

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ
по дисциплине «Введение в нереляционные базы данных»
Тема: Статистика покупок пользователями товаров,
находящихся в разных категориях

Студент гр. 6381	_____	Кухарев М.А.
Студент гр. 6382	_____	Глазков Д.М.
Студент гр. 6383	_____	Клименко К.В.
Преподаватель	_____	Заславский М.М.

ЗАДАНИЕ

Студенты

Кухарев М.А.

Глазков Д.М.

Клименко К.В.

Группы 6381,6382,6383

Тема проекта: Статистика покупок пользователями товаров, находящихся в разных категориях.

Исходные данные:

Разработать веб-приложение для анализа и статистики финансовых потоков с использованием нереляционной базы данных.

Содержание пояснительной записки:

«Введение», «Качественные требования к решению», «Сценарии использования», «Модель данных», «Разработанное приложение», «Выводы», «Приложения», «Литература».

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 27.09.2019

Дата сдачи реферата: 20.12.2019

Дата защиты реферата: 20.12.2019

Студент

Кухарев М.А.

Студент

Глазков Д.М.

Студент

Клименко К.

Преподаватель

Заславский М.М.

АННОТАЦИЯ

В индивидуальном домашнем задании изучаются методы создания приложений на базе технологий Firebase, Node.js, Express.js, React, с использованием нереляционной базы данных. В качестве проекта была выбрана сфера финансов, а именно статистика, анализ и менеджмент личными финансами. Решается проблема распределения денежного потока между сферами жизни и мониторинга изменений потребностей в каждой сфере. В рамках курса “Введение в нереляционные БД” рассматриваются основные операции: хранение, анализ, статистика, представление, импорт, экспорт.

SUMMARY

In an individual homework, methods for creating applications based on the technologies of Firebase, Node.js, Express.js, React, using non-relational databases are studied. As a project, the sphere of finance was chosen, namely statistics, analysis and personal finance management. The problem of distributing cash flow between life spheres and monitoring changes in needs in each sphere is being solved. The course “Introduction to Non-Relational Databases” covers the basic operations: storage, analysis, statistics, presentation, import, export.

Содержание

1.	Введение.....	5
1.1.	Актуальность решаемой проблемы.....	5
1.2.	Постановка задачи.....	5
1.3.	Предлагаемое решение	5
2.	Качественные требования к решению	5
2.1.	Текущие.....	5
2.2.	Перспективные	5
3.	Сценарии использования.....	6
3.1.	Макет UI.....	6
3.2	Сценарии использования для задач:	9
4.	Модель данных	12
4.1	Нереляционная модель данных	12
4.1.1.	Графическое представление модели данных.....	12
4.1.2.	Описание назначений коллекций, типов данных и сущностей.	12
4.1.3.	Оценка удельного объема информации, хранимой в модели	13
4.1.4.	Запросы к модели, с помощью которых реализуются сценарии использования.	14
4.2.	Аналог модели данных для SQL СУБД – графическое представление данных и сравнение с моделью данных для NoSQL БД.....	14
4.2.1.	Графическое представление модели данных.....	14
4.2.2.	Описание назначений коллекций, типов данных и сущностей.	14
4.2.3.	Оценка удельного объема информации, хранимой в модели	15
4.2.4.	Запросы к модели, с помощью которых реализуются сценарии использования	16
4.3.	Вывод:	16
5.	Разработанное приложение	16
5.1.	Краткое описание	16
5.2.	Схема экранов приложения.....	17
5.3.	Использованные технологии.....	18
5.4.	Ссылки на приложение.....	18
6.	Выводы	18
6.1.	Достигнутые результаты	18
6.2.	Недостатки и пути для улучшения полученного решения	18
6.3.	Будущее развитие решения	19
7.	Приложения	19
7.1.	Документация по сборке и развертыванию приложения	19
7.2.	Снимки экрана приложения	20

1. Введение

1.1. Актуальность решаемой проблемы.

Решаемая проблема – мониторинг денежного потока человека с целью его последующего расширения. Каждого человека волнуют вопросы, связанные с личными финансами, с пониманием куда и в каком количестве уходят деньги. Данная проблема будет оставаться актуальной, до тех пор, пока существуют денежные отношения между людьми.

1.2. Постановка задачи

Необходимо разработать web-приложение, позволяющее отслеживать изменение потребностей человека.

Основные функции:

- Статистика денежного потока
- Импорт и экспорт расходов
- Транзакции
- Импорт и экспорт пользователей

1.3. Предлагаемое решение

Предлагается разработать продукт с использованием современных технологий.

2. Качественные требования к решению

2.1. Текущие

Системные требования:

- Удобный и динамичный интерфейс с максимально допустимой задержкой (2 секунды)
- Должно предоставляться публичное API с целью создания мобильного приложения на основе (Ionic Framework) в будущем
- Приложение должно иметь возможность отображать статистику по пользователю за определенный период.
- Должны быть карточки пользователей и транзакций.
- Должна быть возможность просмотра статистики всех пользователей.
- Также необходимо предусмотреть импорт и экспорт данных по финансам и пользователям.

2.2. Перспективные

При росте популярности приложения система должна отвечать следующим требованиям:

1. Выдерживать нагрузку 10000 человек;
2. Нагрузка на application server и базу данных должна балансироваться;
3. Серверная часть должна быть stateless;

3. Сценарии использования

3.1. Макет UI

Список всех пользователей(Рис.1).

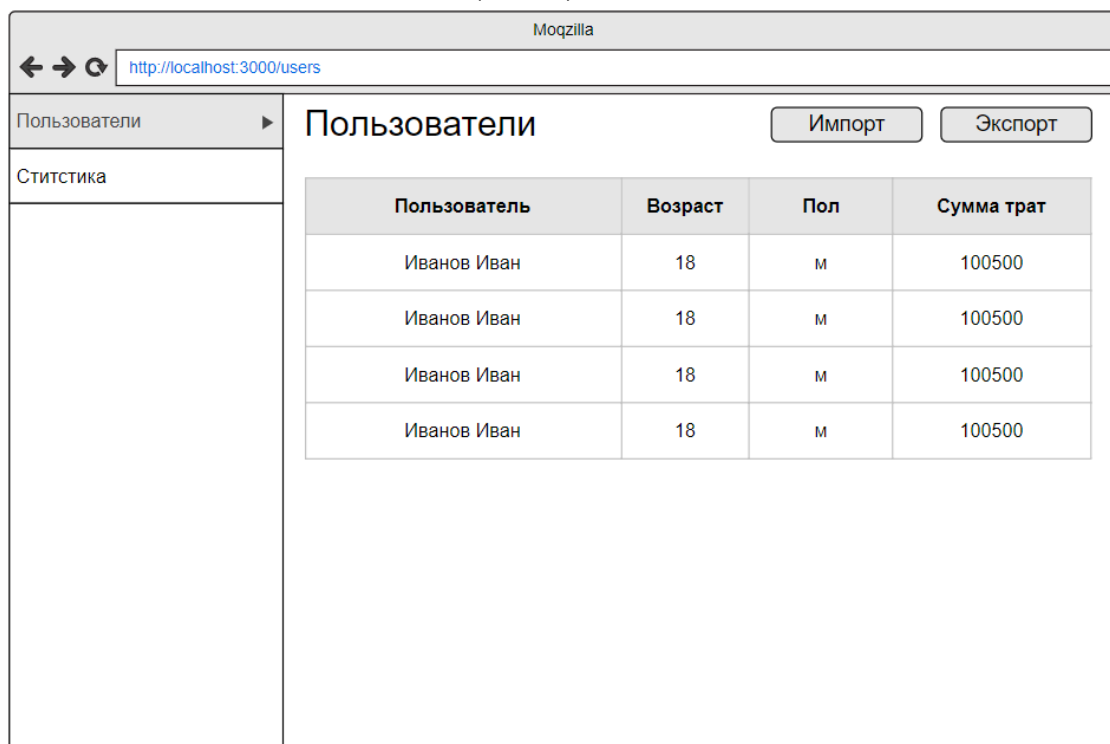


Рисунок 1. Список всех пользователей

Профиль пользователя(Рис.2)

