UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Uroš Modlic

RAZVOJ SPLETNE APLIKACIJE Z ORODJEM GOOGLE WEB TOOLKIT

DIPLOMSKO DELO

NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor:

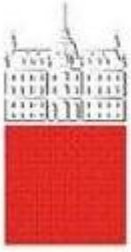
doc. dr. Marko Bajec

Ljubljana, 2009

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
fax: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si



Št. naloge: 00414/2008
Datum: 15.10.2008

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

Kandidat: **UROŠ MODLIC**

Naslov: **RAZVOJ SPLETNE APLIKACIJE Z ORODJEM GOOGLE WEB TOOLKIT**

WEB APPLICATION DEVELOPMENT USING GOOGLE WEB TOOLKIT

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

Google Web Toolkit je razvojno okolje podjetja Google, namenjeno razvoju spletnih aplikacij. V diplomski nalogi omenjeno razvojno okolje preučite in opišite. Na primeru predstavite način razvoja spletnih aplikacij v okolju Google Web Toolkit. Pri razvoju sledite smernicam objektno usmerjenega razvoja.

Mentor:

doc. dr. Marko Bajec

Dekan:

prof. dr. Franc Solina

Original izdane teme diplomskega dela (dvigni v referatu).

Izpolnjena izjava o avtorstvu in soglasje, ki omogoča objavo elektronske oblike v zbirki 'Dela FRI' (obrazec je na spletu: Izjava o avtorstvu diplomskega dela.doc).

Zahvala

Zahvaljujem se svojemu mentorju doc. dr. Marku Bajcu za koordinacijo diplomske naloge in strokovno pomoč.

Za skrb in vzpodbudo se zahvaljujem svojim staršem.

Kazalo

Seznam uporabljenih kratic in simbolov

GWT – Google Web Toolkit – Googlovo orodje za razvoj spletnih aplikacij

AJAX – Asynchronous JavaScript And XML – asinhroni JavaScript in XML

HTML – HyperText Markup Language – standarden jezik za razvoj spletnih strani

XML – Extensible Markup Language – razširljiv označevalni jezik

CSS – Cascading Style Sheets – stilne predloge za oblikovanje spletnih strani

JavaScript – skriptni programski jezik za razvoj interaktivnih spletnih strani

DOM – Document Object Model – objektni model sestave dokumenta spletne strani

Java – objekti programski jezik

RPC – Remote procedure call – klici oddaljenih procedur

RIA – Rich Internet applications – bogate spletne aplikacije

API – Application programming interface – aplikacijski programski vmesnik

UML – Unified Modeling Language – jezik za objektno modeliranje

JDBC – Java Database Connectivity – standard za baze podatkov pod Javo

SQL – Structured Query Language – strukturiran povpraševalni jezik za delo s pod. bazami

URL – Uniform Resource Locator – internetni naslov, na katerem se nahaja vsebina

PHP – PHP Hypertext Preprocessor – programski jezik za razvoj dinamičnih spletnih strani

Povzetek

Diplomska naloga zajema izdelavo spletnega portala in spletne aplikacije za urejanje vsebine z uporabo orodja Google Web Toolkit (GWT). Uporaba omenjenega orodja olajša razvoj Ajax aplikacije s tem, ker se nam pri razvijanju portala in aplikacije ni potrebno ukvarjati z nekompatibilnostjo spletnih brskalnikov in se lahko v celoti posvetimo razvoju v Java okolju.

Orodje GWT omogoča enostavno izdelavo spletnih aplikacij v samo enem jeziku, v Javi. V osnovi so dinamične spletne aplikacije napisane v več različnih jezikih - na strani klienta v označevalnem jeziku HTML za izdelavo spletnih strani in v programskem jeziku JavaScript, ki služi za validacijo in manipulacijo objektov v modelu objektov dokumenta (DOM). Na žalost razvijalcev spletnih aplikacij se je s pojavljanjem novih in novih spletnih brskalnikov, kii so na voljo uporabnikom spleta, začelo pojavljati različno interpretiranje JavaScript kode od brskalnika do brskalnika. Tako so majhne razlike pripeljale do tega, da so morali razvijalci vedno več časa vlagati v to, da so prilagodili spletno aplikacijo za delovanje na vseh različnih brskalnikih, namesto da bi čas vlagali v samo razvijanje.

Tudi na strani strežnika, kjer je možen razvoj v različnih programskih jezikih, z GWT razvijamo aplikacijo samo v Javi. Tako lahko kodo za celotno spletno aplikacijo napišemo samo v enem programskem jeziku, kar poenostavi celotno izdelavo.

Ključne besede

spletna aplikacija, Google Web Toolkit, Ajax, Java

Summary

This work includes making of a web portal and web application for content editing using the Google Web Toolkit (GWT). The use of GWT can facilitate the development of Ajax application because we are not required to deal with incompatibility between web browsers. we can be fully dedicated to the web application development in the Java environment.

With GWT we can make whole web application in just one language, Java. In principal, dynamic web applications are written in several different languages - with markup language HTML and scripting language JavaScript on client side, which serves to validate and handling Document Object Model (DOM). Unfortunately, with emergence of new and new web browsers, a different interpretation of the JavaScript code from browser to browser began to emerge. Small differences have led to this, that the developers needed to invest more time in adaptation of the web application to operate in all the different browsers, rather than just invest their time in the development.

On the server side, where it is possible to develop a web application in different programming languages, with GWT, we develop web applicationjust in Java. The code for the entire web application with GWT is only in one programming language, which simplifies the entire production.

Keywords

web application, Google Web Toolkit, Ajax, Java

1. Uvod

Spletne aplikacije v današnjih časih postajajo vedno bolj napredne in uporabnikom nudijo vedno več funkcionalnosti. Prav tako je delo z njimi lažje, saj za delo z njimi potrebujemo le spletni brskalnik in mrežno, oziroma internetno povezavo, tako da odpade nameščanje programske opreme odjemalca na uporabnikov računalnik. To pa je za razvijalce spletnih aplikacij prednost predvsem zaradi vzdrževanja aplikacije. Pri taki strukturi odjemalec - strežnik, kjer se sama aplikacija nahaja na strežniku, ni potrebno vzdrževati vsakega odjemalca posamezno, temveč vzdržujemo samo aplikacijo na strežniku.

S transformacijo namiznih aplikacij na spletne aplikacije tako naši računalniki postajajo nekakšni terminali, preko katerih dostopamo do aplikacije na strežniku. A se tukaj pojavijo problemi zaradi različnih izvajanj kode spletne aplikacije v brskalniku, ki se prenese s strežnika in se izvede na odjemalcu. Ta služi v večji meri za prenašanje in sinhronizacijo podatkov med strežnikom in odjemalcem.

Z razvojem sodobnih, objektno-usmerjenih jezikov, in z bolj pogosto uporabo tehnologije Ajax v spletnih aplikacijah, so začela nastajati nova orodja, med njimi tudi GWT.

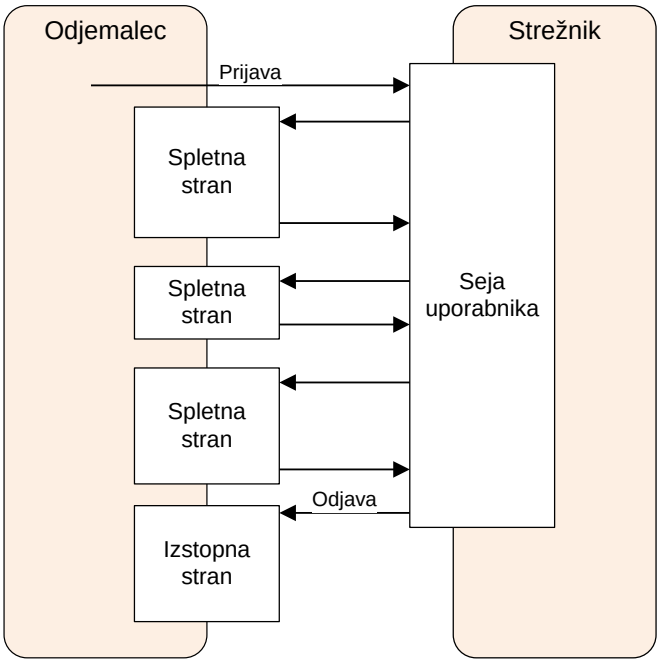
2. Uporaba GWT v spletnih aplikacijah

2.1. Splošno o spletnih aplikacijah

Da bi razumeli, zakaj pravzaprav potrebujemo orodje GWT pri gradnji spletnih aplikacij, si najprej poglejmo njihovo delovanje ter razvoj. Zanima nas predvsem njihova zgradba in delovanje s strani uporabnikov, kot tudi razvijalcev.

2.2. Klasične spletne aplikacije

1. Delovanje klasičnih spletnih aplikacij deluje na principu pošiljanja celotnih spletnih strani s strani strežnika k odjemalcu, ta pa nazaj v smeri strežnika pošilja podatke, katere je uporabnik vnesel v spletno stran. Vneseni podatki se na strežniku ustrezno obdelajo, odjemalcu pa se pošlje povratna informacija o izidu obdelave, ponovno v obliki celotne spletne strani. Tako se poleg podatkov vsakič znova prenašajo tudi deli spletne strani, ki pa za sam prenos podatkov med strežnikom in odjemalcem ter za pravilno delovanje spletne aplikacije pravzaprav sploh ne potrebujemo.

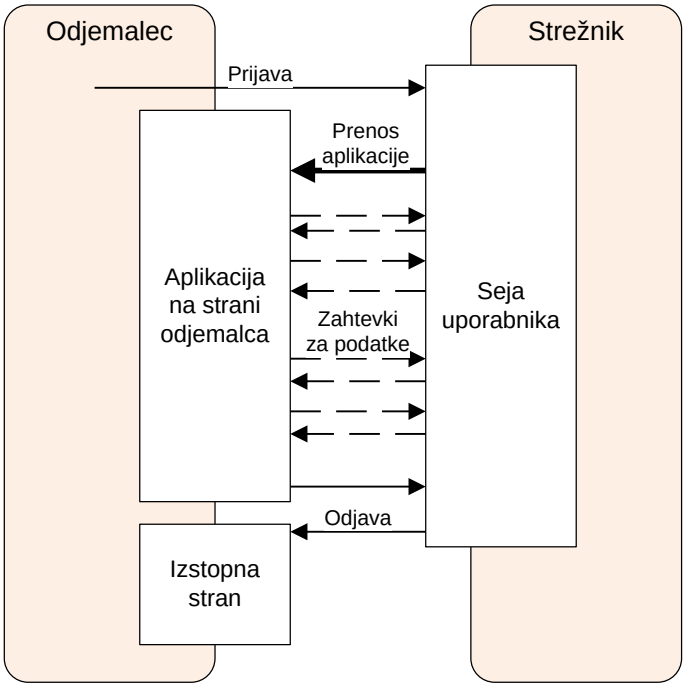


Slika 1. Interakcija med odjemalcem in strežnikom pri klasični spletni aplikaciji.

