Mobilna aplikacija za Windows Phone platformu za prikaz informacija o kulturnoj baštini

Mandić, Luka

Master's thesis / Diplomski rad

2016

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:406776

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-29**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

Mobilna aplikacija za Windows Phone platformu za prikaz informacija o kulturnoj baštini

Diplomski rad

Luka Mandić

Osijek, 2016.

Sadržaj

1. UVOD 1
1.1 Zadatak završnog rada 2
2. RAČUNALNE TEHNOLOGIJE U KULTURNOJ BAŠTINI 3
2.1. Tehnologija u kulturi 3
2.2. Podaci o glagoljičnim spomenicima 5
2.3. Slična tehnička rješenja 7
2.4. Idejno rješenje aplikacije 9
3. KORIŠTENE TEHNOLOGIJE I PROGRAMSKO RJEŠENJE APLIKACIJE 10
3.1. Programsko rješenje aplikacije 10
3.1.1. Korisničko sučelje i univerzalni ulazi 13
3.2. Windows 10 16
3.3. Rad i manipulacija podacima 16
3.4. Arhitektura aplikacije 20
3.4.1. Model podataka 27
4. SLUČAJEVI KORIŠTENJA I DIZAJN APLIKACIJE 29
4.1. Slučajevi korištenja 29
4.2. Dizajn aplikacije 32
4.2.1. Upute za korištenje 34
5. ZAKLJUČAK 39
LITERATURA 40
SAŽETAK 41
ABSTRACT 41
ŽIVOTOPIS 42
PRILOZI 43

1. UVOD

Tema diplomskog rada je izrada aplikacije koja omogućava prikaz i pretraživanje informacija o kulturnoj baštini za Windows Phone 10 platformu. Cilj rada je dizajnirati i izraditi aplikaciju koja korisnicima olakšava pristup informacijama o kulturnoj baštini Republike Hrvatske. Namjena aplikacije je pomoći korisnicima u obrazovanju te aplikaciju prikazati kao alat za učenje o spomenicima kulturne baštine na Odsjeku hrvatskog jezika i književnosti Filozofskog fakulteta u Osijeku. Aplikacija omogućava korisnicima informacije na dlanu štedeći vrijeme i omogućavajući korisniku da na zaslonu njegovog mobilnog uređaja trenutno mogu saznati osnovne informacije kao što su puno ime, mjesto, vrijeme nastanka, pismo, jezik, sadržaj te zanimljivosti o pojedinom spomeniku ili ostavštini. Projekt je nastao kao suradnja dviju sastavnica osječkog Sveučilišta – Filozofskog i Fakulteta elektrotehnike, računarstva i informacijskih tehnologija, a proveli su ga studenti dvaju fakulteta u suradnji s mentorima te tako projekt ima znanstvenu podlogu kao temelj za širu uporabu među korisnicima.

Naziv diplomskog rada opisuje prikaz informacija za Windows 10 univerzalnu platformu sa naglaskom na Windows 10 mobilnu aplikaciju. Aplikacija je izrađena i prilagođena za mobilne uređaje s Windows Phone 10 operacijskim sustavom te stolna računala s instaliranim Windows 10 operacijskim sustavom. Problematika rješenja je bila prilagoditi tekst i informacije u JSON formatu na zaslon mobilnog uređaja te omogućiti korisniku da te informacije u svakom trenutku budu dostupne u aplikaciji. Osim mogućnosti pretraživanja te sortiranja spomenika po stoljećima aplikacija sadrži i vlastiti prevoditelj s latinice na glagoljicu. Aplikacija je razvijena u Visual Studio 2015 programskoj platformi, a korišten je programski jezik C# te opisni jezik XAML.

U poglavlju 2 predstavljen je uvod u kulturnu baštinu Republike Hrvatske i načini povezivanja novih tehnologija i kulturne baštine kroz istraživanja i dostupna rješenja. Poglavlje 3 daje uvid u korištene tehnologije i programsko rješenje aplikacije. Poglavlje 4 daje uvid u korisničko iskustvo pri korištenju aplikacije te upute za rad s aplikacijom. Poglavlje 5 predstavlja zaključak i osvrt na učinjeno.

1.1 Zadatak završnog rada

Zadatak završnog rada je izraditi mobilnu aplikaciju za Windows Phone platformu koja će omogućiti prikaz i pretraživanje informacija o kulturnoj baštini. U radu će se opisati namjena aplikacije, postupak izrade, te potrebne tehnologije i programski jezici. Ostvareno programsko rješenje aplikacije zasnovano na aktualnim tehnologijama ispitat će se na prikladnom skupu ulaznih podataka i na mobilnim uređajima.

2. RAČUNALNE TEHNOLOGIJE U KULTURNOJ BAŠTINI

Kulturna baština, materijalna ili nematerijalna predstavlja bogatstvo čovječanstva zbog svoje raznolikosti i posebnosti, a njena zaštita i promicanje važan je čimbenik za prepoznavanje, definiranje i afirmaciju kulturnog identiteta. Izradom aplikacije o kulturnoj baštini Republike Hrvatske želi se dokumentirati i promicati njene vrijednosti. Kulturnu baštinu čine pokretna i nepokretna kulturna dobra od umjetničkog, povijesnog, paleontološkoga, arheološkoga, antropološkoga i znanstvenog značenja. Upotrebom današnje suvremene tehnologije i informacijskih znanosti omogućava se približavanje i promicanje kulturne baštine kao starosne, povijesne, kulturne, umjetničke i autentične.

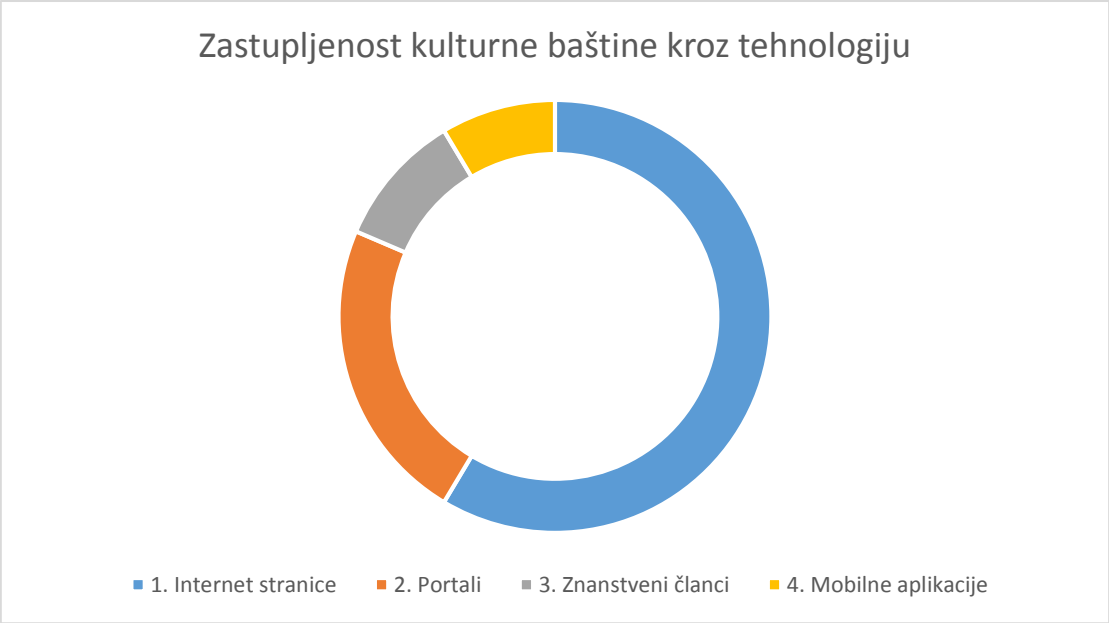
Baština koja je kroz povijest ostala očuvana jedinstvena je i nenadomjestiva, te ostavlja odgovornost očuvanja za sljedeću generaciju. Zahvaljujući snažnom i dinamičnom razvoju moderne tehnologije i znanosti u drugoj polovici 20. stoljeća i početkom 21. stoljeća prvenstveno neprestanom razvoju računarstva i informacijskih tehnologija kao bitan vid zaštite kulture definirao se proces digitalizacije [1]. Digitalizacija kulturne baštine prihvaćena je kao najbolji i najučinkovitiji proces zaštite na svjetskoj razini. Pored zaštite velika prednost digitalizacije je povećanje dostupnosti informacija u digitalnom obliku na dobrobit kako kulturne tako i sveopće javnosti. U toj namjeni digitalizacija je značajno pridonijela i pridonosi izgradnji informacijskog društva i usluga, a posebno usluga i ponuda sadržaja e-kulture [1] koja zahvaljujući suvremenoj tehnologiji postaje sve veći i dostupniji oblik korištenja svih kulturnih sadržaja. Digitalizacija baštine je važna jer predstavlja značajan prostor i važan medij u razmjeni i stjecanju novih znanja, spoznaja, vrijednosti i iskustva. Važnošću i modernizacijom kulturnih dobara znatno se povećava dostupnost, a tako i daljnje povezivanje te povećanje i omogućavanje pristupačnije usluge korisnicima [2].

2.1. Tehnologija u kulturi

Razvojem tehnologije i komunikacije mijenja se i način pristupa korisnicima. Korisniku je potrebno približiti kulturu i baštinu na razumljiv, primamljiv i jednostavniji način koji više odgovara modernoj manipulaciji s podacima. Kulturna baština je oduvijek veoma povezana sa raznim vještinama, zanimanjima, sposobnostima i starim-novim tehnologijama. Komunikacija unutar baštine počinje od kad je i baštine i tehnologija joj u tom razvoju pomaže. Krenuvši od izrade tinte, obrade papira, raznog tiska na tekstil, obrade raznih tehnologija boje, obrade drvoreza do izrade prve fotografije. S erom industrijalizacije dolazi do pojave mehanike,

analogne i digitalne elektronike koja se i danas koristi, a nakon toga pojave radija, filma, suvremene televizije i svih ostalih tehnologija koje su obogatile baštinu i omogućile njen razvoj kakvog danas poznajemo. Tehnologija je omogućila baštinu kakvu danas poznajemo kroz stvaranje, pamćenje, izradu i obradu materijalnih ali i nematerijalnih kulturnih dobara. Tehnologije stvaraju nova znanja i omogućavaju drugačije pristupe kulturnim dobrima. Svaka od tehnologija omogućila je neki novi pristup kulturnoj baštini iz različitih perspektiva i u različitim pogledima i shvaćanjima. Svaka od tehnologija omogućava kreativne i komunikacijske pristupe i pruža nove mogućnosti učenja o baštini. Komuniciranje putem interneta i web stranica već dugo nije novost ni za kulturnu baštinu, a mobilne i desktop aplikacije su trenutno posljednja novost u povijesnom nizu kulturne baštine. Projekti digitalnog dokumentiranja baštine kod nas još uvijek ne obuhvaćaju u već mjeri oblikovanje i proizvodnju digitalnih sadržaja, namijenjenih široj korisničkoj zajednici. Zbog raznih poteškoća i ograničenja kao što su financijska, tehničkih i profesionalna razne ustanove u kulturi ipak se okreću lokalnom digitalnom dokumentiranju i stvaranju različitih kataloga i digitalnih knjiga. Kulturna baština u digitalizaciji ipak pomaže i razjašnjava određene nejasnoće ali i proširuje funkcionalnost baštine pomoću raznih multimedijskih računalnih programa te tako pruža mogućnost povezivanja i pretraživanja cjelokupnih digitalnih materijala prikupljenih o baštini. Kulturna baština donosi mnoge dobrobiti, a kada je potpomognuta novom tehnologijom i digitalizacijom, veliki broj korisnika može pristupiti digitalnim sadržajima, obrazovati se putem interneta, a od nedavno i kroz mobilne aplikacije [2].

U graf na slici 2.1 mogu se vidjeti rezultati dobiveni kroz istraživanje zastupljenosti mrežnih stranica, znanstvenih članaka, portala i mobilnih aplikacija institucija, zajednica i pojedinaca koji se brinu o tradicijskoj baštinu.



Slika 2.1. Zastupljenost kulturne baštine kroz tehnologiju

2.2. Podaci o glagoljičnim spomenicima

Podaci o glagoljičnim spomenicima protežu se od srednjovjekovne do današnje hrvatske kulturne baštine. Dosada nije bilo značajnih promjena i interesa kako bi se modernizirao pristup hrvatskoj jezičnoj povijesti putem mobilnih aplikacija. Napravljen je popis svih glagoljskih spomenika koji su nastali na hrvatskom kulturnom prostoru kako bi se doprinijelo boljem razumijevanju, učenju ali i poznavanju spomenika glagoljice. Aplikacija je nastala kroz projekt u kojem su studenti prikupili podatke i sve potrebne informacije te završno izradili mobilne aplikacije.

