Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

Кафедра информационных систем и программной инженерии

Отчет о прохождении предипломной практики

По направлению обучения

09.03.04 – Программная инженерия.

Выполнил:

студент гр. ПРИ-120

Парахин К.В.

Приняла:

доц. кафедры ИСПИ

Озерова М.И.

Владимир 2024 г.

Оглавление

[Ход работы 3](#_Toc166238599)

[Календарный график прохождения практики 3](#_Toc166238600)

[Введение 4](#_Toc166238601)

[**1** АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 6](#_Toc166238602)

[1.1. Описание предметной области 6](#_Toc166238603)

[1.2. Задачи приложения и обоснование его актуальности 7](#_Toc166238604)

[1.3. Описание набора функций системы 9](#_Toc166238605)

[1.4. Описание категорий пользователей системы и прецедентов 10](#_Toc166238606)

[2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ 12](#_Toc166238607)

[2.1. Функциональная декомпозиция системы 12](#_Toc166238608)

[2.2. Описание состава данных 16](#_Toc166238609)

[2.3. Требования к разрабатываемой системе 18](#_Toc166238610)

[3 РАЗРАБОТКА ПРОГРАММНОЙ СИСТЕМЫ 22](#_Toc166238611)

[3.1. Выбранный вариант разработки программной системы 22](#_Toc166238612)

[3.2. Реализация серверного приложения 24](#_Toc166238613)

[3.3. Описание структуры серверного приложения 25](#_Toc166238614)

[3.4. Спецификация API 27](#_Toc166238615)

[3.5. Описание разработки клиентского приложения 28](#_Toc166238616)

[3.6. Описание интерфейса мобильного приложения 30](#_Toc166238617)

[3.10. Тестирование системы 35](#_Toc166238618)

[3.11. Развертывание приложения 37](#_Toc166238619)

[3.12. Ведение репозиториев программной системы 38](#_Toc166238620)

[4 ИНФОРМАЦИОННЫЙ МЕНЕДЖМЕНТ 39](#_Toc166238621)

[4.1. Календарный план проект и диаграмма Ганта 39](#_Toc166238622)

[4.2. Экономическое обоснование проекта 41](#_Toc166238623)

[4.3. Анализ бизнес-процессов 44](#_Toc166238624)

[Описание бизнес-процесса 44](#_Toc166238625)

[4.6. Анализ рисков проекта 47](#_Toc166238626)

[4.7. Подведение выводов о результате ввода системы в использование 48](#_Toc166238627)

[Вывод 50](#_Toc166238628)

# Ход работы

Место прохождения практики: ООО «АИСТСОФТ» (аккредитованная IT-компания Altenar)

Срок проведения производственной практики: с 15 апреля 2023 года по 18 мая 2024 года (продолжительность - 5 недель)

Календарный график прохождения практики

|  |  |  |  |
| --- | --- | --- | --- |
| № п.п. | Виды работ | Рабочее место студента | Время в днях (или неделях) |
| 1 | Настройка окружения, установка ПО и доступов. Получение ТЗ | рабочее место в офисе | 1 неделя |
| 2 | Проектирование системы | удаленное рабочее место | 1 неделя |
| 3 | Разработка серверного приложения с БД | удаленное рабочее место | 2 недели |
| 4 | Разработка мобильного приложения | удаленное рабочее место | 1 неделя |
| 5 | Составление отчета, исследовательская часть | удаленное рабочее место | 2 дня |

Введение

Целью прохождения производственной практики было получение теоретических, практических и аналитических навыков по выполнению требований, предоставляемых сотрудникам ИТ-компаний (в частности – разработчикам программного обеспечения и аналитикам процессов работы организации).

Конструкторская часть представляет собой реализацию программной системы «Мобильный органайзер» для автоматизации и оптимизации работы IT-отделов предприятия.

Данный проект 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.

Исследовательская часть представлена в главе «Информационный менеджмент». Она 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 и обзор бизнес-процессов после автоматизации системы.

# 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 поддержку.

* 1. Задачи приложения и обоснование его актуальности

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

а) Автоматизация деятельности и взаимодействия сотрудников IT-отдела

б) Создание иерархической организационной структуры проектов внутри IT-отдела

в) Введение автоматической системы отчетности и регистрации инцидентов

г) Хранение данных о сотрудниках, их мероприятиях, задачах, сообщениях

д) Обеспечение своевременного обмена информации в системе путем рассылки уведомлений

е) 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.