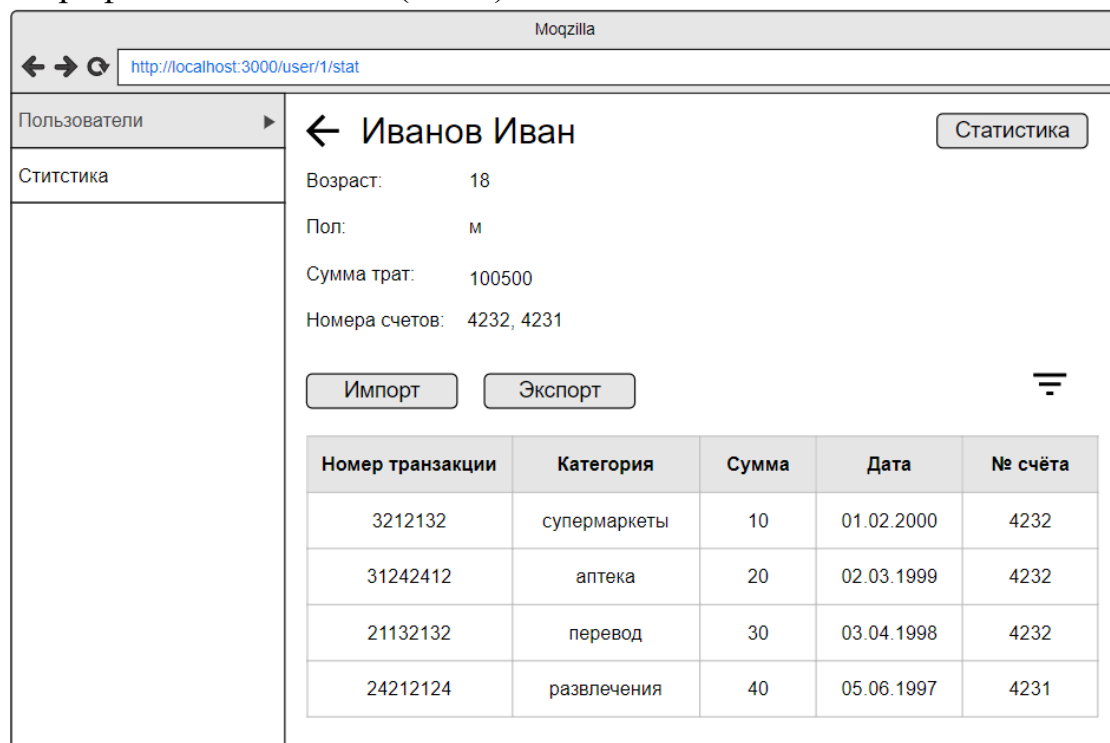


Рисунок 2. Профиль пользователя

Фильтрация списка транзакций пользователя(Рис.3)

Пользователи

Статистика

← Иванов Иван

Статистика

Возраст: 18

Пол: м

Сумма трат: 100500

Номера счетов: 4232, 4231

Импорт Экспорт

Номер транзакции	Категория	Сумма
3212132	супермаркеты	10
31242412	аптека	20
21132132	перевод	30
24212124	развлечения	40

По дате

С

По

По категории

☒ Супермаркеты

☒ Аптека

☒ Перевод

☒ Развлечения

☒ Другое

Применить

счёта

Рисунок 3. Фильтрация списка транзакций пользователя

Страница транзакции(Рис.4)

Пользователи

Статистика

← № 3212132

Категория: супермаркеты

Сумма: 10

Дата: 01.02.2000

Рисунок 4. Страница транзакций

Статистика пользователя за всё время(Рис.5)

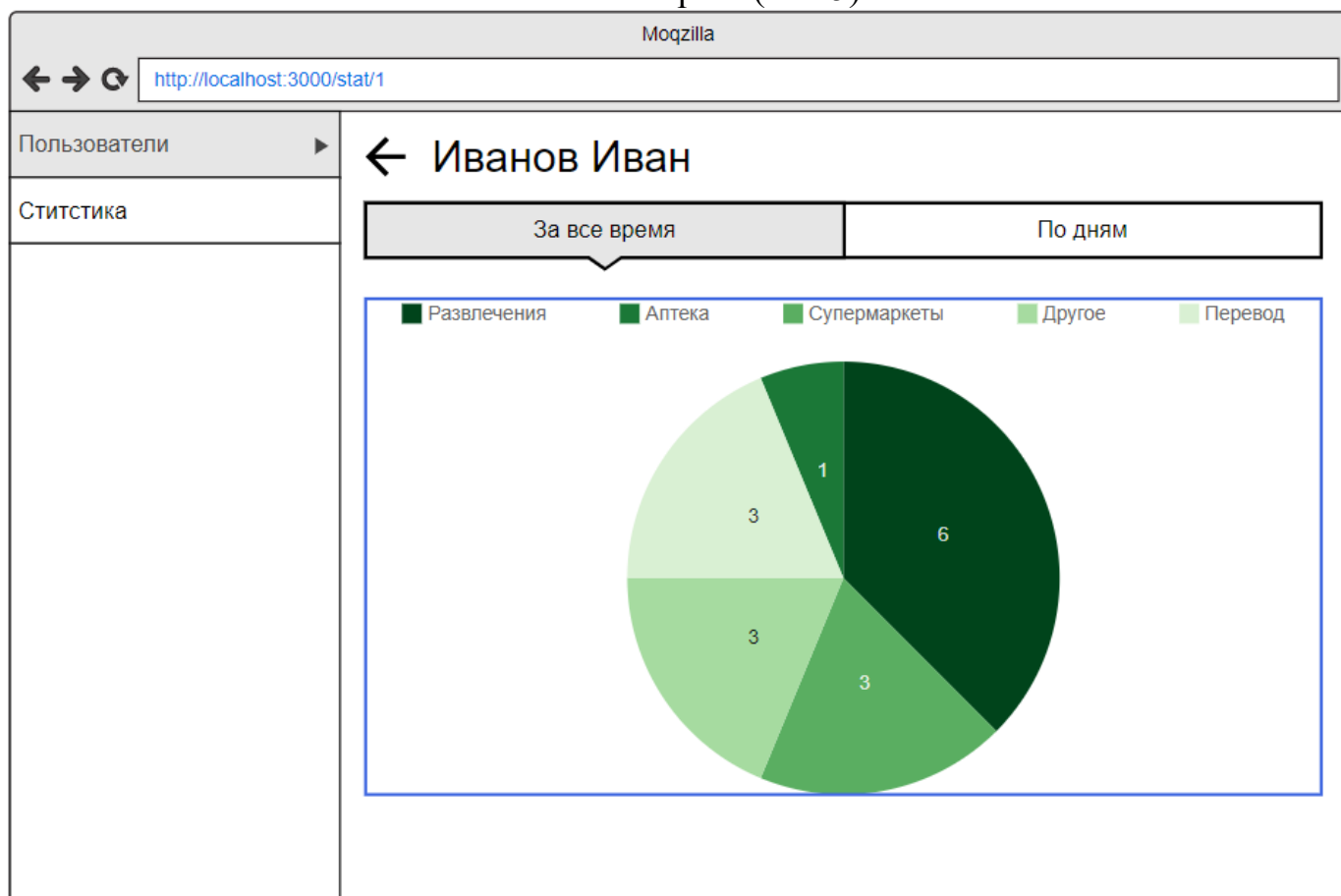


Рисунок 5. Статистика пользователя за всё время

Статистика пользователя по дням(Рис.6)

