UNIVERZA V MARIBORU
FAKULTETA ZA ELEKTROTEHNIKO,
RAČUNALNIŠTVO IN INFORMATIKO

Jernej Ofič, Domen Perko, Gregor Sulcer, Matevž Semprimožnik

DOKUMENTACIJA
ODLOČITEV

Tehnična dokumentacija

TechBuild
Maribor, april 2023

KAZALO VSEBINE

1 ARHITEKTURNE ODLOČITVE

1. 1 Arhitektura rešitve

Aplikacijo sestavlja tehnološki sklad:

- Next.js
- tRPC
- TypeScript
- PlanetScale
- Vercel
- Tailwind CSS
- Prisma
- Clerk
- Open AI API
- GitHub API

1. 1. 1 Next.js

Next.js je močno orodje za razvoj spletnih aplikacij, ki temeljijo na Reactu. Ponuja izboljšano, enostavnejše in zmoglivejše usmerjanje na večstranskih aplikacijah. Vgrajena je podpora za dinamično usmerjanje in prerisovanje komponent, ki omogoča boljšo uporabniško izkušnjo. Omogočeno je tako SSR in SSG. S tem izboljšamo hitrost nalaganja strani in SEO. Vključenih je več funkcionalnosti za optimizacijo spletnih strani (npr. optimizacija prenosa in predponjenje strani). Next podpira tudi API usmerjanje, ki omogoča enostavno definiranje API-jev in poenostavi komunikacijo med odjemalcem in strežnikom. Omogoča ustvarjanje lastnih API-jev ali uporabo obstoječih za vzpostavitev hitrega in zanesljivega podatkovnega toka. Vključen je sistem za zajemanje napak, ki pripomore pri odrivanju in odpravljanju napak v aplikacijah. Prav tako je ogrodje priljubljeno in podporto s strani široke skupnosti razvijalcev. Obstaja obsežna dokumentacija, vodiči in vzorčni projekti, ki nam pomagajo hitro začeti in reševati morebitne težave.

1. 1. 2 TypeScript

TypeScript je programski jezik, ki predstavlja razširitev JavaScripta z možnostjo statične tipizacije. To pomeni, da lahko pred izvajanjem programa preverimo, ali so spremenljivke, funkcije in objekti pravilno uporabljeni glede na njihove tipe, kar lahko zmanjša število napak in pohitri razvoj. Izbrali smo ga zaradi dolgoročne vzdržljivosti projekta. TypeScript zagotavlja večjo varnost pri razvoju z manjšim številom napak, saj preprečuje nepričakovane vrednosti, ki bi lahko sprožile napake pri izvajanju. Poleg tega olajša razumevanje kode, saj je zaradi tipizacije lažje prepoznati namen posameznih delov kode. Podpira dobre prakse razvoja sodobnih aplikacij. TypeScript ima tudi veliko podporo razvijalcev in skupnosti, zato obstaja

veliko orodij in knjižnic, ki olajšajo razvoj v jeziku. Čas razvoja se z uporabo jezika zmanjša, saj je lažje prepoznati in odpraviti napake. Prav tako omogoča lažjo refaktorizacijo kode, saj zaradi tipizacije lažje prepoznamo, kje so potrebne spremembe.

1. 1. 3 tRPC

tRPC (TypeScript RPC) je odprtokodna knjižnica za izgradnjo odjemalcev in strežnikov za oddaljene klice (RPC) med programskimi jeziki. Prinaša številne prednosti. Je zelo lahek in enostaven za uporabo. Z uporabo tRPC-ja enostavno ustvimo RPC storitev, ki se lahko uporablja na več platformah, saj podpira različne protokole prenosa podatkov. Poleg tega je zelo učinkovit, kar pomeni, da lahko aplikacija hitro in učinkovito odgovori na zahteve. Vgrajene so funkcije za ročno upravljanje z napakami in podatkovnimi pretvorbami, kar poenostavlja razvoj. Poleg tega tRPC omogoča enostavno uporabo avtentikacije in avtorizacije. Podpira različne modele podatkov, vključno s težkimi objektno-relacijskimi modeli (ORM) in grafičnimi modeli, kar pomeni, da se lahko prilagodi skoraj vsem potrebam našega projekta. Omogoča tipsko varno komunikacijo s strežnikom na osnovi TypeScripta. Zasnovan je z uporabo sodobnih razvojnih praks, kot so funkcionalne komponente in generiki. To omogoča enostavno in pregledno razvojno izkušnjo, ki omogoča hitro in učinkovito odzivanje na spremembe. tRPC je aktivno vzdrževan in ima veliko skupnost podpornikov in razvijalcev.

