

Univerzitet u Sarajevu  
Elektrotehnički fakultet Sarajevo



**Završni rad – praksa**

**Stručni studij – Razvoj softvera**

**Web aplikacija za upravljanje uposlenicima**

**Student:**

Bakir Karović

**Mentor:**

Doc. dr Vedran Ljubović, dipl.ing.el.

---

Sarajevo, septembar 2020. godine

Univerzitet u Sarajevu  
Elektrotehnički fakultet Sarajevo  
Stručni studij: Razvoj softvera

**Postavka zadatka završnog rada stručnog studija:**  
Web aplikacija za upravljanje uposlenicima

U radu je potrebno:

- opisati šta je sve rađeno u sklopu industrijske prakse,
- navesti koje su sve stvari naučene iz industrijske prakse,
- dati tehnički izvještaj o realiziranom projektu u okviru industrijske prakse

Sarajevo, .....

Potpis:

---

Mentor: Doc. dr Vedran Ljubović, dipl.ing.el.

## **Izjava o autentičnosti radova**

Završni rad  
Stručnog studija

Ime i prezime: Bakir Karović  
Naslov rada: Web aplikacija za upravljanje uposlenicima  
Vrsta rada: Završni rad stručnog studija  
Broj stranica: 35

### **Potvrđujem:**

- da sam pročitao dokumente koji se odnose na plagijarizam, kako je to definirano Statutom Univerziteta u Sarajevu, Etičkim kodeksom Univerziteta u Sarajevu i pravilima studiranja koja se odnose na stručni studij, I i II ciklus studija, integrirani studijski program I i II ciklusa i III ciklus studija na Univerzitetu u Sarajevu, kao i uputama o plagijarizmu navedenim na web stranici Univerziteta u Sarajevu;
- da sam svjestan univerzitetskih disciplinskih pravila koja se tiču plagijarizma;
- da je rad koji predajem potpuno moj, samostalni rad, osim u dijelovima gdje je to naznačeno;
- da rad nije predat, u cjelini ili djelimično, za stjecanje zvanja na Univerzitetu u Sarajevu ili nekoj drugoj visokoškolskoj ustanovi;
- da sam jasno naznačio prisustvo citiranog ili parafraziranog materijala i da sam se referirao na sve izvore;
- da sam dosljedno naveo korištene i citirane izvore ili bibliografiju po nekom od preporučenih stilova citiranja, sa navođenjem potpune reference koja obuhvata potpuni bibliografski opis korištenog i citiranog izvora;
- da sam odgovarajuće naznačio svaku pomoć koju sam dobio pored pomoći mentora i akademskih tutora/ica.

Sarajevo, .....

Potpis:

---

Bakir Karović

## Sadržaj

Lista slika .....	6
Lista primjera .....	6
Lista isječaka kôda .....	7
Sažetak .....	8
1. Uvod.....	9
1.1 O Sass-u.....	9
1.1.1 Sass varijable.....	10
1.1.2 Nesting (ugnježdavanje).....	10
1.1.3 Extend .....	10
1.1.4 Parcijale.....	11
1.1.5 Operatori .....	11
1.2 O Compass-u.....	12
1.2.1 Instalacija Compass-a.....	12
1.2.2 Kreiranje radne .css datoteke .....	12
1.2.3 Mixins .....	13
1.3 O Typescript-u.....	14
1.3.1 Statički tipovi .....	14
1.3.2 Interfejsi .....	14
1.3.3 Klase.....	15
1.3.4 Konverzija tipova.....	16
1.4 O Angular-u.....	16
1.4.1 Instalacija Angulara .....	16
1.4.2 Visual studio code setup.....	17
2 Korisnički zahtjevi projekta.....	18
2.1 Početna stranica.....	18

2.2	Meni .....	18
2.3	Upravljanje uposlenicima.....	18
2.4	Nadogradnja u budućnosti.....	19
2.5	Internacionalizacija i lokalizacija.....	19
3	Izgradnja projekta .....	20
3.1	Build menija iz JSON-a.....	20
3.2	Sidenav i toolbar.....	21
3.3	Rutiranje .....	23
3.4	Tabela uposlenika.....	24
3.4.1	Tema tabele .....	25
3.4.2	Definisanje kolona .....	25
3.5	Brisanje i dodavanje redova tabele.....	26
3.6	Inline datepicker.....	27
3.7	Filtriranje datuma .....	28
3.8	Sortiranje datuma .....	28
3.9	Validacija .....	29
4	Frontend izgled .....	31
4.1	Početna stranica.....	31
4.2	Ekran za prikaz uposlenika .....	32
	Zaključak.....	33
	Bibliografija .....	34

## Lista slika

Slika 1 – izgled automatski generisanog početnog projekta .....	16
Slika 2 – lokacija tslint.json datoteke na root novou projekta .....	17
Slika 3 – JSON prikaz postavki .....	17
Slika 4 – prikaz početnog ekrana sa otvorenim menijem .....	31
Slika 5 - prikaz ekrana za upravljanje uposlenicima sa zatvorenim menijem .....	32

## Lista primjera

Primjer 1 – izgled .scss datoteke .....	9
Primjer 2 – izgled .css datoteke.....	9
Primjer 3 – korištenje Sass varijabli.....	10
Primjer 4 – Sass ugnježdavanje.....	10
Primjer 5 – Izlaz Sass ugnježdavanja.....	10
Primjer 6 – Sass naslijeđivanje .....	11
Primjer 7 – izlaz Sass naslijeđivanja.....	11
Primjer 8 – korištenje Sass operatora.....	11
Primjer 9 – deklaracija i korištenje mixina .....	13
Primjer 10 – definisanje mixin-a.....	13
Primjer 11 – deklaracija varijabli u typescript-u.....	14
Primjer 12 – deklaracija funkcije i opcionalnih parametara .....	14
Primjer 13 – lambda funkcije.....	14
Primjer 14 – implementacija interfejsa unutar funkcije .....	15
Primjer 15 – definisanje klase u typescriptu .....	15
Primjer 16 – konverzija tipova.....	16

## Lista isječaka kôda

Isječak kôda 1 – klasa meni .....	20
Isječak kôda 2 - import JSON objekata.....	20
Isječak kôda 3 – HTML templejt koji kreira pojedinačne stavke menija .....	21
Isječak kôda 4 – templejt koji prikazuje komponentu koja je kliknuta u meniju .....	21
Isječak kôda 5 – tempjelt za pomjeranje teksta.....	21
Isječak kôda 6 – CSS koji prikazuje ikone menija kada je zatvoren .....	22
Isječak kôda 7 – deklaracija lokalne reference na tempjelt tag sa # tagom .....	22
Isječak kôda 8 – korištenje @ViewChild dekoratora i inicijalizacija servisa.....	22
Isječak kôda 9 – označavanje ulaznog atributa .....	22
Isječak kôda 10 – prosljeđivanje reference na niz ikonica i meni .....	22
Isječak kôda 11 – app-routing modul sa putanjama na pojedinačne komponente .....	23
Isječak kôda 12 – anchor tag sa router linkom.....	23
Isječak kôda 13 – import mixin teme .....	25
Isječak kôda 14 – niz sa podacima tabele .....	26
Isječak kôda 15 – Font awesome ikona u HTML templejtu .....	26
Isječak kôda 16 – funkcija za brisanje redova u tabeli.....	26
Isječak kôda 17 – funkcija za dodavanje redova u tabelu .....	27
Isječak kôda 18 – modul za inline prikaz datepickera .....	27
Isječak kôda 19 – definicija kolone Hire date .....	28
Isječak kôda 20 – validacija za required polja .....	29
Isječak kôda 21 – dio modula za validaciju unosa broja telefona.....	30
Isječak kôda 22 – definicija kolone za broj telefona gdje se koristi cellEditor.....	30

## Sažetak

Prvi dio ovoga rada će opisati nove stvari koje su naučene kroz obavljenju praksu, dok će drugi dio služiti kao „user-guide“ odnosno detaljan opis onoga što je rađeno u sklopu industrijske prakse.

