**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное бюджетное

образовательное учреждение высшего образования

**«Владимирский государственный университет имени**

**Александра Григорьевича и Николая Григорьевича Столетовых»**

**(ВлГУ)**

**ВЫПУСКНАЯ**

**КВАЛИФИКАЦИОННАЯ РАБОТА**

Студент Парахин Кирилл Валерьевич

Институт Информационных технологий и электроники

Направление Программная инженерия

**Тема выпускной квалификационной работы**

Программная система «Мобильный органайзер»

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись инициалы, фамилия

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Парахин К.В.

подпись инициалы, фамилия

**Допустить выпускную квалификационную работу**

**к защите в государственной экзаменационной комиссии**

Заведующий кафедрой ИСПИ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.Е. Жигалов

подпись инициалы, фамилия

«\_\_\_» \_\_\_\_\_\_\_\_ 2024 г.

**АННОТАЦИЯ**

В данной выпускной квалификационной работе производилась разработка программной системы «Мобильный органайзер», целью которого является автоматизация и оптимизация рабочих процессов деятельности IT-отделов организации.

Проект состоит из 4 основных этапов, таких как: «Анализ предметной области», «Проектирование системы», «Разработка программной системы», «Информационный менеджмент».

Реализованная система может применяться для таких целей, как, например,

создание и хранение мероприятий, групп, задач и отчетов пользователей, просмотр и редактирование календаря с мероприятиями, получение уведомлений и напоминаний о начале событий.

Реализованная система использует разделение ролей на 3 типа: обычный пользователь, менеджер отдельной рабочей группы и системный администратор.

Выпускная квалификационная работа представлена на 96 страницах, основных рисунков - 37, основных таблиц – 15, использованных источников – 23, приложений – 7, иллюстрационный материал на 8 листах формата А1.

**ABSTRACT**

In this final qualification work, the development of the "Mobile Organizer" software system was carried out, the purpose of which is to automate and optimize the work processes of the organization's IT departments.

The project consists of 4 main stages, such as: "Domain analysis", "System Design", "Software system development", "Information Management".

The implemented system can be used for such purposes as, for example,

create and store events, groups, tasks and user reports, view and edit the calendar with events, receive notifications and reminders about the start of events.

The implemented system uses the division of roles into 3 types: a regular user, a manager of a separate workgroup and a system administrator.

The final qualifying work is presented on 96 pages, the main figures - 37, the main tables – 15, the sources used – 21, applications – 7, illustrative material on 8 sheets of A1 format.

**Содержание**

[ВВЕДЕНИЕ 4](#_Toc167206000)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 6](#_Toc167206001)

[1.1. Описание предметной области 6](#_Toc167206002)

[1.2. Сравнительный обзор аналогов 8](#_Toc167206003)

[1.3. Описание набора функций системы 16](#_Toc167206004)

[1.4. Описание категорий пользователей системы и прецедентов 17](#_Toc167206005)

[1.5. Словарь предметной области 18](#_Toc167206006)

[1.6. Цели и задачи работы 20](#_Toc167206007)

[2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ 21](#_Toc167206008)

[2.1. Функциональная декомпозиция системы 21](#_Toc167206009)

[2.2. Описание состава данных 26](#_Toc167206010)

[2.3. Построение схемы базы данных 27](#_Toc167206011)

[2.4. Требования к разрабатываемой системе 28](#_Toc167206012)

[3 РАЗРАБОТКА ПРОГРАММНОЙ СИСТЕМЫ 31](#_Toc167206013)

[3.1. Выбранный вариант разработки программной системы 31](#_Toc167206014)

[3.2. Описание взаимодействия приложений 33](#_Toc167206015)

[3.3. Реализация серверного приложения 34](#_Toc167206016)

[3.4. Описание структуры серверного приложения 35](#_Toc167206017)

[3.5. Спецификация API 37](#_Toc167206018)

[3.6. Описание особенностей реализации клиент-серверного приложения 38](#_Toc167206019)

[3.7. Описание разработки клиентского приложения 41](#_Toc167206020)

[3.8. Описание интерфейса мобильного приложения 43](#_Toc167206021)

[3.9. Реализация push-уведомлений 52](#_Toc167206022)

[3.10. Тестирование системы 53](#_Toc167206023)

[3.11. Развертывание приложения 56](#_Toc167206024)

[3.12. Ведение репозиториев программной системы 58](#_Toc167206025)

[4 ИНФОРМАЦИОННЫЙ МЕНЕДЖМЕНТ 60](#_Toc167206026)

[Заключение 78](#_Toc167206027)

[Список использованных источников 79](#_Toc167206028)

[ПРИЛОЖЕНИЕ А. ДИАГРАММА ПРЕЦЕДЕНТОВ СИСТЕМЫ 82](#_Toc167206029)

[ПРИЛОЖЕНИЕ Б. ЛОГИЧЕСКАЯ СХЕМА БАЗЫ ДАННЫХ 83](#_Toc167206030)

[ПРИЛОЖЕНИЕ В. ФИЗИЧЕСКАЯ СХЕМА БАЗЫ ДАННЫХ 84](#_Toc167206031)

[ПРИЛОЖЕНИЕ Г. ДИАГРАММА РАЗВЕРТЫВАНИЯ ПРИЛОЖЕНИЯ 85](#_Toc167206032)

[ПРИЛОЖЕНИЕ Д. СХЕМА АЛГОРИТМА РАССЫЛКИ УВЕДОМЛЕНИЙ 86](#_Toc167206033)

[ПРИЛОЖЕНИЕ Е. ИНФОРМАЦИОННЫЙ МЕНЕДЖМЕНТ 87](#_Toc167206034)

# ВВЕДЕНИЕ

Проект представляет собой работу по анализу, проектированию и разработке программной системы «Мобильный органайзер». Проект состоит из 4 основных глав, включающих в себя основные этапы выполнения проекта. Основными из них являются этапы: «Анализ предметной области», «Проектирование системы», «Разработка клиент-серверного приложения», «Информационный менеджмент».

Глава «Анализ предметной области» содержит в себе описание предметной области и задач, которые будет решать программная система, также включает основные позиции по набору функций системы, категории пользователей системы и словарь терминов. По результатам проведенного в этой главе сравнительного обзора аналогов был сделан вывод о том, на какие основные критерии необходимо обратить внимание при реализации программной системы.

Глава «Проектирование системы» содержит в себе функциональную декомпозицию систему, сопряженную с диаграммой прецедентов и анализом некоторых из них, описание состава данных, выполненного как в концептуальном виде – с использованием логической схемы БД, так и в физическом виде – с помощью физической схемы БД. Также в главе фигурирует обзор функциональных и нефункциональных требований.

Глава «Разработка программной системы» содержит в себе описание основных этапов, содержащих как описание выбора средств разработки, варианта взаимодействия серверного и клиентского приложения, непосредственно обзор реализации программной системы, базы данных, некоторых важных алгоритмов. В конце главы приводится раздел тестирования, содержащий в себе как ручное, так и автоматизированное (модульное и интеграционное тестирование), а также раздел развертывания приложения и обзор репозиториев программной системы.

Глава «Информационный менеджмент» содержит в себе описание шагов по анализу окружения проекта, обзору его жизненного цикла и возможных рисков, также календарный план реализации проекта. Для обоснования проекта в данной главе также было проведено экономическое обоснование системы, анализ инвестиций и функционально-стоимостной анализ, подтверждающий возможность реализации проекта в назначенные сроки с определенным набором денежных средств.

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

* 1. Описание предметной области

Предметная область разрабатываемой программно-информационной системы связана с автоматизацией и оптимизацией деятельности и функционирования IT-отделов предприятия.

Деятельность IT-предприятия чаще всего связана с разработкой программного обеспечения для удовлетворения потребностей бизнеса или внешних коммерческих организаций, а иногда направлена только на внутреннюю разработку системных компонентов, никак не связанных с аутсорсингом.

Большая часть современных коммерческих предприятий имеют собственные информационные системы, которые хранят, обрабатывают и передают данные о клиентах, сотрудниках (или вещественных сущностей – например, о товарах). Поэтому каждое такое предприятие должно иметь хотя бы один IT-отдел, занимающийся разработкой, внедрением и поддержкой программно-информационных систем, направленных на получение прибыли.

Каждый IT-отдел состоит из нескольких проектов, разбитых на основе какой-то классификации, обычно связанной с разделением групп людей по ответственности или специальности. Так, например, зачастую в IT-отделе существуют проекты по разработке (в которых могут присутствовать несколько групп backend и frontend разработчиков, автоматических и ручных тестировщиков, devops-инженеров, групп мониторинга и аналитики).

Для автоматизации деятельности отдела необходимо иерархически разбить его на отдельные группы, каждая из которых имеет своего менеджера (руководителя), обладающего отдельными полномочиями по надзору и контролю подчиненных сотрудников.

Также практически любая программная система должна иметь собственный отдел поддержки и обработки проблем пользователей и системных ошибок, чтобы осуществлять контроль происходящего функционирования системы и своевременно производить отладку и поддерджку.

* 1. Сравнительный обзор аналогов

Для того, чтобы погрузиться в предметную область автоматизации деятельности IT-отделов предприятия - необходимо провести сравнительный анализ основных программных компонентов (представленных на рынке), предоставляющих соответствующий функционал.

Для проведения сравнительного анализа было проведено изучение рынка программных компонентов систем автоматизации деятельности IT-отделов предприятия и выделены 4 основных приложений типа IT Automation Software.

Список выделенных программных компонентов категории IT Automation для сравнения:

а) Active Banch

б) Jira Service Management

в) Atera

г) Puppet

Краткое описание каждого из этих сервисов:

а) ActiveBatch Workload Automation: <https://www.advsyscon.com/en-us/activebatch>

Особенности программного продукта:

- «Продукт лучше всего подходит для организации ваших автоматизированных процессов. Подходит для среднего и крупного бизнеса»[10].

- ActiveBatch имеет архитектуру, управляемую событиями, и, следовательно, поддерживается широкий спектр триггеров событий, таких как электронная почта, события с файлами FTP и очереди сообщений.

- Продукт предоставляет возможность запускать рабочие процессы, когда это необходимо, запланировав рабочий день.

- Продукт позволяет детально отслеживать рабочие процессы.

- Продукт имеет функции моделирования процессов, которые будут полезны разработчикам для оптимизации и тестирования рабочих процессов перед запуском в производство.

- ActiveBatch имеет интеллектуальные средства автоматизации, использующие машинное обучение и искусственный интеллект.

По данному программному продукту можно сделать следующие выводы:

- «Программный продукт подходит для организации деятельности как малого, так и большого бизнеса.

- Предоставляет возможности для локального, облачного и гибридного развертывания компонентов.

- Предоставляется демо-версия и 30-дневная бесплатная пробная версия, сам продукт является платным» [10].

б) Jira Service Management: <https://www.atlassian.com/ru/software/jira>

Особенности программного продукта:

- Продукт лучше всего подходит для оптимизированного управления ИТ-сервисами.

- «Jira Service Management позволяет ИТ-командам использовать лучшие практики ITSM для обеспечения превосходного качества обслуживания клиентов. Эта единая платформа может использоваться ИТ-командами разных отделов для максимально быстрой помощи как клиентам, так и сотрудникам»[10].

По данному продукту можно сделать следующие выводы:

- «Программный продукт представляет возможности для управления практически любыми IT – сервисами.

- Предоставляет возможности для локального, облачного и мобильного развертывания компонентов.

- Компонент Jira является одним из продуктов подписки от Atlassian Cloud. Стоимость премиум-плана начинается с 47 долларов за агента. Также доступен индивидуальный корпоративный план»[10].

в) Atera RMM Software: <https://www.atera.com/>.

Особенности программного продукта:

- Продукт лучше всего подходит для автоматизации ИТ и написания сценариев для MSP, корпоративных компаний и поставщиков ИТ-услуг.

- «Atera предоставляет своим пользователям множество инструментов автоматизации ИТ, которые полностью оптимизируют вашу работу и обслуживание. Atera позволяет вам разрабатывать и внедрять мощные правила автоматизации ИТ, которые облегчат жизнь вам и вашим сотрудникам»[10].

- Продукт предоставляет пользователям набор надежных сценариев, которые могут использоваться и изменяться по требованиям;

- Продукт предоставляет возможность устранения уязвимостей в безопасности программных систем;

По данному программному продукту можно сделать следующие выводы:

- «Программный продукт предоставляет возможности для управления MSP (managed service providers – аутсорсинговых провайдеры IT-сервисов), корпоративными компаниями и поставщиками остальных IT-услуг»[10].

- Продукт предоставляет возможности для гибридного развертывания компонентов.

- Компонент предоставляется на платной основе, от 100 долларов в месяц.

г) Puppet infrastructure && IT Automation: <https://www.puppet.com/>

Особенности программного продукта:

- Продукт лучше всего подходит для малого и крупного бизнеса.

- «Puppet поможет вам в автоматизации, управляемой событиями. Он может подключать облачных провайдеров, инструменты DevOps и другие API. В соответствии с сигналами от ваших существующих инструментов DevOps, Relay запускает рабочий процесс для организации действий с нижестоящими службами. Вы сможете выстроить правильный рабочий процесс, выбрав шаги из постоянно растущей библиотеки шагов»[10].

- Puppet содержит функции триггеров на основе событий, подключений и модульных шагов. Это позволит вам добавить шаг утверждения в ваш рабочий процесс.

- Puppet позволит вам увидеть все операции с высоты птичьего полета.

- Версия Puppet Enterprise предоставляет предопределенные рабочие процессы для создания, тестирования и развертывания инфраструктуры.

По данному программному продукту можно сделать следующие выводы:

- «Программный продукт Puppet Enterprise предоставляет возможность выбора для управления и контроля любое облачное хранилище, инфраструктуру или сервис»[10].

- Компонент представляется на платной основе, имеет пробную демонстративную версию.

На рисунке 1.1 ниже приведен скриншот главной страницы Jira Service Management – популярной программной системы для трекинга задач крупных проектов:

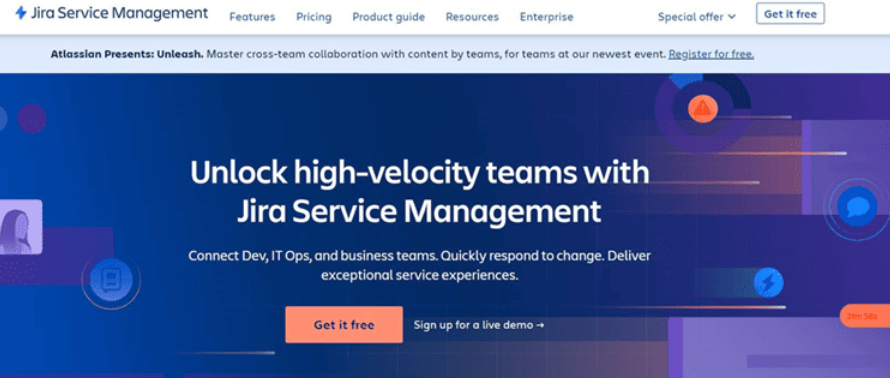


Рисунок 1.1 - Программный компонент Jira Service Management

Описание критериев для сравнения

Затем были выделены критерии, по которым можно провести сравнительный анализ выделенных компонентов – они приведены ниже в таблице 1.1:

Таблица 1.1 - Сравнительные критерии оценки программных компонентов

|  |  |
| --- | --- |
| Критерии | Краткое описание |
| Удобный пользовательский интерфейс | Наличие удобного графического пользовательского интерфейса, решающего основные задачи приложения |
| Интеграция с другими программными компонентами | Возможность интеграции с другими программными компонентами, например, Grafana, Microsoft Outlook или графический SMTP-клиент |
| Наличие бесплатной версии | Модель распространения программного компонента и наличие бесплатной версии продукта (возможно, пробной) |
| Автоматизация бизнес-процессов | Наличие встроенных программных средств и технологий для удобного представления и автоматизации бизнес-процессов |
| Наличие Workflow - механизмов | Наличие и гибкое использование Workflow-механизмов (например, для представления жизненного цикла задач или процессов) |
| Наличие трекинга задач | Наличие трекинга задач в системе и сопутствующего механизма |
| Наличие планировщика мероприятий | Наличие встроенного планировщика мероприятий и интеграция их с другими процессами |
| Наличие автоматического мониторинга | Наличие средств мониторинга, представляющего Devops средства для настройки компонентов системы для их автоматического контроля и исправления неисправностей |
| Наличие систем генерации отчетов | Наличие функционала по сбору статистики и данных и формированию по ним отчетов |
| Использование ИИ | Использование возможностей искусственного интеллекта для гибкого использования |

В приведенной ниже таблице 1.2 приведено сравнение четырех программных компонентов IT Automation: ActiveBatch, Jira, Atera, Puppet по выделенным 10 критериям.

Для оценки использовалась трехбальная шкала – в которой оценки могут проставляться от 0 до 2, где 0 – это означает, что данный критерий полностью не удовлетворяется (либо отсутствует) в системе, 1 – частично представлен/либо имеет альтернативные механизмы, 2 – удовлетворяет заявленному критерию в достаточной мере.

Таблица 1.2 - Расчет средней оценки программных компонентов по выделенным критериям

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерии/Cервисы | Active Batch | Jira | Atera | Puppet |
| Удобный пользовательский интерфейс | 2 | 2 | 2 | 2 |
| Интеграция с другими программными компонентами | 2 | 2 | 2 | 2 |
| Наличие бесплатной версии | 1 | 0 | 0 | 1 |
| Автоматизация бизнес-процессов | 2 | 1 | 2 | 1 |
| Наличие Workflow - механизмов | 2 | 2 | 2 | 0 |
| Наличие трекинга задач | 0 | 2 | 1 | 0 |
| Наличие планировщика мероприятий | 0 | 0 | 0 | 0 |
| Наличие автоматического мониторинга | 1 | 0 | 1 | 2 |
| Наличие систем генерации отчетов | 1 | 0 | 2 | 0 |
| Использование ИИ | 0 | 1 | 2 | 1 |
| Средняя оценка | 1,3 | 1,3 | 1,4 | 0,9 |

Итоги и выводы по проведенному сравнительному анализу:

Наивысшую среднюю оценку по итогу сравнения набрал сервис Atera (средняя оценка 1,4), затем с небольшим отставанием идут компоненты Active Batch и Jira Service Management (1,3), с большим отставанием идет программный компонент Puppet (0,9).

При этом практически все рассмотренные программные компоненты категории IT Automation оказались достаточно удобны, красивы (то есть имеют приятный и утонченный дизайн), информативны и динамичны.

Так как по итогу сравнения наибольшую оценку набрал программный компонент Atera – то для проектирования и разработки собственного мобильного программного компонента для автоматизации деятельности IT-отделов предприятия был выбран этот программный компонент (на рисунке 1.2 представлен скриншот главной страницы данного системного компонента).

