UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Denis Mesarič Štih

Mobilni večuporabniški planer

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: viš. pred. dr. Aleksander Sadikov

Ljubljana, 2015

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Denis Mesarič Štih

Mobile multi-user task manager

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: viš. pred. dr. Aleksander Sadikov

Ljubljana, 2015

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge: Mobilni večuporabniški planer

V diplomski nalogi predstavite postopek izdelave preprostega spletišča ter ga tudi implementirajte. Spletišče naj bo namenjeno skupinskem planiranju opravil in naj se prilagaja velikosti zaslona za potrebe uporabe na mobilnih napravah. Uporabite ogrodje ASP.NET z implementacijo arhitekturnega vzorca Model-pogled-kontroler (MVC).

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Denis Mesarič Štih sem avtor diplomskega dela z naslovom:

Mobilni večuporabniški planer

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Aleksandra Sadikova
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 10. februarja 2015 Podpis avtorja:

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
MVC	Model-View-Controller	model-pogled-kontroler
HTML	HyperText Markup Language	jezik za označevanje nadbesedila
CSS	Cascading Style Sheets	metoda podpornih vektorjev
SQL	Structured Query Language	strukturiran povpraševalni jezik za delo s podatkovnimi bazami
ASP	Active Server Pages	microsoftova tehnologija, ki omogoča aktivno kreiranje spletnih strani na našem strežniku
UML	Unified Modeling Language	splošno namenski modelni jezik

Povzetek

V diplomski nalogi je opisan celoten postopek izdelave preprostega spletišča za planiranje opravil (angl. to-do list) - od analize do kodiranja.

Začne se z analizo sistema, kjer so prikazani primeri uporabe (angl. Use Cases). Vsak primer uporabe se potem podrobno razčleni. Uporabljajo se za opis delovanja spletišča. V tej fazi je odgovorjeno na vprašanje: »Kaj naj sistem počne?«

Opisan in prikazan je tudi primer načrtovanja izgleda strani spletišča s pomočjo prototipov (angl. Mock-up).

Sledi opis podatkovne baze, ki se uporablja pri projektu. Predstavljen je diagram oz. model podatkovne baze, ki predstavlja vse tabele in njihove medsebojne povezave. Pomen tabel je opisan tudi v tekstu.

V zadnjem delu so opisani arhitektura in uporabljeni vzorci sistema. Med njimi je glavni vzorec Model-pogled-kontroler. Ta predstavlja način programiranja spletišča oz. način, kako optimalno strukturiramo elemente programa in njihove povezave. Uporablja se za bolj pregledno kodo in večjo prilagodljivost za spremembe ali nadgradnje.

Ključne besede: MVC, spletno programiranje, podatkovna baza, analiza sistema, primeri uporabe, arhitektura sistema

Abstract

This thesis describes the entire process of making a simple site for planning tasks (also called a To-do list) - from analysis to coding.

It begins with an analysis of the system, which shows examples of use (Use Cases). Each use case is then broken down in detail. They are used to describe the functionality of the website. In this phase we get an answer to the question: "What is the system supposed to do?"

It also contains a description of the planning stages for the visual aspect of the website. Followed by a comparison between the prototype (Mock-up) and final design of the website.

The next part describes the database used in the project. In addition, there is a diagram (model) of the database that represents all the tables and their interconnections. The meaning of each individual table is described in the text.

The final part of the thesis contains a description of the architecture and the patterns used to create the system. The main pattern used is Model-View-Controller. MVC is a way of programming the website i.e. a way to optimize the structure of the programs elements and their connections. It is used to make the code more understandable. Furthermore it increases the flexibility of the program for future changes or upgrades.

Keywords: MVC, web programming, database, system analysis, Use Cases, system architecture

Poglavje 1 Uvod

V diplomski nalogi je opisan sistematičen razvoj spletišča za vodenje evidence opravil. Cilji, ki sem jih v diplomski nalogi zasledoval so bili:

- Določiti primerne inženirske prijeme in metode za izdelavo preprostega spletišča,
- dokumentirati uporabniške zahteve, arhitekturo in načrt spletišča,
- na podlagi izdelane dokumentacije razviti načrtovano spletišče,
- za razvoj uporabiti sodobne Microsoftove tehnologije: C#, ASP.NET, Entity Objects, Razor in vzorec MVC in pridobiti izkušnje z njimi,
- razširiti spletišče tako, da deluje na namiznih računalnikih in na mobilnih napravah ter se prilagaja okolju.

Načrtovanje spletišča se začne z analizo primerov uporabe. Z njo skušamo določiti uporabniške zahteve. Vsak primer uporabe se posamično razčleni do globine, ko je dovolj razumljiv da lahko na njegovi podlagi oblikujemo zaslone (angl. Mock-ups) in modeliramo podatke. V diplomskem delu je predstavljen način oblikovanja izgleda spletišča s pomočjo prototipov. V nadaljevanju je prikazan podakovni model, ki je zgrajen iz objektnega po metodi »najprej kodiranje« (angl. Code-First).

Na podlagi teh dokumentov je razvito spletišče. Z primerno izbiro tehnologij (predvsem tehnologije Bootstrap) je zagotovljeno, da do spletišča uporabnik lahko dostopa z večino današnjih mobilnih naprav. Vse kar potrebuje je spletni brskalnik.

Samo spletišče uporabniku omogoča: registracijo in avtentikacijo, dodajanje in shranjevanje svojih delovnih nalog, poročanje o njihovem izvajanju, ustvarjanje skupin in njihovo vodenje, dodeljevanje delovnih nalog članom skupine, pridruževanje delovnim skupinam in izvajanje delovnih nalog, ki so mu jih dodelili vodje drugih skupin.

Za izdelavo spletišča so bile uporabljene naslednje tehnologije in vzorci:

- programiranje v jezikih C#, JavaScript, HTML, jQuery, CSS,
- Microsoftovo ogrodje Entity Framework,
- strežnik SQL server in pripadajoča orodja,
- vzorec ASP.NET MVC in
- ogrodje Bootstrap.

Dodaten motiv za diplomsko delo je možnost, da se nanj sklicujem pri iskanju dela po zaključku študija. Pri razvoju sem zasledoval načeli enostavnosti in minimalnosti, kot se poslovnem okolju od zaposlenega pričakuje.

Poglavje 2 Uporabljene tehnologije

V tem poglavju so opisane uporabljene tehnologije. Zanje sem se odločil skladno s cilji diplomske naloge:

- ker so primerne za problem, ki ga rešujem in
- ker gospodarstvo v tem trenutku ta znanja pričakuje.

2.1 C#

C# je objektno orientiran programski jezik. Razvil ga je Microsoft leta 2001 za razvoj v okolju .NET [1]. Sintaksa jezika C# blago temelji na jeziku C++, pri čemur je C# za neizkušene programerje bistveno prijetnejši za delo in predvsem varnejši od C++. Poglavitno zato, ker je eden največjih virov programskih hroščev – upravljanje in čiščenje pomnilnika (angl. Garbage collection) – prepuščeno programskemu jeziku oziroma okolju .NET.

Zadnja različica jezika C# omogoča vse sodobne programerske prijeme: objektno orientirane metode (razrede, dedovanje, polimorfizem, itn.), močne tipe, prenosljivost ter deskriptivno programiranje in funkcionalno programiranje.

Zaradi enostavnosti in čistosti jezika, obsežne .NET knjižnice, dobre dokumentacije in zelo dobrega razvojnega okolja Visual Studio je programiranje z njim mnogo bolj učinkovito, kot v okolju C++ in je mogoče rezultate doseči zelo hitro.

C# je najpopularnješi programski jezik za programiranje v .NET. C# temelji na odprtih standardih ECMA (ECMA 344) in je v omejenem obsegu na voljo na večih platformah [1]. Najnovejša različica jezika je trenutno različica 5.0. Različica v pripravi pa je 6.0.

2.2 HTML

HTML (angl. HyperText Markup Language) je standardni označevalni jezik za spletno programiranje [2, 18]. Ena izmed poglavitnih funkcij spletnega brskalnika je, da spletno stran v jeziku HTML (in skriptnih tehnologijah) prikaže uporabniku na prijazen način. Prikaz elementov HTML se lahko med brskalniki razlikuje, vedar je ta težava z vsako novo različico HTML manjša. V različici jezika HTML 5.0 so razlike minimalne.

HTML je drevesno strukturiran oziroma njegove elemente lahko gnezdimo. Večina elementov nastopa v paru, kot začetna in končna oznaka. Začetna oznaka se zapiše tako, da v par kotnih oklepajev napišemo ime elementa. Pri končni oznaki pa pred ime dodamo poševnico. Med parom oznak piše vsebino npr. tekst, ki se pojavi na sami strani (primer: `<p>Ta vsebina predstavlja odstavek</p>`).

Obstajajo tudi izjeme, kjer se element piše samo z eno oznako (primer: `<br/>`).

Elementom lahko določimo lastnosti, ki so znotraj začetne oznake (primer: `<p align="left"></p>`).

V HTML dokumentu ponavadi najdemo vsaj tri glavne elemente. To so: *html*, *head* in *body*. *html* element se uporablja za označitev začetka in konca celotnega dokumenta. V njem sta vgnezdena *head* in *body* elementa. *head* element ponavadi definira informacije o dokumentu. V *body* element je pa vgnezdena celotna vsebina dokumenta. Primer preprostega HTML dokumenta:

```
<html>
  <head>
    <title>Hello HTML</title>
  </head>
  <body bgcolor=white>
       <p>This is the home of the Hello HTML web page. </p>
  </body>
</html>
```

HTML nudi omejen nabor možnosti za interaktivnost (npr. HTML forme). Zato se za uporabniško interakcijo danes večinoma uporablja v kombinaciji z jezikoma JavaScript in CSS.

