Dizajniranje i razvoj web aplikacije u MERN Stack razvojnom okviru

Šamanić, Lorenzo

Sveučilište u Rijeci – Fakultet informatike i digitalnih tehnologija
Preddiplomski studij informatike

Lorenzo Šamanić

Dizajniranje i razvoj web aplikacije u MERN Stack razvojnom okviru

Završni rad

Mentor: Prof. dr. sc. Nataša Hoić-Božić

Rijeka, rujan 2022.

FAKULTET INFORMATIKE I DIGITALNIH TEHNOLOGIJA
SVEUČILIŠTE U RIJECI

Rijeka, 1.7.2022.

Zadatak za završni rad

Pristupnik: Lorenzo Šamanić

Naziv završnog rada: Dizajniranje i razvoj web aplikacije u MERN Stack razvojnom okviru

Naziv završnog rada na eng. jeziku: Web application design and development in the MERN Stack development framework

Sadržaj zadatka:

Dizajniranje korisničkog iskustva i korisničkog sučelja primjenjuje se na razvoj različitih softverskih proizvoda uključujući web sjedišta, web, mobilne te desktop aplikacije. Iako se u procesu dizajniranja u kontekstu primjene multimedije prvenstveno promišlja o elementima i principima vizualne komunikacije, u stvaranju uspješnog korisničkog iskustva važni su i svi ostali aspekti od sadržaja i funkcionalnosti proizvoda do korištene tehnologije za njegovu izradu i dostavljanje.

Zadatak završnog rada je opisati proces dizajniranja i razvoja web aplikacije u tehnologiji MERN Stack-a odnosno mongoDB, express.js, React i node.js. Za kreiranje izgleda i funkcionalnosti web aplikacije koristiti će se React kao *frontend* Javascript okvir, dok će u *backend*-u izvršavati zahtjevi prema mongoDB bazi podataka. Kao praktični dio rada potrebno je napraviti vlastitu web aplikaciju koja ilustrira proces dizajna prema UX modelu te proces izrade u odabranim tehnologijama.

Mentor

Prof. dr. sc. Nataša Hoić-Božić

Voditelj za završne radove

Doc. dr. sc. Miran Pobar

Zadatak preuzet: 15.7.2022

(potpis pristupnika)

Sadržaj

Sažetak

U ovom završnom radu prikazati će se dizajn i razvoj cijele aplikacije, od ideje do realizacije. Započeti će se s dizajnom u alatu Figma. Potom će se proći kroz *backend* dio te također i kroz *frontend* kako bi se vidjeli svi dijelovi aplikacije i lakše shvatilo kako što radi. Za izradu aplikacije će se od tehnologija koristiti MERN stack te će sve biti prikazano na aplikaciji Trading Bible u kojoj se unose te potom spremaju podatci koji se često ponavljaju u forexu kako bi se potom mogli pregledati i vidjeti što se događalo kroz povijest.

Ključne riječi:

Web aplikacija, Figma, MERN stack, MongoDB, React, JavaScript

1. Uvod

Tema ovog završnog rada je izrada web aplikacije u MERN Stack razvojnom okviru gdje će se detaljnije proći kroz sve procese dizajna i razvoja aplikacije. Krenuti će se s Figmom u kojoj će biti kreirani *wireframe* i *mockup* aplikacije. Proći će se kroz elemente korisničkog iskustva te kroz sve ravnine modela elemenata UX. Potom se prelazi na *backend* odnosno na kreiranje baze podataka i API-a na koji se može spojiti *frontend* strana gdje se prikazuju i unose podatci.

Za izradu cijele aplikacije koristiti se jedan od najpopularnijih *stack*-ova, MERN *stack* odnosno platforma koja objedinjuje bazu podataka MongoDB, Express.js, React koji će biti oplemenjen s Next.js-om te Node.js poslužitelj. Ideja korištenja ovakvog *full stacka* je da se cijela aplikacija može razviti s jednim programskim jezikom, u ovom slučaju s JavaScriptom.

Aplikacija Trading Bible koja je napravljena pomoću MERN-a je aplikacija na temu praćenja događaja u forex-u. Forex odnosno *foreign exchange* je trgovanje vrlo slično dionicama s glavnom razlikom da se ovdje trguje iz jedne valute u drugu, npr. iz dolara u euro po određenom omjeru koji se mijenja iz sekunde u sekundu kroz svih 5 radnih dana. Trgovati se može na mnogo načina, no način koji je u ovom slučaju zanimljiv je tehnička analiza gdje se na grafu promatra kako se je cijena ponašala kroz vrijeme te se traže određeni uzorci koji se često ponavljaju. Još jedna od velikih stvari koje utječu na pomak cijene su vijesti te postoje vijesti koje se redovito ponavljaju, neke na tjednoj, neke na mjesečnoj, a neke na višemjesečnoj bazi. Npr. sredinom svakog mjeseca Australija objavljuje razliku u nezaposlenosti te potom te vijesti utječu na pomak australskog dolara. Također prvi petak u mjesecu dolaze vijesti o non-farm payroll-u koji jako utječu na jačinu dolara. Non-farm payroll je naziv za promjenu u broju novozaposlenih u Sjedinjenim Američkim Državama u odnosu na zadnji utvrđeni podatak te uključuje sva zanimanja osim osoba zaposlenih u poljoprivredi, kućanstvima i ne profitnim organizacijama.

Trading Bible aplikacija pomaže u pohranjivanju ovakvih podataka gdje se takve događaje sortira po nazivima te se potom za svaki može vidjeti povijest kada se je i što dogodilo nakon objave vijesti. Primjerice, ukoliko nekog korisnika zanimaju svi podatci o non-farm payroll-u, nakon otvaranja mu se prikazuju svi prijašnji zapisi gdje može vidjeti detaljan prikaz što se događalo u prošlosti, s cijenom u tom periodu prikazanom pomoću slike i brojevnih zapisa.

2. Dizajn web sjedišta

Kako bi došlo do finalnog rezultata odnosno gotovog web sjedišta, prvi korak je dizajn vizualnog prikaza na osnovu kojeg korisnici mogu vidjeti njen potencijalni krajnji izgled. Za brzi razvoj aplikacije, prvo se ide u razvoj dizajna kako se ne bi dogodilo da se u developmentu stalno mijenjaju funkcionalnosti i izgled. Upravo zbog toga se prvo kreira izgled aplikacije u nekom od programa predviđenih za to poput Figme, Sketch-a, Adobe XD-a i ostalih. To je software kojem je jedna od glavnih namjena izrada dizajna web aplikacija te ima sve potrebne alate koji su potrebni za brz i lak razvoj dizajna.

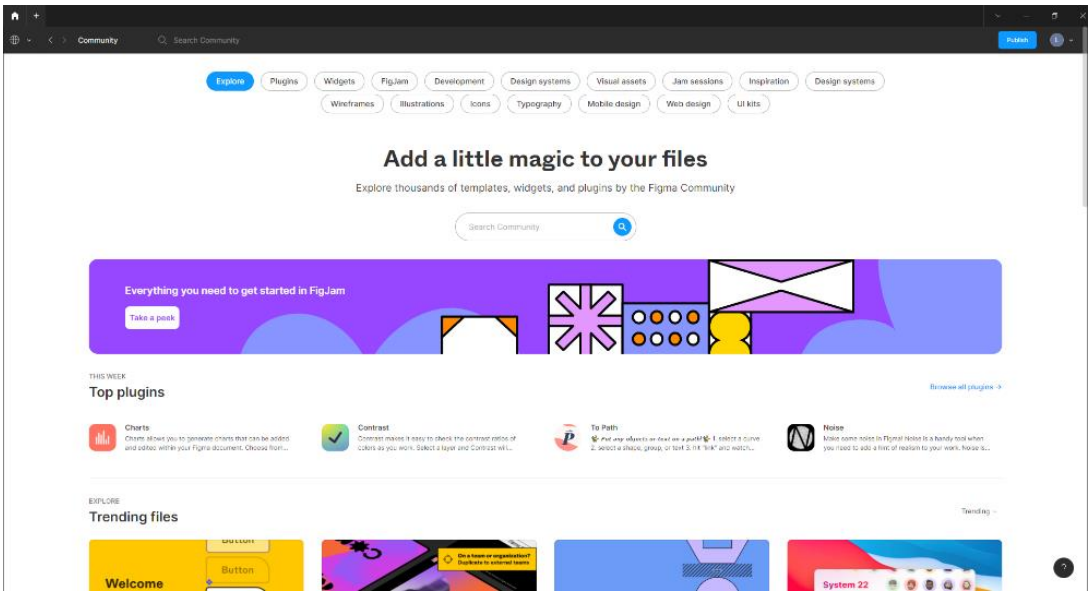
Za dizajn nije bitan samo vizualni dio odnosno UI nego i UX odnosno korisničko iskustvo koje je čak i bitnije. Uz korištenje grafičkih elemenata poput tipografije, boje, geometrijskih oblika, ilustracija ostalih elemenata, jako je bitan dizajn korisničkog iskustva gdje treba razmišljati što će i kako korisnik shvatiti te potom i koristiti. Kako bi poboljšali korisničko iskustvo, usluga mora biti napravljena dovoljno laganom, intuitivnom i lako pamtljivom za korištenje [5].

Bitno je voditi računa o rasporedu sadržaja, isticanju elemenata, standardizaciji, brzini učitavanja te atraktivnosti. Kod rasporeda sadržaja je bitno da korisnik lako može promotriti uredan raspored elemenata kako bi dobio na vremenu i lako shvatio što se događa. Bitno je i istaknuti određene elemente ukoliko je to potrebno kako bi se nešto naglasilo. Standardizacija je također vrlo bitna kako se korisnik ne bi mučio i stalno morao prilagođavati na nešto novo, važno je da je većina elemenata konstanta i dosljedna kroz dizajn. Također, bitno je aplikaciju optimizirati kako bi se učitavala u prihvatljivom vremenu, kao i je li sam dizajn atraktivan, zanimljiv i ugodan oku.

Za razvoj Trading Bible aplikacije odabran je program Figma koji će ukratko biti opisan u nastavku rada.

