Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Andrej Kokelj

Spletni sistem za upravljanje z viri v pralnici

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Aleš Smrdel

Ljubljana, 2018

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomskem delu razvijte spletni sistem, ki bo služil kot podpora pri rezervaciji terminov in upravljanju s pralnimi stroji v ustanovah, ki so namenjene daljšemu bivanju. Razvita rešitev naj implementira več različnih vlog. Tako naj zagotavlja vlogo navadnega uporabnika, ki omogoča izvajanje rezervacij, vlogo receptorja, ki omogoča nadzor in upravljanje s ključi pralnih sob, tajništvo, ki omogoča izvaja obračuna uporabe pralnih strojev za neko časovno obdobje, in administratorja, ki omogoča nadzor nad pralnimi sobami, pralnimi stroji in uporabniki. Pri realizaciji aplikacije izberite tiste tehnologije na strani strežnika in na strani odjemalca, ki bodo omogočale hitro in odzivno delovanje aplikacije. Za zagotavljanje dobre uporabniške izkušnje pa pred objavo aplikacijo tudi testirajte.

Zahvaljujem se vsem sošolcem, prijateljem, družini in mentorju za vso pomoč pri študiju in izdelavi diplomskega dela.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
HTML	Hyper Text Markup Language	Jezik za označevanje besedila
CSS	Cascading Style Sheets	Kaskadne stilske podloge
SQL	Structured Query Language	Strukturirani povpraševalni jezik za delo s podatkovnimi bazami
JSON	JavaScript Object Notation	JavaScript objekt za izmenjavo podatkov
ID	Identification	Identifikacija
REST	Representational State Transfer	Predstavitveni prenos stanja
CRUD	Create, read, update, and delete	Ustvarjanje, branje, posodobitev in brisanje
URL	Uniform Resource Locator	Enolični krajevnik vira

Povzetek

Naslov: Spletni sistem za upravljanje z viri v pralnici

Avtor: Andrej Kokelj

V dijaških in študentskih domovih pa tudi v kakšnih drugih namestitvah, ki omogočajo daljše bivanje, se pogosto nahajajo pralni stroji, katere stanovalci lahko uporabljajo za pranje perila. Ker pa pralnih strojev ni dovolj, morajo stanovalci za uporabo pralnih strojev rezervirati termin. Za rezervacijo terminov pa se še vedno pogosto ročno vodi evidenco. Cilj diplomske naloge je bil razvoj rešitve, ki bi uporabnikom pralnih strojev omogočala rezervacijo termina preko spleta - spletne aplikacije. Vodenje in vpogled v evidenco rezervacij v razviti rešitvi ima odgovorna oseba doma. Uporabniki pralnice se v spletno aplikacijo lahko registrirajo in s tem pridobijo uporabniško ime in geslo, s katerim se prijavijo v aplikacijo. Spletna aplikacija pa izvaja klice na spletno storitev, ki skrbi za manipulacijo s podatki, te pa se nahajajo v podatkovni bazi. Ta rešitev omogoča lažje vodenje evidence in hitrejšo ter bolj učinkovito rezervacijo terminov ter obračun stroškov. Spletna aplikacija je trenutno postavljena v testno okolje, po končanem testnem obdobju in odpravi zaznanih pomanjkljivosti pa bo postavljena v produkcijsko okolje.

Ključne besede: spletna aplikacija, rezervacija sredstev, avtomatska poročila.

Abstract

Title: Web system for managing resources in laundry

Author: Andrej Kokelj

Student dorms and similar accommodations where people stay for longer periods of time often have laundry machines, which inhabitants can use for washing clothes. Since there is usually insufficient number of washing machines, the inhabitants have to reserve a term to use them. These reservations are often still managed manually, on paper. The aim of this thesis was to develop a solution that would enable washing machine users to make reservations online via a web app. The person responsible for managing these reservations would be an authorized person of the dorm. The users of the laundry are able to register into the web app, where they receive a username and a password, which they can later use to login into the web app. This web app then makes calls to the web service which is responsible for manipulating the data, which are stored the database. This solution enables a faster and more efficient way to make reservations for the laundry and for calculating costs. The web app is currently in a testing environment , and after this testing period and the elimination of detected deficiencies the web app will be put into a production environment.

Keywords: Web application, Resource planing, Automated reports.

Poglavje 1

Uvod

V dijaških in študentskih domovih pa tudi v kakšnih drugih stanovanjskih namestivah, ki omogočajo daljše bivanje, se nahajajo pralni stroji, katere stanovalci redno uporabljajo za pranje perila. Ker pa je potencialnih uporabnikov ponavadi več kot pa razpoložljivih pralnih strojev, je potrebno opraviti rezervacijo terminov za pranje. Za rezervacijo terminov za pranje pa se še vedno pogosto ročno vodi evidenco. Ker velikokrat ni ena sama oseba zadolžena za vodenje evidence, se lahko izgublja veliko časa pri iskanju trenutne odgovorne osebe, ki vodi evidenco terminov, zato je zaželena elektronska evidenca terminov in njihovo upravljanje. Zaradi pomanjkanja prilagojenih informacijskih rešitev na tem področju, je cilj diplomske naloge implementacija spletne aplikacije za pralnico, preko katere bo uporabnik rezerviral določen termin za pranje.

Aplikacija je razdeljena na štiri dele. Prvi del aplikacije predstavlja uporabniški del preko katerega se uporabnik prijavi v aplikacijo z uporabniškim imenom in geslom ter prek katerega lahko nato rezervira rezerviral prosti termin. Drugi del aplikacije predstavlja receptorski del, v katerem mora receptor ob predaji ključa za pralnico označiti, da je oddal ključ in ob vračilu ključa tudi označiti, da ga je uporabnik vrnil. Tretji del predstavlja tajniški del, ki vsebuje evidenco števila pranj bodisi po posameznih mesecih bodisi po uporabnikih. Zadnji del predstavlja administratorski del, v katerem pa je

implementiran nadzor nad uporabniki, se pravi dodeljevanje in odvzemanje dostopa do spletne aplikacije uporabnikom, konfiguriranje aplikacije ter dodajanje, brisanje in urejanje pralnih sob in števila pralnih strojev v pralnih sobah.

Spletne aplikacije za rezervacijo določenih stvari oziroma za upravljanje z viri že obstajajo. Glede na zahteve, ki jih imamo glede evidence terminov je podobna spletna aplikacija, kot smo jo izdelali v diplomski nalogi, aplikacija Eventbrite, dostopna tudi na naslovu www.eventbrite.com. Eventbrite je spletna aplikacija, ki omogoča uporabnikom kreiranje, iskanje in promocijo lokalnih dogodkov. To je sicer aplikacija, ki se prvenstveno uporablja za prodajo vstopnica za razne dogodke. Aplikacija sama pa je v svoji osnovi zasnovana tako, da omogoča upravljanje z viri in ni omejena samo na prodajo kart, čeprav se uporablja bolj ali manj samo za to, tako da je primerna tudi za rokovanje s poljubnimi viri preko daljšega časovnega obdobja. Eventbrite bi lahko uporabili tudi za pralnico. Možnost uporabe bi bila, da bi kreirali nov dogodek z dnevom in časom, omejili bi število gostov na enega in nastavili ceno karte. V takem primeru bi bilo kreiranje dogodkov zamudno, ker bi morala odgovorna oseba kar naprej dodajati nove dogodke in bi s tem izgubljala veliko časa. Druga možnost pa bi bila, da bi naredili svoj ospredni del in bi se na to aplikacijo povezovali preko javnega API-ja, kar bi bilo časovno manj zamudno, bi pa morali ravno tako implementirati uporabniški vmesnik, težko pa bi bilo tudi implementirati vse željene funkcionalnosti. Zato smo se odločili, da kreiramo celotno spletno aplikacijo sami. Naša spletna aplikacija avtomatsko generira nove termine, ko se začne nov dan. Prav tako naša aplikacija na koncu meseca avtomatsko izdela poročilo za tajništvo, preko katerega v tajništvu obračunajo porabo preko položnice ter ima možnost prepovedi uporabe za določene uporabnike.

V diplomi so najprej predstavljene tehnologije in metoda razvoja programske rešitve. V nadaljevanju sledi analiza zahtev in načrtovanje celotne rešitve. Nato sledi predstavitev izvedbe, zatem pa testiranje. Na koncu pa sledijo še sklepne ugotovitve.

Poglavje 2

Tehnologije in metode razvoja

V okviru diplomske naloge smo uporabili različne programske jezike. Pri izdelavi smo upoštevali tudi dobre programerske prakse, sledili pa smo tudi metodologiji razvoja. Ker je programska rešitev razdeljena na tri dele, to je podatkovno bazo, spletni strežnik in spletno aplikacijo, smo te teme razdelili smisleno v tri podpoglavja katerim sledi še opis metodologije razvoja, ki smo jo uporabili pri izdelavi aplikacije.

2.1 Podatkovna baza

Pri modernih spletnih aplikacijah je izjemno pomembno rokovanje s podatki. Pri tem so še vedno glavno orodje relacijske podatkovne baze s svojim jezikom, ki je za različne sisteme za rokovanje s podatkovnimi bazami bolj ali manj enak, in sam sistem za upravljanje relacijskih podatkovnih baz.

SQL

SQL (Structured Query Language) je v uporabi že več kot 30 let v različnih bazičnih tehnologijah. Je standarden jezik za ustvarjanje, poizvedovanje, spreminjanje in brisanje podatkov v podatkovnih bazah [15]. Jezik je sestavljen iz različnih ukazov, ki so namenjeni upravljanju s podatkovnimi bazami.

Nekaj najbolj uporabljenih ukazov, ki smo jih uporabljali tudi v okviru naše diplomske naloge [14]:

- CREATE - za kreiranje novih podatkovnih baz in tabel,
- SELECT – za pridobivanje podatkov iz podatkovne baze,
- UPDATE – za spreminjanje podatkov v podatkovni bazi,
- DELETE – za brisanje podatkov v podatkovni bazi,
- INSERT INTO – za vnos novih podatkov v podatkovno bazo.

SQL Server Management Studio

SQL Server Management Studio (SSMS) [13] je integrirano okolje za upravljanje katere koli infrastrukture SQL. Uporablja se za dostopanje, konfiguracijo, administracijo in razvijanje komponent SQL serverja. SSMS združuje široko skupino grafičnih orodij s številnimi bogatimi urejevalniki skript, ki omogočajo dostop do SQL serverja. Namenjen je razvijalcem in administratorjem podatkovnih baz.

2.2 Spletna storitev

Za dostopanje do podatkovne baze skrbi strežniški program, ki je v našem primeru implementiran kot spletna storitev. Spletna storitev je aplikacija, ki se izvaja na oddaljenem računalniku, do nje pa lahko dostopamo preko standardnega aplikacijskega programskega vmesnika (API) za neko storitev. Spletno storitev smo razvili na platformi ASP.NET Core, pri razvoju pa smo uporabili jezik C#.

C#

To je objektno usmerjen programski jezik, ki ga je razvil Microsoft v okviru ogrodja .NET. Zgleduje se po številnih programskih jezikih najbolj pa po

Javi, C in C++ [3]. Za C# smo se odločili zaradi preteklih izkušenj, dobre podpore in dobre integracije v platformi ASP.NET Core.

ASP.NET CORE

To je zmogljiva odprta platforma razvita s strani Microsofta in je modularno ogrodje. V njem lahko razvijamo spletne aplikacije in mobilne storitve za Windows, macOS in Linux sisteme [5].

Spletna storitev REST

Representational State Transfer (REST) je novejši način h komunikaciji med raznimi napravami, ki so povezane v svetovni splet in predstavlja alternativo standardu SOAP (Simple Object Access Protocol). Pri tem načinu lahko naprave komunicirajo med sabo z uporabo protokola HTTP (HyperText Transfer Protocol). REST uporablja osnovne HTTP metode POST, GET, PUT in DELETE, ki se preslikajo v operacije CRUD (Create, Read, Update in Delete) nad podatkovno bazo ter metodo OPTIONS [11]:

- POST: Posodabljanje podatkov na strežniku,
- GET: Pridobi podatke iz strežnika,
- PUT: Ustvarjanje novih podatkov na strežniku,
- DELETE: Brisanje podatkov s strežnika,
- OPTIONS: Uporabljen za pridobivanje možnih operacij na strežniku.

Odjemalec preko protokola HTTP pošlje zahtevo na dogovorjen naslov URI (Uniform Resource Identifier), na katero strežnik odgovori. Odgovor je sestavljen iz statusne kode in glave ter podatkov. Statusna koda je trimestna koda, ki predstavlja informacijo o odgovoru. Razdeljena je na pet vrst. Vsaka vrsta ima svojo predpono. Statusne kode odgovora HTTP:

- 1XX informacija,

- 2XX uspeh,
- 3XX preusmeritev,
- 4XX napaka zahteve,
- 5XX napaka na strežniku.

Podatki, ki so vrnjeni s strani strežnika, so lahko v različnih bolj ali manj standardnih oblikah, največkrat pa se uporablja format JSON, pogosto pa tudi XML [10].

JSON

JavaScript Object Notation (JSON) je oblika podatkov, ki je preprosta za branje in pisanje. Program jo enostavno razčleni in generira. JSON je tekstovni format, ki je popolnoma neodvisen od jezika, čeprav je bil na začetku mišljen kot podmnožica jezika JavaScript. Pozna ga več programskih jezikov vključno s C, C++, C#, Java, JavaScript, Perl, Python ter mnogi drugi. Te lastnosti omogočajo, da je JSON idealen jezik za izmenjavo podatkov. JSON je zgrajen na dveh strukturah.