Prema [3], popis svih spomenika seže od desetog i jedanaestog stoljeća do kraja devetnaestog i početka dvadesetog stoljeća. Hrvatski filolozi nisu, do sada, pokazali interes za usustavljivanje glagoljičnih spomenika hrvatske kulturne baštine. U manjoj mjeri, neki povjesničari usustavili su najreprezentativnije primjerke počevši od Bašćanske ploče preko Vinodolskog zakona, Istarskog razvoda do Hrvojeva misala, prve tiskane glagoljične knjige, Misala po zakonu rimskoga dvora pa sve do pravnih spisa, pisama i isprava kroz novovjekovno razdoblje. U mobilnoj aplikaciji skupljen je veliki broj glagoljičnih spomenika (preko 200) što je i bio cilj rada i projekta. Zbog nepristupačnosti je bilo gotovo nemoguće skupiti sve glagoljične spomenike kroz povijest, ali ovom aplikacijom je prvi put napravljen ovakav širi popis spomenika. Svakako treba izdvojiti da nisu analizirani svi dokumenti *Acta Croatica*-e čiji je sadržaj pravnog karaktera, a nisu ni svi spomenici liturgijskog i književnog karaktera u

razdoblju od X. do XV. stoljeća analizirani. Kasniji spomenici, koji su već bili tiskani također nisu svi obrađeni, a posebno Matice rođenih iz većine župa na obalnim područjima nisu obrađeni budući da su to uglavnom popisi imena ljudi iako su pisani glagoljičnim pismom. Spomenici su kategorizirani kroz stoljeća budući da je takav format najbolji, ali i najjednostavniji za aplikaciju. Popisani su glagoljični spomenici u razdoblju od X. stoljeća do XX. stoljeća. Također, zbog boljeg vizualnog dojma aplikacije, fotografije koje su korištene za ikone su bili prepoznatljivi primjeri glagoljičnih spomenika pa je za XI. stoljeće korištena fotografija Plominskoga natpisa, XII. stoljeća Bašćanska ploča, XV. stoljeća Hrvojev misal pa sve do XX. stoljeća Vajsov misal. Također, pomno su odabrani primjerci za pojedina stoljeća pa su tako prva dva stoljeća izraziti primjerci kamenih spomenika i fragmenata, a daljnja stoljeća prikazani primjera fragmenata pisanih spomenika na pergameni, zatim iluminiranim i ukrašenim rukopisnim knjigama pa sve do tiskanih izdanja knjiga.

Najreprezentativniji spomenik je svakako Bašćanska ploča na temelju koje su pisani brojni radovi, stručne jezikoslovne i književne analize pa je poslužila kao ikona aplikacije. Bašćanska ploča je simbol bogate hrvatske tropismene i trojezične kulturne baštine, a ova aplikacija obuhvatila je samo područje glagoljice. Spomenici su raznoliki svojim sadržajima, jezičnim elementima, a i samo glagoljično pismo je kroz stoljeća korištenja mijenjalo svoj izgled. Zbog toga je u aplikaciji bilo potrebno definirati kojim jezikom su pisani spomenici pa je tako za jezik liturgijskih spomenika korišten termin „hrvatskostaroslavenski jezik“, zatim za jezik književnosti korišten je „hrvatsko-staroslavenski jezik“, a za pravni jezik korišten je termin „starohrvatski jezik“. Na taj način je jasnije definiran jezik i jezični elementi pojedinih spomenika jer nisu svi spomenici pisani istim jezikom. Tako je, primjerice, Vinodolski zakon kao pravni spis pisan starohrvatskim jezikom, a Hrvojev misal kao liturgijski spis pisan hrvatskostaroslavenskim jezikom, a isti slučaj je s pismom. Prvi spomenici pisani su oblom glagoljicom, primjeri su Plominski natpis, Krčki natpis te Konavoski glagoljski natpis, zatim slijedi razdoblje formativne glagoljice, odnosno predstandardizacijsko razdoblje glagoljice, svakako je najbolji primjer Bašćanska ploča. Zatim slijedi razdoblje ustavne glagoljice (primjer je Vinodolski zakon), potom poluustavne glagoljice (Regula Sv. Benedikta). Kurzivnom glagoljicom pisane su isprave, pisma i razni dokumenti (Darovnica Dragoslava, Isprava Anža Frankapana i druge). U samoj aplikaciji, u dijelu „O aplikaciji“ objašnjeni su i neki drugi razlozi ujednačavanja terminologije kako bi spomenici bili što pregledniji i jednostavniji korisnicima ove mobilne aplikacije. Kao što je već rečeno, bilo je gotovo nemoguće skupiti sve spomenike pisane glagoljicom pa su skupljeni i obrađeni svi oni spomenici koji su u stručnoj literaturi

analizirani i čije su podatke skupili studenti Filozofskog fakulteta u Osijeku. Projekt su osmislili studenti i profesori Filozofskog fakulteta, a zajedničkom suradnjom je realiziran. Budući da se radi o fleksibilnoj metodi rada, odnosno o mobilnoj aplikaciji, ostavljen je prostor i mogućnost nadogradnje aplikacije nakon analize do sada nepoznatih spomenika, a aplikacija se može nadograditi i ispravljanjem podataka postojećih spomenika u slučaju novih saznanja iz hrvatske filologije. Podaci o spomenicima korišteni su iz više izvora, a broj stručne literature kojom su se studenti Filozofskog fakulteta koristili prelazi dvadeset jedinica stručnih radova, znanstvenih članaka te monografija.

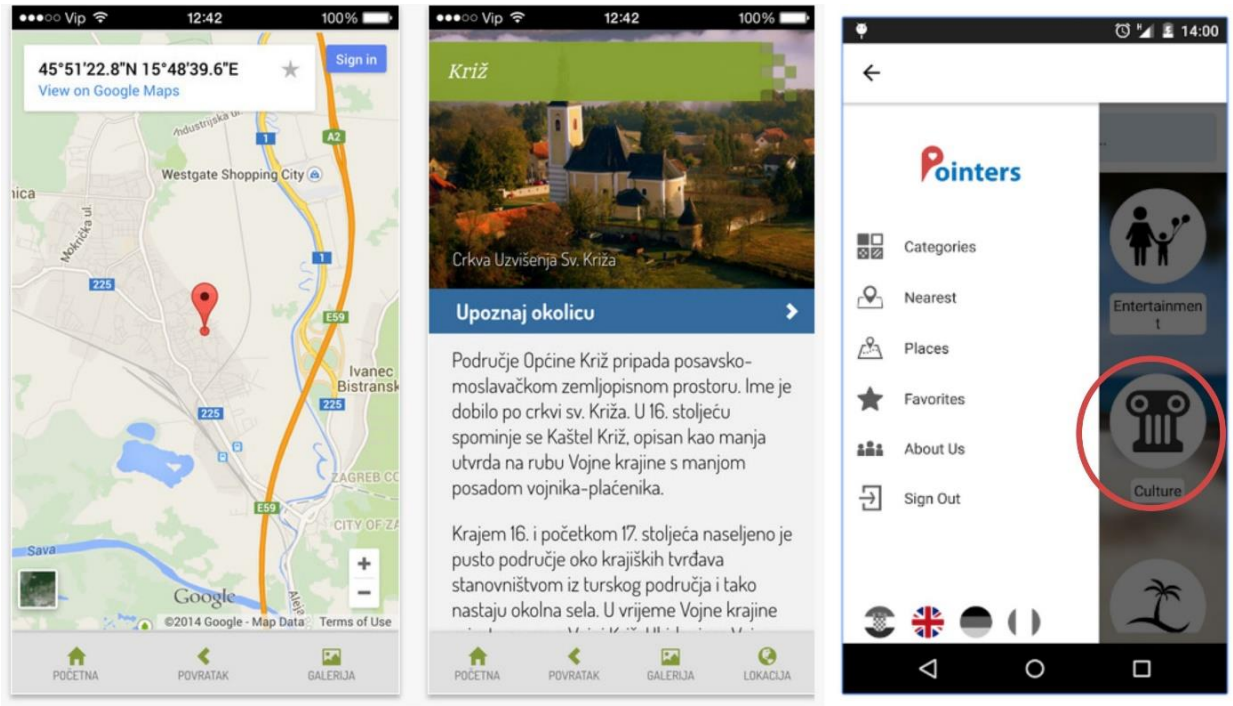
Na slici 2.2. ikonu aplikacije simbolično predstavlja Bašćanska ploča kao jedan od najbitnijih spomenika hrvatske kulturne baštine [3**].**

Slika 2.2. Ikona aplikacije [3]

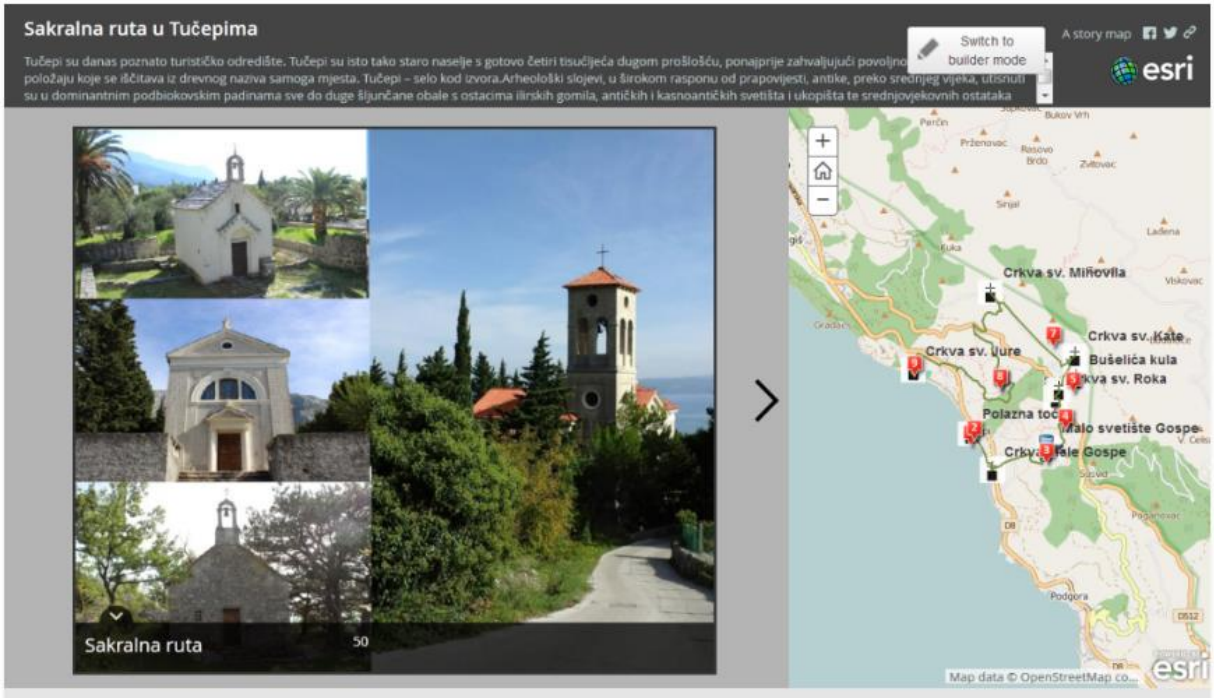
2.3. Slična tehnička rješenja

Provedeno istraživanje pokazuje kako projekti digitalizacije putem aplikacijskih rješenje nisu značajnije zastupljene u hrvatske baštini niti među ustanovama ali ni korisnicima. Krajnji korisnici još ne mogu očekivati aplikacijska rješenja koja obuhvaćaju širu kulturnu baštinu. Također, u Hrvatskoj kulturi još uvijek nije modernizirana i tehnološki napredna tehnologija u promidžbi baštine u a da to nisu web stranice, a tako ni stav ustanova i pojedinaca prema primjeni novih tehnologija u modernizaciji baštine. Osim ovog, provedeno je još jedno službeno istraživanje 2011. godine od strane muzeja grada Zagreba [2] te ne postoji ni jedan projekt ili rješenje koje obuhvaća cjelokupnu ponudu kulturne baštine na jednom mjestu. Prilikom pretraživanja aplikacijskih rješenja za tri mobilne platforme, Windows, Android i IOS, moguće je pronaći aplikacijska rješenja za Android i IOS o bogatstvu i vrijednosti kulturne baštine Zagrebačke županije kroz vrijedne sakralne i druge povijesne građevine te kroz aplikaciju

Pointers Krk o vrijednostima i kulturnoj baštini otoka Krka kao što je prikazano na slici 2.3. Od aplikacijskih rješenja u izradi je web GIS (Geografsko informacijski sustav) mapa u Tučepima prikazanu na slici 2.4.



