Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matjaž Ravbar

Sodoben razvoj prototipov uporabniških vmesnikov z orodjem Microsoft Expression Blend 4

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: doc. dr. Matjaž Kukar

Ljubljana 2012

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00294/2012
Datum: 13.04.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MATJAŽ RAVBAR**

Naslov: **SODOBEN RAZVOJ PROTOTIPOV UPORABNIŠKIH VMESNIKOV Z ORODJEM MICROSOFT EXPRESSION BLEND 4**

A MODERN APPROACH FOR PROTOTYPING USER INTERFACES WITH MICROSOFT EXPRESSION BLEND 4

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Kandidat naj opiše sodoben pristop k razvoju prototipov uporabniških vmesnikov z orodjem Microsoft Expression Blend 4. Opiše naj uporabljene tehnologije, orodja in načine, ki omogočajo interakcijo in dialog med razvijalcem uporabniškega vmesnika in njegovimi bodočimi uporabniki. Opisano naj ilustrira s primerom razvoja uporabniškega vmesnika praktične aplikacije.

Mentor:
doc. dr. Matjaž Kukar

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Matjaž Ravbar, z vpisno številko **63080034**, sem avtor diplomskega dela z naslovom:

Sodoben razvoj prototipov uporabniških vmesnikov z orodjem Microsoft Expression Blend 4

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Matjaža Kukarja,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 28. september 2012 Podpis avtorja:

Zahvalil bi se vsem, ki so kakorkoli pripomogli k nastanku diplomskega dela še posebej mentorju doc. dr. Matjažu Kukarju in oddelku za razvoj upravljanja s podatki in integracijo IS v UCIT NLB d.d.. Tukaj gre še posebna zahvala dipl. inž. rač. in inf. Petru Konda, ki je zasnoval tako zanimiv projekt in me je tekom raziskovalnega projekta usmerjal na pravo pot ter ob težavah priskočil na pomoč z svojim znanjem. Zahvaljujem se tudi vodji oddelka ga. Maji Harvatin Starkelj za oddobritev delovne prakse, v okviru katere je nastal projekt, ki ga bom predstavil v diplomskem delu. Poleg tega bi se zahvalil tudi svojim sošolcem in prijateljem, brez katerih bi bil študij nedvomno težji in staršem, ki so me podpirali na moji študijski poti.

Kazalo

Povzetek

V diplomskem delu je opisan sodoben pristop k razvoj prototipov uporabniških vmesnikov z orodjem Expression Blend 4. Razvoj prototipov je ilustriran na primeru razvoja prototipov aplikacije Register komitentov, ki je potekal, v okviru projekta prenove Registra komitentov, v podjetju NLB d.d.. Ker Expression Blend ponuja razvoj za številne tehnologije, so v diplomsko delo so zajete in opisane tudi vse uporabljene tehnologije in orodja, ki so neposredno povezane z razvojem SketchFlow prototipov. Hkrati je na podlagi razvoja predstavljen tudi dialog med razvijalcem uporabniškega vmesnika in njegovimi bodočimi uporabniki. Ugotovili smo, da je Microsoft Expression Blend 4 učinkovito orodje za razvoj prototipov uporabniških vmesnikov za platformo Silverlight, ki omogoča hiter in enostaven razvoj ter s svojimi funkcijami izboljšuje sodelovanje med razvijalcem in bodočim uporabnikom.

Ključne besede: Microsoft Expression Blend 4, Silverlight, SketchFlow prototipi, razvoj prototipov, sodoben pristop k razvoju prototipov

Abstract

The BA thesis describes the modern approach to prototyping user interfaces with Microsoft Expression Blend 4. Development of SketchFlow prototypes was illustrated at the case of the development prototypes for application Register of business partners, which was held in framework of renovations application Register of business partners in the company NLB. In the BA thesis are included and described also all the technology and tools that are directly related to the development SketchFlow prototypes because the Expression Blend provides the development of many technologies. At the same time, on the basis of the development presented the dialogue between developers the user interface and its future users. We have discovered that Microsoft Expression Blend 4 is an effective tool for prototyping user interfaces to Silverlight platform, which enables fast and easy development and improves collaboration between developers and users.

Key words: Microsoft Expression Blend 4, Silverlight, SketchFlow prototypes, prototyping, a modern approach for prototyping

Poglavje 1

Uvod

Kakovosten in hiter razvoj programske opreme je v današnjem času ključen kazalnik uspeha, hkrati pa tudi zapleten pojem. Stvari še posebej postanejo zahtevne, ko se lotimo večjega projekta, kjer za hiter in kakovosten razvoj programske opreme potrebujemo večje razvojne ekipe s katerimi sodelujejo uporabniki na različnih nivojih dela. Na srečo imamo v današnjem času na razpolago številne programske rešitve, ki pospešijo proces razvoja informacijskih rešitev. Eno takih orodij je tudi Expression Blend 4, ki je namenjeno hitremu in enostavnemu razvoju mobilnih, spletnih ali namiznih aplikacij in prototipov v tehnologiji WPF in Silverlight. V ta namen bom v svojem diplomskem delu predstavil razvoj SketchFlow prototipov aplikacije Register komitentov, ki služijo kot podlaga za nadaljnji razvoj aplikacij. SketchFlow prototipi temeljijo na Silverlight tehnologiji, njihovo izdelavo pa omogoča program Expression Blend 4. Na primeru aplikacije Register komitentov bom predstavil praktični primeru razvoja SketchFlow prototipov in aplikacijo Register komitentov ter tehnologije, ki so bile neposredno povezane z razvojem prototipov. Diplomska naloga je sestavljena iz dveh delov. V prvem delu bom najprej govoril o programu Expression Blend 4 in predstavil njegove novosti v primerjavi s preteklimi različicami. Poleg omenjenega je zaradi lažjega razumevanja celotne tematike v prvem delu zajeta tudi predstavitev uporabljenih tehnologij, s katerimi želim bralcu pomagati razumeti tehnološke zmožnosti

in omejitve, ki jih ponuja Silverlight tehnologija in orodje Expression Blend 4. V drugem delu bom predstavil aplikacijo Register komitentov, ki se uporablja za informacijsko podporo poslovanju s komitenti v Novi Ljubljanski banki. Slednja bo zaradi zastarele tehnologije in vse dražjega vzdrževanja dobila svojo novo različico. Ker bo nova aplikacija temeljila na Silverlight tehnologiji, je potrebno izdelati nove predloge uporabniških vmesnikov. Ker razvoj informacijskih sistemov v NLB d.d. poteka po interno sprejetih standardih, so ti vključevali tudi izdelavo prototipov uporabniških vmesnikov. Sprva je razvoj prototipov potekal v programu Microsoft Office Visio. Ker je razvoj potekal na novi razvojni platformi, smo se odločili, da prototipe poskušamo implementirati v Silverlight tehnologiji, ki je skladna s platformo produkcijske aplikacije. V nadaljevanju sledi praktični primer razvoja prototipov na projektu prenove Registra komitentov, ki sem ga izvajal med obvezno študijsko prakso. V tem poglavju sem opisal vse ključne elemente programa Expression Blend in postopno predstavil potek razvoja prototipa, izdelan prototip pa poskusil pretvoriti v produkcijski projekt. Omenili bomo tudi uporabo prototipov v procesu razvoja informacijskega sistema, kakšne so prednosti in slabosti uporabe ter predstavili potek dela znotraj razvojne faze specifikacije zahtev. Vse ugotovitve bom zapisal v sklepu diplomske naloge.

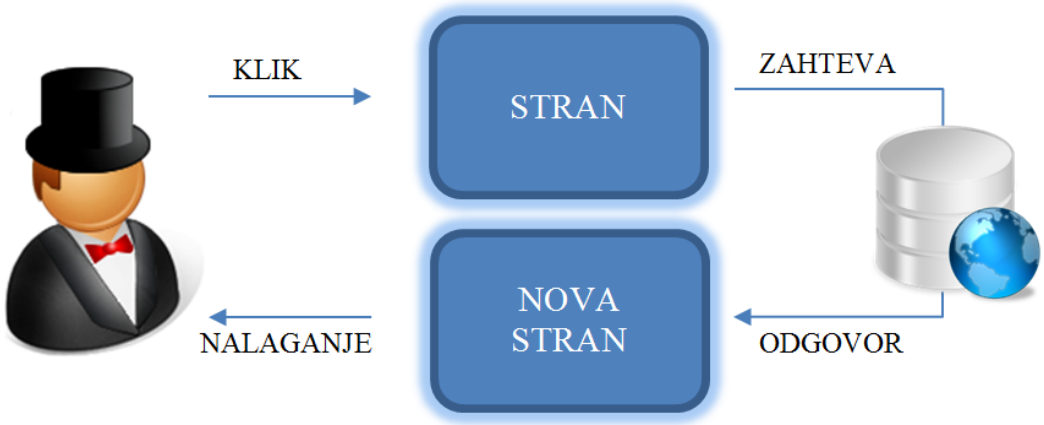
Poglavje 2

Uporabljene tehnologije

Zaradi lažjega razumevanja tehnoloških zmožnosti, ki nam jih ponuja tehnologija Silverlight in Expression Blend 4, bom najprej prestavil tehnologije, ki sestavljajo SketchFlow prototipe. SketchFlow prototipi za spletne aplikacije temeljijo na tehnologiji Silverlight.

2.1 Tehnologija Silverlight

Splet se razvija v smeri, v kateri je uporabniška izkušnja glavna točka uspeha večine spletnih strani ali spretnih aplikacij, uporabniki pa od spleta pričakujejo več kot kadarkoli prej. S koncem obdobja statičnih HTML strani ter razširjanjem in razvojem tehnologij, kot je Adobe Flash platforma in AJAX, se je povečeval poudarek na odzivu, videzu in funkcijah aplikacije, kar je neposredno povečalo uporabniško izkušnjo. Pojav bogatih internetnih aplikacij (RIA) napoveduje obdobje, ko bodo spletne aplikacije in spletne strani postale bolj priljubljene od namiznih aplikacij, informacije se bodo nahajale le klik stran od uporabnika, ob tem pa uporabnik ne bo zaznal sprememb, ki zadevajo stanje in videz vmesnika. Cilj RIA je implementirati atraktiven uporabniški vmesnik, podatke, funkcionalnost in uporabnost v en izdelek, ki preseka tradicionalni page-clik-page model, ki ga danes ponuja vsaka bolj sofisticirana podatkovno usmerjena spletna stran.



Slika 2.1: Model odjemalec- strežnik.

Za oblikovalce in razvijalce to pomeni, da ne potrebujejo samo orodij za razvoj prototipov v končno aplikacijo, ampak tudi nadgradnjo in prenovo produkta ter informacij, ki so ga v preteklosti že razvijali [8]. Uvedba Silverlight platforme in Microsoft Expression Studio[1] aplikacij oblikovalcem, informacijskih arhitektom in razvijalcem uporabniških vmesnikov ponuja platformo ter omogoča, da s sodelovanjem združijo vse svoje obstoječe znanje in sposobnosti pri gradnji in razvoj izdelka v končni produkt oziroma aplikacijo. Andrej Tuzon, eden izmed zunanjih razvijalcev v NLB, je v Microsoftovemu sporočilu za medije ob prihodu tehnologije Silverlight 3 zapisal:

> "Silverlight razvijalcem namizne programske opreme nudi orodja za preprost prehod na razvoj bogatih spletnih rešitev. V primerjavi s tradicionalnimi spletnimi aplikacijami je mogoče v Silverlightu oblikovati bolj kakovostno uporabniško izkušnjo, ki na eni strani razbremeni strežnik, na drugi pa zagotavlja bogat uporabniški vmesnik na odjemalcu" [9].

S tem je želel povedati, da se strežnik razbremeni predvsem zaradi večje

[1]Microsoft Expression Studio je zbirka Microsoftovih orodij za oblikovanje in razvoj bogatih namiznih in spletnih aplikacij.

količine programske logike, ki jo lahko implementiramo na strani Silverlight odjemalca in s tem zmanjšamo delo in interakcijo s strežnikom. Enostaven primer je preverjanje pravilnosti vnosa v tekstovno polje.

Silverlight predvajalnik

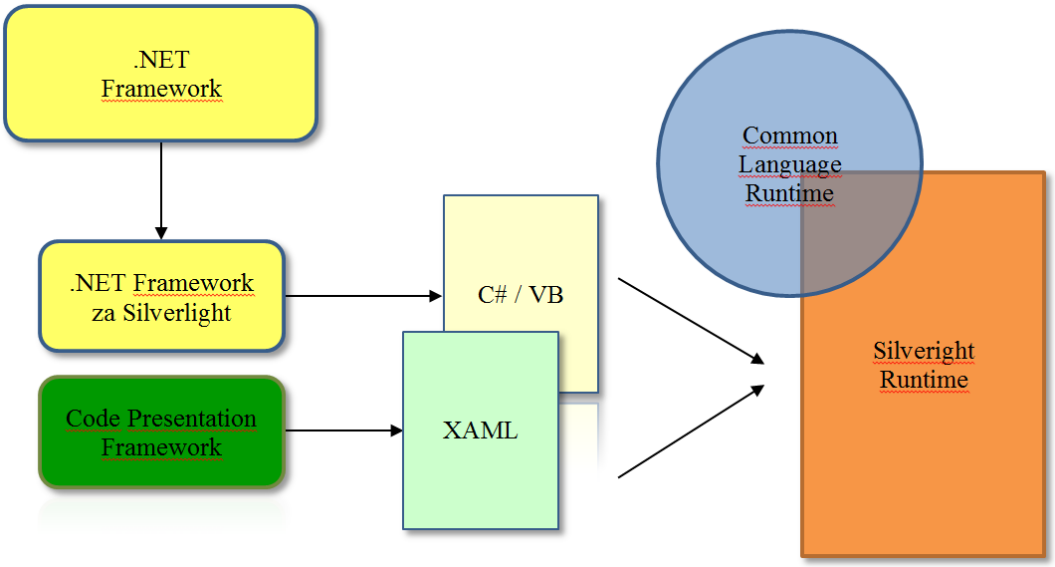
Silverlight predvajalnik (imenovan tudi Silverlight runtime) je vtičnik, ki deluje neodvisno od platforme in brskalnika. Uporabniku omogoča, da vidijo bogate spletne aplikacije (RIA), ustvarjene z Expression Blend, Visual Studio ali s katerokoli drugo aplikacijo, ki podpira Silverlight platformo. Silverlight aplikacije so zgrajene na optimizirani, okrnjeni različici Microsoftovega .NET Framework[2] razvojnega okvirja, ki programerjem ponuja programske rešitve in knjižnice, ki zagotavljajo rokovanje s skupnimi nalogami, kot so predvajanje večpredstavnostnih vsebin, izrisovanje uporabniškega vmesnika in kontrol, rokovanje s podatki in povezovanje s podatkovno bazo. Podobno kot druge tehnologije (Adobe Flash Player) lahko tudi Silverlight ustvari sliko uporabniškega vmesnika, prenaša video ter nalaga in upravlja s podatki v realnem času, poleg tega pa ponuja tudi visoko stopnjo interaktivnosti za podporo igram in grafično zahtevnim spletnim aplikacijam. Silverlight arhitektura je sestavljena iz 2 primarnih komponent:

- Silverlight .NET Framework in
- Core Prezentation Framework.

Silverlight .NET Framework je okrnjena različica Microsoftovega razvojnega okvirja Microsoft .NET Framework in zagotavlja standardne komponente ter osnovne knjižnice razredov, integracijo podatkov, mreženje, smetarjenje (garbage colector - za upravljanje s pomnilnikom) in skupno izvajalno okolje CLR[3].

[2].NET framework - je del sistema Windows, ki se uporablja za oblikovanje, uvajanje in zagon programov.

[3]CLR – Common Language Runtime je del .NET Framework, ki skrbi za izvajanje programske kode.



Slika 2.2: Silverlight arhitektura.

Če definiramo, kako se izriše uporabniški vmesnik v tehnologiji Silverlight, lahko rečemo, da Core Presentation Framework vključuje vse funkcije, ki so neposredno vezane na uporabniški vmesnik, kot so na primer vektorska grafika, tekst, rokovanje z večpredstavnostnimi vsebinami in seveda razširitveni aplikacijski označevalni jezik XAML [8].

2.1.1 Primerjava med Silverlight 3 in Silverlight 4

Da bi lahko primerjali med Silverlight 3 in 4, je zelo priporočljivo poznati vse verzije Silverlight, zato bom na kratko najprej opisal prvi dve verziji. Prva izdaja Silverlighta se je imenovala Silverlight 1.0. Razvoj programske opreme je potekal po programskem modelu JavaScript, preko katerega se je izvajala interakcija s Silverlight objekti. To je pomenilo, da takratne Silverlight aplikacije niso imele BCL[4] podpore. JavaScript se je uporabljal tudi za izvajanje aplikacij v Silverlight predvajalniku, ki je bil nameščen v brskalniku preko

[4]BCL – standardna knjižnica za vse programske jezike, ki uporabljajo .NET ogrodje.

vtičnika. V letu 2008 so pri podjetju Microsoft izdali novo različico, ki se je imenovala Silverlight 2.0 [10]. Z izidom Silverlight 2.0 smo imeli na razpolago izbirati med različnimi CLR programskimi jeziki, ki jih je Microsoft ponujal na razpolago že pri razvoju okenskih aplikacij, ASP.NET[5] in WPF[6] aplikacij. Poleg tega je za interakcijo s Silverlight objekti uporabljal .NET okvir. Ne glede na vse pa še vedno nismo imeli možnosti hitrega razvoja aplikacij, saj je bilo potrebno vse vire in kontrole razviti ročno brez pomoči aplikacije oziroma generatorja kode. Po devetih mesecih je Microsoft ponudil tretjo različico platforme Silverlight, ki je uvajala več sto pomembnih novih možnosti in kontrolnikov, s katerimi je razvojnim ekipam omogočila, da lahko ustvarjajo bogate internetne aplikacije in večpredstavne rešitve. Prihod verzije Silverlight 3 predstavlja nadgradnjo predhodnih različic, ki se z vsako novo verzijo dopolnjujejo. Verzija 3 predstavlja velik korak naprej za platformo Silverlight, ki se ponaša s številnimi novostmi na področju delovnega procesa, grafike, videa in drugih lastnosti aplikacij. Nekaj funkcij, ki so vključene v Silverlight 3, bom naštel tukaj [8]:

- Expression SketchFlow je orodje, preko katerega lahko izdelamo prototipe za spletne in namizne aplikacije za platformo Silverlight in WPF.

- Delovanje izven brskalnika nudi možnosti za delovanje aplikacije izven brskalnika na različnih platformah in brez povezave, kar v primerjavi z običajnimi bogatimi internetnimi aplikacijami nudi več storilnosti in dodatne možnosti.

- Pretočne vsebine brez zatikanj s strežnika IIS so del strežnika Internet Information Services 7.0 (IIS7), v povezavi s tehnologijo Silverlight 3 pa možnost IIS Smooth Streaming zagotavlja video izkušnje v visoki ločljivosti 1080p za video v živo ali na zahtevo.

[5]ASP.NET – kratica za Microsoftovo spletno aplikacijsko ogrodje, ki programerjem omogoča razvoj dinamičnih spletnih strani.

[6]WPF – angl.Windows Presentation Fundation, je računalniški programski grafični podsistem, ki skrbi za izrisovanje uporabniških vmesnikov namiznih aplikacij.

- Podpora aplikacijam s 3D perspektivo nam omogoča, da 2D aplikacije pretvorimo v 3D perspektivo, ki nam omogoča vpeljavo nesimetrične ravnine in rotiranja vmesnika.
- Novi vizualni efekti, ki so bili do tedaj na voljo samo v WPF aplikacijah, so od verzije Silverlight 3.0 naprej na razpolago tudi v XAML.
- Novi poenostavljeni efekti za animacije pripomorejo k bolj tekočemu uporabniškemu vmesniku.
- Neposredno povezovanje med lastnostmi elementov poenostavlja povezovanje med elementi in odpira nove možnosti za integracijo podatkov v aplikacijo.

Silverlight prinaša več sto novih možnosti in kontrolnikov, ki povečujejo storilnost razvijalcev in vključujejo tudi podporo za strojno pospeševanje grafičnih in tridimenzionalnih elementov.

S prihodom četrte različice Silverlight Novembra 2009 lahko aplikacije še dodatno grafično obogatimo s funkcionalnostmi, od katerih bom povzel le najbolj pomembne:

- Podpora za tiskanje nam omogoča izdelavo pogleda tiskanja, ki nam virtualno prikaže, kako bo izgledal natisnjen dokument.
- Izboljšana lokalizacija z dvosmernim besedilom ter podporo za pisanje od desne proti levi in kompleksne pisave.
- Managed Extensibility Framework omogoča razvoj kompleksnih modularnih aplikacij, ki omogočajo hiter zagon in prenos iz interneta, učinkovit razvoj in testiranje kot tudi hitro vzdrževanje in prilagajanje poslovnim potrebam.
- Storitve Windows Communication Foundation RIA
- Izboljšane animacijske sposobnosti, ki poskrbijo za bolj dinamične in interaktivne predstavitve podatkov v seznamih.

- Podpora za spletno kamero in mikrofon, ki nam omogoča zajemanje videa in zvoka v aplikacijah.

- Zmogljivost lokalnega snemanja videa in zvoka nam omogoča pošiljanje videa in zvoka preko elektronske pošte ali urejanja posnetega zvoka in slike brez predhodnega shranjevanja ter zajemanje videa v RAW formatu brez interakcije s strežnikom.

- Modeli interakcije omogočajo razvijalcem vpeljavo naprednih funkcij, ki se navezujejo na uporabo desnega gumba miške in drsnika.

- Podpora multitouch omogoča različne gibe in interakcijo z zaslonom na dotik, ki povečuje uporabniško izkušnjo.

- Multicast povezovanje omogoča podjetjem združljivost z obstoječo infrastrukturo storitev Windows Media Services, ki omogočajo zniževati stroške posredovanja internetnih storitev.

- Varovanje vsebine je sedaj skozi Silverlight DRM[7] sedaj na voljo tudi za H.264 format video datotek. Zaščita za izhodni avdio/video podatkovni tok lastnikom zagotavlja, da lahko zaščiteno vsebino pregledujejo le preko varne video povezave.

- Programi z varnostnimi omejitvami razvijalcem omogočajo vpeljavo HTML v aplikacije, kar omogoča tesnejšo integracijo z vsebino is spletnih strežnikov.

- Obvestilna okna omogočajo vpeljavo pojavnih okenskih obvestil, ki zagotavljajo takojšen odziv in obveščanje uporabnika.

- Zaupanja vredne aplikacije omogočajo uporabniku dostop do lokalnih virov. Uporabniki lahko preko Silverlight aplikacij odpirajo in shranjujejo svoje datoteke, ki so shranjene na računalniku in omogočajo upo-

[7] DRM – Digital Rights Management.

rabniku izdelavo lokalnih kopij poročil in medijskih datotek. COM[8] avtomatsko omogoča dostop do naprav in drugih virov sistema. Za kreiranje poročil se lahko uporabniki poslužujejo drugih namiznih programov, kot je Microsoft Office Excel in podobno.

- Izboljšane lastnosti aplikacij zaradi optimizacij zagotavljajo hitrejše zaganjanje in 200 procentov hitrejše delovanje Silverlight 4 aplikacij od ekvivalentnih Silverlight 3 aplikacij.

- Veliko novih kontrol. Celoten sklop kontrol z več kot 60 prilagojenimi kontrolami pripomorejo k lažjemu in hitrejšemu razvoju obrazcev.

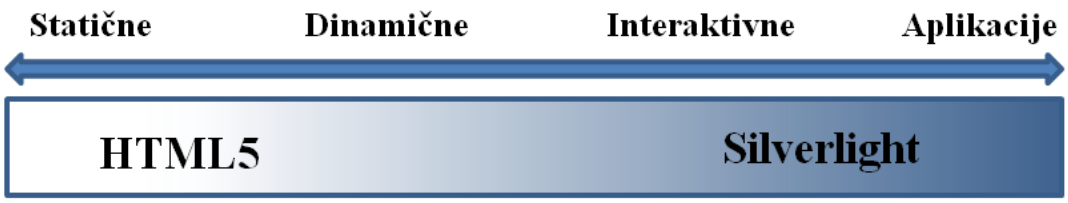
2.1.2 Silverlight in HTML5

Ker je bilo v zadnjem času zelo veliko govora o prihodnosti in uporabi HTML5 in Silverlight, bom na kratko povzel obe tehnologiji. S pojavom HTML5 je Silverlight dobil konkurenta na področju razvoja bogatih internetnih aplikacij. Razvoj HTML5 aplikacij je uradno podprl tudi Microsoft s prihodom nove različice orodja Visual Studio 2012. Pri primerjavi med Silverlight in HTML5 rešitvami je potrebno upoštevati mnogo različnih dejavnikov, najpomembnejši je, kakšno aplikacijo razvijamo in kakšna je naša razvojna ekipa, saj se je potrebno zavedati vse bolj pomembnih ekonomskih dejavnikov, ki so tudi povezani s stroški razvoja in vzdrževanja spletne strani ali aplikacije. Silverlight v primerjavi s HTML5, ki ponuja izboljšan dostop in boljšo kompatibilnost s CSS stili, ne ponuja statičnih spletnih strani, zato na tem področju prevladuje HTML5. Pri razvoju dinamičnih spletnih strani ponuja HTML5 razne kontrolnike za povezovanje z različnimi podatkovnimi viri. Z AJAX podporo se osvežujejo samo določeni deli strani, povezave med dokumenti pa še vedno potekajo po hiperpovezavi. Obstajajo razna orodja, ki omogočajo asinhrone povezave, urejene tabele in podobno. Prav tako kot

[8]COM – Component Object Model predstavlja binarne vmesnike in komunikacijo med različnimi komponentami.

HTML5 je tudi Silverlight zelo dobro opremljen z dinamični in interaktivnimi "out of box" kontrolami, ki zagotavljajo dinamično posredovanje vsebine. Ne glede na to, da obe tehnologiji ponujata dinamične spletne strani bi lahko sklenil zaključek, da je v primeru razvoja statičnih in dinamičnih spletnih strani najverjetnejši zmagovalec HTML5 zaradi vrhunske statične razporeditve dokumentov in lahkega pristopa. Ko se usmerimo v primerjavo po interakciji med HTML5 in Silverlight, počasi zgubljamo povezavo s klasičnimi spletnimi stranmi in začnemo govoriti o spletnih aplikacijah. Take strani lahko vključujejo znatno večjo poslovno logiko, validacijo, grafe in diagrame. Ti višji konstrukti za razvoj uporabniškega vmesnika v HTML5 niso prisotni, zato jih je potrebno zgraditi samostojno. V Silverlight imamo na voljo višje konstrukte uporabniških vmesnikov, saj ima Silverlight vnaprej pripravljene zbirke kontrol, standardnih vzorcev in najboljših praks, ki so nam na voljo za razvoj aplikacij. V smislu, kaj lahko dosežemo z enako tehnologijo, se zdi, da sta obe tehnologiji enakovredni, vendar pa bi razvoj interaktivne spletne strani s HTML5 v praksi trajala bistveno dlje kot razvoj enakovredne rešitve v Silverlight tehnologiji. To je predvsem posledica razlike med "plugin" pristopom, ki daje predvidljivo in dosledno izvajalno okolje, medtem ko HTML5 aplikacije zahtevajo več pozornosti pri podpori brskalniških funkcij in zmogljivosti. Kakorkoli to še ne potisne HTML5 iz uporabnega okvirja, saj ima tudi HTM5 tu nekaj prednosti pri aplikacijah, ki nimajo ciljne skupine uporabnikov (splošen uporabnik) [20]. Na koncu nam ostane samo še primerjava HTML5 in Silverlight za aplikativno podporo, kjer HTML5 zgubi svojo vrednost. Kombinacija programskega jezika in razvojnega okolja, kontrol, uporabniške komponente, modularni okvir skupaj s predvidljivim izvajalnim okoljem poskrbita za sorazmerno enostaven razvoj aplikacij s Silverlight tehnologijo v primerjavi s HTML5. Čeprav je HTML5 v tem primeru zelo neprimeren za uporabo, je potrebno povedati, da to ne pomeni, da HTML5 ne ponuja razvoja spletnih aplikacij.

Na koncu lahko sklenem, da lahko identične spletne strani in aplikacije razvijemo z obema tehnologijama. HTML5 se bo vedno bolj uporabljal za



Slika 2.3: Primerjava med HTML5 in Silverlight tehnologijo.

podporo statičnim in dinamičnim spletnim stranem ter se uveljavljal kot vsestranska mobilna platforma. Silverlight se bo še naprej uveljavljal pri razvoju dinamičnih spletnih strani in uporabljal predvsem za razvoj kompleksnih spletnih aplikacij še posebej v zaprtih korporacijskih okoljih, kjer prevladuje.

2.2 SketchFlow

SketchFlow je orodje za izdelavo prototipov v programu Expression Blend 4. Omogoča nam hitro ustvarjanje in izdelavo prototipov in učinkovito predstavitev slednjih našim strankam od samega začetka pa vse tja do zaključne faze projekta. Tradicionalni prototipi so v ponavadi neuporabni in se jih po končani konceptualni zavrže. Povsem drugače je pri SketchFlowu, saj je zasnovan tako, da prototip, ki smo ga razvili v začetni fazi uporabimo v naslednjih fazah vse do zaključne faze projekta, s tem pa vseskozi razvijamo in nadgrajujemo naše konceptualne ideje oziroma zamisli, kontrole, obnašanja aplikacije in oblike, ki smo jih zasnovali v SketchFlow prototipu. Zaradi zmožnosti hitrega in enostavnega razvoja prototipov je zelo dober način za vzpostavitev komunikacije in sodelovanje z uporabnikom, saj lahko v zelo kratkem času razvijemo več različnih primerkov aplikacije z več različnimi funkcionalnimi postavitvami elementov zaslona, ki jih predstavimo uporabniku kot interaktivni prototip. SketchFlow poskuša uporabnika osredotočti na celotno rešitev. Ker ne želimo, da bi se uporabnik že v zgodnjih fazah osredotočil in ubadal s podrobnosti konceptualne faze in postavitvijo Sket-

chFlow kontrol, nam SketchFlow pomaga zadržati uporabnika ves čas osredotočenega na celotno rešitev. Ko je prototip pripravljen za demonstracijo, lahko od uporabnika na našo prošnjo takoj pridobimo povratne informacije o prototipu. Brez kakršne koli dodatne programske opreme lahko stranka (uporabnik) enostavno pregleduje in testira prototipe, svoje komentarje in pripombe pa lahko uporabnik oddaja preko SketchFlow predvajalnika, ti pa se razvijalcu direktno prikažejo v SketchFlow razvojnem orodju. To omogoča hiter odziv na zahteve in komentarje uporabnikov, ki so jih podali ob pregledovanju prototipa. Ko uporabnik konča s pregledovanjem prototipa, lahko zelo hitro premaknemo popravljeno verzijo v produkcijo, da ima uporabnik omogočen takojšen ponoven pregled nad prototipom. Vse te prednosti maksimizirajo vložek razvijalca v fazi konceptualnega razvoja prototipov. Expression SketchFlow je vključen v:

- Expression Blend (verzija 3 in 4) in
- Expression Professional Subscription, ki ponuja virtualen dostop do vseh Microsoftovih produktov za oblikovanje, razvoj in testiranje aplikacij.

Najpomembnejše funkcije:

- Učinkovito prototipiranje.
- Interaktivno pregledovanje – SketchFlow predvajalnik.
- Odlična dokumentacija – hitro ustvarjena podrobna dokumentacija za naš projekt.

Skiciranje in prototipiranje sta zelo uspešni tehniki, ki nam omogočata zelo uspešen in hiter razvoj številnih idej in prototipov v Expression Blend brez prekomerne porabe časa in virov.