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:

а) Отсутствие единой системы, которая обеспечивает хранение данных сотрудников организации;

б) Отсутствие автоматической системы ведения отчетности по рабочей деятельности отдела предприятия;

в) Отсутствие системы мониторинга отказоустойчивости собственного программного продукта;

г) Отсутствие единой системы обмена информации и бизнес-сообщениями в системе;

д) Отсутствие грамотной оптимизации отдела аналитики и продуктового бизнеса.

* 1. Описание набора функций системы

Основные функции разрабатываемой системы, определяющие границы предметной области:

а) зарегистрироваться в системе

б) войти в систему

в) создавать группы пользователей, выбирая их тип по классификации

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

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

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

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

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

к) получать отчеты от приложения за определенный период времени)

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

м) рассылать уведомления на зарегистрированные мобильные устройства пользователей системы

н) получать push-уведомления и электронные сообщения по почтеОписание категорий пользователей системы и прецедентов

* 1. Описание категорий пользователей

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

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

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

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

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

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

1. ПРОЕКТИРОВАНИЕ СИСТЕМЫ
   1. Функциональная декомпозиция системы

Основные прецеденты (доступные всем пользователям системы):

а) Управлять своим аккаунтом

1) Зарегистрироваться

2) Авторизоваться

3) Выйти из системы

4)Подать заявку на получение роли системного администратора

б) Просматривать календарь мероприятий пользователя

1) Просматривать свой календарь

2) Просматривать календарь пользователей в группе

в) Управлять трекингом задач

1) Распределять задачи в группе по исполнителям

2) Контролировать смену статусов задач

3) Отослать уведомление о назначении задачи

г) Управлять мероприятиями

1) Выбрать тип мероприятия

2) Выбрать время и продолжительность

3) Добавить пользователей из группы на мероприятие

4) Отослать уведомления о приглашении на мероприятие

д) Создать отчет о личной деятельности

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

2) Выбрать период для получения отчетности

3) Выбрать тип представления отчета (текстовый или графический)

е) Регистрировать проблемы рабочего процесса

1) Фиксировать и отправлять инциденты о неисправностях работы системы

2) Сигнализировать о прочих нарушениях рабочего процесса

ж) Получать уведомления

1) Получать уведомления о назначении задачи на исполнение

2) Получать уведомления о приглашении на мероприятие

3) Получать уведомления о начале мероприятий

Прецеденты менеджера группы:

а) Заниматься менеджментом группы

1) Выбрать тип группы

2) Добавить участников в группу

3) Редактировать параметры группы

4) Удалить участника группы

б) Создать отчет по деятельности группы

Прецеденты системного администратора системы:

а) Осуществлять поддержку

1) Получать уведомления о неисправностях в работе системы

2) Получать alert-сообщения в авто-режиме

3) Менять статусы инцидентов при их закрытии

Диаграмма прецедентов в нотации UML изображена ниже на рисунке 1:



Рисунок 1 - Расширенная диаграмма прецедентов

* 1. Описание состава данных

Выделим следующие абстракции данных (сущности системы):

а) User (пользователь с некоторыми учетными данными).

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

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

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

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

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

ж) Alert (проблема в работе системы).

з) Snapshot (отчет с данными).

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

Диаграмма классов (уровня проектирования) системы в нотации UML представлена ниже на рисунке 2:

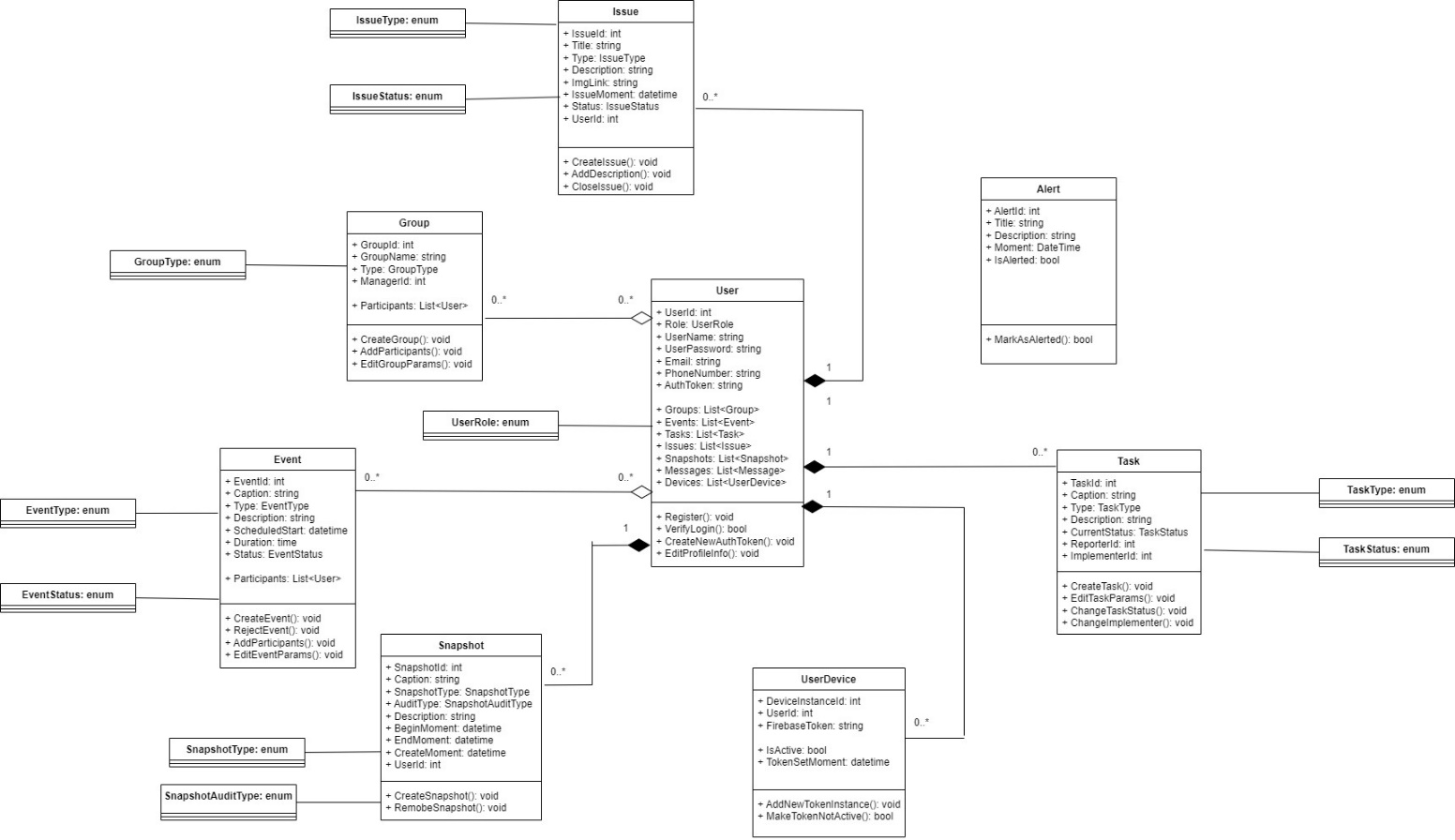


Рисунок 2 - Диаграмма классов уровня проектирования

## Требования к разрабатываемой системе

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

Функциональные требования

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

а) Функция A

1) Название - возможность использовать авторизацию пользователей;

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

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

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

б) Функция Б

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

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

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

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

в) Функция В

1) Название – возможность распределения пользователей по группам;

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

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

г) Функция Г

1) Название – возможность планирования пользовательских мероприятий;

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

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

д) Функция Д

1) Название – возможность создания пользовательских задач;

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

3) Назначение – сохранение пользовательских задач в базе данных.

е) Функция Е

1) Название – возможность регистрирования проблем в работе системы;

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

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

ж) Функция Ж

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

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

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

з) Функция З

1) Название – получать уведомления о событиях в системе;

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

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

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

Нефункциональные требования

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

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

в) Масштабируемость компонентов системы;

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

# РАЗРАБОТКА ПРОГРАММНОЙ СИСТЕМЫ

## Выбранный вариант разработки программной системы

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

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

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

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

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

Также планируется использовать следующие прикладные инструменты разработки и тестирования:

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

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

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

* 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:

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

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

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

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

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

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

Ниже на рисунке 3 показан скриншот обозревателя решений с полной структурой всех проектов

