Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Blaž Rogelj

Jezikovno prilagajanje proizvodnih IS z uporabo konteksta zaslonskih mask

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: viš. pred. dr. Aljaž Zrnec

Ljubljana, 2019

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Pogosto se zgodi, da imajo prevajalci aplikacij težave pri prevajanju besedil, saj iz seznama besed za prevod, ki ga dobijo, ne poznajo konteksta besedila. Z enako težavo se srečuje podjetje, ki se ukvarja z razvojem informacijskih sistemov. Da bi prevajalec videl kontekst besedila, mora poiskati stran na katerem se beseda uporablja, kar pa je zaradi obsežnosti informacijskega sistema lahko zelo zamudno. V okviru diplomske naloge razvijte prototip orodja, ki bo prevajalcem omogočal prevajanje besedil na podlagi spletnih strani informacijskega sistema. Nad posamezno besedo ali stavkom naj ima prevajalec možnost urejanja besede ali kreiranja novega prevoda v drug jezik.

Iskreno se zahvaljujem mentorju viš. pred. dr. Aljažu Zrnecu, ki mi je strpno pomagal in me vodil tekom pisanja diplomskega dela.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
SQL	structured query language	strukturirani povpraševalni jezik
VPN	virtual private network	navidezno zasebno omrežje
CSS	cascading style sheets	kaskadne stilske podloge
IIS	internet information services	internetne informacijske storitve

Povzetek

Naslov: Jezikovno prilagajanje proizvodnih IS z uporabo konteksta zaslonskih mask

Avtor: Blaž Rogelj

Prevajalci aplikacij se pogosto srečajo s težavo, da ne najdejo pravega prevoda podani besedi, ali pa ne vedo, kako bi jo lahko prevedli, ker je ta lahko dvoumna. Do teh težav pride zaradi tega, ker ne poznajo konteksta aplikacije, kjer se beseda uporablja. Pogosta praksa je, da razvijalci izvozijo besede v nek seznam in ta seznam pošljejo prevajalcem. Ko prevajalci prevedejo besede, pošljejo seznam nazaj, kjer morajo razvijalci aplikacije te besede uvoziti v svojo aplikacijo, s tem pa se tudi upočasni razvoj. Če se prevajalci za določeno besedo ne znajo odločiti, kateri prevod bi bil primeren, potem prosijo razvijalce za dodatna pojasnila o kontekstu besede. V manjših informacijskih sistemih konteksta besede ni težko odkriti, večji problem predstavljajo kompleksnejši informacijski sistemi, kjer je včasih težko ugotoviti, kje se določena beseda uporablja.

Pri reševanju tega problema bomo razvili rešitev, ki bo prevajalcem omogočala prevajanje besedil v kontekstu spletnih strani informacijskega sistema. Prevajalec bo v našo aplikacijo vnesel povezavo do spletne strani, iz katere bo aplikacija izluščila besede, ki jih je mogoče prevesti. Ker bo aplikacija omogočala neposredno prevajanje brez vmesnega dela programerjev, ki skrbijo za vnos prevodov v podatkovno bazo, bo prevajanje posledično hitrejše.

Ključne besede: prevajanje, kontekst, informacijski sistem, MX ogrodje,

spletna stran.

Abstract

Title: Language localization of production ISs using the masks context

Author: Blaž Rogelj

Application translators often face a problem of not finding an appropriate translation for a specific word or are unable to translate it, as it has multiple interpretations. These problems arise when a translator is unaware of the context the word is used in. So, the common practice for the developers is to export the words into a list, which then gets send to the translators. When the words are translated, they are then sent back to the developers, whose task it is to import them into their application, with which the application development process is greatly slowed down. If the translators come across the same problems of not knowing which is the appropriate translation, they are forced to ask for additional information. In smaller information systems the context is easily identified; whereas the more complicated require significantly more effort.

In our attempt at fixing this issue We will develop a solution, which will enable translators to translate text based on the context of the information systems website. Translator will be required to enter a link to the website, from where our program will extract the words it needs to translate. Since our application will offer a direct translation, without the intermediate of developers, which would normally be tasked with inputting the information into a websites database, the whole translation process will improve in speed.

Keywords: translating, context, information system, MX framework, web

page.

Poglavje 1

Uvod

V kontekstu globalnega gospodarstva resni informacijski sistemi oziroma aplikacije podpirajo uporabniške vmesnike v različnih jezikih. Razvijalci jih običajno razvijajo v svojem jeziku ali v angleščini, če želijo omogočiti širšo uporabo informacijskih sistemov, jih morajo prevesti. Podjetje, v katerem smo si pridobili motivacijo za pisanje dotičnega diplomskega dela, se ukvarja z razvojem informacijskih sistemov. Za razvoj uporabljajo ogrodje MX-Frame (v nadaljevanju MX), ki so ga izdelali sami, saj le-to omogoča hitrejši razvoj. Ogrodje v osnovi podpira prevajanje besedil, vendar pri večjih količinah besed ta način ni najbolj primeren, saj mora uporabnik, da pride do iskane besede, preklapljati med različnimi zaslonskimi maskami informacijskega sistema. Pri večjih količinah besed tak način postane zelo zamuden, saj lahko uporabnik tako zapravi med eno do dve minuti časa na besedo.

Druga pomanjkljivost ogrodja je, da imajo zaradi pomanjkanja konteksta prevajalci aplikacij, včasih težave pri samem prevajanju. Kontekst oziroma z drugo besedo sobesedilo, je pomembna informacija za prevajalce, da lahko prevedeni besedi določijo pravi pomen, glede na druge besede ali stavke, ki se v besedilu uporabljajo z njo [10]. Ker prevajalci dobijo le seznam besed, ki jih morajo prevesti, in tako ne vidijo celotne slike o tem kje se beseda uporablja, se pri njenem prevodu lahko zmotijo. Če prevajalec ne zna prevesti besede, lahko naredi dvoje; lahko poizkusi uganiti pravilen prevod

ali pa razvijalce prosi za dodatna pojasnila o besedi ter se tako lažje odloči, kateri prevod bo pravilen. Nobena od zgornjih dveh možnosti ni dobra. Pri prvi možnosti lahko dobimo prevode, ki niso smiselni, pri drugi možnosti pa pride do zakasnitve pri prevajanju, saj mora prevajalec čakati na odgovor programerja, ki mu pomaga z dodatnimi pojasnili.

Podjetje za prevajanje svojih informacijskih sistemov uporablja dva načina prevajanja. Prvi način je izvoz besed v *Excel-ov* seznam, drugi način pa je uporaba funkcije za prevajanje, ki je del ogrodja MX. Oba načina imata svoje pomanjkljivosti, katere bomo podrobneje razdelali v naslednjih poglavjih.

1.1 Težave pri prevajanju s seznamom besed

Razvijalci iz podatkovne baze izvozijo besede, ki jih je mogoče prevesti. Pri tem si napišejo poizvedbo, ki vrne vse relevantne besede in jo izvozijo v *Excel-ov* seznam. Na sliki 1.1 vidimo delček realnega primera seznama besed, ki ga morajo prevajalci iz angleščine prevesti v ruščino. V prvem stolpcu je beseda, ki jo morajo prevesti, v drugem stolpcu pa je prevod te besede. Če je slednja obarvana rdeče, potem prevajalec ni prepričan v svoj prevod in o besedi želi dodatna pojasnila. Čeprav so besede v prvem stolpcu na prvi pogled samoumevne, imajo lahko v različnih jezikih dvoumen pomen. Besedo rdeče obarvajo prevajalci, ko niso prepričani o pravilnosti svojega prevoda in želijo dodatne informacije o kontekstu. Rdeče obarvana beseda ni nujno napačna, vendar prevajalci niso povsem prepričani o njeni ustreznosti.

Beseda *Welcome* je pravilno prevedena, vendar je vseeno označena z rdečo barvo, saj prevajalec ni bil prepričan, ali je bila beseda uporabljena v kontekstu pozdrava ali zahvale, vseeno pa je uganil pravilen prevod. Stavka *Sign out* in *Please Enter Your Credentials* sta bila prav tako prevedena pravilno, vendar so tudi tu prevajalci dvomili v pravilnost prevoda. Za stavek *Sign In* se je izkazalo, da ni bil preveden pravilno, na možnost česar je prevajalec opozoril, z uporabljeno rdečo barvo. Stavek se uporablja v kontekstu prijave uporabnika v sistem. Slovenski prevod bi bil *Prijava* ali pa *Prijavi se*,

medtem ko so prevajalci besedo razumeli kot *Registriraj se*. Ruski prevod pomeni nekaj v smislu *Pridružite se* ali pa *Registrirajte se*.

Welcome	Добро пожаловать
Sign Out	Выписаться
My Settings	Мои Настройки
Please Enter Your Credentials	Пожалуйста, Введите Свои Полномочия
Sign In	Зарегистрироваться
Remember me	Напомнить мне
Control Panel	Панель Управления

Slika 1.1: Prevedene besede brez konteksta v ruščino

Pri tem pa ne smemo pozabiti tudi na možnost, da se prevajalec zmoti in se tega ne zaveda. Lahko se zgodi, da besedo prevede napačno in je ne označi z rdečo barvo, saj sklepa, da je le-ta pravilna. V tem primeru bodo napačno besedo opazili uporabniki informacijskega sistema, česar si pa ne želimo.