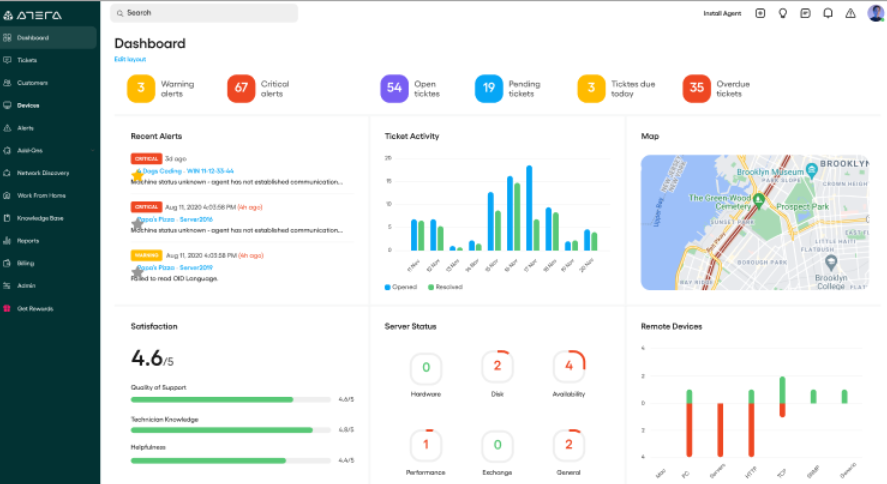


Рисунок 1.2 - Программный компонент Atera

* 1. Описание набора функций системы

Основные функции разрабатываемой системы, определяющие границы предметной области:

а) зарегистрироваться в системе (пользоваться функционалом только имея учетную запись)

б) войти в систему (авторизоваться в приложении с любого другого мобильного устройства)

в) создавать группы пользователей, выбирая их тип по классификации – выбирать параметры интеграции пользователей в своей группе

г) добавлять задачи в отдельном разделе приложения, прикреплять их к своим событиям в календаре (либо к другим пользователям)

д) добавлять общие задачи и цели для развития IT-проекта

е) планировать мероприятия для себя или для группы пользователей, к которой он принадлежит

ж) просматривать свой календарь, удалять из него мероприятия, редактировать их (то есть редактировать расписание)

з) регистрировать и отсылать в поддержку сообщения с проблемами в стабильности работы системы и взаимодействии сотрудников проекта

и) просматривать, оформлять и закрывать сообщения о проблемах в работе системы

к) получать отчеты от приложения за определенный период времени)

л) создавать отчеты и получать диаграммы (например, в виде гистограмм) о деятельности пользователей в группе (проекте)

м) рассылать уведомления на зарегистрированные мобильные устройства пользователей системы

н) получать push-уведомления и электронные сообщения по почте

* 1. Описание категорий пользователей системы и прецедентов

Система по умолчанию не включает в себя явное разделение пользователей по ролям, то есть все пользователи после регистрации получают статус обычного пользователя, по умолчанию.

Регистрация не требует подтверждения отдельными ответственными лицами (все делается автоматически).

Деятельность по модерации в системе тоже никакая не планируется проводиться, поэтому отдельно роль модератора не выделяется.

Но при этом в системе можно получить роль системного администратора (при выполнении некоторых условий) и выполнять обработку тикетов пользователей системы и сообщений о проблемах в стабильности работы программного компонента.

Эти условия заключаются в следующем: пользователь должен быть зарегистрирован в системе хотя бы 2 недели назад и знать пароль системного администратора.

Также стоит отметить, что пользователи могут создавать группы (в рамках собственных иерархических проектов или внутренних групп) – в данных группах должен быть назначен свой собственный менеджер (иначе говоря руководитель проекта), который имеет возможности назначать задачи другим пользователям, цели развития проекта, планировать общие мероприятия, создавать отчеты и диаграммы о деятельности проекта, выполнять прочие организационные действия.

* 1. Словарь предметной области

Задача (task) – некоторая абстрактная деятельность (или цель - goal), которую пользователь хочет (или должен) выполнить в определенный срок (или в рамках какого-то мероприятия).

Goal (цель) – поставленная задача, отражающая развитие пользователя в рамках деятельности в течение какого-то срока.

Календарь (calendar) – набор событий, задач и мероприятий пользователя в рамках определенного промежутка времени. Графически показывается в виде таблицы, сгруппированной по датам и часовым промежуткам.

Мероприятие (meet) – некоторая совместная деятельность группы пользователей (или одного человека), которое планируется провести в определенное время (оно создается в календаре).

Видеоконференция (или по-простому «созвон» - call) – мероприятие, которое проводится в рамках удаленного аудио или видео - разговора в какой-то программе (например, Google Meet, Zoom, Discord).

Snapshot (отчет) – совокупность данных, связанных с некоторым пользователем системы, оформленных в удобном для просмотра и анализа виде (в текстовом описании или в виде гистограммы).

Гистограмма – это способ представления табличных данных в графическом удобном для восприятия виде (относится к категории столбчатых диаграмм).

Feature – это определенная деталь программного продукта, которая имеет свои специфические характеристики.

Bug – это некоторая ошибка в работе программы, связанная с упущением внутри программного компонента, допущенного в ходе его реализации.

Issue (проблема) – это некоторая зафиксированная проблема (bug), выявленная участником жизненного цикла разработки программного продукта (например, тестировщиков или отделом мониторинга).

Alert (Alert system) – это система экстренного оповещения специалистов технической поддержки и разработчиков о неисправностях, произошедших при работе программного продукта.

Тикет (ticket) – это запрос от клиентов в службу поддержки.

* 1. Цели и задачи работы

Целью выпускной квалификационной работы является разработка программной системы «Мобильный органайзер».

Особенностью разработки является создание мобильного клиент-серверного приложения, целью которого является автоматизация деятельности IT-отделов предприятия.

Основные задачи, которые необходимо решить во время выполнения выпускной квалификационной работы:

* + Провести анализ предметной области;
  + Рассмотреть основные аналоги и провести их сравнительный обзор;
  + Выделить основные требования, предъявляемые системе;
  + Провести проектирование системы;
  + Определить и обосновать средства разработки;
  + Осуществить разработку серверного приложения и базы данных;
  + Произвести разработку мобильного приложения;
  + Выполнить развертывание и тестирование приложения;
  + Подготовить информационный менеджмент проекта по разработке и внедрению ПС.

1. ПРОЕКТИРОВАНИЕ СИСТЕМЫ
   1. Функциональная декомпозиция системы

Основные прецеденты (доступные всем пользователям системы):

а) Управлять своим аккаунтом

1) Зарегистрироваться

2) Авторизоваться

3) Выйти из системы

4)Подать заявку на получение роли системного администратора

б) Просматривать календарь мероприятий пользователя

1) Просматривать свой календарь

2) Просматривать календарь пользователей в группе

в) Управлять трекингом задач

1) Распределять задачи в группе по исполнителям

2) Контролировать смену статусов задач

3) Отослать уведомление о назначении задачи

г) Управлять мероприятиями

1) Выбрать тип мероприятия

2) Выбрать время и продолжительность

3) Добавить пользователей из группы на мероприятие

4) Отослать уведомления о приглашении на мероприятие

д) Создать отчет о личной деятельности

1) Выбрать критерий отчета (отчет по задачам, мероприятиям, проблемам)

2) Выбрать период для получения отчетности

3) Выбрать тип представления отчета (текстовый или графический)

е) Регистрировать проблемы рабочего процесса

1) Фиксировать и отправлять инциденты о неисправностях работы системы

2) Сигнализировать о прочих нарушениях рабочего процесса

ж) Получать уведомления

1) Получать уведомления о назначении задачи на исполнение

2) Получать уведомления о приглашении на мероприятие

3) Получать уведомления о начале мероприятий

Прецеденты менеджера группы:

а) Заниматься менеджментом группы

1) Выбрать тип группы

2) Добавить участников в группу

3) Редактировать параметры группы

4) Удалить участника группы

б) Создать отчет по деятельности группы

Прецеденты системного администратора системы:

а) Осуществлять поддержку

1) Получать уведомления о неисправностях в работе системы

2) Получать alert-сообщения в авто-режиме

3) Менять статусы инцидентов при их закрытии

Диаграмма прецедентов в нотации UML представлена в Приложении А.

Описание некоторых прецедентов

Описание прецедентов любого пользователя системы, связанных с управлением его аккаунтом:

Прецедент «Зарегистрироваться»

Предусловие: Пользователь решил впервые воспользоваться мобильным приложением «Многофункциональный календарь»

Действующее лицо: пользователь

Основной поток: пользователь открывает приложение, нажимает кнопку «Зарегистрироваться», вводит учетные данные для регистрации, затем получает ссылку на указан адрес электронной почты, по которой он должен перейти – для завершения процесса создания новой учетной записи

Альтернативный поток:

Пользователь не перешел по отправленной ссылке в течение 10 минут. Тогда процесс создания новой учетной записи отменяется и промежуточные данные удаляются.

Пользователь уже зарегистрирован в системе. Тогда при попытке зарегистрироваться с уже сохраненными в системе учетными данными, он будет уведомлен о наличии зарегистрированной учетной записи с введенными данными.

Постусловие: При наличии зарегистрированной учетной записи, пользователь будет перенаправлен на страницу авторизации.

Описание функционального прецедента менеджера группы:

Прецедент «Создать отчет по деятельности группы»

Предусловие: Пользователь имеет роль менеджера группы и находится на странице группы.

Действующее лицо: менеджер группы.

Основной поток: пользователь заходит на страницу группы, в которой он является менеджером – в результате он видит ссылку на форму создания отчета о деятельности группы – где он может выбрать параметры отчета (тип отчета: отчет по задачам, мероприятиям или решению проблем в работе системы, временные рамки отчета)

Альтернативный поток:

Пользователь не является менеджером текущей группы – тогда при заходе на страницу группы он увидит соответствующее информативное диалоговое окно.

Пользователь ввел некорректные данные – тогда они будут зафиксированы клиентским приложением при валидации запроса на создание нового отчета – и страница будет очищена от неправильно введенных полей.

Постусловие: После отправки запроса на создание нового отчета – сервер обработает данный запрос и пошлет результат пользователю – после чего он имеет возможность перейти на страницу нового отчета и выполнить его просмотр как в текстовой форме (с описанием деятельности как в общем группы, так и ее участников по отдельности), так и в графической форме (в виде гистограммы, выстроенной согласно рассчитанным коэффициентам KPI – о полезной типовой деятельности пользователей группы за определенный период).

Описание функционального прецедента системного администратора:

Прецедент «Осуществлять поддержку»

Предусловие: Пользователь является системным администратором

Действующее лицо: системный администратор

Пользователь с такой ролью, авторизованный в системе, получает push-уведомления о происходящих неисправностях в работе системы – далее он может перейти по уведомлению к описанию ошибки, создать запрос на исправление (issue) и отправить его разработчикам

Альтернативный поток:

Пользователь не имеет роли системного администратора. Для ее получения он должен подождать, чтобы с момента его регистрации в системе прошел срок в 2 недели – после чего ввести специальный root-пароль.

Пользователь не авторизован со своего устройства в системе – тогда он может произвести авторизацию и перейти на страницу со списком неисправностей – где дублируются все сообщения, полученные в автоматическом режиме с сервера.

Постусловие: При получении ответа от разработчиков о решении проблемы – администратор может перевести проблемный запрос в терминальный статус и закрыть его.

* 1. Описание состава данных

Выделим следующие абстракции данных (сущности системы):

а) User (пользователь с некоторыми учетными данными).

б) UserDeviceMap (токены мобильных устройств существующих клиентов системы).

в) Group (часть некоторого сообщества, к которой прикреплен пользователь).

г) Task (задача, поставленная на исполнение администратором группы другому пользователю (или себе)).

д) Event (событие (или мероприятие), предполагающее встречу/созвон – или какую-то другую совместную деятельность пользователей из одной группы).

е) Issue (проблема, зафиксированная пользователем системы или автоматически сервером).

ж) Alert (проблема в работе системы).

з) Snapshot (отчет с данными).

Дополнительно используются перечисления UserRole (роль пользователя системы), GroupType (тип группы), TaskType (тип задачи), TaskStatus (текущий статус задачи), EventType (тип мероприятия), EventStatus (текущий статус мероприятия), IssueType (тип проблемы), IssueStatus (текущий статус проблемы), SnapshotType (тип критерия, на основе которого составляется отчет), AuditType (для определения аудитории, для которой создается отчет).

Логическая схема базы данных представлена в приложении Б.

* 1. Построение схемы базы данных

Далее требуется построить структурную схему базы данных Postgre SQL. Как вариант – это можно сделать с помощью инструмента построения ER-диаграмм в программе Dbeaver – информация бралась из источника [12]. Благодаря этому, можно сразу получить список самих таблиц, их первичных и внешних ключей, а также связей между ними.

Во многом, вид физической схемы данных похожа на ее логическую модель – исключая способ представления связей многие ко многим, возникающие между таблицами «users» - «groups» и таблицами «users» - «events». Для этих связей были созданы специальные переходные таблицы «groups\_users\_maps» и «events\_users\_maps» - которые содержат в себе ссылки между пользователями, находящимися в определенных группах или приглашенных в мероприятия.

Также существует таблица «alerts» без связей с другими таблицами – так сделано было, потому что алерт – это автоматическая зарегистрированная неисправность в системе, которая не имеет никакого отношения к пользователю (системному администратору), ее обнаружившему.

Физическая схема базы данных представлена в приложении В.

## Требования к разрабатываемой системе

Выделим основные категории требований, которые необходимо предъявить к анализируемой выше разрабатываемой системе:

Функциональные требования

Список функциональных требований, необходимых для мобильного приложения «Мобильный органайзер»:

а) Функция A

1) Название - возможность использовать авторизацию пользователей;

2) Описание - пользователь получает доступ к функционалу на сервере с помощью специального сгенерированного токена, получаемого после авторизации (после входа в систему). Используется аутентификация Firebase;

3) Назначение – предоставить доступ только запросам от авторизованных пользователей;

4) Зависимости – фреймворк ASP.NET Core 5, Firebase Authentification.

б) Функция Б

1) Название – осуществление регистрации пользователя и подтверждения учетной записи;

2) Описание – возможность регистрации пользователя в системе;

3) Назначение – возможность сохранять учетную запись пользователя в базе данных и использовать ее повторно для доступа к личным данным;

4) Зависимости – библиотека System.NET.Mail.Kit и средства работы с SMTP – сервером;

в) Функция В

1) Название – возможность распределения пользователей по группам;

2) Описание – возможность создания пользователем групп различных типов, добавления в него пользователей и разграничения прав и прав на просмотр чужих мероприятий;

3) Назначение – сохранение групп с пользователями для создания общих мероприятий и назначения задач на исполнение участникам группы.

г) Функция Г

1) Название – возможность планирования пользовательских мероприятий;

2) Описание - возможность создавать мероприятия, приглашать туда других пользователей;

3) Назначение – сохранение групповых мероприятий с ссылками на проведение их в системах видеоконференции.

д) Функция Д

1) Название – возможность создания пользовательских задач;

2) Описание – возможность прикреплять к пользователям задачи, синхронизировать их с событиями в их календарях;

3) Назначение – сохранение пользовательских задач в базе данных.

е) Функция Е

1) Название – возможность регистрирования проблем в работе системы;

2) Описание – возможность получать автоматические alert-уведомления о неисправностях в работе компонентов системы;

3) Назначение – предоставление функционала для тех поддержки по уведомлению о неисправностях системы и возможности регистрации тикетов для разработки по устранению данных неисправностей.

ж) Функция Ж

1) Название – возможность создавать автоматические и ручные отчеты о работе системы;

2) Описание – возможность создавать отчеты по задачам/мероприятиям;

3) Назначение – создание, сохранение и просмотр пользовательских отчетов в базе данных.

з) Функция З

1) Название – получать уведомления о событиях в системе;

2) Описание – возможность организации системы уведомлений и напоминаний в системе;

3) Назначение – получение mail-сообщений или push-уведомлений на свой адрес электронной почты;

4) Зависимости – библиотека System.NET.Mail.Kit и средства работы с SMTP – сервером, библиотека Firebase и средства работы с FCM.

Нефункциональные требования

а) Обеспечение уровня производительности – трафика запросов не менее 10 000 запросов к серверу в секунду (10 000 RPS);

б) Отказоустойчивость и надежность работы системы, регистрация неисправностей на сервере;

в) Масштабируемость компонентов системы;

г) Платформо-ориентированность мобильного приложения (запуск на OC Android).

# РАЗРАБОТКА ПРОГРАММНОЙ СИСТЕМЫ

## Выбранный вариант разработки программной системы

Для разработки серверной части приложения планируется использовать фреймворк ASP.NET Core на платформе .NET 7. Выбор данной платформы связан с ее удобством, наличием всего необходимого функционала для создания Restful WEB-API приложения, а также моими личными предпочтениями и опытом разработки. «В отличие от многих других платформ, ASP .NET Core имеет продвинутый набор автоматически подключаемых библиотек (Nuget пакетов), поддержку и интеграцию с мощнейшими технологическими решениями и компонентами»[8].

Для разработки клиентской части приложения можно использовать фреймворк разработки мобильных кроссплатформенных приложений - Flutter. Выбор фреймворка «Flutter связан с его популярностью на рынке, удобством и простотой использования, а также наличием огромного количества плагинов и виджетов»[12] – благодаря которым возможно воплотить все основные визуальные решения с минимальными временными затратами.

Кроме того, Flutter разрабатывается и поддерживается компанией Googlе, предоставляющей облачную систему обмена сообщений Firebase. Благодаря этому становиться возможно проводить отправку push-уведомлений на клиентские устройста пользователей используя встроенный в Google Play Services облачный proxy сервер Firebase Cloud Messaging.

Для работы с хранилищем данных удобнее всего использовать ORM – фреймворк Entity Framework и СУБД Postgre SQL. Выбор в сторону Entity Framework был сделан в результате выбора реализации серверной части приложения на платформе .NET. «Подход Entity Framework Code First позволяет создавать и обновлять таблицы базы данных в автоматическом режиме при изменении моделей хранения данных внутри проекта»[9].

Выбор в сторону Postgre SQL был сделан из-за популярности данной СУБД на рынке, а также удобством использования: как с точки зрения взаимодействия со стороны кодовой базы и ее поддержкой провайдером Entity Framework.

Также планируется использовать следующие прикладные инструменты разработки и тестирования:

а) Для разработки серверного приложения планируется использовать IDE Microsoft Visual Studio – как наиболее удобное, популярное, а также свободно распространяемое программной средство разработки приложений на .NET.