2.3 CSS

CSS (angl. Cascading Style Sheets) je jezik, ki se uporablja za fleksibilnejši opis izgleda spletnih strani v kombinaciji z jezikom HTML [3, 18]. Z jezikom CSS se lahko brskalniku pove kakšen izgled naj ima določen HTML element.

Omogoča nam povezati HTML elemente z njihovim načinom izrisovanja. Po oznaki, razredu ali id lastnosti lahko vsakemu HTML elementu določimo pozicijo, barvo, font teksta ipd.

CSS lahko v HTML dokumente dodamo na dva načina:

- z uporabo HTML elementa style znotraj samega HTML dokumenta.
- kot samostojno datoteko na katero dodamo povezavo v HTML dokument.

Osnovni gradnik sintakse CSS je selektor. Z njim določimo, za kateri HTML element (id ali razred) določamo lastnosti oz. stil. Nato pa sledijo lastnosti oz. stil, ki so vnešene med parom zavitih oklepajev in ločene s podpičji.

Npr. če želimo določiti, da bodo v dokumentu vsi elementi h1 rdeči in levo poravnani, tedaj sta selektor in lastnosti zapisana takole: `h1 { color:red; text-align:left; }`.

V arhitekturnem smislu nam kombinacija HTML in CSS omogoča ločiti strukturo dokumenta od samega izgleda.

2.4 JavaScript

JavaScript se uporablja v kombinaciji z HTML in CSS, da se stranem doda dinamičnost in interaktivnost.

Primeri, kako lahko z JavaScript obogatimo spletno stran so npr. [4, 18, 19]:

- preproste animacije,
- sortiranje tabel brez ponovnega nalaganja celotne strani,
- odzivanje na določene dogodke (npr. ko se stran naloži sprožimo animacijo),
- dinamično spreminjanje teksta na strani,
- preverjanje pravilnosti uporabnikovega vnosa itd.

Večina današnjih spletnih brskalnikov že v osnovi podpira JavaScript zato za uporabo ali delo z njim ne potrebujemo dodatnih orodij. Za uporabo v HTML kodi se uporabi element *script* pri katerem se lahko med začetno in končno oznako piše kodo JavaScript ali pa samo vstavi povezavo na zunanjo datoteko, ki vsebuje kodo Javascript (primer: `<script src="mojJavaScript.js"></script>`).

JavaScript se izvaja na odjemalčevi strani. S tem se lahko del procesiranja prenese iz strežnika na odjemalec in razbremeni strežnik. Novejše tehnologije, kot npr. Node.js nudijo tudi možnost uporabe JavaScript na strežnikovi strani.

> *Poleg spletnega programiranja se JavaScript lahko uporablja za razvoj aplikacij in iger. Sodobna orodja namreč znajo JavaScript kodo spremeniti v aplikacije za najpopularnejše mobilne operacijske sisteme kot so Android in iOS.*

2.4.1 jQuery

JQuery je ena izmed najpopularnejših knjižnic za JavaScript, podprta v praktično vseh najbolj razširjenih spletnih brskalnikih [5, 19]. JQuery olajša in pohitri uporabo programskega jezika JavaScript pri spletnem programiranju.

Ključna lastnost knjižnice jQuery je lažji dostop do HTML DOM (angl. Document Object Model) elementov. HTML DOM elementi zajemajo [6]:

- sam dokument,
- vse HTML elemente,
- vse HTML lastnosti,
- tekste znotraj HTML elementov in
- komentarje.

Osnovna sintaksa jQuery vsebuje tri stvari:

1. Na začetku se doda znak »$«, ki služi kot oznaka za začetek sintakse jQuery.
2. Nato sledi selektor t.j. element nad katerim se bo izvajala akcija. Selektor se piše znotraj para okroglih oklepajev.
3. Tretja stvar osnovne sintakse pa je akcija. Akcija je v osnovi funkcija, ki se izvede nad selektorjem (primer, ki skrije HTML element h1: `$("h1").hide()`).

JQuery je bližnjica do funkcionalnosti, ki bi v jeziku JavaScript zahtevale veliko programiranja. Poleg manipuliranja s strukturo spletne strani preko HTML DOM se pogosto uporablja za pisanje elegantne kode za odziv na dogodke ali animacije na spletni strani.

2.5 SQL

SQL (angl. Structured Query Language) je programski jezik četrte generacije. Uporablja se ga za povpraševanje (in za upravljanje) v relacijskih podatkovnih bazah [7]. Jezik SQL omogoča strukturirane poizvedbe z uporabo relacijske algebre.

> *Relacijska podatkovna baza je organizirana zbirka podatkov, tipično v obliki množice tabel, relacij, indeksov, pogledov, procedur in sprožilcev. Uporablja se za shranjevanje podatkov in pohitritev iskanj [8].*

Jezik SQL se običajno izvaja na sistemu za upravljanje podakovnih baz (angl. database management system), ki običajno deluje kot strežnik. Za izdelavo programske rešitve je uporabljen Microsoft SQL strežnik. Poleg strežnika je bil pri razvoju uporabljen SQL Server Management Studio, ki je del SQL strežnika.

SQL Management Studio omogoča celovit pregled nad podatkovno bazo. Olajša nam upravljanje podatkovnih baz, saj nudi tako konfiguriranje strežnika, kot izvajanje poizvedb in njihovo analizo ter optimizacijo. Ena izmed uporabnejših lastnosti orodja je generiranje diagramov v obe smeri (iz baze v diagram in iz diagrama v bazo), kar je zelo priročno za predstavitev podatkovne baze.

Ker podatkovna baza za mobilni večuporabniški planer sodi v kategorijo majhnih podatkovnih baz bi verjetno lahko zanjo uporabil tudi manj močen strežnik, kot je SQL server. Med kandidati so bili baza MySQL, knjižnica SQLLite in SQL Server Express (manj zahtevna verzija SQL Serverja). Vendar je bil eden izmed ključnih ciljev diplomske naloge tudi, da osvojim nove in kompleksne tehnologije. Zato sem se odločil za SQL Server.

Na koncu je programska rešitev zahtevala uporabo le manj kompleksnih gradnikov SQL baze: tabel, relacij in indeksov. Vsi podatki so SQL strežniku shranjeni v tabelah. Vsaka tabela mora imeti primarni ključ. Primarni ključ je stolpec v tabeli izbranega tipa (npr. int, string ipd.) pri katerem ima vsaka vrstica tabele unikaten vnos. Ponavadi se za primarni ključ uporabljajo samo generirane številke v obliki števca, ni pa nujno. Za primarni ključ v tabelah lahko uporabimo karkoli, dokler se vnosi ne podvajajo. Za primarni ključ lahko uporabljamo tudi več kot en stolpec. Tabele imajo povezave med seboj v obliki tujih ključev. Tuji ključi so v osnovi povezave na primarne ključe drugih tabel. Med tabelami so vzpostavljene relacije. Glavni ključi in pogosta iskalna polja so indeksirana, kar pohitri iskanje.

2.6 Entity Framework

Entity Framework je ogrodje za mapiranje objektov v programskem jeziku (C#) v tabele na SQL strežniku [9]. Pri izdelavi spletišča je bil uporabljen Code-First pristop. To pomeni, da se podatkovna baza ustvari na osnovi razredov, ki se uporabljajo v kodi projekta. Med postopkom razvijanja je ta opcija zelo priročna. S tem pristopom se podatkovna baza posodablja samodejno, ko pride do sprememb v uporabljenih razredih.

2.7 Bootstrap

Z ogrodjem Bootstrap lahko zelo hitro izdelamo spletišče, ki se prilagaja večini današnjih mobilnih naprav [10].

Bootstrap je eno izmed najpopularnejših ogrodij pri izdelavi spletišč. Z vgrajenimi elementi pohitri izdelavo vizualnega dela spletišča. Uporablja kombinacijo HTML, CSS, JS in jQuery. Najprivlačnejša lastnost ogrodja je, da ne glede na velikost zaslona ohrani kompozicijo elementov na logičen in lep način.

2.8 MVC

MVC (angl. Model-View-Controller) je hkrati Microsoftova tehnologija in arhitekturni vzorec, ki se uporablja pri izdelovanju aplikacij [11, 20]. Kot Microsoftova tehnologija je MVC predvsem način kako elegantno in ponovljivo strukturirati spletni projekt v okolju .NET. Kot vzorec pa je podrobneje predstavljen v poglavju arhitektura. Tam so prikazani primeri iz praktičnega dela diplomske naloge.

Poglavje 3 Analiza sistema

V tem poglavju so najprej opisane uporabljene metode in notacije za analizo sistema. Potem identificiramo primere uporabe sistema in vsakega posebej razčlenimo.

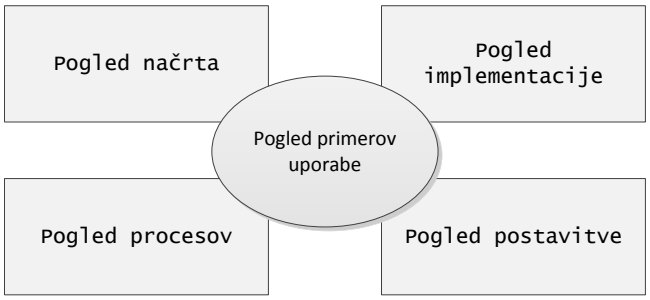
3.1 UML

UML (angl. Unified Modeling Language) je notacija, s katero lahko opišemo modele na področju programskega inženirstva na standarden način in z diagrami [12, 17].

3.1.1 UML je notacija

Čeprav generičnost UML omogoča opisovanje poljubnega programskega sistema in procesa – od modeliranja uporabniških zahtev (s primeri uporabe), algoritmov in procesov (diagram poteka/aktivnosti/interakcij/stanj) do strukture (razredni diagrami, diagrami komponent/namestitve/postavitve) - pa UML ne vsiljuje samega procesa razvoja programske opreme. To pomeni, da standard ne predpisuje nekega zaporedja uporabe diagramov ali cikla razvoja programske opreme. Predpisuje le osnovne elemente (diagrame, objekte in relacije) notacije in pravila njihovega povezovanja.

