Šolski center Novo mesto

Srednja elektro šola in tehniška gimnazija

Šegova ulica 112

8000 Novo mesto

**Aplikacija za upravljanje s delovnimi nalogi**

(Izdelek oziroma storitev in zagovor)

Avtor: Luka Žohar, r4.a

Mentor: Simon Vovko, naziv

Kanižarica, februar 2019

Povzetek in ključne besede

V tem projektnem delu sem naredil aplikacijo, ki deluje na 'full stack' tehnologijah. Te so:

* **Front-end tehnologije**:
  1. Angular,
* **Back-end tehnologije**:
  1. Node.js,
  2. Express.js,
  3. Mongoose,
* **Podatkovne baze**:
  1. MongoDB.

Vse tehnologije bom v teoretičnem delu opisal njihovo zgodovino, za kaj so namenjene, za kaj jih lahko uporabimo, v praktičnem delu bom pa še predstavil, na kakšen način sem jih uporabil v svoji aplikaciji.

**Ključne besede:** spletna aplikacija, delovni nalog, angular, node.js, express.js, mongoose, mongodb, javascript, typescript, html, css, podatkovna baza.

200 do 250 besed

Kazalo

Vsebina

[1 Uvod 1-1](#_Toc7170006)

[2 Jedro naloge 2-2](#_Toc7170007)

[2.1 Teoretični del 2-3](#_Toc7170008)

[2.1.1 Angular 2-4](#_Toc7170009)

[2.1.2 Node.js 2-5](#_Toc7170010)

[2.1.3 Express.js 2-6](#_Toc7170011)

[2.1.4 MongoDB in NOSQL 2-7](#_Toc7170012)

[2.1.5 Mongoose 2-8](#_Toc7170013)

[2.1.6 Ostale NPM knjižnice 2-9](#_Toc7170014)

[2.2 Praktični del 2-10](#_Toc7170015)

[2.2.1 Registracija 2-10](#_Toc7170016)

[2.2.2 Prijava 2-10](#_Toc7170017)

[2.2.3 Upravljanje s računom 2-10](#_Toc7170018)

[2.2.4 Dodajanje in upravljanje s delovnim nalogom 2-10](#_Toc7170019)

[3 Zaključek 3-11](#_Toc7170020)

[4 Sklici 4-12](#_Toc7170021)

[5 Priloge 5-13](#_Toc7170022)

**Nobenega vnosa v kazalo slik ni bilo mogoče najti.**

**Nobenega vnosa v kazalo slik ni bilo mogoče najti.**

# Uvod

Ker imamo doma podjetje, ki se ukvarja z vezenjem na tekstil, sem bil v naše majhno podjetje vključen že pred leti. Do sedaj smo imeli vedno delovne naloge le v fizični obliki, ki so se lahko hitro izgubili, poškodovali, ni bilo neke stalnice glede tega, če je delovni nalog narejen ali ne,… Zato sem se v začetku 4. letnika srednje šole odločil, da bom naredil aplikacijo za upravljanje delovnih nalogov v elektronski obliki.

# Jedro naloge

## Teoretični del

Ko sem začel delati aplikacijo, se na temo razvoja spletnih aplikacij spoznal, a ne dovolj da bi znal samostojno narediti aplikacijo težavnosti, ki sem si jo zastavil.

Za razvijanje aplikacije sem najprej za 'front-end' uporabil Angular, 'back-end' pa ASP.NET orodje ter MSSQL podatkovno bazo ter Entity Framework za upravljanje z podatkovno bazo. Naredil sem celoten ER model podatkovne bazo, ki mi ga je uspelo narediti še kar dobro. A ko sem začel bazo dejansko ustvarjati ter poskušal s podatki upravljati preko Entity Frameworka, sem ugotovil, da mi za to tehnologijo primanjkuje znanja na tem področju in sem vedel, da bom potreboval nekaj drugega za izdelavo aplikacije in sem začel iskati alternativo.

