12-Faktorska aplikacija - Tehnički izveštaj

Ova dokumentacija opisuje kako je fullstack CRUD aplikacija implementirana u skladu sa 12-Factor App principima. Aplikacija se sastoji od:

- Frontenda razvijenog u ReactJS framework-u,
- Backenda u .NET Core tehnologiji,
- MySQL baze podataka,
- Svi servisi se deploy-uju pomoću Docker-a.

1. Codebase (Jedna baza koda, više deploy-ova)

Status: Ispunjeno

Obrazloženje: Aplikacija koristi centralizovanu codebase strukturu verzionisanu preko GitHub repozitorijuma. Svaki servis (frontend, backend) nalazi se u svom direktorijumu i ima svoj Dockerfile. Deployment se vrši iz jednog izvora (Docker image), bez dupliranja koda.

Rezultat primene metodologije:
Ovo omogućava da se svi servisi razvijaju zajedno i testiraju integrisano. Iako se svaki servis može zasebno deploy-ovati, svi dele zajedničku bazu koda koja se koristi kroz različita okruženja (dev, test, prod).

Preporuke za dalji razvoj: Redovno održavanje grana (main, dev, feature) i CI/CD integracija kako bi se dodatno poboljšala automatizacija.

2. Dependencies (Eksplicitno deklarisane zavisnosti)

Status: Ispunjeno

Obrazloženje: Svaki servis eksplicitno definiše svoje zavisnosti:

- React frontend koristi package.json
- .NET backend koristi *.csproj

Rezultat primene metodologije:
Zavisnosti se ne oslanjaju na sistemske biblioteke host OS-a, već se tačno navodi koje verzije su potrebne. To obezbeđuje konzistentno okruženje prilikom buildovanja Docker image-a, čime se izbegavaju "it works on my machine" problemi.

- Backend koristi *.csproj za .NET dependency-e.
- Frontend koristi package.json za Node/React dependency-e.

- Dockerfile za svaki servis izoluje dependency-e unutar kontejnera, bez oslanjanja na globalni sistem.

Preporuke za dalji razvoj: /

3. Config (Konfiguracija u okruženju)

Status: Ispunjeno

Obrazloženje: Kompletna konfiguracija se nalazi u docker-compose fajlu.

Rezultat primene metodologije:
Na ovaj način, isti kod se može koristiti u više okruženja samo promenom konfiguracije, bez menjanja izvornog koda. Konekcija ka bazi (ConnectionStrings__DefaultConnection) se nalazi kao env var u docker-compose.yml.

Preporuke za dalji razvoj: Za produkciju koristi .env fajlove (ne commit-ovati ih) ili secret menadžere (Vault, AWS Secrets Manager ako se prelazi na Cloud).

4. Backing Services (Spoljni servisi tretirani kao prikačeni resursi)

Status: Ispunjeno

Obrazloženje:
MySQL baza i drugi pomoćni alati se tretiraju kao spoljašnji servisi koji mogu da se zamene bez menjanja aplikacije. MySQL baza se koristi kao backing service preko Docker kontejnera i povezuje se putem servisa. Zamena baze sa eksternim DB servisom (npr. RDS) ne bi zahtevala promene u kodu, samo u konfiguraciji.

Rezultat primene metodologije:
Npr. ako migriramo sa lokalne MySQL baze na managed bazu u cloudu, dovoljno je promeniti DATABASE_URL u env varijablama, bez promene aplikacije.

Preporuke za dalji razvoj: Moguće je lako preći na managed tip baze u Cloud okruženju.

5. Build, Release, Run (Razdvajanje faza)

Status: Ispunjeno

Obrazloženje: Proces je podeljen u tri koraka:

- build: kreiranje Docker image-a (npr. docker build -t aleksandarmilanovic/democrud-frontend:latest .)
- release: deploy image-a
- run: pokretanje kontejnera koji izvršava kod sa datom konfiguracijom

Rezultat primene metodologije:
Na ovaj način se osigurava da isti image prolazi kroz različita test/prod okruženja bez ponovnog buildovanja.
Dockerfile i docker-compose omogućavaju:

- **Build**: Kompilacija i priprema frontend i backend aplikacije.
- **Release**: Definisanje konfiguracija putem docker-compose.yml.
- **Run**: Pokretanje kontejnera.

Preporuke za dalji razvoj: Uvesti CI/CD pipeline (GitHub Actions, GitLab CI, ArgoCD) koji odvaja ove faze i omogućava rollback ako je potrebno.

6. Processes (Aplikacija kao jedan ili više stateless procesa, bez čuvanja stanja u memoriji)

Status: Ispunjeno

Obrazloženje:
Svaki servis (.NET, React) je izolovan proces u svom kontejneru i ne zavisi od lokalnog stanja (disk, memorija).

