Završni ispit iz Programiranja za veb – Septembar 2019/20 25.08.2020.

Ispit se polaže najviše 180 minuta. Raspodela poena je data u narednoj tabeli:

	Zac	laci					Teo	rija					
Redni broj	1	2	1	2	3	4	5	6	7	8	9	10	Suma
Najviše poena	15	20	5	5	5	5	5	5	5	5	5	5	70*

^{*}Poeni za teoriju se skaliraju sa ukupnih 50 na 35.

Da bi se položio završni ispit, potrebno je ostvariti najmanje: 5 poena na Node.js zadatku, 10 poena na Angular zadatku i 12 poena na teorijskom delu. Rešenja zadataka koja imaju sintaksnih grešaka nose 0 poena. Kod pod komentarima neće biti pregledan.

Teorijska pitanja

Uputstvo: U radnom direktorijumu na *Desktop*-u nalazi se datoteka *PrezimeIme_alasNalog_grupa_teorija.txt* u kojoj se nalaze teorijska pitanja. U ovoj datoteci ćete smestiti odgovore na pitanja. Obavezno preimenovati datoteku da sadrži Vaše informacije. Obavezno pratiti sve instrukcije koje su date u datoteci.

Napomena za zadatke

Ukoliko neki zahtevi u tekstovima zadataka mogu da se urade na više načina, onda:

- Ako je tekst u zahtevu **podebljan**, to znači da je potrebno da se taj zahtev implementira na opisan način.
- U suprotnom, ostavljeno je Vama da odlučite na koji način ćete implementirati. Slično, ako neki deo zahteva nije eksplicitno naveden (na primer, ako nije naglašeno da se neki formular validira ili da se vrši obrada u slučaju grešaka), onda se podrazumeva da ne morate da vodite računa o tome u Vašim rešenjima.

Zadaci

Implementirati aplikaciju koja omogućava praćenje meteoroloških informacija o gradovima.

- 1. Napisati Node.js aplikaciju koja ispunjava naredne zahteve:
 - U MongoDB bazi podataka pveb_septembar1_2020 nalazi se kolekcija meteos koja sadrži informacije o temperaturama po gradovima:
 - jedinstveni identifikator unosa $(_id)$, ime grada (city), ime države (country) i niz vrednosti temperatura (temperatures).

Kreirati shemu korišćenjem Mongoose biblioteke na osnovu informacija iz date kolekcije. Sva polja u shemi postaviti na **obavezna**. Potom kreirati model *Meteo*.

- Korišćenjem Express i Mongoose biblioteka implementirati naredni REST API:
 - GET /api/meteos
 - * Značenje: dohvatanje svih informacija o svim gradovima.
 - * Odgovor: JSON reprezentacija rezultujućeg niza i statusni kod 200.
 - GET /api/meteos/{city}
 - $\ast\,$ Značenje: dohvatanje svih informacija o zadatom gradu.
 - * Odgovor: JSON reprezentacija rezultujućeg objekta ili *null* ukoliko zadati grad ne postoji. Statusni kod prilagoditi uspešnosti zahteva.
 - PUT /api/meteos
 - * Značenje: zamena niza temperatura za grad čij je ime navedeno u telu zahteva nizom čije su vrednost, takodje, navedene u telu zahteva.
 - * Odgovor: JSON reprezentacija poruke o uspešnosti izvršenog zahteva uz odgovarajući statusni kod.
- Voditi računa da se odgovarajući elementi serverske aplikacije nalaze u odgovarajućim datotekama.

- Sve greške proslediti funkciji srednjeg sloja koja radi na nivou grešaka (definisanoj u app. js datoteci).
- 2. Napisati jednostraničnu Angular aplikaciju koja ispunjava naredne zahteve**:
 - Slanje svih HTTP zahteva koji su navedeni ispod implementirati u jednom **Angular servisu**. Obezbediti postojanje tačno jedne instance ovog servisa na nivou cele aplikacije.
 - Kreirati komponentu zaglavlja HeaderComponent koja sadrži link ka stranici za pregled informacija o svim gradovima http://localhost:4200/ i link ka stranici za pregled infomacija o Beogradu http://localhost:4200/details/Belgrade.
 - Kreirati komponentu MeteoComponent koja na početnoj stranici sa adresom http://localhost:4200/ prikazuje u formi table (slika 1) informacije o svim gradovima. Informacije o svim gradovima se mogu pročitati slanjem odgovarajućeg zahteva serverskom delu aplikacije. Svaki red tabele se odnosi na jedan grad, a kolone $Min\ Temp$, $Avg\ Temp$ i $Max\ Temp$ predstavljaju redom izračunatu minimalnu, prosečnu (zaokruženu na dve decimale) i maksimalnu temperaturu. Za tražene statistike potrebno je implementirati filtere koji ih izračunavaju. Posebno, svako ime grada je link ka stranici za pregled detaljnih informacija o odabranom gradu.

Za stilizovanje tabele mogu se koristiti klase table i table-striped biblioteke Bootstrap (tabela mora imati zaglavlje thead i telo tbody). Klasa text-center se može iskoristiti za centriranje teksta.