Našel sem jo, ko sem brskal po platformi YouTube našel video, ki je pokazal kako uporabljati Node.js s 'frameworkom' Express.js ter NOSQL podatkovno bazo MongoDB. Tehnologije so mi bile zelo všeč , ker so bile preproste za uporabo, a niso bile nič slabše od prejšnjih, le da je bilo potrebne manj kode, zato sem se odločil, da jih bom uporabil.

\*Mean stack slika\*

Glavne tehnologije, ki sem jih uporabil :

* Angular
* Node.js in Express
* MongoDB
* Mongoose
* Web token
* Uporaba HTTP protokola

### Angular

Ko se podamo v vode spletnih aplikacij, po navadi začnemo s osnovami spletnih strani, tj. HTML, CSS, nekaj osnovnih funkcionalnosti jezika JavaScript, a ko želimo izdelovati kompleksnejše aplikacije se v njih lahko hitro 'zgubimo'. Zato se je dobro naučiti 'framework' ali knjižnice, saj nam olajšajo in pohitrijo delo. Na voljo jih je veliko, od osnovnega jQuery-a, do kompleksnejših, kot so React, Vue, Meteor in Angular. Med temi sem se osebno odločil za Angular, saj so mi ga priporočili ter mi je všeč.

**Zgodovina:** 'Framework'Angular obstaja od leta 2010, ko so ga razvili v podjetju Google. Prvi izid se je imenoval AngularJS, saj je bil programski jezik, ki je bil uporabljen v njem, JavaScript. To je trajalo do leta 2014, ko je podjetje 'framework' zelo spremenilo. Odločilo se je, da bodo v njem uporabili programski jezik TypeScript, ki je programski jezik s statičnimi tipi, veliko funkcionalnosti so premaknili v module,… Od takrat posodobitve prihajajo na približno pol leta z novostmi, ki ga še poboljšajo.

Angular deluje na principu MVC (view model controller) modela, v katerem je v tem primeru view HTML stran, model podatki in controller - funcionalnost spletne strani, ki upravlja s podatki (TypeScript). (Wikipedia, 2019)

### Node.js

Če želimo izdelati spletne aplikacije, ki imajo funkcionalnost shranjevanja, posodabljanja, dodajanja in brisanja podatkov oziramo so dinamične, potrebujemo aplikacijo, ki deluje na strežniku in lahko upravlja s podatkovnimi bazami ali datotekami. To dosežemo z 'backend' tehnologijami, ki so tega zmožne. Te tehnologije so na primer Laravel za PHP, .NET za C# in ostale. Na 'back-endu' lahko s podatki upravljamo s obdelovanjem le-teh (na primer računanje veljavnosti vnesene forme), zapisanjem v tabel,… Front-end je po navadi povezan preko API-ja, ki deluje preko HTTP protokola.

Node.js je spletni strežnik, ki izvaja JavaScript kodo zunaj brskalnika. JavaScript se primarno uporablja za skripte v brskalniku, a se s pomočjo Node.js tehnologije lahko uporablja tudi na strežniku. Čeprav obstaja šele od leta 2009, v njem prednosti vidi veliko velikih podjetij, kot so PayPal, Microsoft, Amazon, Walmart in ostali. S Node.js strežnikom po navadi uporabljamo NPM (node packet manager), ki upravlja s paketi, ki jih dobimo s knjižnice paketov. Te paketki nam delo močno skrajšajo, saj zmanjšajo kodo, ki bi jo sicer morali napisati. Na primer lahko uporabimo knjižnico Generate-Password, ki nam generira naključno geslo. Če tega paketa nebi uporabili, bi morali napisati veliko kode, ki bi nam vzela veliko časa ter še mogoče nebi bila zanesljiva. V njem lahko pišemo v vseh jezikih, ki se prevedejo v JavaScript (TypeScript, Dart, CoffeScript,…).

### Express.js

Programiranje v Node.js-u lahko s povečevanjem velikosti in težavnosti lahko postane zamudno in manj pregledno, posledično za dodajanje funkcionalnosti projekta porabimo več časa in v primeru, da projekt zapustimo in pride za nami nekdo, ki kodo napisano v samem Node-u se bo verjetno težko ali se pa sploh ne bo znašel. Zato so po objavi Node-a začeli razvijati 'frameworke', ki nam delo zelo pohitrijo in polepšajo. Poznamo jih več, najbolj popularni so pa Koa2, Sails, Nest, Meteor in veliko ostalih, daleč najpopularnejši je pa Express.js, ki sem ga uporabil v projektu.

