

Projekt iz spletnega programiranja – 2. faza:

Izvedba strežniškega dela aplikacije

December 2016, v1.0

Rok oddaje na učilnici: 15. 1. 2017, možni (in priporočeni) predhodni zagovori.

Predavatelj: dr. Aleš Smrdel

Asistenta: Matevž Jekovec, dr. Gašper Fele-Žorž

Uvod

V drugi fazi boste nadaljevali z delom na spletni aplikaciji in razvili strežniški del. Razvoj je v grobem sestavljen iz naslednjih korakov: (I) načrtovanje podatkovnega modela in diagrami poteka s pripadajočimi URL-ji, (II) izvedba podatkovnega modela v aplikaciji, (III) izvedba poslovne logike in pogledov in (IV) objava, testiranje ter dokumentacija aplikacije.

Izbira primerne ogrodja

Izberite ogrodje, ki se vam zdi najbolj primerno in upošteva načela MVC (angl. *model-view-controller*) pristopa razvoja aplikacij. Nekaj priporočenih ogrodičev za izbrane programske jezike:

- **.NET:** ASP.NET in ASP.NET MVC;
- **Ruby:** Ruby on Rails, Rack, Sinatra, Lotus;
- **Javascript:** Node.js, Meteor, MEAN, DerbyJS, Sails.js;
- **PHP:** Codeigniter, Zend framework, CakePHP, Symfony, Laravel, Nette;
- **Python:** Django, Flask, Pyramid;
- **Java:** Spring, Dropwizard, Play, Spark.

Načrtovanje podatkovnega modela

Glede na namen, zasnovu in načrt aplikacije, ki ste jih definirali v prvi fazi, boste v drugi fazi načrtali podatkovni model in ustrezno pripravili strukturo podatkovne zbirke, vključno s:

- tabelami (z ustreznim indeksiranjem, omejitvami in relacijami),
- pogledi, procedurami in funkcijami,
- prožilci (angl. *triggers*),
- transakcijami.

Narišite ustrezno strukturo tabel z medsebojnimi relacijami. Uporabite lahko orodja za zasnovu in administracijo baz (MySQL Workbench, Microsoft SQL Server), splošno-namenski Graphviz s temo za izris baz (<https://code.djangoproject.com/wiki/DjangoGraphviz>) ali pa svinčnik in papir. Načrt podatkovnega modela shranite v mapo docs/.

Izvedba podatkovnega modela

Pri izvedbi podatkovnega modela glede na izbrano ogrodje in programski jezik uporabite ustrezen ORM (angl. *Object Relational Mapping*). Običajno imajo ogrodja ORM že vključen oz. obstaja priporočen ORM za dotičen programski jezik. V programski kodi definirajte razrede in njihove attribute, ki se preslikajo v ustrezno tabelo in stolpec.

ORM zagotavlja robustnejše delovanje aplikacije in enostavnejši razvoj, vendar pri velikih aplikacijah lahko upočasni njihovo delovanje. Zgodi se tudi, da pri nekaterih poizvedbah ORM ni dovolj in morate ročno zgraditi poizvedbo SQL. Dober ORM vsebuje bogato zbirko že vgrajenih operacij, obenem pa omogoča čisto vgradnjo lastnih poizvedb SQL.

Pri oddaji naloge morate priložiti eno ali več zagonskih skript SQL, ki vzpostavijo relacijsko bazo. Skripte lahko pišete ročno, lahko uporabite orodja za načrtovanje baz, najbolje pa je, da jih zgenerira vaše ogrodje neposredno iz podatkovnega modela (npr. ukaz `migrate` v Rails in Django). Skripte shranite v mapo `install/`.

Diagram poteka in preslikava naslovov

Pri laboratorijskih vajah v prvi fazi smo risali diagrame poteka uporabe aplikacije z vidika uporabnika. Sestavite celoten diagram za vse vloge uporabnikov v vaši aplikaciji in k vsakemu stanju v diagramu pripišite pripadajoč relativen URL. Za risanje diagrama lahko uporabite prosto dostopna spletna orodja (npr. www.draw.io) ali pa namizne aplikacije (npr. ArgoUml). Diagram poteka oddajte kot sliko ali PDF skupaj z dokumentacijo projekta v mapo `docs/`.

Izvedba poslovne logike, ACL, obrazci in beleženje

Poslovna logika povezuje podatkovni model in pogled ter glede na vsebino seje uporabnika krmili delovanje aplikacije. Pri spletnih aplikacijah je vsaka vstopna točka oz. akcija pokrita z enim izmed krmilnikov (angl. *controllers*). Vaša naloga je, da glede na načrt strani zasnujete in sprogramirate vse krmilnike in s tem poslovno logiko.

