Введение

Утверд.

Н. контр.

Провер.

Разраб.

*Смелов В.В.*

*Нистюк О.А*.

*Пилик С.И.*

*Смелов В.В.*

Ф.И.О.

Подпись

Дата

1

*Введение*

Лист

1

*ДП 02.00.ПЗ*

Лит.

Листов

*БГТУ 74217084, 2021*

В современном мире большинство видов деятельности каждого человека связано с деньгами. Хотя сегодня у людей намного больше денег, чем они имели несколько поколений назад, объём знаний о том, как управлять этими деньгами, отстаёт. Для детей деньги — это пара монет в копилке, но для взрослого человека это нечто большее. Плохие решения по личным финансам могут привести к снижению, либо вовсе потере имеющихся средств. Для решения этой проблемы необходимо контролировать свои финансовые операции, для чего используются персональные финансовые менеджеры.

На рынке приложений представлено огромное количество различных Финансовых менеджеров, однако большая часть из них имеет ряд схожих недостатков, таких как:

* неудобство создания финансовой операции;
* кража личной информации пользователя с целью анализа и персонализации рекламы;
* привязанность к конкретной платформе (мобильная, десктоп и т.д.);
* невозможность вести совместные финансы.

Таким образом, целью дипломного проекта является разработка кроссплатформенного приложения для менеджмента персональных финансов. Для достижения поставленной цели сформулированы следующие задачи:

* провести обзор аналогичных решений и патентов;
* спроектировать архитектуру и структуру базы данных;
* разработать модуль для учета долгов;
* разработать модуль для учета накоплений;
* разработать модуль для отображения графической информации о финансовых операциях;
* разработать модуль для учета расходов и доходов;
* разработать сервис для распознавания финансовой информации из торговых чеков;
* разработать способ синхронизации нескольких пользователей для совместного менеджмента финансов;
* разработать способ синхронизации нескольких платформ
* разработать способ оповещения обо всех возможных изменениях финансового состояния;
* протестировать приложение на всех доступных платформах.

1 Обзор литературы и аналогов

Утверд.

Н. контр.

Провер.

Разраб.

*Смелов В.В.*

*Нистюк О.А*.

*Пилик С.И.*

*Смелов В.В.*

Ф.И.О.

Подпись

Дата

6

*1 Обзор литературы*

*и аналогов*

Лист

1

*ДП 02.00.ПЗ*

Лит.

Листов

*БГТУ 74217084, 2021*

1.1 Требования к приложению для персонального финансового менеджмента

Поскольку финансовый менеджмент приносит значительную выгоду пользователям и пользуются большим спросом, разработанное приложение для управления личными финансами должно быть конкурентоспособным, а значит соответствовать всем или большинству требованиям к такому приложению. Из основных требований можно выделить следующие:

1. Финансовая информация очень конфиденциальна, поэтому лучше защитить ее с помощью двухфакторной аутентификации с использованием биометрических данных [1];
2. Очевидным преимуществом приложения по персональному финансовому менеджменту является возможность связать все учетные записи пользователя вместе. Благодаря этому, приложение может отслеживать и контролировать все финансы пользователей в одном месте. Для этого приложение должно поддерживать интеграцию с различными банками и платежными системами [2];
3. Дизайн финансового приложения должен быть простым и интуитивно понятным. Кроме того, рекомендуется использовать вызывающие доверие цвета, такие как белый, синий или светло-зеленый [1];
4. Как уже упоминалось, финансовая информация представляет собой крайне конфиденциальные данные, соответственно они должны быть особенно защищены. Кроме того, согласно опросу, проведенному организацией *RSA* в 2019 году, финансовые / банковские данные — это те данные, над которыми люди больше всего боятся потерять контроль [3]. Совет по стандартам безопасности данных индустрии платежных карт составил список стандартов безопасности данных индустрии платежных карт (англ. - *The Payment Card Industry Data Security Standard* (*PCI DSS*)), описывающий меры безопасности, которые должны быть встроены в приложение, которое передаёт, обрабатывает или хранит информацию о платёжных картах. Список состоит из 12 требований, а именно: установить и поддерживать конфигурацию межсетевых экранов для защиты данных о держателях карт, не использовать пароли к системам и другие параметры безопасности, заданные производителем по умолчанию, защищать хранимые данные о держателях карт, шифровать данные о держателях карт при их передаче через общедоступные сети, защищать все системы от вредоносного ПО и регулярно обновлять антивирусное ПО, разрабатывать и
5. поддерживать безопасные системы и приложения, ограничивать доступ к данным о держателях карт в соответствии со служебной необходимостью, идентифицировать и аутентифицировать доступ к системным компонентам, ограничивать физический доступ к данным о держателях карт, отслеживать и вести мониторинг всего доступа к сетевым ресурсам и данным о держателях карт, регулярно тестировать системы и процессы безопасности, поддерживать политику информационной безопасности для всех работников [4];
6. Формы отчетности должны быть максимально четкими и наглядными. Для лучшего восприятия имеет смысл использовать диаграммы, разные цвета и графики [1];
7. Для приложения по персональному финансовому менеджменту, геймификация — это способ удержания пользователей. Таким образом пользователи остаются мотивированными чаще взаимодействовать с приложением [1].

Исходя из вышесказанного, основными требованиями к приложениям для персонального финансового менеджмента являются: двухфакторная аутентификация, интеграция с различными банками и платежными системами, простой и понятный дизайн, соответствие стандартам безопасности, четкая отчетность, геймификация.

1.2 Аналогичные программные решения

1.2.1 Приложение Money Manager Expense & Budget

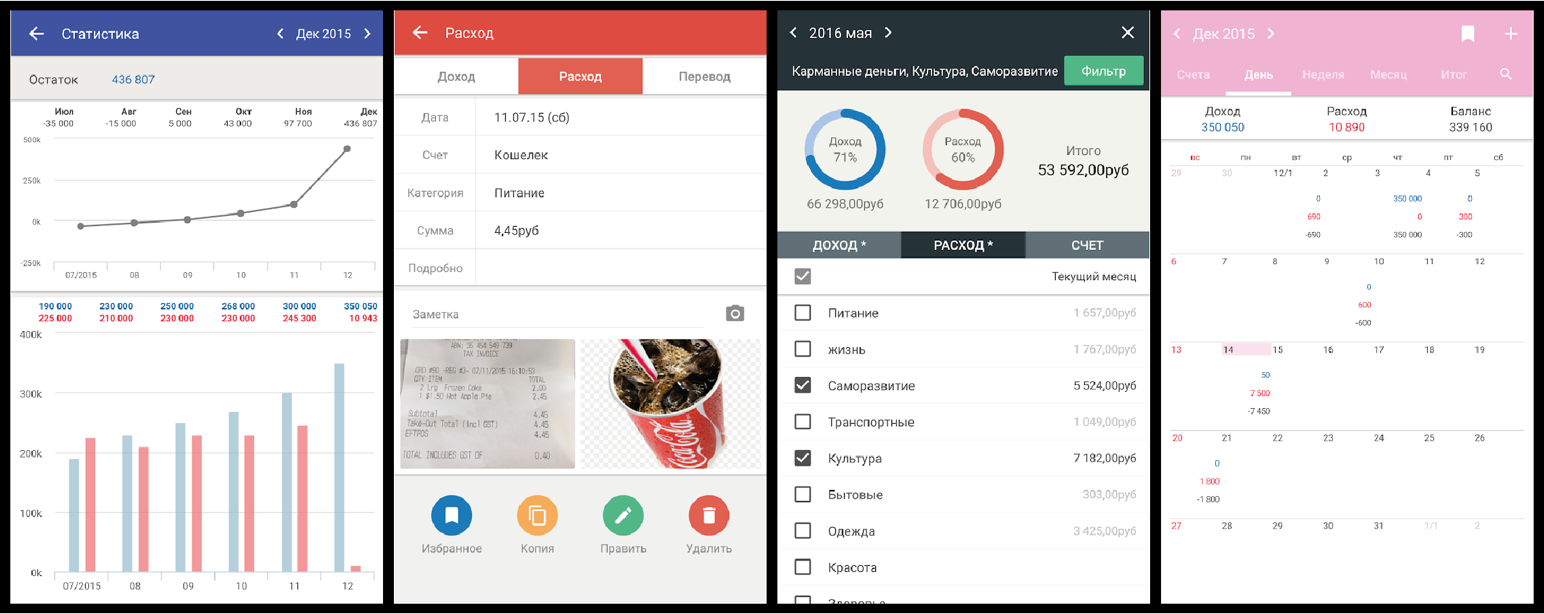
*Money Manager Expense & Budget* [5] — это универсальное приложение с разнообразным функционалом, есть версии для iOS и Android. В последнем предусмотрены два варианта ­– бесплатный и за 399 российских рублей (13,66 белорусских рублей на 18.04.2021). В платной программе отсутствует реклама и ограничения по количеству учитываемых активов (в бесплатной версии максимум 10). Используя подключение *WiFi* есть возможность редактировать и сортировать операции, категории доходов и расходов, группы счетов и другие данные с персонального компьютера.

