Dizajn i razvoj sustava za upravljanje odnosima s klijentima

Nikšić, Mateo

Sveučilište u Rijeci, Fakultet informatike i digitalnih tehnologija

Sveučilišni prijediplomski studij Informatika

Mateo Nikšić

Dizajn i razvoj sustava za upravljanje odnosima s klijentima

Završni rad

Mentor: Doc. dr. sc. Lucia Načinović Prskalo

Rijeka, rujan 2023.

ZADATAK ZA ZAVRŠNI RAD

Sveučilište u Rijeci
FIDIT Fakultet informatike i digitalnih tehnologija
www.inf.uniri.hr

Rijeka, 18.4.2023.

Zadatak za završni rad

Pristupnik: Mateo Nikšić

Naziv završnog rada: Dizajn i razvoj sustava za upravljanje odnosima s klijentima

Naziv završnog rada na engleskom jeziku: Design and development of a customer relationship management system

Sadržaj zadatka: Zadatak završnog rada je opisati postupak dizajna te razvoja sustava za upravljanje odnosima s klijentima korištenjem modernih tehnologija za izradu web aplikacija. U radu će biti opisane faze razvoja sustava: analiza karakteristika postojećih sustava, dizajn korisničkog sučelja i izrada prototipa, dizajn i implementacija baze podataka, razvoj klijentskog dijela aplikacije te sve korištene tehnologije.

Mentor

Doc. dr. sc. Lucia Načinović Prskalo

Načinović Prskalo

Voditelj za završne radove

Doc. dr. sc. Miran Pobar

Zadatak preuzet: 25.4.2023.

Mateo Nikšić

(potpis pristupnika)

Adresa: Radmile Matejčić 2
51000 Rijeka, Hrvatska
Tel: +385(0)51 584 700
E-mail: ured@inf.uniri.hr
OIB: 64218323816
IBAN: HR1524020061400006966

uniri

SAŽETAK

Završni rad naziva “Dizajn i razvoj sustava za upravljanje odnosima s klijentima” obrađuje temu i opisuje postupak dizajna te razvoja web aplikacije odnosno sustava za upravljanje odnosima s klijentima korištenjem modernih tehnologija za izradu web aplikacija. Postupak opisuje četiri faze razvoja web aplikacije: analiza karakteristika postojećih sustava, dizajn korisničkog sučelja i izrada prototipa, dizajn i implementacija baze podataka, razvoj klijentskog dijela aplikacije. Tehnologije koje se koriste pri izradi web aplikacije su Supabase koji u pozadini koristi PostgreSQL bazu podataka i React knjižnica te popratni paketi poput React Router, React Context API, React Query, React Hook Form, React Error Boundary.

Ključne riječi: sustav za upravljanje odnosima s klijentima, web aplikacija, supabase, postgresql, rest api, frontend, react, react router, react context api, react query, react hook form, react error boundary, javascript, html, css, styled components, figma.

ABSTRACT

The thesis titled "Design and Development of a Customer Relationship Management System" addresses the topic and describes the process of designing and developing a web application, specifically a customer relationship management (CRM) system, using modern web application development technologies. The process includes four phases of web application development: analysis of existing software solutions, user interface and prototype design, database design and implementation, and client-side application development. Technologies used in web application development include Supabase, which uses a PostgreSQL database in the background, and the React library with associated packages such as React Router, React Context API, React Query, React Hook Form, and React Error Boundary.

Keywords: customer relationship management system, web application, supabase, postgresql, rest api, frontend, react, react router, react context api, react query, react hook form, react error boundary, javascript, html, css, styled components, figma.

SADRŽAJ

1. UVOD 1
2. ANALIZA KARAKTERISTIKA POSTOJEĆIH SUSTAVA 3
2.1. Karakteristike UntitledCRM sustava 5
3. DIZAJN KORISNIČKOG SUČELJA I IZRADA PROTOTIPA 6
3.1. O korištenim alatima 6
3.2. Korisnička persona 6
3.3. Dijagram korisničkog toka 7
3.4. Dizajn hijerarhije informacija 9
3.5. Dizajn korisničkog sučelja 11
3.6. Usporedba prototipa i konačne web aplikacije 12
3.6.1. Prikazi i objašnjenje ekrana 12
4. DIZAJN I IMPLEMENTACIJA BAZE PODATAKA 28
4.1. O korištenim tehnologijama 28
4.2. EV dijagram baze podataka 28
4.3. Relacijski model baze podataka 30
4.4. Implementacija baze podataka 30
5. RAZVOJ KLIJENTSKOG DIJELA APLIKACIJE 33
5.1. O korištenim tehnologijama 33
5.2. Postavke i struktura projekta 34
5.3. Objašnjenje i primjeri koda 37
5.3.1. Komponenta “App.jsx” 37
5.3.2. Komponenta “ContactTable.jsx” 39
5.3.3. Kuka “useContactsTable.js” 41
5.3.4. Usluga “apiContact.js” 42
5.3.5. Komponenta “DealForm.jsx” 44
5.3.6. Komponenta “DashboardCard.jsx” 45
5.3.7. Komponenta “main.jsx” 46
5.3.8. Komponenta “ErrorFallback.jsx” 47
6. ZAKLJUČAK 49
7. LITERATURA 50
8. POPIS SLIKA 52
9. POPIS TABLICA 54
10. POPIS PRILOGA 55

1. UVOD

Pronalazak potencijalnih klijenata, održavanje odnosa sa postojećim i prethodnim klijentima je cilj svake tvrtke koja želi izgraditi dugoročne i kvalitetne poslovne odnose te povećati prihode, a jedan od boljih načina za postizanje takvog cilja je korištenje sustava za upravljanje odnosima s klijentima (engl. *customer relationship management system*).

Ovim radom opisan je postupak dizajna i razvoja pojednostavljenog sustava za upravljanje odnosima s klijentima naziva UntitledCRM korištenjem modernih tehnologija za izradu web aplikacija. Ključne karakteristike pojednostavljenog sustava uključuju osnovnu analitiku, upravljanje kontaktima i ponudama, dodavanje članova u radni prostor te postavke profila korisnika i tvrtke, a razvoj je podijeljen u četiri faze:

1. analiza karakteristika postojećih sustava,
2. dizajn korisničkog sučelja i izrada prototipa,
3. dizajn i implementacija baze podataka,
4. razvoj klijentskog dijela aplikacije.

Ukratko, u prvoj i drugoj fazi cilj je bio definirati temeljne karakteristike koje svaki sustav za upravljanje odnosima s klijentima treba sadržavati i izraditi funkcionalan prototip koji pruža moderno korisničko sučelje te jednostavno korištenje aplikacije, dok se u drugoj i trećoj fazi na temelju prototipa dizajnira baza podataka te razvija aplikacija.

Za bazu podataka je korištena otvorena i besplatna platforma naziva Supabase koja uvelike olakšava razvoj aplikacija jer na temelju baze podataka automatski stvara REST API koji omogućava manipulaciju podataka poput unosa, čitanja, izmjene i brisanja podataka putem HTTP zahtjeva te uklanja potrebu za razvojem poslužiteljskog dijela (engl. *backend*) aplikacije.

Za razvoj klijentskog dijela (engl. *frontend*) aplikacije korištena je popularna i moderna biblioteka React koja se temelji na Javascript skriptnom jeziku uz popratne pakete te jezici HTML (engl. *HyperText Markup Language*) i CSS (engl. *Cascading Style Sheets*) koji se koriste za oblikovanje hijerarhijske strukture i određivanje izgleda stranice.

U narednom poglavlju "Analiza karakteristika postojećih sustava" detaljno su opisane karakteristike postojećih rješenja za upravljanje odnosima s klijentima, njihova osnovna svojstva te su posebno istaknuta obilježja UntitledCRM sustava koji je predmet istraživanja u ovom radu.

Poglavlje "Dizajn korisničkog sučelja i prototipa" obuhvaća opis korištenih alata za dizajn korisničkog sučelja i prototipa, potencijalnih korisnika odnosno korisničkih persona, dijagram toka, dizajn hijerarhije informacija te korisničkog sučelja, uključujući i prikaz loga te palete boja, uz usporedbu prototipa s konačnom web aplikacijom.

Nakon dizajna korisničkog sučelja i prototipa, korak dalje u procesu uključuje odabir alata i tehnologija za izradu baze podataka, stvaranje dijagrama baze podataka i relacijskog modela, te konačno implementaciju same baze podataka. Ovaj cijeli proces detaljno je opisan u poglavlju "Dizajn i implementacija baze podataka".

Razvoj klijentskog dijela aplikacije detaljno je opisan u poglavlju "Razvoj klijentskog dijela aplikacije" koje pruža informacije o korištenim tehnologijama, postavkama, i strukturi samog projekta, uz objašnjenja i primjere koda zbog boljeg razumijevanja načina rada aplikacije.

Zaključak ovog rada pruža dublji uvid u razmišljanje koje je usmjeravalo različite faze razvojnog procesa, argumentirajući zašto su odabrane određene tehnologije, alati i metode.

2. ANALIZA KARAKTERISTIKA POSTOJEĆIH SUSTAVA

Pretraživanjem ključnih riječi na internetu možemo utvrditi da na tržištu postoji mnoštvo sustava za upravljanje odnosima s klijentima od kojih su pojedina rješenja specijalizirana za određenu industriju ili primjenu, primjerice rješenja specijalizirana za upravljanje odnosima između pravnog i fizičkog lica (engl. *business to customer*) ili rješenja specijalizirana za upravljanje odnosima između pravnih lica (engl. *business to business*), rješenja specijalizirana za internet trgovine ili marketinške agencije. Većina tvrtki koje razvijaju sustave za upravljanje odnosima s klijentima fokusiraju se na razvoj rješenja koja su primjenjiva neovisno o industriji odnosno generičkih rješenja kako bi obuhvatili što veće tržište te upravo pod takva rješenja možemo svrstati i UntitledCRM.

Za definiranje temeljnih karakteristika sustava provedena je analiza konkurentskih aplikacija na način da se za aplikacije koje nude opciju besplatnog perioda testiranja napravi korisnički račun i zabilježe snimke zaslona te se zabilježene snimke zaslona grupiraju s obzirom na funkcionalnost, a zatim se detaljnom analizom odrede zajedničke odnosno temeljne karakteristike koje svaki sustav za upravljanje odnosima s klijentima mora sadržavati kako bi bio funkcionalan. Za analizu karakteristika odabrane su tri postojeće aplikacije: Nutshell, Hubspot i Pipedrive, a prema autoru pripadaju u među 7 najboljih alata na tržištu (Bianco, 2023).

S obzirom da navedene aplikacije postoje dugi niz godina te su razvijene od strane stručnjaka, uz osnovne funkcionalnosti nude i mnoštvo naprednih funkcionalnosti, primjerice automatizirano slanje SMS poruka te e-pošte, stvaranje kalendara te rezervacija, napredno filtriranje podataka i stvaranje prilagođenih izvješća, obavljanje internetskih transakcija te razne integracije s uslugama trećih strana (Wikipedia, 2023).

Na temelju testiranja, usporedbe, analize i zabilježenih snimki zaslona konkurentskih aplikacija dolazimo do zaključka da temeljne funckionalnosti web aplikacije odnosno sustava za upravljanje odnosima s klijentima moraju uključivati sustav za autentifikaciju koji uključuje funkcionalnosti kao što su prijava, registracija te promjena lozinke pojedinog korisnika, nadzornu ploču koja prikazuje osnovne informacije o ponudama i napretku ponuda kroz prodajni lijevak u određenom vremenskom periodu, upravljanje kontaktima

odnosno spremanje osnovnih informacija o kontaktima i ponudama, dodavanje novih članova u radni prostor te postavljanje korisničkog i poslovnog profila.

Slika 1: *Grupiranje snimki zaslona s obzirom na temeljne funkcionalnosti*

Na slici 1 prikazan je način grupiranja zabilježenih snimki zaslona s obzirom na funkcionalnosti, a prema prethodno donesenim zaključcima o temeljnim i minimalnim karakteristikama koje sustav za upravljanje odnosima s klijentima mora zadovoljavati u nastavku se nalazi tablica karakteristika za UntitledCRM sustav.

2.1. Karakteristike UntitledCRM sustava

U tablici 1 nabrojane su i opisane karakteristike sustava za upravljanje odnosima s klijentima naziva “UntitledCRM”.

Tablica 1: **Karakteristike UntitledCRM sustava**

Naziv sustava		UntitledCRM
Opis sustava		Sustav za upravljanje odnosima s klijentima
Naziv karakteristike		**Opis karakteristike**
1.	Autentifikacija korisnika	Registracija novih korisnika aplikacije, prijava postojećih korisnika aplikacije.
2.	Nadzorna ploča	Filtriranje s obzirom na vremenski period te prikaz statistike o financijskoj vrijednosti otvorenih, zatvorenih, izgubljenih i nekvalificiranih ponuda, tablica sa prikazom stadija prodajnog lijevka te broju ponuda i kumulativne financijske vrijednosti za određeni stadij, tortni grafikon sa prikazom distribucije stadija prodajnog lijevka.
3.	Upravljanje kontaktima	Stvaranje, čitanje, ažuriranje i brisanje kontakata. Pregled pojedinog kontakta i ponuda koje pripadaju kontaktu. Prikaz tablice kontakata te pretraga kontakata s obzirom na ime ili prezime kontakta.
4.	Upravljanje ponudama	Stvaranje, čitanje, ažuriranje i brisanje ponude. Filtriranje ponuda s obzirom na status ponude te stadij u prodajnom lijevku.
5.	Dodavanje članova	Registracija novih korisnika koji pripadaju radnom prostoru. Svi korisnici radnog prostora imaju jednake ovlasti.
6.	Korisničke postavke	Promjena postavki profila korisnika poput imena, prezimena te avatara. Promjena postavki korisničkog računa poput adrese e-pošte i lozinke.
7.	Postavke aplikacije	Unos i izmjena informacija tvrtke povezane sa radnim prostorom te postavljanje karakteristika radnog prostora poput valute.

Izvor: Autor

Dodatna karakteristika koja nije navedena pod temeljne karakteristike sustava je opcija svijetlog ili tamnog načina rada, a svrha je prilagođavanje korisničkim preferencijama kako bi korisnik bio zadovoljniji prilikom korištenja aplikacije.

3. DIZAJN KORISNIČKOG SUČELJA I IZRADA PROTOTIPA

Dizajn korisničkog sučelja je proces koji dizajneri koriste za oblikovanje sučelja softvera, a uključuje definiranje rasporeda elemenata i stila pojedinih komponenata, kao što su tipografija, sheme boja, gumba i sličnih elemenata, s ciljem da impresioniraju, privuku i zadrže korisnike izgledom aplikacije (Lamprecht, 2023).

Izrada prototipa omogućava vizualizaciju i testiranje korisničke putanje prije nego što uložimo resurse poput vremena i novca u razvoj proizvoda (Hotjar, 2023.).

3.1. O korištenim alatima

Za dizajn korisničkog iskustva i izradu prototipa korišten je alat naziva Figma. Figma je popularan alat koji omogućava korisnicima poput dizajnera i programera da dizajniraju, podijele i testiraju različite vrste dizajna za digitalne proizvode kako bi mogli napraviti brže i bolje odluke prije nego što se upuste u razvoj digitalnog proizvoda (Figma, 2020.).

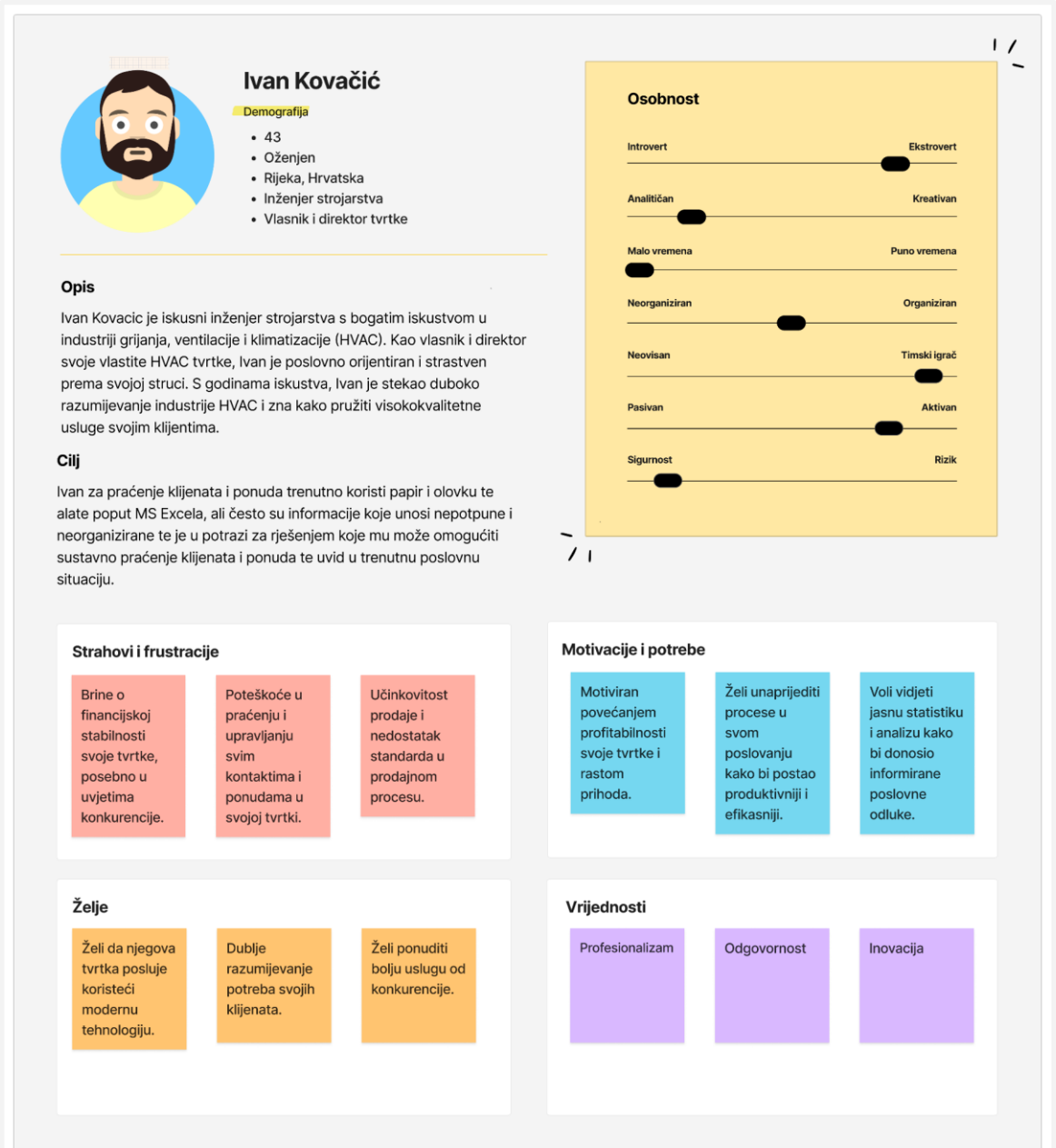
3.2. Korisnička persona

Kao što je prethodno navedeno, UntitledCRM sustav je jednostavna web aplikacija općenite namjene te korisnici aplikacije mogu biti vlasnici ili zaposlenici tvrtki koje posluju u različitim industrijama.

Korisnička persona je definicija korisnika digitalnog proizvoda koja može biti izmišljena, stvorena na temelju istraživanja ili osnovana na temelju intervjua s trenutnim ili potencijalnim korisnicima (Marković, 2019).

