­­­­­­­­­МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования

**«Южно-Уральский государственный университет»**

**(национальный исследовательский университет)**

**Высшая школа электроники и компьютерных наук**

**Кафедра системного программирования**

|  |  |
| --- | --- |
| РАБОТА ПРОВЕРЕНА  Рецензент  Директор ООО «Наполеон Айти»  \_\_\_\_\_\_\_\_\_\_ П.С. Подкорытов  “\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 2019 г. | ДОПУСТИТЬ К ЗАЩИТЕ  Заведующий кафедрой,  д.ф.-м.н., профессор  \_\_\_\_\_\_\_\_\_\_ Л.Б. Соколинский  “\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 2019 г. |

**Разработка мобильного сканера QR-кода чека для ведения личного бюджета на языке Dart   
с использованием SDK Flutter   
на платформе Android**

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ЮУрГУ – 02.03.02.2019.11-032-19091.ВКР

|  |  |
| --- | --- |
|  | Научный руководитель  доцент кафедры СП, к.п.н.  \_\_\_\_\_\_\_\_\_\_ О.Н. Иванова  Автор работы,  студент группы ВМИ-401  \_\_\_\_\_\_\_\_\_\_ А.И. Панченко  Ученый секретарь  (нормоконтролер)  \_\_\_\_\_\_\_\_\_\_ О.Н. Иванова  “\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 2019 г. |

Челябинск–2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования

**«Южно-Уральский государственный университет»**

**(национальный исследовательский университет)**

**Высшая школа электроники и компьютерных наук**

**Кафедра системного программирования**

УТВЕРЖДАЮ

Зав. кафедрой СП

\_\_\_\_\_\_\_\_\_\_ Л.Б. Соколинский

10.02.2019

**ЗАДАНИЕ1**

**на выполнение выпускной квалификационной работы бакалавра**

студенту группы КЭ-401 Панченко Артёму Игоревичу,

обучающемуся по направлению 02.03.02  
«Фундаментальная информатика и информационные технологии»

1. **Тема работы** (утверждена приказом ректора от 28.04.2017 № 835)

Разработка мобильного приложения для формирования и оформления заказов на продукцию в розничной сети магазинов

1. **Срок сдачи студентом законченной работы:** 04.06.2019.
2. **Исходные данные к работе: 2**
3. Марк Д., Наттинг Дж. iOS 6 SDK Разработка приложений для iPhone, iPad и iPod touch. – США: APress, 2013. – 672 p.

2. Арлоу Дж., Нейштадт А. UML 2 и унифицированный процесс. – М.: Символ-Плюс, 2007. – 624 p.

3. Hegarty P. Standford University Lecture – Developing iOS7 App. [Электронный ресурс] URL: https://www.youtube.com/watch?v=ZqKbN\_C4Yvg&list=PLnOdYr35FyvhDUAIW17vo7nGfHJAyikUp (дата обращения: 11.03.2017).

1. **Перечень подлежащих разработке вопросов3**
2. Произвести постановку задачи по созданию мобильного приложения.
3. Произвести сравнительный обзор существующих аналогов для разрабатываемого мобильного приложения.
4. Изучить современные платформы и средства разработки для операционной системы Android.
5. Определить требования и спроектировать мобильное приложение.
6. Реализовать и протестировать мобильное приложение.
7. **Дата выдачи задания:** 10.02.2019.

**Научный руководитель4**

Доцент кафедры СП, к.п.н. О.Н. Иванова

**Задание принял к исполнению** А.И. Панченко

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 5](#_Toc7211867)

[1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 8](#_Toc7211868)

[1.1.Анализ предметной области 8](#_Toc7211869)

[1.2. Обзор аналогов 13](#_Toc7211870)

[1.3. Средства разработки 21](#_Toc7211871)

[2. ПРОЕКТИРОВАНИЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ 23](#_Toc7211872)

[2.1. Определение требований к проектируемому приложению 23](#_Toc7211873)

[2.2. Разработка диаграммы вариантов использования 24](#_Toc7211874)

[2.3. Проектирование интерфейса приложения 26](#_Toc7211875)

[2.4. Выбор средств и проектирование базы данных приложения 27](#_Toc7211876)

[3. РЕАЛИЗАЦИЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ 30](#_Toc7211877)

[3.1. Компоненты и структура проекта 30](#_Toc7211878)

[3.2. Реализация базы данных приложения 31](#_Toc7211879)

[3.3. Реализация функции отображения доходов и потраченных средств 32](#_Toc7211880)

[3.4. Реализация CRUD - функций 34](#_Toc7211881)

[3.5. Реализация запросов к операторам фискальных данных 36](#_Toc7211882)

[3.7. Реализация функции сканирования QR-кода чека 37](#_Toc7211883)

[3.8. Реализация страницы статистики расходов по категориям 39](#_Toc7211884)

[4. ТЕСТИРОВАНИЕ 41](#_Toc7211885)

[4.1. Выбор способов тестирования системы 41](#_Toc7211886)

[4.2. Функциональное тестирование 42](#_Toc7211887)

[4.3. Интеграционное тестирование 47](#_Toc7211888)

[4.4. Юзабилити-тестирование 48](#_Toc7211889)

[ЗАКЛЮЧЕНИЕ 49](#_Toc7211890)

[ЛИТЕРАТУРА 50](#_Toc7211891)

[ПРИЛОЖЕНИЕ 53](#_Toc7211892)

# ВВЕДЕНИЕ

Контроль личного бюджета стал важнейшим фактором социально-экономического развития страны и каждого гражданина, в частности [6]. Это объясняется принципиальными изменениями в жизни современного общества, требующими иных подходов к уровню необходимых знаний, пониманию ответственности личности и общества за благосостояние граждан. В качестве причин стоит выделить экономический кризис, в силу которого у граждан увеличивается необходимость учета и распределения личных финансов при росте цен, снижение стоимости накоплений, «урезание» заработной платы, отсутствие знаний в области финансов, нежелание или невозможность их получать, резкое изменение курса национальной валюты по отношению к иностранной и иные изменения экономической ситуации в стране.

Применение мобильных приложений для оперативного ведения в любое удобное для пользователя время – веяние современности. Развитие технической составляющей портативных устройств, а также покрытие большей части страны сетью 3G позволяет использовать смартфон в качестве устройства доступа к удаленным данным. Приложения для ведения личного бюджета популярны, однако не соответствуют основной идее мобильного устройства: быстрота и простота управления, передачи данных.

***Актуальность темы***

В современных условиях повышенного требования к уровню финансовой грамотности субъектов хозяйствования, вкупе с остро вставшей проблемой снижения уровня реальных доходов граждан, на фоне затянувшегося финансово-экономического кризиса, актуальность эффективного управления личными финансами на уровне отдельно взятой личности и семьи чрезвычайно высока.

Данная работа направлена на создание мобильного приложения, обеспечивающего сбор данных чека, обеспечивая оптимизацию затрат времени и ресурсов, потраченных на выполнение однотипной рутинной работы. Приложение позволит наглядно и быстро анализировать свой бюджет, что в дальнейшем приведет к улучшению планирования бюджета. Качественное планирование бюджета является залогом эффективного ведения персонального бюджета, которое помогает грамотно и результативно распределять свои денежные средства.

Ведение бюджета положительно влияет на экономическое благополучие человека [7] и позволяет:

* создать накопления без издержек по отношению к повседневной жизни;
* спланировать крупные покупки;
* исключить необоснованные кредитные договоренности;
* снизить финансовые риски;
* предупредить факт банкротства физического лица;
* раскрыть финансовый потенциал накоплений, экономическую активность гражданина.

Основной способ ведения бюджета для гражданина основывается на сборе данных чека, который гарантированно будет выдан продавцом в момент расчета.[[1]](#footnote-1)

***Цель и задачи***

Целью данной работы является анализ существующих решений и трендов в развитии ПО для ведения личного бюджета, а также разработка мобильного приложения, способного с помощью сбора и анализа данных с чека предоставлять обработанную информацию о финансах пользователю.

Для достижения цели работы, необходимо решить следующие задачи:

* провести анализ литературы и смежных проектов, связанных с ведением личного бюджета;
* определить требования к системе ведения бюджета и разработать варианты ее использования;
* разработать архитектуру мобильного приложения для ведения личного бюджета;
* разработать схему взаимодействия пользователя с интерфейсом приложения;
* разработать архитектуру базы данных для информации с чека;
* разработать мобильное приложение для устройств с операционной системой Android;

***Структура и объем работы***

Работа состоит из введения, четырех глав, заключения, библиографического списка и одного приложения. Объем работы составляет 39 страниц, объем библиографии – 24 источника. Объем приложения – 3 страницы.

В главе «Теоретическая часть» описана постановка задачи, произведен обзор существующих мобильных приложений для розничных сетей магазинов и программных средств для разработки мобильных приложений для операционной системы iOS.

Глава «Проектирование мобильного приложения для формирования и оформления заказов на продукцию в розничной сети магазинов» посвящена определению требований к разрабатываемому мобильному приложению. В этой же главе описываются диаграммы прецедентов и последовательности.

В третьей главе, «Реализация мобильного приложения для формирования и оформления заказов на продукцию в розничной сети магазинов», рассмотрена методология разработки приложения, описана диаграмма компонентов реализованного мобильного приложения и приведены некоторые методы.

Глава «Тестирование» посвящена результатам тестирования мобильного приложения. Представлены результаты функционального тестирования, выполненные в работающем приложении, и интегрального тестирования в различных условиях окружения.

В заключении сделаны выводы о проделанной работе и сформулированы перспективы дальнейшей разработки.

# 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 1.1.Анализ предметной области

В соответствии с поставленными задачами, необходимо проанализировать текущее состояние прикладной области, теоретические наработки и общие тенденции в прикладной отрасли, тем самым выделить и сформировать оценки и требования к проектируемому ПО.

Ограничениями в поиске подходящей информации, а также в обзоре текущих решений являются:

* обязательное использование мобильного устройства, преимущественно смартфона или планшета;
* приоритет, который заключается в анализе технологий и реализаций, использующих кассовый чек, в том числе старого образца;
* принадлежность к тематике «личный бюджет».

Основным объектом исследования является персональный(личный) бюджет, который можно представить, как систему данных денежных доходов и расходов человека, составляемая обычно на месячный срок в виде таблицы создающую баланс с соответствующими графами. Источники предлагают различные алгоритмы и способы составления, ведения бюджета: большинство работ направлены на пояснение практической части составления личного бюджета, однако анализ остается без должных комментариев [7,8, 12]. Можно выделить составляющие исследуемого объекта:

* личный бюджет — это план доходов и расходов на определенный промежуток времени.
* личный бюджет — это смета, которая позволяет контролировать движение денежных средств.

Для составления бюджета необходимы следующие действия:

* определение статей доходов и расходов;
* запись доходов и расходов;
* группировка доходов и расходов;
* расчет разницы между полученными доходами и произведенными расходами;
* постоянный учет доходов и расходов.

Структура доходов может быть следующая:

* Периодические доходы: ежемесячные доходы, ежегодные доходы;
* Разовые (прочие) доходы.

Аналогичной может быть структура расходов:

* Периодические расходы: ежемесячные расходы, ежегодные расходы;
* Разовые (прочие) расходы.

Описанная выше система по времени входящих данных не имеет ценности в разработке ПО, поскольку любое мобильное устройство позволяет создать динамическую структуру (еженедельную, ежедневную, конкретные даты), удобную для пользователя.

Однако есть такая категория как обязательные расходы – это платежи, повлиять на сроки и размер которых мы не можем, сами по себе являются де-юре обязательными и постоянными на длительное время (некоторые из них), остальные составляют ориентировочную постоянную (плата за коммунальные платежи не может превзойти двойную норму для человека, при использовании индивидуального подсчета расходов):

* налоги;
* платежи по кредитам;
* плата за коммунальные услуги и/или аренду жилья;
* плата за детский сад, дополнительное образование и т.п.

На современном этапе выделяют следующие причины низкой финансовой грамотности среди россиян: отсутствие навыков по составлению личного финансового плана, формированию денежных резервов на случай форс-мажорных обстоятельств; недостаточный уровень платежной дисциплины и надлежащего исполнения обязательств, вытекающих из договоров с кредитными организациями.

Опрос населения, прояснил, что почти половина респондентов – физических лиц принимает решения о выборе той или иной финансовой услуги или продукта не на основе анализа или сопоставления условий, а на основе иных факторов, например, по рекомендации друзей (таблица 1).

**Таблица 1. Результаты выборочного опроса населения в рамках получения информации об уровне финансовой грамотности**2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | 2009 | 2013 | 2017 |
| 1 | Ведете ли Вы или кто-то из Вашей семьи учет доходов и расходов? | | | |
|  | Да, учет ведется, все поступления и расходы зафиксированы | 11 | 12 | 21 |
| Да, учет ведется, но не все поступления и расходы зафиксированы | 13 | 19 | 21 |
| Нет, учет не ведется, но в целом доходы и расходы за месяц известны | 59 | 53 | 41 |
|  | Нет, учет не ведется, и неизвестно, каковы доходы и расходы за месяц | 14 | 15 | 14 |
| 2 | Считаете ли вы себя финансово грамотным человеком? | | | |
|  | Да, считаю | 15 | 13 | 12 |
| Считаю, но не по всем вопросам | 50 | 38 | 50 |
| Нет, не считаю | 35 | 49 | 38 |

Подводя промежуточный итог, мы имеем достаточные знания о том, как должен работать с личным бюджетом пользователь, какие общие вещи его интересуют, что имеет смысл систематизировать и обобщать и по каким признакам. Анализ такого критерия, как финансовой грамотности населения, задает определенные условия для проектируемого ПО:

* простота и доступность информации с точки зрения экономических терминов;
* совместимость с принятыми нормами ведения бюджета;
* пригодность полученных данных к анализу, статистической обработке;
* структурированность и научный подход к организации данных.

Долгое время федеральный закон «О применении контрольно-кассовой техники» фактически не учитывал особенностей дистанционной торговли и ведения расчетов через интернет, возможности получить цифровую копию чека. Новые изменения в ФЗ о ККТ вступившие в силу 1 июля направлены на решение проблем с сохранением прав потребителя при онлайн покупке и предоставляют потребителю дополнительную возможность доказать факт совершения сделки [9].