Программа позволяет учитывать доходы и расходы, разбивая их помесячно и отражая статистику, планировать бюджет по различным направлениям, управлять картами, настраивая автоматические платежи. Встроена функция поиска, защита с помощью пароля и возможность резервного копирования данных, а также их экспорт в *Microsoft Excel*.

Достоинства приложения:

* условно бесплатный доступ к возможностям приложения;
* возможность экспорта в *Excel*;
* возможность работы в ПК;

Недостатки приложения:

* присутствует реклама;
* ограничения по количеству учитываемых активов (в бесплатной версии максимум 10);
* перегруженный интерфейс;
* нет возможности ведения долгов.

Скриншоты приложения представлены на рисунке 1.1.

Рисунок 1.1 – Скриншоты приложения *Money Manager*

1.2.2 Приложение *Monefy*

Программа предназначена для использования на платформах iOS и Android. Простая версия бесплатна, за формат *Pro* придётся заплатить 7,84 и 5,48 белорусских рублей соответственно.

***Monefy*** — идеально подходит чтобы вести учет финансов. Вы можете быстро и, что не менее важно, безопасно синхронизировать любые устройства через Ваш личный *Dropbox* аккаунт. Это самый безопасный способ, так как только Вы имеете доступ к данным.

Функции *Monefy* [6] привлекательны: простой интерфейс, удобные графики расходов, быстрое занесение информации, разнообразные категории трат. Предусмотрена поддержка нескольких счетов и использование пароля. Отличительная особенность — возможность синхронизации отдельных устройств через аккаунт *Dropbox*.

Достоинства приложения:

* простой и интуитивно понятный интерфейс;
* возможность синхронизации через *Dropbox*;
* виджеты для быстрого доступа;
* поддержка различных валют;
* выбор отчетного периода;
* режим бюджета для более эффективного контроля затрат за определенный период;
* экспорт данных;
* возможность установки пароля;
* возможность использования нескольких учетных записей.

Недостатки приложения:

* нет возможности ведения долгов;
* присутствует реклама;

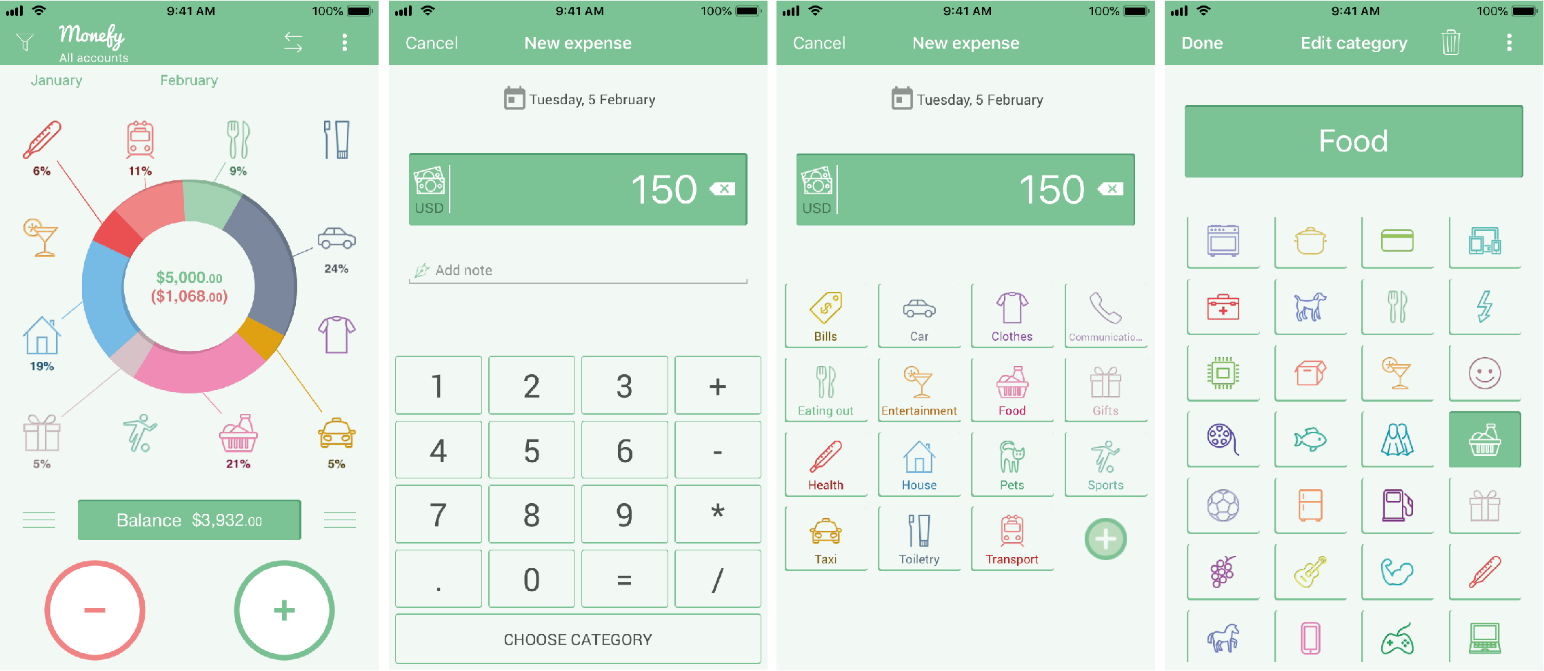
Скриншоты приложения представлены на рисунке 1.2.

Рисунок 1.2 – Скриншоты приложения *Monefy*

1.2.3 Приложение Дзен-мани

Среди функций «Дзен-мани» [7] — синхронизация с банком и автоматическое разнесение операций по категориям, ведение совместного бюджета нескольких пользователей, учёт долгов и планирование трат. Приложение актуально для малого бизнеса и фрилансеров — есть функции разделения личных и рабочих расходов, контроль за поступлениями от контрагентов и возможность оценки прибыльности бизнеса.

Программа доступна для владельцев устройств на Android и iOS. Цена за платную версию составит от 3,39 белорусских рублей до 68,49 рублей. Бесплатный формат ограничен в возможностях, но всё равно удобен для контроля за личными финансами.

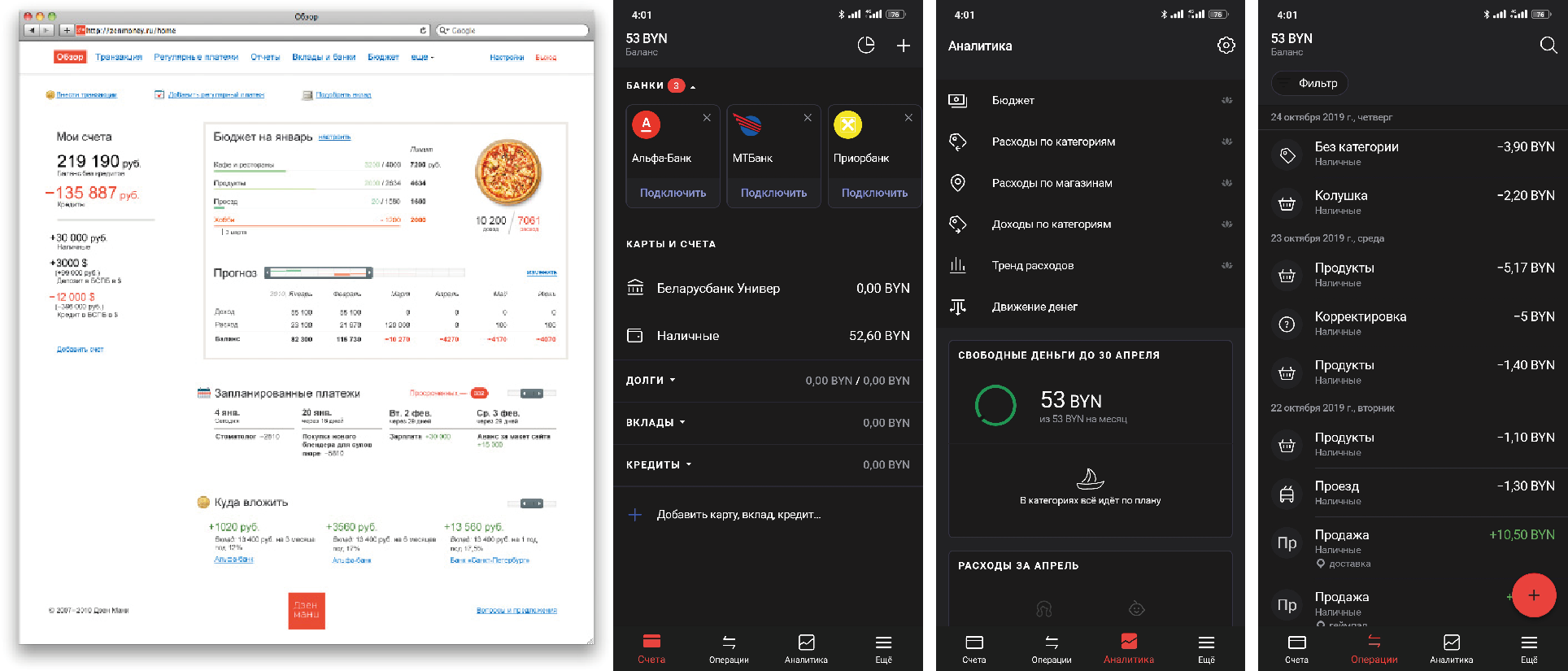
Скриншоты приложения представлены на рисунке 1.3.