SketchFlow je orodje za razvoj in prototipiranje uporabniških vmesnikov, s katerim lahko na podlagi hitrega in učinkovitega razvoja prototipov

v zelo kratkem času predstavimo svojo vizijo za aplikacijo, ki jo želimo razviti. Predstavlja neformalni in hiter način za raziskovanje, prenavljanje in razvoj scenarijev prototipa uporabniškega vmesnika, ki nam omogoča, da iz svojih zamisli razvijemo koncepte v žive prototipe, ki predstavljajo realno sliko končne aplikacije, ki ustreza projektnim oziroma uporabniškim zahtevam, ki so zapisane v projektni dokumentaciji. Ta hiter, ponovljiv in cenovno učinkovit pristop prototipiranja nam omogoča, da se osredotočimo na dve najbolj pomembnejši stvari, ki sta kreativnost in pravočasen razvoj najboljše rešitve za stranko v okviru našega proračuna. SketchFlow nam omogoča hiter in učinkovit zemljevid in eksperiment s tokom uporabniških vmesnikov aplikacije, mrežo posameznih zaslonov in pove kako aplikacija prehaja iz enega stanja aplikacije v drugega. SketchFlow ima hiter in učinkovit zemljevid, na katerem so razporejeni zasloni oziroma eksperimenti, ki prikazujejo tok uporabniških vmesnikov v aplikaciji in prehod iz enega stanja aplikacije v naslednjega. S kontrolami in komponentami, ki so zgrajeni v obliki skice, lahko ideje zelo dobro in hitro vizualiziramo ter nadgradimo. Hiter in tekoč način dela nam pomaga, da lahko zamisli hitro ustvarimo, preverimo, popravimo ali zavržemo in s tem raziščemo različne rešitve ter izberemo najoptimalnejšo pot, ki vodi k najboljši rešitvi z minimalnimi stroški. Zaradi hitrosti in nizkih stroškov vizualizacije oziroma demonstracije prototipov lahko že pred začetkom projekta ocenimo, ali bo projekt strokovno učinkovit in donosen. Uporabnik oziroma stranka že na samem začetku procesa načrtovanja pridobi uporabniške izkušnje, saj v SketchFlow prototipu zelo hitro prikažemo, kako bo aplikacija tekla in prehajala med različnim stanji. Hitra predstavitev nam omogoča, da uporabnik izrazi svoje uporabniške izkušnje s prototipom aplikacije, pomanjkljivosti v prototipu pa lahko označi takoj, ko se z njimi sreča, saj se s tem zmanjša strošek, čas in težavnost odpravljanja pomanjkljivosti v prototipu. Zbiranje učinkovitih in pravočasnih odzivov uporabnika je prav tako pomembno kot zmanjševanje stroškov in čas razvoja v procesu razvoja in načrtovanja prototipa. Ker je SketchFlow predvajalnik prosto dostopen, zagotavlja uporabnikom zelo učinkovito pred-

stavitev prototipov preko standardnih brskalnikov ne glede na lokacijo, kjer se uporabnik nahaja. Uporabniki preko brskalnika pregledajo in testirajo različne scenarije prototipa, obenem pa lahko po potrebi v SketchFlow predvajalniku razvojni skupini sporočijo opazke in izkušnje, ki so jih pridobili ob testiranju in pregledovanju zaslonov. Ko uporabnik shrani svoj komentar, ga lahko shrani in posreduje razvojni ekipi. Razvojna ekipa lahko komentarje uporabnikov uvozi direktno v Expression Blend 4. Komentarji so v programu prikazani v posebnem polju na vmesniku programa Expression Blend, prav tako pa se vidijo vse označbe na prototipu, ki jih je označil uporabnik, ko je pregledoval prototip. To je za razvijalce oziroma oblikovalce zelo priročno orodje, saj lahko v vsakem trenutno vidijo kaj je potrebno popraviti. Tak pristop bistveno zmanjša čas razvoja in razbremenjuje tako uporabnika kot tudi razvijalca, saj v tem primeru ne porabimo toliko časa za komunikacijo z uporabniki. Koncepti, zgrajeni v zgodnji fazi načrtovanja, morda po videzu izgledajo kot prototip, ki pa dejansko predstavljajo pravi Silverlight oziroma WPF projekt, ki ga lahko ustvarimo v programu Expression Blend in Visual Studio, s katerimi si lahko Expression Blend deli projekte. To pomeni, da že od prvega trenutka delamo na realnem projektu, ki ga lahko raziščemo in predstavimo uporabniku, razen če ta predstavlja le zaporedje skiciranih skic oziroma zamisli brez kakršnega koli resničnega uporabniškega vmesnika. SketchFlow omogoča ustvarjanje kompenzivne projektne dokumentacije preko funkcije za izvažanje v Microsoft Word, ki ustvari dokument idejnega projekta v celoti s kazalom vsebine, komponentnimi, navigacijskimi in drugimi okni aplikacije znotraj projekta. To nam prihrani mnogo časa in omogoča, da projekt hitro, temeljito in enostavno opišemo v sprejemljivih časovnih okvirjih kar med samim razvojem projekta. SketchFlow prototipi so hitri, enostavni in poceni za gradnjo, zaradi česar je mogoče ustvariti, raziskati in med seboj primerjati številne prototipe, še preden nadaljujemo z razvojem rešitve. Po končani konceptualni fazi razvoja postanejo prototipi običajno neuporabni in se jih zato zavrže. SketchFlow nam omogoča, da celotno delo na konceptu, virih in komponentah, ki smo ga ustvarili, ponovno

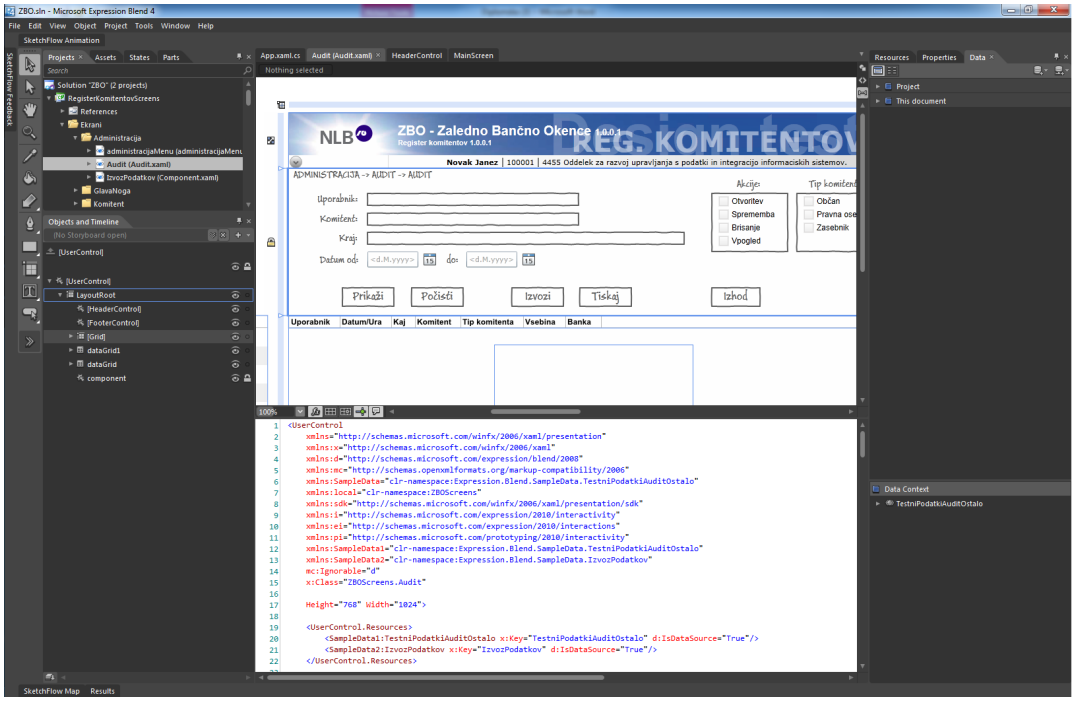
uporabimo za naš produkcijski projekt – nič se ne izgubi. Če izkoristimo vse funkcionalnosti, ki jih nam ponuja Expression Blend 4, lahko hitro razvijemo prototip brez kakršnih koli omejitev, ki se navezujejo na obseg prototipa.

Za preproste prototipe, ki so zelo natančni, interaktivni in podatkovno vodeni, nam SketchFlow ponuja fleksibilen razvoj in demonstracijo naših prototipov in zamisli za našo stranko na najbolj učinkovit način. S SketchFlow imamo fleksibilno in popolno kontrolo nad razvojem rešitve od prototipa do končne aplikacijem [7].

2.3 Označevalni jezik XAML

XAML je jezik, ki nam po eni strani zagotavlja enostaven pregled rešitev in nam poskuša pravilno razumeti out of the box rešitve, po drugi strani pa oblikovalcem, IT arhitektom in razvijalcem omogoča, da so del procesa razvoja, po katerem poteka celotno delo na enem mestu. Razširljiv aplikacijski označevalni jezik, bolj poznan kot XAML (pogovorno "zam-el"), je temelj Silverlight in WPF aplikacij. XAML predstavlja uporabniški vmesnik s strukturnega, hierarhičnega vidika in v nekaterih primerih tudi z vidika obnašanja aplikacije. XAML bazira na XML označevalnem jeziku, ki je zasnovan kot zaporedje oklopljenih značk in ključev (keywords). Opazimo lahko, da sta XML in XHTML kodi zelo podobni XAML-u, tako po videzu kot tudi pristopu. Vsak XAML element predstavlja nekakšen objekt v Silverlight aplikaciji, kot je na primer: črta, okvir, uporabnikova kontrola ali panel z animacijami. XAML podaja ukaze kako in kje naj Silverlight izriše točno določen element na zaslon, v nekaterih primerih pa XAML pove tudi, kako se bo ta objekt obnašal kot odzivi na dogodke. Expression Blend upodablja XAML na otipljiv način, saj ponuja pogled oblikovanja (Design View), tako da imamo vizualni pregled nad aplikacijo (glej sliko 2.4).

Za oblikovalce XAML predstavlja prijazen prikaza uporabniškega vmesnika v Silverlight aplikacijah, ki zagotavlja enostavno kreiranje in prilagajanje uporabniškega vmesnika brez uporabe kompleksne programske kode.



Slika 2.4: Razčlenjen pogled v Expression Blend prikazuje kodo in izrisan XAML.

Uporabniški vmesnik lahko razvijamo in izpopolnjujemo bodisi v pogledu kode ali pa preko orodij in panelov v oblikovalskem pogledu.

2.3.1 Zgradba XAML elementa

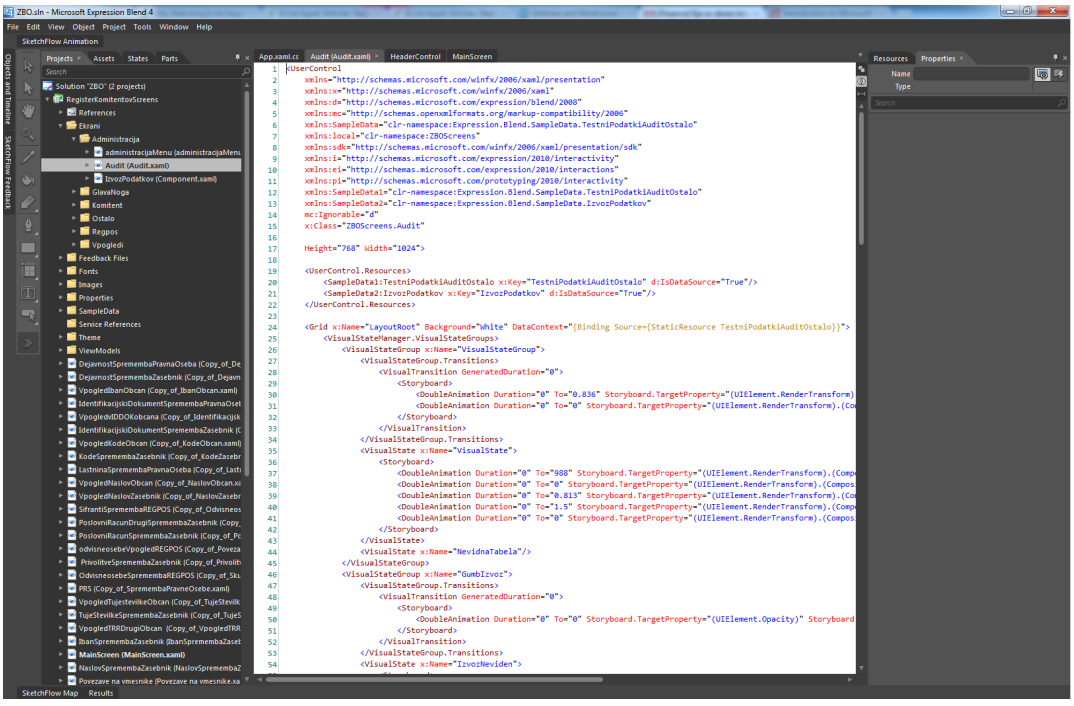
XAML elementi sestavljajo uporabniški vmesnik Silverlight aplikacije. Ti vključujejo slike, večpredstavnostne datoteke, scenarije in animacijo. Element XAML je izdelan, kot je prikazano v kodi:

```
<Ellipse x:Name="myEllipse" Height="31" HorizontalAlignment=
"Left"></Ellipse>
```

XAML elementi definirajo (ustvarijo) objekte, katerim lahko skozi kopico lastnosti in vrednosti nastavimo videz in obnašanje. Vzemimo pod drobnogled XAML element, ki definira elipso:

```
<Ellipse Height="60" HorizontalAlignment="Right" Margin="0,0,100,
10" VerticalAlignment="Bottom" Width="60" Fill="#FFCC0000"/>
```

Tukaj je kreiran objekt Ellipse, ki je polnjen z rdečo barvo, je širok in visok 60 slikovnih pik (60 px). Brez težav lahko ugotovimo, kako bo objekt izgledal na zaslonu, saj je jezik pisan na kožo človeku in je jasno razvidno, kaj bo prevajalnik ustvaril ter kje. Lastnosti HorizontalAlignment, VerticalAlignment in Margin objekt Ellipse postavijo relativno v vsebovalnik. XAML je osnova za zgled celotnega vmesnika Silverlight aplikacije, vključno z mediji, razporeditvijo elementov in animacijami lahko ustvari ter lahko prikazuje več kot samo grafiko v aplikacijah. Ko so scenariji in animacije narejeni, lahko interaktivno (kot so odzivi na klik in premikanje skozi element s kazalcem) nadzorujemo kontrole. Zaradi tega lahko v kodi poleg grafike vidimo še veliko druge XAML kode, ki nadzoruje interaktivne elemente in se ustvari ob kreiranju elementov.



Slika 2.5: Primer XAML izvorne kode za enostavno aplikacijo.

Poglavje 3

Zunaj brskalniške aplikacije (Out of browser applications)

Silverlight ima vgrajeno podporo, ki omogoča delovanje aplikacij izven brskalnika. Za delovanje aplikacije v načinu izven brskalnika je potrebno aplikacijo najprej nastaviti, nato pa jo lahko ponudimo uporabnikom is spletnega mesta. Po namestitvi lahko aplikacijo zaženemo brez brskalnika ali kakršnekoli spletne povezave, kakor druge samostojne aplikacije. Za nastavitev zunaj brskalniškega načina delovanja ne potrebuje spreminjati programske kode aplikacije. Poleg tega lahko v znaj brskalniško aplikacijo vpeljemo še dodatne funkcije, kot je na primer podpora za predpomnjenje v nepovezanem načinu delovanja ter avtomatsko posodabljanje aplikacije [11]. Konfiguracija zunaj brskalniške aplikacije je zelo enostavna in zagotavljanja dodatnih informacije o aplikaciji. Silverlight uporablja te informacije za prikaz namestitvenega uporabniškega vmesnika, bližnjice do aplikacije in zunaj brskalniškega aplikacijskega okna. Te informacije se nahajajo v manifestu aplikacije, zato ni potrebno ponovno prevajati, da aplikacija delovala izven brskalnika. Zunaj brskalniške aplikacije lahko tečejo brez omrežne povezave. Če aplikacija uporablja spletne vire, lahko v aplikacijo implementiramo detekcijo omrežja in implementiramo dodatno podporo za delo brez omrežne povezave, kadar ta ni na voljo. Zunaj brskalniške aplikacije lahko nastavimo tudi tako, da te

zahtevajo povečano zaupanje (varnost). Take aplikacije so overjene z digitalnim certifikatom, zato jim pravimo, da so zaupanja vredne. Za te aplikacije je značilno, da lahko obidejo nekatere varnostne omejitve mehanizma, ki skrbi za aplikacijsko varnost (ang. soundbox) [12].

3.1 Posebnosti zunaj brskalniške aplikacije

Zunaj brskalniške aplikacije, ki temeljijo na tehnologiji Silverlight, do verzije 4 lahko uporabljajo naslednje lastnosti, ki so onemogočene v aplikacijah, ki se nahajajo na spletnih strežnikih[12]:

- Upravljanje aplikacijskega okna: aplikacijsko okno lahko minimiziramo, maksimiziramo in ga nastavljamo po meri.
- Prilagajanje aplikacijskega okna: v zaupanja vrednih aplikacijah lahko skrijemo naslovno vrstico in okvir aplikacijskega okna.
- Gostovanje za HTML: prikažemo lahko HTML vsebino, s katero nadomestimo funkcionalnosti spletnih aplikacij.
- Obvestilna okna: v zunaj brskalniških aplikacijah lahko prikažemo okna z začasnimi obvestili.
- Podpora za upravljanje z digitalnimi pravicami medijskih datotek brez povezave.
- Povečano zaupanje: zaupanje vredne aplikacije lahko integriramo z osnovnimi funkcionalnostmi, ki nimajo enakih varnostnih omejitev kakor običajne Silverlight aplikacije.
- Dostop do datotečnega sistema: zaupanja vredne aplikacije lahko dostopajo do System.IO (omogoča dostop do datotečnega sistema) in podobnih knjižnic, ki so drugače onemogočeni v Silverlightu.

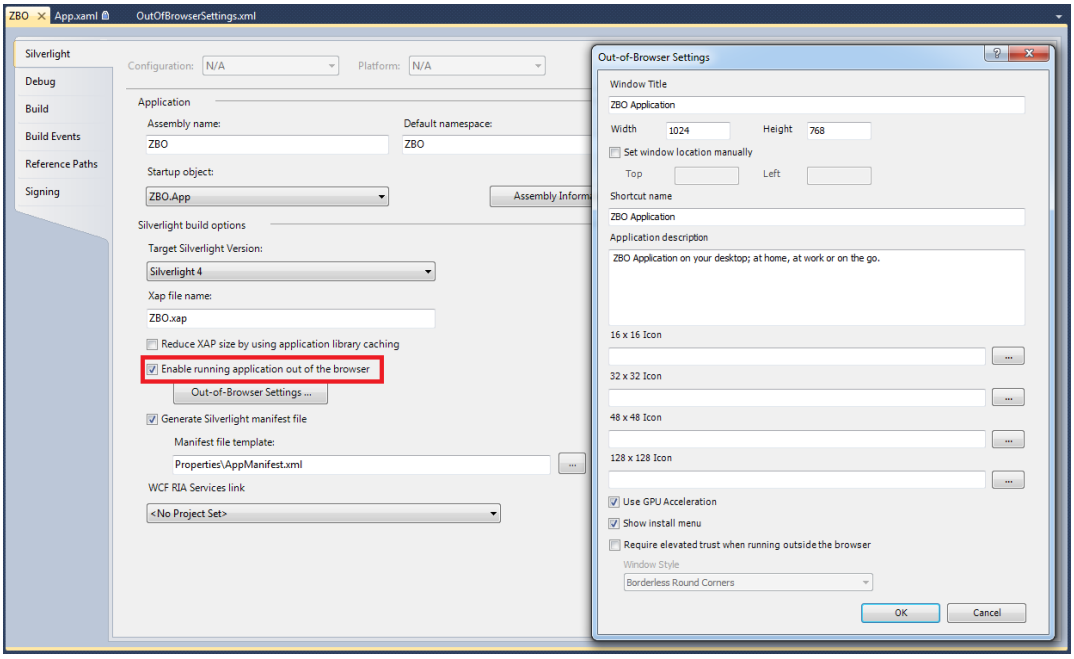
V zunaj brskalniških aplikacijah, ki temeljijo na Silverlight verziji 5, lahko poleg zgornjih funkcij dodatno vpeljemo še naslednje funkcionalnosti, ki v starejših verzijah niso na voljo:

- Večje število aplikacijskih oken: ustvarimo lahko več instanc oken, s katerimi lahko povečamo fleksibilnost uporabniškega vmesnika.
- Dodatne zaupanja vredne zmogljivosti.

3.2 Namestitev zunaj brskalniške aplikacije

Za prevajanje in poganjanje Silverlight aplikacij je potrebno slednje predhodno nastaviti. Expression Blend 4 nam posredno ne omogoča prevajanje in poganjanje aplikacij izven brskalnika, zato je potrebno projekt odpreti v programu Visual Studio 2010, kjer imamo na voljo nastavitve za poganjanje aplikacij in prototipov v načinu zunaj brskalnika. Ko se nam odpre projekt v Visual Studiu, poiščemo zavihek Project in v njem izberemo ZBO[1] Properties. V programu se nam naloži nov zavihek z nastavitvami za prevajanje, v katerem izberemo možnost Enable running application out of the browser. Ko izberemo ukaz za pogon aplikacije zunaj brskalnika, lahko v program vnesemo še dodatne nastavitve, kot je privzeta velikost aplikacijskega okna, uporaba grafičnega pospeševalnika in podobno. Prav tako bi lahko ob poznavanju ukazov vse zgoraj opisane nastavitve ročno vnesli v datoteko OutOfBrowserSettings.xml [13].

[1] ZBO - krajšava za ime aplikacije Zaledno Bančno Okence.



Slika 3.1: Nastavitve za izvajanje programa izven brskalnika.

App.xaml OutOfBrowserSettings.xml

```
1  <OutOfBrowserSettings ShortName="ZBO Application" EnableGPUAcceleration="True" ShowInstallMenuItem="True">
2    <OutOfBrowserSettings.Blurb>ZBO Application on your desktop; at home, at work or on the go.</OutOfBrowserSettings.Blurb>
3    <OutOfBrowserSettings.WindowSettings>
4      <WindowSettings Title="ZBO Application" Height="768" Width="1024" WindowStyle="BorderlessRoundCornersWindow" />
5    </OutOfBrowserSettings.WindowSettings>
6    <OutOfBrowserSettings.SecuritySettings>
7      <SecuritySettings ElevatedPermissions="Required" />
8    </OutOfBrowserSettings.SecuritySettings>
9    <OutOfBrowserSettings.Icons />
10 </OutOfBrowserSettings>
```

Slika 3.2: Datoteka z nastavitvami zunaj brsklaniške aplikacije.

3.3 Namestitev, brisanje in poganjanje zunajbrsklniške aplikacije

Namestitev zunaj brskalniške aplikacije lahko izvedemo na 2 načina, in sicer preko:

- spletne strani, na kateri se nahaja povezava za namestitev aplikacije in
- SLLauncher.exe programa, ki se ga uporablja za namestitev, odstranjevanje in poganjanje Silverlight oziroma xap datoteke.

V prvem primeru se nam po prenosu datoteke odpre namestitveno okno, v katerem lahko nastavimo, kje naj program ustvari bližnjice. Take aplikacije je potrebno predhodno nastaviti za delovanje izven brskalnika. V drugem primeru bomo za namestitev najprej potrebovali odpreti ukazno vrstico, preko katere bomo izvedli namestitev in ni potrebne spreminjati nastavitvenih datotek za aplikacijo. Ko se nam odpre ukazno okno, priporočam, da se najprej postavite na lokacijo, kjer se nahaja datoteka sslauncher.exe. Lokacija je odvisna od vrste operacijskega sistema. Običajno se program nahaja v

```
C:\<%ProgramFiles%>\Microsoft Silverlight
```

. V tabeli 3.1 se nahaja razlaga ukazov, ki sestavljajo program SLLauncher.exe.

Za namestitev programa potrebujemo vse 4 ukaze in xap datoteko, ki se nahaja v našem Expression Blend 4 projektu v mapi Bin*Debug* [14].

```
C:\ProgramFiles (x86)\Microsoft Silverlight>sllauncher
/install:\"C:\Users\matjaz\Desktop\ZBO - Register
komitentov\ZBO\Bin\Debug\ZBO.xap"
/origin:"http://localhost"
/shortcut:desktop+startmenu
/overwrite
```

Ukaz	Vrednost	Opis
/install	Pot do xap datoteke.	Lokacija, kjer se nahaja xap datoteka, ki jo bomo namestili na sistem.
/origin	URI do aplikacije.	Lokacija, na kateri program preverja za nove posodobitve aplikacije.
/shortcut	Desktop desktop $\|\|$*startmenu* desktop + startmenu.	Bližnjice do aplikacije.
/overwrite		Ali naj obstoječi program prepišemo z novim, ko so na voljo posodobitve.

Tabela 3.1: Razlaga ukazov za program SLLauncher.exe.

Ko vnesemo zgornje zaporedje ukazov v ukazno vrstico, se nam na namizju in v meniju start pojavi bližnjica z napisom ZBO Application. Če se premaknemo na lokacijo

```
C:\<%ProgramFiles%>\Microsoft
```

, lahko opazimo, da je program, preko katerega smo nameščali aplikacijo, prekopiral ZBO.xap datoteko v domačo mapo. Ob kliku na bližnjico se preko programa SLLauncher.exe požene ZBO aplikacijo, ki smo jo predhodno nastavili in prevesti z Expression Blend 4 ali Visual Studio. Brez težav lahko sledečo aplikacijo tudi odstranimo iz sistema s preprostim ukazom, kot ga vidite spodaj.

```
C:\ProgramFiles (x86)\Microsoft Silverlight>sllauncher
/uninstall
/origin:"http:"//localhost"
```

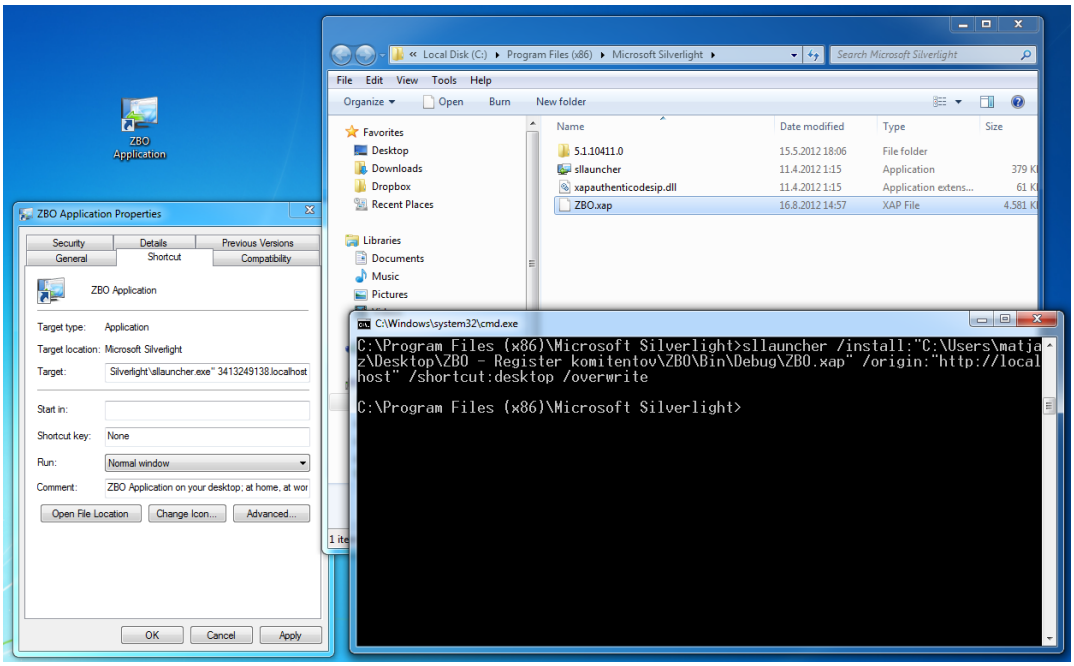
Ob brisanju aplikacije lahko opazimo, da se ZBO.xap datoteka ne odstrani is lokacije, na katero jo je namestil program sllauncher.exe.

Zunaj brskalniške aplikacije lahko brez težav tečejo tudi brez namestitve na obstoječi operacijski sistem. Zaganjanje ni prijetno za uporabnike, saj jo

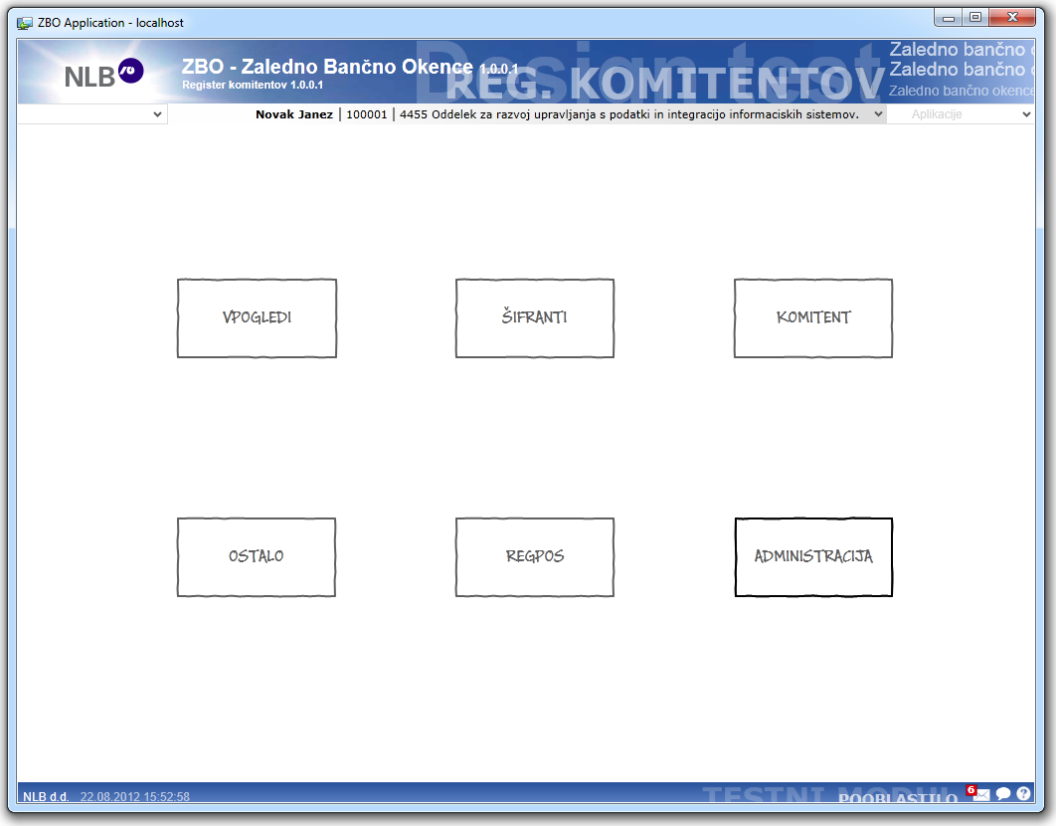
je potrebno poganjati iz ukazne vrstice preko sllauncher.exe programa. Za to potrebuje v ukazno okno vpisati naslednje zaporedje ukazov:

```
C:\ProgramFiles (x86)\Microsoft Silverlight>sllauncher
/install:"C:\Users\matjaz\Desktop\ZBO - Register
komitentov\ZBO\Bin\Debug\ZBO.xap"
/origin:"http://localhost"
/overwrite
```

Po potrditvi zgornjih ukazov se nam v brskalniku odpre aplikacijsko okno, v katerem teče Silverlight aplikacija.



Slika 3.3: Zaganjanje programa SLLauncher v CMD.



Slika 3.4: Prototip v Out-of-browser načinu delovanja.

Poglavje 4

Expression Blend 4

Microsoft Expression Blend 4 je orodje, namenjeno oblikovalcem za gradnjo privlačnih namiznih (WPF), spletnih ali mobilnih Silverlight aplikacij. Ker je dokaj novo orodje, je na začetku potrebna dodatna krivulja učenja, da lahko začnemo graditi smiselne in bogate aplikacije [2]. Sam program Expression Blend je bil napisan v tehnologiji. NET in WPF. Expression Blend je učinkovito interaktivno orodje, ki deluje na principu WYSIWYG[1]. Vmesniki so bazirani na XAML strukturi, ki je implementirana v WPF in Silverlight tehnologiji. Program omogoča načrtovanje namiznih in spletnih aplikacij, ki so sposobne izkoriščati tehnologijo na principu strojne (grafično pospeševanje) in programske opreme. Expression Blend ima za delo v naprej pripravljene funkcijske kontrole, kot so na primer meniji, gumbi, drsniki in seznami. V program lahko uvozimo razna grafične elemente in jih integriramo z Visual Basic ali C programsko kodo kot zaledni del uporabniškega vmesnika, ki ga kreiramo v Expression Blend [3]. V kombinaciji z najnovejšimi orodji v aplikacijah Microsoft Expression Studio, ki nadgrajuje okolje Visual Studio in se povezuje s strežnikom Windows Server, ima sedaj globalna skupnost spletnih razvijalcev na voljo vse, kar je potrebno za oblikovanje naprednih

[1]WYSIWYG je angleška kratica za What You See Is What You Get za programe pri katerih uporabniški vmesnik omogoča,da uporabnik med delom vidi kakšen bo končni rezultat rešitve.

bogatih internetnih aplikacij in večpredstavnih izkušenj [4].