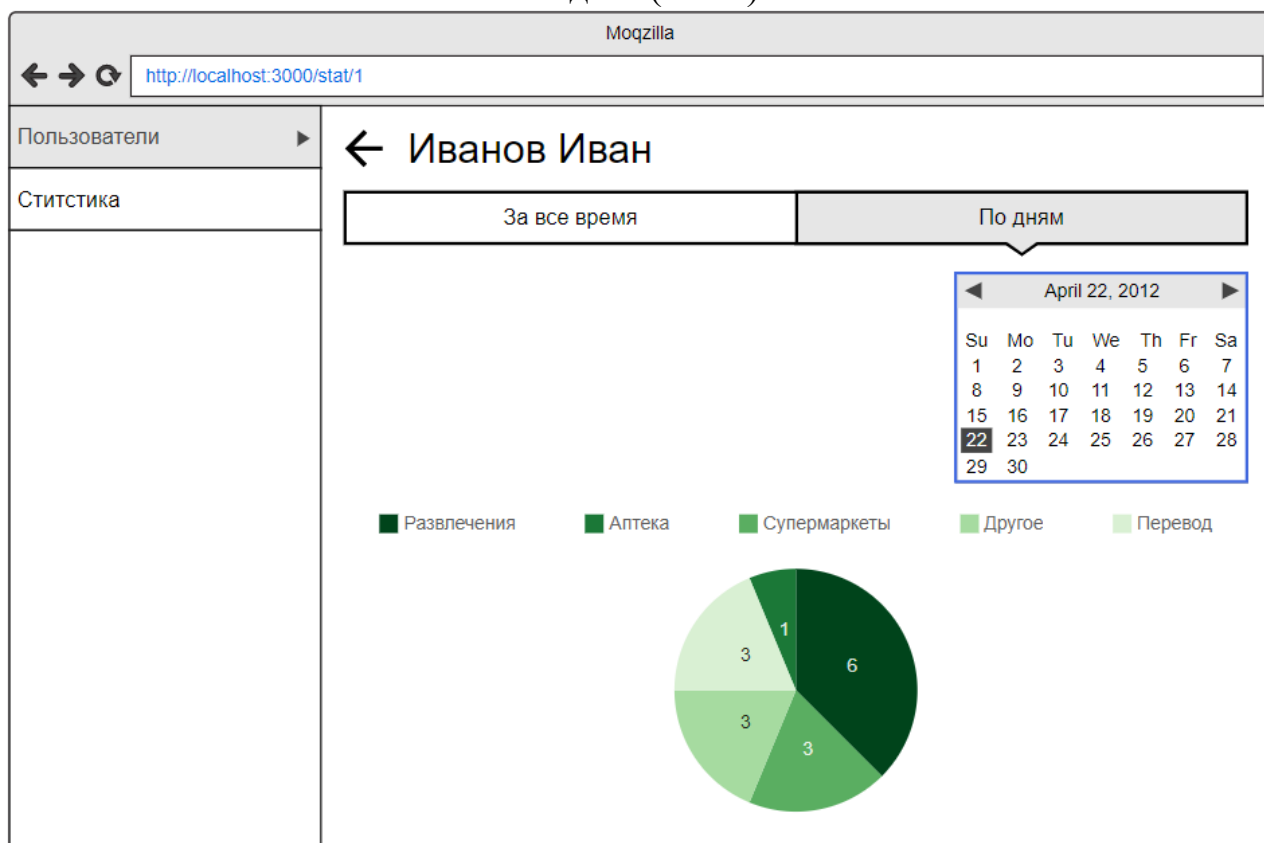


Рисунок 6. Статистика пользователя по дням

Схема экранов приложения(Рис.7)

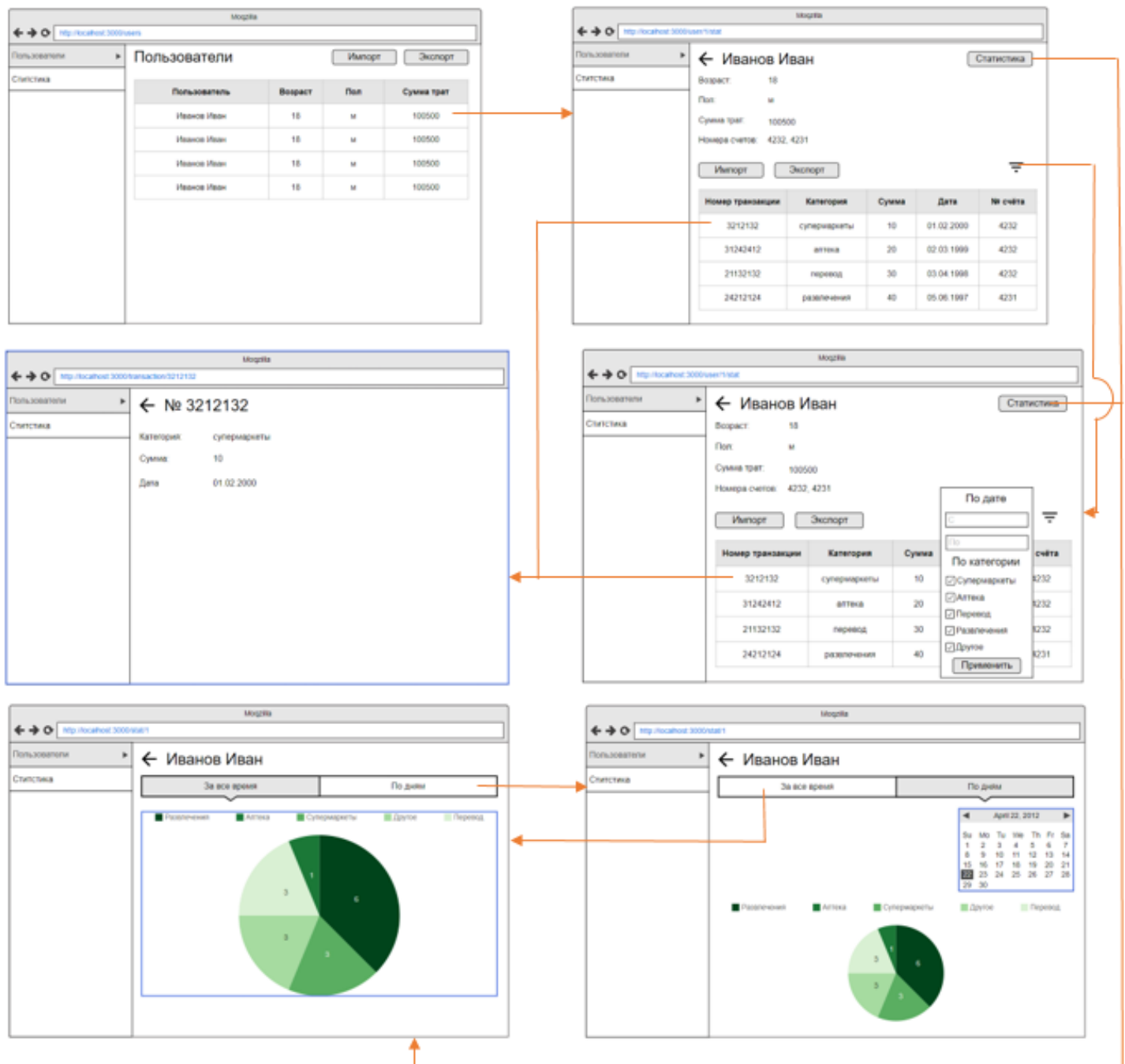


Рисунок 7. Схема экранов приложения

3.2 Сценарии использования для задач:

Для отображения таблицы всех пользователей в системе:

1. В главном меню нажать вкладку "Пользователи"
2. Подождать завершения загрузки таблицы
3. В результате успешного завершения, на экране пропадет иконка загрузки
4. Если загрузка не завершается длительное время, то попробовать в другое время

Для перехода на страницу определенного пользователя:

1. На странице "Пользователи" в таблице всех пользователей навести на строку нужного пользователя
2. Нажать на выбранную строку таблицы
3. Подождать завершения загрузки страницы
4. В результате успешного завершения, на экране пропадет иконка загрузки
5. Если загрузка не завершается длительное время, то попробовать в другое время

Для фильтрации списка транзакций по дате:

1. Перейти на страницу нужного пользователя
2. Рядом со списком транзакций нажать иконку "Фильтр"
3. Ввести в поле "С" дату с которой надо фильтровать
4. Ввести в поле "По" дату по которую надо фильтровать
5. Нажать на кнопку "Применить"
6. Подождать завершения загрузки таблицы
7. В результате успешного завершения, на экране пропадет иконка загрузки
8. Если загрузка не завершается длительное время, то попробовать в другое время