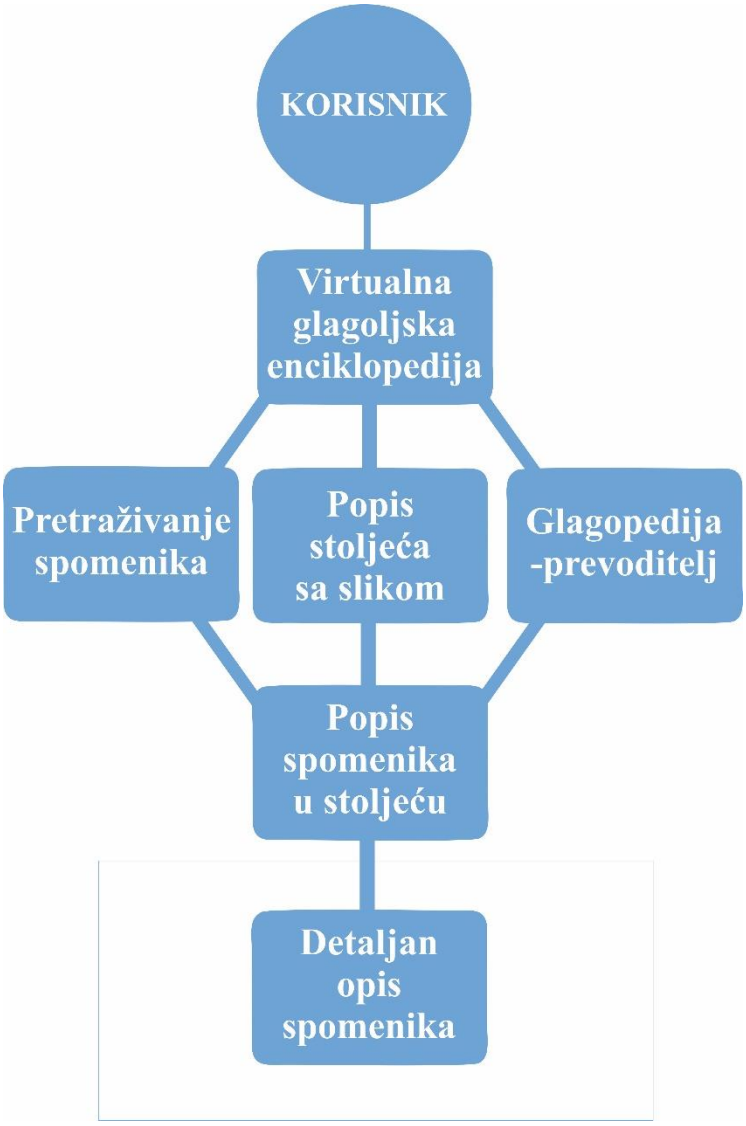
Slika 2.3. Mobilne aplikacije : Kulturna baština i Pointers Krk [4]



Slika 2.4. Web aplikacija: Sakralna ruta u Tučepima (Siječan, 2016.) [5]

2.4. Idejno rješenje aplikacije

Pozivajući se na provedeno istraživanje te uvid u već postojeće mobilne, ali i web aplikacije slične tematike može se zaključiti kako ne postoji ovakvo aplikacijsko rješenje te kako ova aplikacija donosi potpuno novi uvid i prezentaciju glagoljičnih spomenika kroz aktualnu tehnologiju. Aplikacija ispravno radi na mobilnom uređaju Nokia 830 s Windows Phone 10 mobilnim sustavom. Aplikacija se može namijeniti i za neku drugu obitelj uređaja i tako omogućava veću dostupnost korisnicima te joj se time povećava vrijednost. Idejno rješenje aplikacije prikazano je na slici 2.5.



Slika 2.5. Idejno rješenje aplikacije

Prema slici 2.5, prikazane su sve funkcionalnosti aplikacije koje omogućuju korisniku detaljan prikaz strukture aplikacije i put do dohvaćanja krajnje informacije.

3. KORIŠTENE TEHNOLOGIJE I PROGRAMSKO RJEŠENJE APLIKACIJE

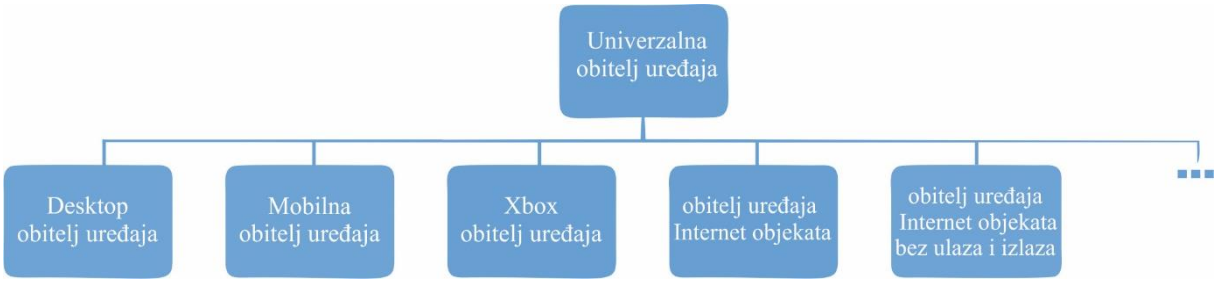
3.1. Programsko rješenje aplikacije

Prvotna inačica aplikacije "Glagopedija" izrađena je za Windows Phone 8.1 platformu. Napretkom tehnologije ali i operacijskog sustava Windows nastala je univerzalna Windows platforma (engl. *UWP*). Platforma je prvotno predstavljena na Windows 8 operacijskom sustavu kao Windows Runtime i već tada je nastao svojevrsni razvoj univerzalnih Windows aplikacija. Kada je Windows Phone 8.1 postao dostupan, Windows Runtime bio je poveznica između Windows i Windows Phone aplikacija. To je usmjerilo i omogućilo programerima razvoj univerzalnih Windows 8 aplikacija koje su dijelile zajednički kod za jedne i druge aplikacije. U samoj jezgri univerzalnih aplikacija nalazi se ideja zajedničke arhitekture za sve Windows aplikacije. Cilj je postići da se aplikacija prilagođava i izvršava na svim Microsoftovim uređaje počevši od mobitela do desktop računala. Windows 10 olakšava razvoj univerzalnih aplikacija koristeći samo jedno aplikacijsko programsko sučelje, jedan paket i jednu trgovinu za sve Windows 10 uređaje. Aplikacijsko programsko sučelje (engl. *application programming interface*) je sučelje za programiranje aplikacija, skup određenih pravila i specifikacija koje programeri slijede kako bi se služili uslugama te resursima operacijskog sustava ili nekog drugog složenog programa kao standardne biblioteke rutina, funkcija, procedura, metoda, struktura podataka, objekata i protokola. Korištenje API-ja omogućava programerima koristiti rad već definiranih programa štedeći vrijeme i trud koji je potreban da se napiše neki složeni program, pri čemu svi programeri koriste iste standarde. Napretkom u operacijskim sustavima, osobito napretkom u grafičkom korisničkom sučelju API je nezaobilazan u stvaranju novih aplikacija.

S ovog stajališta, aplikacije koje su usmjerene kao univerzalne, osim WinRT aplikacijskog programskog sučelja koji je zajednički za sve uređaje, koriste i aplikacijsko programsko sučelje koje uključuje Win32 i .NET aplikacijska programska sučelja, koja su jedinstvena za sve uređaje na kojima se aplikacije pokreću. Univerzalna Windows aplikacija osigurava temeljni API sloj za različite uređaje. To znači da se može stvoriti jedan aplikacijskih paket koji može biti instaliran na širok raspon uređaja. Jednim paketom Windows trgovina postaje jedinstveni distribucijski kanal koji obuhvaća sve uređaje na kojima se aplikacija može instalirati. Budući da univerzalne aplikacije rade na različitim uređajima s različitim karakteristikama i mogućnostima, potrebno je prilagoditi i brinuti o načinu rada sklopovlja svakog zasebnog uređaja pri izradi aplikacije. Svaki uređaj ima svoje zasebno aplikacijsko programsko sučelje.

Moguće je razviti programe koji pristupaju pojedinim aplikacijskim programskim sučeljima kako bi osigurali ispravan rad aplikacije u ovisnosti o karakteristikama određenog uređaja. Prilagodljiva korisnička sučelja i njihove kontrole pomažu programerima prilagoditi korisničko sučelje u ovisnosti o razlučivosti zaslona na kojem se aplikacija prikazuje.

Windows 8.1 i Windows Phone 8.1 aplikacije prilikom izrade usmjerene su na operacijski sustav uređaja, s obzirom na Windows 10 u kojem nije potrebno aplikaciju usmjeriti na određeni operacijski sustav već na skupinu uređaja koje se aplikacijom želi obuhvatiti. Skup uređaja ili „obitelj“ uređaja definira se aplikacijskim programskim sučeljima, karakteristikama sustava, i okruženjem tj. očekivanjima rada aplikacije s obzirom na željeni skup uređaja. To također pokazuje skup uređaja na koje se željena aplikacija može instalirati iz Windows trgovine. Razvrstavanje skupine uređaja na temelju nadređenosti možemo vidjeti na slici 3.1.



Slika 3.1. Obitelji uređaja

Obitelj uređaja predstavlja skup svih aplikacijskih programskih sučelja pojedinog uređaja zajedno s nazivom i brojem verzije uređaja. Obitelj uređaja je osnova operacijskog sustava. Osobno računalo pokreće operacijski sustav koji je baziran na uređajima razvijanim za desktop računala. Mobilni uređaj, prijenosni tableti uređaji, pokreću mobilni operacijskih sustav koji je razvijan za mobilnu obitelji uređaja. Univerzalna obitelj uređaja je posebna. Nije direktno osnova ni jednog operacijskog sustava. Umjesto toga skup API-ja za univerzalnu obitelji uređaja nasljeđuje se ovisno o uređaju na kojem se pokreće. Stoga univerzalna obitelj uređaja omogućava pokretanje na svakom operacijskom sustavu neovisno o uređaju. Svaki uređaj unutar obitelji uređaja dodaje svoj vlastiti API onima koji nasljeđuje. Rezultat unije svih API-ja osigurava uređaju da će zasigurno raditi na operacijskom sustavu koji je baziran na toj obitelji uređaja te slijedno i na svakom uređaju koji pokreće taj operacijski sustav. Jedna od prednosti obitelji uređaja je da se željena aplikacija može izvoditi na bilo kojem uređaju kako je i navedeno uz prilagodljivi kod koji dinamički otkriva i koristi značajke uređaja koji su izvan dosega univerzalnog skupa uređaja.

Odluka o tome na koji se skup ili skupove obitelji uređaja želi usmjeriti aplikacija označava i određene odluke na koje uvrstiti u računicu prilikom razvoja :

- Skup dostupnih API-ja koji su potrebni za rad i razvoj aplikacije.
- Skup API-ja koji se pozivaju i osiguravaju ispravne uvjeta rada.
- Skup uređaja na koje je moguće instalirati aplikaciju i slijedno karakteristike i značajke koje definiraju uređaje.

Dva su osnovna izbora prilikom odabira ili smjera prilikom odabira skupa uređaja:

- API sučelje koje može biti pozvano neovisno o aplikaciji.
- broj uređaja koji aplikacija može obuhvatiti.