Рисунок 1.3 – Скриншоты приложения Дзен-мани

Достоинства приложения:

* есть веб-сервис и мобильные приложения. Вносить траты удобно на телефоне, а планировать и анализировать дома на большом экране;
* умеет распознавать банковские СМС и создавать операции;
* есть автоматический импорт операций из электронных кошельков.

Недостатки приложения:

* цена за *Pro* версию;
* распознаёт СМС только от российских банков;
* непонятный интерфейс на *iOS*.

1.2.4 Приложение Mint

*Mint* [8] считается одним из самых успешных приложений для управления личными финансами. Это приложение включает в себя не только возможность составления бюджета, но и составления финансового плана. К функциям приложения Mint относятся: агрегация счетов, категоризация и бюджетирование, постановка финансовых целей, отслеживание счетов и расходов, советы по экономии денег, отчеты, многофакторная аутентификация, уведомления и оповещения (включая оповещения о мошенничестве), синхронизация с Mint.com, персонализированные предложения, помощь и поддержка.

Достоинства приложения:

* легкость в использовании;
* быстрый обзор всех финансов пользователя;
* привлекательный UI/UX;
* высокая безопасность;
* отслеживание кредитного рейтинга.

Недостатки приложения:

* нет поддержки для пользователей за пределами США и Канады.

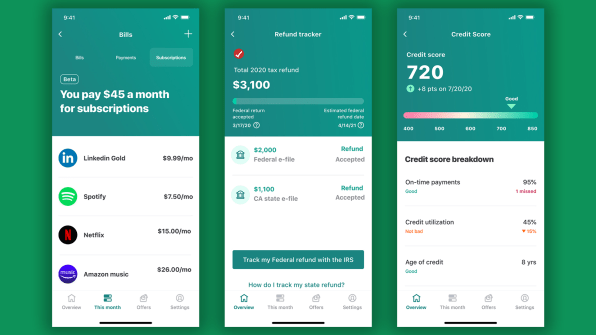
Скриншоты приложения представлены на рисунке 1.4

Рисунок 1.4 – Скриншоты приложения *Mint*

1.3 Патентный поиск

При проведении дипломного проекта основной целью проведения патентного поиска является определение уровня развития технологии и новизны существующего технического решения.

Процесс поиска определяет, как решалась поставленная задача до появления технических решений, защищенных авторскими свидетельствами, патентами и каковы перспективы их развития.

В результате патентного поиска были обнаружены следующие патенты, представленные в таблицах 1.1, 1.2, 1.3.

Таблица 1.1 – Описание патента №1

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Номер патента | Опубликовано | Авторы |
| Метод резервного копирования базы данных в облачном хранилище | CN103309772A | 02.07.2013 | Цао Женнан, Чен Фей, Хи Мужун, Лю Гуанчуань, Ван Ён, Жан Синфэн |

Способ резервного копирования файлов базы данных на основе облачного хранилища. Метод состоит из следующих шагов:

* установка режима автоматической синхронизации для клиента;
* создание каталога локальной базы данных в каталоге облачного хранилища;
* автоматическое копирование файла базы данных в каталоге локальной базы данных в соответствующий каталог облачного диска заданным способом резервного копирования [9].

Таблица 1.2 – Описание патента №2

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Номер патента | Опубликовано | Авторы |
| Системы и методы управления личными финансами | US2016125524A1 | 05.05.2016 | Loskamp Powell |

Патент представляет способы и системы, облегчающие управление личными финансами пользователей. Согласно варианту осуществления, способ управления финансами пользователя включает в себя:

* предоставление графического интерфейса, включающего в себя варианты управления финансами на вычислительном устройстве пользователя;
* получение от пользователя выбора по меньшей мере одного из вариантов управления финансами;
* запрос ввода денежных потоков от пользователя на основе выбора варианта управления финансами;
* генерация графического представления, содержащего по меньшей мере один узел;
* отображение графического представления на вычислительном устройстве пользователя [10].

Таблица 1.3 – Описание патента №3

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Номер патента | Опубликовано | Авторы |
| Способ устройство и защищенный элемент для выполнения безопасной финансовой транзакции в устройстве | RU2639690C2 | 21.12.2017 | Фонтен Себастьян, Дольчино Люк,  Дю Эй Бенжамин,  Де Нанкла Максим,  Альберти Ксавье |

Изобретение относится к средствам для проведения безопасной финансовой транзакции. Техническим результатом является расширение арсенала технических средств для выполнения безопасной финансовой транзакции в устройстве [11].

1.4 Вывод по разделу

По результатам исследования реализаций аналогов создаваемого программного средства, анализа литературы и патентного поиска был обнаружено, что на данном этапе не существует программного средства, которое реализовало бы все возможности разрабатываемого программного средства.

Суммируя все вышеперечисленное, можно сказать, что работа, проделанная по аналитическому обзору литературы, является крепким фундаментом для разрабатываемого программного средства. На данном этапе можно приступить к рассмотрению технических приемов программирования.

2 Обоснование технических приемов программирования

Утверд.

Н. контр.

Провер.

Разраб.

*Смелов В.В.*

*Нистюк О.А*.

*Пилик С.И.*

*Смелов В.В.*

Ф.И.О.

Подпись

Дата

6

*2 Обоснование*

*технических приемов программирования*

Лист

1

*ДП 02.00.ПЗ*

Лит.

Листов

*БГТУ 74217084, 2021*

2.1 Фреймворк Flutter

*Flutter* [12] представляет фреймворк от компании *Google*, который позволяет создавать кроссплатформенные приложения, которые могут использовать один и тот же код. Спектр платформ широк — это веб-приложения, мобильные приложения под *Android* и *iOS*, графические приложения под настольные операционные системы *Windows, MacOS, Linux*, а также веб-приложения.

Особенностью работы с *Flutter* является то, что приложения под разные платформы могут иметь один и тот же код. Поскольку используемые платформы не эквиваленты, то какие-то отдельные части кода необходимо настраивать под определенную ОС, например, под iOS, но тем не менее большая часть кода может совпадать. Это позволяет разработчикам существенно сэкономить время и ресурсы на создание приложений под все поддерживаемые платформы.

В качестве языка разработки используется язык программирования *Dart* [14].

При построении приложения *Flutter* транслирует код на *Dart* в нативный код приложения с помощью Dart AOT (компиляция приложения перед его запуском), которое можно запускать на Android или iOS или другой платформе. Однако при разработке приложения для ее ускорения *Flutter* использует JIT (компиляция приложения в процессе его запуска).

Надо отметить, что *Flutter* относительно новый фреймворк. Хотя прототип фреймворка появился еще в 2015 году, а первая альфа-версия вышла в мае 2017 года, но первый стабильный релиз - Flutter 1.0 увидел свет только в декабре 2018 года. Тем не менее регулярно с небольшой периодичностью выходят подверсии, добавляя новую функциональность и исправляя имеющиеся баги. На момент публикации данной статьи последней стабильной версией является Flutter 2.0, которая вышла в марте 2021 года.

Для написания кода программы можно использовать любой полюбившийся текстовый редактор и затем с помощью утилит командной строки из Flutter SDK компилировать приложение. Однако для таких сред как Android Studio и IntelliJ IDEA, а также текстового редактора Visual Studio Code компания Google выпустила специальные плагины, которые позволяют упростить разработку. Поэтому зачастую для разработки под Flutter используются именно Android Studio и IntelliJ IDEA [13].

2.2 Язык программирования Dart

*Dart* представляет язык программирования общего назначения от компании Google, который предназначен прежде всего для разработки веб-приложений (как на стороне клиента, так и на стороне сервера) и мобильных приложений. Это также

значит, что одну и ту же программу на Dart можно компилировать под различные платформы - Windows (x86/64), Android, iOS.

*Dart* — объектно-ориентированный язык. Все значения, которые используются в программе на *Dart*, представляют объекты.

В своем развитии *Dart* испытал влияние более ранних языков, таких как Smalltak, Java, JavaScript. Его синтаксис похож на синтаксис других си-подобных языков.

*Dart* быстро развивается. Текущая версия — 2.12 [15].

2.3 Среда разработки Visual Studio Code

*Visual Studio Code* [16] — это легкий, но мощный редактор исходного кода, который работает на вашем рабочем столе и доступен для Windows, MacOS и Linux. Он поставляется со встроенной поддержкой JavaScript, TypeScript и Node.js и имеет богатую экосистему расширений для других языков (таких как C ++, C #, Java, Python, PHP, Go) и среды выполнения (например, .NET и Unity).