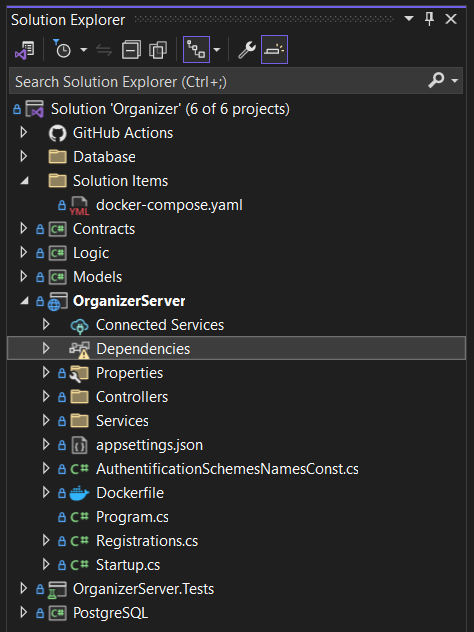


Рисунок 3 - Структура проектов серверного приложения

* 1. Спецификация API

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

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

Все запросы, содержащие тело – должны иметь заголовок вида:

*Content-type: Application/json*

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

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

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

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

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

## Описание разработки клиентского приложения

Клиентская часть приложения была разработана с использованием фреймворка Flutter.

Описание структуры клиентского приложения

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

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

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

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

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

Ниже на рисунке 4 показана структура папок и виджетов данного клиентского мобильного приложения:

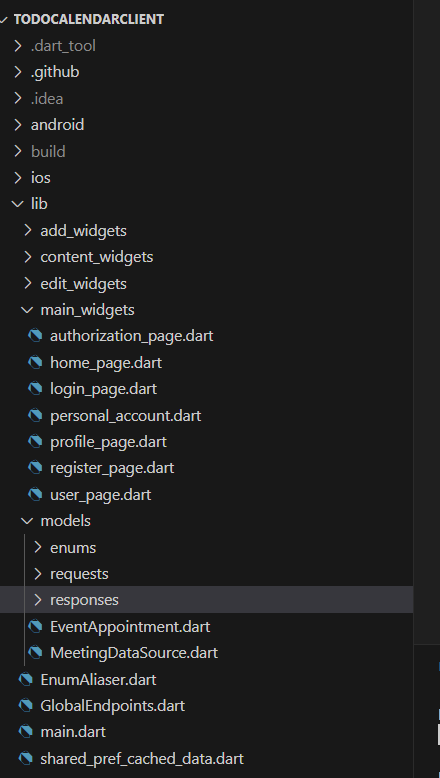
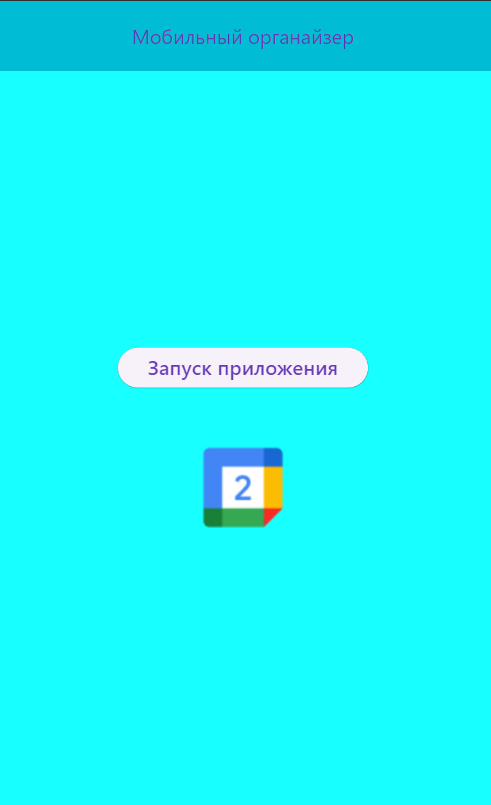
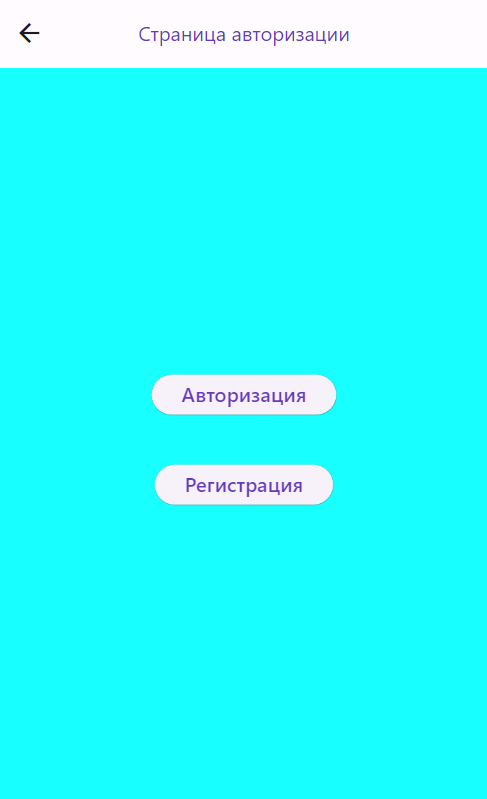