4.1 Kaj je novega v Expression Blend 4

Expression Blend 4 se od svojih predhodnih različic na prvi pogled ne razlikuje veliko. Spremembe so v uporabniškem vmesniku komaj opazne in se odražajo predvsem v širšem izboru razvojnih projektov in nekaterih novih funkcionalnostih, za katere pa je potrebno programe Expression Blend že dodobra poznati, da jih razlikujemo. Expression Blend 4 ima dodane številne nove, izboljšane in nadgrajene funkcije iz predhodnih različic, hkrati pa so Microsoftovi inženirji v programu izboljšali delovni tok razvijalcev in oblikovalcev. Med številnimi izboljšavami, ki jih ponuja Microsoft Expression Blend 4, bomo našteli le nekaj izmed njih [5]:

1. Siverlight 4.0 (podrobnejši opis je na strani 6):
    - Silverlight podpora za Windows Phone 7.
    - ViewBox kontrola.
    - RichTextBox kontrola.
    - PathListBox kontrola.
    - Nova razporeditev elementov ListBoxItem.
    - Nove oblike.
    - Novi prehodi med zasloni.
        - Pogojni izvajanje prehodov.
    - Hitrejši oblikovanje predlog.
2. Združljivost s programom Visual Studio 2010.
3. WPF 4.0:
    - Upravitelj vizualnih stanj kontrol.
    - Poenostavljene funkcionalnosti za delo z animacijami.

4. SketchFlow:
    - Izboljšave v SketchFlow predvajalniku.
    - Izboljšave na področju avtorstva.
5. Podpora za razvoj aplikacij po principu MVVM[2].
6. Podatkovna shramba (Data Store):
    - Hitrejši čas načrtovanja.
    - Podpora za vzorce podatkov tipa CLR.

4.2 Alternativna orodja za prototipiranje uporabniških vmesnikov

Na tržišču imamo široko paleto različnih programskih rešitev za razvoj programske opreme v tehnologiji Silverlight. Ko si postavimo vprašanje, ali na tržišču obstaja konkurenčno orodje s katerim bi lahko v celoti nadomestili program Microsoft Expression Blend 4, zelo težko najdemo pritrdilen odgovor. Če želimo podati zadovoljiv odgovor na vprašanje, si je potrebno postaviti vprašanje kakšen pa je naš namen uporabe oziroma kaj bi želeli početi v Expression Blend? Če imamo namen le oblikovati uporabniški vmesnik za platformo Silverlight, ima Expression Blend kar nekaj konkurence. Za razvoj XAML kode lahko poleg programa Expression Blend izbiramo med številnimi orodij, od katerih bi izpostavil nekaj najbolj uporabnih:

- Expression Design
- Visual Studio
- Kaxaml

[2]MVVM - kratica za Model View View - model. Princip razvoja po katerem poteka razvoj Silverlight programske opreme.

- XDraw
- XamlPad (integrirano v .NET framework)
- Vector Arhitect
- ZAM 3D

Vsa zgoraj našteta orodja so namenjena razvoju XAML kode za spletne aplikacije na platformi Silverlight. V primeru, ko želimo v Silverlight rešitev implementirati tudi programsko logiko, je programu Expression Blend enakovreden le program Microsoft Visual Studio, ki je kompatibilen z vsemi Expression Blend projekti. Visual Studio se v večini primerov uporablja pri razvoju in implementaciji poslovne logike v aplikacijo. Program Expression Blend sicer na razpolago ponuja tudi razvoj programske oziroma poslovne logike, vendar pa je precej okoren za razvijalce, saj ne ponuja nekaterih priročnih funkcij (inteli-sense), ki razvijalcem omogočajo lažje in hitrejše delo. Poleg tega je potrebno poudariti, da nobeno od zgoraj navedenih orodij ne zagotavlja razvoja tako imenovanih SketchFlow prototipov, ki nam jih ponuja na razpolago Expression Blend. Zgoraj našteta orodja pokrivajo le del funkcionalnosti, ki jih ima na razpolago Expression Blend. Ker lahko z njimi po večini oblikujemo le grafično podobo za uporabniške vmesnike bomo sedaj predstavil še nekaj orodij preko katerih lahko razvijemo prototipe aplikacije. Tudi tukaj imamo na razpolago kar nekaj priročnih orodij, preko katerih lahko razvijamo skice oziroma prototipe za uporabniške vmesnike. Nekaj orodij bom naštel tukaj [6]:

- Slike
- Microsoft Office Visio
- Microsoft Power Point
- Axure RP Pro
- iRise

- Balsamiq Mockups
- Pencil
- MockupScreens

Razvoj prototipov s podobnimi programi je dokaj poenostavljen. Vsi ti programi nam ne ponujajo razvoja interaktivnih prototipov, s katerimi lahko povečamo uporabniško izkušnjo. Poleg tega je potrebno poudariti, da je velika prednost SketchFlowa enostavno izvažanje prototipov na SharePoint strežnik, ki uporabnikom zagotavlja enotno vstopno točko, preko katere uporabniki in stranke pregledujejo prototipe ter nam v istem trenutku podajo svoje mnenja in videnja. Te lahko razvijalci prototipov dobijo neposredno v Expression Blend, kjer jih lahko popravijo. Naslednja izmed slabosti, ki jih lahko pripišemo zgornjim programom, je ta, da se vsi prototipi po končani konceptualni fazi zavržejo, saj se jih ne da uporabiti pri nadaljnjem razvoju produkcijskega izdelka. Prototipi so v zgornjih programih videti kot preprosta skica, narejena s svinčnikom na listu papirja.

Poglavje 5

Predstavitev aplikacije Register komitentov

Celovita informacija o komitentu je danes ena najpomembnejših prvin vsake finančne institucije. Podatki o poslovanju komitentov s finančno institucijo nastajajo v različnih notranjih transakcijskih informacijskih sistemih organizacije. Za upravljanje in pregledovanje podatkov o komitentih finančne inštitucije uporabljajo razne aplikativne sisteme, ki jih v splošnem poimenujemo Registri komitentov. Te aplikativne sisteme morajo inštitucije tekom delovanja nadgrajevati in vzdrževati ter jih usklajevati z zakonskimi zahtevami. Ko potrebe prerastejo tehnološke zmožnosti, napoči čas, ko jih je potrebno prenoviti. Aplikacija Register komitentov je usmerjena na komitenta ali potencialnega komitenta finančne institucije. Zato mora zagotavljati enotno razpoznavanje komitenta, ki temelji na podatkovni bazi Registra komitentov. V podatkovni bazi komitente ločujemo na pravne in fizične osebe zaradi njihove specifičnosti, nekoliko pa se razlikuje tudi način poslovanja finančne institucije z obema skupinama [1]. Podpora Registra komitentov je v NLB nameščena oziroma se ob vsaki novi verziji namesti na približno 3500 delovnih postaj. V preteklih desetih letih so bile v letnem povprečju uspešno izdelane in nameščene 3 do 4 verzije. Pri tem je potrebno poudariti, da so se spremembe izvajale v skladu z zakonskimi zahtevami. Preko podpore Re-

gistra komitentov je do februarja 2010 bilo na poslovnem sistemu NLB d.d. ustvarjenih 1,64 milijona komitentov. Register komitentov uporablja skupno število, približno 3700 ljudi. V povprečju uporablja Register komitentov dnevno okoli 2600 uporabnikov iz NLB d.d., od tega 1600 uporabnikov izvaja tudi spremembe na podatkih komitentov. Uporabniki registra komitentov so izključno zaposleni z dodeljeno matično številko v kadrovski evidenci NLB d.d..

5.1 Register komitentov - pravnih oseb

Register združuje splošne podatke o pravnih osebah in samostojnih podjetnikih (matična, davčna številka, naziv, pravno organizacijska oblika, sedež, velikost, lastnina, dejavnost pravne osebe itd., podatke o vodstvu, lastništvu in povezanih osebah). Vse te podatke je mogoče dobiti iz uradnih virov. Posamezni viri podatkov imajo svoje prednosti in slabosti (obseg pravnih oseb, vsebina podatkov - atributi registra, ažurnost podatkov), zato slabo zadovoljujejo potrebe podatkovnega skladišča. S sestavljanjem podatkov iz različnih virov, pa je mogoče zagotoviti dober register komitentov podatkovnega skladišča. Kot ključ za povezavo podatkov se uporablja matična številka pravne osebe. Viri podatkov za Register komitentov podatkovnega skladišča so:

- Poslovni register Slovenije - PRS,
- Imenik pravnik oseb - IPO,
- Seznam davčnih zavezancev in
- administriran vnos.

Potencialna vira podatkov sta še:

- sodni register RS (omogoča vpogled v podatke posamezne pravne osebe, ni dosegljiv za prenos večjega obsega podatkov)in

- RIRZAPO (vsebuje podatke o direktorjih in ustanoviteljih, ne zagotavlja ločljivosti na nivoju osebe).

Register je zasnovan odprto in omogoča vključevanje dodatnih virov podatkov o pravnih osebah (npr. povezava splošnih podatkov o pravnih osebah iz internih sistemov finančne institucije).

5.2 Register komitentov - fizičnih oseb

Pri poslovanju s fizičnimi osebami se mora banka nasloniti na interne podatke, ko sestavljajo register fizičnih oseb. Pogosto se zaradi zgodovinsko pogojene ločene gradnje operativnih sistemov isti komitenti pojavljajo v različnih sistemih šifriranja z različnimi identifikatorji, zato imajo banke ponavadi tu implementirane posebne aplikacije, ki prečiščujejo Register komitentov (težava hranilne knjižice). Pogosto se razlikujejo tudi po pomenu enaki podatki o komitentih (npr. naslov komitenta v različnih registrih). Podatkov, ki bi lahko enolično opredelili komitenta (npr. EMŠO ali davčna številka), navadno v izvornih sistemih ni. Pri polnjenju podatkov iz različnih izvornih sistemov v register komitentov tako dobimo vrsto podvojenih komitentov, kar daje napačno sliko o poslovanju finančne institucije s komitentom (finančni podatki in transakcije istega komitenta so razdrobljeni na več matičnih podatkih, kar onemogoča celovito informacijo o poslovanju finančne institucije s komitentom), napačni pa so tudi nekateri globalni kazalniki poslovanja finančne institucije (npr. št. komitentov, s katerimi finančna institucija posluje) [1]. Zaradi vse bolj zastarele tehnologije, v kateri je aplikacija implementirana, se je leta 2011 začel projekt prenove Registra komitentov, v katerega spada tudi moja raziskava o možnosti vpeljave novega pristopa razvoja prototipov aplikacij, ki se bodo v bodoče prenavljale in razvijale, v okviru projektov razvoja v UCIT NLB d.d.. Tako sem v okviru diplomske naloge raziskal možnost vpeljave Expression Blend 4 v proces razvoja informacijskih sistemov v NLB. V nadaljevanju bom predstavil razvoj SketchFlow prototipov uporabniških vmesnikov na primeru prenove aplikacije Register

komitentov, ki se je izvajal v sklopu projekta prenove Registra komitentov v Novi Ljubljanski banki d.d.. Delo sem opravljal v oddelku za upravljanje s podatki in integracijo informacijskih sistemov banke, ki je organizacijsko umeščen v Sektor za razvoj informacijskega sistema banke, ki se nahaja v Upravljavskem centru za informacijsko tehnologijo NLB. V oddelku razvijajo in vzdržujejo številne infrastrukturne aplikacije, med katerimi je najpomembnejša Register komitentov, s katerim banka v elektronski obliki zagotavlja vse podatke o strankah za namen vključevanja le-teh v ciljne sisteme.

Poglavje 6

Razvoj SketchFlow prototipa na primeru prenove aplikacije Register Komitentov

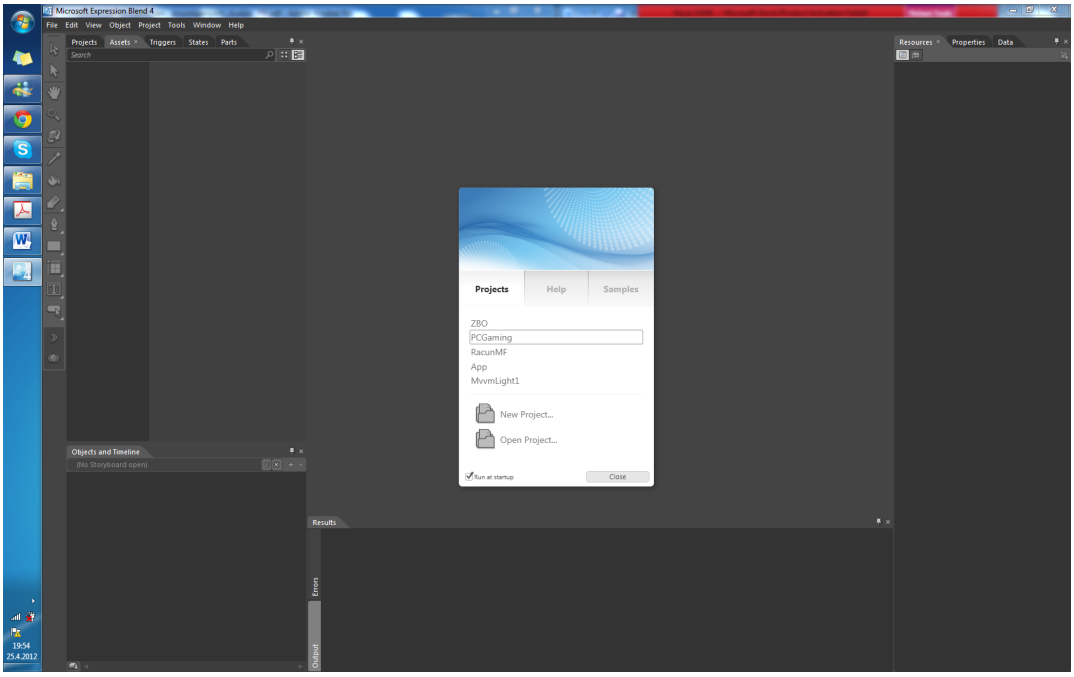
V okviru obsežnega projekta, si bomo ogledali, kako poteka razvoj SketchFlow prototipov v programu Expression Blend 4. Prva aplikacija, ki smo jo poskušali v NLB realizirati z novo tehnologijo, za oblikovanje in razvoj prototipov uporabniških vmesnikov, je bila Register komitentov, ki ga banka planira v celoti preseliti na Silverlight arhitekturo v sklopu ogrodja ZBO[1]. Prenova oziroma razvoj aplikacije Register komitentov poteka po metodologiji, ki je definirana v internem dokumentu banke in temelji na zaporednem slapovnem modelu razvoja. Razvoj SketchFlow prototipov uvrščamo v fazo specifikacije zahtev, prototipi pa se lahko uporabljajo v nadaljnjih razvojnih fazah od načrtovanja do kodiranja oziroma izdelave. Ker je obstoječa aplikacija predhodno že obstajala, je bilo pred tem pri analizi obstoječega stanja že narejenih 20 zaslonov, ki so bili skicirani v obliki Visio diagramov. Na podlagi teh zaslonov, ki predstavljajo obstoječo aplikacijo Register komitentov, so se istočasno, v procesu analize že specificirali primeri uporabe, v katerih so bile definirane akcije in dogodki zaslonskih mask. Glavni namen uporabe proto-

[1]ZBO - Zaledno bančno okence.

tipov je posredovati celovito sliko aplikacije vsem akterjem razvoja: ključnim uporabnikom, poslovnim analitikom, načrtovalcem in programerjem pri razvoju aplikacije Register komitentov. Zelo pomembno je dejstvo, da z njihovo pomočjo dosežemo maksimalne rezultate že v začetnih fazah razvoja, saj je od vseh akterjih, ki so vključeni v projekt, praktično nemogoče pričakovati enakovredno poznavanje tehnologije s katero bomo implementirali končno rešitev, uporabniki pa lahko že v samem začetku pridobijo sliko ciljnega sistema.

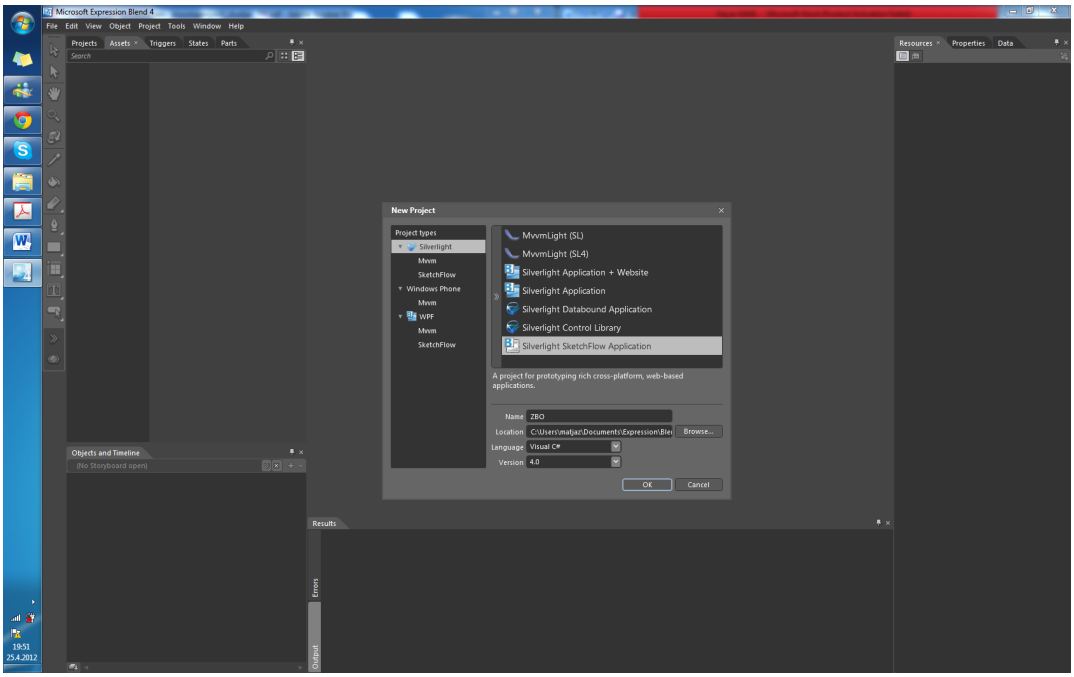
6.1 Vzpostavitev začetnega projekta

Ko prvič zaženemo Expression Blend, se nam prikaže panel z meniji. Enako kot v vseh predhodnih različicah programa lahko vidimo na zaslonu panel z zadnjimi odprtimi projekti. Na panelu lahko odpremo nov projekt ali odremo že obstoječi projekt, ki smo ga pred tem že ustvarili za nadaljnje delo.



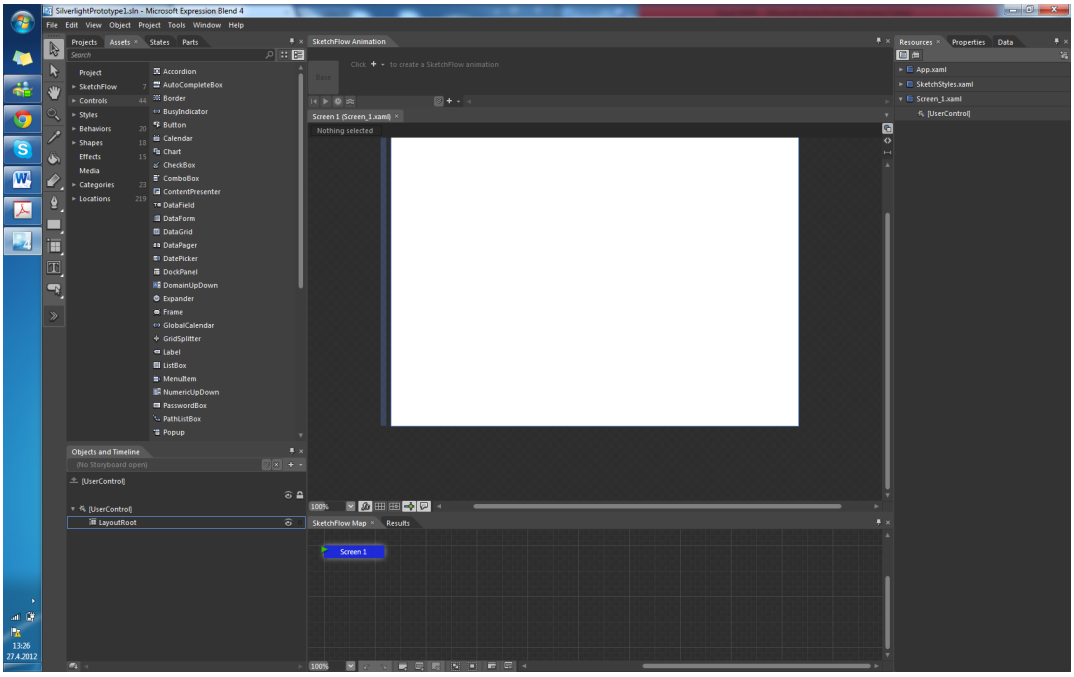
Slika 6.1: Pregled projektov v Expression Blend 4.

Če izberemo zavihek Projects, se nam prikaže modalno okno, ki nam ponuja številne možnosti. Za začetek bomo ustvarili nov projekt, ki se imenuje Silverlight SketchFlow Application. V modalnem oknu pod zavihkom Project izberemo New Project (glej sliko 6.2). Na zaslonu se nam pojavi novo modalno okno, v katerem nastavimo ustrezen tip projekta, določimo ime projekta, nastavimo jezik, v katerem bomo razvijali ter povemo, kam naj se projekt shrani (glej sliko 6.3).



Slika 6.2: Seznam projektov, ki jih lahko ustvarimo v Expression Blend 4.

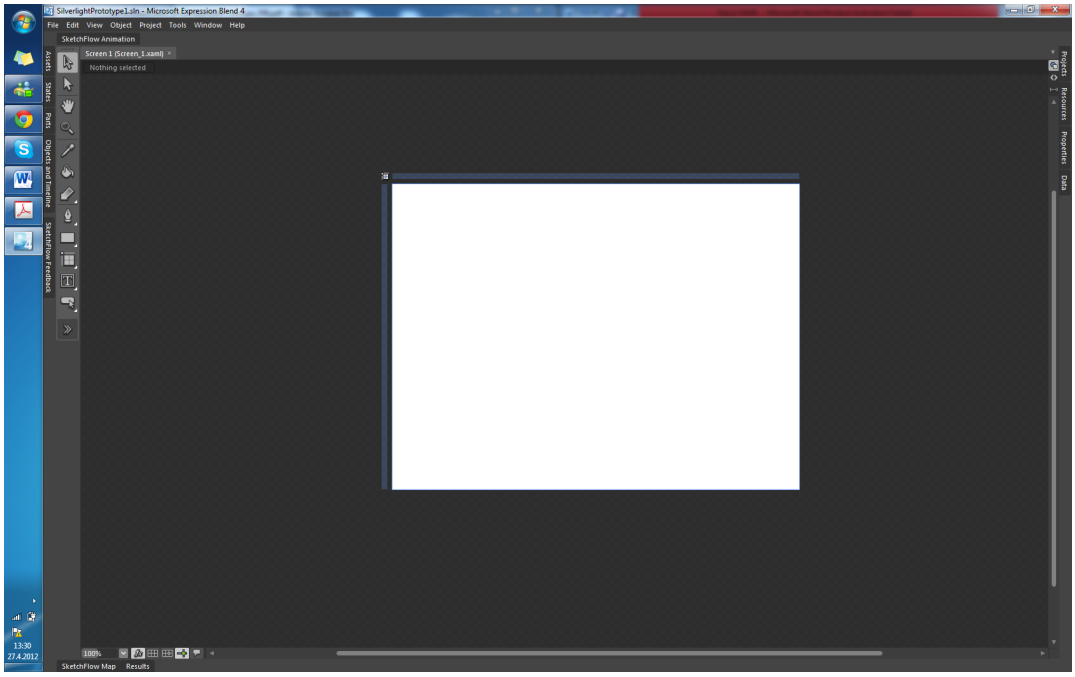
V našem primeru smo izbrali projekt Slivelight SketchFlow Application, saj bomo razvijali SketchFlow prototipe v tehnologiji Silverlight. Ko ustvarjamo nov projekt, se moramo seznaniti tudi z opcijo Language (jezik), ki nam na izbiro ponuja dva programska jezika, izmed katerih izberemo enega, v katerem bomo razvijali novo aplikacijo. Expression Blend nam na razpolago ponuja programski jezik Visual C in Visual Basic. Ker Silverlight SketchFlow projekt podpira možnost, da ga zapakiramo in odpremo v Silverlight Sket-



Slika 6.3: Delovni prostor ZBO projekta.

chFlow predvajalniku, lahko naš paket enostavno delimo z brskalnikom na Windows operacijskem sistemu ali drugih računalniških platformah. Alternativno bi lahko uporabili tudi WPF SketchFlow projekt, če bi želeli oblikovati rešitev s poudarkom na okenskih aplikacijah, vendar pa se je pri tem potem potrebno zavedati, da bomo tak tip aplikacije nato lahko uporabljali, pregledovali in delili samo na Windows platformi. Po pritisku na gumb OK (V redu) se nam odpre delovni prostor, ki zgleda približno tako, kot vidimo na sliki 6.4.

To je privzeti videz uporabniškega vmesnika za oblikovanje aplikacij v Expression Blend 4. Za lažje razumevanje uporabniškega vmesnika in njegovih funkcij bomo v nadaljevanju vsak posamezen panel opisali. Na zgornji sliki vidimo zaslon programa Expression Blend z razširjenimi paneli. Da bo predstavitev panelov bolj pregledna, bomo nekaj panelov, ki je za sedaj ne potrebujemo, skrčili oziroma zaprli (glej sliko 6.5).

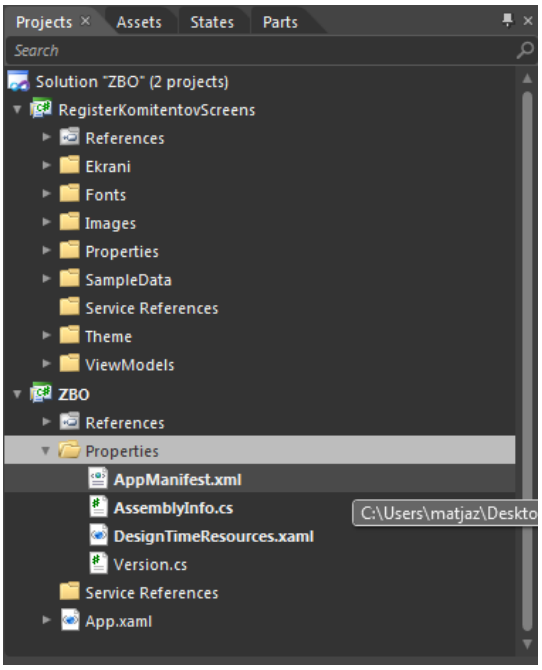


Slika 6.4: Okrnjen pogled na UV Expression Blend 4.

Če pogledamo zaslon od leve proti desni, na vrhu najprej opazimo meni, v katerem se nahajajo nastavitve programa, ki pa jih bomo v tem trenutku izpustili, saj niso relevantne za naše delo in bomo o njih po potrebi govorili v naslednjih poglavjih. Pod menijem se na levi strani nahaja zavihek SketchFlow Animation, na katerem se nahajajo funkcije, ki jih potrebujemo za delo z animacijami. Ker se z animacijami pri našem projektu ne bomo posebej ukvarjali, saj ponavadi poslovne aplikacije nimajo pretiranih vizualnih efektov, bomo ta zavihek odstranili iz našega delovnega prostora (workspace), da bomo imeli več dodatnega prostora.

6.1.1 Projekti (Projects panel)

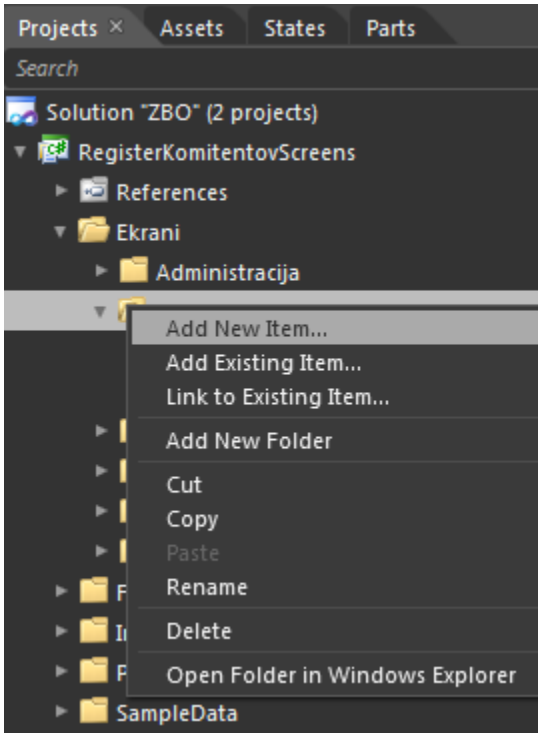
Plošča projektov vsebuje vse datoteke, ki jih uporabljamo v naših projektih. SketchFlow projekt je dejansko zbirka programov, datotek, knjižnic in povezana sredstva (reference assets), ki so namenjeni, da se projekt oziroma



Slika 6.5: Okrnjen pogled na UV Expression Blend 4.

aplikacija izvaja. Tipično tak tip projektov vsebuje zbirko sredstev, kot so vektorske grafike, pisave, slike in drugo. Običajno je plošča projektov razdeljena v 2 hierarhiji. Na vrhu prvega projekta, ki se imenuje RegisterKomitentovScreens, se nahajajo mape, ki vsebujejo datoteke, v katerih so definirani slogi pisave, oblike, kontrole, testni podatki, teme in podobno. Na začetku se v tem zavihku nahajaj tudi XAML datoteka, ki vsebuje začetni za Silverlight prototip. V drugem projektu se nahajajo datoteke, ki jih SketchFlow predvajalnik potrebuje za predvajanje prototipov. V imeniku References se nahajajo datoteke, ki jih imenujemo DDL in imajo datotečno strukturo, ki se konča z .dll. To so dinamične povezovalne knjižnice, ki skrbijo, da naša Silverlight aplikacija deluje pravilno. Takoj pod imenikov References se nahaja še imenik, ki se imenuje Properties. V Properties so zapisane nastavitve programa, ki identificirajo celotno skupno kodo, ki jo specifična aplikacija zahteva, in določajo, kako se poveže vsa druga koda, ki ni v skupni rabi in je del programa. V tem imeniku se nahajajo tudi povezave do kode, ki se

izvaja v ozadju (datoteke s končnico .cs) in XAML datotek, kjer so definirani slogi, kontrole in druga sredstva, ki jih uporabljamo v aplikaciji in se končajo s predpono .xaml. Preko plošče projektov lahko z miško enostavno vključujemo nove in že obstoječe predmete v projekt (slika 6.6).



Slika 6.6: Enostavno dodajanje predmetov z desnim miškinim klikom na projekt ali datoteko.

6.1.2 Kontrole (The Assets Panel)

Do vseh kontrol, ki so nam na razpolago v Expression Bend 4, lahko dostopamo preko menija Orodja, ki se nahaja običajno na skrajni levi strani zaslona vendar pa je bolj uporabno in priročno, če uporabljamo panel s kontrolami, ki se v našem programu imenuje Assets. Panel se nahaja zraven panela Project in States. Vsebuje veliko količino elementov, ki jim v računalniškem žargonu pravimo tudi kontrole, vendar pa vse niso relevantne oziroma tipične

za uporabo pri Silverlight SketchFlow projektu. Ko filtriramo, kontrole s pomočjo iskanja ali skupine, ki so fiksno definirane, nam upravljavec kontrol v nekaterih primerih ponudi podvojene kontrole, ki se nahajajo v več skupinah. Kontrole lahko iščemo v panelu kontrol, razdeljene so v naslednje kategorije [6]:

- **Project:** vsebuje vse specifične kontrole, ki jih imamo definirane v našem projektu, kot so zasloni, komponenta okna, uporabniške kontrole in podobno.