2.4 Расширение Flutter для Visual Studio Code

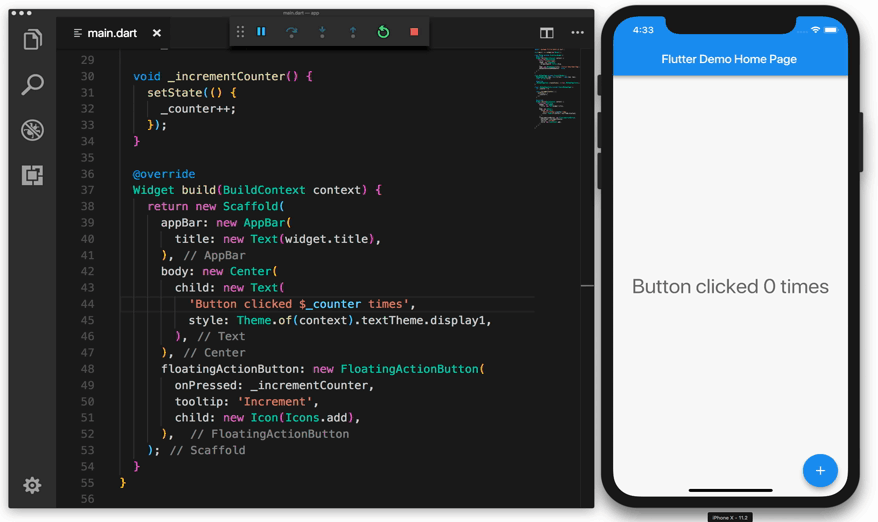
Это расширение VS Code добавляет поддержку для эффективного редактирования, рефакторинга, запуска и перезагрузки мобильных приложений Flutter, а также поддержку языка программирования Dart [17]. Пример использования изображен на рисунке 1.4.

Рисунок 2.1 – Пример использования расширения Flutter для Visual Studio Code

2.5 Локальная база данных SQLite

*SQLite* [18] — это библиотека на языке C, которая реализует быстрый, автономный, высоконадежный, полнофункциональный механизм базы данных SQL. *SQLite* — это наиболее часто используемая база данных в мире. *SQLite* встроен во все мобильные телефоны и большинство компьютеров, входит в состав множества других приложений, которые люди используют каждый день.

Формат файла *SQLite* является стабильным, кроссплатформенным и обратно совместимым, разработчики обещают сохранить его в таком виде до 2050 года. Файлы базы данных *SQLite* обычно используются в качестве контейнеров для передачи контента между системами и как долгосрочный архивный формат для данных. Активно используется более 1 триллиона баз данных *SQLite*.

Исходный код *SQLite* находится в открытом доступе и может быть использован любым желающим для любых целей.

2.6 Платформа Google Cloud

В течение последних 15 лет компания Google работала над созданием одной из самых быстрых, мощных и высокопроизводительных облачных инфраструктур. Сегодня эта инфраструктура используется в качестве базы для нескольких высокоскоростных глобальных сервисов Google, включая Gmail, Карты, YouTube и Поиск.

Все сервисы требуют эффективного управления, которое обеспечивается целым набором инструментов. Облачная платформа Google дает возможность использовать инфраструктуру, ресурсы и инструменты управления. Платформа объединяет 7 регионов, 20 зон и собственную хорошо подготовленную глобальную сеть, состоящую из сотен тысяч километров оптоволоконного кабеля.

*Google Cloud Platform* [19] дает возможность работать с различными инструментами для вычислений и хостинга. Вы можете выбрать управляемую прикладную платформу, использовать контейнеры для получения большей гибкости или создать собственную облачную инфраструктуру, максимально контролируемую. С этим решением у вас, с одной стороны, появляются широкие возможности самостоятельного управления ресурсами, а с другой, – если это не требуется, – Google может взять все управление инфраструктурой на себя! В этом случае вам останется заниматься только разработкой приложений [20].

2.6 Платформа Firebase

*Firebase* [21] — это Backend-as-a-service, или сокращенно BaaS, платформа для упращенной разработки серверной части приложения. Она включает в себя множество

*Firebase* служит базой данных, которая изменяется в реальном времени и хранит данные в JSON. Любые изменения в базе данных тут же синхронизируются между всеми клиентами, или девайсами, которые используют одну и ту же базу данных. Другими словами, обновление в *Firebase* происходят мгновенно.

Вместе с хранилищем, *Firebase* также предоставляет пользовательскую аутентификацию, и поэтому все данные передаются через защищенное соединение SSL. Разработчики могут выбрать любую комбинацию email и пароля для аутентификации, будь то Facebook, Twitter, GitHub, Google, или что-то другое.

У *Firebase* есть SDK для Android и JavaScript, также для iOS и Flutter.

Подробнее об используемых сервисах в программном средстве GoMoney будет описано в пунктах 2.7 и 2.8.

2.7 Служба Firebase Authentication

*Firebase Authentication* [22] предоставляет серверные службы, простые в использовании SDK и готовые библиотеки пользовательского интерфейса для аутентификации пользователей в вашем приложении. Он поддерживает аутентификацию с использованием паролей, номеров телефонов, популярных федеративных поставщиков удостоверений, таких как Google, Facebook и Twitter, и других.

*Firebase* *Authentication* тесно интегрируется с другими службами Firebase и использует отраслевые стандарты, такие как OAuth 2.0 и OpenID Connect, поэтому его можно легко интегрировать с любым пользовательским сервером.

2.8 Служба Firebase Cloud Messaging

*Firebase* *Cloud* *Messaging* далее FCM [23] — это бесплатное кроссплатформенное решение для обмена сообщениями.

Используя FCM, разработчики могут уведомить клиентское приложение о том, что новое электронное письмо или другие данные доступны для синхронизации. Разработчики могут отправлять уведомления, чтобы стимулировать повторное вовлечение и удержание пользователей. В случаях использования, таких как обмен мгновенными сообщениями, сообщение может передавать полезную нагрузку размером до 4 КБ в клиентское приложение.

2.9 Язык SQL

*SQL* (*Structured Query Language*) — это специализированный язык для написания запросов к реляционной базе данных. Основной единицей этого языка является команда *SQL*. Запрос БД – это один или несколько языковых операторов, которые интерпретируются СУБД и позволяют создавать, редактировать или удалять саму БД, а также ее объекты.

Изначально *SQL* был основным способом работы пользователя с базой данных и позволял выполнять следующий набор операций:

* создание в базе данных новой таблицы;
* добавление в таблицу новых записей;
* изменение записей;
* удаление записей;
* выборка записей из одной или нескольких таблиц;
* изменение структур таблиц.

Основные достоинства языка *SQL* заключаются в следующем:

* стандартность – как уже было сказано, использование языка *SQL* в программах стандартизировано международными организациями;
* независимость от конкретных СУБД – все распространенные СУБД используют *SQL*, т.к. реляционную базу данных можно перенести с одной СУБД на другую с минимальными доработками;
* возможность переноса с одной вычислительной системы на другую – СУБД могут быть ориентированы на различные вычислительные системы, но приложения, созданные с использованием *SQL*, могут быть использованы для локальных баз данных и больших многопользовательских систем;
* реляционная основа языка – *SQL* – это язык реляционных баз данных, поэтому он стал популярным, когда модель представления реляционных данных расширилась. Структура таблицы реляционной базы данных хорошо изучена, поэтому язык *SQL* легко усваивается;
* возможность создания интерактивных запросов – *SQL* предоставляет пользователям мгновенный доступ к данным, а в интерактивном режиме можно получить результат запроса за очень короткое время без написания сложной программы;
* возможность программного доступа к БД – *SQL* легко использовать в приложениях, которым необходим доступ к базам данных. Один и тот же *SQL* используется как для интерактивного, так и для программного доступа, поэтому части программ, содержащие доступ к базе данных, могут быть сначала проверены интерактивно, а затем вставлены в программу;
* обеспечение различного представления данных – *SQL* может представлять структуру данных таким образом, что конкретный пользователь может видеть различные представления. Кроме того, данные из разных частей базы данных могут быть объединены и представлены в виде одной простой таблицы, что означает, что представления подходят для повышения безопасности базы данных и настройки ее в соответствии с конкретными требованиями отдельных пользователей;
* возможность динамического изменения и расширения структуры БД – язык *SQL* позволяет управлять структурой базы данных, обеспечивая тем самым гибкость в плане адаптации базы данных к изменяющимся требованиям области.

Основными недостатками языка *SQL* является следующими:

* сложность. Хотя *SQL* был обозначен как инструмент конечного пользователя, он в итоге стал настолько сложным, что стал инструментом программиста;
* отступления от стандартов. Многие разработчики СУБД вносят изменения в язык *SQL*, используемый в разрабатываемых СУБД, отклоняясь от стандарта;
* сложность работы с иерархическими структурами [25].

2.11 Вывод по разделу

В результате обоснования технических средств и анализа поставленной задачи был выбран фреймворк *Flutter* для реализации клиентской части программного средства и язык *Dart* для реализации серверной части, работающей на HTTP-сервере *Nginx*, так как они имеют ряд преимуществ, которые были описаны в данной главе.

Суммируя все вышеперечисленное, можно преступать к проектированию и разработке модели базы данных, интерфейса, сервисов программного средства.

3 Проектирование программного средства

Утверд.

Н. контр.

Провер.

Разраб.

*Смелов В.В.*

*Нистюк О.А*.

*Пилик С.И.*

*Смелов В.В.*

Ф.И.О.

Подпись

Дата

4

*3 Проектирование*

*программного средства*

Лист

1

*ДП 02.00.ПЗ*

Лит.