2.1. Figma

Figma je vrlo moćan dizajnerski alat koji nam pomaže da vrlo brzo i lako kreira bilo što poput web sjedišta, desktop aplikacija, web i mobilnih aplikacija ili nečeg sličnog. Jedna od najvećih prednosti Figme je to što se može koristiti *online* te na taj način omogućava veliku fleksibilnost pri korištenju. Može se svakom dizajnu pristupiti s bilo kojeg mjesta gdje ima interneta te također se dizajniranju može pridružiti bilo tko ukoliko mu se da dopuštenje te potom mogu zajedno u stvarnom vremenu paralelno dizajnirati. Također jedna od bitnijih stvari je mogućnost dodavanja *plugin*-ova pomoću kojih se može ubrzati razvoj dizajna, te se može koristiti zajednica kako bi se dobila inspiracija ili vidjelo kako je netko napravio ono što nekome treba [3]. Nakon gotovog dizajna može se napraviti *prototype* odnosno prikazati kako bi aplikacija prikazivala ekrane ukoliko bi došlo do daljnje realizacije.

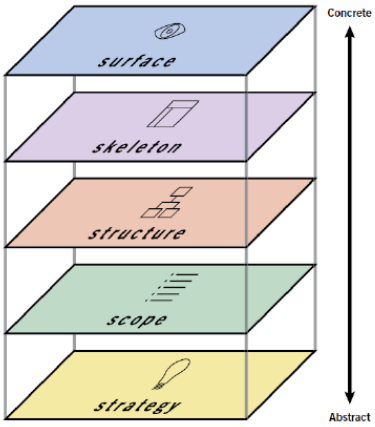


Slika 1 - Figma community

2.2. Model elemenata korisničkog iskustva

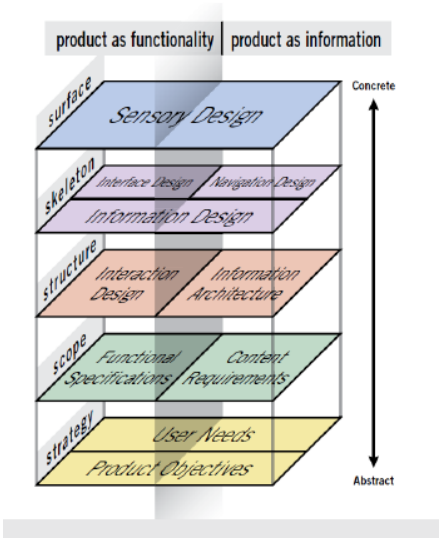
Model elemenata UX je konceptualni okvir za promišljanje o UX i UI dizajnu te je izvorno nastao za web. Sastoji se od pet ravnina (Slika 2) te se prilikom projektiranja pristupa odozdo prema gore jer svaka ravnina ovisi o onoj ispod [5]. Ravnine:

1. Površina (*Surface*)
2. Kostur (*Skeleton*)
3. Struktura (*Structure*)
4. Opseg (*Scope*)
5. Strategija (*Strategy*)



Slika 2 - Osnovni UX model

Proces nije linearan te rad na višoj razini počinje iako još nije dovršen na nižoj razini. Također imamo i potpuni UX model (Slika 3) koji još detaljnije prikazuje elemente pojedine ravnine dijeleći ih prema funkcionalnosti i informaciji.



Slika 3 - Potpuni model elemenata UX

2.3. Ravnina strategije

Kako bi dizajn web sjedišta bio uspješan prvo se treba jasno definirati strategija kako bi se odredili ciljevi proizvoda odnosno u našem slučaju web aplikacije. Treba odrediti što aplikacija mora ostvariti za klijenta te također što se želi postići za korisnike [2].

U slučaju Trading Bible, riječ je o web aplikaciji kojoj je cilj olakšati zapisivanje podataka i lako dohvaćanje istih kroz vrijeme. Podaci koji se zapisuju su brojevi objavljeni na forex kalendaru za neki događaj poput non-farm payroll-a. Zapisuje se i datum događaja kako bi ga se moglo pratiti kroz vrijeme te postoji mogućnost dodavanja slike gdje se vidi na grafu što se je točno dogodilo na burzi nakon objave događaja.

U aplikaciji se događajem smatra samo ime nekog događaj kao primjerice non-farm payroll te potom svaki događaj ima primjere u koje se zapisuju gore navedeni podaci. Takva struktura je odabrana jer se isti događaj ponavlja redovno te zbog toga postoje primjeru u koje se zapisuju skupine podataka povezane s datumom događaja kako bi korisnici mogli pratiti što se zbiva s određenim događajem kroz vrijeme. Korisniku se želi omogućiti jednostavno i intuitivno korištenje aplikacije.

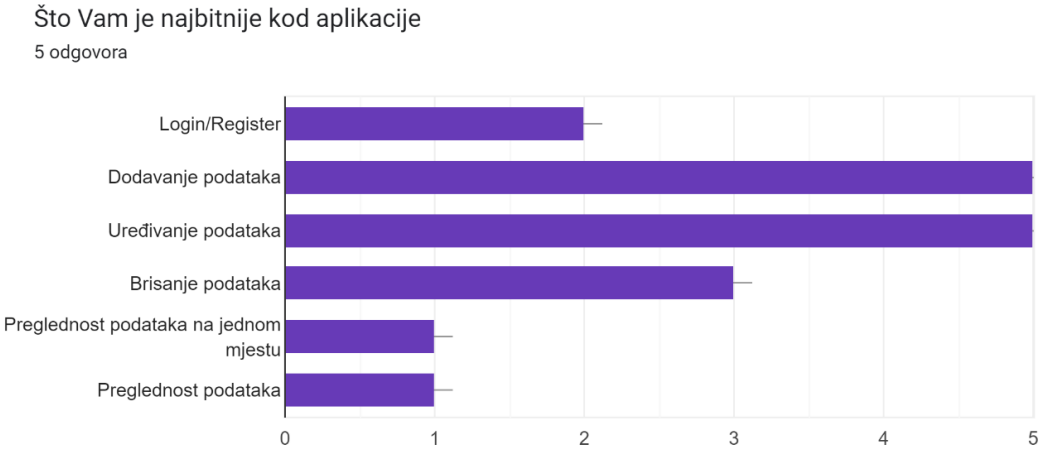
This Week: Aug 28 - Sep 3 — Up Next

Date	8:36pm	Currency	Impact	Detail	Actual	Forecast	Previous	Graph
Sun Aug 28								
Mon Aug 29	All Day	GBP		Bank Holiday				
	8:15pm	USD		FOMC Member Brainard Speaks				
Tue Aug 30	All Day	EUR		German Prelim CPI m/m	0.3%	0.3%	0.9%	
	4:00pm	USD		CB Consumer Confidence	103.2	97.6	95.3	
		USD		JOLTS Job Openings	11.24M	10.37M	11.04M	
	5:00pm	USD		FOMC Member Williams Speaks				
Wed Aug 31	11:00am	EUR		CPI Flash Estimate y/y	9.1%	9.0%	8.9%	
		EUR		Core CPI Flash Estimate y/y	4.3%	4.1%	4.0%	
	2:00pm	USD		FOMC Member Mester Speaks				
	2:15pm	USD		ADP Non-Farm Employment Change	132K	300K	128K	
	2:30pm	CAD		GDP m/m	0.1%	0.1%	0.0%	
	3:45pm	USD		Chicago PMI	52.2	52.5	52.1	
Thu Sep 1	3:45am	CNY		Caixin Manufacturing PMI	49.5	50.1	50.4	
	8:30am	CHF		CPI m/m	0.3%	0.2%	0.0%	
	2:30pm	USD		Unemployment Claims	232K	250K	237K	
	4:00pm	USD		ISM Manufacturing PMI	52.8	52.1	52.8	
Fri Sep 2	2:30pm	USD		Average Hourly Earnings m/m		0.4%	0.5%	
		USD		Non-Farm Employment Change		295K	528K	
		USD		Unemployment Rate		3.5%	3.5%	
Sat Sep 3								

More

Slika 4 - Forex kalendar s podacima [4]

Također prilikom izrade web aplikacije treba voditi računa o brandu odnosno logu, tipografiji, paleti boja te emocijama i dojmovima korisnika kako bi aplikacija bila privlačna korisniku. Kako bi doznali što točno korisnik želi, provedena je analiza na skupini ljudi koji su potencijalni korisnici buduće aplikacije odnosno forex trgovci koji prate forex kalendar te žele imati mogućnost spremanja te potom lakog uvida u podatke kroz povijest koji su im bitni za stvaranje strategije trgovanja. Nakon dobivenih odgovora na upitnik s postavljenim pitanjima, kroz analizu smo došli do sljedećih potreba korisnika (Slika5).



Slika 5 - Prikaz odgovora iz upitnika

Potrebe korisnika:

- Mogućnost prijave korisnika u aplikaciju
- Dodavanje podataka događaja i primjera (naslov, datum, slika, brojevi)
- Uređivanje podataka događaja i primjera (naslov, datum, slika, brojevi)
- Brisanje podataka događaja i primjera (naslov, datum, slika, brojevi)
- Jednostavan prikaz s puno podataka (naslov, datum, slika, brojevi)

Također pomoću ostalih odgovara na pitanja kreirana je persona koja bi najviše sličila krajnjem korisniku Trading Bible aplikacije (Slika 6).

Slika 6 - Persona korisnika

Korisniku se omogućuje jednostavno spremanje podataka o događaju u forexu poput datuma, brojeva i slika događaja. Potom korisnik može pregledati sve spremljene podatke na preglednom mjestu kako bi mogao odlučiti o daljnjim taktikama trgovanja. Aplikacija nema naručitelja već je namijenjena svim trgovcima forex-a kako bi im olakšala trgovanje odnosno svim ljudima koji u bilo kojem obliku već trguju forex-om na nekoj platformi.

Trading Bible aplikacija je dodatan alat trgovca forex-om kojoj je cilj olakšati stvaranje taktike za buduće trgovanje na drugim platformama te sama aplikacije ne nudi mogućnost kupnje i prodaje valuta. Uvidom u prijašnja zbivanja trgovac može špekulirati što će se zbiti ovog puta te si s time povećati šansu za uspješno trgovanje.

2.4. Ravnina opsega

Nakon jasno određenih strateških ciljeva treba se osmisliti kako sve te zahtjeve zadovoljiti odnosno ugoditi i dizajnerima i onome što klijent želi. Nakon uobličavanja potreba korisnika i ciljeva u funkcije koje će proizvod ponuditi korisnicima i posebne zahtjeve za sadržaj, tada strategija postaje opseg [5].