Для фильтрации списка транзакций по категории:

1. Перейти на страницу нужного пользователя
2. Рядом со списком транзакций нажать иконку "Фильтр"
3. Отметить названия категорий, по которым нужно фильтровать
4. Нажать на кнопку "Применить"
5. Подождать завершения загрузки таблицы
6. В результате успешного завершения, на экране пропадет иконка загрузки
7. Если загрузка не завершается длительное время, то попробовать в другое время

Для перехода на страницу транзакции

1. Перейти на страницу нужного пользователя
2. В таблице всех транзакций выбрать строку с нужной транзакцией
3. Нажать на выбранную строку
4. Подождать завершения загрузки страницы
5. В результате успешного завершения, на экране пропадет иконка загрузки
6. Если загрузка не завершается длительное время, то попробовать в другое время

Для перехода на страницу статистики за все время нужного пользователя

1. Перейти на странице нужного пользователя
2. Нажать на кнопку "Статистика"
3. В меню выбрать пункт "За все время"
4. Подождать завершения загрузки статистики
5. В результате успешного завершения, на экране пропадет иконка загрузки
6. Если загрузка не завершается длительное время, то попробовать в другое время

Для перехода на страницу статистики по дням нужного пользователя

1. Перейти на странице нужного пользователя
2. Нажать на кнопку "Статистика"
3. В меню выбрать пункт "По дням"
4. В календаре выбрать необходимую дату
5. Подождать завершения загрузки статистики
6. В результате успешного завершения, на экране пропадет иконка загрузки
7. Если загрузка не завершается длительное время, то попробовать в другое время

Для перехода на страницу статистики за все время всех пользователей

1. В главном меню выбрать пункт "Статистика"
2. В меню выбрать пункт "За все время"
3. Подождать завершения загрузки статистики
4. В результате успешного завершения, на экране пропадет иконка загрузки
5. Если загрузка не завершается длительное время, то попробовать в другое время

Для перехода на страницу статистики по дате всех пользователей

1. В главном меню выбрать пункт "Статистика"
2. В меню выбрать пункт "По дате"
3. В календаре выбрать необходимую дату
4. Подождать завершения загрузки статистики
5. В результате успешного завершения, на экране пропадет иконка загрузки
6. Если загрузка не завершается длительное время, то попробовать в другое время

Для массового добавления и получения данных из таблиц сайта необходимо нажать на кнопки импорт и экспорт соответственно. Страницы статистики всех пользователей и конкретного пользователя различаются минимально.

4. Модель данных

4.1 Нереляционная модель данных

4.1.1. Графическое представление модели данных

Графическое представление модели данных:

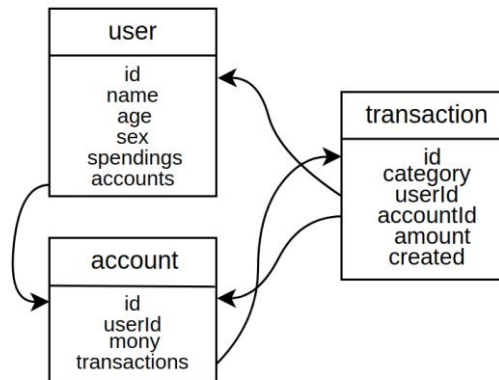


Рис 8. Модель данных для NoSQL СУБД

4.1.2. Описание назначений коллекций, типов данных и сущностей.

В качестве СУБД используется Firebase. В ней в виде коллекций документов хранятся данные о пользователях, их счетах и транзакциях

1. User

- *id* – String - Идентификатор пользователя. $V = 2b \cdot 20 = 40b$ (Firebase по умолчанию генерирует 20-ти значный id)
- *name* - String - Полное имя пользователя. $V = 2b \cdot 25 = 50b$
- *age* – Number - Возраст пользователя. $V = 4b$
- *sex* – String - Пол пользователя. $V = 2b$ (одна буква м/ж)
- *spendings* - Number - Общие траты пользователя за всё время. $V = 2b \cdot 25 = 50b$
- *accounts* - String[] - Список счетов пользователя. $V = 2b \cdot 20 \cdot N = 40b$ (N количество счетов)

2. Account

- *id* - String - Идентификатор счёта. $V = 2b \cdot 20 = 40b$
- *userId* - String - Идентификатор владельца счёта. $V = 2b \cdot 20 = 40b$
- *money* - Number - Объем средств на счету. $V = 2b \cdot 15 = 40b$
- *transactions* - String[] - Список всех транзакций, совершенных на этом счету. $V = 2b \cdot 20 \cdot N = 40b$ (N количество транзакций)

3. Transaction

- *id* - String - Идентификатор финансовой транзакций. $V = 2b * 20 = 40b$
- *category* - String - Категория покупки. $V = 2b * 10 = 20b$
- *userId* - String - Идентификатор пользователя, проводившего транзакцию. $V = 2b * 20 = 40b$
- *accountId* - String - Номер счёта с которого проводилась транзакция. $V = 2b * 20 = 40b$
- *amount* - Number - Размер транзакции. $V = 2b * 4 = 8b$
- *crated* - Date - Дата проведения транзакции. $V = 8b$

Стоит отметить, что самые дорогостоящие операции используются для статистики, когда необходимо сложным образом обработать коллекцию и структурировать информацию так, чтобы клиентской части было удобно с ней работать.

4.1.3. Оценка удельного объема информации, хранимой в модели

«Чистый» объем:

- $user_V = 106b$
- $account_V = 40b$
- $transaction_V = 76b$

Фактический объем:

- $user_V = 146b + 40b * accounts_N$
- $account_V = 120b + 40b * transactions_N$
- $transaction_V = 156b$

Избыточность:

Предположим, что в среднем у пользователя 3 счета, и он совершил за некоторое время по 100 транзакций на каждый счёт. За N примем количество пользователей.

$$clear_V = N * user_V + N * account_V * 3 + N * transaction_V * 3 * 100 = 23026 * N b$$

$$fact_V = 28226 * N b$$

Итого: 1.225

Направление роста:

Основной причиной роста объёма данных БД являются транзакции. При добавлении одной транзакции создаётся новый документ транзакции (268b) плюс добавление новой записи в список транзакций счёта (40b). В итоге 308b на одну запись. При расчёте направления роста можно не учитывать новых пользователей и их счета, т.к в сравнении с количеством транзакций их вклад не велик.

4.1.4. Запросы к модели, с помощью которых реализуются сценарии использования.

- Запрос на получение списка пользователей:

```
firestore.collection('users').get()
```

- Запрос на получение транзакций пользователя, отсортированных по времени:

```
firestore.collection('transactions').where('userId', '==',  
[USER_ID]).orderBy('created', 'desc').get()
```

- Запрос на получение счетов пользователя:

```
firestore.collection('accounts').where('userId', '==', [USER_ID]).get()
```

4.2. Аналог модели данных для SQL СУБД – графическое представление данных и сравнение с моделью данных для NoSQL БД.

4.2.1. Графическое представление модели данных

Графическое представление модели данных:

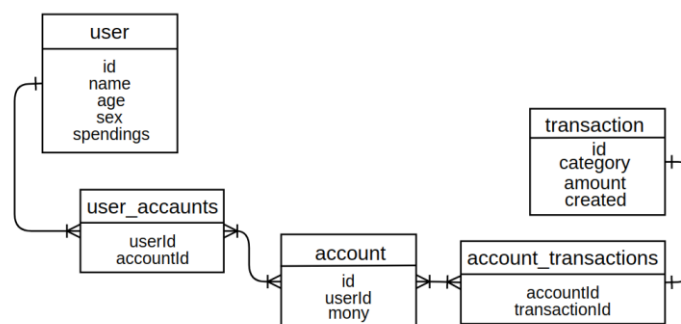


Рис 9. Модель данных для SQL СУБД

4.2.2. Описание назначений коллекций, типов данных и сущностей.