UML na vsak sistem gleda s petih različnih zornih kotov. Z zornega kota primerov uporabe so sistem njegova obnašanja, kot jih vidijo uporabniki. Z zornega kota načrta so sistem razredi, vmesniki in sodelovanje med njimi. Z zornega kota procesov so sistem niti in procesi ter usklajevalni mehanizmi. Z implementacijskega zornega kota pa so sistem strežniki, komponente in datoteke, ki so njegov sestavni del. Zorne kote sistema predstavlja spodnja slika (Slika 1). Za opisovanje vsakega izmed njih je na voljo nekaj diagramov [13].



Slika 1. Pet zornih kotov sistema, ki jih pokriva UML

3.1.2 Obseg rabe UML v diplomski nalogi

Obstajajo tudi formalni (npr. Rational Unified Process) in neformalni procesi in načini, kako si z UML diagrami pomagati pri načrtovanju sistema. Običajno se s primeri uporabe najprej specificira uporabniške zahteve. Posamičen primer uporabe se podrobneje opiše z diagrami poteka/aktivnosti. Iz njih pa je mogoče določiti minimalne razredne diagrame za zadovoljitev zahtev iz primerov uporabe. Vendar pa v diplomski nalogi takšna izpeljava ni bila smiselna, saj bi pripeljala do teoretičnega minimalnega objektnega modela – uporabljena orodja pa že predpisujejo kako morajo biti objekti strukturirani in na kakšen način morajo medsebojno komunicirati (MVC). Če bi želel, da bi se rezultati analize in načrta skladali s kodo bi moral od začetka vedeti, kaj želim in analizo in načrt usmerjati tja – to pa pomeni, da ne bi imela nikakršnega smisla.

Zato je bila notacija UML uporabljena za primere uporabe, nadaljna struktura programskega sistema pa je bila zgrajena neposredno iz primerov uporabe, brez vmesnih korakov. Bolj kot je sistem obsežen in kompleksen, bolj smiselno se zdi poglabljati analizo z drugimi inženirskimi prijemi. Načeloma je analizo in načrt smiselno poglabljati do stopnje, ko je sistem zadostno opisan za kodiranje.

3.2 Analiza primerov

Eden izmed učinkovitih postopkov kako do objektnega modela je, da se napiše esej o problemih, ki jih sistem rešuje. V obliki enostavčnih povedi. Nato se v njem podčrta vse samostalnike in vse glagole ter vse medsebojne povezave. Samostalniki postanejo razredi, glagoli postanejo metode, medsebojne povezave pa relacije med njimi.

Na podoben način je mogoče iz začetne zamisli razviti primere uporabe; pri čemur so samostalniki akterji in glagoli primeri uporabe.

Primeri uporabe za Mobilni večuporabniški planer so bili razviti iz naslednjega teksta:

»Modeliramo mobilni večuporabniški planer s katerim bodo uporabniki urejali in pregledovali svoja opravila. Uporabniki v sistemu lahko nastopajo kot obiskovalci (če se še niso registrirali), kot navadni uporabniki ali kot lastniki skupin. Lastnik skupine je navaden uporabnik, ki ustvari skupino uporabnikov. Lastnik skupine lahko v skupino povabi druge uporabnike. Uporabnik lahko ureja (ustvari, popravi, briše) opravilo zase ali za tiste skupine uporabnikov, katerih lastnik je. Če imajo uporabniki v skupini dovoljenje, pa niso lastniki, lahko tudi urejajo skupinska opravila. Uporabnik lahko označi opravilo kot zaključeno.«

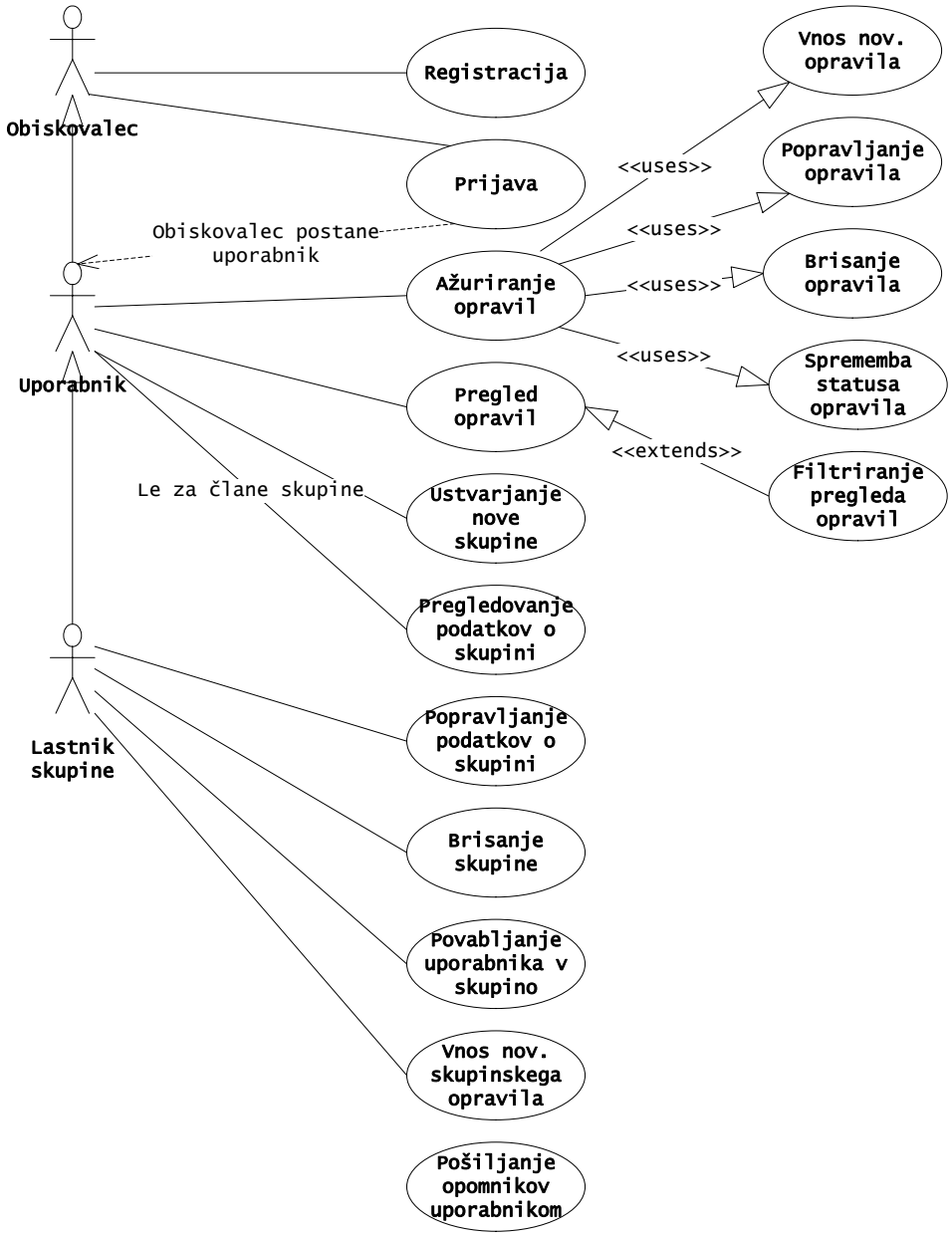
Samostalniki iz teksta so:

- Mobilni večuporabniški planer,
- Navadni uporabnik,
- Lastnik skupine,
- Obiskovalec,
- Opravilo,
- Skupinsko opravilo in
- Dovoljenje.

Glagoli iz teksta so:

- Urejanje (ustvarjanje, popravljanje, brisanje) svojih opravil,
- Pregled svojih opravil,
- Ustvarjanje skupine,
- Vabljenje uporabnikov v skupino,
- Urejanje (ustvarjanje, popravljanje, brisanje) skupinskih opravil in
- Pregled skupinskih opravil.

Na naslednjem diagramu (Slika 2) so združeni in nekoliko podrobneje opisani primeri uporabe.



Slika 2. Primeri uporabe sistema

3.3 Razčlenitev primerov uporabe

V tem poglavju podrobneje razčlenimo vsakega izmed identificiranih primerov uporabe iz prejšnjega poglavja. Vsak primer uporabe najprej opišemo s tekstom. Nato določimo akterje oz. udeležence primera uporabe. Na koncu tekst zapišemo v obliki glavnega t.j. najpogostejšega scenarija uporabe (npr. vnos uporabniškega imena in gesla) in manj pogostih, alternativnih primerov uporabe (npr. kaj se zgodi, če uporabnik vpiše napačno geslo).

3.3.1 Registracija

Da bi lahko začel uporabljati spletišče se mora obiskovalec najprej registrirati. Registracija pomeni, da obiskovalec določi svoje uporabniško ime (naslov svojega elektronskega poštnega predala, da se zagotovi unikatnost up. imena) in svoje geslo. Ko se obiskovalec registrira postane uporabnik sistema in se avtomatično prijavi v sistem. Registracija je prikazana v Tabeli 1.

Ime	Registracija
Udeleženci	Obiskovalec
Predpogoji	-
Scenarij	1. Obiskovalec obišče spletišče. 2. Spletišče ga ne prepozna in mu ponudi možnost registracije ali prijave. 3. Obiskovalec se odloči postati član spletišča. 4. Obiskovalec se registrira s pomočjo spletnega obrazca. 5. Obiskovalec si določi uporabnišo ime in geslo. 6. Obiskovalec je uspešno registriran.
Alternative	5.1. Uporabniško ime je že zasedeno. 5.2. Uporabnik mora vnesti novo uporabniško ime .