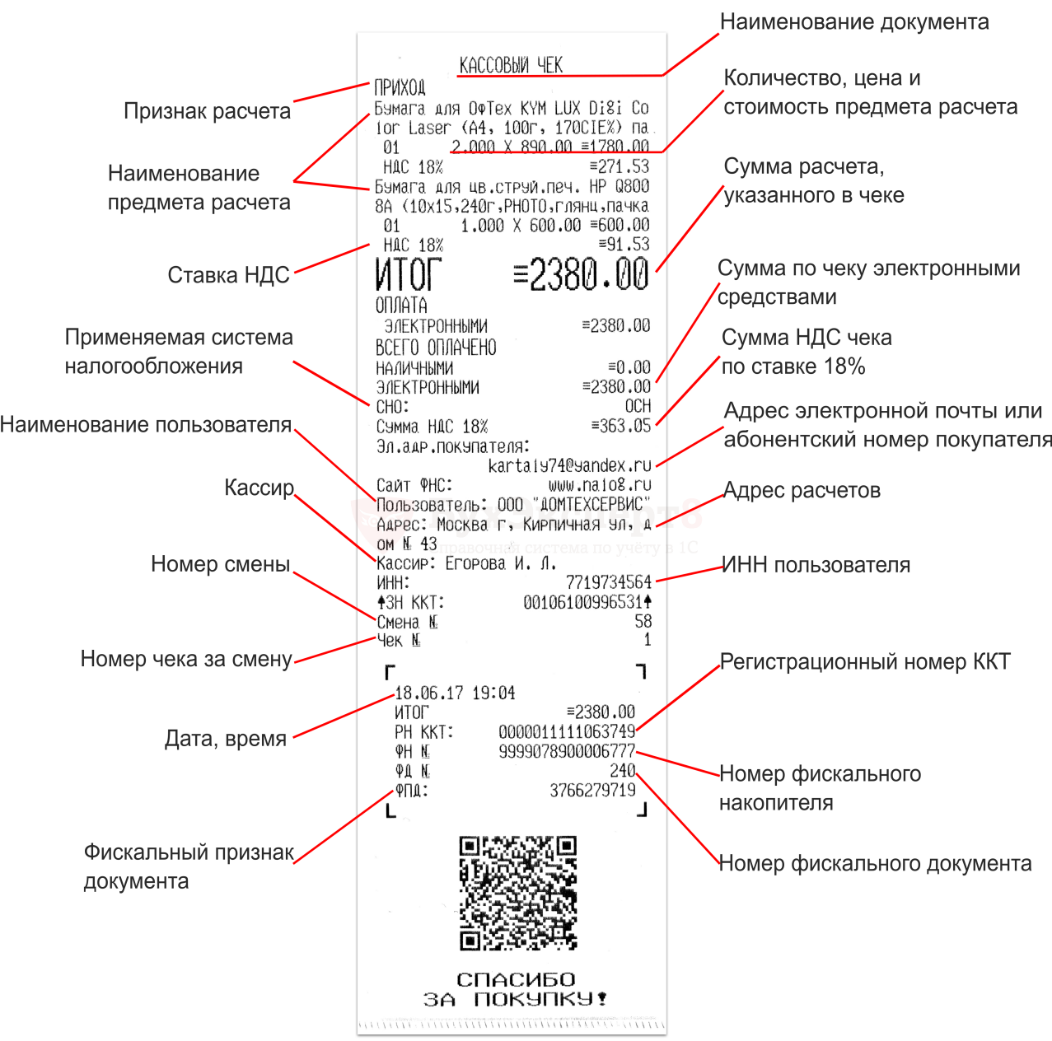
 Кассовые чеки претерпели изменения, заключающиеся в увеличении числа обязательных реквизитов, а также в присвоении QR-кода каждому чеку. Ниже представлен формат чека нового образца и приписанные новые реквизиты (Рис.1).

Рис.1 Экземпляр чека и данные содержащиеся в документе.

Это делает возможным доступ к электронной версии чека, посредством сканирования QR-кода для получения представленных выше реквизитов чека. [5]

Организации, занимающиеся предоставлением услуг населению (в большей части ритейл), начинают предоставлять чек нового образца с 2017 года. Таких организаций более чем 2 млн., однако зарегистрированных аппаратов по состоянию на январь 2017 г. было всего 20000. К июлю 2018 года число зарегистрированных онлайн-касс выросло до 3,5 – 4 млн [10]. Этому росту онлайн-касс способствует дальнейшая работа законопроекта: c 1 июля 2018 года обязаны применять онлайн-кассы и предоставлять чек нового типа, предоставляющий фискальные данные для создания запроса:

1. Организации и ИП на ЕНВД, которые ведут деятельность по пункту 2 статьи 346.26 НК РФ[[2]](#footnote-2);
2. ИП на патенте;
3. Организации и ИП при оказании услуг населению;

Претерпел изменение формат проведения чека. Теперь каждая продажа фиксируется так: как только продавец пробьет чек, онлайн-касса сформирует фискальный признак и отправит его на проверку оператору фискальных данных - посредник между кассой и ФНС. Таким оператором может быть российская организация, получившая соответствующее разрешение от государства. Эту информацию оператор сохранит, а обратно продавцу передаст уникальный номер чека. На это будет уходить не более 1,5 секунды. Без передачи данных оператору фискальных данных сформировать чек с номером будет попросту невозможно. Также оператор фискальных данных передаст информацию о совершенной продаже в ИФНС.

Стоит отметить, что онлайн-кассы могут довольно продолжительное время работать в автономном режиме при потере связи с интернетом, а когда она будет восстановлена, данные в полном объеме будут переданы в ФНС через операторов фискальных данных.

Описанное выше позволяет спроектировать и реализовать систему контроля личного бюджета с удобным и быстрым способом подсчета потраченных средств на различные категории товаров и услуг, подразумевающие законодательством РФ выдачу чека нового образца, который с каждым годом внедряется в большее число предприятий и организаций.

## 1.2. Обзор аналогов

Одно из направлений применения смартфонов связано с решением задач планирования личного или семейного бюджета, основанных на анализе текущих расходов. Использование мобильного приложения может существенно облегчить процесс отслеживания трат на различные категории товаров для пользователя. Появление смартфонов ускорило создание систем учета персональных доходов и затрат [8].

В настоящее время существует множество программ, предоставляющих классические функции: таблицы доходов и расходов, каталогизация позиций, возможность вывести информацию графически за определенное время. Популярностью пользуются программы с кроссплатформенной реализацией и использующие облачные хранилища для синхронизации между устройствами.

Можно выделить несколько реализаций для ведения бюджета:  
1. Классическое приложение, где предусмотрен лишь ручной ввод информации;

2. Приложения как дополнительный сервис приложения онлайн-банка;

3. Комбинированные приложения: возможность совмещения API банков (сервисов онлайн-банка) и сканирования QR-кода чека;

4. Системы распознавания текста чека по фотографии.

Параметры для оценивания приложения:

1. Для оценки приложения станут полученные ранее критерии и другие важные условия:
2. способы определения статей доходов и расходов;
3. способы записи статей доходов и расходов;
4. способы группировки полученных данных;
5. расчет разницы между доходом и расходом;
6. возможность получения детальной информации по каждой операции;
7. возможность получения информации в виде графиков, диаграмм, схем;
8. наличие автоматизации сбора данных, предлагаемые возможности автоматического сбора данных;
9. наличие сервисов хранения данных, получения данных с помощью сторонних сервисов или API;
10. информативность и простота пользовательского интерфейса;
11. эргономика приложения.

**Мобильное приложение CoinKeeper.**

CoinKeeper - классический представитель приложений для ведения бюджета [15].

Способ определения статей: ручной. Пользователь сам определяет категории товаров (может разделить коммунальные услуги и вести их раздельно, может их объединить).

Способ записи статей также самостоятельный. Существует возможность отследить суммы от банковских СМС, однако распределять полученную сумму из сообщения придется вручную. Функции сканирования QR-кода нет.

Группировать данные можно по дням. Не представляется возможным по адресу покупки, по бренду магазина и т.п., то есть предусмотрена группировка по фактическим расходам на конкретные категории товаров, созданные пользователем. Стоит учесть, что есть функция запланированных трат, по мере трат по определенной категории заполняется индикатор-иконка принятым пользователем цветом. Если заполнять доходы и расходы своевременно, то можно получить точную группировку по дням, однако она строго зависит от человека и это не всегда возможно в угоду возможных обстоятельств.

Присутствует ведение баланса: разницы между доходом и расходом, также присутствуют запланированные доходы и текущие расходы. Не ведется отдельная графа оставшихся средств с прошлых месяцев и накоплений отдельно, то есть все денежные средства сведены в одну сумму, что не приемлемо.

Отсутствует возможность детально описывать покупки. В целом, учитывая самостоятельное заполнение, эта функция была бы не востребована, однако для осмысленного ведения бюджета она необходима.

Присутствует вывод информации в виде диаграммы из созданных пользователем категорий. Описывается процентное соотношение и конкретная сумма. Можно получить диаграмму для доходов и расходов.

Сторонних сервисов не предусмотрено, все данные хранятся на устройстве.

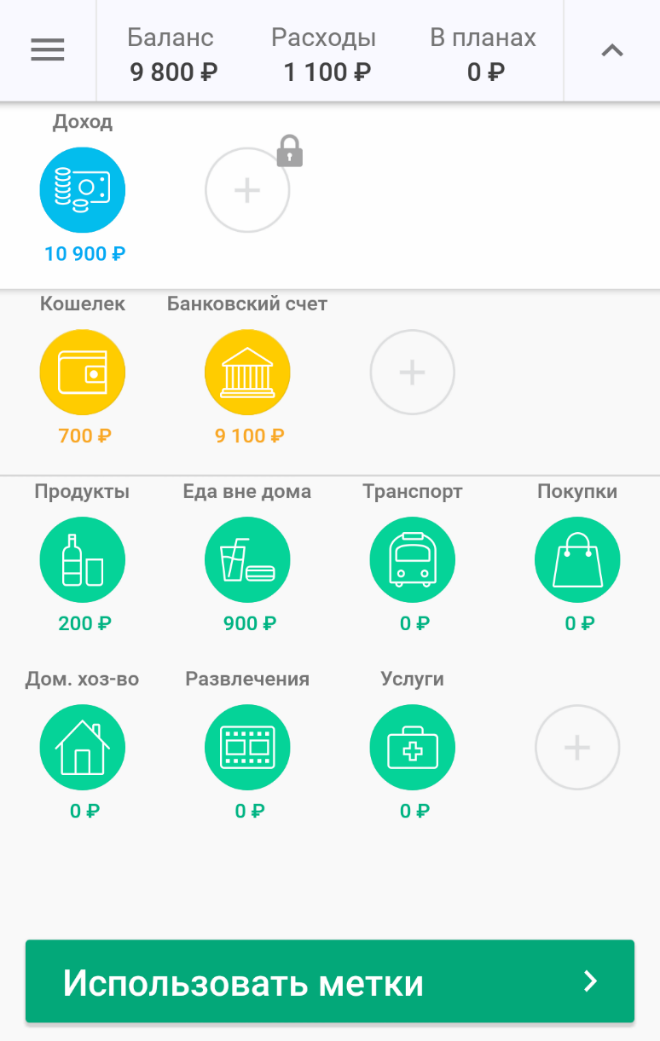
Пользовательский интерфейс (Рис.2) не однозначен: при первом использовании не сразу ясно, что нужно делать, чтобы заполнить данными накопления. Удобен при малом количестве категорий источников доходов, накоплений, расходов т.к. основной принцип взаимодействия – перетаскивание иконок на другую иконку. С точки зрения управления денежными потоками такое решение логичное и приемлемое. Присутствует разумное разделение на категории: накопления, источники доходов, расходы. Главная страница информативна, использование в качестве мобильного приложения в условиях не комфортных для подсчетов, вдумчивых операций приемлемое. Присутствуют напоминания, которые определяются пользователем, однако оставленные комментарии пользователей на странице загрузки приложения в Play Market, говорят о низкой эффективности напоминаний и сложностях в повседневном ведении бюджета с помощью этого приложения.

Рис.2. Главная страница приложения CoinKeeper.

Приложение удобно для ведения денежного баланса и учета денежных потоков, но, чтобы получить подробную информацию о тратах придется тратить значительное время, в таком случае преимуществом перед обычным ведением в домашней книге заключается лишь в мобильности, относительной быстроте заполнения. Отсутствие возможности сканировать QR-код, как эффективный способ получения всей информации чека создает трудности в повседневном использовании данного приложения.

**Мобильное приложение «Сбербанк Онлайн»**

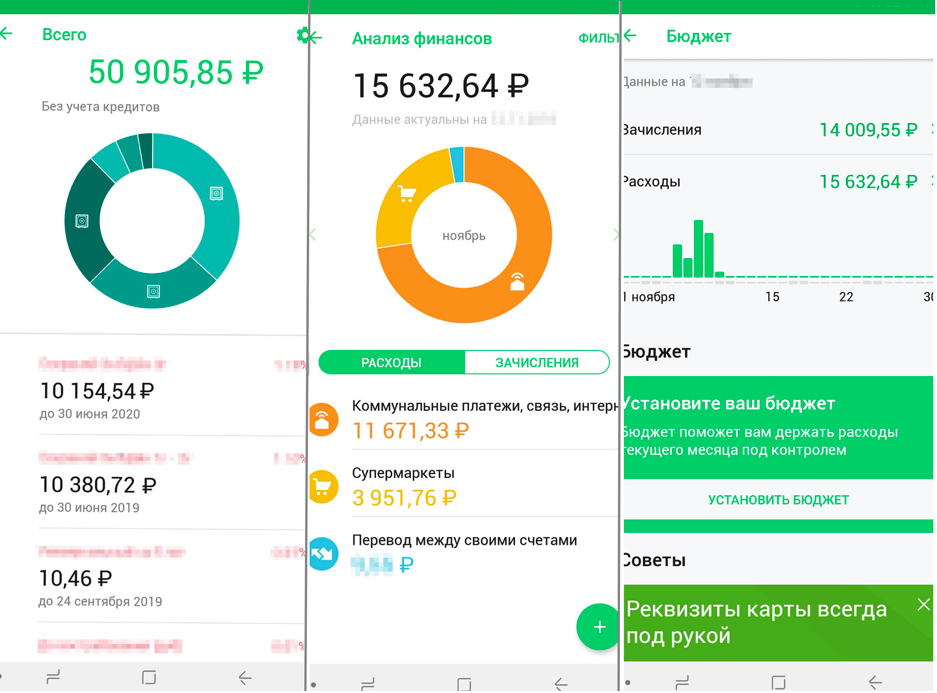
Эффективные, лидирующие на рынке приложений для ведения бюджета в Play Store (Google Play) на территории РФ являются программные решения, взаимодействующие с данными банка, в основном, с помощью парсинга СМС с интеграцией с онлайн-банком. Наглядным примером может послужить «Сбербанк Онлайн» (Рис.3), как яркий представитель систематизации и автоматизации доходов и расходов, основанный на транзакциях по безналичному расчету банковской картой [16].