User:

- id* – varchar - Идентификатор пользователя. $V = 2b \cdot 20 = 40b$
- name* – varchar - Полное имя пользователя. $V = 2b \cdot 25 = 50b$ (25 - примерная средняя длина ФИО)
- age* - int - Возраст пользователя. $V = 4b$
- sex* – varchar - Пол пользователя. $V = 2b$ (одна буква м/ж)
- spendings* – int - Общие траты пользователя за всё время. $V = 2b \cdot 25 = 50b$

Account:

- id* - varchar - Идентификатор счёта. $V = 2b \cdot 20 = 40b$
- userId* - varchar - Идентификатор владельца счёта. $V = 2b \cdot 20 = 40b$
- money* - int - Объем средств на счету. $V = 2b \cdot 15 = 40b$

Transaction:

- *id* - varchar - Идентификатор финансовой транзакций. $V = 2b * 20 = 40b$
- *category* - varchar - Категория покупки. $V = 2b * 10 = 20b$
- *amount* - int - Размер транзакции. $V = 2b * 4 = 8b$
- *crated* - date - Дата проведения транзакции. $V = 8b$

User_account:

- *userid* - varchar - Идентификатор пользователя. $V = 2b * 20 = 40b$
- *accountld* - varchar - Идентификатор аккаунта. $V = 2b * 10 = 20b$

Account_transaction:

- *transactionid* - varchar - Идентификатор пользователя. $V = 2b * 20 = 40b$
- *accountId* - varchar - Идентификатор аккаунта. $V = 2b * 10 = 20b$

4.2.3. Оценка удельного объема информации, хранимой в модели

«Чистый» объем:

- $user_V = 106b$
- $account_V = 40b$
- $trnsaction_V = 76b$

Фактический объем:

- $user_V = 146b$
- $user_accounts_V = 80b * accounts_N$
- $account_V = 120b$
- $account_transactions_V = 80b * transactions_N$
- $transaction_V = 76b$

Избыточность:

Предположим, что в среднем у пользователя 3 счета, и он совершил за некоторое время по 100 транзакций на каждый счёт. За N примем количество пользователей.

$$clear_V = N * user_V + N * account_V * 3 + N * transaction_V * 3 * 100 = 23026 * N b$$

$$fact_V = N * user_V + N * account_V * 3 + N * user_accounts_V * 3 + \\ N * transaction_V * 3 * 100 + N * account_transactions_V * 300 = 47546 * N b$$

Итого: 2.064

4.2.4. Запросы к модели, с помощью которых реализуются сценарии использования.

- Запрос на получение списка пользователей:

```
SELECT * FROM user
```

- Запрос на получение транзакций пользователя, отсортированных по времени:

```
SELECT transaction.*  
FROM transactions  
LEFT JOIN account_transactions ON transaction.id =  
account_transactions.transactionId  
LEFT JOIN user_accounts ON user_accounts.accountId =  
account_transactions.accountId  
WHERE user_accounts.userId = [USER_ID]  
ORDER BY transaction.created
```

- Запрос на получение счетов пользователя:

```
SELECT *  
FROM accounts  
WHERE userId = [USER_ID]
```

4.3. Вывод:

Можно сделать следующие выводы:

- Для необходимых запросов в SQL необходимо делать дополнительные таблицы связности
- Таблицы связности занимают больше места, чем просто массив хранящий идентификаторы, т.к массив хранится прямо в документе пользователя (занимает $user_id + transaction_id * transactions_N$), а таблица хранит каждую запись отдельно ($(user_id + transaction_id) * transactions_N$)
- Аналогичные запросы в SQL являются более громоздкими

5. Разработанное приложение

5.1. Краткое описание

Результатом разработки приложения на клиентской стороне стало SPA (Single Page Application).

Интерфейс приложения написан на языке TypeScript для типизации данных. Использовались библиотеки React.js и Redux.js для реализации SPA. Для выполнения синхронных запросов на сервер использовалась библиотека redux-saga. Компоненты, использованные в интерфейсе приложения, были получены из библиотеки material-ui.

Серверная часть приложения написана с помощью фреймворка express, для выполнения запросов в базу данных.

5.2. Схема экранов приложения

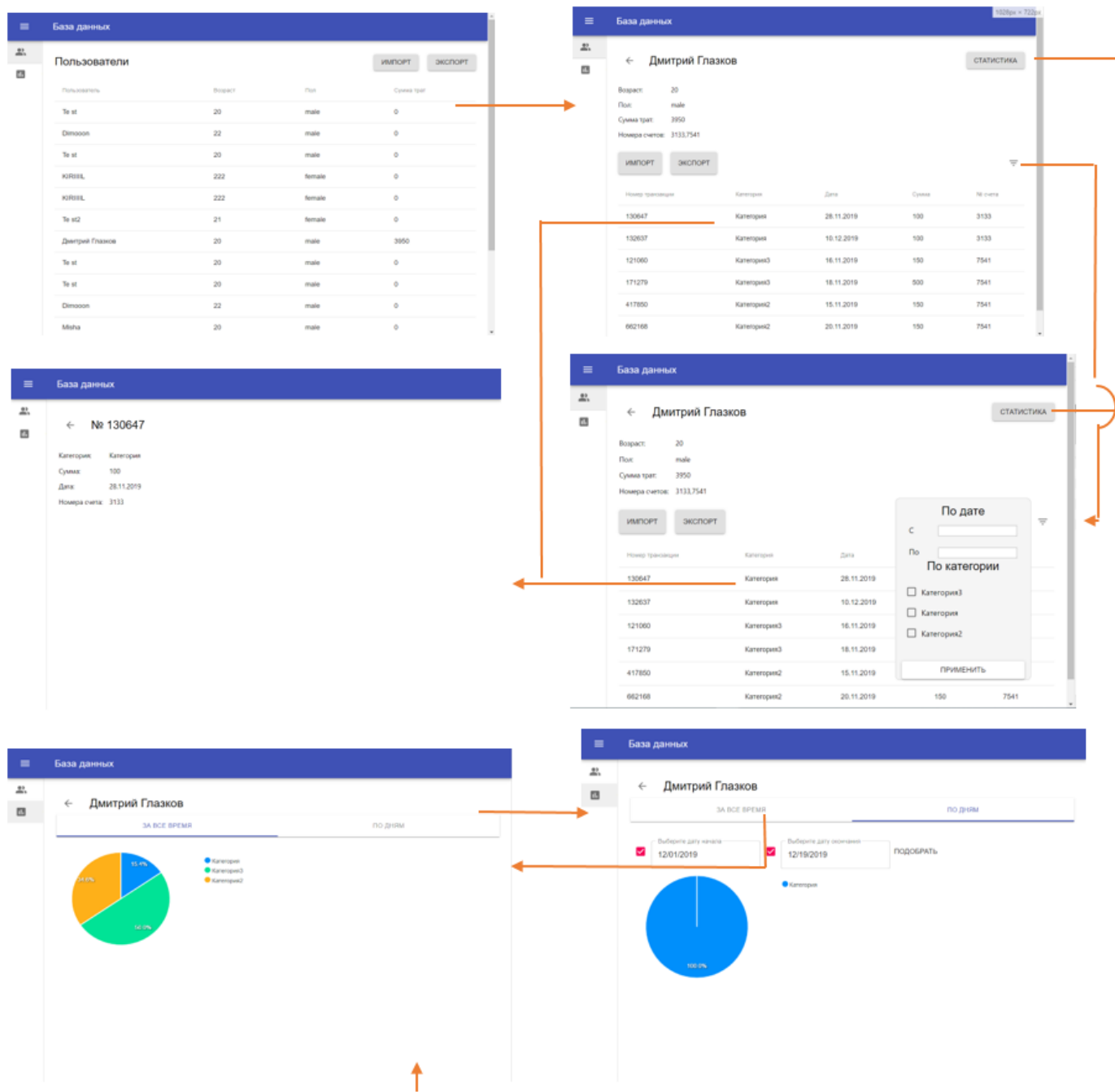


Рисунок 10. Схема экранов приложения

5.3.Использованные технологии

Использованные технологии:

- Firebase — документоориентированная СУБД;
- Express.js — каркас веб-приложений, работающий поверх Node.js;
- React.js/Redux — JavaScript-библиотека для разработки веб-интерфейса.
- Node.js — JavaScript платформа для серверной разработки.