Tabela 1. Primer uporabe za registracijo

3.3.2 Prijava

V spodnji tabeli (Tabela 2) je prikazana prijava. Ko obiskovalec, ki se je v preteklosti že registriral obišče spletišče in vnese svoje uporabniško ime in geslo se s tem prijavi v sistem in postane uporabnik. Če je vnesel napačno uporabniško ime in geslo ga sistem vrne na izhodiščno pozicijo in od njega znova pričakuje uporabniško ime in geslo.

Ime	Prijava
Udeleženci	Obiskovalec
Predpogoji	Obiskovalec se je predhodno uspešno registriral na spletišču.
Scenarij	1. Obiskovalec vpiše uporabniško ime in geslo. 2. Obiskovalec vstopi v sistem in postane uporabnik.
Alternative	1.1. Napačno uporabniško ime ali geslo. 1.2. Obiskovalec ponovno vpiše uporabniško ime in geslo.

Tabela 2. Primer uporabe za prijavo

3.3.3 Pregled opravil

Uporabnik (t.j. obiskovalec, ki se je prijavil v sistem) pregleduje vsa svoja opravila na eni spletni strani. Pregled opravil je prikazan v spodnji tabeli (Tabela 3). Če uporabnik nima nobenih opravil, mu jih sistem ne prikaže. Za boljši pregled so opravila prikazana v tabelarični obliki. Za vsako opravilo se izpiše: opis opravila, naslov opravila, avtor opravila, rok opravila in če je bilo opravilo zaključeno. Če gre za skupinsko opravilo se pri pregledu izpiše še, komu je opravilo namenjeno.

Ime	Pregled opravil
Udeleženci	Uporabnik
Predpogoji	-
Scenarij	1. Sistem uporabniku prikaže vsa njegova opravila.
Alternative	1.1. Uporabnik nima nobenih opravil. 1.2. Sistem mu prikaže prazen zaslon.

Tabela 3. Primer uporabe za pregled opravil

3.3.4 Filtriranje pregleda opravil

Spodnja tabela (Tabela 4) prikazuje filtriranje pregleda opravil. Uporabnik lahko pregled opravil filtrira: a) po roku opravila, b) po imenu opravila ali c) po lastniku opravila: vsa opravila, samo uporabnikova opravila, samo opravila skupin, ki jim uporabnik pripada ali si jih lasti. Uporabnik lahko urejenost nastavi na padajoče ali naraščajoče.

Ime	Filtriranje pregleda opravil
Udeleženci	Uporabnik
Predpogoji	-
Scenarij	1. Uporabnik izbere pregledovanje opravil. 2. Sistem uporabniku prikaže vsa njegova opravila. 3. Uporabnik izbere način filtriranja in razvrščanja opravil. 4. Sistem uporabniku prikaže izbrana razvrščena opravila.
Alternative	-

Tabela 4. Primer uporabe za filtriranje pregleda opravil

3.3.5 Vnos novega opravila

Uporabnik lahko vnese novo opravilo. V spodnji tabeli (Tabela 5) je prikazan primer uporabe vnosa novega opravila. Opravilo mora poimenovati oz. podati krajši opis in določiti rok do katerega mora biti zaključeno. Rok opravila lahko vnese samo za prihodnost. Na voljo ima tudi polje za vnos podrobnejšega opisa opravila, ki pri vnosu ni obvezno. Opravilo se bo pojavilo v uporabnikovem pregledu opravil, kot osebno opravilo.

Ime	Vnos novega opravila
Udeleženci	Uporabnik
Predpogoji	-
Scenarij	1. Uporabnik izbere pregledovanje opravil. 2. Sistem uporabniku prikaže vsa njegova opravila. 3. Uporabnik izbere vnos novega opravila. 4. Sistem mu pokaže stran za vnos opravila. 5. Uporabnik določi ime, rok in podrobni opis opravila. 6. Uporabnik potrdi podatke. 7. Sistem vnese novo opravilo.
Alternative	5.1. Uporabnik je pozabil določiti ime ali rok. 5.2. Izpiše se obvestilo o napaki. 5.3. Uporabnik vpiše manjkajoče podatke. 6.1. Uporabnik ne potrdi podatkov. 6.2. Sistem prekine operacijo vnos opravila.

Tabela 5. Primer uporabe za vnos novega opravila

3.3.6 Urejanje opravila

Kot je prikazano v spodni tabeli (Tabela 6) lahko uporabnik posamezna opravila ureja. Za opravila, ki si jih je uporabnik zastavil sam, lahko ureja vse podrobnosti. Če pa gre za skupinsko opravilo, lahko ureja opravilo samo lastnik skupine ali član skupine, ki ima ustrezna dovoljenja. Med urejanjem opravila se lahko spremeni ime, rok, status in podrobnosti opravila. Kot pri vnosu opravila sta ime in rok obvezna, podrobnosti pa opcijske. Rok je lahko samo v prihodnosti (opravil za nazaj ni mogoče vnašati). V primeru napačnega vnosa podatkov, sistem uporabnika opozori kaj je napaka in ne dovoli zaključiti vnosa.

Ime	Urejanje opravila
Udeleženci	Uporabnik
Predpogoji	Uporabnik ima dovoljenje za urejanje opravila.
Scenarij	1. Uporabnik izbere urejanje opravila in opravilo. 2. Sistem mu pokaže stran za urejanje opravila. 3. Uporabnik popravi ime, rok in/ali podrobni opis opravila. 4. Uporabnik potrdi popravljene podatke. 5. Sistem spremeni podatke opravila.
Alternative	3.1. Uporabnik je izbrisal ime ali rok. 3.2. Izpiše se obvestilo o napaki. 3.3. Uporabnik vpiše manjkajoče podatke. 4.1. Uporabnik ne potrdi podatke. 4.2. Sistem prekine operacijo urejanja opravila.

Tabela 6. Primer uporabe za urejanje opravila

3.3.7 Pregled opravila

Uporabnik izbere opravilo in se mu prikažejo detajlni (vsi) podatki opravila. Prikaže se: opis opravila, naslov opravila, avtor opravila, rok opravila in če je bilo opravilo zaključeno. Pri skupinskih opravilih se izpišejo še člani skupine, ki so zadolženi zanj. Dostop do pregleda skupinskih opravil je dovoljen samo lastniku skupine, članom z ustreznimi dovoljenji in zadolženim članom. Primer uporabe za pregled opravila je prikazan na spodnji tabeli (Tabela 7).

Ime	Pregled opravila
Udeleženci	Uporabnik
Predpogoji	Uporabnik ima dovoljenje za pregled opravila.
Scenarij	1. Uporabnik izbere opravilo. 2. Sistem uporabniku prikaže stran opravila.
Alternative	-

Tabela 7. Primer uporabe za pregled opravila

3.3.8 Brisanje opravila

V primeru, da uporabnik pozameznega opravila ne potrebuje več ima na voljo brisanje opravila. Če gre za osebno opravilo, lahko uporabnik opravila briše poljubno. Pri skupinskih opravilih pa lahko opravila brišejo samo lastnik skupine in člani z ustreznimi dovoljenji. Preden sistem dokončno izbriše opravilo od uporabnika zahteva, da svojo odločitev še enkrat potrdi. Spodaj (Tabela 8) je prikazan primer uporabe za brisanje opravila.

Ime	Brisanje opravila
Udeleženci	Uporabnik
Predpogoji	Uporabnik ima dovoljenje za brisanje opravila.
Scenarij	1. Uporabnik izbere brisanje opravila in opravilo. 2. Sistem mu predlaga, da ponovno potrdi svojo odločitev. 3. Uporabnik potrdi svojo odločitev. 4. Sistem izbriše opravilo.
Alternative	3.1. Uporabnik ne potrdi svoje odločitve. 3.2. Sistem prekine operacijo brisanja.

Tabela 8. Primer uporabe za brisanje opravila

3.3.9 Sprememba statusa opravila

Spreminjanje statusa opravila je prikazano v spodnji tabeli (Tabela 9). Vsako izmed opravil je v enem izmed dveh stanj. Lahko je zaključeno ali pa aktivno. Uporabnik lahko stanje oz. status svojih osebnih opravil poljubno spreminja. Pri skupinskih opravilih lahko status spreminja lastnik skupine, člani z ustreznimi dovoljenji in člani zadolženi zanj. Status se lahko spreminja preprosto s klikom na opravilo v pregledu opravil ali pa med urejanjem opravila.

Ime	Sprememba statusa opravila
Udeleženci	Uporabnik
Predpogoji	Uporabnik ima dovoljenje za spreminjanje statusa opravila.
Scenarij	1. Uporabnik izbere pregledovanje opravil. 2. Uporabnik izbere opravilo in spremeni status. 3. Uporabnik potrdi svojo odločitev. 4. Sistem spremeni status opravila.
Alternative	3.1. Uporabnik ne potrdi svoje odločitve. 3.2. Sistem prekine operacijo spremembe statusa opravila.

Tabela 9. Primer uporabe za spremembo statusa opravila

3.3.10 Ustvarjanje nove skupine

Vsak uporabnik lahko ustvari eno ali več skupin. Pri ustvarjanju skupine uporabnik določi ime skupine in potrdi svojo odločitev. V primeru praznega vnosa imena skupine, sistem uporabnika opozori kaj je napaka in ne dovoli zaključiti vnosa. Po uspešnem vnosu se ustvari nova skupina in uporabnik postane lastnik skupine. To je prikazano tudi v naslednji tabeli (Tabela 10).

Ime	Ustvarjanje nove skupine
Udeleženci	Uporabnik
Predpogoji	-
Scenarij	1. Uporabnik izbere pregledovanje opravil. 2. Sistem uporabniku prikaže vsa njegova opravila. 3. Uporabnik izbere ustvaritev nove skupine. 4. Sistem mu pokaže stran za ustvarjanje nove skupine. 5. Uporabnik določi ime skupine. 6. Uporabnik potrdi podatke. 7. Sistem ustvari novo skupino.
Alternative	5.1. Uporabnik je pozabil določiti ime skupine. 5.2. Izpiše se obvestilo o napaki. 5.3. Uporabnik vpiše manjkajoče podatke. 6.1. Uporabnik ne potrdi podatke. 6.2. Sistem prekine operacijo ustvarjanja nove skupine.