б) Для разработки мобильного клиентского приложения планируется использовать IDE Android Studio – как приложение, предложенное в качестве основного для обучения работы с мобильными приложениями, а также свободно распространяемое компанией Jet Brains – информация по использованию бралась из источника [6].

в) Для тестирования REST API HTTP-запросов планируется использовать приложение Postman – наиболее популярное и удобное приложение, используемое для задач тестирования веб-приложений.

## Описание взаимодействия приложений

Развертывание и контейнеризация серверного приложения необходимо выполнить с использованием с использованием технологии compose и программного обеспечения Docker. Он будет выполнять обязанности проведения взаимодействия и передачи данных между контейнером сервера .NET приложения, контейнером PostgreSQL базы данных, хранящей информацию о сущностях и контейнером облачной распределенной базы данных Redis, хранящей информацию о уведомлениях.

Диаграмма развертывания программной системы представлена в Приложении Г.

Диаграмма ресурсов приложения, включающая в себя такие основные компоненты, как: «Компилятор .NET приложения», «СУБД PostgreSQL», «СУБД Redis», «Docker-контейнер приложения», «HTTP-сервер поставщика», «Мобильное устройство пользователя» представлена в Приложении Д на рисунке 3.

* 1. Реализация серверного приложения

Используемые технологии

В качестве платформы для разработки серверной части приложения был выбран «ASP.NET Core благодаря его высокой производительности и кроссплатформенности» [8].

В качестве СУБД была выбрана PostgreSQL — свободная объектно-реляционная система управления базами данных. Существует в реализациях для множества UNIX-подобных платформ, включая AIX, различные BSD-системы, HP-UX, IRIX, Linux, MacOS, Solaris/OpenSolaris, Tru64, QNX, а также для Microsoft Windows.

Для хранения системных уведомлений в распределенном облачном кэше была выбрана «СУБД Redis - резидентная система управления базами данных класса NoSQL с открытым исходным кодом, работающая со структурами данных типа «ключ — значение»» [22].

Аутентификация была использована на «основе Identity от ASP.NET Core» – с моими собственными доработками (генерацией токена на основе «токеном JWT» и проверкой токенов на уровне middleware серверного приложения) [11].

Используемые библиотеки

Newtonsoft.Json – библиотека для сериализации и десериализации объект в JSON.

Serilog – сторонняя библиотека для логгирования приложения (как в консоли, так и в Google Cloud)

Entity Framework Core - современный модуль сопоставления "объект — база данных" для .NET.

xUnit - инструмент тестирования (тестирование покрывает только основные интеграционные сценарии – типа сценария регистрации. И тестируется в принципе работоспособность приложения – тест на hc (health checks)) [16].

* 1. Описание структуры серверного приложения

Разработка серверного приложения производилась на платформе ASP.NET Core. Для создания проекта под веб-приложение использовался шаблон проекта под названием ASP WEP-API. Для остальных проектов использовался шаблон библиотеки классов.

Были созданы следующие проекты:

А) Contracts (проект, в котором располагаются основные транспортные модели, необходимые для десериализации моделей запросов к API и сериализации моделей ответов данного API)

Б) Logic (проект, в котором располагаются основные инфраструктурные классы-обработчики, например, классы, отвечающие за логику создания и подтверждения учетных записей пользователя, за отправку сообщений на электронную почту и т.д.)

В) Models (проект, в котором располагаются основные доменные и сторожевые модели, необходимые для ведения процесса взаимодействия с обработанными внешними контрактами и взаимодействием с объектной моделью базы данных)

Г) PostgreSQL (проект, в котором располагаются основные классы, отвечающие за создания контекста объектно-реляционной модели базы данных, а также репозитории, отвечающие за операции с базой данных Postgre SQL).

Д) Organaizer (основной проект приложения, в котором находится основной сервис, работающий асинхронно все время с времени запуска приложения, а также точка входа в приложение. В отдельной папке лежат RESTFul – контроллеры, благодаря которым осуществляется взаимодействие по протоколу REST API).

Е) Organaizer.Tests (тестовый проект, проверяющий корректную сборку и работу сервисного веб-приложения, являющегося точкой входа в данное приложение).

Ниже на рисунке 3.2 показан скриншот обозревателя решений с полной структурой всех проектов

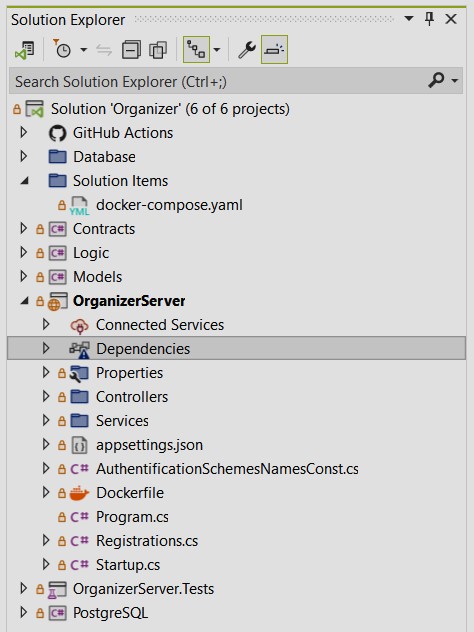


Рисунок 3.2 - Структура проектов серверного приложения

* 1. Спецификация API

Все запросы начинаются с пути к выделенному серверу порту – а именно порту <https://localhost:5201>.

Данный порт может быть переопределен при развертывании серверного приложения в контейнере – что делается в файлах compose для docker контейнеризации. Благодаря этому, запросы могут быть гибко перенаправлены – в соответствии с настроенной маршрутизацией на контроллере клиента и сервера.

Все запросы, содержащие тело – должны иметь заголовок вида:

*Content-type: Application/json*

При ошибке авторизации возвращается ошибка 401 и текстом «*Unauthorized*».

При ошибке доступа возвращается ошибка со статусом 403 и текстом «*Forbidden*» или ошибка со статусом 400 и текстом «*Bad Request*»

При ошибке исполнения запроса, связанной с данными из тела, возвращается ответ со статусом 500 –«*Internal Server Error*».

Токен пользователя (необходимый для подтверждения на сервере факта аутентификации пользователя) – должен передаваться в качестве заголовка запроса (либо альтернативный вариант – в самом теле запроса: *"token": "0895439408").*

Явную авторизацию сервер не использует (используется собственная схема аутентификации) – поэтому в запросе не требуется указывать дополнительно заголовок *Authorization: <User-token>*

Спецификацию API здесь не добавляю, по причине его большого размера и нецелесообразности его размещения в данном разделе пояснительной записки.

* 1. Описание особенностей реализации клиент-серверного приложения
     1. Реализация взаимодействия клиентского и серверного приложений

В статье «Обзор приложения, связанного с автоматизацией деятельности IT-отделов предприятия», подготовленной при разработке проекта по теме ВКР [7] – было рассмотрено описание реализации некоторых важных алгоритмов и компонентов системы.

Разработанное мобильное приложение представляет собой распределенное клиент-серверное приложение. Оно включает в себя облачное серверное приложение, которое имеет собственный жизненный цикл. Само серверное приложение вместе с контейнером базы данных развернуто в Docker – контейнере

Клиентские приложения при этом могут быть установлены пользователями на свои мобильные устройства Android – а далее при попытках пользоваться функционалом системы приложения делают HTTP-запросы к серверу и получают данные в формате JSON. Общение организовано по протоколу REST, имеет двусторонний вариант взаимодействия (то есть как сервис имеет возможность отправлять запросы к клиенту, так и клиент может отправлять запросы на сервер).

Часть из этих данных мобильное приложение сразу может отобразить для пользователя в визуальном виде, а часть – сохранить в локальной памяти устройства (закэшировать) – и затем предоставлять их при посещении связанных страниц. Например, когда пользователь делает запрос к списку своих чатов – приложение получает в запросе список всех чатов пользователя для текущей страницы (а также короткую информацию о каждом из этих чатов).

Благодаря этому приложение не делает слишком много запросов – но при этом и не заставляет единственным первым запросов перегрузить сервер и его базу данных большим набором транзакций. Кэширование и наличие локальной памяти клиентского устройства, а также подгрузка данных по «надобности» частично снимает эту проблему и не сильно перегружает базу данных – то есть дает возможность удовлетворить требование, выдвигаемое к производительности сервера ПС.

Схема работы процесса авторизации приложения приведена в Приложении Д на рисунке 1.

Также отдельно хотел бы рассмотреть логику мобильного приложения, связанного с авторизацией в сессии и получения уведомлений, как push-уведомлений, так и электронных сообщений.

Во время регистрации (или авторизации) пользователь совершает запуск Firebase-приложения на своем Android-устройстве. На облачном Google-сервере генерируется уникальный Firebase-токен – по которому пользователя становиться возможным идентифицировать в системе. Затем, когда клиентское приложение отправляет результат авторизации/регистрации – оно к тому же отправляет на сервер полученный токен, который закрепляется за пользователем и сохраняется в одной из таблиц базы данных. При этом, пользователь при авторизации с другого мобильного устройства будет генерировать новый токен и этот токен будет независимо от первого токена сохранен в базу данных.

* + 1. Реализация сохранения истории системы

В рамках реализации приложения особый упор был сделан на том, чтобы сохранять всю историю событий, происходящих в системе – это могут быть события, связанные как с аккаунтом пользователя (регистрация, авторизация, выход из системы, изменение данных/роли в системе), так и события, связанные с созданием, изменением, удалением основных сущностей системы, таких как: группа пользователей, мероприятие, задача, issue (запрос о проблеме для рассмотрения администрацией), snapshot (отчет пользователя) и alert (неисправность в работе системы).

Также внимание было уделено событиям, связанным с пользователем, например, приглашение/удаление из группы, мероприятия, назначение задачи, а также смена статуса событий, задач и запросов на терминальные статусы.

Для хранения данных о системных событиях использовался внешний распределенный кэш от Redis, который предоставляет нереляционное хранилище данных.

Алгоритм можно описать так: когда происходит какое-то изменение в базе данных – после подтверждения транзакции о его изменении создается и посылается в БД Redis событие с уникальным ключом. В самом приложении существует специальный сервис, который читает обновления по событиям, приходящим из Redis – прочитывая их, он их передает в обработчик событий, который уже на основании типа события принимает решение, что с ним делать.

Некоторые пользовательские события используются для рассылки уведомлений пользователям как по указанной ими в аккаунте электронной почте, так и по Firebase Cloud Messaging в форме push-уведомлений.

Схема алгоритма работы рассылки уведомлений представлена в Приложении Д на рисунке 2.

## Описание разработки клиентского приложения

Клиентская часть приложения была разработана с использованием фреймворка Flutter.

Описание структуры клиентского приложения

Для реализации мобильного клиентского приложения использовался фреймворк Flutter. «Данная библиотека «из коробки» предоставляет большое количество возможностей – например, реализация как Android, так и IOS мобильного приложения, а также настройку используемых пакетов и внешних библиотек и модификацию скриптов по сборке и отладке»[12].

Для создания визуальной части мобильного приложения во Flutter использовались классы-виджеты, которые были разделены на следующие группы:

а) Виджеты без состояния. К таким виджетам можно отнести следующие виджеты: виджет главной страницы (main.dart), виджет домашней страницы, предоставляющей точку входа в календарь (home.dart), виджеты для выполнения регистрации (register.dart) и авторизации (login.dart).

б) Виджеты с изменением состояния. К таким виджетам можно отнести следующие виджеты: виджет главной страницы календаря (user\_page.dart), виджет личного кабинета пользователя (user\_info\_map.dart), виджеты с добавлением/редактирование данных, а также с получением списка аггрегированных данных от сервера.

Для передачи данных между двумя HTTP-клиентами (клиентским и серверными REST API приложениями) будут использоваться json-объекты, которые для выполнения передачи между слушателями необходимо корректно обрабатывать (то есть сериализовать/десериализовать в формат json). Поэтому в клиентском приложении (в папке models) были созданы модели по аналогии с серверными моделями – задача которых состоит в том, чтобы послать запрос на сервер и получить от него ответ – информация бралась из источника [13].

Ниже на рисунке 3.3 показана структура папок и виджетов данного клиентского мобильного приложения:

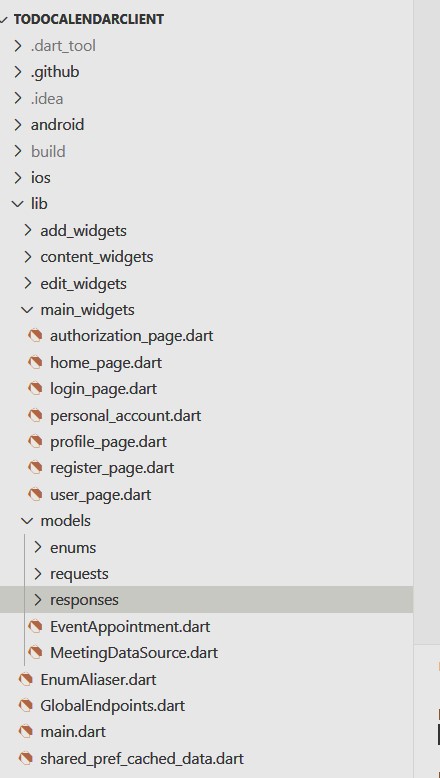


Рисунок 3.3 - Структура папок клиентского Flutter – приложения

* 1. Описание интерфейса мобильного приложения

Создадим основные страницы (виджеты) – для регистрации, авторизации, для личного кабинета пользователя (с перечислением мероприятий, групп, задач и отчетов пользователя), а также дополнительные виджеты для добавления/редактирования мероприятия в календарь, новой группы, новой задачи на реализацию, нового отчета (по мероприятиям или отчетам).

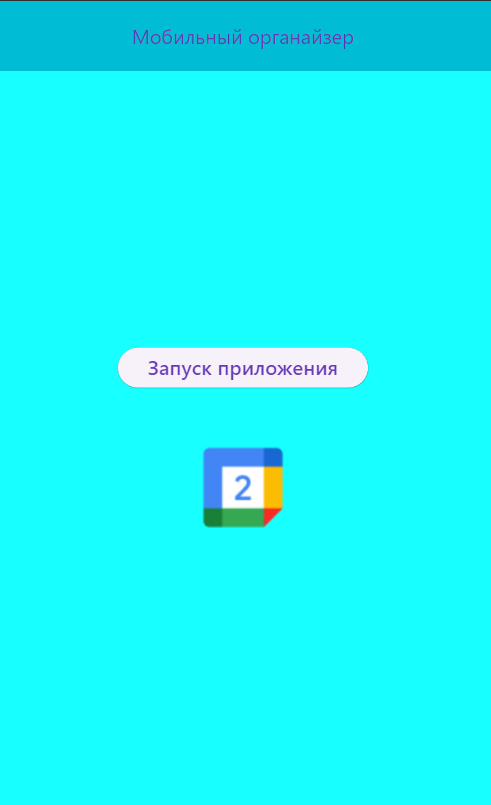
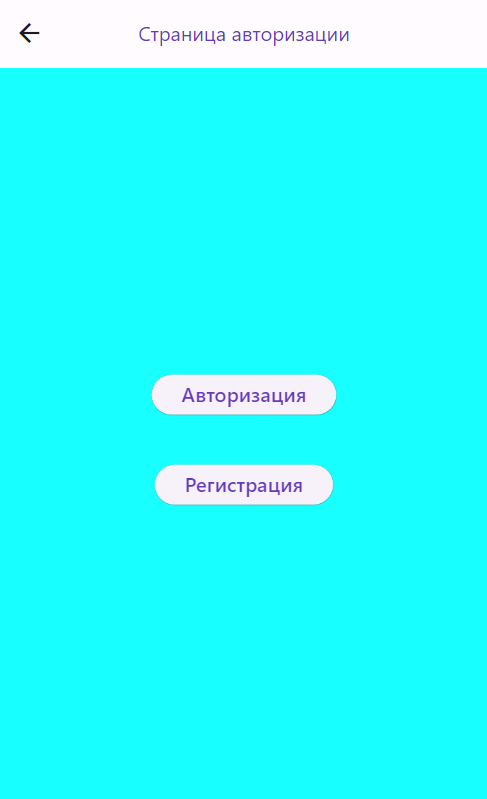
Ниже приведены компоновка скриншотов стартовой страницы и страницы для неавторизованного в системе пользователя (рис. 3.4 – 3.5):

Рисунок 3.4 - Стартовая страница приложенияРисунок 3.5 - Страница неавторизованного пользователя

В случае успешного ответа – загружается страница личного кабинета (user\_page.dart), при этом параметры идентификатора пользователя и его токен аутентификации сохраняются в локальном кэше с использованием пакета shared\_preferences.

Благодаря этому при аутентификации (входе в аккаунт) соответствующие пользователю данные в кэше будут добавляться (или обновляться), а при выходе из аккаунта – локальный кэш будет очищаться.

Краткий листинг реализации кэширования (которое производится в локальном файле текущего мобильного устройства) – информация бралась из источника [14]:

class MySharedPreferences {

static const String \_keyData = 'myData';

static const String \_keyExpiration = 'expirationTime';

Future<bool> saveDataWithExpiration(String data, Duration expirationDuration) async {

try {

SharedPreferences prefs = await SharedPreferences.getInstance();

DateTime expirationTime = DateTime.now().add(expirationDuration);

await prefs.setString(\_keyData, data);

await prefs.setString(\_keyExpiration, expirationTime.toIso8601String());

print('Data saved to SharedPreferences.');

return true;

} catch (e) {

print('Error saving data to SharedPreferences: $e');

return false;

}

}

Future<String?> getDataIfNotExpired() async {

try {

SharedPreferences prefs = await SharedPreferences.getInstance();

String? data = prefs.getString(\_keyData);

String? expirationTimeStr = prefs.getString(\_keyExpiration);

if (data == null || expirationTimeStr == null) {

print('No data or expiration time found in SharedPreferences.');

return null; // No data or expiration time found.