Prevajalci seznam prevedenih besed vrnejo v *Excel-ovi* razpredelnici, kjer prevedene besede programerji združijo v vnosne SQL stavke katere bodo kasneje vnesli v podatkovno bazo. Prevedeni besedi in njeni originalni vrednosti dodajo ključ in kratico jezika, v katero je bila beseda prevedena. Pri tem si pomagajo s programom *Excel*, kjer z združevanjem celic med seboj ustvarijo vnosne stavke. Celico v kateri je SQL-ova vnosna sintaksa (oz. insert stavek) združimo z drugimi celicami, ki nosijo podatke o imenih atributov in njihovimi vrednostmi. Če se v celici s prevodom pojavi vsaj en enojni narekovaj, bo prišlo do narobe ustvarjenega vnosnega stavka, saj bo ta narekovaj pokvaril sintakso stavka. Pred uvozom prevedenih besed v podatkovno bazo, morajo programerji vse stavke ročno pregledati. Ko so vnosni stavki pripravljeni, se jih iz programa *Excel* izvozi v SQL skripto, ki skrbi za pravilno preslikavo besed v podatkovno bazo. Namreč lahko se zgodi, da prevod nekatere besede z istim ključem že obstaja. V tem primeru ne smemo izvesti vnosnega stavka, saj bo prišlo do referenčne napake, ampak moramo narediti posodobitev stare vrednosti na novo. Na sliki 1.2 je primer vnosnih stavkov v programu Excel, ki jih z kopiranjem vnesemo v SQL vnosno skripto.

Obstoječi način prevajanja s seznamom besed ni dober, saj je potrebno besede ročno izvažati v Excel-ove tabele in ustvarjati vnosne skripte, ki jih je potrebno tudi ročno pregledati in ugotoviti, ali vsebujejo znake, ki lahko povzročijo težave ob vnosu v podatkovno bazo.

```
insert into languageTempTable (id, default_value, translation, language)values(77, 'Welcome', 'Добро пожаловать', 'ru');
insert into languageTempTable (id, default_value, translation, language)values(80, 'Sign Out', 'Выписаться', 'ru');
insert into languageTempTable (id, default_value, translation, language)values(81, 'My Settings', 'Мои Настройки', 'ru');
insert into languageTempTable (id, default_value, translation, language)values(85, 'Please Enter Your Credentials', 'Пожалуйста,
insert into languageTempTable (id, default_value, translation, language)values(86, 'Sign In', 'Зарегистрироваться', 'ru');
insert into languageTempTable (id, default_value, translation, language)values(87, 'Remember me', 'Напомнить мне', 'ru');
insert into languageTempTable (id, default_value, translation, language)values(88, 'Control Panel', 'Панель Управления', 'ru');
```

Slika 1.2: SQL stavki za vnos prevedenih besed v aplikacijo

1.2 Težave pri prevajanju znotraj ogrodja MX

Obstoječi način prevajanja, ki deluje znotraj ogrodja MX (več o ogrodju MX je podrobneje razdelano v poglavju 2.2) se srečuje s težavo, da ni enostaven za uporabo, saj je med različnimi zaslonskimi maskami potrebno veliko preklapljanja, čeprav bi bilo lahko vse na eni maski. Zaradi organizacije podatkovnega modela, je na treh različnih zaslonskih maskah enaka funkcionalnost, vendar se v ozadju vsaka funkcionalnost obnaša rahlo drugače. Na nekaterih zaslonskih maskah je potrebno nastaviti določene filtre, da najdemo želen nabor besed, vendar tej filtri pogosto prikazujejo napačne privzete vrednosti. V praksi se pogosto zgodi, da besede niso pravilno urejene po kategorijah, katere filtriramo in je potrebno uganiti, v katerem sklopu besed se nahaja iskana beseda. Prevajanje večjih količin besedil z obstoječim sistemom ni optimalno. Poleg tega se tudi prevajalci težje znajdejo, saj ne poznajo konteksta besedila, ki ga prevajajo. Pri tem načinu prevajanja obstaja tudi napaka, ki je nastala pri razvoju ogrodja MX. Do tega načina smejo dostopati le razvijalci ozirama ljudje z administratorskimi pravicami. Administratorskih pravic navadnim uporabnikom ne dodeljujemo, ker bi lahko zaradi neznanja uporabe ogrodja pokvarili delovanje informacijskega sistema.

1.3 Rešitev problema prevajanja in namen diplomskega dela

Naša naloga je, da omogočimo prevajalcu čim bolj preprost način prevajanja informacijskih sistemov v izbrani ciljni jezik, ki ne bo vključeval programerjev in njihove pomoči pri iskanju besed. Rešitev mora zmanjšati število napačnih prevodov (omenjeno v poglavju 1.1) in tudi pospešiti samo prevajanje. Omogočeno mora biti enostavno popravljanje napačnih prevodov, če se je prevajalec pri prevajanju zmotil.

Namen diplomskega dela je izdelati prototip prevajalne aplikacije v katero bo uporabnik vnesel spletno povezavo do zaslonske maske informacijskega sistema. Iz maske se bodo z algoritmom izluščile vse besede, ki jih je možno prevesti. Le-te se bodo pojavile v prevajalni aplikaciji. Uporabnik se bo nato odločil, v kateri jezik bo prevedel ponujeno besedo, ali pa bo popravil že obstoječi prevod.

Poglavje 2

Uporabljena orodja

2.1 Uporabljene tehnologije

V tem poglavju si bomo ogledali uporabljene spletne tehnologije, posamezne knjižnice in orodja, ki smo jih uporabili pri razvoju.

2.1.1 ASP.NET

Prototip prevajalnega orodja smo razvijali s pomočjo Visual Studia 2015. Visual Studio je integrirano razvojno okolje podjetja Microsoft. Namenjeno je razvoju aplikacij za operacijske sisteme Windows vendar se ga lahko uporabi tudi za razvoj na drugih platformah. Z njim je mogoče razvijati aplikacije na osnovi ogrodja *.NET* [3]. Ogrodje je odprtokodno in vključuje obsežno knjižnico orodij za razvoj uporabniških vmesnikov, podatkovni dostop, spletno komunikacijo, razvoj spletnih aplikacij itd.

Za razvoj prevajalne aplikacije smo izbrali ASP.NET ogrodje, ki temelji na *.NET* ogrodju. Za to smo se odločili zaradi predhodnih izkušenj z dotično tehnologijo. Razvilo jo je podjetje Microsoft, namenjena pa je za izdelavo spletnih strani. Ogrodje podpira različne programske module kot so *Web Forms*, *MVC*, *Web API* itd.

2.1.2 Firefox

Pri razvoju smo si pomagali tudi s spletnim brskalnikom Firefox. Med razvojem smo naleteli na težave s samodejnim vpisom v informacijski sistem. Pri tem smo uporabili razvojno konzolo Firefox, kjer smo uporabili orodje za zajemanje spletnega prometa.

2.1.3 SQL Server Express

Za vzpostavitev testnega okolja smo uporabili okolje SQL Server Express 2017. Okolje je razvilo podjetje Microsoft in služi za upravljanje relacijskih podatkovnih baz. S pomočjo okolja smo ustvarili podatkovni strežnik s podatkovno bazo, ki je bila potreba za pravilno delovanje ogrodja MX.

2.1.4 SQL Server Management Studio

Za SQL poizvedbe, ki jih potrebuje prevajalna aplikacija smo uporabili integrirano okolje SQL Managment Studio. Tudi to okolje je razvilo podjetje Microsoft, ki služi kot vmesnik med uporabnikom in podatkovnimi strežniki. Orodje ponuja grafične in skriptne urejevalnike, ki delujejo nad objekti podatkovnega strežnika [13]. V tem okolju smo uporabili Transact-SQL, ki je Microsoftova razširitev SQL-a. Uporablja se za poizvedovanje po relacijskih podatkovnih bazah in nudi dodatne možnosti kot njegov predhodnik SQL. [14].

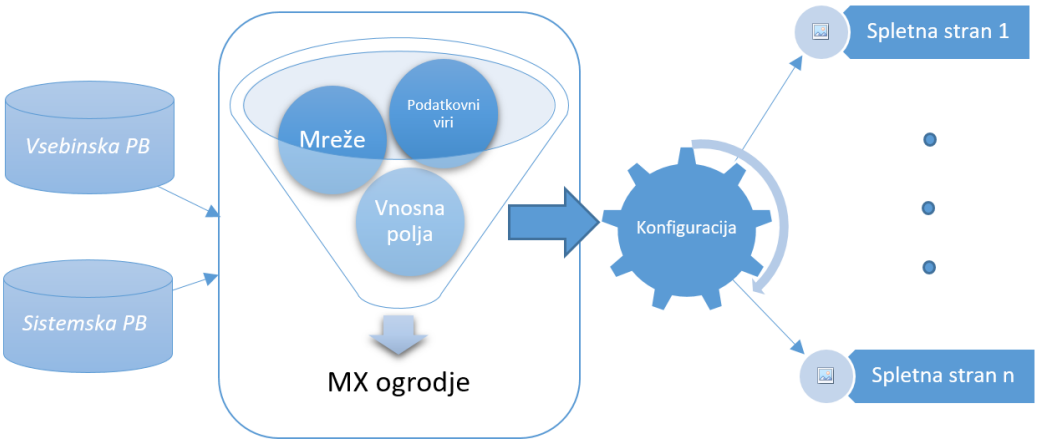
2.2 Ogrodje MX

Pri razvoju aplikacije smo uporabili ogrodje MX, ki ga je razvilo podjetje, v katerem sem zaposlen. Uporablja se za lažji razvoj informacijskih sitemov. Naša rešitev - prevajalna aplikacija je odvisna od tega ogrodja, zato ga bomo podrobneje predstavili. Eden izmed pogojev pri razvoju naše rešitve je ta, da ne smemo posegati v izvorno kodo ogrodja MX, vendar morajo nekatere funkcije vseeno delovati z njim.