Tabela 10. Primer uporabe za ustvarjanje nove skupine

3.3.11 Pošiljanje vabil

Kot je prikazano v Tabeli 11 lahko lastnik skupine v svojo skupino povabi katerega koli uporabnika spletišča. To naredi z vnosom naslova elektronskega poštnega predala uporabnika, ki ga želi povabiti v skupino. Ta mora biti identičen naslovu, ki ga je uporabnik vnesel med registracijo. Če lastnik skupine vnese napačen naslov se mu izpiše napaka in se pošiljanje vabila prekine. Po vnosu, sistem uporabniku pošlje vabilo za pridružitev skupini.

Ime	Pošiljanje vabil
Udeleženci	Lastnik skupine, uporabnik
Predpogoji	-
Scenarij	1. Lastnik skupine se odloči povabiti uporabnika v skupino. 2. Lastnik skupine vpiše uporabnikov elektronski naslov. 3. Sistem pošlje uporabniku vabilo za pridružitev skupini.
Alternative	2.1. Lastnik skupine vpiše napačen elektronski naslov. 2.2. Izpiše se sporočilo o napaki in mora ponoviti postopek.

Tabela 11. Primer uporabe za pošiljanje vabil

3.3.12 Pregledovanje podatkov o skupini

Vsi člani skupine, vključno z lastnikom skupine, imajo na voljo pregledovanje podatov o skupini. Tukaj so prikazani: ime skupine, imena članov skupine ter vsa aktivna in zaključena opravila skupine. Lastnik skupine in člani z ustreznimi dovoljenji imajo na voljo tudi gumb za vnos novega skupinskega opravila. Poleg tega ima lastnik skupine tudi na voljo gumb za urejanje podatkov o skupini. Spodnja tabela (Tabela 12) prikazuje primer uporabe za pregledovanje podatkov o skupini.

Ime	Pregledovanje podatkov o skupini
Udeleženci	Uporabnik
Predpogoji	Uporabnik je lastnik ali član skupine.
Scenarij	1. Uporabnik izbere skupino. 2. Sistem uporabniku prikaže ime skupine, imena uporabnikov v skupini in vsa opravila skupine.
Alternative	-

Tabela 12. Primer uporabe za pregledovanje podatkov o skupini

3.3.13 Urejanje podatkov o skupini

Lastnik skupine lahko ureja podatke o svoji skupini. Urejanje podatkov o skupini je prikazano v Tabeli 13. Na voljo ima opcijo spremembe imena skupine in odstranitev izbranih uporabnikov iz skupine. Če je polje za spremembo imena skupine prazno, se lastniku skupine izpiše napaka in se urejanje podatkov prekine.

Ime	Urejanje podatkov o skupini
Udeleženci	Lastnik skupine
Predpogoji	-
Scenarij	1. Lastnik skupine izbere urejanje skupine in skupino. 2. Sistem mu pokaže stran za urejanje skupine. 3. Lastnik skupine popravi ime skupine in/ali odstrani uporabnike iz skupine. 4. Lastnik skupine potrdi popravljene podatke. 5. Sistem spremeni podatke skupine.
Alternative	3.1. Lastnik skupine je izbrisal ime skupine. 3.2. Izpiše se obvestilo o napaki. 3.3. Lastnik skupine vpiše manjkajoče podatke. 4.1. Lastnik skupine ne potrdi podatke. 4.2. Sistem prekine operacijo urejanja skupine.

Tabela 13. Primer uporabe za urejanje podatkov o skupini

3.3.14 Brisanje skupine

Če se zaradi kakršnega koli razloga lastnik skupine odloči, da ne potrebuje ali ne želi več skupine, ima na voljo brisanje skupine. Na strani za urejanje podatkov o skupini lahko lastnik izbere tudi brisanje skupine. Po izbiri brisanja skupine sistem od lastnika skupine zahteva, da ponovno potrdi svojo odločitev. Po potrditvi sistem izbriše skupino in vsa opravila, ki so bila ustvarjena v sklopu te skupine. Primer uporabe brisanja skupine je prikazan v naslednji tabeli (Tabela 14).

Ime	Brisanje skupine
Udeleženci	Lastnik skupine
Predpogoji	-
Scenarij	1. Lastnik skupine izbere brisanje skupine in skupino. 2. Sistem mu predlaga, da ponovno potrdi svojo odločitev. 3. Lastnik skupine potrdi svojo odločitev. 4. Sistem izbriše skupino.
Alternative	3.1. Lastnik skupine ne potrdi svoje odločitve. 3.2. Sistem prekine operacijo brisanja.

Tabela 14. Primer uporabe za brisanje skupine

3.3.15 Vnos novega skupinskega opravila

Lastnik skupine ali član z ustreznimi dovoljenji lahko vnese novo skupinsko opravilo. V spodnji tabeli (Tabela 15) je prikazan primer uporabe vnosa novega skupinskega opravila. Kot pri osebnih opravilih se mora skupinsko opravilo obvezno poimenovati in določiti rok. Rok lahko vnese samo za prihodnost. Ima tudi opcijo vnosa podrobnejšega opisa opravila (neobvezno). Posebnost pri skupinskih opravilih pa je, da lahko vnašalec določi kateri člani skupine so zadolženi za posamezna opravila. To opravilo se bo pojavilo v njihovem pregledu opravil.

Ime	Vnos novega skupinskega opravila
Udeleženci	Lastnik, uporabnik
Predpogoji	Uporabnik mora biti član skupine in imeti dovoljenje za vnos novega opravila.
Scenarij	1. Lastnik ali uporabnik izbere pregled skupine. 2. Sistem lastniku ali uporabniku prikaže stran skupine. 3. Lastnik ali uporabnik izbere vnos novega opravila. 4. Sistem mu pokaže stran za vnos opravila. 5. Lastnik ali uporabnik določi ime, rok, podrobni opis opravil in izbere ljudi za katere je opravilo namenjeno. 6. Lastnik ali uporabnik potrdi podatke. 7. Sistem vnese novo opravilo.
Alternative	5.1. Lastnik ali uporabnik je pozabil določiti ime ali rok. 5.2. Izpiše se obvestilo o napaki. 5.3. Lastnik ali uporabnik vpiše manjkajoče podatke. 7.1. Lastnik ali uporabnik ne potrdi podatke. 7.2. Sistem prekine operacijo vnos opravila.

Tabela 15. Primer uporabe za vnos novega skupinskega opravila

3.3.16 Pošiljanje obvestil uporabnikom

Sistem uporabnika opozori na dogodke. Dogodek je povabilo v skupino ali približevanje roka opravila. Ko se dogodek sproži, sistem uporabniku pošlje obvestilo. Ta se mu prikaže v zgornjem meniju, kot števec vseh obvestil. Sam števec pa je povezava na stran, kjer so obvestila podrobneje opisana. V primeru, da gre za obvestilo o vabilu za pridružitev skupini, lahko uporabnik povabilo sprejme ali pa zavrne. V obeh primerih se obvestilo izbriše. Če gre za sprejem vabila se uporabnik tudi pridruži skupini. Naslednje obvestilo, ki ga uporabnik lahko prejme gre za približevanje roka kakšnega aktivnega opravila. Če se datum približuje roku kakšnega opravila za katerega je zadolžen uporabnik, mu sistem pošlje obvestilo. Za to obvestilo ima uporabnik na voljo samo potrditev, da je obvestilo videl. Sledeč potrdilu se obvestilo izbriše in se uporabnika za to opravilo v prihodnosti ne obvešča. V Tabeli 16 je prikazan primer uporabe pošiljanja obvestil uporabnikom.

Ime	Pošiljanje obvestil uporabnikom
Udeleženci	Uporabnik
Predpogoji	Uporabnik je dobil povabilo skupine ali se približuje rok kakšnega opravila.
Scenarij	1. Sistem zazna povabilo ali približanje roka opravila. 2. Sistem doda novo obvestilo uporabniku. 3. Uporabniku se prikaže novo obvestilo.
Alternative	-

Tabela 16. Primer uporabe za pošiljanje obvestil uporabnikom

Poglavje 4 Uporabniška izkušnja

V tem poglavju je opisan primer načrtovanja izgleda strani spletišča in orodja, ki se za to uporabljajo.

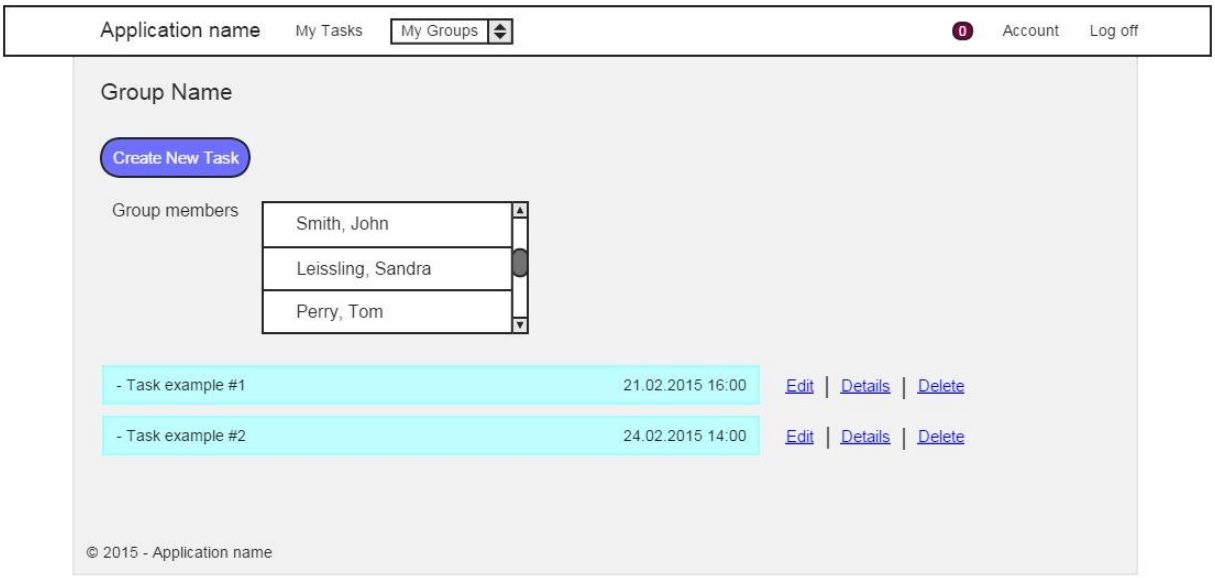
4.1 Mock-up grafika

Mock-up grafika je poenostavljena predstavitev izgleda končnega izdelka [14]. Uporablja se za predstavitev izgleda sistema oz. spletišča. Predstavitev se naredi tako, da se za vsako stran spletišča izriše mock-up grafika.