Zahtjevi sadržaja

Korisnik će tražiti:

- Sign in/Sign up stranicu
- Stranicu s pregledom podataka o događajima u forexu, kao što je non-farm payroll
- Detaljniji prikaz podataka po valuti, kao što su GPBUSD (Funta naspram Američkog Dolara), EURUSD (Euro naspram Američkog Dolara) i druge
- Pregled slika događaja

Zahtjevi funkcionalnosti

Korisnik će tražiti:

- Mogućnost prijave i kreiranja računa
- Dodavanje novih događaja poput non-farm payroll-a
- Dodavanje novih primjera unutar događaja
- Uređivanje događaja (naslov i valuta)
- Uređivanje primjera (datum, slika, brojevi)
- Brisanje događaja
- Brisanje primjera
- Mogućnost povećanja slika na cijeli ekran

2.5. Ravnina strukture

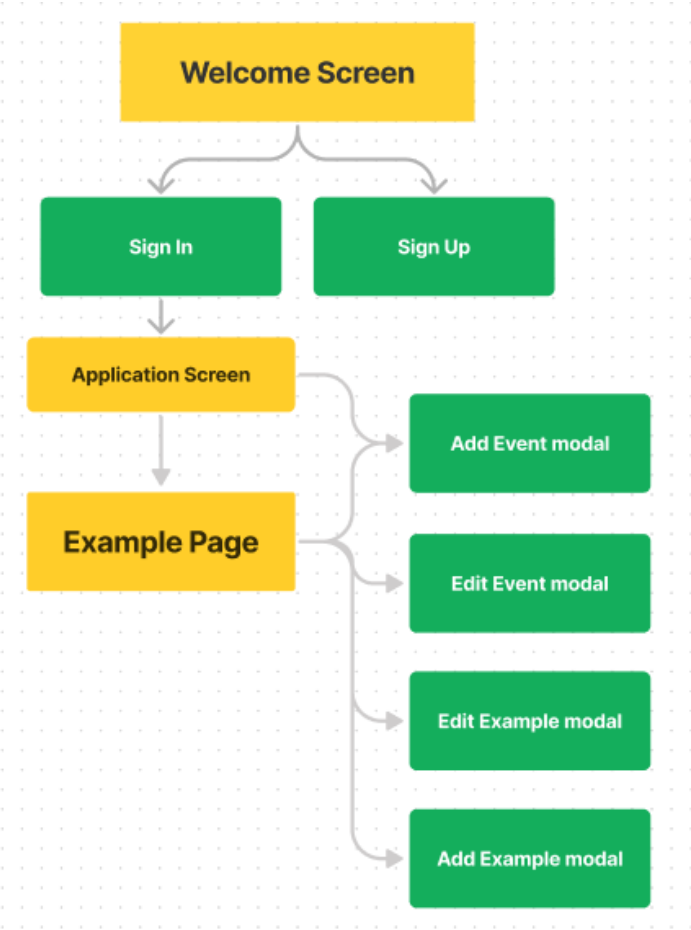
Nakon popisivanja zahtjeva u ravnini opsega koji su proizašli iz strateških ciljeva, definira se konceptualna struktura koja počinje davati oblik mnoštvu zahtjeva. Na osnovu dobivenih zahtjeva i potrebnih informacija koje krajnji produkt treba sadržavati, određuje se kako ih posložiti u koherentnu strukturu [2].

Moguća ponašanja korisnika te definiranje kako će se sustav prilagoditi i odgovoriti na ponašanja opisuje dizajn interakcije. Također, dizajn interakcije obuhvaća upravljanje korisničkim pogreškama odnosno što sustav radi kada se dogode pogreške.

Za proizvode orijentirane na informacije, ali i za one orijentirane na funkcionalnost je važna arhitektura informacija. Bitno je strukturiranje sadržaja te kategorizacija kako bi se korisnik mogao efikasno kretati kroz njega.

Čvor je osnovna jedinica strukture informacija koji može graditi mnogo različitih struktura poput hijerarhijske, linearne, strukture matrica i prirodne strukture.

Trading Bible je strukturno dosta jednostavna aplikacija te se s jedne stranice može otići na većinu ostalih. Glavni razlog tome je da su korisniku vrlo brzo i lako dostupne informacije te da ih ne mora previše tražiti i gubiti se po aplikaciji. Korisnik se nalazi na glavnoj stranici te se pomoću navigacije dolazi do podataka koji se potom izmjenjuju.



Slika 7 - Trading Bible sitemap

2.6. Ravnina kostura

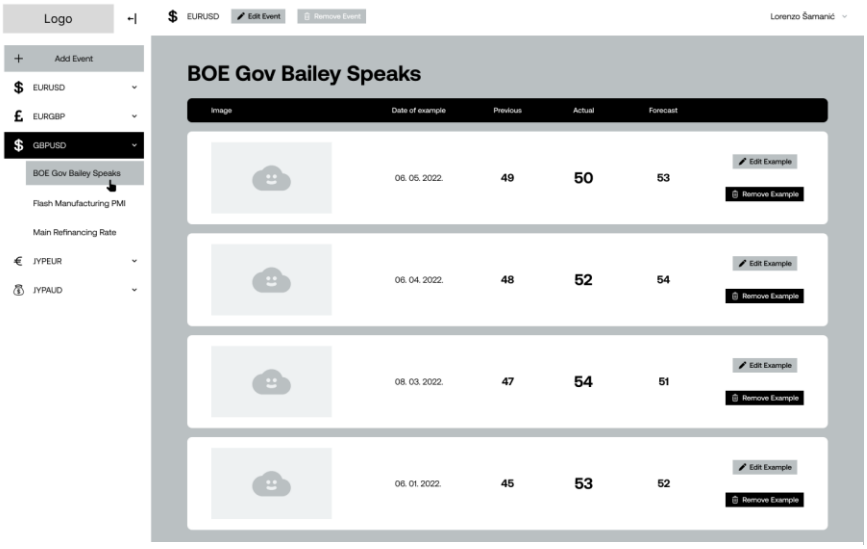
Ravnina kostura dodatno usavršava konceptualnu strukturu definiranu u ravnini ispod. Identificiraju se navigacija, specifični aspekti sučelja ali i informacijski dizajn koji će učiniti apstraktnu strukturu konkretnom. Bitno je misliti o razmještaju koji je najbolji za korisnika odnosno koji omogućuje korisniku maksimalnu efikasnost [5].

U dizajnu sučelja je bitan odabir pravih elemenata sučelja za zadatak koji korisnik treba izvršiti. Također je bitan raspored elemenata na ekranu te je cilj da korisniku aplikacija bude razumljiva i laka za korištenje.

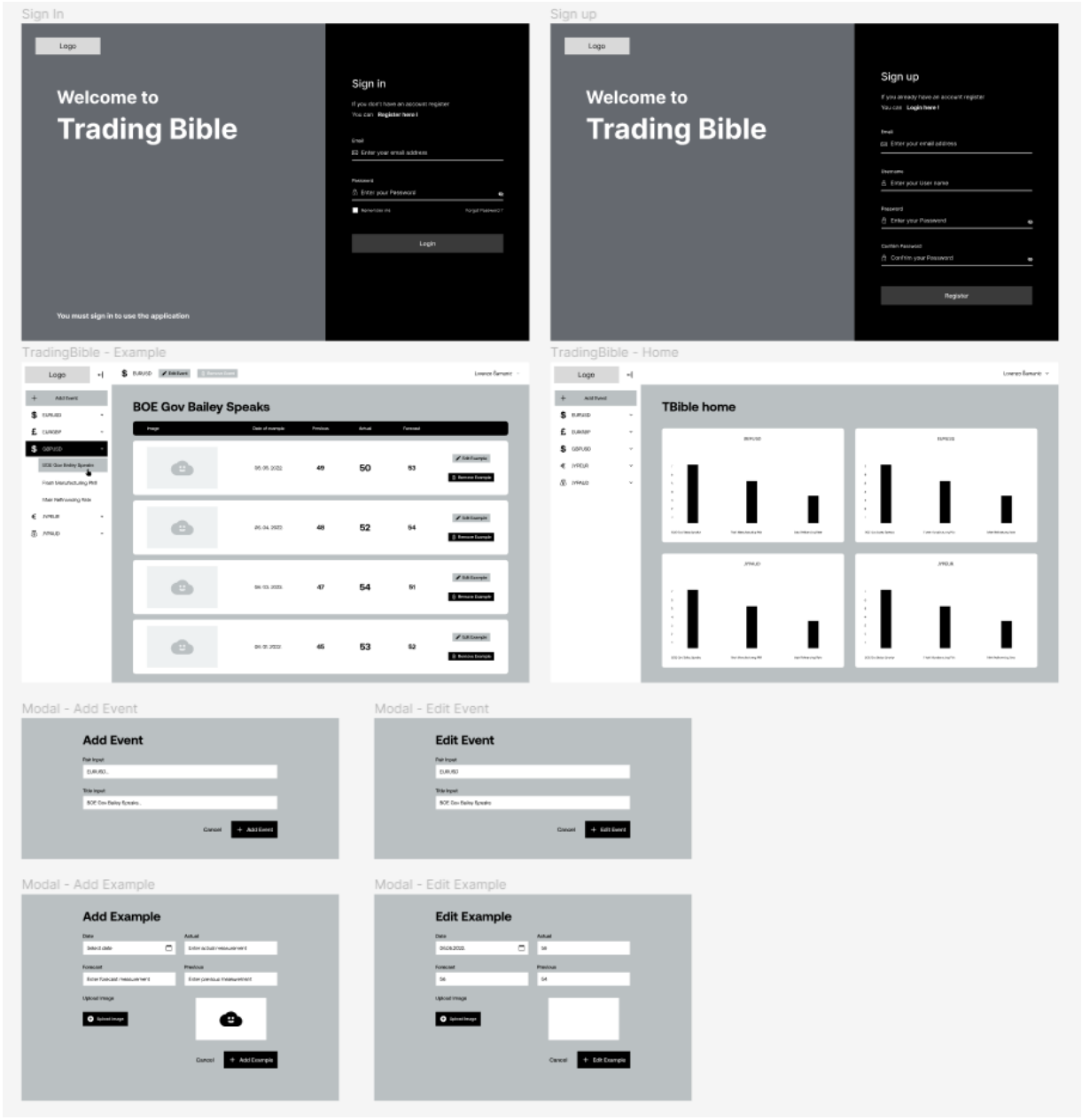
Također bitna stavka je i dizajn navigacije gdje je cilj olakšati kretanje korisnika kroz aplikaciju. Imamo više razina navigacije koje možemo koristiti poput glavne, globalne, lokalne i druge.