Cilj prakse je bio razviti jednostavnu web baziranu aplikaciju za upravljanje uposlenicima. Za razvoj aplikacije je korišten Angular 9 koji se koristio u kombinaciji sa Sass-om i Compass-om te Typescript-om.



## 1. Uvod

Cilj ovog poglavlja je da opiše nove naučene stvari o jezicima koji su korišteni u okviru industrijske prakse te da pokaže sam proces upoznavanja sa novim programskim jezikom.

### 1.1 O Sass-u

Sass je skraćenica za „*syntactically awesome style sheets*”. Sass je jezik koji se interpretira i kompajlira u CSS. Ono što on omogućava je puno lakši način pisanja CSS-a jer nam omogućava korištenje varijabli, ugnježdavanje, naslijeđivanje atributa.

Da bi instalirali Sass, nije nam potreban Ruby kao u prethodnim verzijama. Sve što je potrebno uraditi je preuzeti paket sa GitHub-a na linku<sup>1</sup>, raspakovati ga i dodati u PATH varijablu.

Drugi način zahtijeva instaliran NodeJs, i instalira se komandom ispod koja instalira Javascript implementaciju Sass-a koja je malo sporija.

```
npm install -g sass
```

Sljedeći korak je kreiranje „**scss**“ datoteke te pokretanjem Sass-a da sluša promjene u toj datoteci i kompajlira u „**css**“ datoteku.

```
sass --watch style.scss:style.css
```

Prilikom spašavanja „**style.scss**“ datoteke, „**style.css**“ datoteka se automatski ažurira.

```
$variable: 10px;
body {
  padding: $variable;
}
```

Primjer 1 – izgled .scss datoteke

```
body {
  padding: 10px;
}
```

Primjer 2 – izgled .css datoteke

---

<sup>1</sup> <https://github.com/sass/dart-sass/releases/tag/1.26.3>

### 1.1.1 Sass varijable

Sve css vrijednosti je moguće spasiti u varijable i koristiti ih na mjestima gdje se očekuje CSS vrijednost. Na taj način smanjujemo potrebu za hard-kodiranjem jer se za to brine Sass. Varijable je moguće miješati sa običnim CSS vrijednostima kao na primjeru ispod.

```
$primary-color: #ffcc00;
$element-lr-margin: 12px;

a {
  color: $primary-color;
}
header {
  margin: 20px $element-lr-margin;
}
```

Primjer 3 – korištenje Sass varijabli

### 1.1.2 Nesting (ugnježdavanje)

U Sass-u je moguće definisati stil selektora unutar drugog selektora bez ponavljanja stabla. Izlazni CSS kôd će ponavljati pomenuto stablo ali je lakše i razumljivije čitati Sass.

```
ul {
  list-style-type: none;
  li {
    display: inline;
    float: left;
    a {
      padding: 20px 80px;
    }
    &.subclass {
      margin: 0;
    }
  }
}
```

Primjer 4 – Sass ugnježdavanje

Izlaz u „css“ datoteku ovog kôda je:

```
ul { list-style-type: none; }
ul li { display: inline; float: left; }
ul li a { padding: 20px 80px; }
ul li.subclass { margin: 0; }
```

Primjer 5 – Izlaz Sass ugnježdavanja

### 1.1.3 Extend

Sass efektivno daje mogućnost naslijeđivanja drugih svojstava sa „@extend” oznakom. Na taj način preuzimamo sav stil selektora od drugog selektora, bez potrebe da ih pišemo ponovo, uz mogućnost dodavanja novih stilova.

```

.class1 {
  background-color: grey;
  border: 1px solid black;
  margin: 0 auto;
}
.class2 {
  @extend .class1;
  font-weight: bold;
}
.class3 {
  @extend .class2;
  font-size: large;
}

```

Primjer 6 – Sass naslijeđivanje

Izlaz u „css“ datoteku izgleda ovako:

```

.class1, .class2, .class3 { background-color: grey; border: 1px solid black;
margin: 0 auto; }
.class2, .class3 { font-weight: bold; }
.class3 { font-size: large; }

```

Primjer 7 – izlaz Sass naslijeđivanja

### 1.1.4 Parcijale

Standardno je da se Sass kôd odvaja u više datoteka koje se onda importuju u glavnu „app.scss“ datoteku. Na ovaj način razdvajamo različite opisne module i smanjujemo prenatrpanost jedne datoteke. Parcijale počinju sa donjom crtom i imaju istu ekstenziju „.scss“. Importujemo ih sa „@import“ i putanjom bez donje crte kod imena datoteke. Primjer za importovanje „\_partial.scss“ datoteke je:

```
@import "../partial";
```

Sass spaja importovane datoteke u jednu koju će poslati pretraživaču za razliku od CSS-a koji za svaku import datoteku šalje poseban zahtjev.

### 1.1.5 Operatori

Sass podržava više operatora za računanje. Cijeli spisak se može pronaći na linku<sup>2</sup>.

```

.body {
  height: (5px * 1 + 55px % 50) / 2;
}

```

Primjer 8 – korištenje Sass operatora

<sup>2</sup> <https://sass-lang.com/documentation/operators>

## 1.2 O Compass-u

Compass je Sass framework, dizajniran da ubrza i olakša stiliziranje stranica programerima. Compass predstavlja skup korisnih alata i najboljih praksi pisanja kôda za Sass. Compass je izgubio podršku od strane developera ali se i dalje ponekad koristi.

### 1.2.1 Instalacija Compass-a

Za instalaciju Compass-a je potreban Ruby koji se može preuzeti sa [lnika](#)<sup>3</sup>. Nakon toga, Compass instaliramo sa komandom ispod:

```
gem install compass
```

Za kreiranje projekta koristimo komandu ispod.

```
compass create projectname
```

Drugi način instalacije je korištenjem Foundation framework-a koji zahtijeva NodeJS i Ruby. Foundation se instalira sa dvije komande date ispod.

```
npm install -g bower grunt-cli  
gem install foundation
```

Sljedeći korak je kreiranje projekta.

```
foundation new projectname
```

Dalje je potrebno instalirati sve biblioteke unutar projekta koje se koriste sa komandama ispod. Treća komanda je bitna jer bez nje se ne može pravilno pokrenuti compass watch.

```
gem install bundler  
bundle  
bundle update
```

Pokretanje Compass-a koje nadgleda projekat radimo sa:

```
bundle exec compass watch
```

Sav Sass kôd koji pišemo u „[scss/app.scss](#)“ će se prevesti u CSS i dodati na kraj datoteke „[stylesheets/app.css](#)“.