Рис.3. Пример графического интерфейса приложение «Сбербанк Онлайн».

Для клиентов ПАО Сбербанк, имеющих банковскую карту Сбербанка и подключенный СМС-сервис «Мобильный банк», есть специальный раздел «Мои финансы и бюджет». Он позволяет свести воедино информацию о действующих депозитах клиента, осуществляемых им расходах по банковским картам Сбербанка, а также о накопленных бонусах «Спасибо». Вся информация подаётся в виде графиков и диаграмм. Есть возможность запланировать бюджет на месяц, и приложение будет предупреждать, когда траты приблизятся к заданной планке.

Функционал интернет-банка позволяет также создавать свои собственные категории расходов в дополнение к уже имеющимся, устанавливать расходный бюджет по каждой из категорий. «Научить» банк распознавать новые категории расходов или скорректировать категорию конкретной расходной операции можно как в интернет-банке, так и в мобильном приложении «Сбербанк Онлайн».

Статьи доходов и расходов определяет пользователь. Максимальная совместимость с банковскими картами Сбербанка.

Записи статей доходов и расходов преобладают от СМС, присутствует самостоятельный ввод.

Группировка полученных данных от сообщений предусмотрена, пользователь должен сам настроить категорию для соответствующего магазина.

Ведется точный баланс с точки зрения безналичного расчета, пользователь всегда может получить информацию о текущем балансе на его банковских счетах.

Сбор автоматизации данных автоматизирован посредством считывания СМС предлагаемые возможности автоматического сбора данных;

Уровень автоматизации сравнительно тот же, что и у приложения CoinKeeper, однако есть преимущества в виде быстроты и удобства работы с этим приложением, здесь же предоставляются сервисы банка. Принципы ведения личного бюджета здесь не соответствуют необходимым требованиям, они лишь предоставляют сжатые данные и общее представление о потраченных/полученных суммах и о текущем балансе.

**Мобильные приложения систем распознавания чеков как фотоснимка**

В настоящее время системы распознавания чеков не имеют широкого распространения в силу того, что большинство из них не обеспечивает должного качества работы. Это обусловлено тем, что не существует стандартного шаблона чека, и каждое предприятие, при печати на чеке обязательных данных может самостоятельно выбрать для себя его внешний вид.

Некоторые проекты, например, BilliBox и «33 бакса», закрылись, не оправдав ожиданий пользователей. В проекте BilliBox время распознавания чека занимало порядка 3-х и более часов. При этом изображения пользователей загружались на сервер и там обрабатывались. Кроме большего времени распознавания недостатком данной системы было то, что в некоторых случаях выдавалось следующее сообщение об ошибке: «К сожалению, Ваше фото не содержит изображение розничного чека. Пожалуйста, используйте BilliBox по прямому назначению, для всего остального есть Инстаграм». Программа «33 бакса» являлась стартапом, и этот проект даже не удалось запустить, поскольку был сделан вывод о нецелесообразности ее использования.

Время распознавания чеков напрямую зависит от выбора алгоритма распознавания и от базы чеков, так как чеки из разных магазинов чаще всего различаются и надо найти общие параметры, по которым и будет осуществляться анализ. Приложения такого типа опытным путем показали свою несостоятельность, не позволяют получить нужный, качественный результат в быстрые сроки в максимально безотказном порядке. Развитием такового типа программ является «FinPix», использующая технологию распознавания QR-кода.

**Мобильное приложение «FinPix»**

На момент первичного исследования мобильных приложений (июнь 2018 года) не существовало приложений по использованию QR кода, однако уже на момент сбора информации по исследуемой теме (февраль 2019 года), было выпущено обновление с поддержкой сканера QR кода в приложении для смартфонов с операционной системой Android “FinPix” (Рис.4) [17].

Изначально программа использовала алгоритмы распознавания текста на чеке. Является условно бесплатным продуктом: ранее распознавание текста производилось на сервере, определенное количество чеков можно отсканировать бесплатно каждый месяц. Теперь используется распознавание QR кода. Стоит сразу сказать, что сканирование чека в этом приложении происходит медленно и часто не получается. Отзывы пользователей и свой личный опыт говорят о ненадлежащей реализации сканера QR кода.

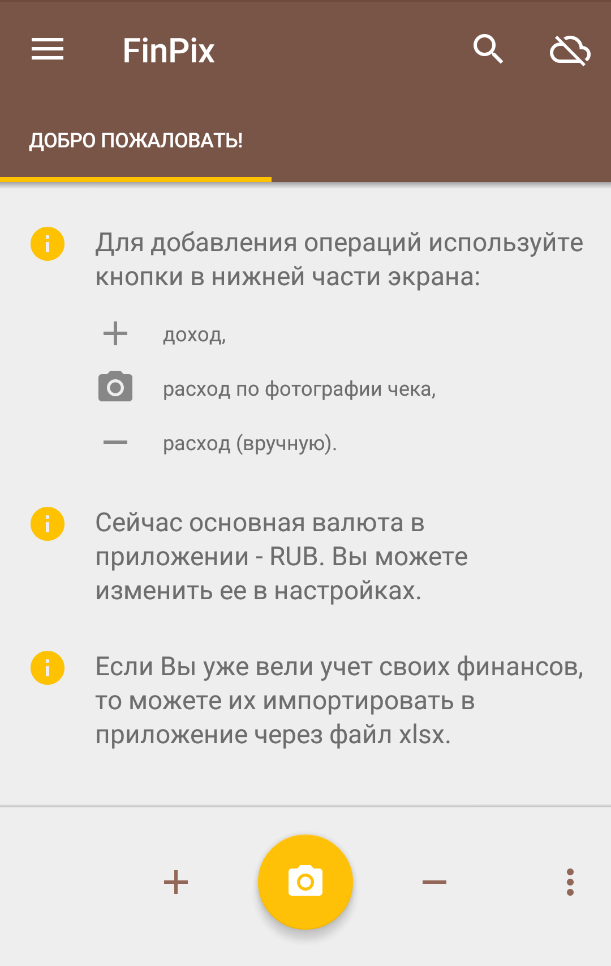
Способ определения статей доходов и расходов: самостоятельный. Способ записи статей доходов и расходов: ручной, с помощью QR-кода. Расчет разницы между доходом и расходом: ведется баланс. Возможность получения детальной информации по каждой операции: присутствует, возможность в ручном режиме заполнять информацию точнее и полнее, чем в аналогах. Возможность получения информации в виде графиков, диаграмм, схем: присутствует. Наличие автоматизации сбора данных, предлагаемые возможности автоматического сбора данных: присутствует (берется итоговая сумма чека).

Рис.4. Главная страница приложения «FinPix».

Простота приложения, удобство в использовании, обучение при первичном использовании, практичный дизайн, удобство в использовании, удобно формировать категории товаров, однако есть ограничение на их количество. Медленное сканирование и частые ошибки при сканировании являются критичным недостатком в работе с этим приложением.

Анализ источников по теме работы показывает, что направление мобильных приложений для ведения личного бюджета на сегодняшний день обладает большой актуальностью и потенциалом. Целый ряд научно-исследовательских работ посвящен тому, как применение мобильных технологий позволяет значительно улучшить качество жизни пользователя, а также предоставляет возможность прогнозирования своего материального благополучия. Исходя из анализа конкурентов, можно сказать, что на данный момент только один из аналогов реализует функцию сканера QR-кода на чеке. Это говорит о том, что решение данной задачи является возможным и актуальным.

## 1.3. Средства разработки

Современное сообщество Android разработчиков использует язык Kotlin в качестве основного. Он является оптимизированной и адаптированной версией Java для разработки мобильных приложений. В процессе ознакомления, был выбран менее популярный объектно-ориентированный кроссплатформенный язык Dart и платформа для разработки Flutter, разрабатываемые Google [11]. Принципиальные отличия:

1. Отсутствие горячего обновления (изменение некоторых параметров без повтороной сборки проекта);
2. Не требует значительное время на создание графических элементов их отображение и размещение, графические элементы представлены объектами класса;
3. Простой C-образный синтаксис, где графическая составляющая представляется как каскад вложенных классов (такой конструктив можно видеть в React Native);
4. Большое количество пакетов (библиотек, готовых классов);
5. Поддержка облачных сервисов Google;

Эти отличительные преимущества стали главной причиной выбора языка программирования Dart и платформы Flutter.

Android SDK является бескомпромиссной полной средой разработки мобильных приложений для устройств с ОС Android [20]. Платформа Flutter использует средства Android SDK для создания пакета приложения.

Для удобства написания кода, его отладки и выполнения использовался текстовый редактор Visual Studio Code с расширениями, предоставляемыми Google (подсветка синтаксиса, подсказки по базовым методам и классам) [21].

В качестве сервиса, предоставляющего облачную базу данных использовался Firestore – NoSQL БД, предоставляемая Firebase – подразделением Google [22]. Данный сервис является лучшим, поскольку предопределен выбором языка и возможностями бесплатного тарифа.   
 Данные технологии находятся на стадии развития, однако представляют основные функциональные возможности и не имеют значительных недостатков.

# 2. ПРОЕКТИРОВАНИЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ

## 2.1. Определение требований к проектируемому приложению

Система будет представлять собой мобильное приложение, обеспечивающее сбор, анализ и хранение данных из чеков. С помощью мобильного приложения, пользователь сможет сканировать QR-код чека, добавлять категории для товаров из чеков.

Исходя из полученных из QR-кода данных, приложение составляет смету о произведенных покупках, подсчитывает текущий баланс пользователя на основе ранее введенных им доходах.

Для использования ПО пользователю потребуется устройство с операционной системой Android 5.1 и выше.

Платформа будет представлять собой клиент-серверное решение вида мобильное приложение-облачное хранилище, для работы которого доступ в интернет будет требоваться только для синхронизации и передачи данных.

**Функциональные требования к проектируемой системе**

Можно определить следующий набор функциональных требований к системе.

1. Система должна обеспечивать регистрацию и авторизацию пользователя в приложении.

2. Система должна позволять пользователю сканировать QR-код.

3. Система должна получать данные от ФНС.

4. Система должна позволять пользователю добавлять данные о доходах.

5. Система должна позволять пользователю создавать категории товаров для их сортировки, сбора статистики.

6. Система должна хранить данные, полученные при сканировании QR-кода.

7. Система должна отображать все имеющиеся пользовательские данные о бюджете в наглядном виде.

8. Система должна рассчитывать баланс доходов и расходов пользователя.

9. Система должна отображать потраченные средства на определенные категории товаров.

10. Пользователь должен иметь возможность определять период времени, за который выводится статистический отчет.

11. Пользователь должен иметь возможность вручную вводить позиции чеков.

**Нефункциональные требования к проектируемой системе**

1. Система должна быть написана на языке Dart.

2. Система должна быть доступна на устройствах с операционной системой Android 5.1 и выше.

3. Система должна хранить анонимные данные пользователей на внешнем сервере с базой данных.

## 2.2. Разработка диаграммы вариантов использования

Диаграмма прецедентов (вариантов использования) отражает отношения между актерами и прецедентами системы и позволяет описать систему на концептуальном уровне. Прецедент – возможность моделируемой системы, часть ее функциональности, благодаря которой актер может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов ее использования и описывает типичный способ взаимодействия пользователя с системой. Диаграмма прецедентов обычно применяется для спецификации внешних требований к системе.

В ходе проектирования был выделен следующий актер.

*Пользователь*– пользователь приложения, которому доступна возможность использования всех его функций.

На основе функциональных требований к мобильному приложению была создана диаграмма прецедентов, которая представлена на рис. 5.

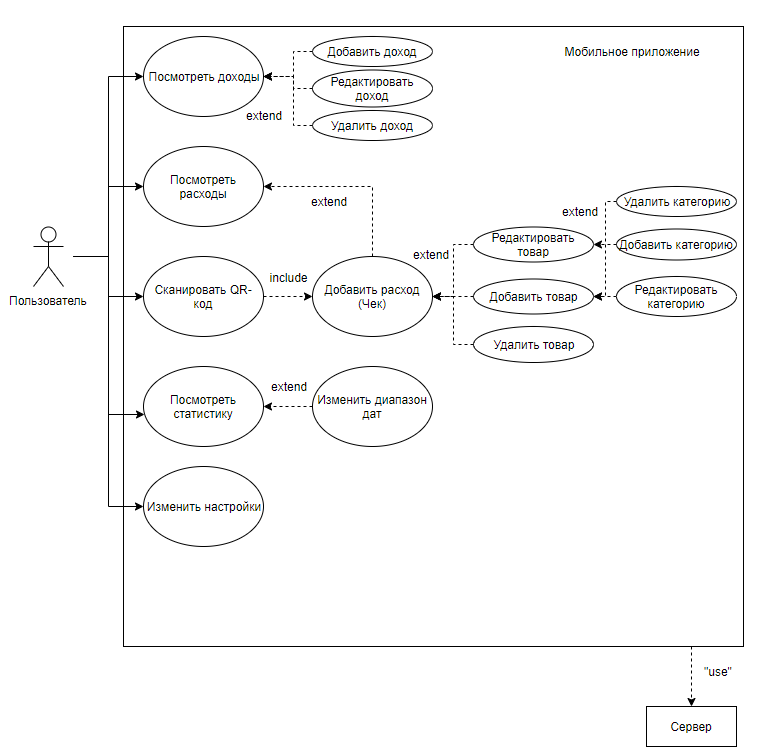


Рис. 5. Диаграмма прецедентов

*Посмотреть доходы –* посмотреть список полученных средств пользователем приложения.

*Добавить доход* – добавить дату и сумму дохода пользователя.

*Редактировать доход* – изменить дату, сумму дохода.

*Удалить доход* – удалить данные о доходе.

*Посмотреть расходы -* посмотреть список купленных товаров их наименование и стоимость.

*Добавить расход* – добавить вручную или с помощью сканера информацию о чеке: дата, наименование, цена.

*Сканировать QR-код* – сканировать QR-код чека и получить данные чека.