5.4.Ссылки на приложение

Исходный код приложения и инструкция по установке находится по ссылке:

<https://github.com/moevm/nosql2h19-shop>

6. Выводы

6.1.Достигнутые результаты

Было разработано приложение, способное отслеживать потребности человека, импортировать расходы из банка, предоставлять в удобном виде статистику по совершенным операциям.

Достигнутые результаты:

- Приложение разработано меньше, чем за 80 часов;
- Приложение достаточно динамичное и быстрое;
- Приложение решает минимальные нужды пользователя.

6.2.Недостатки и пути для улучшения полученного решения

В рамках курса “ Введение в нереляционные базы данных ” приложение можно считать хорошо разработанным, но в рамках будущего продукта существует ряд недостатков:

- Технические недочеты

Причина: отсутствие опыта в выбранных технологиях.

Решение: потребуется около 16 часов для рефакторинга, оптимизации запросов и улучшения юзабилити приложения.

- Недостаточный функционал для продукта

Текущее состояние приложение можно оценивать, как прототип продукта, т.к. функционала для желаемого отслеживания денежного потока недостаточно.

Причина: недостаточное время для разработки и отсутствие опыта у разработчика.

Решение: продолжать разработку продукта.

6.3. Будущее развитие решения

В ближайших планах планируется реализовать:

- Личный кабинет каждого пользователя;
- Улучшить дизайн и user-experience;
- Добавить инструкцию по использованию на сайт и всплывающие подсказки.

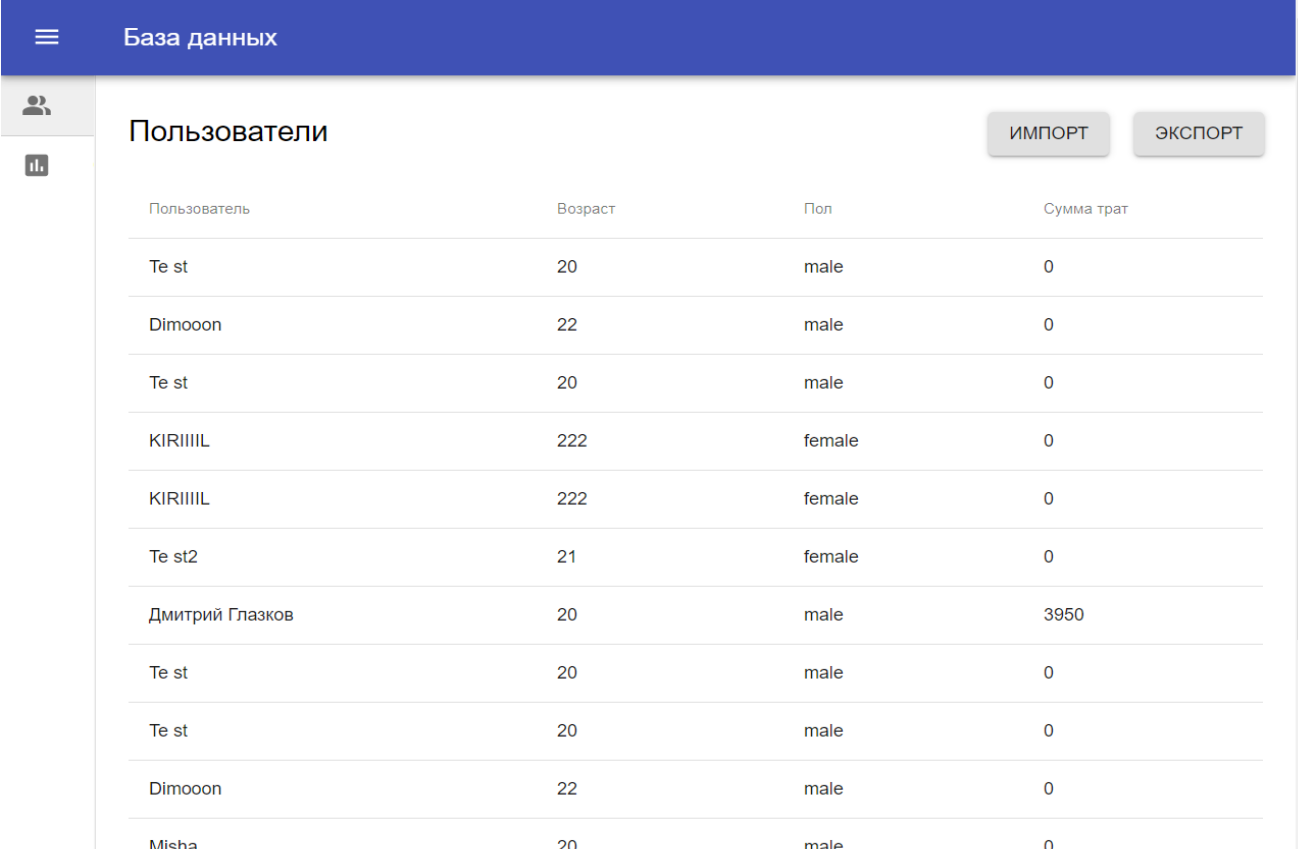
7. Приложения

7.1. Документация по сборке и развертыванию приложения

Инструкция по сборке и запуску:

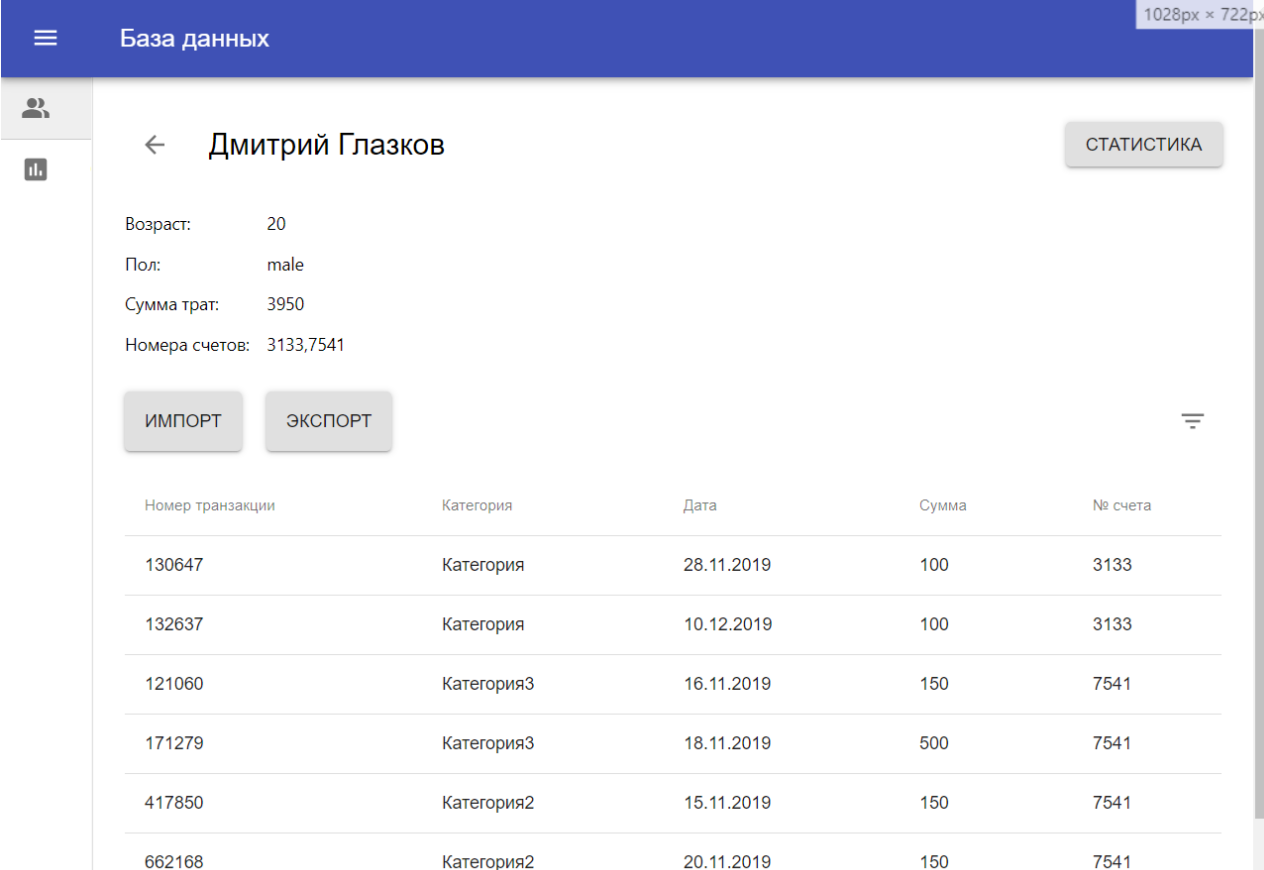
1. Скачать проект из репозитория;
2. Перейти в корневую папку проекта и в терминале ввести: `yarn install`;
3. Собрать приложение: `yarn build`;
4. Перейти в папку `functions` и в терминале ввести: `npm install`;
5. Запустить Firebase-сервер: `firebase serve`;
6. Перейти в браузере по адресу: `http://localhost:5000`.