- Zbirka parov (ključ, vrednost), ki je v večini realiziran kot objekt, zapis, slovar, seznam, matrika.
- Urejen seznam vrednosti. V večini jezikov je to realizirano kot matrika, vektor, seznam ali zaporedje.

Skoraj vsi sodobni programski jeziki podpirajo rokovanje s formatom JSON v eni ali drugi obliki [7].

2.2.1 JWT

JSON Web Token (JWT) je odprt standard za varen prenos informacij po internetu med dvema strankama. Informacije se prenašajo kot JSON objekt. JWT se uporablja za avtentikacijo uporabnikov ali prenos informacij med dvema strankama. JWT objekt je sestavljen iz treh delov [19]:

- Glava: vsebuje dve lastnosti: tip žetona in kateri šifrirni algoritem je bil uporabljen.
- Podatki: vsebuje različne podatke o entitetah itd. Podatki so lahko privatni ali javni.
- Podpis: vsebuje izvleček šifrirnega algoritma, ki vzame glavo, podatke in privatni ključ ter vse skupaj šifrira. Podpis nam pove, da sporočilo ni bilo spremenjeno.

Visual Studio 2017

To je integrirano razvojno okolje (IDE), ki ga je razvil Microsoft. IDE je program, bogat s funkcijami, ki se lahko uporabljajo za razvoj programske opreme. Poleg standardnega urejevalnika in razhroščevalnika, ki ga zagotavlja večina IDEjev, vključuje Visual Studio tudi prevajalce, orodja za dokončanje kode, grafične oblikovalce in še mnogo funkcij za lažji razvoj programske opreme [16].

2.3 Spletna Aplikacija

Za uporabnika najpomembnejši del aplikacije je uporabniški vmesnik, ki se prikaže v brskalniku in uporabniku omogoča upravljanje s sredstvi, ki so na razpolago. Za implementacijo tega dela imamo na voljo nekaj tehnologij, ki so opisane v nadaljevanju.

HTML

HyperText Markup Language (HTML) [20] je standardni jezik za oblikovanje spletnih strani. HTML opisuje strukturo spletnih strani z oznakami, te oznake pa so gradniki spletnih strani. Oznake označujejo koščke vsebine kot so naslov, odstavek, tabela, slike, video posnetki itd. Brskalnik ne prikazuje oznak HTML, vendar jih uporablja za prikaz vsebine strani.

CSS

Kaskadne slogovne podloge (CSS) [4] je sintaksa za določanje slogov dokumentov. Opisuje, kako naj bodo elementi HTML prikazani na zaslonu. Določamo lahko barve, velikosti, odmike, poravnave, itd. Prihrani nam veliko dela, hkrati pa lahko nadzorujemo več spletnih strani, saj se CSS datoteke lahko nahajajo nekje na strežniku. Na njih se lahko sklicujemo iz različnih HTML strani.

Material Design

Gre za oblikovni jezik, ki združuje klasična načela uspešnega oblikovanja skupaj z inovacijami in tehnologijo [9]. Razvil ga je Google za potrebe oblikovanja svojih aplikacij, da bi omogočali enotno uporabniško izkušnjo na vseh svojih izdelkih.

JavaScript

To je programski jezik za komunikacijo s HTML-om oziroma objekti, ki predstavljajo nek dokument [6]. Prvenstveno je jezik namenjen izvajanju v brskalniku. Omogoča izdelavo dinamičnih spletnih strani in programsko dostopanje do DOM elementov HTML-a ter njegovo spreminjanje. Z njim lahko programiramo efekte, tranzicije, omogoča pa tudi izdelavo svojih komponent za spletne strani.

TypeScript

Je razširjen JavaScript, ki je bil narejen za razvijanje velih JavaScript aplikacij [18]. TypeScript je močno tipiziran in je objektno usmerjen. Izdelal ga je Anders Hejlsberg (oblikovalec C#) pri Microsoftu. Je tipizirana nadgradnja JavaScripta, ki se na koncu preko prevajalnika pretvori v JavaScript. Z drugimi besedami: TypeScript je JavaScript z nekaj dodatnimi funkcijami.

Angular

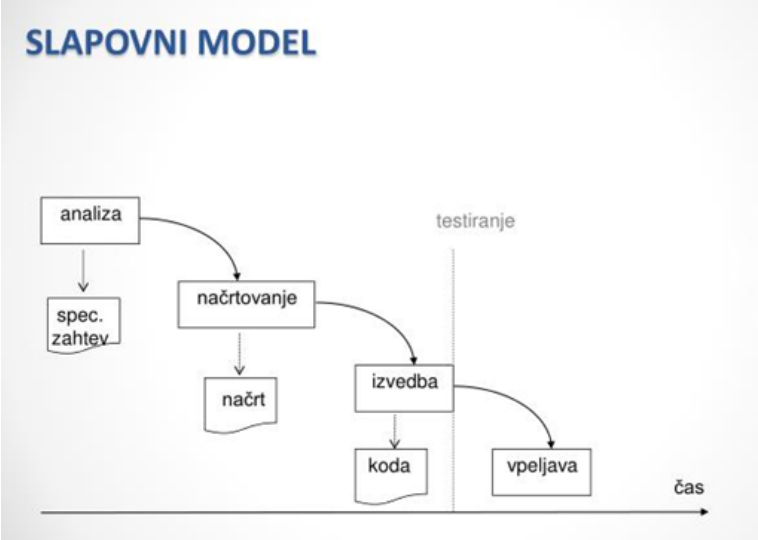
Angular [1] je platforma, ki omogoča lažje razvijanje spletnih aplikacij in združuje dekorativne predloge. Integrirane ima najboljše rešitve za programske izzive. Angular daje razvijalcem možnost za razvijanje spletnih, mobilnih ali namiznih aplikacij. Za izdelavo aplikacij se uporablja HTML, CSS ter TypeScript. Angular je napisan v TypeScriptu in implementira osnovne in opcijske funkcionalnosti, kot knjižnice, ki so spisane v TypeScriptu [2].

Visual Studio Code

Visual Studio Code [17] je program za urejanje izvorne kode programa. Razvil ga je Microsoft za Windows, Linux in macOS. Vsebuje orodja za odpravljanje napak, vgrajeno kontrolo Git itd. Prav tako je prilagodljiv, saj omogoča, da lahko uporabniki spremenijo uredniško temo in bližnjice na tipkovnici. Je odprtokoden in brezplačen program.

2.4 Slapovni model

Slapovni model razvoja informacijskega sistema [12] je najstarejši model, ki temelji na zaporednem izvajanju faz. Faze si sledijo v naslednjem zaporedju: analiza, načrtovanje, izvedba, testiranje in vpeljava, kot je prikazano na sliki 2.1. Ko se ena faza v celoti končna, se začne druga, vračanje v predhodne faze pa ni mogoče. Omogoča natančno in dobro projektno vodenje. Primeren je za relativno kompleksne projekte, ko zadeve dobro razumemo in se med projektom zahteve ne bodo bistveno spreminjale. Primeren je za projekte, kjer se da načrtovanje v celoti izvesti vnaprej, saj ni fleksibilen, vsaka naknadna sprememba pa zahteva veliko dodatnega dela. Težava pri tem modelu je tudi, da v je naravi težko pričakovati, da se bo nek postopek zaključil v celoti, preden se bo začel naslednji postopek. Ne omogoča paralelnega izvajanja delov postopkov.



Slika 2.1: Slapovni model razvoja

Analiza

V tej fazi se zbirajo zahteve projekta od poslovnega konteksta stranke do ravni zmogljivosti izdelka. Ustvari se specifikacijo zahtev na katerih se analizira funkcionalne (to so operacije, ki jih aplikacija vsebuje, npr. izpis mesečnega poročila) in nefunkcionalne (to so zmogljivosti sistema, npr. dano aplikacijo lahko uporablja do 100 uporabnikov) zahteve. Tu se opiše obseg in omejitve projekta.

Načrtovanje

Z zaključkom predhodne faze analize se delo nadaljuje na zasnovi sistema vključno s programsko in strojno arhitekturo, podatkovno bazo ter uporabniškim vmesnikom. Nastala specifikacija projekta opisuje, kako bo projekt izveden s tehničnega vidika.

Izvedba

Ko je potrjeno načrtovanje, se začne pisanje kode v skladu z nastalo specifikacijo projekta.

Testiranje

Ko se zaključi izvedba, se začne testiranje vseh funkcionalnosti, da bi našli napake in pomanjkljivosti ter se prepričali, če program deluje pravilno.

Vpeljava

Ta faza obsega postavitev programske opreme v realno, to je produkcijsko okolje. Vključuje tudi vzdrževanje in naknadno podporo izdelku.

Poglavje 3

Analiza in načrtovanje

V tem poglavju sta predstavljena koraka analize in načrtovanja spletne aplikacije.

3.1 Analiza

V okviru analize smo preučili vse funkcionalne in nefunkcionalne zahteve aplikacije in jih razdelili na več sklopov, opisanih v nadaljevanju.

Sistemske zahteve

- Spletna aplikacija bo delovala na internem strežniku in bo dostopna preko intraneta.
- Potrebno je zavarovati dostop do podatkov preko uporabniškega imena in gesla.
- Omogočati mora rezervacijo terminov.
- Pralni termini so štirje: od 6:00 do 9:00, od 10:00 do 13:00, od 14:00 do 17:00, od 18:00 do 21:00.
- Možnost dodajanja novih pralnih sob ter možnost spreminjanja in brisanja obstoječih pralnih sob.

- V primeru, da uporabnik zamudi pri vračilu ključa več kot 15 minut, se mu na koncu meseca zaračuna zamudnina.

- Drugačen prikaz glede na skupino uporabnika: uporabnik, tajništvo, administrator ali receptor.

Uporabniške zahteve

- Uporabnik se mora registrirati v aplikacijo z naslednjimi podatki: ime, priimek, e-pošta, številka sobe, geslo in uporabniško ime.

- Po registraciji mora biti uporabnik odobren s strani administratorja.

- Ko se uporabnik prijavi v spletno aplikacijo, mora imeti na voljo pregled prostih terminov, na katerih lahko opravi rezervacijo.

- Zaradi večjega števila uporabnikov od števila razpoložljivih pralnih strojev, lahko uporabnik pere le enkrat na pet dni.

- Uporabnik lahko opravi rezervacijo termina ali pa izbriše rezervacijo le v primeru, če to ni isti dan.

- Uporabnik ne sme videti, kdo je rezerviral preostale termine.

Receptorske zahteve

- Receptor dobi uporabniško ime in geslo.

- Ob prijavi v aplikacijo se mu prikažejo rezervacije za tekoči dan.

- Ob predaji ključa mora v aplikaciji označiti, da je uporabnik prevzel ključ.

- Ob vrnitvi ključa mora označiti, da je uporabnik vrnil ključ.

Tajniške zahteve

- Tajnik dobi uporabniško ime in geslo.
- Ob prijavi v aplikacijo ima ta na voljo dva pregleda.
- Eden je pregled po posameznih uporabnikih.
- Drugi je pregled po mesecih za vse uporabnike, ki vključuje število vseh pranj in zamudnin.

Administratorske zahteve

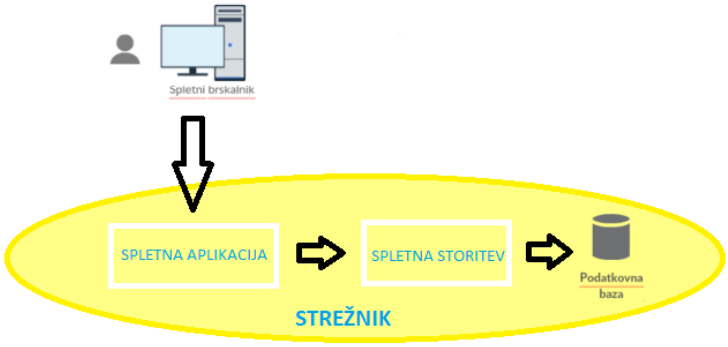
- Administrator dobi uporabniško ime in geslo.
- Ima pregled po uporabnikih.
- Uporabnikom omogoča in onemogoča dostop do spletne aplikacije.
- Vsakemu novemu uporabniku mora najprej odobriti dostop do aplikacije.
- Ima pregled in možnost brisanja rezervacij terminov.
- Pregled po mesecih za vse uporabnike, ki vključuje število vseh pranj in zamudnin.
- Ima možnosti dodajanja, urejanja in brisanja pralnih sob ter urejanja števila pralnih strojev v posamezni pralni sobi.
- Lahko označi sobo za neaktivno, kar pomeni, da nihče ne more rezervirati termina za tisto pralno sobo.

3.2 Načrtovanje

V koraku načrtovanja pa je bila določena arhitektura sistema, narejen je bil podatkovni model, prav tako pa je bil določen tudi uporabniški vmesnik aplikacije.

Arhitektura sistema

Podatkovna baza je postavljena na Microsoft SQL Server Express 2017. Spletna storitev REST je napisana v ogrodju .NET CORE, postavljena pa je na strežniku IIS verzije 10. Na strežniku IIS je postavljena tudi spletna aplikacija, ki je spisana v ogrodju Angular in se povezuje na spletno storitev. Slika 3.1 prikazuje arhitekturo programske rešitve.