*Добавить товар* – дополнить список товаров вручную (наименование и цена, присвоить категорию товара).

*Редактировать товар* – изменить данные конкретного товара из списка расходов.

*Удалить товар* – удалить конкретный товар.

*Добавить категорию* – создать категорию товара.

*Редактировать категорию* – изменить параметры категории товара.

*Удалить категорию* – удалить категорию товара.

*Посмотреть статистику* – посмотреть статистику по расходам за определенное время.

*Изменить настройки* – изменить настройки приложения.

Все данные, имеющееся в приложения – информация о товарах и магазинах – постоянно актуализируются с сервера. Также в ходе работы приложения на сервер отправляются данные полученные в процессе использования.

## 2.3. Проектирование интерфейса приложения

Исходя из функциональных требований, можно определить интерфейсы мобильного приложения (Рис.6):

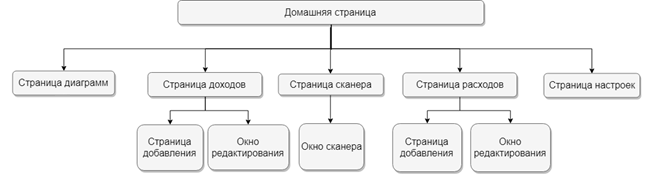


Рис.6 Схема интерфейсов приложения.

1. Домашняя страница предполагает навигацию по основным функциям, отображение краткой информации (доход и расход за текущий месяц, недавние действия пользователя);
2. Страница диаграмм подразумевает отображение круговой диаграммы с пользовательскими категориями и затратами на каждую из них.
   1. Страница графиков по категориям отображает информацию каждой категории отдельно, где можно определять исследуемые временные границы (еженедельно, ежемесячно, определенные даты).
3. Страница доходов позволяет увидеть историю пользовательских доходов, отображает дату поступления средств, сумму, возможность редактировать и удалять. Также позволяет увидеть сумму доходов за месяц. Здесь можно добавить графу доходов, редактировать, удалять запись
   1. Страница добавления доходов представляет собой страницу с полями для ввода: дата, наименование, сумма.
   2. Окно редактирования – окно предоставляющее возможность изменения записи.
4. Страница сканера – реализует графическую оболочку для окна сканера.
   1. Окно сканера представляет отображение для видеопотока камеры устройства и реализует функцию чтения QR-кода.
5. Страница расходов позволяет увидеть историю пользовательских расходов, группирует товары по дате приобретения, позволяет редактировать и удалять каждый элемент списка. Здесь можно добавить запись о приобретенных товарах, а также редактировать, удалять запись.
   1. Страница добавления доходов представляет собой страницу с полями для ввода: дата, наименование, сумма.
   2. Окно редактирования - окно предоставляющее возможность изменения записи.
6. Страница настроек позволяет настраивать параметры приложения.

## 2.4. Выбор средств и проектирование базы данных приложения

Для платформы Flutter разработаны библиотеки для взаимодействия с такими СУБД как MongoDB, SQLite. Эти библиотеки реализуют ORM и «raw-запросы» позволяют интерпретировать запросы языка Dart, в запросы СУБД. Реализация базы данных с помощью подобных библиотек практичны, но предполагают локальное хранилище, создание собственных функций для взаимодействия базы данных.

Практичным решением считается Firebase, реализующий BaaS [14,22]. Основной сервис — облачная СУБД класса NoSQL, позволяющая разработчикам приложений хранить и синхронизировать данные между несколькими клиентами. Поддержаны особенности интеграции с приложениями под операционные системы Android и iOS. Управление настройками, создание коллекций. Хранение данных в облаке от Google, присутствует локальное хранение данных. Реализована гибкая интеграция средств платформы Flutter и Firebase: возможность эффективно использовать StreamBuilder: получая элементы из базы данных, создается список виджетов (в этом случае базовые элементы списка/таблицы), с которыми можно взаимодействовать (сортировать/редактировать/удалять), то есть построение графического отображения в соответствии с базой данных и условиями построения (можно здесь же реализовать первичную выборку, сортировку) основанном на идентификаторах элементов и их содержимом.

Возможность анализа числа запросов к облаку, что позволяет определить эффективность приложения и его оптимальности в работе с данными. Существует потенциальная возможность масштабирования базы данных, что трудно выполнимо в SQL базах, возможность хранения файлов (чеков в PDF формате), поддержка средств аутентификации пользователя, гибкая настройка доступа к данным и безопасность хранения (через конфигурационный файл). Всё это предопределило выбор Firebase как основы для ее использования в реализации приложения.

База данных (Рис.7) содержит 2 коллекции: категории товаров(category), где содержатся данные о пользовательских категориях(name) и дата(date), с помощью которой организуется хранение в двух коллекциях: чек(receipt) и доход(earning). Чек содержит запись о покупке: товар(name), сумма(sum), ссылка на дату(date), ссылка на категорию товара(category). Доход(earning) содержит ссылку на дату поступления средств(date), и сумму(sum).

C:\Users\hydra\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2.png

Рис 7. Схема базы данных.

# 3. РЕАЛИЗАЦИЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ

## 3.1. Компоненты и структура проекта

Проект мобильного приложения содержит 5 основных компонентов, взаимодействие которых представлено на рис. 8.

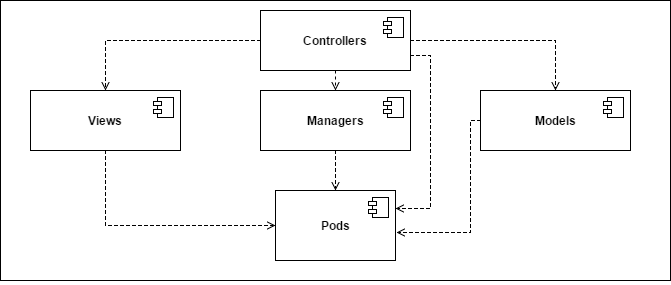


Рис. 8. Диаграмма компонентов

Проект содержит 5 компонентов – Models, Controllers, Views, Managers и Pods. Компонент Controllers содержит все классы контроллеров системы. Контроллеры реализуют логику экранов мобильного приложения.

Компонент Views содержит классы отдельных элементов экрана мобильного приложения (ячейка таблицы, поле поиска и т.д.). Часть логики, связанная с этими элементами, перенесена в их классы, чтобы разгрузить классы контроллеров.

Компонент Models включает в себя реализацию классов моделей. Каждый класс, хранимый в каталоге Models, реализует отдельную структуру данных.

Компонент Managers содержит классы для связи мобильного приложения с сервером. Для удобства связь отдельного модуля приложения реализована в классе отдельного менеджера.

Компонент Pods включает в себя все сторонние библиотеки, подключенные в проекте.

Также стоит отметить файлы, которые нельзя отнести к конкретному компоненту, но они являются неотъемлемой частью проекта. Файл xcDataModel содержит описание структур данных, используемых приложением. Класс RequestManager содержит реализацию механики отправки запросов на сервер. К классу Request Manager обращаются все классы из компонента Managers. Файл storyboard содержит структуру экранов приложения и связь их друг с другом.

## 3.2. Реализация базы данных приложения

Настройка базы данных происходит через веб-интерфейс, обращающийся к облачному сервису, там же определяются коллекции и первые элементы этих коллекций. Ниже представлен пример инициализации и использования коллекции, связанной с чеком (Рис. 9). Метод getData(base) – предоставляет доступ к данным коллекции base, возвращает Stream<QuerySnapshot> - список элементов коллекции. Это классический пример реализации функции чтения с помощью встроенных средств Firebase.

|  |  |
| --- | --- |
| |  | | --- | | getData(base) async {  return await Firestore  .instance  .collection('user')  .document('date')  .collection(base)  .snapshots();  } | |

Рис.9. Пример реализации списка чеков.

# 3.3. Реализация функции отображения доходов и потраченных средств

Отображение доходов и расходов происходит с помощью StreamBuilder (рис.10). Сначала определяется коллекция, из которой нужно извлекать данные для построения, затем описание взаимодействия с полученными данными (сортировка, действие во время загрузки данных, отрисовка).

|  |
| --- |
| return StreamBuilder<QuerySnapshot>(  stream: Firestore.instance  .collection('user')  .document('date')  .collection('check')  .snapshots(),  builder: (BuildContext context, AsyncSnapshot<QuerySnapshot> snapshot) {  if (!snapshot.hasData) return Loading();  List<DocumentSnapshot> documents = snapshot.data.documents;  documents.sort((a, b) => b.data['date'].compareTo(a.data['date']));  return Scaffold(  backgroundColor: Colors.white,  appBar: AppBar(  backgroundColor: Color(0xFF00897B),  title: Text('Расходы'),  actions: <Widget>[  IconButton(  icon: Icon(Icons.add),  onPressed: () {  newProduct(context);  },  ),  ],  ),  body: ExpansionTileList(  documents: documents,  ),  );  },  );  }  } |

Рис. 10. Фрагмент кода функции AddSpending, реализующий отображение списка товаров.

Извлекая данные из базы данных, происходит построение списка по датам, в элементах которого создаются свои списки (рис.10). Каждый элемент содержит информацию, кнопки редактирования и удаления. Каждый отдельный раздел (по дате), содержит клавишу «Добавить товар», также есть клавиша на панели сверху, для добавления товара в новую дату (которой нет или пользователь не хочет искать или не может найти интересующую дату).

# 3.4. Реализация CRUD - функций

CRUD-функции (Create, Read, Update, Delete) – это четыре базовые функции для работы с базами данных, реализующие создание, чтение, обновление(редактирование), удаление данных.

Ниже представлен пример реализации функции создания поля (Рис.11) в облачном хранилище Firestore. Входные параметры: передаваемые данные (data) и путь в хранилище (имя коллекции, base).

|  |
| --- |
| addData(data, base) {  Firestore.instance.runTransaction((Transaction crudTransaction) {  CollectionReference reference =  Firestore.instance.collection('user').document('date').colletion(base);  reference.add(data);  });  } |

Рис.11. Реализация функции создания записи в базе данных.

Асинхронная функция getData (Рис.12) реализует получение (чтение) данных на запрос пользователя. Передаваемый параметр – имя коллекции или определенный документ коллекции (base).

|  |
| --- |
| Future getData(base) async {  return await Firestore.instance.collection('user').document('date').collection(base).snapshots();  } |

Рис.12. Реализация функции чтения данных.

Функция обновления данных (Рис.13) представляет собой запрос на изменение конкретного документа коллекции по его идентификатору (docId), новыми данными (newValues) и имя коллекции (base).

|  |
| --- |
| updateData(docId, newValues, base) {  Firestore.instance  .collection('user').document('date')  .collection(base).document(docId).updateData(newValues)  .catchError((e) { print(e); });  } |

Рис.13. Реализация функции обновления данных.

Функция удаления данных (Рис.14) использует переданный идентификатор документа (docId) коллекции, для создания запроса на удаление этого документа в Firestore.

|  |
| --- |
| deleteData(docId, base) {  Firestore.instance  .collection('user')  .document('date')  .collection(base)  .document(docId)  .delete()  .catchError((e) {  print(e);  });  } |

Рис.14. Реализация функции удаления данных.

# 3.5. Реализация запросов к операторам фискальных данных

На примере запроса к ОФД «ТаксКом» отображен типичная строка запроса (Рис.15), где передаваемые параметры fp – фискальный, fpd – фискальный идентификатор, s – сумма чека.

|  |
| --- |
| String taxcom =  'https://receipt.taxcom.ru/v01/show?fp=$fpd&s=${receipData.sum}&sf=False&sfn=False'; |

Рис. 15. Пример адреса для запроса к ОФД «Такском»

Ниже представлен фрагмент функции запроса (Рис.16), использующий адрес, описанный выше. Используется библиотека http [23], для реализации запросов. После получения ответа, данные отбираются и преобразуются в удобные для обработки и хранения списки, затем вызывается переход к странице с параметрами чека, где пользователь должен отнести товар к определенной категории.

|  |
| --- |
| await http.get(taxcom, headers: {"Content-Type": "application/json"}).then((response) {  var document = parse(response.body);  var nameList = document.getElementsByClassName('value receipt-value-1030');  var sumList = document.getElementsByClassName('value receipt-value-1043');  RegExp exp = new RegExp(r"(\s+)");  List<ReceiptData> receiptList = [];  List<String> buff = [], names = [];  List<double> sum = [];  ...  for (var i = 0; i < nameList.length; i++) {  ReceiptData data = new ReceiptData(names.elementAt(0),sum.elementAt(0));  receiptList.add(data);  names.removeAt(0);  sum.removeAt(0);  }  Navigator.push( context,  MaterialPageRoute(  builder: (context) => QRSpending(receiptList: receiptList, receiptData: receiptData,  ),  ),  );  }  ); |

Рис. 16. Фрагмент реализации запроса к ОФД «Такском».

# 3.7. Реализация функции сканирования QR-кода чека

Функция сканирования QR-кода реализована с помощью библиотеки qr\_mobile\_vision [19] для платформы Flutter. Ниже представлена функция (Рис. 17), вызывающая метод scan из библиотеки, обрабатывающая события, которые могут произойти, если сканирование произошло с ошибками.

|  |
| --- |
| Future<bool> scanner(BuildContext context, String qrEncryption) async {  QrCamera qr;  return showDialog(  context: context,  barrierDismissible: false,  builder: (BuildContext context) => Scaffold(  backgroundColor: Colors.white,  appBar: AppBar(  title: Text('Сканирование'),  ),  body: Container(  padding: EdgeInsets.all(8),  child: Column(  mainAxisAlignment: MainAxisAlignment.center,  children: <Widget>[  Row(  mainAxisAlignment: MainAxisAlignment.center,  children: <Widget>[  SizedBox(  width: 300,  height: 300,  child: qr = QrCamera(  qrCodeCallback: (code) {  if (code.isNotEmpty) {  qr.createState().deactivate();  Navigator.pop(context);  }  callback(code);  print(code);  qrEnryption(code);  },  ),  ),  ],  )  ],  ),  ),  ));  } |

Рис.17. Функция \_scanQR().

После вызова метода scan, открывается окно, отображающая видеопоток, поверх которого отображается область сканирования (Рис.18). После успешного чтения QR-кода, окно автоматически закрывается. При неудачной попытке сканирования, камера начнет фокусироваться на объекте заново. Отклонения камеры в плоскостях допустимо до 15-20 градусов от каждой из осей. Важным критерием при сканировании является качество печати, общее состояние бумаги (общая целостность чека, помятость, проколы), при тестировании сканера на группе чеков (50 образцов), было прочитано 47, три из которых изначально бы отбракованы (качество печати не удовлетворительно для человеческого глаза).



