МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ   
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-40 05 01 «Информационные системы и технологии»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА**

по дисциплине «Скриптовые языки программирования»

Тема «Web-сайт про Финансы»

**Исполнитель**

студент 2 курса 8 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Яцкевич В.И.

подпись, дата

**Руководитель**

Доцент каф. ИСиТ\_\_\_\_\_\_\_\_\_\_ Жиляк Н. А.

должность, ученая степень, ученое звание подпись, дата

Допущен(а) к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата, подпись

Курсовой проект защищен с оценкой

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Жиляк Н. А.

подпись дата инициалы и фамилия

Оглавление

[Введение 5](#_Toc121705776)

[1 Постановка задачи 6](#_Toc121705777)

[2 Проектирование веб-сайта 7](#_Toc121705778)

[2.1 Обзор аналогичных решений 7](#_Toc121705779)

[2.2 Выбор средств реализации программного продукта 9](#_Toc121705780)

[2.3 Проектирование страниц 10](#_Toc121705781)

[2.4 Роли пользователей 11](#_Toc121705782)

[2.5 Выбор стилевого оформления 11](#_Toc121705783)

[2.6 Разработка пользовательских элементов 12](#_Toc121705784)

[2.7 Выводы 14](#_Toc121705785)

[3 Реализация структуры веб-сайта 15](#_Toc121705786)

[3.1 Структура JSX-документа 15](#_Toc121705787)

[3.2 Структура Redux-Toolkit 18](#_Toc121705788)

[3.3 Структура axios для http запросов на сервер 20](#_Toc121705789)

[3.4 Структура серверной части 20](#_Toc121705790)

[4 Тестирование веб-сайта 22](#_Toc121705792)

[4.1 Адаптивный дизайн веб-сайта 22](#_Toc121705793)

[Заключение 23](#_Toc121705794)

# Введение

На сегодняшний день подавляющее число финансовых операций происходит в интернете. Из-за этого сложно уследить за всеми тратами и пополнениями. Для этого существует множество интернет-сервисов, предоставляющие пользователям различные услуги по систематизации доходов и расходов, соответствующие различным категориям и удовлетворяющие запросы потребителя.

Целью курсового проектирования является разработка веб-сайта учета финансовых операций с использованием технологий frontend-программирования и применением полученных в этой области знаний.

Тема является актуальной и востребованной, имеется множество сайтов аналогов, предоставляющих большой выбор услуг. И поэтому сложно выбрать одно подходящее решение для всего.

Для разработки будут использоваться технологии: JS, React, MongoDB, а также HTML/CSS3.

# 1 Постановка задачи

Основные задачи для web-сайта:

* Разработать регистрацию и авторизацию пользователей;
* Разработать возможность добавления и удаления транзакций;
* Разработать Кроссплатформенность для web-сайта;
* Разработать структуру web-сайта;
* Разработать аналитику транзакций;

# 2 Проектирование веб-сайта

## 2.1 Обзор аналогичных решений

В глобальной сети Интернет существует множество аналогов разрабатываемого веб-сайта, как белорусских, так и зарубежных. Каждый из них имеет свои преимущества и недостатки, которые были учтены при разработке.

**fincalculator.ru**

На сайте реализован обширный каталог всевозможных вычислений, с фильтрацией по многим параметрам. Также реализована статистика, которая отражает все транзакции во время пользования. (рис. 1)