S obzirom da je aplikacija koju razvijamo jednostavna zbog nedostatka ljudskih i vremenskih resursa, ne postoji više od jednog načina korištenja aplikacije jer su svi korisnici međusobno jednaki odnosno imaju jednake ovlasti sa jednakim potrebama i ciljevima te se zbog toga ne pojavljuje potreba za stvaranjem više od jedne korisničke persone. Jedina razlika u načinu korištenja aplikacije odnosno korisničkoj putanji između korisnika koji se

sam registrirao, izvornog korisnika i člana radnog prostora je to što je član radnog prostora dodan odnosno registriran od strane izvornog korisnika te poslije dodavanja u radni prostor ima jednake ovlasti kao izvorni korisnik.

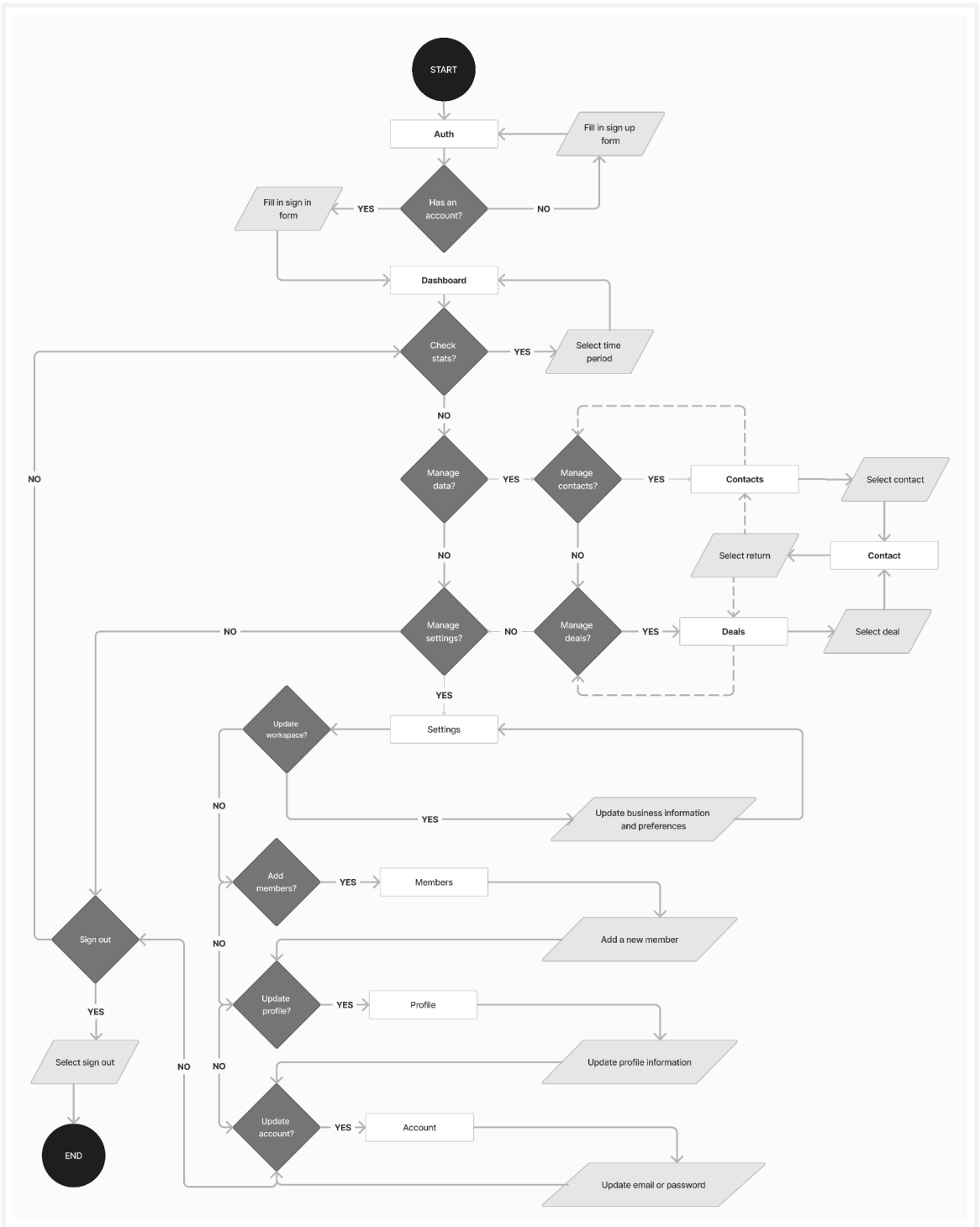


Slika 2: *Korisnička persona "Ivan Kovačić"*

Na slici 2 prikazana je korisnička persona, a za izradu je korišten predložak u alatu Figma.

3.3. Dijagram korisničkog toka

Dijagram korisničkog toka prikazuje svaki korak koji korisnik napravi kako bi ostvario željeni cilj od početka do kraja interakcije, a svrha je evaluacija postojećeg ili stvaranje novog digitalnog proizvoda s intuitivnim korisničkim sučeljem te bolje razumijevanje kako će korisnik koristiti aplikaciju (Browne, 2023).



Slika 3: *Dijagram korisničkog toka za korisničku personu "Ivan Kovačić"*

Na slici 3 prikazan je dijagram korisničkog toka te možemo uočiti različite simbole poput kruga koji označava početak ili kraj interakcije, dijamanta koji predstavlja odluku koju korisnik mora donijeti, pravokutnika koji označava proces ili ekran, paralelograma koji zahtjeva korisnički unos podataka te linije koje povezuju dva simbola i označavaju smjer toka procesa, a iscrtana linija označava alternativni put (Cflowapps, 2023).

Korisnik interakciju započinje na ekranu "Auth". Ako korisnik nema postojeći račun mora ispuniti obrazac za registraciju koji se nalazi na tom ekranu te nakon ispunjavanja ponovno dolazi na ekran "Auth" i s obzirom da ovaj put korisnik ima postojeći korisnički račun ispunjava obrazac za prijavu te dolazi na ekran "Dashboard". Korisnik mora donijeti odluku želi li pogledati statistiku za određeni vremenski period ili ne. U slučaju da želi, korisnik odabere vremenski period za koji želi vidjeti statistiku te se ponovno nalazi na ekranu "Dashboard", ali ovaj put s novim informacijama.

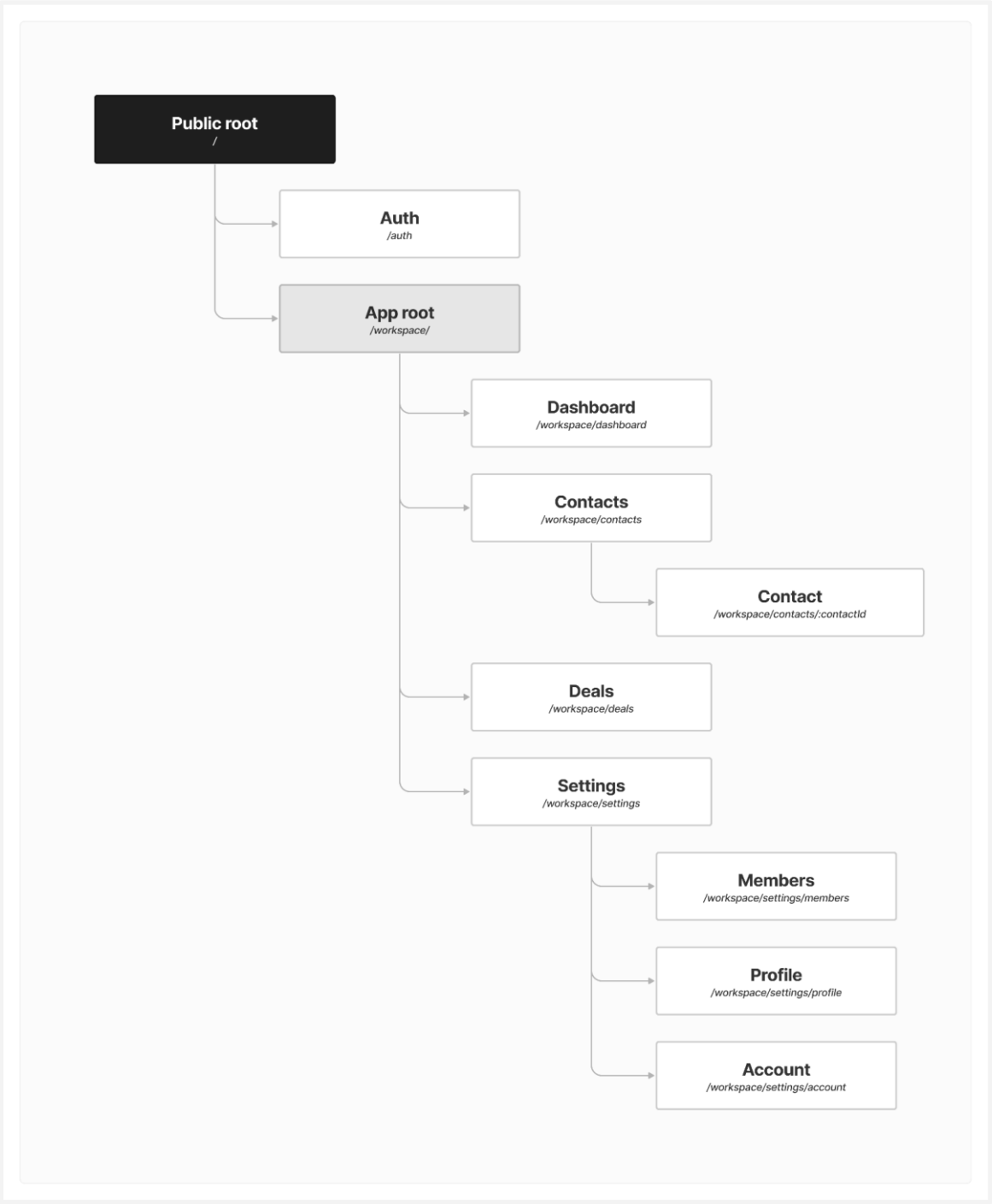
U suprotnom, korisnik može odabrati želi li upravljati s podacima ili postavkama. Ako želi upravljati podacima, mora odabrati želi li upravljati s kontaktima na ekranu "Contacts" ili ponudama na ekranu "Deals" te nakon odabira, ovisno o ekranu na kojem se nalazi korisnik može odabrati da vidi detalje o kontaktu ili ponudi na način da odabere određeni kontakt ili ponudu što ga vodi na ekran "Contact". Korisnik se može odabirom opcije "Return" vratiti na prethodni ekran.

Na ekranima "Contacts" i "Deals" korisnik može vidjeti sve kontakte ili ponude te stvarati nove, ažurirati i brisati postojeće zapise. Ako korisnik želi upravljati postavkama, može odabrati koje postavke želi promijeniti te na kojim ekranima, a ponuđeni ekrani su "Settings", "Members", "Profile" te "Account".

Ako korisnik odluči da ga ne interesira statistika, ne želi upravljati kontaktima niti promijeniti postavke, može prilikom odluke "Sign out" odabrati opciju "Sign out" te završiti interakciju.

3.4. Dizajn hijerarhije informacija

Mapa aplikacije je vizualna i hijerarhijska reprezentacija svih stranica web aplikacije sa svrhom pokazivanje odnosa između njih (Ahmed, 2023).

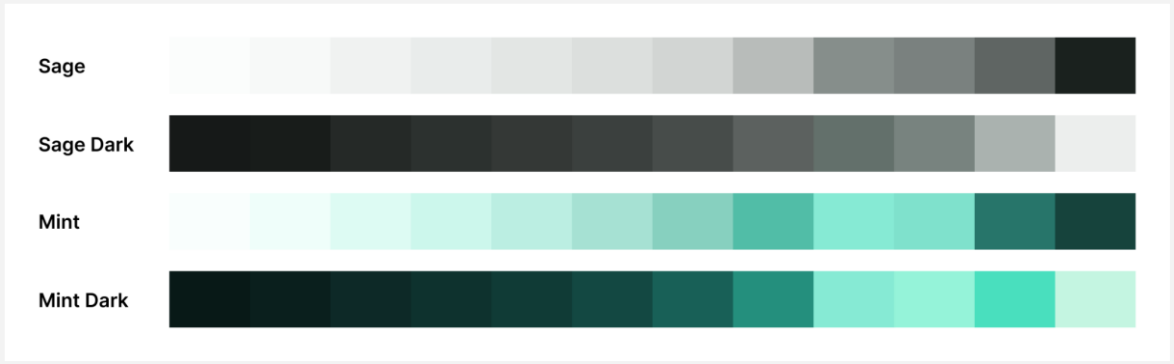


Slika 4: *Hijerarhija informacija ili mapa aplikacije*

Na slici 4 prikazana je hijerarhija informacija, a u nastavku slijedi objašnjenje. Prilikom posjete korijena "/" web aplikacije korisnik je automatski preusmjeren na stranicu ili ekran "Auth". Sve stranice koje su zaštićene te zahtijevaju autentifikaciju korisnika počinju s "/workspace". Nakon uspješne prijave, korisnik je preusmjeren na stranicu "Dashboard", a nakon odjave korisnik je ponovno preusmjeren na stranicu "Auth".

3.5. Dizajn korisničkog sučelja

"Inter" je naziv primarnog fonta aplikacije koji je posebno dizajniran za ekrane računala te je jedan od popularnijih fontova koji se koriste u web aplikacijama. Sustav boja aplikacije je napravljen od strane tvrtke "WorkOS" pod nazivom projekta "Radix" s namjerom da sustav boja bude vizualno privlačan, pristupačan te automatski podržava tamni način rada.



Slika 5: *Paleta boja*

Na slici 5 prikazana je paleta boja za svijetli i tamni način rada, korištena prilikom dizajna korisničkog sučelja aplikacije.

Slika 6: *Logo*

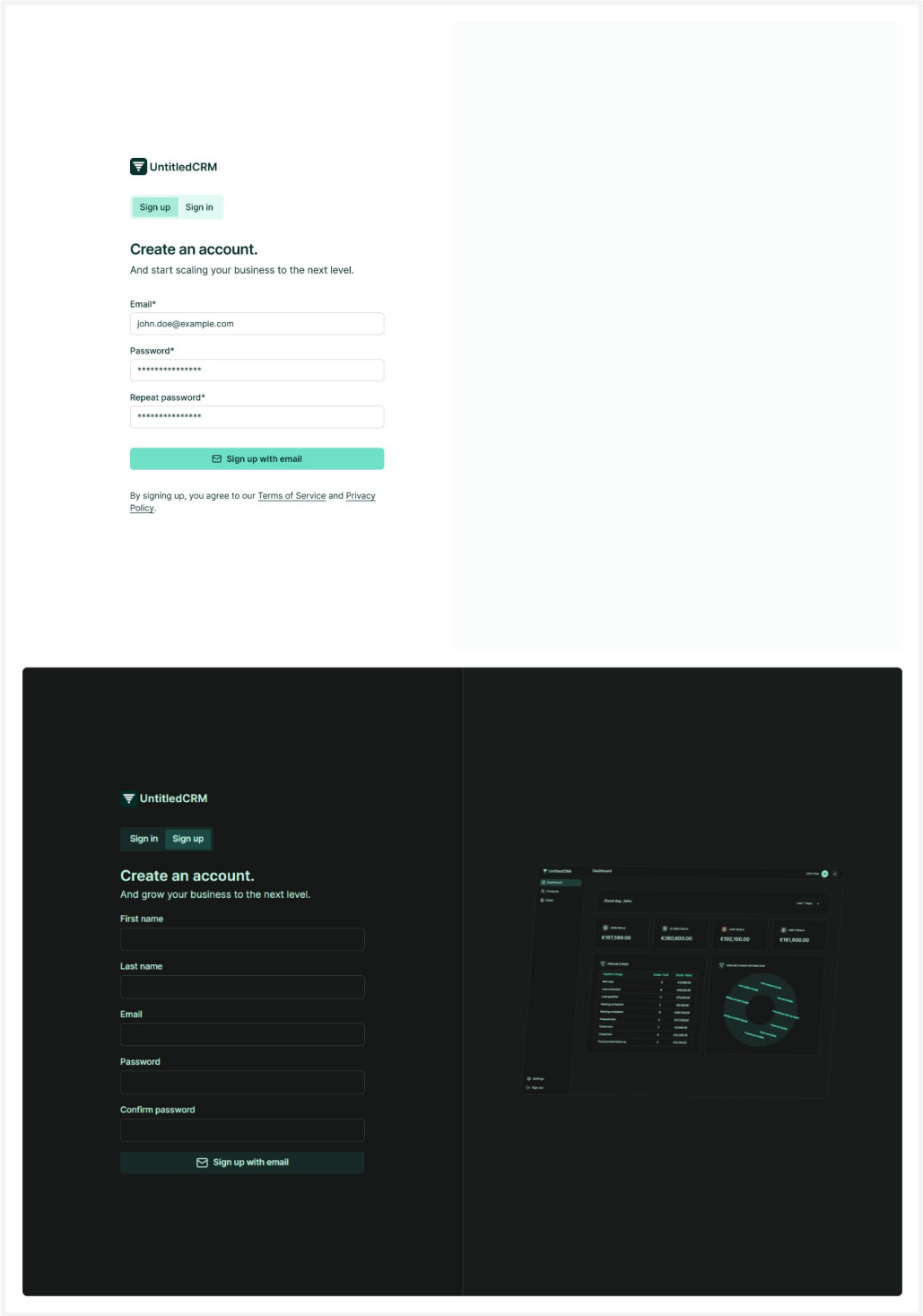
Logo (Slika 6) je inspiriran stadijima prodajnog lijevka gdje prvi stadij predstavlja stvaranje ponude za potencijalnog klijenta koji je tek dodan u kontakte (engl. *new lead*) i sadrži najviše ponuda pa je stoga i najširi te postepeno svaki slijedeći stadij sadrži sve manje i manje ponuda.

3.6. Usporedba prototipa i konačne web aplikacije

Usporedbom prototipa i konačne web aplikacije možemo uočiti odstupanja dizajna konačne web aplikacije te prototipa. Do odstupanja može doći zbog tehničkih ograničenja ili promjene u zahtjevima tijekom implementacije.

