Interaktivni web kalendar

Begić, Filip

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

INTERAKTIVNI WEB KALENDAR

Diplomski rad

Filip Begić

Osijek, 2020.

SADRŽAJ

1. UVOD

Napretkom tehnologije i boljitkom života općenito, potreba za organiziranjem čovjekovog vremena sve je veća. Potrebno je uklopiti različite obaveze jedne s drugima i pronaći ravnotežu između poslovnog i privatnog planiranja. Često je teško odrediti važniju obavezu među onima iste vrste jer se obaveze mogu izvršavati u isto vrijeme ili mogu zahtijevati različita vremena izvršavanja. Usklađivanje više obaveza u isto vrijeme podrazumijeva da su one različitih važnosti, a treba uzeti u obzir i da vrijeme trajanja obaveza može biti različito.

Obaveze često ne budu izvršene na vrijeme ili se izvrše ranije od planiranog. Prilikom planiranja projekata poput web kalendara „.o“ često u početku nisu poznati svi parametri, pa je rok za završetak projekta teško procijeniti.

Koncept kalendara i planera za praćenje obaveza otprije je poznat. Planeri podrazumijevaju vremenske crte, rokovnike i bilo koji oblik bilježenja obaveza s ciljem lakše organizacije.

Web kalendari su programske izvedbe koje nude rješenja za pametnu organizaciju vremena i obaveza korištenjem informacijsko-komunikacijskih tehnologija. Korisnici web kalendara imaju svoje račune, tako da web kalendaru mogu pristupiti s različitih računala. Događaji unutar web kalendara spremljeni su u bazi podataka i, osim ako se radi o timskom kalendaru, vezani su samo uz jednog korisnika. Timski kalendari omogućuju podjelu obaveza i vremena između članova nekog tima. Osim navedenog, današnji moderni web kalendari korisnike privlače intuitivnim korisničkim sučeljem s mnoštvom mogućnosti: dinamičkim obavijestima, mogućnošću uređivanja zadataka, različitim vrstama prikaza itd.

U sklopu ovog rada potrebno je implementirati web kalendar koji će korisniku omogućiti interaktivni pristup svojim obavezama i olakšati mu organizaciju. Svaki korisnik treba imati vlastiti račun u sklopu kojega može izraditi neograničen broj obaveza kojima samo on može upravljati. Uz to, korisnik mora imati mogućnost biti obaviješten o početku bilo kojeg zadatka. Osim obavijesti, korisnik mora imati mogućnost pristupu različitim statistikama zadataka, što služi kao sredstvo dodatne motivacije korisnika.

Kao odgovor na navedene zahtjeve izrađen je web kalendar „.o“ . Programska izvedba tog web kalendara omogućuje izradu korisničkih računa i unutar svakoga korisnik može napraviti vlastiti skup obaveza. Korisnik obavezama može pristupiti u bilo kojem trenutku i može ih uređivati. Među korisničkim postavkama se također nalazi mogućnost obavještavanja korisnika o početku zadataka. Obavijesti su razdvojene na obavijesti e-pošte i obavijesti vidljive u korisničkom sučelju.

Korisnik u bilo kojem trenutku može pristupiti statistici zadataka, koja je dostupna prema dva kriterija: obavljenosti i duljini trajanja.

Ostatak rada organiziran je na sljedeći način: poglavlje 2. daje pregled postojećih izvedbi web kalendara, dok su u poglavlju 3. opisani zahtjevi na model interaktivnog web kalendara. Implementacija interaktivnog web kalendara i vrednovanje istog objašnjeni su u poglavlju 4. Zaključak je dan u poglavlju 5.

2. PREGLED POSTOJEĆIH IZVEDBI WEB KALENDARA

Ljudi su organizirali vlastite obaveze na različite načine kroz povijest, ovisno o razdoblju, i uglavnom su to činili pomoću neke vrste kalendara. U 18. stoljeću se pojavljuju i drugi načini bilježenja, više vrsta istih itd. Od pojave prvih planera do danas prošlo je puno vremena, ali cilj je i dalje bolja organizacija vremena. U današnje vrijeme obaveze se zbog količine sve teže organiziraju, a planer ili kalendar je postao dio svakodnevice i trebao bi biti uvijek dostupan. Iz tog je razloga danas poseban naglasak na web kalendarima i sličnim izvedbama koji su lako dostupni s obzirom da je pristup tehnologiji olakšan. Konkretno, web kalendari obaveze prikazuju na više načina i mogu imati različitu namjenu. U modernim tvrtkama su, primjerice, sve popularnije izvedbe web kalendara za timsku organizaciju.

U nastavku poglavlja bit će opisana povijest, vrste kalendara i planera s primjerima te usporedba istih. Na taj način će se definirati kontekst unutar kojega će se predstaviti zahtjevi na današnje moderne web kalendare i na osnovu tih zahtjeva biti predstavljena vlastita izvedba web kalendara..

2.1. Organizacija vremena kroz povijest

Organizacija vremena problem je koji javlja već od mlađeg kamenog doba. Prvi problem bio je planiranje vremena za sadnju i žetvu, što su događaji usko vezani uz točno određena godišnja doba. Ljudi su, među ostalim, bilježili prolazak sunca i mjeseca kroz fiksne točke i tako primjećivali ponavljajuće uzorke, a onda pravili kalendare. Neki od tih kalendara bili su:

- solarni,
- lunarni,
- babilonski,
- rimski.

Nešto kasnije pojavio se julijanski kalendar, preteča gregorijanskog kalendara koji se koristi i danas, a više o kalendarima u [1] Napretkom civilizacije bilo je potrebno bilježiti i planirati sve veći broj događaja. Događaji su bilježeni po zemlji, klesani u kamen, pisani na papir. Pisani su na zidne kalendare, u rokovnike, u džepne planere itd. Pojavom računala ostvareni su prvi digitalni kalendari, a pojavom interneta web kalendari i planeri.

2.2. Web kalendari

Web kalendari su online programske izvedbe za organizaciju vremena i obaveza. Glavna karakteristika koja ih odvaja od ostalih je da im se može pristupiti s bilo kojeg računala s pristupom internetu. Namijenjeni su za više korisnika, a svakome je preporučeno imati vlastiti račun.

Uglavnom podržavaju više jezika i više formata zapisa vremena i datuma. Imaju i mogućnost prilagodbe korisniku promjenom izgleda, funkcionalnosti, itd. Sve više je kalendara namijenjenih timskom radu gdje korisnici članovi jedne grupe imaju uvid u obaveze drugih korisnika iste grupe. Više o svojstvima web kalendara, klasifikaciji i usporedbi istih uz pridružene primjere u nastavku poglavlja.

2.3. Klasifikacija i usporedba današnjih web kalendara

Web kalendari mogu se podijeliti prema raznim svojstvima, a neki su:

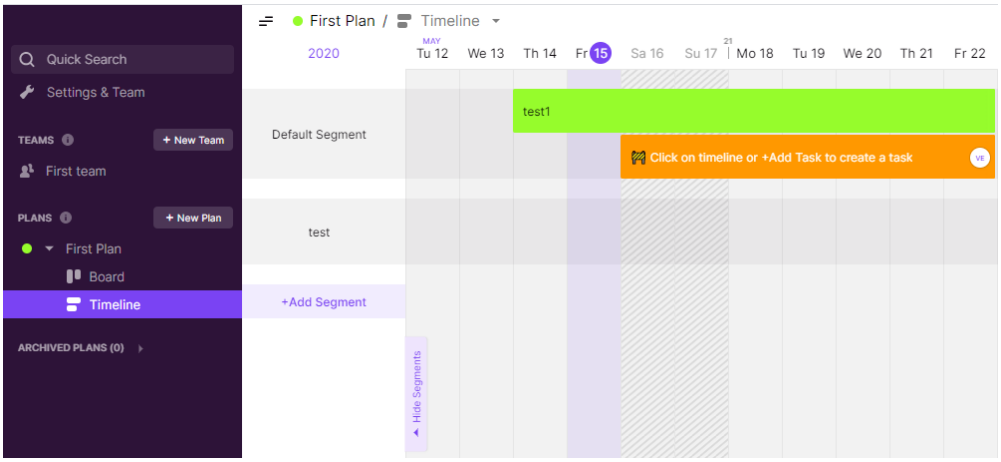
- namjena,
- prikaz,
- opseg,
- dostupnost.

Podjela po namjeni podrazumijeva privatne i poslovne web kalendare. Isti se prema prikazu dijele na one koji događaje smještaju u mjesečni, tjedni ili dnevni prikaz, kao i smještanje na druge razne vremenske crte (*engl. timeline*). Kalendari mogu služiti za organizaciju jedne osobe ili timsku organizaciju što predstavlja podjelu po opsegu. Isti mogu biti dostupni besplatno i mogu zahtijevati plaćanje. Većina web kalendara sadrži više svojstva pa su višenamjenski i sadrže nekoliko ili sve prikaze ili opseg. Također, neke funkcionalnosti mogu biti besplatne, a neke se plaćaju, međutim sve više programskih izvedbi koristi model parcijalnog plaćanja.

Nešto rjeđi od ostalih kalendara su oni s prikazom obaveza u obliku vremenskih crta. Neki od takvih kalendara su:

- Preceden,
- iSpring Suite,
- Sutori,
- Toggl,
- Visme.

Od takvih kalendara ističe se Toggl. Ostali web kalendari su uglavnom uže vezani uz poslovno planiranje. Toggl kalendar, [2], nudi prikaz po mjesecu u obliku vremenskih crta s naznakom kalendarskog prikaza, što je vidljivo i na slici 2.1. Namjena mu je mješovita pa je lako organizirati privatne uz poslovne obaveze, a koristi model plaćanja dijela usluga. Za razliku od ostalih navedenih kalendara koji podržavaju vremenske crte, podržava timsku organizaciju.

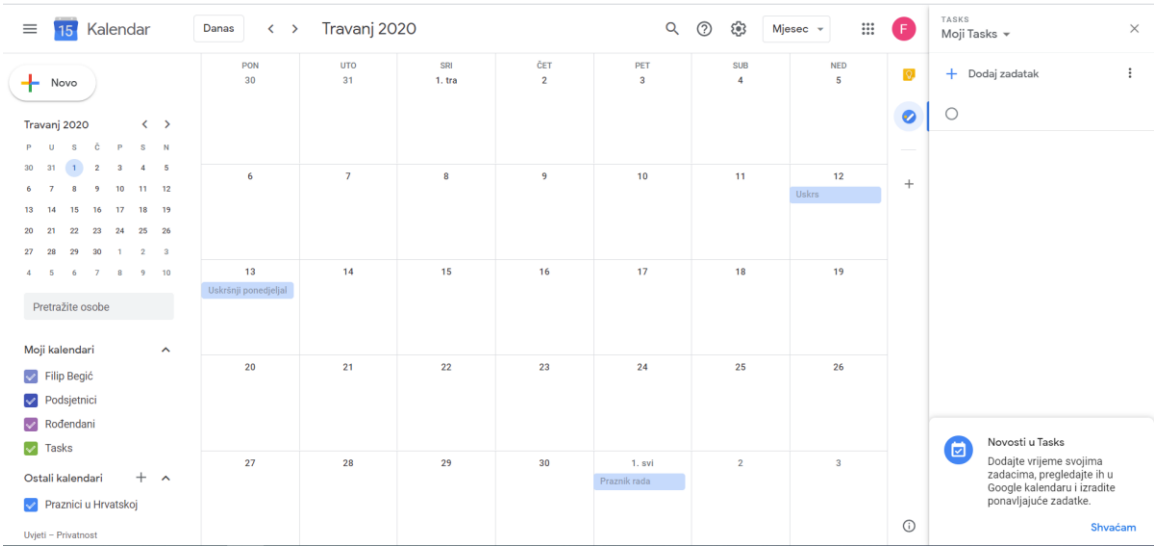


Slika 2.1. Toggl kalendar.

Potpuno besplatni web kalendari sve su češći, a uglavnom dolaze u paketu neke veće usluge. Neki od takvih su:

- Google kalendar,
- Yahoo kalendar,
- Lightning kalendar,
- Posteo.de,
- Mailbox.org.

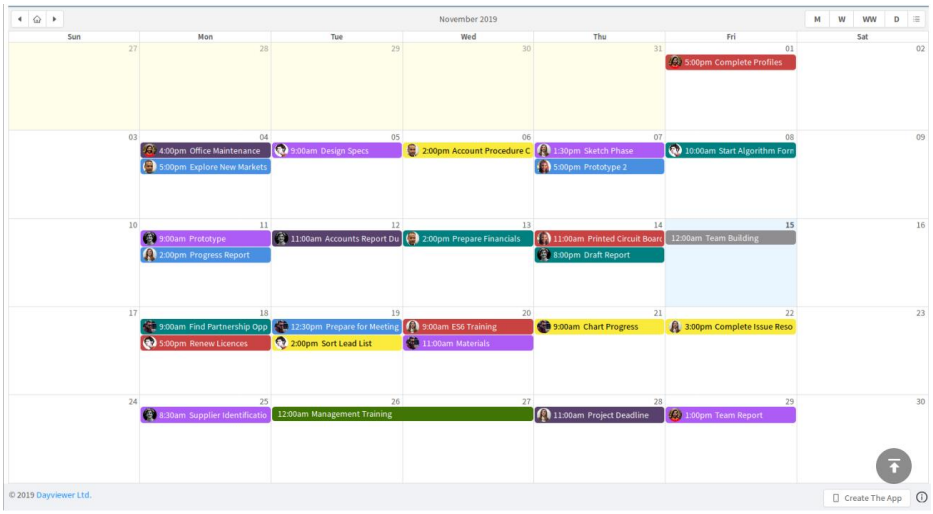
Google kalendar, [3], nudi prikaze svih oblika, a namijenjen je za osobne i poslovne potrebe. Izdvojen je od ostalih prvenstveno zbog puno veće korištenosti i izgleda sučelja. Služi za organizaciju jedne osobe i potpuno je besplatan, ali zahtijeva Google račun, no to zahtijevaju i ostali prethodno navedeni kalendari. Mjesečni prikaz kalendara (Slika 2.2.) kao i dnevni bit će polazna točka pri kreiranju web kalendara „.o“ čiji model će biti dan u sljedećem poglavlju.



Slika 2.2. Google kalendar.