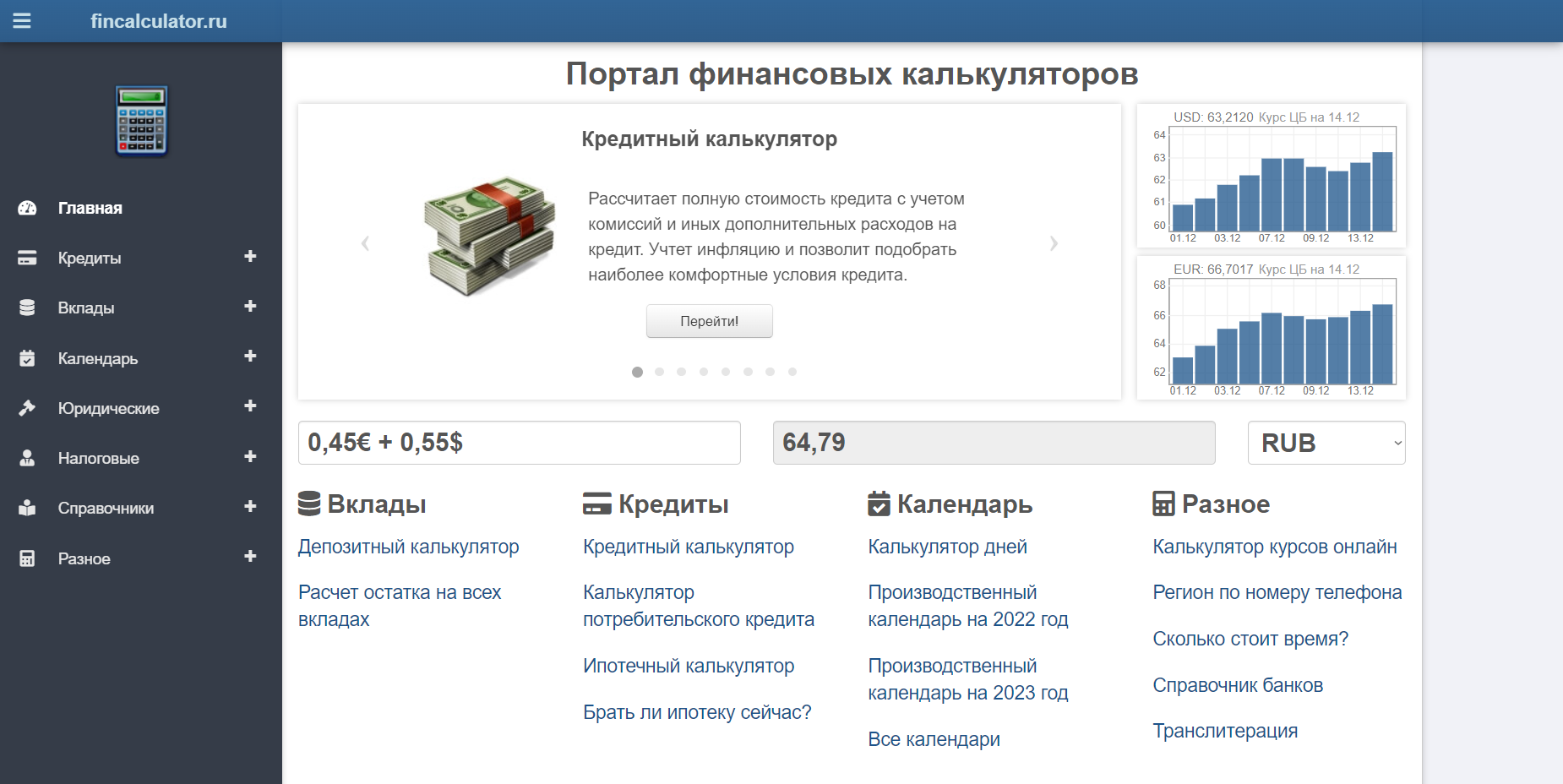


Рисунок 1

**calc.by**

На сайте реализована навигация по каталогу с помощью различных категорий(Таможенные калькуляторы, Конвертеры, Калькуляторы для строительства и ремонта). Также реализовано добавление пользовательского калькулятора (рис. 2)

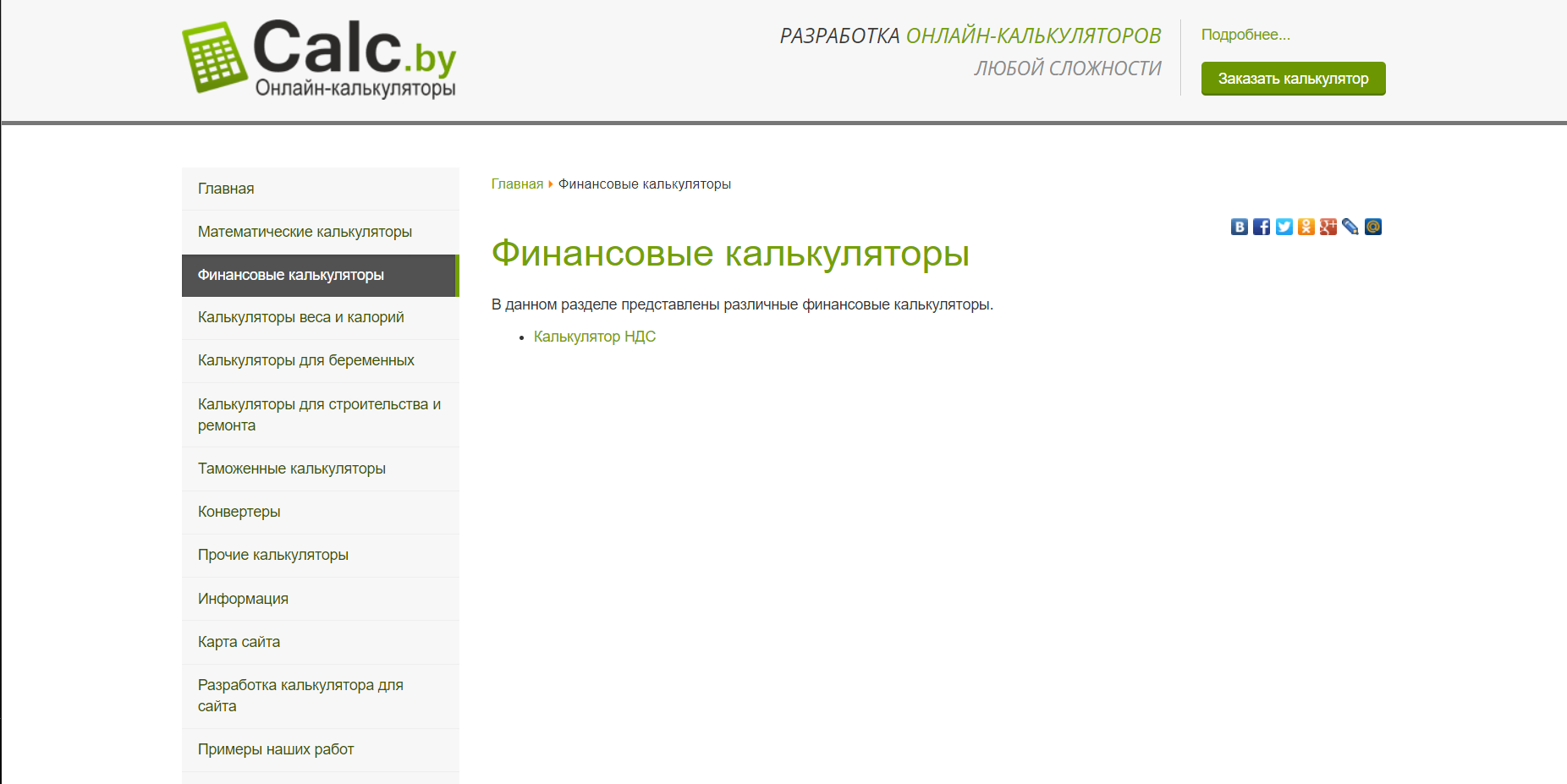


Рисунок 2

**fincult.info**

На данном веб-сайте реализовано самое удачное из рассмотренных выше дизайнерское решение. Присутствует широкий выбор предлагаемых операций и их периодичность (Ежемесячно, ежеквартально, ежедневно). (рис. 3)

