UNIVERZITET U TUZLI
FAKULTET ELEKTROTEHNIKE
ELEKTROTEHNIKA I RAČUNARSTVO

INFRASTRUKTURA I SERVISI U OBLAKU
PROJEKAT BR. 1

Amina Hasić
Tuzla, april 2025.

1. Opis aplikacije

Task Manager je jednostavna web aplikacija dizajnirana da pomogne korisnicima u organizaciji i praćenju njihovih zadataka na efikasan način. Pruža intuitivan interfejs koji omogućava korisnicima da kreiraju, pregledaju, ažuriraju i brišu zadatke u centralizovanom sistemu.

Svrha aplikacije

Ova aplikacija služi kao jednostavno rješenje za upravljanje zadacima, omogućavajući korisnicima da:

- Kreiraju nove zadatke sa naslovom i opcionim opisom
- Pregledaju listu svih kreiranih zadataka
- Uređuju postojeće zadatke kada se zahtjevi promijene
- Brišu zadatke nakon što su završeni ili više nisu potrebni

2. Softverski preduslovi za pokretanje Task Manager aplikacije

Za uspješan rad Task Manager aplikacije, potrebno je zadovoljiti sledeće softverske preduslove:

1. **Operativni sistem (OS)**

Aplikacija može biti pokrenuta na bilo kojem operativnom sistemu koji podržava Docker. Preporučeni operativni sistemi su:

- **Linux** (preporučeno za rad u produkciji zbog bolje podrške za Docker):
  -Ubuntu 20.04 ili novije verzije
  -CentOS 8 ili noviji
  -Debian 10 ili noviji

- **Windows**:
  -Windows 10 (64-bit) ili noviji (preporučeno koristiti WSL2 - Windows Subsystem for Linux)

- **macOS:**

  -macOS 10.15 (Catalina) ili noviji

2. Docker je neophodan za kontejnerizaciju svih komponenata aplikacije (frontend, backend i MongoDB), kao i za upravljanje svim servisima unutar kontejnera.

- **Preporučena verzija**: Docker **v24.0.7** ili novija

- **Komponente**:
    - **Docker Engine** – omogućava pokretanje i upravljanje kontejnerima

    - **Docker Compose** – (ako je korišten) omogućava definisanje i pokretanje više kontejnera kao jednog servisa

    - **Docker CLI** – komandna linija za interakciju sa Dockerom

3. Node.js i npm (za backend)

-Node.js je potrebno za pokretanje backend servera, jer je aplikacija zasnovana na **Node.js** (Express.js).

-npm (Node Package Manager) je takođe potreban za instaliranje zavisnosti (kao što su Express, Nodemon, MongoDB klijent) u backendu.

Preporučena verzija:

-**Node.js v18.x** ili novija
-**npm v8.x** ili novija

Instalacija Node.js:

-Preporučuje se korišćenje zvanične Node.js distribucije:
Node.js Download

4. MongoDB

MongoDB je NoSQL baza podataka koja se koristi za skladištenje podataka o zadacima. U ovoj aplikaciji, MongoDB je kontejnerizovan pomoću Docker-a, tako da nije potrebna zasebna instalacija na računaru osim ako to ne želite (ako ne koristite Docker).

- **MongoDB verzija**: Preporučena verzija je **MongoDB 5.x** ili novija.

5. Bash (za izvršavanje skripti)

Skripte koje ste pripremili (pripremi_aplikaciju.sh, pokreni_aplikaciju.sh, itd.) koriste **bash** za izvršavanje, tako da je **bash** neophodan na operativnim sistemima kao što su Linux i macOS. Na Windowsu, možete koristiti **WSL** (Windows Subsystem for Linux) ili Git Bash.

- **Instalacija Bash-a**:
    - Na **Linuxu** i **macOS-u** je već instaliran.
    - Na **Windows-u** možete preuzeti Git Bash ili omogućiti WSL.