Рисунок 4 - Структура папок клиентского Flutter – приложения

* 1. Описание интерфейса мобильного приложения

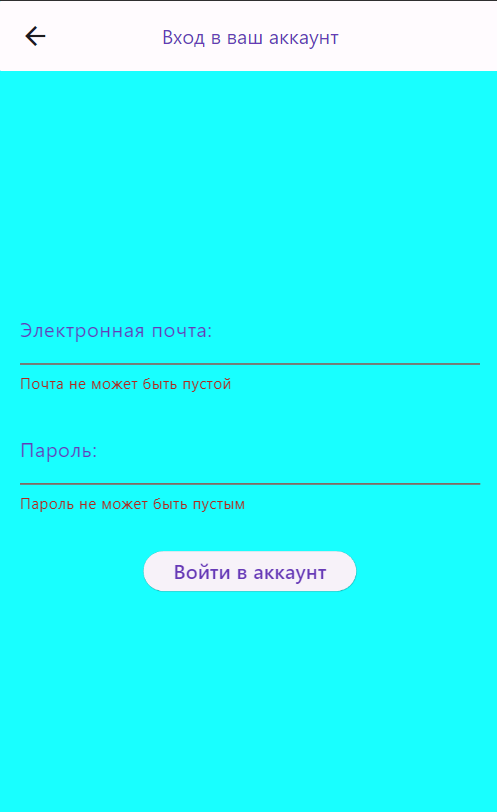
Создадим основные страницы (виджеты) – для регистрации, авторизации, для личного кабинета пользователя (с перечислением мероприятий, групп, задач и отчетов пользователя), а также дополнительные виджеты для добавления/редактирования мероприятия в календарь, новой группы, новой задачи на реализацию, нового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

Ниже приведены компоновка скриншотов стартовой страницы и страницы для неавторизованного в системе пользователя (рис. 5.1 – 5.2):

Рисунки 5.1 – 5.2 - 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.

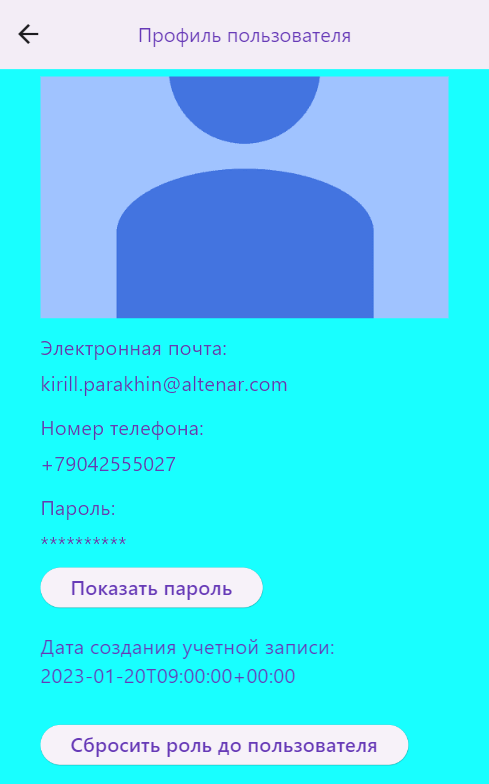
Страницы авторизации и регистрации представлены на рисунках 5.3 – 5.4 ниже. На данных страницах, как и на всех остальных страницах, на которых происходит ввода каких-то данных – происходит валидация вводимого текста:



Рисунки 5.3 – 5.4 - Страницы авторизации и регистрации

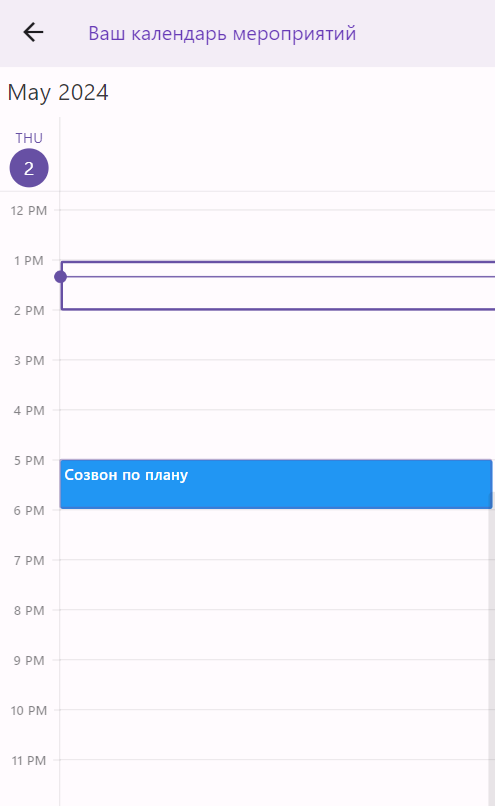
Главная страница пользователя, представляющая собой пример виджета с изменением состояния (в зависимости от изменения локального кэша меняется ее содержимое – в случае отсутствия данных о авторизованном пользователе – осуществляется перенаправление пользователя на страницу для авторизации).

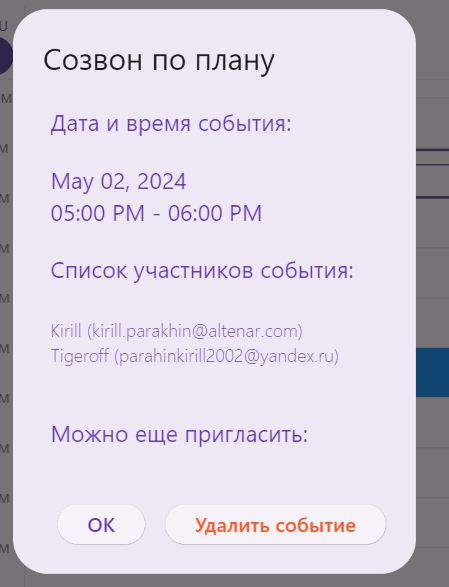
На главной странице пользователя (рисунок 5.5) и странице личного кабинета пользователя (рисунок 5.6) присутствует нижнее навигационное меню с различными режимами текущего окна. По умолчанию, открывается главная страница. При нажатии на другую панель – откроется другое состояние виджета.



Рисунки 5.5 – 5.6 - Главная страница и страница личного кабинета пользователя

Страница с календарными мероприятиями пользователя и выпадающими диалоговыми виджетами представлены ниже на рисунках 5.7 – 5.8:





Рисунки 5.7 – 5.8 - 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).

Ниже на рисунках 5.9 – 5.10 показана страница менеджера группы, на которой представлены отчеты по работе группы и страница конкретного отчета группы:

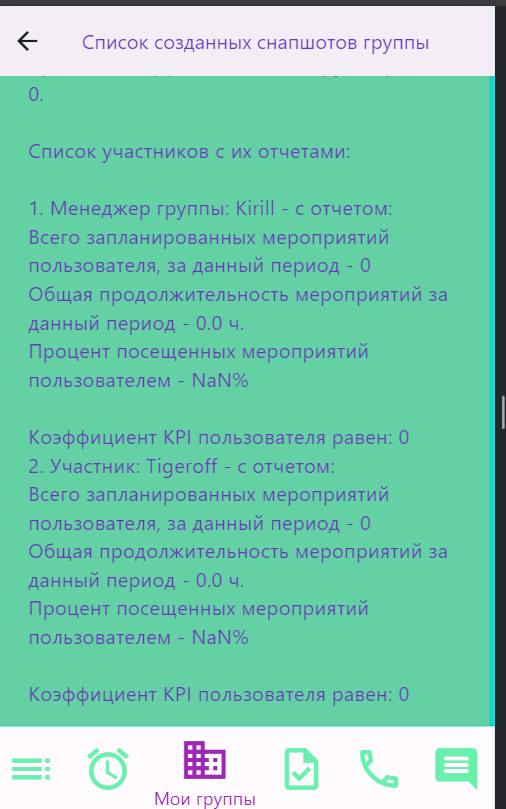
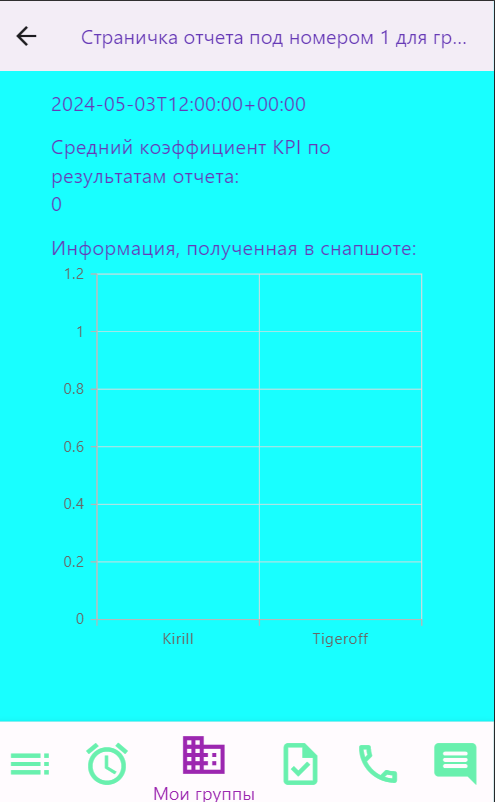