Ogrodje MX nudi podporo pri razvoju proizvodnih informacijskih sistemov. Razvili so ga v *ASP.NET* tehnologiji in se močno prepleta z uporabo *SQL* procedur. V procedurah je določena vsa logika, kako se bo informacijski sistem obnašal, ogrodje MX pa pri tem skrbi za vizualizacijo informacijskega sistema. Njegov namen je lažje in hitrejše razvijanje informacijskih sistemov, saj avtomatizira nekatere standardne dele kode, ki se pojavljajo pri razvoju, kot so: urejanje podatkov, prikazovanje podatkov, grafično oblikovanje in postavitev strani itd. Povezavo podatkov s podatkovno bazo in njihov prikaz na spletni strani je možno narediti enostavno in zelo hitro, saj nam pri tem ni potrebno programirati, temveč le konfigurirati komponente za prikaz in vnos podatkov. To je zgolj le en primer uporabe ogrodja MX, ki se uporablja najbolj pogosto. Ogrodje MX nudi tudi obsežen nabor drugih ukazov, ki podpirajo delovanje informacijskih sistemov. Pri delovanju ogrodja MX je potrebno vedeti, da teče s pomočjo IIS, kjer se računalnik, na katerem je nameščen, obnaša kot strežnik, kar bo kasneje vplivalo na razvoj aplikacije.

Slika 2.1 prikazuje shemo ogrodja. Ogrodje ima navadno ločeno vsebinsko in sistemsko podatkovno bazo. Za ločitev se odločijo razvijalci sami. Če je vsebinska baza ločena od sistemske, je potrebno manj paziti pri nadgradnjah informacijskega sistema, saj se sistemske tabele pri tem ne spreminjajo, vendar pa je lahko ta način nadležen pri pisanju procedur, saj se je pri tem potrebno referencirati na drugo podatkovno bazo.V vsebinski podatkovni bazi so shranjeni podatki, ki jih stranka uporablja pri svoji vsakodnevni uporabi informacijskega sistema. V sistemski podatkovni bazi pa je shranjena konfi-

guracija sistema. H konfiguraciji spadajo tudi prevodi aplikacije, ki jih bomo morali kasneje urejati.



Slika 2.1: Shema ogrodja MX

Ogrodje MX služi za kreiranje spletnih strani in njihovo povezavo s podatki. Programiranje spletnih strani ni potrebno, saj se delovanje strani določi z konfiguracijo komponent za vnos in prikaz podatkov. Ker ogrodje nudi zgolj podporo konfiguraciji, je delovanje informacijskega sistema odvisno od logike v procedurah, ki jih pripravi programer. Logični del, ki določa obnašanje sistema je shranjen v vsebinski podatkovni bazi, medtem ko so v sistemski podatkovni bazi shranjene konfiguracije informacijskega sistema. Vsebinska podatkovna baza lahko zajema tudi sklop različnih podatkovnih baz na različnih zunanjih sistemih.

Poglavje 3

Organizacija tabel

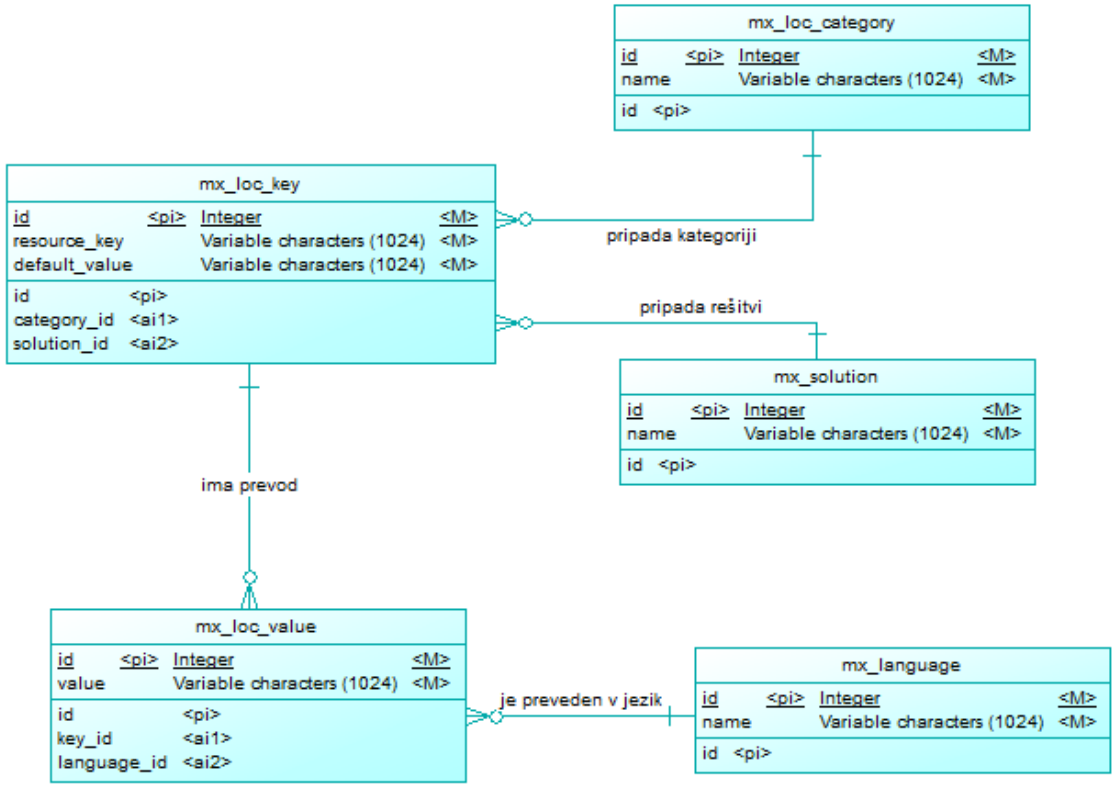
Besede, ki jih je možno prevajati, so shranjene v sistemski podatkovni bazi. Slika 3.1 prikazuje logični podatkovni model baze, v kateri so shranjeni podatki, ki jih bomo urejali. S tako organizacijo tabel je ena beseda lahko prevedena v več različnih jezikov, medtem ko ima en prevod natančno eno privzeto vrednost.

V tabeli `mx_loc_key` so shranjene vse besede, ki se uporabljajo v ogrodju MX. Vsaka beseda ima unikatno ime *resource_key* in privzeto vrednost *default_value*. Pri razvoju informacijskega sistema obstaja pravilo, da pri dodajanju nove besede v polja *resource_key* in *default_value* vedno vnesemo angleški zapis besede. To naredimo zaradi notranje logike ogrodja, da lahko le-to prikaže privzeto vrednost besede (ki je v angleščini), če prevod dotične besede še ne obstaja.

Ker so besede razporejene po kategorijah, v katerih se uporabljajo, sta v tabeli še dva tuja ključa *solution_id* in *category_id*. Ta dva atributa lahko uporabimo kot filter, da omejimo prevajanje besed na ožji nabor.

Tabela *mx_loc_value* je vmesna tabela med tabelama *mx_loc_key* in *mx_language*. V tabeli so shranjene vse prevedene besede tega sistema. Vsak prevod ima svoj enolični identifikator *id*, tuji ključ *key_id*, ki izhaja iz tabele *mx_loc_key* in ključ jezika *language_id*, ki izhaja iz tabele *mx_language*.

Tabela *mx_language* je šifrant jezikov. Ob namestitvi informacijskega sistema se vanj samodejno vneseta angleščina in slovenščina. Za tem sledita še dve tabeli s šifranti *mx_loc_category* in *mx_solution*, v katerih so šifranti kategorij, in programskih rešitev v katerih se določena beseda uporablja. Pri starem načinu prevajanja sta bili ti dve tabeli bolj pomembni, saj se je izvoz besed izvrševal na podlagi teh dveh šifrantov. Na sliki 3.1 so prikazani samo pomembnejši atributi, nepomembne atribute smo zaradi lažjega razumevanja namreč izpustili.



