Федеральное государственное бюджетное образовательное учреждение высшего образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

Кафедра информационных систем и программной инженерии

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к выпускной квалификационной работе по дисциплине

на тему

Программная система «Мобильный органайзер»

Выполнил:

студент гр. ПРИ-120

Парахин К.В.

Принял:

Владимир, 2024

**Аннотация**

В данной выпускной квалификационной работе производилась разработка программной системы «Мобильный органайзер», целью которого является автоматизация и улучшения 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 2 типа: обычный пользователь или системный администратор.

Курсовой проект представлен на 87 страницах, основных рисунков - 29, использованных источников – 5, приложений – 3, иллюстрационный материал на 2 листах формата А1.

**ABSTRACT**

In this course project, a prototype of the mobile application "Multitasking Calendar" was developed

The project consists of 4 main stages, including the main three stages, such as: "Domain analysis", "System Design", "Client-server application development", "System testing", "Mobile application deployment".

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

creating and storing events, groups, tasks and user reports, viewing and editing a calendar with events, receiving notifications and reminders about the beginning of events.

The implemented system does not use separation by user roles.

The course project is presented on 87 pages, the main figures - 29, the sources used – 5, appendices – 4, illustrative material on 2 sheets of A1 format.

**Содержание**

[ВВЕДЕНИЕ 4](#_Toc164761697)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 6](#_Toc164761698)

[1.1. Описание предметной области 6](#_Toc164761699)

[1.2. Задачи приложения и обоснование его актуальности 7](#_Toc164761700)

[1.3. Сравнительный обзор аналогов 10](#_Toc164761701)

[1.4. Описание набора функций системы 18](#_Toc164761702)

[1.5. Описание категорий пользователей системы и прецедентов 19](#_Toc164761703)

[1.6. Словарь предметной области 20](#_Toc164761704)

[1.7. Цели и задачи работы 22](#_Toc164761705)

[2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ 23](#_Toc164761706)

[2.1. Функциональная декомпозиция системы 23](#_Toc164761707)

[2.2. Описание состава данных 29](#_Toc164761708)

[2.3. Построение схемы базы данных 31](#_Toc164761709)

[2.4. Требования к разрабатываемой системе 33](#_Toc164761710)

[3 РАЗРАБОТКА ПРОГРАММНОЙ СИСТЕМЫ 36](#_Toc164761711)

[3.1. Выбранный вариант разработки программной системы 36](#_Toc164761712)

[3.2. Описание взаимодействия приложений 38](#_Toc164761713)

[3.3. Реализация серверного приложения 40](#_Toc164761714)

[3.4. Описание структуры серверного приложения 41](#_Toc164761715)

[3.5. Спецификация API 43](#_Toc164761716)

[3.6. Описание особенности реализации клиент-серверного приложения 44](#_Toc164761717)

[3.7. Описание разработки клиентского приложения 46](#_Toc164761718)

[3.8. Описание интерфейса мобильного приложения 48](#_Toc164761719)

[3.9. Реализация push-уведомлений 60](#_Toc164761720)

[3.10. Тестирование системы 61](#_Toc164761721)

[3.11. Развертывание приложения 68](#_Toc164761722)

[3.12. Ведение репозиториев программной системы 70](#_Toc164761723)

[4 ИНФОРМАЦИОННЫЙ МЕНЕДЖМЕНТ 72](#_Toc164761724)

[4.1. Календарный план проект и диаграмма Ганта 72](#_Toc164761725)

[4.2. Экономическое обоснование проекта 74](#_Toc164761726)

[4.3. Анализ инвестиций, трудоемкости и ресурсоемкости 76](#_Toc164761727)

[4.4. Проведение информационного менеджмента 78](#_Toc164761728)

[4.5. Анализ рисков проекта 82](#_Toc164761729)

[4.6. Подведение выводов о результате ввода системы в использование 84](#_Toc164761730)

[Заключение 86](#_Toc164761731)

[Список использованных источников 87](#_Toc164761732)

[ПРИЛОЖЕНИЕ А. ДИАГРАММА ПРЕЦЕДЕНТОВ СИСТЕМЫ 89](#_Toc164761733)

[ПРИЛОЖЕНИЕ Б. ЛОГИЧЕСКАЯ СХЕМА БАЗЫ ДАННЫХ 90](#_Toc164761734)

[ПРИЛОЖЕНИЕ В. ДИАГРАММА РАЗВЕРТЫВАНИЯ ПРИЛОЖЕНИЯ 91](#_Toc164761735)

# 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. Задачи приложения и обоснование его актуальности

Основные задачи, которые должна решать программная система «Мобильный органайзер»:

* + Автоматизация деятельности и взаимодействия сотрудников IT-отдела
  + Создание иерархической организационной структуры проектов внутри IT-отдела
  + Введение автоматической системы отчетности и регистрации инцидентов
  + Хранение данных о сотрудниках, их мероприятиях, задачах, сообщениях
  + Обеспечение своевременного обмена информации в системе путем рассылки уведомлений
  + Оптимизация работы отделов мониторинга и аналитики

«Мобильный органайзер» - это программное обеспечение, которое содержит 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.

В зависимости от экономического положения организации, новизны и масштаба ее проектов – принимается решение о использовании готовых решений, предоставляющих некоторые обособленные возможности (например, трекинг задач, ведение календаря мероприятий, ведение чатов и истории сообщений, отслеживание и мониторинг системы, документация системы и т.д.).

Для организаций среднего масштаба зачастую встает выбор о принятии решения по разработке собственного программного продукта, который способен объединить все необходимые функции, убрать лишние и оптимизировать их количество согласно требованиям бизнеса. Благодаря этому, компания перестает тратить средства на оплату подписки за пользование платным сторонним программным обеспечением, а также может сэкономить средства на оплату автоматизированных приложением рабочих мест.

Рассмотрим разработку данного проекта на примере организации ООО «АИСТСОФТ».

Предприятие ООО «АИСТСОФТ» - компания, занимающаяся разработкой высоконагруженного программного обеспечения для сбора, обработки и анализа данных в реальном времени, и их последующего преобразования в информацию.

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 по рабочей деятельности отдела предприятия;
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 1.1) приведена матрица структурированных бизнес-выгод, которые имеют место в случае успешной реализации проекта.

Таблица 1.1 - 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 |

## 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:

1. Active Banch
2. Jira Service Management
3. Atera
4. Puppet

Краткое описание каждого из этих сервисов:

1. ActiveBatch

URL-ссылка: <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].

1. Jira Service Management

URL-ссылка: <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].

1. Atera

URL-ссылка: <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.

1. Puppet

URL-ссылка: <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:

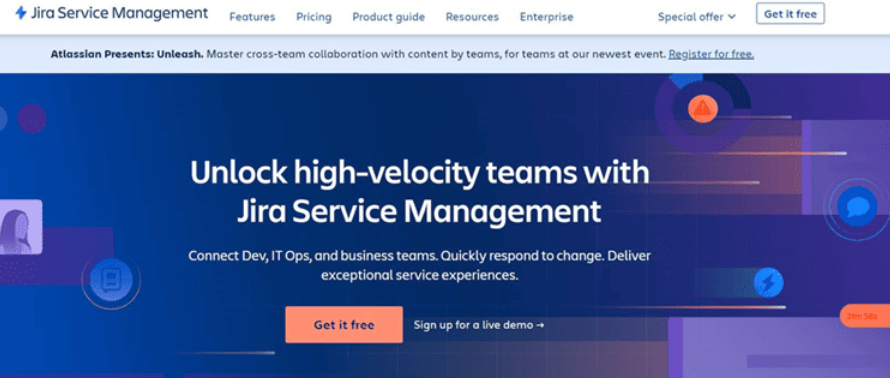


Рисунок 1.1. Программный компонент Jira Service Management

Описание критериев для сравнения

Затем были выделены критерии, по которым можно провести сравнительный анализ выделенных компонентов – они приведены ниже в таблице 1.2:

Таблица 1.2. Сравнительные критерии оценки программных компонентов

|  |  |
| --- | --- |
| Критерии | Краткое описание |
| Дизайн | Оформление контента, совокупность всех графических элементов на веб-странице сервиса. |
| Удобство использования | Понятная и доступная структура сервиса, хорошая видимость и оформление основных элементов сервиса. |
| Функциональность приложения категории IT Automation | Практичность использования программного компонента, наличие на нем всех нужных опций и возможностей:  - наличие трекинга задач и целей проекта  - ведение мероприятий и расписания работы  - возможность создания отчетов и диаграмм  - отслеживание стабильности работы системы. |
| Динамичность работы компонентов | Скорость работы сервиса, то есть постоянное обновление информации на нем, адаптивность и скорость работы. |
| Интеграция с другими компонентами | Возможность интеграции сервиса планирования мероприятий в другие программные продукты. |
| Модель распространения | Модель распространения программного компонента, то есть наличие бесплатной/платной версии, пробной версии, размер и тип тарифного плана. |

В приведенной ниже таблице 1.3 с сравнением четырех программных компонентов IT Automation: ActiveBatch, Jira, Atera, Puppet по выделенным 6 критериям, таким как: дизайн, удобство использования, функциональность приложения категории IT Automation, динамичность работы компонентов, интеграция с другими программными компонентами, модель распространения.

Для оценки использовалась пятибальная шкала (где 1 – это очень плохая оценка, когда продукт не соответствует описанию, 2 – плохое качество и реализация, 3 – удовлетворительная реализация и недостаточный функционал, 4 – неплохой показатель при некоторых недостатках, 5 – идеальное значение показателя).

Таблица 1.3. Расчет средней оценки программных компонентов по выделенным критериям

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерии/Cервисы | Active Batch | Jira | Atera | Puppet |
| Дизайн | 4 | 5 | 5 | 5 |
| Удобство использования | 5 | 5 | 5 | 4 |
| Функциональность приложения категории IT Automation | 4 | 4 | 5 | 4 |
| Динамичность работы компонентов | 5 | 5 | 5 | 5 |
| Интеграция с другими компонентами | 3 | 4 | 5 | 5 |
| Модель распространения | 5 | 3 | 4 | 4 |
| Средняя оценка | 4,3 | 4,3 | 4,8 | 4,5 |

Итоги и выводы по проведенному сравнительному анализу:

Наивысшую среднюю оценку по итогу сравнения набрал сервис Atera (средняя оценка 4,8), затем с небольшим отставанием идет сервис Puppet (4,5) и чуть хуже оценки получили компоненты Active Batch и Jira Service Management (4,3).

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).

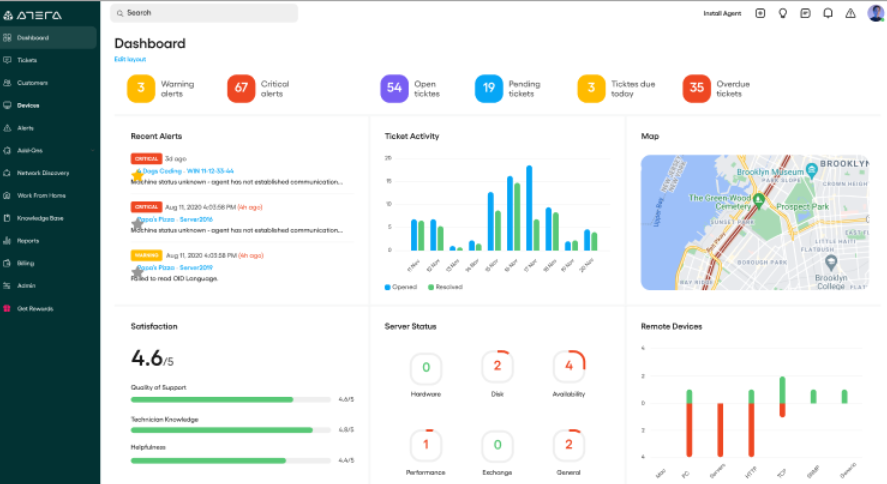


Рисунок 2. Программный компонент Atera

## Описание набора функций системы

Основные функции разрабатываемой системы, определяющие границы предметной области:

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

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

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

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

- добавлять общие задачи и цели для развития IT-проекта

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

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

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

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

- получать отчеты от приложения за определенный период времени)

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

- рассылать уведомления на зарегистрированные мобильные устройства пользователей системы

- 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.

## 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;
  + Провести разработку серверного приложения и базы данных;
  + Провести разработку мобильного приложения;
  + Провести развертывание и тестирование приложения;
  + Провести информационный менеджмент проекта по разработке программной системы;
  + Выполнить экономическое обоснование внедрения системы.

Предприятие, взятое для примера (связанное с местом работы) – ООО «АИСТСОФТ» - компания, занимающаяся разработкой высоконагруженного программного обеспечения для сбора, обработки и анализа данных в реальном времени, и их последующего преобразования в информацию. Предметная область данной программной систем связана с оптимизацией деятельности IT-отделов предприятия и взаимодействием сотрудников внутри ее.

1. ПРОЕКТИРОВАНИЕ СИСТЕМЫ
   1. Функциональная декомпозиция системы

Основные прецеденты (доступные всем пользователям системы):

- Управлять своим аккаунтом

- Зарегистрироваться

- Авторизоваться

- Выйти из системы

*-* Подать заявку на получение роли системного администратора

- Просматривать календарь мероприятий пользователя

- Просматривать свой календарь

- Просматривать календарь пользователей в группе

- Управлять трекингом задач

- Распределять задачи в группе по исполнителям

- Контролировать смену статусов задач

- Отослать уведомление о назначении задачи

- Управлять мероприятиями

- Выбрать тип мероприятия

- Выбрать время и продолжительность

- Добавить пользователей из группы на мероприятие

- Отослать уведомления о приглашении на мероприятие

- Создать отчет о личной деятельности

- Выбрать критерий отчета (отчет по задачам, мероприятиям, проблемам) - Выбрать период для получения отчетности

- Выбрать тип представления отчета (текстовый или графический)

- Регистрировать проблемы рабочего процесса

- Фиксировать и отправлять инциденты о неисправностях работы системы

- Сигнализировать о прочих нарушениях рабочего процесса

- Получать уведомления

- Получать уведомления о назначении задачи на исполнение

- Получать уведомления о приглашении на мероприятие

- Получать уведомления о начале мероприятий

Прецеденты менеджера группы:

- Заниматься менеджментом группы

- Выбрать тип группы

- Добавить участников в группу

- Редактировать параметры группы

- Удалить участника группы

- Создать отчет по деятельности группы

Прецеденты системного администратора системы:

- Осуществлять поддержку

- Получать уведомления о неисправностях в работе системы

- Получать alert-сообщения в авто-режиме

- Регистрировать произошедшие инциденты

- Менять статусы инцидентов при их закрытии

Диаграмма прецедентов в нотации UML изображена ниже на рисунке 2.1:



Рисунок 2.1. Расширенная диаграмма прецедентов

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.

