**Dokumentacija projekta**

**Crypto Wallet App**

**Miljan Petrić 4696**

**Web aplikacija za praćenje i upravljanje digitalnim valutama**

**10. jun 2025.**

**FIT Metropolitan** **IT354**

**Sadržaj**

1. Uvod i teoretska postavka izabrane teme

1.1. Opis teme projekta

1.2. Cilj projekta i funkcionalnosti

1.3. Pregled tehnologija i alata

1. Planiranje i dizajn sistema

2.1. Definicija zahteva

2.2. Use-case dijagram

1. Razvoj korisničkog interfejsa (UI)

3.1. Opis tehnologija za stilizaciju

3.2. Pregled stranica

1. Implementacija funkcionalnosti

4.1. Korisničke funkcionalnosti

4.2. Administratorske funkcionalnosti

1. Simulacija backend-a

5.1. Korišćenje json-servera

5.2. Povezivanje frontenda i backend-a

5.3. Alternativne backend tehnologije

1. Zaključak

6.1. Pregled rezultata

6.2. Izazovi i prepreke

6.3. Mogućnosti za buduća unapređenja aplikacije

1. Liteeratura

**1. Uvod i teoretska postavka izabrane teme**

**1.1. Opis teme projekta** Crypto Wallet App je web aplikacija dizajnirana da korisnicima omogući efikasno praćenje, organizaciju i upravljanje njihovim digitalnim valutama. Aplikacija pruža intuitivan interfejs za vođenje evidencije o stanju novčanika, praćenje istorije transakcija i interakciju sa simuliranim tržišnim podacima. Kroz implementaciju različitih funkcionalnosti, cilj je stvoriti funkcionalno i korisno okruženje za upravljanje osnovnim aspektima kripto-portfolija.

**1.2. Cilj projekta i funkcionalnosti** Glavni cilj projekta je razvoj kompletne CRUD web aplikacije koristeći React framework, fokusirajući se na sledeće funkcionalnosti:

* Registracija i prijavljivanje korisnika: Siguran pristup aplikaciji.
* Pregled stanja novčanika: Prikaz balansa po kriptovalutama i ukupne vrednosti portfolija.
* Istorija transakcija: Detaljan uvid u poslate i primljene kriptovalute.
* Slanje kriptovaluta: Mogućnost prenosa digitalnih sredstava drugim registrovanim korisnicima.
* Simulacija fluktuacije cena: Realistično modelovanje promena tržišnih cena kriptovaluta (posebna funkcionalnost).
* Vizuelni prikaz portfolija: Grafička reprezentacija raspodele imovine u novčaniku korišćenjem eksterne biblioteke.
* Administrativni panel: Centralizovano upravljanje korisnicima, kriptovalutama i transakcijama.

**1.3. Pregled tehnologija i alata** Za razvoj aplikacije korišćene su sledeće ključne tehnologije i alati:

* **Frontend**: React 18 (za komponentnu arhitekturu) sa JavaScript-om.
* **Build alat**: React Scripts (uz Craco za naprednu konfiguraciju).
* **UI komponente**: Prilagođeni HTML elementi stilizovani čistim CSS klasama, bez upotrebe spoljnih UI biblioteka komponenti.
* **Validacija formi**: Standardne JavaScript provere i HTML5 validacija.
* **Rutiranje**: react-router-dom za upravljanje navigacijom i zaštitom ruta.
* **Backend simulacija**: JSON Server za kreiranje REST API mock servera.
* **API komunikacija**: axios biblioteka za slanje HTTP zahteva.
* **Grafikoni**: react-chartjs-2 biblioteka za vizuelizaciju podataka (eksterna biblioteka).
* **Pomoćni alati**: concurrently za istovremeno pokretanje razvojnog servera i JSON Servera.

**{**

  "name": "crypto-wallet-app",

  "version": "0.1.0",

  "private": true,

  "dependencies": {

    "@testing-library/jest-dom": "^5.17.0",

    "@testing-library/react": "^13.4.0",

    "@testing-library/user-event": "^13.5.0",

    "axios": "^1.6.8",

    "chart.js": "^4.4.2",

    "concurrently": "^8.2.2",

    "json-server": "^0.17.4",

    "react": "^18.2.0",

    "react-chartjs-2": "^5.2.0",

    "react-dom": "^18.2.0",

    "react-router-dom": "^6.23.0",

    "react-scripts": "5.0.1",

    "web-vitals": "^2.1.4"