Листов

*БГТУ 74217084, 2021*

3.1 Архитектура приложения

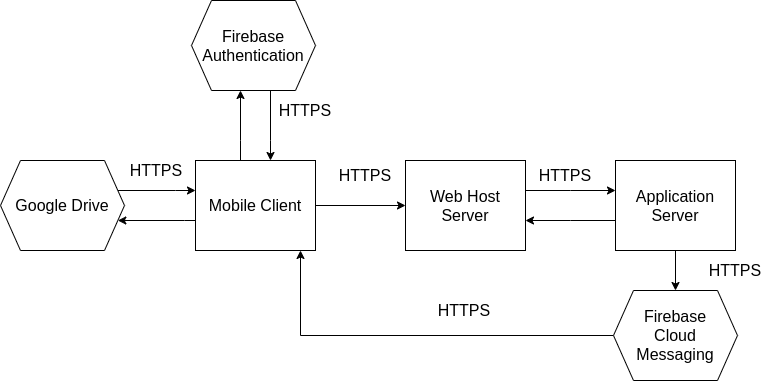
Для разрабатываемого проекта была разработана диаграмма архитектуры приложения, представленная на рисунке 3.1.

Рисунок 3.1 – диаграмма архитектуры приложения

Само приложение будет работать по принципу клиент-серверного взаимодействия. Серверная часть приложения будет работать на *Dart*, преимущество которого заключается в своей кроссплатформенности. В качестве веб-сервера будет использоваться *Nginx*. Также приложение будет обращаться к сервисам Google Drive (для создания бэкапов), Firebase Cloud Messaging (для общения между участниками группы) и Firebase Authentication (для регистрации и аутентификации) по протоколу HTTPS.

Клиентская часть приложения будет работать на базе фреймворка *Flutter*.

3.2 Проектирование серверной части приложения

Серверная часть приложения предназначена для рассылки push-уведомлений участникам группы через сервис Firebase Cloud Messaging. На вход приложение получает сообщение и адресатов и пересылает их сервису Firebase Cloud Messaging.

В таблице 3.1 приведены технологии и пакеты языка Dart, с помощью которых происходила разработка программного обеспечения серверной части приложения.

Таблица 3.1 – Технологии и пакеты, используемые в серверной части приложения

|  |  |
| --- | --- |
| Название | Назначение |
| Dart | Язык разработки |
| dia | Пакет языка Dart для разработки серверных приложений |
| dia\_router | Расширение пакета dia позволяющее задать middleware для определенных url и http методов |
| dia\_body | Расширение пакета dia разбирающее http запрос и возвращающее из него переданные параметры и загруженные файлы |
| encrypt | Пакет языка Dart для работы с шифрованием данных |

Для написания кода серверной части приложения использована *Visual Studio Code – IDE* для разработки программного обеспечения.

3.3 Проектирование клиентской части приложения

Диаграмма вариантов использования представлена в приложении Б.

3.3.1 Технологии и пакеты клиентской части приложения

В таблице 3.2 приведены технологии и пакеты языка Dart, с помощью которых происходила разработка программного обеспечения клиентской части приложения.

Таблица 3.2 – Технологии и пакеты, используемые в клиентсокй части приложения

|  |  |
| --- | --- |
| Название | Назначение |
| Flutter | Фреймворк для создания клиентских приложений |
| flutter\_local\_notifications | Пакет языка Dart для отображения уведомлений |
| flutter\_screen\_util | Пакет языка Dart для управления размерами виджетов на разных размерах экрана |
| shared\_preferences | Пакет языка Dart для простого хранения информации |
| firebase\_messaging | Пакет языка Dart для взаимодействия с Firebase Cloud Messaging |
| permission\_handler | Пакет языка Dart для получения разрешений к компонентам системы |
| flutter\_tesseract\_ocr | Пакет языка Dart для получения текстовой информации с изображений |
| image\_gallery\_saver | Пакет языка Dart для сохранения изображений в галерею |
| image\_picker | Пакет языка Dart для получения изображений с устройства |
| photo\_view | Пакет языка Dart для отображения изображений |
| image\_cropper | Пакет языка Dart для редактирования изображений |
| flutter\_isolate | Пакет языка Dart для работы с потоками |
| o\_color\_picker | Пакет языка Dart для выбора цвета пользователем |
| google\_sign\_in | Пакет языка Dart для авторизации через Google |
| Окончание таблицы 3.2 |  |
| path\_provider | Пакет языка Dart для управления файлами системы |
| firebase\_core | Пакет языка Dart для взаимодействия с Firebase |
| firebase\_auth | Пакет языка Dart для авторизации через сервисы Firebase |
| currency\_picker | Пакет языка Dart для выбора валюты пользователем |
| package\_info | Пакет языка Dart для получения информации о приложении |
| url\_launcher | Пакет языка Dart для обращения по url |
| hive\_flutter | Пакет языка Dart для связи локальной базы данных hive и flutter |
| opencv\_4 | Пакет языка Dart для обработки изображений |
| flutter\_svg | Пакет языка Dart для использования svg-изображений в проекте |
| line\_icons | Пакет языка Dart для использования иконок |
| googleapis | Пакет языка Dart для удобного обращения к Google API (используется для обращения к Drive API) |
| bot\_toast | Пакет языка Dart для отображения сообщений на экране |
| fl\_chart | Пакет языка Dart для отображения графиков |
| sqflite | Пакет языка Dart для взаимодействия с локальной базой данных |
| encrypt | Пакет языка Dart для работы с шифрованием |
| qr\_flutter | Пакет языка Dart для генерации qr кодов |
| qrscan | Пакет языка Dart для получения информации из qr кода |
| get\_it | Пакет языка Dart для построения архитектуры |
| http | Пакет языка Dart для работы с http и https |
| uuid | Пакет языка Dart для генерации уникальных идентификаторов |
| hive | Пакет языка Dart для взаимодействия с локальной базой данных |
| path | Пакет языка Dart для получения путей к файлам |
| dio | Пакет языка Dart для удобной работы с http и https |

Для написания кода клиентской части приложения использована *Visual Studio Code – IDE* для разработки программного обеспечения.

3.3.2 Структура базы данных клиентской части приложения

При проектировании приложений разного уровня сложности рано или поздно встает вопрос хранения данных.

Поэтому одним из самых важных этапов разработки является также построение схемы базы данных. Это необходимо как для общего понимания того, какую именно информацию мы будем хранить, так и для последующего повышения производительности.

Проектирование базы данных – процесс создания схемы базы данных и определения необходимых ограничений целостности.

Основные задачи проектирования баз данных:

– обеспечение хранения в БД всей необходимой информации;

– обеспечение возможности получения данных по всем необходимым запросам;

– сокращение избыточности и дублирования данных;

– обеспечение целостности базы данных.

В общей сложности база данных будет содержать 12 таблиц.

Физическая схема проектируемой базы данных представлена в приложении А.

Список таблиц базы данных и их описание представлены в таблице 3.3.

Таблица 3.3 – Таблицы базы данных

|  |  |
| --- | --- |
| Таблица | Описание |
| Users | Содержит данные о доступных пользователях |
| Groups | Содержит данные о доступных группах |
| UserGroups | Содержит данные о пользователях в группах |
| Partner | Содержит данные о партнерах |
| MoneyBox | Содержит данные о копилках |
| Category | Содержит данные о категориях |
| Scans | Содержит данные об отсканированных чеках |
| Planned | Содержит данные о запланированных операциях |
| GroupPartners | Содержит данные о партнерах в группах |
| GroupMoneyBoxes | Содержит данные о копилках в группах |
| GroupCategories | Содержит данные о категориях в группах |
| FinTransaction | Содержит данные о финансовых транзакциях |

Все таблицы связаны между собой первичными и вторичными ключами.

3.4 Вывод по разделу

Программный продукт будет включать в себя следующие компоненты:

* сервер приложения на языке Dart;
* клиентское приложение для мобильных устройств на базе операционных систем Android и iOS на фреймворкe *Flutter*.

Для проекта была выбрана клиент-серверная архитектура.

Для клиентской части приложения спроектировано 12 таблицы базы данных под управлением *SQLite*. Для коммуникации с сервером будет использоваться текстовый формат *JSON* по протоколу *HTTPS*.

По результатам проведения проектирования следует, что можно переходить к разработке всех модулей программного средства.

4 Разработка программного средства

Утверд.

Н. контр.

Провер.

Разраб.

*Смелов В.В.*

*Нистюк О.А*.

*Пилик С.И.*

*Смелов В.В.*

Ф.И.О.

Подпись

Дата

10

*4 Разработка программного средства*

Лист

1

*ДП 02.00.ПЗ*

Лит.

Листов

*БГТУ 74217084, 2021*

4.1 Серверная часть приложения

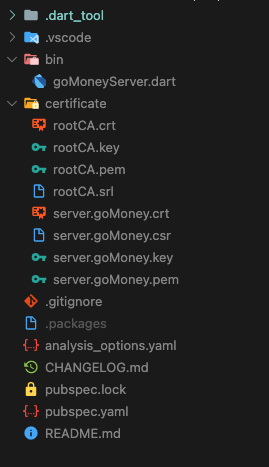
Для разработки серверной части приложения был использован язык Dart. Строение проекта представлено на рисунке 4.1.