Prvi korak u izradi vizualnog dizajna web aplikacije je kreiranje *wireframe*-a. *Wireframe* je dizajn pomoću kojeg se dobiva prvi dojam o tome kako bi nešto trebalo izgledati gdje ne dizajniramo gotov proizvod već je cilj dobiti grubu sliku, odnosno osnovne linije i oblike koji će se koristiti u daljnjem razvoju dizajna sučelja. Prikazuju se samo osnovni elementi i ne brine se o krajnjoj estetici nego o pozicioniranju elemenata. Glavni poticaj izrade *wireframe*-a je brz i lak razvoj koji se može puno puta promijeniti te se ne gubi previše vremena ukoliko se cijela ideja ne realizira.

Dizajn Trading Bible aplikacije je baziran na lakom i preglednom pristupu podataka gdje je sam prikaz podataka vrlo bitan. Cilj je bio napraviti web aplikaciju u kojoj se vrlo brzo i lako mogu vidjeti bitni podatci koji su potrebni korisniku. Dizajn je napravljen tako da se pomoću navigacije kreće kroz parove i naslove te se potom na stranici prikazuju primjeri s detaljnim izvještajem podataka. Dok smo na toj stranici i dalje možemo lako vidjeti u kojem smo paru i naslovu te lako možemo pobrisati, dodavati i izmjenjivati primjere kao i događaje.



Slika 8 - Wireframe prikaza podataka



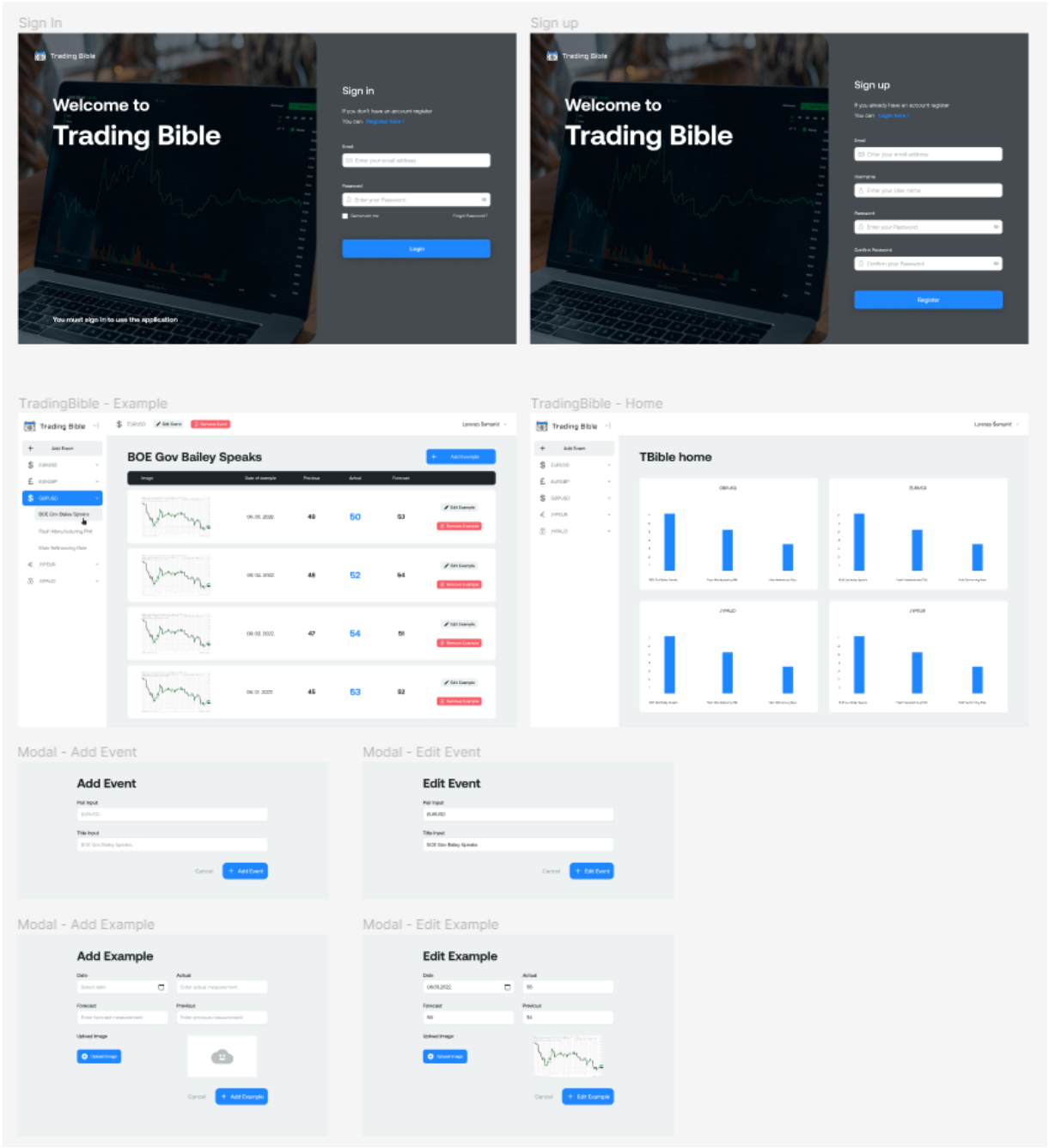
Slika 9 - Wireframe-ovi Trading Bible aplikacije

2.7. Ravnina površine

U zadnjoj ravnini su oni dijelovi aplikacije koje će korisnici prvo primijetiti odnosno vidjeti. Ovdje se sadržaj, funkcionalnost i estetika udružuju kako bi dobili gotov dizajn koji je korisniku ugodan te ispunjava sve ciljeve ostale četiri ravnine [2].

Mockup je nacrt u kojem prikazujemo više vizualnih elemenata poput boje, tipografije i ostalih elemenata uz osnovnu funkcionalnost. Zbog dodavanja elemenata izgled postaje puno atraktivniji te puno bolje dočarava finalni izgled stranice. Lako se uočavaju i rješavaju problemi oko boja, tipografije ili rasporeda i navigacije.

Kod Trading Bible dizajna aplikacije su na temelju *wireframe*-a dodane boje, slike, font i ostali detalji koji vizualno doprinose privlačnosti aplikacije. Izabrane su primarne i sekundarne boje koje se koriste kroz cijelu aplikaciju. Kreirani su standardizirani dijelovi kako bi kroz aplikaciju bili isti te kako bi iskustvo korisnika bilo dosljedno. Boje i font su izabrani kako bi najviše doprinijeli ugodnom osjećaju korištenja aplikacije. Aplikacija koristi puno bijele i varijacije svijetlo sive boje kako korisnika ne bi odvlačili od bitnih podataka koji su istaknuti drugim, lako uočljivim bojama. Korištena je i crna boja kako bi dobili najveći kontrast za prikaz crnog teksta na bijeloj pozadini kao i bijeli tekst na crnoj. Plava boja je izabrana kao primarna boja aplikacije te se koristi kako bi istaknula bitne dijelove aplikacije. Koristi se kao pozadina gumba, u logu, kao istaknuti tekst te također kao pozadina za odabrani tekst kako bi znali što je trenutno izabrano u navigaciji. Također, koristi se i crvena boja kao pozadina u gumbima za brisanje jer sugerira da će se napraviti nešto „opasno," odnosno da će podatak zauvijek nestati.

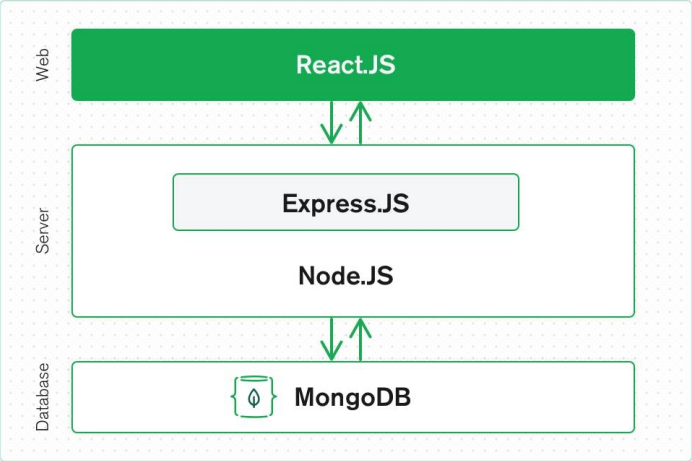


Slika 10 - Mockup-ovi Trading Bible aplikacije

3. Razvoj aplikacije u MERN platformi

MERN *stack* je baziran na JavaScriptu te se koristi kako bi se lakše i brže moglo razvijati *full stack* web aplikacije [7]. MERN se sastoji od 4 tehnologije a to su **M**ongoDB, **E**xpress, **R**eact i **N**ode.js. Osmišljen je kako bi se cijeli proces razvoja olakšao. Svaku od tehnologija će se detaljnije objasniti prema modelu od tri sloja ili razine (Slika 11) [7].

Dakle, MERN se sastoji od tri glavne razine a to su *frontend*, *backend* i baza podataka gdje se kompletno koristi JavaScript i JSON. Na prvoj razini nalazi se React.js odnosno deklarativna biblioteka za kreiranje korisničkih sučelja. Potom Express.js i Node.js gdje je Express brza minimalistička biblioteka za Node.js koja se vrti unutar samog Node.js servera. Naposljetku dolazi sloj baze podataka gdje se nalazi MongoDB odnosno ne relacijska (NoSQL) baza podataka u koju šaljemo podatke iz Reacta preko Express API-a.



Slika 11 - MERN stack

3.1. MongoDB

MongoDB je NoSQL odnosno ne relacijska baza podataka gdje je svaki zapis dokument koji sadrži vrijednost ključa koji je sličan JSON (JavaScript Object Notation) objektima. MongoDB je fleksibilan i dozvoljava korisniku da kreira sheme, baze podataka, tablice i ostalo. Nakon instaliranja MongoDB-a korisnik može koristiti Mongo shell odnosno konzolno sučelje kroz koje može slati zahtjeve za dohvaćanjem, obnavljanjem ili brisanjem podataka. Možemo vidjeti prikaz zapisa iz MongoDB baze podataka (Slika 12) [7].