  },

  "scripts": {

    "start": "react-scripts start",

    "build": "react-scripts build",

    "test": "react-scripts test",

    "eject": "react-scripts eject",

    "server": "json-server --watch db.json --port 5000",

    "dev": "concurrently \"npm run server\" \"npm start\""

  },

  "eslintConfig": {

    "extends": [

      "react-app",

      "react-app/jest"

    ]

  },

  "browserslist": {

    "production": [

      ">0.2%",

      "not dead",

      "not op\_mini all"

    ],

    "development": [

      "last 1 chrome version",

      "last 1 firefox version",

      "last 1 safari version"

    ]

  }

}

**2. Planiranje i dizajn sistema**

**2.1. Definicija zahteva** Definisanje zahteva sistema je ključno za postizanje funkcionalnog i korisnog rešenja.

**Funkcionalni zahtevi (prema tipovima korisnika):**

* **Standardni korisnici:**
  + Registracija novog korisničkog naloga (korisničko ime, lozinka).
  + Prijava na sistem sa postojećim kredencijalima.
  + Prikaz ličnog dashboarda sa informacijama o stanju novčanika (balansi po kriptovalutama, ukupna vrednost).
  + Pregled liste ličnih transakcija.
  + Mogućnost slanja kriptovaluta drugim registrovanim korisnicima.
  + Grafički prikaz raspodele imovine u novčaniku.
  + Odjava iz sistema.
* **Administratori:**
  + Pristup svim funkcionalnostima standardnog korisnika.
  + Pristup administrativnom panelu.
  + Upravljanje korisnicima: pregled, brisanje i ažuriranje korisničkih naloga.
  + Upravljanje kriptovalutama: dodavanje novih kriptovaluta, izmena cena i naziva, te brisanje podržanih kriptovaluta.
  + Pregled svih transakcija u sistemu sa opcijom brisanja transakcija.

**Nefunkcionalni zahtevi:**

* **Pouzdanost:** Aplikacija mora biti stabilna i otporna na greške pri komunikaciji sa backendom.
* **Performanse:** Brz odziv UI-ja i efikasno učitavanje podataka.
* **Sigurnost:** Osnovna zaštita korisničkih podataka (iako su lozinke heširane samo na frontend simulaciji). Zaštićene rute.
* **Prilagodljivost:** Kod treba da bude modularan i lak za proširenje.
* **Upotrebljivost:** Intuitivan i čist korisnički interfejs.

**2.2. Use-case dijagram** Use-case dijagrami ilustruju interakcije između aktera i sistema, pružajući vizuelni pregled funkcionalnosti. Postoje dva glavna aktera u sistemu:

1. **Standardni korisnik** - Korisnik koji obavlja finansijske operacije unutar svog novčanika.
2. **Administrator** - Korisnik sa proširenim privilegijama za upravljanje sistemom.

**Dijagram prikazuja sledeće ključne slučajeve korišćenja (use-cases):**

* **Neautentifikovani korisnik:**
  + Pristup Početnoj stranici
  + Registracija
  + Prijava
* **Standardni korisnik:**
  + Prijava (inherits from Neautentifikovani korisnik)
  + Pregled Dashboarda (Novčanik, Transakcije)
  + Slanje kriptovaluta
  + Pregled detalja transakcije
  + Odjava
* **Administrator:**
  + Prijava (inherits from Neautentifikovani korisnik)
  + Pristup Admin panelu
  + Upravljanje korisnicima (Pregled, Ažuriranje, Brisanje)
  + Upravljanje kriptovalutama (Dodavanje, Izmena, Brisanje)
  + Pregled svih transakcija

**3. Razvoj korisničkog interfejsa (UI)**

**3.1. Opis tehnologija za stilizaciju** Za razvoj korisničkog interfejsa aplikacije korišćen je **čisti CSS** (Plain CSS). Odluka o korišćenju čistog CSS-a doneta je zbog potrebe za potpunom kontrolom nad stilizacijom i rešavanja tehničkih izazova sa integracijom drugih CSS frameworka tokom razvoja. Stilovi su organizovani kroz semantičke klase definisane u src/index.css fajlu, koji služi kao centralna tačka za globalne stilove aplikacije.

body {

  font-family: var(--font-primary);

  line-height: 1.6;

  color: var(--text-color-dark);

  background-color: var(--bg-light);

  -webkit-font-smoothing: antialiased;

  -moz-osx-font-smoothing: grayscale;

  display: flex;

  flex-direction: column;

  min-height: 100vh;

}

**3.2. Pregled stranica** Aplikacija sadrži sledeće ključne stranice:

* **Početna stranica (**HomePage.jsx**):** Služi kao ulazna tačka u aplikaciju, pružajući kratak opis i opcije za prijavu/registraciju.

import React from 'react';

import { Link } from 'react-router-dom';

const HomePage = () => {

  return (

    <div className="homepage-wrapper"> { }

      <h1 className="homepage-title">Dobrodošli u CryptoWallet!</h1> { }

      <p className="homepage-description"> { }

        Vaše sigurno mesto za praćenje i upravljanje digitalnim valutama.