Slika 3.1: Logična organizacija tabel

Zaradi razumevanja, kako deluje notranja logika ogrodja MX, ki ustrezno prikaže prevod uporabniku, smo napisali poizvedbo, ki vrne pomembne podatke o posamezni besedi. Primer podatkov je na sliki 3.2. V prvem

stolpcu *id* nam vrne enolični identifikator besede, v stolpcu *resource_key* vidimo enolično ime besede. Za tem sledi privzeta vrednost *default_value*, ki se uporabi, če uporabnik uporablja angleški jezik ali pa prevod besede še ne obstaja. Naslednji je stolpec *translation*, v katerem je prevod besede, če slednji seveda obstaja. Ker je v tabeli *mx_loc_value* možno shraniti več prevodov za isto besedo, je potrebno imeti še podatek, kateremu jeziku pripada, kar je prikazano v stolpcu *language*. Poleg tega prikazujemo še atributa *category* in *solution_id*, ki ju uporabljamo zgolj za zožitev iskalnega nabora.

V primeru, da ima uporabnik nastavljen informacijski sistem na slovenski jezik, bo MX ogrodje samodejno vzelo vrednost iz polja *translation*. Če določen prevod ne obstaja oziroma ima polje *default_value* vrednost *NULL*, se bo vrednost vzela iz polja *default_value*. V primeru angleškega jezika poteka prevajanje nekoliko drugače. Če je informacijski sistem v angleščini, se bo vrednost vedno vzela iz polja *default_value*.

	id	resource_key	dafult_value	translation	language	category	solution_id
1	6952	customer_first_name	First name	Ime	Slovenski	Solution Content	7
2	6953	customer_last_name	Last name	Priimek	Slovenski	Solution Content	7
3	6956	customer_address	Address	NULL	NULL	Solution Content	7
4	6959	MasterData	MasterData	NULL	NULL	Solution Content	NULL
5	6948	DIPCustomer	Customer	NULL	NULL	Solution Content	6
6	6957	customer_telephone	Telephone	NULL	NULL	Solution Content	7

Slika 3.2: Primer podatkov za prevajanje

Poglavje 4

Funkcionalnosti aplikacije

Poleg osnovne funkcionalnosti prevajanja spletne strani v kontekstu smo prevajalcem dodali še dve dodatni funkcionalnosti, ki jih bodo morda potrebovali pri prevajanju. To so:

- Urejanje prevodov izven konteksta strani
- Urejanje šifranta jezikov

Urejanje prevodov v kontekstu strani je osnovno delovanje našega prototipa, vendar je zaradi začetnih nastavitev tega načina, če želi uporabnik prevesti le eno besedo, nekoliko zamudno. Zaradi tega smo prevajalcu omogočili tudi prevajanje izven konteksta strani. Slednji način je hitrejši, ker nima zakasnitve začetnih nastavitev, vendar prevajalcu tudi ne ponudi nobenega konteksta, zato slednji način lahko uporablja zgolj, ko prevajalec natanko ve kaj želi prevesti. V praksi je to le ena ali nekaj besed. V starem načinu prevajanja je v ogrodju MX imel dostop zgolj programer. Prevajalcu zaradi omejitve uporabniških pravic, ta način ni bil omogočen.

Pri tem je prevajalcem omogočeno tudi dodajanje novih jezikov v šifrant. Funkcionalnost je bila dodana, da lahko prevajalci samostojno dodajo ali uredijo nov tip jezika v šifrant. Prej je za dodajanje jezikov v šifrant moral skrbeti razvijalec.

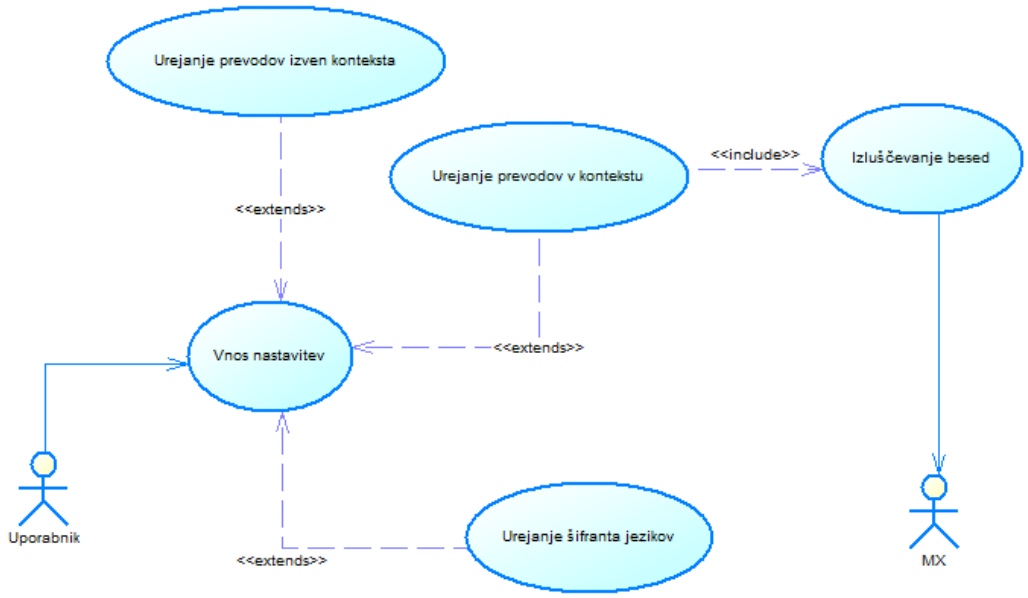
4.1 Urejanje prevodov v kontekstu strani

Urejanje prevodov v kontekstu strani poteka v naslednjih korakih, ki jih nadzira naš prototip:

1. Vnos začetnih nastavitev.
2. Pretvarjanje besed v format za izluščevanje.
3. Osvežitev ogrodja.
4. Dostop do strani.
5. Razčlenjevanje strani.
6. Odpravljanje napak.
7. Urejanje ali dodajanje prevodov.
8. Shranjevanje.

V prvem koraku mora uporabnik vnesti zahtevane nastavitve, ki so potrebne za pravilno delovanje prototipa. Obvezene nastavitve so: povezava do spletne strani, uporabniško ime in geslo informacijeskega sistema, direktorij kjer se nahaja namestitev informacijeskega sistema in povezava do podatkovne baze. Neobvezni nastavitvi sta še kategorija in programska rešitev, ki služita kot filter. Več o pomenu teh nastavitev bomo razdelali v nadaljnih poglavjih. Naslednji koraki od 2. do 5. se nato izvedejo samodejno in neodvisno od uporabnika. Preostali koraki (koraki od 6. do 8.) so odvisni od uporabnika in njegovega namena, kaj želi delati. Lahko se odloči za urejanje prevodov, če se pri urejanju zmoti, je shranjena zgodovina sprememb, ki mu lahko pomaga pri odpravljanju napak. Ko je prevajalec zadovoljen s svojim delom se lahko odloči za uveljavitev sprememb oziroma njihov preklic.

Na sliki 4.1 je diagram primerov uporabe, ki prikazuje funkcionalnosti našega prototipa. Uporabnik mora vnesti začetne nastavitve. Ko so nastavitve vnesene, se uporabnik lahko odloči med načinoma prevajanja aplikacije ali pa se odloči urejati šifrant jezikov. Če se odloči za urejanje prevodov v konteksu se izvede logika, ki iz ogrodja *MX* izlušči besede. V primeru urejanja izven konteksta pa uporabnik ureja prevode iz podatkovne baze.



Slika 4.1: Diagram primerov uporabe

4.1.1 Pretvarjanje besed v format za izluščevanje

Besede, ki jih je možno prevesti v informacijskem sistemu moramo označiti na unikaten način, tako da jih bo računalnik znal razlikovati med besedami, ki jih ne prevajamo. Ker gre za spletni informacijski sistem, so zaslonske maske shranjene v spletnih straneh, katere bomo prebrali s pomočjo prevajalne aplikacije. Besede, ki jih je možno prevesti označimo na unikaten način, tako da se te razlikujejo od ostalih besed. Z iskalnim algoritmom nato najdemo te besede in jih prikažemo uporabniku v aplikaciji.

Pretvarjanje v unikaten format naredimo z SQL poizvedbo, ki spremeni besede v podatkovni bazi. Ker so besede shranjene v dveh različnih tabelah (*mx_loc_key* ali *mx_loc_value*), sta za to potrebni dve podobni poizvedbi. Ogrodje MX bo ob nalaganju spletne strani pokazalo nove unikatne besede na strani.

Poizvedbi smo integrirali v aplikacijo, ker je bila v tem primeru to precej boljša rešitev od, sicer standardnih pristopov, kjer poizvedbe zapisujemo v bazo, v obliki procedur. Glavna prednost našega pristopa je prenosljivost aplikacije. Tako aplikacija za svoje delovanje ne potrebuje lastne baze, dodajanje tujih objektov (npr. naših procedur) v sistemsko bazo MX pa bi bilo strokovno napačno. Način integracije poizvedb v kodo je bolj primeren, saj omogoča lažje popravljanje kode ob spremembi poizvedbe, hkrati pa ne posegamo v podatkovno bazo informacijskega sistema.