Прецедент «Авторизоваться»

Предусловие: Пользователь решил воспользоваться функционалом системы при уже имеющейся учетной записи, то есть пользователь уже зарегистрирован в системе.

Действующее лицо: пользователь

Основной поток: пользователь открывает мобильное приложение, на главной странице видит раздел для авторизации, затем вводит свои данные и получает доступ к своей личной учетной записи и к личному кабинету.

Альтернативный поток: Пользователь ещё не зарегистрирован в системе. Тогда при попытке авторизоваться с некоторыми данными, он будет уведомлен об отсутствии требуемой учетной записи.

Постусловие: При отсутствии зарегистрированной учетной записи, пользователь будет перенаправлен на страницу регистрации.

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):

- User (пользователь с некоторыми учетными данными).

- UserDeviceMap (токены мобильных устройств существующих клиентов системы).

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

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

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

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

- Alert (проблема в работе системы, автоматически зарегистрированная самой системой).

- Snapshot (отчет (снимок данных на определенный момент времени) по определенному критерию).

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

Диаграмма классов (уровня проектирования) системы в нотации UML представлена ниже на рисунке 2.2:

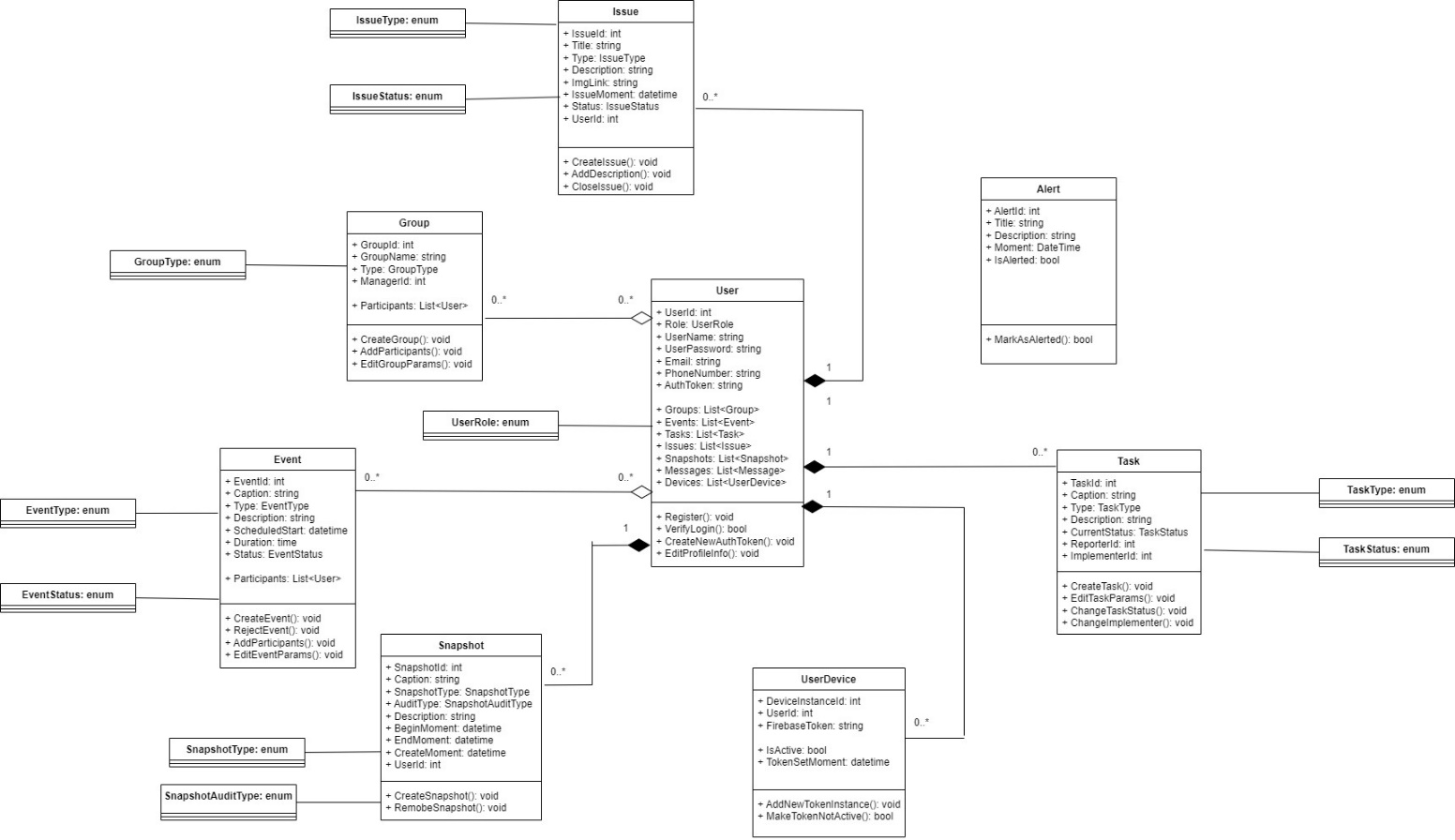


Рисунок 2.2. Диаграмма классов уровня проектирования

* 1. Построение схемы базы данных

Далее требуется построить структурную схему базы данных Postgre SQL – сделать это можно с помощью инструмента построения ER-диаграмм в программе Dbeaver – информация бралась из источника [12].

Физическая схема базы данных представлена ниже на рисунке 2.3:

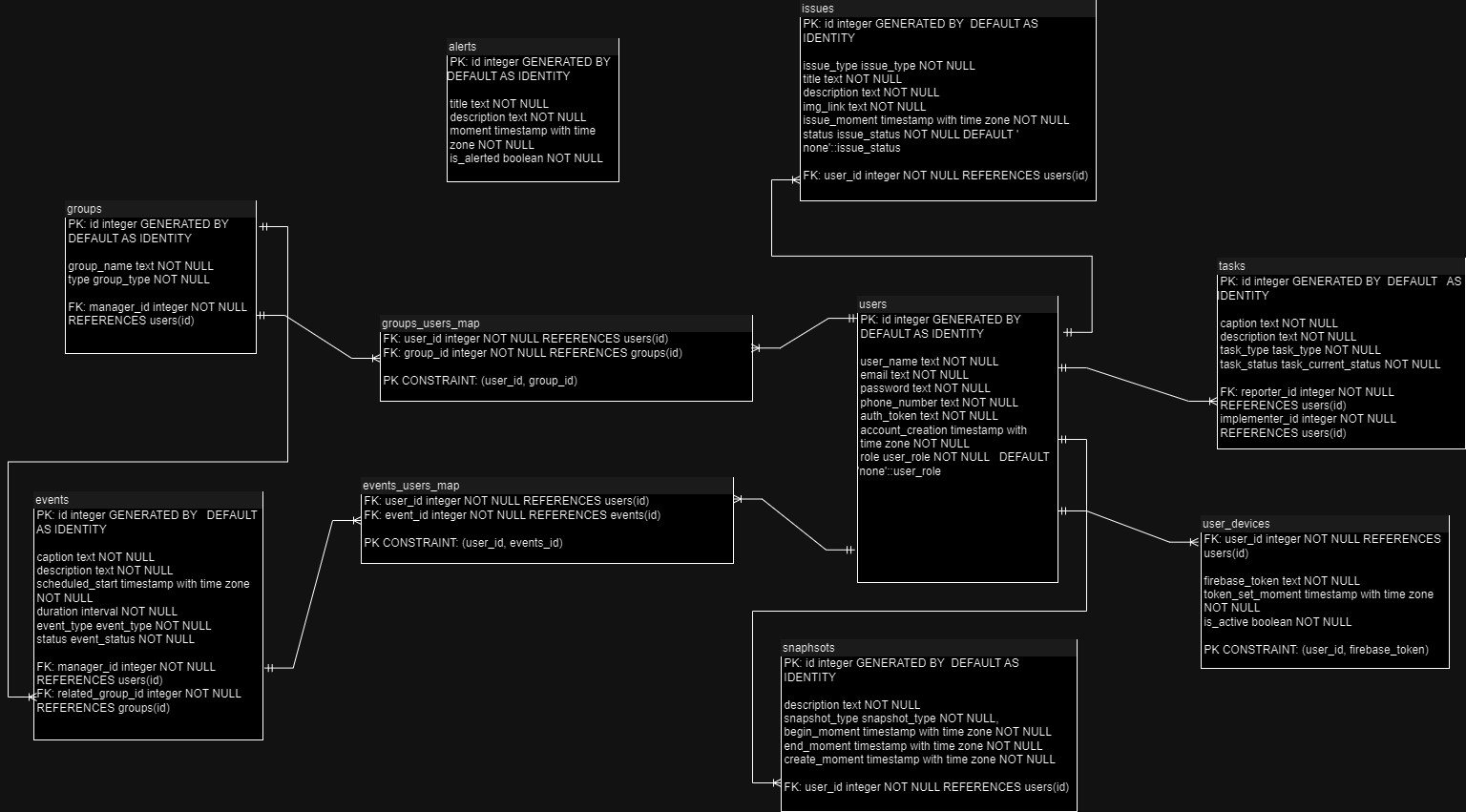


Рисунок 2.3. Физическая схема базы данных

## Требования к разрабатываемой системе

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

1. Функциональные требования

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

1.1) Функция 1

1.1.1) Название - возможность использовать авторизацию пользователей;

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

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

1.1.4) Зависимости – фреймворк ASP.NET Core 5, Google JWT Bearer Authentification.

1.2) Функция 2

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

1.2.2) Описание – возможность регистрации пользователя в системе и возможность подтверждения регистрации новой учетной записи пользователя по электронной почте;

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

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

1.3) Функция 3

1.3.1) Название – возможность распределения пользователей по группам;

1.3.2) Описание – возможность создания пользователем групп различных типов (рабочих, учебных, личных), добавления в него пользователей и разграничения прав и прав на просмотр чужих мероприятий;

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

1.4) Функция 4.

1.4.1) Название – возможность планирования пользовательских мероприятий;

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

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

1.5) Функция 5.

1.5.1) Название – возможность создания пользовательских задач;

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

1.5.3) Назначение – сохранение пользовательских задач в базе данных.

1.6) Функция 6

1.6.1) Название – возможность регистрирования проблем в работе системы;

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

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

1.7) Функция 7

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

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

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

1.8) Функция 8

1.8.1) Название – получать уведомления о событиях в системе;

1.8.2) Описание – возможность организации системы уведомлений и напоминаний (о приглашении на групповое мероприятие, назначении задач) для каждого пользователя-участника мероприятия;

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

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

1. Нефункциональные требования
   1. Обеспечение уровня производительности – трафика запросов не менее 10 000 запросов к серверу в секунду (10 000 RPS);
   2. Отказоустойчивость и надежность работы системы, регистрация неисправностей на сервере;
   3. Стабильность и постоянная работа системы (перезапуск не чаще 5 минут в день);
   4. Масштабируемость компонентов системы;
   5. 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:

1. Для разработки серверного приложения планируется использовать IDE Microsoft Visual Studio – как наиболее удобное, популярное, а также свободно распространяемое программной средство разработки приложений на .NET.
2. 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].
3. Для тестирования REST API HTTP-запросов планируется использовать приложение Postman – наиболее популярное и удобное приложение, используемое для задач тестирования веб-приложений.
   1. 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 и контейнером PostgreSQL базы данных.

Диаграмма развертывания программной системы изображена ниже на рисунке 3.1:

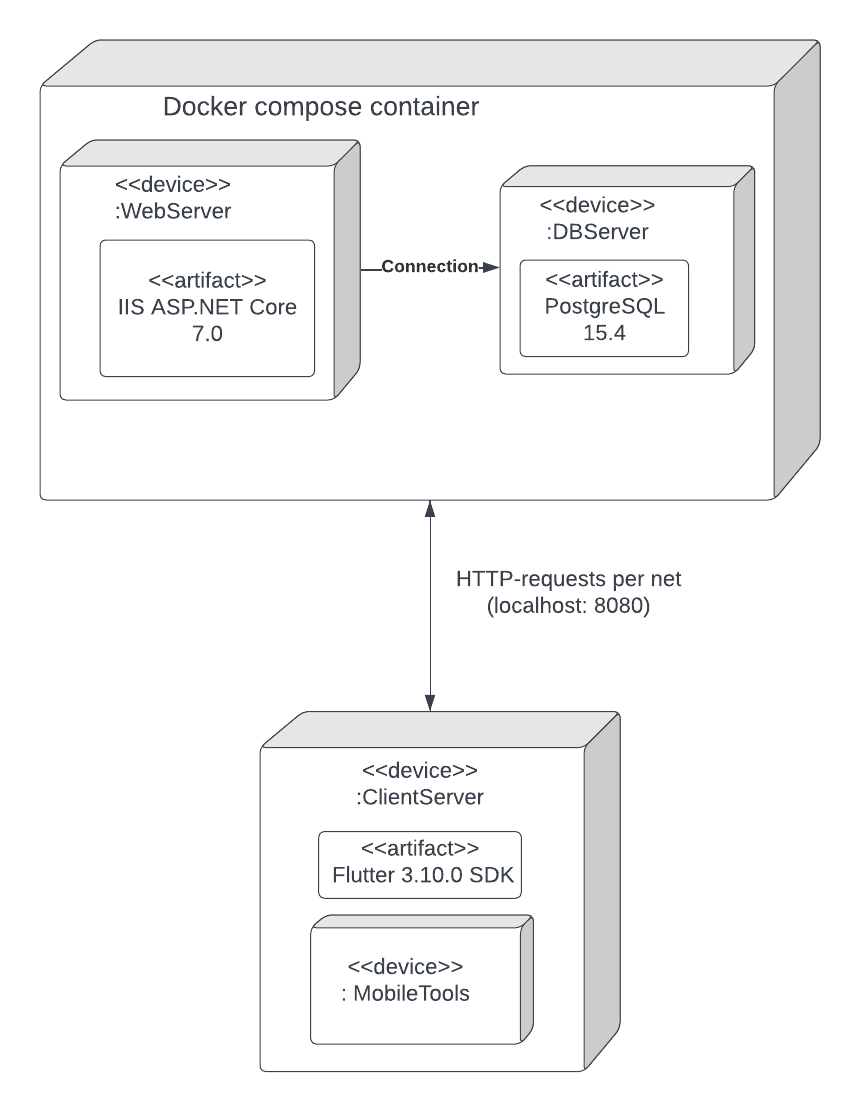


Рисунок 3.1. Диаграмма развертывания ПС

Алгоритм работы приложения

Ниже построим схему одного из основных алгоритмов приложения – схема процесса авторизации пользователя в системе – рисунок 3.2:

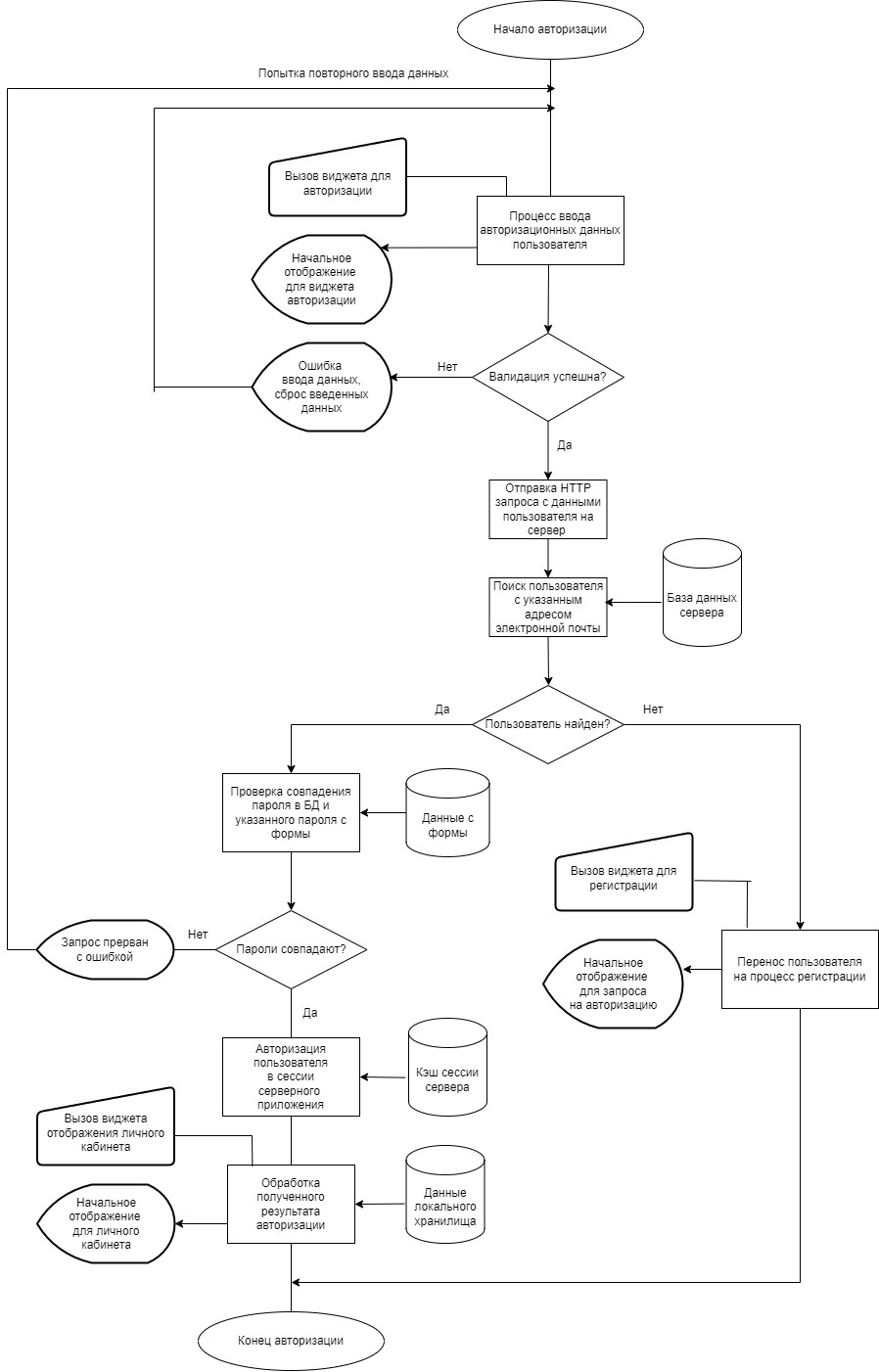


Рисунок 3.2. Схема авторизации пользователя

* 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.

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:

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

- 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.)

- 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)

- 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).

- 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).

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

Ниже на рисунке 3.4 показан скриншот обозревателя решений с полной структурой всех проектов

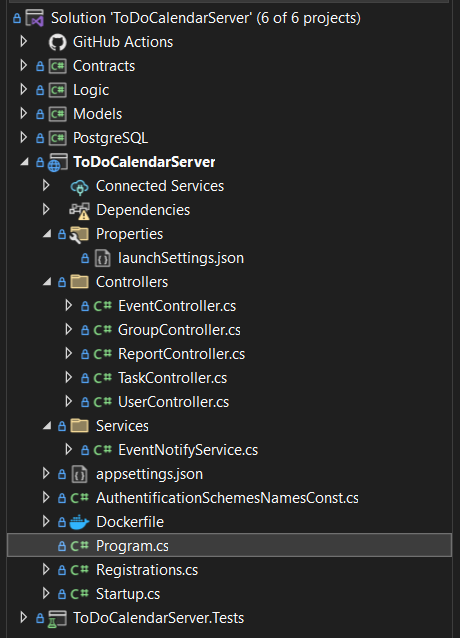


Рисунок 3.4. Структура проектов серверного приложения

* 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 наполнения тела запроса возвращается ответ со статусом 400 и описанием ошибки в объекте errros.

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 особенности реализации клиент-серверного приложения

«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»[7].

«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)»[7].

«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). Затем при обращении к очередному чату – пользователь по мере его открытия сразу получает некоторое закэшированное состояние чата – а при загрузке нового запроса данных с сервера получает уже актуальный набор сообщений на текущий момент времени»[7].

«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 (так как, например, запрос на группу пользователей может включать в себя подгрузку множества связанных с нею данных, а именно – пользователей данной группы, информации о каждом из этих пользователей, личных сообщениях, мероприятиях и задачах любого пользователя). 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»[7].

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

«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 (так как с ним будет связан новый экземпляр Firebase-приложения) и этот токен будет независимо от первого токена сохранен в базу данных. При этом стоит отметить, что в качестве мер безопасности - токены сохраняются в локальной памяти устройства – но в течении времени неактивности могут быть автоматически очищены»[7].

## 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:

- Виджеты без состояния (то есть виджеты, которые не меняют своих данных (и их своевременное отображение) в разное время взаимодействия с ними). 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).

- Виджеты с изменением состояния (те виджеты, на которых необходимо производить подгрузку/изменение данных и т.д.). 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 3.5 показана структура папок и виджетов данного клиентского мобильного приложения:

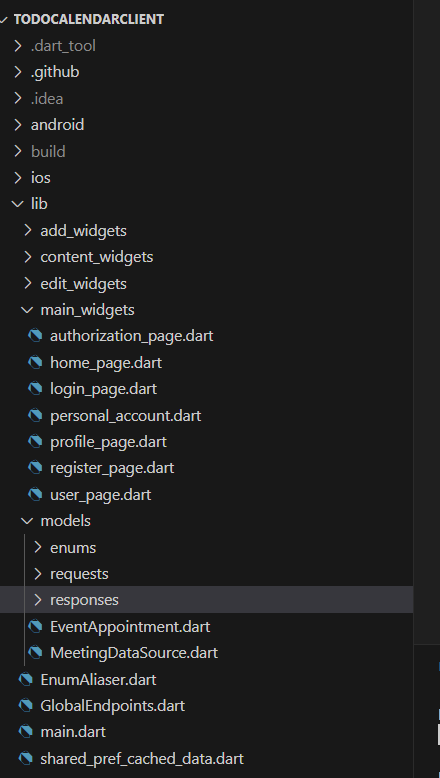
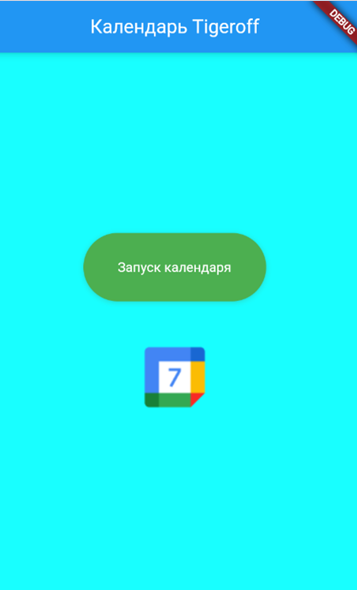


Рисунок 3.5. Структура папок клиентского Flutter – приложения

* 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 (рисунки 3.6 – 3.7):



Рисунки 3.6 – 3.7. Стартовая страница приложения и страница неавторизованного пользователя в системе

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) {

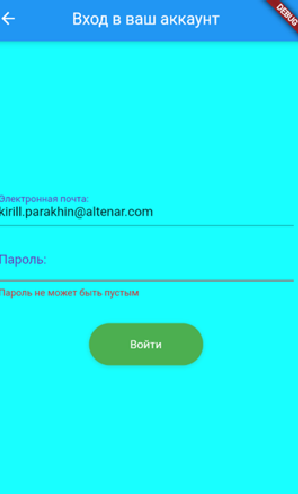
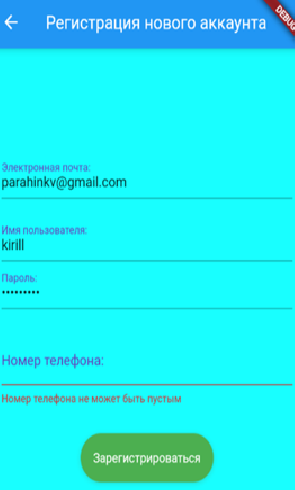
print('Error retrieving data from SharedPreferences: $e');

return null;

}

}

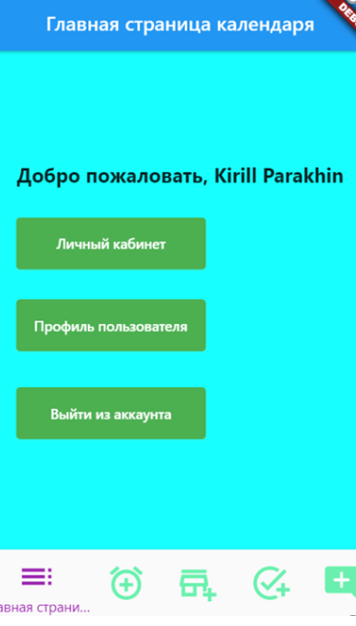
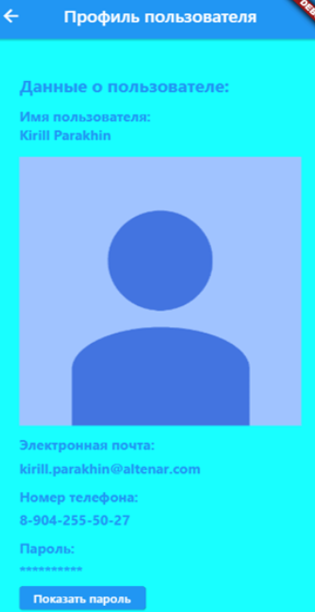
Страницы авторизации и регистрации представлены на рисунках 3.8 – 3.9 ниже. На данных страницах, как и на всех остальных страницах, на которых происходит ввода каких-то данных – происходит валидация вводимого текста:



Рисунки 3.8 – 3.9. Страница авторизации и регистрации

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

На главной странице пользователя (рисунок 3.10) и странице личного кабинета пользователя (рисунок 3.11) присутствует нижнее навигационное меню с различными режимами текущего окна. По умолчанию, открывается главная страница. При нажатии на другую панель – откроется другое состояние виджета.

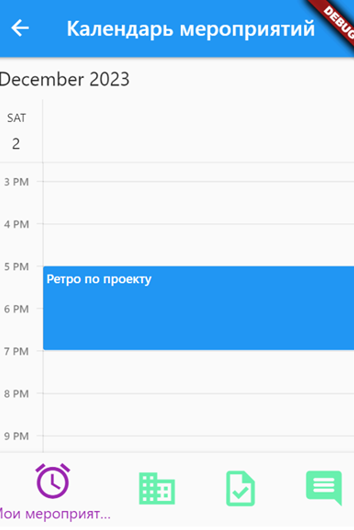
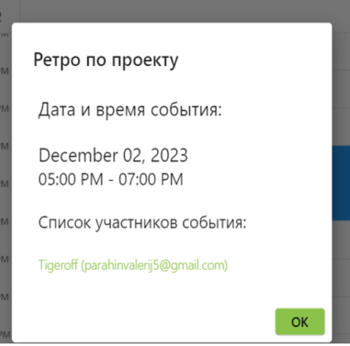
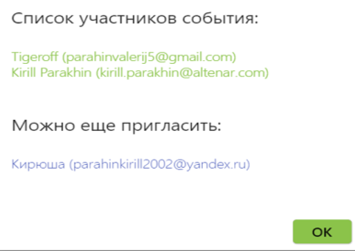


Рисунки 3.10 – 3.11. Главная страница и страница личного кабинета пользователя

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.