        Pratite svoje stanje, transakcije i budite u toku sa tržištem.

      </p>

      <div className="homepage-buttons"> { }

        <Link

          to="/register"

          className="btn-primary"

        >

          Registrujte se

        </Link>

        <Link

          to="/login"

          className="btn-secondary"

        >

          Prijavite se

        </Link>

      </div>

    </div>

  );

};

export default HomePage;

* **Stranice za registraciju i prijavu (**RegisterPage.jsx**,** LoginPage.jsx**):** Omogućavaju korisnicima da se registruju, kreiraju svoj nalog, i prijave u sistem. Implementirana je validacija unosa.

return (

    <div className="page-center"> { }

      <div className="form-card"> { }

        <h2 className="form-title">Registracija</h2> { }

        <form onSubmit={handleSubmit}>

          <div className="form-group"> { }

            <label className="form-label" htmlFor="username"> { }

              Korisničko Ime:

            </label>

            <input

              type="text"

              id="username"

              className="form-input"

              value={username}

              onChange={(e) => setUsername(e.target.value)}

              required

            />

          </div>

          <div className="form-group"> { }

            <label className="form-label" htmlFor="password"> { }

              Lozinka:

            </label>

            <input

              type="password"

              id="password"

              className="form-input"

              value={password}

              onChange={(e) => setPassword(e.target.value)}

              required

            />

          </div>

          <div className="form-group"> { }

            <label className="form-label" htmlFor="confirmPassword"> { }

              Potvrdi Lozinku:

            </label>

            <input

              type="password"

              id="confirmPassword"

              className="form-input"

              value={confirmPassword}

              onChange={(e) => setConfirmPassword(e.target.value)}

              required

            />

          </div>

          {error && <p className="text-error">{error}</p>} { }

          <div className="form-actions"> { }

            <button

              type="submit"

              className="btn-primary"

            >

              Registruj se

            </button>

          </div>

        </form>

      </div>

    </div>

  );

* **Dashboard Stranica (**DashboardPage.jsx**):** Glavna stranica za prijavljenog korisnika. Prikazuje pregled novčanika (trenutni balansi, ukupna vrednost portfolija), listu transakcija i grafikon raspodele imovine. Sadrži i formu za slanje kriptovaluta drugim korisnicima.

return (

        <div className="page-container">

            <h1 className="main-title">Moj Dashboard</h1>

            <div className="dashboard-grid-1">

                {wallet && <WalletOverview wallet={wallet} allCryptos={allCryptos} />}

                {wallet && <PortfolioChart wallet={wallet} allCryptos={allCryptos} />}

            </div>

            <div className="dashboard-grid-2">

                <div className="card">

                    <h3 className="card-title">Pošalji Kriptovalutu</h3>

                    <form onSubmit={handleSendCrypto}>

                        <div className="form-group">

                            <label className="form-label" htmlFor="receiverUsername">

                                Korisničko ime primaoca:

                            </label>

                            <input

                                type="text"

                                id="receiverUsername"

                                className="form-input"

                                value={receiverUsername}

                                onChange={(e) => setReceiverUsername(e.target.value)}

                                required

                            />

                        </div>

                        <div className="form-group">

                            <label className="form-label" htmlFor="sendAmount">

                                Iznos:

                            </label>

                            <input

                                type="number"

                                id="sendAmount"

                                className="form-input"

                                value={sendAmount}

                                onChange={(e) => setSendAmount(e.target.value)}

                                step="any"

                                required

                            />

                        </div>

                        <div className="form-group">

                            <label className="form-label" htmlFor="sendCurrency">

                                Kriptovaluta:

                            </label>

                            <select

                                id="sendCurrency"

                                className="form-select"

                                value={sendCurrency}

                                onChange={(e) => setSendCurrency(e.target.value)}

                            >

                                {Object.keys(wallet?.balances || {})

                                    .filter(symbol => wallet.balances[symbol] > 0)