Rezultat primene metodologije:
Svaki servis (frontend, backend) je nezavistan proces. Backend ne čuva nikakvo stanje — stanje se nalazi u bazi. To omogućava horizontalno skaliranje.

Preporuke za dalji razvoj: /

7. Port Binding (Samostalno izlaganje servisa putem porta)

Status: Ispunjeno

Obrazloženje: Servisi se sami izlažu na portovima (React na 80, .NET na 5000), bez potrebe za spoljnim web serverima.

Rezultat primene metodologije:
Aplikacija ne zavisi od spoljnog web servera (npr. Apache, nginx) već se sama izlaže preko HTTP porta, čime se omogućava bolja fleksibilnost u orkestraciji (ako se postavi na K8s Kubernetes klaster).

- Frontend: Nginx servira React aplikaciju na portu 80.
- Backend: .NET API se expose-uje na portu 5000 (host:8080), samostalno.
- Baza je dostupna na portu 3306 (za testiranje).
- Port binding omogućava direktan pristup aplikaciji bez dodatnog web servera.

Preporuke za dalji razvoj: U produkciji razmotriti izlaganje preko Ingress kontrolera sa HTTPS podrškom.

8. Concurrency (Procesi se skaliraju pokretanjem više instanci)

Status: Ispunjeno

Obrazloženje: Aplikacija može biti skalirana pokretanjem više instanci kontejnera (npr. u Kubernetesu ili sa docker-compose up --scale).

Rezultat primene metodologije:
Svaki servis se može skalirati horizontalno. Time se omogućava rad sa više korisnika paralelno i veća otpornost na greške (npr. kubectl scale deployment democrud-frontend --replicas=3)

Preporuke za dalji razvoj: Uvesti health-check i load balancing (npr. putem Nginx, Traefik ili Kubernetes ingress-a).

9. Disposability (Brzo startovanje i pravilno gašenje)

Status: Ispunjeno

Obrazloženje: Svi kontejneri startuju brzo (u sekundama), i mogu se bezbedno restartovati bez gubitka stanja (jer je stanje van kontejnera – u PVC).

Rezultat primene metodologije:
Kontejneri se brzo startuju i restartuju. Backend koristi retry mehanizam za migracije baze ako ona kasni, što je najbolja praksa. Kontejneri se mogu bezbedno zaustaviti/ponovo pokrenuti.

Preporuke za dalji razvoj: /

10. Dev/Prod Parity (Minimalne razlike između razvojne i produkcione verzije)

Status: Ispunjeno

Obrazloženje:
Korišćenje Docker image-a i docker-compose u lokalnom razvoju omogućava potpuno isto okruženje kao u produkciji (Kubernetes).

Rezultat primene metodologije:
Isti kod, isti image, isti config. Jedina razlika je način deploy-ovanja, čime se smanjuju greške koje nastaju "samo u produkciji". Docker omogućava pokretanje iste aplikacije u različitim okruženjima.

Preporuke za dalji razvoj: /

11. Logs (Logovi kao tokovi događaja)

Status: Ispunjeno

Obrazloženje:
Svi servisi ispisuju logove na standardni izlaz i oni mogu biti prosleđeni u neki log sistem (npr. Elasticsearch, Grafana, Promtail, Loki). Kontejneri prikazuju logove preko docker logs.

Rezultat primene metodologije:
Logovi nisu zapisani lokalno na disk unutar kontejnera, što omogućava skalabilno i centralizovano praćenje aplikacije. Logovi se trenutno ispisuju u konzolu (Console.WriteLine, console.error).

Preporuke za dalji razvoj: Dodati log framework kao što je Serilog (za .NET) i proslediti logove u servis (ELK, Grafana Loki, Promtail).

12. Admin processes (Jednokratne administrativne komande kao deo aplikacije)

Status: Ispunjeno

Obrazloženje: Migracije baze se automatski pokreću u kodu (db.Database.Migrate() sa retry metodom).

Rezultat primene metodologije:
Na ovaj način se administrativne komande ne mešaju sa glavnim kodom aplikacije i mogu se izolovano pokrenuti npr. kao Job u Kubernetesu.

Preporuke za dalji razvoj: Dodati mogućnost admin komandi kroz posebne endpoint-e ili Kubernetes Job definicije. Omogućiti pokretanje migracija, seed podataka i backup kroz posebne CLI komande (npr. dotnet ef database update, node seed.js, ili dedicated admin container).

Zaključak

Aplikacija je projektovana i implementirana u skladu sa 12-Factor App metodologijom. To omogućava:

- lakše testiranje i deploy,
- bolju otpornost na greške,
- fleksibilnije okruženje,
- lakšu migraciju na cloud-native okruženja.

Posebno su dobro implementirani build/release procesi i izolacija servisa. Poboljšanja su moguća u oblasti logovanja, admin task-ova i boljeg odvajanja konfiguracije po okruženju.