7.2. Снимки экрана приложения



Пользователь	Возраст	Пол	Сумма трат
Te st	20	male	0
Dimooon	22	male	0
Te st	20	male	0
KIRIIIL	222	female	0
KIRIIIL	222	female	0
Te st2	21	female	0
Дмитрий Глазков	20	male	3950
Te st	20	male	0
Te st	20	male	0
Dimooon	22	male	0
Misha	20	male	0

Рис 11. Страница списка пользователей



Номер транзакции	Категория	Дата	Сумма	№ счета
130647	Категория	28.11.2019	100	3133
132637	Категория	10.12.2019	100	3133
121060	Категория3	16.11.2019	150	7541
171279	Категория3	18.11.2019	500	7541
417850	Категория2	15.11.2019	150	7541
662168	Категория2	20.11.2019	150	7541

Рис 12. Страница пользователя

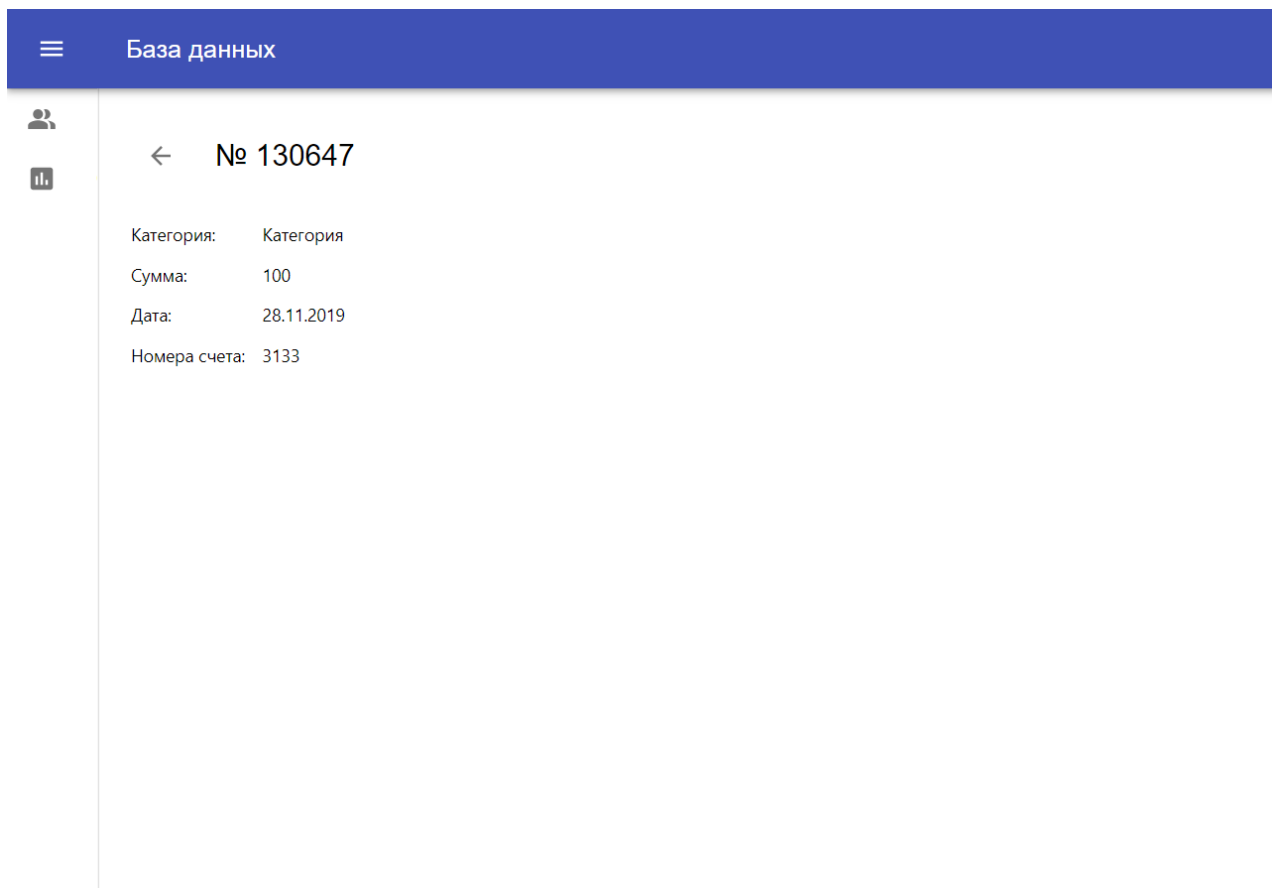


Рис 13. Страница транзакции