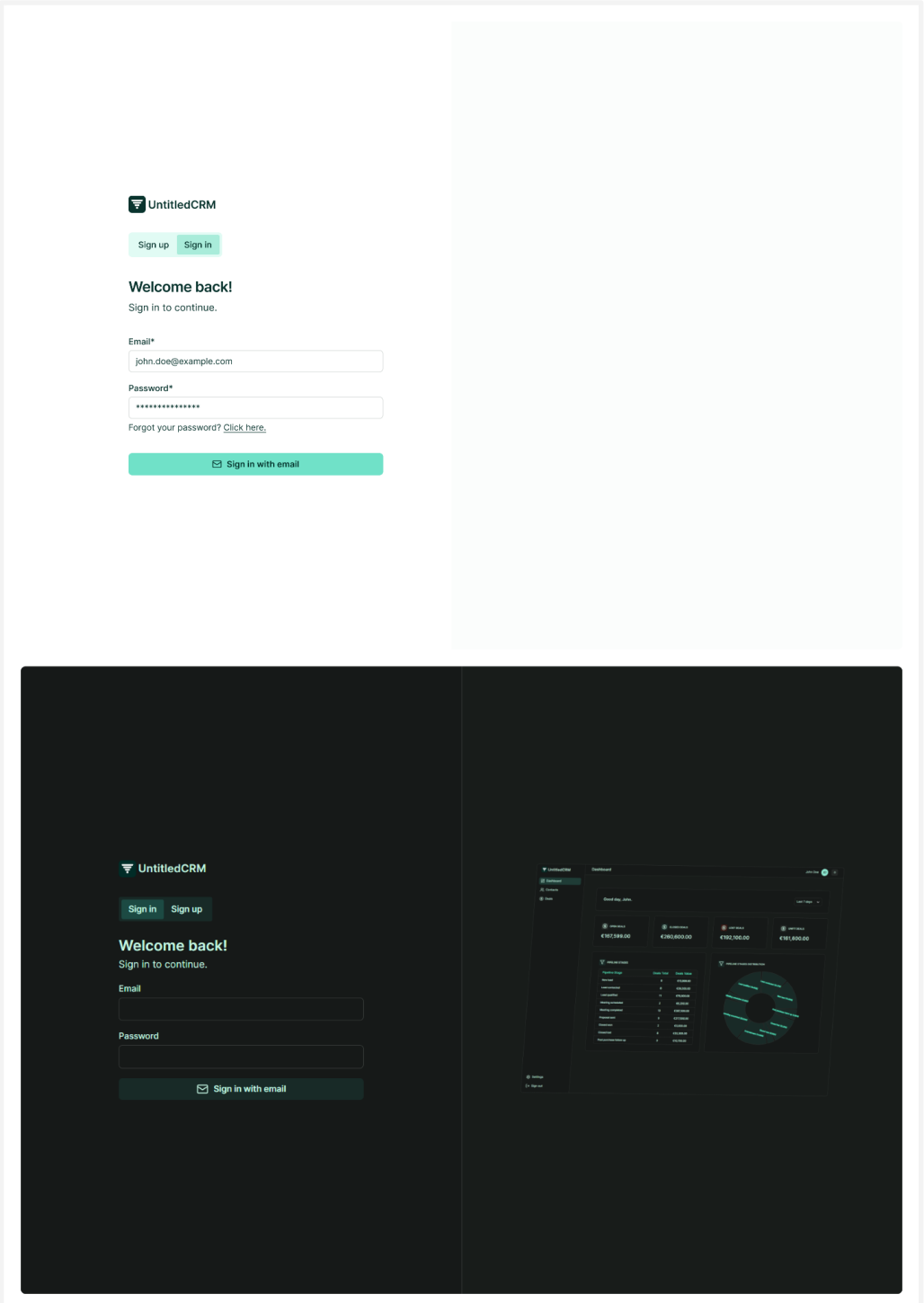
3.6.1. Prikazi i objašnjenje ekrana

U nastavku se nalazi niz slika koje prikazuju dizajn korisničkog sučelja napravljenog u alatu Figma u svijetlom načinu rada i dizajn korisničkog sučelja završne verzije aplikacije UntitledCRM koja prikazuje verziju u tamnom načinu rada.



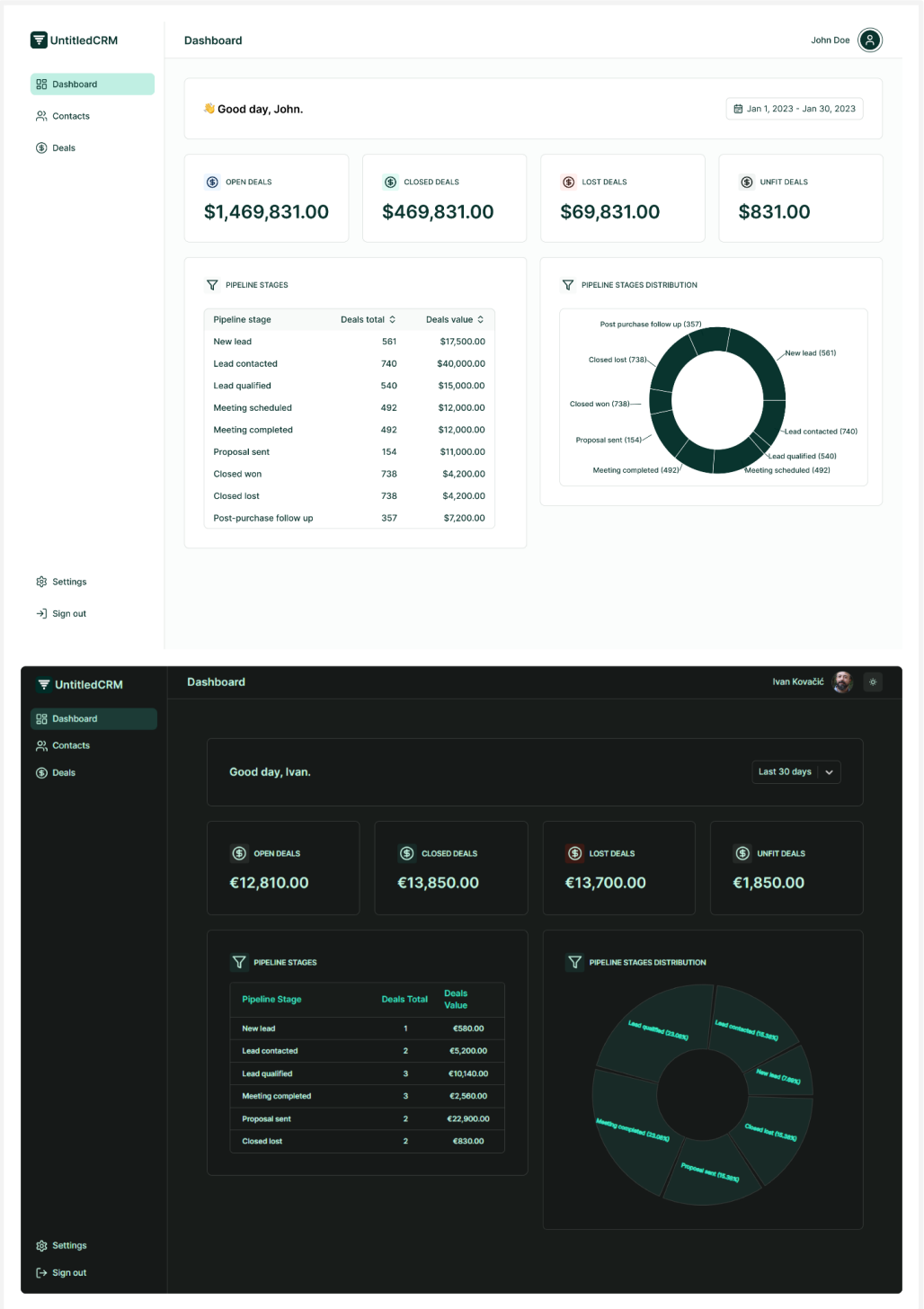
Slika 7: *Ekran "Auth" - obrazac za registraciju*

Korisnik prvo mora napraviti korisnički račun ispunjavanjem obrasca za registraciju. Za uspješno ispunjavanje obrasca korisnik mora unijeti ispravnu adresu e-pošte i lozinke se moraju podudarati te sadržavati najmanje osam znakova (Slika 7).



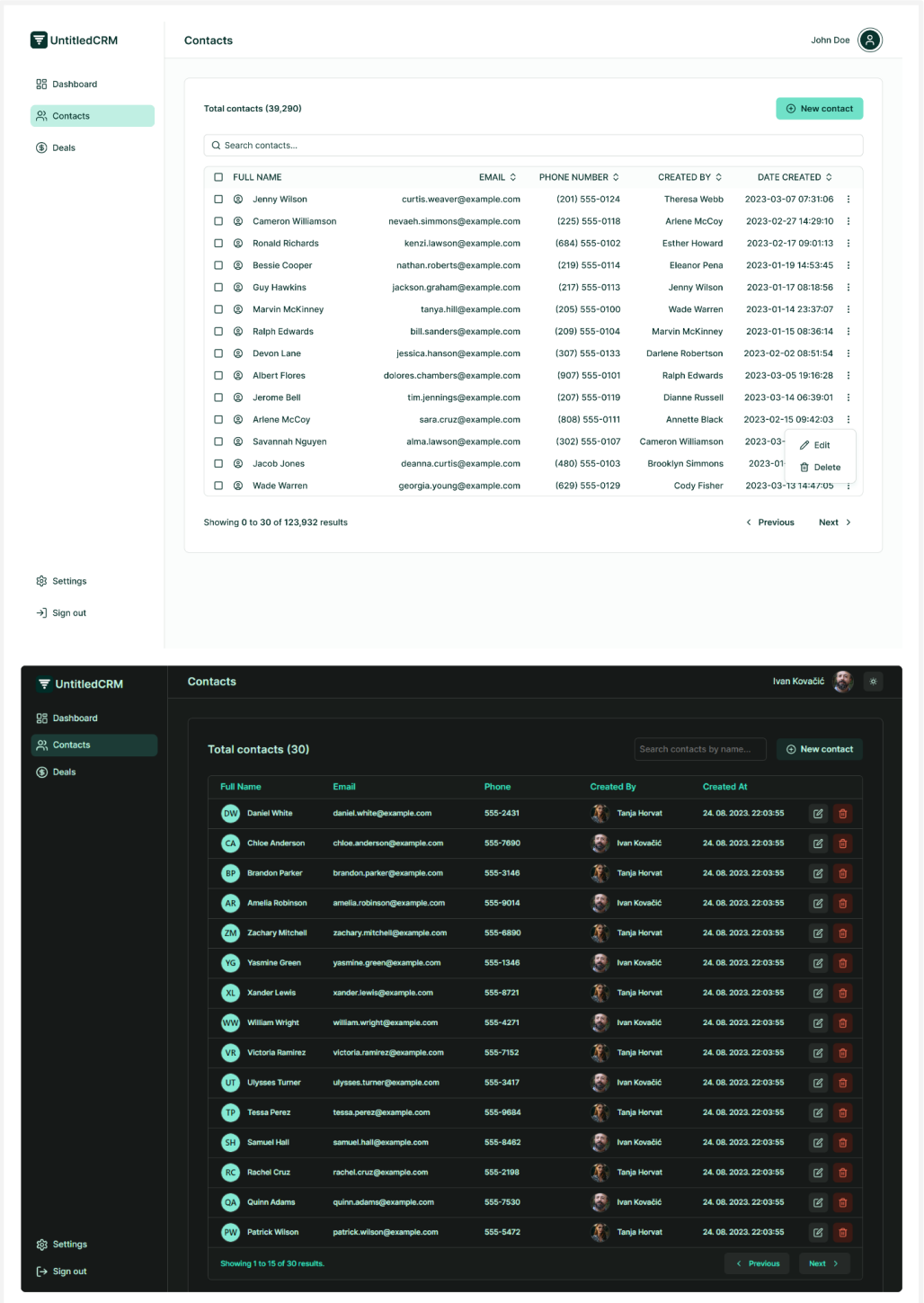
Slika 8: *Ekran "Auth" - obrazac za prijavu*

Nakon što je korisnik napravio korisnički račun, može se prijaviti u aplikaciju (Slika 8) te će nakon uspješne autentifikacije biti preusmjeren na ekran "Dashboard".



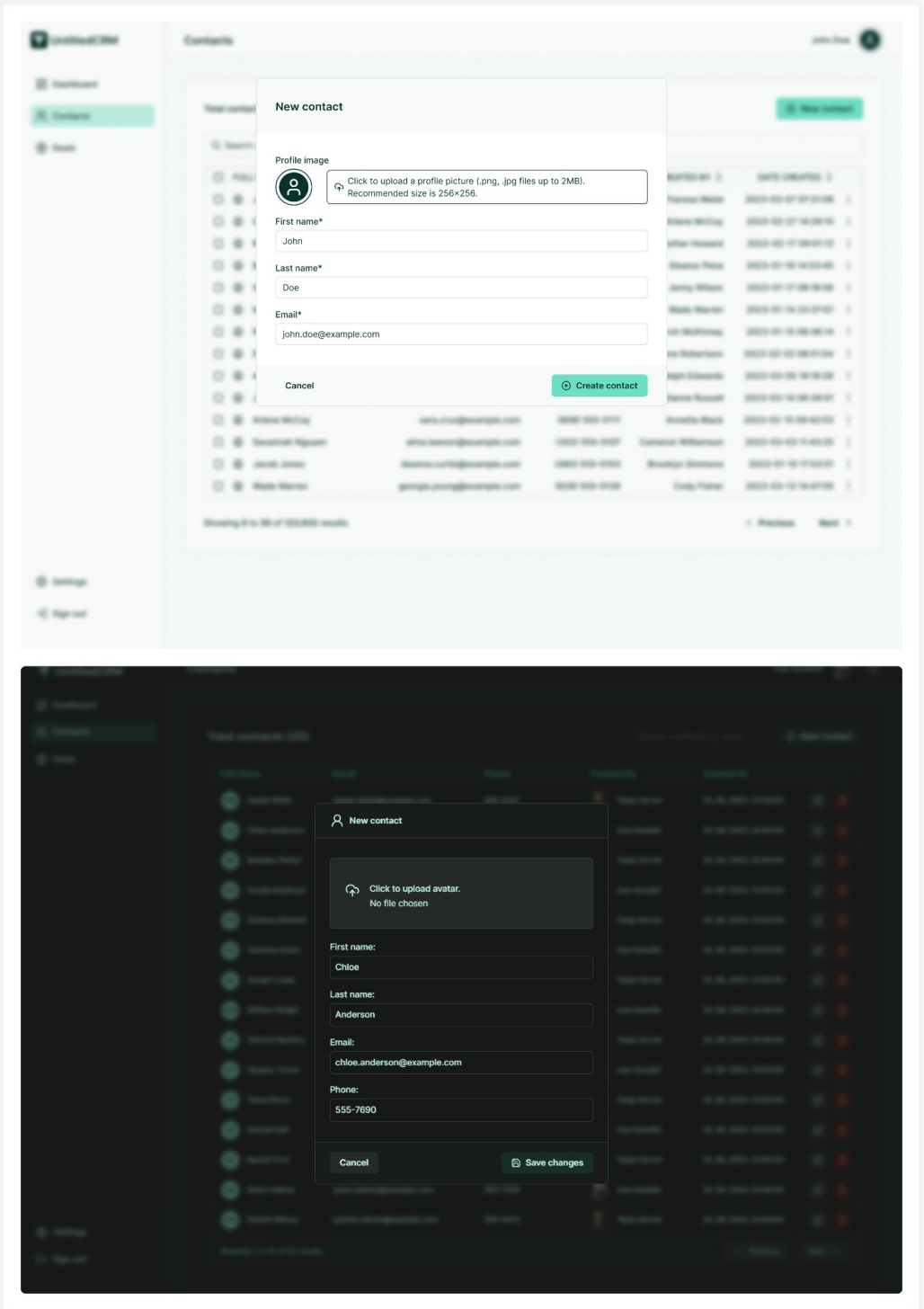
Slika 9: *Ekran "Dashboard"*

Na ekranu "Dashboard" korisnik može odabrati želi li vidjeti statistiku za prethodni dan ili zadnjih 7, 30, 60 ili 90 dana (Slika 9).



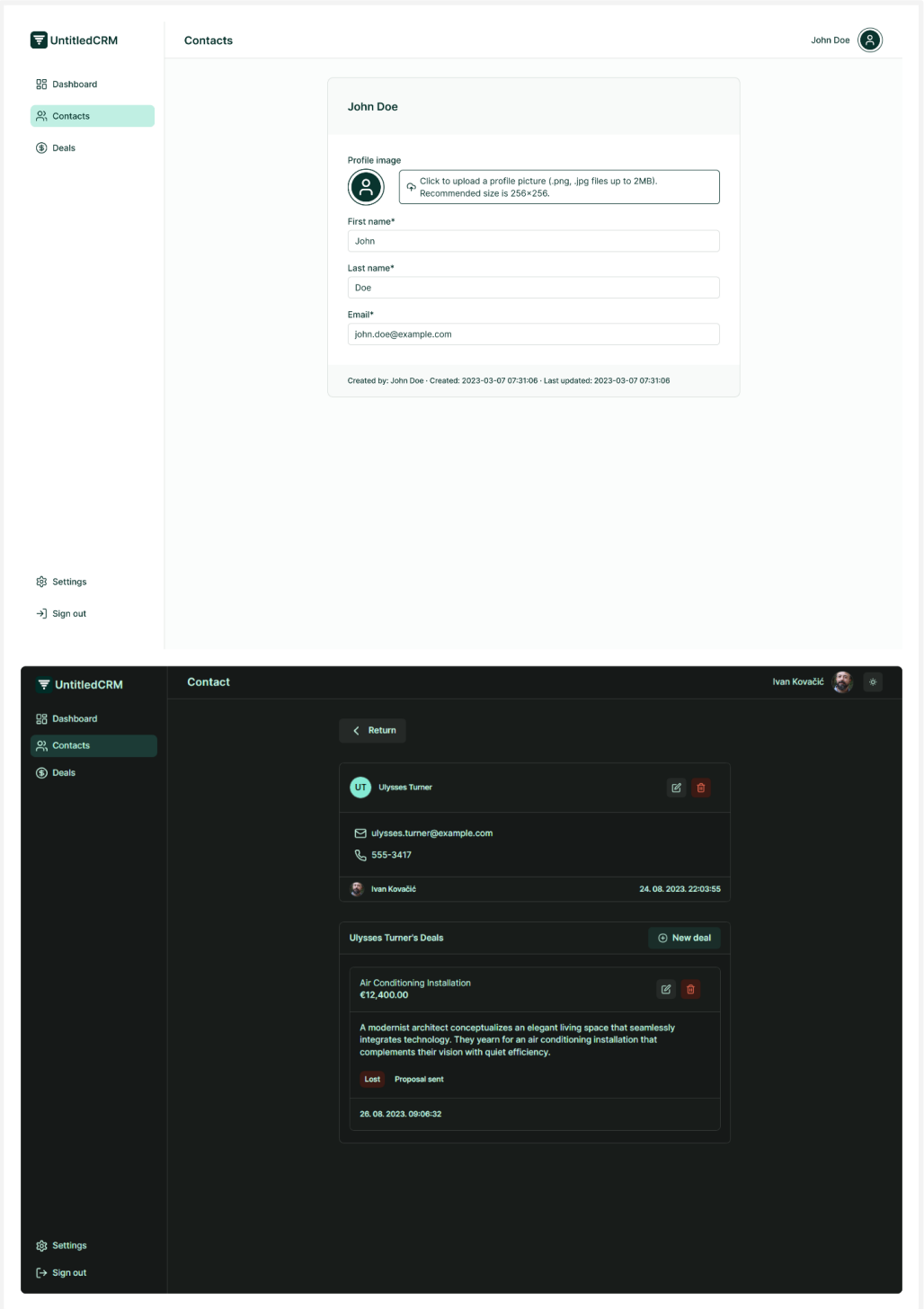
Slika 10: *Ekran "Contacts"*

Na ekranu "Contacts" korisnik može dodati novi, ažurirati ili obrisati postojeći kontakt (Slika 10). Uz to, može pretraživati kontakte s obzirom na ime ili prezime te klikom na puno ime i prezime kontakta može otvoriti ekran "Contact" gdje može vidjeti više detalja o kontaktu.