Besede pretvorimo v unikaten format, katere bomo kasneje iskali z algoritmom za razčlenjevanje. Besede so v formatu $|X¤X¤X|$, znak X predstavlja poljubno vrednost. Prvi znak X nosi podatek o ključu besede, drugi znak podatek o ključu jezika, tretji znak pa nosi samo besedo. Ključ besede potrebujemo zaradi kasnejših poizvedb, da lahko določimo, katero besedo bomo urejali v podatkovni bazi. Jezik besede je prav tako pomemben, saj na podlagi le-tega ugotovimo, katero tabelo bomo morali urejati (*mx_loc_key* ali *mx_loc_value*). Čeprav je uporabniku že samoumevno, v katerem jeziku je beseda, ko jo prebere, računalniku ni, zato potrebuje tudi ključ jezika. Poleg teh dveh podatkov prikazujemo tudi besedo, ki se trenutno uporablja na

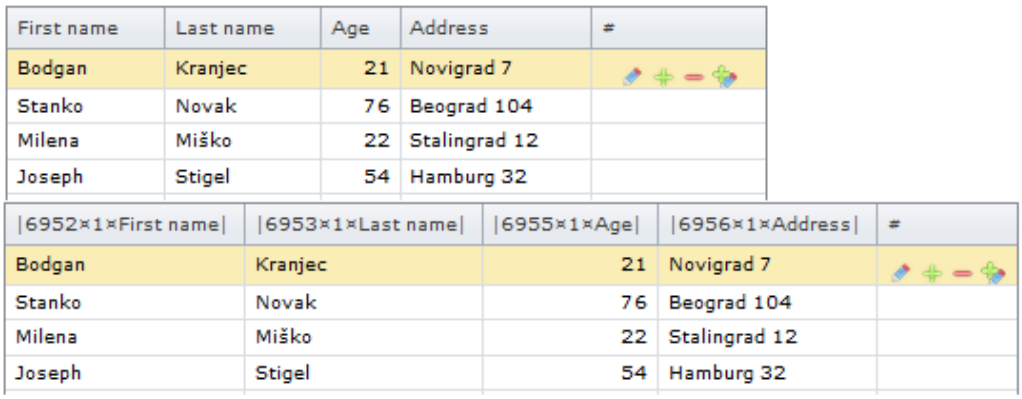
strani. Čeprav imamo že dovolj podatkov, da znamo določiti, katero besedo urejamo, jo uporabnik vseeno potrebuje zaradi konteksta, da lahko vidi, kje na strani se ta beseda uporablja.

Spodnja psevdopoizvedba prikazuje, kako deluje spreminjanje besed v nov format. Poleg besede se prilepi podatek o ključu besede in podatek o id-ju jezika. Dejanska poizvedba je nekoliko drugačna saj je zakodirana v našem prototipu. Povrnitev besed v prvotno vrednost deluje v obratni smeri. Procedura poišče besede spremenjene besede in jim izbriše dodane znake.

```
UPDATE dbo.mx_loc_key
SET default_value = '|'+id+'¤'+'1'+'¤'+default_value+'|'
WHERE
(solution_id = @solution_id OR @solution_id IS NULL)
AND (category_id = @category_id OR @category_id IS NULL)

UPDATE dbo.mx_loc_value
SET value = '|'+lv.key_id+'¤'+lv.language_id+'¤'+lv.value+'|'
FROM dbo.mx_loc_value lv
INNER JOIN dbo.mx_loc_key lk ON lk.id = lv.key_id
WHERE
(lk.solution_id = @solution_id OR @solution_id IS NULL)
AND (lk.category_id = @category_id OR @category_id IS NULL)
```

Slika 4.2 prikazuje uporabniški vmesnik s tabelaričnim seznam uporabnikov na informacijskem sistemu pred in po spreminjanju besed v novi format. Besede, ki jih je možno prevajati so v glavi mreže (vse ostalo so podatki, ki so neodvisni od lokalizacije). V prvi mreži je prvotna vrednost besed, v drugi pa je pretvorjena beseda v format za razčlenjevanje.



First name	Last name	Age	Address	#
Bodgan	Kranjec	21	Novigrad 7	
Stanko	Novak	76	Beograd 104	
Milena	Miško	22	Stalingrad 12	
Joseph	Stigel	54	Hamburg 32	

|6952×1×First name|	|6953×1×Last name|	|6955×1×Age|	|6956×1×Address|	#
Bodgan	Kranjec	21	Novigrad 7	
Stanko	Novak	76	Beograd 104	
Milena	Miško	22	Stalingrad 12	
Joseph	Stigel	54	Hamburg 32	

Slika 4.2: Besede pred in po pretvorbi v nov format

4.1.2 Osvežitev ogrodja

Ko zaženemo poizvedbo za pretvorbo besed v nov format, se vrednosti v podatkovni bazi posodobijo, na strani informacijskega sistema pa se spremembe še ne prikažejo. To se zgodi, ker ogrodje svoje podatke shranjuje v predpomnilnik, zaradi česar prikazuje stare vrednosti. Spraznitev predpomnilnika dosežemo tako, da ponovno zaženemo spletni strežnik IIS[12]. Na spletu obstaja mnogo programskih rešitev, kako ponovno zagnati spletni strežnik IIS, vendar v našem primeru nobena rešitev ni delovala dobro. Glavni problem je bil, da naš prototip ne bi tekel na istem strežniku, kjer tečejo instance MX-a. Tukaj so se pojavile različne težave s pravicami. Problem smo rešili tako, da smo izkoristili drug način delovanja informacijskega strežnika, ki ga bomo v nadaljevanju diplomskega dela podrobnejše razdelali.

Aplikacije spletnega strežnika IIS tečejo v zbirki aplikacij. Na eni zbirki, lahko teče tudi več strani. Vsaka stran je povezana z nekim direktorijem, kjer teče izvorna koda informacijskega sistema. Spraznitev MX-ovega predpomnilnika lahko osvežimo tako, da osvežimo ali zbirko aplikacij ali samo aplikacijo ali spletno stran, ki teče na aplikaciji. Slednja se samodejno osveži, če se spremeni katera-koli od datotek, ki je v direktoriju ali poddirektoriju kjer teče aplikacija. To smo izkoristili za posodobitev sistema, tako da smo programsko posodabljali *Last write time* atribut na trenutni čas datoteke *web.config*. V tej datoteki so shranjene vse globalne nastavitve, ki veljajo za celotno aplikacijo. Ta datoteka je tudi vedno prisotna v vseh ASP-jevih straneh, saj ob ustvaritvi projekta se ustvari samodejno. Ker vsebine datoteke ne spreminjamo, na aplikaciji ni nobenih posledic.

Pred razčlenjevanjem spletne strani mora uporabnik v prototip vnesti pot do direktorija, kjer teče ogrodje. Ker je ta direktorij vedno javno viden v omrežju, nimamo težav s pravicami in dostopom do te datoteke. Za dostop do tega direktorija izven omrežja se mora uporabiti VPN povezava.

Spodnja koda spremeni atribut *Last wite time* datoteki *web.config* na trenutni čas. V primeru, da pot ni bila podana, se metoda zaključi. Zaradi boljše robustnosti prototipa uporabniku dovolimo vnesti dva tipa poti. Pot je lahko vnesena z datoteko *web.config* ali brez nje.

```
public static void Restart(string path)
{
if (string.IsNullOrEmpty(path))
          return;

if (File.Exists(path))
          File.SetLastWriteTimeUtc(path, DateTime.UtcNow);
else if (Directory.Exists(path))
          File.SetLastWriteTimeUtc(path +
          @"\web.config", DateTime.UtcNow);
}
```

4.1.3 Dostop do strani

Pred prevajanjem besed s prototipom je potrebno vnesti povezavo do spletne strani, ki jo prevajamo. Ker je stran varovana z uporabniškim imenom in geslom, ju je pred prevajanjem potrebno vnesti, da lahko dostopamo do izvorne (HTML) kode strani. Izvorno kodo strani potrebujemo, ker kasneje z algoritmom iščemo besede, ki so na tej strani v unikatnem formatu [7, 8]. Na sliki 4.3 je primer, kako izgleda beseda, ki jo je mogoče prevesti v izvorni kodi. Da smo lahko dostopali do izvorne kode, smo morali napisati funkcijo, ki informacijskemu sistemu pošlje piškotek z uporabniškim imenom in geslom. Če je bila avtentikacija uspešna se vrne piškotek z izvorno kodo strani.