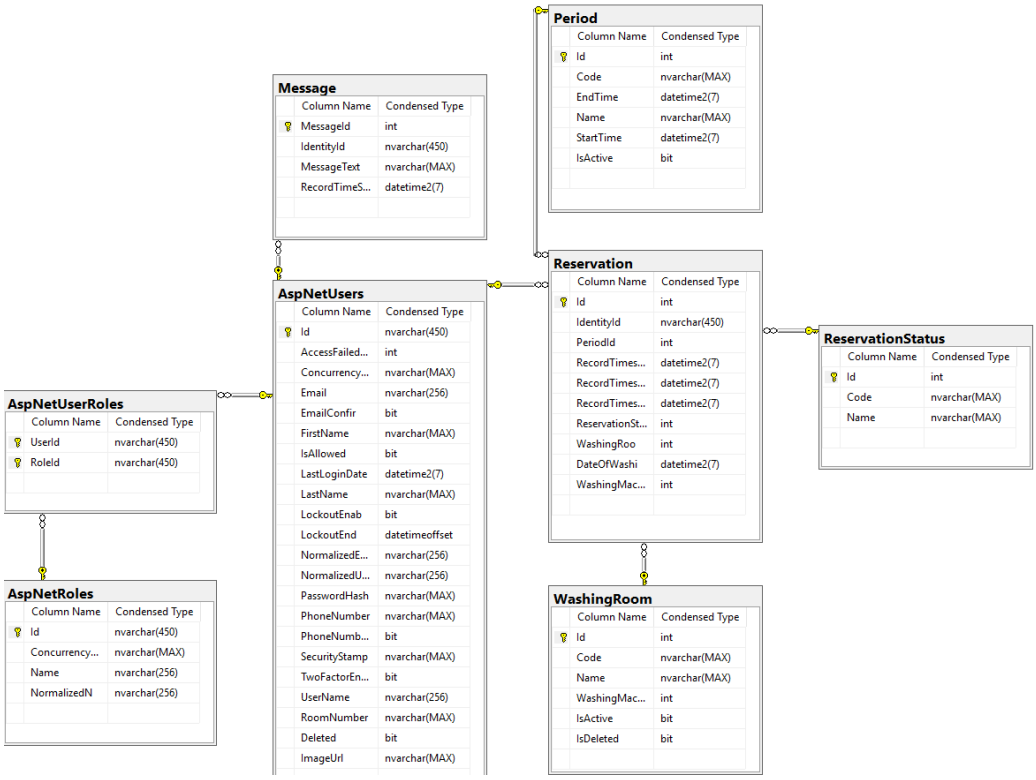
Slika 3.1: Arhitektura sistema.

Podatkovni model

Po izvedeni analizi je bilo potrebno nadaljevati z načrtovanjem podatkovne baze. Med razvojem programske rešitve se je zaradi spreminjanja in dodajana funkcionalnosti podatkovna baza spreminjala. Model je bil narejen s pomočjo ASP.NET CORE Entity frameworka pri katerem je bil uporabljen „code first“ način za izdelavo podatkovnega modela. Ob zaključku programske rešitve je bil podatkovni model sestavljen iz osmih tabel. Te so prikazane na sliki 3.2.

Tabela AspNetUsers

V tej tabeli so shranjeni podatki o registriranih uporabnikih. Tabela izhaja iz ASP.NET CORE Identity, ki je uporabljen za pokrivanje funkcionalnosti



Slika 3.2: Podatkovni model.

prijave in registracije v ASP.NET. Zaradi tega so v tabeli nekatera polja neuporabljena. Spodaj so opisana zgolj uporabljena polja.

- Id – enolična oznaka (ID) posameznega uporabnika,
- Email – e-poštni naslov uporabnika,
- FirstName - ime uporabnika,
- LastName – priimek uporabnika,
- LastLoginDate – čas zadnje prijave uporabnika,
- RoomNumber – številka sobe, v kateri uporabnik prebiva,
- PasswordHash – zgoščeno geslo uporabnika,
- UserName – uporabniško ime uporabnika,
- Delete – polje, ki pove ali je uporabnik izbrisan iz seznama,
- IsAllowed – polje, ki pove ali ima uporabnik pravice, da se lahko prijavi v sistem,
- ImageUrl – URL do prikazne slike uporabnika.

Tabela AspNetRoles

Ta tabela prav tako izhaja iz ASP.NET CORE Identityja v njej pa so shranjene skupine uporabnikov.

- Id – enolična oznaka (ID) posameznega uporabnika,
- Name – ime skupine,
- NormalizedName – opis skupine.

Tabela AspNetUserRoles

Tudi ta tabela izhaja iz ASP.NET CORE Identityja. Je povezovalna tabela, ki pove, v katero skupino pripada določen uporabnik.

- UserId – enolična oznaka (ID) uporabnika,
- RoleId – enolična oznaka (ID) skupine.

Tabela Message

To je tabela, v kateri se nahajajo sporočila, ki so jih poslali uporabniki o napakah sistema za rezervacijo in so namenjena administratorju.

- MessageId – enolična oznaka (ID) sporočila,
- IdentityId – enolična oznaka (ID) uporabnika,
- MessageText – vsebina sporočila,
- RecordTimeStamp – čas, ko je bilo sporočilo poslano.

Tabela Period

Tabela, v kateri so shranjeni možni termini za rezervacijo.

- Id – enolična oznaka (ID) termina,
- Code – enolična oznaka termina,
- EndTime – končni čas termina,
- StartTime – začetni čas termina,
- Name – ime termina,
- IsActive – polje, ki pove, če je termin v uporabi.

Tabela WashingRoom

Tabela, v kateri so shranjeni podatki pralne sobe.

- Id - enolična oznaka (ID) pralne sobe,
- Code – enolična oznaka sobe,
- WashingMachineCount – število pralnih strojev v sobi,
- IsActive – polje, ki pove, če je soba na voljo,
- IsDeleted – polje, ki pove, če je soba izbrisana iz seznama.

Tabela ReservationStatus

Tabela, ki ima shranjen status posamične rezervacije.

- Id – enolična oznaka (ID) statusa rezervacije,
- Code – enolična oznaka statusa,
- Name – opis statusa.

Tabela Reservation

To je glavna tabela aplikacije. V njej so shranjene vse rezervacije uporabnikov, vsebuje pa naslednja polja:

- Id – enolična oznaka (ID) rezervacije,
- IdentityId – enolična oznaka (ID) uporabnika,
- PeriodId – enolična oznaka (ID) termina,
- RecordTimestamp – čas rezervacije,
- RecordTimestampFinish – čas ko je uporabnik vrnil ključ,
- RecordTimestampStart – čas ko je uporabnik prevzel ključ,

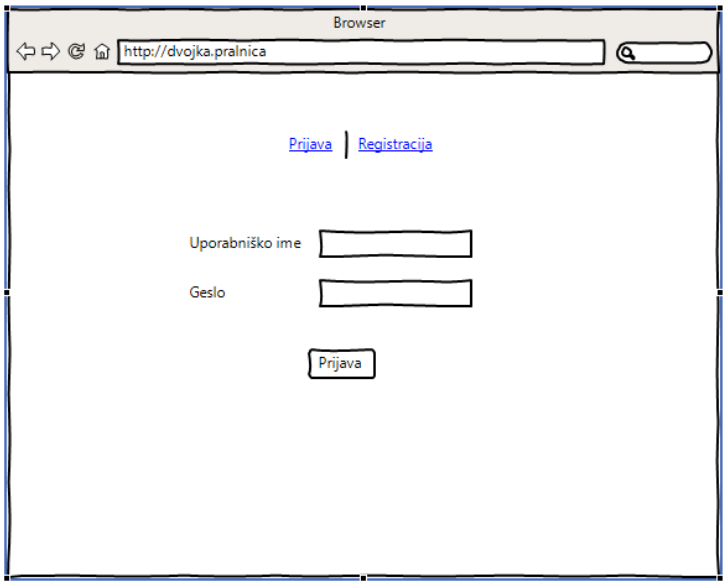
- ReservationStatusId – enolična oznaka (ID) statusa rezervacije,
- WashingRoomId – enolična oznaka (ID) sobe,
- DateOfWashing – datum pranja,
- WashingMachineCount – število pralnih strojev.

Spletna aplikacija

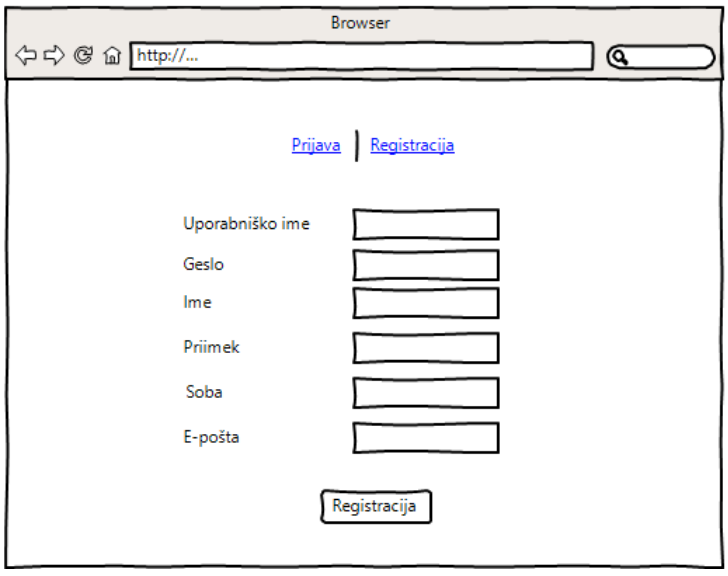
Spletna aplikacija je namenjena vsem uporabnikom pralnice. Ta omogoča nove rezervacije, vsebuje pregled mesečnih terminov pranj in rezervacij, omogoča brisanje, dodajanje ter urejanje pralnih sob, omogoča ali onemogoča prijavo uporabnikov ter brisanje uporabnikov iz aplikacije. V fazi načrtovanja smo na podlagi analize narisali tudi prototip aplikacije oziroma strani, ki jih mora le-ta imeti. Na vrhu spletne aplikacije se nahajajo navigacija, ki vsebuje povezave na druge strani in logotip pralnice ter gumb za odjavo iz aplikacije.

Slika 3.3 prikazuje papirnati prototip prijavne strani spletne aplikacije. V sredini okna sta polji za vnos uporabniškega imena in gesla ter gumb za prijavo v aplikacijo. V primeru nepravilne prijave pa se mora prikazati napis „Napačno uporabniško ime ali geslo“.

Slika 3.4 prikazuje papirnati prototip spletne strani za registracijo novega uporabnika. Ta stran vsebuje polja za vnos uporabniškega imena, gesla, imena, priimka, sobe in e-poštnega naslova. Poleg tega pa vsebuje tudi gumb za registracijo uporabnika. Ob uspešni registracije se mora pojaviti napis „Uspešna registracija“.

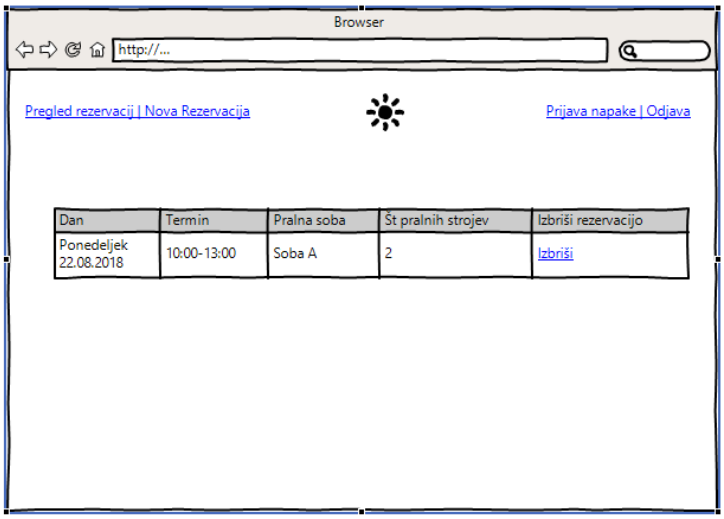


Slika 3.3: Primer okna za prijavo.



Slika 3.4: Primer spletnega okna za registracijo.

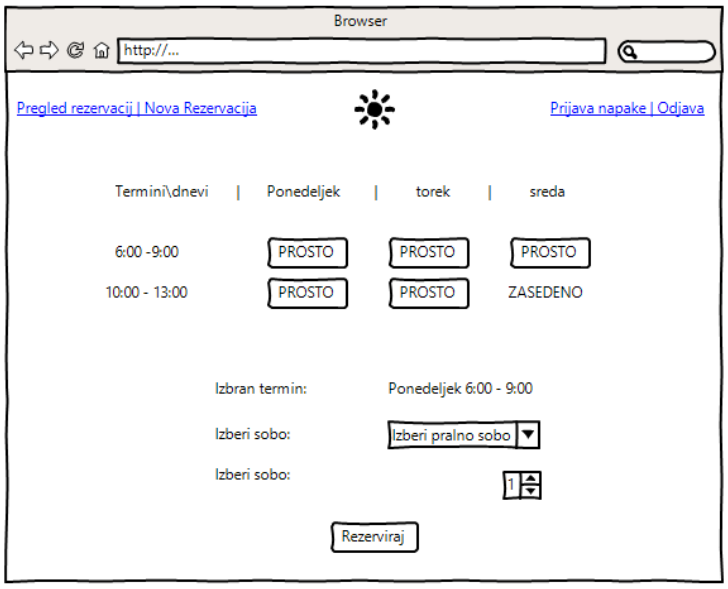
Slika 3.5 prikazuje papirnati prototip prikaza pregleda rezervacij za uporabnika. V tem prikazu vidi svoje pretekle in sedanje rezervacije. Vsebuje tabelo s pregledom rezervacij, ki vsebuje stolpce Dan, Termin, Pralna soba, Št. pralnih strojev in Izbriši rezervacijo, ki vsebuje gumb za izbris rezervacije, če se ta še ni zgodila. Prav tako ima gumb „Odjava“, kjer se uporabnik lahko odjavi.