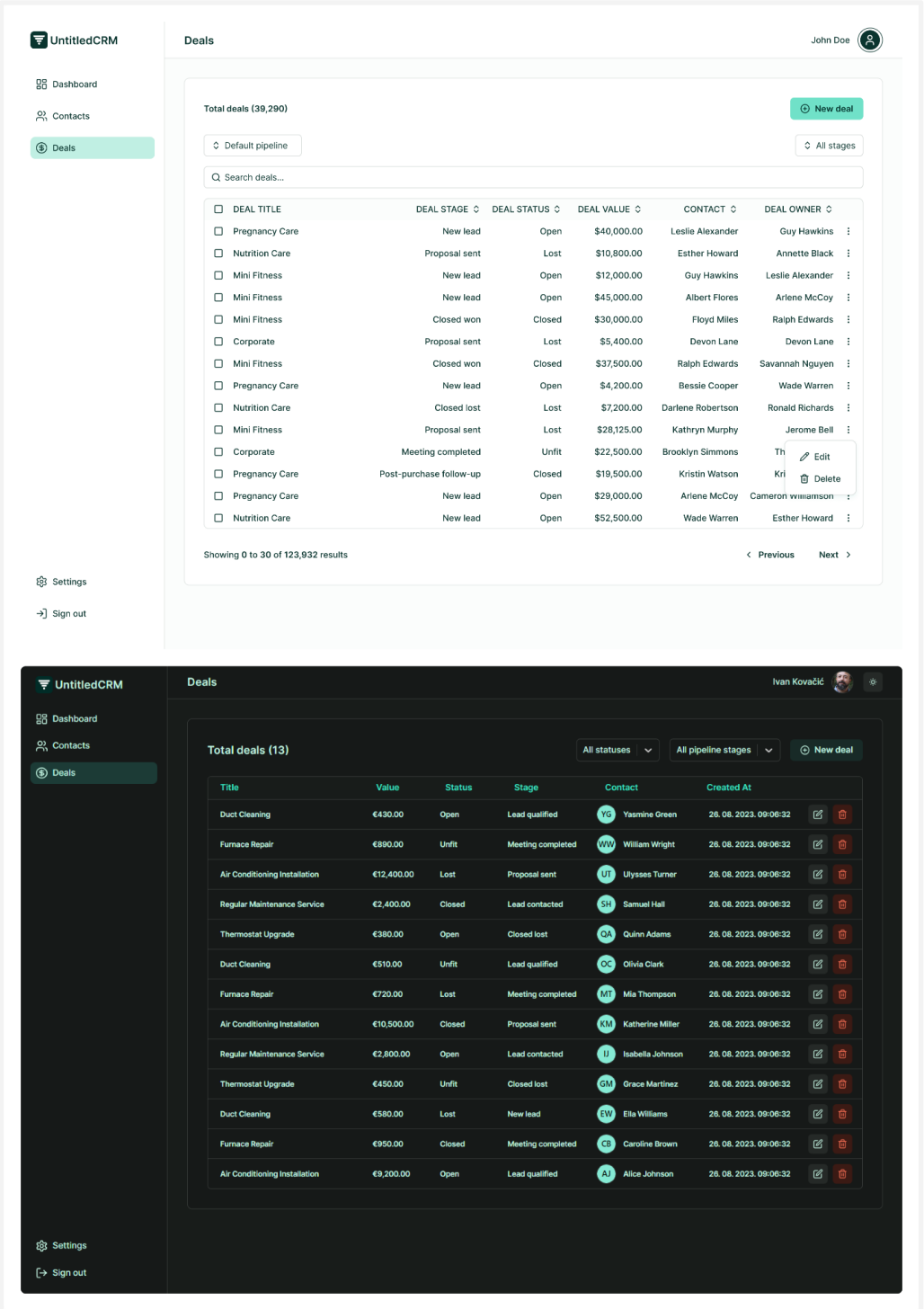
Slika 11: *Ekran "Contacts" - obrazac za stvaranje i ažuriranje kontakta*

Na prethodnoj slici (Slika 11), prikazan je obrazac za stvaranje novog ili ažuriranje postojećeg kontakta. Ime i prezime kontakta te adresa e-pošte su obavezna polja za unos dok je avatar kontakta opcionalno polje.



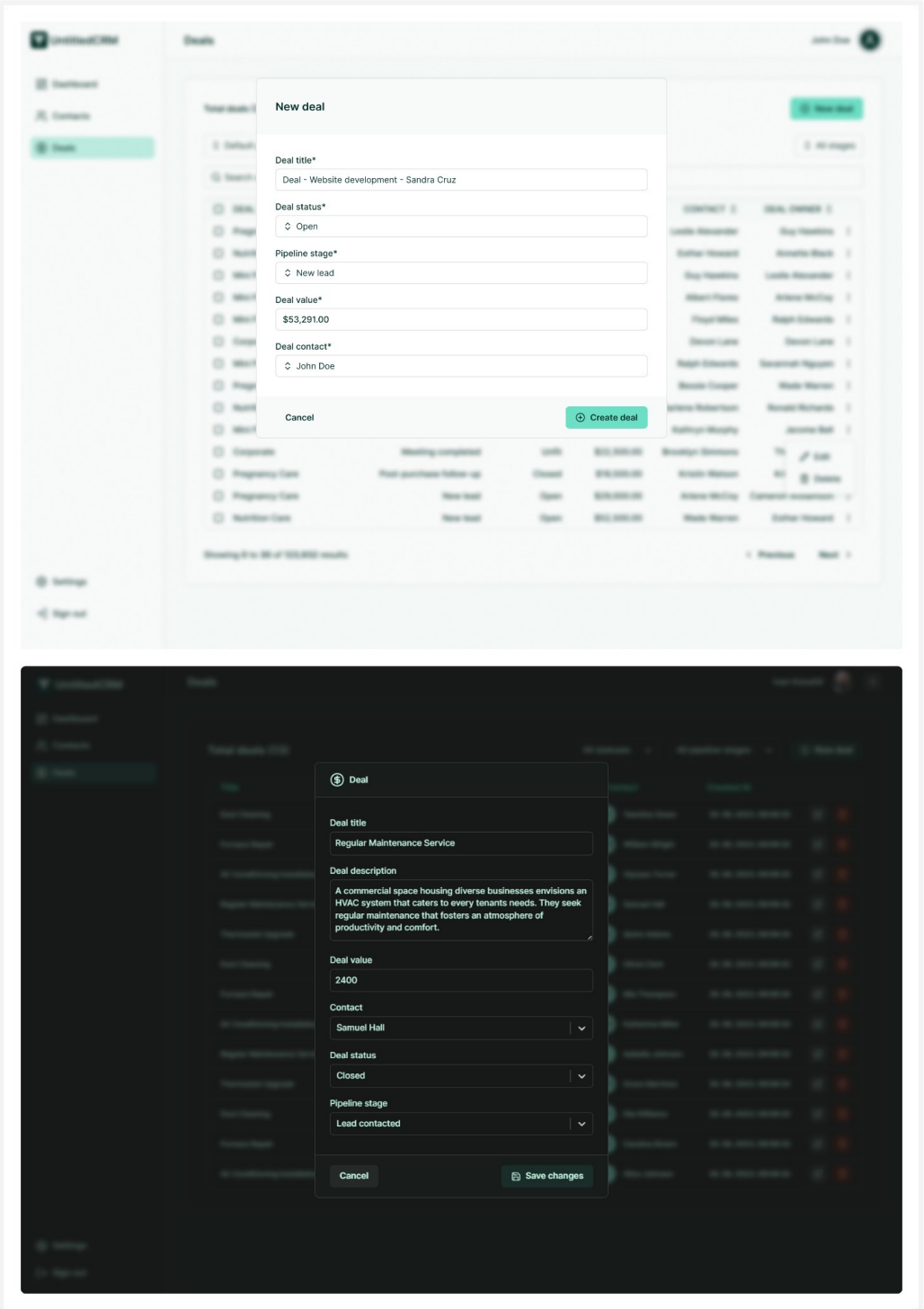
Slika 12: *Ekran "Contact"*

Na ekranu "Contact" prikazane su sve informacije povezane sa odabranim kontaktom te je moguće ažurirati ili obrisati informacije o kontaktu ili ponudi te stvoriti novu ponudu (Slika 12).



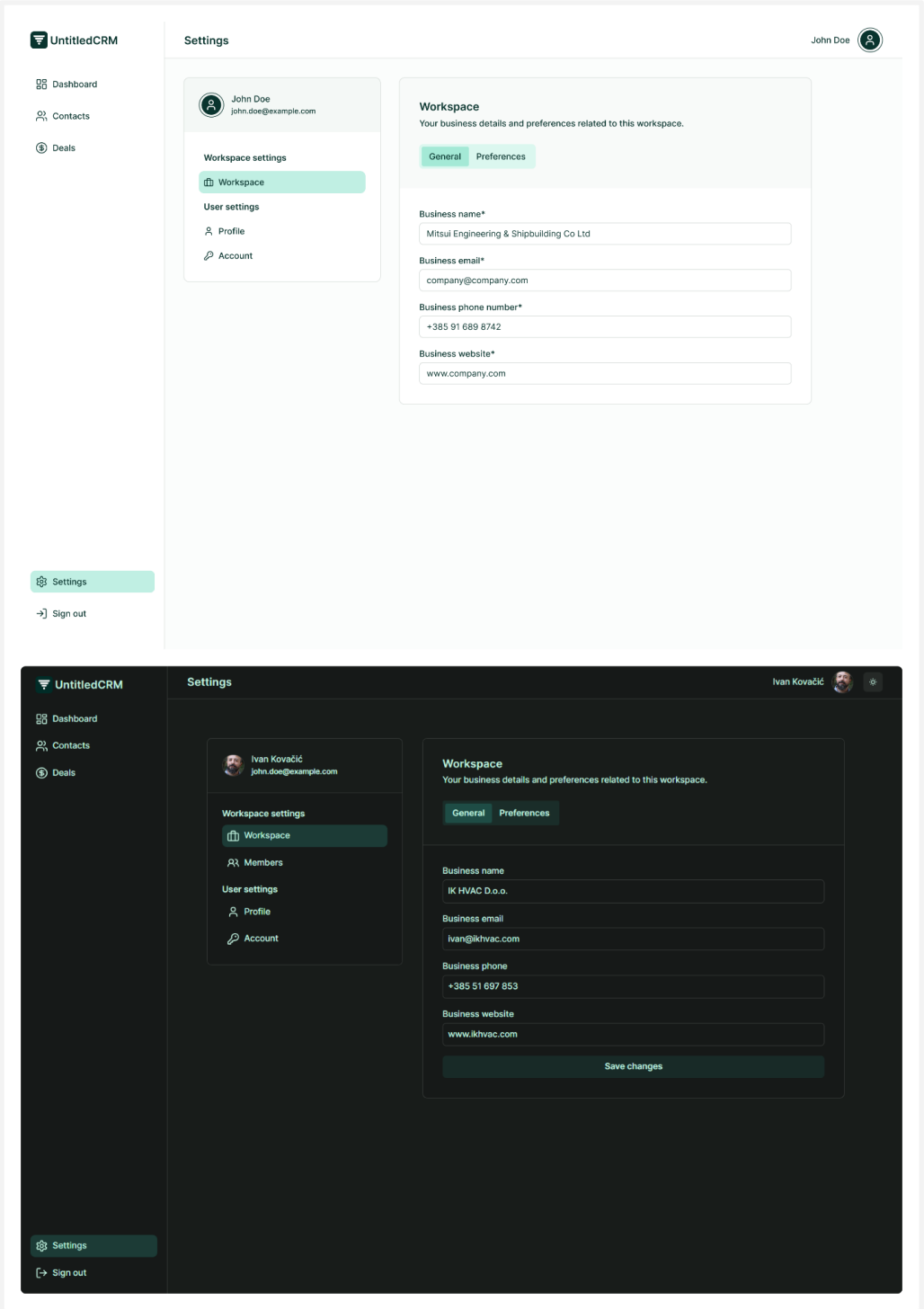
Slika 13: *Ekran "Deals"*

Na ekranu "Deals" korisnik može dodati novu, ažurirati ili obrisati postojeću ponudu. Može filtrirati ponude s obzirom na status te stadij ponude u prodajnom lijevku (Slika 13). Klikom na naziv ponude može otvoriti ekran "Contact" gdje može vidjeti više detalja o ponudi te povezanom kontaktu.



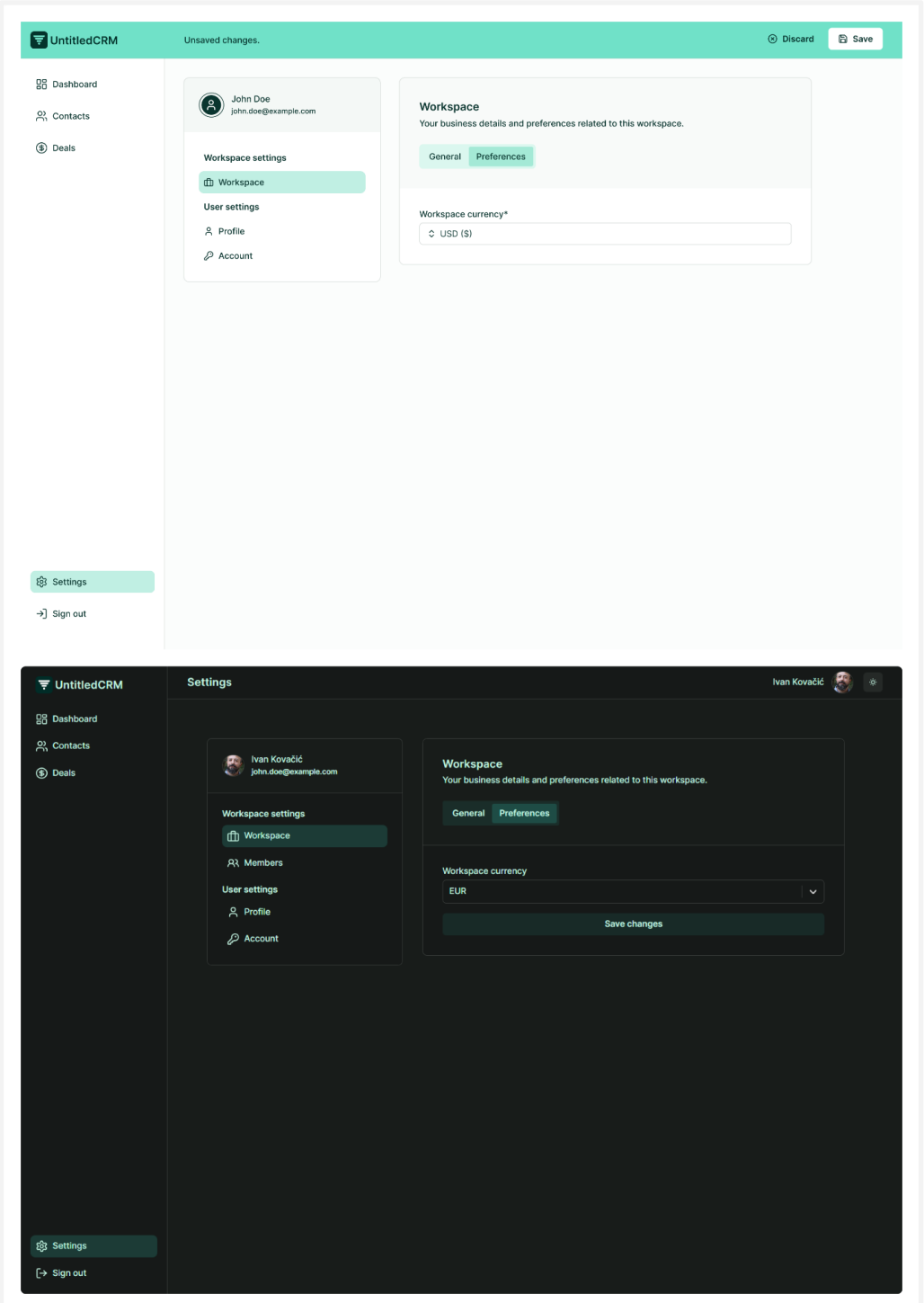
Slika 14: *Ekran "Deals" - obrazac za stvaranje i ažuriranje ponuda*

Na prethodnoj slici, prikazan je obrazac za stvaranje nove ili ažuriranje postojeće ponude. Naziv ponude, vrijednost ponude, povezani kontakt, status te stadij u prodajnom lijevku su obavezna polja dok je opis ponude opcionalno polje. U opis ponude možemo zapisati aktivnosti i detalje vezane za ponudu (Slika 14).



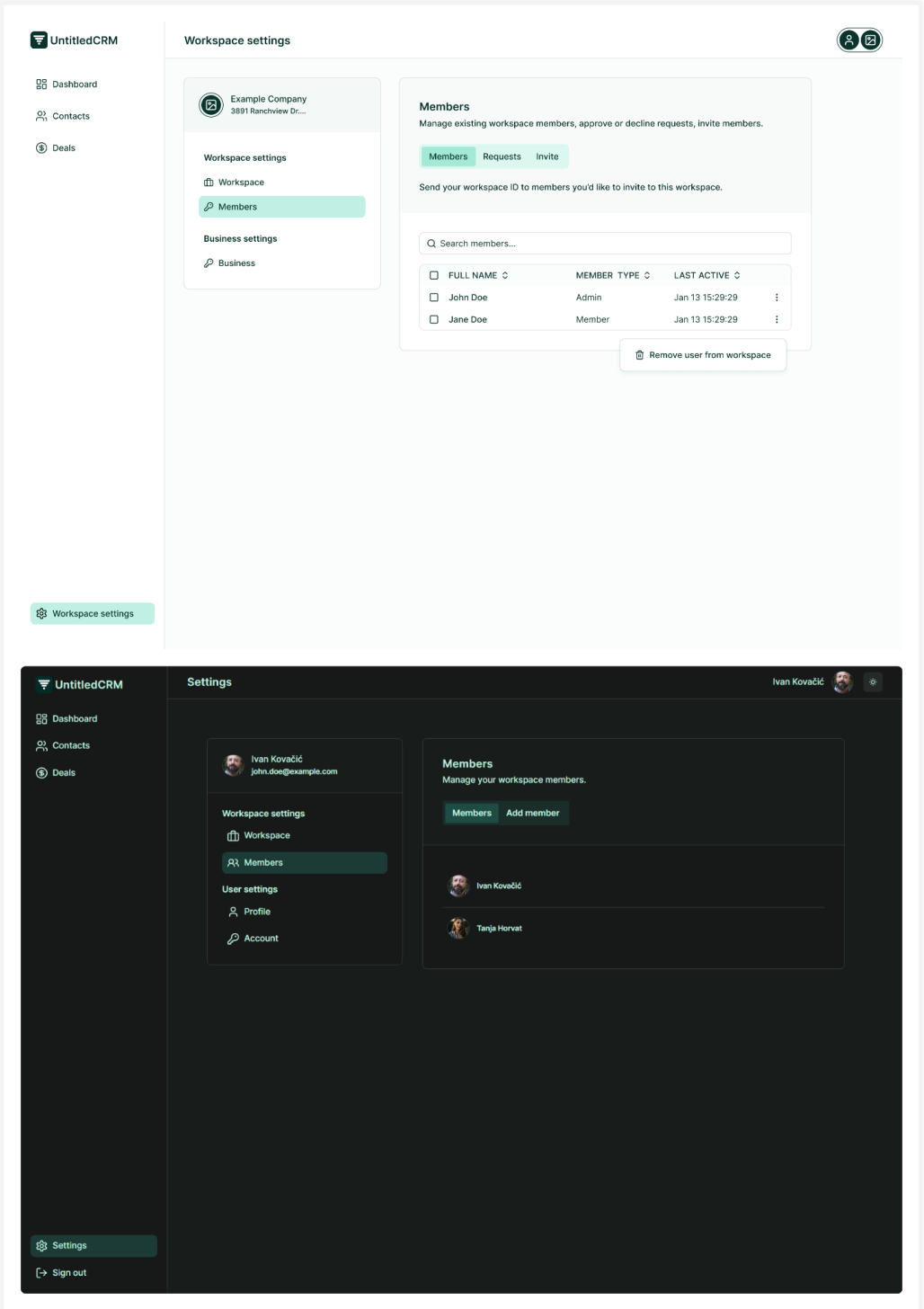
Slika 15: *Ekran "Settings" - obrazac za ažuriranje poslovnih informacija*

Klikom na gumb "Settings" korisnik dolazi na istoimeni ekran gdje može ažurirati podatke o tvrtki kojoj pripada radni prostor. Polja koja korisnik mora ispuniti uključuju naziv, adresu e-pošte, adresu i web sjedište tvrtke (Slika 15).