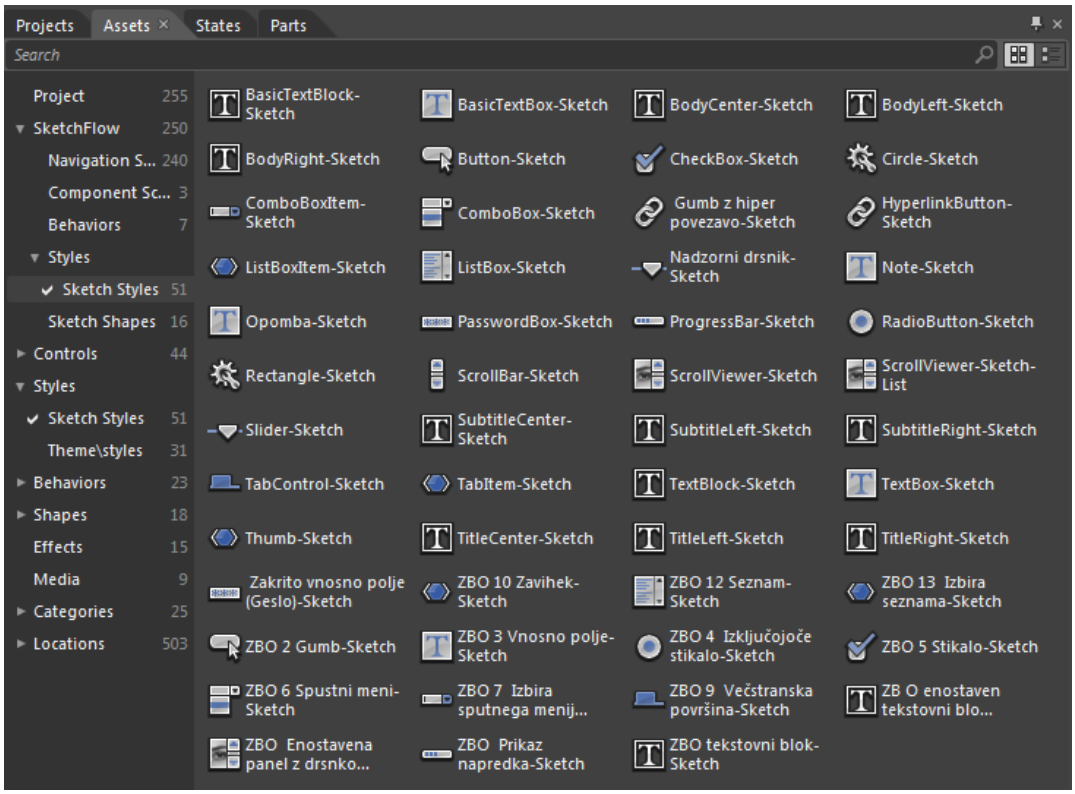
- **SketchFlow:** vsebuje navigacijska okna, komponentna okna, behaviours in stile, ki jih uporabljamo v aplikacijah vključno s SketchFlow stili. Podkategorije, ki sestavljajo skupino SketchFlow kontrolnikov, ponujajo bolj specifičen pogled kontrol. SketchFlow kategorija kontrol vsebuje vse kontrole v našem projektu, ki uporabljajo SketchFlow stile.

- **Controls:** vsebuje seznam vseh kontrol, ki so na voljo v projektu in seznam osnovnih kontrol, ki imajo zelo preprosto strukturo in jim je zato zelo lahko spremeniti njihov izgled v obliko, ki ustreza naši ureditvi ali podatkom, s katerimi se rokujejo.

- **Styles:** vsebuje enoličen seznam vseh stilov, ki smo jih ustvarili za naš projekt. Pod Styles spada vse, kar smo ustvarili, spremenili ali pretvorili v kontrolo (control) ali pa karkoli, kar bi lahko uporabili za kontrolo. SketchFlow Styles, ki ga uporabljamo v SketchFlow projektu, je dejansko knjižnica slogov, ki so narejeni izključno za uporabo v SketchFlow projektih.

- **Behaviours:** je kategorija, kjer shranjujemo vse specifične akcije, ki jih uporabljamo v projektu. Privzeto imamo v knjižnici na voljo že nekaj osnovnih kontrolnikov. V primeru, da želimo povečati naš nabor behaviour kontrolnikov, lahko te naložimo iz galerije, ki se nahaja na spletnem mestu Expression Gallery.

- **Effects:** ta kategorija vsebuje efekte, ki jih lahko uporabimo v naši aplikaciji. V programu je prvotno na voljo že nekaj efektov, ki pa jih lahko še dodamo, če jih namestimo enako kot v zgornjem primeru.
- **Medija:** je kategorija, ki nam ponuja kontrole, ki podpirajo 3D - objekte in objekte, ki podpirajo predvajanje avdio in video vsebin.
- **Category in Location:** sta kategoriji, ki vsebujeta kontrole in zbire (assemblies), ki so značilne za druge Silverlight ali WPF projekte.

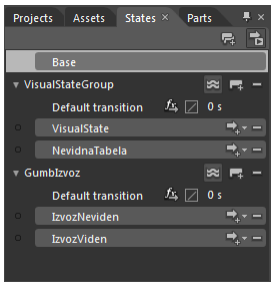


Slika 6.7: Kontrole lahko preko panela enostavno dodajamo na vmesnik.

6.1.3 Stanja (the states panel)

Panel stanj je namenjen ustvarjanju različnih stanj zaslonov v našem SketchFlow projektu. Če želimo prikazati zaslone v nekem danem stanju, moramo

le-te tudi ustvariti, da bi tako prikazali zaslon v nekem specifičnem stanju, ki je različen od osnovnega zaslona. Če se ne nahajamo v nekem stanju ali če ne ustvarimo nobenega stanja, smo v tako imenovanem osnovnem (base) stanju oziroma kategoriji. Stanja lahko vplivajo na vse kontrole oziroma elemente, ki sestavljajo zaslon ali pa samo na posamezne. Za vsak zaslon našega projekta imamo lahko poljubno število stanj.



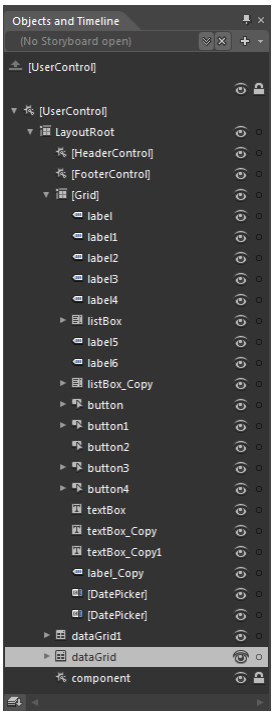
Slika 6.8: Na panelu stanj lahko posnamemo različne scenarije vmesnika.

6.1.4 Panel objektov in časovnica (Object and the timeline panel)

Panel objektov predstavlja logično postavitev elementov vmesnika po principu od spodaj navzgor glede na njihovo deklaracijo v XAML kodi, ki tvori naš prototip in aplikacijo.

6.1.5 Obdelovalno okno (Artboard panel)

V zgornjem delu našega zaslona se nahajajo zavihki, ki se nahajajo v obdelovalnem oknu (ang. Artboard). Njihovo število nam pove, koliko zaslonov imamo trenutno odprtih v SketchFlow projektu. Beli zaslon je podoben pladnju in predstavlja prazen list papirja oziroma zaslona, na katerem bomo zgradili prototip. Ta zaslon je delovna površina, na katero lahko po principu potegni in spusti dodajamo elemente is panela kontrol, ki smo jih uvozili



Slika 6.9: Panel Object and Timeline vsebuje vse objekte, ki se nahajajo na zaslonu.

v naš SketchFlow projekt ali pa smo jih ustvarili direktno preko obdelovalnega okna. Obdelovalno okno lahko tudi poljubno prilagodimo. V ta namen imamo na voljo nekaj nastavitev, ki nam omogočajo prilagajanje velikosti zaslona. Nastavimo lahko poljubno barvo ozadja, povečamo zaslon, definiramo zaklopljene linije n podobno. Na desni strani panela se nahajajo ikone, s katerimi preklapljamo med različnimi pogledi obdelovalne površine. Najpogosteje delamo v pogledu dizajna (ang. Design view), preklapljamo pa lahko še med pogledom kode in razdeljenem pogledu, ki nam omogoča, da imamo hkrati pregled nad obliko prototipa in kodo, ki se generira ob postavljanju kontrol na prototip ali obratno. Na spodnjem delu obdelovalnega okna imamo na razpolago še nekaj drugih ikon, kjer si z leve proti desni sledijo [6]:

- Ikona Zoom nam omogoča povečavo zaslona, s katero si lahko poljubno

približamo ali pomanjšamo zaslon, na katerem delamo.

- Ikona Efects vključi ali izključi prikazovanje efektov v realnem času, ki jih bomo mogoče dodali v objekte.

- Ikona Grid prikaže ali skrije mrežo v našem obdelovalnem oknu. Uporabljamo jo za natančno in lažje pozicioniranje objektov na vmesniku.

- Ikona Grid Snapping omogoča, da program potisne ali pa avtomatsko poravna kontrolo v linijo na mreži, ki smo jo predhodno definirali.

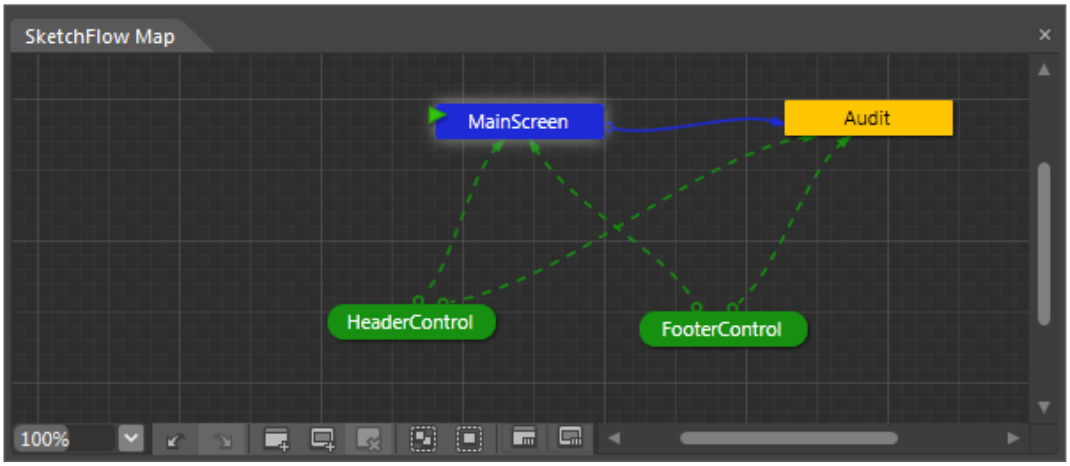
- Ikona Snapping to Snaplines: omogoča, da program potisne kontrolo na zaklopljeno linijo, ki smo jo predhodno nastavili v dokumentu. Zaklopljene linije ustvarimo za mrežo ali druge vrste postavitve, s katero razdelimo dokument po horizontalni in vertikalni smeri.

- Ikona zaznamek (Annotation) prikazuje zanamke, ki smo jih mogoče dodali v naš projekt.

6.1.6 SketchFlow zemljevid (SketchFlow Map)

SketchFlow zemljevid je panel, kjer lahko ustvarimo in urejamo zemljevid naše aplikacije. Zemljevid predstavlja reference med zasloni, kar si lahko predstavljamo kot zemljevid, ki pove, kateri zaslon je naslednji v interakciji, ki se bo prikazal na zaslonu. Na dnu panela se nahajajo kontrole za kontroliranje stopnje povečave zemljevida, funkciji za razveljavitev in uveljavljanje, dodajanje in odstranjevanje elementov in kontroli za prikaz izbranih ali vseh elementov na našem zaslonu.

Modro polje na zemljevidu, ki je označeno z napisom "Audit", se imenuje vozlišče (angl. node). Vozlišče na zemljevidu predstavlja zaslon oziroma belo platno, ki se nahaja v obdelovalnem oknu. Ko gremo z miško skozi vozlišče, se nam prikaže panel z dodatnimi možnostmi, kjer imamo na razpolago možnost ustvariti novo okno oziroma vozlišče z lastnostmi, ki so podobne obstoječim vozliščem ali pa ustvariti poseben tip vozlišča, ki predstavlja komponento



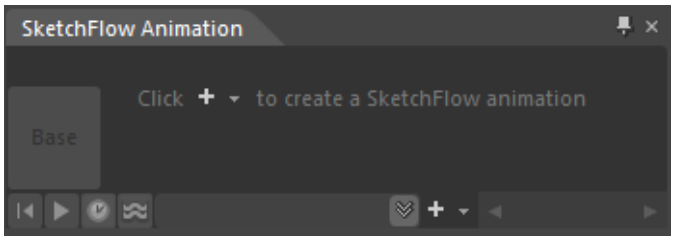
Slika 6.10: SketchFlow zemljevid nam omogoča enostavno povezovanje med zasloni.

okno (ang. the component screen). Da okna med sabo lažje ločimo, imamo na voljo tudi funkcijo "Change visual tag", preko katere lahko vozlišču nastavimo poljubno barvo polja. Ta možnost je še posebej uporabna pri večjih projektih, kjer z barvo ločimo posamezne sklope projekta.

6.1.7 Animacije (Animation panel)

Če želimo pokazati, kako bo izgledal naš vmesnik ob različnih interakcijah z aplikacijo, lahko uporabimo animacije. Expression Blend 4 ima zelo napreden sistem za animacije, ki se imenuje SktechFlow Animation. Ta nam omogoča hitro izdelavo animacij, ki simulirajo napredno interakcijo vmesnika. Animacija v SketchFlow je osnovana na sistemu Visual State Manager, ki je implementiran v Expression Blend. Animacije je mogoče povsem enako zgraditi tudi preko programske kode ali pa v okiru napredne funkcije Behaviours, ki jo ima SketchFlow. Ker je včasih potrebno izdelati veliko rešitev, na podlagi katerih se nato odločimo, katera rešitev dejansko deluje najbolje, je to v veliko primerih najboljši pristop, saj lahko animacijo izdelamo zelo hitro. Animacije so včasih najboljše orodje za hitro iskanje najbolj optimalnih

rešitev.

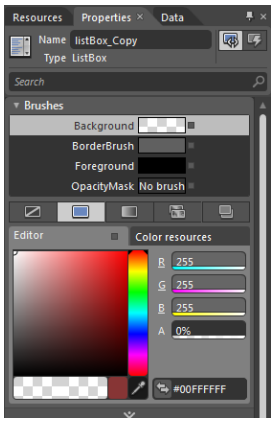


Slika 6.11: Panel z animacijami.

6.1.8 Lastnosti (Properties panel)

Panel z lastnostmi je konteksten.Prikazuje samo elemente, ki jih lahko urejamo ali spreminjajmo dani objekt. To pomeni, da se razlikuje glede na izbrano kontrolo. Panel je razdeljen na kategorije:

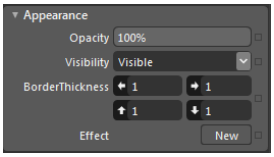
- V Brushes nastavimo barvo ozadja, pisave in črt. V tem razdelku lahko nastavljamo tudi prosojnost, ki se nahaja tudi v razdelku Apperance.



Slika 6.12: V panelu z lastnostmi lahko prilagajamo kontrole uporabniškim zahtevam.

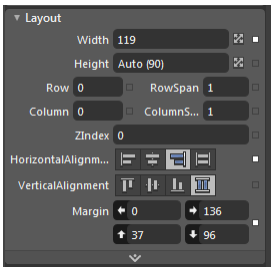
- Apperance vsebuje lastnosti, ki so zelo uporabne, saj vplivajo na videz objekt v vmesniku. Najpogosteje se v tem sklopu nastavitev uporablja

kontrola za nastavitev motne slike (opacity), s katero lahko objekt tudi skrijemo. V meniju lahko posameznemu objektu po potrebi nastavimo tudi nekaj enostavnih efektov (npr.: blur in drop shadows), ki jih ponavadi ne nastavimo, dokler ne ustvarimo že zelo dovršenega prototipa.



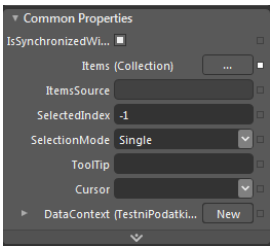
Slika 6.13: Panel z lastnomi Apperance.

- Layout kategorija nastavitev ponuja osnovne in nekaj naprednih nastavitev našega objekta. Nastavimo lahko lastnosti, ki določajo velikost, postavitev, poravnavo in rob objekta, ki je vgrajen v vmesnik.



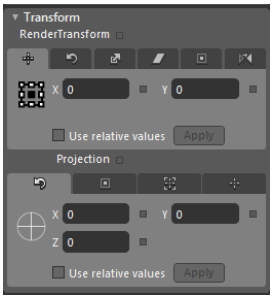
Slika 6.14: Panel z lastnostmi Layout.

- Common Properties kategorija ima številne dodatne funkcije, ki včasih niso primerne za SketchFlow in jih v takem primeru Expression Blend ne upošteva. Tukaj je nekaj parametrov, ki so pomembni za dinamično prototipiranje. Dodamo lahko tudi vsebino, ki se bo prikazala na vmesniku, ko se bomo z miško postavili na kontrolo (ToolTip funkcija) in pod oznako Cursor določimo, kakšna naj bo oblika miške, ko bomo šli skozi element.



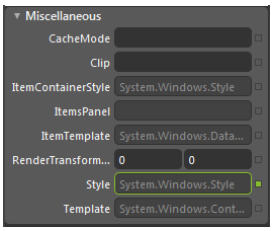
Slika 6.15: Panel z lastnostmi Common Properties.

- Transform se uporablja pri vseh vizualnih kontrolah, ki jih želim preoblikovati. Ponavadi objekt prilagodimo že v obdelovalnem oknu, ne glede na vse pa jih imamo možnost nastavljati tudi tukaj.



Slika 6.16: Panel z lastnostmi Tranform.

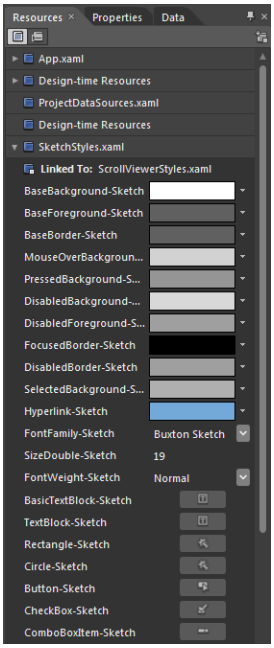
- Miscellaneous se ponavadi ne uporablja pri izdelavi prototipov, razen če smo napreden uporabnik, ki želi prilagoditi objektu vse lastnosti.



Slika 6.17: Panel z lastnostmi Miscellaneous.

6.1.9 Viri (Resources panel)

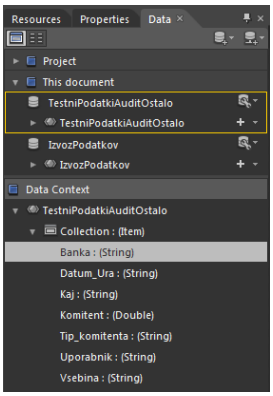
Panel virov hrani vse stile in vire (kontrole), ki se nahajajo trenutno izbranem vmesniku ali v našem projektu. V virih se nahajajo knjižnice, viri podatkov in vsi specifični elementi iz katerih je sestavljen trenutno odprt vmesnik. Podobno kot v panelu kontrol ga lahko uporabljamo za dodajanje kontrol na vmesnik ali pa z njim dodamo sloge za obstoječe kontrole v projekt.



Slika 6.18: Panel virov.

6.1.10 Podatki (Data panel)

Kadar želimo naše prototipe narediti bolj dinamične, se poslužimo orodij za delo s podatki. Podatkovni panel je namenjen delu z živimi ali simuliranimi podatkovnimi viri.



Slika 6.19: Podatkovni viri za projekt.

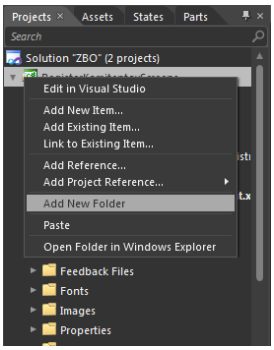
6.1.11 Povratne informacije (Feedback panel)

Panel za povratne informacije omogoča pregledovanje in uvažanje povratnih informacij v našem SketchFlow projektu.

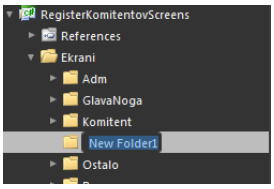
6.2 Kreiranje zaslona - okna

Ob vzpostavitvi začetnega projekta se nam že avtomatsko ustvari začetno okno, ki se imenuje "Start". Ker nam tako poimenovanje ni všeč, bomo okno preimenovali v "MainScreen". Prvo okno bo namenjeno meniju, v katerem bo uporabnik izbral, kaj želi početi v aplikaciji. Po uspešnem vnosu dodamo še eno okno, ki ga bomo poimenovali Audit. V tem oknu se bo nahajal prototip uporabniškega vmesnika, ki bo vključeval skoraj vse funkcionalnosti, ki nam jih ponuja Expression Blend 4. Okno lahko ustvarimo na več načinov. Najlažji način dodajanja oken poteka preko SketchFlow zemljevida, vendar pa ima ta način tudi eno pomanjkljivost. Okna v tem primeru praktično ne moremo prestavljati med mapami v projektu. To zna biti velik problem v primeru, ko postane aplikacija velika. Panel Projekti zaradi tega postane nepregleden, kar pa uporabnika upočasnjuje pri svojem delu.

Ker vemo, da bo naš projekt obsegal več 10 prototipov uporabniških vmesnikov bomo ustvarili mapo "Ekrani", ki bo služila kot splošna lokacija, kjer



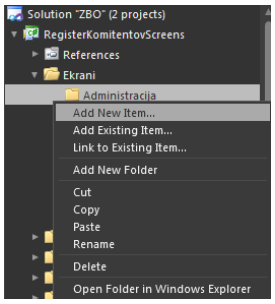
Slika 6.20: Dodajanje datotek v projekt.



Slika 6.21: Preimenovanje nove datoteke Ekrani.

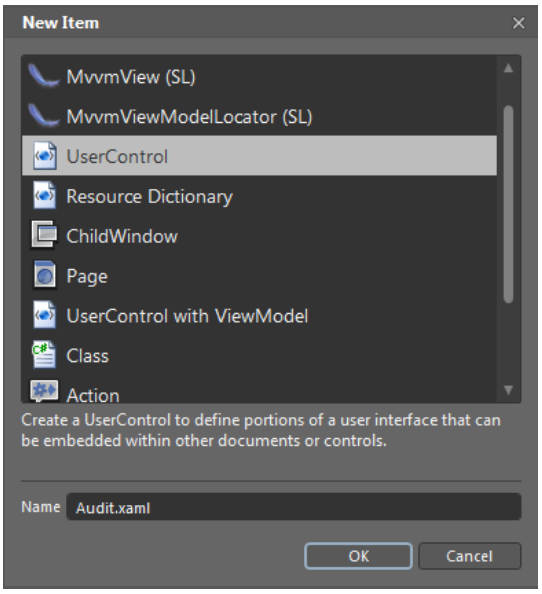
se nahajajo vsi vizualni elementi. Mape v projektu dodajamo na sledeči način. Z miško se postavimo na ikono RegisterKomitentovScreens in kliknemo desnim gumb na miški. Pokaže se nam modalno okno, kjer s klikom ali preko kurzorja na tipkovnici izberemo "Add new folder"(slika 6.2). V panelu Projects se nam pokaže nova mapa, ki ji moramo samo še spremeniti ime v "Ekrani" (slika 6.2). Poleg tega bomo za začetek ustvarili še podderiktorij Administracija, v katerem se bo nahajal zaslon, ki ga bomo sedaj dodali. Postopek dodajanja je enak kot v zgornjem primeru, le da se tukaj s kurzorjem miške postavimo na mapo "Ekrani". Novo okno dodamo v mapo Administracija, tako da se z miško postavimo na njo in z desnim miškinim klikom se nam prikaže modalno okno, v katerem izberemo izbiro "Add new item" (slika 6.2).

Na zaslonu se nam odpre novo modalno okno (slika 6.23), v katerem lahko izbiramo številne tipe zaslonov. Ker imamo projekt tipa Silverlight SketchFlow in želimo dodati novo okno, izberemo izbiro "UserControl". Preden



Slika 6.22: Dodajanje novega zaslona v projekt.

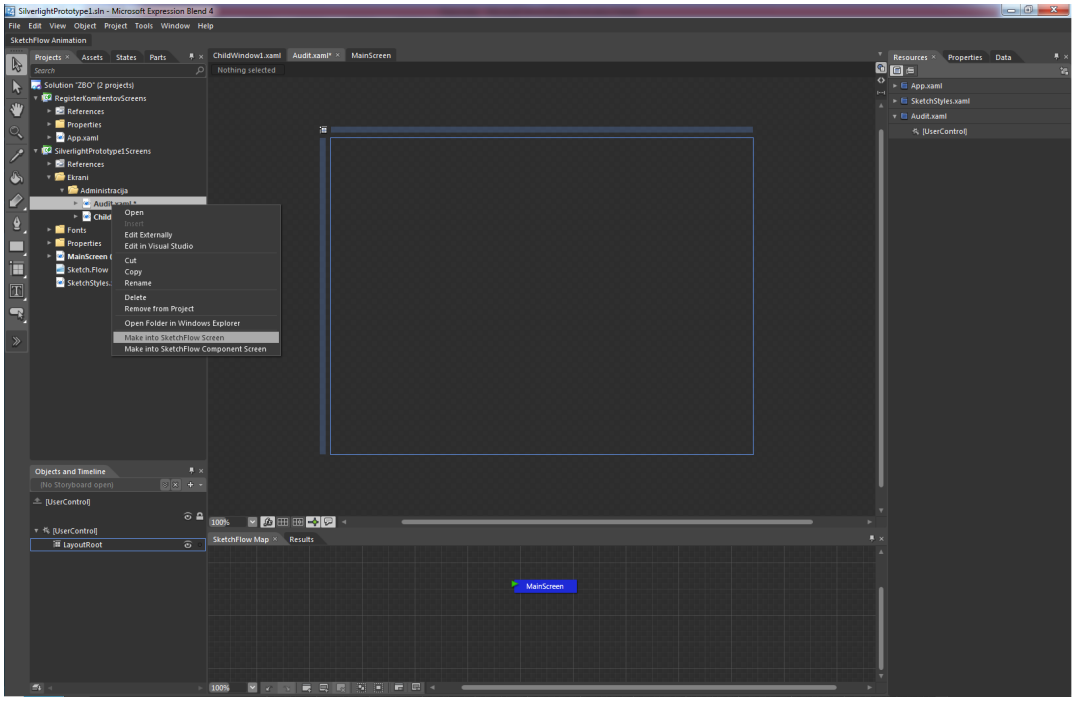
potrdimo izbiro za novo okno je potrebno v polju Name nastaviti še ime, ki je prvotno že nastavljeno, ki pa nam v našem primeru zaradi lažjega ločevanja med zasloni ne ustreza. Poimenovali ga bomo "Audit" in potrdili izbiro s klikom na gumb OK. Ko potrdimo izbiro, se nam v obdelovalnem oknu odpre prazen panel.



Slika 6.23: New Item panel za dodajanje elementov v projekt.

Ker bomo v aplikaciji prehajali med obema zaslona, jih je potrebno povezati v tako imenovane "povezane zaslone" (ang. Conected screens). To

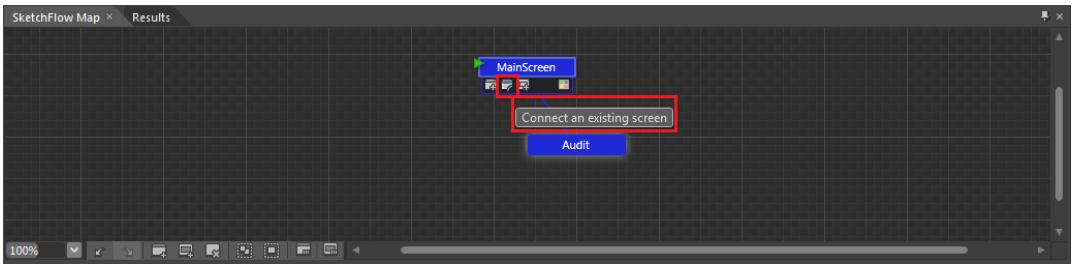
se opravi v SketchFlow zemljevidu, kjer zaslone med seboj enostavno poveznemo po principu povleči in spusti prvi zaslon v drugega (slika **??**). Preden se lahko zgornjega opravila lotimo, je še potrebno poskrbeti, da naše prototipe pretvorimo v SketchFlow zaslone, da jih bomo dodali na SketchFlow zemljevid. Tam jih bomo nato povezali med seboj.



Slika 6.24: SketchFlow zaslon predstavlja vozlišče na SketchFlow zemljevidu.

SketchFlow zaslon se ustvari preko panela Project v katerem izberemo ciljni zaslon in v modalnem oknu, ki se nam prikaže ob desnem miškinem kliku na ikono Audit.xaml, izberemo možnost "Make into SketchFlow screen" (slika 6.24). Ko zaslon pretvorimo v SketchFlow zaslon, se ta prikaže na našem SketchFlow zemljevidu, kjer ga lahko nato povežemo z drugimi zasloni, spreminjamo barvo vozliča, na njega obesimo komponentna okna in podobno.

Ko ustvarimo povezave, lahko z uporabo navigacije krmarimo med zasloni. Povezan zaslon ustvarimo v SketchFlow zemljevidu z enostavnim potegom MainScreen vozlišča v vozlišče Audit ali preko ikone, ki se nam pokaže



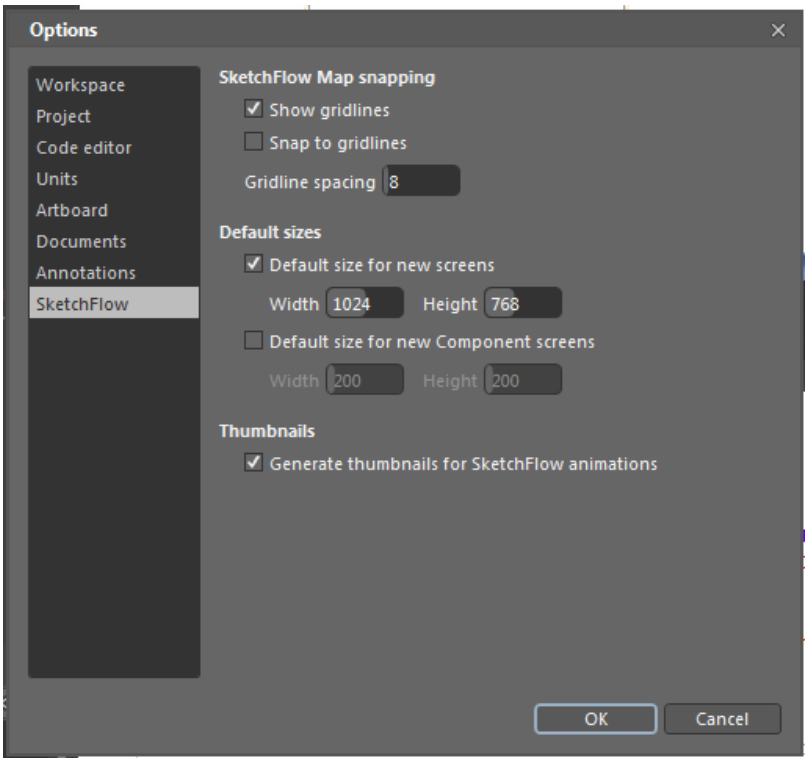
Slika 6.25: Povezovanje zaslonov (Connected screen).

v modalnem oknu vozlišča, če se z miško postavimo nanj, kot je prikazano na sliki 6.25.