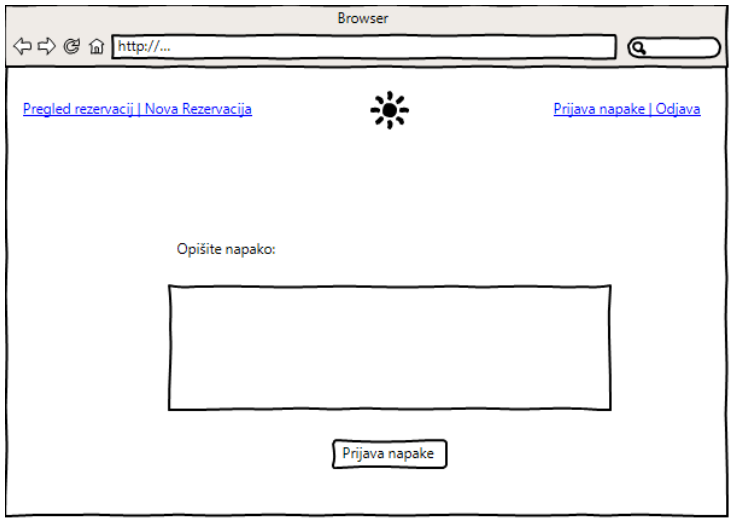
Slika 3.5: Primer vstopnega spletnega okna za uporabnika.

Slika 3.6 prikazuje papirnati prototip spletne strani, v kateri uporabnik naredi novo rezervacijo. Najprej izbere termin, tako da klikne na gumb „PROS“. Potem izbere pralno sobo ter število pralnih strojev. S spodnjim gumbom „Rezerviraj“ pa izvede dokončno rezervacijo.

Slika 3.7 prikazuje papirnati prototip spletne strani za prijavo napake, ki se je zgodila ob uporabi spletne aplikacije. Vsebuje vnosno polje za opis napake in gumb za prijavo napake, s katerim se pošlje sporočilo administratorju.

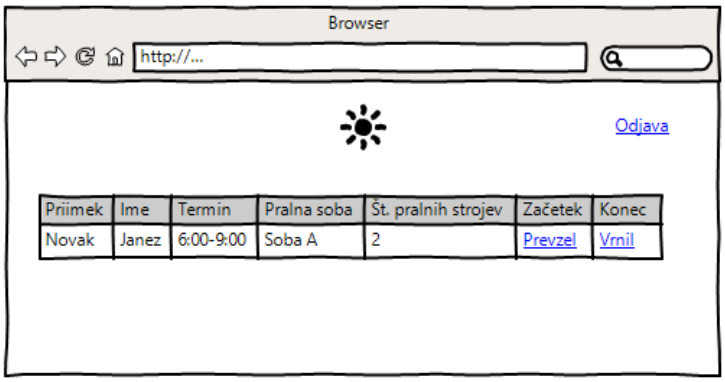


Slika 3.6: Primer spletnega okna za rezervacijo termina.



Slika 3.7: Spletno okno za prijavo napake.

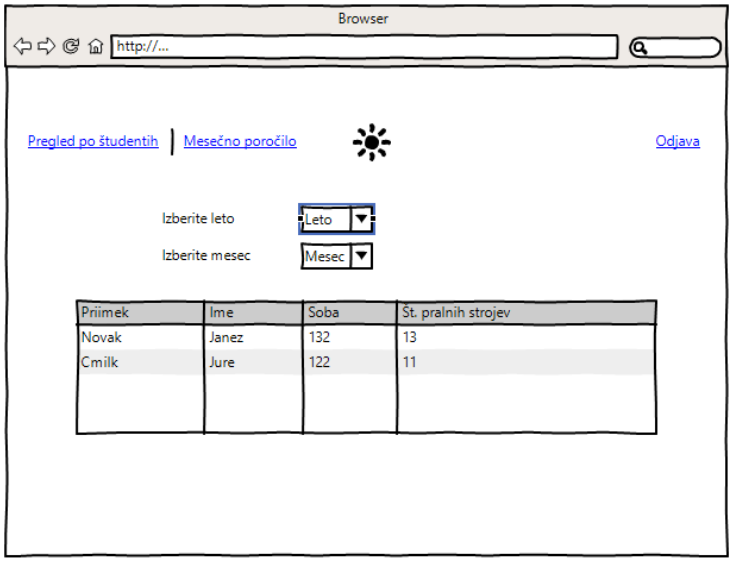
Slika 3.8 prikazuje papirnati prototip receptorskega pregleda rezervacij, v katerem se nahaja seznam rezervacij za tekoči dan. S klikom na gumb „Odjava“ se odjavi iz spletne aplikacije.



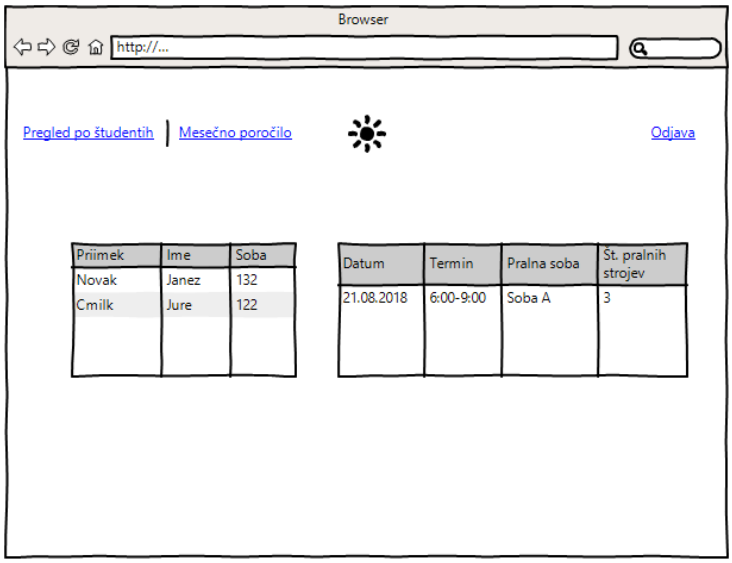
Slika 3.8: Spletno okno ki ga vidi receptor.

Slika 3.9 prikazuje papirnati prototip spletne strani za prikaz mesečnega poročila, namenjeno tajništvu in administratorju, kjer lahko izbereta leto in mesec. Po izbiri želenega termina se prikažejo podatki za izbrano leto in mesec v obliki tabele. S klikom na gumb „Odjava“ se administrator ali tajništvo odjavita iz spletne aplikacije.

Slika 3.10 prikazuje papirnati prototip spletne strani za prikaz pregleda uporabnikov za tajništvo in administratorja. Ta stran prikazuje v desni tabeli vsa pranja za izbranega uporabnika iz tabele uporabnikov, ki je na levi strani strani.

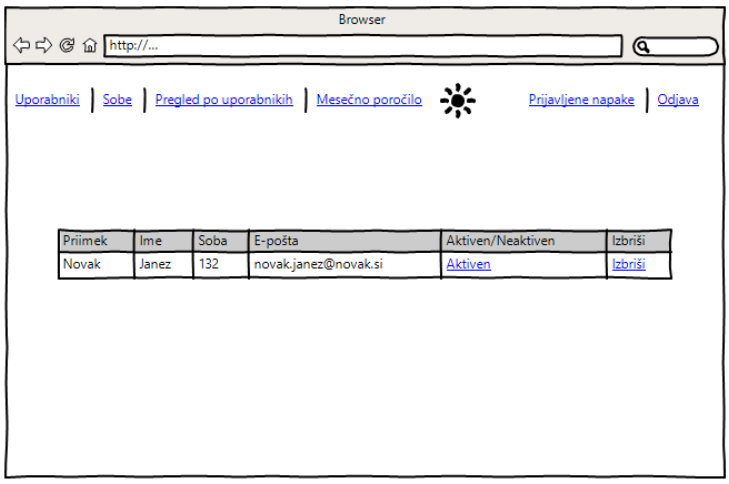


Slika 3.9: Spletno okno prikazuje mesečno poročilo števila pranj.



Slika 3.10: Spletno okno prikazuje podatke o pranju za izbranega uporabnika.

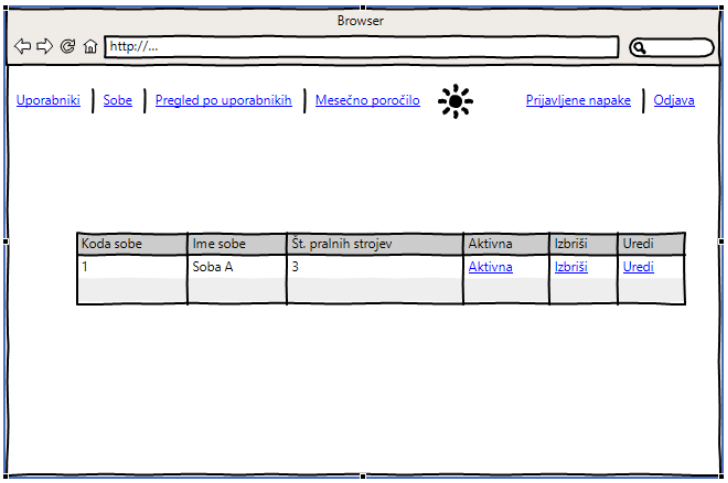
Slika 3.11 prikazuje papirnati prototip pregleda po uporabnikih v tabelarični obliki s stolpci Priimek, Ime, Soba, E-pošta, Aktiven/Neaktiven, Izbriši. S klikom na gumb „Aktiven“ se spremeni status uporabnika v status aktiven s čimer omogočimo uporabniku, da se lahko prijavi. S klikom na gumb „Izbriši“ se uporabnika izbriše iz aplikacije.



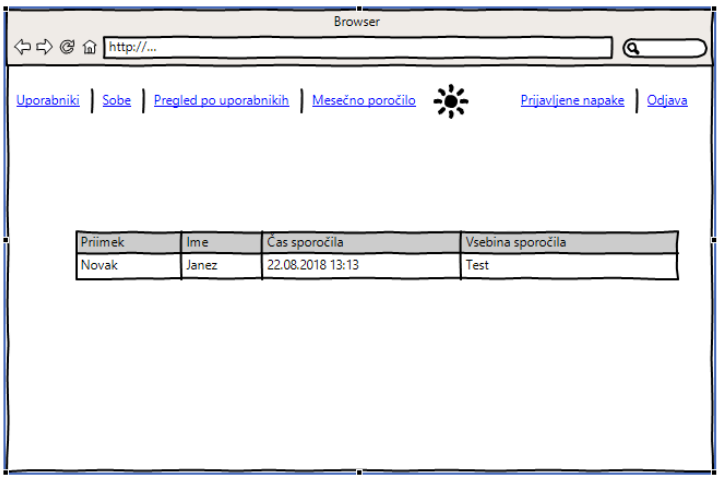
Slika 3.11: Spletno okno, kjer so prikazani uporabniki.

Slika 3.12 prikazuje papirnati prototip spletne strani s pralnimi sobami, v kateri se nahaja tabela s sobam, ki vsebuje stolpce Koda sobe, Ime sobe, Št. pralnih strojev, Aktivna, Izbriši in Uredi. S klikom na gumb „Izbriši“ se pralno sobo izbriše iz aplikacije. S klikom na gumb „Uredi“ pa se lahko pralno sobo ureja.

Slika 3.13 prikazuje spletno stran prijavljene napake, v kateri se nahaja tabela s poslanimi sporočili, ki vsebuje priimek in ime uporabnika, čas odposlanega sporočila in vsebino sporočila.



Slika 3.12: Spletno okno za dodajanje in urejanje informacij o pralnih sobah.



Slika 3.13: Spletno okno v katerem so prikazana sporočila uporabnikov.

Poglavje 4

Izvedba

Pri izvedbi smo se najprej lotili izdelave podatkovne baze, katere struktura je že prikazana v načrtovanju. Nato smo se lotili razvoja spletne storitve REST v ogrodju ASP.NET Core. Dostop do spletne storitve smo zavarovali z JSON Web Token (JWT) avtentikacijo. Za tem smo razvili še spletno aplikacijo v Angularju.