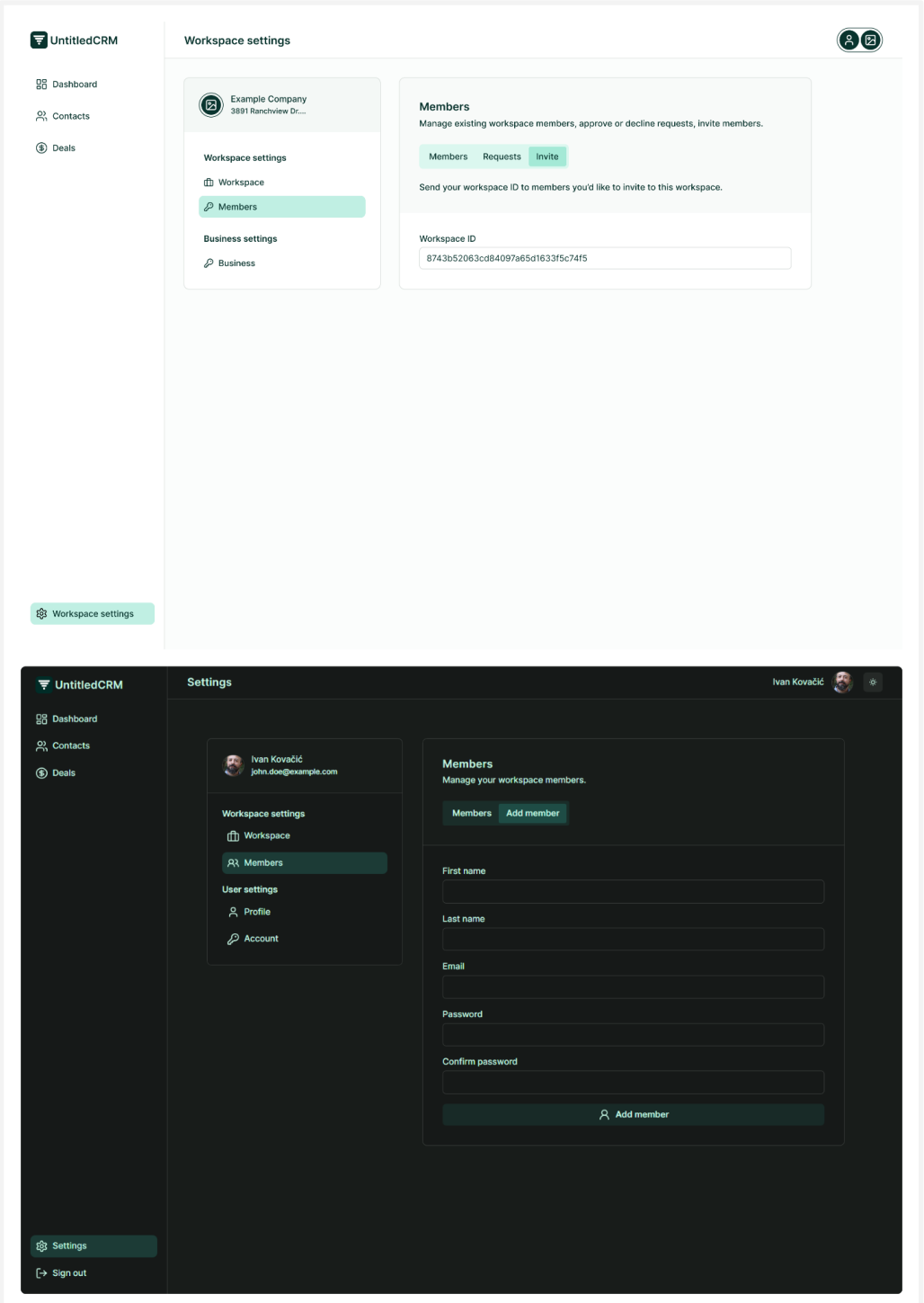
Slika 16: *Ekran "Settings" - obrazac za ažuriranje preferencija*

Klikom na gumb "Preferences" korisnik može postaviti preferencija poput valute koja će se koristiti za prikaz vrijednosti ponuda unutar radnog prostora (Slika 16).



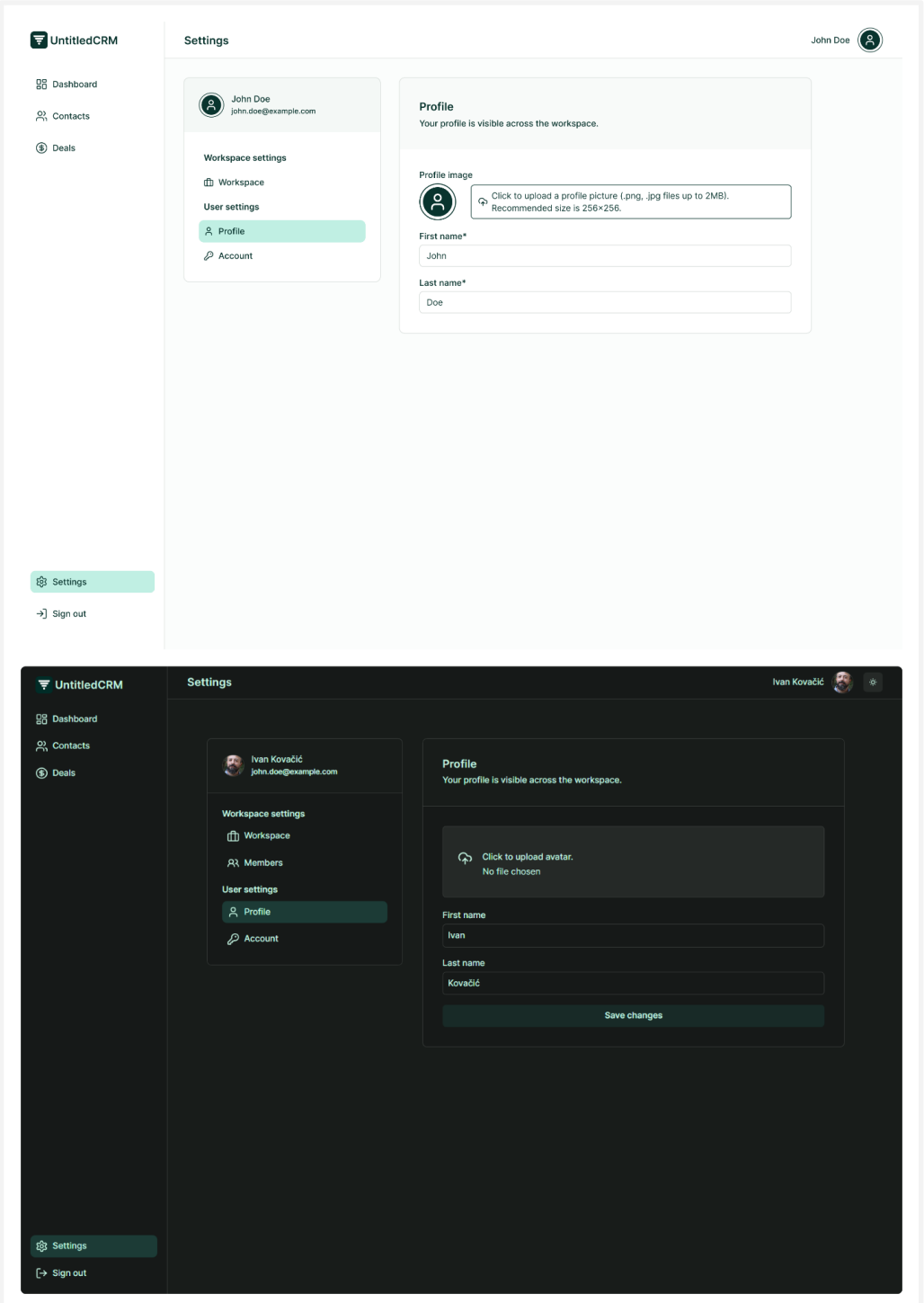
Slika 17: *Ekran "Members" - popis članova radnog prostora*

Na ekranu "Members" korisnik može pregledati postojeće članove radnog prostora te klikom na gumb "Add member" može otvoriti obrazac za dodavanje novog korisnika (Slika 17).



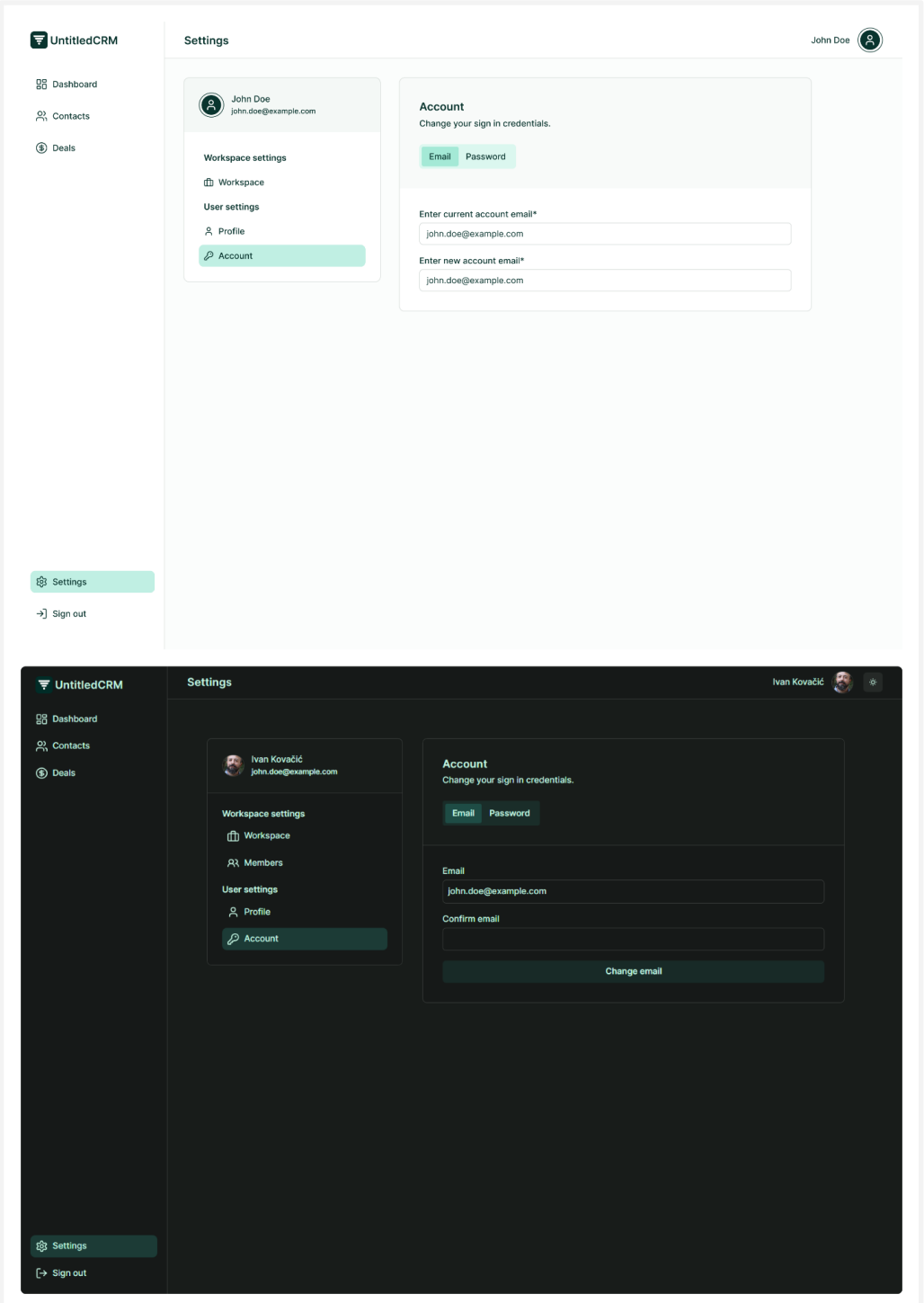
Slika 18: *Ekran "Members" - dodavanje novog člana u radni prostor*

Za dodavanje novog korisnika obavezno je ispuniti tražena polja koja uključuju ime i prezime, adresu e-poštu te lozinku člana koji se dodaje u radni prostor (Slika 18).



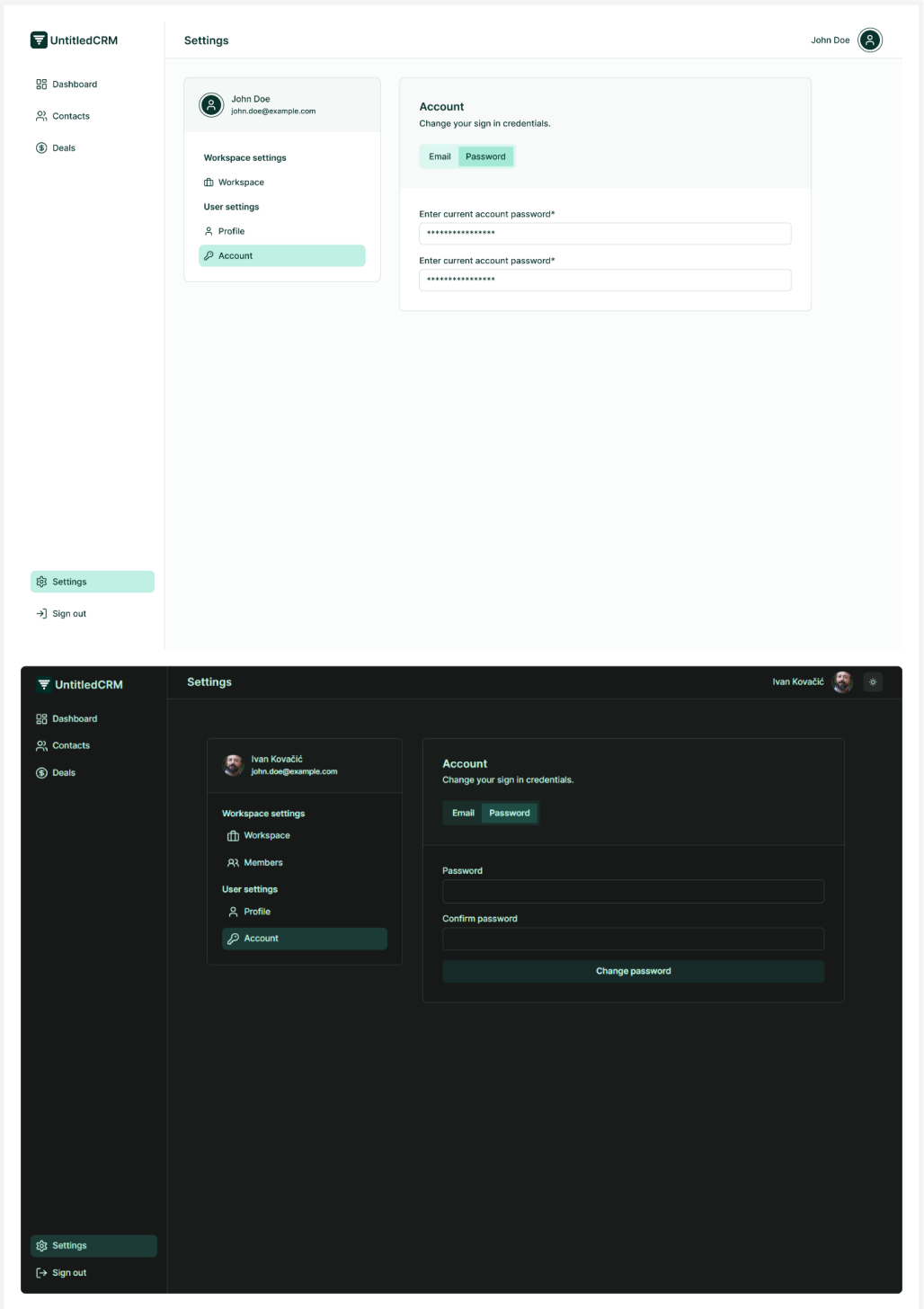
Slika 19: *Ekran "Profile" - obrazac za ažuriranje korisničkog profila*

Posjetom ekrana "Profile" korisnik može ažurirati korisnički profil odnosno promijeniti ime, prezime te avatar (Slika 19). Ako korisnik ne priloži avatar prikazati će se alternativni avatar s inicijalima njegovog imena i prezimena.



Slika 20: *Ekran "Account" - obrazac za ažuriranje adrese e-pošte korisničkog računa*

Navigiranjem na ekran "Account", korisnik može promijeniti e-poštu povezanu s korisničkim računom (Slika 20). Nakon promjene adrese e-pošte korisniku će biti poslana e-pošta gdje mora potvrditi promjenu, u suprotnom neće doći do promjene.



Slika 21: *Ekran "Account" - obrazac za ažuriranje lozinke korisničkog računa*

Klikom na gumb "Password" korisniku se pojavljuje obrazac gdje može promijeniti lozinku povezanu s korisničkim računom (Slika 21). Nova lozinka postaje odmah aktivna odnosno nije potrebna potvrditi promjenu lozinke korištenjem e-pošte.

4. DIZAJN I IMPLEMENTACIJA BAZE PODATAKA

Baza podataka je relacijskog tipa te je dizajnirana na temelju prototipa UntitledCRM aplikacije, a implementirana je korištenjem platforme Supabase.

4.1. O korištenim tehnologijama

Supabase je BaaS (engl. "*Backend as a Service"*) platforma otvorenog koda, smještena u oblaku koja programerima pruža niz alata za stvaranje i upravljanje poslužiteljskim dijelom aplikacije. Budući da se nalazi u oblaku, ne zahtjeva instalaciju kako bi se mogla koristiti već je potrebno napraviti korisnički račun prije stvaranja novog projekta (Adservio, 2022).

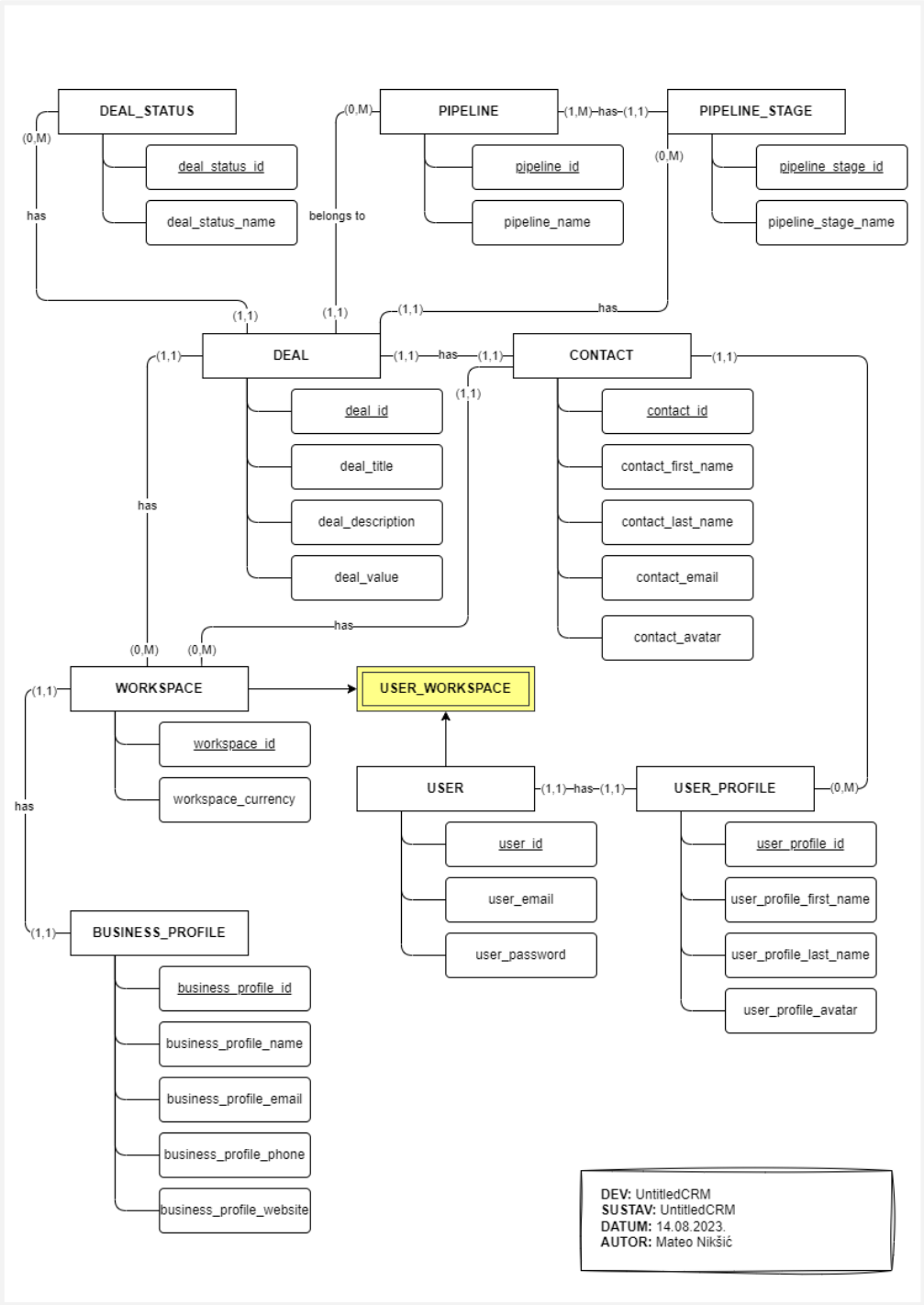
Prednosti razvoja aplikacije koristeći Supabase uključuju intuitivno sučelje koje štedi vrijeme pri razvoju aplikacije, omogućuje pristup izvornom kodu tako da je moguće prilagoditi kod specifičnim potrebama aplikacije, korisna dokumentacija te rastuća zajednica koja je spremna ponuditi odgovor i podršku u pronalaženju rješenja za bilo koji problem (Adservio, 2022).