Slika 4.3: Beseda v izovrni kodi spletne strani

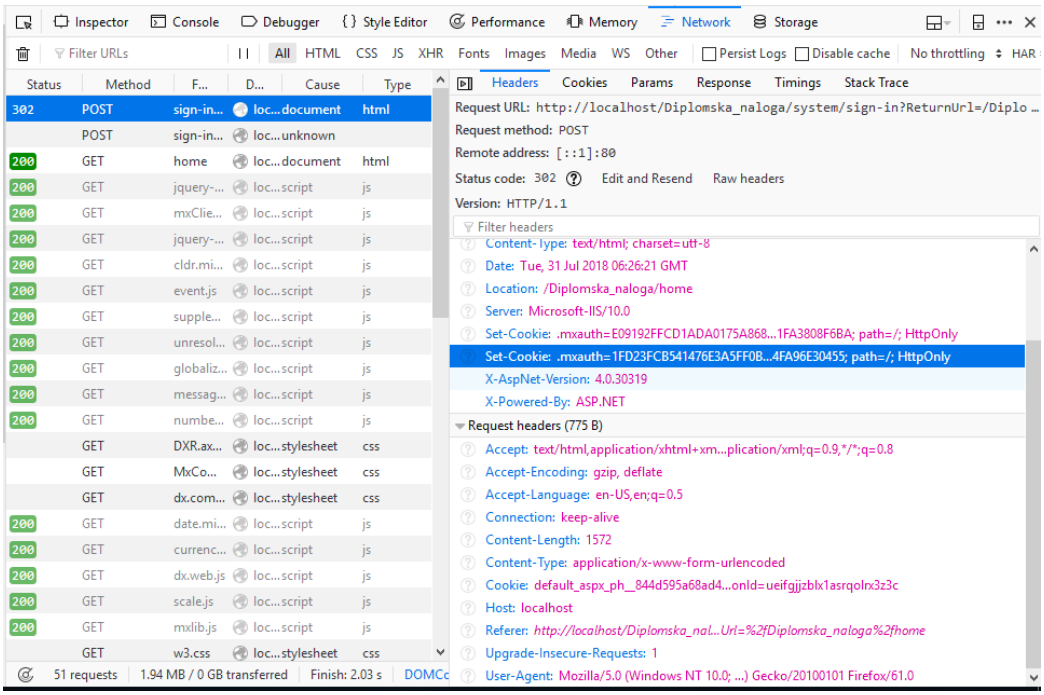
Metoda *getCookies* na spletno stran pošlje zahtevo za prijavo, v katero sta zapakirana uporabniško ime in geslo. Če je prijava uspešna, stran odgovori s piškotkom *.mxauth*, ki je pomemben pri branju izvorne kode strani. Če prijava ni bila uspešna, potem dotični piškotek ni prisoten. Prijavo se pošlje preko *View stat-a*.

```
private string[] getCookies(string userName,
 string password, string loginURL)
{
string parameters = string.Format(
       @"__VIEWSTATE={0}&ctl00$ph$textUserName$State={1}
       &ctl00$ph$textUserName={2}&ctl00$ph$textPassword$State={3}
       &ctl00$ph$textPassword={4}&ctl00$ph$ASPxButton1={5}");

HttpWebRequest requestHeader =
(HttpWebRequest)WebRequest.Create(loginURL);

requestHeader.ContentType = "application/x-www-form-urlencoded";
requestHeader.Method = "POST";
requestHeader.AllowAutoRedirect = false;
byte[] bytes = Encoding.ASCII.GetBytes(parameters);
requestHeader.ContentLength = bytes.Length;
using (Stream os = requestHeader.GetRequestStream())
       os.Write(bytes, 0, bytes.Length);
return requestHeader.GetResponse().Headers["Set-cookie"].Split(',');
 }
```

Če je bila avtentikacija s podanimi parametri uspešna, potem nam stran vrne prijavne piškotke, ki so na sliki 4.4. Prijavna piškotka sta prikazana pod poljem *Set-Cookie*.



Slika 4.4: Seznam piškotkov ov uspešni prijavi

Metoda *sendCookie* ustvari zahtevo po branju izvorne kode strani. Da se zahteva pravilno ustvari, je potrebno narediti pravilen klic s pravilno nastavljenimi piškotki. V prvem koraku metoda *sendCookie* s pomočjo metode *getCookies* pridobi seznam piškotkov spletne strani. Ob uspešni avtentikaciji uporabnika je v seznamu piškotek *.mxauth*, ki ga stran vrne. Ta piškotek zapakiramo v zahtevo po branju spletne strani. V primeru, da piškotka ni ali je le-ta neveljaven, zahteva ne bo odobrena in izvorne kode strani ne bomo dobili nazaj. Ko je zahteva uspešno poslana in potrjena s strani strežnika, nam slednji vrne izvorno kodo strani, ki jo nato preberemo.

```
public string sendCookie(string userName, string password,
       string pageURL, string loginURL)
{
try
{
       string pageSourceCode;
       string[] cookies = getCookies(userName, password, loginURL);

       HttpWebRequest request =
        (HttpWebRequest)WebRequest.Create(pageURL);
       request.Headers.Add("Cookie", cookies[1]);
       HttpWebResponse response =
        (HttpWebResponse)request.GetResponse();

       using (StreamReader sr = new StreamReader(
              request.GetResponse().GetResponseStream())
              )
       {
           pageSourceCode = sr.ReadToEnd();
       }
       return pageSourceCode;
}
catch (Exception e)
{
       Console.WriteLine(e.GetType().FullName);
      Console.WriteLine(e.Message);
       return null;
}
}
```

4.1.4 Razčlenjevanje strani

Po pridobljenem viru spletne strani, iščemo pojavitve znaka ¤. Pri pretvarjanju besed v format za razčlenjevanje smo uporabili ta znak kot ločilo med ključem besede in jezika ter samo besedo. Znak *currency sign* (¤) se na splošno uporablja za nedefinirane oznake valut [6]. Če ima podatek neko valuto in ta ni znana, lahko uporabimo ta znak. Ker je znak zelo redko uporabljen v informacijskem sistemu pa imamo definirane vse valute in ga tudi ne uporabljamo, lahko pričakujemo, da se pri prevajanju ne bo uporabil. V primeru, da bi se znak uporabil v neki besedi, potem je algoritem za razčlenjevanje nebi znal več pravilno razčleniti in bi jo bilo potrebno popraviti v podatkovni bazi.

Algoritem po viru strani išče ali se v besedi pojavi znak ¤. V tem primeru se preveri, ali je beseda v vzorcu $|X¤X¤X|$. Kljub temu da najdemo vzorec, še ne moremo vedeti, ali pripada besedi, ki je namenjena prevajanju. Prvi znak X nosi podatek o ključu besede, drugi znak podatek o ključu jezika, tretji znak pa nosi samo besedo. Če sta prva dva znaka numerična in tretji tekstovni, potem lahko sklepamo, da smo našli besedo za prevajanje.

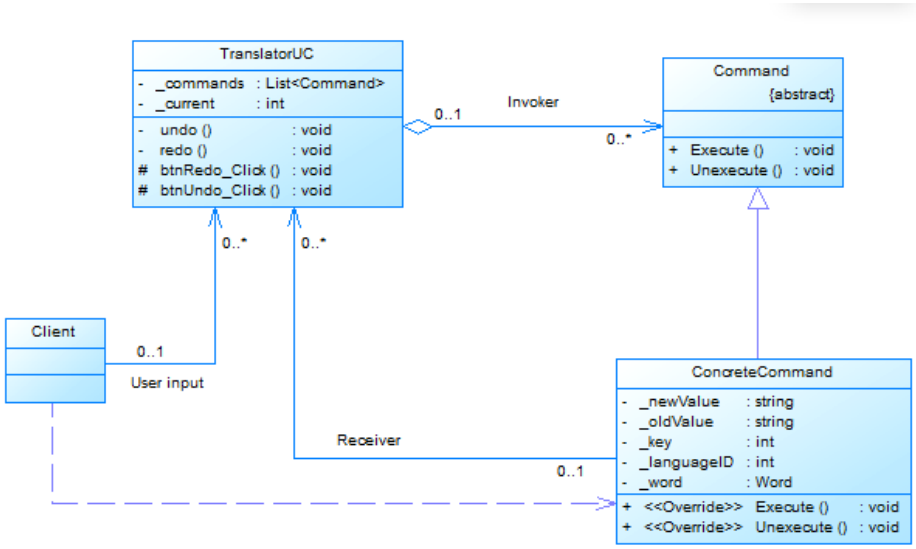
Najdene besede prikažemo uporabniku v ogrodju. Uporabnik se nato odloči, kaj želi narediti z njimi. Lahko posodobi obstoječo besedo ali pa ustvari nov prevod dotične besede v spustnem meniju. Program v ozadju poskrbi, da obstoječe besede posodablja, nove pa dodaja.

4.1.5 Odpravljanje napak

Odpravljanje napak pri prevajanju je bilo dodano, da se lahko prevajalcu omogoči enostavno povrnitev napačno prevedene besede v prejšnjo vrednost. V primeru, da je bila beseda večkrat prevedena so zabeležena vsa stanja besed za nazaj in jo je možno obnoviti na katero koli prejšnjo vrednost. Pomikanje med različnimi stanj besede naprej in nazaj deluje nad posamezno besedo. Omogočena je tudi funkcionalnost *Reset*, ki povrne vse besede v prvotno stanje

Na vsakem koraku prevajanja je uporabniku omogočeno, da prevajalec razveljavi vse svoje spremembe od začetka prevajanja. Razveljavi lahko vse spremembe za nazaj, torej se vrnete na začetek, ali pa se lahko premika po korakih naprej (*redo*) in nazaj (*undo*). To smo rešili s pomočjo načrtovalskega vzorca *Command*.