Z razvojem spletnih brskalnikov in izrabljanjem njihovih zmožnosti se začne povečevati uporaba Ajaxa v spletnih aplikacijah.

2.3. Ajax in bogate spletne aplikacije

Ime Ajax je kratica za dve tehnologiji, ki se uporabljata pri razvoju bogatih spletnih aplikacij – asinhrona uporaba JavaScripta in XMLja. Take vrste spletnih aplikacij so vse bolj podobne namiznim aplikacijam, saj je njihovo delovanje v primerjavi s klasičnimi spletnimi aplikacijami bolj interaktivno in se hitreje odzivajo.



Slika 2. Interakcija med odjemalcem in strežnikom pri uporabi tehnologije Ajax.

Hitrejša odzivnost spletnih aplikacij, narejenih s tehnologijo Ajax, je v asinhroni komunikaciji med odjemalcem in strežnikom. Uporabniški vmesnik se med tem, ko čaka na odgovor strežnika, še vedno odziva, saj vsaka komunikacija med njima teče v novi niti. Tako ima uporabnik v takšnem načinu delovanja prikazano v brskalniku samo eno stran, spreminjajo pa se le določeni gradniki spletne strani.

Pošiljanje podatkov med odjemalcem in strežnikom poteka s serializacijo objektov za zahtevek. Ti se pošljejo strežniku, ki jih deserializira in obdela, ter pošlje odgovor na isti način – s serializacijo ter deserializacijo na odjemalcu. Kljub možnosti počasnega in nezanesljivega postopka komunikacije preko omrežja, je zaradi asinhrone komunikacije in stalne odzivnosti spletne aplikacije uporabniška izkušnja v tem primeru boljša.

2.4. Opredelitev Ajaxa s štirimi koncepti

2.4.1. Spletni brskalnik hrani aplikacijo in ne vsebine

Klasične spletne aplikacije, ki prikazujejo samo vsebino, ne hranijo informacij o tem, kje v toku dogodkov se nahaja uporabnik. Te informacije se hranijo na strežniku, na katerem se tudi vodi uporabnikova seja, ki služi za vodenje uporabnika čez celoten tok aplikacije. Brskalnik v takem primeru samo prikazuje spletne strani – HTML dokument z vsebino in obliko. Z vsakim klikom uporabnika na spletni strani se pošlje zahtevek strežniku, ki pošlje novo stran, staro pa brskalnik zavrže.

Ob začetku dela z Ajax spletno aplikacijo pa strežnik pošlje odjemalcu bolj kompleksno spletno stran, ki je v veliki večini sestavljena iz JavaScript kode, ki bo poskrbela za spreminjanje posameznih delov spletne strani in za odzivanje na uporabnikove akcije. Takšna spletna stran je v celotnem času seje pri odjemalcu. V tem primeru se logika za vodenje čez tok aplikacije porazdeli med strežnikom in odjemalcem, odvisno od tega, kako je spletna aplikacija zgrajena. Za primer, vsebina spletne košarice v spletni trgovini se lahko skozi čas nakupa shranjuje pri odjemalcu, ob zaključku postopka pa se prenese in shrani v podatkovno bazo na strežniku.

2.4.2. Spletni strežnik pošilja podatke in ne vsebine

Pri klasičnih spletnih aplikacijah se za vsako spremembo, ki jo naredi uporabnik na uporabniškem vmesniku, pošlje celotna vsebina in oblika spletne strani k odjemalcu. Tako se, za recimo majhno spremembo v spletni trgovini, ko uporabnik doda izdelek v košarico, pošilja celotna vsebina ter oblika spletne strani k odjemalcu, čeprav je sprememba le majhen podatek o vsebini uporabnikove košarice.

Bolj učinkovite so v takšnih primerih Ajax aplikacije, saj pošljejo strežniku samo podatke o spremembi, v našem primeru dodan izdelek v košarico. Tako je velikost odgovora odjemalcu veliko manjša, saj strežnik ne vrne celotne spletne strani v obliki HTML dokumenta, temveč podatke vrne v XML dokumentu, JavaScript kodi ali pa kot preprosto besedilo. S tem se izognemo prenašanju nepotrebnih delov spletne strani. Količina prometa v smeri od strežnika k odjemalcu je v klasičnih aplikacijah, poleg podatkov, določena tudi z velikostjo same spletne strani z oblikovanjem vred. Pri Ajax aplikacijah je prenos na začetku večji, ko strežnik pošlje celoten uporabniški vmesnik in del logike spletne aplikacije odjemalcu. Nadaljnja komunikacija pa je bolj učinkovita, sej so odgovori strežnika samo podatki, ne pa celotna spletna stran.

2.4.3. Uporabnikova interakcija z aplikacijo je lahko nepretrgana in tekoča

Spletne povezave in HTML obrazci sta dve možnosti za vnos podatkov uporabnika, ki ju ponujajo spletni brskalniki.

Povezave so lahko preprosto besedilo, lahko pa jih oblikujemo s stilnimi predlogami CSS, jim dodamo slike, določimo stile ob prehodu miške čez povezavo in s tem dosežemo lepši ter bolj privlačen uporabniški vmesnik. HTML obrazce pa sestavljajo različni tipi gradnikov, namenjeni vnašanju podatkov – vnosna polja, potrditvena polja, izbirna polja in spustni seznami. Manjkajo pa komponente, katere poznamo iz namiznih aplikacij, kot so drevesa, tabele z možnostjo urejanja in podobne naprednejše komponente.

Pri klasičnih spletnih aplikacijah, ko se podatki preko spletne povezave ali HTML obrazca pošljejo strežniku, mora uporabnik počakati na odgovor strežnika, kar pripelje do neodzivnosti spletne aplikacije. To pa je pri delu s spletno aplikacijo moteče, zato so Ajax aplikacije v prednosti zaradi tekoče uporabe, saj delo poteka nemoteno in brez čakanja, da se v brskalnik naloži nova spletna stran. Tako se lahko uporabnik posveti samemu delu. Prav tako lahko Ajax aplikacije s pomočjo JavaScripta realizirajo napredne gradnike za vnos podatkov ter tako povečajo učinkovitost in zadovoljstvo uporabnika pri delu s spletnimi aplikacijami.

2.4.4. JavaScript je resen programski jezik, potrebno je vzdrževati red

Do uporabe JavaScripta v Ajax aplikacijah je JavaScript veljal za nekakšen neresen jezik, ki se je uporabljal za majhne dinamične dodatke pri spletnih straneh, in na stil kodiranja ni bilo potrebno pretirano paziti. Z uporabo v Ajaxu pa so tudi programerji začeli na JavaScript gledati kot na resen programski jezik.

Tako se je trajanje delovanja JavaScript kode povečala na celotno trajanje seje, pri tem pa ne sme prihajati do napak, počasnosti ali prevelike porabe pomnilnika. Tako je tudi za JavaScript začel veljati red in urejenost kode, postal je resen programski jezik. Prav tako je zaradi večje obsežnosti JavaScript kode potrebna modularnost in preglednost kode, kar pripelje do večje učinkovitosti pri razvoju kode.

2.5. Kaj pridobimo z GWT?

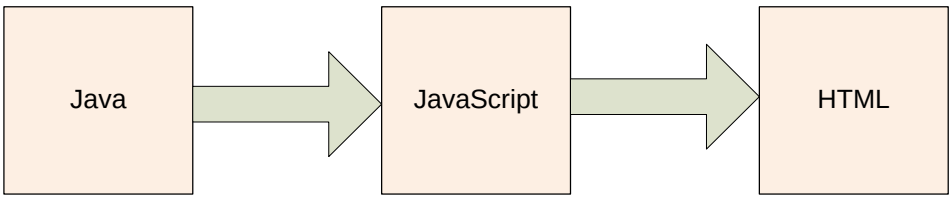
GWT združi programsko kodo za strežnik in odjemalca v eno samo aplikacijo, napisano v enem samem programskem jeziku – Javi.

To nam prinese naslednje prednosti:

- večje število programerjev zna programirati v Javi kot JavaScriptu,
- večje število orodji za razvoj v Javi,
- podoben razvoj kot ga poznamo pri razvoju namiznih aplikacij,
- več že vnaprej pripravljenih naprednih gradnikov spletnih strani,
- olajšano odkrivanje napak v kodi.

S pisanjem kode v istem jeziku za odjemalca in strežnik lahko isto kodo uporabimo na obeh straneh. Tako na primer lahko za validacijo vnosnih polj uporabimo isto kodo, prvič za takojšen odziv v brskalniku na strani odjemalca, drugič pa na strežniku za ponovno validacijo in tako dosežemo večjo varnost.

GWT pred programerjem skrije različno interpretacijo JavaScript kode različnih spletnih brskalnikov. Sicer so razlike v interpretaciji majhne, a še kako pomembne za pravilno izvajanje spletne aplikacije. Tako je koda brez ukvarjanja s problemi samih brskalnikov, delujoča na vseh podprtih brskalnikih, ki se uporabljajo dandanes.



Slika 3. GWT pristop: izvorna koda v Javi se prevede v JavaScript, ki se izvede v brskalniku kot HTML / JavaScript / CSS.

2.6. Kako začeti?

Spletne aplikacije z GWT lahko razvijamo na operacijskih sistemih Windows ali Linux. Potrebujemo še Javo in pa seveda GWT. Za pisanje kode lahko seveda uporabljamo preproste, navadne urejevalnike besedil, vsekakor pa je lažje pisanje z za to namenjenimi programi, kot je na primer Eclipse.

2.7. Struktura GWT projekta in datoteke

V GWT imamo že pripravljeni dve skripti, *projectCreator* and *applicationCreator*, s katerimi kreiramo projekt in samo aplikacijo.