Neke od usluga koje pruža, a koje su korištene tijekom izrade aplikacije su autentifikacija korisnika, spremište za datoteke (avatare korisnika u formatu .png ili .jpg) te izgradnja relacijske baze podataka koristeći PostgreSQL. Na temelju strukture baze podataka automatski stvara REST API koji omogućuje interakciju s bazom podataka u JSON formatu koristeći HTTP i HTTPS protokole.

4.2. EV dijagram baze podataka

Model baze podataka izgrađen je korištenjem metodologije MIRIS te metode entiteti - veze (skraćeno metoda EV).

Metodologija MIRIS ili "Metodologija za Razvoj Informacijskih Sustava" je skup metoda sa ciljem projektiranja i izgradnje informacijskog sustava koju je razvio prof. dr. sc. Mile Pavlić. Koriste se tri osnovne metode: metoda za modeliranje aplikacija, procesa te podataka (Pavlić, Informacijski sustavi, 2011.).



Slika 22: *EV dijagram*

Metoda EV je grafički prikaz međusobno povezanih grupa podataka promatranog sustava, a osnovni koncepti su: tip entiteta, slabi tip entiteta, tip veze, atribut, agregracija, povratna veza, generalizacija (Pavlić, Oblikovanje baza podataka, 2011.). Na slici 22 je prikazan EV dijagram baze podataka "UntitledCRM" sustava.

4.3. Relacijski model baze podataka

U nastavku slijedi relacijski model baze podataka izveden iz EV dijagrama:

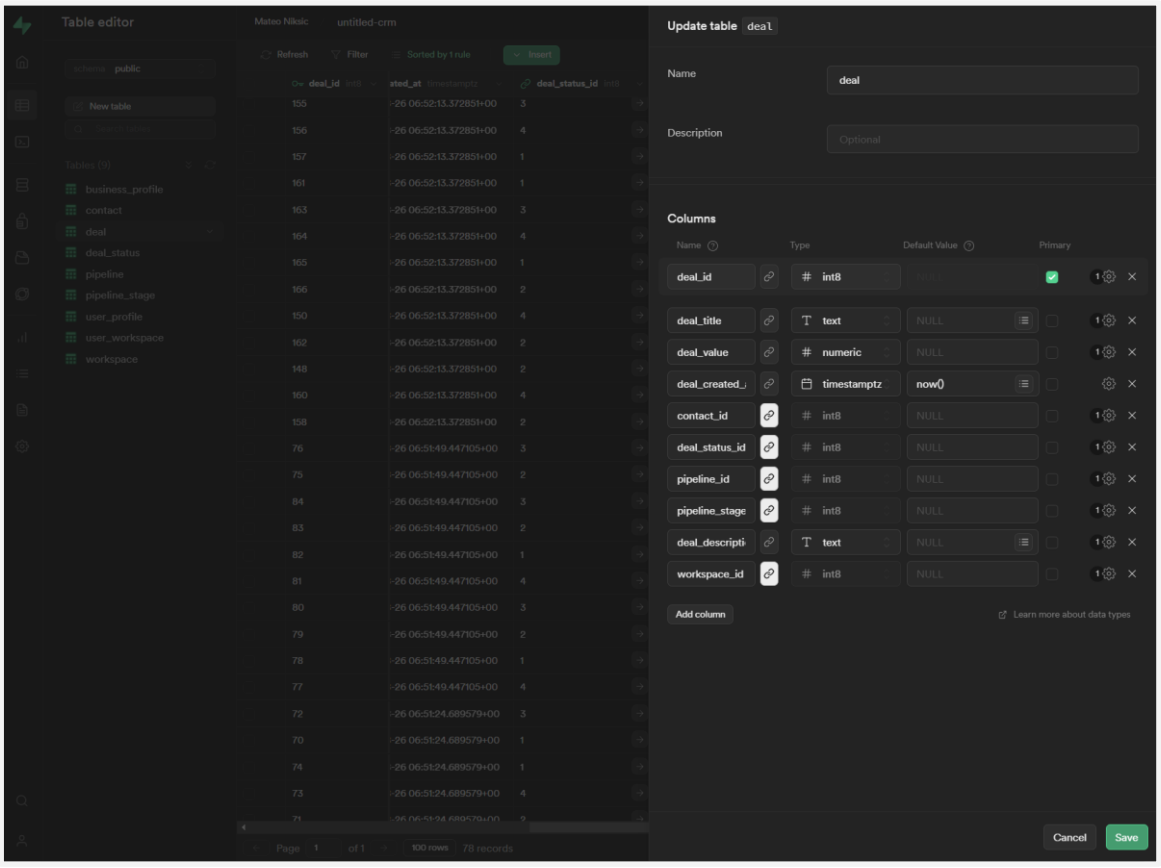
- USER (**user_id**, user_email, user_password)
- USER_PROFILE (**user_profile_id**, user_profile_first_name, user_profile_last_name, user_profile_avatar, *user_id*)
- WORKSPACE (**workspace_id**, workspace_currency)
- USER_WORKSPACE (**user_id, workspace_id**)
- BUSINESS_PROFILE (**business_profile_id**, business_profile_name, business_profile_email, business_profile_phone, business_profile_website, *workspace_id*)
- CONTACT (**contact_id**, contact_first_name, contact_last_name, contact_phone, contact_email, contact_avatar, *user_profile_id*, *workspace_id*)
- DEAL (**deal_id**, deal_title, deal_description, deal_value, *deal_status_id*, *contact_id, pipeline_id*, *pipeline_stage_id*, *workspace_id*)
- DEAL_STATUS (**deal_status_id**, deal_status_name)
- PIPELINE (**pipeline_id**, pipeline_name)
- PIPELINE_STAGE (**pipeline_stage_id**, pipeline_stage_name, *pipeline_id*)

Važno je napomenuti da se tablica "USER" nalazi u privatnoj shemi uz dodatne atribute dodane od strane Supabase platforme te su operacije i pristup REST API-ju ograničeni u odnosu na tablice koje se nalaze u javnoj shemi.

4.4. Implementacija baze podataka

Tijekom implementacije baze podataka napravljen je novi korisnički računa na Supabase platformi te novi projekt naziva "untitled-crm". Za stvaranje tablica korišteno je grafičko sučelje, a prvo se stvaraju jednostavne tablice koje ne sadrže složene i strane ključeve te zatim složene tablice sa složenim i stranim ključevima.

Za stvaranje testnih podataka korišten je "SQL Editor" alat koji je dio Supabase platforme te omogućava izvršavanje SQL upita.

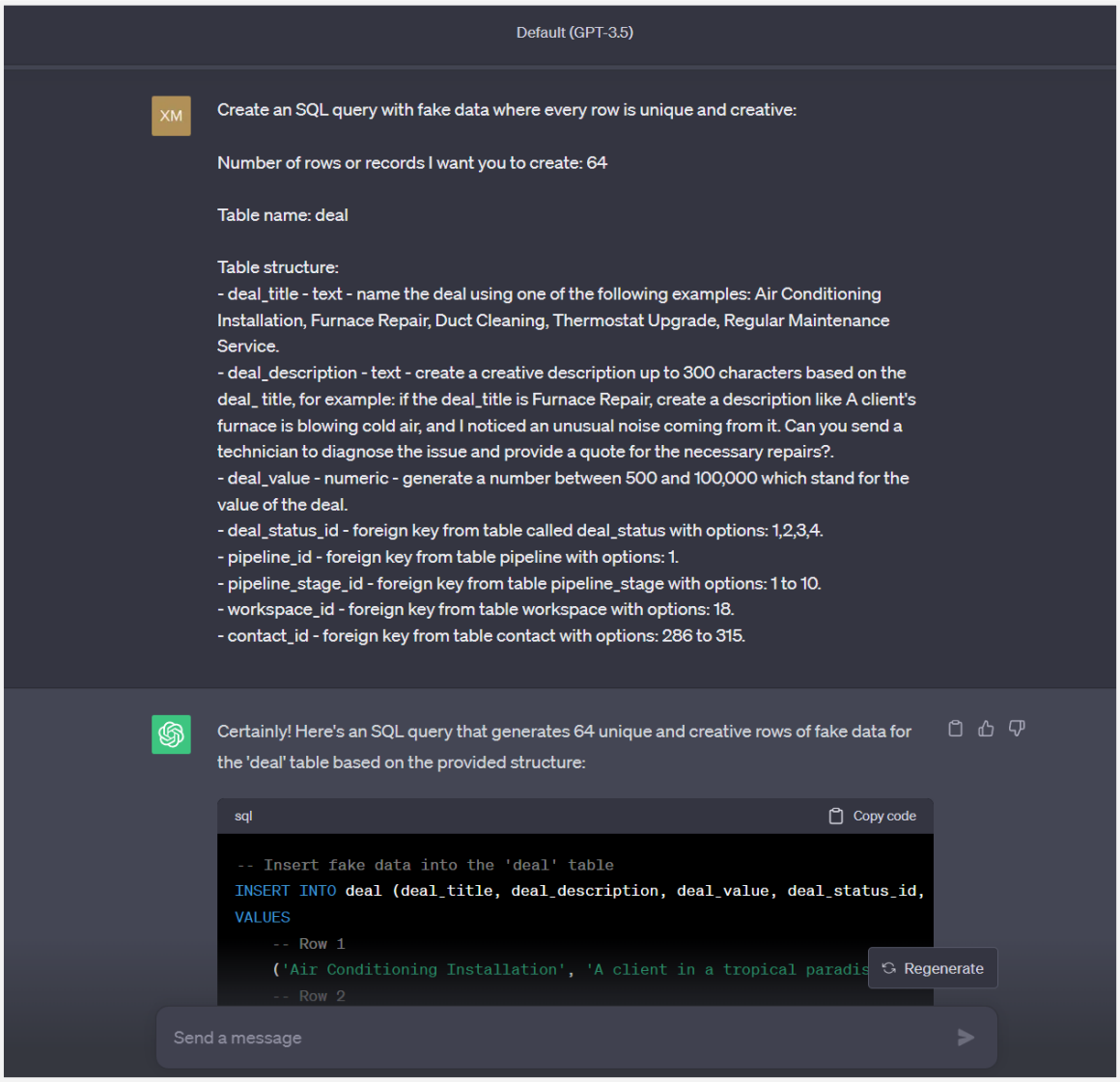


Slika 23: *Grafičko sučelje za stvaranje tablica*

Na prethodnoj slici (Slika 23) se nalazi prikaz grafičkog sučelja za stvaranje tablica na platformi Supabase.

Testni podatci su stvoreni korištenjem ChatGPT umjetne inteligencije. ChatGPT je chatbot umjetne inteligencije (AI) koji koristi obradu prirodnog jezika za stvaranje ljudskog razgovora. Jezični model može odgovoriti na pitanja i sastaviti različite pisane sadržaje, uključujući članke, objave na društvenim mrežama, eseje, kod i e-poštu (Hetler, 2023).

Za uspješno stvaranje testnih podataka ChatGPT mora prvo stvoriti SQL upit gdje unosi testne podatke za jednostavne tablice te se prije stvaranja složenih tablica napravi popis mogućih ključeva koje ChatGPT može koristiti za stvaranje složenih tablica.



Slika 24: *Prikaz ChatGPT naredbe za stvaranje testnih podataka*

Na primjer (Slika 24), napišemo stupce koje tablica “DEAL” ili neka druga tablica koristi i vrijednosti ili ograničenja koje ti stupci imaju, a za stupce koji su strani ključevi navedemo opcije koje ChatGPT može koristiti. U nastavku se nalazi prikaz naredbe za stvaranje testnih podataka.

5. RAZVOJ KLIJENTSKOG DIJELA APLIKACIJE

Za razvoj klijentskog dijela aplikacije korištene su moderne tehnologije za razvoj dinamičkih web aplikacija poput knjižnice React te popratnih paketa ili knjižnica temeljenih na Javascript skriptnom jeziku te HTML jezika za strukturu i organizaciju tekstualnog sadržaja i CSS jezika za stilizaciju HTML dokumenata.

5.1. O korištenim tehnologijama

React je Javascript knjižnica koju je razvio Facebook u svibnju 2023. godine, a sada je jedna od najčešće korištenih knjižnica za razvoj web aplikacija. Omogućava jednostavan razvoj dinamičkih aplikacija korištenjem virtualnog DOM-a (engl. *virtual DOM*) koji poboljšava performanse zato jer ažurira samo promijenjeni dio dokumenta u odnosu na tradicionalne web aplikacije koje koriste Javascript. U Reactu se izrađuju JSX komponente odnosno građevne jedinice od kojih se gradi aplikacija te komponenta može sadržavati više drugih komponenti, a prijenos podataka se odvija u jednom smjeru od vrha prema dnu odnosno od roditeljske komponente prema djeci. Sintaksa JSX omogućava pisanje HTML jezika unutar Javascript koda što olakšava razvoj korisničkog sučelja. Komponente React knjižnice podatke mogu prihvaćati i prosljeđivati putem varijable "props" koja predstavlja svojsta (engl. *properties*) (Simplilearn, 2023).

"React hooks" su komponente koje započinju svoj naziv s "use", a dio su React knjižnice te omogućavaju komponentama da koriste stanja, efekte, kontekst, reference i ostale React funkcionalnosti koje omogućuju lakše dijeljenje stanja i logike među komponentama, ali i poboljšavaju čitljivost koda (Adhikary, 2022).

"React context API" omogućava prosljeđivanje podataka kroz stablo komponenti bez da se podaci ručno prosljeđuju na svakoj razini što čini dijeljenje podataka između komponenti puno lakšim (Boateng, 2023).

"React router DOM" je ekstenzija popularne knjižnice "React Router" koja upravlja dinamičkim prikazom različitih komponenti odnosno ekrana i sinkronizira ih s URL adresom. Ključne komponente koje se nalaze u paketu su "BrowserRouter" koji u sebi sadrži djecu: "Routes" za preusmjeravanje, "Route" za renderiranje komponente te "Link" za navigaciju (Dev, 2020).

“React query” je intuitivni API za dohvat podataka koristeći REST API, a njegove su glavne prednosti jednostavno pisanje logike, inteligentno keširanje i dohvaćanje podataka u pozadini. Nakon dohvaćanja vraća podatke, status i pogrešku ako podaci nisu uspješno dohvaćeni (Starčević, 2023).

“React hook form” je knjižnica koja olakšava upravljanje sa složenim obrascima koji imaju puno polja te zahtijevaju validaciju (Hygraph, 2022).

“React error boundary” se koristi za hvatanje greški unutar komponenti koje obuhvaća te za prikaz ekrana za oporavljanje od greške umjesto da se aplikacija sruši i postane nekorisna (Apte, 2023).

“Styled components” je knjižnica koja omogućava pisanje CSS jezika koristeći Javascript jezik direktno unutar komponente (Makode, 2023).

“Recharts” je knjižnica korištena za vizualizaciju podataka korištenjem grafikona.

Za razvoj aplikacije korišten je Vite alat za izgradnju (engl. *build tool*) čija je svrha pružiti brže i jednostavnije razvojno iskustvo, a sastoji se od poslužitelja te naredbe za izgradnju koda koja optimizira kod i statične datoteke za produkciju.

5.2. Postavke i struktura projekta

Naziv korijenskog direktorija je “project-untitled-crm”, a sadrži niz mapa i datoteka objašnjenih u nastavku. Datoteke poput “node_modules”, “.vite”, “.eslintrc.cjs”, “prettier.config.cjs”, “vite.config.cjs” su konfiguracijske datoteke ili datotke koje sadrže vanjske knjižnice ili module te se stvaranju prilikom inicijalizacije projekta. Datoteka “dist” sadrži kod aplikacije za distribuciju, dobije se pokretanjem naredbe “npm build”, a definirana je u datoteci “package.json” koja sadrži opis, skripte i ovisnosti aplikacije.

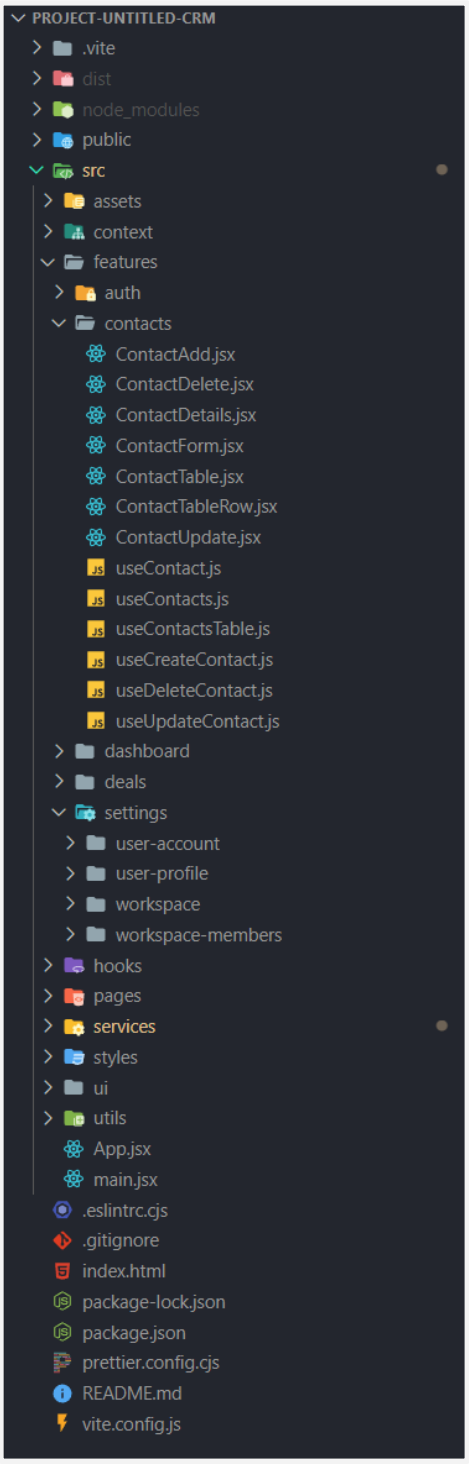
Datoteka “main.jsx” je glavna datoteka ili ulazna točka koja se koristi za pokretanje React aplikacije te definira korijensku komponentu aplikacije “App.jsx” koja se pomoću ReactDOM komponente i render metode povezuje s “root” elementom “index.html” datoteke koji predstavlja tijelo dokumenta ili “body”.