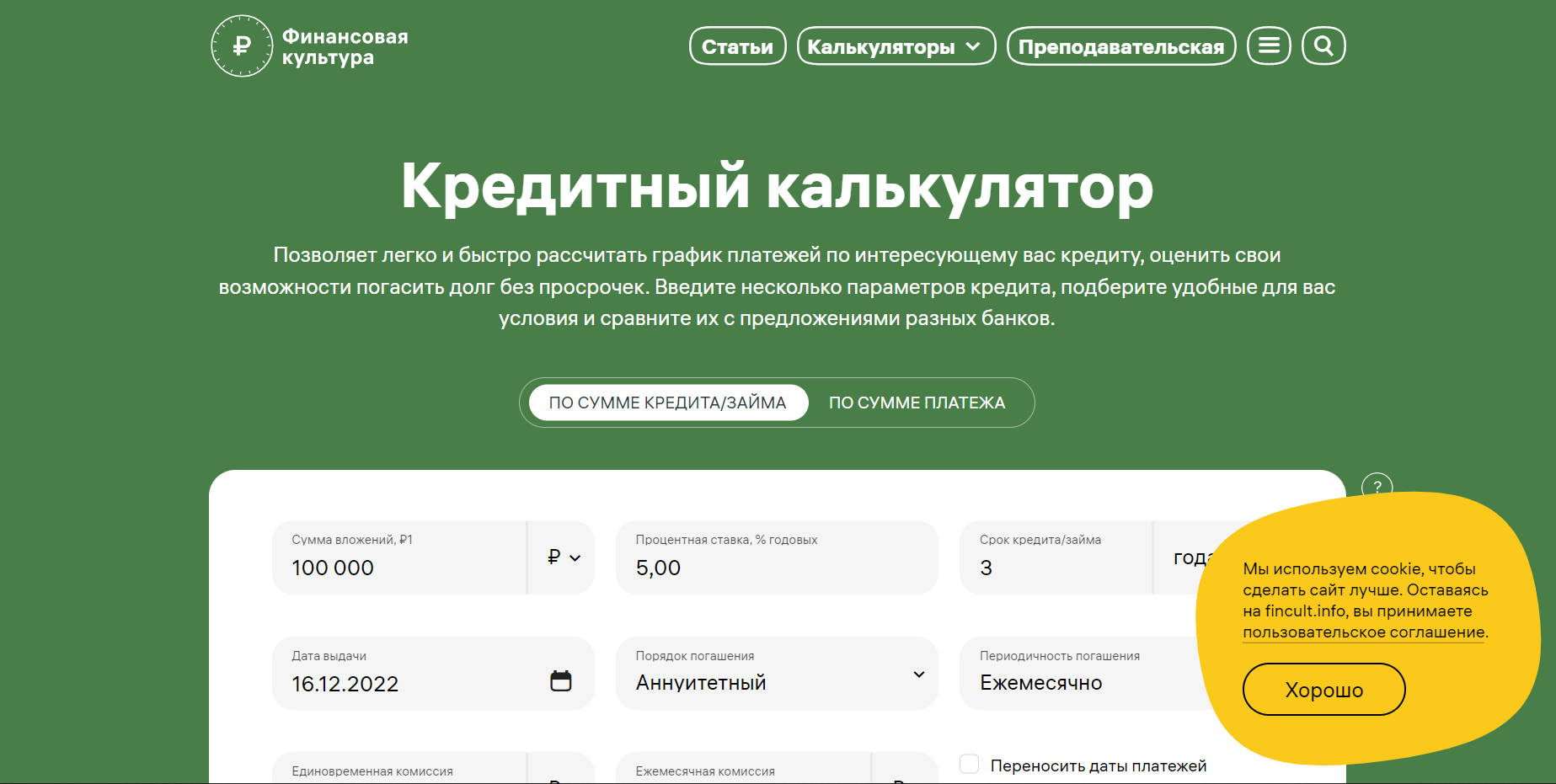


Рисунок 3

**100calc.ru**

На главной странице сайта расположена навигационная панель типа для выбора нужных калькулятора. Так же присутствует навигация по основным категориям (Строительные, финансовые, бытовы). (рис. 4)