Slika 12 - MongoDB zapis

3.2. Express.js

Express je Node.js biblioteka. Express omogućava lakše pisanje *backend* koda kako se ne bi morao pisati kod koristeći Node.js. Express pomaže u dizajniranju dobrih web aplikacija i API-a. Express podržava mnoge *middleware*-ove koji programiranje skraćuju i olakšavaju.

```
const express = require('express'),
http = require('http');

const hostname = 'localhost';
const port = 8080;
const app = express();

app.use((req, res) => {
  console.log(req.headers);
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/html');
  res.end('<html><body><h1>This is a test server</h1></body></html>');

});
const sample_server = http.createServer(app);

sample_server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Slika 13 - Express testni server

3.3. React

React je JavaScript biblioteka koja se koristi za izradu korisničkih sučelja. Koristi se zbog svoje sposobnosti da brzo čita i mijenja podatke. React je poznat po virtualnom DOM-u, a to je objekt koji prezentira DOM (Document Object Model) objekt. Virtual DOM je kopija originalnog DOM-a što znači da se prilikom neke modifikacije promjeni virtualni DOM te se potom uspoređuju oba DOM-a i potom se razlike usklade s originalnim DOM-om [2]. Ovime se dobiva da se osvježi samo mali dio koji se je ažurirao te se time štede resursi i vrijeme. Još jedna od bitnijih stvari koje React koristi su komponente. Komponente u React-u su elementi za slaganje korisničkog sučelja gdje svaka komponenta ima svoju logiku i doprinosi cjelokupnom sučelju. Komponente također promoviraju iskorištavanje koda kako se ne bi ponavljao što potom znači da je kod lakši za shvatiti i lakši za učitavanje.

3.4. Node.js

Node.js stvara JavaScript okruženje u kojem se korisniku dozvoljava pokretanje koda na serveru odnosno izvan preglednika. Kod Node.js je važno spomenuti node package manager odnosno npm koji dozvoljava korisniku da bira između više tisuća besplatnih paketa odnosno modula za skidanje koje može koristiti u developmentu.

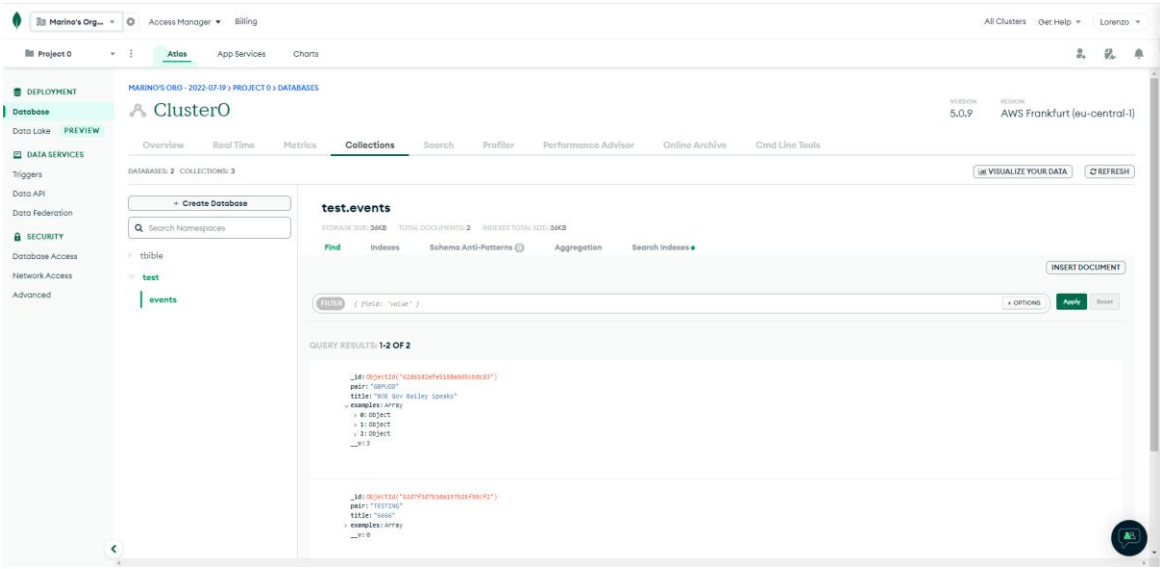
Slika 14 - Node.js logo [10]

4. Backend

Kako bi svaka aplikacija dobro radila treba kvalitetno razviti backend stranu aplikacije. Glavni cilj backend-a je dio aplikacije koji se vrti na strani servera te ga se ne vidi, no svi podatci i komuniciranje s frontend-om ide do backend-a koje osigurava da bi sve s klijentske strane radilo kako treba i prikazivalo podatke koji su zatraženi. Kad se popuni neko polje ili forma, internet preglednik šalje zahtjev serverskoj strani koja mu potom vraća tražene podatke u obliku frontend koda koje preglednik može interpretirati. Glavne komponente backend-a su API i baza podataka. API je programsko sučelje koje pomaže u komunikaciji s backend servisima poput servera. U Trading Bible aplikaciji korišten je MongoDB Atlas kao baza podataka te Apollo Server s GraphQL-om gdje se potom može komunicirati s frontend-om preko Apollo Client-a. Prijenos slika je napravljen pomoću Node.js *middleware*-a Multer koji se primarno koristi za učitavanje datoteka.

4.1. MongoDB Atlas

MongoDB Atlas je baza podataka u oblaku koja je kreirana od strane istih ljudi koji su napravili MongoDB. Atlas pojednostavljuje kreiranje i praćenje baza te mu je velika prednost što je u oblaku i nema potrebe za kreiranjem baze podataka na vlastitom serveru. Može se vidjeti sučelje MongoDB Atlasa (Slika 15) te Trading Bible bazu s pohranjenim podatcima koji se mogu potom prikazati na frontend-u.



Slika 15 - MongoDB Atlas

4.2. Apollo Server - GraphQL

Apollo Server je kod otvorenog tipa koji podržava GraphQL server koji je kompatibilan sa bilo kojim GraphQL klijentom uključujući i Apollo Client kojeg i Trading Bible koristi. Baza je deklarirana u *backend*-u po GraphQL shemi gdje je definirano koje polje je koji tip (Slika 16).

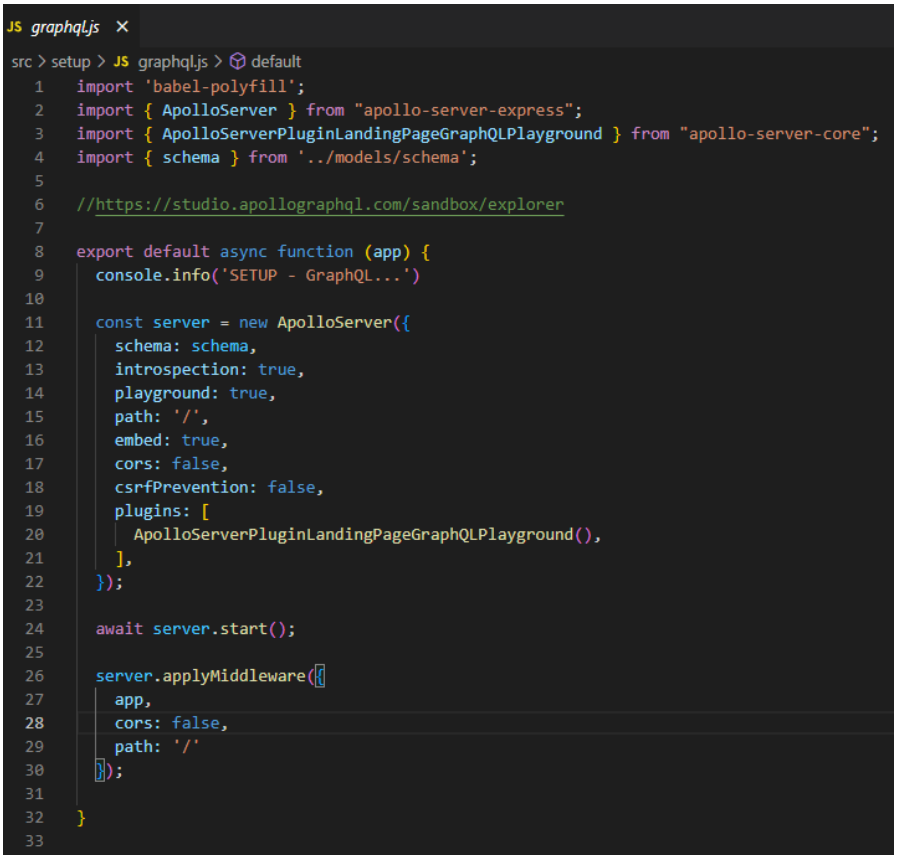
JS event.js

src > models > JS event.js > ...

```
1   import mongoose from "mongoose";
2   import { composeWithMongoose } from "graphql-compose-mongoose";
3
4   const Example = new mongoose.Schema({
5     date: Date,
6     measureActual: String,
7     measureForecast: String,
8     measurePrevious: String,
9     image: String,
10  });
11
12  export const Event = new mongoose.Schema({
13    pair: String,
14    title: String,
15    examples: [Example],
16  });
17
18  export const eventModel = mongoose.model("Event", Event);
19  export const eventTC = composeWithMongoose(eventModel);
20
21  const extendedResolver = eventTC.getResolver("findMany").addFilterArg({
22    name: "titleRegexp",
23    type: "String",
24    description: "Search by regExp",
25    query: (query, value) => {
26      query.title = new RegExp(value, "i"); // eslint-disable-line
27    },
28  });
29  extendedResolver.name = "findMany";
30  eventTC.addResolver(extendedResolver);
```

Slika 16 - Shema i resolver

Baza se sastoji od Eventa u kojem se nalaze varijable pair i title koje su obje string, te od examples-a koji je niz. Unutar examples-a se nalaze varijable date koja je po tipu Date, te mesureActual, mesureForecast, mesurePrevious i image koje su sve string-ovi. Nakon deklariranja tipova varijabli treba se definirati i resolver jer Apollo Server ne zna da bi trebao koristiti te podatke prilikom izvršavanja upita. Nakon definirane sheme, podataka i resolvera moramo dati i podatke Apollo Serveru za inicijalizaciju (Slika 17).



Slika 18 - Inicijalizacija Apollo Server-a

Nakon svega može se lokalno pokrenuti http server ili se može deklarirati da prilikom developmenta bude podignut lokalno, dok se prilikom *productiona* može koristiti hosting server na nekoj odabranoj domeni (slika 18). Ovdje se može vidjeti upravo to te također istu stvar i za MongoDB bazu koja se u *productionu* spaja na MongoDB Atlas.

Slika 17 - Inicijalizacija u developmentu i productionu

4.3. Multer – učitavanje slika na server

Multer je Node.js *middleware* koji se primarno koristi za učitavanje datoteka [9]. Multer dodaje body objekt i file objekt u request objekt. Body objekt sadrži vrijednosti tekstualnih vrijednosti iz forme dok file objekt sadrži datoteke učitane u formi.

JS upload.js ×

src > setup > JS upload.js > ...