Ove dvije značajke uključuju kompromise i međusobno su povezane. Uzmimo za primjer univerzalnu aplikaciju koja je usmjerena na obitelj univerzalnih Windows Phone uređaja i posljedično je dostupna za sve uređaje. Aplikacija koja je namijenjena za obitelj univerzalnih Windows Phone uređaja može pretpostaviti dostupnost onih API-ja koji su bazirani za navedenu obitelj uređaja jer to nam je ciljana skupina. Ostali API-ji moraju se pozivati ovisno o potrebi. Također takva aplikacija mora imati vrlo prilagodljivo korisničko sučelje i sveobuhvatne ulazne karakteristike i sposobnosti jer se može pokretati na širokom rasponu uređaja. Windows mobilne aplikacije su aplikacije koje su predviđene za mobilnu obitelj uređaja i kao takve dostupne su za uređaje čiji je operacijski sustav baziran na mobilnoj obitelji uređaja koja uključuje pametne mobitele, prijenosne dlanovnik uređaje, i ostale slične uređaje. Aplikacije za mobilnu obitelj uređaja mogu pretpostaviti dostupnost svih API-ja unutar mobilne obitelji uređaja, a korisničko sučelje mora biti prilagodljivo ovisno o uređaju. Aplikacija čiji je cilj pokretanje na Internet objekata uređajima(engl. *Internet of Things*) može biti instalirana jedino na uređajima baziranim na obitelji uređaja Internet objekata. Ta aplikacija može biti vrlo specijalizirana svojim korisničkim sučeljem i ulaznim mogućnostima jer se pretpostavlja da će raditi na samo ovoj vrsti uređaja.

Da bi aplikacija bila dostupna za što više uređaja, te da se omogući ispravan rad na što više zasebnih uređaja aplikacija bi trebala biti razvijana za univerzalnu obitelj uređaja. Na taj način, aplikacija automatski cilja sve skupine uređaja koje se temelje na univerzalnim uređajima. Znači da aplikacija radi na svim operacijskim sustavima tih obitelji uređaja, a i na svim uređajima koji pokreću te operativne sustave. Jedini API-ja koji je zajamčeno dostupan na svim tim uređajima je skup definiran po pojedinoj verziji obitelji univerzalnog uređaja koji ciljan,

npr. mobilna aplikacija. S trenutnim izdanjem, ta inačica je uvijek 10.0.x.0. Aplikacija može pozivati i API izvan svoj obiteljske verzije uređaja.

U rijetkim slučajevima može se postići da željena aplikacija radi na svim uređaja iste verzije osim određenim uređajima s određenom verzijom te obitelji uređaja. Uzmimo primjer kako aplikacija ovog rada ima ciljanu skupinu 10.0.x.0 univerzalnih aplikacija. Kada se inačica operacijskog sustava nadogradnjom promjeni na 10.0.x.2, u tom slučaju moguće je odrediti da se aplikacija pokreće normalno na svim uređajima osim na uređajima s verzijom 10.0.x.1 Xbox konzole. Aplikacija će normalno raditi na verziji Xbox 10.0.x.1 i ranijim inačicama ali neće biti dostupna inačica 10.0.x.2. Po zadanom Microsof Visual Studio određen je za Windows.

Po zadanom, Microsoft Visual Studio navodi *Windows.Universal* ciljani uređaj u aplikaciji *Package.appxmanifest* datoteci. Da bi se odredila obitelj uređaja koje aplikacija nudi unutar trgovine, potrebno je ručno konfigurirati elemente *TargetDeviceFamily* u unutar *Package.appxmanifest* datoteke. Zaključno, univerzalne aplikacije se prvenstveno odnose na uređaj, a ne na sustav. Sama obitelj uređaja identificira aplikacijsko programsko sučelje, obilježja sustava i ponašanje koje se može očekivati na uređaju. Distribucija se odvija preko *.AppX* formata čime je osigurana pouzdana instalacija i neprimjetno ažuriranje. Elementi sučelja koriste efektivan broj elemenata slike, što osigurava automatsku adaptaciju u ovisnosti o dostupnom broju elemenata slike (engl. *pixel*). A uz SDK ekstenzije moguće je dodati i posebno aplikacijskog programsko sučelje pojedinoj grupi uređaja.

3.1.1. Korisničko sučelje i univerzalni ulazi

Univerzalna aplikacija se može izvoditi na više različitih vrsta uređaja koji imaju različite ulaze i različite razlučivosti ekrana, gustoću i druge jedinstvene karakteristike. Windows 10 pruža nove univerzalne kontrole, izgled ploče i alati koji pomažu pri prilagodbi korisničkog sučelja uređaju na kojem se aplikacija pokreće. Na primjer, korisničko sučelje aplikacije može se prilagoditi na način da se iskoriste prednosti razlike u razlučivosti ekrana kada se aplikacija pokreće na desktop računalu u odnosu na mobilni uređaj. Neki dijelovi korisničkog sučelja aplikacije automatski će se prilagoditi različitim uređajima. Kontrole kao što su gumbi i klizači automatski se prilagođavaju svim obiteljima uređaja i načinima unosa. Za kvalitetni korisnički dizajna aplikacije, međutim, možda će se aplikacija morati prilagoditi ovisno o uređaju na kojem se izvodi. Na primjer, aplikacija za fotografije treba prilagoditi korisničko sučelje pri radu na malom uređaj kako bi se osiguralo da se korištenje je idealno za korištenje jednom

rukom. Kada je aplikacija za fotografije na stolnom računalu, korisničko sučelje treba se prilagoditi kako bi iskoristilo prostor zaslona.

Windows pomaže prilagoditi sučelje aplikacija za više uređaja koristeći sljedeće značajke :

- Univerzalne kontrole i izgled kontrolne ploče pomaže poboljšati i prilagoditi korisničko sučelje razlučivosti zaslona.
- Zajedničke ulazne kontrole omogućavaju primanje ulaza kroz dodir, olovku, miš ili tipkovnicu ili kontrolu kao što je Microsoft Xbox.
- Alati unutar UWP okruženja olakšavaju razvoj dizajna korisničkog sučelja kako bi se moglo prilagoditi različitim zaslonima s različitim razlučivostima

Univerzalne kontrole i izgledi ploče:

- Windows 10 uključuje nove kontrole kao što su kalendar i podjela pogleda. Kontrola pivot, koja je prije bila dostupna samo za Windows Phone, je sada dostupan za sve obitelji uređaja.
- Kontrole su ažurirane kako bi radile na većim ekranima, prilagođavanje se temelji na broju elemenata slike (engl. *pixel*) zaslona dostupnih na uređaju, a te rad s više vrsta ulaza kao što su tipkovnica, miš, dodir, olovkom, i kontrolora, kao što je Xbox.
- Moguće je prilagoditi ukupni izgled korisničkog sučelja na temelju razlučivosti zaslona uređaja na kojem se aplikacija izvodi. Na primjer, aplikacija za komunikaciju može se pokretati na radnoj površini i uključivati slika-u-slici pozivatelja i kontrole dobro prilagođene za unos mišem na slici 3.3 preuzetoj iz [6].

Slika 3.2. Primjer prilagodbe korisničkog sučelja na različitim razlučivostima zaslona [6]

Kada se aplikacija pokreće na telefonu, zbog manjeg zaslona za rad, aplikacija može eliminirati pogled slika-u-slici i napraviti gumb za poziv veći kako bi se olakšao rad jednom rukom.

Slika 3.3. Primjer prilagodbe korisničkog sučelja na razlučivosti mobilnog zaslona [6]

Kako bi se olakšala prilagodba ukupnog izgleda korisničkog sučelja na temelju količine dostupnog prostora na zaslonu, Windows 10 uvodi različite prilagodbe i kontrole za dizajn.

3.2. Windows 10

Windows 10 operacijski sustav za mobilne uređaje donosi integraciju sa Windows 10 instaliranim na osobnim računalima. U poslovnom okruženju u kojem se trenutno nalazi cjelokupno društvo, vrlo je bitna brzina i protočnost podataka. Kroz smjernice navedene u poglavlju 3.1 Windows 10 operacijski sustav za mobilne uređaje donosi potrebnu brzinu i prilagodljivo grafičko sučelje, zadovoljava tražene uvijete uvodeći nove promjene kao što su obavijesti, organizacija postavki, univerzalne aplikacije koje rade isto na svim uređajima te novosti kao što ikone u postavkama i novi akcijski centar. Akcijski centar nudi tri reda opcija za modificiranje prečaca, uz to nova mogućnost je i korištenje umjetna inteligencije *Cortana* te poboljšani pretraživač i dizajn operacijskog sustava.

Unutar projekta korišten je testni uređaj Nokia 830 i nadogradnjom Windows 10 sustava testirane su navedene mogućnosti. Od korisnih aplikacija Office tim nadogradio je nove mogućnosti u univerzalne aplikacije za Word, Excel i PowerPoint koje rade isto neovisno o uređaju na kojem se koriste. Korisnici Windows Phone 10 uređaja imaju mogućnost koristit aplikaciju „Glagopedija“.

3.3. Rad i manipulacija podacima

Podaci koje su najvažniji dio ovog u rada spremljeni su json formatu. JSON (engl. *JavaScript Object Notation*) je jedan od lakših tekstualnih otvorenih standarda dizajniran za čitljivu razmjenu podataka. Jednostavan je za čitanje i pisanje, kao i za generiranje i analiziranje. JSON je izveden iz JavaScript jezika i koristi se za predstavljanje jednostavnih struktura podataka i asocijativnih nizova, odnosno objekata [7].

JSON je prvi put korišten u State Software poduzeću kojeg je osnovao Douglas Crockford te s kojim JSON i nastaje. Web stranica JSON.org osnovana je 2002. godine, a u prosincu 2005. godine Yahoo počinje koristiti JSON-a u nekima od svojih web servisa dok Google godinu kasnije također počinje sa korištenjem JSON-a za Google Drive web protokole.Koncept JSON-a je jednostavan. JSON prosljeđuje podatke do klijenta kao znakovni niz (engl. *string*) te korištenjem određenih funkcija ovisno o programskom jeziku pretvara znakovi niz u nizove i objekte. JSON je (*engl. Lightweight*) otvoreni format za razmjenu tekstualno baziranih podataka koji je pogodan za upotrebu u web aplikacijama ali i aplikacijama koje pristupaju podacima spremljenim u ovom formatu [7]. JSON je tekstualni format koji je kompletno neovisan jezik s mogućnošću prevođenja izvornog koda za mnoge druge jezike kao što su C, C++, C#, Java, JavaScript, Perl, Python i mnogi dugi što ga čini idealnim za razmjenu podataka.

Također JSON format je često korištena i za serijalizaciju i prijenos strukturiranih podataka između servera i web aplikacija, služeći kao alternativa za XML.

Ekstenzija JSON fromata .json.

JSON je izgrađen je od dvije strukture :

- Zbirki parova naziva/vrijednosti. U različitim jezicima, to je definirano kao objekt, struktura, asocijativni niz, lista sa ključevima, *hesh* tablice. Ova struktura se obilježava sa **{ }** vitičastim zagradama, a podaci u obliku ime/vrijednost su odvojeni sa zarezom. Ime su niska slova pod dvostrukim navodnicima i slijedi vrijednost.
- Niz-uređena lista vrijednosti. U većini jezika to je definirano kao niz, vektor, lista ili slijed. Niz se obilježava sa **[ ]** uglastim zagradama, a članovi niza u odvojeni zarezom.

Univerzalne strukture podataka i tipovi podataka koji se mogu smjestit u ove zbirke su:

- Objekt
- Znakovni niz
- Broj
- Niz
- Točno/Netočno
- Prazno

Možemo napraviti ili čitati JSON datoteku koristeći :

- JavaScriptSerializer (ugrađen unutar .Net okruženja).
- JSON.NET alat (preuzimanje putem Nuget paket sučelja) [7].