Рисунок 4.1 – Строение проекта серверной части приложения

Целью серверной части приложения является пересылка push-уведомлений между пользователями. Так как служба *Firebase Cloud Messaging* не позволяет пересылать push-уведомления между клиентами напрямую.

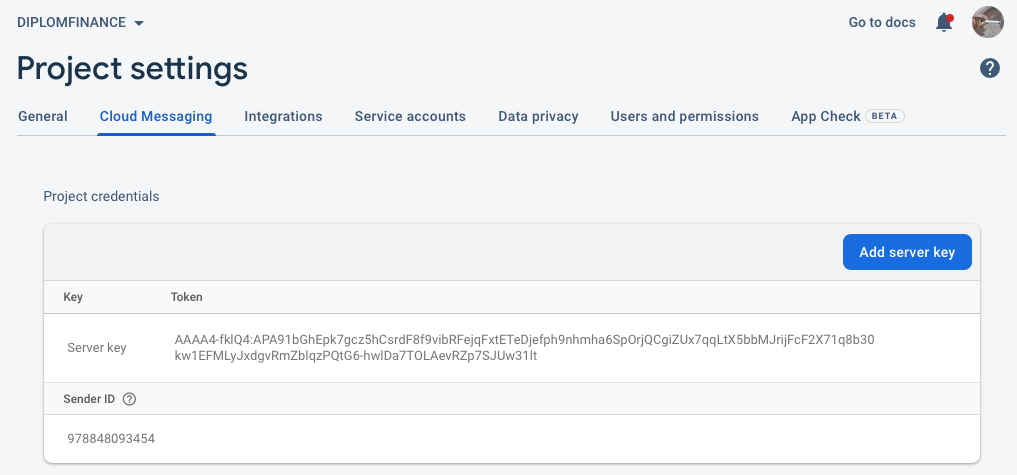
На сайте Firebase был создан проект “*DiplomFinance*”. На странице настроек проекта во вкладке «*Cloud Messaging*» был скопирован ключ сервера, он понадобится для отправки push-уведомлений. Страница настроек проекта Firebase отображена на рисунке 4.2.

Рисунок 4.2 – Страница настроек Firebase проекта

В проекте 3 пути для разных событий:

– подтверждение приглашения;

– удаление из группы;

– оповещение об операции в группе.

Каждое из трёх событий принимает зашифрованное сообщение, расшифровывает его, генерирует уведомление и отсылает сервису Firebase Cloud Messaging, который, в свою очередь, рассылает его адресатам. Методы генерации уведомлений представлены на листинге 4.1.

|  |
| --- |
| Map<String, dynamic> createOperationNotification(Map<String, dynamic> data) {  var title = '';  var body = '';  dynamic operation = data['data']['operation'];  title += 'Group ' + data['group\_name'];  body += data['from\_name'] + ' ';  body += operation['operation'] + ' ';  body += 'in ' + operation['type'] + ' ';  body += '"' + operation['name'] + '"';  return {'title': title, 'body': body};  }  Map<String, dynamic> createConfirmNotification(Map<String, dynamic> data) {  var title = 'Group Invite';  var body = '';  body += data['from\_name'] + ' ';  body += (data['status'] ? 'confirmed' : 'refused') + ' ';  body += 'your invite to group "' + data['group\_name'] + '"';  return {'title': title, 'body': body};  }  Map<String, dynamic> createDeleteFromGroupNotification(  Map<String, dynamic> data) {  var title = 'Group Delete';  var body = '';  body += data['from\_name'] + ' ';  body += 'deleted ';  body += 'from group "' + data['group\_name'] + '"';  return {'title': title, 'body': body};  } |

Листинг 4.1 – Методы генерации уведомлений

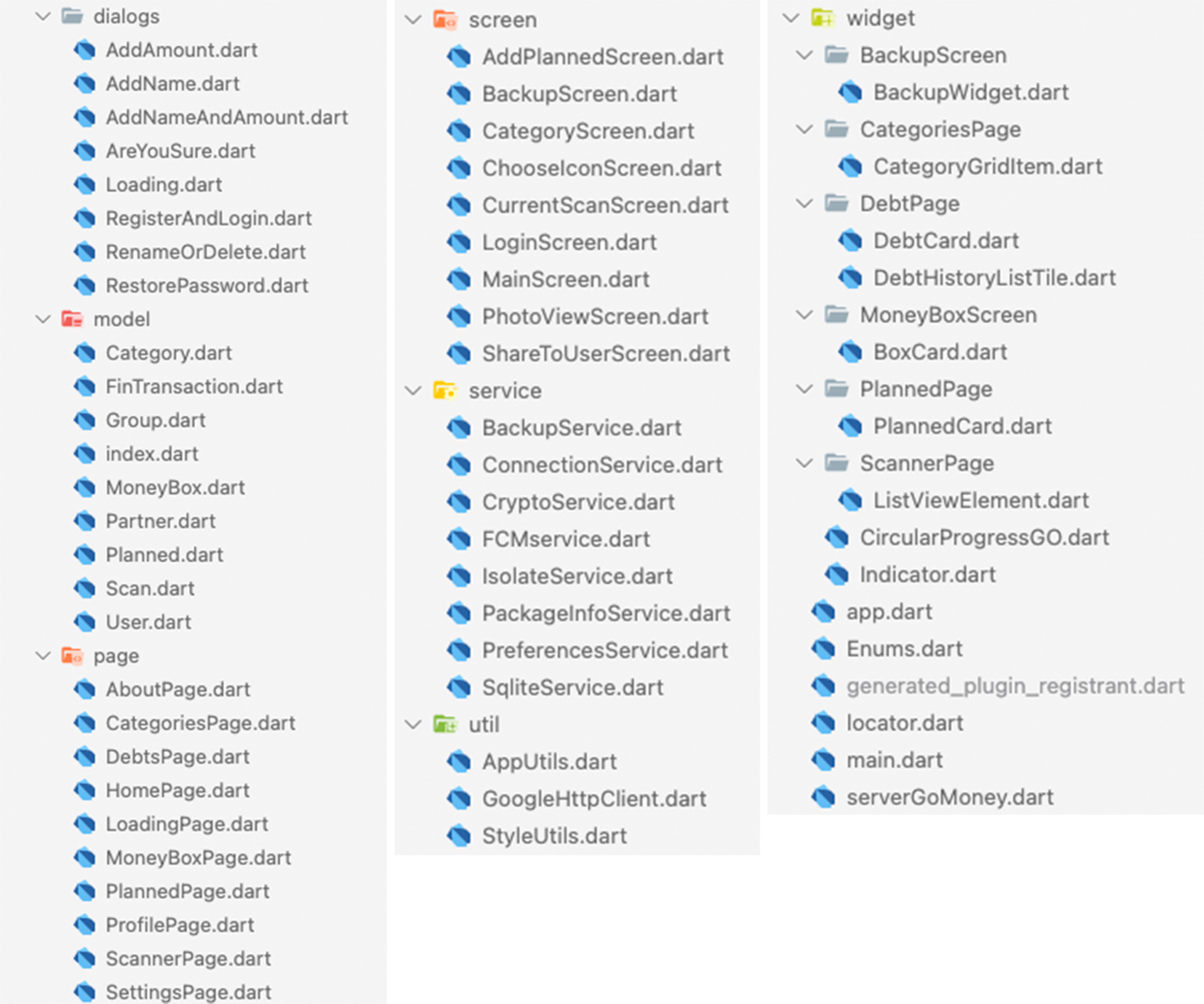
Для того чтобы сервер шифровал *HTTP* подключение были сгенерированы ключ и подписан сертификат с помощью *OpenSSL* и переданы серверу. Метод подключения сертификата и ключа представлен на листинге 4.2.

|  |
| --- |
| Future<SecurityContext> initSsl() async {  final serverContext = SecurityContext.defaultContext;  serverContext.useCertificateChain('certificate/server.goMoney.pem');  serverContext.usePrivateKey('certificate/server.goMoney.key');  return serverContext;  } |

Листинг 4.2 – Метод подключения сертификата и ключа OpenSSL

Код серверной части приложения представлен в приложении В.

4.2 Клиентская часть приложения

Для начала разработки клиентской части приложения был использован пустой шаблон проекта *Flutter*. Итоговое строение проекта представлено на рисунке 4.3.