```
1   // Imports
2   import path from 'path'
3   import multer from 'multer'
4   import bodyParser from 'body-parser'
5   var express = require('express')
6
7
8   export default function (server) {
9     console.info('SETUP - UploadServer...')
10
11    // Request body parser
12    server.use(bodyParser.json())
13    server.use(bodyParser.urlencoded({ extended: false }))
14
15    var uploadsPath = path.join(__dirname, '../uploads');
16    server.use('/uploads', express.static(uploadsPath));
17
18    var storage = multer.diskStorage({
19      destination: function (req, file, callback) {
20        callback(null, uploadsPath);
21      },
22      filename: function (req, file, callback) {
23        callback(null, Date.now() + '_' + file.originalname.toLowerCase());
24      }
25    });
26    var upload = multer({ storage: storage }).single('file');
27
28    server.post('/uploads', function (req, res) {
29      upload(req, res, function (err) {
30        if (err) {
31          return res.end("");
32        }
33        res.end(req.file.filename);
34      });
35    });
36
37  }
```

Slika 19 - Multer učitavanje slika

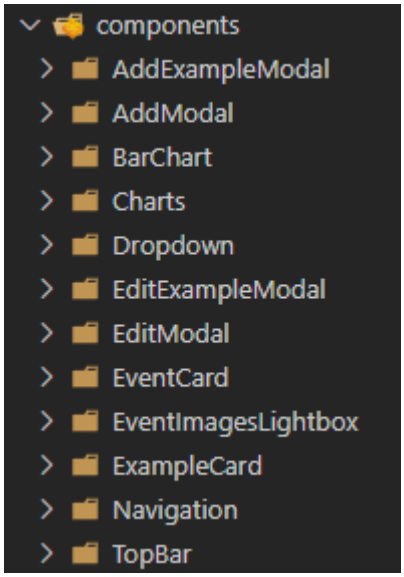
(Slika 19) Ovdje se može vidjeti kako multer radi u *backend*-u aplikacije Trading Bible. Sve slike biti će učitane na server u direktorij */uploads* što je deklarirano varijablom uploadsPath. Također možemo vidjeti imenovanje same aplikacije koja se imenuje tako da se uzme vrijeme učitavanja slike koje se nadoda na originalno ime slike kako ne bi došlo do problema ukoliko se učitava više slika s istim početnim imenom.

5. Frontend

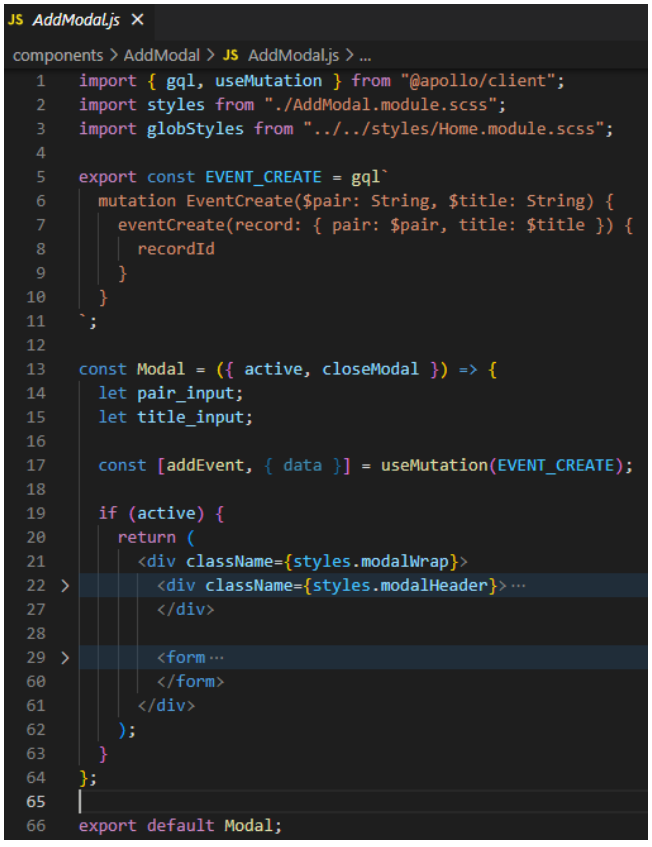
Frontend je dio aplikacije koji se vizualno vidi i koji omogućuje lagano korištenje aplikacije bez da se zna kako i zašto nešto u pozadini radi. To je klijentska strana gdje je bitno korisničko sučelje.. Osnove svakog web sjedišta i aplikacije su HTML, CSS i JavaScript, no MERN stack koristi JavaScript biblioteku React.js. Za izradu Trading Bible aplikacije odlučeno je da će se koristiti još jedna JavaScript biblioteka po imenu Next.js koja koristi React.js.

5.1. Next.js

Next.js je jedna od najpoznatijih biblioteka otvorenog koda za React koja je izgrađena na Node.js-u te omogućuje React baziranim web aplikacijama funkcionalnosti poput server-side rendering-a i generiranja statičnih web sjedišta. Server-side rendering učitava HTML na svakom upitu, dok se kod statičnog generiranja HTML generira na početku te se potom ponovno iskorištava na svakom upitu. Također bitna stavka React-a je da se kod piše koristeći komponente. Zbog toga se u aplikaciji modali odnosno prozori za dodavanje i uređivanje kao zasebne komponente (Slika 20) koje se mogu potom pozvati u glavnoj index.js datoteci.



Slika 20 - Komponente Trading Bible aplikacije



Slika 21 - Modal za dodavanje eventa

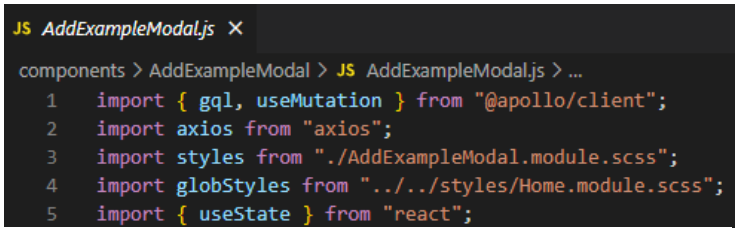
5.2. Apollo Client

Apollo Client je JavaScript biblioteka koja omogućuje upravljanje lokalnim i udaljenim podatcima s GraphQL-om. Koristi se za dohvaćanje, mijenjanje i brisanje podataka. Preko GraphQL-a postoje dvije mogućnosti za slanje upita a to su query i mutate. Query se koristi za dohvaćanje podataka te se time dobivaju podatci iz baze podataka koji su zatraženi dok se pomoću mutacija mogu brisati i uređivati podatci te se mogu i slati podatci u bazu. U sljedećoj slici (Slika 22) se može vidjeti korištenje Apollo Client-a već u prvoj liniji gdje se iz njega uključuje dio gql koji treba za kreiranje query-a i mutacija. Prvi QUERY_EVENTS služi kako bi se dobili svi podatci o eventu te ih se dalje može prikazivati gdje trebaju. Sljedeća je mutacija REMOVE_EVENT kojoj se šalje id eventa kako bi se mogao obrisati te potom ima EDIT_EVENT gdje se šalje id eventa kojeg se želi mijenjati te također se šalje cijeli zapis kako bi ga se promijenilo.

Slika 22 - Prikaz query-a i mutacija

5.3. Axios – učitavanje slika na server

Axios je asinkrono baziran HTTP klijent za preglednik i Node.js koji može slati http zahtjeve preko Node.js-a. U aplikaciji se koristi kako bi se poslala slika na server stoga se koristi samo POST zahtjev [1].



Slika 23 - Import Axios-a

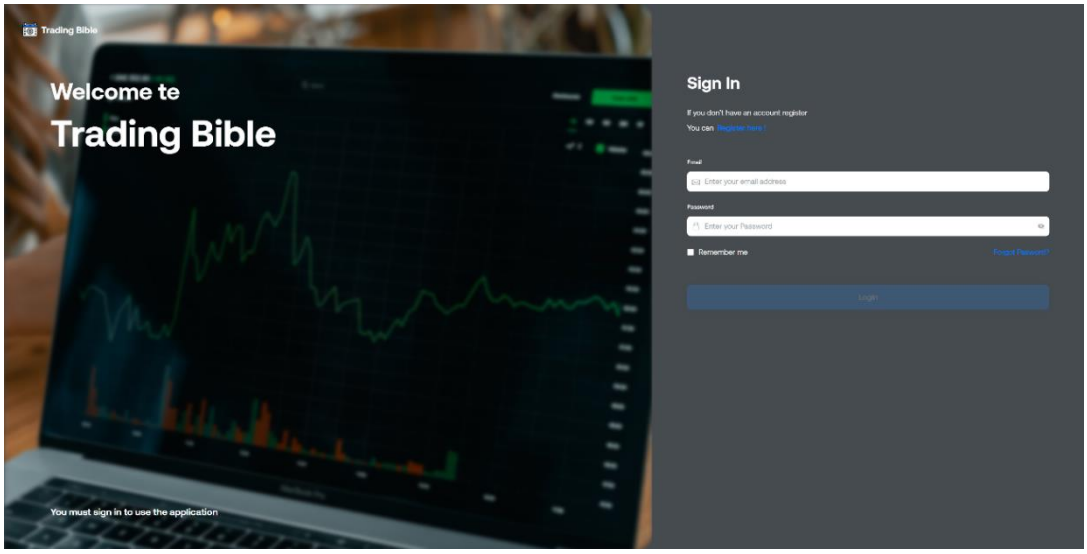
U sljedećoj slici (Slika 24) se može vidjeti kako se koristi axios u Trading Bible aplikaciji. Prvo se pregledava ima li uopće odabrana slika te ukoliko ima šalje se na server te se čeka odgovor od multer-a. Nakon što multer izvrši svoj dio pohranjivanja slike, on vraća novo ime slike odnosno ono ime koje je pohranjeno na serveru te potom ide s daljnjim dodavanjem podataka u bazu.