4.1 Podatkovna baza

Podatkovna baza je bila implementirana s pomočjo ASP.NET Entity Frameworka, v načinu „code first“. S programskim jezikom C# smo najprej naredili entitetne razrede Message.cs, AppUser.cs (izpeljan iz ASP.NET Identityja), Period.cs, Reservation.cs, ReservationStatus.cs ter WashingRoom.cs, ki vsebujejo atribute in lastnosti atrtibutov. Slika 4.1 prikazuje primer razreda entitete. Prav tako vsebuje razred ApplicationDbContext.cs izpeljan iz IdentityDBContext, ki predstavlja sejo s podatkovno bazo in omogoča dodajanje, branje, urejanje in brisanje podatkov iz podatkovne baze za potrebe CRUD operacij spletne storitve. Slika 4.2 prikazuje primer ApplicationDbContext.cs.

```
6 references | 0 changes | 0 authors, 0 changes
public class Message
{
    0 references | 0 changes | 0 authors, 0 changes
    public int MessageId { get; set; }
    0 references | 0 changes | 0 authors, 0 changes
    public string MessageText { get; set; }
    1 reference | 0 changes | 0 authors, 0 changes
    public DateTime? RecordTimeStamp { get; set; }
    0 references | 0 changes | 0 authors, 0 changes
    public string IdentityId { get; set; }
    0 references | 0 changes | 0 authors, 0 changes
    public AppUser Identity { get; set; }

}
```

Slika 4.1: Primer razreda entitete.

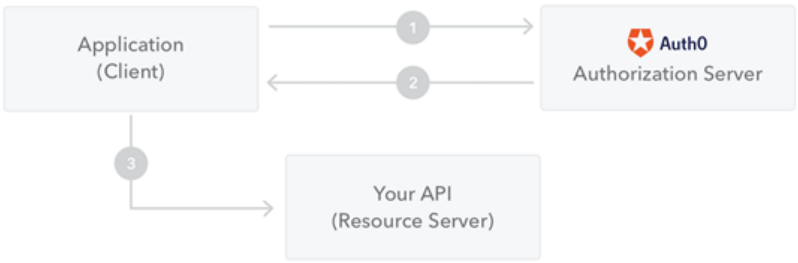
```
21 references | 0 changes | 0 authors, 0 changes
public class ApplicationDbContext : IdentityDbContext<AppUser>
{
    0 references | 0 changes | 0 authors, 0 changes
    public ApplicationDbContext()
    {
    }

    0 references | 0 changes | 0 authors, 0 changes
    public ApplicationDbContext(DbContextOptions options) : base(options)
    {
    }

    0 references | 0 changes | 0 authors, 0 changes
    public DbSet<Period> Periods { get; set; }
    4 references | 0 changes | 0 authors, 0 changes
    public DbSet<Reservation> Reservations { get; set; }
    5 references | 0 changes | 0 authors, 0 changes
    public DbSet<WashingRoom> WashingRooms { get; set; }
    0 references | 0 changes | 0 authors, 0 changes
    public DbSet<ReservationStatus> ReservationStatuses { get; set; }
    1 reference | 0 changes | 0 authors, 0 changes
    public DbSet<Message> Message { get; set; }

    0 references | 0 changes | 0 authors, 0 changes
    protected override void OnModelCreating(ModelBuilder builder)
    {
        base.OnModelCreating(builder);
        builder.Entity<Period>().ToTable("Period");
        builder.Entity<Reservation>().ToTable("Reservation");
        builder.Entity<WashingRoom>().ToTable("WashingRoom");
        builder.Entity<ReservationStatus>().ToTable("ReservationStatus");
        builder.Entity<Message>().ToTable("Message");
    }
}
```

Slika 4.2: Razred, ki izhaja iz EntityFramework kontektsa.

4.2 Spletna storitev

Za razvoj naše spletne storitve REST smo izbrali ASP.NET Core ogrodje. Za avtentikacijo klicev smo implementirali JWT varnost, na podlagi katere se uporabnik prijavi v aplikacijo in dobi žeton, ki je veljaven le nekaj minut. V vsakem naslednjem klicu spletne storitve mora uporabnik v glavi HTTP zahteve poslati še JWT žeton. Za delovanje spletne storitve smo ustvarili štiri kontrolne razrede, ki skrbijo za dostopne točke spletne storitve in vsebujejo funkcije, ki se izvedejo ob klicu kontrole točke.

Slika 4.3 prikazuje, kako smo v naši aplikaciji implementirali varnost pri spleti storitvi. Najprej se je bilo potrebno prijaviti, da nam je strežnik poslal JWT žeton. V vseh naslednjih klicih spletnega strežnika pa smo morali ta žeton pošiljati v glavi HTTP zahtevka.



Slika 4.3: Primer, kako izgleda JWT avtentikacija uporabnikov [8].

Za delovanje spletne storitve smo potrebovali kontrolne razrede, ki so skrbeli za odziv na klic spletne storitve, ti razredi so:

- ReservationsController.cs – skrbi za potek rezervacij,
- UserController.cs – skrbi za prijavo in registracijo uporabnikov,
- UsersController.cs – skrbi za urejanje uporabnikov s strani administratorja.

- WashingRoomController.cs – skrbi za urejanje pralnih sob s strani administratorja.

Slika 4.4 prikazuje primer kontrolerja in funkcije, ki se izvede ob klicu dostopne točke users/id. Spodaj je opisan kontrolni razred in funkcija.

- [Authorize(Roles="Admin")] – definiramo, kdo ima pravico klicati dostopne točke v kontrolerju. V tem primeru zgolj administrator spletne aplikacije, druge dostopne točke pa lahko kličejo tudi drugi uporabniki aplikacije, ki so lahko: uporabnik, tajništvo ali receptor. Prav tako lahko za vsako dostopno točko posebej definiramo, kdo jo lahko kliče.

- [Route("api/[controller]")] – tukaj definiramo pot do dostopnih točk. V tem primeru vzame ime razreda „users“ brez Controller.

- [HttpDelete("{id}")] – tu definiramo tip metode. V tem primeru je metoda DELETE. Drugi tipi metode: GET, POST, PUT.

- IActionResult deleteUser(string id) – tu definiramo funkcijo, ki se izvede za prejet „string id“. Funkcija lahko vrača različne HTTP odgovore z različnimi statusi.

Datoteka PralnicaRepository.cs vsebuje funkcije, ki manipulirajo s podatki nad podatkovno bazo, in se kličejo v kontrolerjih. Na podatkovno bazo smo se povezovali na dva načina:

- S pomočjo Entity framework seje, preko katere smo nad bazo izvajali osnovne CRUD operacije. Primer je prikazan na sliki 4.5.

- Z metodo SqlConnection smo na podatkovni bazi klicali SQL procedure. Te skrbijo za težje izvedljive funkcionalnosti aplikacije. Primer je prikazan na sliki 4.6.

Primer SQL procedure je prikazan na sliki 4.7. Ta procedura vsebuje dva vhodna parametra: „@date“, ki v proceduro vnese dan in „@period_id“, ki v proceduro vnese izbrani termin. Procedura vrne vse informacije o pralnih sobah, ki so še na voljo za vnešen dan in termin.

```
[Authorize(Roles = "Admin")]
[Route("api/[controller]")]
1 reference | 0 changes | 0 authors, 0 changes
public class UsersController : Controller
{
    IPralnicaRepository _repository;
    0 references | 0 changes | 0 authors, 0 changes
    public UsersController(IPralnicaRepository repository)
    {
        _repository = repository;
    }


    [HttpDelete("{id}")]
    0 references | 0 changes | 0 authors, 0 changes
    public IActionResult deleteUser(string id)
    {
        var user = _repository.GetUser(id);
        if (user == null)
        {
            return BadRequest();
        }
        user.IsAllowed = user.Deleted = true;

        _repository.UpdateUser(user);

        if (!_repository.Save())
        {
            return BadRequest();
        }
        return Ok();
    }
```

Slika 4.4: Primer kontrolnega razreda in funkcije.

```
2 references | 0 changes | 0 authors, 0 changes
public class PralnicaRepository : IPralnicaRepository
{
    private ApplicationDbContext _context;
    private IConfiguration _config;
    0 references | 0 changes | 0 authors, 0 changes
    public PralnicaRepository(ApplicationDbContext context, IConfiguration config)
    {
        _context = context;
        _config = config;
    }

    1 reference | 0 changes | 0 authors, 0 changes
    public void CreateAppUser(AppUser appUser)
    {
        _context.Users.Add(appUser);
    }
```

Slika 4.5: Primer klica z Entity frameworkom.

```
2 references | 0 changes | 0 authors, 0 changes
public List<GetMessage> GetAllMessages()
{
    List<GetMessage> newList = new List<GetMessage>();
    try
    {
        using (SqlConnection connection = new SqlConnection(_config.GetConnectionString("DefaultConnection")))
        {
            connection.Open();
            SqlCommand command = new SqlCommand("[dbo].[show_messages_sel]", connection);
            command.CommandType = CommandType.StoredProcedure;
            command.CommandTimeout = 5;
            GetMessage r;
            using (SqlDataReader oReader = command.ExecuteReader())
            {
                r = new GetMessage();
                while (oReader.Read())
                {
                    r.MessageText =oReader["MessageText"].ToString();
                    r.UserLastFirstName = oReader["userLastFirstName"].ToString();
                    newList.Add(r);
                    r = new GetMessage();
                }
                connection.Close();
            }
        }
    }
    catch (Exception ex) { /*Handle error*/ }
    return newList;
}
```

Slika 4.6: Primer klica na bazo z SqlConnection metodo.

```sql
-- =============================================
-- Author:      KAN
-- Create date: 12.07.2018
-- Description: Sobe ki so še na voljo za določen datum in termin
-- EXEC dbo.user_washing_room_sel @date ='2018-07-10', @period_id = 2
-- =============================================
ALTER PROCEDURE [dbo].[user_washing_room_sel]
    @date date,
    @period_id int
AS
BEGIN

SELECT Id,
       Code,
       Name,
       WashingMachineCount
FROM dbo.WashingRoom
WHERE IsActive =1 and IsDeleted = 0
AND Id NOT IN (SELECT WashingRoomId FROM dbo.Reservation WHERE DateOfWashing = @date and PeriodId = @period_id)


END
```

Slika 4.7: Primer SQL procedure.

4.2.1 Dostopne točke

V okviru aplikacije smo definirali tudi več dostopnih točk, ki omogočajo izvajanje CRUD operacij nad podatkovno bazo:

- user/login [GET] - ob veljavni prijavi vrne JWT žeton in skupino uporabnika,
- user/register [POST] - dostopna točka za registracijo uporabnika,
- user/reset [POST] - dostopna točka za spremembo gesla uporabnika,
- users [GET] - vrne vse uporabnike aplikacije,
- users/{id} [PUT] - spremeni status uporabnika iz aktivnega v neaktivnega uporabnika ali obratno, za uporabnika s podanim ID-jem,
- users/{id} [DELETE] - izbriše uporabnika s podanim ID-jem,
- washingroom [GET] - vrne vse pralne sobe aplikacije,
- washingroom [POST] - dodajanje nove pralne sobe,
- washingroom [PUT] - urejanje pralne sobe,
- washingroom/{id} [PUT] - spremeni status pralne sobe iz „Neaktivne“ v „Aktivno“ oziroma obratno, za pralno sobo s podanim ID-jem,
- washingroom/{id} [DELETE] - izbriše pralno sobo s podanim ID-jem,
- reservations [GET] - vrne vse možne rezervacije terminov za en teden,
- reservations [POST] - opravi rezervacijo za določenega uporabnika,
- reservations/{id} [DELETE] - izbriše rezervacijo s podanim ID-jem,
- reservations/{user_id} [GET] - vrne trenutne rezervacije uporabnika z ID-jem ”user_id”,

- reservations/h/{user_id} [GET] - vrne pretekle rezervacije uporabnika z ID-jem "user_id",
- reservations/{user_id}/{date} [GET] - vrne podatke za mesečni pregled rezervacij za uporabnika z ID-jem "user_id",
- reservations/{user_id}/image [POST] - doda ime prikazne slike v polje ImageUrl za uporabnika z ID-jem "user_id",
- reservations/report/{date} [GET] - vrne vse rezervacije za izbrani mesec,
- reservations/receptor [GET] - vrne vse rezervacije za trenutni dan,
- reservations/receptor/start/{id} [POST] - zabeleži čas prevzema ključa za rezervacijo s podanim ID-jem,
- reservations/receptor/end/{id} [POST] - zabeleži čas vrnitve ključa za rezervacijo s podanim ID-jem,
- reservations/message [GET] - vrne vsa sporočila uporabnikov,
- reservations/message [POST] - dodajanje novega sporočila s strani uporabnika.

4.3 Spletna aplikacija

V tem podpoglavju so najprej opisani glavni gradniki Angular aplikacije, nato je prikazan še praktičen primer vsakega gradnika iz naše aplikacije, zatem pa sledi še predstavitev aplikacije.

Struktura Angular aplikacije

Angular aplikacija je sestavljen iz različnih gradnikov.

Modul

Angular je sestavljen iz majhnih modulov, ki predstavljajo določene funkcionalnosti aplikacije. Osnovni moduli Angularja so moduli *NgModule*. Vsaka Angular aplikacija vsebuje vsaj en korenski (glavni) modul, ki omogoča zagonsko nalaganje aplikacije ter pogosto več funkcijskih modulov.

Komponenta

Vsaka Angular aplikacija vsebuje tudi vsaj eno komponento, ki je korenska (glavna) in povezuje druge komponente v hierarhijo z uporabo dokumentnega objektnega modela (DOM). Vsaka komponenta vsebuje razred, ki vsebuje aplikacijske podatke in logiko ter je povezna s HTML predlogo. Predloga predstavlja izgled komponente. Pred opredelitvijo komponente je dekorator @Component(), ki identificira razred neposredno pod njo kot komponento. Dekorator vsebuje metapodatke o komponenti. To so podatki, ki so potrebni, da sistem ve, kakšen je razred v komponenti in kako deluje.