                                    .map(symbol => (

                                        <option key={symbol} value={symbol}>{symbol}</option>

                                    ))}

                            </select>

                        </div>

                        {sendError && <p className="error-message">{sendError}</p>}

                        {sendSuccess && <p className="success-message">{sendSuccess}</p>}

                        <button

                            type="submit"

                            className="btn-primary"

                        >

                            Pošalji

                        </button>

                    </form>

                </div>

                <TransactionList transactions={transactions} userId={userId} />

            </div>

        </div>

    );

* **Stranica detalja transakcije (**TransactionDetailPage.jsx**):** Prikazuje detaljne informacije o specifičnoj transakciji.

return (

    <div className="bg-white p-8 rounded-lg shadow-lg w-full max-w-2xl mx-auto mt-8">

      <h2 className="text-3xl font-bold text-gray-800 mb-6 text-center">Detalji Transakcije</h2>

      <div className="space-y-4 text-lg">

        <p><strong>ID Transakcije:</strong> {transaction.id}</p>

        <p><strong>Tip:</strong> <span className={typeColor}>{type}</span></p>

        <p><strong>Iznos:</strong> {transaction.amount} {transaction.currency}</p>

        <p><strong>Pošiljalac:</strong> {transaction.senderId}</p>

        <p><strong>Primalac:</strong> {transaction.receiverId}</p>

        <p><strong>Datum:</strong> {new Date(transaction.date).toLocaleString()}</p>

      </div>

      <div className="mt-8 text-center">

        <Link to="/dashboard" className="bg-indigo-600 hover:bg-indigo-700 text-white font-bold py-2 px-4 rounded focus:outline-none focus:shadow-outline transition duration-300">

          Nazad na Dashboard

        </Link>

      </div>

    </div>

  );

* **Administrativni panel (**AdminDashboardPage.jsx**):** Centralna tačka za administratore, sa linkovima ka stranicama za upravljanje korisnicima, kriptovalutama i pregledom svih transakcija.

 return (

    <div className="admin-page-wrapper"> { }

      <h1 className="admin-title">Admin Panel</h1> { }

      <div className="admin-grid"> { }

        <Link

          to="/admin/users"

          className="admin-card"

        >

          <span className="admin-icon">👥</span> { }

          <h2 className="admin-card-title">Upravljanje Korisnicima</h2> { }

          <p className="admin-card-description">Kreiranje, ažuriranje i brisanje korisničkih naloga.</p> { }

        </Link>

        <Link

          to="/admin/cryptos"

          className="admin-card"

        >

          <span className="admin-icon">₿</span> { }

          <h2 className="admin-card-title">Upravljanje Kriptovalutama</h2> { }

          <p className="admin-card-description">Dodavanje, uređivanje i brisanje podržanih kriptovaluta.</p> { }

        </Link>

        <Link

          to="/admin/transactions"

          className="admin-card"

        >

          <span className="admin-icon">💸</span> { }

          <h2 className="admin-card-title">Pregled Transakcija</h2> { }

          <p className="admin-card-description">Pregled svih transakcija u sistemu.</p> { }

        </Link>

      </div>

    </div>

  );

* **Upravljanje korisnicima (**UserManagementPage.jsx**):** Omogućava administratorima pregled liste svih korisnika, kao i opcije za brisanje i ažuriranje korisničkih naloga.

 return (

    <div className="py-8">

      <h1 className="text-4xl font-bold text-gray-900 mb-8 text-center">Upravljanje Korisnicima</h1>

      <div className="bg-white p-6 rounded-lg shadow-xl mb-8">

        <h2 className="text-2xl font-semibold mb-4">{editingUser ? 'Ažuriraj Korisnika' : 'Dodaj Novog Korisnika'}</h2>

        <form onSubmit={editingUser ? handleUpdateUser : handleCreateUser}>

          <div className="mb-4">

            <label className="block text-gray-700 text-sm font-bold mb-2" htmlFor="username">

              Korisničko Ime:

            </label>

            <input

              type="text"

              id="username"

              className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline"

              value={newUsername}

              onChange={(e) => setNewUsername(e.target.value)}

              required

            />

          </div>

          <div className="mb-4">

            <label className="block text-gray-700 text-sm font-bold mb-2" htmlFor="password">

              Lozinka:

            </label>

            <input

              type="text"

              id="password"

              className="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline"

              value={newPassword}

              onChange={(e) => setNewPassword(e.target.value)}

              required

            />

          </div>

          <div className="mb-6">

            <label className="block text-gray-700 text-sm font-bold mb-2" htmlFor="role">

              Uloga:

            </label>

            <select

              id="role"

              className="shadow border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline"

              value={newRole}

              onChange={(e) => setNewRole(e.target.value)}

            >

              <option value="user">User</option>

              <option value="admin">Admin</option>

            </select>

          </div>

          {addUserError && <p className="text-red-500 text-xs italic mb-4">{addUserError}</p>}

          <div className="flex items-center justify-between">

            <button

              type="submit"

              className="bg-indigo-600 hover:bg-indigo-700 text-white font-bold py-2 px-4 rounded focus:outline-none focus:shadow-outline transition duration-300"

            >

              {editingUser ? 'Ažuriraj' : 'Dodaj Korisnika'}

            </button>

            {editingUser && (

              <button

                type="button"

                onClick={() => { setEditingUser(null); setNewUsername(''); setNewPassword(''); setNewRole('user'); }}

                className="bg-gray-500 hover:bg-gray-700 text-white font-bold py-2 px-4 rounded focus:outline-none focus:shadow-outline transition duration-300 ml-4"

              >

                Poništi

              </button>

            )}

          </div>

        </form>

      </div>

      <div className="bg-white p-6 rounded-lg shadow-xl">

        <h2 className="text-2xl font-semibold mb-4">Lista Korisnika</h2>

        {error && <p className="text-red-500 text-xs italic mb-4">{error}</p>}

        <table className="min-w-full bg-white">

          <thead>

            <tr>

              <th className="py-2 px-4 border-b text-left">ID</th>

              <th className="py-2 px-4 border-b text-left">Korisničko Ime</th>

              <th className="py-2 px-4 border-b text-left">Uloga</th>

              <th className="py-2 px-4 border-b text-left">Akcije</th>

            </tr>

          </thead>

          <tbody>

            {users.map(user => (

              <tr key={user.id} className="hover:bg-gray-50">

                <td className="py-2 px-4 border-b">{user.id}</td>

                <td className="py-2 px-4 border-b">{user.username}</td>

                <td className="py-2 px-4 border-b">{user.role}</td>

                <td className="py-2 px-4 border-b">

                  <button

                    onClick={() => handleEditUser(user)}

                    className="bg-blue-500 hover:bg-blue-700 text-white text-sm py-1 px-2 rounded mr-2"

                  >

                    Uredi

                  </button>

                  <button

                    onClick={() => handleDeleteUser(user.id)}

                    className="bg-red-500 hover:bg-red-700 text-white text-sm py-1 px-2 rounded"

                  >

                    Obriši

                  </button>

                </td>

              </tr>

            ))}

          </tbody>

        </table>

      </div>

    </div>

  );

* **Upravljanje kriptovalutama (**CryptoManagementPage.jsx**):** Stranica gde administratori mogu dodavati nove kriptovalute, menjati njihove nazive i cene, te brisati postojeće valute.

return (

    <div className="page-wrapper">

      <h1 className="page-title">Upravljanje Kriptovalutama</h1>

      <div className="card form-section-card">

        <h2 className="card-title">{editingCrypto ? 'Ažuriraj Kriptovalutu' : 'Dodaj Novu Kriptovalutu'}</h2>

        <form onSubmit={editingCrypto ? handleUpdateCrypto : handleCreateCrypto}>

          <div className="form-group">

            <label className="form-label" htmlFor="symbol">

              Simbol (npr. BTC):

            </label>

            <input

              type="text"

              id="symbol"

              className="form-input"

              value={newSymbol}

              onChange={(e) => setNewSymbol(e.target.value)}

              required

              disabled={!!editingCrypto}

            />

          </div>

          <div className="form-group">

            <label className="form-label" htmlFor="name">

              Naziv (npr. Bitcoin):

            </label>

            <input

              type="text"

              id="name"

              className="form-input"

              value={newName}

              onChange={(e) => setNewName(e.target.value)}

              required

            />

          </div>

          <div className="form-group">

            <label className="form-label" htmlFor="price">

              Trenutna Cena (USD):

            </label>

            <input

              type="number"

              id="price"

              className="form-input"

              value={newPrice}

              onChange={(e) => setNewPrice(e.target.value)}

              step="any"

              required

            />

          </div>

          {addCryptoError && <p className="error-message">{addCryptoError}</p>}

          <div className="form-actions">

            <button

              type="submit"

              className="btn-primary"

            >

              {editingCrypto ? 'Ažuriraj' : 'Dodaj Kriptovalutu'}

            </button>

            {editingCrypto && (

              <button

                type="button"

                onClick={() => { setEditingCrypto(null); setNewSymbol(''); setNewName(''); setNewPrice(''); }}

                className="btn-secondary spaced-left"