U programskom kodu 3.1 i navedenoj strukturi možemo vidjeti jednostavan prikaz JSON objekta{}, uključujući imena / vrijednosti.

```
[ //početak JSON niz

{     "id": "0",

      "ime": "Budimpeštanski odlomci",

      "stoljece": "11",

      "mjesto": "$",

      "godina": "XI. - XII. stoljeće",

      "pismo": "Formativna glagoljica",

      "jezik": "Hrvatskostaroslavenski jezik",

      "sadrzaj": "Ispisan je dio žitja Simeona Stylita/Šimuna Stupnika",

      "velicina": "2 ulomka pergamenta",

      "zanimljivosti": "Najstariji očuvani hrvatskoglagoljski hagiografski
tekst"

   }, // JSON objekt

{ "id": "1", ...//nastavak
```

Programski kod 3.1. Sadržaj podaci.json datoteke

Niz može biti zbirka bilo čega, uključujući i gore navedene objekte. Ovi objekti imaju deset elemenata, id, ime, stoljece, mjesto, pismo, jezik, sadrzaj, velicina, zanimljivost.

```
public class Spomenik
    {
        public string id { get; set; }
        public string ime { get; set; }
        public string stoljece { get; set; }
        public string mjesto { get; set; }
        public string godina { get; set; }
        public string pismo { get; set; }
        public string jezik { get; set; }
        public string sadrzaj { get; set; }
        public string velicina { get; set; }
        public string zanimljivosti { get; set; }

   }
```

Programski kod 3.2. Klasa spomenik sa definiranom varijablama

Klasa Spomenik se sastoji se od deset pristupnika (engl. *Acessors*) s istim imenom kao i unutar JSON niza. Klasa Spomenik pristupa elementima unutar JSON niza. GET i SET su pristupne metode koje omogućavaju pristup podacima i informacijama u privatnim poljima dok oni imaju javno deklarirana svojstva. *GET* metodom pristupamo elementu, a SET metodom postavljamo vrijednost. { get; set; } je automatsko svojstvo koje olakšava pisanje koda. U primjeru se može vidjeti stvarna primjena.

```
private string ime;
public string Ime
{
    get
    {
        return this.ime;
    }
    set
    {
        this.ime = value;
    }
}
```

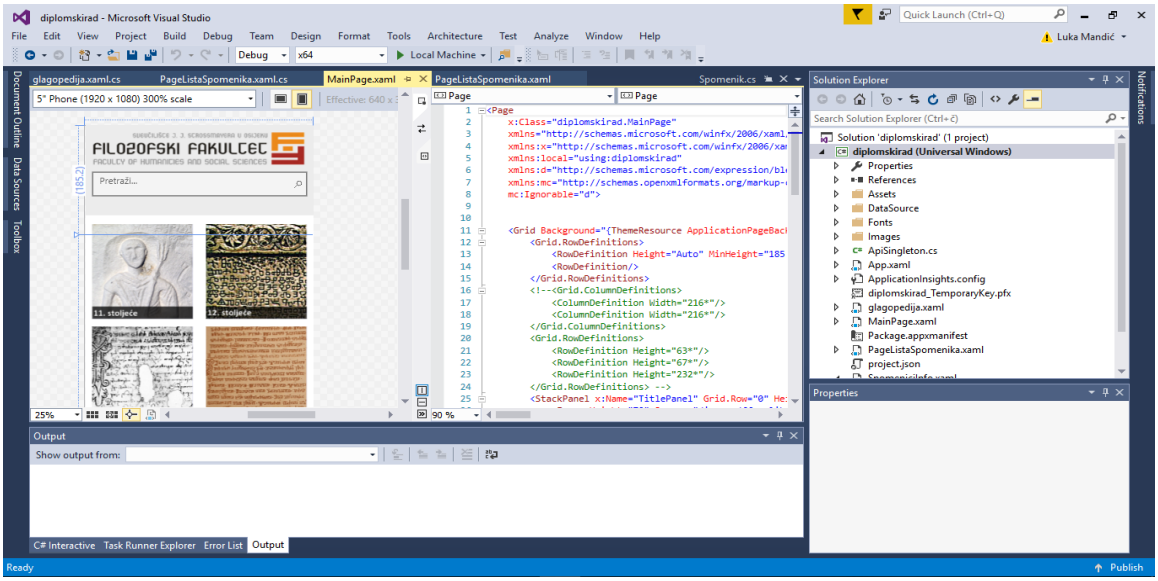
Programski kod 3.3. Ključ/vrijednost

3.4. Arhitektura aplikacije

Aplikacija je izrađena je u *Microsoft Visual Studio 2015* razvojnom okruženju, a pisana je u C# programskom jeziku te je kao predložak za izradu aplikacije korišten je *Windows Blank App (universal application).*

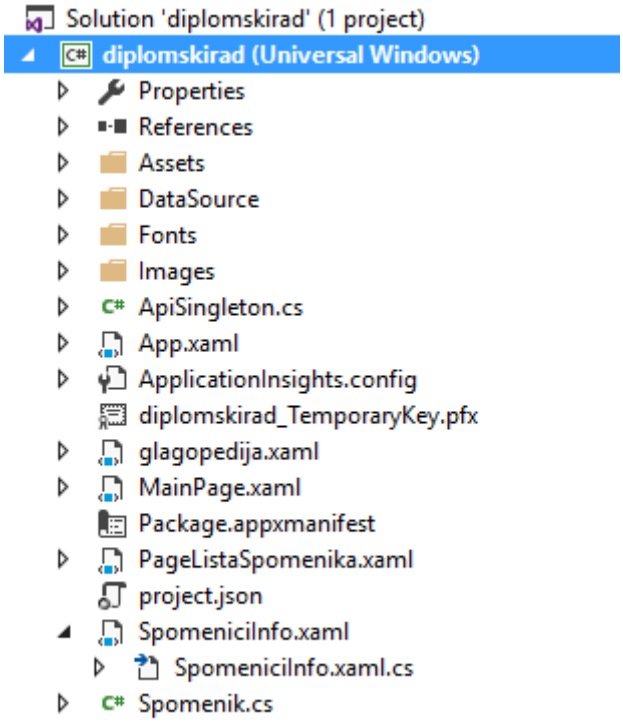
Važno teorijsko predznanje i vještina rada unutar Visual Studio presudna je za razvoj Windows apliakcija. Microsoft Visual Studio je platforma za razvoj računalnih programa. Visual Studio je platforma s korisničkim sučeljem namijenjenim kvalitetnijem radu programera. Podržava rad s raznim programskim jezicima, a najčešći su Microsoft Visual Basic, C++ i C# koji je korišten pri izradi projektne aplikacije. Microsoftov razvojni alat Visual Studio koristi se za razvoj aplikacija i dostupan je velikoj populaciji jer njegova jednostavnija i besplatna inačica Visual Studio Express omogućava početnicima brže svladavanje programerskih vještina i razvoj manjih aplikacija za učenje i jednostavniju upotrebu. Sučelje Visual Studio okruženja možemo vidjeti na slici 3.5 [8] .

Visual Studio podržava i dizajnerski pristup izradi programskog koda – Microsoft Blend. Microsoft Blend omogućava i olakšava dizajniranje aplikacije raspoređivanjem grafičkih ikona koje simboliziraju dijelove koda pri čemu se upisuju samo neke osnovne značajke tih dijelova koda.



Slika 3.4. Programsko sučelje unutar Visual Studio okruženja

Projekt je nazvan diplomski rad i može se podijeliti na nekoliko glavnih dijelova kao što je prikazano na slici 3.6.

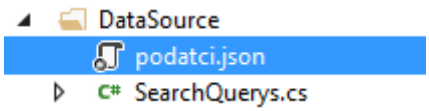


Slika 3.5. Cjeline aplikacije

Aplikacija je prvotno zamišljena za mobilne uređaje te je sučelje aplikacije dizajnirano za ciljana skupina uređaja iz mobilne obitelji.

Neki od tih dijelova su automatski kreirani pri izradi projekta, kao što je mapa *Assets, References, Properties, Package.appxmanifest* te *App i MainPage.xaml* dok su ostali dijelovi aplikacije dodani tokom samog razvoja. Struktura aplikacije se temelji na programskom sučelju koje se sastoji od glavnog pozadinskog programa *MainPage.xaml.cs* koji sadrži kod pozadinske logike aplikacije.

Podatci s kojim je rađeno su spremljeni u datoteku *DataSource* pod nazivom podatci.json kao što je prikazano na slici 3.7.



Slika 3.6. Sadržaj datoteke DataSource

Aplikacija se sastoji od glavnog korisničkog sučelja s grafičkim prikazom stoljeća te tražilicom koja korisniku omogućuje pretragu spomenika po imenu i prevoditeljem koji vrši prijevod latiničnog teksta u glagoljični. U datotekama *Assets i Image* nalaze se slike i ikone različitih dimenzija koje su potrebne za ispravan prikaz aplikacije te slike koje se koriste unutar aplikacije, kao što su slike stoljeća te slika zaglavlja aplikacije na početnoj stranici. U datoteci *Fonts* spremljeni su fontovi koji se koriste unutar aplikacije, izvorni font te glagoljični font koji se koristi u prevoditelju. *ApiSingleton.cs* klasa najvažniji je dio aplikacije. *ApiSingleton.cs* klasa koristi *Task* klasu koja omogućava pretvorbu sinkronih metoda u asinkroni tip. Asinkorno programiranje se koristi kako bi se osigurao kvalitetniji i brži rad aplikacije tj. kako bi se oslobodila glavna programska nit (engl. *Thread*). Sve kontrole kao što su *TextBox, Button* itd. po standardnim postavka su vezane za glavnu programsku nit i izvršavanje tih kontrola se uvijek vrši na glavnoj programskoj niti. U navedenom slučaju imamo funkciju *readFile()* koja pronalazi datoteku *podaci.json* te asinkrono izčitava datoteku pomoću *streamReader* klase pohranjujući podatke u *JSONString* polje. Ukoliko funkcija *readFile()* ne bi bila asinkrona prilikom pokretanja aplikacije događalo bi se da prvo izvršavaju sinkrono sve standardne kontrole unutar aplikacije te bih tek onda na red došlo čitanje i generiranje, serijalizacija JSONa. To može biti vrlo loše za korisničko iskustvo aplikacije. Upravo zato postoji asinkroni način programiranja, koji sinkrone metode pretvara u asinkrone na način da se u određenom trenutku pozove ključna riječ *await* te se stvara nova nit na kojem se izvršava funkcija ili metoda te se oslobađa glavnu nit kako bi ispravno radila.

Ukoliko se ne pozove ključna riječ *await* metoda ili funkcija iako je asinkrona neće se izvršavati na drugoj niti, tj. samo dio koji se nalazi pod ključnom riječi *await* izvršavat će se asinkrono. Stil asinkronog programiranja koji se koristi u radu zove se TAP (engl. *Task Async Pattern*) i prije samog principa, važno je poznavati osnovna pravila :

- Asinkrona metoda uvijek ima povratni tip *Task <T>*, gdje je T tip podataka koju metoda vraća
- Parametri asinkrone metode moraju biti isti kao i kod sinkrone metode
- Navedena asinkrona metoda vraća *Task* u kojem se nalazi vrijednost koju je zapravo vratila metoda [8].
- Nakon pristupnog modifikatora slijedi ključna riječ *async* [9].

```
// parsiranje jsona
        public async Task<String> readFile()
        {
            try
            {

                var fileStream = File.OpenRead("DataSource/podatci.json");

                using (var streamReader = new StreamReader(fileStream, Encoding.UTF8))
                {
                    JSONString = streamReader.ReadToEnd();
                }

                listaSpomenika =
Newtonsoft.Json.JsonConvert.DeserializeObject<List<Spomenik>>(JSONString); //*

                // razvrstavanje spomenika *

                    foreach (Spomenik spomenik in listaSpomenika)
                 {
                    if (spomenik.stoljece == "11")
                    {
                        jedanaestoLista.Add(spomenik);
                    }
                ...//nastavak
                }
                foreach (Spomenik spomenik in listaSpomenika)                {
                    if (spomenik.stoljece == "17")
                    {
                        sedamnaestoLista.Add(spomenik);
                    }
                }
                foreach (Spomenik spomenik in listaSpomenika)
                {
                    if (spomenik.stoljece == "18")
                    {
                        osamnaestoLista.Add(spomenik);
                    }
                }
                foreach (Spomenik spomenik in listaSpomenika)
                {
                    if (spomenik.stoljece == "19")
                    {
                        devetnaestoLista.Add(spomenik);
                    }
                }
                foreach (Spomenik spomenik in listaSpomenika)
                {
                    if (spomenik.stoljece == "20")
                    {
                        dvadesetoLista.Add(spomenik);
                    }
                }
            }
            catch (AggregateException e)
            {
                return e.ToString();
            }
            return JSONString;
        }
```

Programski kod 3.3. Čitanje, parsiranje i sortiranje JSON datoteke.

Ukoliko se želi sačuvati sadržaj objekta u datoteci, poslati objekt drugom procesu ili prenositi objekt unutar aplikacije, potrebno je brinuti o tome kako je objekt predstavljen, jer ga je potrebno trebati pretvoriti u drugi format. Serijalizacija je proces konverzije stanja objekta u oblik koji je postojan i koji se može prenositi [10].

```
listaSpomenika = Newtonsoft.Json.JsonConvert.DeserializeObject<List<Spomenik>>(JSONString)
```

Programski kod 3.4. Čitanje, parsiranje i sortiranje JSON datoteke.