Рис.18. Экран сканирования с примером отображаемого изображения камеры.

# 3.8. Реализация страницы статистики расходов по категориям

Страница диаграммы расходов по категориям (Рис. 19) содержит в себе круговую диаграмму (chart), и функциональную кнопку (IconButton), где пользователю предлагается выбрать граничные даты исследуемого временного диапазона (\_initialFirstDate, \_initialLastDate) с помощью функции showDatePicker класса DateRagePicker.

|  |
| --- |
| Scaffold(  backgroundColor: Colors.white,  appBar: AppBar(title: Text('Расходы'), actions: <Widget>[  IconButton(  icon: Icon(Icons.date\_range),  onPressed: () async {  DateTime date = DateTime.now();  DateTime \_initialFirstDate =  new DateTime(date.year, date.month, 1);  DateTime \_initialLastDate =  new DateTime(date.year, date.month + 1, 0);  List<DateTime> picked = await DateRagePicker.showDatePicker(  context: context,  initialFirstDate: \_initialFirstDate,  initialLastDate: \_initialLastDate,  firstDate: new DateTime(2018),  lastDate: new DateTime(DateTime.now().year + 1));  if (picked != null && picked.length == 2) {  print(picked);  setDate(\_initialFirstDate, \_initialLastDate);  }  },  )  ]),  body: chart,  ); |

Рис 19. Фрагмент реализации страницы статистики расходов пользователя.

Ниже представлена реализация диаграммы расходов и список категорий и затрат пользователя по ним (Рис. 20). Объект chart реализуется с помощью библиотеки charts\_flutter [18] и содержит список series.

|  |
| --- |
| var series = [  charts.Series(  colorFn: (Spendings sales, \_) => sales.color,  domainFn: (Spendings sales, \_) => sales.name,  measureFn: (Spendings sales, \_) => sales.sum,  id: 'Sales',  data: categoryList,  labelAccessorFn: (Spendings sales, \_) =>  '${sales.name}: ${sales.sum.toString()}')  ];  var chart = charts.PieChart(  series,  defaultRenderer: charts.ArcRendererConfig(  arcRendererDecorators: [charts.ArcLabelDecorator()]),  ); |

Рис 20. Фрагмент реализации объекта диаграммы и его составляющих.

# 4. ТЕСТИРОВАНИЕ

## 4.1. UI-тестирование, тестирование производительности

Из-за тесной связи логики мобильного приложения с его интерфейсом, unit-тестирование является неэффективным способом тестирования системы, т.к. не может обеспечить достаточный процент покрытия исходного кода. Для автоматизации тестирования Android-приложения использовался сервис Firebase Test Lab, который позволяет, в частности, разрабатывать UI-тесты. Для реализации UI-тестов в фреймворке используется машинный обход всех страниц (при условии достаточности данных для перехода на другую страницу). Ниже представлены результаты тестирования скорости инициализации страницы и загрузки иных страниц (Рис.21). В качестве виртуального аппарата для тестирования используется Google Pixel 2.

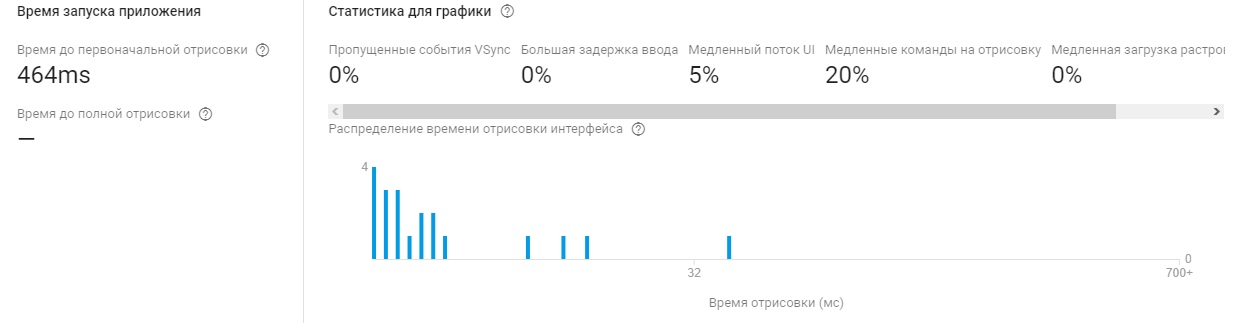


Рис.21. Статистика тестирования отрисовки экрана.

Скорость запуска приложения от нажатия кнопки запуска, до отрисовки экрана: 464мс. При использовании в реальности это значение значительно больше (от 1 до 3 секунд в зависимости от числа приложений, объема кэша, общей производительности устройства). С точки зрения пользователя такие результаты удовлетворительны. Дальнейшие страницы открываются гораздо быстрее.

Также ведется метрика производительности приложения (Рис.22).

Можно увидеть, что инициализация при запуске требует 30% от производительности процессора и чуть больше чем 100 МБ оперативной памяти.

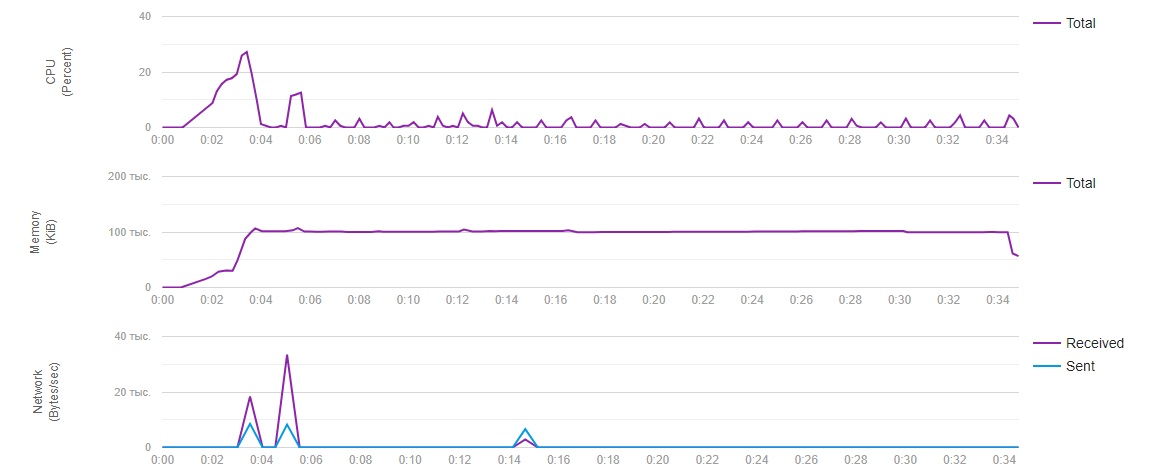


Рис.22. Метрика производительности приложения.

## 4.2. Функциональное тестирование

Функциональное тестирование – это тестирование программного обеспечения в целях проверки реализуемости функциональных требований [4]. Функциональные требования определяют, что именно делает программное обеспечение, какие задачи оно решает. Используя методологию функционального тестирования, проверим работу мобильного приложения.

Тест № 1. Корректное отображение экрана главной страницы приложения

Входные данные: -

Цель: Проверить корректность отображения экрана, панели навигации и сумм затрат/доходов пользователя.

Результат: тест пройден.

Тест № 2. Корректное отображение экрана расходов.

Входные данные: Приложение открыто на экране расходов.

Цель: Проверить корректность перехода на экран расходов пользователя и отображение данных по расходам. Данные должны быть отсортированы от ранней к поздней датам. Разделы должны предоставлять информацию о товарах (Раскрытие вложенности и возможность листать список). Функциональные клавиши (добавления, редактирования, удаления) должны быть доступны.

Результат: тест пройден.

Тест № 3. Корректность работы сканера QR-кода.

Входные данные: Приложение открыто на экране сканера штрих-кода товара

Цель: Проверить корректность работы сканера штрих-кода

Ход проведения:

1. отсканировать штрихкод товара, которого нет в сети магазинов;
2. получить ошибку о том, что такого товара нет в сети магазинах;
3. нажать на кнопку «Оставить отзыв»;
4. отсканировать штрихкод товара, который есть в сети магазинов;
5. убедиться, что приложение открыл экран карточки отсканированного товара.

Результат: тест пройден.

Тест № 5. Корректность выбора магазина.

Входные данные: Приложение открыто на экране общего каталога, город выбран автоматически либо пользователем вручную, магазин не выбран.

Цель: Проверить обновление данных о товарах при выборе магазина.

Ход проведения:

1. открыть экран категории;
2. проверить, что у товаров стоит средняя цена и не показывается количество товаров в магазине;
3. перейти на экран магазинов;
4. выбрать магазин из списка;
5. перейти обратно на экран категории;
6. убедиться, что надпись «средняя цена» исчезла, появилось количество единиц товара в магазине.

Результат: тест пройден.

Тест № 6. Корректность добавления товара в корзину без выбранного магазина.

Входные данные: Приложение открыто на экране общего каталога, город выбран автоматически либо пользователем вручную, магазин не выбран, определение геопозиции включено.

Цель: Проверить корректность добавления товара в корзину в случае, когда магазин не выбран. Должен автоматически выбраться ближайший к пользователю магазин, товар должен добавиться в корзину.

Ход проведения:

1. открыть экран категории;
2. проверить, что у товаров стоит средняя цена и не показывается количество товаров в магазине;
3. нажать на кнопку добавления товара;
4. перейти на экран магазинов;
5. убедиться в том, что выбран ближайший магазин;
6. перейти на экран корзины;
7. убедиться, что выбранный товар был добавлен в корзину.

Результат: тест пройден.

Тест № 7. Невозможность оформления заказа при общей стоимости товаров в корзине менее 500 руб.

Входные данные: Приложение открыто на экране общего каталога, город выбран, магазин выбран.

Цель: Проверить невозможность оформления заказа при общей стоимости товаров в корзине менее 500 руб.

Ход проведения:

1. открыть экран категории;
2. добавить несколько товаров на сумму менее 500 рублей;
3. перейти на экран корзины;
4. убедиться, что посчитанная общая сумма товаров верна;
5. нажать на кнопку «оформить заказ»;
6. проверить появление уведомления о недостаточной общей стоимости товаров в корзине для оформления заказа.

Тест № 8. Возможность оформления заказа пи общей стоимости товаров в корзине от 500 руб.

Входные данные: Приложение открыто на экране общего каталога, город выбран, магазин выбран.

Цель: Проверить возможность оформления заказа пи общей стоимости товаров в корзине от 500 руб.

Ход проведения:

1. открыть экран категории;
2. добавить несколько товаров на сумму от 500 рублей;
3. перейти на экран корзины;
4. убедиться, что посчитанная общая сумма товаров верна;
5. нажать на кнопку «оформить заказ»;
6. проверить, что приложение показало экран оформления заказа.

Результат: тест пройден.

Тест № 9. Корректность проверки данных при оформлении заказа.

Входные данные: Приложение открыто на экране оформлении заказа, все поля пусты.

Цель: Проверить корректность проверки данных при оформлении заказа

Ход проведения:

1. нажать на кнопку «Отправить»;
2. убедиться, что красным подсвечены все поля, кроме комментария;
3. убедиться, что приложение не дает выбрать любую дату и время, кроме часов работы магазина на текущий и следующий день;
4. выбрать корректную дату;
5. ввести некорректный телефон (менее 11 цифр);
6. ввести некорректный email;
7. нажать на кнопку «Отправить»;
8. убедиться, что поля «номер телефона» и «email» подсвечены красным;
9. ввести корректные данные;
10. нажать на кнопку «Отправить»;
11. проверить, что приложение показало экран общего каталога с уведомлением, что заказ успешно сформирован.

Результат: тест пройден.

Тест № 10. Корректность работы отзывов о товаре.

Входные данные: Приложение открыто на карточке товара

Цель: Проверить корректность работы отзывов о товаре

Ход проведения:

1. нажать на кнопку «Отзывы»;
2. убедиться, что появился экран с отзывами о товаре или экран с надписью «нет отзывов», если ранее никто не оставлял отзывы об этом товаре;
3. нажать на кнопку «Оставить отзыв»;
4. ввести некорректные данные;
5. нажать на кнопку «Отправить»;
6. убедиться, что все поля подсвечены красным;
7. ввести корректные данные;
8. нажать на кнопку «Отправить» повторно;
9. убедиться, что приложение вновь показало экран отзывов с уведомлением о том, что ваш отзыв успешно принят;
10. убедиться в том, что кнопка «Оставить отзыв» исчезла для этого товара;
11. через несколько дней убедиться, что отзыв о товаре появился в списке после прохождения модерации.

Результат: тест пройден.

## 4.3. Интеграционное тестирование

Интеграционное тестирование – полная проверка программного продукта после его сборки с целью выявления ошибок, возникающих в процессе интеграции программных модулей или компонентов [4].

Собранный проект был проверен на виртуальных устройствах с различными характеристиками экрана (проверяется некорректность отображения элементов интерфейса): 1280х768px, 1280x800px, 1980x1080px, 2160x1080px; версиями: Android 5.1.1, Android 6, Android 7, Android 8. Ошибок не было выявлено, все элементы на всех разрешениях отобразились корректно. В ходе проверки на различных версиях операционной системы также не было выявлено ошибок, мобильное приложение работает корректно на всех поддерживаемых версиях ОС.

Все скриншоты приложения представлены в приложении.

## 4.4. Юзабилити-тестирование

Интеграционное тестирование – полная проверка программного продукта после его сборки с целью выявления ошибок, возникающих в процессе интеграции программных модулей или компонентов [4].

# ЗАКЛЮЧЕНИЕ

Роль мобильных приложений в повседневной жизни растет постоянно. Многие люди пользуются несколькими десятками приложений ежедневно. Практически каждый день выпускаются новые мобильные приложения.

Целью работы являлась разработка мобильного приложения для получения данных чека для составления бюджета пользователя. Для достижения данной цели были решены следующие задачи:

1. произведена постановка задачи;
2. произведен анализ литературы и смежных проектов, связанных с ведением личного бюджета;
3. изучены современные платформы и средства разработки для операционной системы Android;
4. определены требования к системе ведения бюджета и разработаны варианты ее использования;
5. разработана архитектура базы данных;
6. разработана архитектура мобильного приложения для ведения личного бюджета;
7. разработана схема взаимодействия пользователя с интерфейсом приложения;
8. реализовано мобильное приложение для устройств с операционной системой Android.
9. протестировано мобильное приложение.