Osnovno GWT aplikacijo sestavljajo naslednje datoteke:

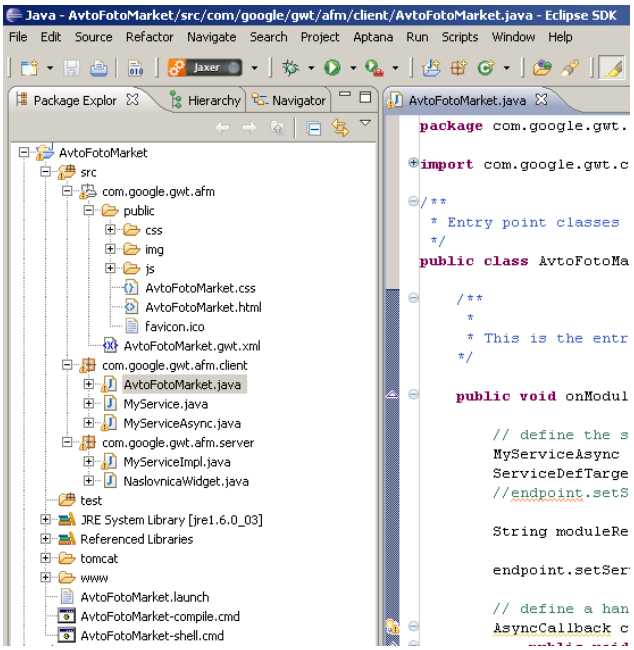
- MyProject\src\com\xyz\MyApp.gwt.xml
- MyProject\src\com\xyz\public\MyApp.html
- MyProject\src\com\xyz\client\MyApp.java
- MyProject\MyApp-shell.cmd
- MyProject\MyApp-compile.cmd

V XML datoteki **MyApp.gwt.xml** se nahaja zapis GWT modula: katere dodatne module GWT aplikacija uporablja, Javanski razred kjer se aplikacija začne, ter katera stilna predloga CSS naj se uporabi za oblikovanje spletne aplikacije.

MyApp.html datoteka je statični HTML dokument, kjer so z označevalnim jezikom definirani statični deli spletne strani.

Datoteka **MyApp.java** vsebuje kodo, ki se izvaja na strani klienta.

MyApp-shell.cmd je skripta, ki odpre našo spletno aplikacijo v GWT razvojni lupini in v GWT brskalniku, **MyApp-compile.cmd** pa skripta za prevajanje kode.



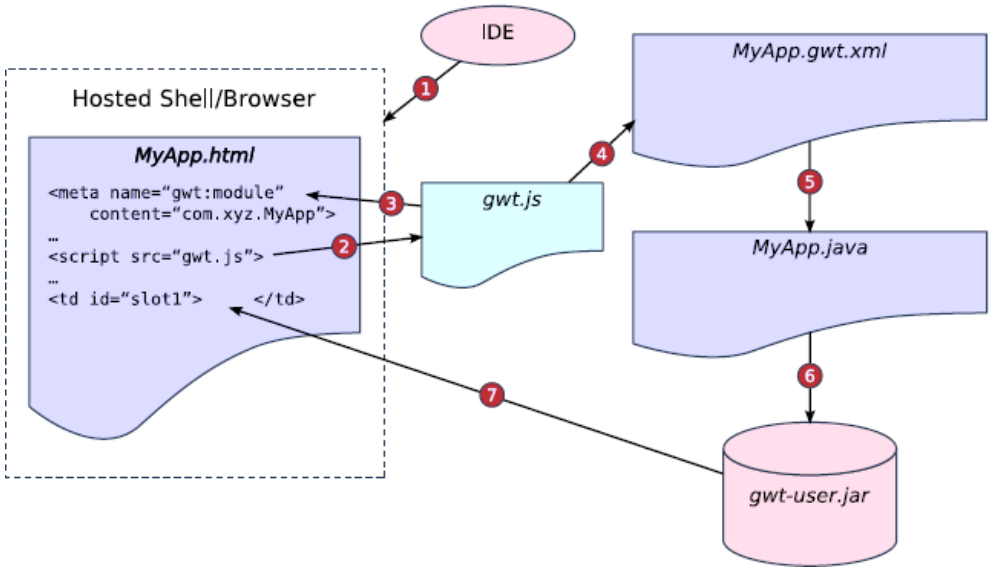
Slika 4. Primer organizacije datotek GWT projekta v Eclipsu.

2.8. Gostujoči način za razvoj aplikacije

Za razvoj GWT aplikacije se uporablja MyApp-shell.cmd skripta, ki nam odpre GWT razvojno lupino in GWT brskalnik. V ozadju GWT razvojne lupine deluje strežnik Tomcat.

Slika 5. GWT razvojna lupina in GWT brskalnik.

V gostujočem načinu se izvaja Javanska koda od začetka do konca spletne aplikacije. Zato je tudi mogoče iskati napake v aplikaciji z ustreznimi orodji za razvoj Java programov.



Slika 6. Odpiranje GWT aplikacije v gostujočem načinu.

Potek odpiranja aplikacije v gostujočem načinu:

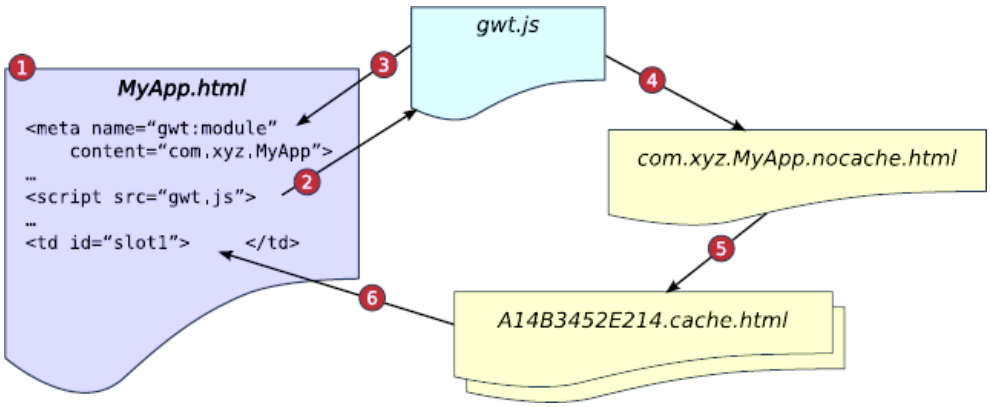
1. Program Shell odpre okno gostujočega GWT brskalnika, v katerem odpre MyApp.html.
2. MyApp.html naloži datoteko gwt.js.
3. gwt.js iz datoteke MyApp.html prebere ime modula MyApp.gwt.xml.
4. GWT prebere modul MyAll.gwt.xml in najde razred MyApp.java z vstopno točko.
5. Instanca razred MyApp.java kliče metodo onModuleLoad( ) in spletna aplikacija se prične izvajati.
6. Spletna aplikacija kliče funkcije v knjižnici gwt-user.jar.
7. Funkcije v knjižnici gwt-user.jar manipulirajo z DOM spletne aplikacije v gostujočem brskalniku - dodajo gradnike uporabniškega vmesnika in poslušalce za dogodke v brskalniku.

Ko kodo za spletno aplikacijo napišemo do konca in vse deluje kot mora, jo prevedemo v takšno obliko, da bomo lahko spletno aplikacijo odprli tud v navadnem brskalniku. Tukaj pridemo do spletnega načina GWT aplikacije.

2.9. Spletni način za postavitev aplikacije na strežnik

Ko v gostujočem načinu v gostujočem brskalniku kliknemo na gumb »Compile/Browse«, GWT prevajalnik paket .client prevede v JavaScript in spletno aplikacijo odpre v navadnem brskalniku na strežniku Tomcat. To aplikacijo pa lahko sedaj poženemo tudi z datotečnega sistema ali pa naložimo na kakšen drug strežnik.

Ob klicu GWT prevajalnika, ta programsko kodo spletne aplikacije združi z JavaScript verzijo GWT API v eno samo datoteko z JavaScript kodo. Shrani jo v www mapo projekta spletne aplikacije, kjer so tudi že ostale pripadajoče datoteke, ki skrbijo za delovanje aplikacije. Prav tako v mapo www skopira še vsebino mape public.



Slika 7. Odpiranje GWT aplikacije v spletnem načinu.

Potek odpiranja aplikacije v gostujočem načinu:

1. Spletni brskalnik naloži MyApp.html.
2. Naloži se JavaScript datoteka gwt.js.
3. gwt.js iz datoteke MyApp.html prebere ime modula com.xyz.MyApp.
4. gwt.js vstavi na spletno strani gradnik iframe, v katerega se naloži vsebina strani com.xyz.MyApp.noncache.html.
5. JavaScript v strani com.xyz.MyApp.noncache.html preveri, kateri brskalnik uporablja uporabnik spletne aplikacije in iframe usmeri na ustrezno datoteko za tip brskalnika.
6. JavaScript koda, ki je ekvivalentna metodi onModuleLoad( ), se izvede.

Tako prevedene preproste spletne aplikacije, ki ne uporabljajo interakcije s strežnikom, delujejo na vseh strežnikih, saj se vsa koda izvede na strani odjemalca. Takšno spletno aplikacijo lahko naložimo na poljuben strežnik, lahko pa jo poganjamo tud z datotečnega sistema.

2.10. Klici oddaljenih procedur

Ali naj se koda izvaja na strežniku, kjer imamo večjo kontrolo in večjo varnost? Ali naj se izvaja pri odjemalcu, kar prinese boljšo odzivnost in interaktivnost spletne aplikacije? GWT lahko kodo izvaja na obeh straneh, tako na strežniku kot pri odjemalcu. Za vmesno povezavo pa skrbijo klici oddaljenih procedur (RPC).