Svaka ozbiljnija tvrtka ima neki način timske organizacije koji je nerijetko u obliku web kalendara. Primjeri takvih kalendara su:

- Trello,
- Dayviewer,
- Airtable,
- Dapulse,
- Teams planer.



Slika 2.3. Dayviewer kalendar.

Izdvaja se Dayviewer kalendar, [4], izgleda sučelja sa slike 2.3. Poput Google kalendara nudi više vrsta prikaza. Međutim nudi i timsku organizaciju, ali ne nudi mogućnost obavještavanja korisnika o početku događaja, zbog čega je i izdvojen. Ostala pružaju barem jedan način obavještavanja korisnika o timskim zadacima. Taj nedostatak dijelom je nadomješten plaćanjem dodatnih opcija

koje su u vidu većeg izbora prikaza, kontrolnih ploča i proizvoljnih oznaka zadataka kako bi bili uočljiviji. Korisnik ima mogućnost uređivanja računa korištenjem postavki koje se nalaze u korisničkom sučelju.

Mlađim korisnicima koji imaju potrebu planiranja obaveza izgled sučelja web kalendara jedna je od bitnijih stavki. Neki od kalendara s mogućnosti promjene sučelja prema željama korisnika su:

- Canva planer,
- iSpring Suite,
- Sutori,
- Toggl.

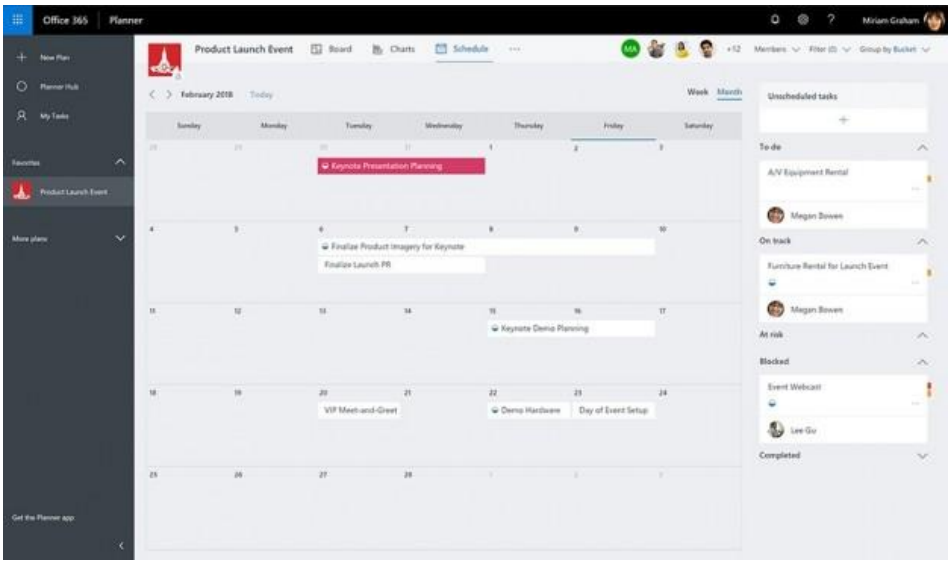
Canva planer, [5], je kalendar s manjim brojem funkcionalnosti u okviru planiranja. Podržava više vrsta prikaza i potpuno je promjenjivog sučelja, zbog čega je i izdvojen. Korisnik izabire između nekoliko predložaka izgleda i pomiče elemente unutar dozvoljenog područja. Glavni nedostatak je nedostatak korisničkih obavijesti – korisnik mora sam pregledati kalendar kako bi vidio kada neki događaj počinje. Isti nedostatak primijećen je u više web kalendara, a smatra se bitnom stavkom.

Najčešće zamijećen nedostatak web kalendara je u okviru obavijest. Jedni od poznatijih kalendara koji nude razne oblike obavještavanja korisnika su:

- Microsoft planer,
- Google kalendar,
- Webplanner,
- Any.do,
- The Muse.

Među navedenima izdvaja se Microsoft planer. Sustav obavještavanja korisnika istog na daleko je većoj razini od ostalih. Microsoft planer nudi obavijesti unutar korisničkog sučelja, putem e-pošte i unutar operacijskog sustava Windows ako to korisnik želi. Ostali navedeni kalendari obavijesti nude uglavnom putem e-pošte, uz iznimku Google kalendara.

Microsoft planer (Slika 2.4.) jedan je od poznatijih Microsoftovih kalendara čija je glavna namjena timska organizacija. Nudi pregled događaja po mjesecu i tjednu, a ima i listu događaja u obliku vremenske crte. Korisnik je obaviješten o počecima događaja putem obavijesti. Također ima i prikaz statistike uspješno odrađenih događaja i zadataka. Obje funkcionalnosti prisutne su u web kalendaru „.o“, a bit će opisani u sljedećim poglavljima. Više o Microsoft kalendaru na [6].



Slika 2.4. Microsoft planer.

Web kalendar izrađena u okviru ovog diplomskog rada naziva „.o“ je hibrid navedenih izvedbi web kalendara. Kao što je navedeno i za Microsoft planer, ima mogućnost obavijesti unutar sučelja, ali i putem emaila za slučaj da na uređaju web kalendar nije otvoren. Microsoft planer služi kao uzor za statistiku događaja web kalendara „.o“. Prikazi događaja i zadataka u svakoj od navedenih izvedbi web kalendara su slični pa tako i u web kalendaru „.o“, a čine ga mjesečni prikaz, dnevni prikaz te popis koji je u okviru dodatnih mogućnosti. Korisnik ima mogućnost upravljanja računom kao kod sa slike 2.3., uz mogućnost upravljanja obavijestima. Svi zahtjevi i svojstva navedeni su u poglavlju 3., a izvedbe i implementacije istih u poglavlju 4.

Na slici 2.5. uspoređeni su predstavnici skupina web kalendara prema važnijim osnovnim svojstvima. Predstavnici su već navedeni i detaljno pojašnjeni u ovom potpoglavlju. Navedena su samo osnovna svojstva jer većinu ostalih svojstava kalendari dijele.

Tablica 2.1. Usporedba predstavnika kalendara.

	Toggl kalendar	**Google kalendar**	**Dayviewer kalendar**	**Canva planer**	**Microsoft planer**
Mjesečni prikaz	DA	DA	DA	DA	DA
Tjedni prikaz	NE	DA	DA	DA	DA
Dnevni prikaz	DA	DA	DA	DA	DA
Vremenska crta	DA	NE	NE	DA	NE
Timska organizacija	DA	DA	DA	NE	DA
Obavijesti	NE	DA	NE	NE	DA
Statistika	NE	NE	NE	NE	DA
Potpuno besplatan	NE	DA	NE	NE	DA

3. MODEL INTERAKTIVNOG WEB KALENDARA

Zadatak ovog diplomskog rada bio je napraviti web kalendar koji će olakšati organiziranje obaveza i događaja s mogućnostima koje su zadane u potpoglavlju 3.1. Web kalendar treba primarno biti namijenjen za osobne potrebe korisnika.

Oslanjajući se na postojeća izvedbe web kalendara ističe se potreba za različite mogućnosti prikaza zadataka, ali i mogućnost uređivanja zadataka u svakom trenutku. Zbog motivacije korisnika treba postojati prikaz statistike riješenosti zadataka. Korisnik mora moći imati vlastiti račun sa zadacima o kojima mora biti obaviješten putem e-pošte i unutar korisničkog sučelja web kalendara.

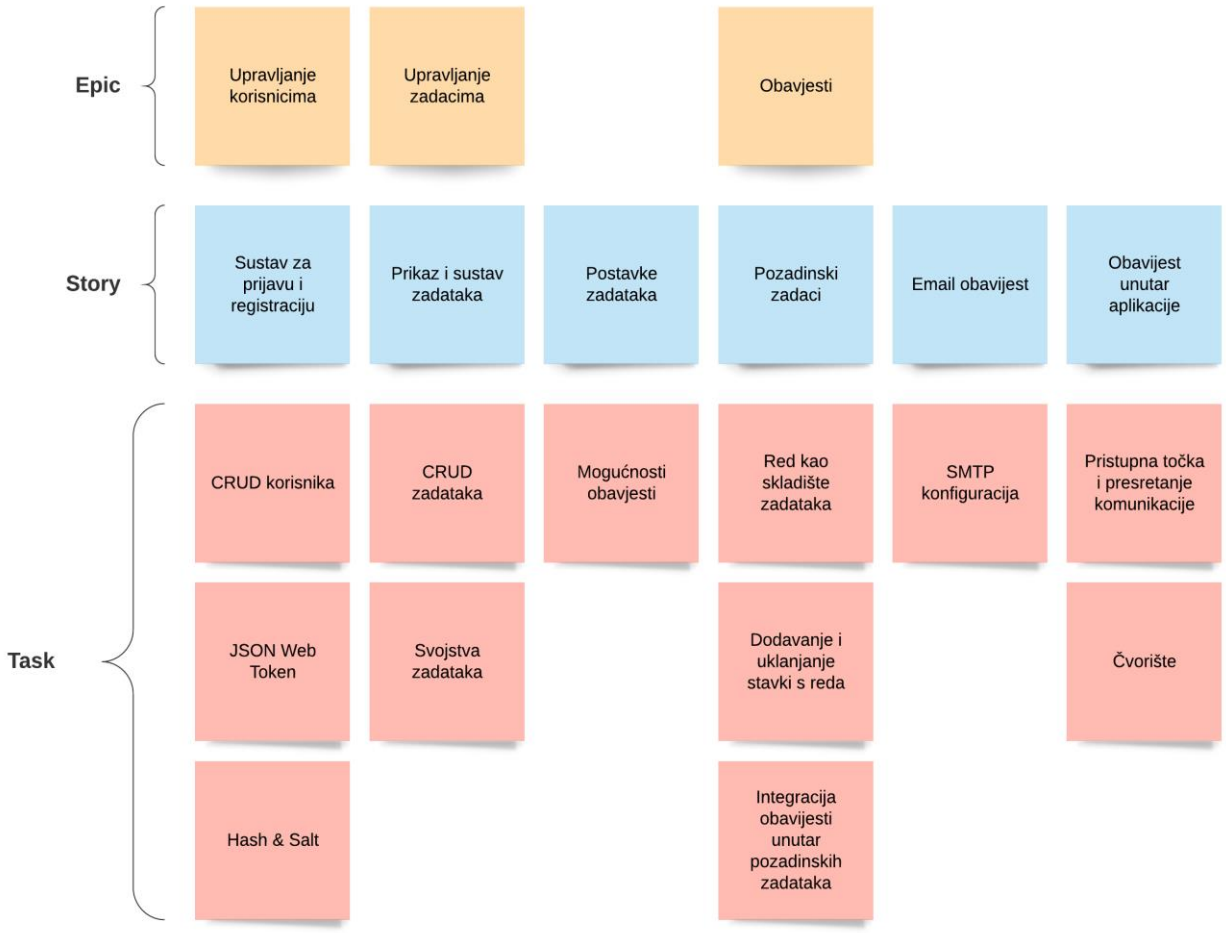
U nastavku poglavlja bit će detaljnije opisani zahtjevi na web kalendar i korištene tehnologije za implementaciju istog. Na temelju predočenih zahtjeva bit će predstavljen smjer razvoja web kalendara „.o“, a na temelju korištenih tehnologija bit će prikazana arhitektura programske izvedbe web kalendara „.o“ upotpunjena modelom.

3.1. Zahtjevi na web kalendar

Zahtjevi na web kalendar zadani su kao korisnička priča (*engl. User story*) koja je prikazana na slici 3.1. Prema toj korisničkoj priči treba izraditi interaktivni web kalendar s podrškom za popis zadataka. Obavezno je sučelje za registraciju i prijavu korisnika. Korisnici mogu imati različite ovlasti. Svaki od zadataka ima period do kojega bi trebao biti riješen. Web kalendar mora imati mogućnost prikaza liste zadataka za svaki pojedini dan. Potrebno je korisnika obavijestiti na email za sve važnije događaje oko zadataka. Ako korisnik želi, mora moći odgoditi zadatak. Ovisno o krajnjem datumu zadatka, istima se dodjeljuje odgovarajuća boja na popisu. Korisnička priča, među ostalim, prikazuje tri **epica** koji ispod sebe u hijerarhiji zadataka imaju priče i zadatke. **Epic** predstavlja okvirnu cjelinu unutar koje se nalaze priče i zadaci koji su međusobno povezani.

Prvi **epic** definira zahtjeve za korisničku registraciju, prijavu, ovjeru autentičnosti (*engl. authentication*) i autorizaciju, a drugi objedinjuje prikaze i upravljanje zadacima. Treći **epic** sadrži priče i zadatke koji definiraju korisničke obavijesti.

Svaki **epic** s pričama i zadacima bit će detaljnije objašnjen u narednim potpoglavljima.



Slika 3.1. Zahtjevi na web kalendar.

Od tri **epica**, dva dijele sličnu strukturu – upravljanje korisnicima i zadacima. Osvrtom na priče i zadatke **epica** obavijesti može se zaključiti da bi bilo idealno njegove programske niti odvojiti od druga dva **epica**. Odvajanjem bi se omogućilo neometano slanje zahtijeva poslužitelju dok bi u isto vrijeme druge programske niti bile spremne za zadatke obavijesti. Neke od tehnologija pogodne za odvajanje programskih niti **epica** su:

- ASP.NET Core,
- razni Java okviri,
- Node.js.

ASP.NET Core, kao dio okvira .NET Core ističe se kao brzo rastući moderni programski okvir. Podržava više programskih jezika i jedna je od rijetkih tehnologija otvorenog koda tvrtke Microsoft.

Novi standardi i REST principi nalažu odvajanje korisničkog (*engl. front-end*) i programskog sučelja (*engl. back-end*) što je karakteristika današnjih najprihvaćenijih programskih izvedbi.

S obzirom na prethodnu podjelu poslova na poslužitelju bilo bi dobro na sličan način podijeliti i korisničko sučelje. Time bi pojedini dijelovi korisničkog sučelja bili više puta uporabljivi, a struktura bila u skladu s popularnim principima poput REST-a. Potrebna je tehnologija koja omogućuje grupiranje pojedinih dijelova korisničkog sučelja veću logičku cjelinu. Neke od tehnologija koje omogućuju takvo grupiranje su:

- Angular,
- React,
- Vue.js.