Все поставленные задачи были решены, цель достигнута.

В перспективе планируется реализовать следующие возможности:

* расширить процент чеков, по которым можно получить детальную информацию;
* разработать интеллектуальный алгоритм подбора категории каждому товару.

# ЛИТЕРАТУРА

1. App Store. [Электронный ресурс] URL: https://itunes.apple.com/ru/genre/ios/id36?mt=8 (дата обращения: 01.05.2017).
2. Chris Buckett, Dart in action
3. Balbaert I. - Dart Cookbook – 2014
4. Kaner Cem, Falk Jack, Nguyen Hung Quoc. Testing Computer Software. – USA: Wiley Computer Publishing, 1999. – 479 p.
5. Понявина, М. Б., Новые нестандартные методы использования QR-кодов в маркетинговой политике современных компаний // Экономика и современный менеджмент. — 2014. — С. 82-86.
6. Коркин, Р. И. Финансовая грамотность населения как фактор обеспечения экономической безопасности государства // Теория и практика общественного развития. — 2017. — № 12.
7. Санникова Л.В., Сичинава З.И. МОБИЛЬНОЕ ПРИЛОЖЕНИЕ ДЛЯ АВТОМАТИЗАЦИИ СЕМЕЙНОГО БЮДЖЕТА// Инновационная наука. — 2017. — № 5. – С. 124-126.
8. Кирпичников А.П., Ляшева С.А., Пекерман И.Л., Шлеймович М.П. Мобильная система учета расходов // Вестник казанского технологического университета. — 2014. — №24. – С. 331-334.
9. Перевощикова С.М. Онлайн-кассы – это новые возможности контроля за оборотом розничной торговли или новые проблемы предпринимателей? // Вектор экономики. — 2016. — №5. – с. 9.
10. Лашманова Ю.Ю., Свиненкова А.Д., Подгайкова К.С. Внедрение онлайн-касс // International Innovation Research. – 2017. – C. 194 – 197.
11. Рябоконь О.С., Кукарцев В.В. Новый язык структурного веб-программирования Dart // Актуальные проблемы Авиации и Космонавтики. - 2013. – № 9. – С. 436-437.
12. Чернова М.В., Моделирование семейного бюджета // Финансы и кредит. – 2011. – №17. – С. 60-63.
13. Кирюшкина А.Н., Полтева Т.В., Персональное финансовое планирование // Карельский научный журнал. – 2017. – №4. – C. 185-188.
14. Тимошенко Н.С., Вдовых П.Е., Кутумбаев Р.Е., Хоркуш А.В. Создание базы данных при помощи сервиса Firebase // Инновационные технологии в образовании и науке. – 2017. – С. 78-81.
15. CoinKeeper. [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=com.disrapp.coinkeeper.material&hl=ru (дата обращения: 10.02.2019).
16. Сбербанк Онлайн. [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=ru.sberbankmobile (дата обращения: 10.02.2019).
17. FinPix. [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=com.finpix.app (дата обращения: 10.02.2019).
18. Библиотека визуализации данных Charts\_flutter. [Электронный ресурс] URL: https://pub.dartlang.org/packages/charts\_flutter (дата обращения: 27.03.2019).
19. Библиотека сканера QR-кода qr\_mobile\_vision. [Электронный ресурс] URL: https://pub.dartlang.org/packages/qr\_mobile\_vision (дата обращения: 04.04.2019).
20. Документация Android SDK [Электронный ресурс] URL: https://developer.android.com/studio (дата обращения: 23.01.2019)
21. Расширения для Visual Studio Code [Электронный ресурс] URL: https://medium.com/flutter-community/flutter-visual-studio-code-shortcuts-for-fast-and-efficient-development-7235bc6c3b7d (дата обращения: 02.02.2019).
22. Firebase Guides [Электронный ресурс] URL: https://firebase.google.com/docs/guides (дата обращения: 25.02.2019).
23. Библиотека http запросов [Электронный ресурс] URL: https://pub.dartlang.org/packages/http (дата обращения: 15.03.2019).