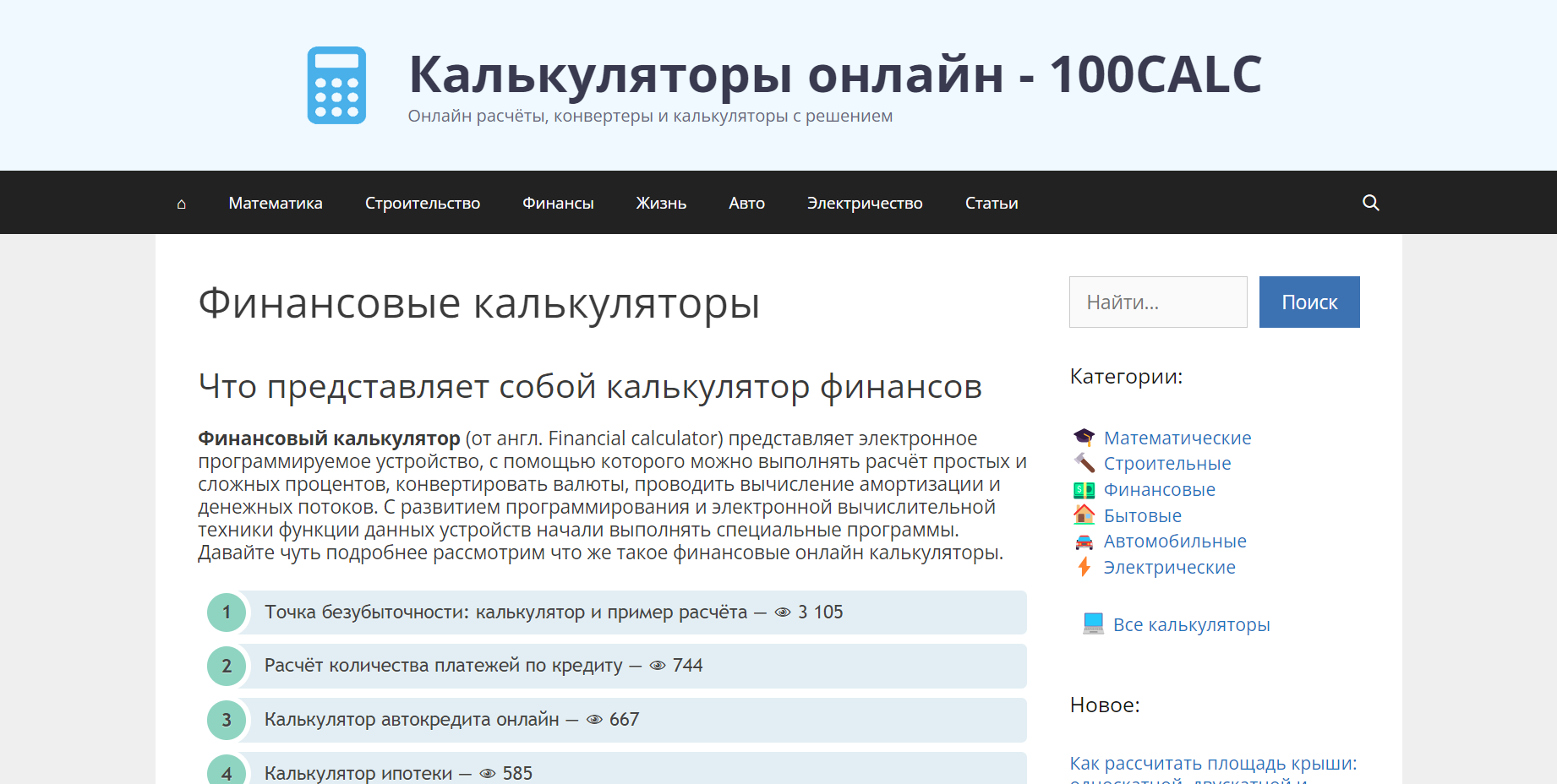


Рисунок 4

## 2.2 Выбор средств реализации программного продукта

Главной задачей сайта является предоставление возможности добавления тарт и пополнений. Данный сай будет предназначен для людей, которые хотят контролировать свой финансовый поток, отслеживать каждый шаг, а также видеть наглядную и понятную статистику. Сайт будет адаптирован под мобильные устройства, а также будет кроссплатформенным.

Сайт будет разработан в текстовом редакторе Visual Studio Code. Страницы сайта могут просматриваться в таких браузерах как Microsoft Edge, Safari, Opera, Mozilla Firefox, Google Chrome.

Visual Studio Code — редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений.

Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса и средства для рефакторинга.

Для реализации будущего сайта будет использована следующая архитектура:

1. Front-end

2. Back-end

3. База данных

Front-end — это часть сайта, с которой взаимодействует пользователь. В данном случае это будет внешний вид сайта, его дизайн, а также навигация по сайту. В данном случае будет использованы следующие технологии: React JS, CSS3, Redux-Toolkit, Axios, React Router.

JavaScript – это язык программирования, который используется для создания динамических веб-страниц.

React JS — это библиотека, которая использует компонентный подход, который позволяет разрабатывать сложные интерфейсы из простых компонентов. С помощью React разработчики создают веб-приложения, которые изменяют отображение без перезагрузки страницы. Благодаря этому приложения быстро реагируют на действия пользователя, например, заполнение форм, применение фильтров, добавление товаров в корзину и так далее. React применяют для отрисовки компонентов пользовательского интерфейса. Также библиотека может полностью управлять фронтендом. В этом случае React используют с библиотеками для управления состоянием и роутинга, например, Redux и React Router.

Redux — это библиотека для управления состоянием приложения.

Redux Toolkit — это набор инструментов для упрощения работы с Redux.

Axios — это библиотека для работы с HTTP-запросами. Axios позволяет создавать запросы к API.

Back-end - это часть приложения, которая отвечает за обработку данных, ко-торые приходят от клиента и возвращают данные клиенту. Также Back-end отвечает за работу с базой данных, которая хранит данные приложения. Для создания Back-end будет использованы следующие технологии: Node.js, Express.

Node.js — это JavaScript-окружение, которое позволяет запускать JavaScript-код на сервере.

Express — это фреймворк для Node.js, который позволяет создавать сервер, который будет обрабатывать запросы от клиента и возвращать данные клиенту.

База данных - это совокупность взаимосвязанных данных. Для создания базы данных будет использована PostgreSQL.

MongoDB — это объектно-реляционная система управления базами данных (СУБД), которая позволяет хранить данные в виде таблиц. MongoBD позволяет создавать базы данных, которые будут хранить данные приложения.

## 2.3 Проектирование страниц

В проекте будет использована Flex верстка, так как она позволяет создавать адаптивные веб-страницы, которые будут отображаться корректно на всех устройствах. CSS Flexbox поддерживается всеми современными браузерами. С помощью этой технологии можно очень просто и гибко расставить элементы в контейнере, распределить доступное пространство между ними, и выровнять их тем или иным способом даже если они не имеют конкретных размеров.