6. Dodatni preduslovi

- **Internet konekcija**: Za preuzimanje Docker slika i zavisnosti iz NPM-a i Docker Hub-a, potrebna je stabilna internet konekcija.
- **Portovi**: Uverite se da su portovi koje aplikacija koristi (80 za frontend, 3000 za backend) otvoreni i dostupni na vašem računaru ili serveru.

3. Arhitektura Task Manager aplikacije

Aplikacija koristi troslojnu arhitekturu (three-tier architecture), kontejneriziranu pomoću Docker-a. Svaki sloj (frontend, backend, baza) je izolovan u svom kontejneru i komunicira preko Docker mreže task-app-network.

Opis slojeva:

1. Frontend (klijent)

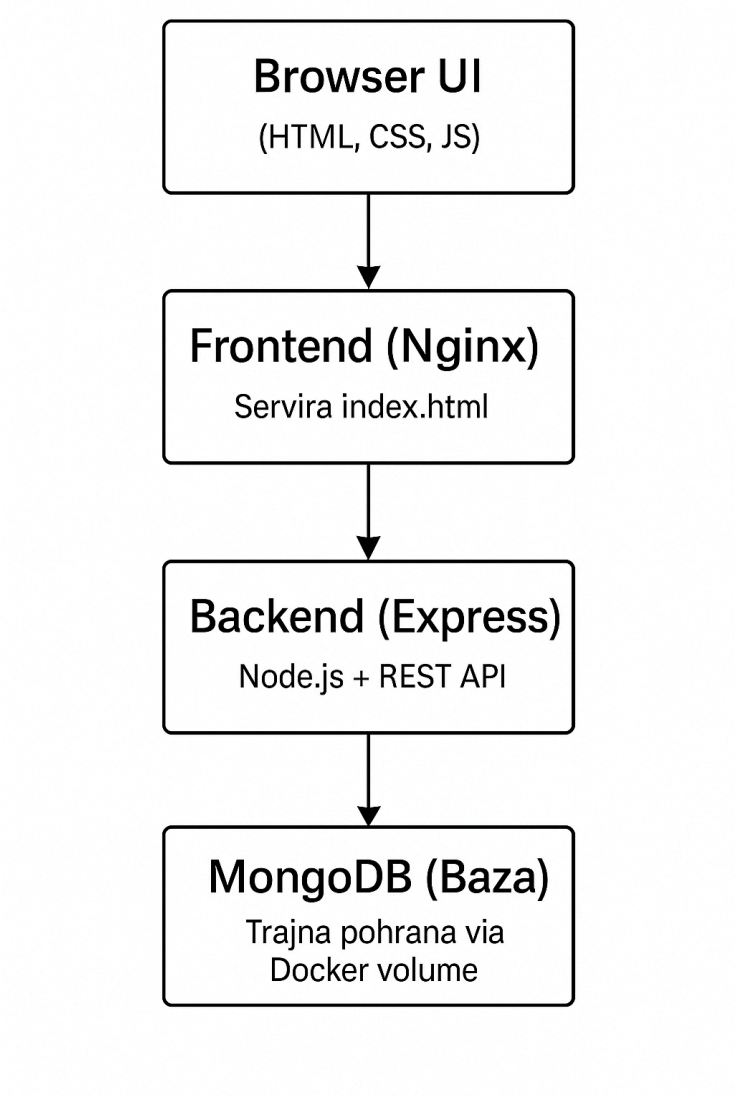
- Izgrađen pomoću: HTML, CSS, JavaScript
- Serviran putem: nginx:alpine (zvanična Docker slika)
- Uloga: Interfejs za korisnika – prikaz zadataka, forma za unos/izmjenu, dugmad za brisanje