              >

                Poništi

              </button>

            )}

          </div>

        </form>

      </div>

      <div className="card table-section-card">

        <h2 className="card-title">Lista Kriptovaluta</h2>

        {error && <p className="error-message">{error}</p>}

        <table className="data-table">

          <thead>

            <tr>

              <th className="table-header">Simbol</th>

              <th className="table-header">Naziv</th>

              <th className="table-header">Cena (USD)</th>

              <th className="table-header">Akcije</th>

            </tr>

          </thead>

          <tbody>

            {cryptos.map(crypto => (

              <tr key={crypto.symbol} className="table-row">

                <td className="table-data">{crypto.symbol}</td>

                <td className="table-data">{crypto.name}</td>

                <td className="table-data price-data">${crypto.currentPriceUSD.toFixed(2)}</td> {}

                <td className="table-data actions-data">

                  <button

                    onClick={() => handleEditCrypto(crypto)}

                    className="btn-info spaced-right"

                  >

                    Uredi

                  </button>

                  <button

                    onClick={() => handleDeleteCrypto(crypto.id)}

                    className="btn-danger"

                  >

                    Obriši

                  </button>

                </td>

              </tr>

            ))}

          </tbody>

        </table>

      </div>

    </div>

  );

* **Sve transakcije (**AllTransactionsPage.jsx**):** Prikazuje kompletan spisak svih transakcija koje su se desile u sistemu, dostupno samo administratorima.

**return (**

    <div className="page-wrapper"> { }

      <h1 className="page-title">Sve Transakcije</h1> { }

      <div className="card"> { }

        {error && <p className="text-error">{error}</p>} { }

        {transactions.length === 0 ? (

          <p className="text-muted">Nema transakcija u sistemu.</p>

        ) : (

          <table className="data-table"> { }

            <thead>

              <tr>

                <th className="table-header">ID</th> { }

                <th className="table-header">Pošiljalac</th> { }

                <th className="table-header">Primalac</th> { }

                <th className="table-header">Iznos</th> { }

                <th className="table-header">Valuta</th> { }

                <th className="table-header">Datum</th> { }

                <th className="table-header">Akcije</th> { }

              </tr>

            </thead>

            <tbody>

              {transactions.map(tx => (

                <tr key={tx.id} className="table-row"> { }

                  <td className="table-data table-id">{tx.id}</td> { }

                  <td className="table-data">{getUserNameById(tx.senderId)}</td> { }

                  <td className="table-data">{getUserNameById(tx.receiverId)}</td> { }

                  <td className="table-data">{tx.amount}</td> { }

                  <td className="table-data">{tx.currency}</td> { }

                  <td className="table-data table-date">{new Date(tx.date).toLocaleString()}</td> { }

                  <td className="table-data">

                    <button

                      onClick={() => handleDeleteTransaction(tx.id)}

                      className="btn-danger"

                    >

                      Obriši

                    </button>

                  </td>

                </tr>

              ))}

            </tbody>

          </table>

        )}

      </div>

    </div>

  );

**4. Implementacija funkcionalnosti**

**4.1. Korisničke funkcionalnosti**

**Registracija i prijava korisnika**

* **Registracija korisnika:** Implementirano u komponentama RegisterPage.jsx i LoginPage.jsx, sa korišćenjem AuthContext.js za upravljanje stanjem autentifikacije. Podaci se šalju i proveravaju preko JSON Servera.
* **Prijava korisnika:** Implementirana u komponenti LoginPage.jsx. Funkcionalnosti uključuju:
  + Validacija unosa korisničkog imena i lozinke
  + Provera kredencijala kroz API poziv (GET zahtev)
  + Čuvanje korisničkih podataka u localStorage nakon uspešne prijave
  + Preusmeravanje na dashboard nakon uspešne prijave

**Pregled podataka i interakcija sa njima**

* **Dashboard:** Prikazuje WalletOverview (pregled novčanika i simulaciju cena), PortfolioChart (grafikon raspodele) i TransactionList (lista transakcija).
* **Slanje kriptovaluta:** Omogućava korisniku da pošalje određenu količinu kriptovalute drugom registrovanom korisniku, uz validaciju dovoljnih sredstava i ažuriranje stanja novčanika oba korisnika, kao i kreiranje transakcije u db.json.
* **Simulacija fluktuacije cena:** U komponenti WalletOverview.jsx, useEffect hook simulira promene cena kriptovaluta, dajući dinamičan prikaz vrednosti portfolija. Ovo je specifična funkcionalnost projekta.
* **Vizuelni prikaz portfolija:** PortfolioChart.jsx komponenta koristi react-chartjs-2 za prikaz Pie grafikona raspodele vrednosti kriptovaluta u novčaniku.