Структура веб-сайта будет состоять из данных страниц:

- Главная страница

- Страница с историей транзакций и текущем балансам

- Личный страница пользователя

- Календарь с отмеченными событиями

- Статистика

На главной странице расположено краткие рекомендации по финансовой грамотности, являющиеся обязательными к прочтению и поясняющие работу основных страниц сайта. Так же главная страница содержит пользовательскую статистику, показывающую количество пользователей, которые положительно откликнулись о данном сайте.

С главной страницы мы можем попасть на другие страницы с помощью выпадающего навигационного меню, находящегося с левой стороны.

Следующей страницей, на которую попадает пользователь является страница личного учета финансового положения пользователя. На ней пользователь может добавить финансовую транзакцию (как доход, так и расход) по соответствующей категории. А также на станице присутствует история проведенных операций, показывающая все транзакции. Добавление транзакции осуществляется по двум категориям: доход и расход, далее выбирается ее категория (зарплата, к примеру), указывается сумма, пояснение к ней и последним шагом является ее отправка. Вернувшись в начало страницы данную транзакцию мы можем наблюдать в истории.

Следующая страница – пользовательский календарь, в котором пользователь может добавить события, привязанные к конкретной дате и времени. А также распланировать свой график далеко вперед или посмотреть события, имеющие большую давность.

Следующая из основных страниц является страница пользовательских задач. Она представляет собой таблицу с тремя основными зонами: цель, в процессе решения, в процессе тестирования и выполнено. Задачи размещаются в соответствующих зонах и по мере выполнения переносятся в следующую зону.

## 2.4 Роли пользователей

Как только пользователь заходит на веб-сайт, его ролью является Пользователем. Это означает, что он может производит транзакции, видеть историю и аналитику, но не может видеть личную информацию остальных пользователей.

После авторизации Пользователь может совершать транзакции и после выхода его информация будет храниться в базе данных. Никто, кроме администратора не имеет доступ к этим данным. После входа в свой аккаунт, вся информация будет отображаться на сайте.

Также есть роль администратора, он имеет доступ к всем личным данным пользователе и может сам их редактировать в базе данных при необходимости.

## 2.5 Выбор стилевого оформления

Веб-сайт будет реализован в минималистическом стилевом оформлении. Основными цветами будут чёрный, белый и серый. Нейтральные цвета в интерфейсе веб-сайта позволят подчеркнуть на их фоне яркие элементы, отражающие суммы транзакций к примеру или события в календаре.

В качестве основного шрифта для сайта был выбран популярный шрифт Open Sans в вариациях Regular и Bold.

Данный шрифт лаконично вписывается в общий вид веб-страницы. Отсутствие засечек подчеркивает минималистический стиль оформления веб-страницы.

## 2.6 Разработка пользовательских элементов

**Разработка логотипа**

Логотип для веб-сайта был разработан в формате SVG. Для его создания был использован редактор векторных изображений Adobe Illustrator.

Логотип был разработан на основе шрифта Harlow Solid Italic. Отдельные символы были преобразованы в вектора и изменены для создания логотипа. (рис.5)



Рисунок 5

Иконка для сайта также была создана в Adobe Illustrator на основе логотипа. (рис. 6)

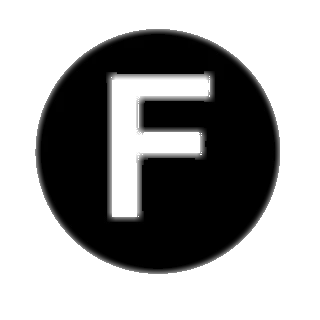


Рисунок 6

В левой части сайта будут использоваться кнопки для навигации по страницам. (рис. 7)

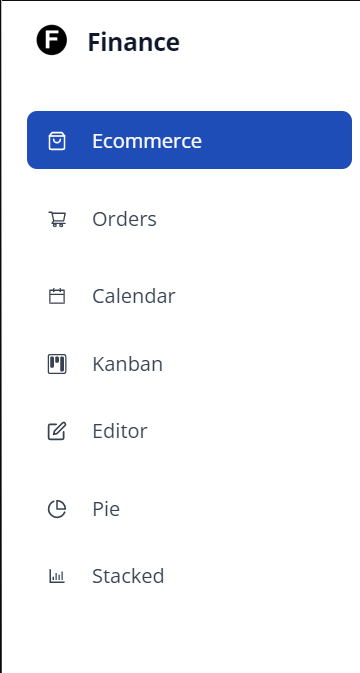


Рисунок 7

В компоненте подсчета баланса будут реализована карточка с имеющимся балансом и таблицей с историей. (рис. 8)

