pokretanje provjere autentičnosti temeljene na pregledniku koji osluškuju povratni poziv na određeni URL registriran u aplikaciji

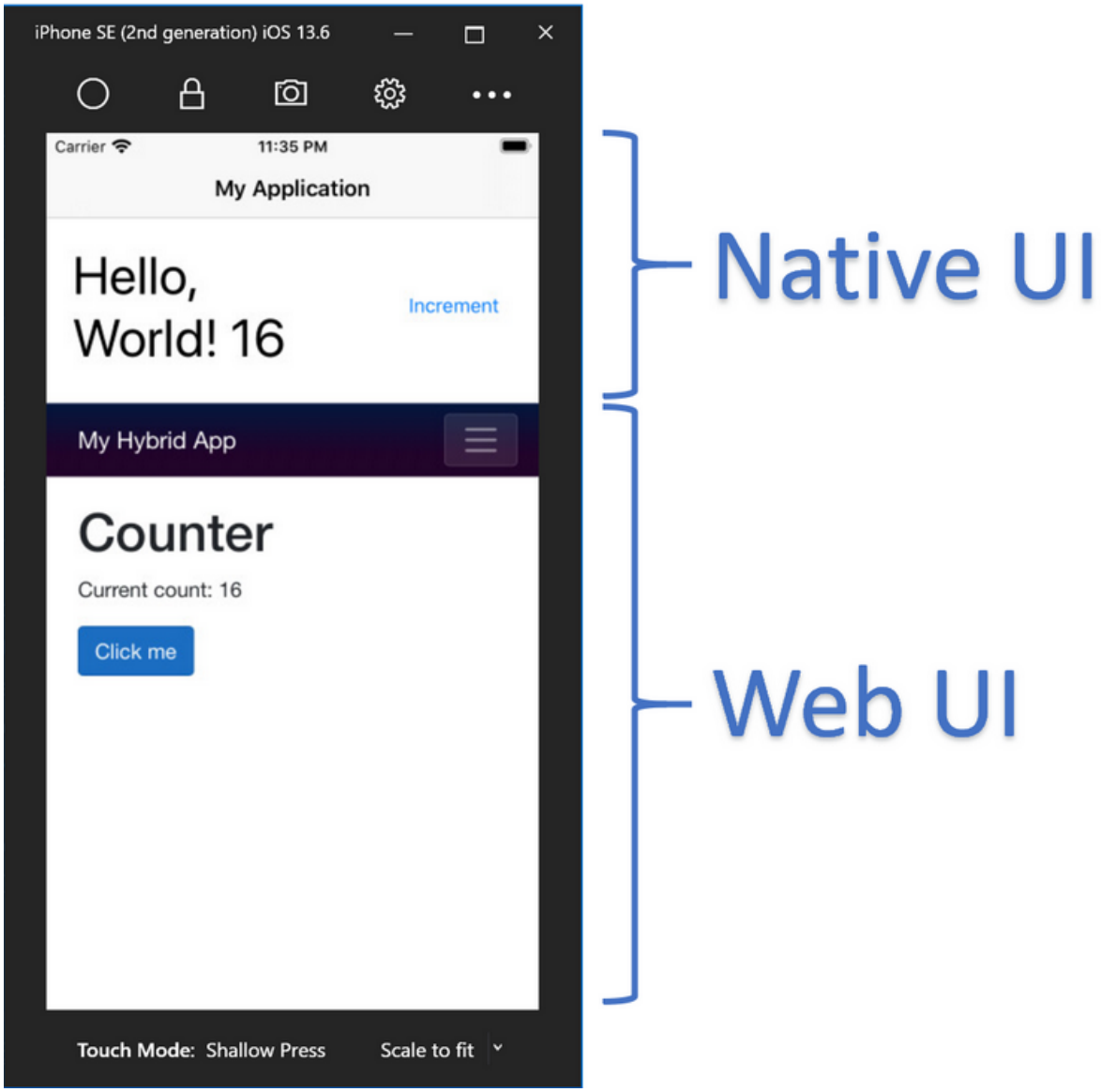
U sljedećem primjeru prikazano je kako pročitati CSV datoteku s diska koristeći *System.IO.StreamReader* iz BLC-a. *WeatherForecastService* je klasa koja se može injektirati u MAUI pregled ili *Blazor WebView*. Kada se pozove metoda *GetForecast* , datoteka s diska se učitava u niz *WeatherForecat.* Servis koristi cross-platform putanju za datoteke *Environment.SpecialFolder.LocalApplicationData* kako bi se osigurala kompatibilnost uređaja. Jednom kada je putanja uspostavljena datoteka se učita u *stream* čitača i obrađuje s pomoću CSV čitača otvorenog koda, *CsvHelper*. U komponenti *FetchData, WeatherForcastService* se ubrizgava i poziva se metoda *GetForecast.* Nakon što su podaci pročitani s diska, Razor sintaksa se koristi za iteriranje kroz podatke koristeći HTML i CSS. Kōd pregleda je identičan Blazor Maui predlošku jer su se samo promijenili unutarnji dijelovi servisa.

```
public WeatherForecast[] GetForecast()
{
    string fileName = Path.Combine(
        Environment.GetFolderPath(
        Environment.SpecialFolder.LocalApplicationData), "temp.csv");

    WeatherForecast[] weather;
    using (var reader = new StreamReader(fileName))
    using (var csv = new CsvReader(reader, CultureInfo.InvariantCulture))
    {
        weather = csv.GetRecords<WeatherForecast>().ToArray();
    }
    return weather;
}
```