Страница с календарными мероприятиями пользователя и выпадающими диалоговыми виджетами представлены ниже на рисунках 3.12 – 3.14:



Рисунки 3.12 – 3.14. Страницы с мероприятиями

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(() {

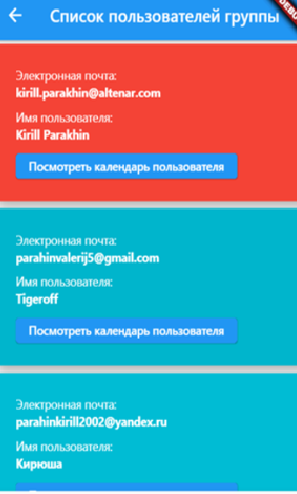
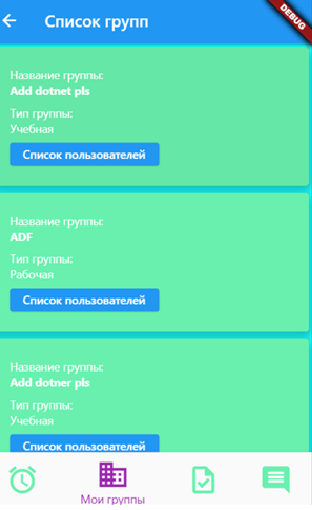
eventsList = fetchedEvents;

});

}

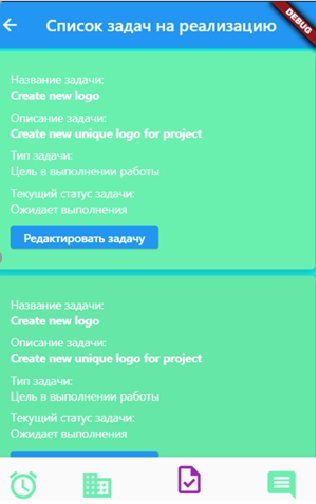
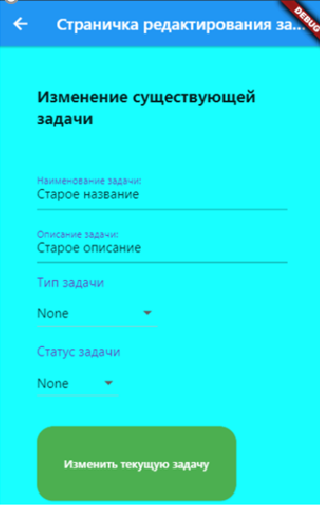
Страницы со списком групп пользователя представлена ниже на рисунке 3.15, страница со списком пользователей группы – на рисунке 3.16.

У каждой группы можно посмотреть список ее пользователей (красным выделяется текущий пользователь) – и посмотреть снимок его календаря:



Рисунки 3.15 – 3.16. Страница со списком групп и страница со списком пользователей группы

Ниже на рисунке 3.17 представлены виджет со списком задач текущего пользователя, а на рисунке 3.18 - виджет для создания/редактирования задачи пользователя. Для задачи можно выбирать ее тип, текущий статус, а также назначать ее исполнение для определенного пользователя – о чем должен быть также уведомлен.



Рисунки 3.17 – 3.18. Страница со списком задач и страница конкретной задачи

Также 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).

В отчете по задачам вычисляются показатели: количество задач для реализации за указанный период, процент не начатых задач, процент задач в процессе выполнения, процент выполненных задач

В отчете по мероприятиям вычисляются показатели: количество запланированных мероприятий за указанный период, общая реальная продолжительность мероприятий за данный период, процент посещения запланированных мероприятий

В отчете по проблемам (issuses) вычисляются показатели: количество зарегистрированных и просмотренных тикетов, процент открытых и закрытых тикетов. Ниже на рисунке 3.19 представлен виджет со списком отчетов пользователей – представленных по умолчанию в текстовом виде:



Рисунок 3.19. Виджет со списком отчетов пользователя

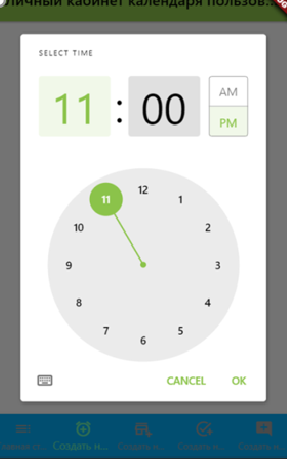
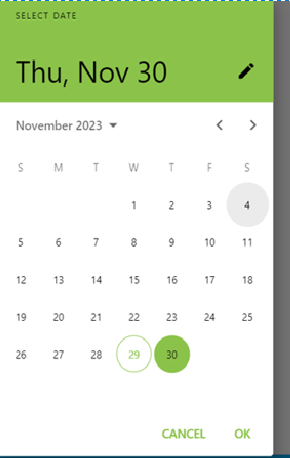
***Виджеты с графиками необходимо реализовать и представить ниже***

Виджеты для создания новых данных

Все виджеты, использующие для создания/редактирования данных имеют встроенную валидацию исходной строки на стороне клиента (например, проверка пустоты строк, сравнение времени окончания и начала планируемого мероприятия (или отчета) и т.д.).

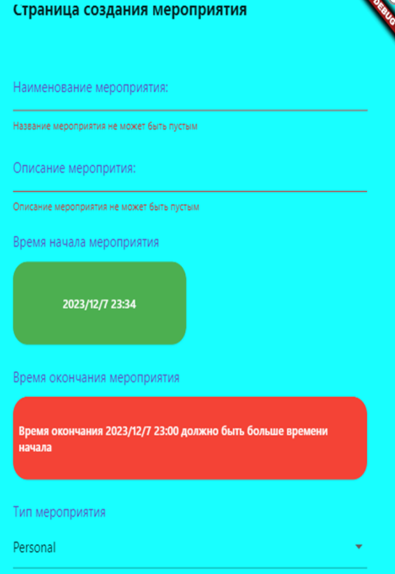
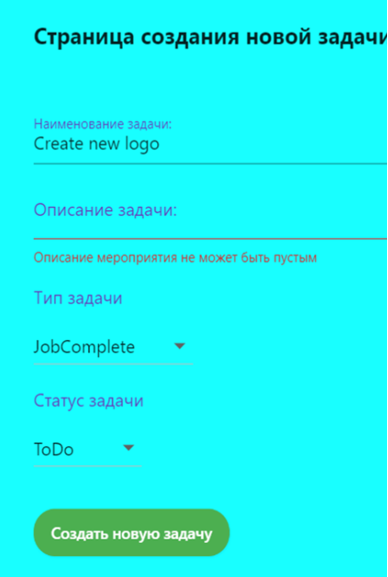
Для значений, соответствующим типам и статусам создаваемых сущностей – будут использоваться выпадающие списки для перечислений. Для выбора даты и времени начала и окончания создаваемого мероприятия использовался специальное визуальное диалоговое окно Flutter.

Ниже на рисунках 3.20 – 3.21 приведены виджеты для создания нового мероприятия (рисунок 3.20 – выбор даты мероприятия, рисунок 3.21 – выбор времени начала мероприятия):



Рисунки 3.20 – 3.21. Виджет для выбора даты и виджет для выбора времени мероприятия

Для создания новых задач, групп пользователей, отчетов, регистрации проблем в работе системы (и так далее) также используются специальные страницы, скриншоты которых приведены ниже (рисунок 3.22 – страница создания нового мероприятия, рисунок 3.23 – страница создания новой задачи):



Рисунки 3.22 – 3.23. Страница создания нового мероприятия и страница создания новой задачи

***Также можно добавить раздел с кабинетом системного администратора и просмотром зарегистрированных пользователями инцидентов***

Для рисования диаграмм (гистограмм), отображающих распределение KPI пользователей групп использовался специальный виджет Flutter SfCartesian Chart – информация по реализации бралась из источника [17].

## 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.

Из моей статьи «Обзор приложения, связанного с автоматизацией деятельности IT-отделов предприятия»:

«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)»[7].

«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 (информация о email адресе храниться у каждого пользователя и требуется при регистрации) и Firebase-клиента для создания push-уведомлений (рассылка происходит на авторизованные мобильные устройства благодаря сохранению токенов)»[7].

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 на рисунках 3.24 – 3.125 – показаны примеры запросов, связанных с созданием и получением информации о пользователях, а на рисунках 3.26 - 3.27 – примеры запросов с получением информации о мероприятиях (events) пользователей:

Запросы к endpointу /users:

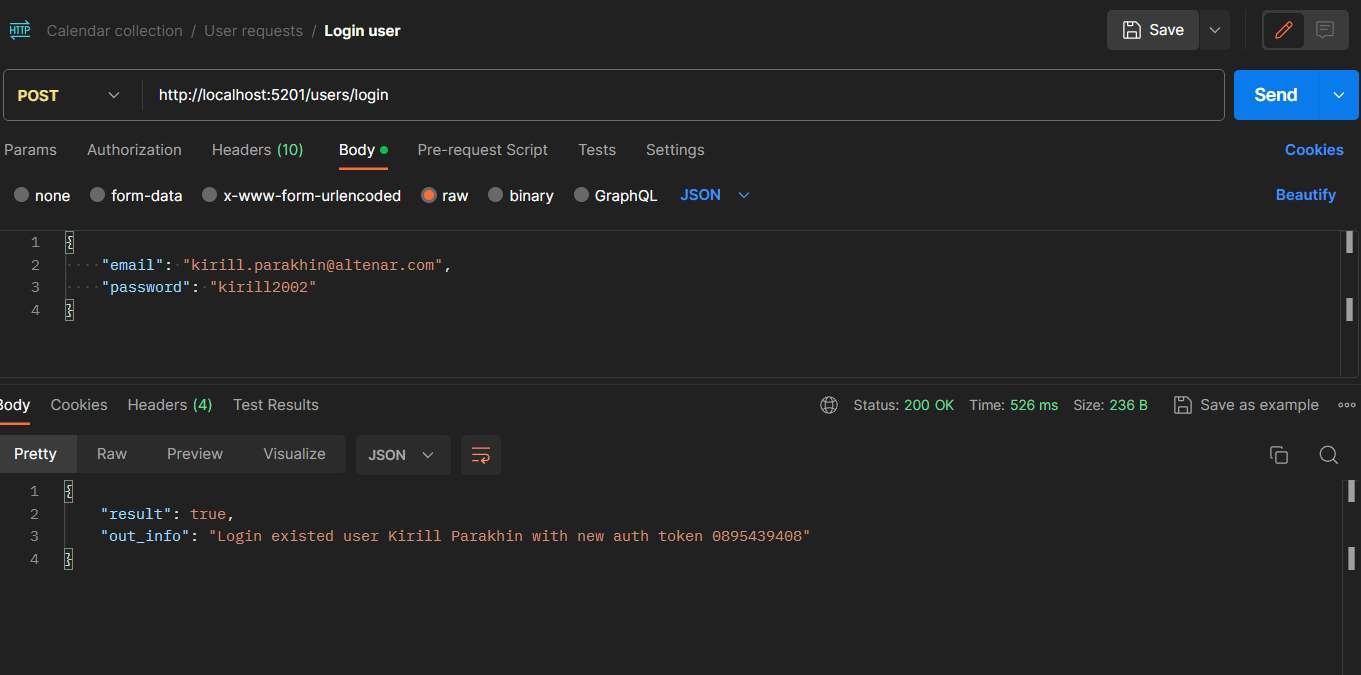


Рисунок 3.24. Тестирование запроса на авторизацию пользователя

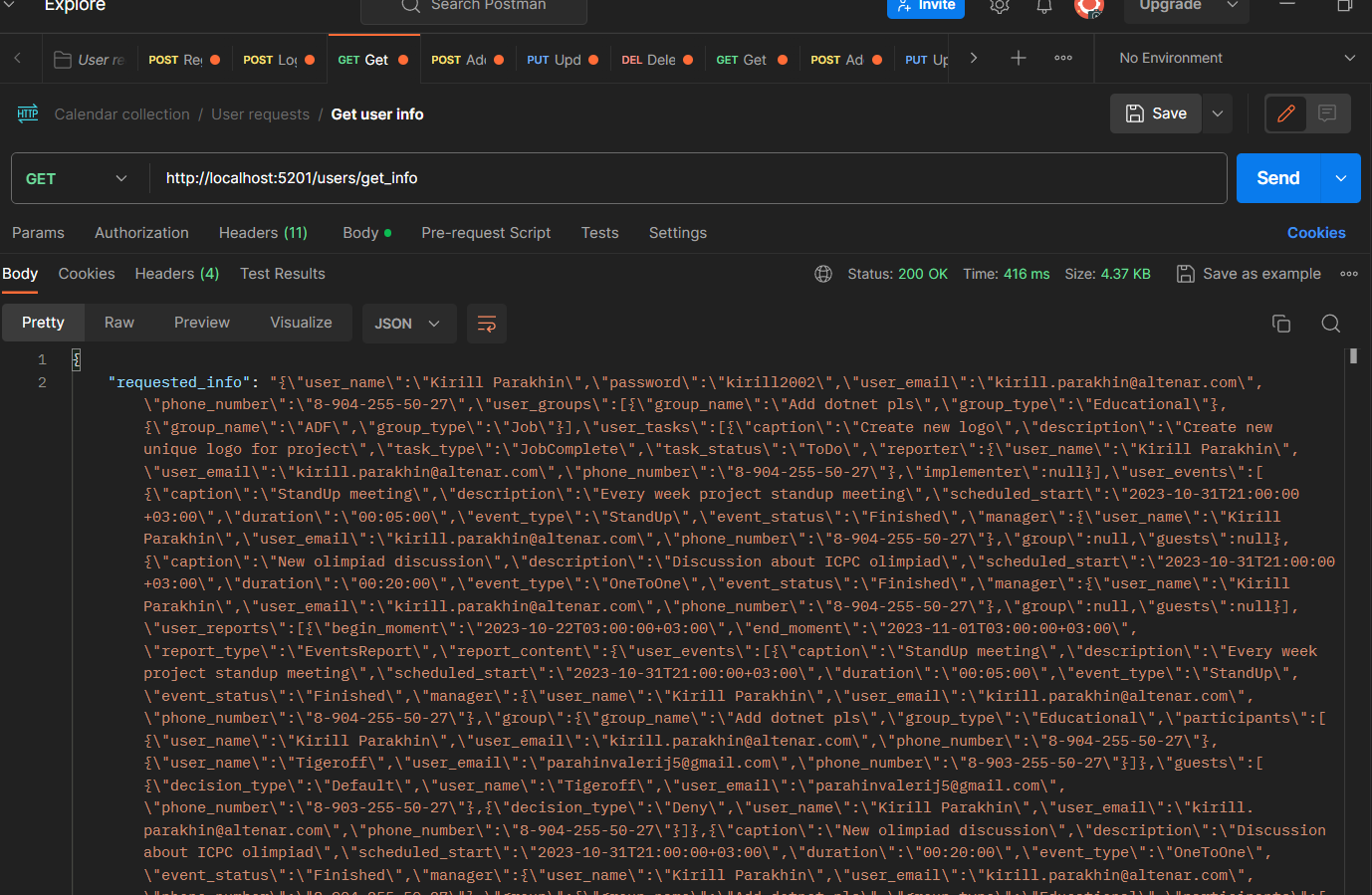


Рисунок 3.25. Тестирование получения информации о пользователе

Запросы к endpointу /events:

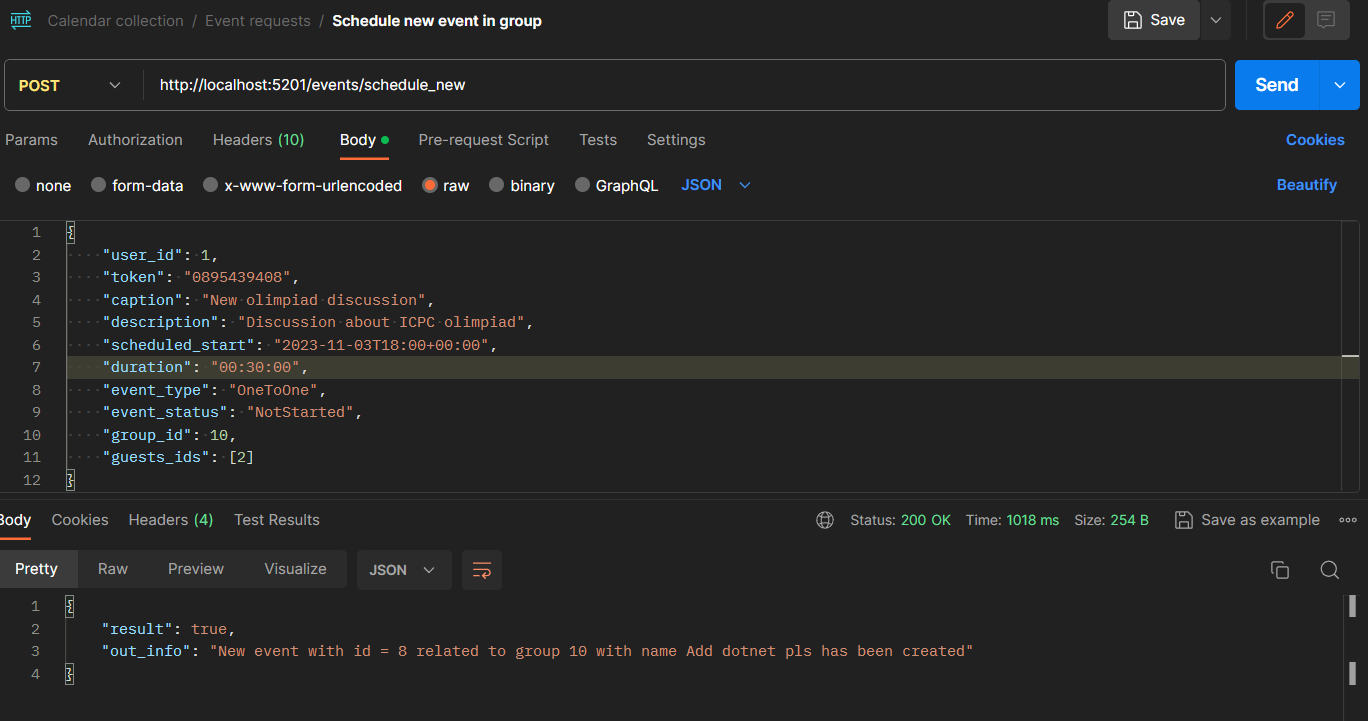


Рисунок 3.26. Тестирование запроса на создание нового мероприятия

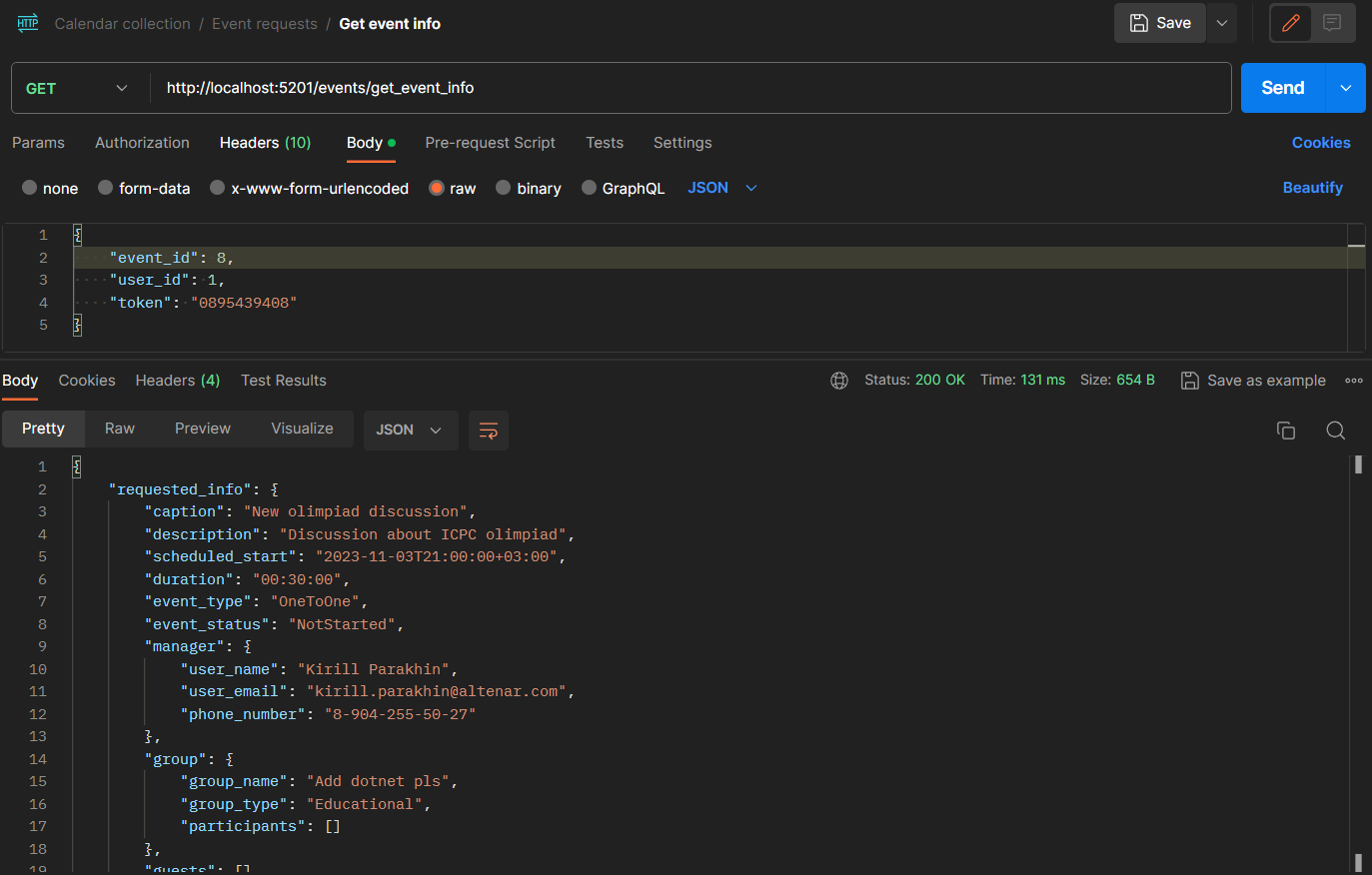


Рисунок 3.27. Тестирование запроса на получение полной информации о выбранном мероприятии

* + 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 3.28 – на нем изображено окно, в котором в графическом виде представлены модульные и интеграционные тесты для сервисного проекта приложения:

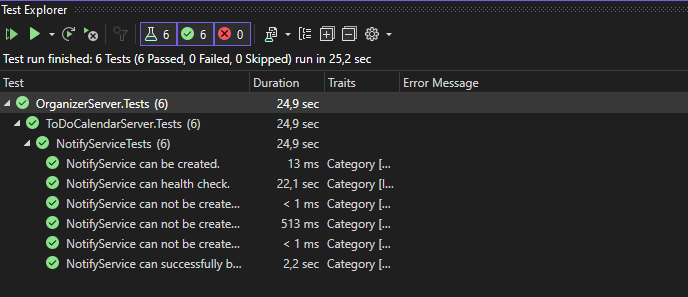


Рисунок 3.28. Окно Теst Explorer в Visual Studio

* + 1. Тестирование мобильного приложения с помощью тестового кейса

В данном разделе напишем несколько сценариев для тестирования виджетов в нашем приложении.

1. Сценарий регистрации нового пользователя в системе
2. Пользователь заходит на страницу регистрации;
3. Пользователь вводить требуемые данные для регистрации;
4. Пользователь получает ответ;
   1. Пользователь может получить успешный результат регистрации;
   2. Пользователь может получить сообщение о провальной регистрации.
   3. Пользователь может получить сообщение о проблеме с соединением к серверу.

Пользователь заполняет данные, требуемые для регистрации, и нажимает кнопку «Зарегистрироваться» (скриншот виджета представлен ниже на рисунке 3.29):

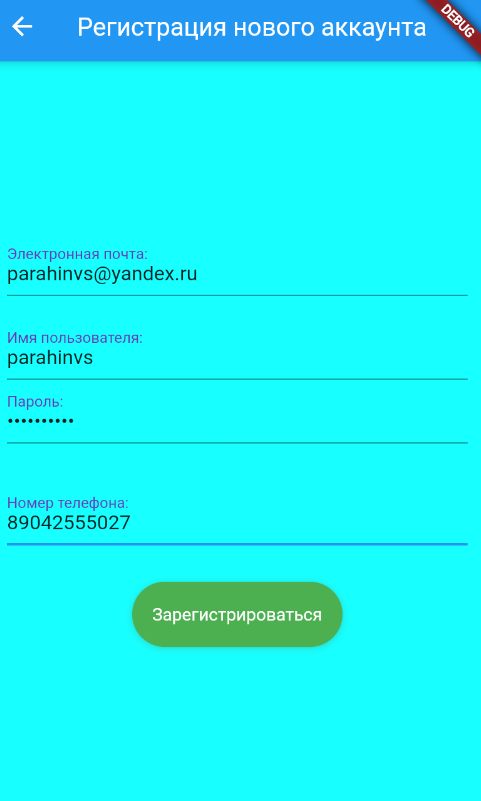


Рисунок 3.29. Скриншот ввода существуюшего аккаунта для регистрации

В том случае, если пользователь введёт электронную почту, с которой уже существует какой-то аккаунт в системе – то ему будет выведено сообщение о провальной регистрации. Для показа диалоговых сообщений был использован виджет showDialog (или AlertDialog).

На стороне сервера идёт проверка на существование пользователя, в нашем случае пользователь существует, возвращается ответ с кодом 400 (BadRequest), обрабатываем ответ и выводим окно с сообщением для пользователя – скриншот виджета представлен ниже на рисунке 3.30:

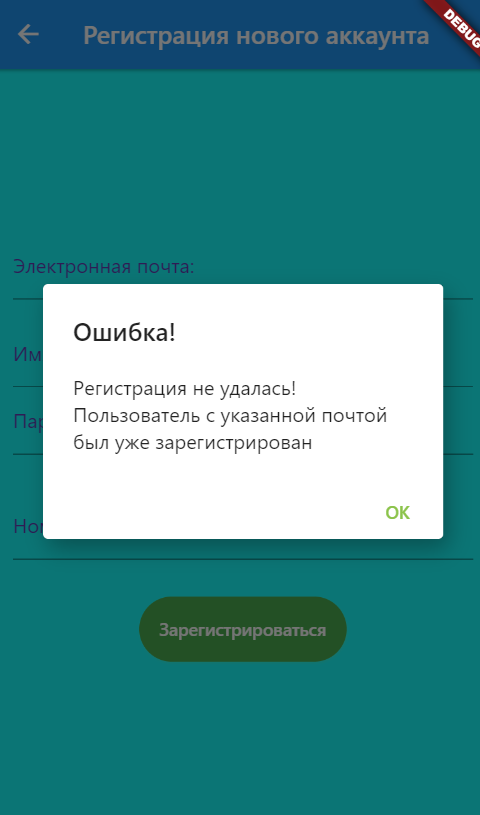


Рисунок 3.30. Скриншот вывода ошибки при помощи showDialog

После того, как пользователь уведомлён о невозможности использования текущей почты, он может ввести другую почту (например, [parahinvs1@yandex.ru](mailto:parahinvs1@yandex.ru) и имя «Кирюша»):

Далее пользователю будет выведено диалоговое окно о необходимости подтверждения его аккаунта в течение 5 минут. Для этого ему необходимо перейти на указанный при регистрации адрес электронной почты – и перейти по ссылке, отправленной ему в письме – скриншот диалогового окна представлен ниже на рисунке 3.31:

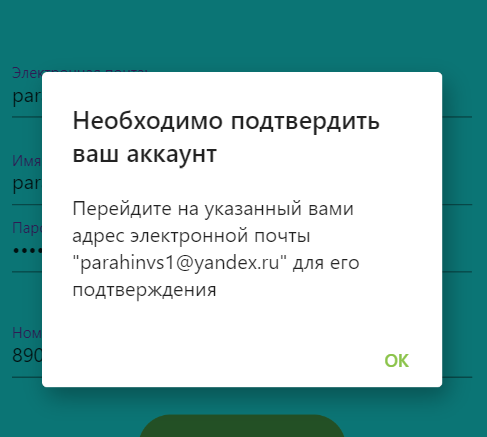


Рисунок 3.31. Уведомление о подтверждении аккаунта

Сообщение о подтверждении регистрации по почте представлено ниже на рисунке 3.32:

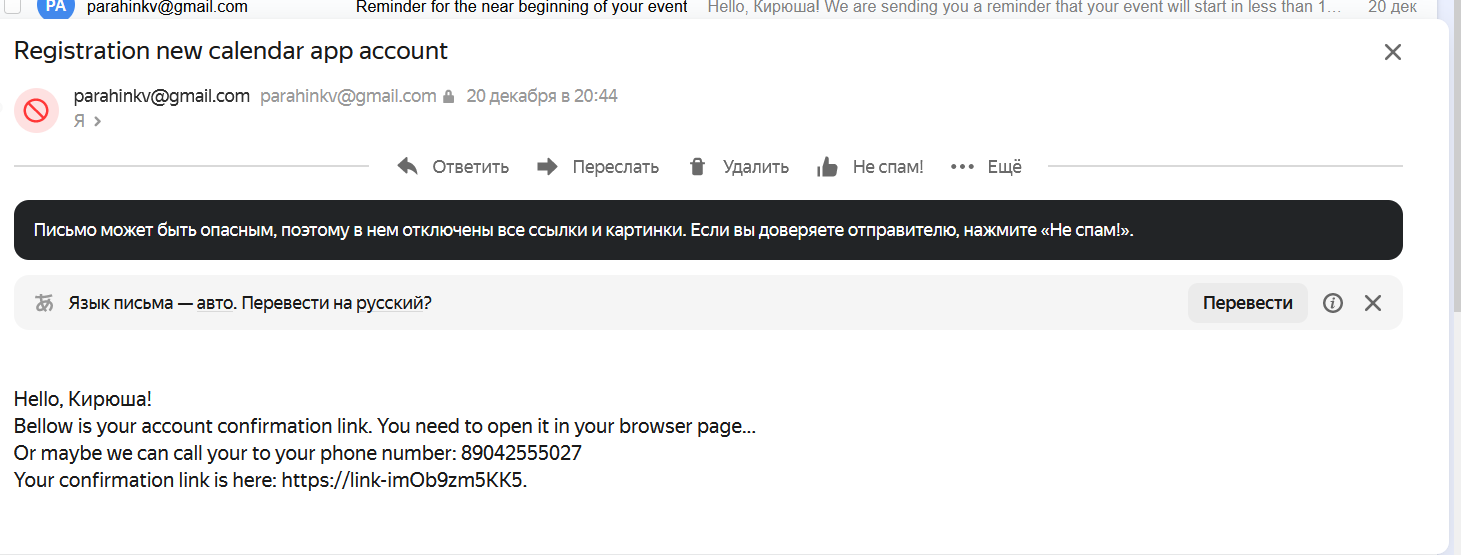


Рисунок 3.32. Сообщение подтверждения учетной записи на почте

После успешной регистрации приложение перенаправляет пользователя на главную страницу приложения – при этом сразу выполнив его авторизацию в системе – снизу можно увидеть сообщение об успешной регистрации в системе. Данное сообщение обрабатывалось при помощи виджета Flutter для показа информационных диалоговых сообщений.

* 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 3.33 представлен скриншот контейнеров, выполненный в программе Docker Desktop.

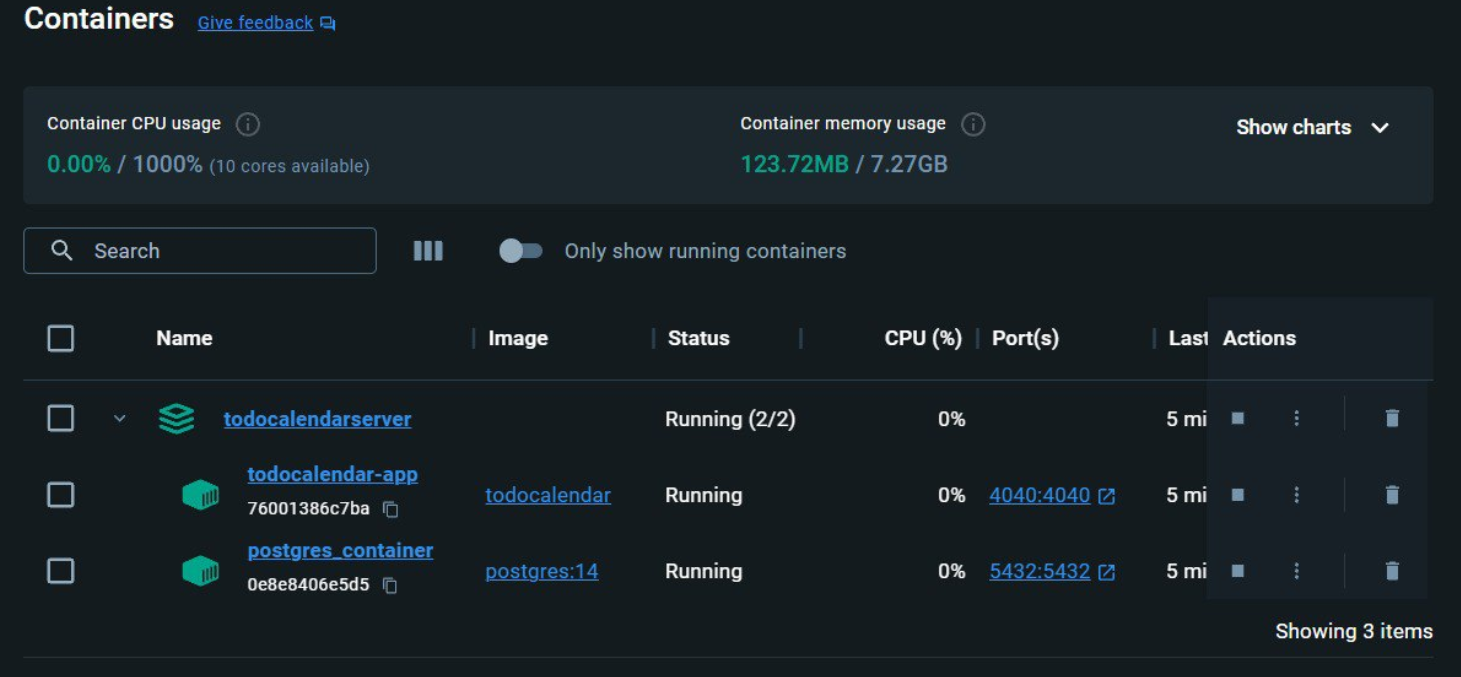


Рисунок 3.33. Развертывание сервера приложения и базы данных в Docker – контейнере

* + 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 3.34).

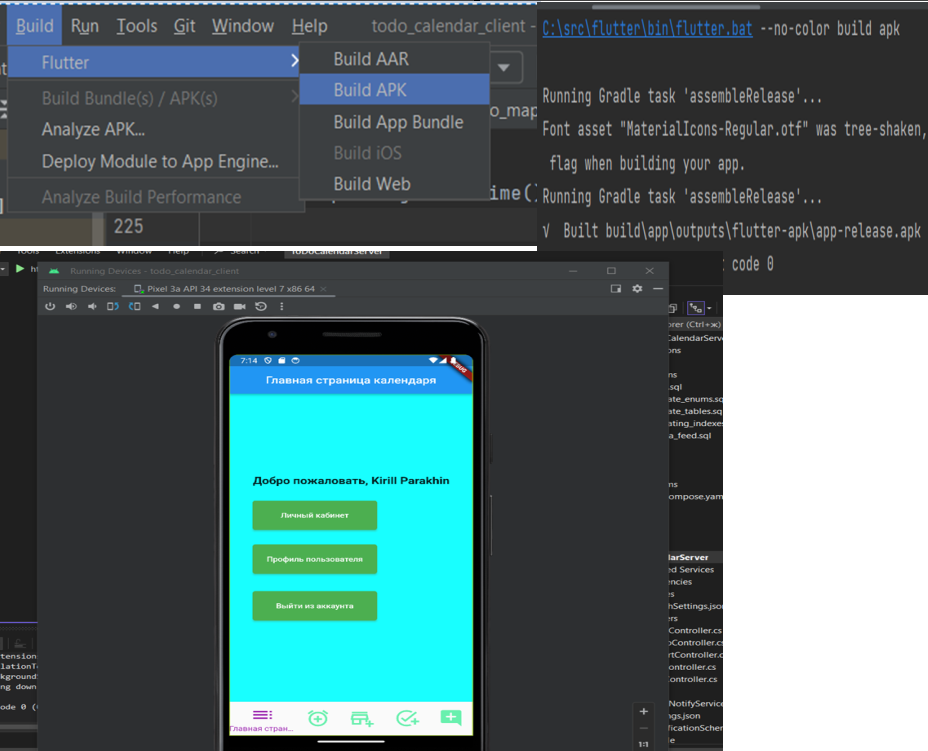


Рисунок 3.34. Развертывание мобильного приложения

* 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. 3.35 - 3.36 – 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.