Predloga

Predloga vsebuje HTML elemente z Angular oznakami, ki lahko spremenijo elemente HTML preden se prikažejo. Direktive predlog zagotavljajo programsko logiko in z oznakami povezujejo podatke aplikacije in HTML elemente. Obstajata dve vrsti vezave podatkov.

- Dogodkovne, ki omogočajo, da se aplikacija odziva na uporabniški vnos, tako da posodablja podatke v aplikaciji.

- Vezava podatkov iz aplikaciji v HTML elemente.

Preden je izgled aplikacije prikazan uporabniku, Angular pregleda direktive in na podlagi teh direktiv spremeni HTML elemente. Angular podpira dvosmerno podatkovno vezavo, kjer spreminjanje elementa HTML povzroči, da se spremenijo tudi aplikacijski podatki.

Storitev

Za podatke in programsko logiko, ki niso povezani le z eno samo komponento, in se jih želi uporabiti v večih komponentah, se ustvari storitev. Pred opredelitvijo razreda storitve se nahaja dekorator @Injectable(). Dekorator zagotavlja metapodatke, ki omogočajo, da se storitev vnese v komponento kot odvisnost.

Usmerjevanje

Usmerjevalnik (Router) je modul *NgModule*, ki omogoča storitev za določanje navigacijskih poti po različnih pregledih in straneh aplikacije. Zgrajena je na običajnih navigacijskih konvencijah:

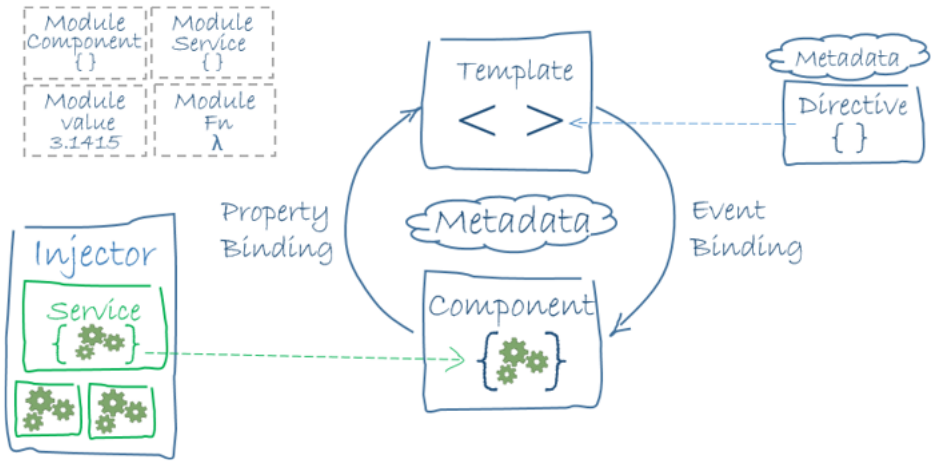
- V naslovno vrstico se vnese URL, brskalnik pa se pomakne na ustrezno spletno stran.
- Ob kliku na povezavo do spletne strani, se brskalnik prestavi na novo spletno stran.
- Ob kliku gumba „Nazaj“ in „Naprej“ se brskalnik premika skozi zgodovino strani.

Pregled Angularja

Slika 4.8 prikazuje, kako so povezani osnovni gradniki Angularja. Komponenta (*Component*) in predloga (*Template*) skupaj predstavljata Angularjev izgled. Dekorator na komponenti doda metapodatke, ki vsebujejo kazalce do predloge (*Template*). Podatkovne vezave (*Event binding* in *Property binding*) na predlogi komponente spremenijo izgled glede na aplikacijske podatke in programsko logiko. Odvisnost (*Injector*) pa doda storitev komponenti npr. *Router* [2].

Primer datotek iz naše aplikacije

V nadaljevanju so predstavljeni gradniki razvite aplikacije Angular.



Slika 4.8: Slika prikazuje, kako so povezani osnovni gradniki Angularja.

Komponenta

Slika 4.9 prikazuje primer komponente RoomList, ki vsebuje metapodatke v @Component dekoratorju. Prav tako vsebuje storitev roomService. Vsebuje tudi funkcije, ki se izvedejo ob nekem dogodku, ki ga povzroči uporabnik.

Predloga

Slika 4.10 prikazuje HTML elemente z Angular oznakami, ki se pred prikazom strani aplikacije pretvorijo v HTML elemente z ustreznimi podatki. Na sliki lahko vidimo tako dogodkovne vezave na HTML elemente „(click)=deleteRoom(room.id)“ kot tudi vezavo podatkov iz aplikacije. Te se pretvorijo v HTML elemente „{{room.code}}“.

Storitev

Slika 4.11 prikazuje primer storitve. Storitev vsebuje meta podatke v @Injectable dekoratorju. Prav tako vsebuje funkcije, ki izvedejo klice na spletno storitev, ki smo jo postavili pred spletno aplikacijo Angular. Funkcija addNewRoom pošlje zahtevek na spletni strežnik s podatki, ki so v telesu in

avtorizacijskim žetonom JWT v glavi. Ta se ob prijavi v aplikacijo shrani v lokalno spremenljivko localStorage('userToken') in je veljaven le petnajst minut po prijavi.

```
import { RoomsService } from '../shared/rooms.service';
import { Component, OnInit } from '@angular/core';
import { Room } from '../shared/room.model';

@Component({
  selector: 'app-rooms-list',
  templateUrl: './rooms-list.component.html',
  styleUrls: ['./rooms-list.component.css']
})
export class RoomsListComponent implements OnInit {
  constructor(public roomsService : RoomsService) { }

  ngOnInit() {
    this.roomsService.getRooms();
  }
  showForEdit(room : Room){
    this.roomsService.selectedRoom =room;
  }
  changeStatus(id : number){
    this.roomsService.changeRoomStatus(id).subscribe(data => {
      this.roomsService.getRooms();
    })
  }
  deleteRoom(id : number){
    this.roomsService.deleteRoom(id).subscribe(data => {
      this.roomsService.getRooms();
    })
  }
}
```

Slika 4.9: Slika prikazuje komponento Angularja.

```
<tbody>
  <tr *ngFor="let room of roomsService.roomsList">
    <td>{{ room.code}}</td>
    <td>{{ room.name}}</td>
    <td>{{ room.washingMachineCount}}</td>
    <td class="brand-logo center">
        <a class="btn-floating btn-small red" (click)="showForEdit(room)">
          <i class="small material-icons">mode_edit</i>
        </a>
    </td>
    <td class="brand-logo center">
      <div >
        <a  class="btn-floating btn-small red" (click)="changeStatus(room.id)">
          <i *ngIf="room.isActive === true" class="small material-icons">check_box</i>
          <i *ngIf="room.isActive === false" class="small material-icons">check_box_outline_blank</i>
        </a>
      </div>
    </td>
    <td class="brand-logo center">
      <a  class="btn btn-medium red" (click)="deleteRoom(room.id)">Izbriši</a>
    </td>
  </tr>
</tbody>
```

Slika 4.10: Slika prikazuje del predloge Angularja.

```
import { Room } from './room.model';
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import 'rxjs/add/operator/map';
import 'rxjs/add/operator/catch';
import 'rxjs/add/observable/throw';

@Injectable({
  providedIn: 'root'
})
export class RoomsService {
  readonly rootUrl = 'http://localhost:63315/api/washingroom/';
  selectedRoom : Room;
  roomsList : Room[];
  constructor(private http: HttpClient) {

  }

  addNewRoom(room : Room){
    let headers = new HttpHeaders({ 'Content-Type': 'application/json',
                                    'Authorization' : 'bearer ' + localStorage.getItem('userToken') });
    let bodyJson = JSON.stringify( room)
    return this.http.post(this.rootUrl, bodyJson, {headers : headers});
  }

  updateRoom(room : Room){
    let headers = new HttpHeaders({ 'Content-Type': 'application/json',
                                    'Authorization' : 'bearer ' + localStorage.getItem('userToken') });
    let bodyJson = JSON.stringify(room)
    return this.http.put(this.rootUrl, bodyJson, {headers : headers});
  }
```

Slika 4.11: Slika prikazuje service Angularja.

Predstavitev aplikacije

V nadaljevanju je predstavljena spletna aplikacija. Vse do zaključka testiranja je spletna aplikacija dostopna na naslovu: https://pralnicakok.azurewebsites.net/

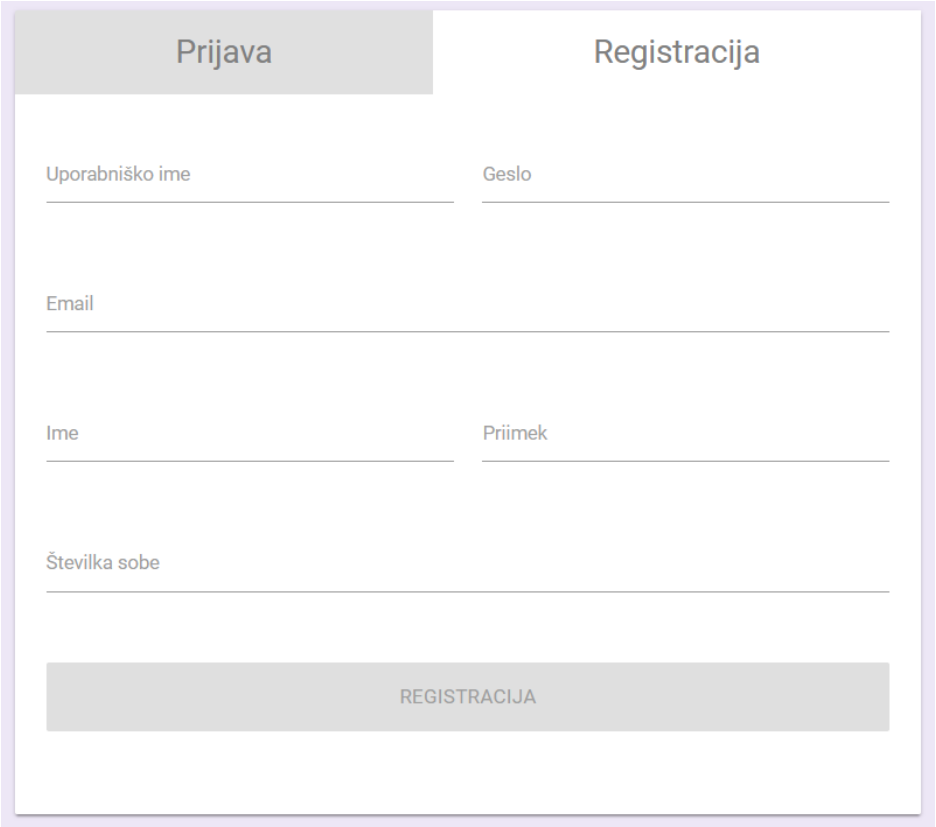
Registracija in prijava

Prijavno okno je začetna stran spletne aplikacije. Vsebuje dva obrazca. To sta obrazec za registracijo uporabnika in obrazec za prijavo že obstoječega uporabnika. Novi uporabniki vnesejo svoje podatke v obrazec za registracijo, ki je prikazan na sliki 4.12, nato pa kliknejo gumb „REGISTRACIJA“. Pri registraciji se preverja, da uporabniško ime in e-pošta še ne obstajata. V primeru, da že obstajata, spletna aplikacija javi napako z napisom „Registracija ni uspešna. Uporabnik z uporabniškim imenom ali e-poštnim naslovom že obstaja“. Pri tem mora biti geslo dolgo najmanj osem znakov. V primeru uspešne registracije dobi uporabnik obvestilo o uspešni registaciji. Spletna aplikacija preusmeri uporabnika na prijavno stran. Pred prvo prijavo mora biti uporabnik odobren s strani administratorja.

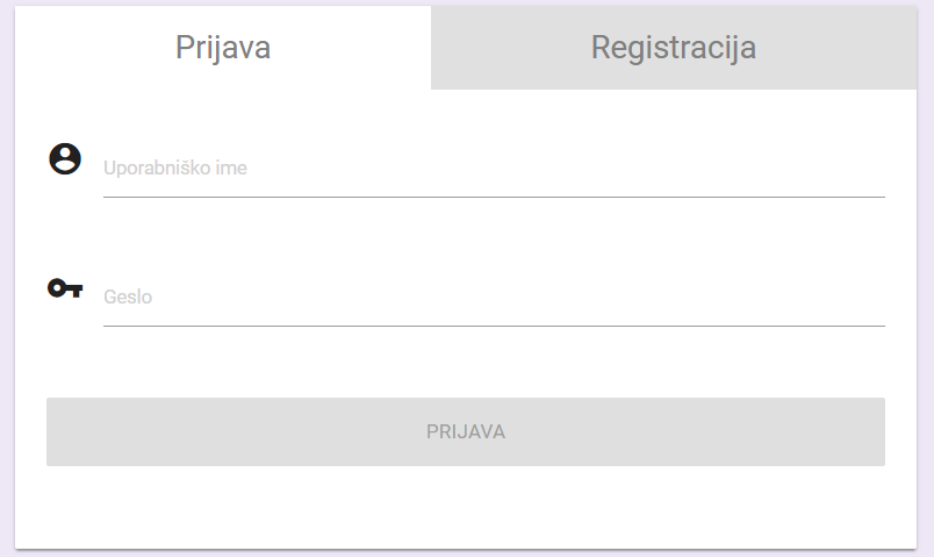
Prijavna stran vsebuje obrazec za prijavo, ki je prikazan na sliki 4.13. Ko uporabnik izpolni obrazec, se ob pritisku na gumb „PRIJAVA“ izvede preverjanje uporabniškega imena in gesla. V primeru napačnega uporabniškega imena in gesla, aplikacija javi napako z napisom „Napačno uporabniško ime ali geslo“. Ob uspešni prijavi pa uporabnika preusmeri na prvo stran, ki je določena glede na skupino uporabnika. V aplikaciji imamo štiri skupine uporabnikov in to so:

- Uporabniki,
- Receptorji,
- Tajništvo,
- Administrator.