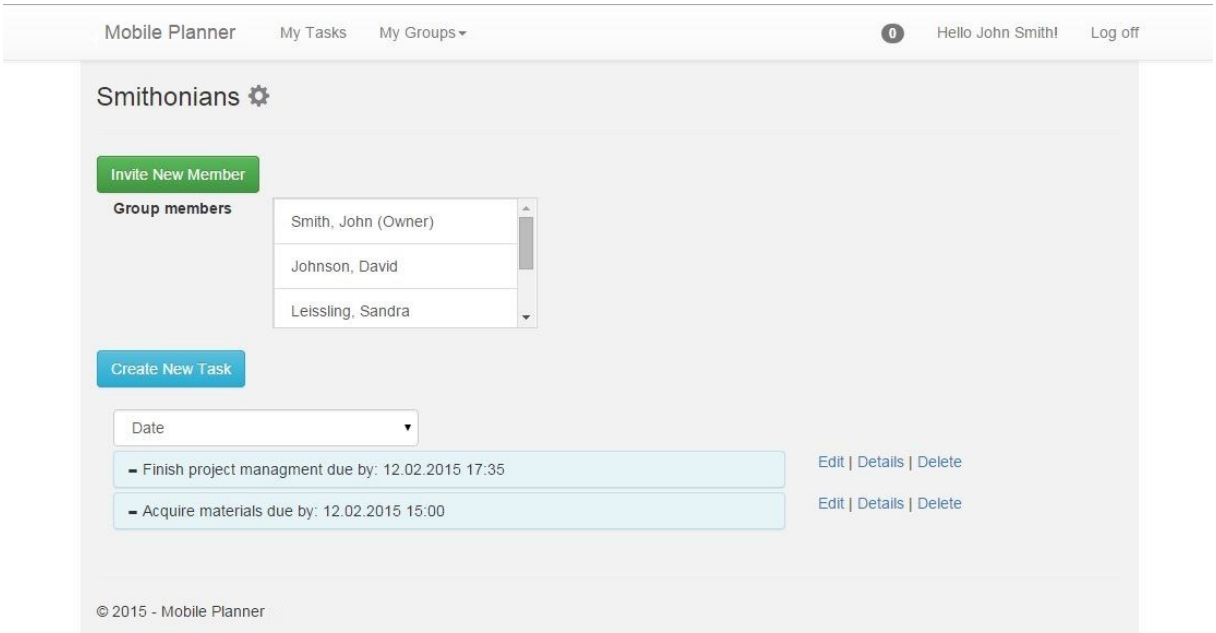
Za izris mock-up grafike se sicer lahko uporabi kakršnokoli orodje za oblikovanje ali risanje. Ker pa je to zelo pogost problem so danes na voljo številna specializirana orodja, posebej za izdelavo mock-up grafike spletnih strani. Ta orodja že vsebujejo osnovne gradnike, ki se pogosto uporabljajo pri spletnih straneh (npr. spustni meni, polje za vnašanje teksta). Vsaka spletna stran je platno. Gradniki pa se potem dodajajo na platno na način povleci in spusti (angl. Drag and drop) ter nadalje oblikujejo po želji.

Poleg orodij za izdelavo mock-up grafike, ki jih lahko namestimo na računalnik obstajajo tudi orodja na spletu oz. v oblaku, kjer se mock-up grafiko oblikuje kar v brskalniku.

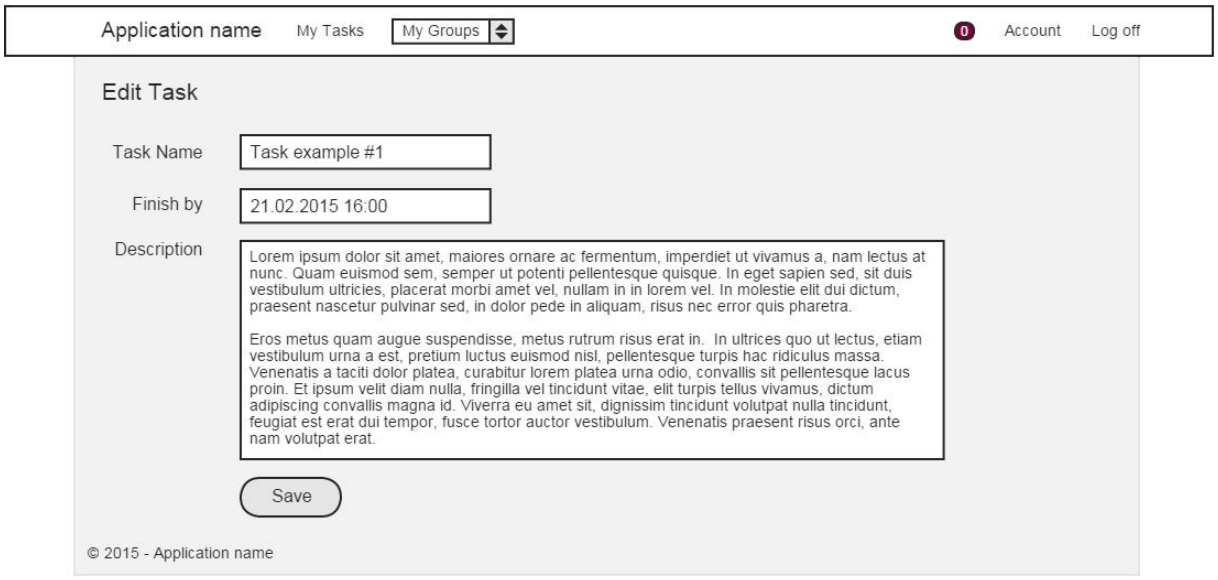
Naslednje slike (Slike 3-6) prikazujejo dva primera mock-up grafike in končnega izgleda spletne strani.



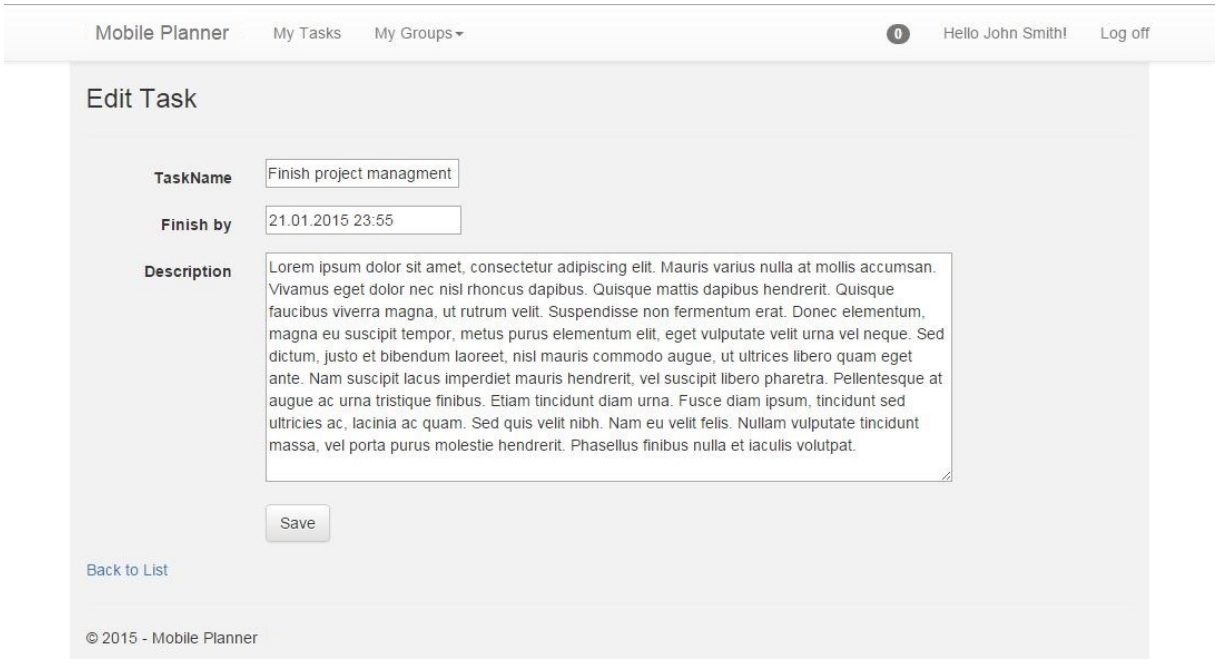
Slika 3. Primer mock-up grafike za pregled skupine