Določite, katere akcije v vaši aplikaciji lahko prožijo posamezne vrste uporabnikov (administrator, navaden uporabnik, ...) in kateri elementi so za njih vidni. Omenjeno obnašanje sprogramirajte z uporabo ACL (angl. *access control list*), ki je običajno že vgrajen v izbrano ogrodje.

Posebno pozornost namenite obrazcem. Vse vnose ustrezno validirajte na strani strežnika. Validacija na strani brskalnika ne zagotavlja veljavno izpolnjenih vnosnih polj, saj zahtevek HTTP uporabnik brez težav sestavi sam, mimo brskalnika, in ga pošlje na strežnik.

Premislite, katera mesta v vaši aplikaciji so kritična in sprogramirajte ustrezno beleženje (angl. *logging*). Npr. če programirate spletno trgovino, je kritičen korak oddaja naročila. V primeru programerske napake je lahko oddaja naročila neuspešna. V tem primeru morate zagotoviti, da je v dnevniških datotekah dovolj podatkov, da administrator lahko restavrira naročilo in ga obdela oz. povpraša stranko o vsebini naročila. Za beleženje uporabite obstoječe mehanizme, ki vam jih prinaša ogrodje. Običajno na enem mestu v konfiguracijski datoteki aplikacije nastavite beleženje v datoteko, v podatkovno zbirko, obvestilo po elektronski pošti itd.

Ravno tako poskrbite za konsistentnost vaše podatkovne baze. npr. pri spletni trgovini proces oddaje naročila vsebuje več korakov (podatki glede plačila, naslov za dostavo, končna potrditev). Vsak korak »dogradi« košarico in ločeno komunicira s podatkovno bazo. Lahko uporabite transakcije in v primeru, da uporabnik ne želi oddati naročila, transakcijo stornirate.

Izvedba pogleda

Pogled razvijte tako, da je popolnoma ločen od poslovne logike (krmilnika) in podatkovnega modela. Smiselno naj uporablja gradnike, ki jih boste pripravili: zaglavje strani, noga strani, tabelarična struktura (ali

kakšna druga, odvisno kako ste si stran zamislili) za prikaz podatkov, del strani za prikaz podatkov o uporabniku ipd.

Pri gradnji končne kode HTML uporabite ustrezen pogon za predloge (angl. *templating engine*), ki vam ga ponuja izbrano ogrodje. Ta lahko teče tudi na strani odjemalca (npr. AngularJS).

CSS iz prve faze lahko sedaj opišete posredno z jezikom Sass ali Less, ki vam zagotovita ves čas veljavno skladno CSS in podprtost na širokem naboru brskalnikov.

Aplikacijo prevedite v vsaj en dodaten jezik. Uporabite mehanizme za lokalizacijo, ki vam jih ponuja ogrodje oz. programski jezik.

Dokumentacija

Uredite dokumentacijo vašega projekta. V datoteko README.md ali INSTALL.md opišite postopek, potreben za objavo vaše aplikacije s povezavami na skripte za namestitev, v kolikor jih imate.

Poleg tega lahko sestavite referenčni priročnik (angl. *reference guide*) iz vaše izvirne kode za bodoče razvijalce ali pa uporabniški priročnik (angl. *user's guide*) za končne uporabnike vaše aplikacije. Pri tem uporabite orodja za samodejno gradnjo priročnikov iz predloge (npr. sphinx, doxygen, autodoc itd.). Končno verzijo dokumentacije shranite v mapo docs/.

Testiranje in profiliranje

Izdelek temeljito preklikajte, preverite če so vsi elementi skladni, če koda deluje pravilno in če se vsi podatki prikazujejo pravilno. Preverite, da lahko uporabniki dostopajo le do njim dovoljenih vsebin.

Predvsem za testiranje modela in poslovne logike uporabite testiranje enot v izbranem programskem jeziku. Za preverjanje pravilnega delovanja funkcionalnosti strani prek uporabniškega vmesnika uporabite katero od orodij za funkcionalno, regresijsko in integracijsko testiranje (z uporabo Selenium, Windmill, CasperJS itd.).

Vse teste shranite v ustrezno mapo, ki jo določa vaše ogrodje.

Za ugotavljanje odzivnosti vaše spletne aplikacije izvedite performančne teste (z uporabo JMeter, Selenium, Gatling, Tsung, The Grinder itd.). Izmerite pričakovano število zahtevkov na sekundo, ki jih lahko vaša aplikacija obdela. Nato z uporabo orodij za profiliranje v izbranem jeziku ugotovite ozka grla v vaši aplikaciji (npr. cProfile, valgrind, Visual Studio Profiling Tools itd.). Izsledke profiliranja shranite v mapo docs/.