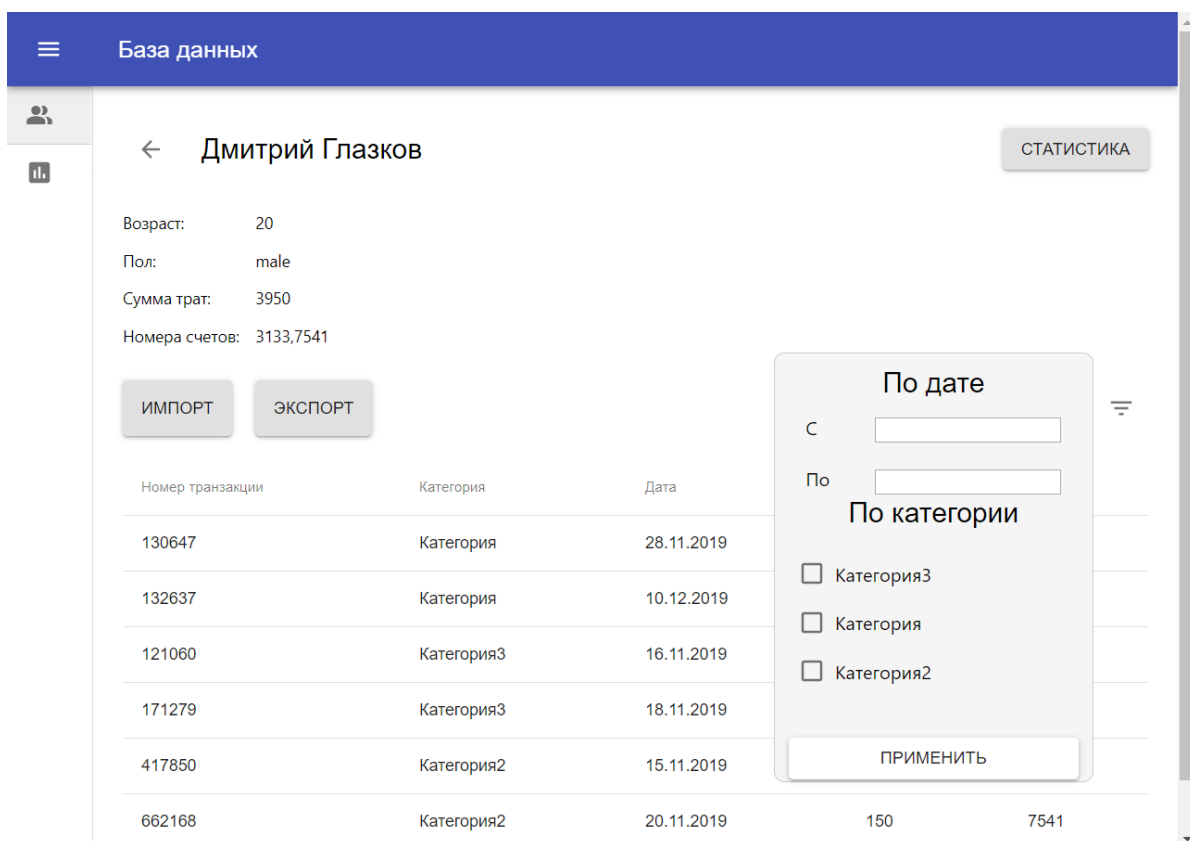


Рис 14. Страница сортировки транзакций по дате

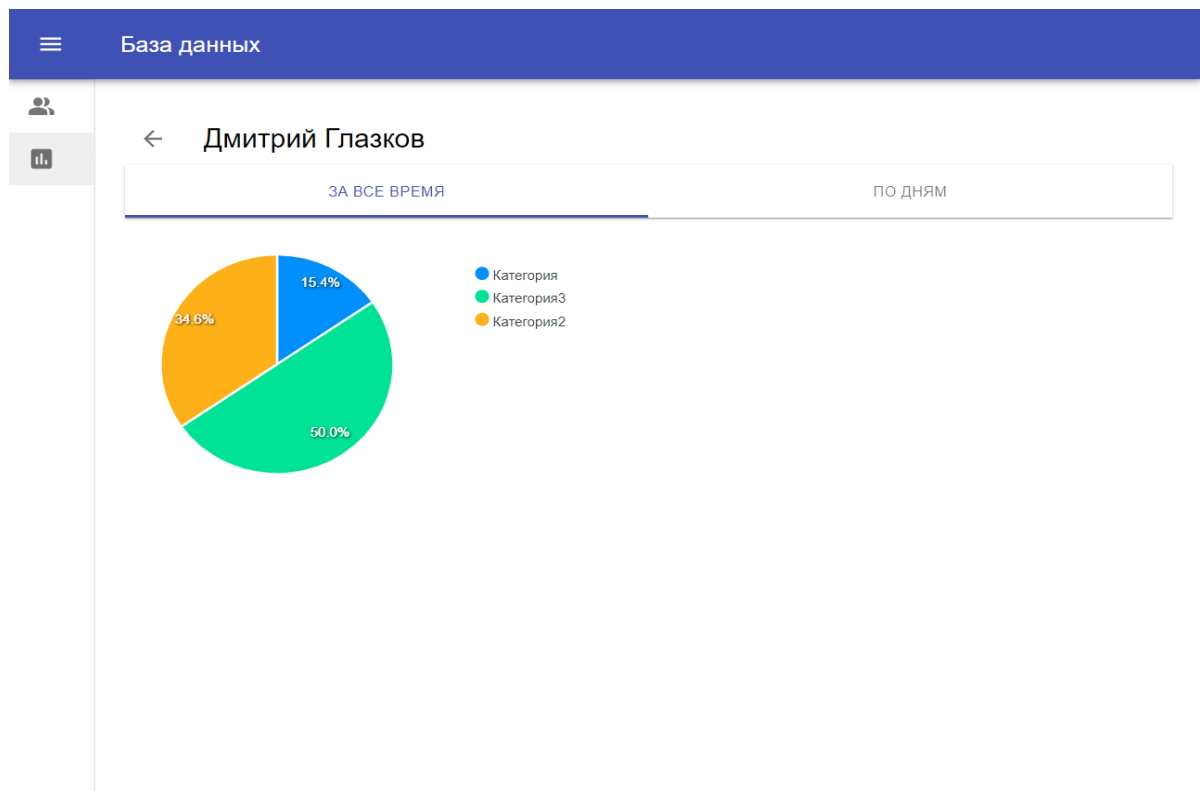


Рис 15. Страница статистики за всё время

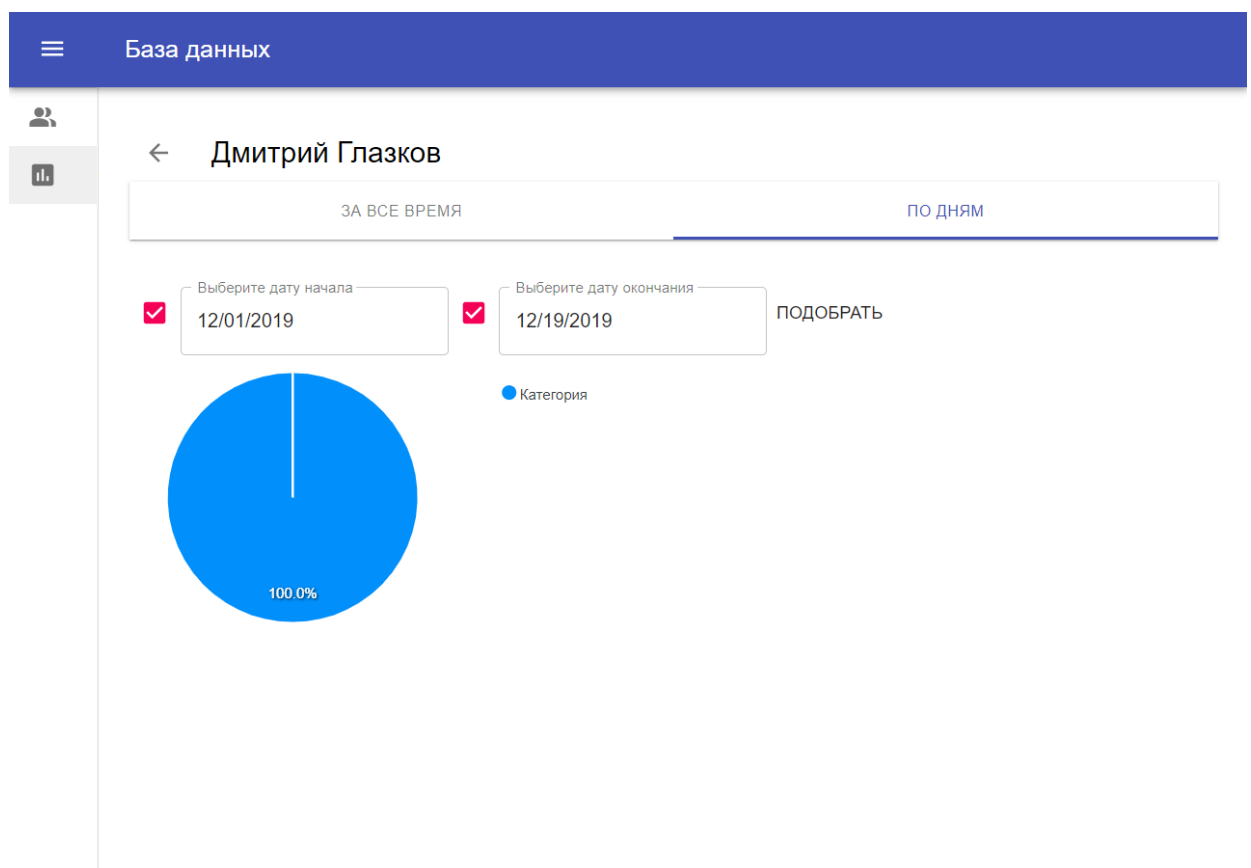


Рис 16. Страница статистики по дням

8. Литература

1. Документация по Firebase: <https://firebase.google.com/docs/guides>
2. Документация по NodeJS: <https://nodejs.org/ru/docs/guides/>
3. Пример отчета: https://github.com/moevm/nosql-2017-lib_card/blob/master/%D0%9F%D0%BE%D1%8F%D1%81%D0%BD%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F%20%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D0%BA%D0%B0.pdf
4. <https://www.youtube.com/watch?v=IVdMNHWWSYw&list=PLlb7e2G7aSpTABCq2ifA8dac39QuxbR1K>