Slika 4. Končni izgled strani za pregled skupine



Slika 5. Primer mock-up grafike za urejanje opravila



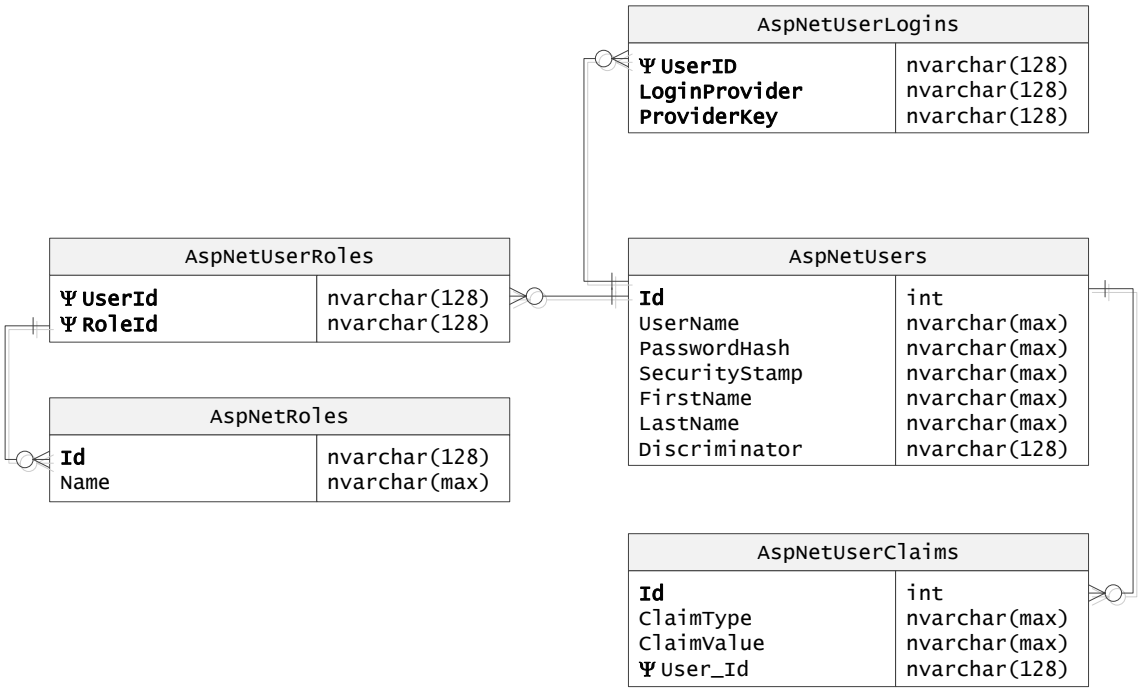
Slika 6. Končni izgled strani za urejanje opravila

Poglavje 5 Podatkovna baza

Skladno s filozofijo »slika pove tisoč besed« so v tem poglavju opisani podatkovni modeli z diagrami in s kratkim tekstom za vsako tabelo. V podatkovnem modelu so opisane vse tabele, njihova polja in tipi. Primarni ključi so odebeljeni, pred sekundarnimi ključi pa stoji simbol Ψ.

5.1.1 Identifikacija uporabnika

Za identifikacijo uporabnika orodje ASP.NET Identity avtomatično ustvari naslednje tabele.



Slika 7. Tabele, ki jih je avtomatično ustvaril ASP.NET Identity

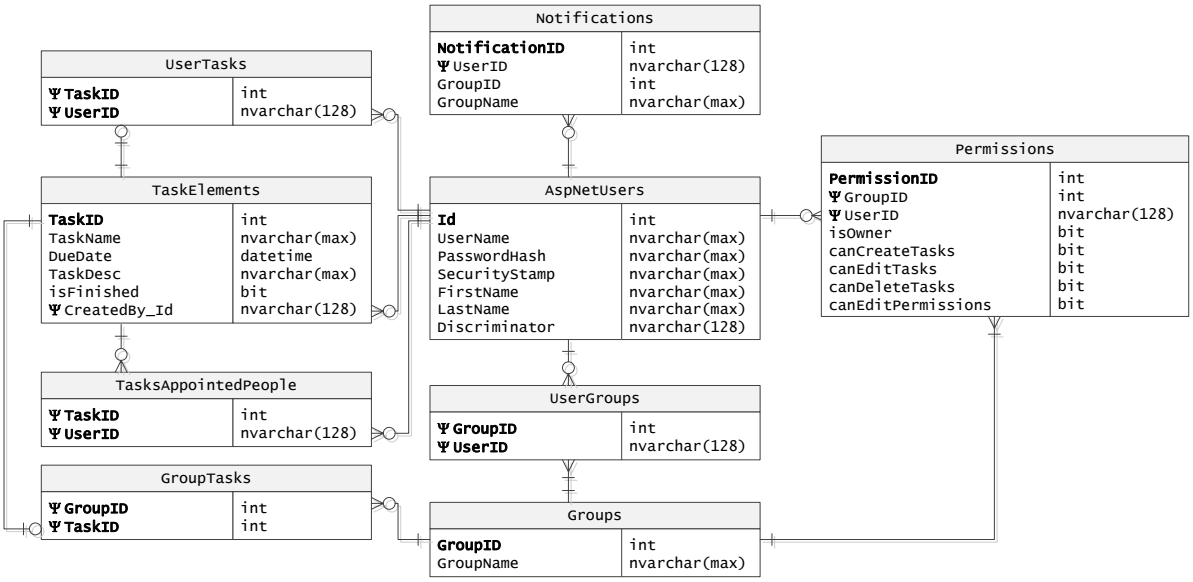
Iz zgornjega diagrama (Slika 7) je razvidno, da je tabela **AspNetUsers** glavna oz. centralna tabela. Zato ima ta tabela nadpovprečno veliko povezav. V drugih diagramih se velikokrat uporablja kot središče diagrama oz. eno izmed središč diagrama zaradi boljše razvidnosti. Takemu diagramu se ponavadi reče *zvezdni* ali *snežni* diagram. Tabela *AspNetUsers* vsebuje podatke o uporabnikih.

AspNetRoles se uporablja za shranjevanje dovoljenj uporabnikov v obliki vlog (npr. vloga »Administrator«) [15]. Za povezavo med uporabniki in vlogami pa se uporablja vmesna tabela **AspNetUserRoles**.

Tabela **AspNetUserClaims** vsebuje sklop izrazov, ki bolj podrobno predstavljajo identiteto uporabnika [15]. Uporablja se predvsem ko z vlogami ne moremo dovolj natančno opisati dovoljenj uporabnika. V **AspNetUserLogins** pa so shranjeni podatki o zunanjem ponudniku za avtentikacijo (npr. Facebook), ko se uporablja za prijavo.

5.1.2 Opravila in skupine

Spodnji diagram (Slika 8) prikazuje tabele, povezane z opravili in skupinami ter njihove relacije.



Slika 8. Podatkovni model za opravila in skupine

5.1.2.1 Centralne tabele

Tri centralne tabele modela, na katere se vse preostale sklicujejo so:

- AspNetUsers,
- TaskElements in
- Groups.

Tabela **AspNetUsers** je bila že opisana v predhodnih poglavjih. Vsebuje podatke o uporabnikih. Tabela **TaskElements** hrani vsa opravila. Tabela **Groups** pa vsebuje podatke o skupinah.

5.1.2.2 Povezovalne tabele

Poleg centralnih tabel so v modelu še povezovalne tabele. Te tabele praviloma vsebujejo dva ključa, ki vzpostavljata povezavo med dvema centralnima tabelama. Z imenom je nakazano, za kakšne vrste povezavo gre. Tabela **UserGroups** povezuje tabeli **AspNetUser** in **Groups**.

UserTasks vsebuje povezave med uporabniki in opravili, ki jih je uporabnik ustvaril sam zase, **GroupTasks** pa vsebuje povezave med opravili in skupinami pod katere ta opravila spadajo.

V tabeli **TasksAppointedPeople** so, podobno kot pri tabeli *UserTasks*, shranjene povezave med opravili in uporabniki. Tabeli se razlikujeta po tem, da so v tej tabeli shranjena izključno samo skupinska opravila. Opravila so nadalje posredno povezana uporabniki s tem, da so ti člani skupine, ki jim je bilo to opravilo dodeljeno.

Tabela **Permissions** vsebuje povezave med uporabniki in skupinami ter vsa dovoljenja, ki jih uporabnik ima znotraj te skupine.

Tabela **Notifications** hrani obvestila za posameznega uporabnika in je neposredno povezana z tabelo uporabnikov.

Poglavje 6 Arhitektura

V tem poglavju so opisani arhitekturni prijemi oz. vzorci, ki so uporabljeni na večih mestih v programu.

Najpomembnejša med njimi sta:

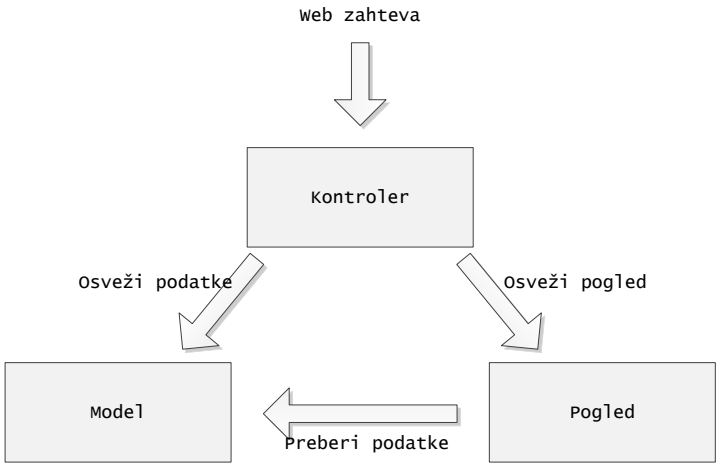
- Model-view-controller (*v nadaljevanju: MVC*), ki je danes najpopularnejši vzorec za razvoj spletnih aplikacij in
- vzorci mapiranja objektov v programskem jeziku (C#) v tabele na SQL strežniku, ki jih uporablja ogrodje Entity Framework.