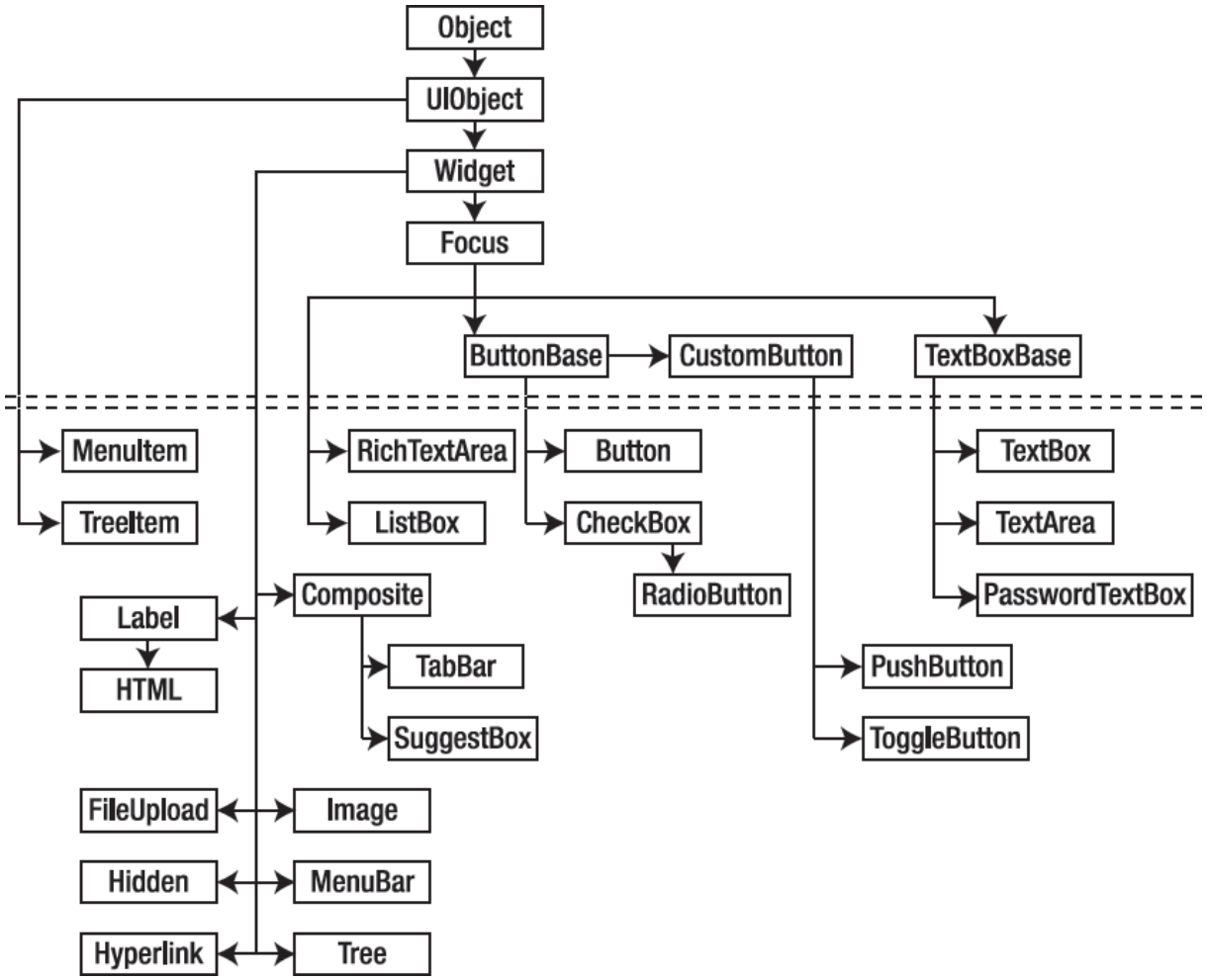
Spletne aplikacije GWT spadajo med bogate spletne aplikacije (RIA), saj se del kode izvaja na strežniku, del kode pa pri odjemalcu. Na strani odjemalca ne potrebujemo nobenih posebnih programov, le klasični spletni brskalnik.

GWT uporablja svoj protokol za klic oddaljenih procedur. Prvi razlog za uporabo lastnega protokola je asinhrono klicanje procedur. Klici lahko ali pa ne vrnejo odgovorov v prihodnosti, aplikacija pa med čakanjem na odgovor še vedno ostane odzivna in uporabnik lahko nemoteno nadaljuje delo. Aplikacija sicer deluje še naprej in se mora kar se da najbolje prilagoditi delu brez podatka, na katerega čaka po klicu oddaljene procedure. Drugi razlog za nov protokol pa je njegova preprostost. Tako je programske kode manj in je čas, ki poteče ob prvem nalaganju spletne aplikacije, krajši. Prav tako veliko JavaScript kode upočasni delovanje brskalnika, česar pa si vsekakor ne želimo.

Serializacija v JavaScriptu ne podpira načina serializacije, kot ga poznamo v Javi. Prav tako ni mogoče dinamično nalaganje razredov, zato so se pri Googlu odločili, da implementirajo lastno serializacijo za JavaScript.

2.11. Programiranje grafičnega vmesnika – osnovni gradniki

V GWT imamo na voljo osnovne gradnike za gradnjo grafičnega vmesnika spletne aplikacije. Ti so namenjeni interakciji med uporabnikom in spletno aplikacijo.



Slika 8. Razredna hierarhija osnovnih gradnikov.

Vsi gradniki se v GWT nahajajo v paketu com.google.gwt.user.client.ui. Med njimi je 6 abstraktnih razredov za gradnike (z nekaj osnovnimi metodami):

- razred UIObject vsebuje metode za vidljivosti gradnikov, nastavljanje dimenzij gradnikom, določanje stilskih predlog gradnikom,
- razred Widget vsebuje metode za preverjan stanj gradnikov na spletni strani in za dodajanje gradnikov na panele,
- razred FocusWidget vsebuje metode za upravljanje s poslušalci na gradnikih,
- razred ButtonBase vsebuje metode za upravljanje z besedilom na gumbih,
- razred TextBoxBase vsebuje metode za delo z vnosnim polji.

3. Implementacija informacijske rešitve

3.1. Opis problemske domene

Spletna aplikacija, ki jo želimo razviti s pomočjo GWT, bi bila namenjena za potrebe uredništva avtomobilistične revije Avto Foto Market. Revija pokriva novice iz sveta avtomobilizma, avtomobilske teste, reportaže s salonov in podobnih dogodkov, ki imajo skupni imenovalec avtomobilizem. Spletno aplikacijo bi lahko razdelili na dva dela. Prvi del bi bil namenjen interni uporabi uredništva, drugi pa spletni različici tiskane revije, namenjene vsem obstoječim bralcem in novim bralcem, ki bi jih pridobili z objavljanjem vsebine na spletu.

Del spletne aplikacije za interno uporabo bi bil namenjen samo članom uradništva, tako bi bil dostop do tega dela mogoč samo z uporabniškim imenom in geslom. Vsak uporabnik bi imel dostop do določenih delov aplikacije, za urejanje katerih bi bil zadolžen.

Tako bi se lahko preko spletne aplikacije oddajalo članke in fotografije, namenjene za objavo v tiskani reviji in kasneje na spletni stran. S tem bi pridobili boljši pregled nad delom in potekom sestavljanja revije v tekočem mesecu. Že v naprej bi bila znana vsebina prihajajoče revije in kdo je za njo zadolžen, prav tako roki oddaje člankov. Ti podatki bi lahko bili zbrani v tabeli, iz katere bi bilo razvidno, kako pripravljanje vsebine za revijo napreduje, prav tako pa bi bil mogoč dostop do besedil za članke in fotografij.

Uporabna bi bila tudi možnost skupnega koledarja, v katerega bi člani uredništva vpisovali datume sestankov, avtomobilskih predstavitev, izida revije in ostale pomembne dogodke. Tako bi imeli boljši pregled nad aktualnimi dogodki iz sveta avtomobilizma, ki bi bili zbrani na enem mestu. Koledar bi lahko bil v obliki tabele oziroma seznama, lahko pa grafično izpopolnjen v pravi koledar, odvisno od potreb in obsega koledarja.

Urejanje vsebine spletnega portala za bralce bi potekalo preko spletne aplikacije za člane uredništva. Spletna verzija tiskane revije bi bila na voljo proti plačilu, starejše številke pa bi bile na voljo brezplačno.

Del spletne aplikacije za oddajo malih oglasov pa bi bil namenjen vnašanju malih oglasov za rabljene avtomobile, dostopen samo pooblaščenim osebam za vnos malih oglasov.

3.2. Analiza in načrtovanje

V procesu razvoja nove informacijske rešitve sta analiza in načrtovanje najpomembnejša dejavnika pri uspešnosti izvedbe. Najlažje bi bilo sicer kar konkretno razviti rešitev, a bi kmalu naleteli na dvome o pravilni smeri razvoja. Z analizo in načrtovanjem dosežemo učinkovito komuniciranje s strankami o želeni strukturi in obnašanju sistema, boljše razumevanje sistema in iskanje možnosti poenostavitve ter ponovne uporabe, vizualizacijo in nadzor nad arhitekturo sistema.