Objava, zvezna integracija in optimizacija

Pripravite skripte za objavo vaše aplikacije na »produkcijskem« strežniku oz. izbranem oblaku. Uporabite skripte za zvezno integracijo Puppet oz. Chef. Če uporabljate oblak Heroku, uporabite in prilagodite njim lastne skripte.

Aplikacijo lahko objavite namesto na strežnik oz. oblak tudi lokalno v izolirano okolje Docker ali Vagrant. V tem primeru morate poskrbeti za namestitev vseh odvisnosti, ki jih vaša aplikacija potrebuje.

Pri objavi poskrbite, da bo vsa vsebina pravilno prevedena (npr. izvirne datoteke Sass/Less v CSS, koščki pogledov v končni pogled, morebitni popravki podatkovne zbirke itd.). Poleg tega v produkcijskem okolju poskrbite za optimalno delovanje vaše spletne aplikacije. Preučite in uporabite mehanizme za predpomnjenje (angl. *caching*) bodisi na nivoju spletnega strežnika, na aplikacijskem nivoju ali na nivoju podatkovnega modela. Poskusite optimizirati velikost vaših virov (npr. optimalne velikosti slik glede na velikost zaslona odjemalca, pomanjšana koda javascript – angl. *minify*). Stran lahko optimizirate tudi tako, da odjemalec v enem zahtevku pridobi celotno spletno stran z vsemi viri, vgrajenimi kar v datoteki HTML. Za doseg tega cilja lahko uporabite tudi HTTP2 v vašem spletnem strežniku in ustrezno pošiljate vsebino.

Vse skripte, namenjene objavi aplikacije, shranite v mapo install/.

Končni izdelek druge faze

Končni izdelek oddajte v zapisu ZIP, zgeneriranem iz repozitorija git. Poleg tega v datoteko ZIP vključite tudi zgenerirano dokumentacijo projekta. Datoteka naj vsebuje:

- Dokumentacijo projekta v mapi doc/:
 - načrt podatkovnega modela.
 - zagonske skripte SQL za vzpostavitev relacijske baze.
 - navodila za namestitev spletne aplikacije (zahtevano programsko okolje, postavitve strežnika, postavitve baze, namestitve aplikacije).
 - diagram in opis vašega krmilnega dela aplikacije s pripadajočimi vstopnimi točkami (mesta uporabe HTTP GET/POST, oblika rezultata HTML ali JSON, uporabljeni pogledi in krmilnik itd.).
- Testi:
 - Kratko poročilo o tipu predpomnjenja;
 - funkcionalnem, regresijskem in performančnem testiranju;
 - Konfiguracija za zvezno integracijo (angl. *Continuous Integration*);
 - HTTP2.
- Izvorno kodo celotne spletne aplikacije s pripadajočimi knjižnicami.

Kriteriji za ocenjevanje

V drugi fazi je možno doseči 125 točk. Pogoji za uspešno opravljanje vaje je v povprečju zbranih vsaj 50 % točk pri obeh fazah (pri posamezni fazi lahko zberete tudi manj kot 50 %, vendar mora biti zato preostala faza toliko boljše narejena).

Točkovačnik za osnovne zahteve pri drugi fazi je naslednji:

Načrt podatkovnega modela	10
Izvedba podatkovnega modela v izbranem ogrodju in migracije	10
Diagram poteka in preslikave naslovov (angl. <i>routes</i>)	5
Uporaba ORM	5
Izvedba poslovne logike v izbranem ogrodju in njena ločitev od modela/pogleda	10
Validacija obrazcev na strani strežnika	5
Nadzor dostopa uporabnikov do virov (angl. <i>access control list</i>)	10
Izvedba pogleda v izbranem ogrodju s pogonom za predloge (angl. <i>templating engine</i>)	10
Dobra programerska praksa, komentarji, izbrana konvencija programiranja	10
Lokalizacija	5
Beleženje z uporabo izbrane metode (angl. <i>logging</i>)	5
Referenčni priročnik ali uporabniški priročnik za aplikacijo (sphinx)	5
Testi enot	5
Funkcionalni, regresijski, integracijski testi (Selenium)	5
Performančni testi in profiliranje (Selenium, JMeter)	10
Objava aplikacije in zvezna integracija (puppet, chef, po potrebi skripte Bash in SQL)	5
Optimizacija nalaganja strani (predpomnjenje, združevanje vsebine v en odgovor, HTTP2, minify)	10

Viri

Pri izdelavi si pomagajte s predlagano literaturo predmeta, tu pa je še nekaj povezav:

- Sass: <http://sass-lang.com/>
- Less: <http://lesscss.org>
- Optimizacija strani: <http://designingforperformance.com/basics-of-page-speed>
- Regresijsko testiranje: <http://smartbear.com/resources/articles/what-is-regression-testing/>
- Selenium: <http://www.seleniumhq.org/>
- JMeter: <http://jmeter.apache.org/>