Листинг 4.2 – Метод подключения сертификата и ключа OpenSSL

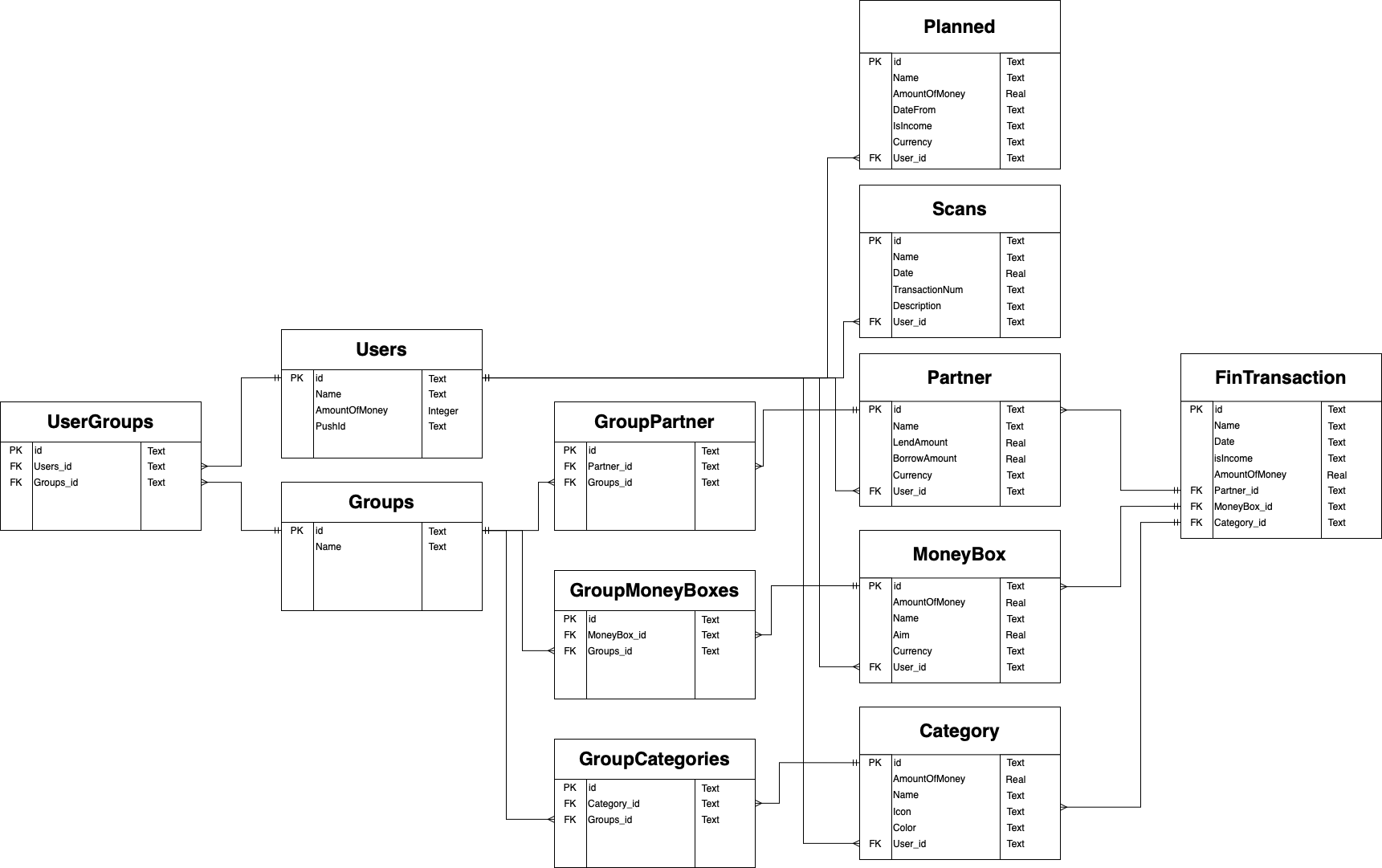
Для удобства разработки был использован пакет языка Dart под названием “*get\_it*” для удобного создания и вызова классов-сервисов. Всего было разработано 8 классов и все классы были размещены в папку “*service*”. Список сервисов и их назначение описано в таблице 4.1.

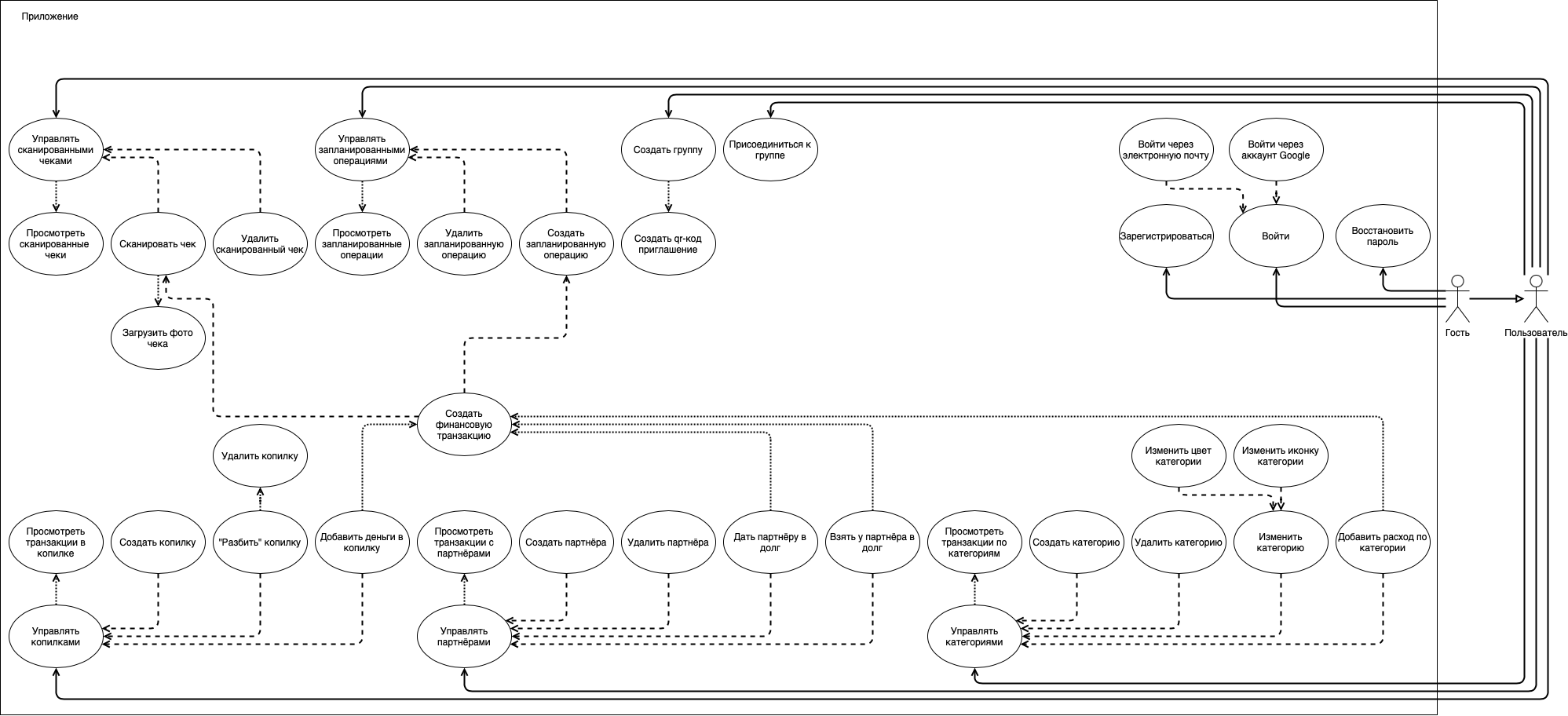
Таблица 4.1 – Сервисы клиентской части приложения

|  |  |
| --- | --- |
| Сервис | Назначение |
| Backup Service | Отвечает за создание бэкапов и отправление их на Google Drive, а также за их выгрузку и распаковку. |
| Connection Service | Отвечает за отправку запросов к серверной части приложения. |
| Crypto Service | Отвечает за шифрование и дешифровку данных. |
| FCM Service | Отвечает за получение данных из сервиса Firebase Cloud Messaging и отображение уведомлений на их основе. |
| Isolate Service | Отвечает за запуск и уничтожение задач в отдельном потоке. |
| Package Info Service | Отвечает за получение информации о приложении. |
| Preferences Service | Отвечает за получение, запись и удаление данных в *Shared Preferences*. |
| SqliteService | Отвечает за получение, запись и удаление данных в *Sqlite*. |