Express.js je minimalističen 'framework', ki ga uporabljamo s Node-om. Prva verzija je bila izdana maja 2010. Najbolj je uporabljen v kombinaciji s MEAN (Mongo Express Angular Node) in MERN (MongoDB Express React Node). Pohitri nam veliko zadev s Node-om, na primer za zagon strežnika s Node-om potrebujemo več vrstic kode kot s Express-om.

### MongoDB in NOSQL

Ko izdelujemo spletne aplikacije, ki shranjujejo podatke, prej ali slej pridemo to točke, ko želimo podatke shranjevati v podatkovni bazi, saj tako podatke shranimo trajno. Vrst podatkovnih baz poznamo več, najbolj popularne so entitetno-relacijske oziroma baze, ki za upravljanje s njimi potrebujejo jezik SQL (MySql, MSSQL, PostgreSQL, MariaDB,…) ter NOSQL podatkovne baze.

**NOSQL podatkovne baze**

Take vrste baz delujejo na sistemu dokumentov. Razlike med NOSQL in SQL bazami lahko razložimo na praktičnem primeru oblike shranjenih podatkov.

**Shranjevanje podatkov v SQL bazah**

SQL baze si lahko predstavljamo kot da imamo več predalov, v katerih shranjujemo podatke o več osebah. V tem primeru je en predal ena tabela. Podatki shranjeni v teh predalih so napisani v enem dokumentu, podatki med predali so pa povezani s indeksi.

**Shranjevanje podatkov v NOSQL bazah**

Shranjevanje podatkov NOSQL bazah lahko na praktičnem primeru predstavimo tako, da imamo za vsako osebo (dokument) svoj predal, v katerega se shranjujejo vsi podatki v njem. Ta način se razlikuje od SQL-a v tem, da ne rabimo po določen podatek iskati v drug predal, kar je bolj pregledno. Najboljši primer je dokumentiranje pacientov pri zdravniku, kjer ima vsak svojo mapo, v kateri so shranjeni vsi podatki o tej osebi, njegovi pregledi, operacije in ostalo.

**MongoDB**

MongoDB je NOSQL podatkovna baza, ki deluje na sistemu shranjevanja dokumentov. Je med najpopularnejšimi podatkovnimi bazami, saj je bil na platformi StackOverflow četrta najbolj uporabljena. Že od začetka je brezplačna. Razvilo in objavilo ga je podjetje MongoDB Inc. februarja 2009. Podatki so shranjeni v formatu, podobnem JSONu, kar omogoča fleksibilnost sheme dokumenta. Model dokumenta se lahko 'mapira' kot objekt, zato je delo s podatki lahko. Za upravljanje z njo lahko uporabljamo konzolo, v katere vpisujemo poizvedbe, MongoDB Compass, ki je grafična namizna aplikacija, priročna če imamo veliko dokumentov, saj si podatke lažje prikažemo grafično. MongoDB Atlas deluje kot podatkovna baza v oblaku, podobno kot Microsoftov Azure.

### Mongoose

Mongoose.js ali bolj uporabljeno Mongoose je orodje za objektno modeliranje, ki ga uporabljamo za modeliranje sheme za interakcijo s MongoDB podatkovno bazo.

### Ostale NPM knjižnice

1. **materialize-css**

Materialize je framework za enostavno grafično izdelovanje spletnih strani. Deluje podobno kot Bootstrap, ki je sicer bolj popularen.

1. **generate-password**

To knjižnico sem uporabil zato, da sem pri registraciji generiral naključna varna gesla, če bi uporabnik hotel naključno generirano geslo, ki je varno.

1. **bcryptjs**

bcryptjs je knjižnica, ki sem jo uporabil za haširanje gesel ter odhaširanje le teh za preverjanje veljavnosti vnesenega gesla.

1. **jsonwebtoken**

Knjižnica, ki se uporablja za avtentkacijo uporabnikov. Knjižnica generira kodirano besedilo, v katerem so podatki o uporabniki in o načinu kodiranja besedila.