**Dodavanje, brisanje i ažuriranje podataka**

* **Slanje kriptovaluta:** Ova funkcionalnost obavlja ažuriranje stanja novčanika (wallets entitet).
* **Ažuriranje korisnika i kriptovaluta:** Ovu funkcionalnost obavljaju administratori.

**4.2. Administratorske funkcionalnosti**

* **Kreiranje, ažuriranje i brisanje podataka:**
  + **Upravljanje korisnicima (**UserManagementPage.jsx**):** Administratori mogu pregledati sve korisnike, ažurirati njihove podatke (uključujući ulogu) i brisati naloge.
  + **Upravljanje kriptovalutama (**CryptoManagementPage.jsx**):** Administratori mogu dodavati nove kriptovalute, ažurirati njihove detalje (npr. cene, nazive) i brisati postojeće kriptovalute.
  + **Pregled transakcija (**AllTransactionsPage.jsx**):** Administratori imaju uvid u sve transakcije u sistemu i mogu ih brisati.
* **Pregled korisničkih aktivnosti ili unosa:** Kroz AllTransactionsPage.jsx i UserManagementPage.jsx, administratori imaju potpuni pregled aktivnosti i unosa u sistem.

**5. Simulacija backend-a**

**5.1. Korišćenje json-servera** Backend funkcionalnosti simulirane su korišćenjem json-server alata. JSON Server omogućava brzo i jednostavno kreiranje REST API mock servera na osnovu običnog JSON fajla (db.json).

* **Postavljanje:** json-server se instalira putem npm-a, db.json fajl se kreira sa željenom strukturom podataka (npr. users, wallets, transactions, cryptos).
* **Pokretanje:** Server se pokreće komandom npm run server (definisanom u package.json), koja nadgleda db.json fajl za promene i automatski generiše REST endpointe (npr. http://localhost:3000/users).

{

  "users": [

    {

      "id": "user1",

      "username": "testuser",

      "password": "password123",

      "role": "user"

    },

    {

      "id": "user2",

      "username": "jovan",

      "password": "lozinka",

      "role": "user"

    },

    {

      "id": "admin1",

      "username": "admin",

      "password": "adminpass",

      "role": "admin"

    }

  ],

  "wallets": [

    {

      "id": "wallet1",

      "userId": "user1",

      "balances": {

        "BTC": 1,

        "ETH": 2.1,

        "XRP": 100

      }

    },

    {

      "id": "wallet2",

      "userId": "user2",

      "balances": {

        "BTC": 1.5,

        "ETH": 5,

        "ADA": 1000,

        "DOT": 50

      }

    }

  ],

  "transactions": [

    {

      "id": "tx1",

      "senderId": "user1",

      "receiverId": "user2",

      "amount": 0.1,

      "currency": "BTC",

      "date": "2023-10-26T10:00:00Z",

      "type": "send"

    },

    {

      "id": "tx2",

      "senderId": "user2",

      "receiverId": "user1",

      "amount": 0.05,

      "currency": "BTC",

      "date": "2023-10-26T11:00:00Z",

      "type": "receive"

    },

    {

      "id": "tx1749661690862",

      "senderId": "user2",

      "receiverId": "user1",

      "amount": 0.5,

      "currency": "BTC",

      "date": "2025-06-11T17:08:10.863Z",

      "type": "send"

    }

  ],

  "cryptos": [

    {

      "id": "crypto1",

      "symbol": "BTC",

      "name": "Bitcoin",

      "currentPriceUSD": 65000

    },

    {

      "id": "crypto2",

      "symbol": "ETH",

      "name": "Ethereum",

      "currentPriceUSD": 3200

    },

    {

      "id": "crypto3",

      "symbol": "XRP",

      "name": "Ripple",

      "currentPriceUSD": 0.5

    },

    {

      "id": "crypto4",

      "symbol": "ADA",

      "name": "Cardano",

      "currentPriceUSD": 0.45

    },

    {

      "id": "crypto5",

      "symbol": "DOT",

      "name": "Polkadot",

      "currentPriceUSD": 7.2

    }

  ]

}

**5.2. Povezivanje frontenda i backend-a** Komunikacija između frontend React aplikacije i simuliranog backend-a ostvarena je putem HTTP zahteva. Korišćena je axios biblioteka za efikasno slanje ovih zahteva.

