TaskManager - Vježba na satu

FIPU Web aplikacije: 17.12.2024.

Zadatak je napraviti *from scratch* jednostavnu **aplikaciju za upravljanje zadacima**.

- **Vue.js frontend** dio aplikacije mora sadržavati mogućnost dodavanja, brisanja i označavanja zadataka kao završene.
- **Express.js backend** mora sadržavati osnovne funkcionalnosti za CRUD operacije nad zadacima na odgovarajućim endpointima.
- Koristite MongoDB Atlas bazu podataka za pohranu zadataka
- Komunikaciju između frontenda i backenda realizirajte pomoću Axios biblioteke.

CHALLENGE🔥: Pokušajmo riješiti bez ChatGPT-a i drugih AI alata.

- Googlajte, koristite dokumentaciju, skripte iz kolegija, komunicirajte s kolegama, bacite oko na Stack Overflow, itd.

1. Klonirajte repozitorij s predloškom: `https://github.com/lukablaskovic/TaskManager/tree/main`
2. Instalirajte sve potrebne pakete: `npm install`
3. Pokrenite lokalno Vue.js aplikaciju: `npm run dev`
4. Proučite definiranu HTML strukturu u `App.vue`

`App.vue` je podijeljen u 3 dijela:

1. `script setup` - Vue 3 script setup dio: U Vue3 ne definiramo više `data`, `methods`, `computed`, `watch` i `props` objekte kao što smo u `Vue2`

```
<script setup></script>
```

2. `template` - predstavlja jednostavnu HTML strukturu s TailwindCSS klasama

```
<template></template>
```

3. `style scoped` - ovdje želimo pisati dodatni CSS kod koji se odnosi **samo na ovu komponentu** (`scoped`)

```
<style scoped></style>
```

Uočite u HTML komentarima osnovne `template` dijelove:

- `<!-- Header -->` Naslov i button za dodavanje novog zadatka
- `<!-- Task List -->` Prozor s naslovom "Vaši zadaci" koji sadrži zadatke (trenutno samo 1)
- `<!-- Task Item -->` Blok koji sadrži jedan zadatak s naslovom, opisom i buttonima (`<!-- Task Actions -->`)

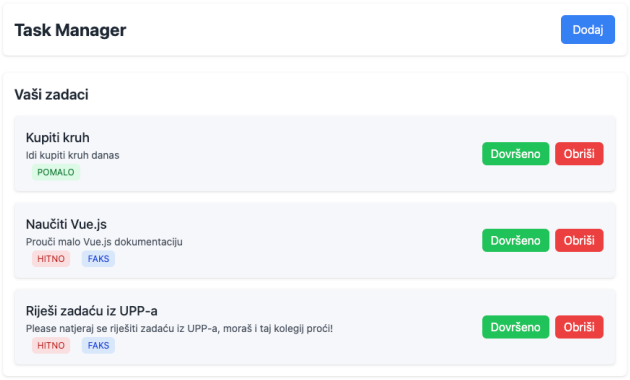
- `<!-- Task Actions -->` Buttoni za označavanje zadatka kao završenog te brisanje

5. Izradite Vue3 komponentu `Task.vue` u `src/components` direktoriju. U njega prebacite `template` dio koji se odnosi na **jedan zadatak**, te definirajte 2 propsa: `naslov` i `opis`
6. Otvorite `TaskManagerBackend` direktorij i pokrenite Express.js server: `nodemon/node index.js`. Prvo treba instalirati sve potrebne pakete: `npm install`
7. Definirajte prvo GET endpoint za dohvaćanje svih zadataka s Expressa. Dodajte endpoint u odgovarajući Router te definirajte nekoliko dummy in-memory zadataka:

```
let tasks = [
  {
    id: 1,
    naslov: "Kupiti kruh",
    opis: "Idi kupiti kruh danas",
    zavrsen: false,
  },
  {
    id: 2,
    naslov: "Naučiti Vue.js",
    opis: "Prouči malo Vue.js dokumentaciju",
    zavrsen: false,
  },
  {
    id: 3,
    naslov: "Riješi zadaću iz UPP-a",
    opis: "Please natjeraj se riješiti zadaću iz UPP-a, moraš i taj kolegij proći!",
    zavrsen: false,
  },
];
```