V nadaljevanju bomo opisali vsako skupino uporabnikov posebej.



Slika 4.12: Registracijsko okno spletne aplikacije.



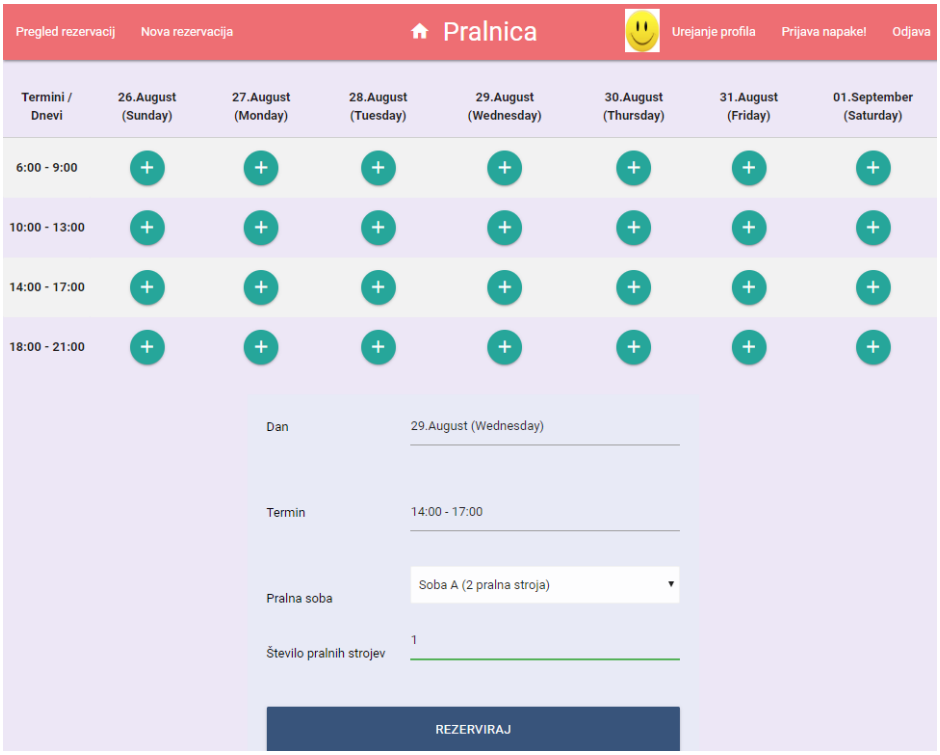
Slika 4.13: Prijavno okno spletne aplikacije.

Uporabniški del

Slika 4.14 prikazuje prvo stran z imenom „Pregled rezervacij“, ki se prikaže uporabniku po prijavi v aplikacijo. Vsebuje navigacijsko vrstico, v kateri imamo gumbe, ki so povezave do drugih strani aplikacije, kjer se nahajajo še druge funkcionalnosti spletne aplikacije. Navigacijska vrstica vsebuje tudi napis „Pralnica“ in prikazno sliko uporabnika ter gumb „Odjavi“, ki poskrbi za odjavo iz spletne aplikacije. Prva stran uporabnika vsebuje dve tabeli z rezervacijami terminov. Leva tabela prikazuje tekoče rezervacije od trenutnega dne dalje, kjer ima uporabnik možnost brisanja rezervacij s pritiskom na gumb „IZBRIŠI“, vendar le za rezervacije, ki bodo v prihodnjih dneh. V desni tabeli pa se nahaja preteklih deset rezervacij terminov.

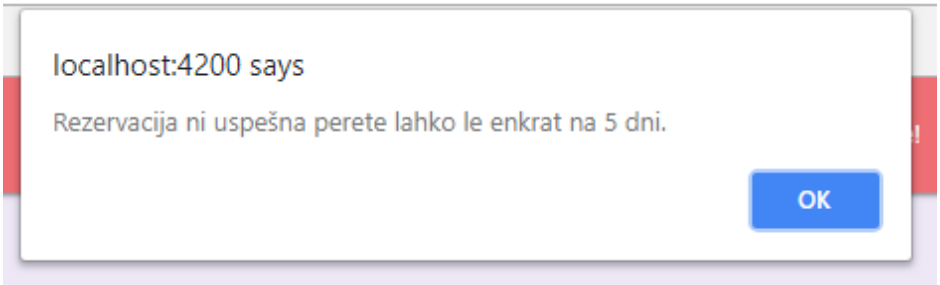
Slika 4.14: Prva stran uporabnika.

Slika 4.15 prikazuje okno, kjer lahko opravimo novo rezervacijo termina za pranje. Najprej v tabeli izberemo željen termin. Nato na podlagi izbranega termina izberemo pralno sobo v spustnem meniju, zatem pa izberemo še število pralnih strojev, ki jih bomo uporabili. Pri izbiri števila pralnih strojev imamo omejitve, in sicer najmanj en stroj in največ toliko strojev, kot jih izbrana pralna soba poseduje. Ob pritisku na gumb „REZERVIRAJ“ se izvede rezervacija termina. Pri rezervaciji imamo omejitve, in sicer da lahko uporabnik pralnico uporablja le enkrat na pet dni. V primeru neuspešne



Slika 4.15: Stran za rezervacijo novega termina pranja.

rezervacije se pojavi opozorilno okno z napisom „Rezervacija ni uspešna. Perete lahko le enkrat na 5 dni.“, kot je prikazano na sliki 4.16. V primeru uspešne rezervacije pa je uporabnik preusmerjen na prvo stran, kjer se v levi tabeli pojavi nova rezervacija.



Slika 4.16: Primer opozorilnega okna.

Slika 4.17 prikazuje stran za urejanje profila, kjer lahko uporabnik spremeni prikazno sliko ali geslo. Pri spremembi prikazne slike mora uporabnik s klikom na gumb „Choose File“ izbrati sliko. Ko izbere sliko, pritisne na gumb „SPREMENI SLIKO“. Ob uspešni spremembi slike se spremeni slika na vrhu aplikacije.

Pri spremembi gesla mora uporabnik v obrazec najprej vpisati trenutno geslo, nato pa v naslednji dve polji obakrat isto novo geslo. S klikom na gumb „SPREMENI GESLO“, se uporabniku spremeni geslo. V primeru vpisa napačnega trenutnega gesla, se geslo ne spremeni in pokaže se opozorilno okno, ki uporabnika obvesti, da geslo ni bilo spremenjeno. V primeru uspešno spremenjenega gesla, se pojavi opozorilno okno, ki obvesti uporabnika, da je bilo geslo uspešno spremenjeno.

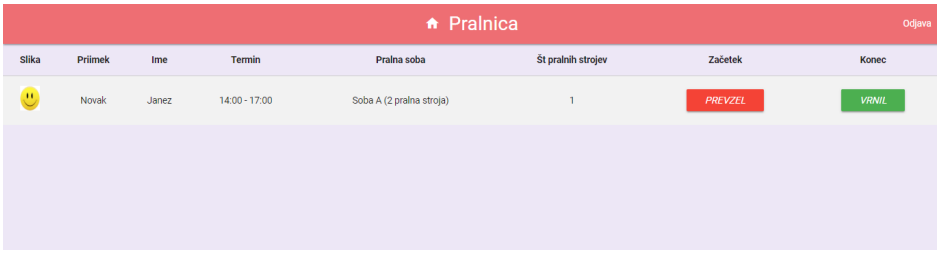
Slika 4.17: Stran za spreminjanje uporabniškega gesla in prikazne slike.

Slika 4.18 prikazuje okno, kjer lahko uporabnik prijavi napako, ki se je zgodila med uporabo spletne aplikacije. Vsebuje polje za vnos vsebine, kamor uporabnik vpiše sporočilo in pa gumb „PRIJAVA NAPAKE“, s katerim pošlje sporočilo administratorju. Ob uspešno poslanem sporočilu se zgodi preusmeritev na prvo okno in prikaže se obvestilno okno z napisom „Uspešno prijavljena napaka“.

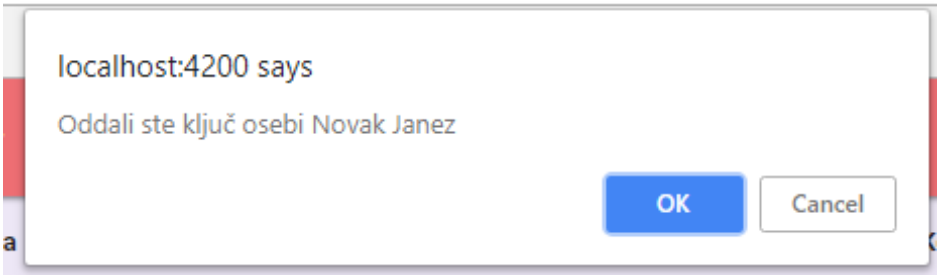
Slika 4.18: Stran za prijavo napake.

Receptorski del

Slika 4.19 prikazuje prvo stran, ki se prikaže receptorju po prijavi v spletno aplikacijo in vsebuje vse rezervacije terminov za tekoči dan. Vsebuje navigacijsko vrstico, kjer se nahaja le gumb za odjavo iz spletne aplikacije. Rezervacije so prikazane v tabeli s polji: Slika, Priimek, Ime, Termin, Pralna soba, Št. pralnih strojev in gumba z ukazom „PREVZEL“ in „VRNIL“. Ko receptor preda ključ uporabniku označenega termina, pritisne na gumb „PREVZEL“, s tem se zabeleži čas, ko je uporabnik začel prati. Obenem se prikaže potrditveno okno z napisom, ki ga obvesti o oddaji ključa. Ob vrnitvi ključa mora receptor pritisniti gumb „VRNIL“, da zabeleži čas vrnitve ključa pralne sobe. Prav tako pa se prikaže potrditveno okno, ki ga obvesti, da je zaključil akcijo vračila. Slika 4.20 prikazuje primer potrditvenega okna.



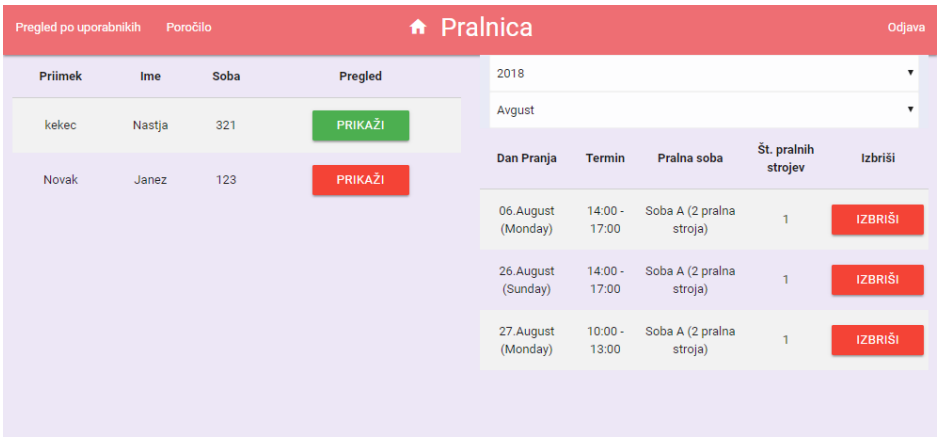
Slika 4.19: Prikaz receptorske strani.



Slika 4.20: Primer potrditvenega okna.

Tajniški del

Slika 4.21 prikazuje prvo stran, imenovano „Pregled po uporabnikih“, ki se prikaže tajništvu. Prav tako vsebuje navigacijsko vrstico, kjer so gumbi s povezavami do drugih strani in gumb „Odjava“. Prva stran prikazuje pregled rezervacij terminov za posameznega uporabnika. Ob levi strani se nahaja tabela s podatki uporabnikov. Na desni strani imamo dva spustna menija. V prvem izberemo leto, v drugem pa mesec nad katerim želimo pregled. S pritiskom na gumb „IZBRIŠI“, lahko izbrišemo posamezno rezervacijo. Ob izbrisu se nam pojavi potrditveno okno, ki nas vpraša, če smo prepričani, da želimo izbrisati to rezervacijo.



Slika 4.21: Stran za pregled po uporabnikih.

Slika 4.22 prikazuje stran imenovano „Poročilo“, ki vsebuje dva spustna menija. V prvem izberemo leto, v drugem pa mesec. Nato se nam prikažejo števila vseh uporabljenih pralnih strojev in število vseh zamudnin za vse uporabnike za izbrano leto in mesec. Z gumbom „Prenesi PDF“ lahko prenesemo podatke v datoteko. Slika 4.23 prikazuje primer PDF datoteke.