Komplement serijalizacije je deserijalizacija objekta omogućuje da se stvori novi objekt na temelju određenih podataka. U osnovi deserijalizacija pretvara tok podataka, tj. datoteku u objekt. Zajedno ovi procesi omogućuju trajnu pohranu i prijenos podataka. U aplikaciji se koristi *Newtonsoftova* metoda *JsonConvert.DeserializeObject* kojom se omogućava pretvorba JSONString polja u generičku listu *List<Spomenik>* gdje je *Spomenik* tip podatka koji metoda vraća [11].

Unutar klase *PageListaSpomenika* pristupa se JSON objektima i pretvara ih se u podatkovni niz te se popunjava lista spomenika koja se prikazuje ovisno o traženom stoljeću i željenom atributu.

```
public void setListView() //postavi listu
        {
            if (dataList != null)
            {
                JArray obj = JArray.FromObject(dataList);

                for (int i = 0; i < obj.Count; i++)
                {

        JObject row = JObject.Parse(obj[i].ToString()); // pristup objektima JSona

                    var item = new Spomenik();
                    item.ime = row["ime"].ToString();
                    item.godina = row["godina"].ToString();
                    item.id = row["id"].ToString();
                    item.jezik = row["jezik"].ToString();
                    item.mjesto = row["mjesto"].ToString();
                    item.pismo = row["pismo"].ToString();
                    item.sadrzaj = row["sadrzaj"].ToString();
                    item.stoljece = row["stoljece"].ToString();
                    item.velicina = row["velicina"].ToString();
                    item.zanimljivosti = row["zanimljivosti"].ToString();

                    list.Items.Add(item);
```

Programski kod 3.5. Metoda *setListView*

Jedan od zahtjeva aplikacije je pretraživač koji je implementiran kroz *AutoSuggestBox* kontrolu. Klasa *SearchQueries* pristupa podacima za pretragu i kroz metodu *IEnumerable* obavlja *fuzzy* pretraživanje.

```
public class SearchQueries
    {
       List<Spomenik> dataList11;

        public List<Spomenik> GetResults(string query)
        {
            return dataList11;
        }

        public void SetResults()
        {
            dataList11 = ApiSingleton.getInstance.getListaSpomenika();

        }

        public SearchQueries()
        {
            SetResults();
            }

        public IEnumerable<Spomenik> GetMatchingSpomenici(string query)
        {
            return dataList11
                .Where(p => p.ime.IndexOf(query,
StringComparison.CurrentCultureIgnoreCase) > -1)
                .OrderByDescending(p => p.ime.StartsWith(query,
StringComparison.CurrentCultureIgnoreCase));
        }
```

Programski kod 3.6. Sadržaj *SearchQueries* klase

Fuzzy pretraživanje je tehnika za pronalaženje uzoraka tipa string koji odgovaraju traženom upitu približno. Dozvoljava pretraživanje pojmova koji sadrže slova slična unesenim pojmovima pretraživanja. Problem približnog string podudaranja dijeli se na dva pod problema

- pronalaženje približnog pod niza tražene riječi
- pronalaženje riječi unutar zadanog niza koje odgovaraju traženom uzorku točno ili približno [12].

Podudaranje uzorka mjeri se u broju primitivnih operacija potrebnih za pretvorbu stringa u potpuno podudaranje. Ovaj broj se zove uređivanje udaljenosti između niza i uzorka. Uobičajene primitivne operacije su :

- Umetanje : Baš -> Baščanska ploča
- Brisanje : Bašćansk -> Bašćanska ploča, Bašćanski ostrišci
- Zamjena : Bašlkćan -> Bašćanska

Najčešća primjena približnih podudaranja donedavno je bila provjera pravopisa, a približno podudaranje koristi se i za pretraživanje DNA baze podataka. Približno podudaranje također se koristi za filtriranje neželjene pošte. Ovakvo znakovno podudaranje ne može se koristiti za većinu binarnih podataka, kao što su slike i glazbe [12]. Kako bi pretraživanje radilo ispravno, potrebno je implementirati *AutoSuggestBox* događaje unutar glavne stranice gdje se pretraživač koristi.

- *TextChanged* događaj pokreće se svaki put pri upisu korisnika unutar *AutoSuggestBox* kontrole.
- *QuerySubmitted* događaj pokreće se nakon što je poslan zahtjev.
- *SuggestionChosen* događaj pokreće se kada je završio zahtjev odabirom stavke i pregledavanjem unutar *AutoSuggestBox* kontrole [13].

```
private void AutoSuggestBox_TextChanged(AutoSuggestBox sender,
AutoSuggestBoxTextChangedEventArgs args)
        {
            if (args.Reason == AutoSuggestionBoxTextChangeReason.UserInput )
            {

                var matchingSpomenici =squeries.GetMatchingSpomenici(sender.Text);
                sender.ItemsSource = matchingSpomenici.ToList();
            }
        }
```

Programski kod 3.7. AutoSuggestBox_TextChanged događaj

```
private void AutoSuggestBox_QuerySubmitted(AutoSuggestBox sender,
AutoSuggestBoxQuerySubmittedEventArgs args)
        {
            if (args.ChosenSuggestion != null)
            {
                SelectSpomenici(args.ChosenSuggestion as Spomenik);
            }
            else
            {
                var matchingSpomenici =
squeries.GetMatchingSpomenici(args.QueryText);
                if (matchingSpomenici.Count() >= 1)
                {
                    SelectSpomenici(matchingSpomenici.FirstOrDefault());
                }
            }
        }
```

Programski kod 3.8. *AutoSuggestBox_ QuerySubmitted* događaj

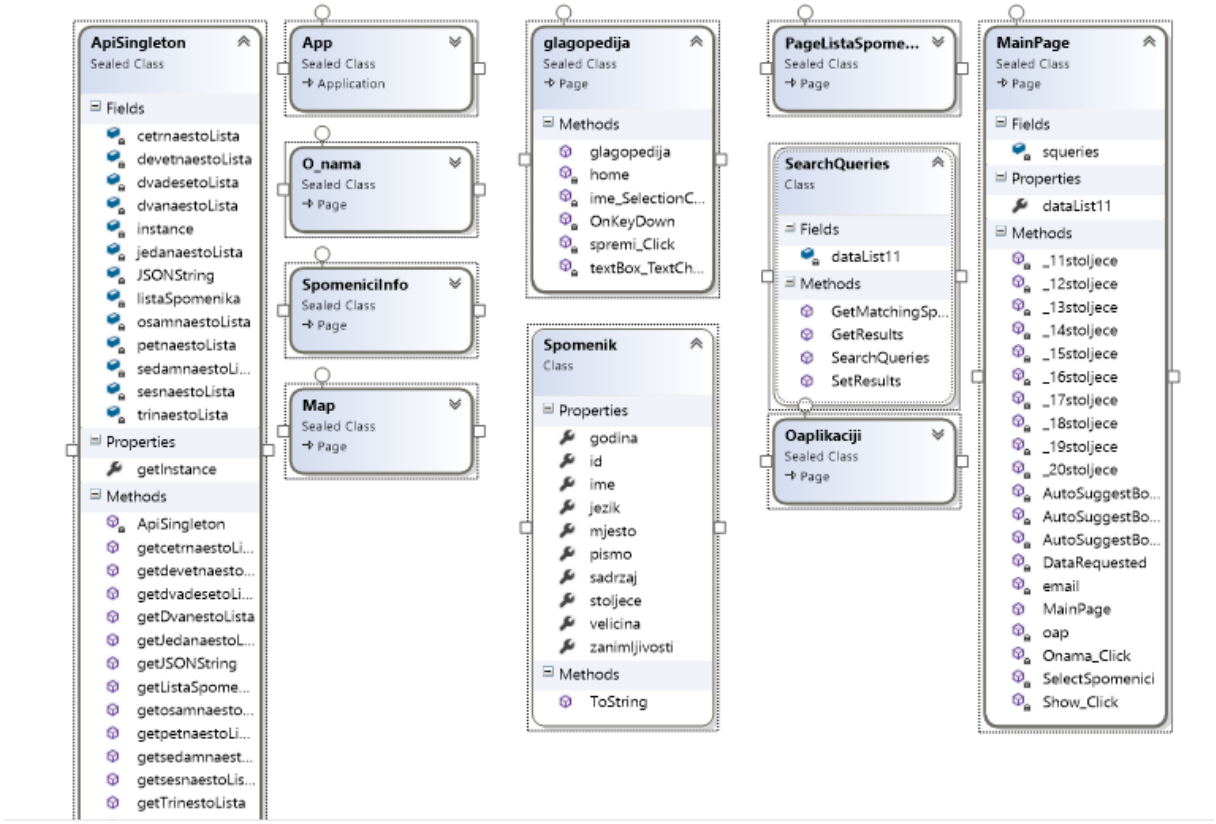
Funkcija unaprijed definiranim argumentima provjerava jeli odabran zahtjev korisnika te ukoliko nije odabran, prikazuje listu spomenika po unaprijed određenim parametrima. Nakon što je zahtjev poslan *SuggestionChosen* događaj pokreće se i omogućava pregledavanjem odabrane stavke unutar *AutoSuggestBox* kontrole.

```
private void AutoSuggestBox_SuggestionChosen(AutoSuggestBox sender,
AutoSuggestBoxSuggestionChosenEventArgs args)
        {
            var dataObject = args.SelectedItem as Spomenik;
           sender.Text = string.Format("{0}", dataObject.ime);
            }
```

Programski kod 3.9. *AutoSuggestBox_ QuerySubmitted* događaj

3.4.1. Model podataka

Slika 3.8. prikazuje model aplikacije i razvoj funkcionalnosti aplikacije kroz dijagram klase automatski kreiran unutar Visual Studio okruženja.



Slika 3.7. Dijagram klase kreiran kroz Visual Studio sučelje

ApiSingleton.cs klasa razvijana je kao *Singleton* model. Singleton ima jednu instancu klase u toku izvršavanja te ima specifičnost da joj konstruktor nikada nije dostupan, što znači da se ni u jednom djelu ostatka koda Singleton klasa ne može instancirati. Klasa ima privatan konstruktor. Kako bi se pristupilo metodama i sadržaju Singleton "klase" postoji javna metoda *getInstance()* koji služi da se pozivaču metode preda kontrolu nad *Singleton* klasom. Ova metoda se koristi u klasi *PageListaSpomenika* i *PageInfo*. Klasa Spomenik sadrži *property-e* koji kontroliraju pristup poljima. Ovim varijablama se pristupa isključivo preko *property-a.* Također imamo glavnu *MainPage.cs* klasu koja sadrži funkcije za navigaciju prema stoljećima ovisno o zahtjevu korisnika ali i izbornike te. Također *MainPage.cs* klasi se poziva *SearchQuerises.cs* funkcija i definira se sadržaj i način pretraživanja za *Autosuggestbox.* U klasi *Glagopedija* koriste se *Textbox i Textblock* u kojem se postavlja glagoljični font te se prilikom upisa u odvija prijevod *prijevod.Text = upis.Text.* Također ova klasa sadrži kao i klasa *SpomeniciInfo* mogućnost spremi, koja dohvaća sve što se nalazi unutar mreže korisničkog sučelja stranice (engl. *Grid*) te omogućava spremanje kao sliku u .png formatu. *SpomeniciInfo* sadrži (poziva se *GetInstance()*) objekt *DataObject* tipa *Spomenik* koji pomoću *OnnavigateTo* funkcije dohvaća odabrani spomenik te prosljeđuje parametar, na primjer *property:* ime određenom *Textblocku* gdje se prikazuje. Klasa *Map* pomoću mape generirane u XAML kontrolama i funkcije *AddMapIcon()* prikazuje lokaciju pomoću širine i dužine. Zbog nepotpunih podataka lokacija spomenika, nije bilo moguće implementirati i omogućiti prikaz svih spomenika na mapi stoga je razvijena samo demo inačica.

```
private void AddMapIcon()
        {
            Geopoint myPoint = new Geopoint(new BasicGeoposition() { Latitude =
44.977987, Longitude = 14.738357 });
            //create POI
            MapIcon myPOI = new MapIcon { Location = myPoint, NormalizedAnchorPoint
= new Point(0.5, 1.0), Title = "Trenutna pozicija", ZIndex = 0 };
            // add to map and center it
            MapControl1.MapElements.Add(myPOI);
            MapControl1.Center = myPoint;
            MapControl1.ZoomLevel = 10;
        }
```

Programski kod 3.9. Prikaz lokacije spomenika na mapi

4. SLUČAJEVI KORIŠTENJA I DIZAJN APLIKACIJE

U tablici 4.1. definirani su opći zahtjevi korisnika s obzirom na prioritete i zahtjeve projekta. Oznaka UC (engl. *User case*) označava moguće korisničke slučajeve prilikom rada u aplikaciji.