WeatherForecastService.cs 9-1

Pokretanje aplikacije na desktop ili mobilnom uređaju pruža jednako iskustvo pregleda. CSV podaci se čitaju s diska i prikazuju u hibridnom pregledu.

Navedeni primjeri su samo jedan način korištenja *Blazor Webview*-a s MAUI-jem. U slijedećem primjeru, Blazor se koristi ekskluzivno unutar ljuske aplikacije pružajući kompletnu UI navigaciju unutar *Main.razor* komponenti, koja je specificirana u *ComponentType* parametru. Preuzimanje cijele aplikacije s *Blazor WebView* komponentom je opcionalno, može se dodati u bilo koji XAML pregled i čak se može i pomiješati s izvornim UI komponentama. Pomiješani UI može dijeliti stanje aplikacije i mogu potpuno međusobno komunicirati.



.NET MAUI Blazor aplikacija pomiješana s izvornim web UI-om i dijeljenim stanjem 9-2

9.3.Hibridni ekosistem

.NET 6 će daljnje ojačati developerski ekosustav koji okružuje MAUI i Blazor. Od samog početka kada je Blazor bio još u eksperimentalnoj fazi, počele su se pojavljivati knjižnice kōda koje podržavaju model aplikacije. Primjer toga je Telerik UI[18], ime koje je već 20 godina sinonim za razvoj .NET aplikacija, koji ima dedicirane MAUI i Blazor UI komponente, tako da developeri imaju izbor između izvorno platformskog UI-a koristeći MAUI i XAML, Blazor s HTML I CSS-om, ili oboje za izradu UI elemenata.[19]

[18] https://www.telerik.com/maui-ui
[19] https://www.codemag.com/Article/2111092/Blazor-Hybrid-Web-Apps-with-.NET-MAUI

10. ZAKLJUČAK

Blazor omogućuje izgradnju klijentskog web sučelja s .NET-om, ali ponekad je potrebno više od onoga što web platforma nudi. .NET MAUI omogućuje izgradnju cross-platform aplikacija koristeći Blazor komponente i web sučelje s potpunim pristupom izvornim funkcionalnostima uređaja. Ovakve hibridne aplikacije mogu se izvoditi na bilo kojoj od platformi poput Windows, Mac, iOS i Android. S obzirom na to da .NET MAUI Blazor omogućuje stvaranje nativnih aplikacija i njihovog sučelja koristeći HTML I CSS posebno je privlačan web developerima koji su već upoznati s takvim tehnologijama. Neke od prednosti korištenja Xamarina za izradu cross-platformskih aplikacija je velika prilagodljivost. :NET MAUI omogućuje developerima da prilagode korisničko sučelje svoje aplikacije da izgleda kao da je napravljena korištenjem nativnog razvoja. Koristeći C# kao osnovni jezik dopušta programerima da koriste bogate značajke jezika tijekom kodiranja, kao i lako međusobno dijeljenje knjižnica klasa. Korištenje Xamarin formsa omogućuje developerima da naprave sučelje koje radi s bilo kojom platformom.

11. POPIS SLIKA

12. POPIS LITERATURE

https://www.codemag.com/Article/2111092/Blazor-Hybrid-Web-Apps-with-.NET-MAUI

https://www.c-sharpcorner.com/article/what-is-blazor-and-how-does-it-works/

https://www.telerik.com/blogs/blazor-dotnet-maui-what-how-when

https://docs.microsoft.com/en-us/aspnet/core/blazor/hybrid/tutorials/maui?view=aspnetcore-6.0

https://docs.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-6.0#blazor-webassembly

https://www.computerhope.com/jargon/t/thread.htm

https://www.techtarget.com/searchsoftwarequality/definition/hybrid-application-hybrid-app

SUMMARY

.Net Maui – framework for developing cross-platform hybrid applications.

This paper will describe the .NET MAUI Blazor framework for creating hybrid cross-platform applications with a single codebase using the C# programming language and Xaml or Blazor, developed by Microsoft Corporation. It will go over Hybrid applications and how they have been used so far to develop applications on the Windows platform, Blazor as a framework for creating web applications and its use in .Net Maui framework, and how Blazor hybrid applications work. The process of developing a cross-platform application using the Visual Studio development tool and the Blazor Maui framework will also be shown with all the steps necessary to create such an application. Finally, we will go over specialty of using Blazor hybrid mode and the possibility of using HTML and CSS for desktop and mobile app development.