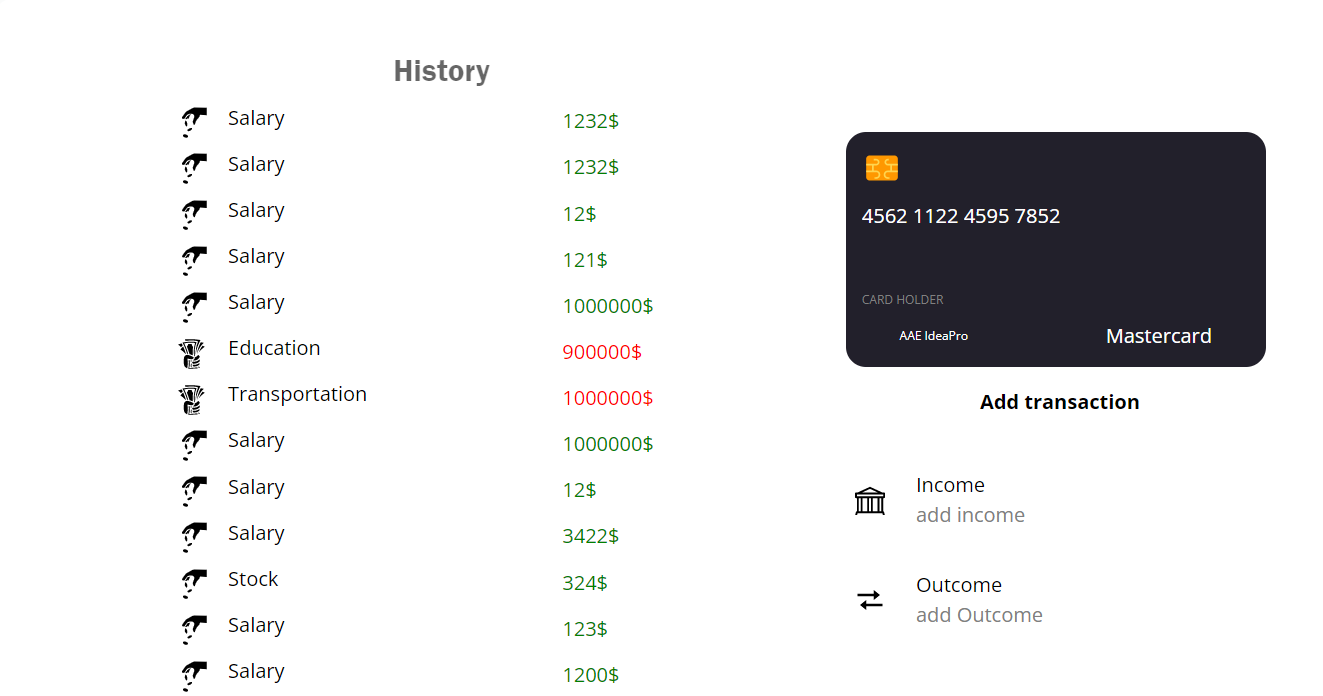


Рисунок 8

Оформление формы транзакции (рис. 9)

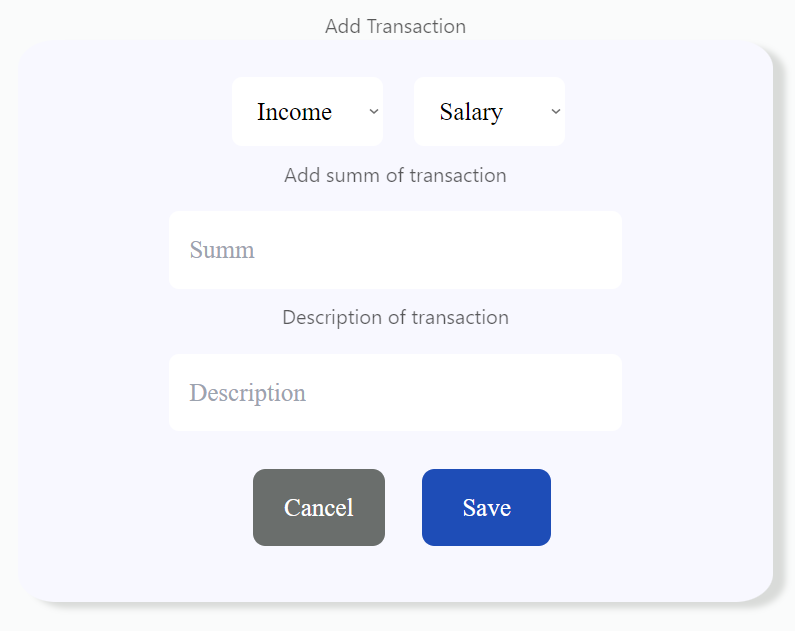


Рисунок 9

## 2.7 Выводы

В заключении этого раздела подведем: были разработан макет и прототип сайта, который будет использоваться для дальнейшего улучшения сайта. Были выбраны основные цветовые решения и шрифты, которые будут использоваться. Были разработаны некоторые динамические эффекты и анимации, которые позволят придать сайту интерактивность и сделают его более привлекательным для пользователя. Самое главное, чтобы сайт был удобным, понятным и простым для пользователя. А также решал все поставленные для него задачи.

# 3 Реализация структуры веб-сайта

## 3.1 Структура JSX-документа

React 18 использует в основном функциональные компоненты для отрисовки веб-страницы. С помощью JSX эти компоненты принимают вид подобный структуре HTML-документа.

Например, структура компонента App приведена в листинге 1.