# ПРИЛОЖЕНИЕ



Рис. 1. Главная страница

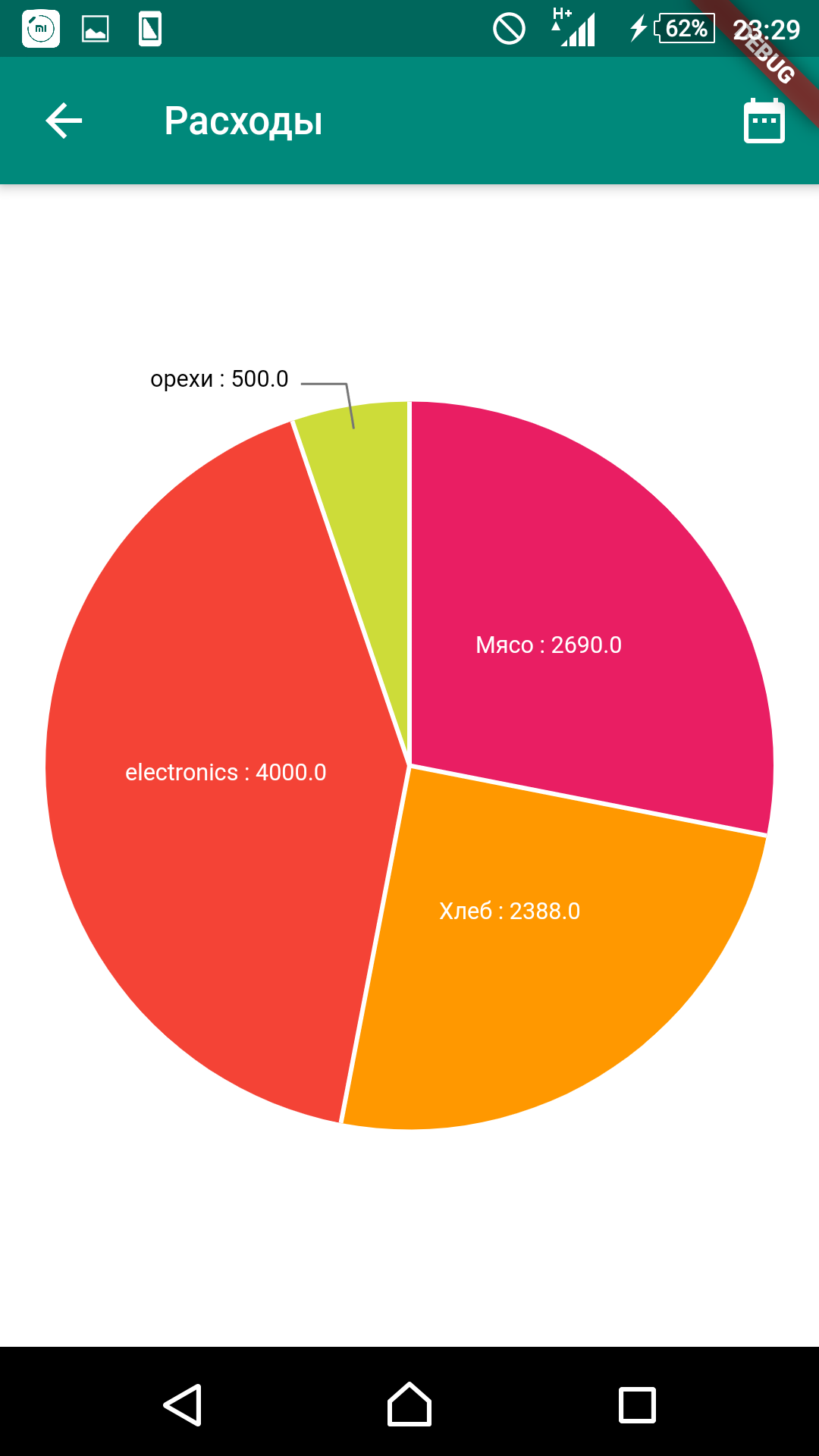


Рис. 2. Статистика расходов

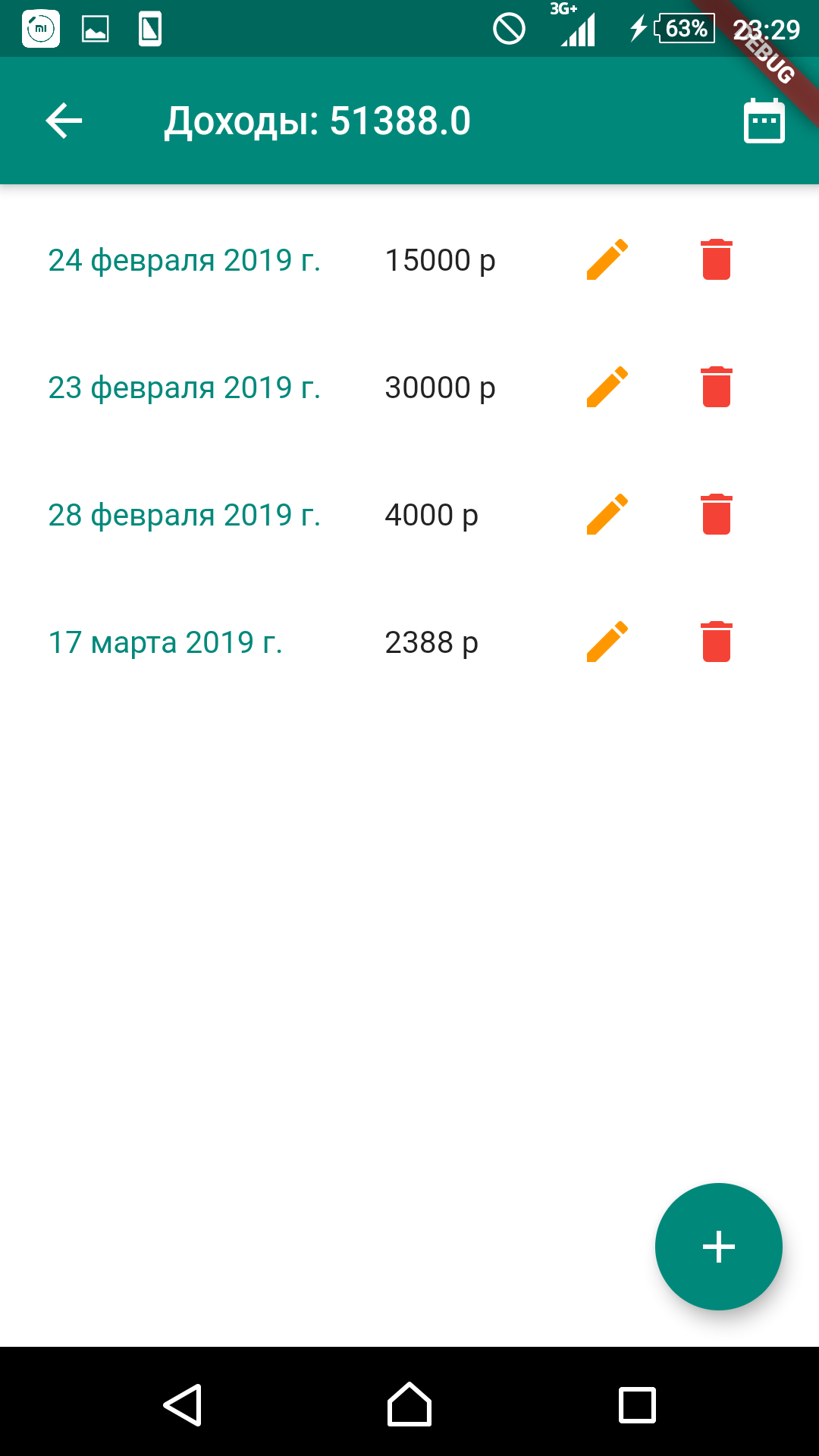


Рис. 3. Страница доходов

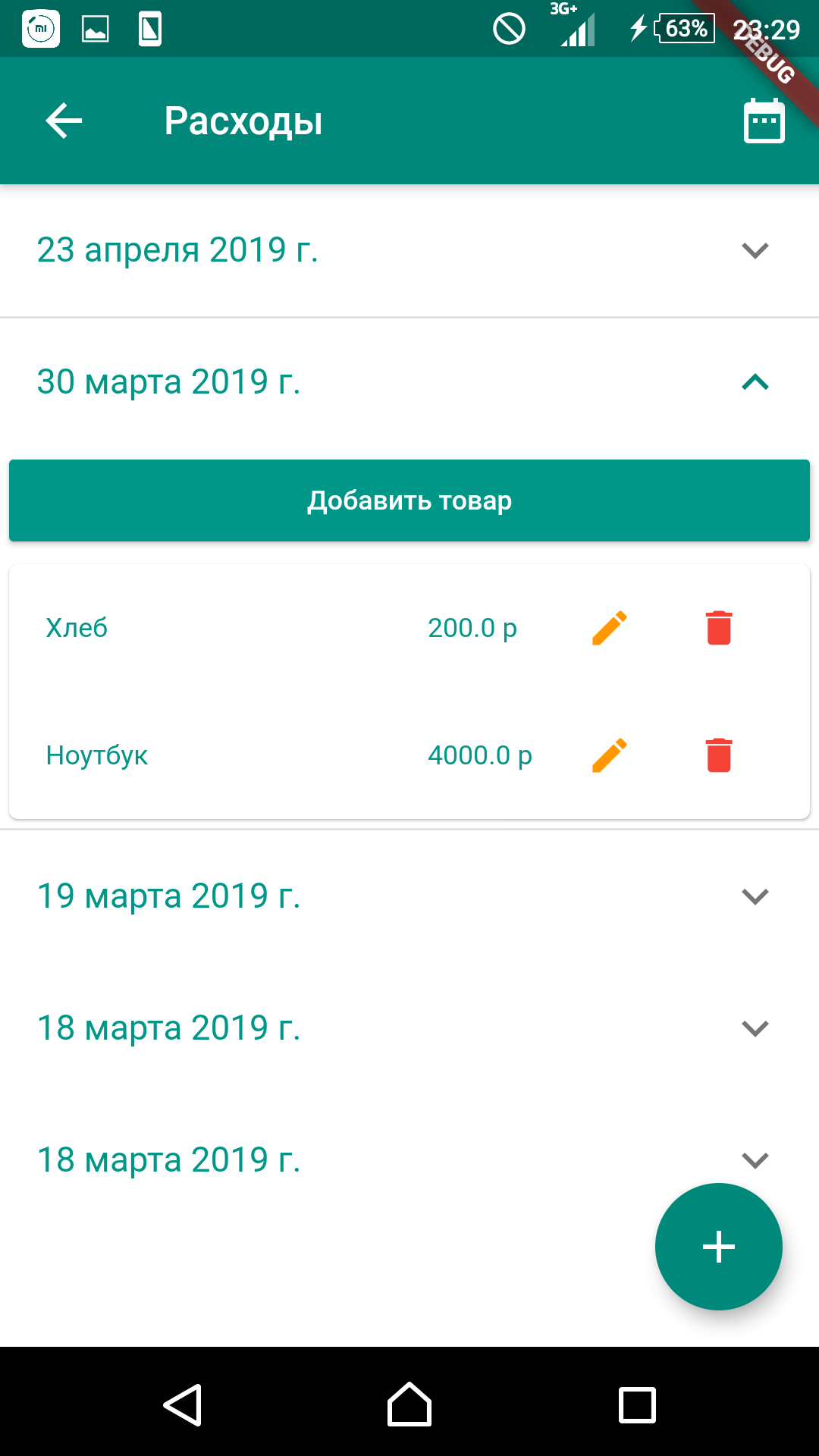


Рис. 4. Страница расходов

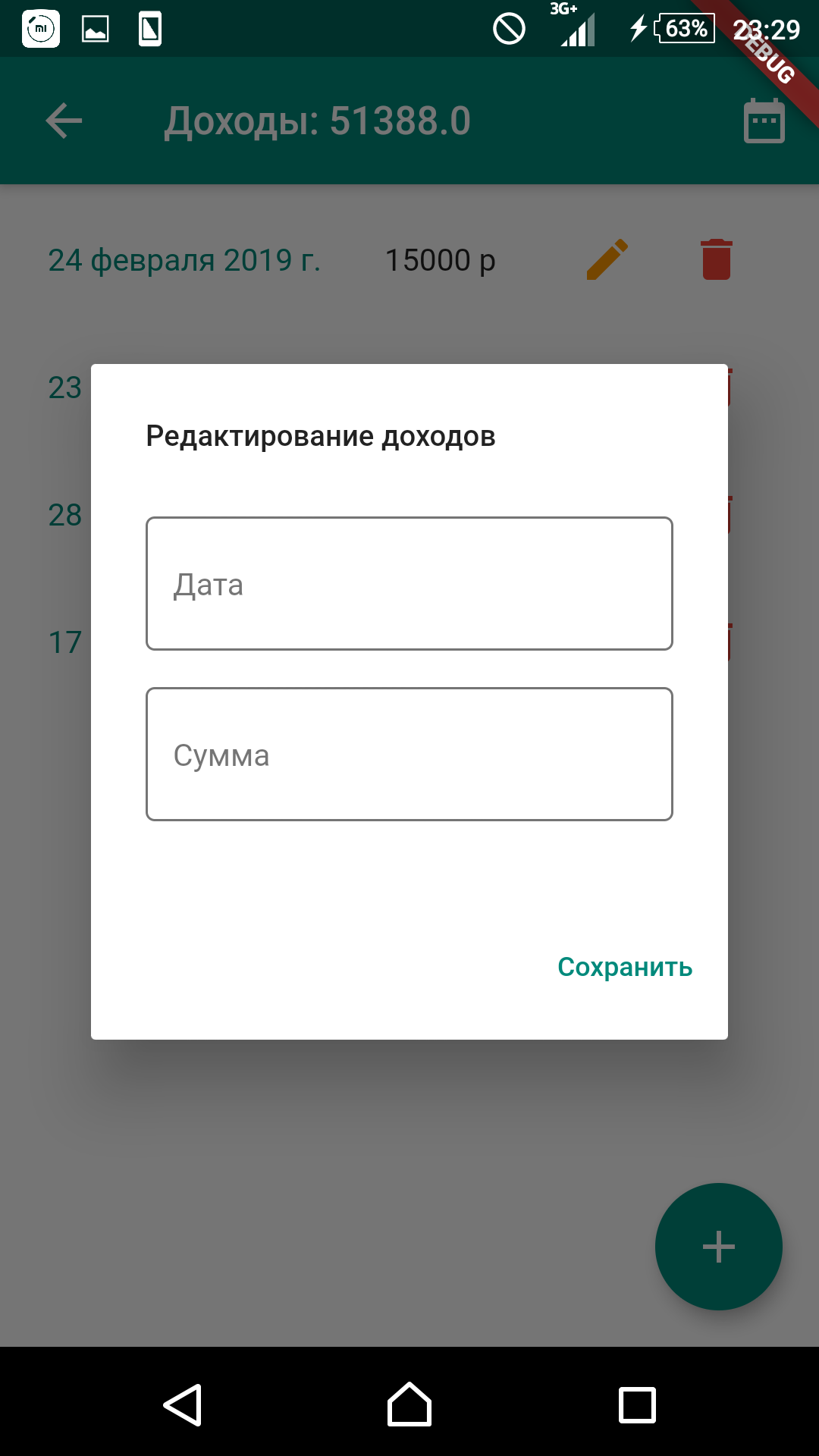


Рис. 5. Редактирование доходов

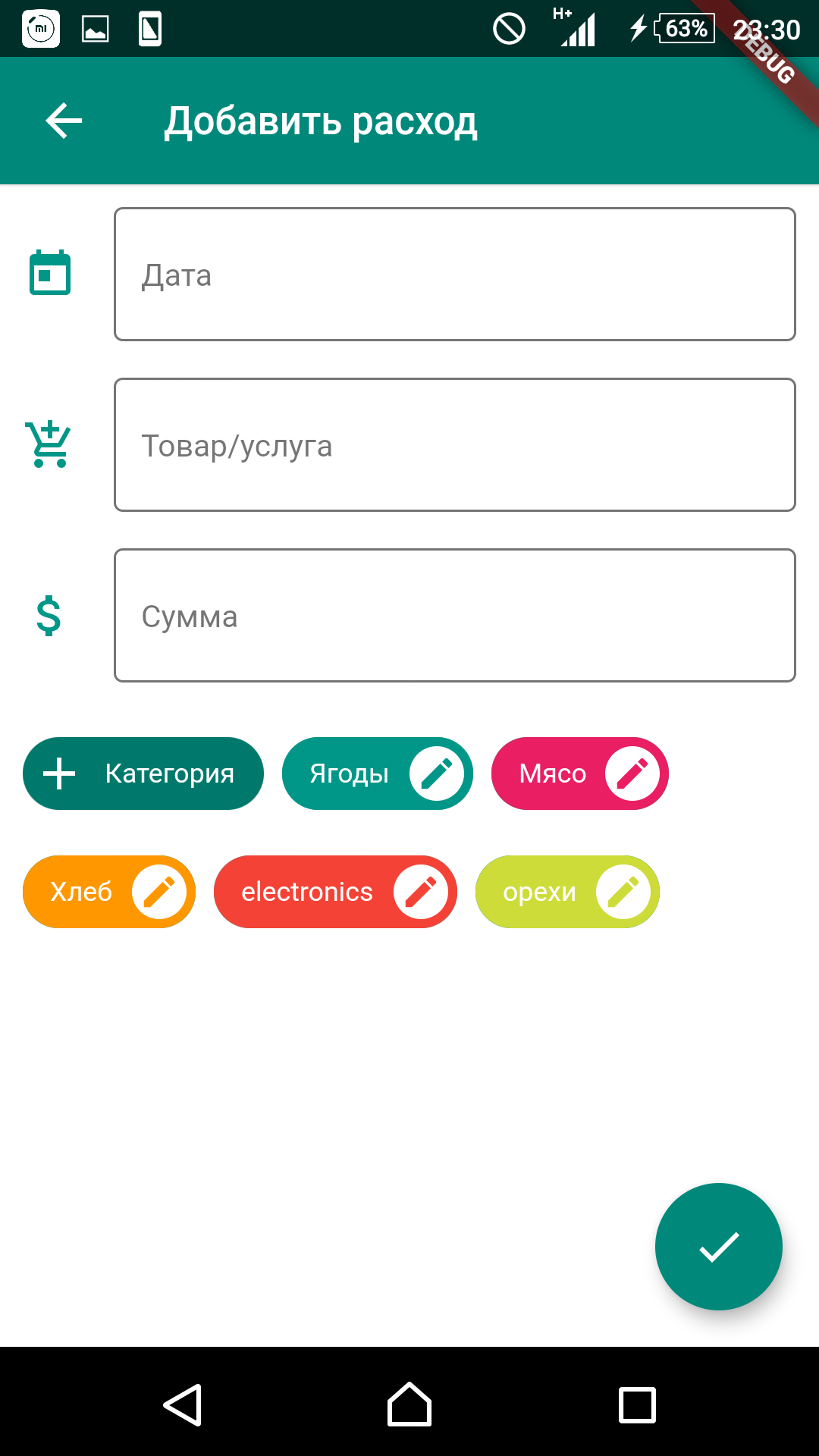


Рис. 7. Добавление расходов

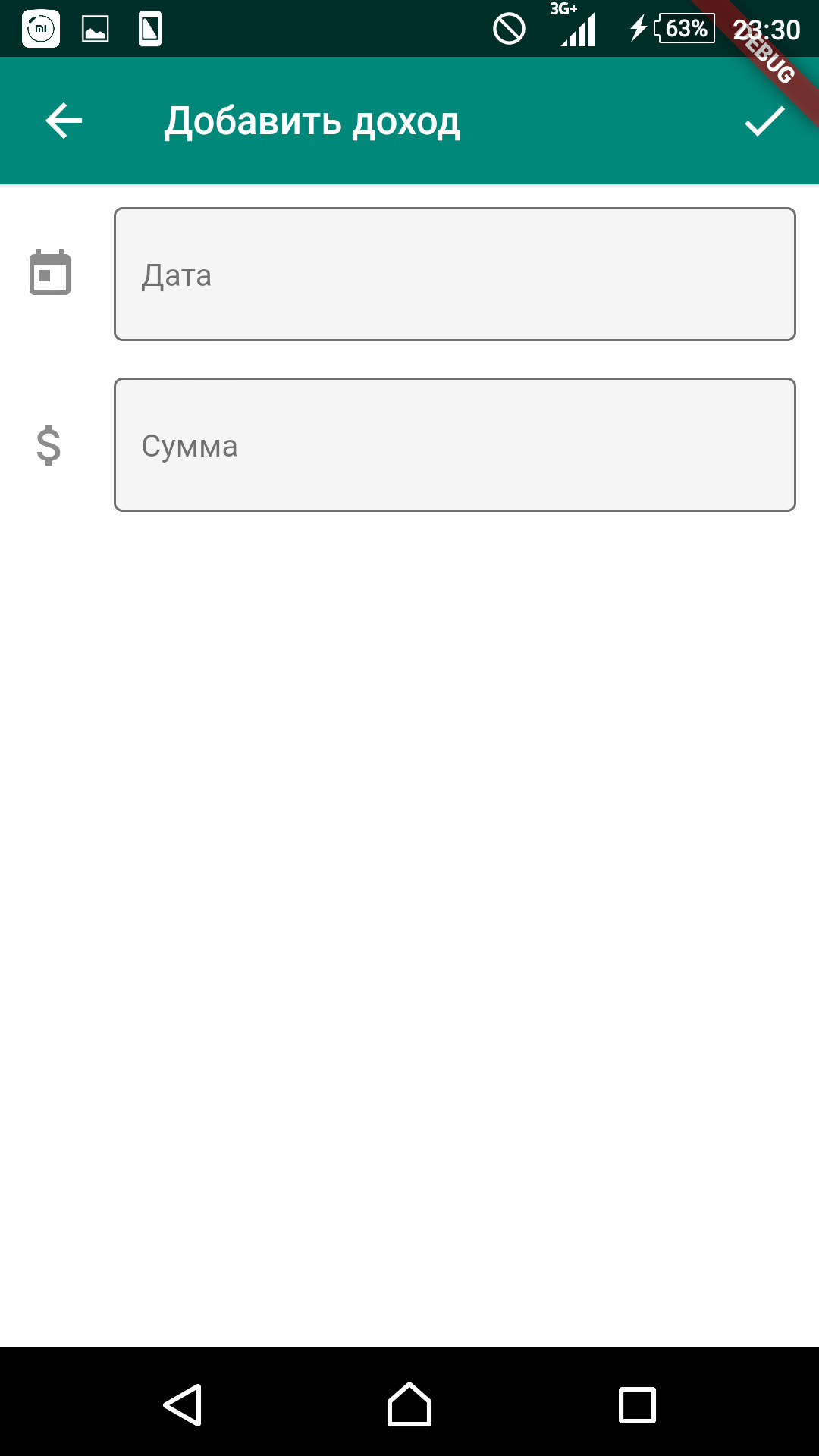


Рис. 6. Добавление доходов

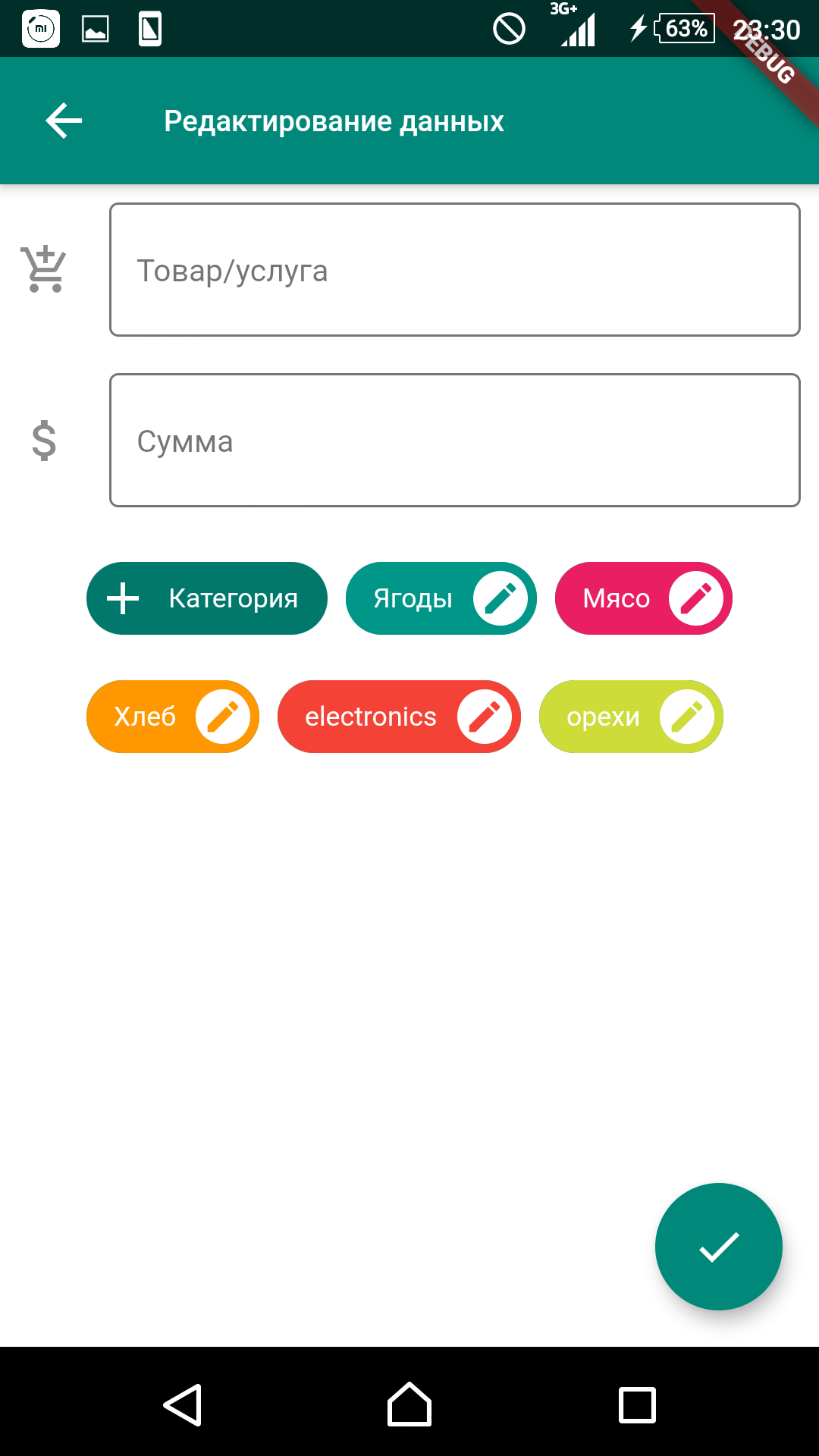


Рис. 8. Редактирование расходов

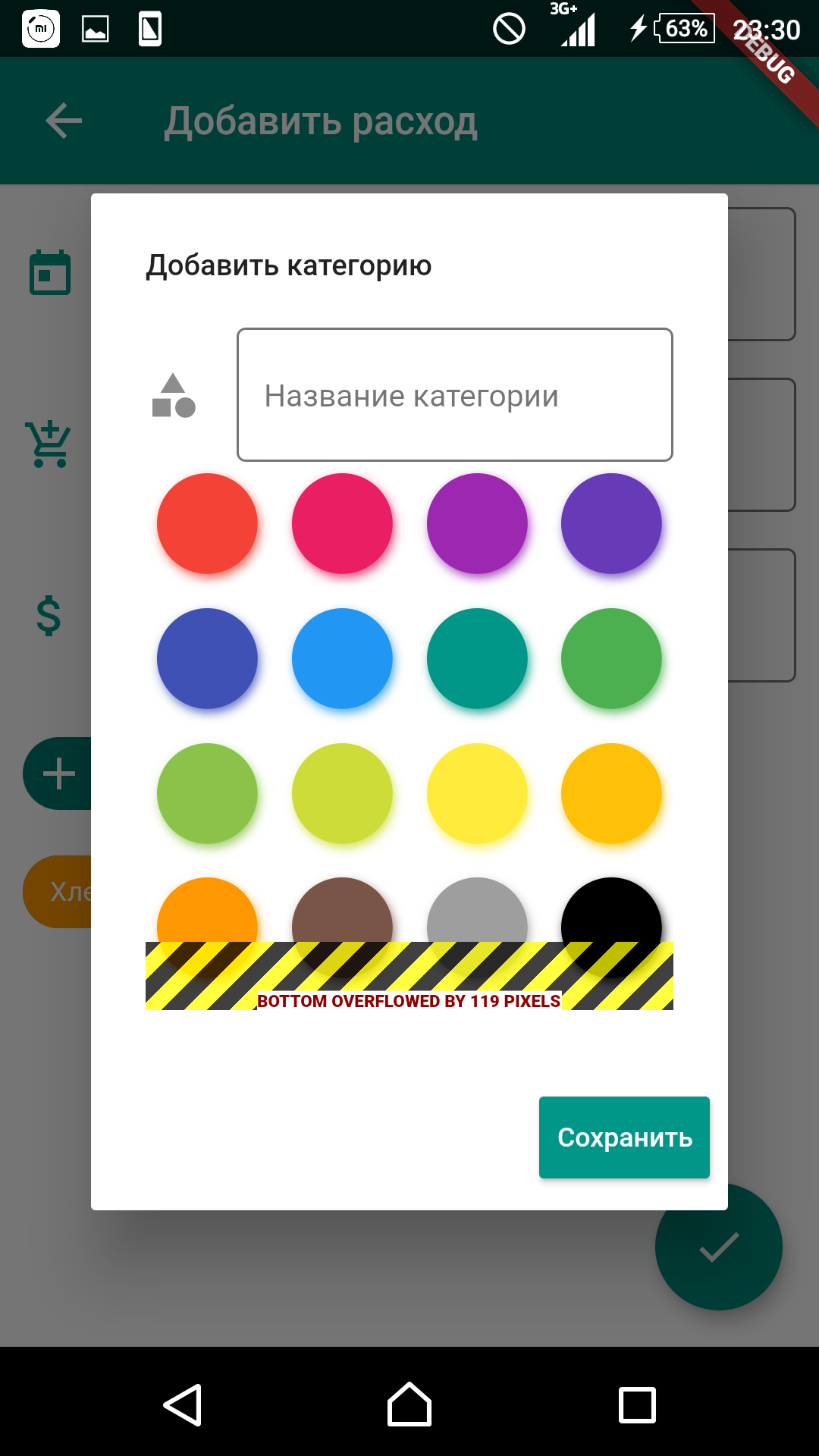