Mapa “context” sadrži datoteku “AppContext.jsx” koja predstavlja globalni kontekst, a sadrži kod za upravljanje svijetlim i tamnim načinom rada te postavljanje naziva stranice u komponenti “AppHeader.jsx”. Mapa “hooks” sadrži “React hooks” poput “useApp” koja omogućuje konzumaciju globalnog konteksta te “useLocalStorage” za privremenu pohranu podataka u lokalno spremište klijentovog preglednika.

Svako svojstvo aplikacije poput “auth” za autentifikaciju, “dashboard” za prikaz nadzorne ploče i grafikona, “contacts” i “deals” za prikaz tablice kontakata ili ponuda i izvođenje operacija nad njima, “settings” za upravljanje postavkama ima svoju mapu unutar mape “features”. Mape svojstva sadrže datoteke tipa “.jsx” koje predstavljaju komponente aplikacije te “.js” koje služe za pisanje “React hooks” koje počinju prefiksom “use”.

U mapi “pages” su definirane stranice aplikacije od kojih svaka ima svoju URL adresu. Mapa “styles” sadrži datoteku “GlobalStyles.jsx” koja postavlja stil HTML elemenata na globalnoj razini te ima definirane varijable za paletu boja i klase “light-mode” i “dark-mode” za promjenu svijetlog ili tamnog načina rada koje se primjenjuju na HTML dokument ovisno o korisničkim preferencijama. Mapa “ui” sadrži komponente koje ne trebaju posebnu logiku, nisu dio određenog svojstva aplikacije te se primjenjuju na globalnoj razini primjerice, “AppHeader.jsx”, “AppLayout.jsx”, “AppSidebar.jsx” i “Spinner.jsx” za prikazivanje stanja učitavanja prilikom dohvaćanja podataka. Mapa “utils” sadrži pomoćne funkcije za formatiranje datuma i valute te konstante.

Mapa “services” je jedna od važnijih jer omogućava povezivanje sa Supabase klijentom za dohvaćanje podataka iz baze podataka. Obuhvaća datoteku “supabase.js” koja sadrži kod za povezivanje s projektom na Supabase platformi. Za povezivanje je potrebno instalirati “supabase” paket i uvesti metodu “createClient” te stvoriti varijablu supabase koja sprema rezultat poziva metode. Zatim se varijabla supabase uvozi u datoteke koje počinju prefiksom “api” i nazivom tablice primjerice, “apiContact.js” ili “apiWorkspace.js”, a sadrže kod i logiku za čitanje, stvaranje, ažuriranje i brisanje podataka te tablice.



Slika 25: *Prikaz strukture projekta*

Na slici 25 prikazana je struktura mape i datoteka projekta. U slijedećem potpoglavlju će biti prikazani različiti primjeri koda uključujući i kod za dohvaćanje podataka iz baze podataka.

5.3. Objašnjenje i primjeri koda

S obzirom da web aplikacija UntitledCRM sadrži pozamašnu količinu koda te nije moguće proći i objasniti svaku liniju koda, cilj ovog odlomka je pružiti uvid u ključne dijelove koda koji se ponavljaju i u ostalim komponentama.

5.3.1. Komponenta "App.jsx"

App.jsx

src > App.jsx > queryClient > defaultOptions > queries

```jsx
function App() {
  return (
    <QueryClientProvider client={queryClient}>
      <GlobalStyles />
      <AppProvider>
        <BrowserRouter>
          <Routes>
            <Route path="/">
              <Route index element={<Navigate replace to="auth" />} />
              <Route path="auth" element={<Auth />} />
            </Route>
            <Route
              path="workspace/"
              element={
                <ProtectedRoute>
                  <AppLayout />
                </ProtectedRoute>
              }>
              <Route index element={<Navigate replace to="dashboard" />} />
              <Route path="dashboard" element={<Dashboard />} />
              <Route path="contacts" element={<Contacts />} />
              <Route path="contacts/:contactId" element={<Contact />} />
              <Route path="deals" element={<Deals />} />
              <Route path="settings" element={<SettingsLayout />}>
                <Route index element={<Workspace />} />
                <Route path="members" element={<WorkspaceMemebers />} />
                <Route path="profile" element={<UserProfile />} />
                <Route path="account" element={<UserAccount />} />
              </Route>
            </Route>
            <Route path="*" element={<PageNotFound />} />
          </Routes>
        </BrowserRouter>
      </AppProvider>
      <AppToaster />
    </QueryClientProvider>
  );
}

export default App;
```

Slika 26: *Kod za komponentu "App.jsx"*

Na slici 26 možemo primijetiti korištenje komponenti “BrowserRouter”, “Routes” i “Route” iz knjižnice “React Router DOM” koje definiraju rute odnosno URL strukturu aplikacije.

Tekst unutar komponente obojan žutom bojom (ne uključujući naziv funkcije “App”) predstavlja “props” odnosno svojstva koja se proslijeđuju u komponentu primjerice, varijabla “path” označava relativnu putanju, a varijabla “element” prosljeđuje određenu komponentu.

Možemo primijetiti da se komponenta “BrowserRouter” koja je roditelj komponentama u mapi “pages”, (a dalje je mapa “pages” roditelj komponentama u mapi “features”) nalazi unutar komponenti “QueryClientProvider” te “AppProvider” koje pružaju kontekst svim komponentama koje su im djeca. “AppProvider” komponenta je uvezena iz datoteke “AppContext.jsx”, a komponenta “QueryClientProvider” je dio knjižnice “React query” te omogućava komponentama koje su joj djeca da koriste njezine metode ili kuke (engl. “hooks”) za dohvaćanje podataka.

5.3.2. Komponenta "ContactTable.jsx"

```jsx
function ContactTable() {
  const {
    user: { workspace_id: workspaceId },
  } = useUser();
  const { contacts, count, isLoadingContacts } = useContactsTable({
    workspaceId,
  });

  const [searchParams, setSearchParams] = useSearchParams();

  function handleSubmit(e) {
    e.preventDefault();
    searchParams.delete('page');
    setSearchParams(searchParams);
  }

  if (isLoadingContacts)
    return (
      <Spinner.Wrapper>
        <Spinner />
      </Spinner.Wrapper>
    );

  return (
    <Row
      $flexDirection="column"
      $alignItems="stretch"
      $backgroundColor="normal"
      $border="1"
      $borderRadius="sm"
      $padding="3.2rem">
      <Row
        $justifyContent="space-between"
        $alignItems="center"
        $gap={'2.4rem'}
        $margin="0 0 2.4rem 0">
        <Text size="large">Total contacts ({count})</Text>
        <Row $gap={'1.6rem'}>
          <Form onSubmit={handleSubmit}>
            <Form.Input
              type="text"
              id="search-contacts"
              placeholder="Search contacts by name..."
              defaultValue={searchParams.get('search') ?? ''}
              onChange={(e) => {
                const queryValue = e.target.value.trim().toLowerCase();
                if (!queryValue) {
                  searchParams.delete('search');
                } else {
                  searchParams.set('search', queryValue);
                }
              }}
            />
          </Form>
          <ContactAdd />
        </Row>
      </Row>
      <Table.Wrapper>
        <Table
          role="table"
          columns="minmax(16rem, 0.4fr) minmax(18rem, 0.75fr) minmax(10rem, 0.5fr)
                   minmax(16rem, 0.5fr) minmax(10rem, 0.5fr) 6.8rem;">
          <Table.Header role="row">
            <Table.Column>Full name</Table.Column>
            <Table.Column>Email</Table.Column>
            <Table.Column>Phone</Table.Column>
            <Table.Column>Created by</Table.Column>
            <Table.Column>Created at</Table.Column>
          </Table.Header>
          <Table.Body
            data={contacts}
            render={(contact) => (
              <ContactTableRow
                contactDetails={contact}
                key={contact.contact_id}
              />
            )}
          />
          <Table.Footer>
            <Pagination count={count} />
          </Table.Footer>
        </Table>
      </Table.Wrapper>
    </Row>
  );
}
```

Slika 27: *Kod za komponentu "ContactTable.jsx"*

Komponenta “ContactTable” služi za prikaz tablice kontakata koja sadrži naslov s ukupnim brojem kontakata, gumb za stvaranje novog kontakta te paginaciju (Slika 27). Koristeći prilagođenu (engl. *custom*) kuku ili “React hook” naziva “useContactsTable” možemo dohvatiti popis kontakata na način da u metodu proslijedimo primarni ključ radnog prostora “workspaceID” kojeg možemo dobiti iz trenutne sesije korisnika koju ponovno možemo dobiti korištenjem kuke naziva “useUser”. U ovom slučaju, kuka “useSearchParams” služi za postavljanje URL parametara za pretragu kontakata po imenu ili prezimenu. Nakon što se obrazac za pretragu pošalje, koristi se kuka koja dohvaća podatke iz URL-a te koristi kuku “useQuery” za dohvaćanje rezultata koristeći metode napisane u datotekama “services” mape. Isto vrijedi i za paginaciju.

Iz korištenih kuka možemo dohvatiti varijablu “isLoadingContacts” koja predstavlja status dohvaćanja podataka odnosno ako se podaci dohvaćaju vrijednost će joj biti “True”, a ako su dohvaćeni vrijednost će joj biti “False”. Na temelju te varijable možemo prikazati stanje učitavanja koristeći komponentu “Spinner”.

Kada su kontakti dohvaćeni, spremljeni su u varijablu “contacts” u obliku polja (engl. *array*) te se prosljeđuju u komponentu “Table.Body” korištenjem “props” varijabli “data” i “render”. Za svaki objekt odnosno kontakt u polju stvara se komponenta “Row” koja označava redak te prikazuje informacije o kontaktu.

5.3.3. Kuka "useContactsTable.js"

useContactsTable.js

src > features > contacts > useContactsTable.js > ...

```
function useContactsTable({ workspaceId }) {
  const queryClient = useQueryClient();
  const [searchParams] = useSearchParams();

  const page = !searchParams.get('page') ? 1 : Number(searchParams.get('page'));
  const search = searchParams.get('search');

  const {
    data: { data: contacts, count } = {},
    isLoading: isLoadingContacts,
    error: contactsError,
  } = useQuery({
    queryKey: ['contacts', workspaceId, page, search],
    queryFn: () => getContactsApi({ workspaceId, page, search }),
  });

  const pageCount = Math.ceil(count / PAGE_SIZE);
  if (page < pageCount) {
    queryClient.prefetchQuery({
      queryKey: ['contacts', workspaceId, page + 1],
      queryFn: () => getContactsApi({ workspaceId, page: page + 1 }),
    });
  }

  if (page > 1) {
    queryClient.prefetchQuery({
      queryKey: ['contacts', workspaceId, page - 1],
      queryFn: () => getContactsApi({ workspaceId, page: page - 1 }),
    });
  }

  return {
    contacts,
    count,
    isLoadingContacts,
    contactsError,
  };
}

export { useContactsTable };
```

Slika 28: *Kod za komponentu "useContactsTable.js"*

Kuka "useContactsTable" sadrži logiku za dohvaćanje podataka koji se koriste u komponenti "ContactTable.jsx" iz baze podataka. Metoda "searchParams.get" služi za dohvaćanje URL parametara te ih na osnovu naziva sprema u varijablu "page" ili "search" (Slika 28). Kuka "useQuery" koristi se za dohvaćanje podataka pomoću funkcije "getContactsApi" u koju se proslijeđuju varijable poput ključa radnog prostora, stranice koje se dohvaća i teksta pretrage. Ova funkcija je definirana u datoteci "apiContact.js". Za keširanje odgovora koristi se "queryKey" koji omogućuje ponovno dohvaćanje podataka u slučaju promjena ili ažuriranja. Metoda "prefetchQuery" varijable "queryClient" služi za

dohvaćanje podataka koji se nalaze jednu stranicu više ili manje u odnosnu na stranicu koja se otvara kako bi korisnik imao bolje iskustvo i ne bi morao čekati na učitavanje.

5.3.4. Usluga "apiContact.js"

Kao što je prije spomenuto, prvo moramo uvesti varijablu "supabase" na koju možemo dodavati razne metode za dohvaćanje i manipulaciju podataka. Funkcija "getContacts" se izveze te prilikom uvoza u kuku "useContactsTable" se preimenuje u "getContactsTableApi" radi boljeg razumijevanja.

apiContact.js

src > services > apiContact.js > getContact

You, 2 weeks ago | 1 author (You)

```
1  import supabase, { supabaseUrl } from './supabase';
2  import { PAGE_SIZE } from '../utils/constants';
3
4  export async function getContacts({ workspaceId, page, search }) {
5    let query = supabase
6      .from('contact')
7      .select(
8        `*,
9      user_profile(*)`,
10       { count: 'exact' },
11     )
12     .eq('workspace_id', workspaceId);
13
14   if (search) {
15     query = query.or(
16       `contact_first_name.ilike.%${search}%,contact_last_name.ilike.%${search}%`,
17     );
18   }
19
20   if (page) {
21     const from = (page - 1) * PAGE_SIZE;
22     const to = from - 1 + PAGE_SIZE;
23     query = query.range(from, to);
24   }
25
26   query = query.order('contact_id', { ascending: false });
27
28   const { data, error, count } = await query;
29
30   if (error) {
31     console.log(error);
32     throw new Error('There was a problem while fetching contacts data.');
33   }
34
35   return { data, count };
36 }
37
```

Slika 29: *Kod za uslugu "apiContact.js"*

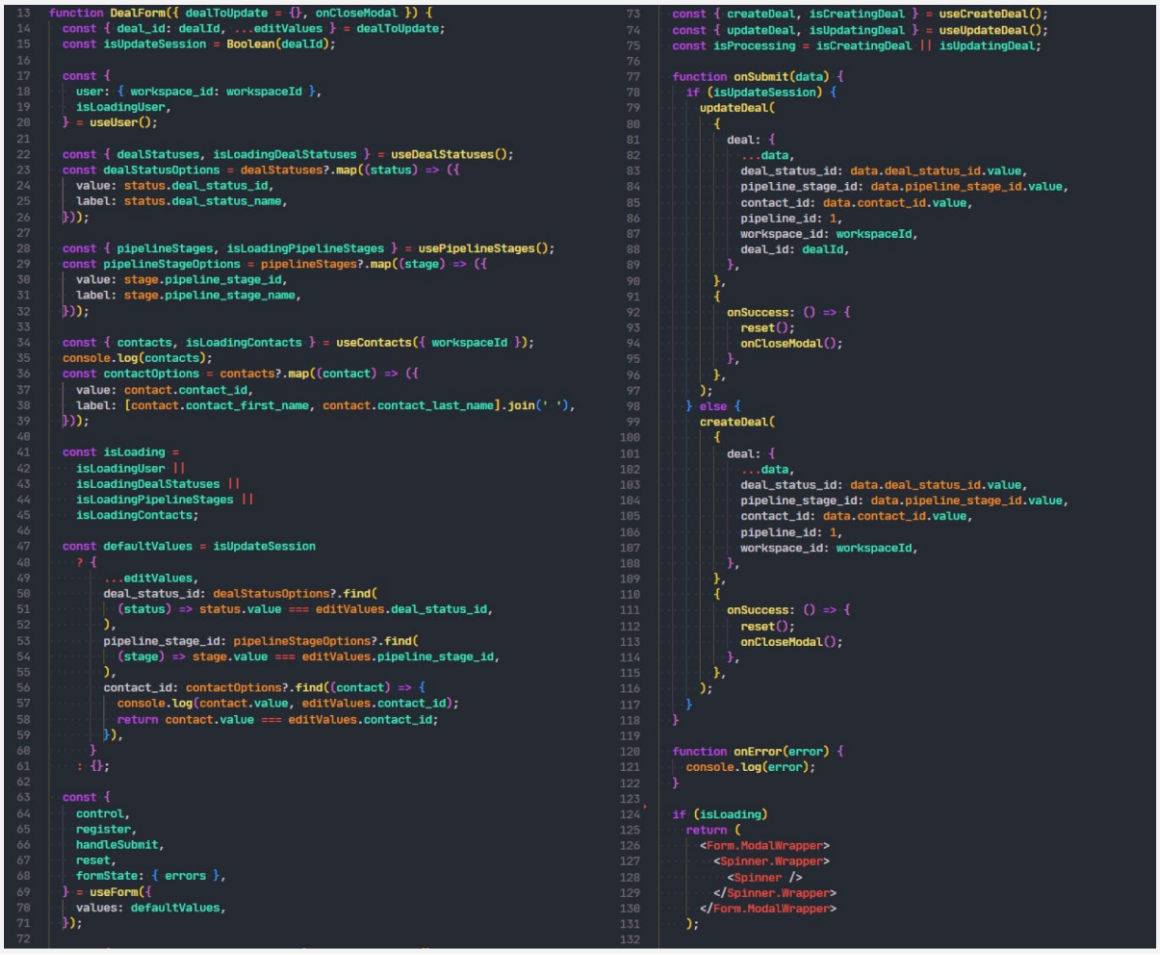
U ovom slučaju (Slika 29), želimo dohvatiti sve podatke koji pripadaju tablici “contact” i dodatno podatke profila korisnika koji je stvorio zapis, a pripadaju korisničkom radnom prostoru.

Korištenjem “if/else” petlje možemo uvjetno dodati i druge metode na inicijalni upit ovisno o postojanju varijabli. Ako varijabla “search” postoji, dodat će metodu “or” u kojoj se izdvajaju korisnici na temelju niza riječi u varijabli “search” po imenu i prezimenu ili ako postoji varijabla “page”, dohvaćamo kontakte za tu stranicu.

Na kraju, dodaje se metoda “order” koja poreda kontakte s obzirom na primarni ključ koji je rastući te se na vrhu uvijek prikazuju novi kontakti. Korištenjem ključne riječi “await query” čekamo na obradu upita te dohvaćamo varijable odnosno podatke poput “data” koja sadrži tražene podatke. Ukoliko je došlo do pogreške “error” sadrži naziv i opis greške te varijabla “count” sadrži broj vraćenih zapisa.

Podaci i broj vraćenih zapisa se vraćaju kao objekt te se mogu izvesti iz varijable “data” kao što je prikazano na slici 28.