}

DateTime expirationTime = DateTime.parse(expirationTimeStr);

if (expirationTime.isAfter(DateTime.now())) {

print('Data has not expired.');

return data;

} else {

await prefs.remove(\_keyData);

await prefs.remove(\_keyExpiration);

print('Data has expired. Removed from SharedPreferences.');

return null;

}

} catch (e) {

print('Error retrieving data from SharedPreferences: $e');

return null;

}

}

Страницы авторизации и регистрации представлены на рисунках 3.6 – 3.7 ниже. На данных страницах происходит валидация вводимого текста:

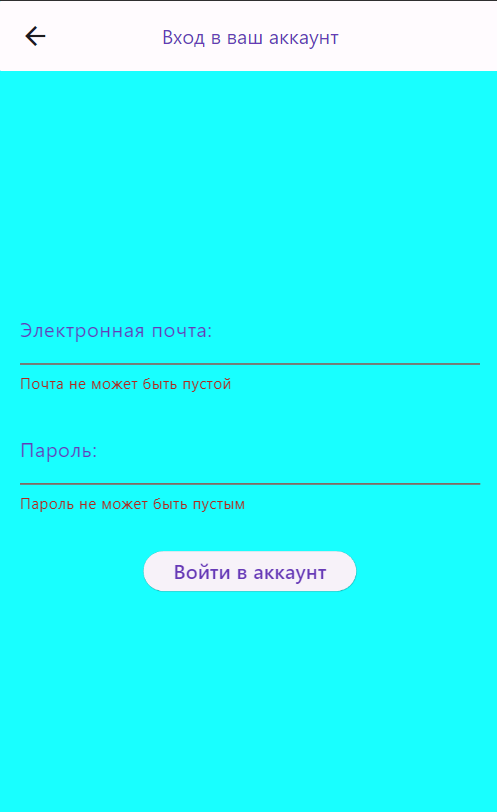


Рисунок 3.6 - Страница авторизации

Рисунок 3.7 - Страница регистрации

Главная страница пользователя, представляющая собой пример «виджета с изменением состояния (в зависимости от изменения локального кэша меняется ее содержимое» [23] – в случае отсутствия данных о авторизованном пользователе – осуществляется перенаправление пользователя на страницу для авторизации).

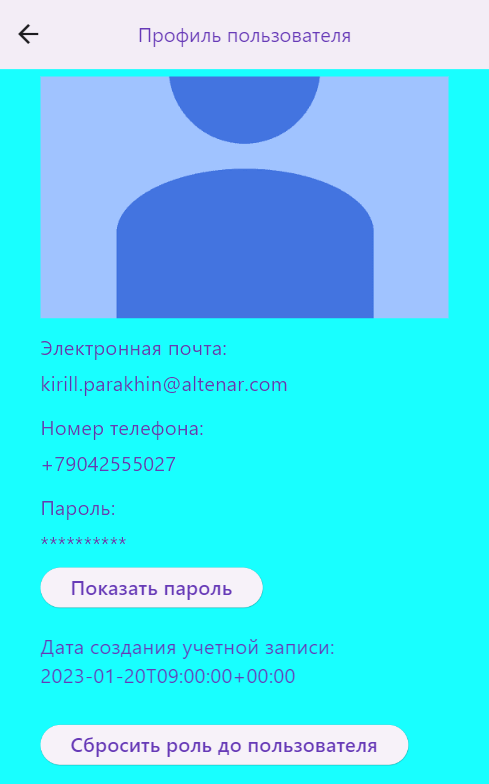
На главной странице пользователя (рисунок 3.8) и странице личного кабинета пользователя (рисунок 3.9) присутствует нижнее навигационное меню с различными режимами текущего окна. По умолчанию, открывается главная страница. При нажатии на другую панель – откроется другое состояние виджета.

Рисунок 3.8 – Главная страница пользователяРисунок 3.9. Профиль пользователя

По аналогии, есть виджеты для работы с группами, задачами и отчетами текущего пользователя – получаемые в качестве результата на GET-запрос по текущему идентификатору пользователя.

Также присутствует инструмент личного кабинета пользователя – для просмотра информации о текущих мероприятий в календаре пользователя, списка его групп, задач и отчетов.

С него можно перейти на виджеты по просмотру календаря мероприятий пользователя, списка его групп, задач, запросов и отчетов.

Страница с календарными мероприятиями пользователя и выпадающими диалоговыми виджетами представлены ниже на рисунках 3.10 – 3.11:

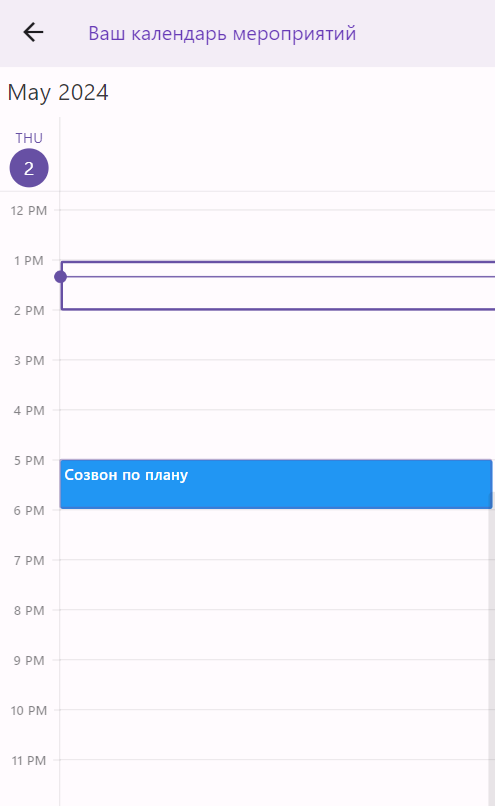
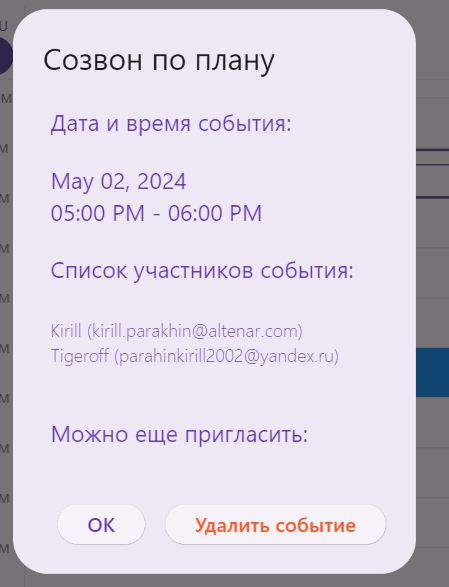


Рисунок 3.10 – Виджет с календарными мероприятиями

Рисунок 3.11 – Виджет с информацией о текущем мероприятии

Мероприятия настраиваются и показываются в режиме дневного просмотра. При этом время в календаре используется локальное время мобильного устройства пользователя (+3 UTC, по умолчанию).

При этом сам виджет календаря и выделенные в нем мероприятия интерактивны – при клике на них можно посмотреть подробную информацию о них – а также увидеть список участников – информация бралась из источника [16].

Фрагмент листинга получения информации о мероприятиях пользователях с API сервера:

var json = jsonDecode(cachedData.toString());

var cacheContent = ResponseWithToken.fromJson(json);

var userId = cacheContent.userId;

var token = cacheContent.token.toString();

var model = new UserInfoRequestModel(userId: userId, token: token);

var requestMap = model.toJson();

var url = Uri.parse(userInfoUri);

final body = jsonEncode(requestMap);

try {

final response = await http.post(url, headers: headers, body: body);

var jsonData = jsonDecode(response.body);

var responseContent = GetResponse.fromJson(jsonData);

if (responseContent.result) {

var userRequestedInfo = responseContent.requestedInfo.toString();

var data = jsonDecode(userRequestedInfo);

var userEvents = data['user\_events'];

var fetchedEvents =

List<EventInfoResponse>

.from(userEvents.map(

(data) => EventInfoResponse.fromJson(data)));

setState(() {

eventsList = fetchedEvents;

});

}

Страница со списком групп пользователя представлена ниже на рисунке 3.12, страница со списком пользователей группы – на рисунке 3.13.

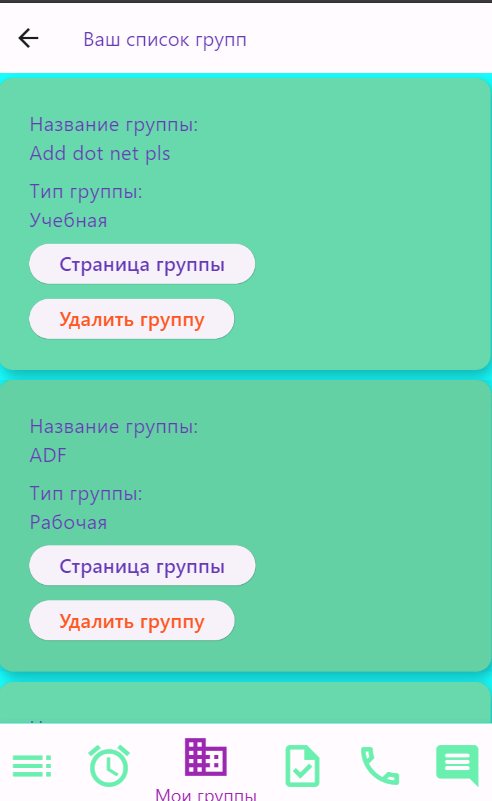
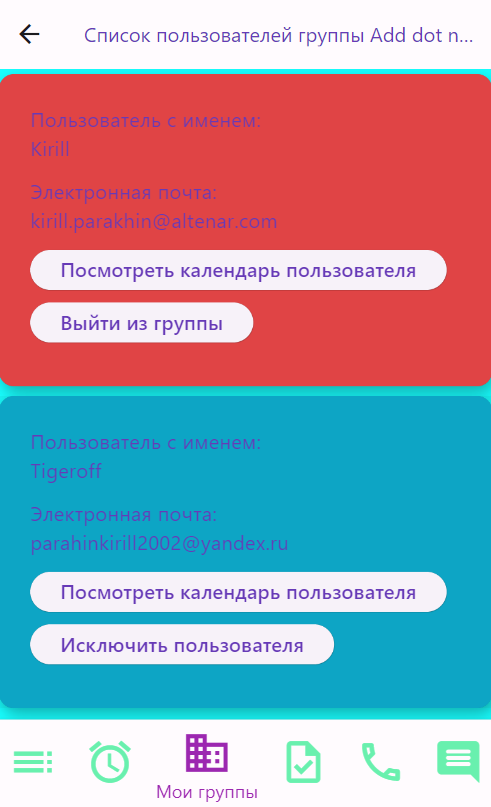


Рисунок 3.12 – Страница со списком групп пользователя

Рисунок 3.13 – Страница со списком пользователей в группе

Ниже на рисунках 3.14 – 3.15 показана страница менеджера группы, на которой представлены отчеты по работе группы и страница конкретного отчета группы:

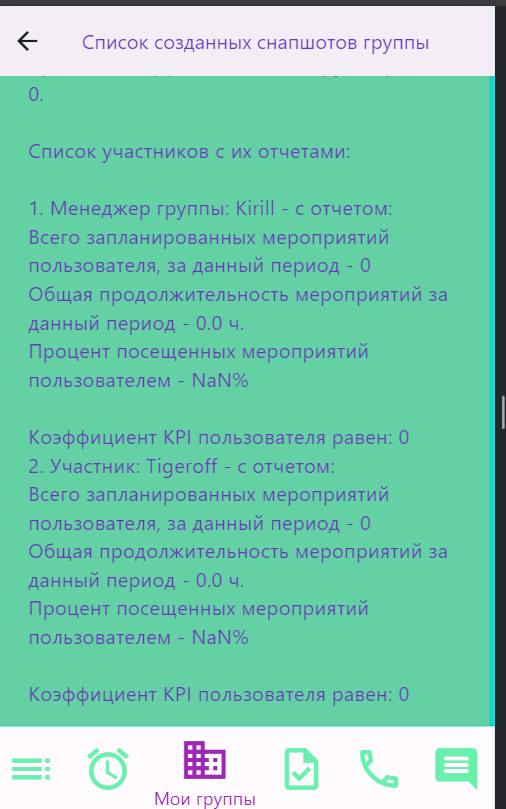
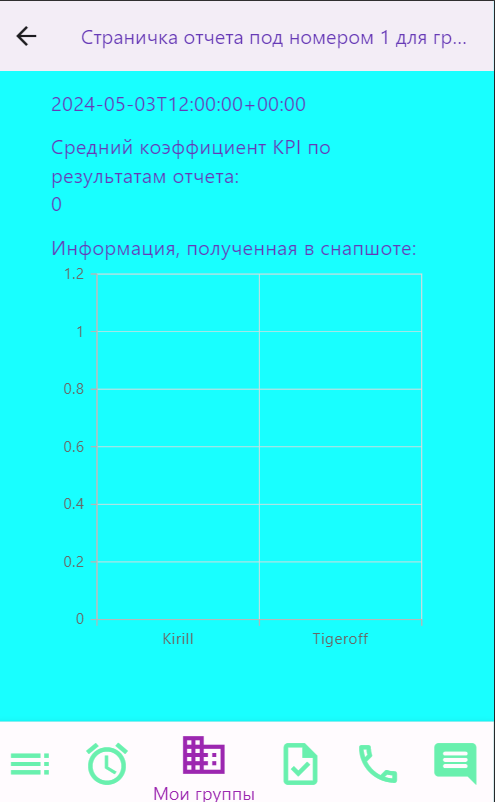


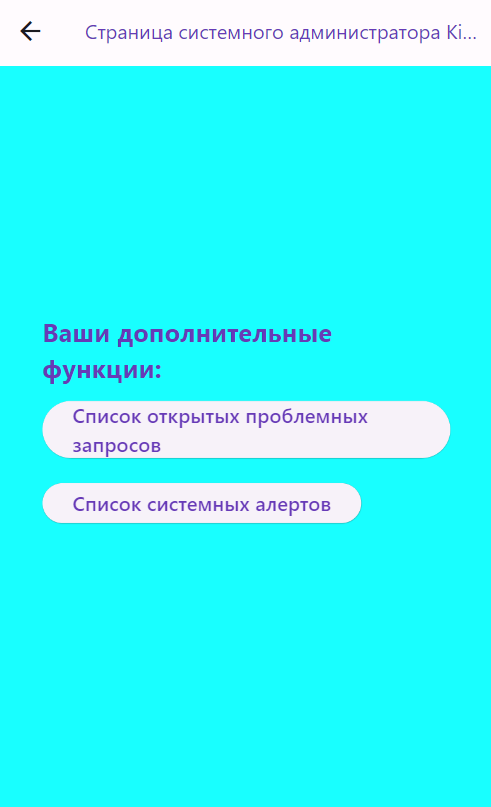
Рисунок 3.14 – Страница со списком отчетов менеджера группы

Рисунок 3.15 – Страница отдельного отчета менеджера

Пользователь может создавать отчеты, как о собственной деятельности, так и о деятельности группы (данная опция доступна исключительно для менеджера проекта). Отчет может создаваться согласно определенному критерию для анализа (это или участие пользователя в мероприятиях, либо его исполнение задач, либо участие в регистрации и исправлении проблем, связанных с работой системы).

На вход подается круг лиц, для которых нужно создать отчет (возможно, только себя) – и промежуток времени, за который создается отчет – на выходе приложение представляет список данных в форме вычисленных значений KPI пользователя(ей) - и представляет их либо в текстовом виде, либо в виде диаграммы (гистограммы – для чего использовался специальный виджет от фреймворка Flutter, предназначенный для визуализации данных).

Также рассмотрим функционал системного администратора. На странице администратора, на которую он может зайти через страницу профиля, он может просматривать и редактировать присланные ему запросы о проблемах, созданные конкретными пользователями – а также просматривать системные алерты о зарегистрированных неисправностях в работе системы – на рисунках 3.16 – 3.17 представлены скриншоты общей страницы администратора и скриншот страницы алертов – которых может не быть, если ошибок в работе не выявлено:



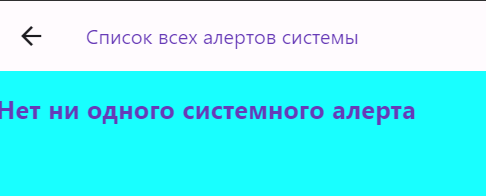


Рисунок 3.17 – Страница со списком системных алертов

Рисунок 3.16 – Страница системного администратора

## Реализация push-уведомлений

Push-уведомления для Android – приложения были реализованы с помощью использования библиотеки Firebase, на клиенте создается сессия Firebase – приложения, на сервере происходит рассылка уведомления с помощью соответствующего nuget – пакета, осуществляющего отправку текущего системного сообщения в FCM.

В статье «Обзор приложения, связанного с автоматизацией деятельности IT-отделов предприятия», подготовленной при разработке проекта по теме ВКР [7] – была подробно рассмотрена логика рассылки уведомлений в системе.

Рассмотрим, как работает логика рассылки уведомлений в системе. На сервере существует отдельный Background сервис, который существует с самого старта приложения и работает в режиме 24/7. Одной из его зон ответственности является проверка изменений по сущностям базы данных (например, информации о мероприятии – приглашение нового участника, наступление начала мероприятия, окончание мероприятия или его перенос).

Реализация отправки уведомлений на электронную почту или прямиком на устройства пользователей практически ничем не отличается, кроме как использованием разных клиентов – SMTP-клиента для создания email-сообщений и Firebase-клиента для создания push-уведомлений (рассылка происходит на авторизованные мобильные устройства благодаря сохранению токенов).

Информация по реализации клиента Firebase и отправки сообщений из .NET приложения в Firebase Cloud Messaging бралась из источника [19].

## Тестирование системы

* + 1. Тестирование запросов к REST API серверного приложения

Для тестирования HTTP-запросов, поступающих на вход и уходящих на выход – для серверного приложения использовался программный продукт Postman. Ниже в таблице 3.1 представлен отрывок из спецификации API по тестированию путей регистрации/авторизации.

Таблица 3.1 - Запросы по регистрации/авторизации пользователей

|  |  |
| --- | --- |
| **Регистрация пользователя** | |
| Метод | URL |
| POST | /users/register |
| Тело запроса | |
| {  "email": "parahinvaleri5@gmail.com",  "name": "Tigeroff1",  "password": "tigeroff2002",  "phone\_number": "8-903-255-50-27"  } | |
| Ответ | |
| Статус | Тело |
| 200 ОК | {  "result": true,  "out\_info": "Code confirmation was performed for user with email parahinvaleri5@gmail.com with code: 009709  } |
| **Авторизация пользователя** | |
| Метод | URL |
| POST | /users/login |
| Тело запроса | |
| {  "email": "parahinvaleri5@gmail.com",  "password": "tigeroff2002"  } | |
| Ответ | |
| Статус | Тело |
| 200 ОК | {  "result": true,  "out\_info": "Login existed user Tigeroff1 with new auth token 2112168000"  } |

* + 1. Модульное и интеграционное тестирование серверного приложения

Для основного сервиса приложения NotifyService было написано несколько модульных тестов на создание экземпляра зависимостей с помощью DI-контейнера .NET. Также были написаны интеграционные тесты, проверяющие, что сервисное веб-приложение ASP.NET Core может быть собрано и запущено, после чего будет успешно принимать запросы на конечные эндпоинты API.

«Для создания интеграционного теста, проверяющего корректный запуск приложения использовался паттерн веб-фабрики WebApplicationFactory, который основывается на методах регистрации зависимостей приложения, подключаемых в стартовом файле Startup.cs»[15].