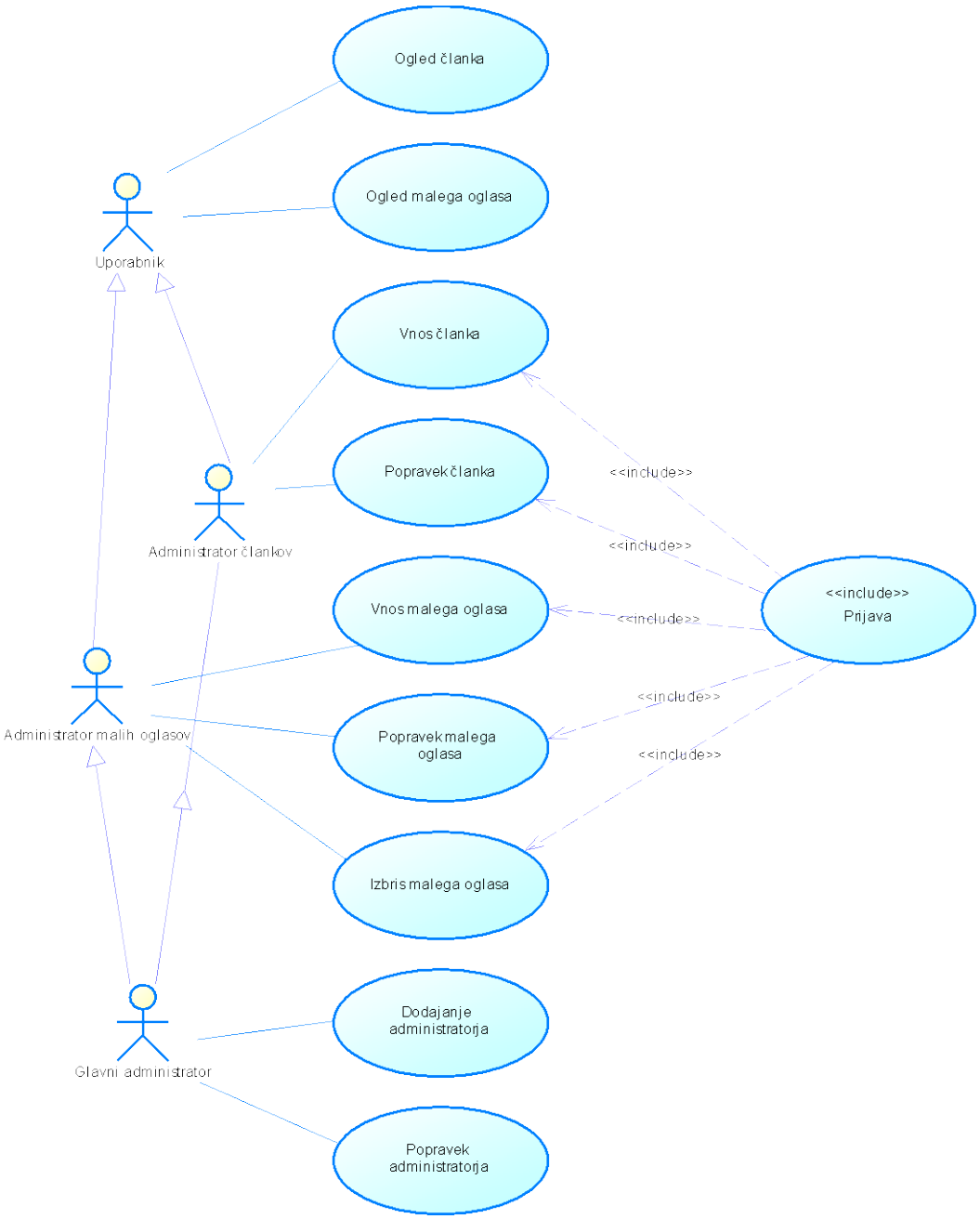
Za analizo in načrtovanje uporabljamo različne diagramske tehnike. Uporabljamo jih za opis specifikacij, načrtov in modelov. UML je grafični jezik, ki nam koristi v tem primeru. To je jezik za vizualizacijo, specificiranje, konstruiranje in dokumentiranje izdelkov informacijskih rešitev.

Z jezikom UML sem zgradil diagram primerov uporabe ter diagrame zaporedja. Ti dve diagramski tehniki služita predvsem za načrtovanje logičnega podatkovnega modela, kar pripomore k hitrejši in kvalitetnejši izdelavi informacijske rešitve.

3.2.1. Diagrami primerov uporabe

Diagrami primerov uporabe opisujejo funkcionalno obnašanje sistema, kot ga vidi uporabnik. Predstavljajo komunikacijo med akterji – uporabnikom in informacijskim sistemom.

Informacijski sistem tvorijo primeru uporabe: Prijava, Ogled članka, Ogled malega oglasa, Iskanje malega oglasa, Vnos članka, Popravek članka, Vnos malega oglasa, Popravek malega oglasa, Izbris malega oglasa, Dodajanje administratorja in Popravek administratorja.



Slika 9. Diagram primerov uporabe

3.2.1.1. Opis toka dogodkov za primer Ogled članka

Primer uporabe omogoča uporabniku ogled članka (novice, testa, reportaže).

Osnovni tok:

0. Uporabnik začne postopek ogleda članka, ko odpre stran, ki v URL naslovu vsebuje ID članka.
1. Sistem prikaže spletno stran z vsebino članka.

Alternativni tok:

0. Uporabnik začne postopek ogleda članka, ko odpre stran, ki v URL naslovu vsebuje ID članka.
1. Sistem iskanega članka na tem naslovu ne najde, zato prikaže obvestilo, da iskani članek ne obstaja.

3.2.1.2. Opis toka dogodkov za primer Ogled malega oglasa

Primer uporabe omogoča uporabniku ogled malega oglasa.

Osnovni tok:

0. Uporabnik začne postopek ogleda malega oglasa, ko odpre stran, ki v URL naslovu vsebuje ID malega oglasa.
1. Sistem prikaže spletno stran z vsebino malega oglasa.

Alternativni tok:

0. Uporabnik začne postopek ogleda malega oglasa, ko odpre stran, ki v URL naslovu vsebuje ID malega oglasa.
1. Sistem iskanega malega oglasa na tem naslovu ne najde, zato prikaže obvestilo, da iskani mali oglas ne obstaja.

3.2.1.3. Opis toka dogodkov za primer uporabe Prijava

Primer uporabe omogoča administratorju prijavo v sistem.

Osnovni tok:

0. Administrator začne postopek prijave, ko odpre naslov strani.
1. Sistem prikaže zaslon z vnosnimi polji za uporabniško ime in geslo.
2. Administrator vnese uporabniško ime in geslo.
3. Sistem preveri uporabniško ime in geslo.
4. Sistem prikaže zaslon za izbiro možnosti dela, ki so administratorju omogočeni glede na njegove pravice.

Alternativni tok:

0. Sistem prikaže zaslon z vnosnimi polji za uporabniško ime in geslo.
1. Administrator vnese uporabniško ime in geslo.
2. Sistem preveri uporabniško ime in geslo.
3. Sistem zavrne dostop do aplikacije zaradi napačnega uporabniškega imena ali gesla.

3.2.1.4. Opis toka dogodkov za primer uporabe Vnos članka

Primer uporabe omogoča administratorju vnos članka (novice, testa, reportaže).

Osnovni tok:

0. Administrator klikne na povezavo za vnos članka.
1. Sistem prikaže zaslon za vnos naslova članka, izbiro za katero vrsto članka gre in izbiro na katero avtomobilsko znamko se članek nanaša.
2. Administrator vnese naslov članka, s seznama izbere vrsto članka in ustrezno avtomobilsko znamko in potrdi podatke.
3. Sistem preveri naslov članka, ID avtomobilske znamke in ID vrste članka, prikaže zaslon za vnašanje vsebine članka.
4. Administrator vnese vsebino članka in potrdi podatke.
5. Sistem preveri vsebino članka in prikaže predogled članka.
6. Administrator potrdi članek.
7. Sistem zapiše članek v podatkovno bazo in shrani fotografije na strežnik.
8. Sistem prikaže sporočilo o uspešni shranitvi članka v podatkovno bazo in fotografij na strežnik.

Alternativni tok:

0. Administrator klikne na povezavo za vnos članka.
1. Sistem prikaže zaslon za vnos naslova članka, izbiro za katero vrsto članka gre in izbiro na katero avtomobilsko znamko se članek nanaša.
2. Administrator ne vnese naslova članka, s seznama izbere vrsto članka in ustrezno avtomobilsko znamko in potrdi podatke.
3. Sistem preveri naslov članka in ugotovi, da tega administrator ni vpisal.
4. Sistem vrne uporabniku sporočilo, da je vnosno polje naslov članka obvezno.
5. Administrator vnese naslov članka in potrdi podatke.
6. Sistem preveri naslov članka in prikaže zaslon za vnašanje vsebine članka.
7. Administrator vnese vsebino članka in potrdi podatke.
8. Sistem preveri vsebino članka in prikaže predogled članka.
9. Administrator potrdi članek.
10. Sistem zapiše članek v podatkovno bazo in shrani fotografije na strežnik.
11. Sistem prikaže sporočilo o uspešni shranitvi članka v podatkovno bazo in fotografij na strežnik.

3.2.1.5. Opis toka dogodkov za primer uporabe Popravek članka

Primer uporabe omogoča administratorju popravek članka.

Osnovni tok:

0. Administrator klikne gumb popravek ob želenem članku na zaslonu seznam člankov.
1. Sistem prikaže zaslon za popravek vsebina članka.
2. Administrator popravi naslov ali/in vsebino članka in potrdi podatke.
3. Sistem preveri naslov ter vsebino članka in prikaže predogled članka.
4. Administrator potrdi članek.
5. Sistem zapiše članek v podatkovno bazo in shrani morebitne nove fotografije na strežnik.
6. Sistem prikaže sporočilo o uspešni shranitvi članka v podatkovno bazo in morebitnih novih fotografij na strežnik.

Alternativni tok:

0. Administrator klikne gumb popravek ob želenem članku na zaslonu seznam člankov.
1. Sistem prikaže zaslon za popravek vsebina članka.
2. Administrator popravi naslov ali/in vsebino članka in potrdi podatke.

3. Sistem preveri naslov ter vsebino članka in ugotovi, da vsebina članka vsebuje nedovoljene znake.
4. Sistem vrne uporabniku sporočilo, da je vnosno polje vsebine članka vsebuje nedovoljene znake.
5. Administrator ponovno vnese vsebino članka in potrdi podatke.
6. Sistem preveri naslov ter vsebino članka in prikaže predogled članka.
7. Administrator potrdi članek.
8. Sistem zapiše članek v podatkovno bazo in shrani morebitne nove fotografije na strežnik.
9. Sistem prikaže sporočilo o uspešni shranitvi članka v podatkovno bazo in morebitnih novih fotografij na strežnik.

3.2.1.6. Opis toka dogodkov za primer Vnos malega oglasa

Primer uporabe omogoča administratorju vnos malega oglasa.

Osnovni tok:

0. Administrator klikne na povezavo za vnos malega oglasa.
1. Sistem prikaže zaslon za vnos malega oglasa.
2. Administrator vnese naslov, vsebino in fotografije malega oglasa, s seznama izbere in ustrezno avtomobilsko znamko ter model in potrdi podatke.
3. Sistem preveri naslov in vsebino malega oglasa, avtomobilsko znamke in modela avtomobila.
4. Sistem prikaže predogled malega oglasa.
5. Administrator potrdi mali oglas.
6. Sistem zapiše mali oglas v podatkovno bazo in shrani fotografije na strežnik.
7. Sistem prikaže sporočilo o uspešni shranitvi malega oglasa v podatkovno bazo in fotografij na strežnik.