5.3.5. Komponenta "DealForm.jsx"



```jsx
function DealForm({ dealToUpdate = {}, onCloseModal }) {
  const { deal_id: dealId, ...editValues } = dealToUpdate;
  const isUpdateSession = Boolean(dealId);

  const {
    user: { workspace_id: workspaceId },
    isLoadingUser,
  } = useUser();

  const { dealStatuses, isLoadingDealStatuses } = useDealStatuses();
  const dealStatusOptions = dealStatuses?.map((status) => ({
    value: status.deal_status_id,
    label: status.deal_status_name,
  }));

  const { pipelineStages, isLoadingPipelineStages } = usePipelineStages();
  const pipelineStageOptions = pipelineStages?.map((stage) => ({
    value: stage.pipeline_stage_id,
    label: stage.pipeline_stage_name,
  }));

  const { contacts, isLoadingContacts } = useContacts({ workspaceId });
  console.log(contacts);
  const contactOptions = contacts?.map((contact) => ({
    value: contact.contact_id,
    label: [contact.contact_first_name, contact.contact_last_name].join(' '),
  }));

  const isLoading =
    isLoadingUser ||
    isLoadingDealStatuses ||
    isLoadingPipelineStages ||
    isLoadingContacts;

  const defaultValues = isUpdateSession
    ? {
        ...editValues,
        deal_status_id: dealStatusOptions?.find(
          (status) => status.value === editValues.deal_status_id,
        ),
        pipeline_stage_id: pipelineStageOptions?.find(
          (stage) => stage.value === editValues.pipeline_stage_id,
        ),
        contact_id: contactOptions?.find((contact) => {
          console.log(contact.value, editValues.contact_id);
          return contact.value === editValues.contact_id;
        }),
      }
    : {};

  const {
    control,
    register,
    handleSubmit,
    reset,
    formState: { errors },
  } = useForm({
    values: defaultValues,
  });

  const { createDeal, isCreatingDeal } = useCreateDeal();
  const { updateDeal, isUpdatingDeal } = useUpdateDeal();
  const isProcessing = isCreatingDeal || isUpdatingDeal;

  function onSubmit(data) {
    if (isUpdateSession) {
      updateDeal(
        {
          deal: {
            ...data,
            deal_status_id: data.deal_status_id.value,
            pipeline_stage_id: data.pipeline_stage_id.value,
            contact_id: data.contact_id.value,
            pipeline_id: 1,
            workspace_id: workspaceId,
            deal_id: dealId,
          },
        },
        {
          onSuccess: () => {
            reset();
            onCloseModal();
          },
        },
      );
    } else {
      createDeal(
        {
          deal: {
            ...data,
            deal_status_id: data.deal_status_id.value,
            pipeline_stage_id: data.pipeline_stage_id.value,
            contact_id: data.contact_id.value,
            pipeline_id: 1,
            workspace_id: workspaceId,
          },
        },
        {
          onSuccess: () => {
            reset();
            onCloseModal();
          },
        },
      );
    }
  }

  function onError(error) {
    console.log(error);
  }

  if (isLoading)
    return (
      <Form.ModalWrapper>
        <Spinner.Wrapper>
          <Spinner />
        </Spinner.Wrapper>
      </Form.ModalWrapper>
    );
```

Slika 30: *Kod za komponentu "DealForm.jsx"*

Komponenta "DealForm.jsx" jedna je od kompleksnijih komponenti, a svrha joj je stvaranje novih ili ažuriranje postojećih ponuda (Slika 30).

Pozivi prilagođenim kukama "useUser", "useDealStatuses", "usePipelineStages", "useContacts" koriste se za dohvaćanje podataka o korisniku, statusima ponude, stadija prodajnog lijevka i kontakata. Ovisno o tome radi li se o sesiji ažuriranja ili stvaranja nove ponude automatski popuniti polja u obrascu ili otvoriti prazan obrazac. Izdvojene varijable koje počinju na "isLoading" služe za prikaz stanja učitavanja i komponente "Spinner".

Kuka "useForm" je posebna kuka iz knjižnice "React hook form" za upravljanje obrascima, postavljanje inicijalnih vrijednost, validacije te praćenje promjene korisničkog unosa.

Kuke “useCreateDeal” i “useUpdateDeal” se koriste za stvaranje ili ažuriranje ponude, a temelje se na kuki “useMutation” koja je dio “React query” knjižnice te omogućava takvu manipulaciju podacima uključujući i brisanje zapisa.

Metode “onSubmit” i “onError” se pozivaju prilikom slanja obrasca. Metoda “onSubmit” prima podatke iz obrasca te koristeći “if/else” petlju odlučuje radi li se o sesiji stvaranja nove ili ažuriranja postojeće ponude i sukladno tome koristi prethodno navedene kuke u koje prosljeđuje informacije prikupljene obrascem.

5.3.6. Komponenta “DashboardCard.jsx”

“DashboardCard.jsx” je komponenta s dva svojstva: “title” koji postavlja naslov kartice na "Pipeline stages distribution" i “icon” koji postavlja ikonu kartice, a sadrži tortni grafikon za prikaz distribucije stadija prodajnog lijevka korištenjem “Recharts” knjižnice (Slika 31). Komponente “PieChartContainer” i “ResponsiveContainer” omogućavaju prilagodbu veličine grafikona prema raspoloživom prostoru. “Pie” komponenta koristi podatke iz varijable “pipelineStages” za stvaranje grafikona, a svojstvo “name” označava naziv stadija prodajnog lijevka.

```jsx
<DashboardCard title="Pipeline stages distribution" icon={<FilterIcon />}>
  {pipelineStages.length ? (
    <PieChartContainer>
      <ResponsiveContainer height="100%" width="100%">
        <PieChart>
          <Pie
            data={pipelineStages}
            nameKey="name"
            dataKey="pipelineShare"
            innerRadius={75}
            outerRadius={180}
            paddingAngle={2}
            style={{ stroke: 'var(--border-non-interactive)' }}>
            <LabelList
              dataKey={(entry) =>
                `${entry.name} (${entry.pipelineShare.toFixed(2)}%)`
              }
              position="inside"
              angle="15"
              style={{
                fontFamily: '"Inter", sans-serif',
                fontWeight: '400',
                fontSize: '0.9rem',
                stroke: 'var(--text-lc)',
              }}
            />
          </Pie>
        </PieChart>
      </ResponsiveContainer>
    </PieChartContainer>
  ) : (
    'No records found.'
  )}
</DashboardCard>
```

Slika 31: *Kod za komponentu "DashboardCard.jsx"*

Komponenta "LabelList" sadrži naziv stadija i postotak udjela stadija u prodajnom lijevku koji je zaokružen na dvije decimale koristeći metodu "toFixed". Ostala svojstva služe za definiranje izgleda grafikona.

5.3.7. Komponenta "main.jsx"

U ovom kodu se nalazi komponenta "ErrorBoundary" iz spomenute knjižnice "React error boundary" koja u sebi sadrži glavnu komponentu React aplikacije "App" te ako se dogodi greška tijekom izvođenja aplikacije odnosno koda u "App" komponenti, greška će biti uhvaćena.

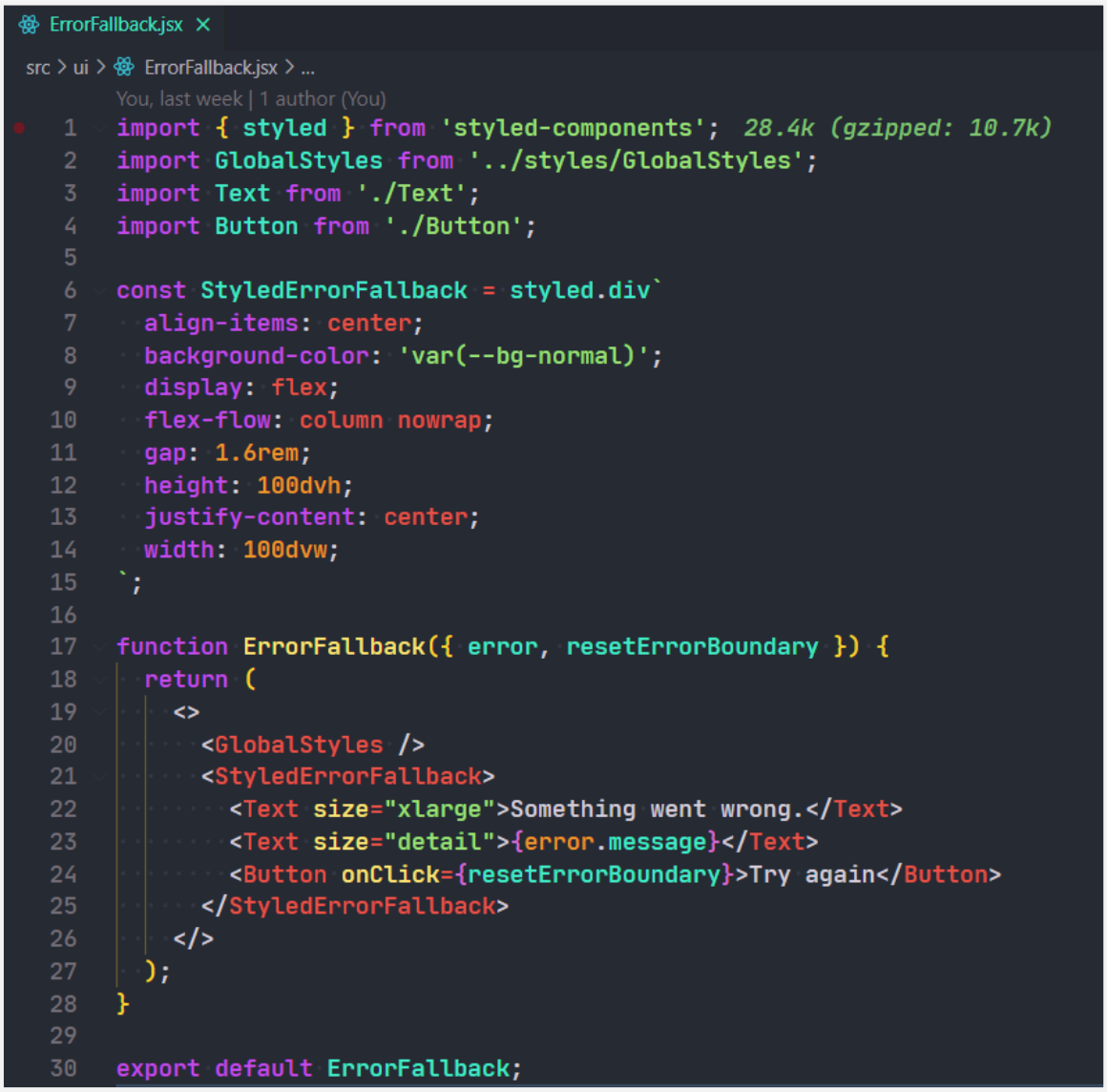
Slika 32: *Kod za komponentu "main.jsx"*

Svojstvo ili "prop" naziva "FallbackComponent" postavlja komponentu koja će se koristiti kao prikaz u slučaju greške, a svojstvo "onReset" sadrži funkciju za preusmjeravanje korisnika na početnu stranicu (Slika 32).

5.3.8. Komponenta "ErrorFallback.jsx"

Komponenta "ErrorFallback" prihvaća svojstvo ili prop odnosno funkciju "onReset" pod nazivom "resetErrorBoundary" te stvara gumb sa sadržajem "Try again" koji prilikom klika aktivira danu metodu.

U kodu komponente možemo primijetiti korištenje metode "styled" iz knjižnice "Styled componentes" koja omogućava pisanje CSS koda unutar Javascript jezika odnosno React komponente.



Slika 33: *Kod za komponentu "ErrorFallback.jsx"*

U ovom slučaju varijabla "StyledErrorFallback" predstavlja komponentu HTML elementa div sa stilom definiranim u kodu (Slika 33).

Zaključno s ovom komponentom objašnjeni su ključni dijelovi koda te primjeri koda za svaku knjižnicu koja je korištena za razvoj web aplikacije UntitledCRM. U popisu priloga je dodana poveznica na Github gdje se može pronaći cijeli kod te uputstva za pokretanje aplikacije na lokalnom računalu.

6. ZAKLJUČAK

Ovim završnim radom dokumentiran je proces dizajna i razvoja sustava za upravljanje odnosima s klijentima. Opisane su sve faze razvoja sustava odnosno web aplikacije od analize karakteristika postojećih sustava, dizajna korisničkog sučelja i izrade prototipa, dizajna i implementacije baze podataka, razvoja klijentskog dijela aplikacije, korištene metode, alati i tehnologije te njihova implementacija.

Za dizajn korisničkog sučelja i izradu prototipa korišten je popularan alat Figma te su izrađene korisničke persone i dijagram korisničkog toka u svrhu vizualizacije i simulacije korištenja konačne web aplikacije. Prikazana je usporedba prototipa aplikacije i konačne verzije aplikacije te možemo zaključiti da konačna verzija aplikacije ima određena odstupanja od inicijalnog dizajna korisničkog sučelja zbog lakše tehničke implementacije.

Može se primijetiti da je korisnicima sustava za upravljanje odnosima s klijentima važno da mogu na sustavan i organiziran način spremati informacije o potencijalnim i trenutnim klijentima, pratiti kretanje ponude kroz prodajni lijevak te kroz osnovno izvješće dobiti uvid u trenutno stanje tvrtke na temelju čega mogu donositi bolje poslovne odluke.

Za modeliranje baze podataka korištena je metodologija MIRIS te dijagram metoda entiteti - veze, a izrađena je pomoću platforme Supabase koja u pozadini koristi PostgreSQL relacijsku bazu podataka i automatski na osnovu modela baze podataka i tablica stvara REST API koji omogućava manipulaciju podataka te uklanja potrebu za razvojem poslužiteljskog dijela aplikacije što rezultira bržim razvojem i smanjenom potrebom resursa za održavanje rada aplikacije.

Klijentski dio aplikacije razvijen je korištenjem standardnih web tehnologija poput HTML, CSS, Javascript, ali i knjižnice React koja je popularna te stoga ima veliku bazu korisnika odnosno dobro je dokumentirana te je lakše pronaći programere koji mogu nastaviti daljnji razvoj aplikacije.

Konačna verzija sustava za upravljanje odnosima s klijentima je jednostavna. Bitno je naglasiti da je potrebno uložiti dosta vremena i znanja za razvoj takve web aplikacije te možemo zaključiti kako postojeće kompanije koje se bave razvojem takvih sustava moraju uložiti jako puno resursa za razvoj korisne, funkcionalne i skalabilne web aplikacije.

7. LITERATURA

Adhikary, T. (2022). React Hooks Fundamentals for Beginners. Preuzeto 9. rujna 2023. s https://www.freecodecamp.org/news/react-hooks-fundamentals/

Adservio. (2022). Introduction To Supabase. Preuzeto 8. rujna 2023. s https://www.adservio.fr/post/introduction-to-supabase

Ahmed, T. (2023). Information Architecture (IA) and Sitemap: Simple and Easy to Understand. Preuzeto 7. rujna 2023. s https://www.linkedin.com/pulse/information-architecture-ia-sitemap-simple-easy-understand-ahmed/

Apte, S. (2023). React error handling with react-error-boundary. Preuzeto 8. rujna 2023. s https://blog.saeloun.com/2023/07/06/react-error-boundaries/

Bianco, G. (2023). 7 of the Best Sales Automation Tools on the Market. Preuzeto 23. rujna 2023. s https://www.nutshell.com/blog/sales-automation-tools

Boateng, D. (2023). How to Use the React Context API in Your Projects. Preuzeto 8. rujna 2023. s https://www.freecodecamp.org/news/context-api-in-react/

Browne, C. (2023). What are User Flows in User Experience (UX) Design? Preuzeto 1. rujna 2023. s https://careerfoundry.com/en/blog/ux-design/what-are-user-flows/

Cflowapps. (2023). A Comprehensive Guide to Flowchart Symbols and Notations for Process Workflows. Preuzeto 7. rujna 2023. s https://www.cflowapps.com/flowchart-symbols/

Dev. (2023). What is react router? Preuzeto 9. rujna 2023. s https://dev.to/duomly/what-is-react-router-ap6

Figma. (2020). What is Figma? Preuzeto 1. rujna 2023. s https://help.figma.com/hc/en-us/articles/14563969806359-What-is-Figma

Hetler, A. (2023). What is ChatGPT? Preuzeto 10. rujna 2023. s https://www.techtarget.com/whatis/definition/ChatGPT

Hotjar. (2023). An introduction to UI prototyping. Preuzeto 1. rujna 2023. s https://www.hotjar.com/ui-design/glossary/prototype/

Hygraph. (2022). React Hook Form - A Complete Guide. Preuzeto 8. rujna 2023. s https://hygraph.com/blog/react-hook-form

Lamprecht, E. (2023). The Difference Between UX and UI Design: A Beginner's Guide. Preuzeto 1. rujna 2023. s https://careerfoundry.com/en/blog/ux-design/the-difference-between-ux-and-ui-design-a-laymans-guide/

Makode, K. (2023). How to Use Styled Components in Your React Apps. Preuzeto 8. rujna 2023. s https://www.freecodecamp.org/news/styled-components-in-react/

Marković, P. (2019). Korisničke persone. Preuzeto 1. rujna 2023. s https://pedja.online/korisnicke-persone/

Pavlić, M. (2011). Informacijski sustavi. Školska knjiga. Preuzeto 8. rujna 2023.

Pavlić, M. (2011). Oblikovanje baze podataka. Rijeka: Odjel za informatiku Sveučilišta u Rijeci. Preuzeto 12. srpnja 2023.

Simplilearn. (2023). The Best Guide to Know What Is React. Preuzeto 8. rujna 2023. s https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs

Starčević, I. (2023). What is React Query: everything you need to know. Preuzeto 8. rujna 2023. s https://decode.agency/article/what-is-react-query

Wikipedia. (2023). Customer relationship management. Preuzeto 23. rujna 2023. s https://en.wikipedia.org/wiki/Customer_relationship_management

8. POPIS SLIKA

Slika 1: Grupiranje snimki zaslona s obzirom na temeljne funkcionalnosti…….4

Slika 2: Korisnička persona “Ivan Kovačić”.....7

Slika 3: Dijagram korisničkog toka za korisničku personu “Ivan Kovačić”.....8

Slika 4: Hijerarhija informacija ili mapa aplikacije……..…..10

Slika 5: Paleta boja……..11

Slika 6: Logo……...11

Slika 7: Ekran “Auth” - obrazac za registraciju……..……13

Slika 8: Ekran “Auth” - obrazac za prijavu…….14

Slika 9: Ekran “Dashboard”.....15

Slika 10: Ekran “Contacts”.....16

Slika 11: Ekran “Contacts” - obrazac za stvaranje i ažuriranje kontakta……17

Slika 12: Ekran “Contact”.....18

Slika 13: Ekran “Deals”.....19

Slika 14: Ekran “Deals” - obrazac za stvaranje i ažuriranje ponuda……...20

Slika 15: Ekran “Settings” - obrazac za ažuriranje poslovnih informacija…….21

Slika 16: Ekran “Settings” - obrazac za ažuriranje preferenca……....22

Slika 17: Ekran “Members” - popis članova radnog prostora…….23

Slika 18: Ekran “Members” - dodavanje novog člana u radni prostor……24

Slika 19: Ekran “Profile” - obrazac za ažuriranje korisničkog profila……25

Slika 20: Ekran “Account” - obrazac za ažuriranje adrese e-pošte korisničkog računa…..26

Slika 21: Ekran “Account” - obrazac za ažuriranje lozinke korisničkog računa…….27

Slika 22: EV dijagram…….29

Slika 23: Grafičko sučelje za stvaranje tablica……31

Slika 24: Prikaz ChatGPT naredbe za stvaranje testnih podataka……...32

Slika 25: Prikaz strukture projekta……..36

Slika 26: Kod za komponentu “App.jsx”.....37

Slika 27: Kod za komponentu “ContactTable.jsx”.....39

Slika 28: Kod za komponentu “useContactsTable.js”.....41

Slika 29: Kod za uslugu “apiContact.js”.....42

Slika 30: Kod za komponentu “DealForm.jsx”.....44

Slika 31: Kod za komponentu “DashboardCard.jsx”.....46

Slika 32: Kod za komponentu “main.jsx”.....47

Slika 33: Kod za komponentu “ErrorFallback.jsx”.....48

9. POPIS TABLICA

Tablica 1: Temeljne karakteristike UntitledCRM sustava…….5

10. POPIS PRILOGA

Kao prilog završnom radu ostavljam poveznicu na web aplikaciju te Github repozitorij gdje se nalazi kod i potrebne datoteke te detaljne upute za preuzimanje i pokretanje web aplikacije UntitledCRM na lokalnom računalu.

Poveznica na Github repozitorij: https://github.com/mateoniksic/project-untitled-crm/

Poveznica na web aplikaciju: https://project-untitled-crm.vercel.app/