Ukoliko je compass instaliran korištenjem Foundation frameworka koji omogućava kreiranje responzivnog grid prikaza stranice, potrebno je uključiti dodatni script tag za migraciju Jquery-ja na stariju verziju poslije osnovnog Jquery tag-a u HTML datoteci. Ovo je potrebno jer Compass zahtijeva staru verziju Jquery-a jer je star i napušten od strane developera.

```
<script src="https://code.jquery.com/jquery-migrate-1.4.1.min.js"></script>
```

### 1.2.2 Kreiranje radne .css datoteke

Moguće je napraviti radnu verziju „[css](#)“ datoteke bez blanko znakova koristeći Compass pokretanjem komande ispod. Kao izlaz je „[css](#)“ datoteka bez blanko znakova u jednoj liniji kôda.

---

<sup>3</sup> <https://rubyinstaller.org/downloads/>

```
compass compile -e production --force
```

### 1.2.3 Mixins

Mixins omogućavaju korištenje više CSS svojstava odjednom bez da ih moramo svaki put pisati ponovo.

```
@mixin box-shadow($arguments) {  
    box-shadow: $arguments;  
    -webkit-box-shadow: $arguments;  
    -moz-box-shadow: $arguments;  
}  
  
header {  
    @include box-shadow(blue 4px 4px 12px);  
}
```

Primjer 9 – deklaracija i korištenje mixina

Compass već unaprijed daje mixine koje se mogu naći na stranici<sup>4</sup> i koje treba importovati pomoću „@import“ taga.

Mixins se mogu gledati kao funkcija koja prima argumente koji mogu biti i opcionalni odnosno mogu imati unaprijed zadanu vrijednost.

```
@mixin background-def($image, $x: 50%, $y: 50%) {  
    background: {  
        image: $image;  
        repeat: no-repeat;  
        position: $x $y;  
    }  
}
```

Primjer 10 – definisanje mixin-a

---

<sup>4</sup> <http://compass-style.org/reference/compass/css3/>

## 1.3 0 Typescript-u

Typescript predstavlja ekstenziju na Javascript te podržava sav Javascript kôd kao i dodatne mogućnosti poput statičnih tipova. Ova znači da je moguće pisati i NodeJS backend kroz Typescript.

Typescript se instalira koristeći „Node package manager“ sljedećom komandom:

```
npm install -g typescript
```

### 1.3.1 Statički tipovi

Statički tipovi se definiraju sa dodavanjem „:tip“ poslije imena varijable. Typescript podržava i Javascript način deklaracije bez implicitnog zadavanja tipa. Varijable koje su deklarirane sa „declare“ se mogu koristiti na nivou cijele aplikacije. Varijable čiji tip još nije definiran su tipa „any“. Od svih tipova je moguće kreirati nizove.

```
declare var temp;
var tekst: string = 'primjer';
var tekst2 = 'js way';
var nepoznato: any;
var names: string[] = ['John'];
const kolicina: boolean = null;
let house: number = undefined;
```

Primjer 11 – deklaracija varijabli u typescript-u

„Null“ tip je podtip svih primitivnih tipova osim „void“ i „undefined“. „Undefined“ tip je podtip svih tipova. Sve varijable koje nisu inicijalizovane su po default-u „undefined“.

Također je moguće specificirati tipove ulaznih parametara u funkcije.

```
function sum(num: number, text: string, optional?: string): string {
    return num + text + optional;
}
```

Primjer 12 – deklaracija funkcije i opcionalnih parametara

Prilikom poziva funkcije, sve prosljeđene varijable moraju se poklapati sa očekivanim ulaznim tipovima. Pored toga, nije moguće poslati vrijednost u funkciju koja ne očekuje nikakvu vrijednost za razliku od Javascripta-a koji samo ignorira prosljeđenu vrijednost. Opcionalni parametri se označavaju sa „?“ za razliku od Javascript-a gdje je svaki parametar opcionalan. Dodatno, provjera tipova se vrši prilikom kompajliranja umjesto prilikom rada kao kod Js-a.

```
var mult = (h: number, w: number) => h * w;
var update : (h: number) => void;
```

Primjer 13 – lambda funkcije

Brži način pisanja povratne vrijednosti je iza „=>“ znaka bez potrebe za pisanjem „return“ i vitičastih zagrada. Ukoliko funkcija ne vraća vrijednost, onda se piše „void“ kao povratni tip.

### 1.3.2 Interfejsi

U Typescriptu i funkcije mogu implementirati interfejs ili više njih istovremeno. Kao povratni tip iz funkcije mora biti objekat sa svim atributima interfejsa odnosno implementacijom svih metoda interfejsa.

```

interface sessionEval {
    addRating: (rating: number) => void;
    calcRating: () => number;
}

function sessionEvaluator(): sessionEval {
    var ratings: number[] = [];
    var addRatingImpl = (rating: number) => {
        ratings.push(rating);
    }
    var calcRatingImpl = () => {
        var sum: number = 0;
        ratings.forEach(function (el) {
            sum += el;
        });
        return sum / ratings.length;
    }
    return {
        addRating: addRatingImpl,
        calcRating: calcRatingImpl
    }
}

```

Primjer 14 – implementacija interfejsa unutar funkcije

### 1.3.3 Klase

Svi atributi klase su automatski „**public**” osim ako nije naglašeno da su „**private**”. Ako su atributi koje prima konstruktor „**public**”, onda će ih Typescript automatski kreirati i inicijalizirati te nije potrebno da ih ručno definišemo. Privatni atribut je potrebno nazvati drugačije od gettera i settera jer će prilikom poziva „**this.atribut**” se pozvati getter odnosno setter a ne direktno atribut.

```

class Engine {
    constructor(public horsepower: number, public engineType: string) { }
}

class Car {
    private _engine: Engine;
    constructor(engine: Engine) {
        this.engine = engine;
    }
    get engine(): Engine {
        return this._engine;
    }
    set engine(value: Engine) {
        this._engine = value;
    }
    start() : void {
        alert('Car engine started ' + this._engine.engineType);
    }
}

```

Primjer 15 – definisanje klase u typescriptu

### 1.3.4 Konverzija tipova

Tipove je moguće pretvoriti u druge uz dodavanje „<Tip>“ ispred vrijednosti koju želimo konvertovati. U primjeru ispod, funkcija „getElementById“ vraća tip „HTML<Element>“ te ga je potrebno konvertovati u „HTMLInputElement“ sa „<HTMLInputElement>“. Konverzija je jako korisna kod pretvorbe baznog tipa u izvedeni tip kada smo sigurni da je izvedeni tip sačuvan u baznom (slično polimorfizmu).

```
var input: HTMLInputElement = <HTMLInputElement>document.getElementById("xId");

var engine: IEngine = new Engine(100, '1.6 CDI');
console.log((<Engine>engine).horsePower); //konverzija je neophodna
```