Ниже приведен листинг одного из тестов на проверку health checks приложения (который дает подтверждение тому, что приложение может быть запущено и может давать корректные ответы при обращении к нему):

[Fact(DisplayName = $"{nameof(NotifyService)}" +

$" can health check.")]

[Trait("Category", "Integration")]

public async Task CanHealthCheckAsync()

{

// Arrange

using var client = \_factory.CreateClient();

string? response = null;

// Act

var exception =

await Record.ExceptionAsync(async () =>

response = await client.GetStringAsync("/hc"));

// Assert

exception.Should().BeNull();

response.Should().NotBeNullOrWhiteSpace()

.And.Be("Healthy");

}

private readonly WebApplicationFactory<Startup> \_factory;

Скриншот Test Explorer из IDE Visual Studio представлен ниже на рисунке 3.18 – на нем изображено окно, в котором в графическом виде представлены модульные и интеграционные тесты для сервисного проекта приложения:

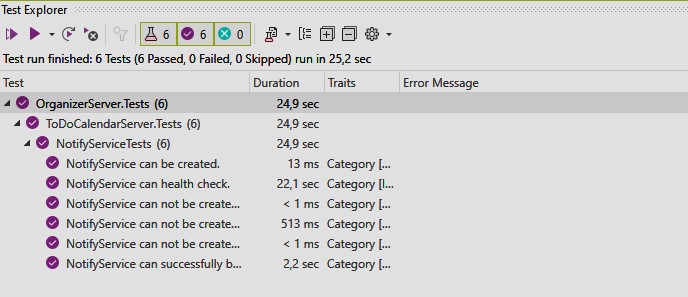


Рисунок 3.18 - Окно Теst Explorer в Visual Studio

* 1. Развертывание приложения
     1. Развертывание сервера

Для использования серверного приложения – его удобно развернуть в отдельном контейнере, например, Docker – контейнере.

«Чтобы произвести развертывание приложения, необходимо ввести в терминале проекта команду: docker compose –up build»[18].

При этом во время развертывания с помощью Dockerfile был создан образ серверного приложения (под названием organizer-app), а также образ базы данных postgres\_container (при этом при создании контейнера – к базе данных будут применены основные миграции по созданию таблиц и наполнению их некоторыми исходными данными).

По итогу, получается контейнер приложения, к которому можно получить доступ в локальной сети по адресу url: 4040:4040 – ниже на рисунке 3.19 представлен скриншот контейнеров, выполненный в программе Docker Desktop.

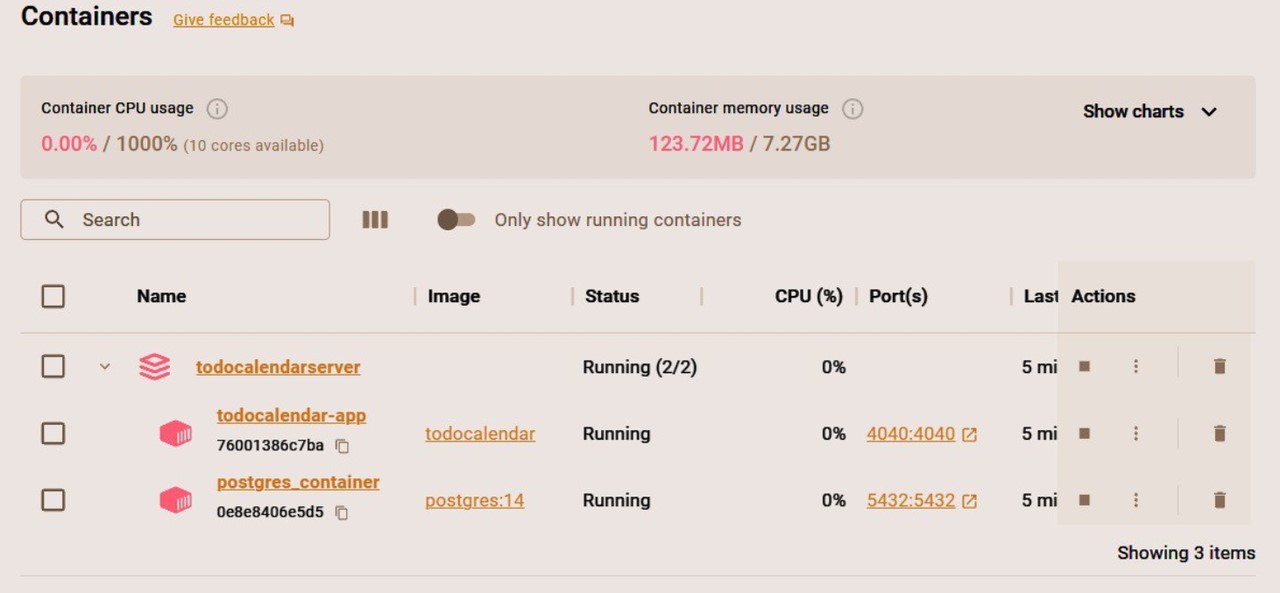


Рисунок 3.19 - Развертывание сервера приложения и базы данных в Docker – контейнере

* + 1. Развертывание мобильного приложения

После проведения разработки и тестирования логики работы мобильного клиент-серверного приложения – необходимо собрать его в виде исполняемого файла для определенной мобильной операционной системы – в данном случае будем использовать OC Android.

Полученный исполняемый файл (с расширением файла .apk) - его возможно запустить как в эмуляторе, так и на реальном мобильном устройстве (развертывание мобильного приложения Flutter, выполненное в IDE Android Studio представлено ниже на рисунке 3.20).

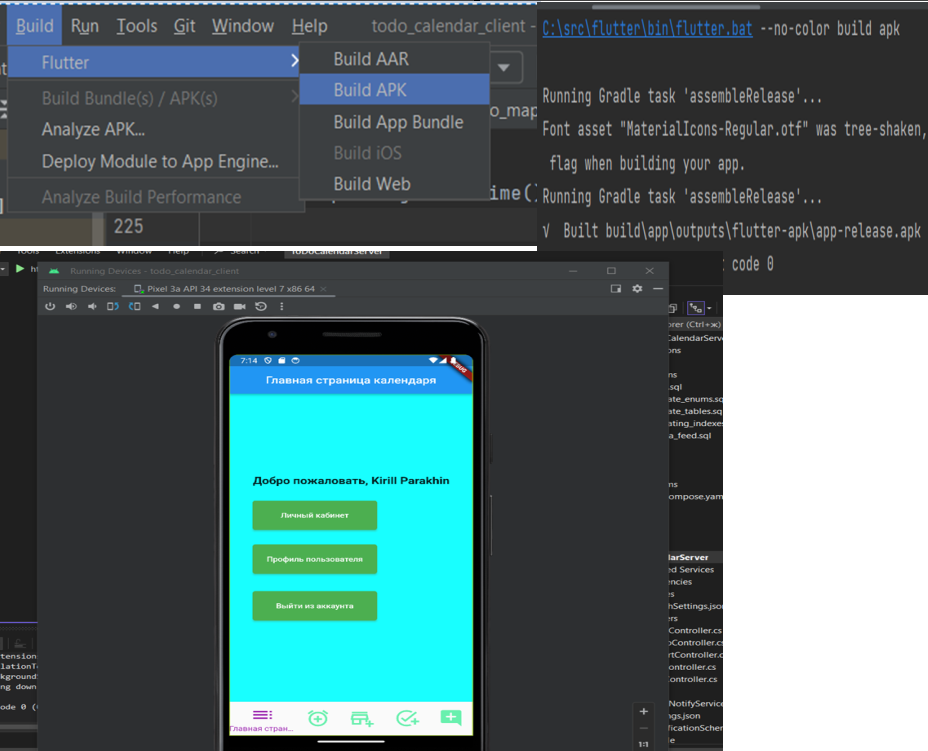


Рисунок 3.20 - Развертывание мобильного приложения в Android Studio

* 1. Ведение репозиториев программной системы

Вся разработка велась с использованием системы контроли версий git и сервиса для хранения удаленных репозиториев Github, на котором размещен код моей программной системы по ссылке: <https://github.com/Tigeroff2002/ToDoCalendarServer> (рис. 3.21 - 3.22 – представлены скриншоты репозиториев с серверной и клиентской частью разрабатываемого курсового проекта, расположенные на сайте GitHub).

Для разработки использовался базовый функционал, предлагаемый системой git. Но, несмотря на то, что разработка велась не в команде (то есть она проводилась только мной) – все равно реализация функционала велась в отдельных ветках, на каждую из задач или изменение создавались коммиты.

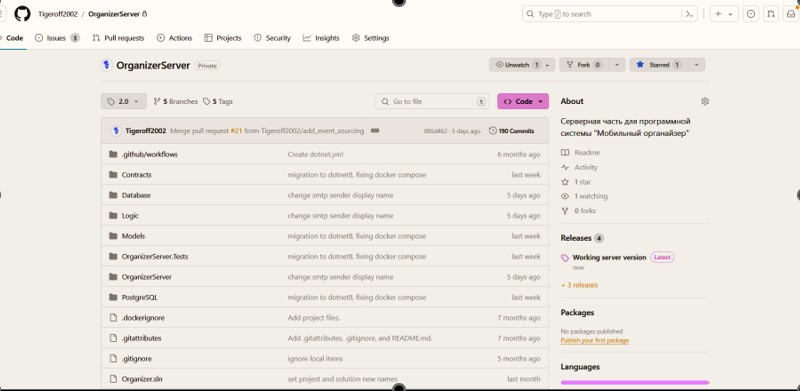


Рисунок 3.21 - Репозиторий серверного ASP.NET - приложения

Репозиторий клиентского приложения доступен на Github по ссылке: <https://github.com/Tigeroff2002/ToDoCalendarClient>

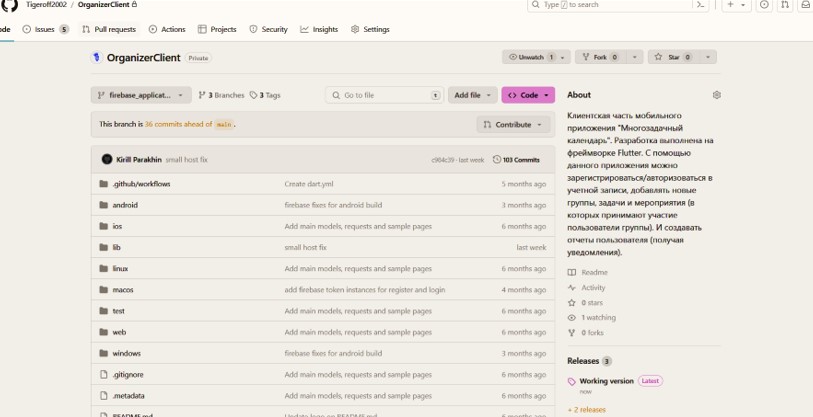


Рисунок 3.22 - Репозиторий клиентского Flutter – приложения

По мере внесения изменений – проводилось автоматическая удаленная сборка серверного приложения, а также его развертывание в Docker – контейнере.

По окончанию разработки были созданы релизы – то есть готовые собранные приложения, готовые к скачиванию и прикладному использованию.

«Вместе с каждым релизом публикуются текстовые примечания, ссылки на бинарный файл и исходный код. Зайдя в раздел Releases, пользователь всегда может найти последнюю версию программы, changelog и полную историю версий. Ссылка на релизы помещена на главную страницу проекта»[19].

1. ИНФОРМАЦИОННЫЙ МЕНЕДЖМЕНТ

Информационный менеджмент рассматривается как совокупность принципов, методов и форм управления информационным процессом, описывающая информационное окружение (пространство) лица, принимающего решение (ЛПР) и его проблемное поле [21].

К задачам информационного менеджмента относятся [21]:

1. формирование технологической среды ИС;
2. развитие ИС и обеспечение ее обслуживания;
3. организация обработки информации;
4. планирование работ в среде ИС;
5. выполнение работ по обработке информации;
6. управление персоналом в сфере информатизации;
7. экономика (управление капиталовложениями);
8. формирование инновационной политики и осуществление инновационных программ;
9. формирование и обеспечение комплексной защищенности информационных ресурсов.

Управление проектом разработки ПС мобильного приложения «Мобильный органайзер» относится к задаче «экономика (управление капиталовложениями)» и включает разработку следующих вопросов:

1. Обоснование проекта;
2. Анализ окружения проекта;
3. Основные положения устава проекта;
4. Содержание проекта;
5. Организационная структура проекта;
6. Календарный план проекта в *MS Project;*
7. План управления рисками проекта;
8. План управления изменениями;
9. Управленческая отчетность по проекту.

**Обоснование проекта**

«Мобильный органайзер» - это программная система, которая соединяет в себе основные функции, которые необходимы для создания единой иерархической структуры проекта предприятия, проведения взаимодействия между сотрудниками, а также создания автоматизированной отчетной и мониторинговой системы по отслеживанию основных положительных и негативных инцидентов функционированию программного продукта компании.

Практически каждая современная коммерческая компания имеет хотя бы один IT-отдел, деятельность которого необходимо оптимизировать – чтобы не тратить на некоторые бизнес-процессы отдельные человеческие временные ресурсы.

Для организаций среднего масштаба зачастую встает выбор о принятии решения по разработке собственного программного продукта, который способен объединить все необходимые функции

Благодаря этому, компания перестает тратить средства на оплату подписки за пользование платным сторонним программным обеспечением, а также может сэкономить средства на выплату зарплату сотрудникам, чья деятельность сможет быть автоматизирована.

Основными бизнес-причинами возникновения необходимости в разработке данного проекта в рамках данной организации являются:

1. Отсутствие единой системы, которая обеспечивает хранение данных сотрудников организации;
2. Отсутствие автоматической системы ведения отчетности
3. Отсутствие системы мониторинга отказоустойчивости собственного программного продукта
4. Отсутствие единой системы обмена информации и бизнес-сообщениями в системе
5. Отсутствие грамотной оптимизации отдела аналитики и продуктового бизнеса

В следующей таблице (таблице 4.1) приведена матрица структурированных бизнес-выгод, которые имеют место в случае успешной реализации проекта.

Таблица 4.1 - Матрица структурированных бизнес-выгод.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | Характер воздействия на бизнес | | |
| Создание новых возможностей | Повышение эффективности операций | Отказ от операций |
| Степень определенности | Финансовые | Сокращение части сотрудников мониторинга | Увеличение качества услуг мониторинга засчет их автоматизации | Отказ от ручных действий по регистрации инцидентов в работе системы |
| Количественные | Хранение и структуризация данных. | Уменьшение времени обработки объектов | Отказ от бумажной работы, за исключение печати кадровых и зарплатных документов |
| Измеримые | Уменьшение времени на планирование мероприятий и задач | Возможность быстрого менеджмента и трекинга событий в системе | Отказ от взаимодействий между людьми на прямую и хранения системо-ориентированных данных у сотрудников |
| Качественные | Централизованное хранение данных | Оперативный доступ к данным | Отказ от децентрал. хранения данных |

**Анализ окружения проекта**

Для проекта программной системы «Мобильный органайзер» рассмотрим, какие внутренние и внешние факторы окружения оказывают влияние.

К внутренним факторам относятся:

• Стиль управления — влияет на принятие ключевых решений в процессе использования и обновления проекта. От того, какие цели преследует руководство проекта, зависит качество результата.

• Организация проекта — должна быть выстроена четкая схема развития проекта для получения наибольшей выгоды.

• Работники — развивать проект должны люди, имеющие определенный опыт в данной сфере деятельности и заинтересованные в ней.

К внешним факторам относятся:

• Законы и право. Основное влияние на проект, оказывает статья от 27.12.1991 N 2116-1 (ред. от 06.08.2001) "О налоге на прибыль предприятий и организаций".

• Технологии. С развитием технологий целесообразно будет развивать проект внедряя их.

Все перечисленные факторы оказывают воздействие на реализацию проекта. Необходимо учитывать их при разработке проекта, сроков его реализации и расчете бюджета проекта – эти меры снизят негативное действие факторов внешней и внутренней среды и помогут успешно реализовать проект.

В таблице 1 в приложении Е представлены заинтересованные стороны проекта.

Разработчик проекта заинтересован в создании напрямую, так как это повысит его квалификацию, увеличит объем знаний, полученных в результате исследования, проектирования и разработки. Владелец продукта заинтересован в получении готовой высокопроизводительной продукции, которая реально поможет провести оптимизацию бизнес-процессов работы проекта.

На рисунке 4.1 ниже представлена карта заинтересованных сторон проекта:

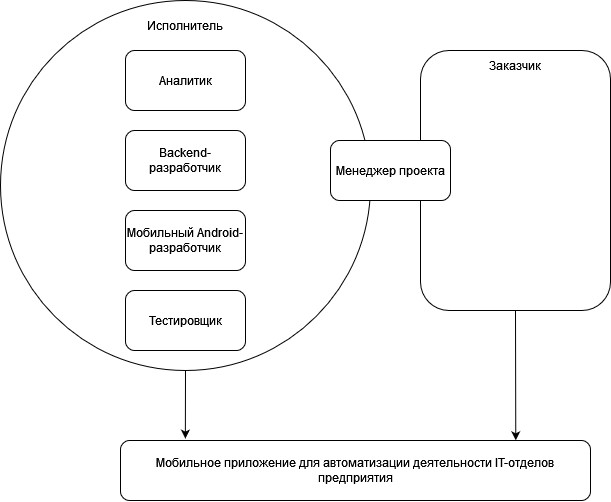


Рисунок 4.1 - Заинтересованные стороны

**Основные положения устава проекта**

Устав проекта - основополагающий документ проекта. Разработка Устава проекта - процесс, который относится к группе процессов инициации проекта.

Для повышения вероятности разделяемой ответственности, приемки результатов проекта, а также удовлетворения заказчиков и других заинтересованных сторон проекта их необходимо привлекать в процессы инициации проекта.

В Уставе проекта документируются первоначальные требования к проекту, удовлетворяющие потребностям и ожиданиям заинтересованных сторон.

Устав проекта мобильного приложения для автоматизации деятельности IT-отделов предприятия формируется из необходимости формулирования всех условий реальной текущей картины функционирования проекта.

Основные положения устава проекта ПС «Мобильный органайзер» приведены в таблице 2 в приложении Е.

**Содержание проекта**

Жизненный цикл проекта представлен ниже на рисунке 4.2.

Разработчик переходит от одной стадии к другой строго последовательно. Сначала полностью завершается этап «Разработка требований», в результате чего получается список требований к ПО. После того как требования полностью определены, происходит переход к проектированию, в ходе которого создаются документы, подробно описывающие для программистов способ и план реализации указанных требований. После того, как проектирование полностью выполнено, программистами выполняется реализация полученного проекта. После того, как реализация завершена, производится тестирование и отладка продукта; на этой стадии устраняются все недочёты, появившиеся на предыдущих стадиях разработки. После этого программный продукт внедряется и обеспечивается его поддержка — внесение новой функциональности и устранение ошибок.