* **GET zahtevi:** Koriste se za dohvatanje podataka, npr. lista korisnika (http://localhost:3000/users), transakcija, novčanika, ili specifičnih entiteta putem ID-a ili filtera (npr. username, password za prijavu).

const cryptoRes = await axios.get('http://localhost:5000/cryptos');

                setAllCryptos(cryptoRes.data);

* **POST zahtevi:** Služe za kreiranje novih entiteta, npr. registracija novog korisnika ili kreiranje nove transakcije.
* **PUT/PATCH zahtevi:** Koriste se za ažuriranje postojećih entiteta (npr. ažuriranje balansa novčanika, izmena podataka o korisniku/kriptovaluti).
* **DELETE zahtevi:** Omogućavaju brisanje entiteta iz baze, npr. brisanje korisnika, transakcije ili kriptovalute.

**5.3. Alternativne backend tehnologije** Iako je JSON Server bio dovoljan za potrebe ovog projekta zbog svoje jednostavnosti i brzog postavljanja, u produkcijskom okruženju ili za složenije funkcionalnosti, mogle bi se koristiti sledeće backend tehnologije:

* Express.js (Node.js)
* Firebase (Backend-as-a-Service)
* Supabase (Open-source alternativa Firebase-u)
* Django REST Framework (Python)

**6. Zaključak**

**6.1. Pregled rezultata** Projekat "Crypto Wallet App" uspešno je implementiran kao kompletna web aplikacija, u potpunosti ispunjavajući postavljene kriterijume. Ključni postignuti rezultati uključuju:

* Potpuno funkcionalan korisnički interfejs razvijen u Reactu, stilizovan čistim CSS-om.
* Robusnu funkcionalnost autentikacije i autorizacije sa odvojenim ulogama korisnika i administratora.
* Implementaciju svih CRUD operacija (kreiranje, čitanje, ažuriranje, brisanje) za centralne entitete sistema (korisnici, novčanici, transakcije, kriptovalute).
* Razvoj sveobuhvatnog administrativnog panela za upravljanje podacima.
* Efikasnu simulaciju backend-a korišćenjem json-server alata.
* Uspešnu integraciju eksterne biblioteke react-chartjs-2 za vizuelizaciju podataka (grafikone).
* Implementaciju posebne funkcionalnosti simulacije fluktuacije cena kriptovaluta, koja doprinosi realističnijem prikazu portfolija.

**6.2. Izazovi i prepreke** Tokom razvoja projekta, suočeni smo sa nekoliko ključnih izazova:

* Implementacija složene logike za čuvanje i proveru korisničkih sesija i autorizacije na strani klijenta.
* Pravilno rukovanje greškama prilikom asinhronih API poziva i validacije unosa korisnika.
* Poseban izazov predstavljalo je rešavanje problema sa konfiguracijom CSS frameworka (Tailwind CSS) i sukobima zavisnosti (npm ERESOLVE greške), što je zahtevalo temeljno otklanjanje grešaka i, na kraju, strateški prelazak na čisti CSS radi stabilnosti projekta.

**6.3. Mogućnosti za buduća unapređenja aplikacije** Budući razvoj aplikacije može uključivati:

* Proširenje funkcionalnosti novčanika: Uvođenje opcija za simuliranu kupovinu/prodaju kriptovaluta, detaljnije finansijske izveštaje, i notifikacije o promenama cena.
* Migracija na pravi backend: Zamena JSON Servera robusnim backend-om (npr. Node.js sa MongoDB/PostgreSQL) za podršku produkcionim zahtevima, skalabilnosti i poboljšanu sigurnost.
* Integracija sa realnim API-jima: Povezivanje sa eksternim servisima (poput CoinGecko API-ja) za dohvatanje stvarnih, živih podataka o cenama i tržišnim informacijama.
* Dodavanje podrške za više jezika.

**7. Spisak literature**

* Corbet, J. (2019). "The Deadline I/O Scheduler." <https://lwn.net/Articles/143323/>
* University of Illinois, Chicago, (2011). "IO Scheduling in Linux." <https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/13_IOSystems.html>
* Microsoft Developer Network (2021). "Windows I/O Processing." <https://docs.microsoft.com/en-us/windows/win32/fileio/i-o-concepts>
* Axboe, J. (2019). "Efficient IO with io\_uring." <https://kernel.dk/io_uring.pdf>