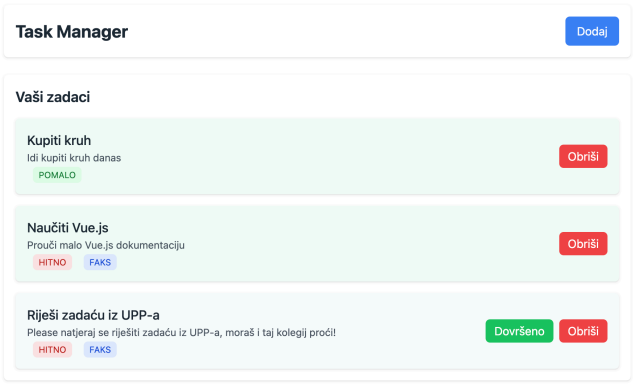
8. Pošaljite zahtjev s Vue.js aplikacije jednom kad se komponenta `App.vue` mounta. Koristite axios biblioteku za slanje GET zahtjeva na `http://localhost:3000/tasks` endpoint.
9. Dohvaćene zadatke s Express.js poslužitelja pohranite u `tasks` reaktivnu varijablu u Vue.js-u te koristeći `v-for` renderirajte odgovarajući broj `Task.vue` komponenti.
10. Napravite novi MongoDB Atlas Cluster. Dodajte kolekciju `tasks` gdje će biti spremljeni zadaci.
11. Dodajte POST metodu na endpoint `/tasks` koja će odjednom dodati sve zadatke koje pošaljete u tijelu zahtjeva. Pozovite metodu preko HTTP klijenta i dodajte postojeće zadatke. Dodajte jednostavnu validaciju podataka za postojanje obaveznih ključeva.
12. Nadogradite endpoint GET `/tasks` na način da vraća zadatke iz MongoDB kolekcije `tasks`.
13. Za svaki task dodajte tagove. `tags` neka bude polje stringova koji predstavljaju tagove zadatka. Npr. `["hitno", "faks", "pomalo"]`. Svaki zadatak može imati proizvoljan broj tagova. Na frontendu prikažite tagove uz svaki zadatak.
14. Prikažite određeni tag različitom bojom. Npr. tag `hitno` neka bude crvene boje, `faks` plave boje, `pomalo` zelene boje. Napravite novu Vue komponentu `TaskTag` koja će definirati 1 tag.

Primjer:



15. Implementirajte sve potrebno (frontend + backend) za označavanje zadatka kao dovršenog. Na frontendu možete "pozeleniti" `div` zadatka koji je završen i sakriti button za označavanje zadatka kao završenog.

- poslužite se skriptom WA5 - koju MongoDB metodu ćete koristiti za ažuriranje zadatka?
- koju HTTP metodu ćete koristiti za ažuriranje zadatka?
- kako će izgledati URL ove rute?



16. Implementirajte promjene na templateu čim se zadatak označi kao završen. Bez potrebe za osvježavanjem stranice (reactive kopija)
17. Implementirajte sve potrebno (frontend + backend) za brisanje zadatka.
    - poslužite se skriptom WA5 - koju MongoDB metodu ćete koristiti za brisanje zadatka?
    - koju HTTP metodu ćete koristiti za brisanje zadatka?
    - kako će izgledati URL ove rute?
18. Promptajte korisnika prije brisanja je li siguran u radnju, funkcija `confirm`
19. Nakon uspješnog brisanja, filtrirajte postojeće zadatke bez ručnog osvježavanja aplikacije (component events).
20. Implementirajte mogućnost dodavanja novog zadatka (frontend + backend). Dodajte novi zadatak u MongoDB kolekciju `tasks`. Poslužite se templateom definiranim u `App_primjer_s_dodavanjem.vue`

Task Manager

Dodaj zadatak

Naslov zadatka:

Prošetaj breka!

Opis zadatka:

Nemoj zaboraviti prošetati breka

Spremi zadatak | Odustani

Vaši zadaci