Переход от одной фазы разработки к другой происходит только после полного и успешного завершения предыдущей фазы.

Фазы жизненного цикла подробно представлены в Приложении Ж на рисунках 1-2.

Также в таблице 3 приложения Е приведен словарь иерархической структуры работ для данного проекта ПС «Мобильный органайзер».

**Смета проекта**

Смета проекта и связанные с ней расчеты приведены в таблице 4 в приложении Е. По итогам рассчетов стоимость проекта равна 511 240 руб.

**Организационная структура проекта**

Ниже в таблице 4.2 представлено штатное расписание проекта, отражающее необходимость в ресурсах, а на рисунке 4.2 ниже приведена организационная диаграмма проекта.

Таблица 4.2 – Штатное расписание проекта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Должность | Структурное подразделение | Количество штатных единиц | Тарифная ставка руб/час |
| 1 | Backend - разработчик | Отдел разработки | 1 | 400 |
|  | Мобильный Android-разработчик | Отдел разработки | 1 | 350 |
| 2 | Аналитик | Отдел аналитики | 1 | 360 |
| 3 | Менеджер проекта | Отдел аналитики | 1 | 600 |
| 4 | Тестировщик | Отдел тестирования | 1 | 350 |

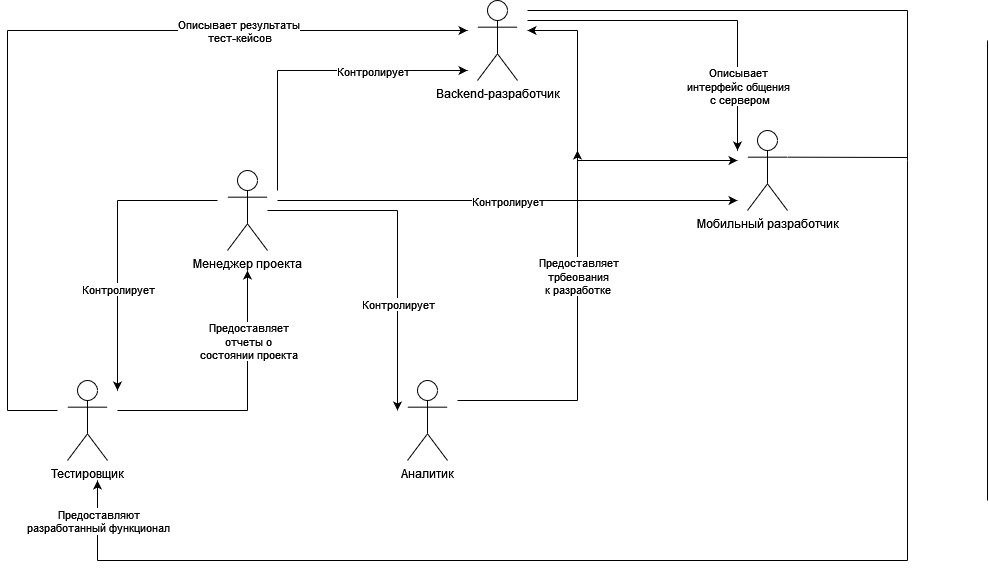


Рисунок 4.2 - Организационная диаграмма проекта

Ниже в таблице 4.3 показаны условные обозначения матрицы отвественности, а в таблице 5 в приложении Е – приведена сама матрица ответственности для сотрудников проекта

Таблица 4.3 - Условные обозначения матрицы ответственности (RACI).

|  |  |  |
| --- | --- | --- |
| Обозначение | Расшифровка | Описание |
| Исп. (R) | Исполнитель (Responsible) | Несет ответственность за непосредственное исполнение задачи. К каждой задаче должно быть приписано не менее одного исполнителя |
| Утв. (A) | Утверждающий (Accountable) | Отвечает за конечный результат перед вышестоящим руководством. На каждую работу должен быть назначен строго один подотчётный |
| Cогл. (C) | Согласующий (Consulted) | Согласует принимаемые решения, взаимодействие с ним носит двусторонний характер |
| Н. (I) | Наблюдатель (Informed) | Его информируют об уже принятом решении, взаимодействие с ним носит односторонний характер |

**Календарный план проекта**

В таблице 6 в приложении Е представлен детализированный календарный план реализации проекта по разработке ПС «Мобильный органайзер»

Диаграмма Ганта, построенная в MS Project согласно описанным в таблице 4.9. задачам проекта представлена ниже на рисунке 4.3:

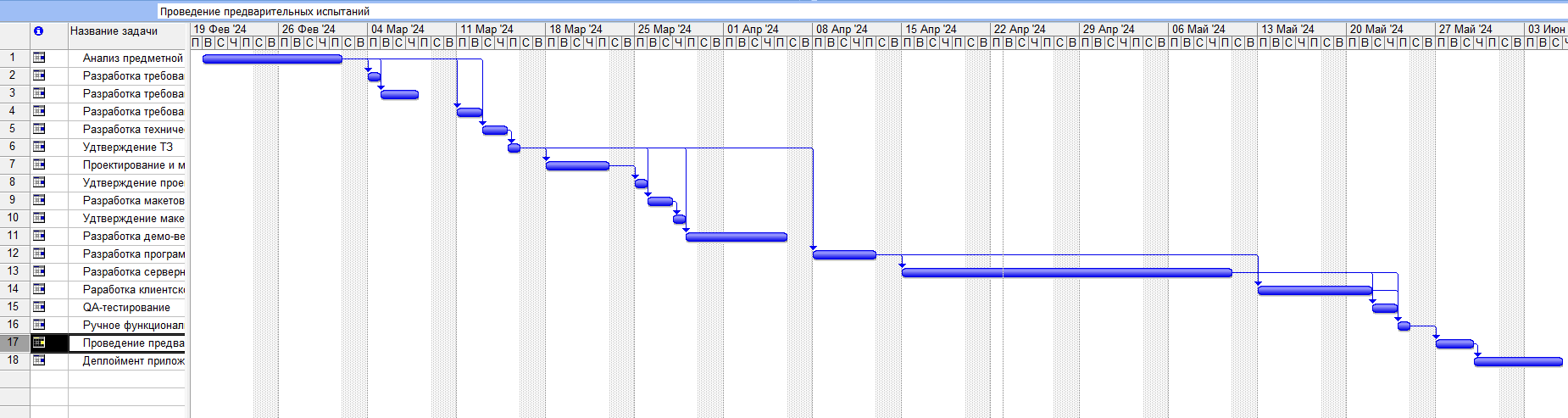


Рисунок 4.3 - Диаграмма Ганта

**План управления рисками**

Управление рисками проекта включает в себя процессы, связанные с определением, анализом и реагированием на риски проекта.

Цели управления рисками проекта – повышение вероятности возникновения и воздействия благоприятных событий и снижение вероятности возникновения и воздействия, неблагоприятных для проекта событий.

Для того, чтобы представить риски управления проектом следует разработать шкалу оценки вероятности рисков. Шкала оценки вероятности рисков представлена ниже в таблице 4.4.

Далее следует разработать шкалу оценки последствий. Шкала оценки последствий представлена ниже в таблице 4.5.

Таблица 4.4 - Шкала оценки вероятности возникновения рисков.

|  |  |  |
| --- | --- | --- |
| Интервал вероятностей | Словесная формулировка | Числовая оценка |
| 1% - 20% | Низкая | 1 |
| 21% - 40% | Ниже среднего | 2 |
| 41% - 60% | Средняя | 3 |
| 61% - 80% | Выше среднего | 4 |
| 81% - 99% | Высокая | 5 |

Таблица 4.5 - Шкала оценки последствий.

|  |  |  |  |
| --- | --- | --- | --- |
| Перерасход  средств | Отставание  от расписания | Описание | Числовая оценка |
| До 5% | 1 неделя | Небольшая потеря  производительности | 1 |
| До 15% | 2 недели | Умеренное снижение производительности | 2 |
| Более 30% | 1 месяц | Серьезный ущерб для производительности | 3 |

На основе имеющихся данных следует построить матрицу оценки риска.

Матрица оценки риска приведена ниже в таблице 4.6.

Таблица 4.6 - Матрица оценки риска.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 |
| 1 | 1 | 2 | 3 | 4 | 5 |
| 2 | 2 | 4 | 6 | 8 | 10 |
| 3 | 3 | 6 | 9 | 12 | 15 |

Как видно из реестра рисков наиболее опасными являются риски под номерами 4 (Нехватка средств) и 6 (Ошибка в планировании проекта).

Для того, чтобы снизить вероятность их появления, менеджер проекта должен составить подробный план расходов, и заложить дополнительные расходы на возникновение незапланированных ситуаций.

Подробный план управления рисками представлен в виде схемы ниже на рисунке 4.4:

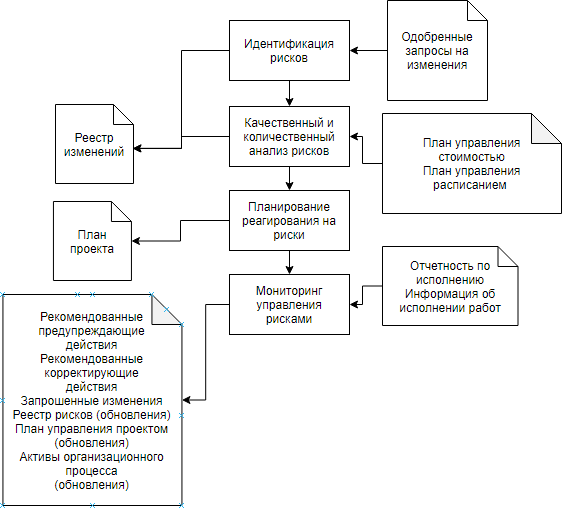


Рисунок 4.4 - План управления рисками

**План управления изменениями**

Любой из участников проекта — заказчик или исполнитель может инициировать запросы на изменение. Любые из этих запросов на функциональное преобразование должны быть задокументированы и пройти через процесс контроля за реализацией изменений.

В общем виде данный процесс регламентирует прохождение изменений через следующие стадии – ниже на рисунке 4.5:

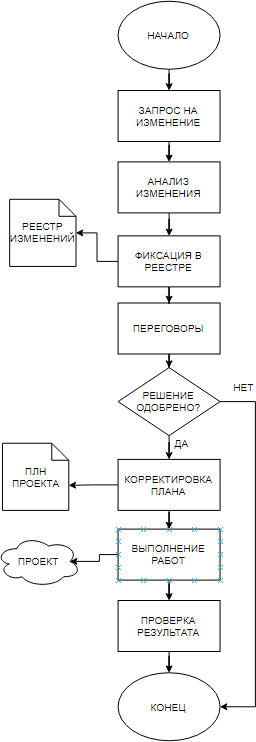


Рисунок 4.5 - Подробная схема процесса изменения.

Далее приведено подробное описание каждого пункта:

а) Запрос на изменение:

Инициатор высказывает требование, которое выходит за рамки проекта и является изменением. Это еще не сигнал к действию, а пока что только запрос.

б) Анализ изменения:

Этот процесс позволяет понять, как предлагаемое изменение повлияет на проект. Также на этапе анализа проводится оценка последствий, что будет, если изменение принять и что проект потеряет, если отказаться от изменения.

в) Фиксация запроса в реестре изменений:

Любые запросы на изменение нужно фиксировать в специальном документе «Реестр запросов на изменение» (или Реестр изменений), где содержится список всех запросов на изменения.

г) Переговоры:

Как правило, одна из сторон настаивает на внесение изменения в проект, а другая противится этому. Поэтому нужно провести переговоры, обсудить варианты реализации и отклонения предлагаемого изменения и принять решение по нему.

д) Корректировка плана:

Внести изменения в базовый план (первоначальный план проекта), так как изначально работа над проектом планировалась по одному сценарию, а теперь этот сценарий изменился. В рабочий план, который должен являться вашим навигатором по проекту.

е) Выполнение работ:

Так как изменение внесено в план, то оно является частью проекта. Работаем с ним, как с обычными задачами проекта. Следует выполнить работы.

ж) Проверка результата:

Если работы приняты, то работы по реализации изменения считаются завершенными.

**Управленческая отчетность проекта**

Сначала нужно удостовериться, что все работы выполнены надлежащим образом и их результаты соответствуют плановым. Полученная информация фиксируется в документе «О выполнении плана работ по проекту», который позже включается в итоговый отчет о реализации проекта.

В итоговый отчет о реализации проекта следует включить описание целей проекта, требований к его результатам, а также перечень первоначальных параметров и характеристик (из бизнес-плана), изменений, внесенных в проект, и их причин.

Важной частью итогового отчета являются описание полученных результатов и раскрытие показателей эффективности.

Ответственность за подготовку данного документа можно закрепить за менеджером проекта.

Модель отчетности показателей для оценки фактической продуктивности проекта разработки ПС «Мобильный органайзер» представлена ниже на рисунке 4.6.

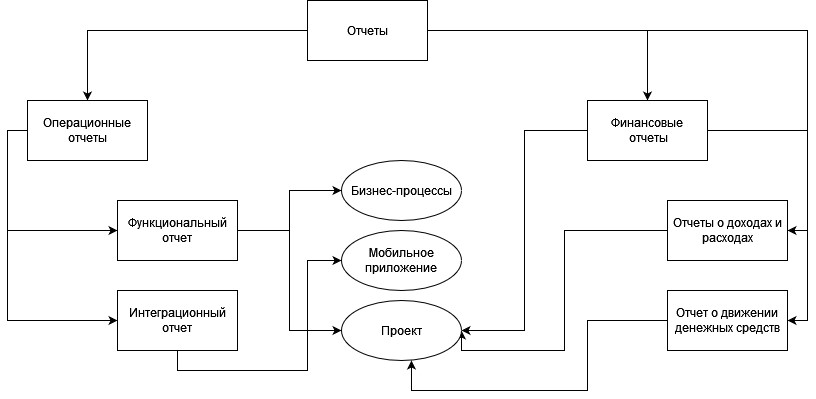


Рисунок 4.6 - Модель отчетности показателей для оценки фактической продуктивности

Во время разработки выделяются следующие виды отчетов:

а) Статус - отчет по проекту

Статус - отчет по проекту необходим для сообщения статуса проекта. Он должен быть достаточно подробным, чтобы можно было оценить процент готовности проекта.

б) Реестр рисков

Реестр рисков содержит результаты идентификации рисков, данные их анализа и планы реагирования для наиболее существенных рисков.

Обзор рисков проекта может включаться в Статус-отчет по проекту как одна из ключевых метрик.

Например, можно указать общее количество еще существующих рисков и классифицировать их по степени негативного влияния на проект.

в) Реестр изменений

Реестр изменений содержит список всех запросов на изменения (увеличение или уменьшение содержания проекта, изменение стратегий, процессов, планов или процедур, изменение цен или бюджетов, или пересмотр расписаний).

**Подведение выводов о результате ввода системы в использование**

Результаты решения комплекса задач по автоматизации процессов мониторинга, составления и аналитики отчетов в IT-отделе текущей организации повлияли на управление предприятием по следующим направлениям:

а) Обеспечение автоматического ведения отчетности

Было: каждый отчет требовалось создавать вручную отдельному аналитику (при этом опираясь зачастую на данные, полученные со слов и записей сотрудников – то есть в теории не самых достоверных).

Стало: добавилась система, которая автоматически выполняет отчеты за определенные периоды времени (неделя/месяц).

б) Обеспечение оперативности получения статистики

Было: статистику по рабочей деятельности сначала создавал руководитель команды, затем мог передать ее лично сотруднику.

Стало: любой сотрудник может получить статистику своей работы

в) Сокращение операций, выполняемых вручную

Было: каждый новый инцидент, влияющий на работоспособность системы необходимо было вручную отслеживать, регистрировать и передавать разработчикам.

Стало: добавилась автоматическая система регистрации инцидентов и создания уведомлений в форме alert – сообщений, посылаемых как мониторингу, так и разработчикам.

г) Обеспечение единого синхронизированного хранилища данных

Было: ведение данных в системе велось не для всех процессов или событий, такие вещи, как текущий статус задачи или инцидента могли иметь у разных сотрудников в один и тот же момент разные состояния.

Стало: за счет внедрения обновляемого синхронизированного хранилища данных, все данные, поступаемые и обрабатываемые в систему корректно обновляются и отображаются для сотрудников в рамках «снимка» текущего состояния базы данных и системных событий.

д) Снижение трудоемкости ввода информации

Было: каждый новый инцидент или задача заполнялись вручную отделом мониторинга, отсутствовала единая форма заполнения.

Стало: добавилась система с автоматическим заполнение полей и данных при создании новых задач или инцидентов.

е) Обеспечение хорошего уровня анализа и аудита

Было: статистику сначала создавал руководитель группы, затем передавал управлению.

Стало: любой человек с необходимым уровнем доступа может сам сгенерировать статистику в программе.

ж) Отказ от использования платных подписок

Было: для разных типов задач (трекинг задач, ведение календаря мероприятий, мониторинг системы, аудит системы) приходилось использовать ряд платных сервисов.

Стало: появилась единая программная система (за использование которой не требуется вносить денежные средства), объединяющая в себе разные функции, необходимые для деятельности IT-организации.

# Заключение

В ходе выпускной квалификационной работы по теме «Программная система «Мобильный органайзер» было разработана программная система, представляющее собой мобильное приложение для организации и автоматизации деятельности IT-отделов предприятия.

Были достигнуты следующие результаты:

* + Проведен анализ предметной области;
  + Были рассмотрены основные аналоги, затем проведен их сравнительный анализ;
  + Выделены функциональные и нефункциональные требования программной системы;
  + Проведено проектирование системы;
  + Определены и обоснованы средства разработки программной системы;
  + Осуществлена разработка серверного приложения и базы данных;
  + Произведена разработка мобильного приложения для операционной системы Android;
  + Программная системы развернута и протестирована различными методами;
  + Был подготовлен информационный менеджмент проекта разработки и внедрения ПС.