6.3 Nastavitev privzete velikosti zaslona v Expression Blend 4

Vsak razvojni projekt imam svoje zahteve, ki se jih je potrebno držati. Edna od osnovnih zahtev, ki naj bi jih imela naša aplikacija, je celostna podoba aplikacije, ki je identična z vsemi drugimi ZBO (ZBO - zaledno bančno okence) aplikacijami. V osnovi to pomeni, da je privzeta velikost zaslona 1024 x 768 slikovnih pik. Zgradba zaslona mora biti sestavljena iz ZBO glave in noge, ki uporabnika vodita oziroma obveščata z informacijami o času, sporočilih, pooblastilih za uporabo aplikacije, lokaciji, kjer se uporabnik nahaja v aplikaciji in še čemu. V aplikacijskem meniju Tools (orodja) se nahaja element Options, kjer bomo nastavili privzeto velikost vsakega SketchFlow zaslona. Ko se nam na namizju pokaže modalno okno Options, v levem padajočem meniju poiščemo izbiro SketchFlow, kjer se nam ob kliku nanj pokaže kopica nastavitev. Da bi program za vsak nov zaslon uporabil našo prednastavljeno velikost je potrebno izbrati privzeto velikost za nov zaslon "Default size for new sceen" (slika 6.26). V primeru, da naša privzeta velikost ni pravilna, v polje Width vpišemo vrednost 1023, v polje Heigh pa vrednost 768. Po uspešnem vnosu naše nastavitve potrdimo s klikom na

gumb OK.



Slika 6.26: Nastavitev privzete velikosti novega zaslona.

6.4 Uvažanje po meri izdelanih virov iz dinamičnih povezovalnih knjižnic v projekt

Naslednji korak pri izdelavi naših prototipov je uvažanje ZBO slogov dinamičnih povezovalnih knjižnic v naš projekt. Kontrole, sloge in efekte, ki jih uvozimo preko .DLL datotek, lahko uporabljamo v obdelovalnem oknu s pomočjo panela kontrol. Ker smo že v začetku povedali, da bo naš projekt ustrezal ZBO arhitekturi aplikacij, bomo uporabljali tudi kontrole, ki se v Expression Blend 4 ne nahajajo. Če želimo v projekt dodati .DLL datoteko, moramo slednjo dodati v oba projekta datoteke References. Ker nekateri

objekti, ki jih vsebuje knjižnica .DLL zahtevajo posebne knjižnice za delovanje je potrebno te tudi dodati notri, saj imamo pri prevajanju našega projekta drugače težave.

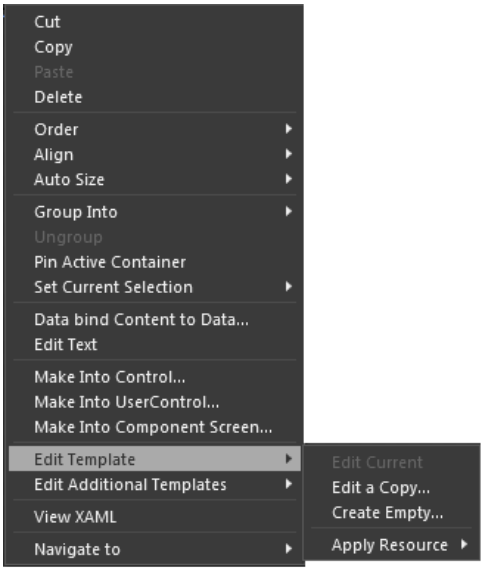
6.5 Spreminjanje SketchFlow kontrol

Ker bo naš projekt služil kot osnova za vse nadaljnje projekte, bomo preimenovali najbolj uporabne kontrole v slovenski jezik in jim določili skupne lastnosti. To bomo naredili zaradi tega, ker ne bodo vsi uporabniki na projektih koderji, ampak tudi poslovni analitiki in drugi uporabniki, ki ne poznajo računalniškega žargona, še posebej, če jim je tuj angleški jezik. Vsaka kontrola se mora začeti z imenom ZBO ¡ zaporedna številka ¿, da jih Expression Blend 4 sortira po vrstnem redu. Slogi se nahajajo v datoteki SketchStyles.xaml, kjer je potrebno za vsako kontrolo spremeniti ime, ki ga definiramo z značko x:key=≪Ime kontrole¿”. Spremenili smo tudi velikost pisave, ki je po novem znašala 16 pikslov, dodatno pa je potrebno nastavili tudi višino kontrol na 23 pikslov. Ker vse kontrole niso originalno namenjene za SketchFlow, jim je potrebno spremeniti tudi pisavo iz FontFamily=”TemplateBinding FontFamily”v FontFamily=”StaticResource FontFamily-Sketch”.

```
3292                </ControlTemplate>
3293            </Setter.Value>
3294        </Setter>
3295    </Style>
3296    <!-- Sketch TextBox -->
3297    <Style x:Key="ZBO 3 Vnosno polje-Sketch" TargetType="TextBox" BasedOn="{StaticResource BasicTextBox-Sketch}">
3298        <Setter Property="FontSize" Value="16"/>
3299        <Setter Property="Height" Value="23"/>
3300    </Style>
3301    <!-- Sketch RadioButton -->
3302    <Style x:Key="ZBO 4 Izključojoče stikalo-Sketch" TargetType="RadioButton">
3303        <Setter Property="Background" Value="{StaticResource BaseBackground-Sketch}"/>
3304        <Setter Property="Foreground" Value="{StaticResource BaseForeground-Sketch}"/>
3305        <Setter Property="HorizontalContentAlignment" Value="Left"/>
3306        <Setter Property="VerticalContentAlignment" Value="Top"/>
3307        <Setter Property="Padding" Value="4,1,0,0"/>
3308        <Setter Property="BorderThickness" Value="1"/>
3309        <Setter Property="BorderBrush" Value="{StaticResource BaseBorder-Sketch}"/>
3310        <Setter Property="Template">
3311            <Setter.Value>
3312                <ControlTemplate TargetType="RadioButton">
3313                    <Grid Background="#00000000" Cursor="Hand">
3314                        <Grid.ColumnDefinitions>
```

Slika 6.27: Primer urejanja vnosnega polja.

Vsako kontrolo lahko spreminjamo tudi preko obdelovalnega okna. Z desnim gumbom miške kliknemo na kontrolo, ki jo želimo spremeniti, da se nam odpre modalno okno, ki nam na razpolago ponudi možnost urejanja predloge "Edit Template".

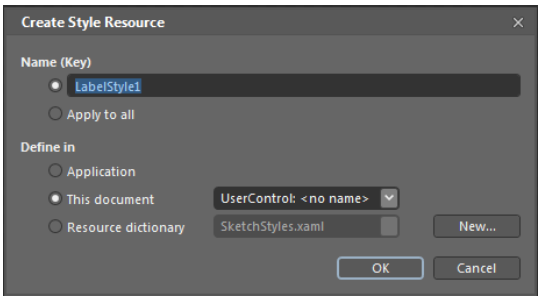


Slika 6.28: Prilagajanje kontrol potrebam aplikacije.

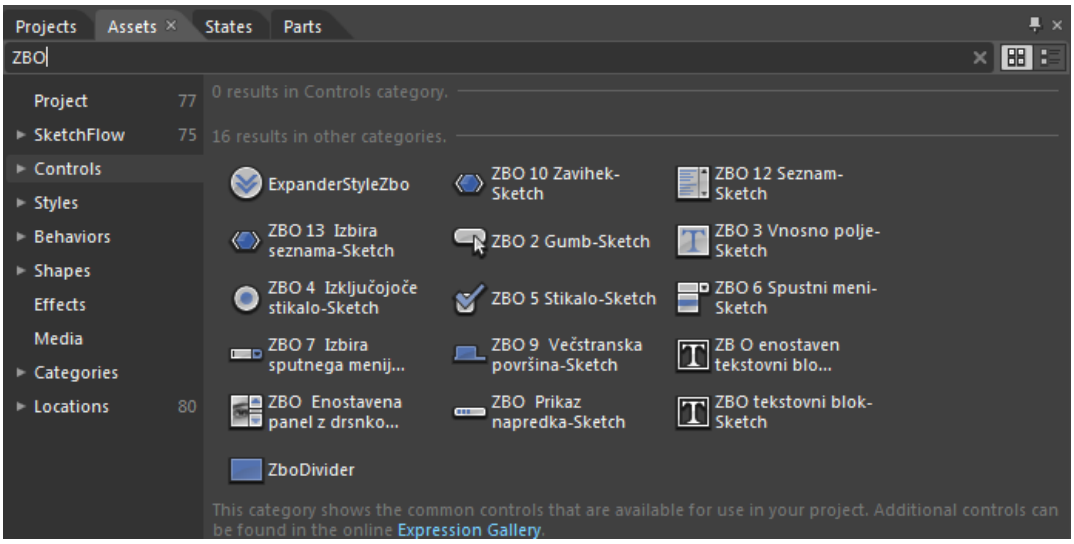
Ob zbiri se nam odpre podmeni, kjer se nahajajo 3 možnosti urejanja:

- uredi trenutni element – Edit Current,
- kopiraj trenutni element in ga preuredi – Edit Copy,
- in ustvari prazno predlogo – Empty Copy.

Prva izbira nam je na voljo, kadar delamo s kontrolami, ki smo jih že pred tem ustvarili sami. Ob izbiri drugih dveh možnosti pa se nam odpre modalno okno, v katerem nastavimo ime stila in izberemo mesto, kamor bomo shranili svoj stil. Ker želimo imeti slog kontrolnika shranjenega v datoteki, ga bomo shranili v datoteko SketchStyles.xaml, kjer so shranjeni vsi stili naših kontrol. V nasprotnem primeru bi lahko ustvarili novo datoteko, v kateri bi se nahajal naš stil kontrole.



Slika 6.29: Preimenovanje kontrole.



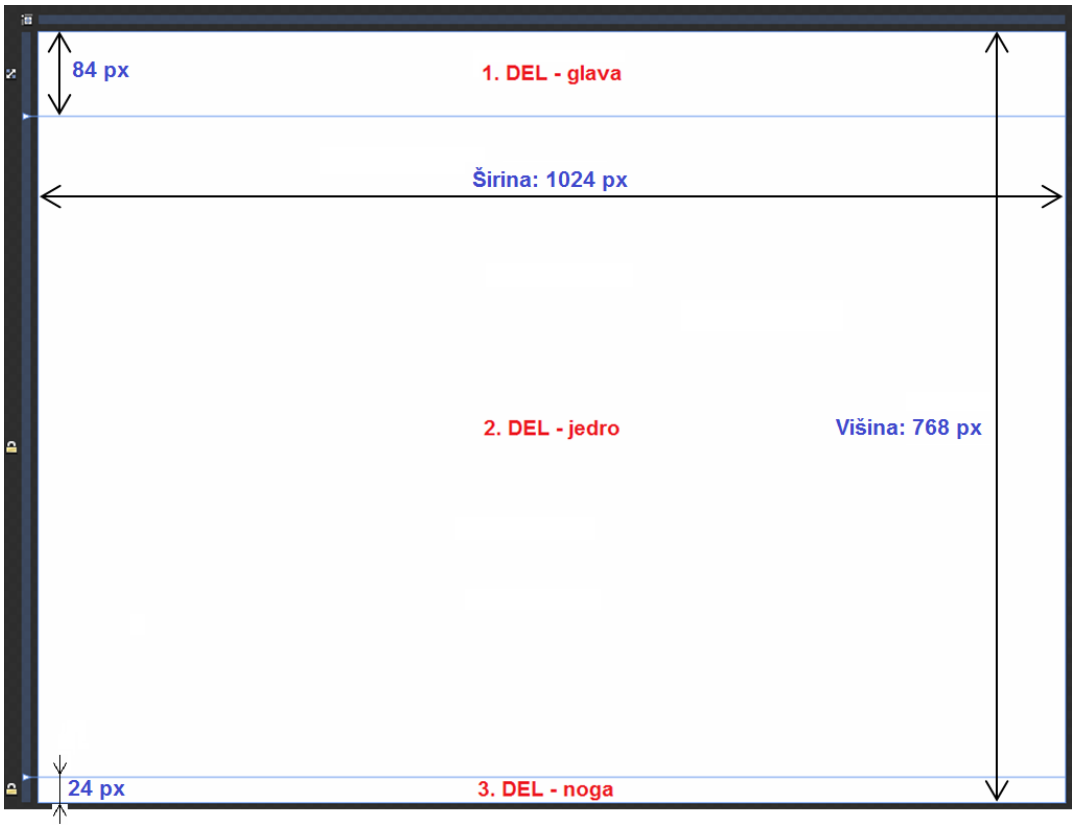
Slika 6.30: Viri, ki so bili preimenovani.

6.6 Zaklopljene črte

Vsak prototipi je sestavljen iz treh delov, ločenih z zaklopljenimi črtami. Zaklopjene črte ustvarimo enostavno s klikom na moder trak, ki se nahaja ob zaslonu in jih lahko enostavno premikamo po širini ali višini zaslona s klikom na zaklopljeno linijo. Zaklopjene linije predstavljajo mrežo, ki jo lahko ustvarimo tudi ročno v urejevalniku XAML kode ali pa s kontrolo Grid.

6.7 Struktura prototipa

Prvi del prototipa predstavlja glava aplikacije, osrednji – glavni del predstavlja jedro prototipa, tretji del pa predstavlja noga, v kateri se nahaja datum in čas ter ikona za dostop do elektronske pošte.



Slika 6.31: Osnovna zgradba za slonov na katerih bomo razvijali prototipe.

Glava je visoka 84 slikovnih pik (ang. pixles - px) in je raztegnjena čez celoten zaslon. Noga je visoka 24 slikovnih pik in se prav tako razteza čez celotno širino zaslona. Preostali del zaslona je namenjen prototipiranju uporabniških vmesnikov registra komitentov. Mrežo lahko ustvarimo tudi ročno preko urejevalnika kode, kjer:

- `<Grid x:Name="Ime" Background="White">`

- predstavlja začetek mreže z lastnostmi, ki pomenijo:

  1. `x:Name = "Ime mreže"`

     ,

  2. `Background="Barva ozadja"`

     .

- `<Grid.RowDefinition>`

  - značka, ki predstavlja začetek definicije mreže.

- `<RowDefinition MinHeight="84">`

  - predstavlja vrstico, katere minimalna višina meri 84 slikovnih pik.

- `</Grid.RowDefinitions>` in
  `</Grid>`

  pa predstavljajo končne značke oziroma konec elementa.

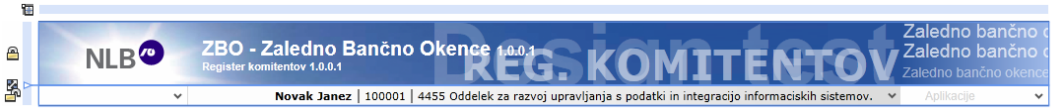
```
23
24      <Grid x:Name="LayoutRoot" Background="White">
25          <Grid.RowDefinitions>
26              <RowDefinition MinHeight="84"/>
27              <RowDefinition Height="659" />
28              <RowDefinition Height="24"/>
29          </Grid.RowDefinitions>
30      </Grid>
31  </UserControl>
```

Slika 6.32: Definicija mreže.

Ročno pisanje kode je boljše od generatorja, saj imamo s tem čistejšo kodo. Poleg tega pa poteka kodiranje ponavljajočih struktur hitreje, če prekopiramo izvorno kodo, kakor pa če se lotimo kopiranja objektov. Tak način je navsezadnje za izkušenega oblikovalca ali razvijalca, ki obvlada XAML kodo hitrejši in omogoča hitrejši razvoj prototipov.

6.7.1 Kontrola HaderControl

Ker bo del vsakega prototipa sestavljena iz enakega dela, bomo najprej ustvarili glavo in nogo. Glavo in nogo bomo sestavili iz virov, ki smo jih uvozili v projekt preko dinamične povezovalne knjižnice NLB.SL.Theme. Oblika glave in noge je sestavljena iz XAML kode. Ogrodje glave je sestavljeno iz mreže (ang. Grid), v katero je vpet logotip z napisom, slikami ter animacijami. Drugi del glave je uporabniški del, ki je prav tako sestavljen iz mreže, ki vsebuje spustni meni in oznake s podatki o uporabniku.



Slika 6.33: Glava prototipa.

6.7.2 Kontrola FooterControl

Pri nogi imamo enako strukturo, le da je ta sestavljena samo iz enega okvirja oziroma mreže, v katero so vpeti "stack" paneli z oznakami in ikonami, ki so programsko nadzorovane. Ker je zgradba glave in noge zelo kompleksna, je za oblikovanje in delo potrebno dobro poznati XAML kodo.

Slika 6.34: Noga prototipa.

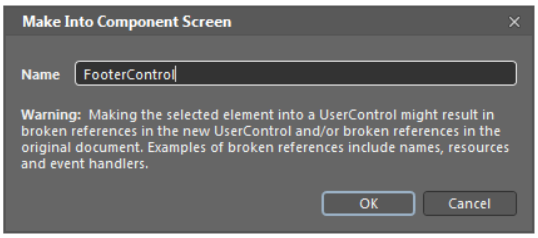
6.8 Komponentno okno (Component screens)

Komponentno okno je množica elementov, ki jih lahko uporabimo na več zaslonih. Uporabljamo jih, ker nam olajšajo načrtovanje prototipov, ki po-

navadi vsebujejo del zaslona s komponentami, ki so na vseh zaslonih enake. V SketchFlow je Komponentno okno uporabno orodje, brez katerega bi bilo potrebno vedno znova in znova ustvarjati enake elemente skozi vse prototipe.

6.8.1 Kreiranje komponentnega okna

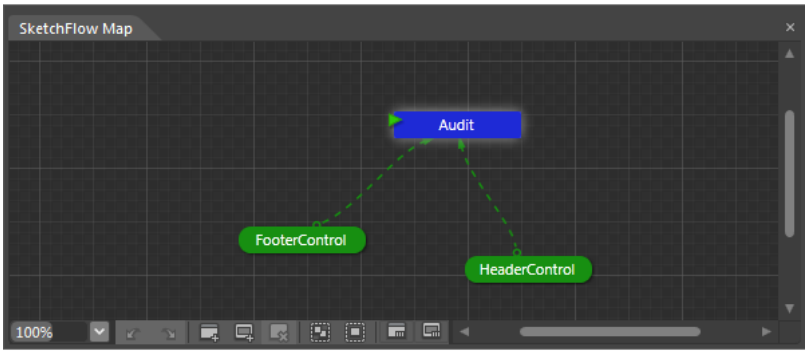
Ko smo končali z oblikovanjem glave in noge, jih je potrebno še pretvorili v SketchFlow komponentno okno (ang. SketchFlow component screen). To naredimo lahko naredimo na dva načina. Izberemo skupino objektov, ki jih bomo pretvorili in z desnim miškinim gumbom miške kliknemo na označene elemente ter izberemo možnost "Make into Component Screen", ki del zaslona pretvori v komponentno okno. Ob izbiri se pojavi panel, v katerem bomo nogo, ki jo želimo pretvoriti v komponentno okno, poimenovali "FooterControl", glavo pa v "HeaderControl".



Slika 6.35: V modalnem oknu lahko poimenujemo komponentno okno.

Po potrditvi se v panelu Projects in na SketchFlow zemljevidu pojavi novo okno v obliki zelenega ovalnega polja, ki predstavlja našo nogo, ki je povezana z MainScreen zaslonom.

Ustvarjeni komponentni okni HeaderControl in FooterControl se po prevajanju projekta nahajata tudi v virih oziroma med kontrolami, kjer jih lahko po principu "potegni & spusti" povlečemo na zaslon in uporabljamo kot druge standardne objekte oziroma elemente s panela, ki vsebuje kontrole (virov).



Slika 6.36: Zelene črte predstavljajo povezave med zasloni in komponentnim oknom.

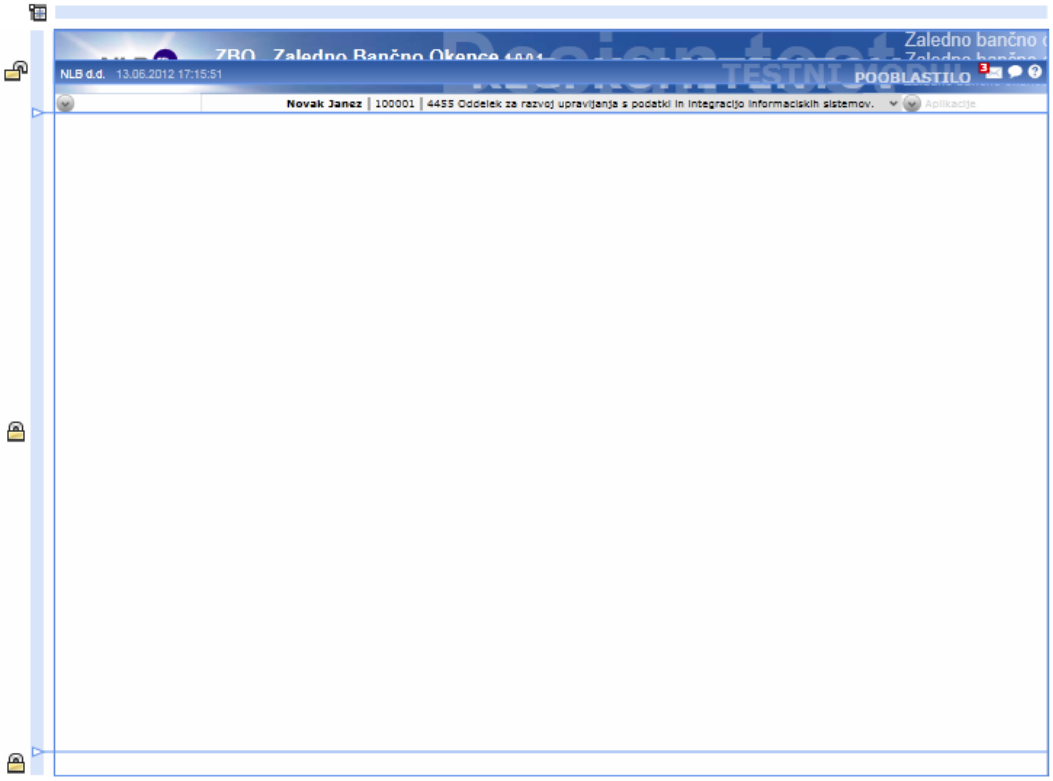
6.8.2 Dodajanje komponentnih oken v prototipe

Komponentna okna lahko dodajamo na več načinov. Prvi in hkrati najbolj enostaven način dodajanja poteka preko SketchFlow zemljevida. Če zeleno polja povežemo z zaslonom oziroma prototipom, se na njem pojavi skupina objektov v obliki kontrole. Drugi način, ki smo ga omenili v prejšnjem podnaslovu, omogoča, da iz panelov Projects na prototip potegnemo komponentno okno ali pa ga preko panela Kontrol (virov) dodamo na zaslon.

Komponentno okno lahko dodamo na zaslon tudi preko urejevalnika XAML kode, v katerem takoj za značko, ki označuje konec definicije mreže (ang. grid), vpišemo naslednjo kodo:

Komponentno okno je potrebno nato še umestiti v ogrodje prototipa MainScren.xaml in Audit.xaml. Ker imamo prototipe v začetnih fazah razdeljene na 3 vrstice (dele), jim je potrebno določiti lego. To lahko nastavimo kar v urejevalniku XAML kode, kjer v značko, s katero kličemo komponentno okno za glavo HeaderControl dodamo ukaz Grid.Row=”0”, za komponento FooterControl pa nastavimo vrednost Grid.Row=”2.

To lahko storimo tudi preko Lastnosti (ang. Properties), na katerem se za ta del nastavitev nahaja panel Layout. V panelu se nahaja polje Row in Column, kjer lahko nastavimo pozicijo vsake kontrole na prototipu. Pri tem je potrebno upoštevati, da se vse vrstice in stolpci v mreži prototipov začnejo



Slika 6.37: Komponenta okna povezana z zaslonom preko SketchFlow zemljevida.

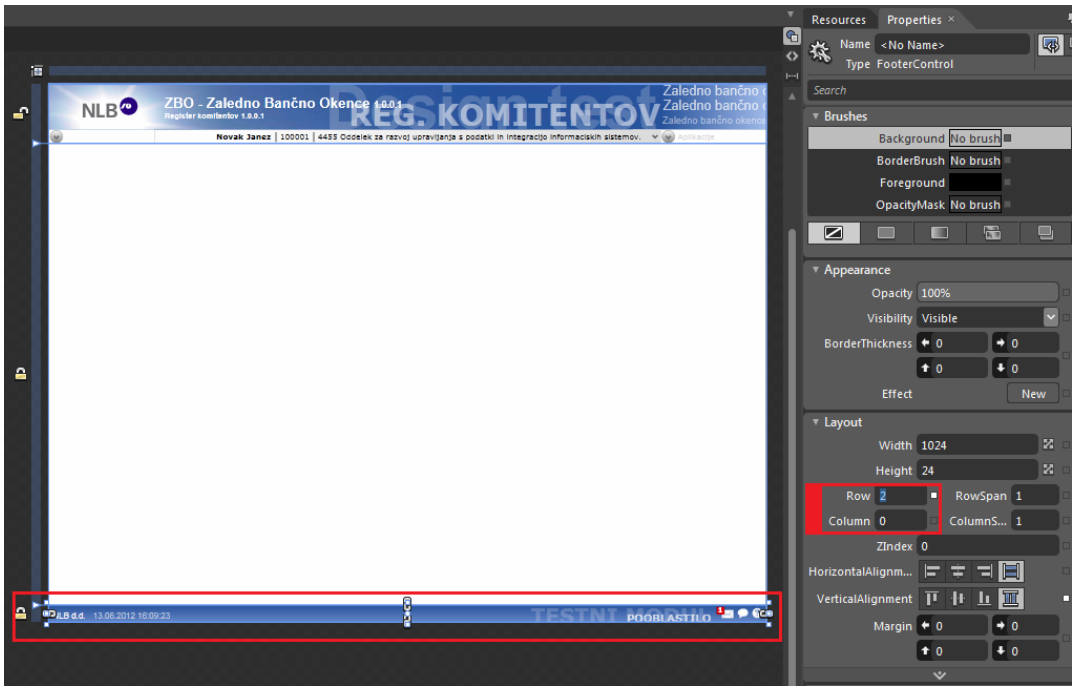
```
        </Grid.RowDefinitions>
        <local:HeaderControl /> <!-- komponentno okno - glava -->
        <local:FooterControl /> <!-- komponentno okno - noga -->
    </Grid>
```

Slika 6.38: Dodajanje komponentnih oken preko urejevalnika XAML kode.

```
    </Grid.RowDefinitions>
    <local:HeaderControl Grid.Row="0" VerticalAlignment="Stretch" /> <!-- komponentno okno - glava -->
    <local:FooterControl Grid.Row="2" VerticalAlignment="Stretch"  /> <!-- komponentno okno - noga -->
</Grid>
```

Slika 6.39: Postavitev kontrol na mrežo.

z 0.



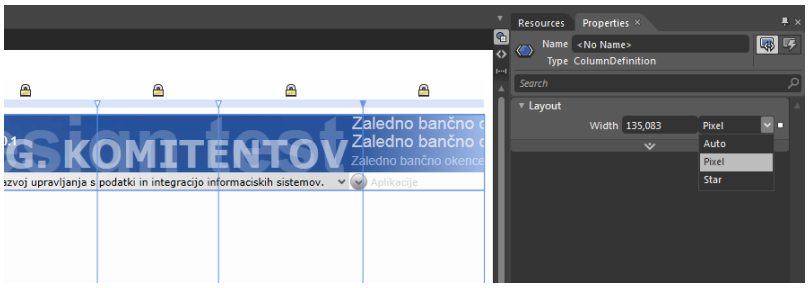
Slika 6.40: Pozicioniranje komponente FooterControl preko panela Properties.

6.9 Dodajanje kontrol na prototip

Naša naslednja naloga bo postavitev gumbov na MainScrean.xaml, preko katerih se bomo sprehajali med prototipi zaslonov. Po uspešni postavitvi komponentnih oken za naše vmesnike razdelimo glavni del (drugo vrstico - jedro) prototipa na simetrična polja, v katere bomo vstavili 9 gumbov, ki nam bodo služili za navigacijo. Jedro razdelimo z zaklopljenimi črtami, ki jih ustvarimo s klikom na trak ob robu prototipa in jim v panelu z lastnostmi natančno določimo njihovo pozicijo. Pri tem moramo upoštevati, da bomo zaradi boljše preglednosti in hitrejše interakcije uporabljali gumbe velikosti 161 x 80 slikovnih pik. Najprej se lotimo pozicionirati zaklopljene črte za

stolpce. Ker je v Expression Blend 4 izhodišče vertikalnih zaklopljenih črt skrajna desna stran prototipa, je potrebno prototip razdeliti z desne proti levi. Ker želimo, da je razmik med gumbi enakomeren, ga izračunamo po naslednjem principu:
Širina prototipa: 1024 px
Širina gumba: 161 px
Št. gumbov v eni vrsti: 3
Razmik med gumbi = 1024 px – 3 * 161 px = 135,083 px

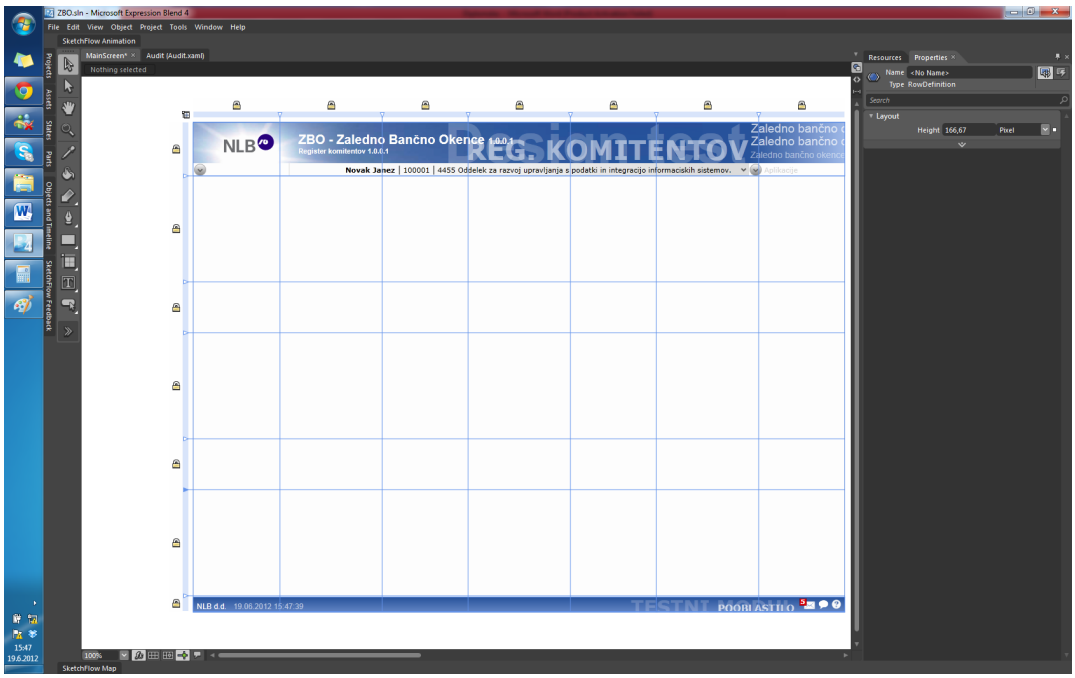
Izračunamo, da je razmik med stolpci oziroma gumbi približno 135 slikovnih pik. Razmik je potrebno vnesti za vsako zaklopljeno črto posebej. To storimo tako, da najprej kliknemo na zaklopljeno črto, nato pa v panelu Lastnosti pod oznako Width nastavimo širino na 135.083 ter za enoto izberemo px - pixles (slika 6.41). Naslednjo linijo zopet po istem principu označimo in se pomaknemo v panel lastnosti in vanjo vnesemo širino gumba (161 px), saj se bo v tem polju nahajal gumb.



Slika 6.41: Pozicioniranje zaklopljenih črt in izbira merske enote.

Ko končamo s postavitvijo stolpcev, definiramo še zaklopljene linije po vertikalni smeri. Gumbe bomo postavili v 2 vrstici višine 20 slikovnih pik. Ker želimo, da so gumbi enakomerno razporejeni tudi v vertikalni liniji, je potrebno izračunati razmik med vrsticami.
Višina prototipa: 768 px
Višina glave: 84 px

Višina noge: 24 px
Višina gumbov: 80 px
Št. vrstic z gumbi: 2
Prazen prostor = 768 px - 2 * 80 px – 84 px – 24 px = 500 px
Ker imamo 2 vrstici gumbov, je potrebno 500 px razdeliti na 3 enakomerne dele, ki predstavljajo razmik med gumbi.
Razmik med gumbi = 500 px / 3 = 166,666 px.
Postopek postavitve je enak kot v zgornjem primeru, le da tukaj postavljamo zaklopljene črte v vodoravni liniji. Črte je potrebno definirati od spodaj navzgor, da so razmiki pravilni. Označimo prvo zaklopljeno črto in lastnostih nastavimo višino (angl. Height) na 161.67 px. Naslednji črti nastavimo višino 20 px, nato pa se ves postopek ponavlja do konca.