Angular je najrobusniji programski okvir od navedenih. Sadrži već ugrađene alate za usmjeravanje (*engl. routing*) kao i puno veći broj programskih paketa i komponenti. Idealan je za srednje i velike projekte zbog odličnih performansi. Dijeli se na module koji se dijele na komponente što u potpunosti odgovara prethodno navedenim zahtjevima. Više o tehnologijama u potpoglavlju 3.2.2.

3.1.1. Upravljanje korisnicima

Upravljanje korisnicima je prvi spomenuti **epic**. Središte pozornosti istoga je sustav prijave i registracije korisnika. Kada korisnik stvori račun potrebno je osigurati njegove podatke. Za osiguranje podataka postoji više načina, a odabran je način kriptiranja korištenjem **hasha** i **salta**. **Salt** je nasumični niz bitova koji se dodaje na lozinku prilikom **hashiranja** iz kojega proizlazi **hash**. **Hashiranje** je vrsta jednosmjernog kriptiranja od kojeg je vrlo teško i neisplativo napraviti inverz. Potrebno je i autorizirati korisnika korištenjem nekog od tokena. Token je objekt unutar kojeg su sadržana korisnička prava. Token treba izdati poslužitelj i isti se treba provjeravati pri svakom pokušaju prijave radi onemogućivanja neželjenog pristupa. Ovo je još jedan razlog koji upućuje na to da bi bilo dobro odvojiti korisničko sučelje od programskog. Više o ovjeravanju autentičnosti (*engl. authentication*) i autorizaciji u poglavlju 4.2.1. Upravljanje korisnicima usko je vezano uz upravljanje zadacima jer svaki korisnik ima svoj skup zadataka.

3.1.2. Upravljanje zadacima

Epic za upravljanje zadacima u hijerarhiji ima dvije priče. Prva priča, prikaz i sustav zadataka, podrazumijeva kreiranje, uređivanje i uklanjanje zadataka s korisnikovog popisa. Potrebno je odrediti parametre zadataka koji će odgovarati većini korisnika. Za web kalendar „.o“ odabrana su četiri parametra koji opisuju zadatke: riješenost, trajanje, naziv i detalji. Zadacima se, kao i korisnicima, treba moći pristupiti metodama iz njihovih kontrolera. Za svaki zadatak treba postojati mogućnost slanja obavijesti o početku.

3.1.3. Obavijesti o zadacima

Obavijesti su, kao što je već spomenuto, **epic** koji se razlikuje od prva dva i koji koristi odvojene programske niti. Treba postojati skladište obavijesti strukture reda gdje će se dodavati i uklanjati informacije za slanje obavijesti. Obavijesti se trebaju slati određeni period prije početka zadataka. Period treba odrediti korisnik. Obavijesti su u dva oblika: putem e-pošte i unutar korisničkog sučelja. Za slanje putem e-pošte treba koristiti SMTP (*engl. Simple Mail Transfer Protocol*) kojega se prethodno konfigurira. Za slanje obavijesti unutar korisničkog sučelja potrebno je postaviti čvorište između poslužitelja i klijenata. Čvorište mora brinuti kojem korisniku se šalje koja obavijest. Više o obavijestima u poglavlju 4.2.4.

3.2. Korištene tehnologije

Prema [7] i naravi zahtjeva, ASP.NET Core se nameće kao najprirodnija tehnologija. Ostale tehnologije za programsko sučelje navedene u poglavlju 3.1. manje su popularne i imaju lošije performanse. ASP.NET Core pruža puno više alata, a zbog svoje popularnosti postoji puno dodataka (*engl. plug-in*) za njega.

Za korisničko sučelje korišten je programski okvir Angular. Kao što je već spomenuto u poglavlju 3.1., u usporedbi sa sličnim tehnologijama Angular nudi više gotovih rješenja u vidu više paketa i komponenti. Zbog toga je Angular glomazan u usporedbi s drugim okvirima.

3.2.1. Microsoft Visual Studio i Visual Studio Code

Visual Studio je Microsoftovo rješenje integriranog razvojnog okruženja (*engl. Integrated Development Environment*). Služi za razvoj računalnih igara, web programskih izvedbi i servisa i mobilnih programskih izvedbi. Uključuje uređivač koda (*engl. Code editor*) uz nadopunjavanje i ispravljanje istog (*engl. IntelliSense*). Sadrži i program za pronalazak i ispravak pogrešaka (*engl. Debugger*). Za Windows i web programske izvedbe starijih programskih okvira (Web Forms, MVC) podržava dizajner sučelja. Osnovna verzija prilično je robustna, a podržava i velik broj proširenja. Više na [8].

Visual Studio Code, [9], je lagan (*engl. Lightweight*) uređivač koda koji je također napravio Microsoft. Njegove značajke uključuju ispravak pogrešaka, nadopunu i ispravak koda, refaktoriranje i ugrađeni git. Osim što podržava proširenja, sučelje je prilagodljivo, a kod istoga je dostupan svima (*engl. Open-source*).

3.2.2. .NET Core i ASP.NET Core

.NET Core je besplatni programski okvir namijenjen za Windows, Linux i macOS platforme, a razvijen od strane Microsofta. Podržava programske jezike C#, Visual Basic i F#. Razvijen je na temelju .NET-a, programskog okvira namijenjenog isključivo za Windows platformu. Nastao je na temelju okvira .NET koji je bio namijenjen isključivo za uređaje s Windows operacijskim sustavom. Stari okvir je zato uglavnom okupljao Windows entuzijaste. Pojavom .NET Corea porastao je i broj korisnika istog. Više o .NET Core na [10]

ASP.NET Core (*engl. Active Server Pages, ASP*), [11], je programski okvir unutar većeg okvira, .NET Core. Kao i veći okvir, ASP.NET Core je nastao na temelju ASP.NET-a čiji je poslužitelj mogao biti samo uređaj s Windows operacijskim sustavom. ASP.NET Core Namijenjen je za razvoj web programskih izvedbi. Fokus razvoja spomenutog programskog okvira su performanse. Izvorni kod nalazi se na GitHub-u i dostupan je svima.

3.2.3. Angular

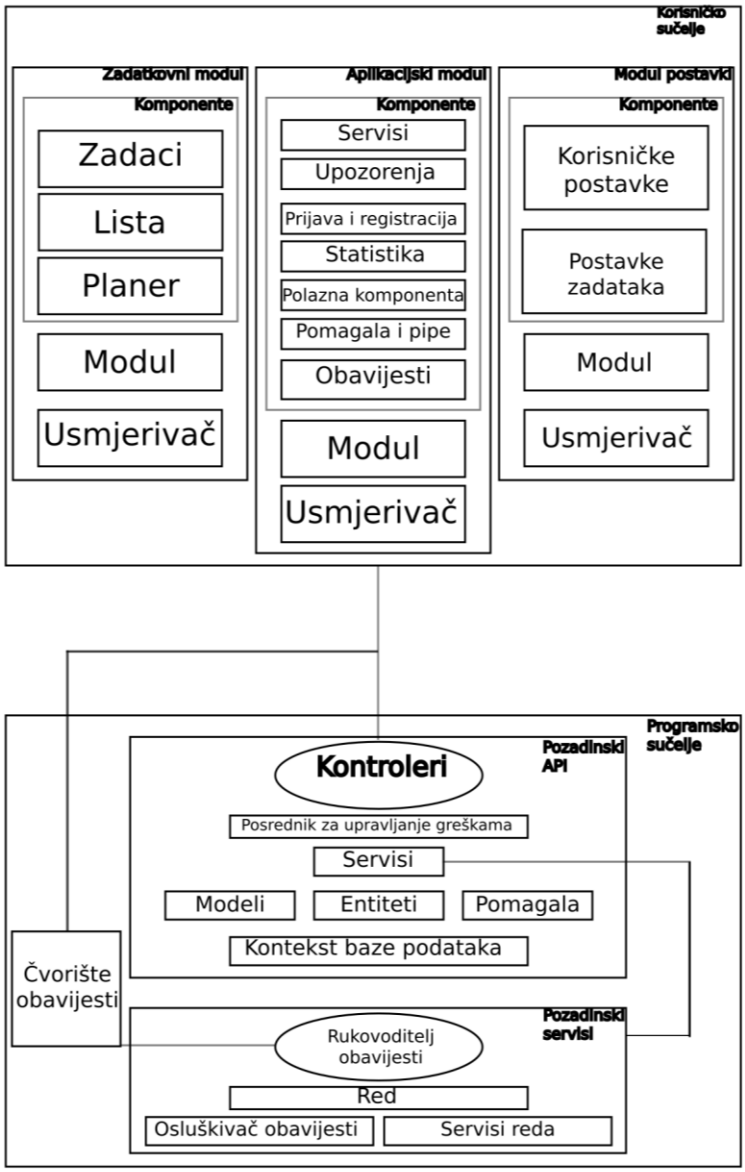
Angular je programski okvir korišten za razvoj web programskih izvedbi. Temelji se na TypeScriptu i razvija ga tim developera iz tvrtke Google. Nastao je po uzoru na AngularJS, okviru temeljenom na JavaScript jeziku, a od kojeg se razlikuje u puno značajki:

- Angular koristi hijerarhiju komponenti, a prethodnik kontrolere,
- drugačija sintaksa,
- Angular je više modularan od svoga prethodnika,
- dinamičko učitavanje komponenti, [12].

3.3. Model programske izvedbe „.o“

Imajući u obziru odabrane tehnologije i zahtjeve na web kalendar kreiran je model programske izvedbe „.o“, a vidljiv je na slici 3.2. Odabrani model posjeduje karakteristike MVC (*engl. Model-View-Controller*) arhitekturalnog stila i komponentnog stila, stoga se može zaključiti kako je riječ o hibridu navedenih stilova. Korisničko sučelje podijeljeno je na tri modula od koji svaki ima nekoliko komponenti: zadatkovni modul, aplikacijski modul i modul postavki. Svaka komponenta predstavlja dio programske izvedbe korisničkog sučelja, a u njoj su definirani izgled i funkcionalnost. Osim komponenti, svaki modul sadrži usmjerivač čija je svrha navođenje krajnjeg korisnika i promjenu aktivne komponente. Aplikacijski modul je glavni modul svake Angular programske izvedbe i služi za upravljanje ostalim modulima. Programsko sučelje dijeli se na dva veća projekta: pozadinski API (*engl. Application programming interface*) i pozadinski zadaci.

Pozadinski API sadrži kontrolere i servise koje servisi korisničkog sučelja koriste povezivanjem na pristupne točke. Pozadinski zadaci brinu se o slanju podsjetnika za korisnike.



Slika 3.2. Model programske izvedbe „.o“.

4. IMPLEMENTACIJA INTERAKTIVNOG WEB KALENDARA „.O“

U ovom poglavlju će biti opisan web kalendar „.o“ po njezinim sastavnim cjelinama. Navedeni web kalendar se sastoji od dvije veće cjeline: korisničko sučelje i programsko sučelje. Korisničko sučelje predstavlja dio programske izvedbe s kojim korisnik ima izravnu interakciju, a napravljen je u programskom okviru Angular. Programsko sučelje dio je programske izvedbe napravljen u okviru ASP.NET Core. Uz važnije funkcionalne izvedbe bit će referenciran odgovarajući **epic** iz poglavlja 3.1., čime će biti obrazložen razlog njihovog implementiranja.

4.1. Korisničko sučelje

Korisničko sučelje podijeljeno je na tri manje cjeline:

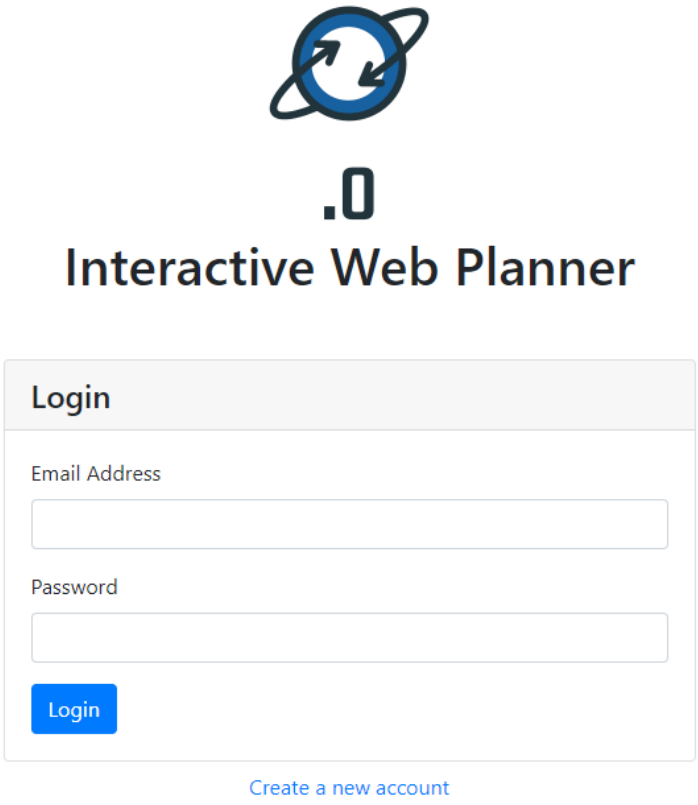
- Sučelje za prijavu i registraciju,
- Sučelja za upravljanje zadacima,
- Sučelja za upravljanje korisničkim postavkama.

Svrha sučelja za prijavu i registraciju je omogućiti korisniku pristup vlastitom računu unutar okvira Angular, što je zahtijevano u **epicu** za upravljanje korisnicima. Sučelja za upravljanje zadacima podrazumijevaju zadatkovni modul unutar kojega su različiti prikazi, statistika te modalni prozori za upravljanje pojedinog zadatka. Sučelja za upravljanje korisničkim postavkama dijele se na sučelje koje upravlja korisničkim podacima i ono koje upravlja postavkama obavijesti.