# Список использованных источников

1. Бёрнс Б. Распределенные системы. Паттерны проектирования. — СПб.: Питер, 2019 — 224 с.: ил. — (Серия «Бестселлеры O’Reilly»). ISBN 978-5-4461-0950-0.
2. Колисниченко Д. Н. Программирование для Android. Самоучитель. — 3-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2020. — 288 с.: ил. ISBN 978-5-9775-6587-5.
3. Рихтер Дж. Clr Via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. — СПб.: Питер, 2013. — 896 с.: ил. — (Серия «Мастер-класс»). ISBN 978-5-496-00433-6.
4. Hans-Jürgen Schönig. Mastering PostgreSQL 13. 1-е изд, 2020 - 459 c. ISBN 978-1-80056-749-8.
5. Моуэт Э. Использование Docker / пер. с англ. А. В. Снастина; науч. ред. А. А. Маркелов. — М.: ДМК Пресс, 2017. - 354 с.: ил. ISBN 978-5-97060-426-7.
6. Пирская Л. В. Разработка мобильных приложений в среде Android Studio. 2019. №1. С.120-230. ISBN 978-5-9275-3346-6.
7. Парахин К.В. Обзор приложения, связанного с автоматизацией деятельности IT-отделов предприятия, 2024. №1. С.86-90. ISBN 978-5-398-03137-9 URL: [http://www.bf.pstu.ru/images/documents/nauka/sborniki/MNRR 2024\_s\_oblozhkoy.pdf](http://www.bf.pstu.ru/images/documents/nauka/sborniki/MNRR%202024_s_oblozhkoy.pdf) (дата обращения: 15.03.2024)
8. Общие сведения об ASP.NET Core [Электронный ресурс]. – 2024. Режим доступа: <https://learn.microsoft.com/ru-ru/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0> (дата обращения: 12.01.2024 г.).
9. Руководство. Начало работы с Entity Framework 6 Code First [Электронный ресурс]. – 2024. Режим доступа: <https://learn.microsoft.com/ru-ru/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application> (дата обращения: 04.02.2024 г.).
10. 10 Best IT Automation Software Reviews 2024 [Электронный ресурс]. – 2024. Режим доступа: <https://www.softwaretestinghelp.com/best-it-automation-tools/> (дата обращения: 17.02.2024 г.).
11. Авторизация с помощью определенной схемы в ASP.NET Core [Электронный ресурс]. – 2024. Режим доступа: [https://learn.microsoft.com/ru ru/aspnet/core/security/authorization/limitingidentitybyscheme?view=aspnetcore-8.0](https://learn.microsoft.com/ru-ru/aspnet/core/security/authorization/limitingidentitybyscheme?view=aspnetcore-8.0) (дата обращения: 20.02.2024 г.).
12. Введение и установка Flutter [Электронный ресурс]. – 2023.

Режим доступа: <https://habr.com/ru/articles/311412/> (дата обращения: 10.03.2024 г.).

1. Создание Flutter Rest API [Электронный ресурс]. – 2023.

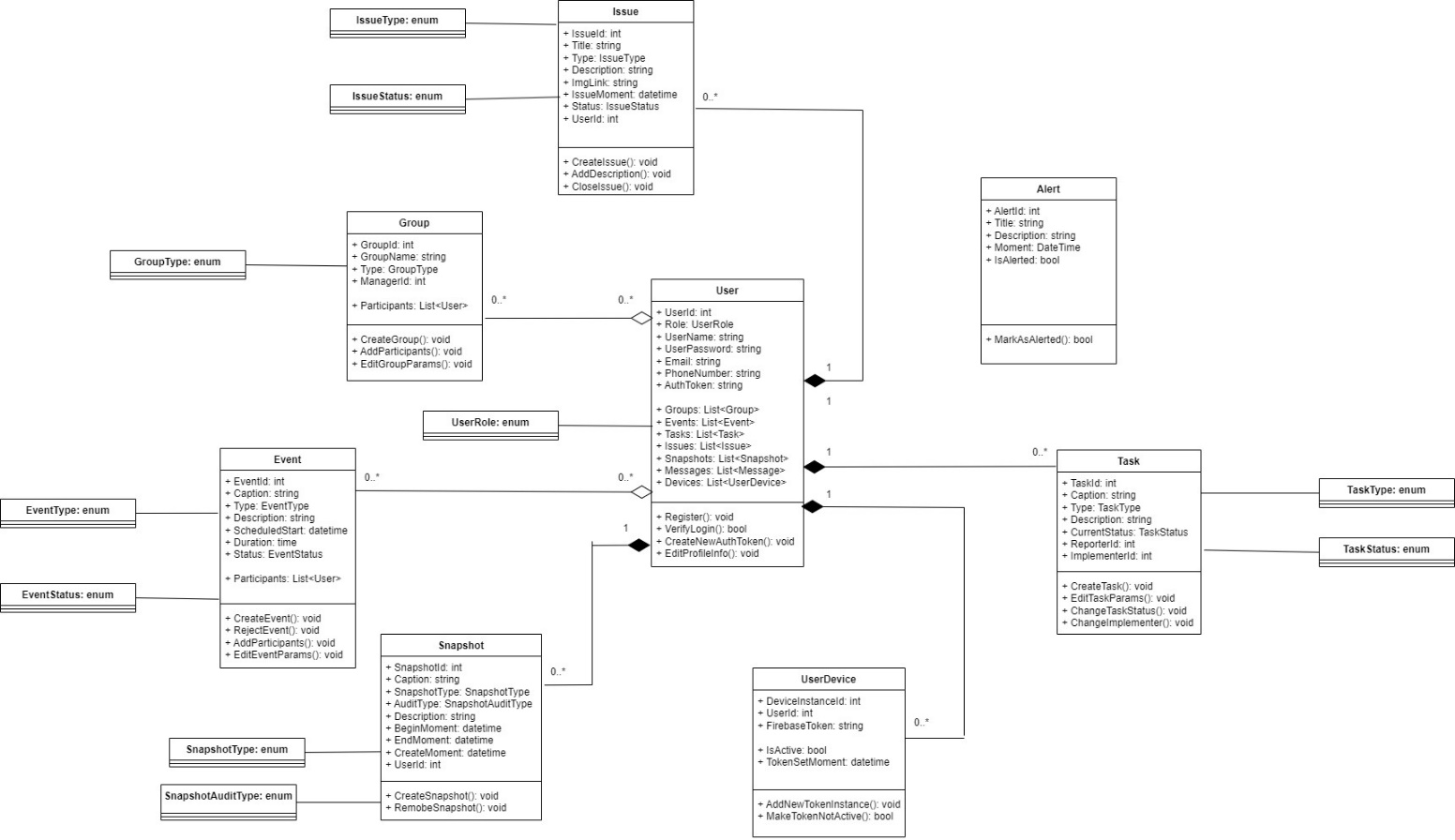
Режим доступа: <https://www.c-sharpcorner.com/learn/learn-flutter/flutter-rest-api> (дата обращения: 17.03.2024 г.).

1. Кэширование данных на устройстве Flutter [Электронный ресурс]. – 2023. Режим доступа: <https://medium.com/@ahmedsherif4175/easiest-way-to-cache-data-in-flutter-a-step-by-step-guide-7e983d64198b> (дата обращения: 24.03.2024 г.).
2. Интеграционные тесты на платформе ASP.NET Core [Электронный ресурс]. – 2024. Режим доступа: <https://learn.microsoft.com/ru-ru/aspnet/core/test/integration-tests?view=aspnetcore-8.0> (дата обращения: 28.03.2024 г.).
3. Работа с Flutter Syncfusion Calendar для показа виджета календаря [Электронный ресурс]. – 2023. Режим доступа: <https://help.syncfusion.com/flutter/calendar/getting-started> (дата обращения: 31.03.2024 г.).
4. Работа с Flutter SfCartesian Chart для построения гистограмм [Электронный ресурс]. – 2023. Режим доступа: <https://help.syncfusion.com/flutter/cartesian-charts/chart-types/histogram-chart> (дата обращения: 07.04.2024 г.).
5. Руководство по Docker Compose для начинающих [Электронный ресурс]. – 2022. Режим доступа: <https://habr.com/ru/companies/ruvds/articles/450312/> (дата обращения: 14.04.2024 г.).
6. Push – уведомления в Android с помощью Firebase Cloud Messaging [Электронный ресурс]. – 2023. Режим доступа: <https://habr.com/ru/articles/302002/> (дата обращения: 21.04.2024 г.).
7. Github Releases: публикация релизов [Электронный ресурс]. – 2022. Режим доступа: <https://habr.com/ru/articles/185444/> (дата обращения: 06.05.2024 г.).
8. Костров А. В. Основы информационного менеджмента: Учеб. пособие / А. В. Костров. – 2-е изд., перераб. и доп.  -  М.: Финансы и статистика, 2009. – 528 с. - ISBN 5-279-02314-0.
9. Джозайя Карлсон. Redis in Action / 1-е изд, 2013 - 320 c. ISBN 78-1617290855.
10. Эрик Уиндмилл. Flutter in Action / 1-е изд, 2019 - 432 c. ISBN 978-5-97060-221-8.