Alternativni tok:

0. Administrator klikne na povezavo za vnos malega oglasa.
1. Sistem prikaže zaslon za vnos malega oglasa.
2. Administrator vnese naslov, vsebino in fotografije malega oglasa, s seznama izbere in ustrezno avtomobilsko znamko ter model in potrdi podatke.
3. Sistem preveri naslov in ugotovi, da naslov malega oglasa vsebuje nedovoljene znake.
4. Sistem vrne administratorju sporočilo, da je vnosno polje naslov malega oglasa vsebuje nedovoljene znake.

5. Administrator ponovno vnese naslov malega oglasa in potrdi podatke.
6. Sistem preveri naslov in vsebino malega oglasa, avtomobilsko znamke in modela avtomobila.
7. Sistem prikaže predogled malega oglasa.
8. Administrator potrdi mali oglas.
9. Sistem zapiše mali oglas v podatkovno bazo in shrani fotografije na strežnik.
10. Sistem prikaže sporočilo o uspešni shranitvi malega oglasa v podatkovno bazo in fotografij na strežnik.

3.2.1.7. Opis toka dogodkov za primer Popravek malega oglasa

Primer uporabe omogoča administratorju popravek malega oglasa.

Osnovni tok:

0. Administrator klikne gumb popravek ob želenem malem oglasu na zaslonu seznam malih oglasov.
1. Sistem prikaže zaslon za popravek malega oglasa.
2. Administrator popravi naslov ali/in vsebino malega oglasa in potrdi podatke.
3. Sistem preveri naslov ter vsebino malega oglasa in prikaže predogled malega oglasa.
4. Administrator potrdi mali oglas.
5. Sistem zapiše mali oglas v podatkovno bazo in shrani morebitne nove fotografije na strežnik.
6. Sistem prikaže sporočilo o uspešni shranitvi malega oglasa v podatkovno bazo in morebitnih novih fotografij na strežnik.

Alternativni tok:

0. Administrator klikne gumb popravek ob želenem malem oglasu na zaslonu seznam malih oglasov.
1. Sistem prikaže zaslon za popravek malega oglasa.
2. Administrator popravi naslov ali/in vsebino malega oglasa in potrdi podatke.
3. Sistem preveri naslov ter vsebino malega oglasa in ugotovi, da vsebina malega oglasa vsebuje nedovoljene znake.
4. Sistem vrne uporabniku sporočilo, da je vnosno polje vsebine malega oglasa vsebuje nedovoljene znake.
5. Administrator ponovno vnese vsebino malega oglasa in potrdi podatke.
6. Sistem preveri naslov ter vsebino malega oglasa in prikaže predogled malega oglasa.
7. Administrator potrdi mali oglas.

8. Sistem zapiše mali oglas v podatkovno bazo in shrani morebitne nove fotografije na strežnik.
9. Sistem prikaže sporočilo o uspešni shranitvi malega oglasa v podatkovno bazo in morebitnih novih fotografij na strežnik.

3.2.1.8. Opis toka dogodkov za primer Izbris malega oglasa

Primer uporabe omogoča administratorju izbris malega oglasa.

Osnovni tok:

0. Administrator klikne gumb izbriši ob želenem malem oglasu na zaslonu seznam malih oglasov.
1. Sistem prikaže zaslon za izbris malega oglasa.
2. Administrator potrdi izbris malega oglasa
3. Sistem izbriše mali oglas v podatkovni bazi in morebitne pripadajoče fotografije.
4. Sistem prikaže sporočilo o uspešnem izbrisu malega oglasa v podatkovni bazi in morebitnih pripadajočih fotografij na strežniku.

Alternativni tok:

0. Administrator klikne gumb izbriši ob želenem malem oglasu na zaslonu seznam malih oglasov.
1. Sistem prikaže zaslon za izbris malega oglasa.
2. Administrator prekliče izbris malega oglasa
3. Sistem prikaže zaslon seznam malih oglasov.

3.2.1.9. Opis toka dogodkov za primer Vnos administratorja

Primer uporabe omogoča administratorju vnos novega administratorja člankov ali administratorja malih oglasov.

Osnovni tok:

0. Administrator klikne na vnos novega administratorja.
1. Sistem prikaže zaslon za vnos novega administratorja in izbiro za katero vrsto administratorja gre.

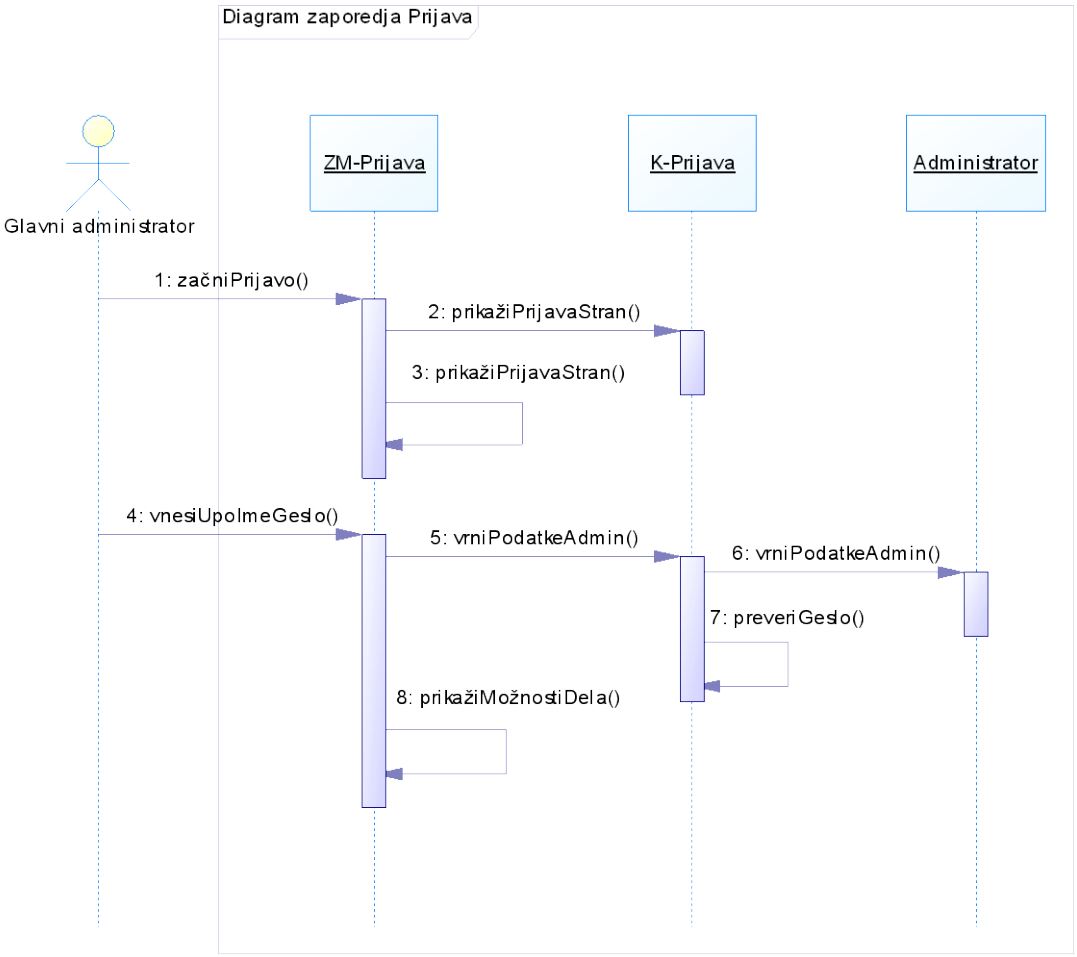
2. Administrator vnese uporabniško ime novega administratorja, njegovo uporabniško im, izbere vrsto administratorja, vnese administratorjeve osebne podatke in potrdi podatke.
3. Sistem preveri naslov uporabniško ime, geslo, podatke in vrsto novega administratorja.
4. Sistem prikaže predogled podatkov o novem administratorju.
5. Administrator potrdi podatke o novem administratorju.
6. Sistem zapiše podatke o novem administratorju v podatkovno bazo.
7. Sistem prikaže sporočilo o uspešni shranitvi podatkov o novem administratorju v podatkovno bazo.

3.2.2. Diagrami zaporedja

Diagrami zaporedja opisujejo dinamično obnašanje med akterji in sistemom ter med objekti sistema. Ti diagrami eksplicitno prikazujejo zaporedje sporočil. So primerni za prikaz celotnega toka dogodkov in za specifikacijo dogodkov, ki potekajo v realnem času.

3.2.2.1. Diagram zaporedja za primer uporabe Prijava

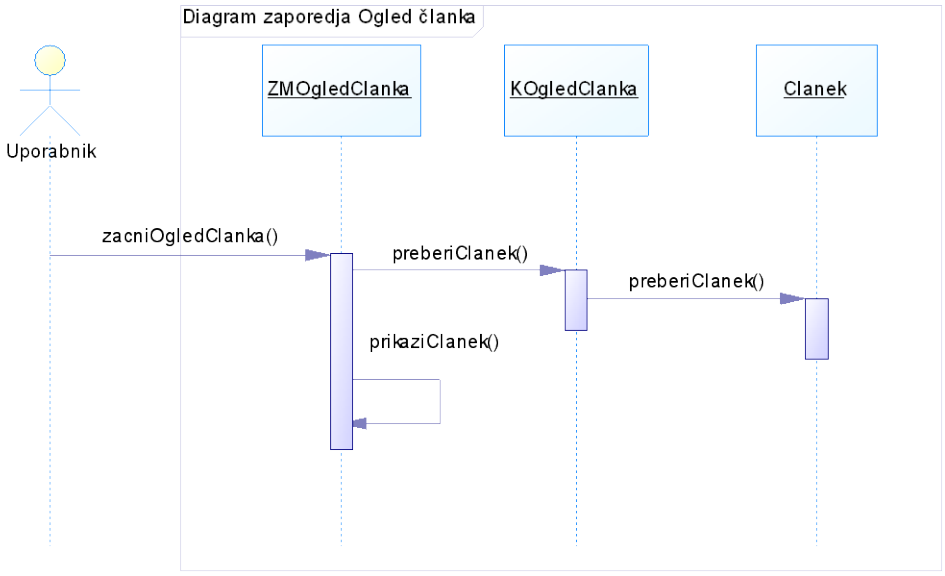
Identificiral sem začetni razred Administrator.



Slika 10. Diagram zaporedja za primer uporabe Prijava.