Primjer 16 – konverzija tipova

## 1.4 O Angular-u

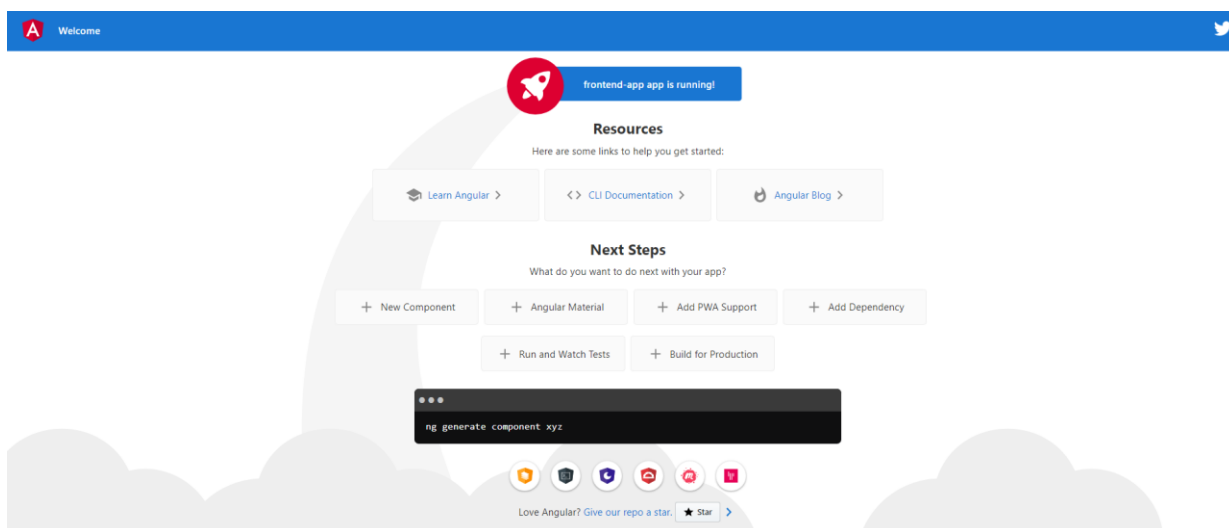
Angular je framework otvorenog kôda baziran na Typescript-u. Koristi se za razvoj web aplikacija, a posebno single-page web aplikacija.

### 1.4.1 Instalacija Angulara

Potrebo je kreirati folder u kojem će se nalaziti naš angular projekat. Potom otvoriti cmd u tom folderu i pokrenuti ispod šest komandi.

```
npm install -g @angular/cli
set-executionpolicy remotesigned
ng new frontend-app --style=scss
set-executionpolicy restricted
cd frontend-app
ng serve --open
```

Poslije toga, početna aplikacija se može otvoriti sa na linku<sup>5</sup> i izgleda ovako:



Slika 1 – izgled automatski generisanog početnog projekta

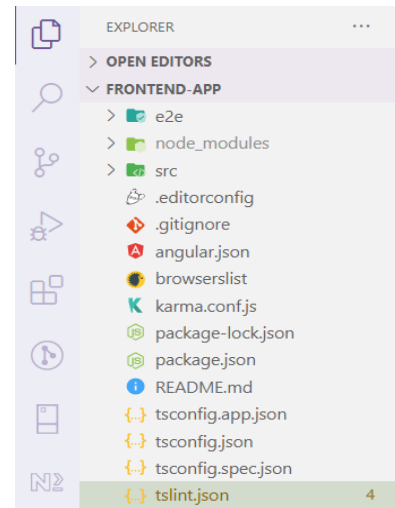
<sup>5</sup> <http://localhost:4200/>



## 1.4.2 Visual studio code setup

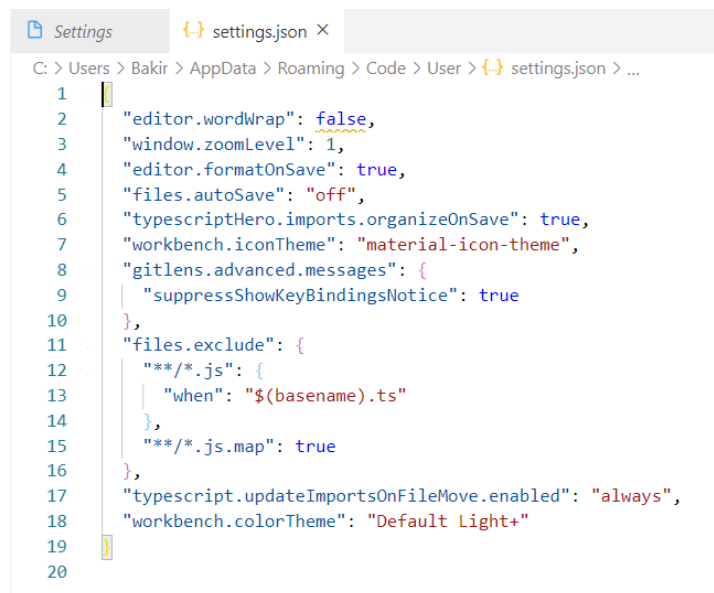
Svi uposlenici firme moraju pisati istim standardom kodiranja. To je potrebno raditi da bi različiti programi napisani od strane različitih programera dobili zajednički izgled, poboljšali čitljivost i olakšali održavanje kôda. Dodatno olakšava detekciju grešaka i povećavaju ponovnu upotrebljivost kôda. Da se ovaj proces ne bi radio ručno, potrebno je podesiti razvojno okruženje da to radi za nas. Radno okruženje će formatirati naš kôd svaki put kada spasimo promjene.

Da bi iskonfigurisali VSC po standarima Zira kompanije, potrebno pruzeti „[tslint.json](#)” datoteku sa Zira Confluence-a te ju importovati u projekat na root nivou umjesto postojeće „[tslint.json](#)” datoteke.



Slika 2 – lokacija tslint.json datoteke na root novou projekta

Sljedeći korak je otvoriti meni za preuzimanje plugina (**ctrl+shift+x**) te instalirati sve potrebne plugine. Tu spadaju: Typescript hero plugin, TSLint, Bracket Pair Colorizer, Auto Rename Tag, Material Icon Theme i Move TS. Nakon toga, potrebno je otvoriti postavke sa „(**ctrl + ,**)” te postaviti prikaz na JSON u gornjem desnom ćošku te unijeti postavke kao na slici ispod:



Slika 3 – JSON prikaz postavki

## 2 Korisnički zahtjevi projekta

Prije početka izrade projekta, najbitnije je što preciznije odrediti zahtjeve i opisati rad aplikacije. Opis aplikacije, skice ekrana i zahtjeve smo dobijali direktno od mentora tako da nije bilo potrebe za anketiranjem korisnika i revizijom zahtjeva. Sada ćemo spomenuti sve zahtjeve koji trebaju biti ispunjeni sa projektovanom aplikacijom.