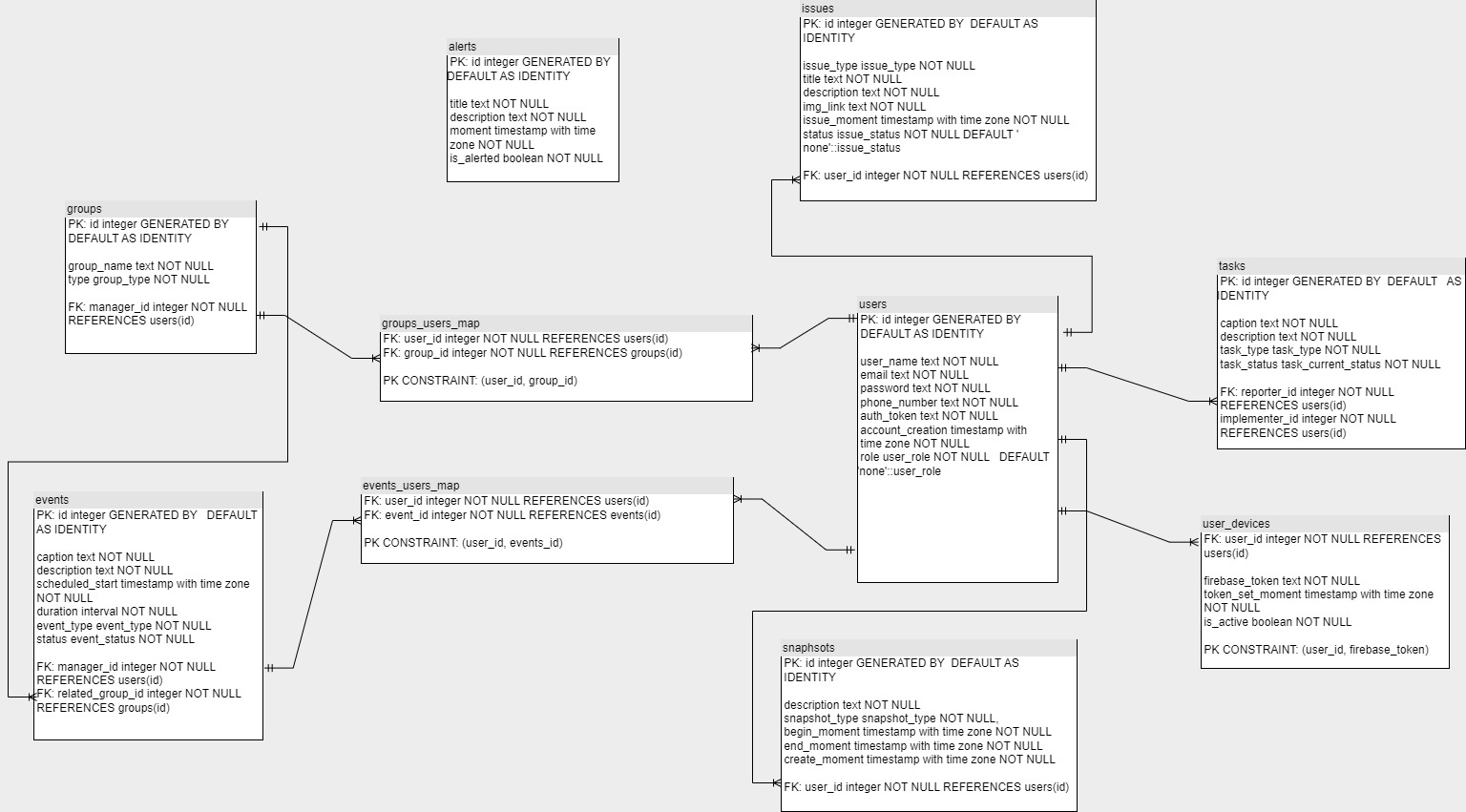
# ПРИЛОЖЕНИЕ А. ДИАГРАММА ПРЕЦЕДЕНТОВ СИСТЕМЫ



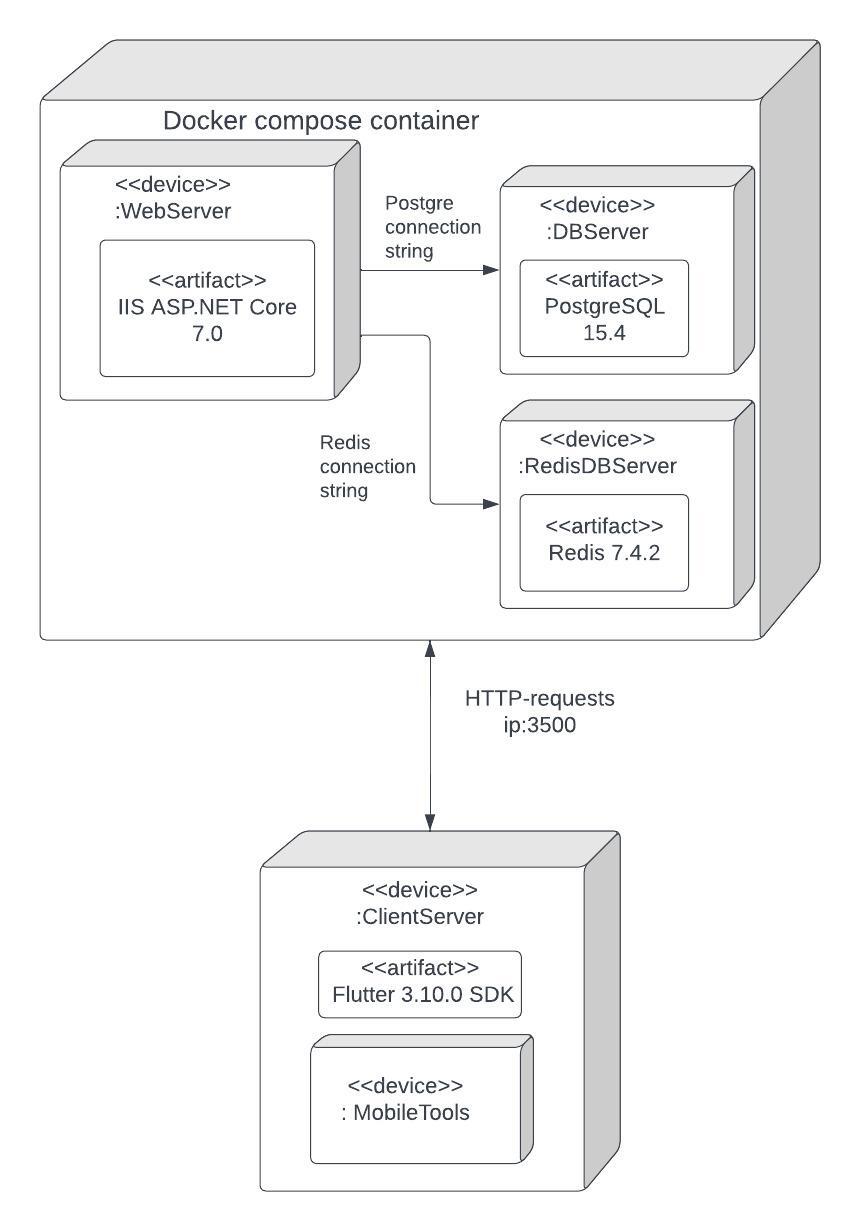
ПРИЛОЖЕНИЕ Б. ЛОГИЧЕСКАЯ СХЕМА БАЗЫ ДАННЫХ



ПРИЛОЖЕНИЕ В. ФИЗИЧЕСКАЯ СХЕМА БАЗЫ ДАННЫХ



ПРИЛОЖЕНИЕ Г. ДИАГРАММА РАЗВЕРТЫВАНИЯ ПРИЛОЖЕНИЯ



ПРИЛОЖЕНИЕ Д. СХЕМЫ ПРОГРАММ

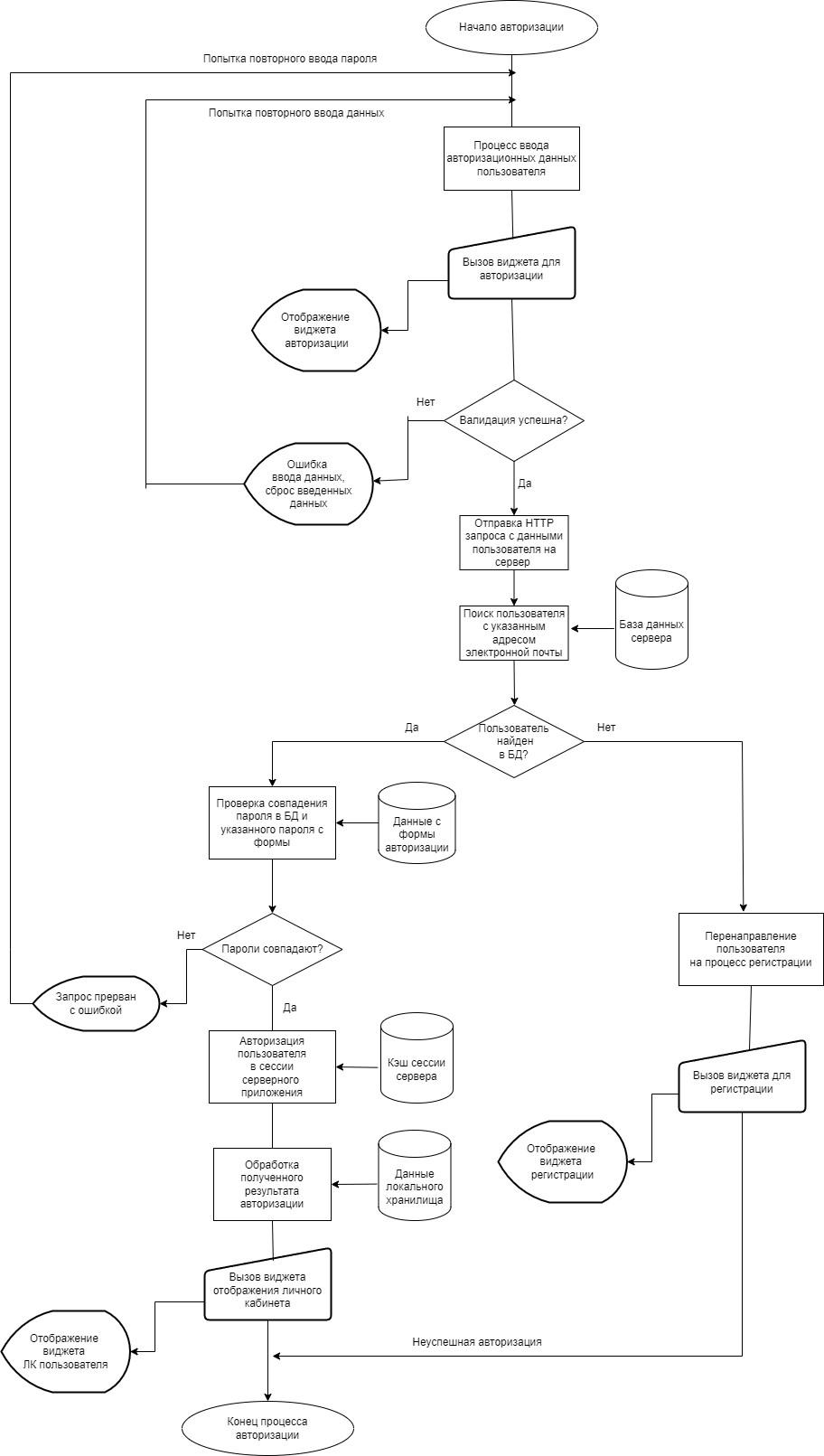


Рисунок 1. Схема алгоритма авторизации пользователя

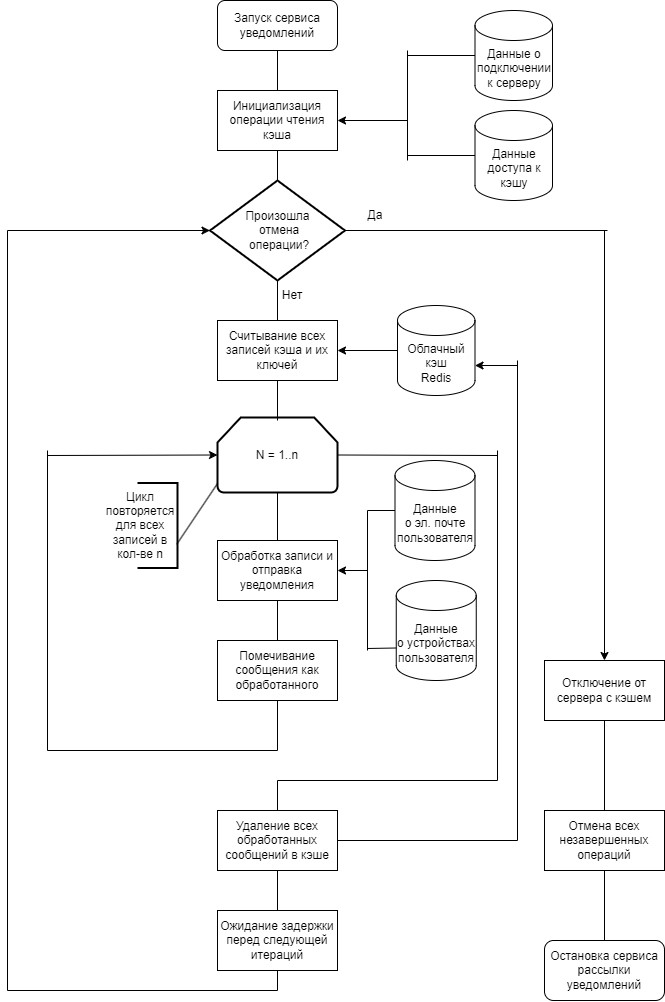


Рисунок 2. Схема рассылки уведомлений

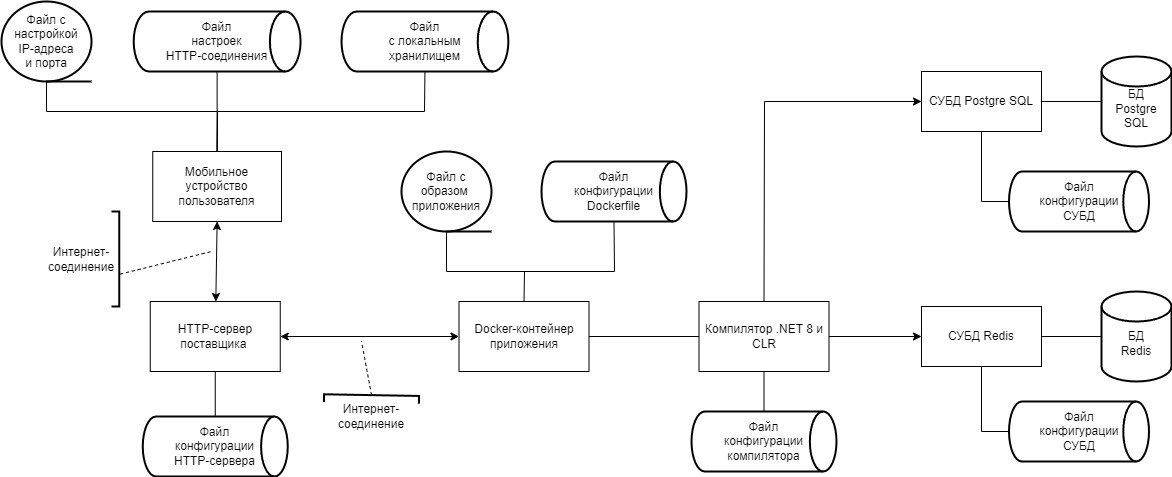


Рисунок 3. Схема ресурсов системы

ПРИЛОЖЕНИЕ Е. ИНФОРМАЦИОННЫЙ МЕНЕДЖМЕНТ

Таблица 1 – Группы заинтересованных лиц

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Группа | ФИО | Должность | Роль в проекте | Влияние на проект | Полномочия | Интерес к проекту |
| Исполнитель\Заказчик | Емельянов Д.В. | Менеджер проекта | Формирование требований. Контроль процесса разработки | Влияние на разработку | Любые действия, связанные с управлением проектом | Высокая степень заинтересованности в реализации. |
| Исполнитель | Савичев А.В. | Аналитик | Анализ и доработка требований заказчика | Влияние на разработку | Любые действия, связанные с архитектурой и анализом требований | Высокая степень заинтересованности. Исследовательская работа |
| Исполнитель | Парахин К.В. | Backend разработчик | Разработка серверной части проекта | Влияние на разработку | Любые действия, связанные с разработкой серверного  приложения | Высокая степень интереса в реализации |
| Исполнитель | Парахин К.В. | Мобильный Android- разработчик | Разработка клиентской части проекта | Влияние на разработку | Любые действия, связанные с разработкой мобильного приложения | Высокая степень интереса в реализации |
| Исполнитель | Парахин К.В. | Тестировщик | Проверка разработанного кода на соответствие всем требованиям | Влияние на разработку | Любые действия, связанные с архитектурой и анализом требований | Ожидание реализации приложения для тестирования |

Таблица 2 – Основные положения устава проекта

|  |  |
| --- | --- |
| Цели проекта | Автоматизация и оптимизация существующих бизнес-процессов функционирования IT-проекта: совершенствование деятельности IT-отдела, создание общей иерархической структуры проектов и их сотрудников внутри организации, ведение системы отчетности и системы мониторинга инцидентов, хранение данных о сотрудниках и событиях в системе, создание системы своевременного обмена сообщениями и уведомлениями в системе, рассылка уведомлений |
| Результаты проекта | Заказчик получает автоматизированную систему, которая:   1. Создает единое оптимизированное хранилище всех данных проекта 2. Автоматизирует и упрощает работу отдела аналитики и мониторинга 3. Автоматизирует систему отчетности и регистрации любых событий 4. Обеспечивает своевременный обмен информации в системе путем рассылки уведомлений |
| Допущения связаны с управлением рисками проекта | Возможность корректировки информации от менеджера в ходе разработки проекта. |
| Ограничения | Разработка мобильного приложения должна производиться с помощью фреймворка Flutter. Разработка серверной части производится строго на платформе .NET с использованием СУБД Postgre SQL и Redis |
| Расписание основных контрольных событий | Время начала и окончания проекта  С 20.02.2024 по 31.05.2024, работы могут быть продлены по требованию заказчика. |
| Бюджет проекта | Бюджет проекта, выделенный заказчиком (включающий в себя все виды расходов) составляет 511 240 рублей (расчеты произведены в рамках составления сметы проекта в таблице 4.5 данного приложения) |
| Критерии приемки | 1. Время обработки одного запроса пользователя не должно занимать более 1 секунды; 2. Все данные должны сохраняться в базах данных на сервере приложения; 3. Время формирования отчетов не должно быть более 2 секунд. |
| Обоснование полезности | После разработки и внедрения мобильного приложения сократится время работы отдела бизнес-аналитики и руководителей (team-leadов проектов). Также автоматизируется деятельность отдела мониторинга – поэтому станет возможным освободить часть рабочих мест. Cоставление отчетов будет осуществляться силами разработанной системы. Также будет сохраняться вся история, к которой может быть получен доступ в течение некоторого промежутка времени. |

Таблица 3 – Словарь иерархической структуры работ

|  |  |  |  |
| --- | --- | --- | --- |
| Этап | Состав работ | Трудоемкость (чел/час) | Результат |
| Разработка требований к хранению данных | Формирование требований к хранилищу разнотипных данных в системе, выбор необходимого типа хранилища и связанной СУБД | 36 ч | Требования к хранению и хранилищу данных |
| Разработка требований к серверной части | Обследование объекта и обоснование  необходимости;  Формирование требований пользователя к серверной части ИС | 64 ч | Формирование требований к серверной части |
| Разработка требований к клиентской части | Обследование объекта и обоснование  необходимости;  Формирование требований пользователя к клиентской части ИС. | 64 ч | Отчёт о выполненной работе |
| Разработка ТЗ | Разработка и утверждение технического задания на создание и разработку проекта | 64 ч | Разработанное техническое задание |
| Проектирование ПС | Проектирование архитектуры программной системы, ее состояний и схем обмена данными | 96 ч | Проект системы |
| Разработка макетов виджетов | Разработка и утверждение визуальных макетов системы | 64 ч | Разработанные макеты виджетов клиентской части |
| Разработка демо-версии проекта | Разработка проектных решений по системе и её частям. | 96 ч | Тестовая демо-версия |
| Разработка серверной части проекта | Разработка серверной части системы и её частей, например, связанных с обменом информацией с хранилищем данных. | 384 ч | Полностью функционирующее серверное приложение |
| Разработка клиентской части проекта | Разработка клиентской части системы и виджетов пользовательского интерфейса | 192 ч | Рабочая версия мобильного клиентского приложения |
| QA-тестирование | Разработка тест-кейсов и тест-сетов, проведение автоматического QA-тестирования | 96 ч | Протестированное серверное приложение |
| Ручное функциональное тестирование | Разработка тест-кейсов и тест-сетов, проведение функционального ручного тестирования | 72 ч | Протестированное клиент-серверное приложение |
| Внедрение разработанного проекта | Проведение предварительных испытаний, развертывание приложения | 72 ч | Функционирующий продукт в облаке компании |
| Подключение сотрудников | Подготовка персонала, предоставление данных учетных записей | 24 ч | Продукт с учетными данными аккаунтов |
| *Поддержка продукта* | *Поддержка продукта* | *192 ч* | *Поддерживаемый продукт* |

Таблица 4 – Смета проекта

|  |  |  |  |
| --- | --- | --- | --- |
| **Стоимость работ:** | | | |
| *Категория специалиста* | *Трудозатраты, час* | *Ставка,*  *руб/час* | *Итого* |
| Менеджер проекта | 288 | *350* | *100 800* |
| Backend - разработчик | 480 | 300 | *144 000* |
| Мобильный Android-разработчик | 192 | 250 | *48 000* |
| Аналитик | 256 | 240 | *61 440* |
| Тестировщик | 216 | 250 | *54 000* |
| **Стоимость оборудования:** | | | |
| *Категория расходов* | *Количество* | *Стоимость за единицу, руб* | *Итого (5 мес.)* |
| Сервер и облако для приложения | 1 | 20 000 (в мес) | 100 000 |
| **Инфраструктурные расходы:** | | | |
| *Категория расходов* | *Количество* | *Стоимость за единицу, руб* | *Итого (5 мес.)* |
| Интернет | 1 | 800 (в мес) | 3 000 |
| **ИТОГО** | 511 240 руб. | | |

Таблица 5 – Матрица ответственности RACI

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Процессы/Роли | Backend-разработчик | Мобильный разработчик | Аналитик | Менеджер проекта | Тестировщик |
| Разработка требований к хранению данных | Н. (I) | Н. (I) | Исп. (R | Утв. (A) | Н. (I) |
| Разработка требований к клиентской части | Н. (I) | Н. (I) | Исп. (R) | Утв. (A) | Н. (I) |
| Разработка клиентской части | Н. (I) | Н. (I) | Исп. (R) | Утв. (A) | Н. (I) |
| Разработка ТЗ | Н. (I) | Н. (I) | Исп. (R) | Утв. (A) | Н. (I) |
| Проектирование и моделирование ПС | Исп. (R) | Н. (I) | Исп. (R) | Утв. (A) |  |
| Разработка макетов виджетов | Н. (I) | Н. (I) | Исп. (R) | Утв. (A) | Н. (I) |
| Разработка демо-версии проекта | Исп. (R) | Исп. (R) | Cогл. (C) | Утв. (A) | Утв. (A) |
| Разработка серверной части приложения | Исп. (R) | Н. (I) | Cогл. (C) | Утв. (A) | Н. (I) |
| Разработка клиентской части приложения | Н. (I) | Исп. (R) | Cогл. (C) | Утв. (A) | Н. (I) |
| Проведение QA тестирования | Н. (I) | Н. (I) | Cогл. (C) | Утв. (A) | Исп. (R) |
| Проведение ручного функционального тестирования | Н. (I) | Н. (I) | Cогл. (C) | Утв. (A) | Исп. (R) |
| Внедрение разработанного продукта | Исп. (R) | Исп. (R) | Н. (I) | Утв. (A) | Н. (I) |
| Поддержка продукта | Исп. (R) | Исп. (R) | Исп. (R) | Утв. (A) | Исп. (R) |

Таблица 6 – Календарный план проекта

|  |  |  |  |
| --- | --- | --- | --- |
| Задача | Длит. (дней) | Начало | Окончание |
| Исследование предметной области | 9 | 20.02.2024 | 01.03.2024 |
| Разработка требований | 6 | 04.03.2024 | 15.02.2024 |
| Разработка требований к хранению данных | 1 | 04.03.2024 | 04.03.2024 |
| Разработка требований к серверной части | 3 | 05.03.2024 | 07.03.2024 |
| Разработка требований к клиентской части | 2 | 11.03.2024 | 12.03.2024 |
| Разработка технического  задания | 2 | 13.03.2024 | 04.03.2024 |
| *Утверждение технического*  *задания* | 1 | 15.03.2024 | 15.03.2024 |
| Проектирование и моделирование ПС | 5 | 18.03.2024 | 22.03.2024 |
| *Утверждение проекта ПС* | 1 | 25.03.2024 | 25.03.2024 |
| Разработка макетов виджетов | 2 | 26.03.2024 | 27.03.2024 |
| Утверждение макетов виджетов | 1 | 28.03.2024 | 28.03.2024 |
| Разработка демо-версии проекта | 6 | 29.03.2024 | 05.04.2024 |
| Разработка программной документации | 5 | 08.04.2024 | 12.04.2024 |
| Разработка серверной части приложения | 16 | 15.04.2024 | 10.05.2024 |
| Разработка клиентской части приложения | 7 | 13.05.2024 | 21.05.2024 |
| QA-тестирование | 2 | 22.05.2024 | 23.05.2024 |
| Ручное функциональное тестирование | 1 | 24.05.2024 | 24.05.2024 |
| Внедрение разработанного продукта | 5 | 27.05.2024 | 31.05.2024 |
| Проведение предварительных испытаний | 3 | 27.05.2024 | 29.05.2024 |
| Развертывание приложения, подключение сотрудников | 2 | 30.05.2024 | 31.05.2024 |
| Итог | 69 | 20.02.2024 | 31.05.2024 |

ПРИЛОЖЕНИЕ Ж. ЖИЗНЕННЫЙ ЦИКЛ ИС

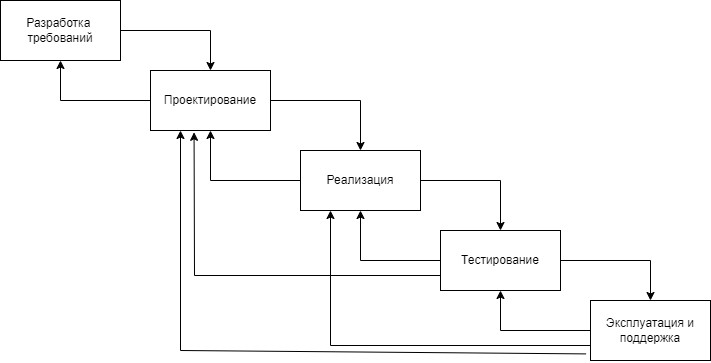


Рисунок 1 – Жизненный цикл проекта

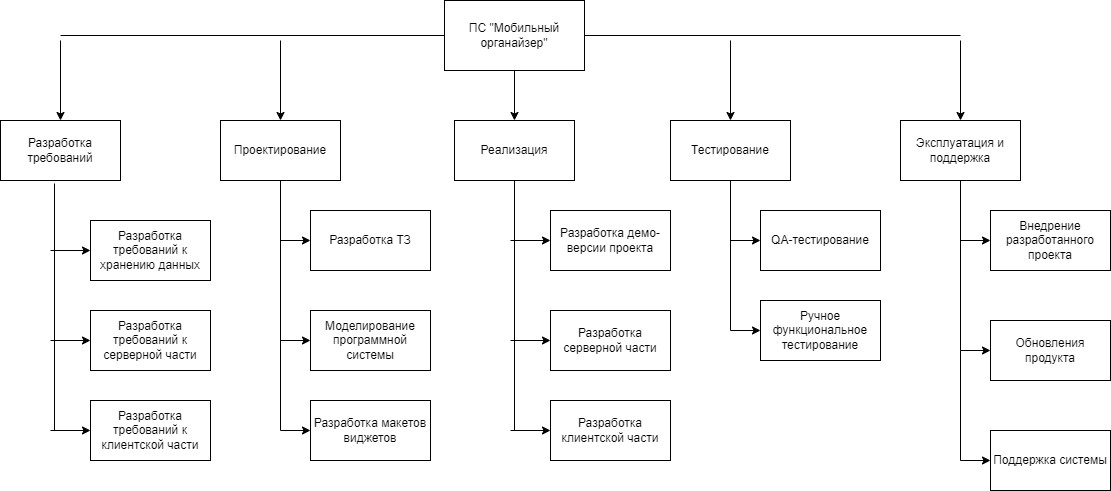


Рисунок 2 - Фазы жизненного цикла проекта