# Projekt – 2. faza:

# Izvedba strežniškega dela aplikacije

## Spletno programiranje

## Oktober 2015

Rok oddaje na učilnici: 17. 1. 2016, možne (in priporočene) predhodne oddaje in zagovori.
Predavatelj: dr. Aleš Smrdel
Asistenta: Matevž Jekovec, Žiga Emeršič

# Uvod

Pri predmetu Spletno programiranje tekom semestra razvijate lastno spletno aplikacijo, katere razvoj je sestavljen iz dveh faz. V prvi fazi ste že razvili spletno stran s statično vsebino, ki nima posebnih funkcionalnosti, predstavlja pa osnutek za drugo fazo.
V drugi fazi pa boste razvili še strežniški del aplikacije. Razvoj strežniškega dela je sestavljen iz naslednjih korakov: (I) načrtovanje podatkovnega modela, (II) izvedba podatkovnega modela v aplikaciji, (III) izvedba poslovne logike in pogledov in (IV) objava ter testiranje aplikacije.

# Izbira primernega ogrodja

Izberite ogrodje, ki se vam zdi najbolj primerno in upošteva načela MVC (angl. *model-view-controller*) pristopa razvoja aplikacij. Nekaj priporočenih ogrodij 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, Flash, Pyramid;
- **Java**: Spring, Dropwizard, Play, Spark.

# Načrtovanje podatkovnega modela

Glede na namen, zasnovo in načrt aplikacije, ki ste jih definirali v prvi fazi, boste v drugi fazi načrtali podatkovni model in ustrezno pripravili strukturo podatkovne baze, kar vključuje:

- tabele (z ustreznim indeksiranjem, omejitvami in relacijami),
- poglede, procedure in funkcije,
- prožilce (angl. *triggers*),
- transakcije.

Narišite ustrezno strukturo tabel z medsebojnimi relacijami. Uporabite lahko orodja za zasnovo in administracijo baz (MySQL Workbench, Microsoft SQL Server) ali pa splošno-namenski Graphviz s temo za izris baz (https://code.djangoproject.com/wiki/DjangoGraphviz).

## Izvedba podatkovnega modela

Pri izvedbi podatkovnega modela imate dve možnosti:

- **SQL**: Lahko ga razvijete neposredno s pomočjo SQL-a, pripravite pripadajoče procedure, funkcije, poglede itd. Na njih se nato sklicujete v modelu vaše spletne aplikacije in dobljene rezultate iz baze ustrezno pretvarjate v podatkovne strukture na aplikacijskem nivoju.
- **ORM**: Glede na izbrano ogrodje in programski jezik uporabite *ORM (angl. Object Relational Mapping)*. ORM je običajno že del izbranega ogrodja. V programski kodi morate nato definirati razrede in njihove atribute, ki se preslikajo v ustrezno tabelo in stolpec.
  Če ne uporabljate ogrodja, lahko uporabite enega izmed naslednjih knjižnic ORM:
  - **Ruby**: ActiveRecord;
  - **Java**: Hibernate, JPA;
  - **PHP**: DataMapper, Doctrine;
  - **Python**: SQLAlchemy.

Vsak izmed načinov ima svoje prednosti in slabosti. Pri uporabi SQL imate več nadzora in pogosto hitrejše delovanje, vendar hkrati tudi več možnosti za napake in težje obvladljivo kodo. ORM zagotavlja robustnejše delovanje in enostavnejši razvoj, vendar pri velikih aplikacijah lahko upočasni delovanje. Hkrati se lahko zgodi, da pri zahtevnejših poizvedbah ORM ni dovolj, tako da morate v nekaterih primerih ročno zgraditi poizvedbo SQL. Dober ORM vsebuje bogato zbirko že vgrajenih operacij, obenem pa omogoča čisto vgradnjo lastnih poizvedb SQL.
Ne glede na izbran način izvedbe podatkovnega modela morate ob oddaji priložiti zagonsko skripto SQL, ki vzpostavi relacijsko bazo. Skripto lahko napišete ročno, lahko uporabite orodja za načrtovanje baz, lahko pa jo zgradi že vaše ogrodje (npr. ukaz `migrate` v Rails in Django).