Na sliki 4.5 je razredni diagram, ki prikazuje shemo načrtovalskega vzorca *Command*. Razred *Command* je abstraktni razred, ki ponudi dve metodi *Execute* in *Unexecute*. Logika teh dveh metod je definirana v razredu *ConcreteCommand*. Metodi znata posodobiti besedo tako na njeno prejšnjo vrednost kot tudi na naslednjo vrednost, če smo besedo že kdaj povrnili v prejšnje stanje. Spodnji diagram se rahlo razlikuje od standardnih diagramov načrtovalskega vzorca *Command*, saj smo si ga prilagodili za svoje potrebe. Razred *TranslatorUC* podpira (povezava *Invoker*) klic funkcij *Execute* in *Unexecute*, hkrati pa tudi ve, kako izvesti operaciji (povezava *Receiver*). Razred *Client* predstavlja uporabnika, ki dejansko izvrši klic metod. [5, 4, 2]



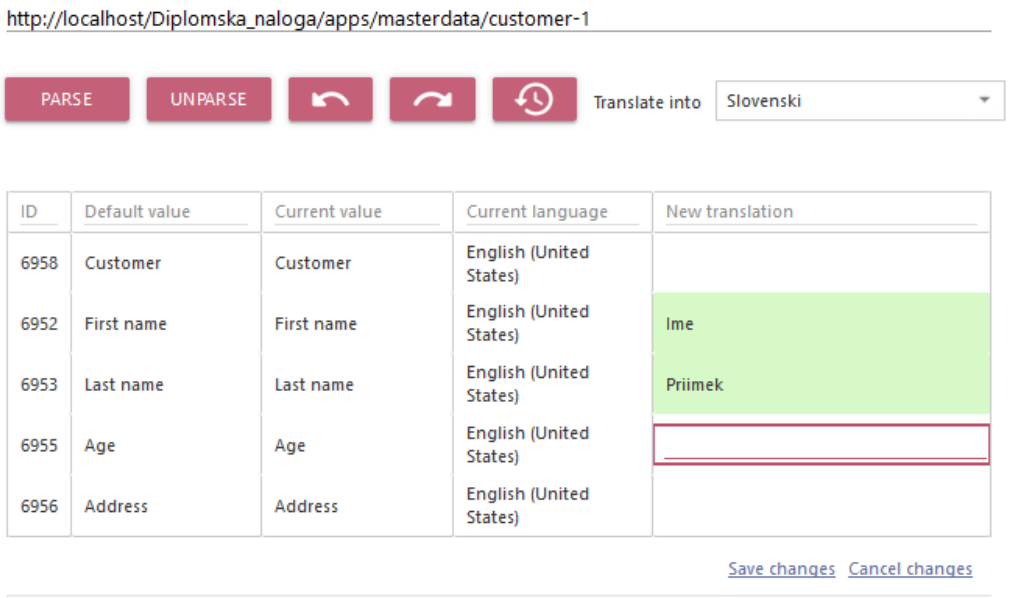
Slika 4.5: Razredni diagram načrtovalskega vzorca Command

4.1.6 Urejanje in shranjevanje

Ko uporabnik vnese začetne nastavitve in klikne na gumb *Parse* se iz zaslonske maske vnešene strani izluščijo vse besede, ki jih je možno prevesti. Seznam teh besed je na sliki 4.6. Seznam omogoča, da uporabnik, preden potrdi shranjevanje, vnese več prevodov besed hkrati. Če bi uporabnik moral potrjevati vsako besedo posebej, bi prišlo do velike zakasnitve, saj se ob vsakem shranjevanju ogrodje MX ponovno osveži. Ker se pri osveževanju ponovno zažene spletni strežnik IIS, lahko to traja tudi dlje časa.

V spustnem meniju *Translate into* uporabnik izbere v kateri jezik želi prevesti besedo. V njem so prikazani vsi jeziki iz šifrantov. Če je pri urejanju polje *Current language* enako vrednosti, ki je v spustnem seznamu (npr. oba sta v angleščini), potem bo prišlo posodabljanja trenutne besede. Če je vrednost polja *Current language* in spustnega seznama različna (npr. eden je v angleščini, drugi je v slovenščini), potem bo dodan nov prevod. Če prevod že obstaja, pa bo prišlo do posodobitve.

Po končanem prevajanju mora uporabnik pritisniti na gumb *Unparse*, da pretvori besede iz formata za prevajanje nazaj v prvotno obliko. V primeru napačnega prevoda se lahko pomika s puščicama naprej ali nazaj po zgodovini svojih prevodov. Če želi povrniti vse besede, mora pritisniti gumb *Revert*

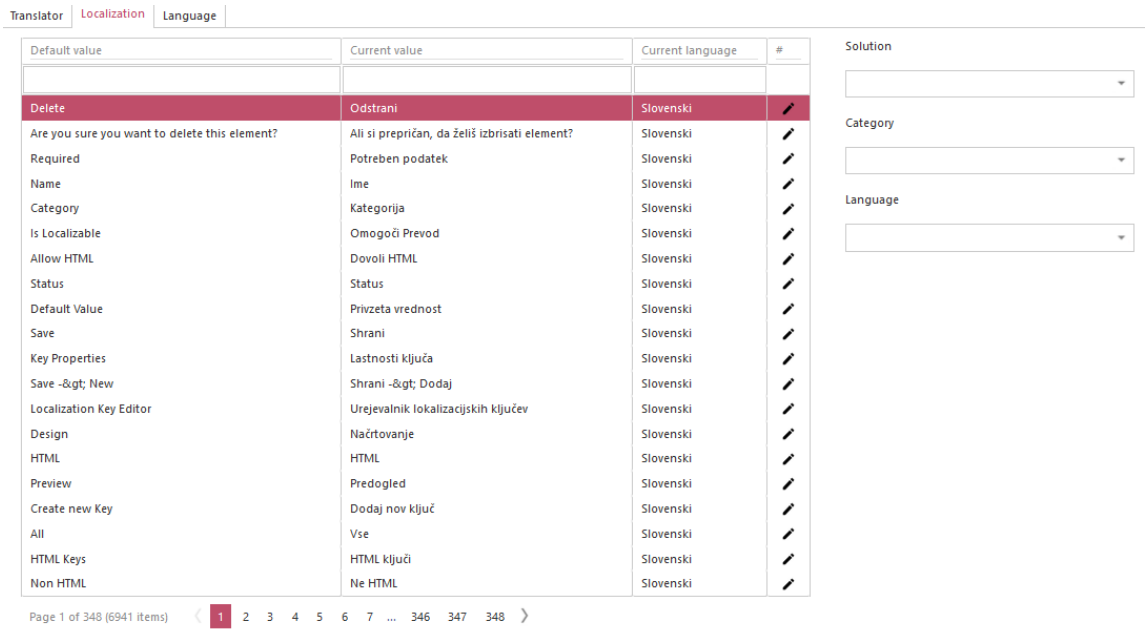


Slika 4.6: Seznam besed za prevajanje

4.2 Urejanje prevodov izven konteksta strani

Uporabniku je omogočeno tudi prevajanje brez konteksta strani. Ta način je ogrodje MX podpiralo že prej, vendar so do njega imeli dostop le razvijalci. Poleg tega je bil tudi nekoliko neroden za uporabo, saj uporabniško ni bil najbolje načrtovan. Naš način je preprostejši za uporabo in omogoča hitro preklapljanje med urejanjem v ali izven konteksta. Uporaba tega načina je bolj primerna v primerih, ko je potrebno urediti le nekaj besed, saj za svoje delovanje potrebuje manj začetnih nastavitev. Uporabnik mora pred tem poznati vsaj nekatere podatke o besedi. Da besedo najde, si lahko pomaga z različnimi filtri, ki zožijo nabor najdenih besed.

Pri prevajanju je omogočeno tudi urejanje šifranta jezikov. Jezike je mogoče dodajati, urejati in tudi brisati, če le-ti še niso v uporabi. Uporabniški vmesnik skrbi za pravilen format vnosa, v nasprotnem primeru uporabniku ne dopusti vnesti željene vrednosti in ga ustavi. Na sliki 4.7 je zaslonska maska izven kontekstnega urejanja.



Slika 4.7: Zaslon izven kontekstnega prevajanja

Poglavje 5

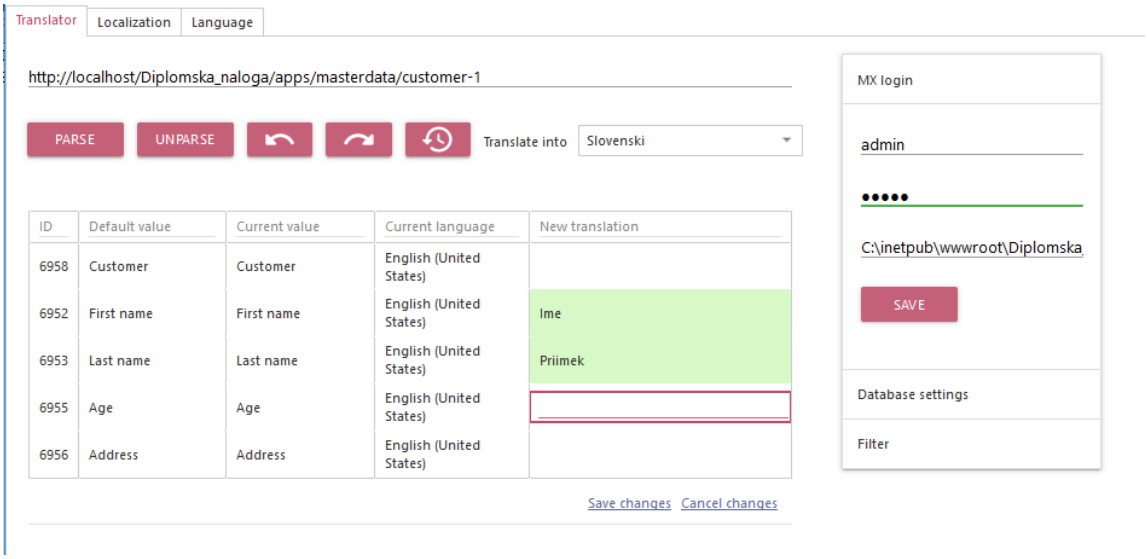
Uporabniški vmesnik prototipa

Uporabniški vmesnik smo zgradili s standardnimi *ASP*-jevimi komponentami, ki so že vključene v razvojno okolje Visual studia. Poleg standardnih komponent smo uporabil tudi plačljive *ASPX* , ki jih je razvilo podjetje *DevExpress*. Licenco za komponente nam je priskrbelo podjetje, s katerim smo sodelovali. Čeprav so te komponente nadstandardne in omogočajo več funkcionalnosti od standardnih smo imeli z njimi nekaj težav, saj je bila licenca stara. Glavna težava je bila, da je bilo nemogoče spreminjati barvo komponent na poljubno barvo, saj tedanja verzija tega ni omogočala, poleg tega se je pojavila tudi težava pri povezovanju podatkov v mreže [11]. V uporabljeni verziji ni bilo možno povezati podatkov v mreže, če so podatki izhajali iz načrtovalskega vzorca *Abstract factory*. V novejših verzijah so to napako odpravili [1].