### 2.1 Početna stranica

- Na početnoj stranici se nalazi jedna od Zira Internship slika.
- Gornja traka je vidljiva na svakoj stranici
- U gornjoj traci se nalazi Zira logo koji klikom vraća na početnu stranicu
- U gornjem desnom ćošku se nalaze ikonice korisnika, obavještenja te gašenja
- Sa lijeve strane se nalazi meni
- Sadržaj treba biti prilagođen mobilnom prikazu
- Svi stilovi se mogu promijeniti i ne moraju izgledati kao u datoj skici

### 2.2 Meni

- Vidljiv na svakoj stranici
- Build-a se iz JSON datoteke
- Nalaze se sljedeće kategorije: Emoloyees, Statistics, Reports, Server, Registries, Project, Availability, Compress
- Svaka kategorija ima odgovarajuću ikonicu koja se također nalazi u JSON datoteci
- Prelazom miša preko ikonice se prikazuje naziv kategorije
- U gornjem lijevom ćošku je hamburger dugme koje sa strane otvara cijeli meni sa svim nazivima
- U mobilnom prikazu ikonice su skrivene i koristi se samo „hamburger“ dugme za prikaz menija
- Klikom na Employees se otvara ekran za upravljanje uposlenicima

### 2.3 Upravljanje uposlenicima

- Uposlenike prikazati tabelarno
- Kolone su sljedeće:
  - First name – text
  - Last name - text
  - Phone number – brojevi formata +387xxxxxxxxx
  - Email - text
  - Status - padajuća lista sa izborima: At work, Holiday absence, Business trip, Sickness absence, Maternity absence
  - Birth date – datum formata dd/MM/yyyy
  - Gender - padajuća lista sa izborima: male, female
  - Hire date – datum formata dd/MM/yyyy
  - Job title – padajuća lista sa izborima: Junior Developer, Senior Developer, System Designer, System Architect, Business Analyst, Administration, Marketing, Sales
  - Id – broj koji je read only
  - Created – datum formata dd/MM/yyyy koji je read only

- Created by – read only tekst uposlenika koji je kreirao unos
- Modified – read only boolean
- Modified by – read only naziv uposlenika koji je izvršio izmjenu
- Omogućiti filtriranje i sortiranje po svakoj koloni
- Omogućiti dodavanje, brisanje te izmjenu uposlenika sa ikonicama u gornjem desnom ćošku tabele
- Prikazati deset uposlenika po stranici
- Sva polja su obavezna

## 2.4 Nadogradnja u budućnosti

Kao i sa svakom drugom aplikacijom, projektovanje aplikacije koju će biti lahko nadograditi u budućnosti je postala standardna praksa. Aplikacija koja je pravljen sa mogućom nadogradnjom u vidu će biti jeftinija za ažurirati. Bitno je koristiti najmodernije tehnologije jer na taj način će cijena ažuriranja u budućnosti biti smanjena. Za projekat je korišten Angular 9 što je tada bila najmodernija verzija Angulara. Dodatno, Angular redovno dobija nove verzije u skladu sa novim dostignućima u industriji.

## 2.5 Internacionalizacija i lokalizacija

Trenutno se traži da aplikacija podržava samo engleski jezik. Zbog toga se neće vršiti proces internacionalizacije. Angular podržava markiranje riječi koje se trebaju prevesti tako da je moguće izvesti lokalizaciju ukoliko dođe do potrebe u budućnosti.

## 3 Izgradnja projekta

Sada ćemo proći kroz detaljni opis izgradnje projekta od samog početka te time pokazati sve naučene stvari o samom Angular-u.

### 3.1 Build menija iz JSON-a

Kôd u Angular-u je razdvojen u komponente koje predstavljaju logički dio aplikacije i zadužene su za prikaz određenog dijela stranice. Komponente se sastoje od HTML templejta, klase koja definiše ponašanje templejta te stilova koji definiraju izgled templejta.

Struktura JSON-a u kojem se nalazi meni je niz objekata sa atributima „name”, „routerLink”, „icon“ te „menus“. Da bi se JSON importovao u komponentu potrebno je definisati model (klasu) sa istim atributima kao i JSON.

```
export class Menu {  
  name: string;  
  menus: Menu[];  
  routerLink: string;  
  icon: string;  
}
```

Isječak kôda 1 – klasa meni

Dalje je potrebno nizu objekata takve klase dodijeliti JSON koji se direktno importuje u komponentu.

```
import * as menu from 'src/assets/menu/menu.json';  
menus: Menu[] = menu.menus;
```

Isječak kôda 2 - import JSON objekata

Sljedeći korak je instaliranje „Material frameworka” i „Font awesome” ikona komandama ispod te generisanje „sidenav” komponente iz „Material-a“:

```
ng add @angular/material;  
ng add @fortawesome/angular-fontawesome@0.6;  
npm install @fortawesome/free-regular-svg-icons  
ng generate @angular/material:material-nav --name=sidenav
```

Da bi se ikonice prikazale, potrebno ih je importovati u jedan objekat u komponenti i nakon toga pristupati atributima tog objekta u zavisnosti koju ikonu iz menija treba prikazati. Da bi se izbjegalo ponavljanje kôda i iskoristio DRY<sup>6</sup> princip, u templejt-u je potrebno iterirati kroz meni „for“ petljom. Poslije svake ikonice se dodaje „mat-divider” koji predstavlja horizontalnu liniju između ikona osim one koja je zadnja.

---

<sup>6</sup> DRY - Don't Repeat Yourself

```

<div *ngFor="let menu of menus; last as isLast">
  <a mat-list-item routerLink="{{ menu.routerLink }}" class="navigation-item">
    <fa-icon
      [icon]="icons[menu.icon]"
      size="lg"
      [fixedWidth]="true"
    ></fa-icon>
    <label class="navigation-item-label">{{ menu.name }}</label>
  </a>
  <div *ngIf="!isLast">
    <mat-divider class="h-navigation-divider"></mat-divider>
  </div>
</div>

```

Isječak kôda 3 – HTML templejt koji kreira pojedinačne stavke menija

## 3.2 Sidenav i toolbar

U „sidenav” komponentu spadaju same ikone, gornja traka (toolbar) te sadržaj tj. ostatak stranice. Sadržaj nije hard-kodiran već je predstavljen kao mjesto za prikaz komponente koju je korisnik kliknuo tj. kao „router-outlet”.

```

<mat-sidenav-content class="sidenav-content">
  <!-- body content goes here -->
  <router-outlet></router-outlet>
</mat-sidenav-content>

```

Isječak kôda 4 – templejt koji prikazuje komponentu koja je kliknuta u meniju

Toolbar se sastoji od hamburger dugmeta koji otvara meni, logo-a, span-a koji će pogurati sve elemente iza njega na desni dio te ikona koje se takođe buildaju kao i ikonice menija. Tekst „Authorized Access” je predstavljen kao tabela sa dva reda tako da se riječi nalaze jedna ispod druge.

```

<!-- This fills the remaining space of the current row -->
<span class="filler"></span>

<div class="far-end">
  <span>Authorized</span>
  <span>Access</span>
</div>

```

Isječak kôda 5 – templejt za pomjeranje teksta

Kako ikone menija sa strane trebaju biti vidljive i kada je meni zatvoren, potrebno je podesiti CSS stil tako da se meni pomjeri u desno a labela sakrije kada je meni zatvoren. To se postiže preko „:not” selektora.

```

/*left sidenav style override*/
.mat-sidenav:not(.mat-drawer-opened) {
  /*makes icons visible*/
  transform: translate3d(0, 0, 0);
  visibility: visible !important;
  width: $nav-width !important;
  overflow: hidden;
}

/*hide items labels*/
.mat-sidenav:not(.mat-drawer-opened) .navigation-item-label {
  display: none !important;
}

```

Isječak kôda 6 – CSS koji prikazuje ikone menija kada je zatvoren

Da bi se dodala sjena na dio gornje trake koja počinje iza „hamburger“ dugmeta, potrebno je sjenu pomjeriti po x-osi za širinu dugmeta. Pomjeranje po x-osi predstavlja prvi parametar „`box-shadow`“ svojstva.

```
box-shadow: 60px 2px 4px -1px rgba(0, 0, 0, 0.2)
```