Slika 6.42: Mreža, na katero bomo postavili navigacijske gumbe.

Mrežo, ki jo prikazuje slika 6.42, lahko ustvarimo tudi v XAML pregledovalniku kode. Koda za mrežo, ki smo jo ustvarili, zgleda takole:

```
<Grid x:Name="LayoutRoot" Background="White">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="0.295*"/>
        <ColumnDefinition Width="161"/>
        <ColumnDefinition Width="0.219*"/>
        <ColumnDefinition Width="160"/>
        <ColumnDefinition Width="0.221*"/>
        <ColumnDefinition Width="160"/>
        <ColumnDefinition Width="0.265*"/>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="85"/>
        <RowDefinition Height="166.67"/>
        <RowDefinition Height="80"/>
        <RowDefinition Height="166.67"/>
        <RowDefinition Height="80"/>
        <RowDefinition Height="166.67"/>
        <RowDefinition Height="24"/>
    </Grid.RowDefinitions>
    <local:HeaderControl Grid.ColumnSpan="7" Grid.Row="0" Height="Auto" VerticalAlignment="Stretch" Width="Auto" d:IsPrototypingComposition="True"/>
    <local:FooterControl Grid.ColumnSpan="7" VerticalAlignment="Stretch" Width="Auto" d:IsPrototypingComposition="True" Margin="0,2,0,-2" Grid.Row="6"/>
</Grid>
```

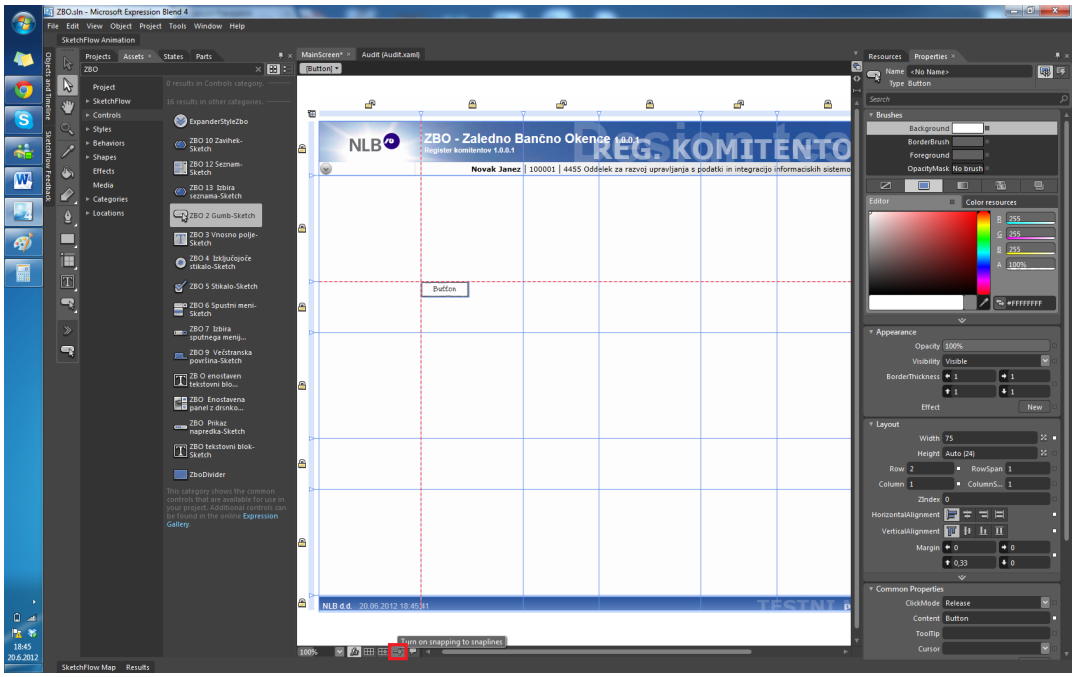
Slika 6.43: Avtomatsko generirana XAML koda za mrežo.

Dodajanje gumbov na mrežo prototipa

Po uspešno izvedenem koraku, v katerem smo si ustvarili podlago za postavitev gumbov, je potrebno na prototip dodati le še gumbe, ki bo prekrivala celotno površino posameznega polja (161 x 80 px). Najprej se odpravimo v panel Kontrol (Virov) in v iskalnik napišemo "ZBO". Iskalnik nam v panelu izpiše vse kontrole, ki vsebujejo iskani niz. V panelu izberemo kontrolo ZBO 2 Gumb – Sketch in ga potegnemo na panel. V primeru, da imamo v SketchFlow zemljevidu vključeno funkcijo "Snaping to snaplines", nam bo program ponudil poravnavo elementa na mrežo, ki smo jo ustvarili (slika 6.43). Gumb lahko pozicioniramo tudi na drugi način preko panela lastnosti, kar bomo sedaj tudi naredili. Ker je pred vsakim nastavljanjem lastnosti v panelu lastnosti potrebno označiti objekt, ki ga urejamo, označimo gumb in se odpravimo v panel Lastnosti, v katerem bomo gumb umestili na mrežo, mu določili velikost, vsebino in drugo.

6.10 Gumbi in navigacije

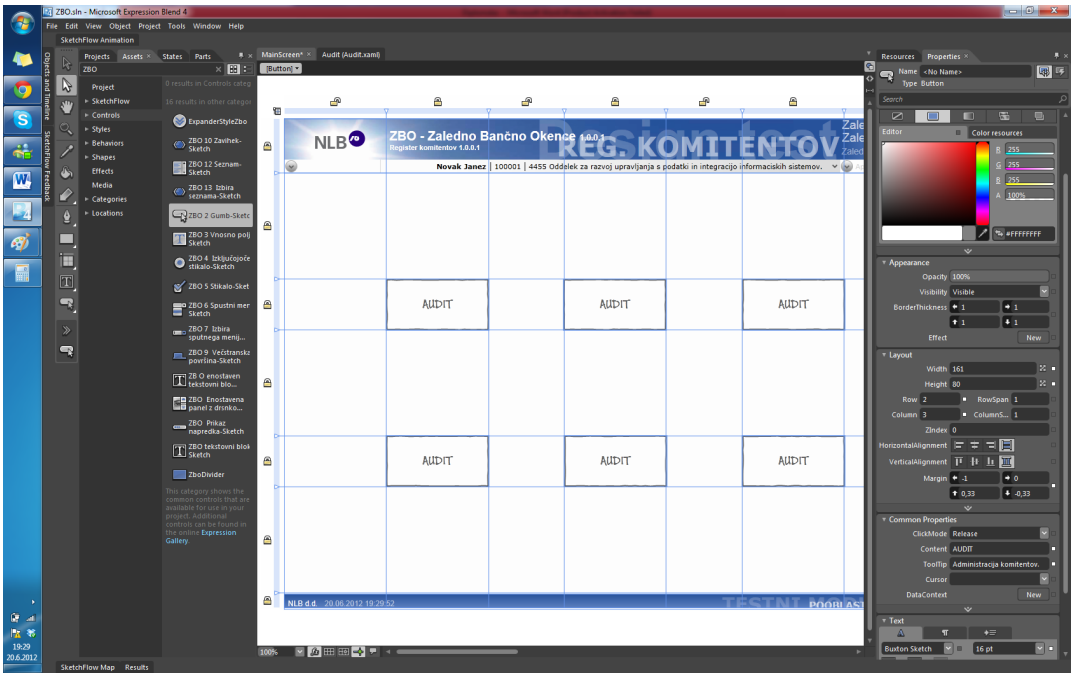
Med prototipi zaslonov se lahko premikamo preko gumbov, ki vsebujejo navigacijo za prehajanje med zasloni. To je zelo uporabna funkcija, ki uporabniku omogoča, da brez posebnega znanja programiranja med seboj poveže zaslone.



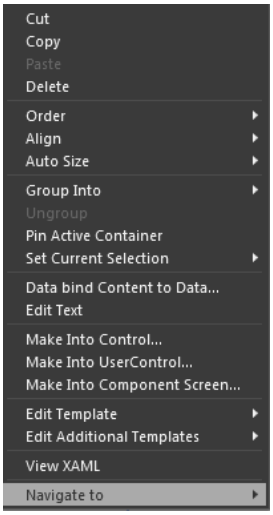
Slika 6.44: Postavitev gumba na rob mreže z funkcijo snapping to snapline.

V nasprotnem primeru bi moral načrtovalec prototipov implementirati reference na druge zaslone z uporabo programske kode, ki se izvaja v ozadju in izvaja tudi delovanje poslovne logike. Expression Blend 4 omogoča, da lahko obstoječi zaslon povežemo s katerimkoli zaslonom, ki se nahaja v našem projektu brez predhodnega znanja programiranja. Gumb AUDIT povežemo tako, da ga označimo in nanj kliknemo z desnim gumbom miške. Odpre se nam dialog, v katerem se v kaskadnem seznamu pomaknemo na element "Navigate to".

Razširi se nam novo okno, v katerem se nahajajo vsi zasloni, ki so v SketcFlow zemljevidu povezani z MainScreen.xaml zaslon. V oknu izberemo Audit oziroma zaslon, na katerega želimo, da kaže povezava in v tem trenutku je navigacija na zaslon ustvarjena (slika 6.46).



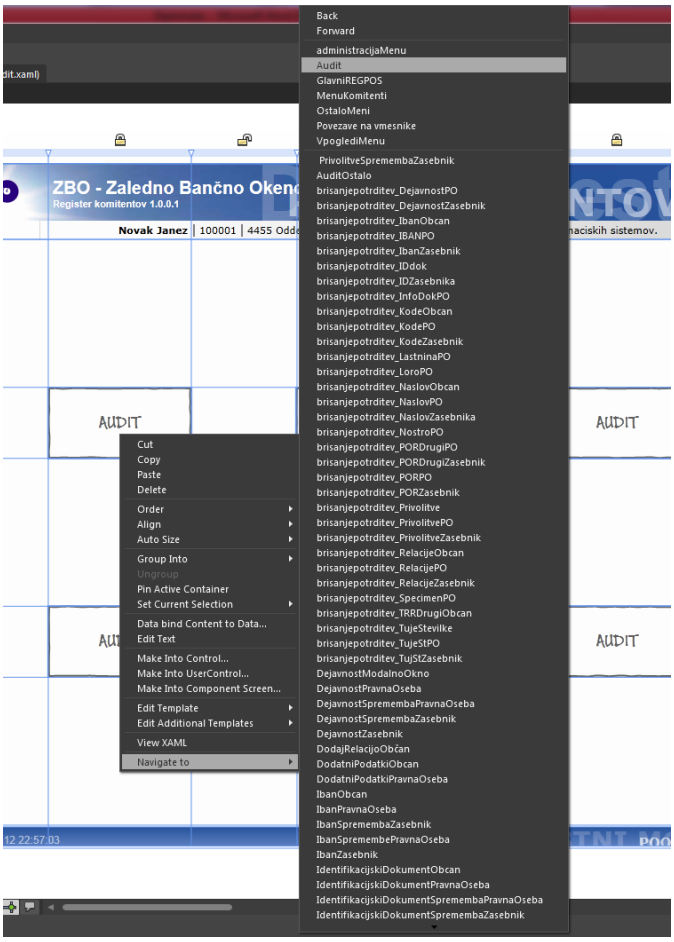
Slika 6.45: Končna oblika začetnega zaslona v Expression Blend 4.



Slika 6.46: S funkcijo Navigate to povežemo dva zaslona med seboj.

6.11 Izdelava vmesnika Audit

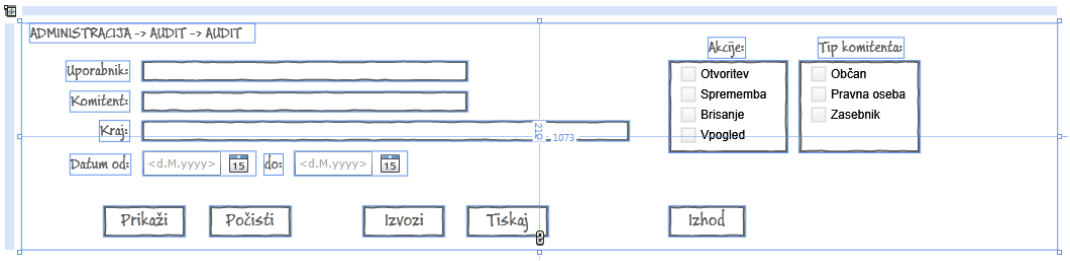
Zaslon Audit je namenjen administraciji Registra komitentov. Glavna naloga vmesnika Audit je omogočiti uporabniku podrobni vpogled v zgodovino upo-



Slika 6.47: Povezava zaslona MainScreen.xaml z Audit.xaml.

rabe Registra komitentov. Zaslon informacije posreduje v obliki tabele oziroma seznama, ki ga lahko uporabniki s pomočjo iskalnih kriterijev poljubno prirejajo svojim potrebam. Zgradba vmesnika Vmesnik Audit je sestavljen iz štirih delov. Prvi in zadnji del sestavljata glava in noga, ki smo ju predstavili na začetku poglavja, zato ju ne bomo omenjali. Drugi del vmesnika sestavlja panel, ki je namenjen iskalnim kriterije, medtem ko tretji del vmesnika predstavlja tabela oziroma seznam, v katerem se nahajajo podrobne informacije o uporabi Registra komitentov.

Prvi del sestavljajo kontrole, preko katerih definiramo iskalne parametre,



Slika 6.48: Prvi (kontrolni) del zgradbe prototipa Audit.xaml

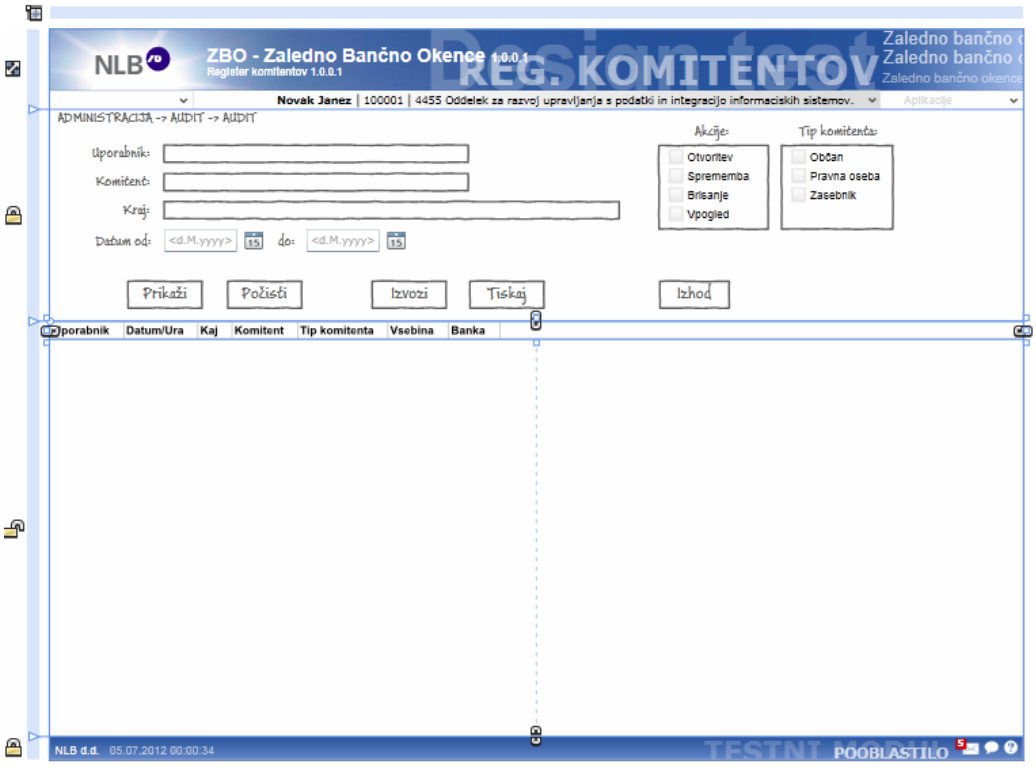
Zap. št.	Polje
1	uporabnik
2	datum /ura
3	kaj
4	komitent
5	tip komitenta
6	vsebina
7	banka

Tabela 6.1: Tabela polj za izpis Audit.

ki uporabniku omogočajo filtriranje vsebine. Vsebino lahko filtriramo po uporabniku, komitentu, tipu komitenta, kraju ter datumu in akcijah, ki so se izvajale nad komitenti. Za iskanje, brisanje, izvažanje in tiskanje rezultatov uporabimo gumbe, ki poskrbijo za ustrezne akcije. Akcije lahko realiziramo s pomočjo programske kode ali animacijami, stanji in vzorčnimi podatki. Tretji del zaslona je sestavljen iz dveh tabel, s katerimi ponazorimo prikaz rezultatov. Na začetku je tabela prazna ob kliku na gumb Prikaži/Počisti pa prikaže/počisti zaslon s podatki oziroma stanje zaslona. To realiziramo s stanji, s katerimi prvo (prazno) tabelo prekrije druga tabela, ki vsebuje vzorčne podatke.

Tabeli implementiramo s kontrolo DataGrid. Prva tabela ne vsebuje podatkov in se nahaja na vmesniku (slika 6.49). Drugo tabelo napolnimo z

vzorčnimi podatki, kot se nahajajo v tabeli. Ko kliknemo na gumb prikaži, se tabela s podatki postavi pred prazno tabelo, ki se že nahaja na zaslonu. S funkcijo State prikažemo vmesnik interaktiven in dinamičen. S stanji realiziramo tudi druge funkcije, kot so počisti, izvozi in tiskaj.

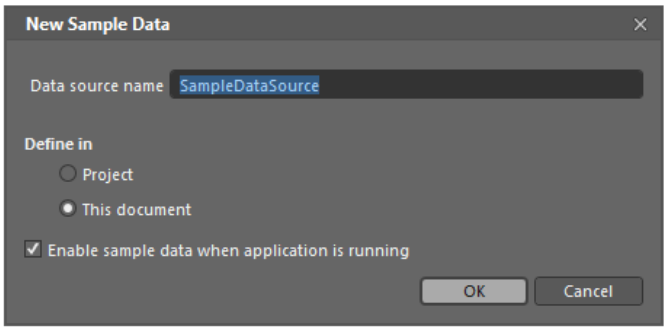


Slika 6.49: Ko smo uspešno ustvarili tabelo, bi moral naš prototip izgledati nekako takole.

Ker razvijamo prototip, ki bo predstavljal približek končne aplikacije, bomo v prototip vključili stanja zaslonov in vzorčne podatke. Te bomo v nadaljevanju predstavili na konkretnem primeru.

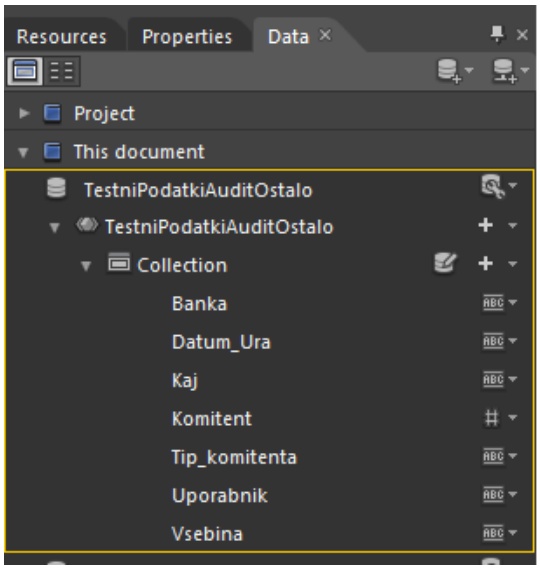
6.12 Vzorčni podatki in primerjava s podatkovno bazo

V Expression Blend 4 imamo na voljo funkcijo, ki nam omogoča vzorce podatkov. Vzorce podatkov uporabimo v prototipih, kjer jih lahko uporabimo za izdelavo različnih seznamov. Podatkovni vzorci se nahajajo na panelu Data. Kreiranje vzorčnih podatkov Vzorčne podatke lahko ustvarimo v panelu Podatki(ang. data), v katerem lahko izbiramo med funkcijami, ki omogočajo, da ustvarimo nov vzorec podatkov, uvozimo podatke iz XML dokumenta ali pa ustvarimo vzorec podatkov programsko iz razreda. Ker želimo v tabeli imeti vzorec podatkov, ki se bo ujemal s podatki, ki se bodo nahajali v končni aplikaciji, izberemo možnost New Sample Data, s katero ustvarimo nov vzorec podatkov. Na zaslonu se nam pokaže novo modalno okno, v katerem je potrebno nastaviti ime zbirke ter mesto oziroma lokacijo, kjer bo podatkovna zbirka definirana.



Slika 6.50: Odpiranje novega vzorca podatkov.

Ustvarjeni vzorec se nam pokaže v panelu Data. Vsak podatkovni vzorec lahko hrani več različnih podatkovnih zbirk in atributov. Ob vsakem novem vnosu atributa je slednjemu potrebno določiti ime in podatkovni tip. Na voljo imamo omejeno število podatkovnih tipov. Dolžino in tip podatkov določimo na desni strani vsakega atributa.



Slika 6.51: Struktura vzorca podatkov.

Povezovanje DataGrid kontrol z vzorčnim podatki

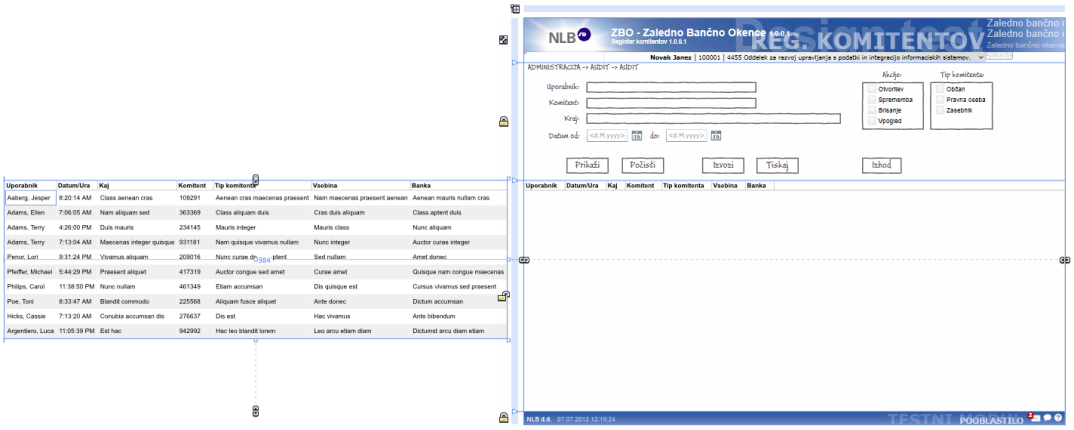
Povezovanje kontrol z vzorcem podatkov je enostavno. Podatkovne vzorce dodajamo v tabel z enostavno funkcijo potegni in spusti. Ob povezavi kontrole DataGrid z zbirko se avtomatsko ustvari struktura tabele s podatki, ki jih ni potrebno bistveno preurejati.

Uporabnik	Datum/Ura	Kaj	Komitent	Tip komitenta	Vsebina	Banka
Aaberg, Jesper	8:20:14 AM	Class aenean cras	109291	Aenean cras maecenas praesent	Nam maecenas praesent aenean	Aenean mauris nullam cras
Adams, Ellen	7:06:05 AM	Nam aliquam sed	363369	Class aliquam duis	Cras duis aliquam	Class aptent duis
Adams, Terry	4:26:00 PM	Duis mauris	234145	Mauris integer	Mauris class	Nunc aliquam
Adams, Terry	7:13:04 AM	Maecenas integer quisque	931181	Nam quisque vivamus nullam	Nunc integer	Auctor curae integer
Penor, Lori	9:31:24 PM	Vivamus aliquam	209016	Nunc curae do 384 iptent	Sed nullam	Amet donec
Pfeiffer, Michael	5:44:29 PM	Praesent aliquet	417319	Auctor congue sed amet	Curae amet	Quisque nam congue maecenas
Philips, Carol	11:38:50 PM	Nunc nullam	461349	Etiam accumsan	Dis quisque est	Cursus vivamus sed praesent
Poe, Toni	8:33:47 AM	Blandit commodo	225568	Aliquam fusce aliquet	Ante donec	Dictum accumsan
Hicks, Cassie	7:13:20 AM	Conubia accumsan dis	276637	Dis est	Hac vivamus	Ante bibendum
Argentiero, Luca	11:05:39 PM	Est hac	942992	Hac leo blandit lorem	Leo arcu etiam diam	Dictumst arcu diam etiam

Slika 6.52: Tabela, v kateri se nahajajo podatki iz vzorca podatkov.

Tabelo, ki hrani vzorec podatkov, pomaknemo ob rob prototipa tako, da

se ne vidi na prototipu (slika 6.52). Ko uporabnik klikne gumb Prikaži, bo sprožil zahtevo za prikaz stanja vmesnika, na katerem se bo nahajala tabela s podatki. Tabela se bo pomaknila z roba zaslona na prototip in prekrila prazno tabelo. Ob kliku na gumb Počisti se bo trenutno stanje postavilo nazaj v osnovno obliko.



Slika 6.53: Postavitev tabel v prototipu.

Primerjava obstoječega podatkovnega vzorca s podatkovno bazo Expression Blend 4 podpira različne vzorčne podatkovne tipe [17]:

- String
- Number
- Boolean
- Image

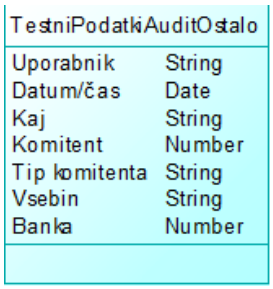
Od tega imamo pri podatkovnem tipu String na voljo različne formate teksta, na podlagi katerih program priredi vsebino. Ti formati so:

- Lorem ipsum
- Address

- Colors
- Company Name
- Date
- Email Address
- Name
- Phone Number
- Price
- Time
- Website URL

Delovanje podatkovnega vzorca se zelo razlikujejo od dejanske podatkovne baze. Če primerja podatkovni vzorec in fizično podatkovno bazo, lahko opazimo razlike že v samem principu uporabe, delovanju kot tudi v podatkovni strukturi. Ne glede na to, da podatkovni vzorci oziroma vzorčne zbirke podatkov v programu Expression Blend nadomeščajo podatkovne baze, je podatkovni vzorec le na videz podoben pravi podatkovni bazi. S tem so želeli doseči enostavno uporabo, hiter razvoj in sorazmerno prilagodljivo strukturo, ki nam omogoča avtomatsko generiranje velike količine podatkov. Podatki v podatkovnem vzorcu se lahko nahajajo v več različnih zbirkah, ki se obnašajo kot tabele. Vsaka zbirka ima lahko poljubno število atributov, število zapisov, ki nam jih generira program, pa lahko ročno spreminjamo. Ker so table med seboj neodvisne in se jih ne da povezovati med seboj, so podatki v tabelah redundantni in ne vsebujejo enoličnih identifikatorjev. Zaradi strukture ne moremo delati poizvedb, prav tako pa je onemogočeno kakršnokoli ažurirati podatkov v podatkovnem vzorcu in posledično izvajanje integritete nad podatki. Ažuriranje podatkov lahko urejamo samo preko panela Data, kjer tudi ustvarimo vzorčni primer podatkov. V zgornji prestavitvi podatkovnih tipov lahko opazimo, da smo poleg ostalih omejitev, omejeni tudi s podatkovnimi

tipi. Enako kot podatkovna baza nam tudi podatkovni vzorec omogoča, da podatke iz vzorca arhiviramo v obliki XML dokumenta, prav tako pa lahko podatke iz vzorca podatkov pridobivamo iz več vmesnikov hkrati. Uporaba vzorca podatkov oziroma zbirke je zelo primerna za uporabo v prototipih, ne pa tudi v dejanskih aplikacijah, kjer se pokažejo potrebe po dnevnem spreminjanju podatkov. Poleg tega pa je tudi prvotni namen vzorčnih podatkov samo pomoč za realizacija realnega stanja aplikacije, ne pa tudi za dejansko uporabo v aplikativnih sistemih.

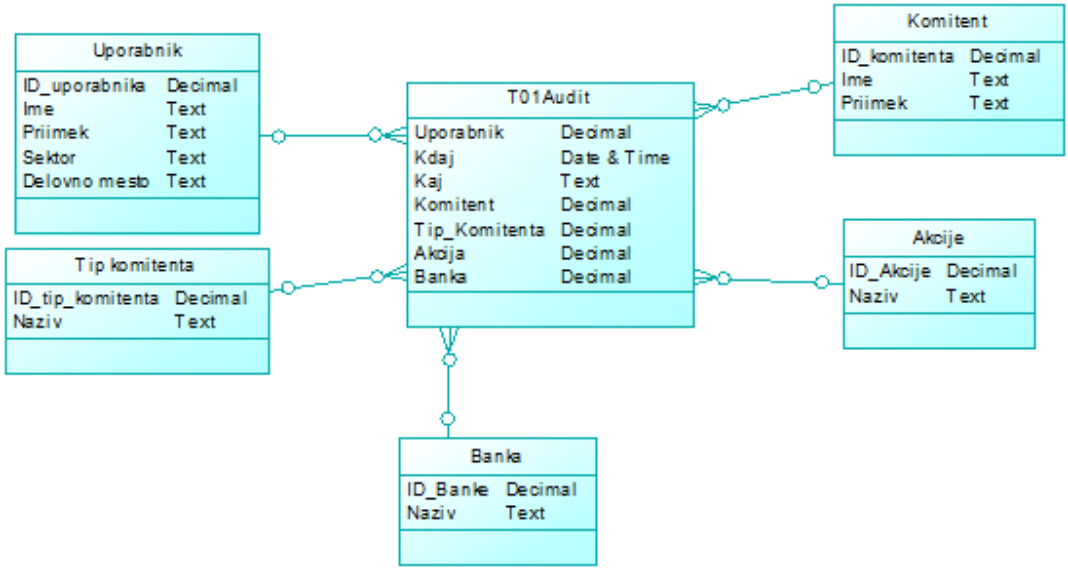


Slika 6.54: Konceptualna shema podatkovnega vzorca.

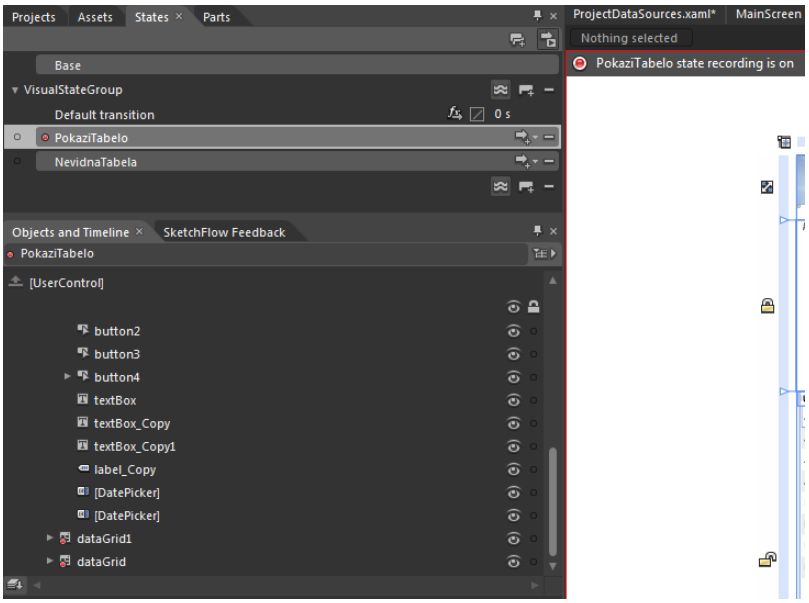
6.13 Stanja vmesnika (State)

Za delo s stanji imamo v programu Expression Blend 4 na voljo panel States (Stanja). Uporabljamo jih, kadar želimo v prototip vključiti različne situacije (stanja). Zato bomo v našem primeru za abstrakcijo realnega stanja aplikacije uporabili States.