2. Backend (API servis)

- Izgrađen pomoću: Node.js + Express.js
- Funkcija: RESTful API – prima zahtjeve od frontenda i komunicira s bazom
- API rute:
    - GET /tasks – dohvat svih zadataka
    - POST /tasks – kreiranje zadatka
    - PUT /tasks/:id – ažuriranje zadatka
    - DELETE /tasks/:id – brisanje zadatka

3. Baza podataka (MongoDB)

- Docker image: mongo:latest (zvanična slika)
- Služi za trajno skladištenje zadataka
- Podaci se čuvaju unutar volumena, tako da restart kontejnera ne briše podatke



4. Docker mreža i servisi

Svi kontejneri su povezani na Docker mrežu task-app-network radi međusobne komunikacije koristeći servise po imenu (mongo, task-app-backend, itd.).

Perzistentnost podataka

- Koristi se Docker volume (npr. mongo-data) za MongoDB kako bi se omogućilo:
    - Čuvanje podataka izvan kontejnera
    - **Bez gubitka podataka prilikom restarta ili rekreiranja MongoDB kontejnera**

Servisi

🔹 1. Frontend servis (task-app-frontend)

- Image: Custom image baziran na nginx:alpine
- Uloga: Servira statičke fajlove (HTML, CSS, JS) korisnicima putem HTTP-a
- Portovi:
    - Lokalni port: 8080
    - U kontejneru: 80
- Dockerfile uključuje:
    - Kopiranje fajlova u /usr/share/nginx/html
    - EXPOSE 80

🔹 2. Backend servis (task-app-backend)

- Image: Custom image baziran na node:18-alpine
- Uloga: REST API za manipulaciju zadacima (GET, POST, PUT, DELETE)
- Portovi:
    - Lokalni port: 3000
    - U kontejneru: 3000
- Komunikacija: API zove MongoDB koristeći hostname mongo

🔹 3. Baza podataka (mongo)

- Image: mongo:latest (zvanična Docker slika)
- Uloga: Trajno skladištenje zadataka u kolekciji tasks
- Portovi:

    - Lokalni port: 27017
    - U kontejneru: 27017
- Volumen: Podaci se čuvaju pomoću volume-a mongo-data

Mreža

🔹 task-app-network (Docker bridge mreža)

- Vrsta: bridge (standardna Docker mreža)
- Uloga: Omogućava kontejnerima da komuniciraju putem DNS imena (npr. mongo)
- Servisi povezani na mrežu:
    - task-app-frontend
    - task-app-backend
    - mongo

Volume

🔹 mongo-data (Docker volume)

- Veže se na: /data/db u MongoDB kontejneru
- Uloga:
    - Omogućava trajnu pohranu podataka
    - Štiti podatke od gubitka pri restartovanju kontejnera

5. Hot Reload u Razvoju

Tokom razvoja, omogućen je hot reload za frontend dio aplikacije. To znači da se promjene u HTML, CSS i JavaScript fajlovima automatski reflektuju u browseru bez potrebe za ručnim osvježavanjem stranice ili restartovanjem servera.

Za razvojni način rada frontend-a koristi se skripta:

./hot_reload.sh

Ova skripta:

- Pokreće lokalni development server sa aktivnim **hot reloadom**
- Prati izmjene fajlova u frontend/ direktoriju
- Automatski otvara aplikaciju na slobodnom portu (npr. http://127.0.0.1:33335)

Primjena

1. Pokreni skriptu: ./hot_reload.sh
2. Otvori navedenu adresu u browseru (npr. http://127.0.0.1:33335)
3. Uredi fajlove index.html, styles.css, ili app.js u frontend/ folderu
4. Promjene će biti prikazane automatski bez potrebe za restartom

6. Kako pristupiti aplikaciji

Otvori web preglednik i idi na:

http://localhost:8080

Tu ćeš vidjeti korisnički interfejs (lista zadataka, forma za unos itd.). Sve statičke stranice se serviraju preko nginx servera.

API je dostupan na:

http://localhost:3000/tasks