Pošto se „hamburger“ dugme nalazi u „`toolbar`“ komponenti, koja je dijete komponente „`navbar`“, klik na njega treba otvarati i zatvarati meni koji je u roditelj komponenti „`navbar`“. Potrebno je referencu na meni poslati u dijete komponentu „`toolbar`“. To se radi sa „`@ViewChild`“ dekoratorom u roditelj komponenti koji dozvoljava da se određeni atribut vidi u dijete komponenti. Templejt tag koji se koristi u Typescript dijelu se mora označiti sa „`#`“ kao lokalna referenca na taj tag.

```
<mat-sidenav #drawer></mat-sidenav>
```

Isječak kôda 7 – deklaracija lokalne reference na templejt tag sa `#` tagom

Dalje je potrebno implementirati „`sidenav`“ servis, registrovati ga kao provider u „`app.module`“ te postaviti referencu na dugme koji će „`toolbar`“ dijete komponenta koristiti za otvaranje i zatvaranje menija u roditelj „`navbar`“ komponenti.

```

@ViewChild('drawer') drawer: MatSidenav;
ngAfterViewInit(): void {
  this.sidenavService.setSidenav(this.drawer);
}

```

Isječak kôda 8 – korištenje `@ViewChild` dekoratora i inicijalizacija servisa

Da bi „`toolbar`“ dijete komponenta primila referencu na meni, potrebo je deklarirati objekat tipa „`MatSidenav`“ koji se importuje iz „`material-a`“, te taj objekat označiti sa „`@Input()`“.

```
@Input() drawer: MatSidenav;
```

Isječak kôda 9 – označavanje ulaznog atributa

Prilikom postavljanja toolbar komponente u sidenav templejtu, potrebno je u komponentnom tagu proslijediti sve atribute koji su označeni sa „`@Input()`“.

```
<app-nav-toolbar [icons]="icons" [drawer]="drawer"></app-nav-toolbar>
```

Isječak kôda 10 – prosljeđivanje reference na niz ikonica i meni

### 3.3 Rutiranje

Da bi se implementirali linkovi na stranice, potrebo je sve komponente koje želimo prikazati korisniku, registrovati u „**app-routing**“ modulu te putanju do njih.

```
const routes: Routes = [
  { path: '',
    component: HomeComponent,
  },
  {
    path: 'form',
    component: FormComponent,
  },
];
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule],
})
export class AppRoutingModule {}
```

Isječak kôda 11 – app-routing modul sa putanjama na pojedinačne komponente

Dalje je potrebno definisati „**routerLink**“ sa putanjom u anchor HTML elementu. Svaki klik na anchor će prikazati komponentu na datoj putanji unutar „**router-outlet**“ taga.

```
<a routerLink="{{ menu.routerLink }}" class="navigation-item"></a>
```

Isječak kôda 12 – anchor tag sa router linkom

Za generisanje komponente u kojoj se nalazi internship slika i koja se prikazuje klikom na Zira logo, korištena je komanda:

```
ng generate component --name=home
```

### 3.4 Tabela uposlenika

Za prikaz tabele uposlenika korišten je „ag-Grid” koji nudi mnogo gotovih funkcionalnosti poput sortiranja i filtriranja. „Ag-Grid” je nezavisni framework za prikaz tabelarnih podataka koji se lahko integriše u Angular. Ima nevjerovatne performanse jer može prikazati više stotina hiljada redova unutar tabele te iste sortirati i filtrirati bez većih zastojanja.

Za instalaciju je potrebno pokrenuti komandu:

```
npm install --save ag-grid-community ag-grid-angular
```

Dodatno je moguće instalirati enterprise verziju koja je besplatna ako ne radimo deployment stranice. Ova verzija nudi dodatne funkcionalnosti i izgled.

```
npm install ag-grid-enterprise
```

Nakon toga je potrebno importovati „ag-Grid-angular” komponentu u aplikaciju te iskoristiti je u template-u za prikaz tabele. Bitna svojstva „ag-Grid-angular” komponente su:

- columnDefs – opisuje definicije pojedinačnih kolona
- rowData – podaci koji se smještaju u kolone, podaci ne moraju imati sve kolone popunjene
- defaultColDef – definicija kolone koja se daje svakoj koloni kao default
- pagination – dozvoljava prikaz podataka po stranicama
- paginationPageSize – postavlja maksimalan broj redova po stranici
- animateRows - dodaje animaciju pri dinamičkom dodavanju i brisanju redova
- components – označava custom komponentu
- frameworkComponents – označava komponente framework-a koje će se koristiti unutar tabele poput custom datepicker-a iz material-a
- rowSelection – opcija „multiple” daje mogućnost selektovanja više redova odjednom te dohvaćanje selektovanih redova ukoliko ih želimo izbrisati



### 3.4.1 Tema tabele

Za postavljanje unaprijed definisane teme za tabelu, potrebno je istu importovati u globalni „`scss`” i postaviti klasu „`ag-Grid`” komponente unutar HTML-a. Moguće je dodatno zadati atribute za „`ag-theme-material`” kao što je različita boja pozadine za neparne redove.

```
@import "node_modules/ag-grid-community/src/styles/ag-grid.scss";
@import "node_modules/ag-grid-community/src/styles/ag-theme-material/sass/ag-
theme-material-mixin.scss";

.ag-theme-material {
  @include ag-theme-material(
    (
      odd-row-background-color: #edf2f7,
    )
  );
}
```

Isječak kôda 13 – import mixin teme

### 3.4.2 Definisanje kolona

Kolone imaju svoje atribute koji definišu njen izgled:

- `width` – širina kolone u pikselima
- `sortable` – dozvola za sortiranje kolone
- `editable` – dozvola za inline izmjenu kolone
- `filter` – specificira filter koji može biti custom ili jedan od tri koja su unaprijed data: `agTextColumnFilter`, `agDateColumnFilter` i `agNumberColumnFilter`
- `resizable` – dozvola za promjenu širine
- `suppressMenu` – sakriva meni sa opcijama sakrivanja kolona i pinovanja kolona
- `floatingFilterComponentParams`: { `suppressFilterButton`: true } – sakriva filter dugme
- `headerClass` – postavlja klasu zaglavlja kolone kako bi mogli definisati svoj CSS izgled
- `checkboxSelection` – daje opciju odabira redova kroz checkbox
- `headerName` – ime kolone
- `field` – ime kolone koju koristimo kada definišemo podatke ili kada dohvatamo kolonu
- `cellRenderer` – postavlja komponentu koja prikazuje izmjenu polja
- `cellEditor` – postavlja izgled prikaza tokom inline izmjene polja
- `cellEditorParams` – postavlja objekat sa nizom vrijednosti koje će biti ponuđene pri izmjeni polja

Podatke koji će se učitati u tabelu je potrebno postaviti u niz objekata gdje svaki objekat opisuje vrijednost kolona (ne nužno svih).

```

rowData = [{
  name: 'Bakir',
  l_name: 'Karovic',
  phone: '+38762409970',
  email: 'bkarovic1@etf.unsa.ba',
  status: 'At work',
  bdate: '23/09/1999',
  gender: 'Male',
  hdate: '01/08/2025',
  title: 'Junior Developer'
}]