## Izvedba poslovne logike

Poslovna logika povezuje podatkovni model in pogled ter glede na vsebino sejo 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.

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.

Dodatno lahko za učinkovitejše delovanje spletne strani poskrbite za predpomnjenje (angl. *caching*), bodisi na nivoju spletnega strežnika, na aplikacijskem nivoju ali na podatkovnem nivoju.

## 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.

CSS iz prve faze lahko sedaj nadgradite z jezikom Sass (angl. *Syntactically Awesome Stylesheets*) in drugimi podobnimi jeziki.

Pripravite minimizacijo (angl. *minify*) kode JavaScript in CSS.

Dodatno lahko stran optimizirate tako, da se vsi podatki pošiljajo v enem zahtevku (http://designingforperformance.com/basics-of-page-speed).

## Testiranje

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.

Dodatno lahko za preverjanje pravilnega delovanja funkcionalnosti strani prek uporabniškega vmesnika uporabite katero od orodij za funkcionalno in regresijsko testiranje: Selenium, Windmill, CasperJS itd.

Izmerite lahko tudi hitrost delovanja vaše spletne strani. Za to uporabite orodja, kot so Jmeter, Selenium, Gatling, Tsung, The Grinder itd.

## Končni izdelek druge faze

Končni izdelek oddajte v arhivski datoteki ZIP. Ta naj vsebuje:

- Dokumentacijo projekta v mapi doc/:
  - Vizualizacija podatkovnega modela.
  - Zagonska skripta SQL za vzpostavitev relacijske baze.
  - Navodila za namestitev spletne aplikacije (zahtevano programsko okolje, postavitev strežnika, postavitev baze, namestitev 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.).
  - Dodatno: Kratko poročilo o tipu predpomnjenja; funkcionalnem, regresijskem in performančnem testiranju.
- Izvorno kodo celotne spletne aplikacije.

## Kriteriji za ocenjevanje

V drugi fazi je možno doseči 125 točk. Pogoj za uspešno opravlje 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).

Točkovalnik za osnovne zahteve pri drugi fazi je naslednji:

| | |
|---|---|
| Načrt podatkovnega modela | 5 |
| Izvedba podatkovnega modela v izbranem ogrodju (za ORM bi posebej povedal koliko točk več prinese kot SQL) | 10 |
| Izvedba poslovne logike v izbranem ogrodju | 10 |
| Izvedba pogleda v izbranem ogrodju | 10 |
| Validacija obrazcev na strani strežnika | 5 |
| Nadzor dostopa uporabnikov do virov (angl. *access control list*) | 10 |
| Popolna ločitev modela, pogleda in poslovne logike | 5 |
| Beleženje z uporabo izbranega ogrodja (angl. *logging*) | 5 |
| Dokumentacija za namestitev in zagonska skripta SQL | 5 |
| Dobra programerska praksa, komentarji, izbrana konvencija programiranja | 10 |

Dodatne zmožnosti aplikacije, zahtevane za nadpovprečno opravljene vaje (ocena 9 in 10) ter bonus točke:

| | |
|---|---|
| Iskanje po celotni vsebini spletne strani | 10 |
| Predpomnjenje na vsaj dveh nivojih | 10 |
| Funkcionalni, regresijski testi (Selenium) | 10 |
| Performančni testi (JMeter, Selenium) | 10 |
| Zmanjšano število zahtevkov do strani (združevanje vsebine v en odgovor) | 10 |

# Viri

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

- http://sass-lang.com/
- http://designingforperformance.com/basics-of-page-speed
- http://smartbear.com/resources/articles/what-is-regression-testing/
- http://www.seleniumhq.org/
- http://jmeter.apache.org/