4.1.1.Sučelje za prijavu i registraciju

Kada korisnik pristupi web kalendaru, početna stranica pruža mogućnost prijave unosom elektroničke pošte i lozinke. Uneseni stringovi moraju biti validni kako bi korisnik uopće mogao zatražiti prijavu. Prisutni su validatori provjere stringa elektroničke pošte i provjere duljine lozinke. Validatori su iz modula Angular Forms. Vežu se uz komponentu Form Group. Ista predstavlja grupaciju polja za unos tekstova (*engl. textbox*), okvira za izbor (*engl. checkbox*) i općenito elemenata u koje korisnik može unijeti neku vrstu podatka. Validatori se povezuju s pojedinim elementom, a svaki element može imati više validatora povezanih na njega. Korišteni su validatori duljine stringa, validatori postojanosti, validatori e-pošte. Podaci se šalju programskom sučelju gdje se obavlja završna provjera autentičnosti podataka. Provjera će biti objašnjena u potpoglavlju 4.2.2. Ako su podaci ispravni, programsko sučelje vratit će potvrdu, a podaci o korisniku bit će spremljeni u lokalno skladište.

Ako korisnik nema izrađen korisnički račun isti može izraditi klikom na poveznicu **Create a new account** ispod forme za prijavu vidljive na slici 4.1.1



Slika 4.1.1. Sučelje za prijavu.

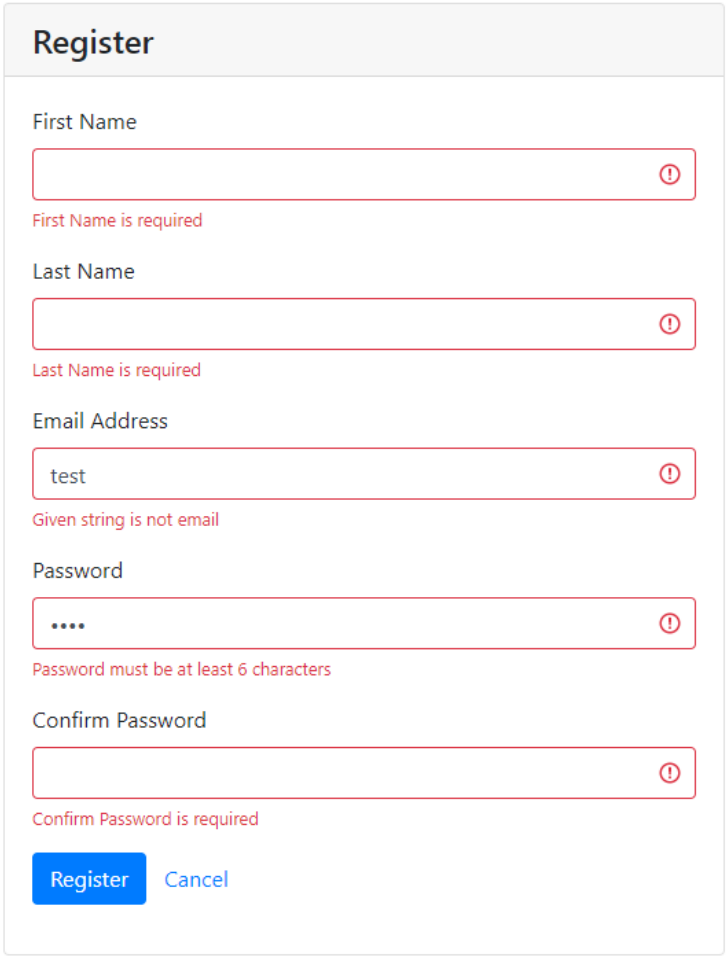
Registracija radi po istom principu kao i prijava. Validacija radi na istom principu kao kod prijave. Dodan je i validator koji provjerava jesu li lozinka i potvrdna lozinka iste. Taj validator nije u sklopu već spomenutog modula Angular Forms i potrebno ga je ručno napraviti. Ako neki od podataka nije ispravnog formata, korisnik će biti obaviješten prikladnom porukom kao na slici 4.1.2. koja je rezultat već spomenutih validatora. Ako je korisnik greškom kliknuo na registraciju i sl., može se vratiti pomoću gumba **Cancel**.

Sučelja za prijavu i registraciju koriste i pomoćne servise za ovjeru autentičnosti čije se metode pozivaju kada je potrebno pozvati neku od metoda iz programskog sučelja. Jedan od servisa je i JSON Web Token presretač koji provjerava je li token validan.

Servis za upozorenja (*engl. Alert*) služi za obavijest korisnika o povratnim informacijama zahtjeva. Osim što vraća potvrdu, programsko sučelje može i poslati poruku o pogrešci. Pogreške se dohvaćaju pomoću servisa za presretanje (*engl. Interceptor*) koji istu poruku prosljeđuje servisu za upozorenja. Servis za upozorenja korisnika obavještava u tri scenarija:

- zahtjev je uspješan,
- zahtjev je neuspješan,
- zahtjev je uspješan, ali postoje upozorenja.

Slanje poruka, uvjeti istih i izdavanje JSON Web Tokena bit će pojašnjeni u potpoglavlju 4.2.2.



Slika 4.1.2. Sučelje za registraciju.

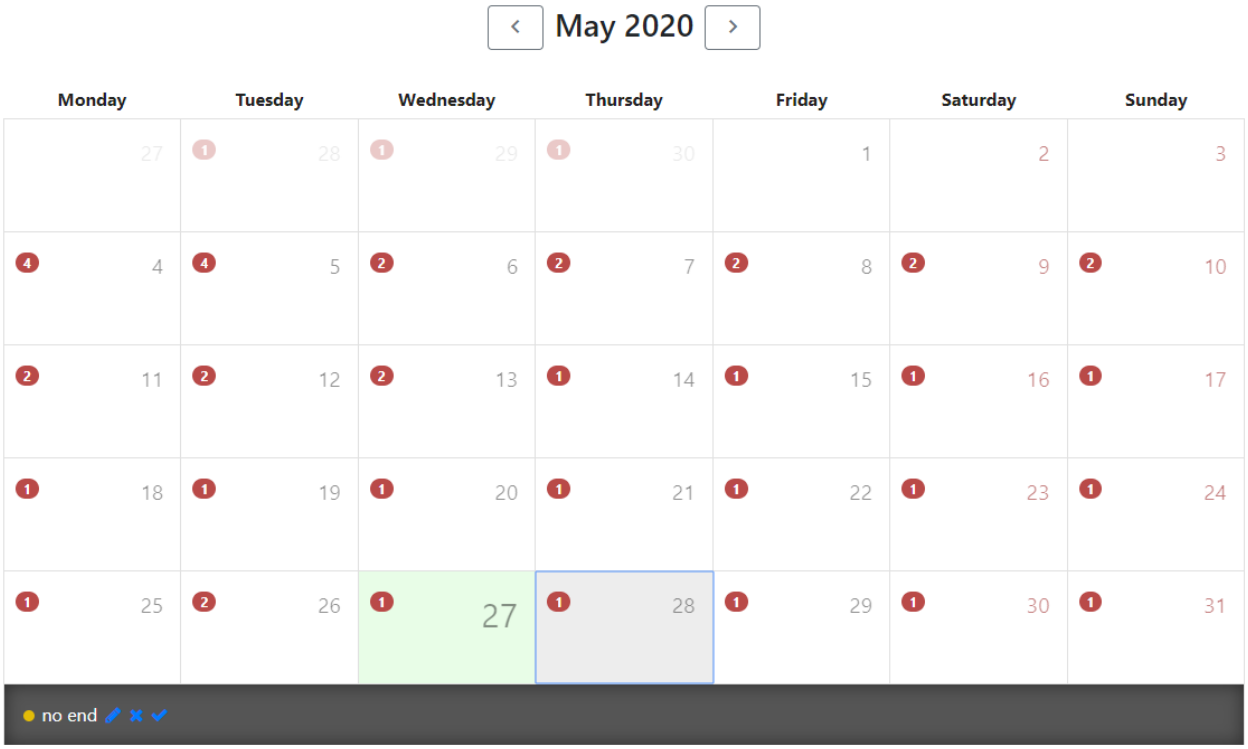
4.1.2. Sučelja za upravljanje zadacima

Za veći dio sučelja korisnika sa zadacima stvoren je zadatkovni modul unutar Angular projekta, a onda uvezen u korijenski modul app.module (*engl. Root*). Modul se sastoji od nekoliko komponenti:

- planner-view,
- event-content,
- list-view.

Ostatak je smješten izravno u korijenski modul u obliku komponenti. **Event-content** komponenta predstavlja modalni okvir za dodavanje, uređivanje i brisanje zadataka. **Planner-view** i **list-view** predstavljaju različite poglede koji su dio zahtjeva **epica** za zadatke postavljenih u poglavlju 3.2.

Nakon prijave, korisnik ima prikaz polazne stranice (*engl. Home Page*) vidljive na slici 4.1.3.

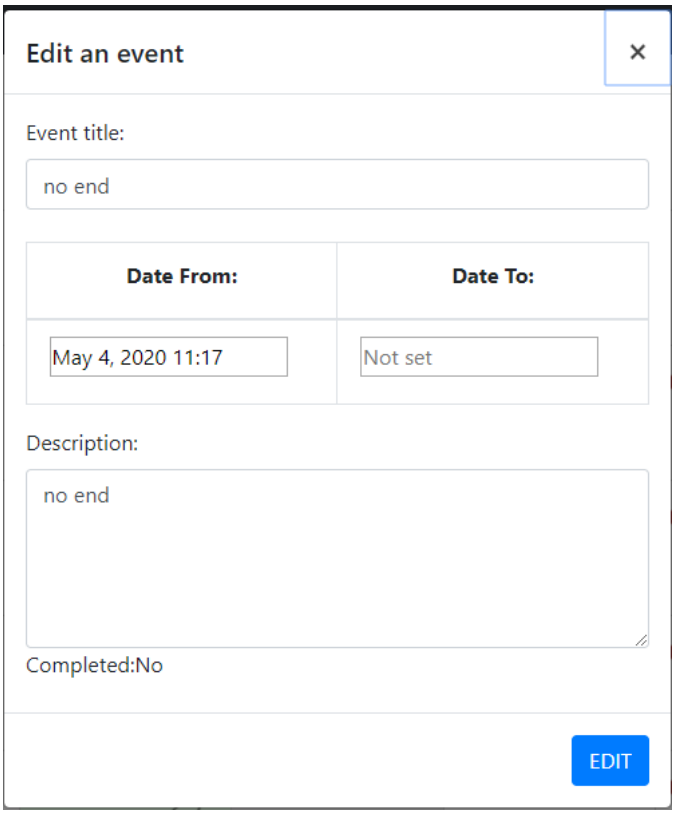


Slika 4.1.3. Početna stranica i prikaz kalendara.

Na polaznoj stranici prikazana je komponenta **Planner-view**. Zadaci su prikazani na kalendaru za trenutni mjesec gdje je trenutni dan naznačen svijetlo zelenom bojom. Klikom na bilo koji dan otvara se manja alatna traka s popisom zadataka za taj dan ako istih ima. Broj zadataka nekog dana vidljiv je u crvenom krugu istoga dana. Svaki zadatak opisan je sljedećim parametrima:

- identifikatorom,
- naslovom,
- datumom i vremenom početka,
- datumom i vremenom završetka,
- bojom,
- mogućnošću uređivanja izravnim upisom,
- mogućnošću uređivanja pomicanjem,
- opisom,
- obavljenosti.

Mogućnost uređivanja izravnim upisom odnosi se na uređivanje zadatka otvaranjem modalnog prozora na slici 4.1.4. Mogućnost uređivanja pomicanjem znači da se zadatku početni i krajnji datum mogu promijeniti pomakom istoga mišem. Mogućnost uređivanja pomakom nemaju samo zadaci bez krajnjeg datuma.



Slika 4.1.4. Modalni prozor za uređivanje zadataka.

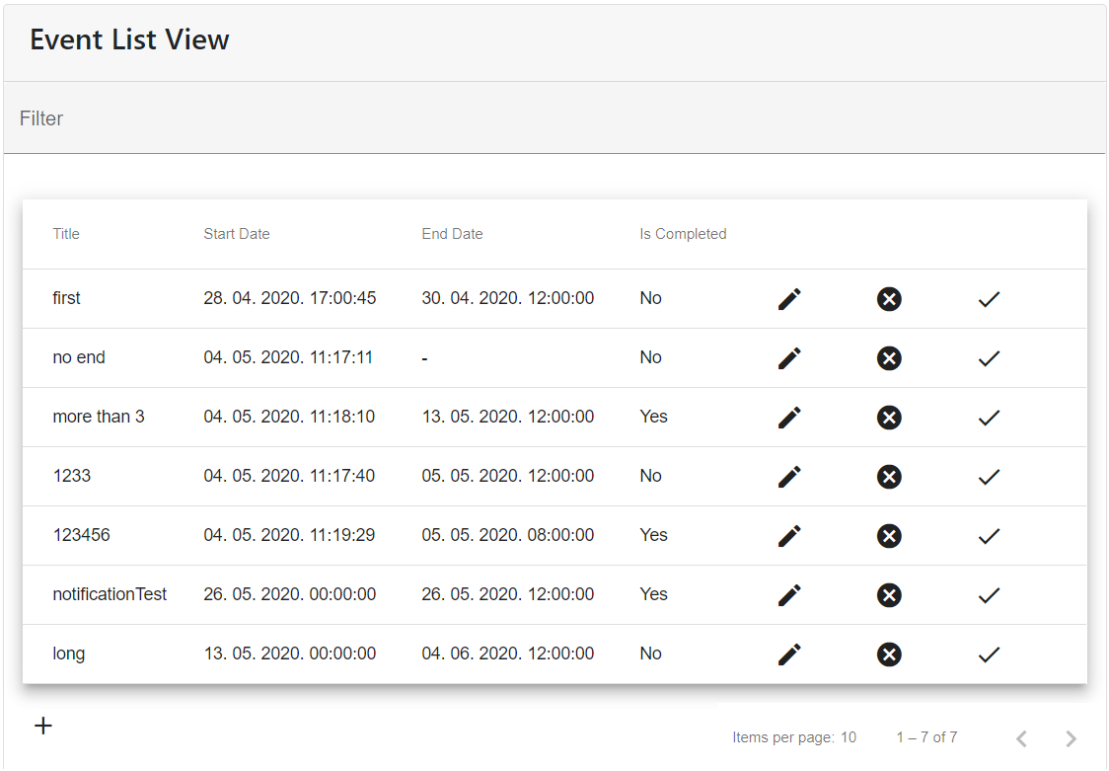
Svojstvo obavljenosti zadatka u modalnom prozoru može se samo čitati. Za mijenjanje istoga potrebno je na već spomenutoj traci zadataka u kalendaru kliknuti na ikonu kvačice pri čemu se otvara modalni prozor za potvrdu. Modalni prozor također posjeduje validatore koji se brinu o ispravnosti uređivanog zadatka. Osim opisa i krajnjeg datuma, svako svojstvo mora postojati. Krajnji datum ne smije biti manji od početnog. Sva ostala vidljiva svojstva mogu se uređivati uz poštivanje validatora. Za kalendarski prikaz korišten je Angular Calendar, dostupan na [13], a alternative istoga navedene su u poglavlju 4.3.