```

Isječak kôda 14 – niz sa podacima tabele

### 3.5 Brisanje i dodavanje redova tabele

Da bi se koristio „ag-Grid” u Typescript-u, potrebno je postaviti lokalnu referencu na samu „agGrid” komponentu u HTML-u i nakon toga dohvatiti referencu na nju. Dalje je potrebno dodati dvije ikone u HTML-u i povezati događaj za klik koji će pozvati funkcije za dodavanje odnosno brisanje redova.

```

<fa-icon
  (click)="deleteSelectedRows()"
  [icon]="icons.faUserTimes"
  size="lg"
  [fixedWidth]="true"
  style="color: red;"
></fa-icon>

```

Isječak kôda 15 – Font awesome ikona u HTML templejtu

Odabrani redovi se dohvataju sa „getSelectedRows” metodom *API* atributa „ag-Grid-a”. Brisanje istih radimo tako što prosljedimo „updateRowData” funkciji odabrane redove pod „remove” atributom objekta.

```

public deleteSelectedRows() {
  const selectedRows = this.agGrid.api.getSelectedRows();
  this.agGrid.api.updateRowData({ remove: selectedRows });
}

```

Isječak kôda 16 – funkcija za brisanje redova u tabeli

Redovi se dodaju tako što prosljedimo jedan prazan objekat u niz kao „add“ atribut „updateRowData“ funkcije. Dodatno je moguće odmah početi mijenjati red tako što ćemo prebaciti fokus na njega i početi izmjenu prve kolone sa „startEditingCell“ metodom gdje prosljeđujemo objekat sa atributom reda i nazivom kolone koji su u ovom slučaju prva kolona i zadnji red.

```
public addRow() {
    this.agGrid.api.updateRowData({
        add: [{}],
    });
    const last = this.gridApi.getDisplayedRowCount() - 1;
    this.gridApi.setFocusedCell(last, 'name');
    this.gridApi.startEditingCell({
        rowIndex: last,
        colKey: 'name',
    });
}
```

Isječak kôda 17 – funkcija za dodavanje redova u tabelu

### 3.6 Inline datepicker

Da bi se formatirao inline prikaz datepicker-a, potrebno je definisati datepicker modul sa svim potrebnim metodama. Dalje je potrebno u definiciji kolone definisati da ova funkcija predstavlja „cellEditor“ te kolone.

```
export function getDatePicker() {
    function Datepicker() {}
    Datepicker.prototype.init = function (params) {
        this.eInput = document.createElement('input');
        this.eInput.value = params.value;
        this.eInput.classList.add('ag-input');
        this.eInput.style.height = '100%';
        $(this.eInput).datepicker({ dateFormat: 'dd/mm/yy' });
    };
    Datepicker.prototype.getGui = function () {
        return this.eInput;
    };
    Datepicker.prototype.afterGuiAttached = function () {
        this.eInput.focus();
        this.eInput.select();
    };
    Datepicker.prototype.getValue = function () {
        return this.eInput.value;
    };
    Datepicker.prototype.destroy = function () {};
    Datepicker.prototype.isPopup = function () {
        return false;
    };
    return Datepicker;
}
```

Isječak kôda 18 – modul za inline prikaz datepickera

### 3.7 Filtriranje datuma

Da bi filtriranje kolone po datumu radilo, potrebno je definisati komparator funkciju u definiciji kolone. Funkcija „`dateFilterComparator`” je komparator funkcija koja vraća -1 ili 1 ako je datum koji se filtrira veći odnosno manji od onog u ćeliji. Ako su datumi isti, onda vraća 0 i samo ti datumi će se prikazati kao rezultat filtriranja.

```
headerName: 'Hire date',
field: 'hdate',
filter: 'agDateColumnFilter',
filterParams: {
  // filter comparator
  comparator: this.dateFilterComparator,
},
// sort comparator
comparator: dateSortComparator,
width: 200,
cellEditor: getDatePicker()
```

Isječak kôda 19 – definicija kolone Hire date

### 3.8 Sortiranje datuma

Funkcija za sortiranje datuma je slična funkciji za filtriranje datuma. Razlikuju se u tome da komparator funkcija za sortiranje datuma kao rezultat vraća razliku između dva datuma koja se porede. Na osnovu te razlike će se znati koji datum je veći a koji manji.

Činjenica da se za dvije slične svrhe ne mogu koristiti iste funkcije na prvi pogled predstavlja loš dizajn. Međutim, framework „`ag-Grid`” je napravljen da bude veoma efikasan i da radi sa stotinama hiljada redova u tabeli. Kako funkcije za filtriranje i sortiranje imaju različitu kompleksnost, korištenje jedne iste zajedničke funkcije bi usporavalo rad, što nije željen efekat.

### 3.9 Validacija

Validacija je rađena na 2 različita načina. Prvi način je korištenjem gettera i settera u definiciji kolone koji će odlučiti da li se vrijednost treba prihvatiti ili ne. Ova vrsta validacije je korištena samo kod polja koja nisu smjela biti prazna.

```
headerName: 'First name',
field: 'name',
valueGetter: function (params) {
  // check for undefined and null
  if (params.data.name) {
    // return value
    return params.data.name;
  } else {
    // return default value
    return 'required';
  }
},
valueSetter: function (params) {
  if (params.newValue) {
    params.data.name = params.newValue;
    // return true to tell grid that the value has changed, so it knows
    // to update the cell
    return true;
  } else {
    // return false, the grid doesn't need to update
    return false;
  }
}
```

Isječak kôda 20 – validacija za required polja

Drugi način validacije je korišten za polja koja moraju imati specifičan format poput maila i broja telefona. Definiše se modul sa funkcijama inicijalizacije, dodavanjem listenera te funkcijama provjere validacije. Modul treba da mijenja CSS stil upisanog teksta u ovisnosti da li je validan ili ne.

```
export function PhoneCellEditor() {}  
// creation of input element  
PhoneCellEditor.prototype.init = function (params) {  
  this.eGui = document.createElement('div');  
  this.eGui.innerHTML = `  
    <input value=${params.value} style='width: 150px;' />  
  `;  
  this.eInput = this.eGui.querySelector('input');  
  // add listener  
  this.eInput.addEventListener('input', this.inputChanged.bind(this));  
};  
  
PhoneCellEditor.prototype.inputChanged = function (event) {  
  const val = event.target.value;  
  if (this.isValid(val)) {  
    this.eInput.classList.remove('invalid-cell');  
    this.eInput.classList.add('valid-cell');  
  } else {  
    this.eInput.classList.remove('valid-cell');  
    this.eInput.classList.add('invalid-cell');  
  }  
};  
  
PhoneCellEditor.prototype.isValid = function (value) {  
  const regex = /^(\+387\d{8}(:\d{1})?)$/;  
  return regex.test(String(value).toLowerCase());  
};
```

Isječak kôda 21 – dio modula za validaciju unosa broja telefona

Dati modul se postavlja kao cellEditor u definiciji kolone.

```
headerName: 'Phone number',  
field: 'phone',  
cellEditor: PhoneCellEditor
```

Isječak kôda 22 – definicija kolone za broj telefona gdje se koristi cellEditor

## 4 Frontend izgled

Svi ekrani su dizajnirani po skicama iz definicije zahtjeva. Za izbor boja je korišten „[Material Design](#)“ koji dodatno uljepšava izgled i poboljšava korisničko iskustvo. Sada ćemo proći kroz ključne stvari interfejsa.