1. 1. 4 Tailwind

Tailwind CSS je odprtokodna knjižnica CSS, ki omogoča, da hitro in enostavno oblikujemo spletna mesta z uporabo predhodno definiranih razredov CSS. Tailwind omogoča izredno prilagodljivost in hitrost razvoja. Tailwind je zasnovan tako, da je enostaven za prilagajanje in razširitev. Omogoča nam kreiranje svojih razredov in definiranje globalnih spremenljivk. Knjižnjica je lahka in ne upočasnjuje naše aplikacije. Olajšan je tudi razvoj odzivnih aplikacij za vse širine naprav.

1. 1. 5 Vercel

Vercel je platforma za gostovanje in razvoj aplikacij, prednosti so enostavnost, hitrost in zmogljivost. Zasnovan je tako, da je enostaven za uporabo. Nudi intuitivno uporabniško izkušnjo, ki omogoča hitro in enostavno nalaganje aplikacij in datotek ter enostavno upravljanje z njimi. Je zelo hitra platforma, ki omogoča hitro nalaganje spletnih strani in aplikacij, to pripomore k boljši uporabniški izkušnji. Omogoča prilagajanje okolja in nastavitev, ki so potrebne za našo aplikacijo.

1. 1. 6 PlanetScale

PlanetScale je platforma za upravljanje podatkovnih baz, ki je preprostosta, zmogljiva in skalabilna. Omogoča enostavno skaliranje podatkovne baze. To je zelo pomembno, če bo aplikacija rastla in se bo število uporabnikov povečevalo. PlanetScale je zasnovan tako, da zagotavlja hitro in učinkovito delovanje podatkovne baze. To je pomembno za zagotavljanje hitrega delovanja aplikacije in za izboljšanje uporabniške izkušnje. Vgrajene so funkcije za varnost in zaščito podatkovne baze. Uporablja MySql podatkovno bazo.

1. 1. 7 Prisma

Prisma je ORM (Object-Relational Mapping) in platforma za dostop do podatkov. Ponuja preprost vmesnik, ki olajša uporabo in integracijo s katero koli tehnologijo. Sintaksa je jasno definirana in je intuitivna za uporabo in razumevanje. Omogoča hitrejše in učinkovitejše dostopanje do podatkovne baze in ustvarjanja poizvedb brez potrebe po ročnem pisanju SQL poizvedb. To zmanjšuje čas razvoja in zagotavlja hitrejšo dostavo aplikacije. Prisma uporablja enotni model podatkov, ki olajša prenos podatkov med različnimi platformami in storitvami. To omogoča boljšo modularnost in skalabilnost aplikacije. Zagotavlja varnost podatkov, saj uporablja pripravljene poizvedbe in pripravljene ukaze, ki preprečujejo SQL vbrizgavanje. Ponuja tudi možnosti zaščite podatkov in dovoljenj, kar pomaga pri zagotavljanju, da se podatki hranijo na varnem. Skaliranje aplikacije je enostavno.

1. 1. 8 Clerk

Clerk je platforma za avtentikacijo in upravljanje uporabniških računov. Zagotavlja enostavno integracijo avtentikacije v aplikacijo. Z uporabo lahko enostavno dodamo funkcionalnosti za registracijo, prijavo in ponastavitev gesla. Ponuja sodobno uporabniško izkušnjo za avtentikacijo in upravljanje uporabniških računov, ki jo lahko enostavno prilagodimo. Zagotavlja visoko raven varnosti pri upravljanju uporabniških podatkov. Ponuja vse varnostne funkcije, kot so varnostno kopiranje podatkov, šifriranje in tokenizacija.

1. 1. 9 Open AI API

OpenAI API je zmogljivo orodje umetne inteligence, ki temelji na GPT-3.5 modelu. Omogoča enostavno integracijo naprednih jezikovnih sposobnosti v različne aplikacije in sisteme. Z možnostjo generiranja visokokakovostnih odgovorov na splošna vprašanja, ustvarjanja vsebine, prevajanja in še več, OpenAI API ponuja vsestransko uporabnost. S podporo več jezikov, dokumentacijo in skalabilnostjo predstavlja zanesljivo rešitev. Moč umetne inteligence smo izkoristili za izboljšanje svojih izdelkov in storitev.

1. 1. 10 GitHub API

GitHub API je programski vmesnik, ki omogoča razvijalcem interakcijo z različnimi funkcionalnostmi in podatki GitHub platforme. S pomočjo GitHub API-ja lahko dostopamo do informacij o repozitorijih, uporabnikih, komitih, vprašanjih, potezah itd. Omogoča tudi izvajanje akcij, kot so ustvarjanje, brisanje in posodabljanje repozitorijev, upravljanje vprašanj, upravljanje potez in še več. GitHub API je orodje, uporabno za avtomatizirane procese ali integriracijo GitHub-a v aplikacijo.