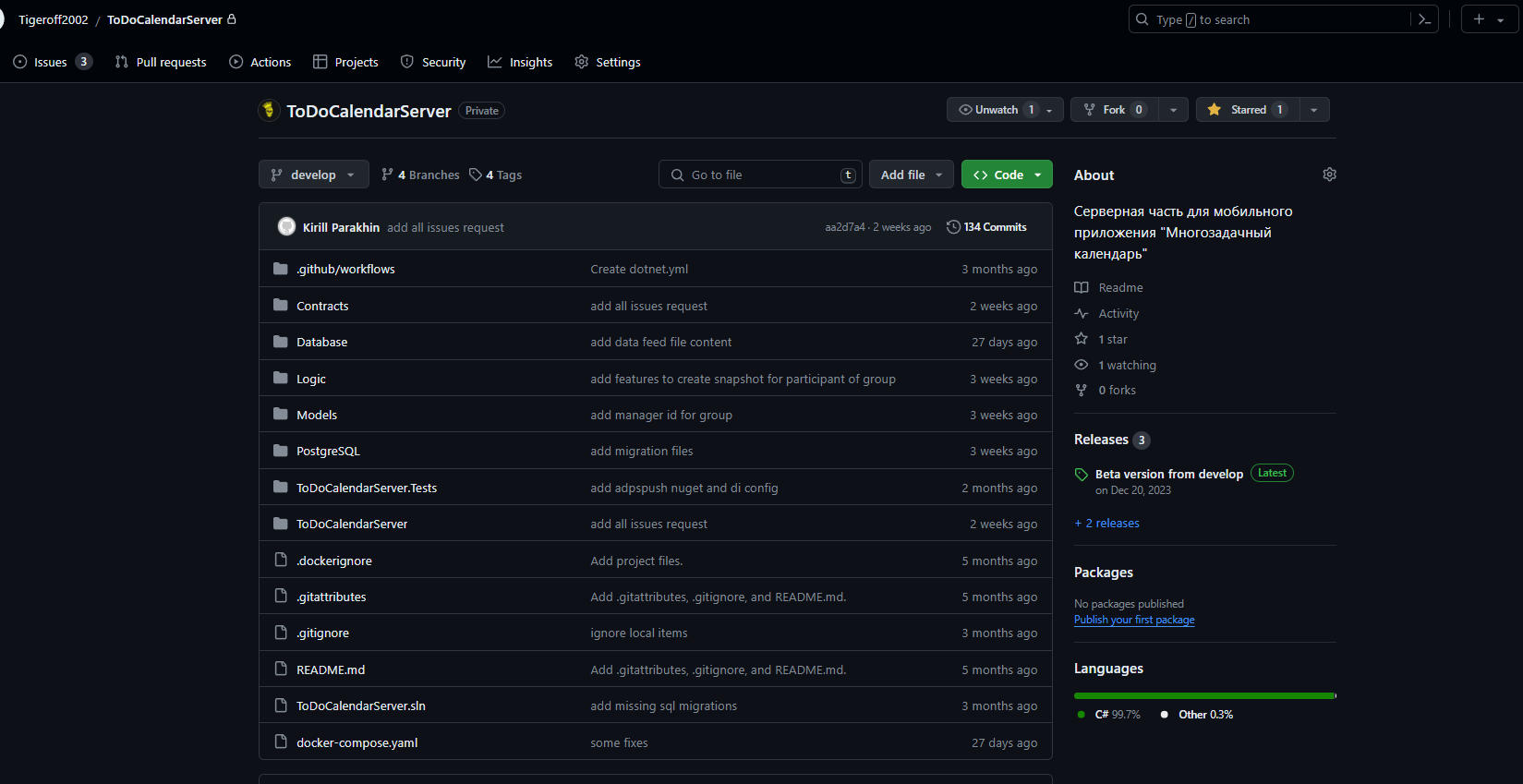


Рисунок 3.35. Репозиторий серверного ASP.NET - приложения

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

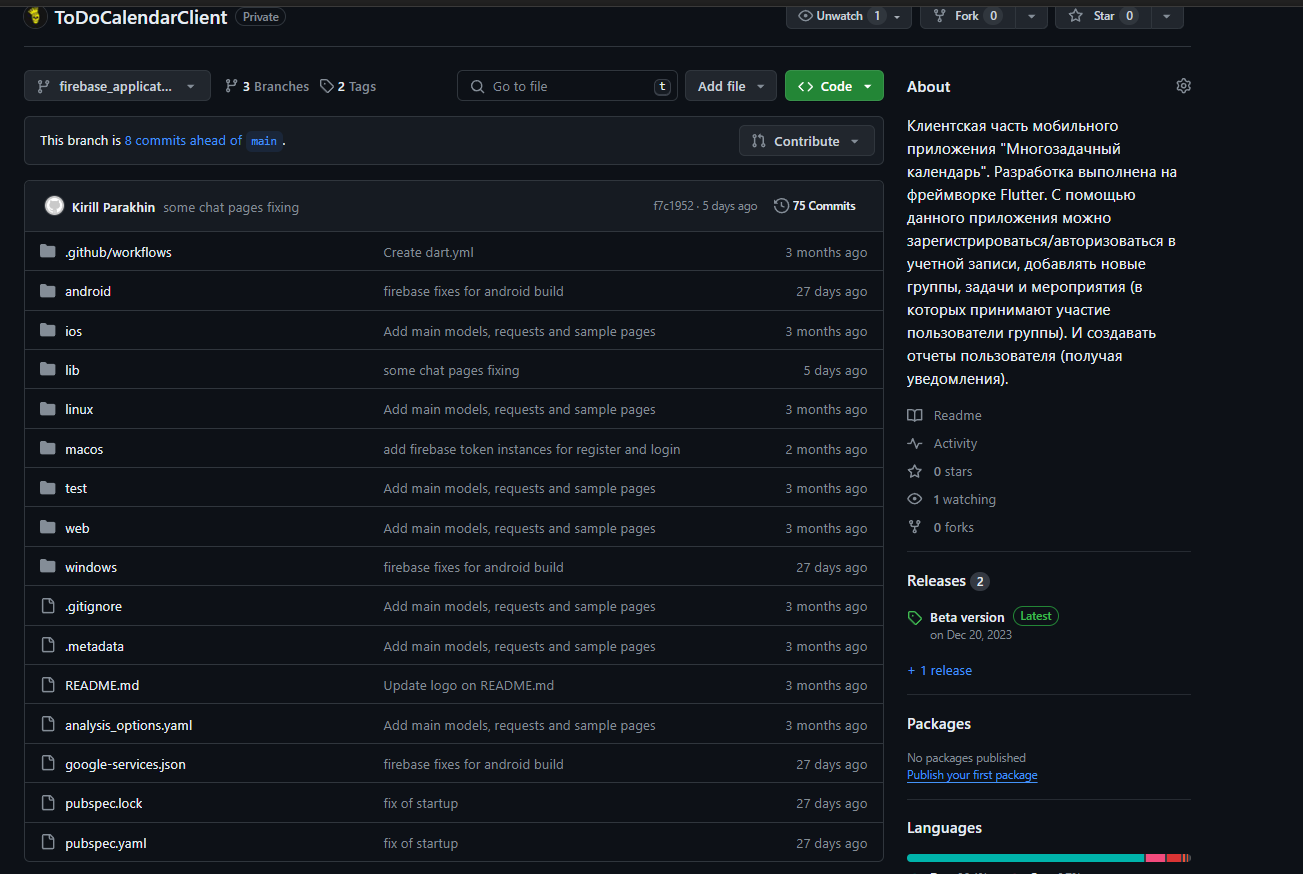


Рисунок 3.36. Репозиторий клиентского Flutter – приложения

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

## Календарный план проект и диаграмма Ганта

Календарный план проекта по разработке программной системы «Мобильный органайзер» расписан согласно задачам и их датам начала-окончания представлен ниже в таблице 4.1:

Таблица 4.1. 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 |

Диаграмма Ганта, построенная в MS Project согласно описанным в таблице 4.1 задачам проекта представлена ниже на рисунке 4.1:

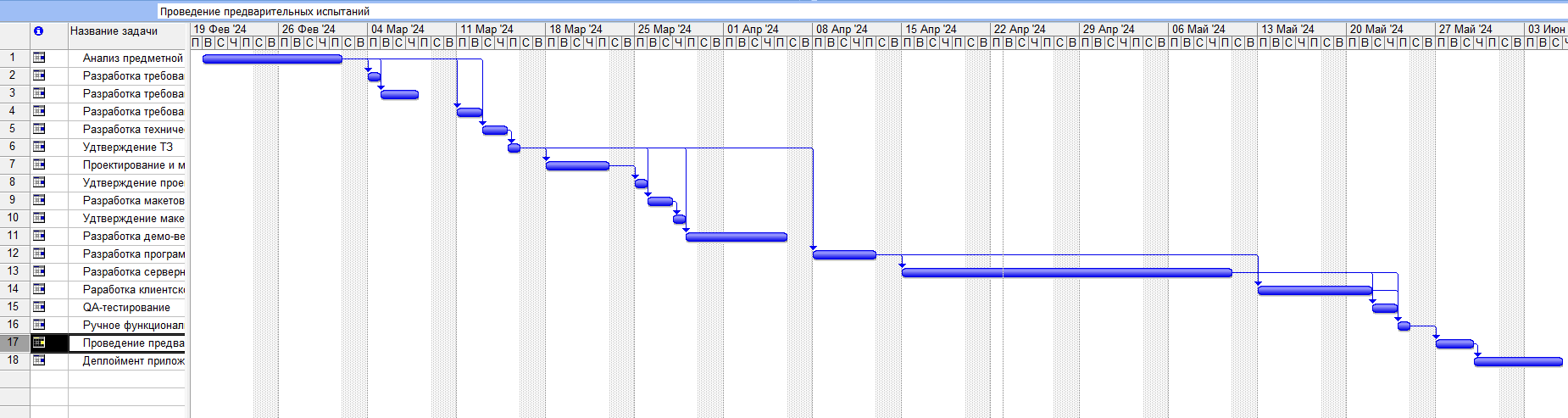


Рисунок 4.1. Диаграмма Ганта

## Экономическое обоснование проекта

Проведем экономическое обоснование целесообразности ведения проекта по разработке и внедрению программной системы «Мобильный органайзер», предназначенной для автоматизации деятельности IT-отделов предприятия. Ниже в таблице 4.2 приведены показатели экономической эффективности процессов до и после внедрения программной системы «Мобильный органайзер»:

Таблица 4.2– Показатели экономической эффективности

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Показатели | Обозначение | Единица измерения | Величина показателя | |
| До внедрения | После внедрения |
| Месячная зарплата сотрудника мониторинга | Z | руб. | 40000 | 40000 |
| Затраты труда сотрудника мониторинга | Т | чел./день | 5 | 2 |
| Месячная зарплата аналитика отчетов | Z2 | руб. | 50000 | 50000 |
| Затраты труда аналитика отчетов | Т2 | чел./день | 2 | 1 |
| Коэффициент накладных расходов | a |  | 0,2 | 0,2 |
| Коэффициент дополнительной зарплаты | b |  | 0,3 | 0,3 |
| Себестоимость часа работы ПЭВМ | Sq | руб./час | - | 40 |
| Время работы ПЭВМ для решения задачи (мес.) | Tz | маш./час | - | 12 |
| Время работы ПЭВМ на обучение, адаптацию, настройку оборудования | T | маш./час | - | 6 |
| Коэффициент настройки оборудования | g |  | - | 0,1 |
| Длительность проектирования | N | год |  | 0,3 |
| Среднее количество рабочих дней в месяце | Q | дней | 22 | 22 |
| Коэффициент прочих расходов | h |  | 0,1 | 0,2 |
| Нормативный коэффициент эффективности капитальных вложений | Еnce | - | - | 0,3 |

Коэффициент экономической эффективности Er рассчитывается по формуле:

**Er = S/К = 131 715 / 80 300 = 1,64** (11)

*Полученное Er> Еnce(0,3), от сюда следует, что технология является эффективной.*

Т - срок окупаемости затрат на решение задачи:

**Т = К/S = 0,61 (7 месяцев и 2 недели)**

*Результаты расчета экономической эффективности проекта - годовая экономия 131 715 руб.*

Конкретные результаты по автоматизации бизнес-процессов:

После автоматизации системы были достигнуты следующие результаты:

- своевременное поступление статистики и аналитики, необходимых для принятия управленческих решений

- снижение трудоемкости обработки экономической информации

- обеспечение автоматической отчетности

- обеспечение оперативности получения статистики

- обеспечение хорошего уровня анализа

При решении комплекса задач по автоматизации процессов в организации IT-отделов предприятия можно выделить следующие направления совершенствования деятельности предприятия:

* Сокращение операций, выполняемых вручную
* Снижение трудоемкости ввода информации
* Обеспечение единого синхронизированного хранилища данных

Изменений в организационной структуре при внедрении новой технологии не произошло.

## Анализ инвестиций, трудоемкости и ресурсоемкости

Проведем рассчет ROI при внедрении программной системы «Мобильный органайзер».

Предлагаемые изменения:

* Новая система поиска и трекинга задач;
* Автоматическая система составления отчетов
* Автоматическое отлавливание и составление отчетов по поводу ошибок в работе системы

Изначально необходимо классифицировать все виды ресурсов. Для сотрудников необходимо определить годовые заработные платы, а для технической инфраструктуры – стоимость содержания в год.

Вторым шагом было определение преимуществ от внедрения.

Преимущества, которые может принести программная система «Мобильный органайзер»:

* Сокращение времени поиска и отслеживания задач;
* Сокращение времени на планирование и отслеживание мероприятий
* Ведение автоматической отчетности о ходе выполнения проекта;
* Система составления отчетов по поводу ошибок в работе системы и алертов о возникающих багах и ошибках

Оценка окупаемости инвестиций.

Окупаемость инвестиций – ROI, определяется как отношение получаемой прибыли, сэкономленных затрат к затратам на внедрение ИТ- решения.

Чем выше прибыль от проекта и ниже затраты по его поддержке, тем больше показатель ROI.

Введем понятие базовой единицы – в качестве базы берется минимальный годовой оклад сотрудника (например, 12 месяцев\*1,3\*МРОТ (19242 руб.)).

Сумму экономии определим равной количеству базовых годовых окладов по должности.

Рассчитаем итоговый ROI за 3 года (предварительно проведя рассчет показателей за каждый год): ROI= = 1,21

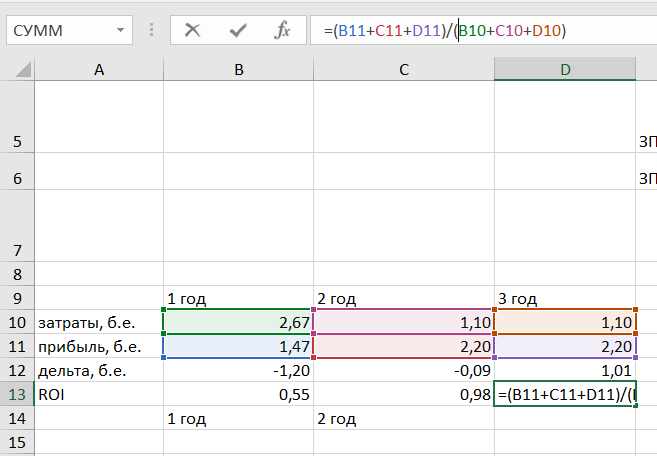


Рисунок 4.2. Расчет ROI по годам

Таким образом, так как коэффициент окупаемости инвестиций больше единицы за три года (равен 1,21), то это говорит о том, что проект по применению ПС «Мобильный органайзер» для совершенствования процесса отслеживания ошибок не будет убыточным и все вложения окупятся через 3 года.

График изменения ROI представлен на рисунке 4.3, расчеты – на рисунке 4.2.

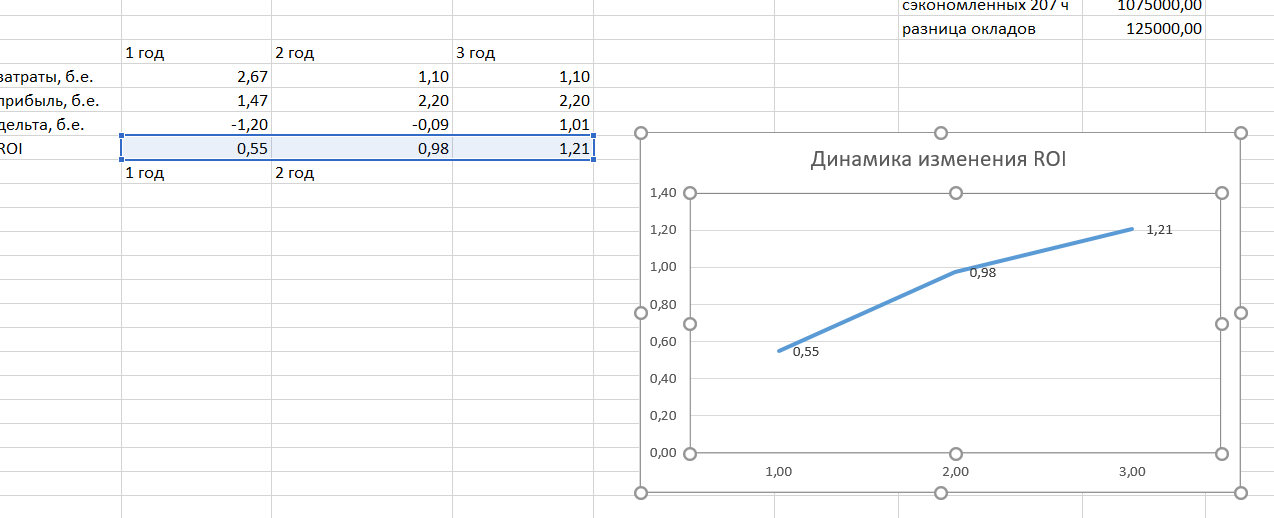


Рисунок 4.3. График изменения ROI

## Проведение информационного менеджмента

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 "О налоге на прибыль предприятий и организаций" [21].

• 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 ходе обдумывания – так как это план из СРС по информационному менеджменту, в данной же ВКР участие принимал лишь только я.