```
31    if (file) {
32      axios
33        .post("https://api.tbible.club/uploads", formData, {})
34        .then((response) => {
35          updateEventById({
36            variables: {
37              id: event?._id,
38              record: {
39                examples: [
40                  ...(event?.examples || []),
41                  {
42                    date: date,
43                    measureActual: measureActual,
44                    measureForecast: measureForecast,
45                    measurePrevious: measurePrevious,
46                    image: response.data,
47                  },
48                ],
49              },
50            },
51          });
52        })
53        .catch((err) => {
54          console.log("aeraer", err);
55        });
56    } else {
```

Slika 24 - Axios učitavanje slike

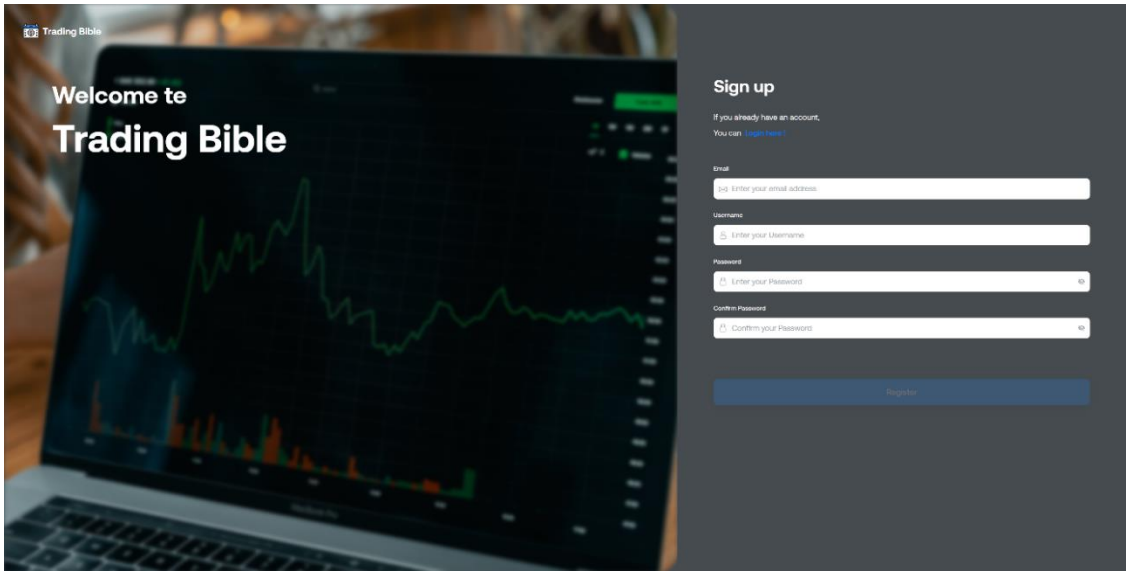
6. Trading Bible Aplikacija

U ovom poglavlju će se prikazati kako aplikacija radi odnosno kako bi ju korisnik koristio. Aplikacija se nalazi na Vercelu odnosno na linku: https://lbible.vercel.app/, te se API nalazi na linku: https://api.tbible.club/. Upisivanjem gore navedenoga linka kako bi se došlo na Trading Bible aplikaciju dolazi se na Sign In ekran odnosno na stranicu za prijavu kako bi se dalje moglo nastaviti s korištenjem aplikacije. (Slika 25).



Slika 25 - Sign In stranica

Ukoliko korisnik još nema svoj račun s kojim se može logirati, klikom na „Register here!“ dolazi se na Sign Up stranicu gdje se može unesti podatke za registraciju kako bi se mogao kreirati novi račun s kojim se potom može logirati u aplikaciju (Slika 26).

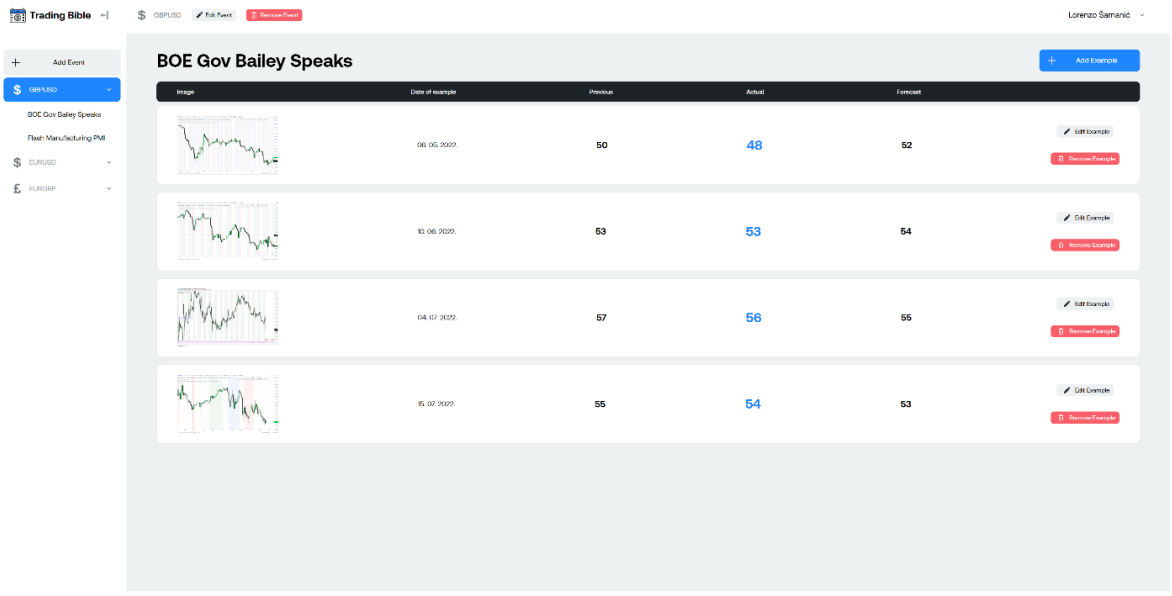


Slika 26 - Sign Up stranica

Ulaskom u samu aplikaciju dolazimo na početnu stranicu gdje nam se prikazuju grafovi s parovima valuta kako bi se moglo vidjeti koliko se događaja nalazi u kojem paru (Slika 27).

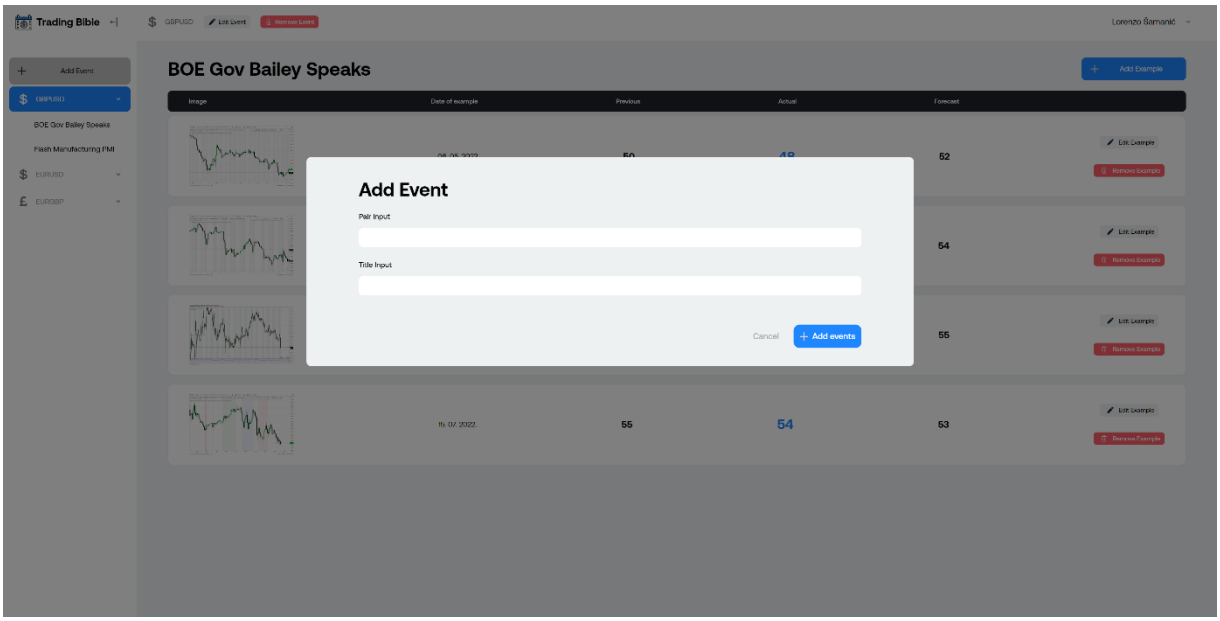
Slika 27 - Home stranica

Odabirom nekog para u lijevom navigacijskom panelu prikazuju se svi događaji unutar tog para te odabirom nekog događaja se prikazuju svi primjeri tog događaja.



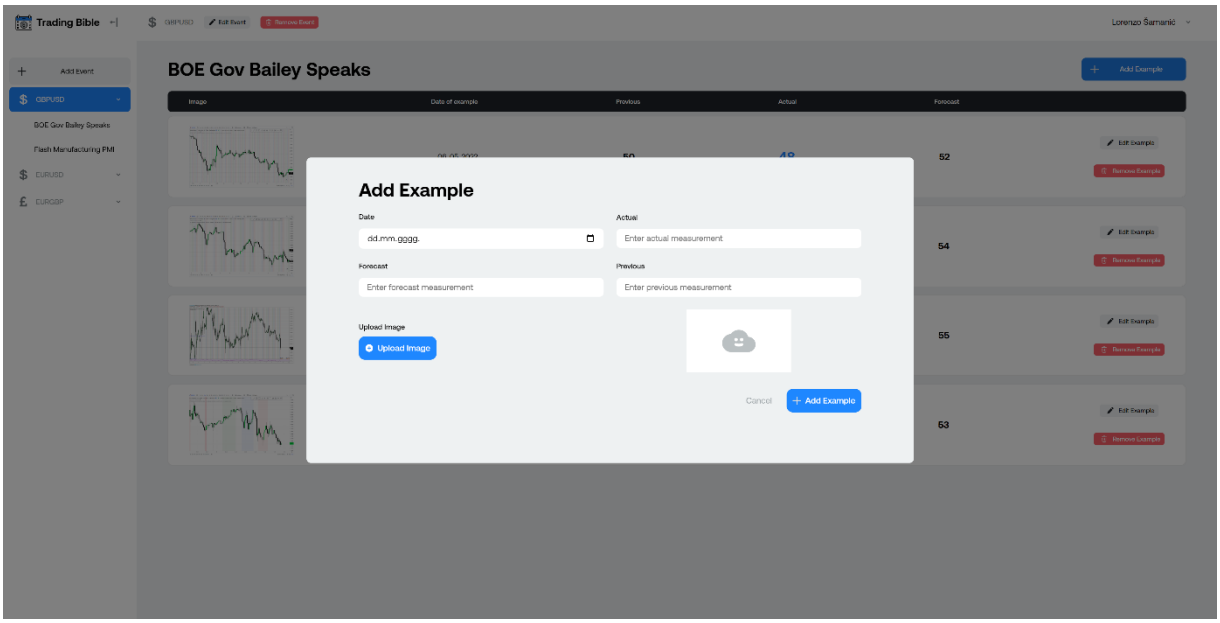
Slika 28 - Prikaz primjera događaja u paru GBPUSD

Klikom na gumb Add Event dolazi nam prozor za dodavanje događaja (Slika 29).



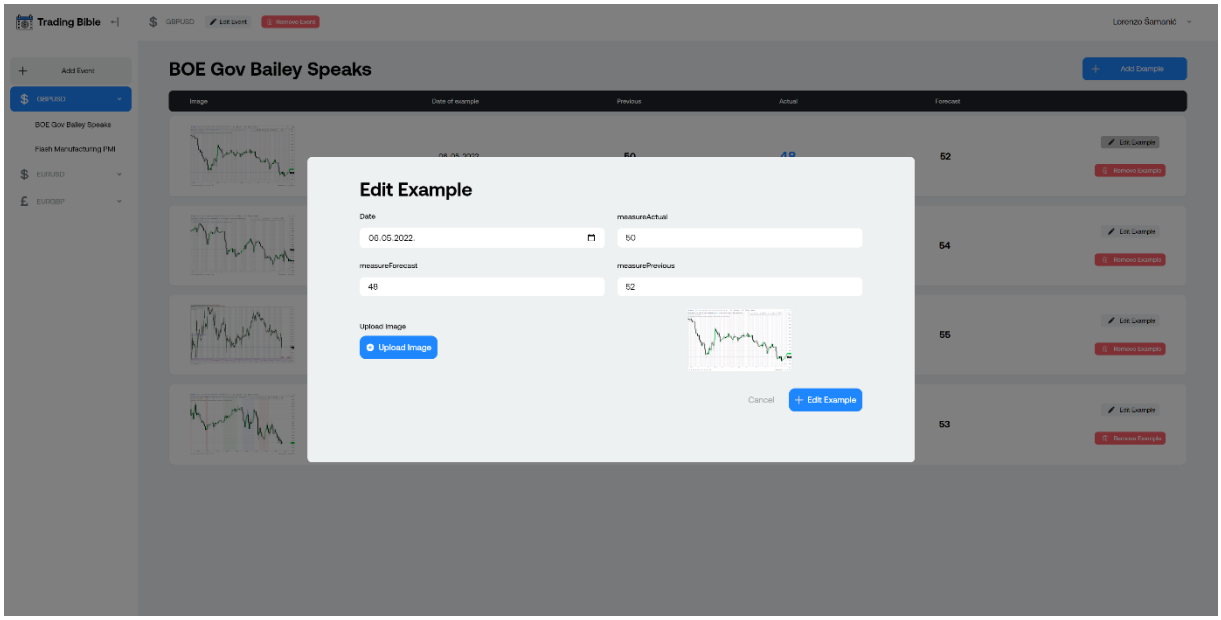
Slika 29 - Dodavanje događaja

Klikom na gumb Add Example dolazi nam prozor za dodavanje novih primjera unutar odabranog događaja (Slika 30).



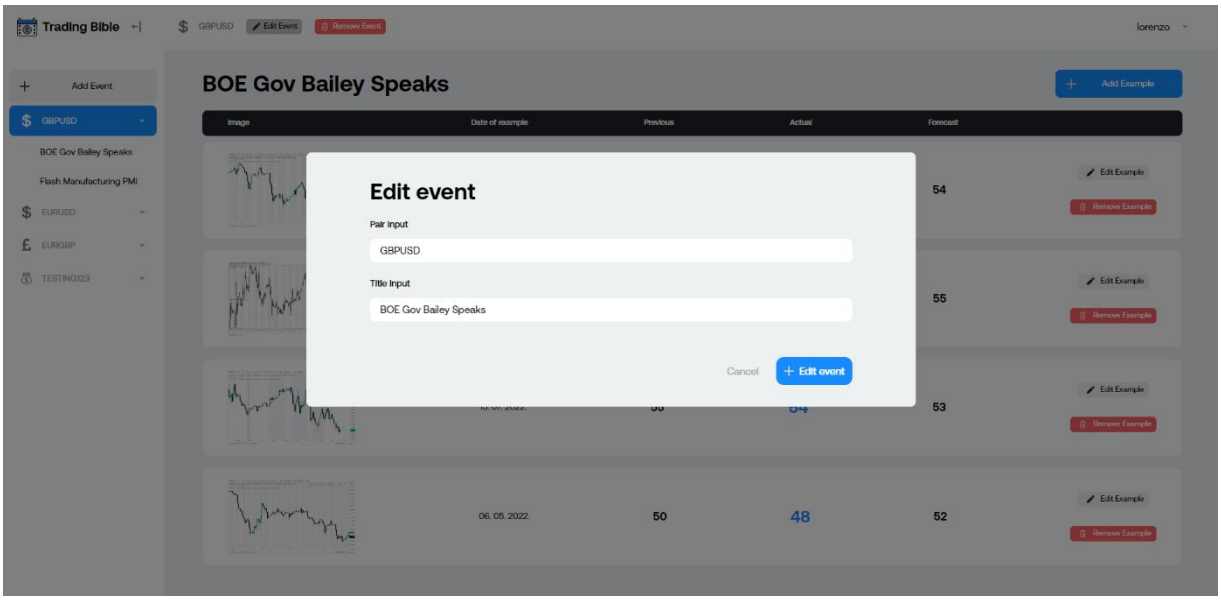
Slika 30 - Dodavanje primjera unutar događaja

Kod primjerima imamo mogućnost brisanja i uređivanja klikom na gumbove Edit i Remove Example. Nakon klika na gumb Edit Example dolazi nam prozor za uređivanje odabranog primjera (Slika 31).



Slika 31 - Uređivanje primjera

Također imamo mogućnost brisanja i uređivanja događaja klikom na gumbove Edit i Remove Event. Nakon klika na gumb Edit Event dolazi nam prozor za uređivanje odabranog događaja (Slika 32).