3.2.2.2. Diagram zaporedja za primer uporabe Ogled članka

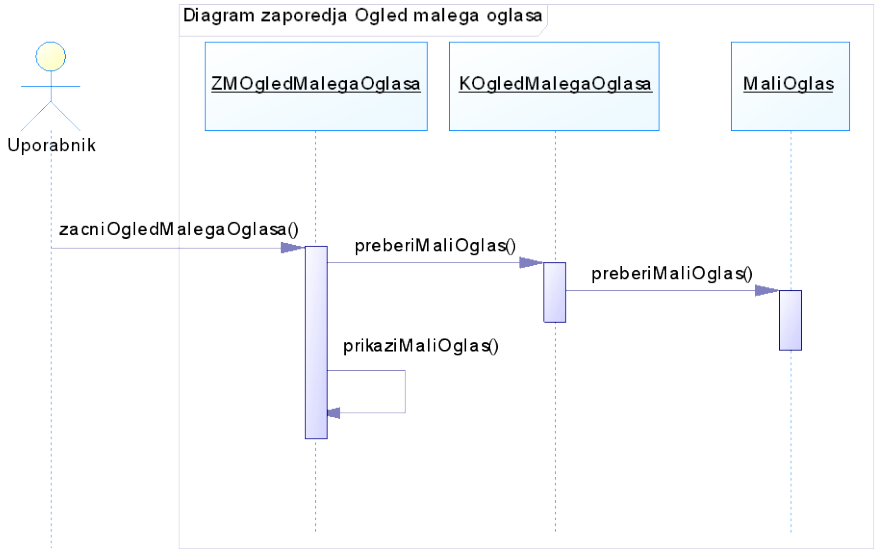
Identificiral sem začetni razred Clanek.



Slika 11. Diagram zaporedja za primer uporabe Ogled članka.

3.2.2.3. Diagram zaporedja za primer uporabe Ogled malega oglasa

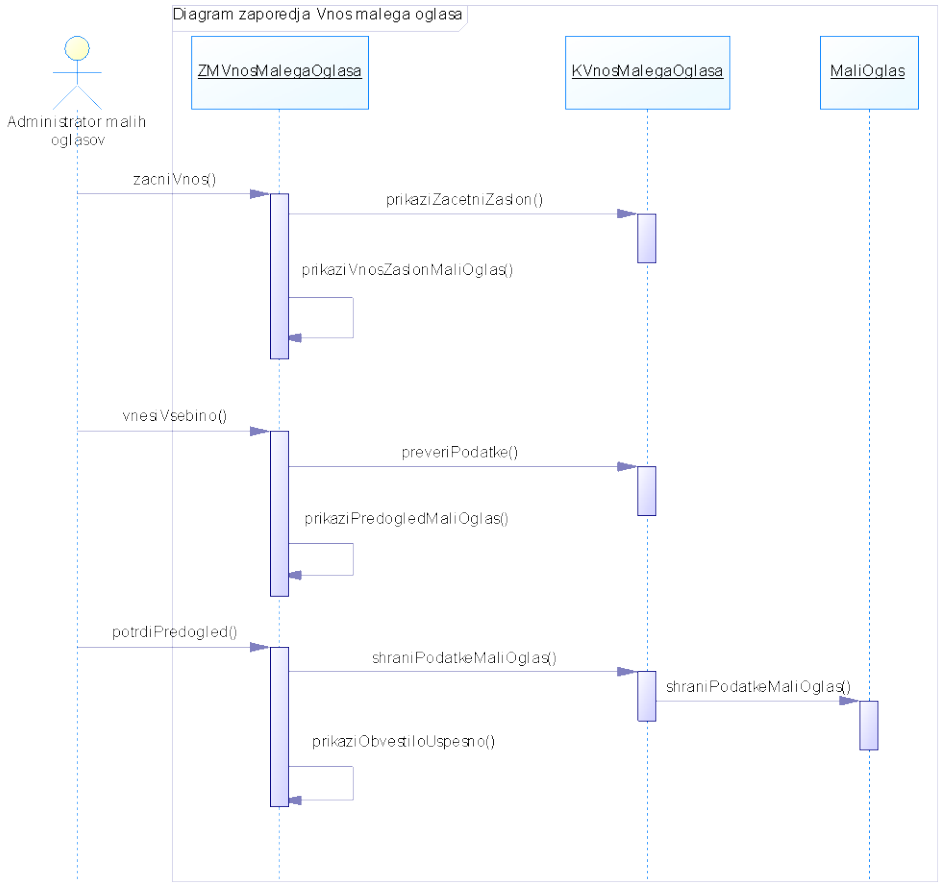
Identificiral sem začetni razred Mali oglas.



Slika 12. Diagram zaporedja za primer uporabe Ogled malega oglasa.

3.2.2.4. Diagram zaporedja za primer uporabe Vnos malega oglasa

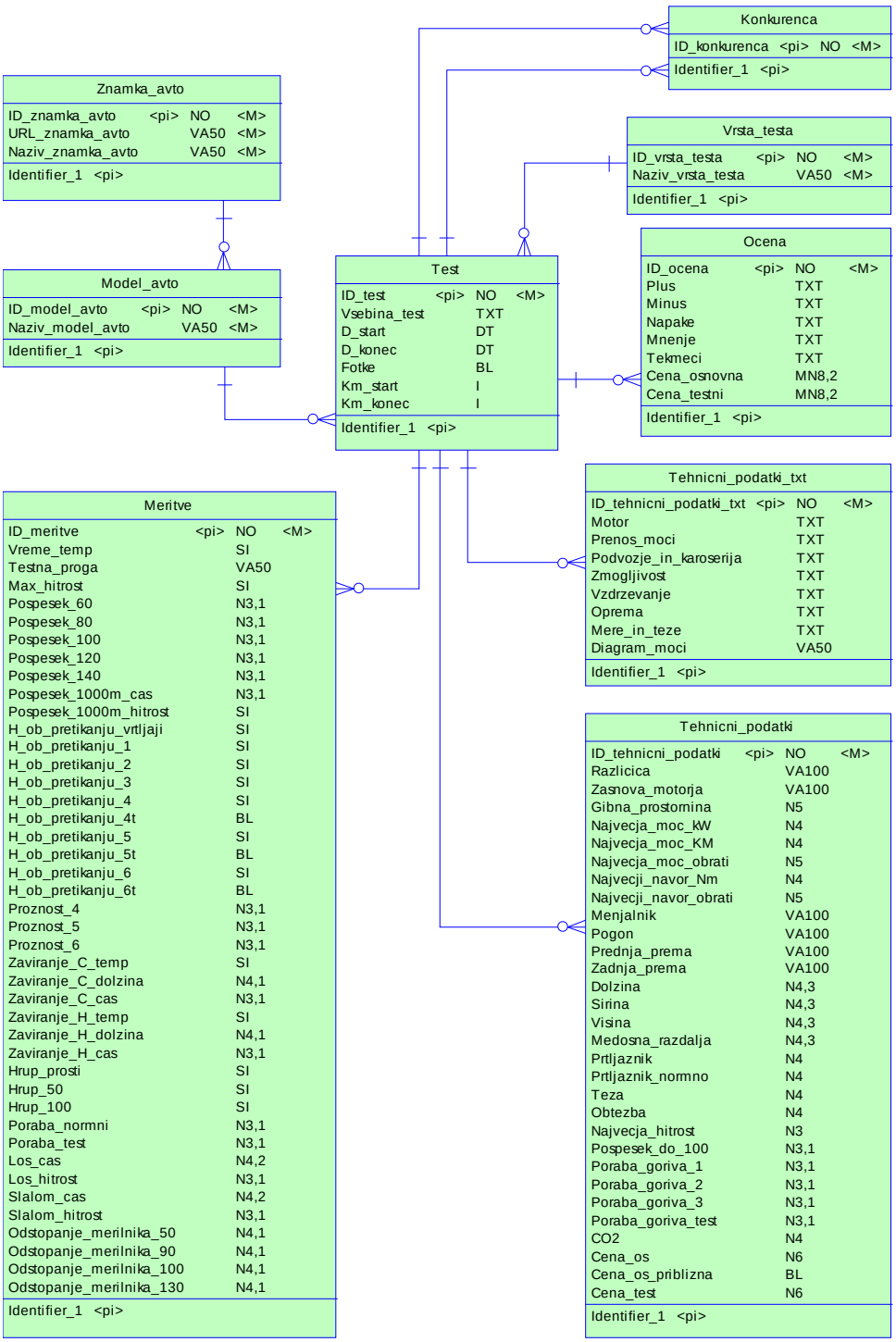
Identificiral sem začetni razred Mali oglas.



Slika 13. Diagram zaporedja za primer uporabe Vnos malega oglasa.

3.3. Realizacija spletne aplikacije

Za realizacijo spletne aplikacije bi uporabili orodje GWT, kar bi pomenilo, da bi bil vsa koda napisana v programskem jeziku Java v kombinaciji s HTML in CSS za oblikovanje. Za strežnik pa bi uporabili Apache Tomcat, podatke bi shranjevali v podatkovno bazo MySQL.



Slika 14. Konceptualni model dela podatkovne baze za teste avtomobilov.

3.4. Spletna različica revije

Del spletne aplikacije, ki bi bil dostopen vsem obiskovalcem spletne strani, bi bila spletna različica revije. Obsegala bi članke in reportaže s fotografijami, tehničnimi podatki, malimi oglasi rabljenih avtomobilov in koledarjem dogodkov iz sveta avtomobilizma. V prihodnosti bi seveda dodajali še nove funkcionalnosti in s tem obdržali obiskovalce spletne strani in pridobili nove.

Slika 15. Avtomobilski test objavljen na spletni strani.

Spletna stran je sestavljena iz gradnikov narejenih po meri z uporabo osnovnih GWT gradnikov. Tako uporabniku ni potrebno čakati, da se naloži celotna spletna stran, ampak vsebino vidi že sproti, ko se nalagajo gradniki na spletno stran. S tem dosežemo boljšo uporabniško izkušnjo in skrajšamo čas čakanja uporabnika na zahtevano vsebino.

3.5. Grafični vmesnik spletne različice revije

Grafični vmesnik spletne strani je v grobem razdeljen na štiri dele: glavo, glavni meni, vsebino in desni meni.

Slika 16. Grafični vmesnik spletne strani za uporabnike.