- Kreirati komponentu MeteoDetailsComponent koja na adresama oblika http://localhost:4200/details/ime-odabranog-grada prikazuje detaljne informacije (slika 2) za odabrani grad. Informacije o gradu se mogu pročitati slanjem odgovarajućeg zahteva serverskom delu aplikacije. Naslov sekcije prilagoditi tako da sadrži ime i zemlju odabranog grada, a vrednosti temperatura prikazati redosledom kojim su očitane sa servera.
- U reaktivnom formularu koji je pridružen komponenti *MeteoDetailsComponent* se mogu odabrati opcije za prikaz samo nekoliko prvih ili nekoliko poslednjih temperatura (slika 3). Broj vrednosti za prikaz se unosi u odgovarajuće tekstualno polje. Validacijom reaktivnog formulara obezbediti da pre klika na dugme "Show filtered temperatures" ova vrednost bude uneta i da bude korektan ceo broj. Radio dugme "od kraja" smatrati inicijalno obeleženim. Ukoliko prilikom validacije dodje do greške, ispisati odgovarajuću poruku u alert prozorčiću (slika 4). Ukoliko je sve u redu, prikazati niz filtriranih temperatura (slika 5).

Nakon prve interakcije sa formularom, pored dugmeta za filtriranje treba obezbediti pojavljivanje dugmeta "Reset temperatures" kojim se niz filtriranih vrednosti vraća na početni 3).

Klikom na link "Save filtered results" potrebno je izvršiti trajno čuvanje filtriranih temperatura u bazi. Čuvanje se može inicirati slanjem odgovarajućeg zahteva serverskom delu aplikacije. Korisniku je potrebno prikazati informaciju o uspešnosti ažuriranja (slika 6).

Za slučaj zadavanja grada koji ne postoji (na primer, direktnom modifikacijom URL-a) prikazati odgovarajuću poruku o grešci (slika 7). Za stilizovanje poruke mogu se koristiti klase alert i alert-danger biblioteke Bootstrap.

• Voditi računa o curenju memorije.

**Da bi Vam bile dostupne klase koje su navedene iznad, u datoteci klijent/src/index.html u sadržaju elementa <head> potrebno je da dodate naredne linije:

```
<link rel="stylesheet" type="text/css" href="/assets/bootstrap.min.css">
<link rel="stylesheet" type="text/css" href="/assets/styles.css">
```

All info | Belgrade info

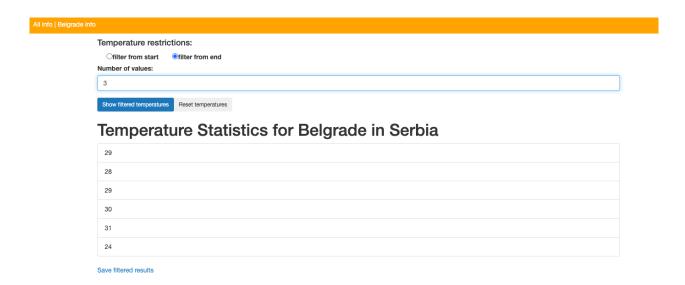
Meteo Statistics

City	Country	Min Temp	Avg Temp	Max Temp
Belgrade	Serbia	24	28.33	31
Novi Sad	Serbia	26	28.60	31
Berlin	Germany	19	21.40	24
Lisbon	Portugal	18	19.17	22

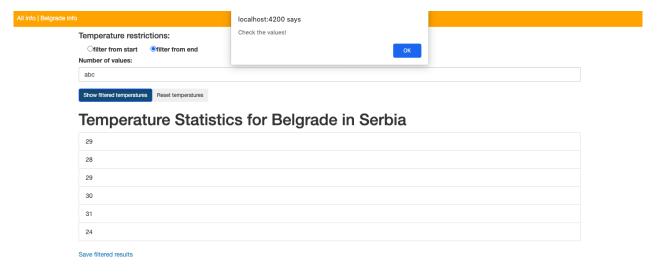
Slika 1: Početna stranica – komponenta MeteoComponent.

Temperature re Ofilter from star Number of values:	
Show filtered tempera	atures
Temper	ature Statistics for Belgrade in Serbia
Temper	ature Statistics for Belgrade in Serbia
	ature Statistics for Belgrade in Serbia
29	ature Statistics for Belgrade in Serbia
29	ature Statistics for Belgrade in Serbia
29 28 29	ature Statistics for Belgrade in Serbia

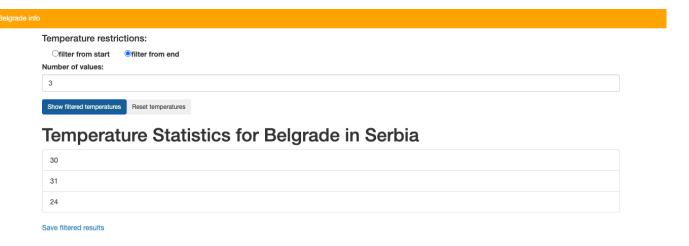
Slika 2: Detaljan prikaz temperatura – komponenta ${\it MeteoDetailsComponent}$.



Slika 3: Interakcija sa formularom – izbor vrednosti za filtriranje.



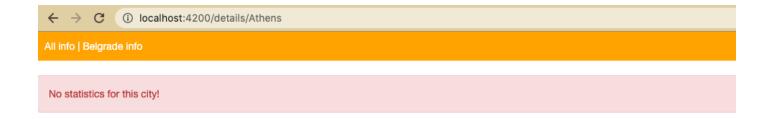
Slika 4: Interakcija sa formularom – unos pogrešnih vrednosti.



Slika 5: Interakcija sa formularom – prikaz temperatura nakon filtriranja.



Slika 6: Prikaz poruke koja je dobijena sa servera nakon klika na link za čuvanje filtriranog sadržaja.



Slika 7: Prikaz poruke o grešci ukoliko se zada nepostojeći grad.