Modalni prozor je prikaz komponente event-content. Ista je uvezena u komponentu **planner-view** kao ViewChild. ViewChild je tzv. ukras (*engl. decorator*) koji služi za ubrizgavanje reference komponente u neku drugu. Ako je zadatak odabran, isti je ulazni parametar za komponentu modalnog prozora, a ako nije, novi zadatak se inicijalizira.

Desnim klikom na bilo koji dan otvara se izbornik s opcijom za dodavanje novog zadatka i opcijom za provjeru dana. Prva opcija otvara već spomenuti modalni prozor koji nudi unos novog zadatka. Druga opcija otvara dnevni prikaz zadataka.

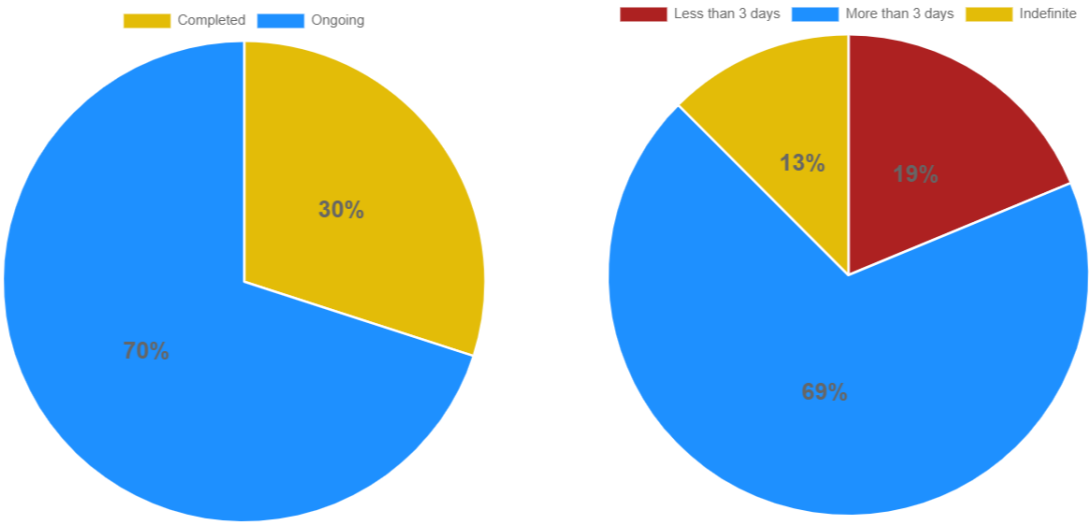
Zadaci su u istom prikazani kao vremenske crte (*engl. Timeline*). Korisniku je dopušteno imati proizvoljan broj zadataka u danu.

Dvije su vrste korisnika: oni s potpunim pristupom i oni s ograničenim pristupom. Korisnici s ograničenim pristupom zadatke mogu vidjeti u kalendaru ili u dnevnoj vremenskoj crti. Korisnici s potpunim pristupom imaju mogućnost pregleda zadataka u tablici. Zadaci su poredani po datumu kreiranja, a omogućeni su filtar, paginacija i sortiranje(slika 4.1.5.). Filtar se primjenjuje na bilo koji stupac u obliku stringa. Omogućene su proizvoljne pipe (*engl. Custom Pipe*) koje prilikom mapiranja zadataka u tablicu pretvaraju svaki podatak u string, pa tako i boolean vrijednosti. Lista ima sve mogućnosti koje pruža i kalendarski prikaz. Za prikaz je korištena komponenta mat-table iz modula Angular Materials o kojemu više na [14].



Slika 4.1.5. Prikaz liste zadataka.

S obzirom na zahtjeve **epica** zadataka postavljene u poglavlju 3.1.2., korisnik može vidjeti statistiku svojih zadataka. Ista je u obliku tortastih dijagrama (*engl. Pie Chart*). Na slici 4.1.6.a je prikaz prema dovršenosti zadataka, dok je na slici 4.1.6.b prikaz prema duljini trajanja. Boje desnog dijagrama ekvivalentne su bojama iz kalendarskog prikaza. Korisnik u jednom trenutku može izabrati prikaz samo po jednoj kategoriji, a izbor je omogućen jednostavnim padajućim izbornikom. Ukupni broj zadataka također je prikazan na stranici.



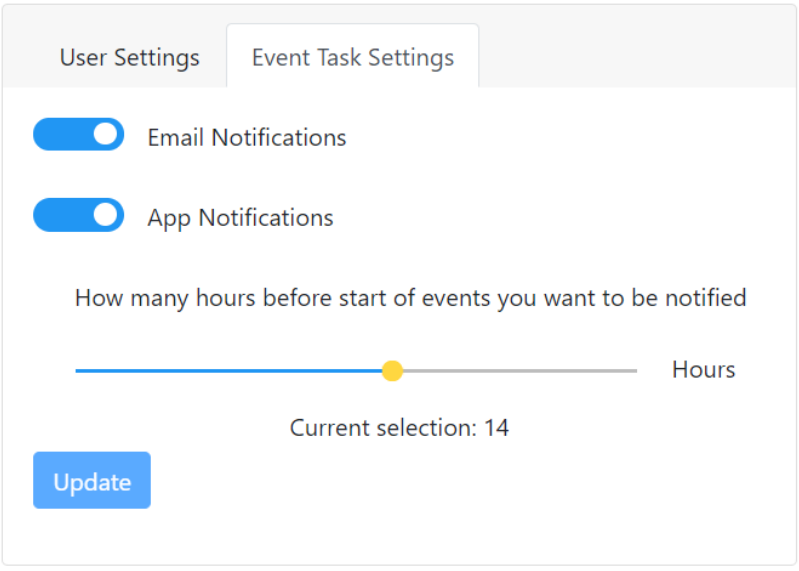
a) *b)*

Slika 4.1.6. Prikaz statistika zadataka.

4.1.3. Sučelja za upravljanje korisničkim postavkama

Korisnik može u svakom trenutku promijeniti postavke. Postavke su podijeljene na one vezane uz račun i one vezane uz zadatke, a u korisničkom sučelju web kalendara su podijeljene u kartice (*engl. tabs*). Prilikom mijenjanja postavki računa potrebno je unijeti trenutnu lozinku. Izgled sučelja za promjenu postavki računa gotovo je isti onom za registraciju korisnika (slika 4.1.2.).

Promjena postavki zadataka odnosi se na obavijesti. Korisnik može birati želi li biti obaviješten unutar korisničkog sučelja web kalendara i/ili putem elektroničke pošte. U početku su obje mogućnosti omogućene. Također može izabrati u kojem periodu prije početka zadatka želi da mu stigne obavijest. Postavljanje vremena je u obliku kliznog izbornika (*engl. Slider*), a dopušteni raspon je unutar 24 sata. Gumbi za spremanje postavki onemogućeni su sve dok barem jedna



Slika 4.1.7. Postavke obavijesti zadataka.

postavka nije uređena. Mogućnost je ostvarena korištenjem svojstva zaprljanosti elemenata (*engl. Dirty*). Postavke su vidljive na slici 4.1.7.

Obavijesti su korisniku dane u modalnom okviru za upozorenja korisnikovog preglednika. Iste se prikazuju bez obzira na kojoj stranici se korisnik nalazi i imaju veći prioritet od modalnih prozora. Ako korisnik dodaje zadatak, a dobije obavijest, istu mora zatvoriti kako bi nastavio s daljnjim radom (slika 4.1.8.). Više o obavijestima u poglavljima 4.2.3. i 4.2.4.

4.2. Programsko sučelje

Programsko sučelje sastoji se od dva veća .NET projekta:

- Backend API,
- EventTasks Background Services (pozadinski servisi).

Prvi od projekata prvenstveno je namijenjen za zahtjev – odgovor vezu između klijenta i poslužitelja. S obzirom na korisničku priču predstavljenu u trećem poglavlju, podrazumijeva **epic** upravljanja zadacima. Sadrži pristupne točke (*engl. Endpoint*) koje poziva Angular klijent. Sadrži čitavu logiku za pristup bazi korištenjem Entity Frameworka i, kao što je navedeno, za interakciju s korisničkim sučeljem. Osim toga definira i neke entitete koji se koriste u drugom projektu.

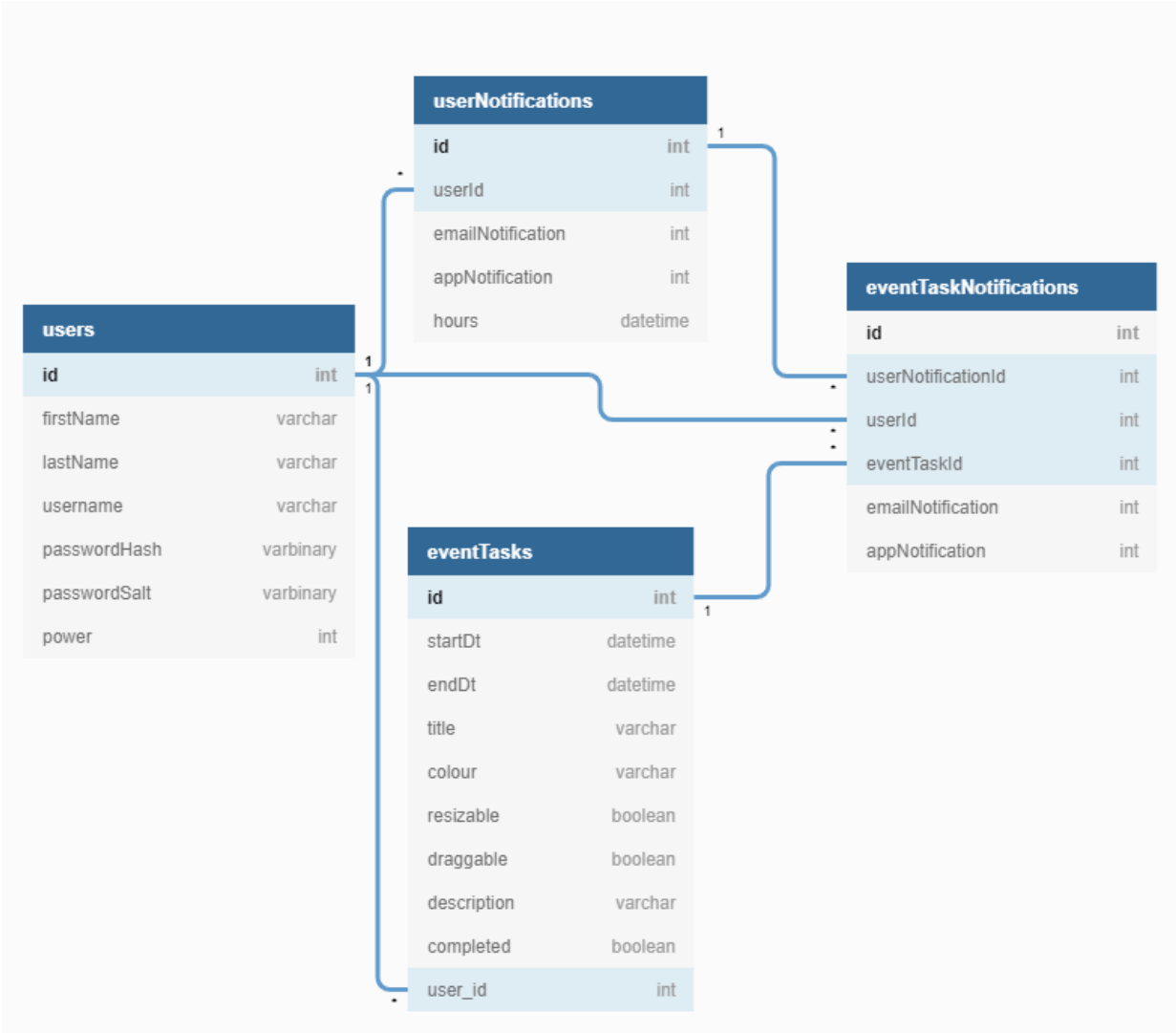
Drugi projekt predstavlja logiku za pozadinske zadatke i implementira čvorište obavijesti kojega poziva isti projekt i Angular klijent. Drugi projekt predstavlja programski dio **epica** za obavijesti naznačenog u trećem poglavlju.

Više o oba projekta bit će objašnjeno u sljedećim potpoglavljima.

4.2.1. Baza podataka i Entity Framework

Po uzoru na zahtjeve na web kalendar, baza podataka izrađena je Code-First pristupom korištenjem Entity Frameworka. Pristup započinje kreiranjem željenih entiteta koji predstavljaju tablice u bazi, a prikazani su dijagramom na slici 4.2.1.

Tablica **users** sadrži korisnička svojstva. Više o svojstvima **password hash** i **password salt** bit će u sljedećem potpoglavlju. Tablica **eventTasks** predstavlja zadatke. Sadrži strani ključ koji ju veže na tablicu users. Tablica **userNotifications** je za korisničke postavke koje uključuju i postavke o obavijestima. Sadrži informacije o tome na koji način korisnik želi biti obaviješten. Tablica **eventTaskNotifications** ima informacije o kojim zadacima je korisnik već obaviješten kao i strane ključeve na sve ostale tablice.



Slika 4.2.1. Baza podataka.

Code-First pristup zahtijeva i migracije prije stvaranja same baze. Migracije se odrađuju korištenjem .NET CLI (*engl. Command Line Interface*) ili Windows PowerShella. Za projekt izrađen u sklopu ovog diplomskog rada korišten je Windows PowerShell, a više o Entity Frameworku na [15]. Nakon uspješne migracije moguće je postaviti dodatna ograničenja (*engl. Constraints*), definirati procedure i sl.