Glavo sestavljata logotip revije, ki tudi predstavlja povezavo na začetno stran, in pa skupina ikon za hiter skok v rubrike testi, reportaže, članki in mali oglasi. Vsebina glave se med celotnim postopkom brskanja po spletni strani ne spreminja, tako da ima uporabnik ves čas na voljo pet povezav za hiter dostop do delov spletne strani.

Glavni meni je sestavljen iz GWT gradnikov po meri. Sestavljajo ga gradnik Naslovnica, v katerem je uporabniku na voljo slika in nekaj datumskih informacij o aktualni številki revije. Slika naslovnice revije deluje tudi kot povezava na seznam z vsebino aktualne številke. Naslovnici sledi gradnik za iskanje avtomobilov in modelov, tako da lahko uporabnik v spustnem seznamu vidi, kaj mu je na voljo. Za klasičen pogled pa je tu še povezava, ki kaže na seznam vseh testov v tabelarični obliki. Glavni meni nato sestavljajo povezave na različne sklope: novice, testi, članki in reportaže. Vsak sklop ima poleg povezave na seznam celotne

vsebine sklopa tudi hitre povezave do zadnjih treh vnosov v pripadajoči sklop. Na koncu vseh povezav sledi še povezava na male oglase in pa prostor za oglasne pasice.

Vsebina se prične s slednikom – vrstica, ki uporabniku sporoča, kje na strani se trenutno nahaja – in služi za boljšo navigacijo po spletni strani. Vsebuje naziv sklopa, kjer se trenutno nahaja uporabnik in datum objave vsebine, ki si jo uporabnik pravkar ogleduje. Temu sledi naslovna fotografija in naslov vsebine, naprej pa je uporabniku na voljo dejanska vsebina članka oziroma malega oglasa.

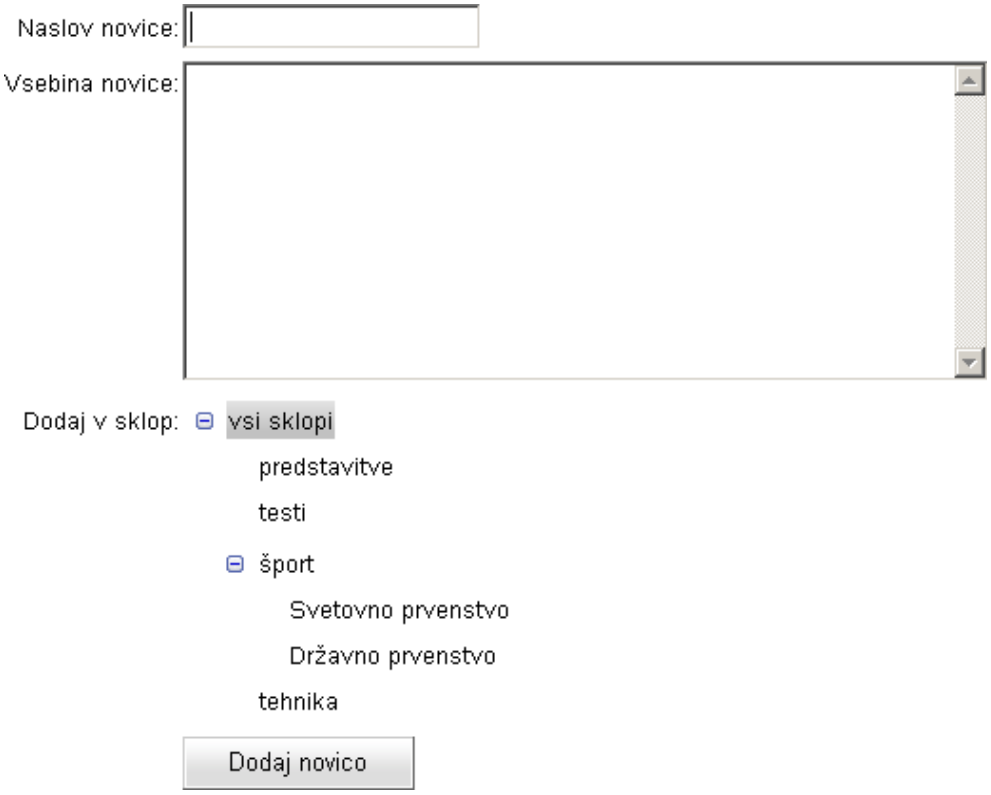
Desni meni je sestavljen iz gradnika za hitro iskanje po malih oglasih. V gradniku lahko nastavljamo želene parametre, po katerih želimo iskati med malimi oglasi. Temu gradniku pa sledijo še trije naključno izbrani mali oglasi, ki služijo kot hitre povezave do vsebine malega oglasa. Tudi na koncu desnega menija je prostor za oglasne pasice.

Prav tako pa je možnost vstavljanja oglasnih pasic med glavo spletne strani in vsebino, kar prikazuje slika 17.

Slika 17. Možnost vstavljanja oglasnih pasic med glavo in vsebino.

3.6. Zaslonska maska za vnašanje novice

Obrazec za vnašanje novice je sestavljen iz GWT gradnikov in ga lahko vidimo na sliki 18. Na tem primeru lahko vidimo, kako lahko z nekaj vrsticami kode ustvarimo drevo. Gradnja drevesa je kompleksna, če sami pišemo kodo HTML in JavaScript. Tako pa nam GWT nudi enostavno in hitro rešitev za bolj kompliciranе gradnike, kot je na primer drevo.



Slika 18. Obrazec za vnos novice.

Poleg drevesa nam GWT lahko pomaga tudi pri:

- spustnih menijih,
- orodnih vrsticah,
- bogatih urejevalnikih besedil,
- delom z zavihki v spletnih aplikacijah,
- polja za vnos besedila, ki uporabnikom predlagajo vnose besedil.

Naprednejši elementi v spletnih straneh do sedaj niso bili pogosti zaradi njihove kompleksne izvedbe in niso delovali v vseh brskalnikih. Z uporabo GWT je za razvijalce prednost tudi ta, da je način gradnje podoben kot pri razvijanju v Swing razredu v Javi za namizne aplikacije.

```
FlexTable postavitev = new FlexTable();

Label naslovNoviceLabel = new Label("Naslov novice:");
TextBox naslovNovice = new TextBox();
postavitev.setWidget(0,0,naslovNoviceLabel);
postavitev.setWidget(0,1,naslovNovice);

Label vsebinaNoviceLabel = new Label("Vsebina novice:");
TextArea vsebinaNovice = new TextArea();
vsebinaNovice.setCharacterWidth(50);
vsebinaNovice.setVisibleLines(10);
postavitev.setWidget(1,0,vsebinaNoviceLabel);
postavitev.setWidget(1,1,vsebinaNovice);

Label seznamSklopovLabel = new Label("Dodaj v sklop:");
TreeItem root = new TreeItem("vsi sklopi");
root.addItem("predstavitve");
root.addItem("testi");
TreeItem sport = new TreeItem("šport");
sport.addItem("Svetovno prvenstvo");
sport.addItem("Državno prvenstvo");
root.addItem(sport);
root.addItem("tehnika");
Tree t = new Tree();
t.addItem(root);
postavitev.setWidget(2,0,seznamSklopovLabel);
postavitev.setWidget(2,1,t);

Button submit = new Button("Dodaj novico");
postavitev.setWidget(3,1,submit);

RootPanel.get("form").add(postavitev);

naslovNovice.setFocus(true);
```

Slika 19. Koda za gradnjo obrazca za vnos novice.

Tako je kodiranje v primerjavi s HTML preglednejše in bolj prijazno do programerjev. Lahko tudi izkoristimo prednosti, ki jih nudi objektno programiranje – posamezni programski problemi so predstavljeni z resničnimi ali abstraktnimi objekti, ki med seboj komunicirajo s pošiljanjem sporočil. Takšne objekte, ki delujejo kot črne škatle (sprejemajo in pošiljajo podatke), lahko enostavno ponovno uporabimo.

4. Sklepne ugotovitve

V okviru diplomske naloge sem razvil spletno aplikacijo v GWT okolju. Realiziral sem cilj, ki smo si ga zastavili na začetku naloge, da je celotna aplikacija napisana v Javi in se s tem izognil problematičnemu kodiranju v JavaScriptu, ki ga je potrebno prilagajati za vsak spletni brskalnik posebej. Tako sem se lahko bolj posvetil vsebinskemu delu projekta.

Med razvojem spletne aplikacije nisem imel kakšnih posebnih težav. Pridobil sem si veliko novega znanja in izkušenj, saj je bil to zame s področja javanskih aplikacij prvi projekt, ki sem se lotil samostojno. Do sedaj sem spletne aplikacije razvijal v jeziku PHP, kar vključuje tudi pisanje HTML kode in JavaScripta, kar pa mi je bilo pri GWT prihranjeno.

Spletna aplikacija je zgrajena tako, da jo lahko brez večjih sprememb dosedanjega dela lahko enostavno nadgradimo. Predstavitveni in poslovni nivo (programska koda aplikacije) sta ločena, tako da lahko izkoriščam vse prednosti objektno usmerjenega programiranja.

Aplikacijo bi lahko še nadgradil tako, da bi se bolj posvetil obliki grafičnega vmesnika administracijskega dela – ga naredil bolj prijaznega do uporabnika in s tem še izboljšal funkcionalnost. Prav tako je še veliko možnosti za razvoj dodatnih modulov in realizacijo idej, ki se bodo pojavile v prihodnosti.

Spletna aplikacija še ni bila vpeljana v delovno okolje revije, saj še ni bila dokončana, ker se v kratkem pripravlja nova grafična ter vsebinska podoba tiskane različice revije, čemur bo sledila tudi spletna stran ter v nekaterih delih tudi spletna aplikacija.

5. Priloge

5.1. Seznam slik

Literatura

[1] Ed Burnette. *Google Web Toolkit: Taking the Pain Out of Ajax,* 2006.

[2] Dave Crane, Eric Pascarello. *Ajax in Action*, Manning, oktober 2005.

[2] Jeff Dwyer. *Pro Web 2.0 Application Development with GWT,* Apress, maj 2005.

[4] Google Web Toolkit – Google Code. Dostopno na: http://code.google.com/webtoolkit/

[5] Google Web Toolkit - Developer's Guide. Dostopno na: http://code.google.com/docreader/#p=google-web-toolkit-doc-1-5&s=google-web-toolkit-doc-1-5&t=DevelopersGuide