Vsak zaslon vsebuje stanja, ki so ločena od stanj drugih zaslonov. V panelu States imamo na začetku samo eno stanje, ki ga Expression Blend 4 poimenuje Base (slo. osnovno) stanje. Ne glede na to lahko enostavno priredimo osnovno stanje našim potrebam in ga posnamemo. Vsako stanje izdelamo na podlagi časovnice, ki se v programu odraža kot posnetek nekega dogodka. Pod osnovnim (ang. Base) stanjem se nahajajo skupine, ki združujejo stanja v skupine. Na sliki 64 lahko opazimo tudi panel Objets



Slika 6.55: Primer realne PB za podatkovni vzorec



Slika 6.56: Panel Stanja (ang. states).

and Timeline, v katerem imamo seznam vseh kontrol in časovnico, s katero določimo čas prehoda kontrole in podobno. Na zgornji levi strani obdeloval-

nega okna se v času snemanja stanja nahaja rdeč gumb, ki nam pove, da Expression Blend 4 snema delovanje našega programa. Ob kliku na gumb se snemanje ugasne in s tem shrani stanje prototipa. Vse kontrole v panelu Objects and Timeline, ki imajo rdečo piko, so vključene v izbrano stanje.

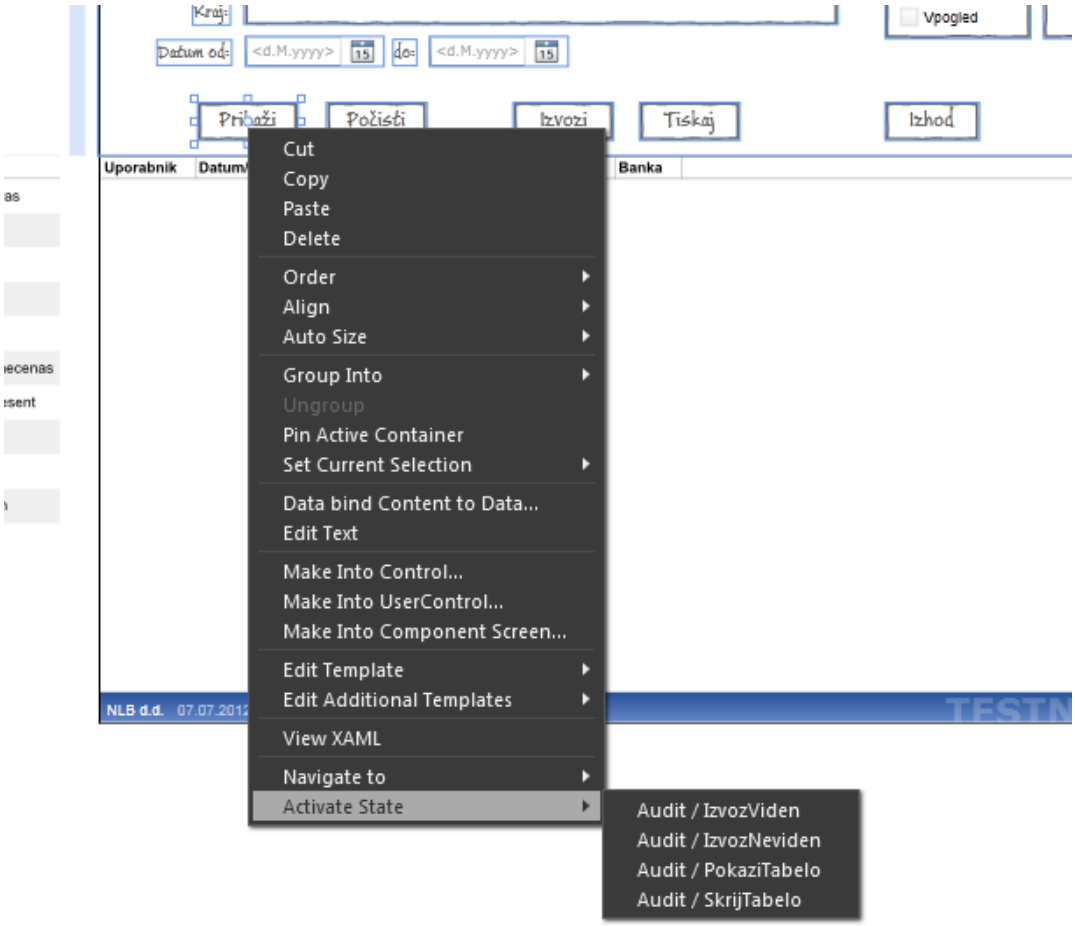
6.13.1 Izdelava posnetka

Ker bomo na zaslonu želeli prikazati tabelo in jo kasneje spet skrili, bomo ustvarili dve stanji. Stanje ustvarimo s klikom na gumb Add State, ki se nahaja v vrstici VisualStateGroup. Ko izberemo Prikazi tabelo, program začne snemati zaslon (slika 6.56). Tabelo, ki vsebuje podatke postavimo na mesto, kjer se bo nahajala v programu in s klikom na rdeč gumb, ki se nahaja v levem zgornjem kotu obdelovalnega okna, ustavimo snemanje. Nato izberemo drugo Stanje, v njem pa tabelo s podatki postavimo izven zaslona. Tako dobimo dva različna stanja zaslona, ki ju bomo uporabili za prikaz funkcionalnosti gumbov Prikaži in Počisti, ki se nahajata na prototipu.

6.13.2 Preklapljanje med stanji

Preklapljanje med stanji je enostavno in ga omogoča funkcija Activate State. Stanje z gumbom povežemo tako, da se postavimo na gumb Prikaži in kliknemo nanj z desnim gumbom miške. Program nam ponudi seznam funkcij, v katerem izberemo Activate State. Odpre se seznam vseh stanj, ki jih imamo na razpolago v prototipu in izberemo enega izmed njih. Prav tako se nato postavimo še na ostale gumbe in jih funkcionalno povežemo s posameznimi stanji.

Ko zaslon uredimo do konca, je potrebno projekt zgraditi in prevesti izvorno kodo v izvršljivo datoteko, ki jo bomo lahko poganjali v brskalnikih.

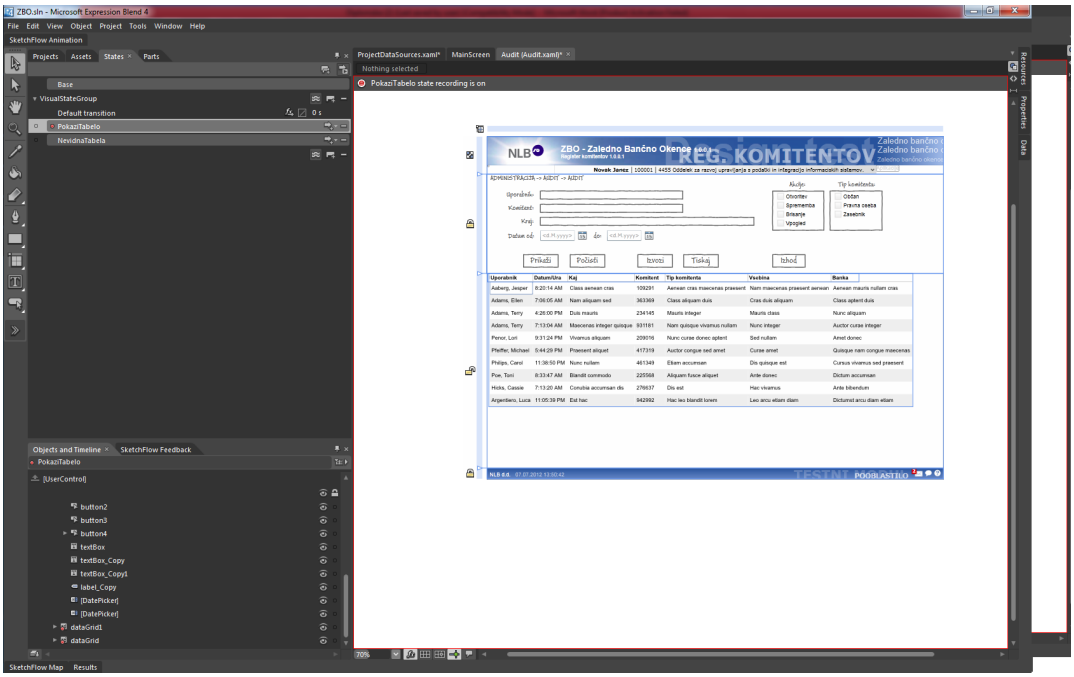


Slika 6.57: Gumb Prikaži povezavo s stanjem "PokaziTabelo".

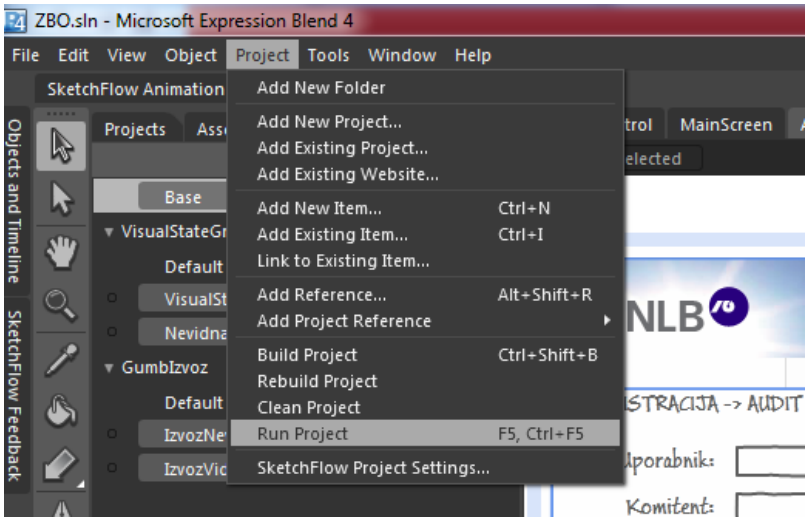
6.14 Prevajanje projekta

Funkcija, ki nam omogoča prevajanje projekta, se nahaja v aplikacijskem meniju Project. Ko se postavimo na element Project, se odpre seznam, na katerem se nahaja funkcija "Run Project", ki prevede in požene projekt. V primeru, da še nismo zgradili oziroma prevedli izvorne kode projekta, lahko to storimo ročno ob izbiri funkcije Build Project, v nasprotnem primeru pa program izpiše napake v aplikacijski kodi.

Ob prevajanju se nam v zavihku Results (rezultati) izpisujejo sporočila,

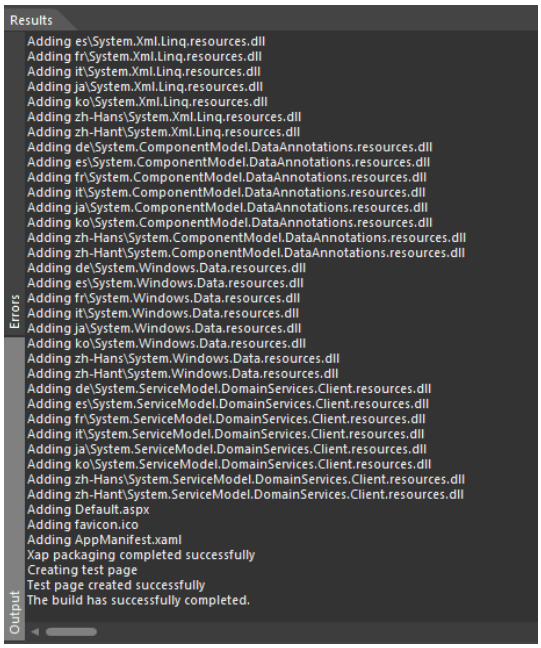


Slika 6.58: Prototip Audit.xaml v stanju "PrikaziTabelo".



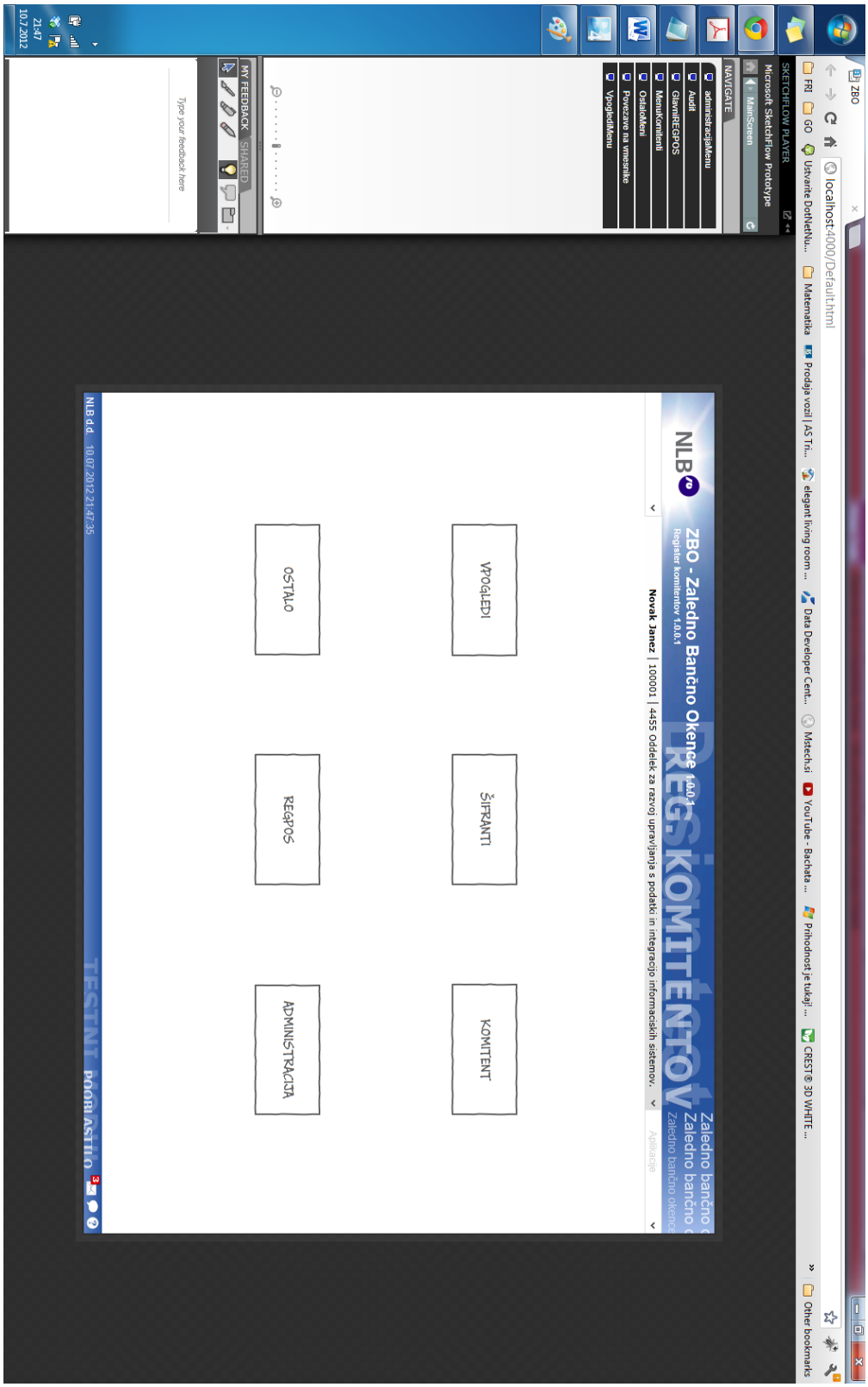
Slika 6.59: Prevajanje izvorne kode.

ki nam povedo, kaj dela prevajalnik. Na koncu nam izstavi informacijo o uspehu prevajanja, v primeru napak pa nam izpiše mesto, vrsto in število vseh napak ter nevarnosti v programski kodi.

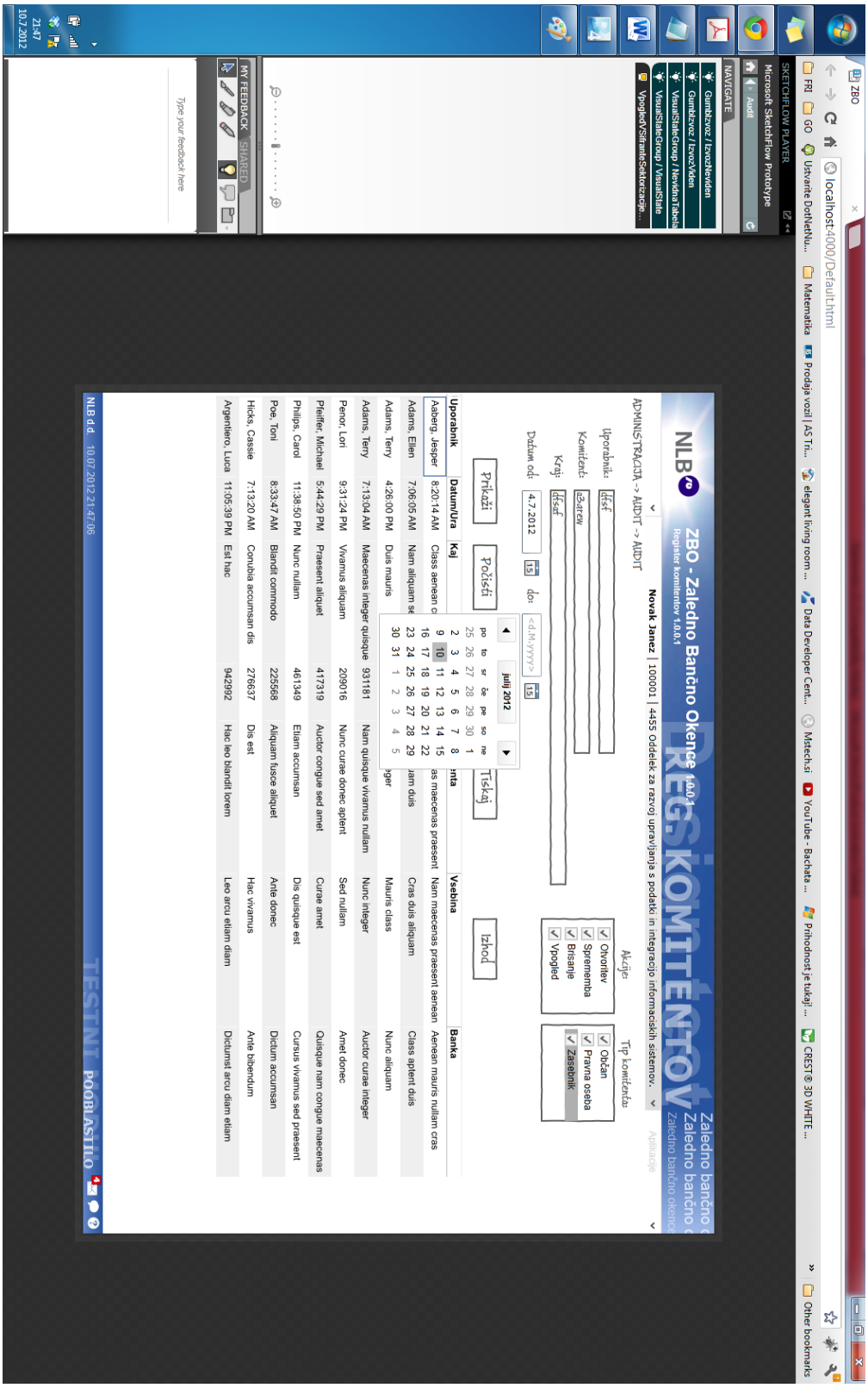


Slika 6.60: Rezultati prevajanja izvorne kode.

Ko zaženemo program, se nam v Start vrstici operacijskega sistema prikaže obvestilo, ki nam sporoča, da je program pognal spletni strežnik, na katerem se bo zagnala aplikacija. Iz naslova spletnega strežnika lahko poganjamo aplikacijo v kateremkoli brskalniku na našem računalniku. Po obvestilu se nam odpre brskalnik, v katerem se nam v Silverlight predvajalniku naloži aplikacijo, kot je vidite na sliki 6.61.



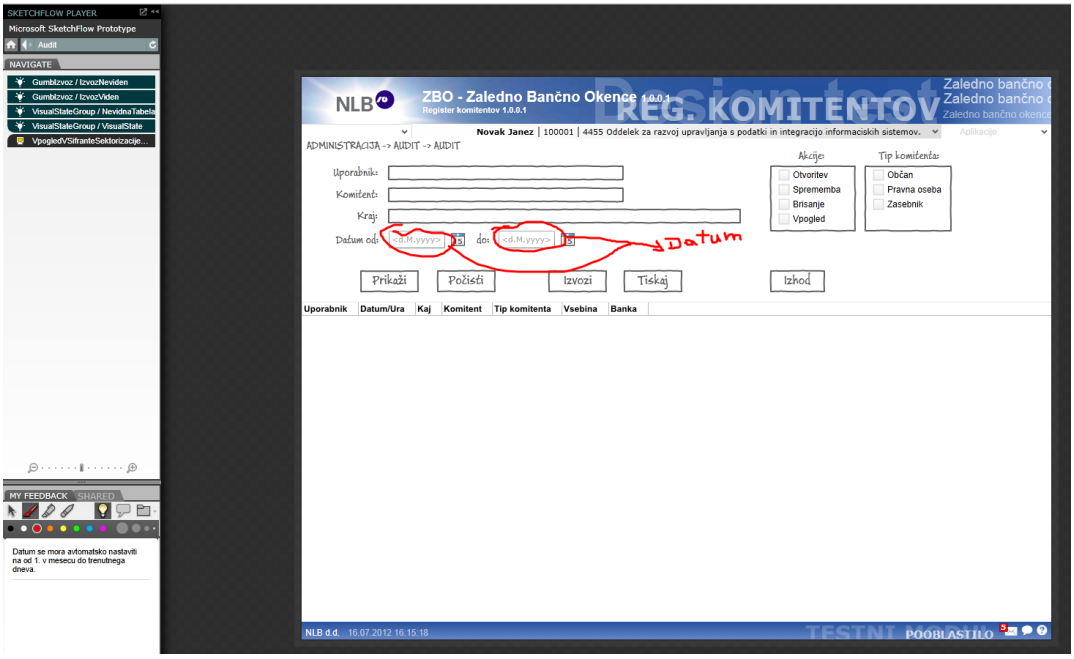
Slika 6.61: Začetno okno prototipa aplikacije Register komitentov.



Slika 6.62: Zaslon audit.xaml v SketchFlow predvajalniku.

6.15 SketchFlow predvajalnik (SketchFlow player)

SketchFlow predvajalnik se uporablja za predstavitev Silverlight prototipov, ki smo jih razvili v Expression Blend 4 ali drugih sorodnih razvojnih orodjih, ki podpirajo razvoj na Silverlight platformi. Leva stran predvajalnika uporabniškega vmesnika je sestavljena iz dveh ključnih področij. Na vrhu se nahaja bližnjica za povezavo do začetnega zaslona poleg nje pa imamo še navigacijski gum, ki nam omogoča pomikanje naprej in nazaj po zaslonih. Takoj pod navigacijskimi gumbi se nahaja panel Navigate, preko katere lahko prehajamo med zasloni, ki so povezani med seboj. Pod panelom imamo povečalo s katerim lahko prilagajamo velikost zaslona našim potrebam. Pod panelom se nahajata še dva dodatna zavihka My Feedback in Shared.



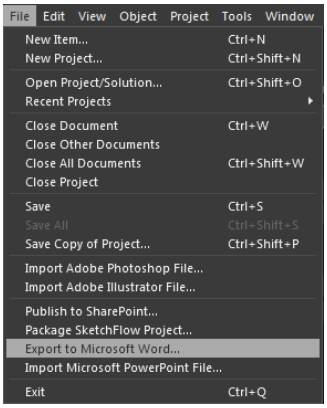
Slika 6.63: Odzivi uporabnikov.

6.16 Odziv uporabnikov (My Feedback)

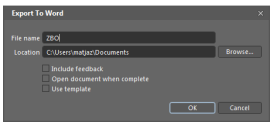
Odziv je namenjen posredovanju komentarjev, ki jih lahko uporabnik vpiše in shrani na strežnik. Iz komentarjev lahko nato razvijalec preko Expression Blend 4 pregleduje in odpravlja pomanjkljivosti o katerih ga je uporabnik obvestil preko uporbniškega vmesnika. V ta namen lahko prototip naložimo na SharePoint spletno mesto, kjer si ga uporabniki ogledujejo in ga testirajo ter sporočajo svoje odzive na izdelan prototip na podlagi katerih lahko nato skozi dodaten razvoj prilagodimo zaslon uporabniškim zahtevam. Primer je na sliki 6.63.

6.17 Generiranje MS Word poročila

Expression Blend 4 ima v svojem integriranem razvojnem okolju tudi možnost generiranja ustvarjanja Microsoft Word poročila, v katerem so zajeti vsi zasloni in zaznamki, ki so jih ustvarili uporabniki ter razvijalci v prototipu. Ta dokument lahko služi kot začetna predloga za produktno dokumentacijo ali hitro poročilo za sestanke, ki so organizirani v domeni razvoja aplikacije. Generiranje dokumenta je zelo enostavno. V aplikacijskem meniju File izberemo izbiro Export to Microsoft Word, s katero generiramo dokument [15]. Odpre se nam okno Export To Word, v katerem nastavimo ime in mesto, kamor naj program shrani dokument, poleg tega pa lahko v poročilo vključimo odzive uporabnikov in uporabimo predlogo.



Slika 6.64: Generiranje MS Word dokumenta za poročilo.



Slika 6.65: Shranjevanje Word dokumenta.

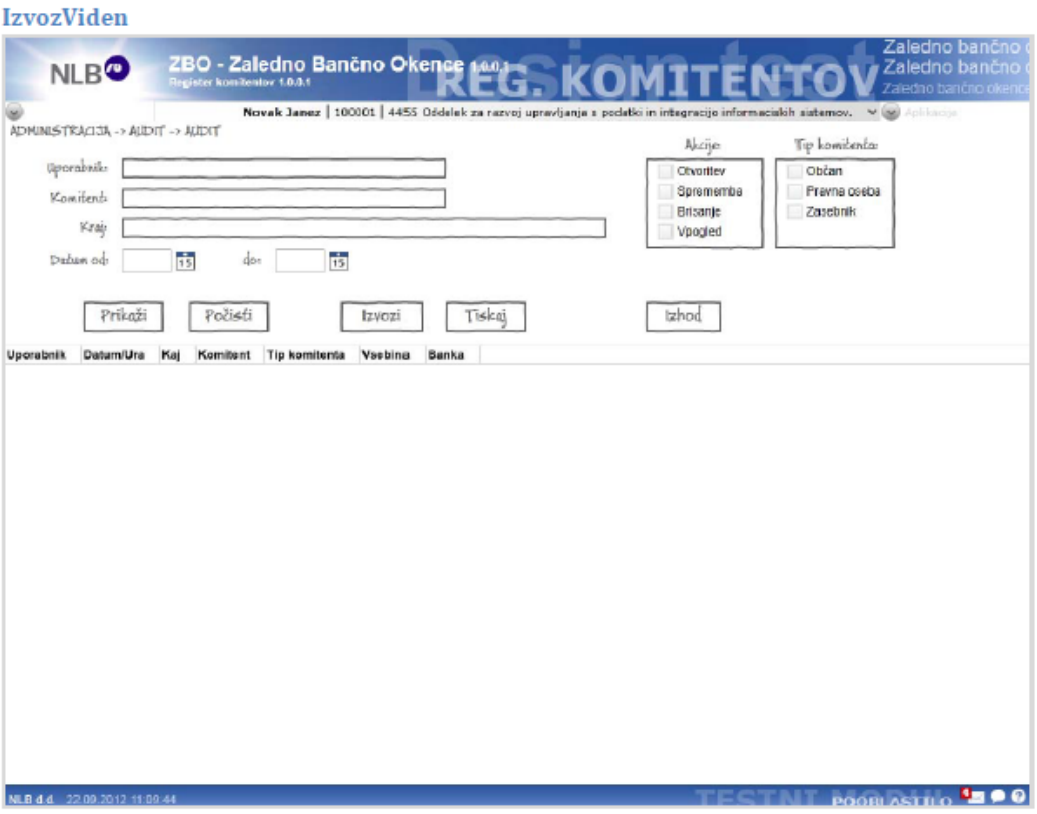


Figure 6: Audit, IzvozViden

Component Screens

FooterControl

Figure 7: FooterControl

HeaderControl

Figure 8: HeaderControl

Orphaned feedback

ZBOScreens.Audit

Feedback from Matjaž Ravbar - 20:28 28.5.2012

Gumb izhod se ne sme nahajati na tem mestu in ga je potrebno prestaviti.

Slika 6.66: Del generiranega poročila v Wordu.

6.18 Časovni okvir razvoja prototipa in število iteracij

1. Časovni okir Ker je na začetku potrebna daljša časovna krivulja za razvoj vmesnika, je potrebno pri postavljanju časovnih okvirjev upoštevati predhodno znanje in izkušnje, pridobljene s programom Expression Blend 4. Uporabnik začetnik bi za razvoj takšnega uporabniškega vmesnika potreboval od 2 do 3 ure dela, medtem ko je pri izkušenih uporabnikih čas, ki ga potrebujejo oblikovalci in razvijalci, bistveno manjši in se ocenjuje na maksimalno 1 uro. Za razvoj slednjega uporabniškega vmesnika sem potreboval 1.5 ure časa, saj sem že poznal proces razvoja prototipov z Expression Blend 4.

2. Število iteracij Število iteracij razvoja posameznega prototipa zaslona je bilo odvisno predvsem od treh dejavnikov:

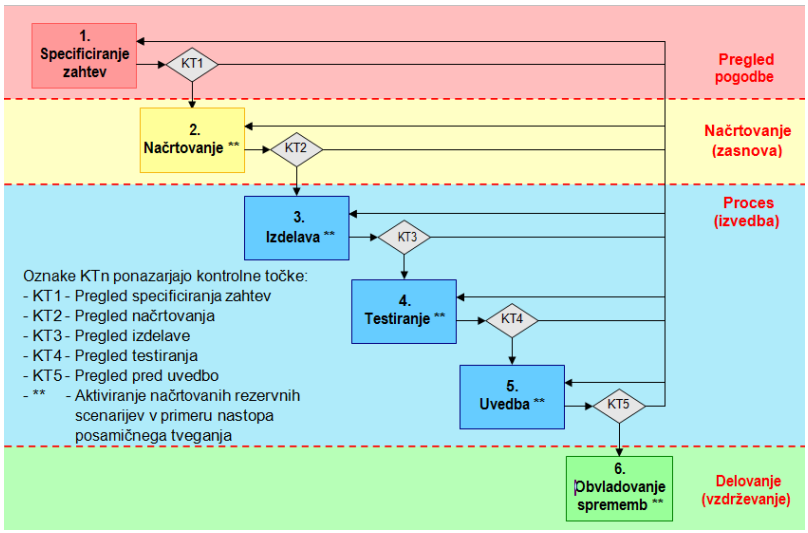
   - Nedvoumne in konsistentne dokumentacije (primeri uporabe), ki jo je imel na razpolago razvijalec prototipov.
   - Natančnosti in znanja razvijalca (oblikovalca).
   - Odobravanja sprememb ključnih uporabnikov na uporabniškem vmesniku.

   Za uspešno potrditev ustreznosti prototipa uporabniškega vmesnika Audit.xaml s strani ključnih uporabnikov sem potreboval dve iteraciji. V prvi iteraciji sem razvil uporabniški vmesnik na podlagi specifikacije zahtev in Visio slike obstoječe aplikacije. Ker uporabniki niso bili zadovoljni z nekaterimi pristopi, sem moral vmesnik popraviti v skladu z uporabniškimi zahtevami. Ključni uporabniki niso dovolili velikega odstopanja prototipa od starega vmesnika.

6.19 Prototipiranje in metodologija razvoja informacijskih sistemov v NLB

NLB d.d. ima v Upravljavskem centru za informacijsko tehnologijo zaposlenih preko 400 ljudi, ki skrbijo za IT podporo in razvoj. Z razvojem informacijskih sistemov banke se ukvarja približno 100 ljudi, ki delujejo na bolj ali manj kompleksnih projektih. Ker so veliki razvojni projekti ob veliki masi ljudi za izpeljavo zelo kompleksni, jih je potrebno podpreti z uvedbo različnih metodoloških pristopov, ki omogočajo konsistenten potek dela od zasnove do končne faze projekta. Razvoj informacijskih sistemov v NLB poteka po prirejenem razvojnem modelu, ki temelji na slapovnem model razvoja. Metodologija je definirana v uradno sprejetem dokumentu Metodologija razvoja in sprememb informacijskih sistemov v NLB. Metodologija se prvenstveno osredotoča na izvajanje razvojnega procesa in na sistematično vodenje zapisov oziroma stvarnih dokazil o korektno opravljenih postopkih. Ti zapisi se vodijo v tako imenovanih Mapah projektov. Dokument izvajalce vodi do čim uspešnejšega zaključka projekta ne glede na organizacijski vidik ali vrsto razvoja. Proces razvoja informacijskih sistemov zajema 6 razvojnih faz. V življenjski cikel razvoja so vpeljane še dodatne kontrole točke, s katerimi pregledujemo rezultate dela znotraj posamezne razvojne faze. Vsaka razvojna faza ima vnaprej določene izvajalce in naloge, ki se lahko odražajo v različnih zahtevkih, analizah, poročilih in podobno.