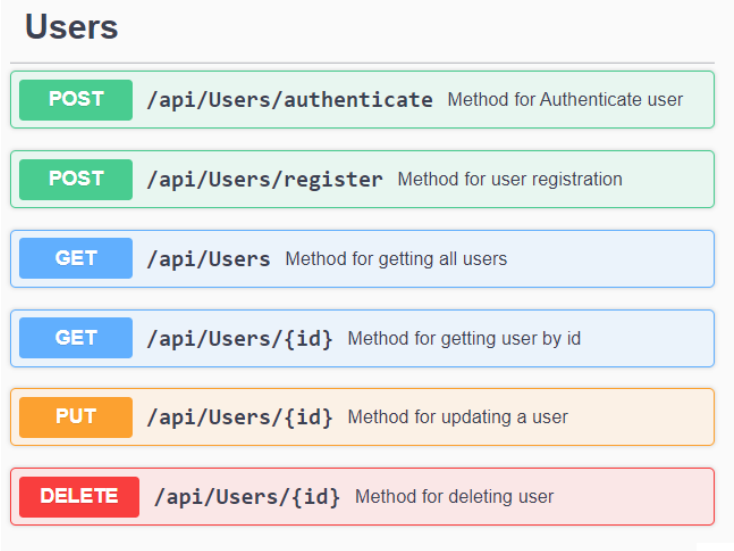
Datoteke s migracijama nalaze se među ostalima u projektu. Iste su vidljive u dijelu sučelja Visual Studio Solution Explorer.

4.2.2. Interakcija s korisničkim sučeljem

Interakcija s korisničkim sučeljem obuhvaća metode pristupa podacima, servise i druge pomoćne metode i ekstenzije koje se prilikom navedene interakcije koriste.

Svaka pristupnih točki u sklopu je kontrolera koji se temelji na jednom od entiteta korištenih za kreiranje baze podataka. Adresa pristupa bilo kojeg kontrolera definirana je u datoteci **appsettings.json**. Ista datoteka sadrži ponavljajuće informacije u JSON formatu. Svaki kontroler ima svoju jednostavnu dokumentaciju napravljenu pomoću Swaggera.

Swagger, [16], je novi standard opisivanja i dokumentiranja RESTful API dizajna. Osim za .NET, ima podršku i u ostalim poznatim programskim okvirima poput Laravela, Node.js-a, Angulara itd. Prilikom izrade API-ja korišteni su neki RESTful principi. Kao što je već napomenuto u prethodnim poglavljima, programsko i korisničko sučelje razdvojeni su što omogućuje odvojeni razvoj. API pristupne točke, osim za prve dvije metode sa slike 4.2.1., imaju općenito ime. Razlikuju se u vrsti metode i **string query** parametrima. U komunikaciji klijent-poslužitelj tip JSON definira elemente za promjenu stanja. Više o REST-u na [17].



Slika 4.2.1. Pristupne točke za entitet korisnika.

Prilikom registracije novog korisnika svi podaci se, ako su validni, šalju u programsko sučelje. Svi podaci, osim lozinke, spremaju se u bazu kakvi su i poslani. Lozinka se kriptira koristeći SHA512 hash funkciju. Lozinka je hashirana u polje od 512 bita (*engl. Hash-based Message Authentication Code, HMAC*). U bazu se također se sprema i ključ korišten za navedeno hashiranje. Ključ i kod se koriste i prilikom ovjere autentičnosti gdje se predana lozinka uspoređuje s onom dobivenom inverznim hashiranjem. Takav način ovjere autentičnosti sprječava administratora baze podataka da vidi lozinku korisnika. HMAC-SHA512, [18], je postojeća klasa unutar ASP.NET Core-a s već postojećim metodama za navedeno kriptiranje.

Osim korištenja hash-salt kriptiranja lozinke, u programskoj izvedbi web kalendara „.o“ se koristi i JWT (*JSON Web Token*) za dodatnu ovjeru autentičnosti, [19]. JWT je otvoreni standard koji

definira sigurni prijenos podataka između klijenata i poslužitelja. Koristi HMAC algoritam za kriptiranje.

JWT se sastoji od tri dijela:

- zaglavlje,
- tijelo,
- potpis.

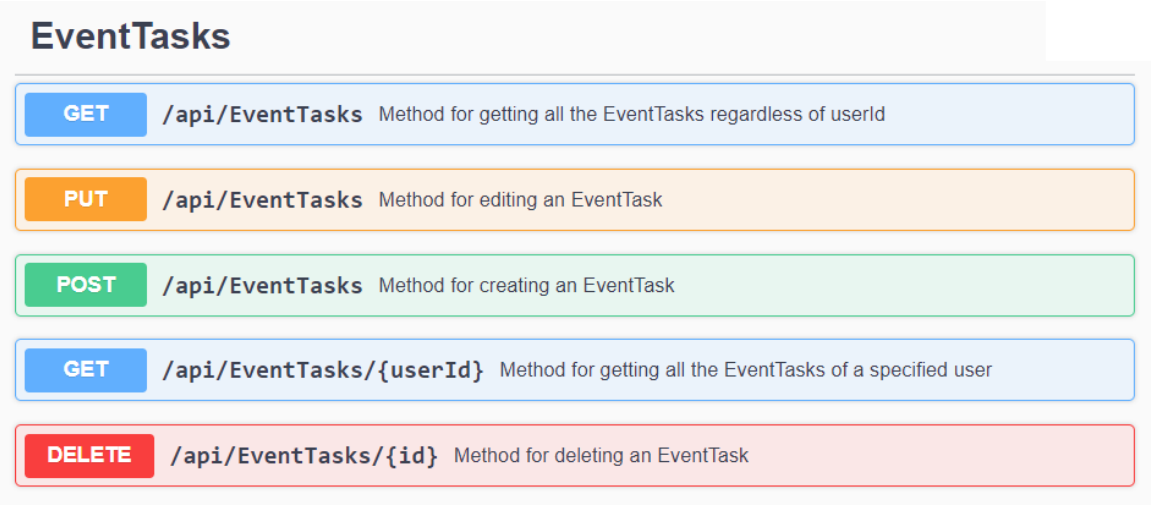
U zaglavlju su informacije o vrsti tokena i algoritmu korištenom za kriptiranje. Tijelo sadrži informacije o pravima. Prava su podaci o entitetu (uglavnom korisnik). Prije kreiranja potpisa potrebno je kriptirati zaglavlje i tijelo. Potpis je uvjerenje o autentičnosti poruke, odnosno potvrda da poruka nije mijenjana putem. U isto vrijeme potpis je i potvrda da je pošiljatelj onaj koji tvrdi da je. U sadržaju ASP.NET Core okvira nalazi se i JWT rukovatelj (*engl. handler*).

Prilikom svakog zahtjeva za prijavu korisnika poziva se metoda Authenticate iz kontrolera za korisnike. U istoj je instancirana klasa JWT rukovatelja. Ako su korisničko ime i lozinka ispravni kreira se token. Za prava je uzet korisnički identifikator i dodano je da token traje sedam dana. U potpisu se koristi tajni ključ postavljen kao string u već spomenutoj datoteci appsettings.json. Nakon kreiranja, token se proslijeđuje klijentu koji ga zahtijeva zajedno s još nekim potrebnim informacijama o korisniku.

Za sve ostale upite dodan je posrednik za tokene (*engl. JWT Bearer middleware*) koji se također nalazi u okviru ASP.NET Core. Svaki upit se presreće i provjerava se ispravnost tokena. URL zahtjev (*engl. Uniform Resource Identifier*) se provjerava kao string posjeduje li token. Ako posjeduje, zahtjev se proslijeđuje kontroleru. Ako ne, klijent dobiva informaciju o greški.

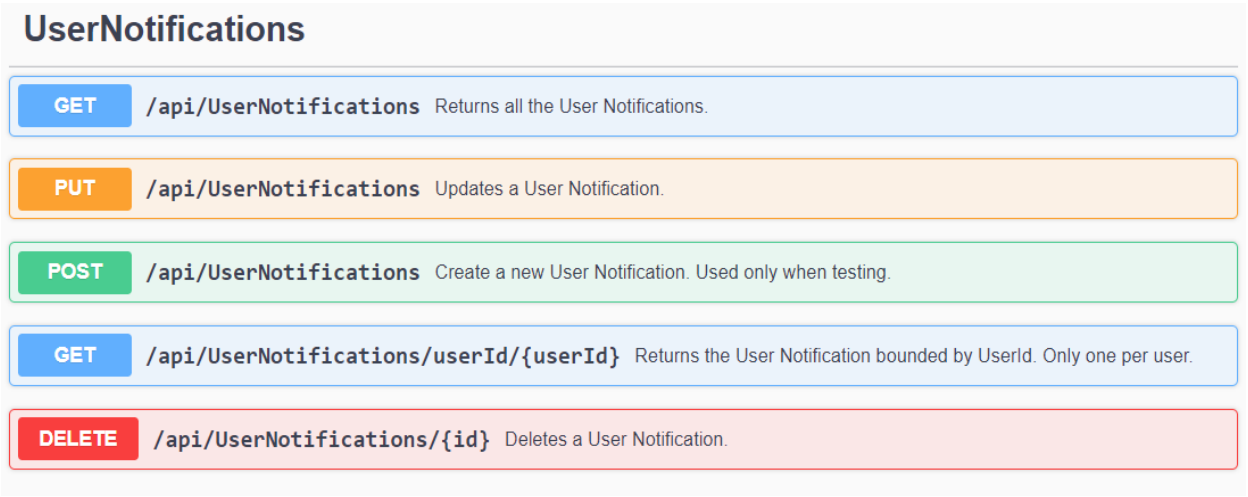
Korisniku se, dakle, ovjerava autentičnost kriptiranjem i dekriptiranjem lozinke SHA512 algoritmom kriptiranja, dok se autorizira identifikacijom JWT-a dobivenog sa strane poslužitelja.

Na slici 4.2.2. predočene su pristupne točke kontrolera za zadatke. Prilikom bilo kojeg zahtjeva odvija se obnavljanje boje. Provjerava se je li razlika između datuma provjere i datuma završetka zadatka promijenjena i prema tome se dodjeljuje nova boja. Crvena boja predstavlja zadatke kojima je datum završetka unutar 3 dana ili je isti prošao. Zadaci koji imaju plavu boju imaju i datum završetka koji je od trenutnog veći od tri dana. Žuta boja je za one zadatke koji nemaju definiran datum završetka. Za definiranje boja korištene su pomoćne klase definirane unutar datotečnog sustava projekta. Mijenjanje boja odvija se i u pozadinskim zadacima koji će biti definirani u sljedećem potpoglavlju.



Slika 4.2.2. Pristupne točke za entitet zadatka.

Pristupne točke korisničkih obavijesti priložene su na slici 4.2.3. Prilikom stvaranja korisnika istome se dodaju i prava koja se, kao što je navedeno u poglavlju 4.1.3. mogu mijenjati.



Slika 4.2.3. Pristupne točke za entitet korisničkih postavki.

4.2.3. Pozadinski servisi

Programsko sučelje podijeljeno je u dva projekta zbog potrebe za neovisnim obavljanjem poslova. Ako se jedan od projekata sruši, drugi i dalje može obavljati svoj posao jer koriste različite programske niti (*engl. thread*).

Neki od gotovih rješenja pozadinskih servisa su:

- Hangfire,
- ASP Scheduler.

S obzirom da jedno od rješenja zahtijeva komercijalnu dozvolu koja se plaća, a drugo ograničava korisnika od prerade programskog koda, implementiran je pozadinski servis web kalendara „.o“.

Osim dozvole, alternativna rješenja uglavnom su glomazna i sadrže funkcionalnosti koje vjerojatno nikada ne bi bile korištene.

Pozadinski servisi koriste više servisa iz Web API projekta koji su ubrizgani (*engl. Dependency Injection*). U okviru ASP.NET Core omogućeno je među-projektno (*engl. cross-project*) ubrizgavanje uz pridržavanje nekih pravila. Svaki korišteni servis potrebno je postaviti unutar **Program.cs** klase projekta u kojega se ubrizgava. Nužno je referencirati se na Startup.cs klasu izvornog projekta zbog različitih životnih ciklusa servisa:

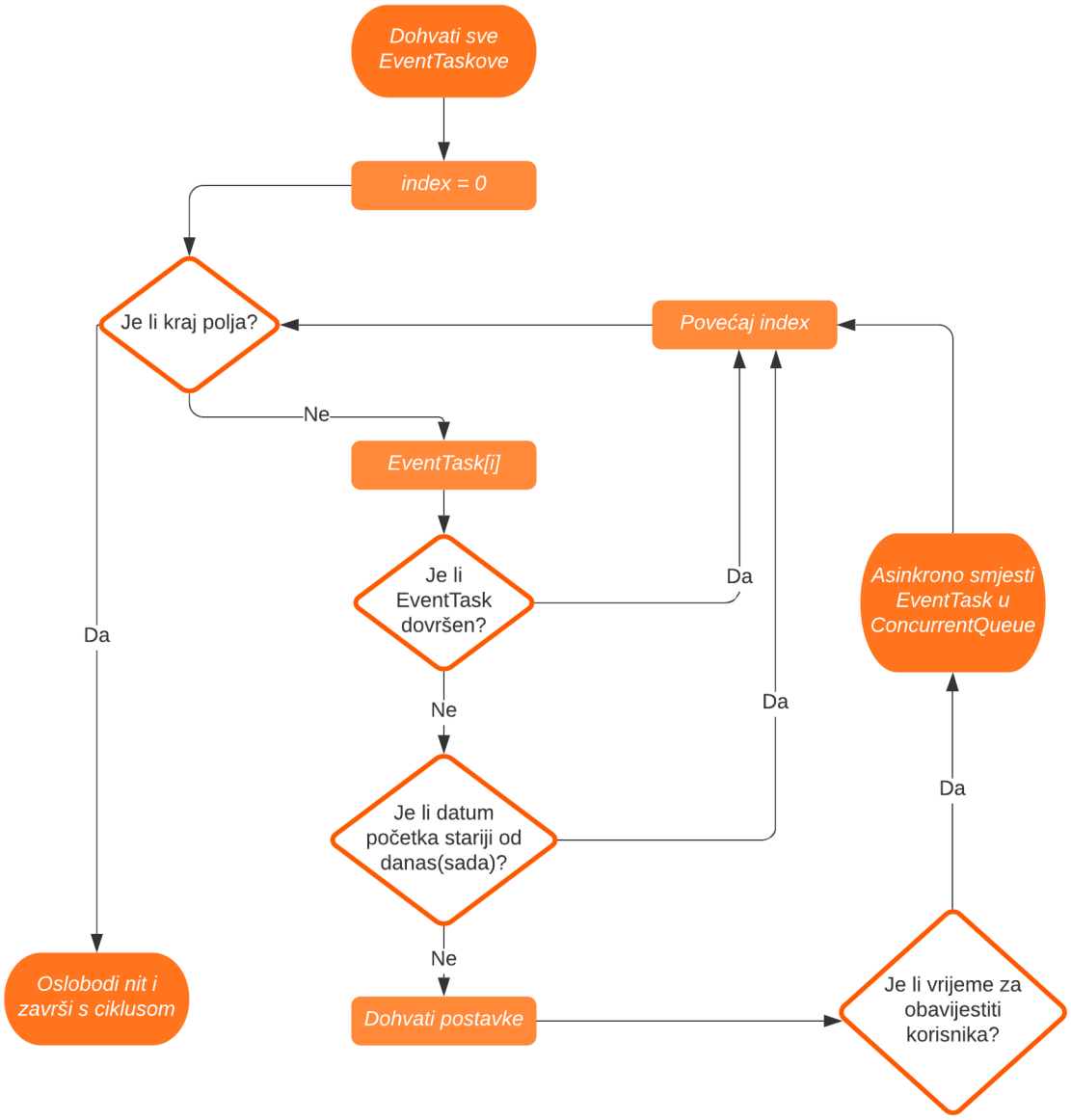
- tranzijentni ciklus,
- ograničeni ciklus,
- jedinični ciklus.