Tablica 4.1. Opći zahtjevi korisnika

ID	Status	Prioritet	Opis	UC
			Opći zahtjevi korisnika	
1	A	1	U sustavu postoje različiti spomenici	-
2	A	1	Za svaki spomenik različiti su podaci.	-
3	A		Svi podaci spremljeni su u lokalnoj datoteci podaci.json	
4	A	1	Korisnik može odabrati stoljeće	UC1
5	A	1	Korisnik može pregledati listu spomenika u stoljeću	UC2
6	A	1	Korisnik može pregledati određeni spomenik kao tekst	UC3
7	A	2	Korisnik može spremiti opis spomenika ili prijevoda kao sliku.	UC4
8	A	3	Korisnik može vidjeti prikaz određenog spomenika na mapi	UC5
9	A	3	Korisnik može prevoditi latinični tekst u glagoljični	UC6
10	A	1	Korisnik može pretraživati spomenike po imenu kroz ugrađeni pretraživač	UC7
11	A	1	Sustav treba biti napravljen za Windows Phone 10 mobilnu platformu	-

4.1. Slučajevi korištenja

U tablicama 4.2 do 4.8 definirani su korisnički slučajevi pod oznakama UC (engl. *User case*) koji označavaju moguće slučajeve rada u aplikaciji.

Tablica 4.2. Korisnički slučaj 1

ID slučaja	UC1
Ime	Odabir stoljeća
Opis	Korisnik odabire stoljeće za pregled spomenika
Preduvjet	Nema
Glavni scenarij	1. Korisnik ulazi u aplikaciju 2. Korisnik odabire stoljeće 3. Sustav otvara korisniku novu stranicu sa listom spomenika
Alternativni scenarij	5. Došlo je do greške prilikom klika stoljeća 1. Povratak na korak 2 glavnog scenarija

Tablica 4.3. Korisnički slučaj 2

ID slučaja	UC2
Ime	Pregled liste spomenika
Opis	Korisnik vidi popis spomenika i odabire željeni sa liste
Preduvjet	Postavljena je odabir stoljeća (UC1)
Glavni scenarij	1. Korisnik odabire spomenik 2. Korisnik se odabire prevoditelj u izborniku 3. Sustav otvara korisniku novu stranicu sa informacijama o traženom spomeniku (UC3)
Alternativni scenarij	1. Korisnik odabire prevoditelj u izborniku 1.1 Sustav otvara korisniku novu stranicu za prijevod 2. Korisnik se vraća na prethodnu stranicu

Tablica 4.4. Korisnički slučaj 3

ID slučaja	UC3
Ime	Pregled spomenika
Opis	Korisnik pregledava spomenik Info stranicu
Preduvjet	UC1- UC2 ili pretraživač (UC8)
Glavni scenarij	1. Korisnik odabire traženi spomenik 2. Sustav otvara stranicu sa detaljima

Alternativni scenarij	1. Korisnik sprema spomenik 2. Korisnik se vraća na početnu stranicu 3. Korisnik pregledava mjesto na mapi

Tablica 4.5. Korisnički slučaj 4

ID slučaja	UC4
Ime	Spremanje informacija ili prijevoda u .png formatu
Opis	Korisnik želi spremiti info ili prijevod za daljnje korištenje ili dijeljenje
Preduvjet	UC1- UC2 ili pretraživač (UC8)
Glavni scenarij	1. Korisnik odabire spremi iz izbornika 2. Korisnik odabire ime i datoteku na uređaju u koju želi spremiti
Alternativni scenarij	1. Korisnik iz izbornika odabire drugu opciju

Tablica 4.6. Korisnički slučaj 5

ID slučaja	UC5
Ime	Mapa
Opis	Korisnik želi vidjeti lokaciju spomenika na mapi -napomena podaci nisu dostupni- napravljena demo inačica
Preduvjet	Odabran željeni spomenik
Glavni scenarij	1. Korisnik odabire spomenik 2. Korisnik odabire lokacija u izborniku 3. Sustav otvara stranicu sa mapom i označava lokaciju
Alternativni scenarij	3. Korisnik odustaje od lokacije 4. Korisnik ima pristup mreži 1. Lokacija potpunija i detaljnija

Tablica 4.7. Korisnički slučaj 6

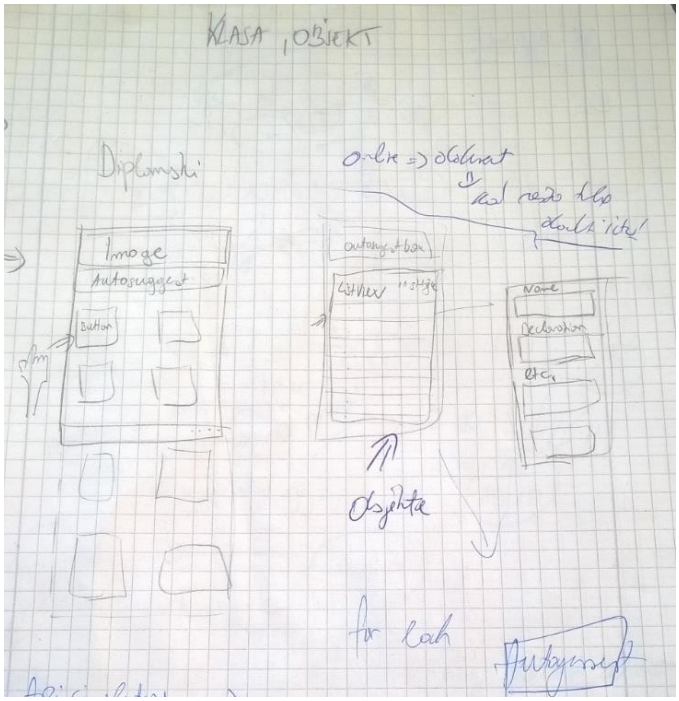
ID slučaja	UC6
Ime	Prijevod
Opis	Korisnik prevodi tekst u glagoljicu
Preduvjet	Korisnik mora unijeti željeni tekst
Glavni scenarij	1. Korisnik odabire prevoditelj iz izbornika 2. Korisnik opisuje tekst 3. Sustav prilikom upisivanja automatski prevodi
Alternativni scenarij	Korisnik unosi znakove koje nije moguće prevesti jer ne postoje unutar glagoljičnog pisma

Tablica 4.8. Korisnički slučaj 7

ID slučaja	UC7
Ime	Pretraživač
Opis	Korisnik želi pretraživati spomenike po imenu
Preduvjet	Dostupnost spomenika
Glavni scenarij	1. Korisnik upisuje riječi u pretraživač 2. Sustav prikazuje popis s kojim se upit podudara 3. Korisnik odabire željeni spomenik
Alternativni scenarij	3. Došlo je do greške prilikom upisa 1. Sustav ne prikazuje traženi rezultat 2. Povratak na 1. korak glavnog scenarija

4.2. Dizajn aplikacije

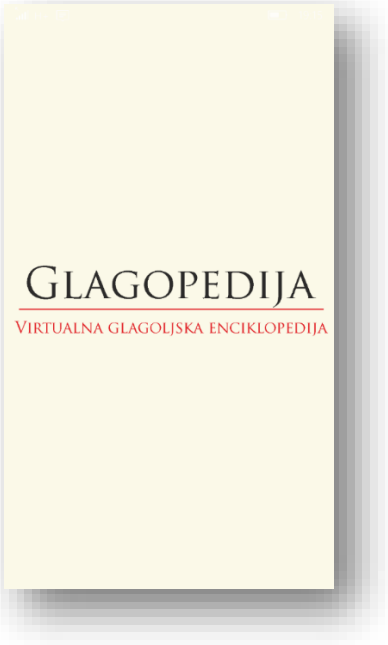
Prvi nacrt aplikacije na slici 4.1 izrađen je kao skica na papiru inspiriran vodećim smjernicama dizajna Windows aplikacija, dok je inspiracija za završnu verziju pronađena u dizajnu nove Windows Phone aplikacije Zagrebačke banke. Nakon skice sljedeći nacrt aplikacije rađen je u Microsoft Blend sučelju s XAML opisnim jezikom. XAML je deklarativni jezik koji se koristi kod Windows Phone-a kako bi se opisao izgled korisničkog sučelja kroz svojstva klasa međutim osim takve statike XAML upravlja i dinamikom kao što su animacije, transformacije, definicija 2D i 3D objekata i još mnogim drugim mogućnostima. Kada se XAML koristi kod Windows Phone-a tada je odgovoran za tijek izvođenja i opis izvođenja logike aplikacije.



Slika 4.1. Nacrt aplikacije na papiru

Visual Studio podržava i dizajnerski pristup izradi programskog koda. Blend je alat u kojemu kod ne treba pisati, nego se programiranje može obavljati raspoređivanjem grafičkih ikona koje simboliziraju dijelove koda pri čemu se upisuju samo neke osnovne značajke tih dijelova koda, gdje također sve promjene ovise isključivo o promjenama koje se naprave u Microsoft Blendu te nakon toga dovoljno je samo spremiti promjene i one se automatski integriraju u XAML kod. *Microsoft Expression Blend for Visual* studio 2015 je sastavni dio dizajnerskog paketa Visual Studia**,** a glavna zadaća mu je izrada aplikacija za Windows Phone. Podržava sve standardne tehnologije kao što su Silverlight, XAML, C#, Visual Basic i WPF [9].

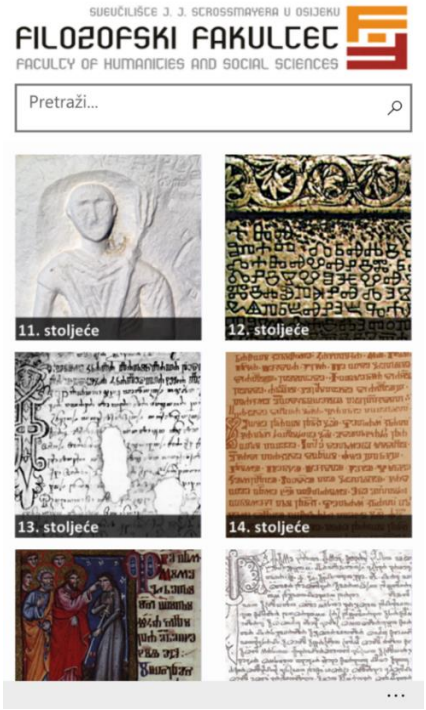
Blend je opremljen različitim predlošcima koji olakšavaju dizajn same aplikacije te omogućavaju dizajneru ili programeru pomoću koda definirati dizajn u aplikaciju. Blend omogućava i dodavanje već gotovog dizajna kroz Photoshop i Illustrator alate u sam program kako bi se dizajnerima omogućila brža i lakša izrada konačnog dizajna. Nakon što je završetka dizajna, potrebno ga je samo uklopiti unutar programa dok će Blend generirati sam potreban dio koda [14]. Dizajn aplikacije „Glagopedija“ napravljen je u Blendu dok su slike i ikone aplikaciji izrađene u CorelDraw programu. CorelDraw je računalni program za uređivanje vektorske grafike i u njemu je izrađena ikona aplikacije te pozadinska boja aplikacije kao i slika koja se pojavljuje prilikom učitavanja aplikacije (engl. *Splash screen image*) prikazana na slici 4.2.



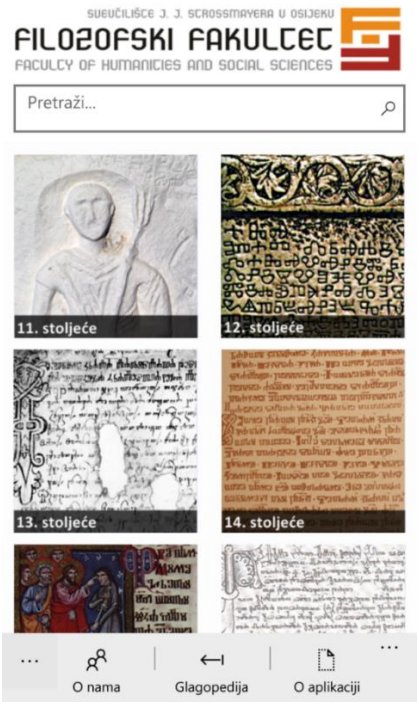
Slika 4.2. Uvodna slika aplikacije

4.2.1. Upute za korištenje

Korisnik ulazi u aplikaciju i odabire stoljeće (Slika 4.3). Osim odabira stoljeća korisnik iz izbornika može odabrati prevoditelj ili opisne stranice (Slika 4.4).