Slika 32 - Uređivanje događaja

7. Zaključak

U ovom završnom radu opisan je cijeli proces razvoja jedne web aplikacije, od ideje do realizacije odnosno od dizajniranja i kreiranja *wireframe*-a i *mockup*-a do samog programiranja. Opisan je cijeli UX i UI proces pomoću modela elemenata UX te je svaka ravnina razrađena kako bi vidjeli kako smo došli do krajnjeg dizajna. Pojašnjeno je što je i za što koristi MERN *stack* te je opisano čemu služi koja tehnologija. Također je prikazana većinu koda koji tvori Trading Bible aplikaciju te se vide svi glavni principi i tehnologije kojima se koristi.

Unatoč prijašnjem poznavanju razvoja web aplikacija pomoću drugih tehnologija i korištenjem tradicionalnog HTML-a, CSS-a i JavaScript-a, odabrao sam MERN *stack* kao razvojni okvir za razvijanje nove aplikacije kako bi stekao neka nova znanja u svijetu web aplikacija.

Kompletnu aplikaciju sam napravio od početka te mi je to prvi put da spajam dizajn i razvoj aplikacije istovremeno. Prilikom izrade sam prošao kroz sve faze dizajniranja prema UI/UX modelu gdje sam puno naučio o tome kako razmišljati kao korisnik te potom udovoljiti tim zahtjevima. Dizajniranje *wireframe*-a i *mockup*-a u Figmi mi je bio najzabavniji dio u procesu dizajniranja aplikacije. Nakon dizajniranja sam prešao na razvoj aplikacije u MERN *stack*-u gdje sam također stekao puno iskustva u rješavanju problema, naučio neke nove koncepte, paradigme i principe programiranja. Osobno mi je ovo prvo susretanje s razvojem *full stack* aplikacije gdje sam sve kreirao iz početka: od podešavanja servera do kreiranja baze podataka u oblaku, dodavanju datoteka na server te potom dohvaćanju svih podataka s klijentske strane i potom kreiranju korisničkog sučelja kako bi sve radilo kako treba. Smatram da će mi novostečeno znanje u dizajniranju i razvoju aplikacija biti od velike koristi u daljnjem osobnom razvoju te uvelike pomoći pri kompetentnosti na tržištu rada i u struci.

8. Literatura

1. *AXIOS*. n.d. https://axios-http.com/ (pokušaj pristupa 20. Srpanj 2022).
2. Costello, Vic. *Multimedia Foundations*. Taylor and Francis, 2016.
3. *Figma*. n.d. https://www.figma.com/ (pokušaj pristupa 19. Srpanj 2022).
4. *Forex Factory*. n.d. https://www.forexfactory.com/ (pokušaj pristupa 1. Rujan 2022).
5. Garrett, Jesse James. *The Elements of User Experience*. 2002.
6. *Medium*. n.d. https://javascript.plainenglish.io/react-the-virtual-dom-comprehensive-guide-acd19c5e327a (pokušaj pristupa 19. Srpanj 2022).
7. *MongoDB*. n.d. https://www.mongodb.com/mern-stack (pokušaj pristupa 19. Srpanj 2022).
8. *MongoDB*. n.d. https://www.mongodb.com/what-is-mongodb (pokušaj pristupa 19. Srpanj 2022).
9. *NPM*. n.d. https://www.npmjs.com/package/multer (pokušaj pristupa 19. Srpanj 2022).
10. *Wikipedia*. n.d. https://en.wikipedia.org/wiki/Node.js (pokušaj pristupa 20. Srpanj 2022).