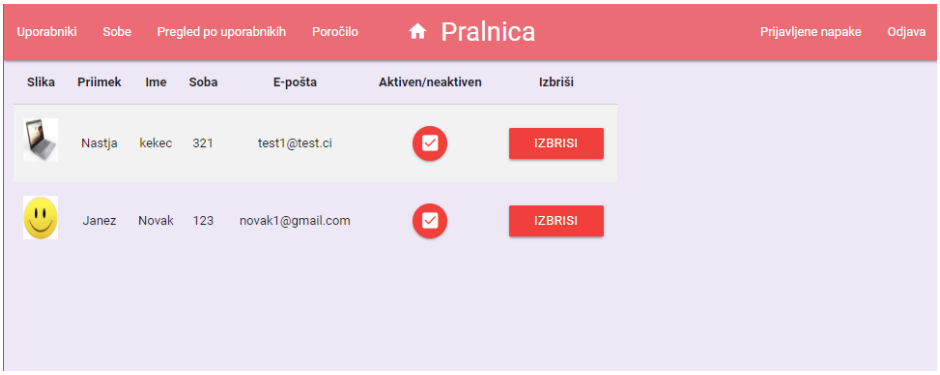
Slika 4.22: Stran za pregled po mesecih.

Slika 4.23: Primer PDF mesečnega poročila.

Administratorski del

Slika 4.24 prikazuje prvo stran, ki se imenuje „Uporabniki“ in se prikaže administratorju. Prav tako vsebuje navigacijsko vrstico, kjer ima gumbe s povezavami do drugih strani in gumb „Odjava“. Stran prikazuje uporabnike aplikacije. Na strani se nahaja tabela z uporabniki. Tabela vsebuje naslednja polja: Slika, Priimek, Ime, Soba, E-pošta, Aktiven/neaktiven in Izbriši. Vsa polja razen Aktiven/neaktiven in Izbriši so samo prikazna. Administrator mora za vsakega uporabnika posebej označiti potrditveno polje, to je „Aktiven/neaktiven“, da se uporabnik lahko prijavi v aplikacijo. V primeru da je polje neoznačeno, se uporabnik ne more prijaviti v aplikacijo. Gumb „IZBRIŠI“ izbriše uporabnika iz aplikacije.

Slika 4.25 prikazuje stran sobe. Ob levi strani se nahaja obrazec za dodajanje nove pralne sobe ali za spreminjanje obstoječih pralnih sob. Ko izpolnimo obrazec, se lahko odločimo za preklic in pritisnemo gumb „Reset“. Z gumbom „Dodaj/spremeni“ pa lahko dodamo novo sobo ali spremenimo obstoječo. Desno se nahaja tabela obstoječih sob. Ta vsebuje polja: Koda sobe, Opis sobe, Št. pralnih strojev, ki se nahajajo v njej. Vsa ta polja so samo prikazna. Ob pritisku na gumb „Uredi“ se nam na levi strani v obrazec preslika pralna soba, katero lahko nato urejamo. Potrditveno polje „Aktiven/neaktiven“ ima enako funkcionalnost kot pri uporabnikih, prav tako tudi gumb „IZBRIŠI“.

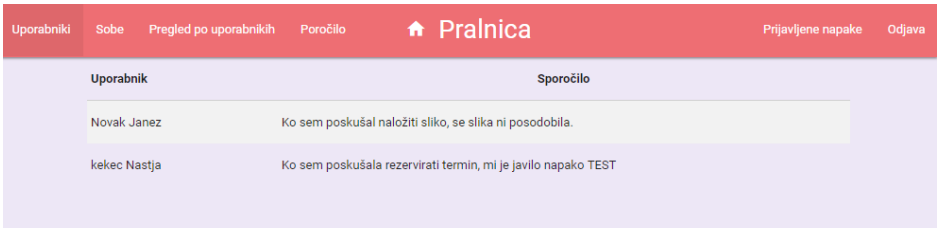


Slika 4.24: Stran uporabniki, ki prikazuje uporabnike.

Slika 4.25: Stran sobe, ki prikazuje pralne sobe.

Slika 4.26 prikazuje stran s prijavljenimi napakami, ki so jih prijavili uporabniki. Vsebuje tabelo s prikazanimi polji uporabniških sporočil. Prikazana so v zaporedju od najnovejšega do najstarejšega.

Poleg zgoraj naštetih strani administrator vidi tudi strani, ki so prikazane v tajniškem delu.



Slika 4.26: Stran za pregled prijavljenih napak.

Poglavje 5

Testiranje

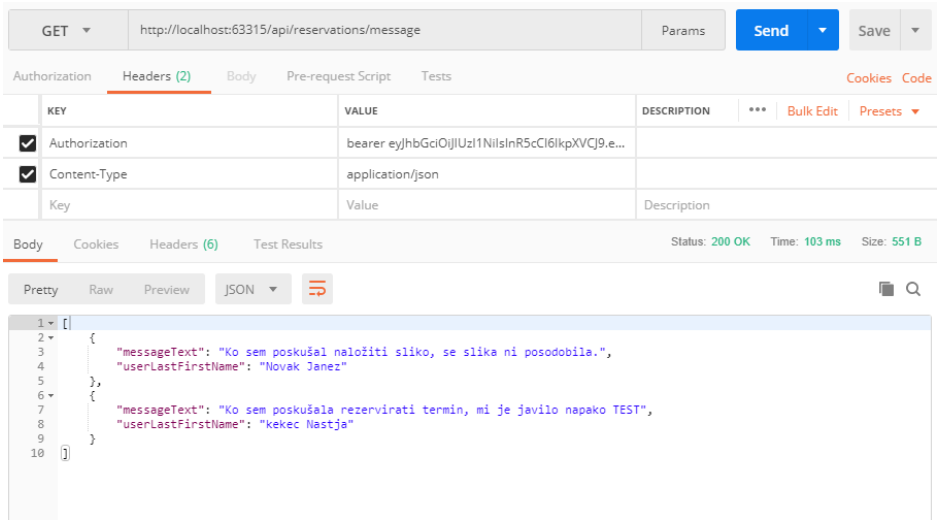
V sklopu testiranja smo najprej lokalno preizkusili delovanje spletne storitve. Zatem smo podatkovno bazo, spletno storitev in spletno aplikacijo prenesli v oblak in jo predali na voljo uporabnikom za testno uporabo.

5.1 Testiranje spletne storitve

Storitev smo najprej testirali lokalno s programom Postman, ki omogoča klicanje dostopnih točk dane spletne storitve. Testirali smo vsako dostopno točko posebej ter spremljali podatke v podatkovni bazi, če so se ti pravilno posodobili. Slika 5.1 prikazuje primer uspešnega klica spletne storitve s programom Postman, kjer v glavi zahtevka GET pošljemo JWT žeton pod poljem Authorization, ob klicu dostopne točke pa nam ta vrne podatke v JSON obliki.

5.2 Postavitev aplikacije v oblaku

Za tesitranje spletne aplikacije smo programsko rešitev postavili v oblaku na Microsofotovi storitvi Azure. Najprej smo postavili bazo in jo s pomočjo programa Microsoft Data Migration Assistant (DMA), ki omogoča prenos podatkovne baze iz enega strežnika na drugega, prenesli na Azure. Nato smo v



Slika 5.1: Primer klica spletne storitve s pomočjo programa Postman.

oblak postavili spletno storitev, zatem pa še spletno aplikacijo. Spletno aplikacijo smo nato predali uporabnikom za testno uporabo. Do konca testiranja bo spletna aplikacija dostopna na naslovu https://pralnicakok.azurewebsites.net/.

Poglavje 6

Sklepne ugotovitve

Cilj dilpomske naloge je bil izdelava programske rešitve za pralnico, ki bo omogočala uporabniku rezervacijo terminov za pranje perila. Tekom diplomske naloge smo implementirali vse načrtovane funkcionalnosti spletne aplikacije, nekaj pa smo jih dodali med samim razvojem. Razvoj programske rešitve je trenutno v stanju testiranja s strani uporabnikov. Ko se bo testiranje končalo, bomo izvedli še izvedbeno fazo projekta, v katerem bomo namestili programsko opremo na produkcijski strežnik. Spletna aplikacija omogoča uporabniku rezervacijo termina za pranje, prijavo napake in pregled vseh rezervacij. Prav tako omogoča tajništvu pregled nad uporabo pralnice za določeno obdobje ali pa pregled po posameznih uporabnikih. Receptorjem omogoča dodajanje zapisa, da je uporabnik prevzel ključe za neko pralno sobo oziroma jih je vrnil. Administratorju poleg vseh tajniških funkcionalnosti, omogoča tudi konfiguracijo spletne aplikacije, urejanje uporabnikov in brisanje uporabnikov, dodajanje in urejanje ter brisanje pralnih sob, ter pregled prijavljenih napak. Celotna aplikacija je izdelana pretežno v slovenskem jeziku. Določeni deli aplikacije vsebujejo še nekaj angleških besedil („Choose file“, „RESET“), kar je bilo odkrito med testiranjem in bo pred namestitvijo na produkcijski strežnik tudi popravljeno.

6.1 Nadaljnji razvoj

Zaradi vse večje informatizacije na vseh področjih, prihaja do vse večje potrebe po informacijskih sistemih na različnih področjih. Naša programska rešitev je šele osnovna različica, ki bo potrebovala še nadgradnje, glede na uporabniški odziv. Prav tako bi bilo smiselno v nadaljevanju aplikacijo dopolniti in približati uporabniku z naslednjimi nadgradnjami:

- Spletna aplikacija bi lahko bila v večih jezikih in ne samo v slovenščini, saj v domovih prebivajo tudi stanovalci, ki ne razumejo slovenskega jezika.
- Razvoj mobilne aplikacije za operacijska sistema Android in iOS. Spletno storitev smo razvili tako, da bi do podatkov lahko dostopali tudi iz mobilne aplikacije preko interneta, v prvi fazi pa bi zadostovala tudi nadgradnja aplikacije v odzivno spletno aplikacijo z uporabo Bootstrap predlog.
- Dodan avtomatiziran izbor pralne sobe, pri katerem bi sistem ob izbiri števila pralnih strojev samodejno preveril, katera pralna soba z najmanjšim številom pralnih strojev je še prosta, saj lahko pralne sobe vsebujejo različno število pralnih strojev.
- Dodali bi lahko urejanje uporabniških lastnosti kot so: ime, priimek, e-pošta in številka sobe.
- Prav tako bi lahko dodali funkcionalnost, ki bi uporabniku na dan koriščenja termina poslala e-poštno sporočilo kot opomnik.
- Popolna avtomatizacija, kjer bi vsak uporabnik imel elektronski ključ, pralnica pa bi vsebovala elektronsko klučavnico. Tako bi razbremenili receptorja in ga izključili iz spletne aplikacije.

Literatura

[1] Angular. Dosegljivo: https://angular.io/docs. ["Dostopano: 22.8.2018"].

[2] Angular Architecture. Dosegljivo: https://angular.io/guide/architecture. ["Dostopano: 22.8.2018"].

[3] C#. Dosegljivo: http://www.webcitation.org/6h8p6fWJk/. ["Dostopano: 22.8.2018"].

[4] CSS. Dosegljivo: https://www.w3schools.com/css/css_intro.asp. ["Dostopano: 22.8.2018"].

[5] Introduction to ASP.NET Core. Dosegljivo: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-2.1/. ["Dostopano: 22.8.2018"].

[6] JavaScript. Dosegljivo: https://www.w3schools.com/js/js_intro.asp. ["Dostopano: 22.8.2018"].

[7] JSON. Dosegljivo: https://www.json.org/. ["Dostopano: 22.8.2018"].

[8] JWT. Dosegljivo: https://jwt.io/introduction/. ["Dostopano: 26.8.2018"].

[9] Material Design. Dosegljivo: https://materializecss.com/about.html. ["Dostopano: 22.8.2018"].

[10] Mobilna aplikacija za ogled razstave, diplomska naloga. Dosegljivo: http://eprints.fri.uni-lj.si/3920/1/63130043-URBAN_FATUR-Mobilna_aplikacija_za_ogled_razstave_v_galeriji-1.pdf. ["Dostopano: 22.8.2018"].

[11] RESTful Web Services. Dosegljivo: https://www.tutorialspoint.com/restful/restful_introduction.htm/. ["Dostopano: 22.8.2018"].

[12] Slapovni model. Dosegljivo: https://ucilnica1516.fri.uni-lj.si/pluginfile.php/792/course/section/1238/p1.pdf. ["Dostopano: 22.8.2018"].

[13] SQL Server Management Studio. Dosegljivo: https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-2017/. ["Dostopano: 22.8.2018"].

[14] Sql syntax. Dosegljivo: https://www.w3schools.com/sql/sql_syntax.asp/. ["Dostopano: 22.8.2018"].

[15] Sql Tutorial. https://www.w3schools.com/sql/default.asp/. ["Dostopano: 22.8.2018"].

[16] Visual Studio. Dosegljivo: https://docs.microsoft.com/en-us/visualstudio/ide/visual-studio-ide/. ["Dostopano: 22.8.2018"].

[17] Visual Studio Code. Dosegljivo: https://code.visualstudio.com/docs/editor/whyvscode. ["Dostopano: 22.8.2018"].

[18] Gavin Bierman, Martín Abadi, and Mads Torgersen. Understanding typescript. In *European Conference on Object-Oriented Programming*, pages 257–281. Springer, 2014.

[19] Michael Jones, John Bradley, and Nat Sakimura. Json web token (jwt). Technical report, 2015.

[20] Chuck Musciano and Bill Kennedy. *HTML*. Number BOOK. O'reilly, 1997.