6.1 Model-view-controller (MVC)

MVC je arhitekturni vzorec, ki se ga uporablja (tudi) pri izdelovanju spletnih aplikacij [20]. Z uporabo tega vzorca se razvoj spletnih aplikacij poenostavi, arhitektura pa postane bolj čista. Delovanje vzorca je prikazano v spodnji sliki (Slika 9).

Osnovna zamisel vzorca MVC je ločiti funkcionalnosti spletne strani na tri sklope [16]:

- Model (model),
- View (pogled) in
- Controller (kontroler).



Slika 9. Predstavitev delovanja Model-view-controller vzorca

- **Model** je tipično razred iz katerega so izpeljani vsi drugi podatkovni razredi. V ASP.NET MVC pa je model abstrakcija razreda. Pojem podatkovni razred pomeni, da model predvsem hrani podatke in ponavadi nima druge funkcionalnosti (npr. podatkovne logike).

  Del ASP.NET MVC je knjižnica DataAnnotations. Z njo lahko nastavimo pravila za posamezna polja (npr. omejitev vnosa za 10 znakov ali omejitev vsebine vnosnega polja na vsebino, ki se ujema z regularnim izrazom) in pa spremenimo ime polja kako bo prikazano v View.

- **Pogled** je v ASP.NET MVC izveden kot HTML šablona, ki se uporablja za prikaz podatkov in polj kamor uporabnik lahko vnese podatke. Iz prejšnje točke vemo, da so podatki shranjeni v razredih Model. Za čisto arhitekturno razmejitev mora biti razred view omejen zgolj na prikaz in vnos podatkov, ne pa tudi na njihovo shranjevanje ali obravnavanje. Za to skrbi razred Controller.

  Microsoft Visual Studio okolje za view omogoča uporabo obogatenega HTML. To tehnologijo imenujejo Razor Engine.

- **Kontroler** je usmerjevalec vsega dogajanja znotraj vzorca MVC. Povezuje se s podatkovno bazo in shranjuje ali vrača podatke (kot razrede Model) ter usmerja proces za katerega je zadolžen.

6.1.1 Model

Spodnja slika (Slika 10) prikazuje enega od modelov programa.

Z rdečo puščico je označen del, ki se navezuje na predhodno omenjeno knjižnico *DataAnnotations*. Ta knjižnica omogoča podajanje dodatnih opisnih navodil kako naj sistem ravna z določenim poljem (v danem primeru za DueDate).

V razredu so razvidna tudi polja tipa »virtual«. Ta polja so reference na različne objekte oz. kolekcije objektov, ki so del programa. S pomočjo ogrodja Entity Framework se nato podatki shranjujejo v bazo na tak način, da se z objektom shranijo tudi povezani objekti.

```
namespace WorkPlanner.Models
{
    public class TaskElement
    {
        [Key]
        public int TaskID { get; set; }

        [Required]
        public string TaskName { get; set; }

        [Required]
        [Display(Name = "Due by")]
        [DataType(DataType.DateTime)]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:dd/MM/yyyy HH:mm}")]
        public DateTime DueDate { get; set; }
        public virtual ICollection<ApplicationUser> AppointedPeople { get; set; }
        public virtual ApplicationUser CreatedBy { get; set; }
        public string TaskDesc { get; set; }
        public Boolean isFinished { get; set; }
        public virtual ICollection<ApplicationUser> AppUsers { get; set; }
        public virtual ICollection<Group> Groups { get; set; }
        public TaskElement()
        {
            AppUsers = new HashSet<ApplicationUser>();
            Groups = new HashSet<Group>();
            AppointedPeople = new HashSet<ApplicationUser>();
        }
    }


}
```

Slika 10. Model

6.1.2 Pogled

Primer pogleda je prikazan v spodnji sliki (Slika 11). Rdeča puščica prikazuje primer sintakse pogona Razor. To je pogon, ki progamerju olajša delo.

Namesto pisanja manj pregledne kode HTML lahko programer piše v psevdojeziku in se sklicuje neposredno na razrede. Razor pa nato ustvari HTML za te razrede. Za sintakso psevdo jezika pogon uporablja kar C#.

```
@model IEnumerable<WorkPlanner.Models.TaskElement>

@{
    ViewBag.Title = "Tasks";
}

<h2>Tasks</h2>

<p>
    @Html.ActionLink("Create New Task", "Create")
    <br />
    <a href="@Url.Action("Create", "Groups")">Create New Group</a>
</p>
<div class="container">
    @foreach (var item in Model)
    {
        <div class="row">
            <div class="col-md-8">
                <button type="button" class="btn btn-default btn-group-justified btn-task">
                    <div class="text-left">
                        <span class="glyphicon glyphicon-minus"></span>
                        @Html.DisplayFor(modelItem => item.TaskName)
                        @Html.DisplayName("due by:")
                        @Html.DisplayFor(modelItem => item.DueDate)
                    </div>
                </button>
            </div>
            <div class="col-md-4">
                @Html.ActionLink("Edit", "Edit", new { id = item.TaskID }) |
                @Html.ActionLink("Details", "Details", new { id = item.TaskID }) |
                @Html.ActionLink("Delete", "Delete", new { id = item.TaskID })
            </div>
        </div>


    }
</div>
```

Slika 11. Pogled

6.1.3 Kontroler

Spodnja slika (Slika 12) prikazuje primer kontrolerja. Spletna zahteva GET povzroči avtomatičen klic metode Index. Ta pa glede na trenutnega uporabnika vrne pogledu seznam njegovih opravil kot model. Kontroler torej v celoti orkestrira pogled in model.

```
public class TasksController : Controller
 {
        private TasksDBContext db = new TasksDBContext();

        // GET: /Tasks/
        public ActionResult Index()
        {
            ApplicationUser currentUser = GetCurrentUser();

            List<TaskElement> tasks = currentUser.TaskElements.ToList();
            tasks.AddRange(currentUser.GroupTaskElements.ToList());

            return View(tasks);
        }
 }
```

Slika 12. Kontroler

Poglavje 7 Sklepne ugotovitve

Načrtovanje programskih rešitev je več kot samo programiranje. Skozi čas so se oblikovale najboljše prakse kako se lotiti vsakega koraka od analize do kodiranja. V diplomski nalogi nisem opisal postopka testiranja, ker bi to bistveno povečalo njen obseg. Poskusil sem prikazati prakse ki sem se jih naučil iz pogovorov z bolj izkušenimi razvijalci.

Z uporabljenim pristopom sem zadovoljen in je dobro deloval zame. Poleg učinkovitega dela omogoča tudi večjo predvidljivost in s tem učinkovitejše načrtovanje. Če so koraki razvoja spletne aplikacije znani, je tudi lažje približno določiti časovni okvir za vsak korak. Seveda pa to ni edini pristop do izdelave spletišča ampak samo eden izmed mnogih.

Cilj diplomske naloge je bil spoznati in predstaviti postopek izdelave spletišča. Želel sem tudi pridobiti izkušnje in projekt, ki mi bo v prihodnosti pomagal predstaviti moje znanje pri iskanju dela.

Pri izdelavi obsežnejših spletišč ponavadi sodeluje več ljudi. Delo se razdeli med člane skupine, ki so strokovnjaki na svojem področju (npr. oblikovanje, načrtovanje pod. baze). To je tudi eden od razlogov, da sem se odločil za enostavnejši primer spletišča. Kljub temu sem z izdelavo spletišča dobil celovit pregled nad takšnim projektom.

Poglavje 8 Literatura

8.1 Seznam virov

[1] *C Sharp (programming language)*. Dostopno na: http://en.wikipedia.org/wiki/C_Sharp_%28programming_language%29.

[2] *HTML*. Dostopno na: http://en.wikipedia.org/wiki/HTML.

[3] *Cascading Style Sheets*. Dostopno na: http://en.wikipedia.org/wiki/Cascading_Style_Sheets.

[4] *JavaScript*. Dostopno na: http://en.wikipedia.org/wiki/JavaScript.

[5] *jQuery*. Dostopno na: http://en.wikipedia.org/wiki/JQuery.

[6] *The HTML DOM Element Object*. Dostopno na: http://www.w3schools.com/jsref/dom_obj_all.asp.

[7] *What is SQL?*. Dostopno na: http://www.sqlcourse.com/intro.html.

[8] *Database*. Dostopno na: http://en.wikipedia.org/wiki/Database.

[9] *What is Entity Framework?*. Dostopno na: http://www.entityframeworktutorial.net/what-is-entityframework.aspx.

[10] *Bootstrap (front-end framework)*. Dostopno na: http://en.wikipedia.org/wiki/Bootstrap_(front-end_framework).

[11] *Model–view–controller*. Dostopno na: http://en.wikipedia.org/wiki/Model-view-controller.

[12] *Unified Modeling Language*. Dostopno na: http://en.wikipedia.org/wiki/Unified_Modeling_Language.

[13] I. Savnik. *Pregled UML*. Dostopno na: http://osebje.famnit.upr.si/~savnik/predmeti/NPB/npb3-uml-intro.pdf.

[14] *Mockup*. Dostopno na: http://en.wikipedia.org/wiki/Mockup.

[15] *Overview of Custom Storage Providers for ASP.NET Identity*. Dostopno na: http://www.asp.net/identity/overview/extensibility/overview-of-custom-storage-providers-for-aspnet-identity#architecture.

[16] *Model View Controller*. Dostopno na: http://c2.com/cgi/wiki?ModelViewController.

[17] M. Fowler. (2004). *UML Distilled: A Brief Guide to the Standard Object Modeling Language*.

[18] J.D. Gauchat. (2012). *HTML5 for Masterminds, 2nd Edition*.

[19] J. Duckett. (2014). *JavaScript & JQuery: Interactive Front-end Web Development*.

[20] J. Galloway. (2014). *Professional ASP.NET MVC 5*.