1. **nodemailer**

Uporablja se za pošiljanje e-pošte. Uporabil sem za pošiljaje pošte v primeru, če uporabnik pozabi geslo.

1. **Passport in passport-jwt**

Passport je knjižnica za avtorizacijo uporabnika, ki ima več načinov avtorizacije. V mojem primeru sem uporabil avtorizacijo s passport-jwt, ki preveri veljavnost tokena, poslanega v HTTP zahtevku.

## Praktični del

V tem delu bom predstavil področja, ki sem jih razvil.

### Registracija

Ker je aplikacija narejena da trajno shranjuje podatke sem moral narediti registraciji, ki na podatkovni bazi naredi prostor kjer se shranjujejo podatki o uporabniku. Pri registraciji je potrebno vpisati uporabniško ime, elektronski naslov, geslo, ime in priimek. Zahteve za veljavnost forme so:

1. Ime in priimek :
   1. Minimalna dolžina je 2 znaka,
   2. Veljavni znaki so male in velike črke od a do ž,
   3. Najdaljša dolžina je 100 znakov.
2. Uporabniško ime :
   1. Minimalna dolžina so 3, najdaljša pa 100 znakov
   2. Veljavno znaki so male in velike črke, številke od 0 do 9 ter znaka \_ in –
   3. Če je uporabniško ime na voljo preverim s HTTP poizvedbo na strežnik, ki odgovori s true ali false. Glede na odgovor se nastavi veljavnost up. imena ter se uporabnika obvesti
3. Elektronski naslov
   1. Veljavnost naslova se preveri s reggex preverjanjem
   2. Če je na voljo se preveri enako kot pri up. imenu, le da preveri elektronski naslov
4. Geslo
   1. Minimalna dolžina je 8 znakov
   2. Mora vsebovati majhno in veliko črko ter številko

**Postopek registracije :**

1. Uporabnik vnaša podatke, pri tem brskalnik sproti preverja veljavnost podatkov. Ko je forma veljavna, uporabnik klikne na gumb za registracijo,
2. Brskalnik pošlje POST zahtevek na strežnik, ki v bazo vpiše novega uporabnika s podanimi podatki in zakriptiranim geslom,
3. Strežnik odgovori s sporočilom ter indeksom novo ustvarjenega uporabnika.

Front-end kode

http poizvedba

backend registracija (add user)

### Prijava

Da uporabnik dobi že podane podatke se mora prijaviti. Za to potrebuje uporabniško ime ali elektronski naslov ter geslo. Če uporabniško ime ali geslo ni pravilno, se uporabnika o tem obvesti. Če je vse pravilno dobi žeton (JWT Token), s katerim lahko dostopa do funkcij, ki so za uporabnike brez njega nedostopni, kot so na primer dodajanje nove podlage ali delovnega lista, saj bi brez avtentikacije lahko vsak upravljal s podatki uporabnika.

**Postopek prijave :**

1. Uporabnik vpiše uporabniško ime ali elektronski naslov ter geslo
2. Brskalnik pošlje http zahtevo na strežnik
3. Strežnik preveri, če obstaja podano up. ime ali el. naslov, če da preveri še, če se ujemata gesla, če se ne odgovori da so podani podatki napačni
4. Če sta podatka pravilna, strežnik odgovori s sporočilom, novo ustvarjenim žetonom, ki je veljaven 14 dni ter vsemi podatki o uporabniku, razen gesla.
5. Brskalnik shrani žeton in podatke ter uporabnika pošlje na začetno stran, kjer lahko zdaj uporablja aplikacijo.

### Upravljanje s računom

### Dodajanje in upravljanje s delovnim nalogom

#### Podlaga

##### Dodajanje nove podlage

##### Urejanje podlag

#### Delovni nalog

##### Dodajanje novega delovnega naloga

# Zaključek

# Sklici

*Wikipedia*. (19. 4 2019). Pridobljeno iz Angular (web framework): https://en.wikipedia.org/wiki/Angular\_(web\_framework)

# Priloge

Dodaj datoteke aplikacije brez NPM modulov.