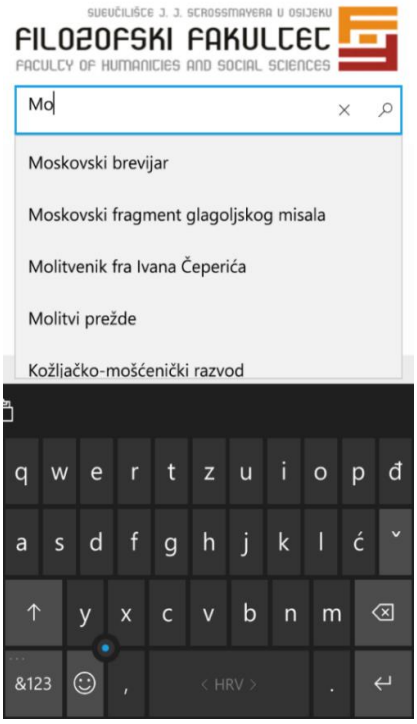
Slika 4.3. Glavna stranica aplikacije s izbornikom



Slika 4.4. Glavna stranica aplikacije

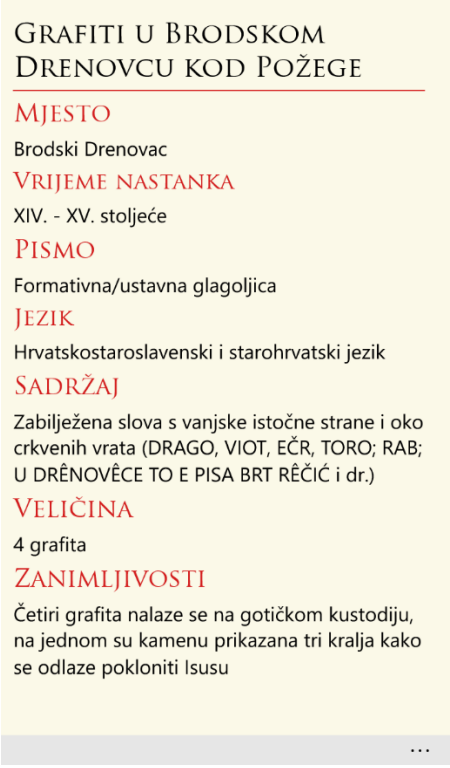
Nakon odabira stoljeća korisnik može odabrati željeni spomenik kroz listu (Slika 4.5) ili pretraživač sa početne stranice (Slika 4.6).

Slika 4.5. Popis stoljeća

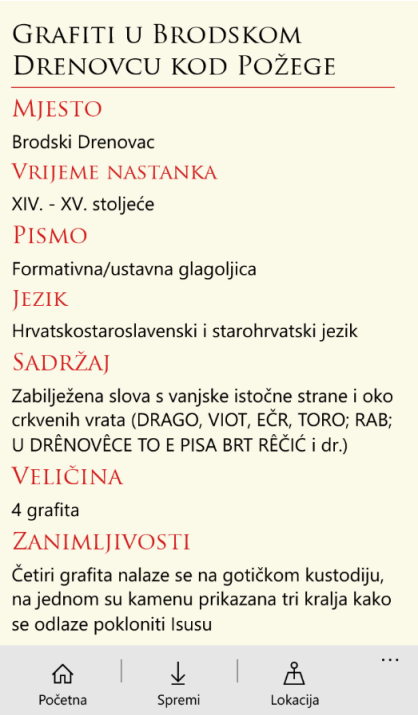


Slika 4.6. Pretraživač spomenika unutar aplikacije

Nakon odabira spomenika korisnik može odabrati vidjeti traženi spomenik (Slika 4.7), otvoriti mapu ili spremiti rezultat (Slika 4.8).



Slika 4.7. Stranica spomenik info



Slika 4.8. Stranica spomenik info s izbornikom

Nakon odabira spomenika korisnik može odabrati lokaciju (Slika 4.9).

Slika 4.9. Prikaz lokacije spomenika na mapi u demo razvoju

Prevoditelju se može pristupiti na stranici stoljeća ili na početnoj stranici (Slika 4.10).

Slika 4.10. Prevoditelj „Glagopedija“

Dizajn i korisničko iskustvo aplikacije važni su koraci pri izradi aplikacije, a završna inačica aplikacije je u potpunosti zadovoljila početne zahtjeve projekta i opće zahtjeve korisnika.

5. ZAKLJUČAK

Svrha ove aplikacije je omogućiti i olakšati studentima Filozofskog fakulteta u Osijeku smjera Hrvatski i Povijest pristup svim spomenicima kulturne baštine ali i zaljubljenicima u kulturu Hrvatske omogućiti virtualnu kulturnu enciklopediju. Razvojem operacijskog sustava Windows 10 nastala je i Windows 10 univerzalna aplikacija „Glagopedija“. Projekt je suradnja dviju znanstveno-nastavnih sastavnica osječkog Sveučilišta – Filozofskog i Fakulteta elektrotehnike, računarstva i informacijskih tehnologija. Minimalna inačice za instalaciju aplikacije je Windows 10 Build 10240. Aplikacija je razvijena u razvojnom okruženju *Microsoft Visual Studio 2015* s instaliranim Windows 10 mobile SDK paketom. Aplikacija je razvijana u programskom jeziku C# i XAML kao univerzalna Windows aplikacija. Aplikacija se može instalirati na mobilne uređaje s Windows Phone 10 mobilnim sustavom, ali isto tako se može namijeniti i za neku drugu obitelj uređaja. Za instalaciju je potrebno prenijeti datoteku na uređaj ili preuzeti aplikaciju iz Windows Trgovine .

Mogući su razni dodaci i nadogradnje na aplikaciju kao što je prijevod na engleski jezik te omogućavanje korisniku da sam unese spomenik kroz korisničko sučelje koristeći bazu podataka. U nadolazećim inačicama aplikacije omogućiti će se dohvaćanje podataka kroz online bazu podataka i potpuni prikaz lokacije spomenika na mapi. Sama aplikacija ali i dokumentacija kroz ovaj diplomski rad može poslužiti kao odličan nastavni izvor i vrijedan materijal za pripremu sveučilišnih ispita, projekata, završnih i diplomskih radova te primjer uspješne suradnje dvaju fakulteta.

Zahvale prof.dr.sc. Goranu Martinoviću na savjetovanju, mentorstvu i definiranju pojedinih rješenja pri izradi diplomskog rada.

LITERATURA

[1] Povijest.net : Digitalizacija u službi očuvanja i promoviranja kulturne baštine, http://povijest.net/digitalizacija-u-sluzbi-ocuvanja-i-promoviranja-kulturne-bastine/ dostupno 10.06.2016.

[2] M. Šojat-Bikić (2011), Hrvatska tradicijska baština online: stanje i mogućnosti, Znanstveni rad, Zagreb.

[3] M. Lukić, K. Blažević Krezić (2016), Projektna dokumentacija Glagopedija.

[4] Google play : Kulturna baština, https://play.google.com/store/apps/details?id=hr.dharmedia.kbastina, dostupno 10.06.2016.

[5] Heritagegopro: Upotreba GIS aplikacije u oblikovanju kulturne i/ili turističke rute, http://heritagegopro.com/upotreba-gis-aplikacije-u-oblikovanju-kulturne-iili-turisticke-rute-primjer/ , dostupno 10.06.2016

[6] MSDN: Vodič kroz univerzalne aplikacije, https://msdn.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide, dostupno 12.06.2016.

[7] Json.org, http://json.org/, dostupno 19.06.2016.

[8] Visual Studio : http://en.wikipedia.org/wiki/Microsoft_Visual_Studio, dostupno 22.06.2016.

[9] Msacademic : Čuda asinkronog programiranja 2.dio, http://msacademic.hr/cuda-asinkronog-programiranja-2-dio/ , dostupno 22.06.2016.

[10] MSDN: https://msdn.microsoft.com/en-us/library/ms973893.aspx, dostupno 22.06.2016.

[11] Newtonsoft : http://www.newtonsoft.com/json/help/html/SerializingJSON.html, dostupno 22.06.2016.

[12] Wikipedia : https://en.wikipedia.org/wiki/Approximate_string_matching, dostupno 3.7.2016.

[13] C-SharpCorner : http://www.c-sharpcorner.com/UploadFile/iersoy/autosuggestbox-control-in-universal-apps/, dostupno 3.7.2016.

[14] MSDN: All Blend : https://msdn.microsoft.com/en-us/expression/cc197141.aspx, dostupno 3.7.2016.

SAŽETAK

Cilj rada bio je izraditi aplikaciju koja korisnicima olakšava pristup informacijama o kulturnoj baštini Republike Hrvatske. Namjena i konačni cilj aplikacije je pomoć korisnicima u obrazovanju, aplikaciju je prikazana kao alat za učenje o spomenicima kulturne baštine na odsjeku Hrvatskog jezika i književnosti Filozofskog Fakulteta u Osijeku. U izradi Windows Phone 10 aplikacije korišten je C# programski jezik te opisni jezik XAML. Aplikacija je korisniku predstavljena kao virtualna glagoljska enciklopedija pod nazivom „Glagopedija“. Dizajn aplikacije izabran je na uzorku od više idejnih rješenja te je kao takav osmišljen i izrađen u CorelDraw X5 i Microsoft Blend okruženju. Ovakva aplikacija može biti vrlo korisna u promicanju kulturne baštine Republike Hrvatske, informiranju korisnika i studenata. Prednosti aplikacije su njena brzina, jednostavnost korištenja, dostupnost informacija te nova platforma za razvoj.

Ključne riječi: Glagopedija, kulturna baština, univerzalna platforma, Windows Phone aplikacija.

ABSTRACT

The goal was to create an app that helps users to access information about the cultural heritage of the Republic of Croatia. The purpose and ultimate goal of the application is to help users in education. Application is presented as a tool for learning about the cultural heritage monuments in the Department of Croatian Language and Literature Faculty of Philosophy in Osijek. Windows Phone 10 application is developed using C # programming language and XAML description language. The application is presented to the user as a virtual Glagolitic encyclopedia titled "Glagopedija". The application design was chosen on a sample of more conceptual ideas. Examples and main design are designed and manufactured in CorelDraw X5 and Microsoft Blend environment. This application can be very helpful in promoting the Croatian cultural heritage and student population. Benefits of application are its speed, ease of use and accessibility of information. New universal development platform is also one of benefit.

Keywords: Glagopedija, cultural heritage, universal platform, Windows Phone app.

ŽIVOTOPIS

Luka Mandić rođen je 17. ožujka 1993. godine u Požegi. Osnovnu školu „Vladimir Nazor“ pohađao u Trenkovu, a nakon osnovne upisuje Tehničku školu u Požegi smjer Tehničar za računalstvo. Nakon završetka srednje škole nastavlja obrazovanje u istom smjeru te 2011. upisuje Elektrotehnički fakultet u Osijeku, smjer Računarstvo. Uz studiranje, radi u obiteljskom poduzeću Pismoreklam gdje stječe iskustvo i ambicije za daljnje obrazovanje i napredovanje. Kroz svoje obrazovanje djelovao je kroz razne udruge, od najznačajnijih su IEEE gdje je bio predsjednik ogranka za računarstvo te kao Microsoft Student Partner gdje je aktivno sudjelovao u programu učeći nove tehnologije i promičući ih kroz razna predavanja i konferencije u Hrvatskoj. Kroz studiranje stekao je različita znanja i vještine. Ističu se prezentacijske vještine znanja o novim tehnologijama i njihove primjene te razvoj Windows i Windows Phone aplikacija. Zanima ga napredak tehnologije i njeznina primjena u praksi ali i poduzetništvo s kojim sve naučeno poprima vrijednost. Luka se koristi engleskim u govoru i pismu, a njemačkim jezikom u pismu.

Luka Mandić

(Potpis studenta)

PRILOZI

Prilog 1. Diplomski rad u docx, doc i pdf obliku.

Prilog 2. Slike korištene u pisanom dijelu završnog rada.

Prilog 3. Programski kod.