Tranzijentni ciklus označava da se servis instancira nanovo prilikom svakog zahtjeva. Ograničeni ciklus (*engl. scoped*) ograničava instanciranje servisa na jedanput po zahtjevu klijenta. Jedinični ciklus (*engl. singleton*) znači da se servis instancira samo prilikom prvog poziva. U svakom daljnjem pozivanju koristi se ista instanca.

Dependency Injection, [20], koristi se unutar svakog od projekata. Unutar jednog projekta je, primjerice, moguće ubrizgati servis jediničnog ciklusa unutar servisa ograničenog ciklusa. Ispravljač greški tada javlja upozorenje da je ciklus ubrizganog servisa promijenjen na ograničeni. Isto upozorenje je prilikom cross-project ubrizgavanja tretirano kao pogreška te se zaustavlja daljnje izvođenje oba projekta.

S obzirom na postavljene zahtjeve u trećem poglavlju, za logičko središte pozadinskih servisa korišten je istodobni red (*engl. concurrent queue*). Uzeta je postojeća struktura iz okvira .NET Core, **ConcurrentQueue** klasa s principom prvi unutra, prvi van (*engl. first in – first out, FIFO*) s thread-safe podrškom. Detaljnije informacije o klasi su na [21]. U vrijeme iniciranja reda inicira se i semafor koji ograničava broj programskih niti ili broj bazena niti koji se mogu koristiti. U projektu je korišten semafor koji ograničava red na jednu programsku nit, istovremeno se samo jedna stavka može obrađivati. Za semafor je korištena postojeća klasa **SemaphoreSlim**, [22], unutar okvira .NET Core.

Svakih deset sekundi pozadinski servis dohvaća sve zadatke iz baze gdje ih filtrira kao dijagram toka na slici 4.2.4. Jedna programska nit iz posebnog bazena zadužena je za taj proces. Prilikom dohvaćanja svih zadataka okida se i funkcija za promjenu boje i zadatak se osvježava u bazi. Prilikom sljedećeg pozivanja korisnik će dobiti novu boju ako je došlo do promjene.



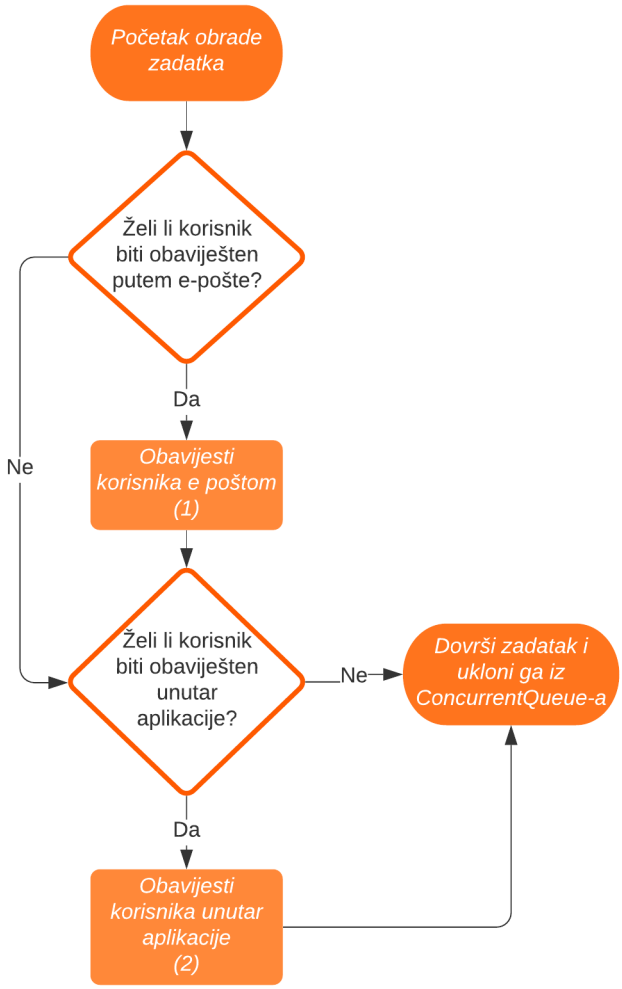
Slika 4.2.4. Dijagram toka dodavanja zadataka u red.

Kada servis završi sa smještanjem zadataka u red, programska nit se oslobađa i počinje s obradom zadataka iz reda. Programska nit se ponovno oslobađa tek kada se obrade svi zadaci. Proces obrade zadataka nešto je kompleksniji od onoga za dodavanje, a pojednostavljeni dijagram toka može se vidjeti na slici 4.2.5.

Korisnik se e poštom (1) obavještava uz korištenje SMTP poslužitelja, [23]. Korišten je besplatan Google SMTP poslužitelj, [24], koji zahtijeva Google račun.

Okvir ASP.NET Core obuhvaća i paket koji podržava slanje e-pošte putem SMTP-a. Osim e-pošte, potrebno je navesti i lozinku iste te definirati portove koji će se koristiti. Spomenuti paket

omogućuje slanje proizvoljne poruke sa svim mogućnostima koje pruža davatelj usluge e-pošte (Google).



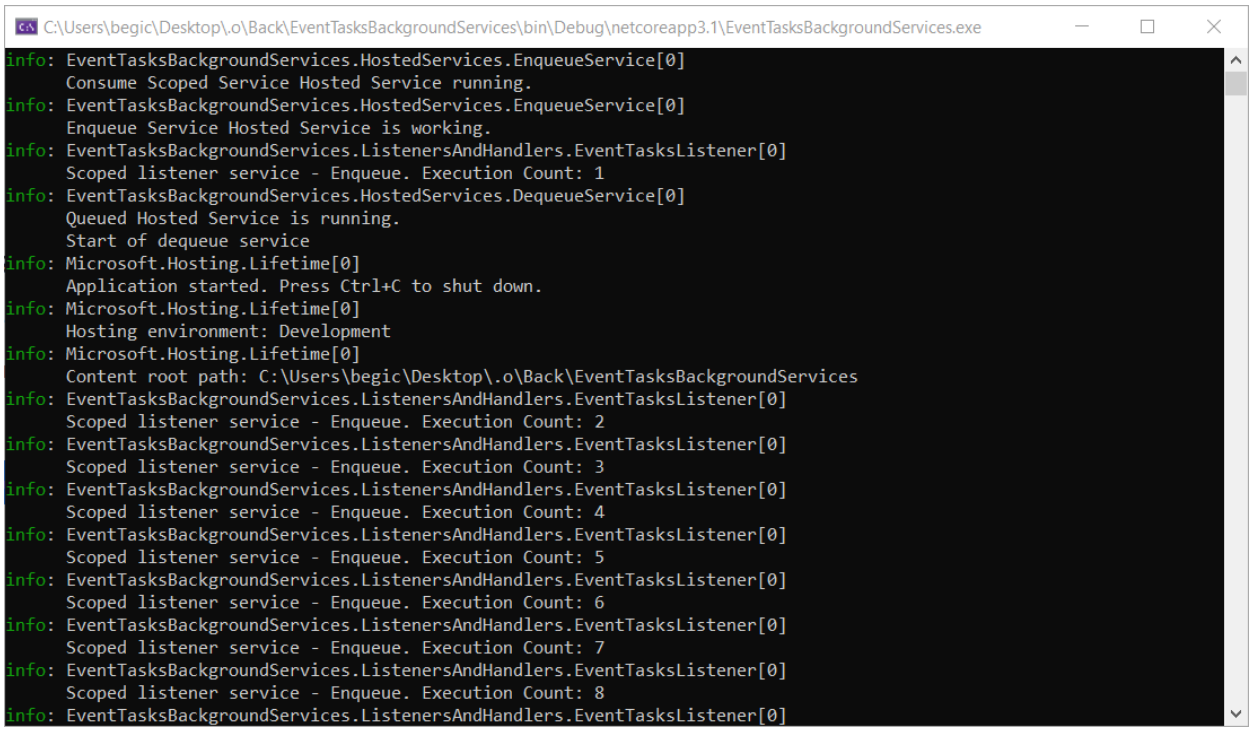
Slika 4.2.5. Dijagram toka obrade zadataka iz reda.

Međutim, moguće je da slanje ne uspije. Pozivanje metode za slanje e-pošte obuhvaćeno je unutar pokušaj-uhvati (*engl. try-catch*) bloka. Ako slanje ne uspije, redak u tablici za obavijesti neće sadržavati istinitu zastavicu za e-mail obavijest. Posljedica će biti da će se isti zadatak ponovno dodati u red i doći će do ponovnog pokušaja slanja. Svaki neuspješni pokušaj bit će zabilježen u zapisima izvedbe (*engl. logger*).

Zapisi izvedbe, osim neuspješnog slanja e-maila, sadrže informacije o većini pozadinskih procesa:

- dodavanje zadataka u red i broj okidanja procesa,
- uklanjanje i obavljanje zadataka,
- greške prilikom dodavanja i uklanjanja,

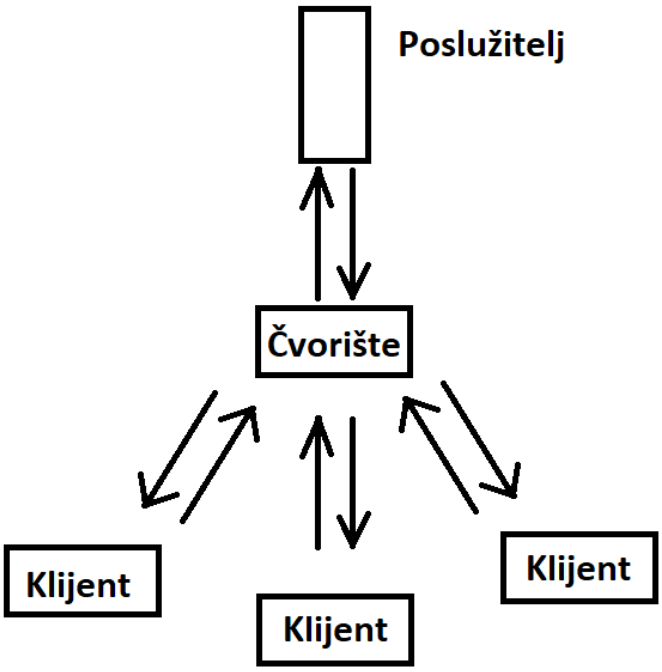
Izgled konzole i primjer zapisa vidljiv je na slici 4.2.6.



Slika 4.2.6. Konzolni zapis procesa pozadinskih servisa.

4.2.4. Čvorište obavijesti

Čvorište obavijesti (*engl. Notification hub*) je posrednik između komunikacije klijent-poslužitelj vezane uz obavijesti za zadatke (slika 4.2.). Brine se o otvorenim vezama između klijenata i poslužitelja i odlučuje kojem klijentu se šalje koja obavijest.



Slika 4.2.7. Čvorište.

Za web kalendar je korišteno SignalR čvorište iz programskog okvira ASP.NET Core, [25]. SignalR omogućuje pozivanje metoda sa servera od strane klijent. Naziv čvorišta se definira navođenjem jednoznačnog imena u obliku stringa koji se koristi prilikom spajanja klijenata i definiranja upravljačkih metoda.

U ASP.NET Core okviru definirani su SignalR klijentski paketi za različite klijente:

- .NET Core klijent,
- Java klijenti,
- Javascript klijenti (uključuje i Typescript).

Za početak veze potrebno je na poslužiteljskoj strani definirati upravljač koji će služiti samo za početak veze. Upravljač mora imati definiranu metodu za povezivanje koja posjeduje atribut s jednakim nazivom kao spomenuti naziv čvorišta. Ako je klijent uspješno spojen, šalje mu se potvrdna zastava.

SignalR definira metode okidače za spajanje i odspajanje klijenata. Zbog sigurnosnih potreba web kalendara iste su pregažene posebnim metodama koje će omogućiti mapiranje veza i identificiranja klijenata.

Prvu ulogu ima posrednik unutar programskog sučelja (*engl. middleware*). Kao što je spomenuto u prethodnim poglavljima, isti presreće spajanja i provjerava postoji li JWT. Ako JWT postoji provjerava se kojem korisniku je izdan. Povezivanje korisničkog identifikatora i konekcijskog identifikatora predstavlja mapiranje veza.