|  |
| --- |
| import React, { useEffect } from 'react';  import { BrowserRouter, Routes, Route } from 'react-router-dom';  import { FiSettings } from 'react-icons/fi';  import { TooltipComponent } from '@syncfusion/ej2-react-popups';  import { useDispatch, useSelector } from "react-redux";  import { fetchAuthMe } from "../src/redux/slices/auth";  import { selectIsAuth} from '../src/redux/slices/auth'  import { Navbar, Footer, Sidebar, ThemeSettings } from './components';  import { Ecommerce, Orders, Calendar, Employees, Stacked, Pyramid, Customers, Kanban, Line, Area, Bar, Pie, Financial, ColorPicker, ColorMapping, Editor } from './pages';  import './App.css';  import AddNew from '../src/components/PaymentComponents/form/AddNew';  import { useStateContext } from './contexts/ContextProvider';  import Login from './components/Login';  import Registration from './components/Registration';  const App = () => {  const { setCurrentColor, setCurrentMode, currentMode, activeMenu, currentColor, themeSettings, setThemeSettings } = useStateContext();  const dispatch = useDispatch();  const isAuth = useSelector(selectIsAuth)  useEffect(()=>{  dispatch(fetchAuthMe())  } , [])  useEffect(() => {  const currentThemeColor = localStorage.getItem('colorMode');  const currentThemeMode = localStorage.getItem('themeMode');  if (currentThemeColor && currentThemeMode) {  setCurrentColor(currentThemeColor);  setCurrentMode(currentThemeMode);  }  }, []);  return (  <div className={currentMode === 'Dark' ? 'dark' : ''}>  <BrowserRouter>  <div className="flex relative dark:bg-main-dark-bg">  <div className="fixed right-4 bottom-4" style={{ zIndex: '1000' }}>  <TooltipComponent  content="Settings"  position="Top"  >  <button  type="button"  onClick={() => setThemeSettings(true)}  style={{ background: currentColor, borderRadius: '50%' }}  className="text-3xl text-white p-3 hover:drop-shadow-xl hover:bg-light-gray"  >  <FiSettings />  </button>  </TooltipComponent>  </div>  {activeMenu ? (  <div className="w-72 fixed sidebar dark:bg-secondary-dark-bg bg-white ">  <Sidebar />  </div>  ) : (  <div className="w-0 dark:bg-secondary-dark-bg">  <Sidebar />  </div>  )}  <div  className={  activeMenu  ? 'dark:bg-main-dark-bg bg-main-bg min-h-screen md:ml-72 w-full '  : 'bg-main-bg dark:bg-main-dark-bg w-full min-h-screen flex-2 '  }  >  <div className="fixed md:static bg-main-bg dark:bg-main-dark-bg navbar w-full ">  <Navbar />  </div>  <div>  {themeSettings && (<ThemeSettings />)}  <Routes>  <Route path="/Login" element={(<Registration/>)}/>  {/\* dashboard \*/}  <Route path="/" element={(<Ecommerce />)} />  <Route path="/ecommerce" element={(<Ecommerce />)} />  {/\* pages \*/}  <Route path="/orders" element={<Orders />} />      <Route path = '/add' element = {<AddNew />} />  {/\* apps \*/}  <Route path="/kanban" element={<Kanban />} />  <Route path="/editor" element={<Editor />} />  <Route path="/calendar" element={<Calendar />} />  {/\* charts \*/}  <Route path="/line" element={<Line />} />  <Route path="/area" element={<Area />} />  <Route path="/bar" element={<Bar />} />  <Route path="/pie" element={<Pie />} />  <Route path="/financial" element={<Financial />} />  <Route path="/color-mapping" element={<ColorMapping />} />  <Route path="/pyramid" element={<Pyramid />} />  <Route path="/stacked" element={<Stacked />} />  </Routes>  </div>  <Footer />  </div>  </div>  </BrowserRouter>  </div>  );  };  export default App; |

*Листинг 1*

В нём использованы импортированные из react-router-dom компоненты BrowserRouter, Routes, Route, необходимые для навигации на веб-сайте

## 3.2 Структура Redux-Toolkit

Redux-Toolkit был использован для работы с состояниями и их изменением, показан на листинге 2.

|  |
| --- |
| import {createSlice, createAsyncThunk} from '@reduxjs/toolkit'  import axios from '../../axios'  export const fetchUserData = createAsyncThunk('auth/fetchUserData', async (params)=>{  const {data} = await axios.post('/auth/login', params);//подзапрос  return data;  })  export const fetchRegister = createAsyncThunk('auth/fetchRegister', async (params)=>{  const data = await axios.post('/auth/register', params)  return data  })  export const fetchAuthMe = createAsyncThunk('auth/fetchAuthMe', async ()=>{  const data = await axios.get('/auth/me')  return data  })  const initialState = {  data: null,  status: 'loading',  }  const authSlice = createSlice({  name:'auth',  initialState,  reducers: {  logout:(state)=>{  state.data = null }  },  extraReducers: {  [fetchUserData.pending]: (state)=>{//загружается  state.status = 'loading';  state.data = null;  },  [fetchUserData.fulfilled]: (state, action)=>{  state.status = 'loaded'  state.data = action.payload;  },  [fetchUserData.rejected]:(state)=>{  state.status = 'error';  state.data = null;  },  [fetchAuthMe.pending]: (state)=>{  state.status = 'loading';  state.data = null;  },  [fetchAuthMe.fulfilled]: (state, action)=>{  state.status = 'loaded'  state.data = action.payload;  },  [fetchRegister.pending]: (state)=>{  state.status = 'loading';  state.data = null;  },  [fetchRegister.fulfilled]: (state, action)=>{  state.status = 'loaded'  state.data = action.payload;  },  [fetchRegister.rejected]:(state)=>{  state.status = 'error';  state.data = null;  }}  })  export const selectIsAuth = state => Boolean(state.auth.data);  export const authReducer = authSlice.reducer;  export const { logout} = authSlice.actions |