Ker so bile tako *ASP-jeve* kot tudi *ASPX-eve* komponente zelo slabo grafično oblikovane, smo za njihov boljši izgled napisali tudi CSS. V osnovi smo izhajali iz odprtokodnega CSS-a, ki ga je razvilo podjetje Google. Odprtokodni CSS je deloval za standardne komponente deloval zelo dobro, nadstandardne pa so bile drugače oblikovane in CSS jih je popačil. Težavo smo rešili tako, da smo uporabili manjše prilagoditve izvornega CSS.

Pri postavljanju spletne strani smo se poskusili držati čim več Nielsonovih principov, ki se uporabljajo kot smernice za dobro načrtovanje uporabniških

vmesnikov. Na vseh zavihkih smo uporabljali konsistenten izgled, ohranili smo tudi podobnost s stranmi informacijskih sistemov, ki jih podjetje razvija. Držali smo se minimalističnega načrtovanja in uporabniku poskusili narediti čim bolj preprosto spletno stran, da jo bo ta že o prvem stiku z njo znal uporabljati. Uporabili smo čim več ikon, ki ponazarjajo, delovanje stvari [9]. V aplikaciji smo uporabili neplačljive ikone, ki jih je razvil Google. Na sliki 5.1 je prikazana glavna stran za prevajanje v kontekstu strani.



Slika 5.1: Uporabniški vmesnik

V prvem besedilnem vnosnem polju je povezava do spletne strani, ki jo bo prevajalec želel prevajati. Pod vnosnim poljem je pet gumbov, ki sprožijo ukaze o pretvorbi besed v format za prevajanje in nazaj. Ostali trije gumbi so za *Undo*, *Redo* in *Revert* prevedenih besed. V spustnem meniju uporabnik izbere jezik, v katerega želi besedo prevesti ali popraviti. Na desni strani aplikacije je dinamično okno, ki zahteva podatke o uporabniškem imenu in geslu do informacijskega sistema, zahteva povezavo do podatkovne baze, neobvezno je na voljo še filter za zožitev nabora besed. Okno se dinamično odpira in zapira glede na to, ali je uporabnik pravilno vnesel vse zahtevane podatke. V sredini sledi seznam besed, ki smo jih pridobili s spletne strani.

V prvem stolpcu je *ID* besede, ki je koristen, če želimo besedo urediti izven konteksta ali popraviti v bazi. V stolpcu *Default value* je privzeta vrednost besede, ki se uporabi, če ni prevoda. Stolpec *Current value* kaže besedo, ki je trenutno na strani. To besedo uporabnik potrebuje zaradi konteksta, da vidi, kje se uporablja. V stolpcu *Current language* je ime jezika v katerem je beseda na strani. V vnosno polje *New translation* lahko vpišemo več novih besed hkrati. Glede na to, ali smo zadovoljni s prevodom, lahko shranimo ali prekličemo spremembe.

Poglavje 6

Sklepne ugotovitve

Cilj diplomske naloge je bil izdelati aplikacijo, ki bo omogočala boljše prevajanje besed v okviru razvoja informacijske rešitve prilagojene določenemu jezikovnemu področju, tako da bodo le-te v kontekstu s spletno stranjo. Nalogo smo rešili tako, da smo naredil prototip aplikacije, ki razbere vse možne besede, ki se jih lahko ureja, uporabniku pa omogoči njihovo urejanje.

V okviru diplomske naloge smo dosegli vse zastavljene cilje. Z uporabo aplikacije se je število napačnih prevodov zmanjšalo, pri prevajanju pa so zdaj potrebni le prevajalci, razvijalcev pa pri tem delu ne potrebujemo več. Nov način prevajanja je pri večjih količinah besedil veliko hitrejši, medtem ko je pri manjših ta razlika manj opazna, v nekaterih primerih pa počasnejša. Počasnejša je, če želimo urediti le eno ali dve besedi, in sicer zaradi zakasnitve začetnih nastavitev, ki jih mora uporabnik nastaviti pred delom. Pokazala se je tudi pomanjkljivost, da aplikacija ni bil tako intuitivna, kot smo si želeli. Novi uporabniki se lažje znajdejo, če jih kdo pouči, kako se aplikacija uporablja.

Prototip je bil razvit tako, da lahko deluje za vse verzije informacijskih sistemov, ki jih je podjetje do sedaj razvilo. Za konec smo naredili kratko analizo, ki je pokazala, da je razviti prototip uspešen. V analizi smo testirali ali aplikacija deluje pravilno. Pri tem smo dodajali različne prevode in urejali šifrant jezikov ter preverili smo kaj se zgodi, če zaslonska maska vsebuje:

- večkrat isti prevod na eni strani,
- dva ali več različnih jezikov na strani in
- uporabo posebnih znakov in šumnikov.

Pri testiranju posebnih znakov smo ugotovili, da ne moremo prevesti besede, ki v vsebini vsebuje znak ¤, saj se ta uporablja pri pretvarjanju besede v unikaten format (opisano v poglavju 4.1.1). To težavo smo predvideli že prej, vendar smo jo zanemarili, saj tega znaka ne uporabljamo. V vseth ostalih kominacijah je aplikacija delovala pravilno. Pomanjkljivosti smo opazili tudi pri odzivnosti aplikacije. Aplikacija včasih deluje počasi pri shranjevanju besed, saj se pri tem ponovno zažene spletni strežnik IIS. Ta problem smo opazili že tekom razvoja, zato se urejene besede ne shranijo takoj po urejanju, ampak jih shranjujemo v predpomnilniku. Ko prevajalec zaključi urejanje sproži akcijo za shranjevanje, ki v podatkovno bazo shrani vse spremenjene besede.

Literatura

[1] AbstractFactory. Abstractfactory. Dosegljivo: https://www.c-sharpcorner.com/article/abstract-factory-design-pattern-in-c-sharp/. [Dostopano: 11. 8. 2018].

[2] Alexander, C. (1995). *Design Patterns*. O'Reilly.

[3] ASP.NET. Asp.net. Dosegljivo: https://en.wikipedia.org/wiki/ASP.NET. [Dostopano: 5. 1. 2019].

[4] Bishop, J. (2008). *C# 3.0 Design Patterns*. O'Reilly.

[5] Command. Command design pattern. Dosegljivo: https://www.dofactory.com/net/command-design-pattern. [Dostopano: 14. 9. 2018].

[6] CurrencySign. Currencysign. Dosegljivo: https://en.wikipedia.org/wiki/Currency_symbol. [Dostopano: 25. 2. 2019].

[7] HttpWebRequest1. Http web request 1. Dosegljivo: https://stackoverflow.com/questions/16642196/get-html-code-from-website-in-c-sharp. [Dostopano: 4. 8. 2018].

[8] HttpWebRequest2. Http web request 2. Dosegljivo: https://stackoverflow.com/questions/975426/how-to-programmatically-log-in-to-a-website-to-screenscape. [Dostopano: 4. 8. 2018].

[9] Icons. Google icons. Dosegljivo: https://material.io/tools/icons/?style=baseline. [Dostopano: 12. 9. 2018].

[10] Kontekst. Language integrated query. Dosegljivo: https://fran.si/iskanje?View=1&Query=kontekst. [Dostopano: 10. 5. 2019].

[11] OldTheme. To old version. Dosegljivo: https://www.devexpress.com/Support/Center/Question/Details/T563180/theme-base-color-not-changing-all-elements. [Dostopano: 20. 8. 2018].

[12] Recycling. Recycling asp.net application. Dosegljivo: https://weblog.west-wind.com/posts/2006/Oct/08/Recycling-an-ASPNET-Application-from-within. [Dostopano: 13. 8. 2018].

[13] SQL. Sql. Dosegljivo: https://en.wikipedia.org/wiki/SQL. [Dostopano: 5. 1. 2019].

[14] Transact-SQL. Transact-sql. Dosegljivo: https://en.wikipedia.org/wiki/Transact-SQL. [Dostopano: 5. 1. 2019].