Таблица 4.3 - Группы лиц заинтересованных.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Группа | Должность | Роль в проекте | Влияние на проект | Полномочия | Интерес к проекту |
| Исполнитель\Заказчик | Менеджер проекта | Формирование требований. Контроль процесса разработки | Влияние на разработку | Любые действия, связанные с управлением проектом | Высокая степень заинтересованности в реализации. Принятие ключевых решений |
| Исполнитель | Аналитик | Анализ и доработка требований заказчика | Влияние на разработку | Любые действия, связанные с архитектурой и анализом требований | Высокая степень заинтересованности. Исследовательская работа |
| Исполнитель | Backend разработчик | Разработка серверной части проекта | Влияние на разработку | Любые действия, связанные с разработкой серверного программного продукта | Высокая степень заинтересованности в реализации. Выполнение заданий. |
| Исполнитель | Мобильный Android- разработчик | Разработка клиентской части проекта | Влияние на разработку | Любые действия, связанные с разработкой мобильного приложения, подстроенного под серверное приложение | Высокая степень заинтересованности в реализации. Выполнение заданий. |
| Исполнитель | Тестировщик | Проверка разработанного кода на соответствие всем требованиям | Влияние на разработку | Любые действия, связанные с архитектурой и анализом требований | Высокая степень заинтересованности. Исследовательская работа |

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 4.4 представлена карта заинтересованных сторон проекта:

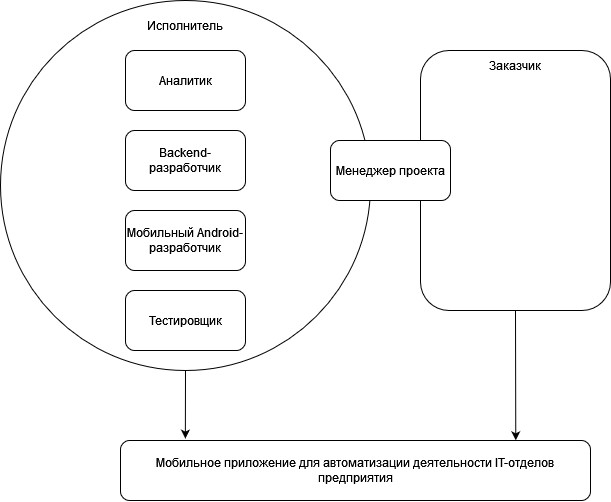


Рисунок 4.4. Заинтересованные стороны

Жизненный цикл проекта

Жизненный цикл проекта и его фазы представлены ниже на рисунках 4.5 – 4.6:

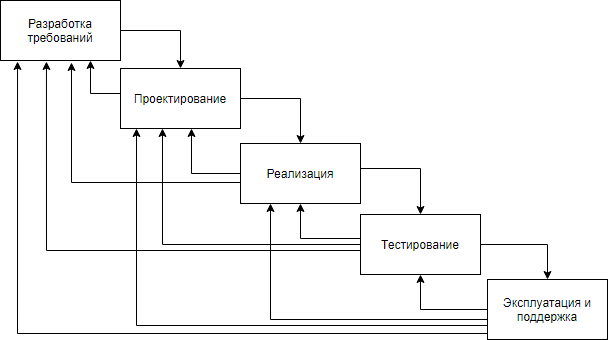


Рисунок 4.5. Жизненный цикл проекта

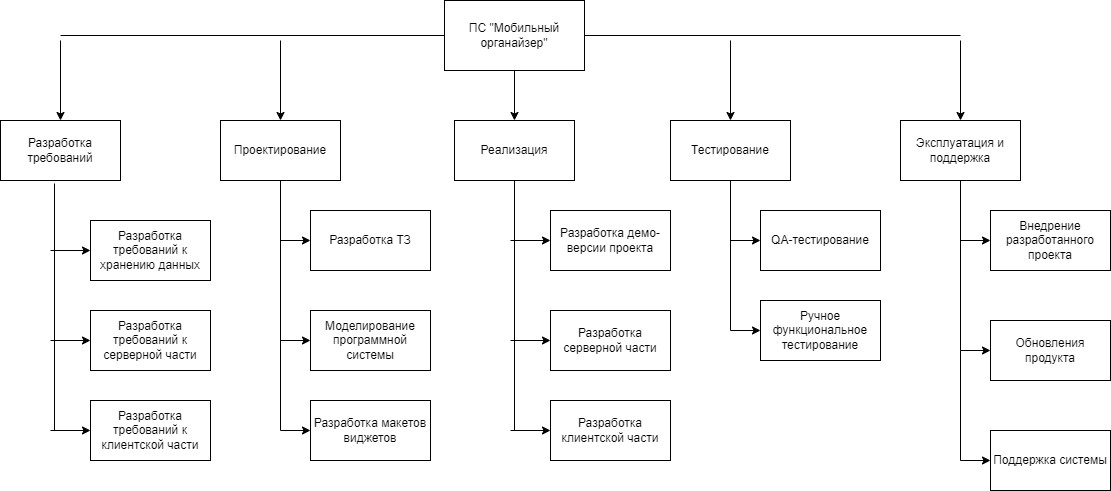


Рисунок 4.6. Фазы жизненного цикла проекта

## Анализ рисков проекта

План управления рисками проекта

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. При оценке воздействия риска определяется потенциальный эффект, который он может оказать на цель проекта (например, время, стоимость, содержание или качество).

Таблица 4.4 - План управления рисками

|  |  |  |  |
| --- | --- | --- | --- |
| Риск | Ответственный | Дата(до) | Мероприятие |
| Нехватка средств | Менеджер проекта | 13.02.2024 | Составить подробный план расходов, и заложить дополнительные расходы на возникновение  незапланированных ситуаций |
| Нарушение сроков | Аналитик | 21.04.2024 | Увеличить размер команды,  нанять новых татуировщиков и разработчиков. |
| Изменение  приоритетов заказчика | Менеджер  проекта | 05.05.2024 | Ввод «штрафов» в виде  дополнительной оплаты каждого нового изменения и  изменение даты релиза проекта. |
| Утечка  персональных данных | Разработчики-сотрудники (Backend-разработчик и мобильный разработчик) | 05.05.2024 | Внедрение в проект технологий шифрования данных. |
| Ошибка в планировании проекта | Менеджер проекта | 14.04.2024 | Заложить дополнительное время на разработку и обсуждение возникших вопросов по проекту, ввести мероприятие – техническая неделя. Перед разработкой команда должна изучить 50% требований, декомпозировать  и оценить их. |
| Проблемы с производительностью | Разработчик серверной части приложения | 05.05.2024 | Провести дополнительные мероприятия по обсуждению текущей архитектуры, пробовать оптимизировать использованные технические алгоритмы. |
| Разногласия между участниками проекта | Команда сотрудников (хотя в конечном случае менеджер проекта) | 26.05.2024 | Проводить обсуждения в голосовых видеоконференциях, фиксировать все стороны и мнения в виде разделов документации. В каждый следующий раз возвращаться к ним – и делать выводы по достигнутым результатам. |

## 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:

1. 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), теперь необходимо только правильно заполнить параметры составления отчета и суметь его корректно проанализировать.

1. 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

1. Сокращение операций, выполняемых вручную

Было: каждый новый инцидент, влияющий на работоспособность системы (например, перезапуск или ошибка в логике программных компонентов) 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.

1. 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.

1. 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.

1. Обеспечение хорошего уровня анализа

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

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

# 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;
  + Произведен выбор и сравнительный обзор аналогов;
  + Выделены функциональные и нефункциональные требования программной системы;
  + Произведено проектирование системы, разработана схема взаимодействия приложений;
  + Выбраны и обоснованы средства разработки программной системы;
  + Произведена разработка серверного приложения и базы данных;
  + Произведена разработка мобильного приложения для операционной системы Android;
  + Программная системы развернута и протестирована различными методами;
  + Проведен информационный менеджмент проекта разработки системы;
  + Выполнено экономическое обоснование для внедрения системы.

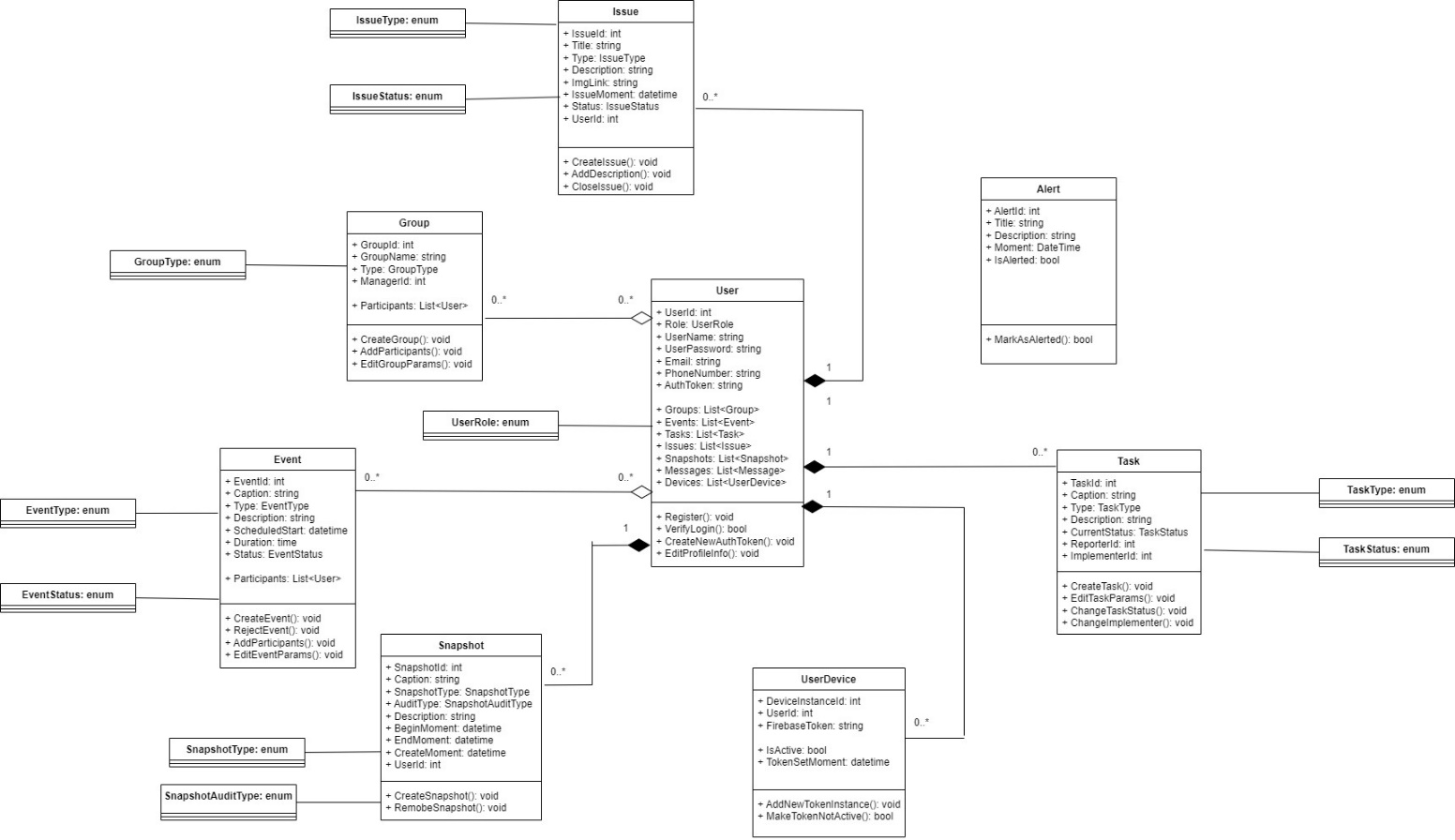
# Список использованных источников

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. Mastering PostgreSQL 13 - Packt Publishing, Birmingham, UK, 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. Л. В. Пирская. Разработка мобильных приложений в среде Android Studio. 2019. №1. С.120-230. ISBN 978-5-9275-3346-6.
7. К.В. Парахин. Обзор приложения, связанного с автоматизацией деятельности IT-отделов предприятия, 2024. №1. С.3-5 УДК 004.4. Дата публикация: 15.03.2024 г. ISBN:
8. Общие сведения об ASP.NET Core: - .URL: <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: - .URL: <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: - .URL: <https://www.softwaretestinghelp.com/best-it-automation-tools/> (дата обращения: 17.02.2024 г.).
11. Авторизация с помощью определенной схемы в ASP.NET Core: - .URL: <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: - .URL: <https://habr.com/ru/articles/311412/> (дата обращения: 10.03.2024 г.).
13. Создание Flutter Rest API: - .URL: <https://www.c-sharpcorner.com/learn/learn-flutter/flutter-rest-api> (дата обращения: 17.03.2024 г.).
14. Кэширование данных на устройстве Flutter: - .URL: <https://medium.com/@ahmedsherif4175/easiest-way-to-cache-data-in-flutter-a-step-by-step-guide-7e983d64198b> (дата обращения: 24.03.2024 г.).
15. Интеграционные тесты на платформе ASP.NET Core: - .URL: <https://learn.microsoft.com/ru-ru/aspnet/core/test/integration-tests?view=aspnetcore-8.0> (дата обращения: 28.03.2024 г.).
16. Работа с Flutter Syncfusion Calendar для показа виджета календаря: - .URL: <https://help.syncfusion.com/flutter/calendar/getting-started> (дата обращения: 31.03.2024 г.).
17. Работа с Flutter SfCartesian Chart для построения гистограмм: - .URL: <https://help.syncfusion.com/flutter/cartesian-charts/chart-types/histogram-chart> (дата обращения: 07.04.2024 г.).
18. Руководство по Docker Compose для начинающих: - .URL: <https://habr.com/ru/companies/ruvds/articles/450312/> (дата обращения: 14.04.2024 г.).
19. Push – уведомления в Android с помощью Firebase Cloud Messaging: - .URL: <https://habr.com/ru/articles/302002/> (дата обращения: 21.04.2024 г.).
20. Github Releases: публикация релизов: - .URL: <https://habr.com/ru/articles/185444/> (дата обращения: 06.05.2024 г.).
21. Закон РФ "О налоге на прибыль предприятий и организаций" от 27.12.1991 N 2116-1 (последняя редакция) : - .URL: <https://www.consultant.ru/document/cons_doc_LAW_200/> (дата обращения: 12.05.2024 г.)

# ПРИЛОЖЕНИЕ А. ДИАГРАММА ПРЕЦЕДЕНТОВ СИСТЕМЫ



ПРИЛОЖЕНИЕ Б. ЛОГИЧЕСКАЯ СХЕМА БАЗЫ ДАННЫХ



ПРИЛОЖЕНИЕ В. ДИАГРАММА РАЗВЕРТЫВАНИЯ ПРИЛОЖЕНИЯ