2 ORGANIZACIJA DELA

Naša ekipa je sestavljena iz štirih članov, pri čemer smo vsi člani ekipe tudi razvijalci. Da bi zagotovili lažje usmerjanje in usklajevanje med nami, smo se odločili, da en član ekipe prevzame tudi vlogo projektnega vodje.

Glavni komunikacijski kanal je bila platforma Discord, kjer smo se tudi dogovarjali za sestanke in jih tudi izvajali.

Za organizacijo dela smo uporabili metodo Scrum. Razvijali smo v eno tedenskih sprintih. Pred začetkom vsakega sprinta smo imeli sestanek planiranja, ki je potekal preko platforme Discord, kjer smo pregledali seznam zahtevanih funkcionalnosti, imenovan tudi "Product Backlog".

Skupaj smo določili prioritete in izbrali naloge, ki jih želimo izvesti v tem sprintu. Naloge smo nato razdelili med člane ekipe glede na njihove spretnosti in interese, ob upoštevanju skupnih ciljev sprinta. Vsak član ekipe je prevzel odgovornost za določeno število nalog, ki jih je bilo potrebno opraviti v skladu s cilji sprinta.

Naloge je projektni vodja vnesel v Trello, kjer smo vodili svoje naloge in morebitne napake. S pomočjo Trello- ta smo imeli vizualen pregled nad opravljenim delom in preostalimi nalogami. To nam je omogočilo učinkovito sledenje napredku projekta in hitro prepoznavanje morebitnih zamud ali težav.

Vsak dan zjutraj smo imeli »stand up« sestanke preko platforme Discord. Ti sestanki so bili namenjeni kratkemu pregledu napredka, usklajevanju in odpravljanju morebitnih ovir. Trajali so okvirno 15 minut.

Dnevno smo po potrebi imeli kratke sestanke med posameznimi člani, kjer smo si izmenjevali informacije o preprekah in morebitnih težavah pri opravljanju nalog.
Na koncu sprinta smo opravili pregled opravljenega dela. Skupaj smo pregledali, kaj smo uspeli doseči v tem sprintu, ocenili kakovost opravljenega dela ter prepoznali morebitna področja za izboljšave, ter se pripravili na naslednji sprint.
Naša organizacija dela po Scrum je pripomogla k uspešnemu izvajanju projektov, povečala preglednost in usklajenost med člani ekipe ter nam omogočila dosego zastavljenih ciljev v optimalnem času.

3 ZAGOTAVLJANJE KAKOVOSTI

Za zagotavljanje kakovosti rešitve smo uporabljali več pristopov. Naša ekipa je redno preverjala kodo. V primeru, da so se pojavili kakršni koli problemi ali napake, smo hitro ukrepali in jih odpravili. Naša ekipa je tudi redno pregledovala in izboljševala postopke razvoja in testiranja, da smo zagotovili, da je naša metoda razvoja čim bolj učinkovita in zanesljiva.

3. 1 Unit testiranje

Za zaledje naše aplikacije smo napisali teste enot s pomočjo ViTest-a. Uporaba ViTest za testiranje Next.js omogoča enostavno namestitev in uporabo, ima dobro integracijo z Next.js, je hiter in učinkovit, ponuja lepo oblikovana poročila o testiranju, podpira različne vrste testov in omogoča preprosto konfiguriranje.

3. 2 Spremljanje napak

Napake v kodi smo spremljali s pomočjo Axiom-a. Axiom je orodje za spremljanje napak in beleženje dogodkov v aplikacijah. S pomočjo Axioma lahko spremljamo in analiziramo različne vrste napak, vključno z napakami v kodi, težavami s konfiguracijo in drugimi težavami, ki lahko vplivajo na delovanje aplikacije. Poleg tega Axiom zazna tudi druge dogodke, kot so zahteve API-jev, obremenitve strežnikov, operacije v bazi podatkov in podobno.

3. 3 Tehnični dolg

Za odpravljanje tehničnega dolga smo uporabili SonarCloud. SonarCloud je orodje za upravljanje kakovosti kode in vodenje tehničnega dolga. S pomočjo SonarClouda lahko odkrijemo težave v kodi, kot so podvajanje kode, slaba berljivost kode, težave z varnostjo, kritična tveganja in druge težave. Pravtako smo si določili standarde, ki smo jih pri razvoju morali dosegati.