*Листинг 2*

## 3.3 Структура axios для http запросов на сервер показан на листинге 3.

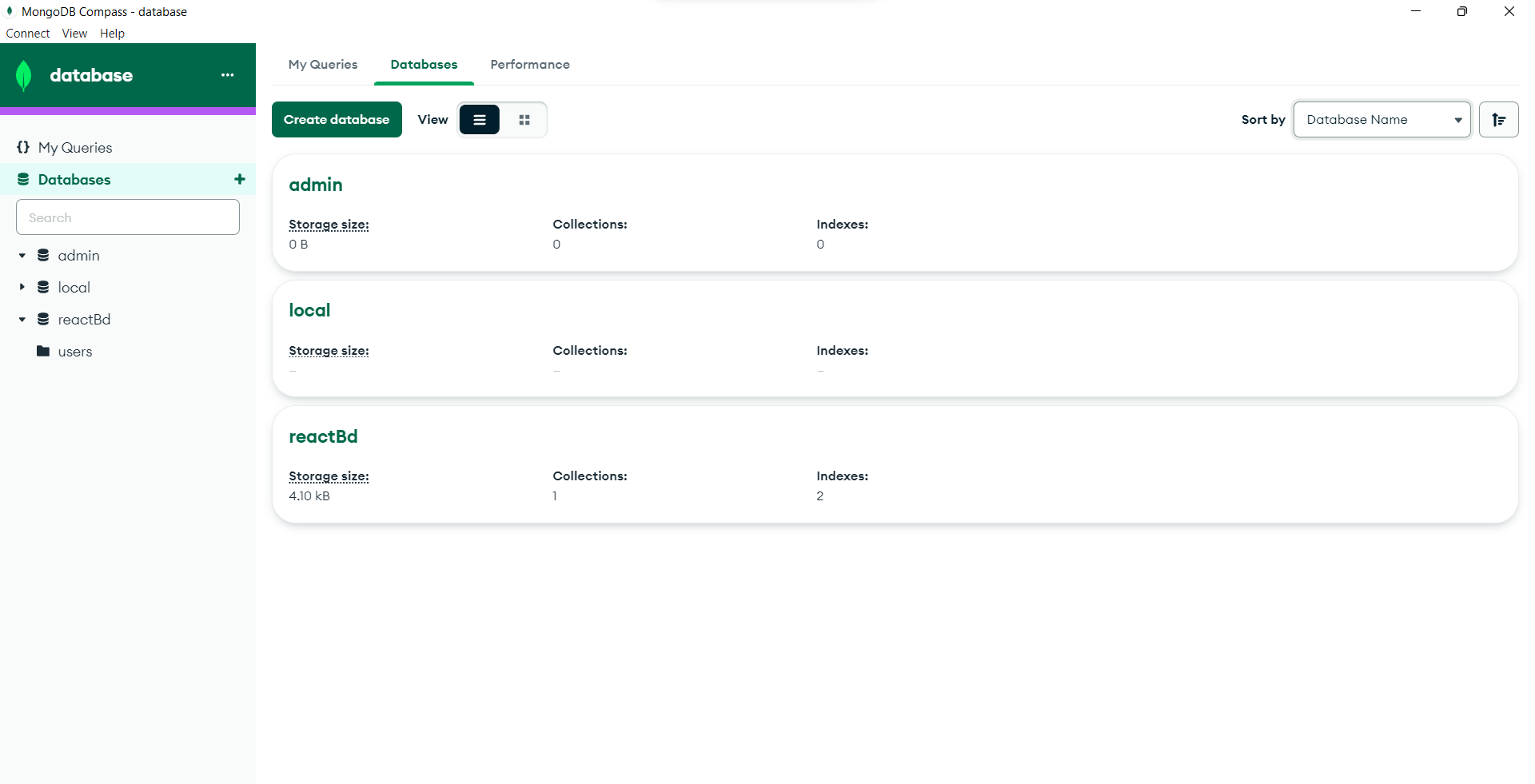
|  |
| --- |
| import axios from 'axios'  const instance = axios.create({  baseURL:'http://localhost:4445'  })  instance.interceptors.request.use((config)=>{  config.headers.Authorization = window.localStorage.getItem('token')  return config  })  export default instance |

*Листинг 3*

## 3.4 Структура серверной части

Серверная часть реализована с помощью фреймворка express для nodeJS.

База данных была реализована с помощью MongoDB.



# 4 Тестирование веб-сайта

## 4.1 Адаптивный дизайн веб-сайта

Адаптивность на веб-сайте была достигнута благодаря flex-вёрстке и медиазапросам. Она позволила добиться того, чтобы компоненты веб-сайта подстраивались под размеры окна. (рис. 12)

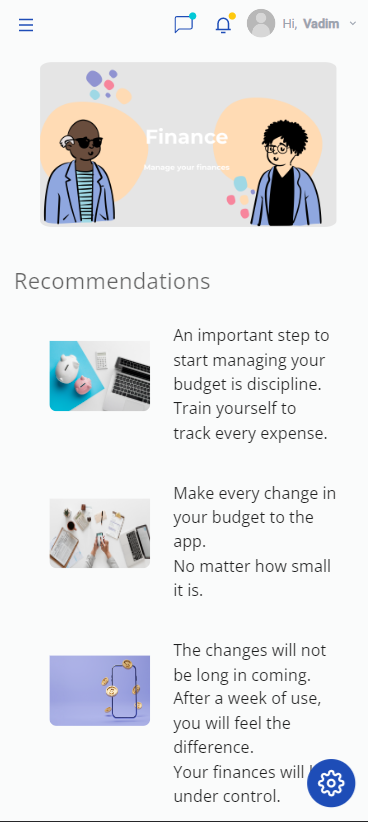


Рисунок 12

# Заключение

Целью на курсовое проектирование было создание веб-сайта для отслеживания финансовых потоков пользователей, аналитика и статистика. Он был реализован с использованием технологий: Express, React, JS, MongoDB и других технологий frontend-программирования.

Данный проект представляет собой реализацию следующего функционала:

- Форма добавления транзакции

- История транзакций

- Спроектирована база данных

Средой для разработки был выбран Visual Studio Code, который послужил отличным решением для разработки веб-сайта.

После разработки последовал этап тестирования, в ходе которого был разработан адаптив для веб-сайта.

Результатом курсового проекта стал готовый веб-сайт, который не уступает своим аналогам.

Таким образом, из выше сказанного можно сделать вывод, что в курсовом проекте раскрываются все теоритические и практические аспекты темы, поставленные в цели и задачах проекта.