Sve metode na poslužiteljskoj strani su asinkrone pa se nekada metode čvorišta pravovremeno ne okinu i obavijest ne stigne do klijenta. Zato je implementiran .NET Core klijent u pozadinskim servisima koji okida metodu za slanje koja je, između ostalih, prikazana na slici 4.2.8.

```
public class NotificationHub : Hub
    {
        static HashSet<HubConnectionsModel> CurrentConnections = new
HashSet<HubConnectionsModel>();

        [HubMethodName("transfernotifications")]
        public async Task Send(EmailNotificationModel emailModel)
        {
            var connectionId = GetAllActiveConnections()
                .Where(x => x.UserName == emailModel.UserId.ToString())
                .FirstOrDefault()
                .ConnectionId;

            await Clients.Client(connectionId)
                .SendAsync("transfernotifications", emailModel);
        }

        public override Task OnConnectedAsync()
        {
            HubConnectionsModel model = new HubConnectionsModel()
            {
                ConnectionId = Context.ConnectionId,
                UserName = Context.User.Identity.Name
            };
            CurrentConnections.Add(model);
            return base.OnConnectedAsync();
        }

        public override Task OnDisconnectedAsync(Exception exception)
        {
            var connection = CurrentConnections
                .FirstOrDefault(x => x.UserName ==
Context.User.Identity.Name);

            if (connection != null)
            {
                CurrentConnections.Remove(connection);
            }
            return base.OnDisconnectedAsync(exception);
        }

        //return list of all active connections
        public List<HubConnectionsModel> GetAllActiveConnections()
        {
            return CurrentConnections.ToList();
        }
    }
```

Slika 4.2.8. Programski kod čvorišta.

4.3. Vrednovanje web kalendara „.o“

Implementirani „.o“ kalendar odgovor je na zahtjeve postavljene u trećem poglavlju. Tijekom implementacije web kalendara „.o“ bilo je više problema, neki zahtjevi su ispunjeni u drugačijem obliku, a neke mogućnosti su ostvarene iako nisu bile u planu.

Prvi veći problem korisničkog sučelja je iskoristiti ranije spomenuti Angular Calendar. Isti omogućuje kalendarski i dnevni prikaz zadataka. Za programski okvir Angular postoji puno kalendarskih komponenti, neke od kojih su:

- angular ui calendar,
- angular bootstrap kalendar,
- ion2 kalendar,
- angular full calendar.

Od pronađenih kalendarskih izvedbi, Angular Calendar ima najbolju recenziju developera i najveće mogućnosti prilagodbe. Više o alternativnim izvedbama na [26].

Najveći problem programskog sučelja je implementiranje pozadinskih servisa i čvorišta obavijesti. Prvi veći bio je dependency injection servisa web kalendara „.o“. Sve operacije unutar pozadinskih servisa asinkrone su, ali prvo je testirana sinkrona izvedba. Uzeta su dva laptopa s izvedbom korisničkog sučelja. Na oba laptopa registrirani su odvojeni korisnički računi, a dodani identični zadaci. Metode dodavanja na red i uklanjanja su blokirajuće i ne poštuje se ciklus od 10 sekundi. Slanje jedne e-pošte zbog potvrde pružatelja SMTP usluge zna potrajati i dvije do tri sekunde. Postavljanjem asinkronih metoda i reda isti problem je riješen. Za više korisnika i više zadataka može se pojaviti problem gdje se red nikada ne isprazni. Za isti je onda moguće povećati broj programskih niti u bazenu i povećati period ponavljanja dovoljno da korisniku obavijest stigne na vrijeme.

Zahtjevi nalažu slanje e-pošte za svaki veći događaj – promjena, stvaranje, brisanje i uređivanje zadataka. Slanje e-pošte za svaki navedeni događaj stvorio bi mnoštvo pošte i stvorio nered u ulaznom spremniku korisnika. Zato se obavijesti šalju samo za početak događaja.

5. ZAKLJUČAK

Od početka civilizacije ljudima se nameće problem organizacije vremena. Prije više tisuća godina ljudi su planirali kada saditi usjeve, a danas planiraju gotovo sve događaje jer vrijeme postaje sve dragocjenije. Danas je jedna od najviše korištenih vrsta kalendara web kalendar, s obzirom da je ljudima stalno dostupan. To je i jedan od razloga zašto je u sklopu ovog rada bilo potrebno izraditi interaktivni web kalendar s mogućnostima stvaranja i upravljanja zadacima.

U današnje vrijeme postoji već mnogo izvedbi web kalendara od kojih se neka razlikuju više, a neka manje. Dostupni kalendari mogu biti namijenjeni za osobne i/ili poslovne obaveze. Osim prema namjeni, kalendari se razlikuju prema vrsti prikaza, prema dostupnosti i prema opsegu. U radu su navedeni primjeri kalendara za svaku podjelu. Od navedenih kalendara neki su više, a neki manje odgovarali prethodno navedenim zahtjevima.

Od zahtjeva na web kalendar koji je trebao biti izrađen u sklopu ovog rada bitno je istaknuti da mora imati podršku za više korisnika koji mogu imati različite ovlasti. Također, potrebna je podrška za prikaz zadataka od kojih svaki ima svoj period da kada treba biti riješen. Kako korisnik ne bi zaboravio na zadatak treba mu slati obavijest putem e-pošte. Zadaci se trebaju razlikovati po bojama ovisno o vremenu isteka istih.

Kao odgovor na navedene zahtjeve, u okviru ovog diplomskog rada napravljen je interaktivni web kalendar „.o“. Navedeni je kalendar hibridna izvedba te sadrži funkcionalnosti koje sadrže neki od danas najpopularnijih web kalendara, ali uz dodatne funkcionalnosti i dorade kako bi ispunio postavljene zahtjeve. Nisu svi zahtjevi ostvareni na postavljeni način zbog zadržavanja jednostavnosti i funkcionalnosti programske izvedbe.

Programska izvedba „.o“ sastoji se od dva glavna dijela: korisničkog sučelja ostvarenog u programskom okviru Angular te programskog sučelja ostvarenog u okviru ASP.NET Core. Korisnik ima mogućnost registracije i prijave, a njegovi podaci osigurani su kriptiranjem lozinke i korištenjem JWT-a.

Od izdvojenih funkcionalnosti korisnik ima mogućnost pregleda zadataka u obliku kalendara i kao vremenske crte za svaki pojedini dan. Korisnik s većim mogućnostima može zadatke vidjeti kao listu u tablici. Ista omogućuje pretraživanje zadataka prema bilo kojem od parametara koji ih opisuju te ih može po istima i sortirati. Također, korisniku je dostupan pregled statistike zadataka prema dva kriterija: obavljenost i duljina trajanja. Korisnik može uređivati vlastite podatke i birati kakve obavijesti želi primati. Obavijesti mogu biti u obliku podsjetnika putem e-pošte ili

podsjetnika unutar korisničkog sučelja web kalendara. Najveći dio prethodno navedenih funkcionalnosti implementiran je u okviru ASP.NET Core u jeziku C#. Programsko sučelje sastavljeno je od dva dijela: pozadinski API i pozadinski servisi. Pozadinski API ima svrhu posrednika između korisničkog sučelja i baze podataka. Sadrži metode za ovjeru autentičnosti i autorizaciju korisnika. Pozadinski servisi okidaju se periodički. Svakih 10 sekundi provjeravaju se zadaci u bazi podataka po kriteriju aktualnosti. Zadacima se osvježavaju boje, a oni koji počinju unutar perioda koji je korisnik odredio dodani su u red za slanje obavijesti. S obzirom na svojstva očekuje se korištenje web kalendara „.o“ u privatne svrhe, no nije isključena poslovna namjena za samo jednu osobu. Kada bi se web kalendar razvijao za komercijalne svrhe bilo bi potrebno zakupiti poslužiteljski stroj i domenu za korisnički pristup.

LITERATURA

[1] N. Marie: The history of western calendar [online], TimeCenter, Helsingborg, Švedska, dostupno na: https://www.timecenter.com/articles/the-history-of-the-western-calendar, [pristupljeno 10.5.2020.]

[2] Toggl tim: Toggl planner [online], Toggl, dostupno na: https://plan.toggl.com, [pristupljeno 10.5.2020.]

[3] Google: Google calendar [online], Google, dostupno na: https://calendar.google.com, [pristupljeno 14.5.2020.]

[4] Dayviewer tim: Dayviewer calendar [online],Dayviewer, dostupno na: https://dayviewer.com, [pristupljeno 14.5.2020.]

[5] Canva tim: Canva planner [online], Canva, dostupno na: https://www.canva.com, [pristupljeno 14.5.2020.]

[6] MessageOps tim: Use Microsoft planner to streamline your work day [online], MessageOps, https://messageops.com/use-microsoft-planner-to-streamline-your-work-day/, [pristupljeno 16.5.2020.]

[7] I. Šuta, ASP.NET Core [online], Codingblast, 2018., https://codingblast.com/why-asp-net-core/, [pristupljeno 11.6.2020.]

[8] Microsoft, Visual Studio [online], Microsoft, https://visualstudio.microsoft.com/, [pristupljeno 25.5.2020.]

[9] Microsoft, Visual Studio Code [online], Microsoft, https://code.visualstudio.com/, [pristupljeno 25.5.2020.]

[10] Microsoft, .NET Core[online], https://dotnet.microsoft.com/, Microsoft, [pristupljeno 25.5.2020.]

[11] Microsoft, ASP.NET Core [online], https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-3.1, Microsoft, [pristupljeno 25.5.2020.]

[12] Google i kolaboratori, Angular [online], 2010., https://angular.io, Google, [pristupljeno 25.5.2020.]

[13] M. Lewis,, Angular Calendar [online], 2016., https://www.npmjs.com/package/angular-calendar, NPM, [pristupljeno 31.5.2020.]

[14] Google i kolaboratori, Angular Materials – table [online], 2010.,https://material.angular.io/components/table/overview, Google, [pristupljeno 31.5.2020.]

[15] Microsoft, Entity Framework Core [online], 2017., https://docs.microsoft.com/en-us/ef/core/managing-schemas/migrations, Microsoft, [pristupljeno 31.5.2020.]

[16] T. Tam, Swagger [online], 2011., https://swagger.io/, Swagger, [pristupljeno 31.5.2020.]

[17] M. Rouse, RESTful API [online], https://searchapparchitecture.techtarget.com/definition/RESTful-API, Tech Target, [pristupljeno 11.6.2020.]

[18] Microsoft, HMAC SHA512 [online], https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.hmacsha512?view=netcore-3.1, Microsoft, [pristupljeno 1.6.2020.]

[19] Auth0, JSON Web Token [online], https://jwt.io/introduction/, Auth0, [pristupljeno 2.6.2020.]

[20] Microsoft, Dependency Injection [online], https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection, Microsoft, [pristupljeno 2.6.2020.]

[21] Microsoft, Concurrent Queue [online], https://docs.microsoft.com/en-us/dotnet/api/system.collections.concurrent.concurrentqueue-1?view=netcore-3.1, Microsoft, [pristupljeno 2.6.2020.]

[22] Microsoft, Semaphore Slim [online], https://docs.microsoft.com/en-us/dotnet/api/system.threading.semaphoreslim?view=netcore-3.1, Microsoft, [pristupljeno 2.6.2020.]

[23] Carnet, SMTP [online], 2006., https://www.cert.hr/wp-content/uploads/2006/08/CCERT-PUBDOC-2006-05-159.pdf, Carnet, [pristupljeno 3.6.2020.]

[24] Google, Send email from printer scanner, or app [online], https://support.google.com/a/answer/176600?hl=en, Google, [pristupljeno 3.6.2020.]

[25] Microsoft, SignalR hub [online], https://docs.microsoft.com/en-us/aspnet/core/signalr/hubs?view=aspnetcore-3.1, Microsoft, [pristupljeno 4.6.2020.]

[26] Openbase, Angular calendars [online], 2019., https://openbase.io/packages/top-angular-calendar-libraries, Openbase, [pristupljeno 6.6.2020.]

SAŽETAK

U današnje vrijeme sve veća je potreba za organizacijom vremena zbog velikog broja obaveza. U tu svrhu razvijena su mnoga rješenja i tehnološka pomagala. Jedni od najpopularnijih takvih rješenja su web kalendari i planeri koji postoje u raznim oblicima i formama.

U sklopu ovog diplomskog rada bilo je izraditi web kalendar koji korisniku omogućuje interaktivnu organizaciju svoga vremena. Web kalendar korisniku mora dati potpunu slobodu mijenjanja zadataka kao i stvaranja istih. Korisnik u svakom trenutku treba moći pristupiti pregledu statistike zadataka koja je prema dva kriterija: obavljenost i duljina trajanja. Za početak svakog zadatka korisnik mora biti obaviješten pravovremeno kako bi reagirao na odgovarajući način.

Kao rezultat je ostvaren web kalendar „.o“ koja u potpunosti odgovara postavljenim zahtjevima. Web kalendar „.o“ moguće je koristiti u privatne i u poslovne svrhe.

Ključne riječi: Angular, ASP.NET Core, web kalendar, .o

ABSTRACT

There is a great effort nowadays regarding organizing time schedule because of a rising number of responsibilities. As a result, there are many solutions and technological aids which help people to organize their time. Most popular of those aids are web calendars and planners.

As a part of this thesis, a web calendar was supposed to be made. This calendar has to enable interactive time organization to the user. User must be able to edit tasks, as well as to create new ones. User also must be able to access statistics of the tasks according to two criteria: task completion and task duration. User should be notified about the upcoming task in a timely manner, in order to be able to react to the notification accordingly.

As a result, application ".o" was made, which entirely fulfils given requirements. Application can be used for private or business purposes.

Key words: Angular, ASP.NET Core, web calendar, .o

ŽIVOTOPIS

Filip Begić rođen je 13. Studenoga 1995. Godine u Požegi, gdje trenutno i živi. Završio je Osnovnu školu Antuna Kanižlića u Požegi, nakon čega upisuje Gimnaziju u Požegi, smjer Prirodoslovno - matematički koju završava 2014. godine. Iste godine upisuje preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Preddiplomski studij završava 2017. godine te upisuje diplomski studij računarstva na istom fakultetu. Sudjeluje na natjecanju STEM Games 2018. i 2019. godine u znanju iz područja inženjerstva.

PRILOZI

[P1] DVD s programskim kodom

[P2] URL Git repozitorija s programskim kodom,

Korisničko sučelje - https://github.com/TruliParadajz/DiplomskiFront

Programsko sučelje - https://github.com/TruliParadajz/DiplomskiBack