### 4.1 Početna stranica

Početna stranica je prva stvar koju korisnik vidi kada otvori stranicu. Meni i gornja alatna traka je vidljiva na svakoj stranici. Na početnoj stranici se još dodatno nalazi i slika koja služi kao natpis. Ova jednostavnost prezentira čist i elegantan prvi pogled na aplikaciju.



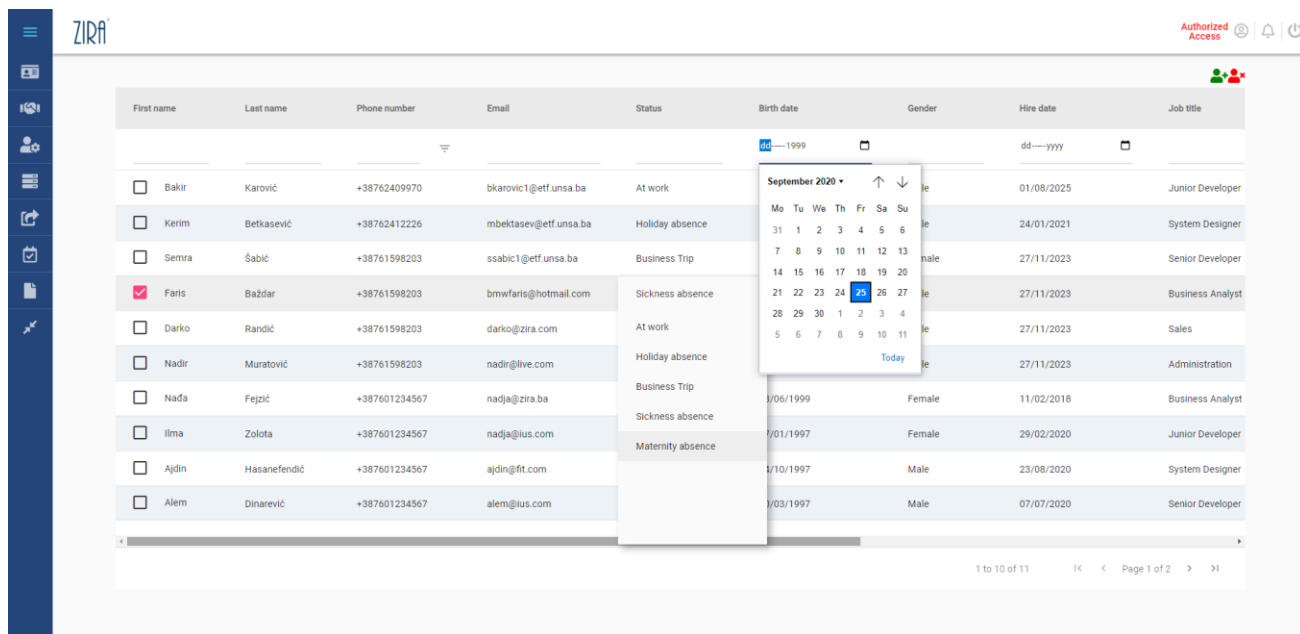
Slika 4 – prikaz početnog ekrana sa otvorenim menijem

## 4.2 Ekran za prikaz uposlenika

Ekran za prikaz uposlenika je dizajniran kao „single-page“ aplikacija. Sav unos, CRUD operacije te validacija se radi direktno nad ćelijom. Na taj način korisnik uvijek zna gdje se nalazi i ne gubi fokus prilikom akcije.

Svaku kolonu je moguće pretražiti i filtrirati čime se ubrzava pretraga pojedinačnog uposlenika. Prilikom pretrage i filtriranja nema potrebe za osvježivanjem stranice jer je prikaz rezultata pretrage i filtriranja dinamičan. To znači da se rezultat prikazuje istovremeno dok korisnik piše pretragu. Ovo daje osjećaj direktne interakcije sa tabelom.

Dodatno, tabela omogućava odabir više uposlenik istovremeno držeći tipku „Ctrl“ ili „Shift“.



The screenshot displays the ZIRA application interface. On the left is a dark blue sidebar with icons for navigation. The main area features a table with columns: First name, Last name, Phone number, Email, Status, Birth date, Gender, Hire date, and Job title. A calendar dropdown menu is open over the Birth date column, showing the month of September 2020. The table contains 11 rows of employee data. The first row is highlighted with a checkbox. The status column contains various entries like 'At work', 'Holiday absence', 'Business Trip', 'Sickness absence', and 'Maternity absence'. The bottom right corner shows pagination: '1 to 10 of 11' and 'Page 1 of 2'.

	First name	Last name	Phone number	Email	Status	Birth date	Gender	Hire date	Job title
<input type="checkbox"/>	Bakir	Karović	+38762409970	bkarovic1@etf.unsa.ba	At work	dd--1999		dd--yyyy	Junior Developer
<input type="checkbox"/>	Kerim	Betkasević	+38762412226	mbektasev@etf.unsa.ba	Holiday absence			24/01/2021	System Designer
<input type="checkbox"/>	Semra	Šabić	+38761598203	ssabic1@etf.unsa.ba	Business Trip		Female	27/11/2023	Senior Developer
<input checked="" type="checkbox"/>	Faris	Baždār	+38761598203	bmfaris@hotmail.com	Sickness absence			27/11/2023	Business Analyst
<input type="checkbox"/>	Darko	Randić	+38761598203	darko@zira.com	At work			27/11/2023	Sales
<input type="checkbox"/>	Nadir	Muratović	+38761598203	nadir@live.com	Holiday absence			27/11/2023	Administration
<input type="checkbox"/>	Nada	Fejzić	+387601234567	nadja@zira.ba	Business Trip	06/1999	Female	11/02/2018	Business Analyst
<input type="checkbox"/>	Ilma	Zolota	+387601234567	nadja@ius.com	Sickness absence	01/1997	Female	29/02/2020	Junior Developer
<input type="checkbox"/>	Ajdin	Hasanefendić	+387601234567	ajdin@fit.com	Maternity absence	10/1997	Male	23/08/2020	System Designer
<input type="checkbox"/>	Alem	Dinarević	+387601234567	alem@ius.com		03/1997	Male	07/07/2020	Senior Developer

Slika 5 - prikaz ekrana za upravljanje uposlenicima sa zatvorenim menijem



## **Zaključak**

Ovaj rad je analizirao i prošao kroz proces izgradnje male web aplikacije za upravljanje zaposlenicima. Definirao je ciljeve i zahtjeve razvoja projekta te diskutovao rezultate postignute datim procesom. Rezultujuće programsko rješenje je moderna mini web aplikacija koja je spremna da postane punopravna aplikacija za upravljanje cijelom firmom.

## Bibliografija

„*Speeding your CSS workflow with Sass and Compass*“ – Gary Simon, 6.10.2015,  
<https://app.pluralsight.com/library/courses/speeding-up-css-workflow-sass-compass-2342>

„*TypeScript Fundamentals*“ – Dan Wahlin i John Papa, 25.3.2016,  
<https://www.pluralsight.com/courses/typescript>

„*Code with us: Angular Quick Start*“ – John papa i Ward Bell, 10.2.2017,  
<https://app.pluralsight.com/library/courses/code-with-us-angular-quick-start>

„*The Pragmatic Programmer : From Journeyman to Master*“ - Hunt Andrew i Thomas David, 1999.