Рис. 9. Добавление категории

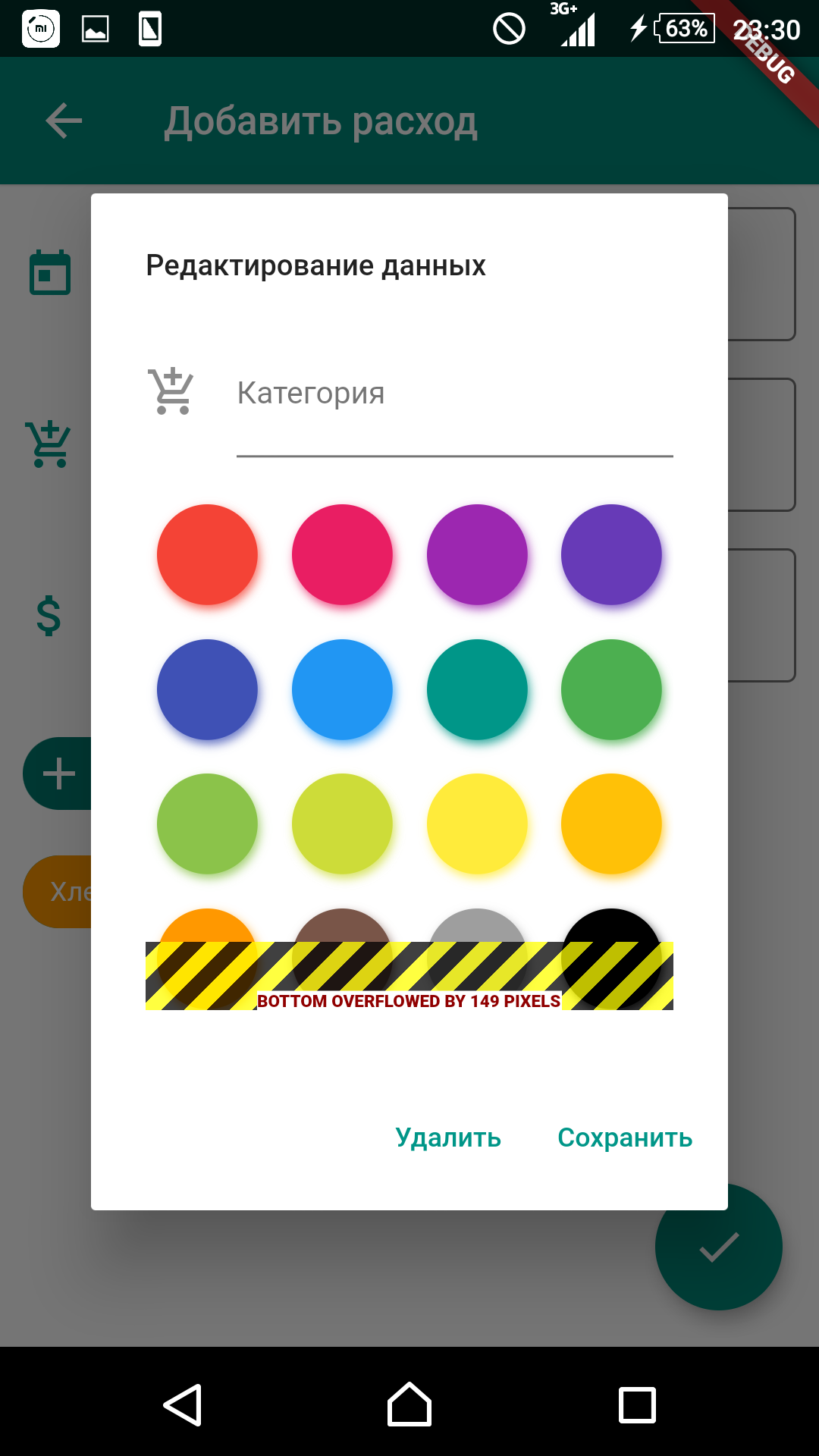


Рис. 10. Редактирование категории

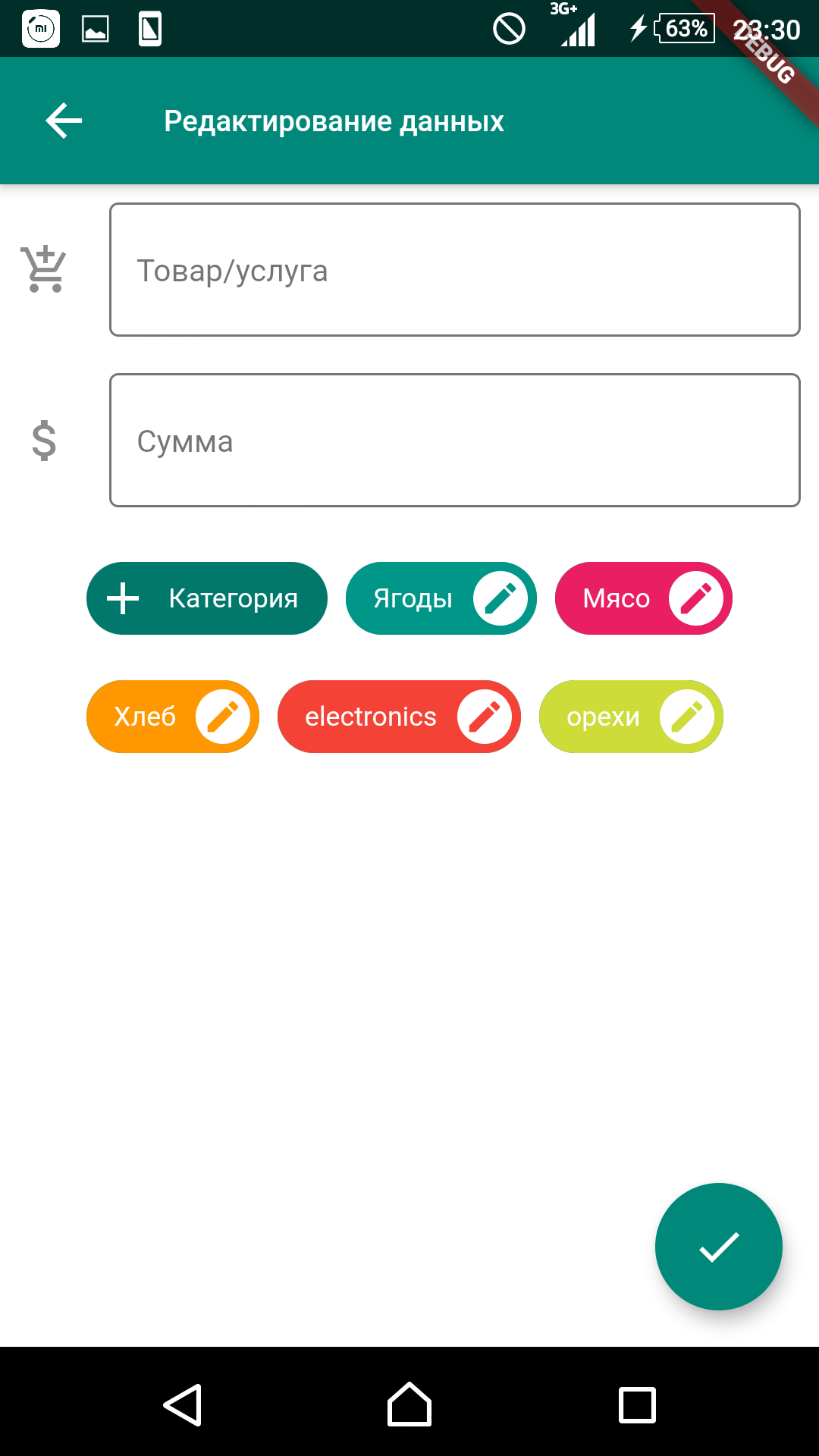


Рис. 11. Редактировать товар

1. 1 О применении контрольно-кассовой техники при осуществлении расчетов в Российской Федерации: Федеральный закон от 22.05.2003 N 54-ФЗ (ред. от 25.12.2018) // Собрание законодательства Р.Ф - 2003 г. N 21 ст. 1957. [↑](#footnote-ref-1)
2. 2 Налоговый кодекс Российской Федерации (часть вторая), от 10.08.2000 г. N 153-154 (ред. 25.12.2018) // Собрание законодательства Р.Ф. - 2000 г. N 32 ст. 3340я [↑](#footnote-ref-2)