Razvoj prototipov uporabniških vmesnikov lahko poteka vzporedno s pisanjem primerov uporabe. Potek projekta v fazi analize je odvisen predvsem od vrste projekta, izvajalcev in tehnologije, v kateri bomo implementirali rešitev. Zelo pomembno je, da so prototipi usklajeni s specifikacijami zahtev in primeri uporabe, zato je v nekaterih primerih priporočljivo nekatere prototipe izdelati pred izdelavo specifikacije zahtev in primerov uporabe. Tak pristop priporočam v primeru, ko želimo prenoviti obstoječi sistem ter snovalci specifikacije zahtev in primerov uporabe ne poznajo tehnologije, v katerih bo implementirana rešitev. V tem primeru poskušamo oblikovalci in razvijalci



Slika 6.67: Življenjski cikel razvoja in sprememb informacijskih sistemov v NLB

prototipov izkoristiti vse možnosti, ki nam jih ponuja tehnologija pri razvoju prototipov, da jih lahko nato predstavimo poslovnim analitikom in drugim akterjem, ki sodelujejo pri pisanju specifikacije zahtev in primerov uporabe. Na razpolago dobijo praktično zelo podrobno sliko končnega sistema, zaradi česar lahko primere uporabe prilagodijo tehnološkim zmožnostim, ki nam jih ponuja Silverlight tehnologija. Velja dejstvo, da je izbira SketchFlow prototipov najbolj optimalna in racionalna rešitev, saj je razvoj prototipov uporabniških vmesnikov za Register komitentov potekal na enaki platformi kakor končna rešitev. Ne glede na vse, pa je potrebno povedati, da se znotraj faze specifikacije zahtev izvede več ponovnih iteracij, v katerih je potrebno dokumente in izdelke popravljati ter dopolnjevati glede na poslovne in uporabniške zahteve. Po končanem prototipiranju je bilo potrebno ponovno še enkrat pregledati vsak primer uporabe in v prototipu popraviti in/ali dopolniti manjkajoče stvari. Razlika je le v tem, da je bilo teh integracij v primeru, ko smo izdelali prototip zaslona pred primeri uporabe, bistveno manj, saj so je v nasprotnem primeru ponekod prototipi razlikovali od specifikacije zahtev. Do takih napak je prihajalo predvsem zaradi omejitev pri

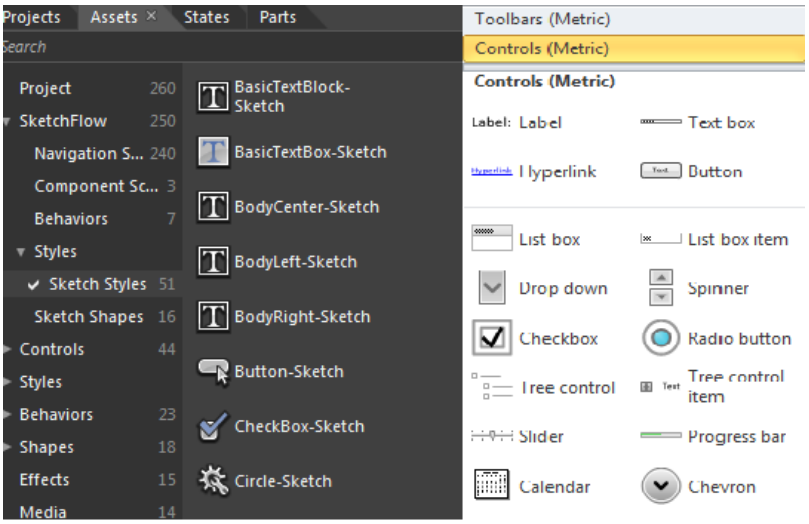
tehnologiji ali boljših tehnoloških rešitev razvijalca prototipov, kakor so jih predvideli v primerih uporabe. Ko izdelamo prototip zaslona, ga lahko z uporabno funkcijo odložimo na SharePoint spletno mesto, ki smo ga predhodno oblikovali za dostop uporabnikov in razvojne skupine do prototipov. Tam si uporabniki lahko ogledajo prototip aplikacije in svoje odzive na izdelek preko Silverlight vtičnika posredujejo razvijalcem. Proces razvoja posameznega prototipa zaslona se zaključi, ko se v celoti izdela ter potrdi prototip zaslona, ki je skladen z zahtevami, ki so zapisane v specifikacije zahtev v primeru uporabe. Pregled in potrditev izvedejo ključni uporabniki s projektnim vodjem. V nasprotnem primeru se izvede nova iteracija, v kateri odpravimo pomanjkljivosti. Ko je faza analize zaključena, se prične faza načrtovanja. Pri načrtovanju uporabniških vmesnikov se še posebej izkaže uporaba prototipov. Ti že po definiciji zajemajo tisto, kar od načrta uporabniškega vmesnika pričakujemo: prikaz izgleda ter osnovne funkcionalnosti. Zato se za pripravo načrta uporabniškega vmesnika prototipi veliko uporabljajo. Če se spomnimo, smo uporabo prototipov omenjali že v fazi analize, ko smo govorili o tehniki zajema zahtev. Uporabljajo se namreč lahko tudi kot tehnika za definicijo zahtev, vendar pa s tem segajo tudi v fazo načrtovanja, saj med drugim predstavljajo tudi načrt uporabniškega vmesnika. Zato velja, da uporaba prototipov pri razvoju IS zmanjšuje mejo med analizo in načrtovanjem. Velja celo več: zmanjšuje mejo tudi med načrtovanjem in izvedbo, saj po metodologijah hitrega razvoja prototipi niso samo načrti ali delovne verzije, ampak predstavljajo osnovo, iz katere se v fazi izvedbe razvije končna aplikacija [18]. Razvoj prototipov uporabniških vmesnikov za aplikacijo Register komitentov se od faze analize naprej ne izvaja več. Prototipi v fazi načrtovanja služijo zgolj kot pomoč razvojni skupini pri izdelavi načrta IT rešitve. Ti so še posebej dobrodošli pri posameznikih, ki ne poznajo Silverlight tehnologije, v kateri bomo izdelali končno aplikacijo, saj s tem pridobijo celostno sliko celotnega sistema, prav tako pa ostali razvojni skupini pomagajo pri načrtovanju. S tem se zmanjša čas, ki ga potrebujejo za organizacijo in izpeljavo sestankov na katerih s člani projektne skupine usklajujejo svoje

ideje in rešitve, poleg tega pa delo poteka hitreje z manjšimi možnosti za dodatne napake in neskladja v povezavi s specifikacijami zahtev in primeri uporabe. Ker je vse skupaj na SharePoint spletnem portalu, imajo izvajalci ves čas na voljo dostop do prototipov, kjer si jih lahko ogledajo. Ko faza načrtovanja prestopi v fazo izdelave, se prototipe prav tako uporablja za vpogled. Ker se prototipov ne da preslikati v produkcijsko aplikacijo, koderji oziroma razvijalci uporabniških vmesnikov zgradijo nove uporabniške vmesnike, ki so izdelani na podlagi prototipov. Ker imajo programerji na voljo žive prototipe, lahko v vsakem trenutku pridobijo tehnične informacije in s tem pospešijo proces razvoja. Glede na to, da se prototipe na nek način lahko pretvori v produkcijski projekt je to mogoče, vendar pa ni priporočljivo, saj je potrebno zamenjati vse vire, ki so povezani s SketchFlow kontrolami in jih ročno spremeniti v kontrole Silverlight. To zahteva veliko časa in truda, poleg tega pa je kvaliteta odvisna tudi od zahtevnosti izdelave prototipa, saj nekateri uporabniki progama Expression Blend 4 niso nujno programerji. Prototipi po fazi izdelave niso več smiselni za uporabo, zato se jih ne uporablja več. Zelo priporočljivo jih je shraniti za primer morebitne naknadne spremembe sistema, kjer jih lahko zopet uporabimo kot celoto.

6.20 Primerjava med Visio in SketchFlow prototipi

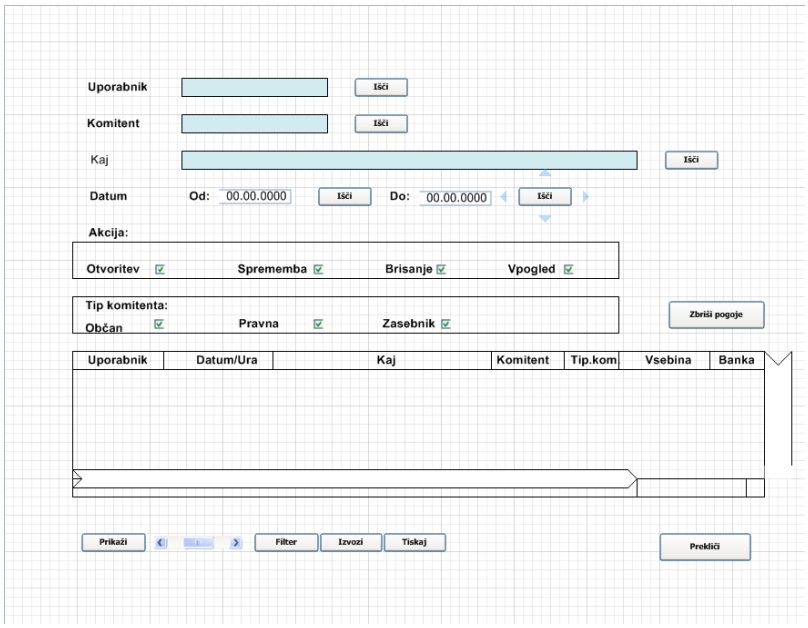
Microsoft Visio in Expression Blend 4 sta orodji, ki nam ponujata možnost razvoja prototipov. Oba orodja sta last programske hiše Microsoft. Orodje Microsoft Visio se prvenstveno uporablja za izrisovanje različnih diagramov, lahko pa ga uporabljamo tudi za načrtovanje in prototipiranje uporabniških vmesnikov. Najprej bi opozoril na nekaj razlik, ki se odražajo v samem procesu izdelave prototipov. Microsoft Visio ne omogoča načrtovanja prototipov za spletne aplikacije, saj je z verzijo Microsoft Visio 2010 odstranil okno brskalnik. Ne glede na vse lahko še vedno razvijamo prototipe. V Microsoft Visiu ustvarimo novo okno s preprostim potegom kontrole, ki predstavlja za-

slon na papir, ki je lahko poljubno velik in lahko vsebuje več oken hkrati. Pri Expression Blend 4 je to malo drugače, saj se za kreiranje okna uporablja funkcija Add Screen, ki jo omogočajo zasloni, ki se nahajajo SketchFlow Zemljevidu. Dodajanje kontrol na zaslon je v obeh programih praktično enako. Razlika je le v grafični podobi in naboru rešitev.

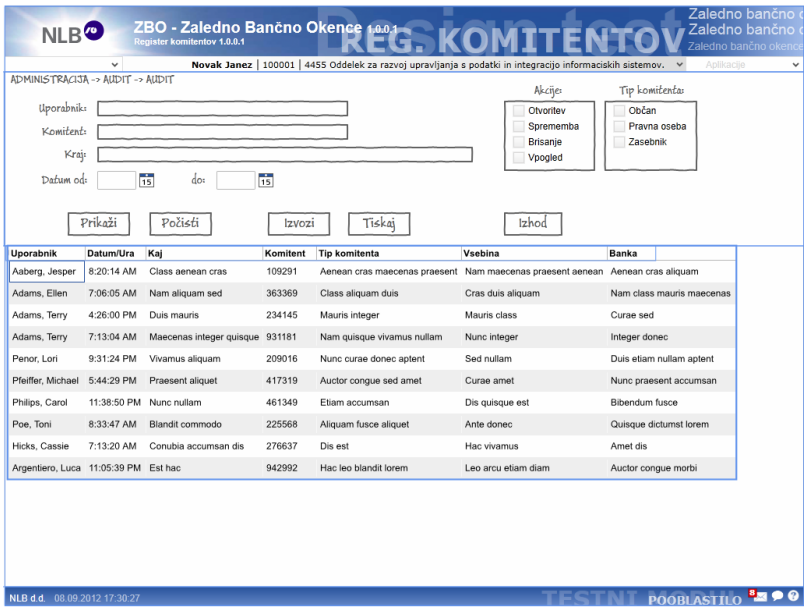


Slika 6.68: SketchFlow in Visio kontrole.

Razlika med prototipi, narejenimi s programom Visio ter Expression Blend 4, se opazijo predvsem v rešitvi in pristopu pregledovanja, ki ga Expression Blend ponuja. Microsoft Visio ponuja razvoj statičnih prototipov zaslonov. Za pregledovanje prototipov ne potrebujemo namestiti programa Visio, saj imajo uporabniki na razpolago Visio pregledovalnik za Internet Explorer, iz katerega lahko slike tiskajo in pregledujejo.



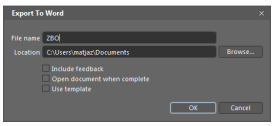
Slika 6.69: Microsoft Visio statični prototip vmesnika.



Slika 6.70: SketchFlow prototip vmesnika.

6.21 Pretvorba SketchFlow projekta v produkcijski projekt

Če želimo pretvoriti SketchFlow projekt v produkcijski projekt, mu je potrebno odstraniti reference iz SketchFlow projekta. Projekt pretvorimo v projekt, ki je pripravljen za razvoj aplikacije, namenjene potrebam produkcije po sledečem vrstnem redu. Zaradi varnosti je potrebno najprej ustvariti varnostno kopijo, ki jo bomo uporabili v primeru, da se nam pri pretvorbi projekt kakorkoli poškoduje ali pa v primeru, če želimo popravljati prototipe ali jih dopolnjevati. Zaradi beleženja in verzioniranja projekta je zelo priporočljivo, da si ustvarimo TFS strežnik, na katerega odlagamo naše prevedene rešitve. V panelu Project z desnim gumbom miške kliknemo na oznako Rešitev ZBO (angl. Solution ZBO) in izberemo Open Folder in Windows Exploler.



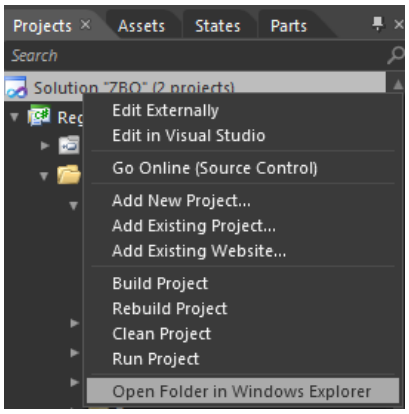
Slika 6.71: Odpiranje projekta v Windows Explorerju.

Odpre se nam Windows Explorler v katerem najdemo datoteke, ki pripadajo SketchFlow projektu. Poiščemo datoteko ZBO.cspoj in jo odpremo z Beležnico (ang. notepad).

V tekstovni datoteki, označimo in izbrišemo naslednji dve vrstici:

```
<ExpressionBlendPrototypingEnabled>false
</ExpressionBlendPrototypingEnabled>
<ExpressionBlendPrototypeHarness>true
</ExpressionBlendPrototypeHarness>
```

Ko izbrišemo tekst, shranimo naše spremembe in zapremo beležnico. Odpremo datoteko References, v kateri poiščemo Microsoft.Expression.Prototyping.Runtime.dll



Slika 6.72: Odpiranje datoteke ZBO.csproj.

in jo odstranimo iz projekta. Ponovno z desnim gumbom miške kliknemo na datoteko Reference in izberemo Add Rederence. V oknu Add Reference poiščemo System.Windows.Controls.Navigation.dll in ga izberemo. Privzeto mesto za 64 bitni operaciski sistem Windows 7, kjer se nahaja dinamična povezovalna knjižnica, je

```
C:\Program Files (x86)\Microsoft SDKs\Silverlight\v3.0\Libraries\
Client
```

.

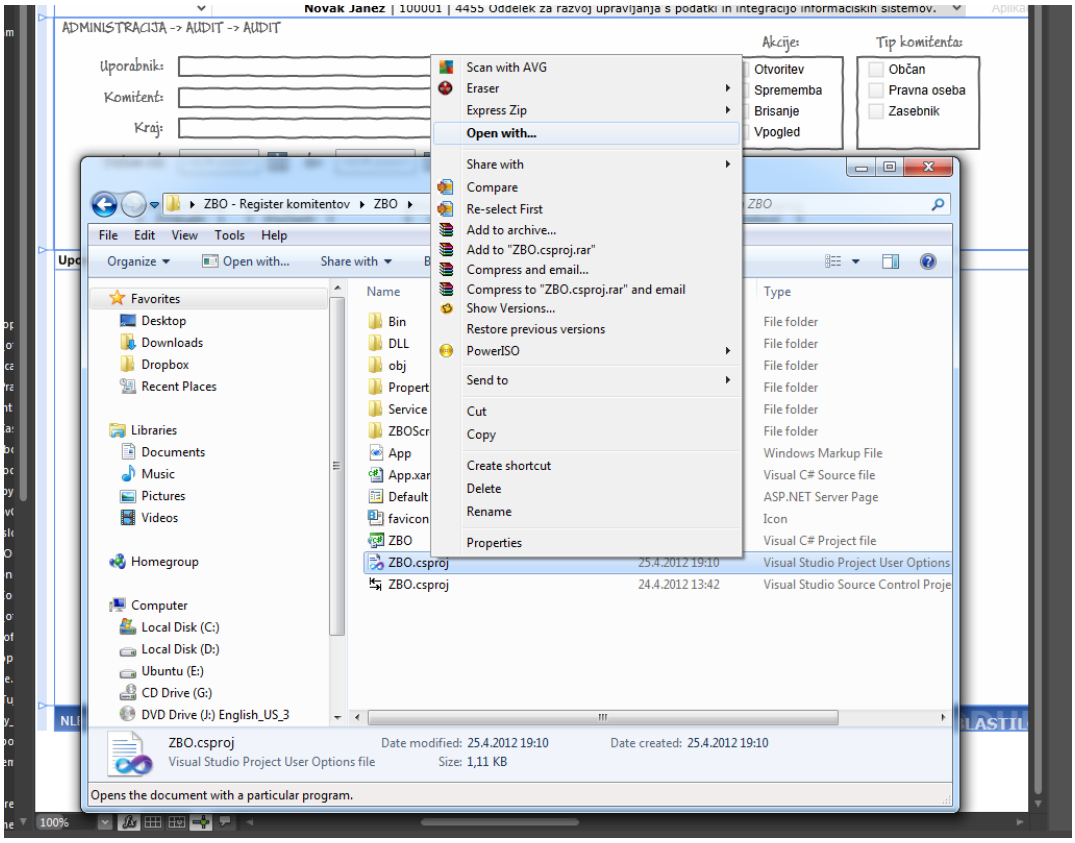
Kliknemo gumb Open (slika 6.73).

V projektu ZBO poiščemo datoteko, ki se imenuje App.xaml, ter jo razširimo. Odpremo datoteko App.xaml.cs.

```
Microsoft.Expression.Prototyping.Services.SketchFlowLibraries(
"ZBO.Screens")]
```

[assembly: V kodi poiščemo vrstico, v kateri se nahaja zgornja koda, jo označimo ter izbrišemo. V isti datoteki poiščemo kodo:

```
this.RootVisual = new Microsoft.Expression.Prototyping.Workspa-
ce.PlayerWindow();
```



Slika 6.73: Dodajanje referenc v rešitev.

in jo zamenjamo z naslednjo kodo:

```
this.RootVisual = new System.Windows.Controls.Frame() { Source
= new Uri("/ZBO.Screens;component/MainScreen.xaml", UriKind.Rel-
ative) };
```

Shranimo projekt in ga prevedemo.

V brskalniku se nam požene projekt, ki ne vsebuje SketchFlow predvajalnika [16].

Potrebno je povedati, da imamo v praksi na voljo dve vrsti razvoja prototipov v Expression Blend 4. V prvem primeru razvijamo prototipe v obliki, ki predstavlja končno rešitev, medtem ko se druga vrsta, ki ji praviloma pravimo tudi SketchFlow prototipi, namensko uporablja izključno za pomoč

Slika 6.74: Končni rezultat pretvorbe prototipa.

pri načrtovanju in kodiranju s tem načrtovalcem ter koderjem ponuja celovit pogled sistema. Tak razvoj omogoča kakovostnejšo gradnjo uporabniških vmesnikov, saj zaradi načina dela nismo zgolj usmerjeni k hitri rešitvi, ampak tudi k najoptimalnejši in najkakovostnejši, ki odpravlja odpor uporabnikov do uporabe aplikacije na praktično minimalno stopnjo.

Poglavje 7

Zaključek

Ker je tehnološki napredek vse večji, bodo banke in druge podjetja, ki se ukvarjajo z razvojem sodobnih poslovnih aplikacij, za potrebe doseganja konkurenčnosti morale uporabljati inovativne pristope k IT razvoju aplikacij. Na trgu se pojavljajo vse bolj zahtevne stranke oziroma ključni uporabniki, ki od razvijalcev pričakujejo nadpovprečne rezultate. Da bi te zahteve lahko maksimalno zagotovili, nam pri razvoju aplikacij na Microsoftovih platformah pomaga tudi program Expression Blend 4, ki oblikovalcem in razvijalcem programske opreme ponuja številne rešitve, s katerimi poskušamo zadovoljiti zahteve uporabnikov.

Ker se nahajamo v času, ko je gospodarska kriza na vrhuncu svoje moči, so številna podjetja zelo omejena s finančnimi sredstvi. Zaradi pomanjkanja finančnih sredstev se je v organizacijah zelo velika pozornost usmerila tudi v zmanjševanje stroškov na vseh ravneh. Ena od možnosti za razvoj platforme Silverlight je, da je Microsoft v ta namen na tržišče ponudil rešitev s katero je združil lastnosti namiznih in spletnih aplikacij v eno platformo, ki delujejo neodvisno od operacijskega sistema, saj jo lahko uporabniki uporabljajo na katerikoli platformi operacijskega sistema. Na Silverlight platformi lahko z orodjem Expression Blend 4 enostavno in hitro razvijamo rešitve, vse od prototipov pa do informacijskih rešitev za podjetja.

Učinkovito vizualno orodje oblikovalcem ter razvijalcem omogoča risanje,

oblikovanje, 2D in 3D animiranje vmesnikov za spletne, namizne ali mobilne aplikacije in prototipe. Za razvoj prototipov je potrebna zelo kratka učna krivulja, tako da je možno pričeti z razvojem v sorazmerno zelo kratkem času, medtem ko je za razvoj uporabniških vmesnikov potrebna nekoliko daljša učna krivulja. Ob tem velja nameniti tudi kritiko Microsoftu, saj bi pričakovali, da program Expression Blend 4 ponuja avtomatično pretvorbo SketchFlow projekta v produkcijski projekt, ki bi ga lahko nato razvili v končno aplikacijo. Ne glede na to, da program ponuja delno možnost pretvorbe projektov, je pretvorba SketchFlow prototipov v Silverlight projekte za potrebe nadaljnjega razvoja nesmiselna in neuporabna. Če bi hoteli SketchFlow prototip preoblikovati v pravo aplikacijo, bi kljub pretvorbi projekta v produkcijski projekt potrebovali veliko časa za pretvorbo, saj je potrebno ročno spremeniti vse statične vire, ki definirajo obliko uporabniškega vmesnika.

Diplomsko delo v večji meri temelji na razvoju SketchFlow prototipov uporabniških vmesnikov v programu Expression Blend 4, ki sem jih razvijal na projektu prenove Registra komitentov v NLB d.d.. Za predstavitev prototipiranja v Expression Blend 4 sem se odločil, ker je orodje praktično edino na tržišču, ki ponuja razvoj interaktivnih prototipov poleg tega pa je združljivo z različnimi Microsoftovimi rešitvami kot je Team Foundation Server, Visual Studio in SharePoint, preko katerega lahko uporabnikom omogočimo dostop do pregledovanja. V povezavi z njimi se zelo olajša delo v projektni skupini, saj lahko člani razvojne skupine v vsakem trenutku dostopajo do zadnje različice rešitev in jo pregledajo, komentirajo, nadgradijo in podobno. S tem razvijalcem zagotovimo več časa za druga bolj pomembna opravila na razvoju.

Z vidika uporabnika je Expression Blend 4 orodje, ki je zelo prijazno uporabnikom, ne samo razvijalcu ekranov, ampak tudi samim uporabnikom končne aplikacije. Zelo uporabna je opcija oddajanja komentarjev, ki jo lahko uporabnik koristi, ko želi komentirati delo razvijalca ekranov v primeru, če mu kaj na ekranu ni všeč, se s čem ne strinja, bi kaj dodal itd. Prav tako Expres-

sion Blend 4 mogoča lažji vpogled v ekrane preko brskalnika, ki uporabniku da občutek prave aplikacije, ki jo bo dobil na koncu v uporabo. Ko sem g. Barbari Hunter iz podjetja NLB d.d. zastavil vprašanje kaj meni o uporabi progama Expression Blend 4 z vidika razvijalca prototipov mi je na zastavljeno vprašanje odgovorila takole: "Expression Blend 4 je super program z vseh vidikov. Razvijalec se lahko po svojih kreiranih ekranih orientira na dva načina: SketchFlow zemljevid in seznam projektov, na katerih dela trenutno ("Projects"); odvisno, kaj mu je bolj všeč. Oba vpogleda sta zelo enostavna in zelo lahko se je "sprehajati" med ekrani. SketchFlow zemljevid ponuja odličen vpogled v povezave med ekrani. Ekrane lahko kreira oziroma nariše hitro, ker je orodje Expression Bled res zelo enostavno tudi za popolnega začetnika. Npr. jaz sem se s pomočjo videov vodičev sama naučila uporabljati Expression Blend 4 v samo nekaj dneh. Kreiranje povezav med ekrani, dodajanje slik, podatkov na ekrane oziroma kreiranje testnih ali pa pravih baz podatkov, kar uporabnik pač potrebuje, je zelo enostavno in hitro. Opcija kreiranja stanj (states) je tudi zelo lahka za uporabo. Z njo se lahko razvijalec veliko igra pri kreiranju "pop-up" ekranov, pojavi določenih polj ali slik na ekranu itd. V primerjavi z drugimi orodji, kot na primer Microsoft Visiom, je orodje Expression Blend 4 v veliki prednosti. Če ne zaradi vseh zgoraj naštetih stvari, že zaradi časa, ki ga prihraniš s tem, da uporabljaš Expression Blend. Če bi jaz na primer uporabljala Visio in ne Expression Blend 4, bi sigurno porabila še enkrat toliko časa za risanje prototipov, kot sem ga z Expression Blend 4".

Program Expression Blend 4 priporočam v uporabo vsem, ki želijo razvijati napredne uporabniške vmesnike, saj lahko v zelo kratkem času razvijemo zelo dobre uporabniške vmesnike. Poleg tega Microsoft redno izdaja nove različice programa, ki nam ponujajo dodatne funkcionalnosti. Ker lahko razvijemo prototipe in aplikacije praktično brez znanja programiranja, ga lahko uporablja vsakdo, ki je vešč uporabe računalnika. S funkcijo povleči in spusti lahko naredimo zadovoljive vmesnike, če pa svoje znanje še poglobimo, pa lahko izdelamo vrhunske aplikacijske vmesnike in prototipe. V primerjavi s

programom Microsoft Office Visio ponuja uporabniku bolj prijazno rešitev, saj uporabnik takoj pridobi uporabniško izkušnjo, medtem ko je pri programu Microsoft Visio možno le grafično predstaviti vmesnik. Zaradi povezljivosti s SharePoint portalom, ki uporabnikom omogoča enotno dostopno točko za pregledovanje in komentiranje prototipov, se zelo izboljša komunikacija med oblikovalci in končnimi uporabniki, poleg tega pa se za polovico zmanjša potreba po organizaciji sestankov. V primeru, ko ima razvojna skupina sestanek, ni več potrebno tiskanje poročil, saj nam Expression Blend4 v povezavi s SharePoint portalom omogoča pregledovanje prototipov kar preko spleta. Če pa se pojavi potreba po izdelavi poročila nam pa to Expression Blend 4 stori v trenutku, saj se v Expression Blend 4 nahaja funkcija za generiranje Word poročil. Microsoftu je tako uspelo narediti zelo dobro in vsestransko orodje, brez katerega bi bila marsikatera ovira na poti do učinkovitega grafičnega vmesnika nerešljiva.

Literatura

[1] (2000) Podatkovna skladišča v finančnih institucijah. Dostopno na: http://www.src.si/library_si/pdf/infosrc/InfoSRC.SI%20-%202000-28.pdf.

[2] (2009)Primeri uporabe Microsoft Expression Blend 3. Dostopno na: http://www.msblogs.si/post/Primeri-uporabe-Microsoft-Expression-Blend-3.aspx.

[3] (2010) Introduction to Expression Blend. Dostopno na: http://www.longhorncorner.com/uploadfile/puranindia/introduction-to-expression-blend.

[4] (2009) Aplikacije prihodnosti že danes. Dostopno na: http://www.buyitc.si/novica/2971/Microsoft-Silverlight-3-in-Expression-3-prinasata-vrsto-novosti-z-zanimanjem-jih-pricakujejo-tudi-slovenski-partnerji.

[5] Brennon Williams, *Microsoft Expression Blend 4 Unleashed 2011*, 2011.

[6] (2012) GUI Prototyping Tools. Dostopno na: http://c2.com/cgi/wiki?GuiPrototypingTools.

[7] (2012) SketchFlow: An Overview. Dostopno na: http://expression.microsoft.com/en-us/ee215229.aspx.

[8] Chris Bernard, Sara Summers, *Dynamic Prototyping with SketchFlow in Expression Blend*, 2011.

[9] (2009) Silverlight 3 in Expression 3 prinašata vrsto novosti, z zanimanjem jih pričakujejo tudi slovenski partnerji. Dostopno na: www.buyitc.si/downloadfile.aspx?fileid=337.

[10] J. Ambrose Little, Jason Beres, Grant Hinkson, Devin Rader, Joe Croney, *Silverlight 3 Programmer's Reference*, 2010.

[11] (2010) Application and Programming Models. Dostopno na: http://msdn.microsoft.com/en-us/library/cc903934(v=vs.95).

[12] (2010) Out-of-Browser Support. Dostopno na: http://msdn.microsoft.com/en-us/library/dd550721(v=vs.95).

[13] (2010) Out-of-Browser Applications. Dostopno na: http://www.silverlight.net/learn/overview/out-of-browser-applications/out-of-browser-applications-(silverlight-quickstart).

[14] (2010) Installing Silverlight applications without the browser involved. Dostopno na: http://timheuer.com/blog/archive/2010/03/25/using-sllauncher-for-silent-install-silverlight-application.aspx.

[15] (2010) More Fun with SketchFlow: Details of the SketchFlow Player. Dostopno na: https://www.intertech.com/Blog/Post/More-Fun-with-SketchFlow-Details-of-the-SketchFlow-Player.aspx.

[16] (2012) Convert into a production project. Dostopno na: http://msdn.microsoft.com/en-us/library/ee371158(v=expression.30).

[17] (2011) Modify sample data. Dostopno na: http://msdn.microsoft.com/en-us/library/ee341407(v=expression.40).aspx.

[18] (2012) Načrt uporabniškega vmesnika in izpisa: tehnika. Dostopno na: http://www2.gov.si/mju/emris.nsf/0/4287A1D6DBB42872C1256EB4007AD57B?OpenI

[19] (2007) Kaj je uporabnost in kaj uporabniška izkušnja. Dostopno na: http://www.mojmikro.si/preziveti/kar_tako/vojne_za_boljso_uporabnisko_izkusnjo.

[20] (2011) Flex, Silverlight or HTML5? Dostopno na: http://www.scottlogic.co.uk/blog/colin/wp-content/uploads/2011/05/Flex-Silverlight-HTML5.pdf

[21] Andrew Troelsen, *Pro Expression Blend 4*, 2011

[22] Victor Gaudiso, *Foundation Expression Blend 4 with Silverlight*, 2010

[23] Elena Kosinska, Chris Leeds, *Expression Blend 4 Step by Step*, 2011

[24] Flex, Silverlight or HTML5? Dostopno na: http://www.scottlogic.co.uk/blog/colin/wp-content/uploads/2011/05/Flex-Silverlight-HTML5.pdf

[25] (2010) Expression Blend 4. Dostopno na: http://www.microsoft.com/expression/products/Blend_Overview.aspx