Список использованных источников

1. Archer Soft, how to build a Money Management App: Requirements and features [Electronic resource] // Archer Soft. – 2021. – Mode of access: https://archer-soft.com/blog/how-build-money-management-app-requirements-and-features#requirements. – Date of access: 06.04.2021.
2. Andrew Burak, How to Develop a Personal Finance App like Mint [Electronic resource] // Andrew Burak. – 2021. – Mode of access: https://relevant.software/blog/personal-finance-app-like-mint/#What\_features\_the\_personal\_finance\_app\_must\_have\_in\_2021. – Date of access: 10.04.2021.
3. RSA, RSA DATA PRIVACY & SECURITY SURVEY 2019: The Growing Data Disconnect Between Consumers and Businesses [Electronic resource] // RSA. – 2019. – Mode of access: https://www.rsa.com/content/dam/en/misc/rsa-data-privacy-and-security-survey-2019.pdf. – Date of access: 10.04.2021.
4. Стандарт Безопасности Данных Индустрии Платежных Карт, Требования и Процедуры Аудита Безопасности [Электронный ресурс] // PSI Security standards council. – Режим доступа: https://ru.pcisecuritystandards.org/\_onelink\_/pcisecurity/en2ru/minisite/en/docs/PCI\_DSS\_v3-2-1\_RU.pdf. – Дата доступа: 12.04.2021.
5. Money Manager Expense & Budget [Электронный ресурс] // Google Play. – Режим доступа: https://play.google.com/store/apps/details?id=com.realbyteapps.moneymanagerfree&hl=ru&gl=US. – Дата доступа: 12.04.2021.
6. Monefy – планирование бюджета и учет расходов [Электронный ресурс] // Google Play. – Режим доступа: https://play.google.com/store/apps/details?id=com.monefy.app.lite&hl=ru&gl=US. – Дата доступа: 12.04.2021.
7. Дзен-мани – онлайн управление личными деньгами, планирование и учет личных финансов [Электронный ресурс] // Дзен-мани. – Режим доступа: https://zenmoney.ru. – Дата доступа: 12.04.2021.
8. Mint: Personal Finance & Money [Electronic resource] // Mint Intuit. – 2021. – Mode of access: https://mint.intuit.com. – Date of access: 06.04.2021.
9. Database file backup method and system based on cloud storage: pat. CN103309772A China, G 06 F 11/14, H 04 L 29/08 / Cao Zhennan, Chen Fei, He Mujun, Liu Guanchuan, Wang Yong, Zhang Xinfeng; applicant Dawning Inf Ind Beijing Co Ltd. – No CN103309772A; 02.07.10; 11.01.12 / Espacenet. – 2010.
10. SYSTEMS AND METHODS FOR PERSONAL FINANCE MANAGEMENT: pat. US2016125524A1 United States of America, G06Q40/00 (EP, US) / LOSKAMP POWELL; applicant Loskamp Powell. – No US2016125524A1; 05.05.16; 04.11.14 / Espacenet. – 2016.
11. Способ, Устройство И Защищенный Элемент Для Выполнения Безопасной Финансовой Транзакции В Устройстве: пат. RU2639690C2 Россия, G 06 Q 20/40 , G 06 Q 10/00, G 06 Q 30/02 / Фонтен Себастьян, Дольчино Люк, Дю Эй Бенжамин, Де Нанкла Максим, Альберти Ксавье; заявитель Моубивэйв, Инк. – No RU2639690C2; 21.12.2017; 21.12.17 / Федеральная Служба По Интеллектуальной Собственности. – 2013 – 2017.
12. Flutter [Electronic resource] // Flutter. – 2021. – Mode of access: https://flutter.dev. – Date of access: 06.04.2021.
13. Введение в Flutter. Что такое Flutter. Установка [Электронный ресурс] // Metanit. – Режим доступа: https://metanit.com/dart/flutter/1.1.php. – Дата доступа: 12.04.2021.
14. Dart [Electronic resource] // Dart. – 2021. – Mode of access: https://dart.dev. – Date of access: 06.04.2021.
15. Введение в язык Dart. Что такое Dart. Первая программа [Электронный ресурс] // Metanit. – Режим доступа: https://metanit.com/dart/tutorial/1.1.php. – Дата доступа: 12.04.2021.
16. Visual Studio Code [Electronic resource] // Visual Studio Code. – 2021. – Mode of access: https://code.visualstudio.com. – Date of access: 13.04.2021.
17. Flutter Extension [Electronic resource] // Visual Studio Code. Marketplace. – 2021. – Mode of access: https://marketplace.visualstudio.com/items?itemName=Dart-Code.flutter. – Date of access: 13.04.2021.
18. SQLite [Electronic resource] // SQLite. – 2021. – Mode of access: https://www.sqlite.org/index.html. – Date of access: 14.04.2021.
19. Google Cloud [Electronic resource] // Google Cloud. – 2021. – Mode of access: https://cloud.google.com. – Date of access: 14.04.2021.
20. Google Cloud Platform: Мощная инфраструктура, большие данные и аналитика, инновационные ЦОДы и собственная сеть, полностью управляемые вычисления без серверов и многое другое [Электронный ресурс] // Softline. – Режим доступа: https://softline.by/solutions/cloud/oblachnye-platformy/google-cloud-platform-virtualnyie-mashinyi-i-setevyie-servisyi. – Дата доступа: 12.04.2021.
21. Firebase [Electronic resource] // Firebase. – 2021. – Mode of access: https://firebase.google.com. – Date of access: 14.04.2021.
22. Firebase: Аутентификация [Электронный ресурс] // Firebase. – Режим доступа: https://firebase.google.com/docs/auth. – Дата доступа: 12.04.2021.
23. Firebase: Обмен сообщениями [Электронный ресурс] // Firebase. – Режим доступа: https://firebase.google.com/docs/cloud-messaging. – Дата доступа: 12.04.2021.
24. Firebase: Обмен сообщениями [Электронный ресурс] // Firebase. – Режим доступа: https://firebase.google.com/docs/cloud-messaging. – Дата доступа: 12.04.2021.
25. SQL [Electronic resourse] // ISO. – Mode of access: https://www.iso.org/standard/45498.html. – Date of access: 14.04.2021.

ПРИЛОЖЕНИЕ А. Логическая схема базы данных

ПРИЛОЖЕНИЕ Б. Диаграмма использования

ПРИЛОЖЕНИЕ В. Листинг кода серверной части приложения

class CustomContext extends Context with Routing, ParsedBody {

CustomContext(HttpRequest request) : super(request);

}

String SEND\_URL = 'https://fcm.googleapis.com/fcm/send';

String API\_KEY =

'AAAA4-fklQ4:APA91bGhEpk7gcz5hCsrdF8f9vibRFejqFxtETeDjefph9nhmha6SpOrjQCgiZUx7qqLtX5bbMJrijFcF2X71q8b30kw1EFMLyJxdgvRmZblqzPQtG6-hwlDa7TOLAevRZp7SJUw31lt';

const SECRET\_KEY = '2020\_PRIVATES\_KEYS\_ENCRYPTS\_2020';

Future<SecurityContext> initSsl() async {

final serverContext = SecurityContext.defaultContext;

serverContext.useCertificateChain('certificate/server.goMoney.pem');

serverContext.usePrivateKey('certificate/server.goMoney.key');

return serverContext;

}

void main(List<String> arguments) async {

final app = App((req) => CustomContext(req));

final router = Router<CustomContext>('/group');

app.use((ctx, next) async {

var start = DateTime.now();

await next();

final time = DateTime.now().difference(start).inMilliseconds;

print('${ctx.request.method} ${ctx.request.uri.path} ${time}ms');

});

router.post('/confirmInvite', (ctx, next) async {

var data = encryptResponse(ctx.parsed['data']);

var resp = await http.post(Uri.parse('https://fcm.googleapis.com/fcm/send'),

headers: {

'Content-Type': 'application/json',

'Authorization': 'key=$API\_KEY'

},

body: jsonEncode({

'notification': createConfirmNotification(data),

'data': data['data'],

'registration\_ids': data['to']

}));

ctx.body = resp.body;

});

router.post('/sendOperation', (ctx, next) async {

var data = encryptResponse(ctx.parsed['data']);

var resp = await http.post(Uri.parse('https://fcm.googleapis.com/fcm/send'),

headers: {

'Content-Type': 'application/json',

'Authorization': 'key=$API\_KEY'

},

body: jsonEncode({

'notification': createOperationNotification(data),

'data': data['data'],

'registration\_ids': data['to']

}));

ctx.body = resp.body;

});

router.post('/deleteFromGroup', (ctx, next) async {

var data = encryptResponse(ctx.parsed['data']);

var resp = await http.post(Uri.parse('https://fcm.googleapis.com/fcm/send'),

headers: {

'Content-Type': 'application/json',

'Authorization': 'key=$API\_KEY'

},

body: jsonEncode({

'notification': createDeleteFromGroupNotification(data),

'data': data['data'],

'registration\_ids': data['to']

}));

ctx.body = resp.body;

});

app.use(body());

app.use(router.middleware);

await app

.listen('localhost', 8080)

.then((info) => print('Server started on https://localhost:8080'));

}

Map<String, dynamic> encryptResponse(String data) {

final key = encrypt.Key.fromUtf8(SECRET\_KEY);

final iv = encrypt.IV.fromLength(16);

final encrypter = encrypt.Encrypter(encrypt.AES(key));

return convert.jsonDecode(encrypter.decrypt64(data, iv: iv));

}

Map<String, dynamic> createOperationNotification(Map<String, dynamic> data) {

var title = '';

var body = '';

dynamic operation = data['data']['operation'];

title += 'Group ' + data['group\_name'];

body += data['from\_name'] + ' ';

body += operation['operation'] + ' ';

body += 'in ' + operation['type'] + ' ';

body += '"' + operation['name'] + '"';

return {'title': title, 'body': body};

}

Map<String, dynamic> createConfirmNotification(Map<String, dynamic> data) {

var title = 'Group Invite';

var body = '';

body += data['from\_name'] + ' ';

body += (data['status'] ? 'confirmed' : 'refused') + ' ';

body += 'your invite to group "' + data['group\_name'] + '"';

return {'title': title, 'body': body};

}

Map<String, dynamic> createDeleteFromGroupNotification(

Map<String, dynamic> data) {

var title = 'Group Delete';

var body = '';

body += data['from\_name'] + ' ';

body += 'deleted ';

body += 'from group "' + data['group\_name'] + '"';

return {'title': title, 'body': body};

}

ПРИЛОЖЕНИЕ Г.