Рисунок 5.9 – 5.10 – Страница отчетов менеджера группы и страница отдельного отчета

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).

## Тестирование системы

Тестирование запросов к REST API серверного приложения

Для тестирования HTTP-запросов, поступающих на вход и уходящих на выход – для серверного приложения использовался программный продукт Postman. Ниже в таблице 1 представлен отрывок из спецификации API по тестированию путей регистрации/авторизации.

Таблица 1 - Запросы по регистрации/авторизации пользователей

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

* 1. Развертывание приложения

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

По итогу, получается контейнер приложения, к которому можно получить доступ в локальной сети по адресу url: 4040:4040 – ниже на рисунке 6 представлен скриншот контейнеров, выполненный в программе Docker Desktop.

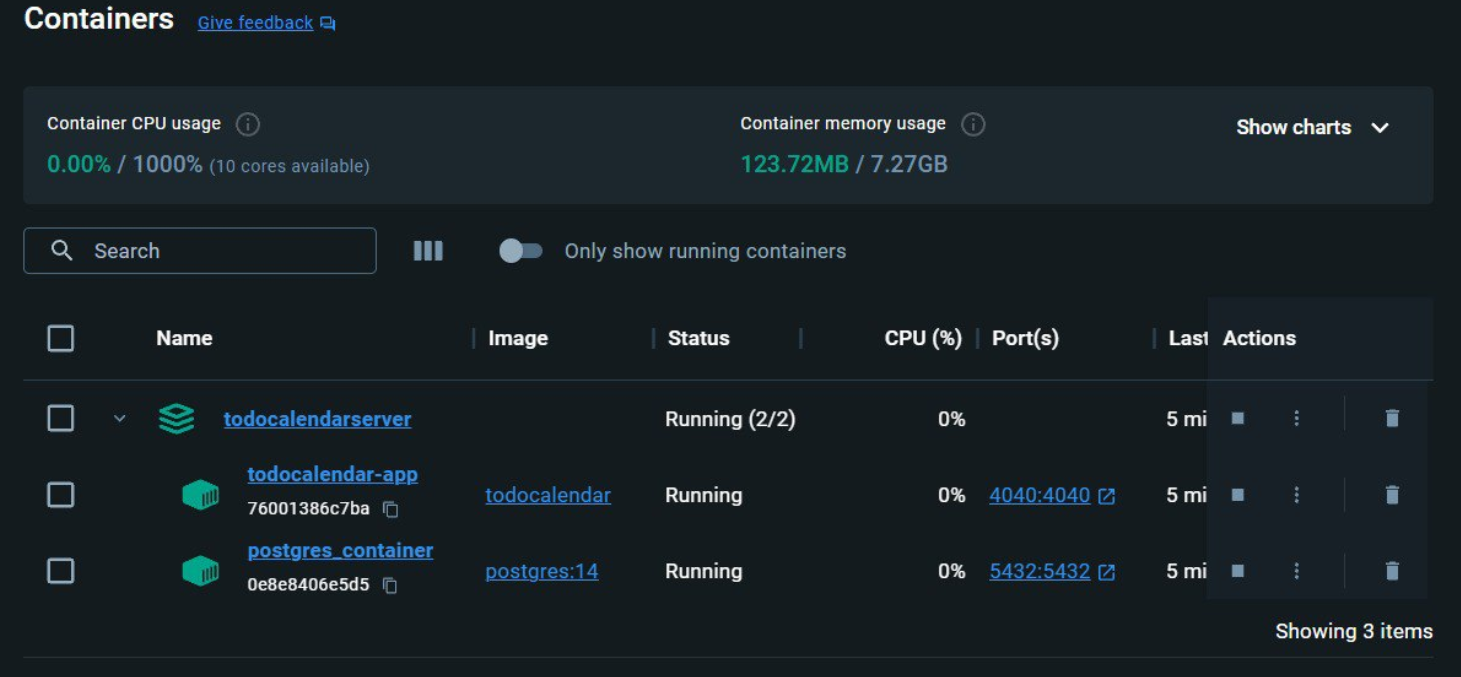


Рисунок 6 - Развертывание сервера приложения и базы данных в Docker – контейнере

* 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:

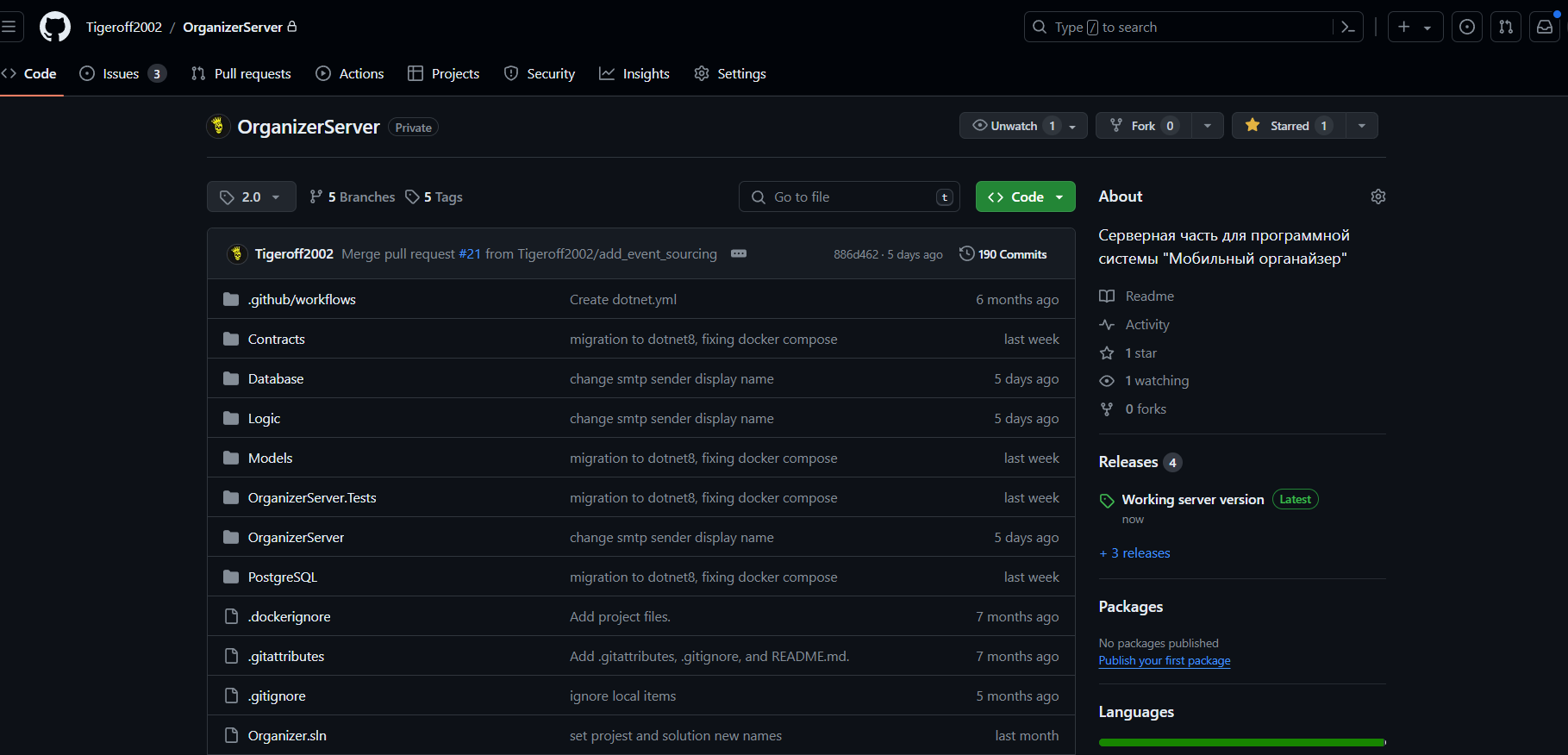


Рисунок 7.1 - Репозиторий серверного ASP.NET - приложения

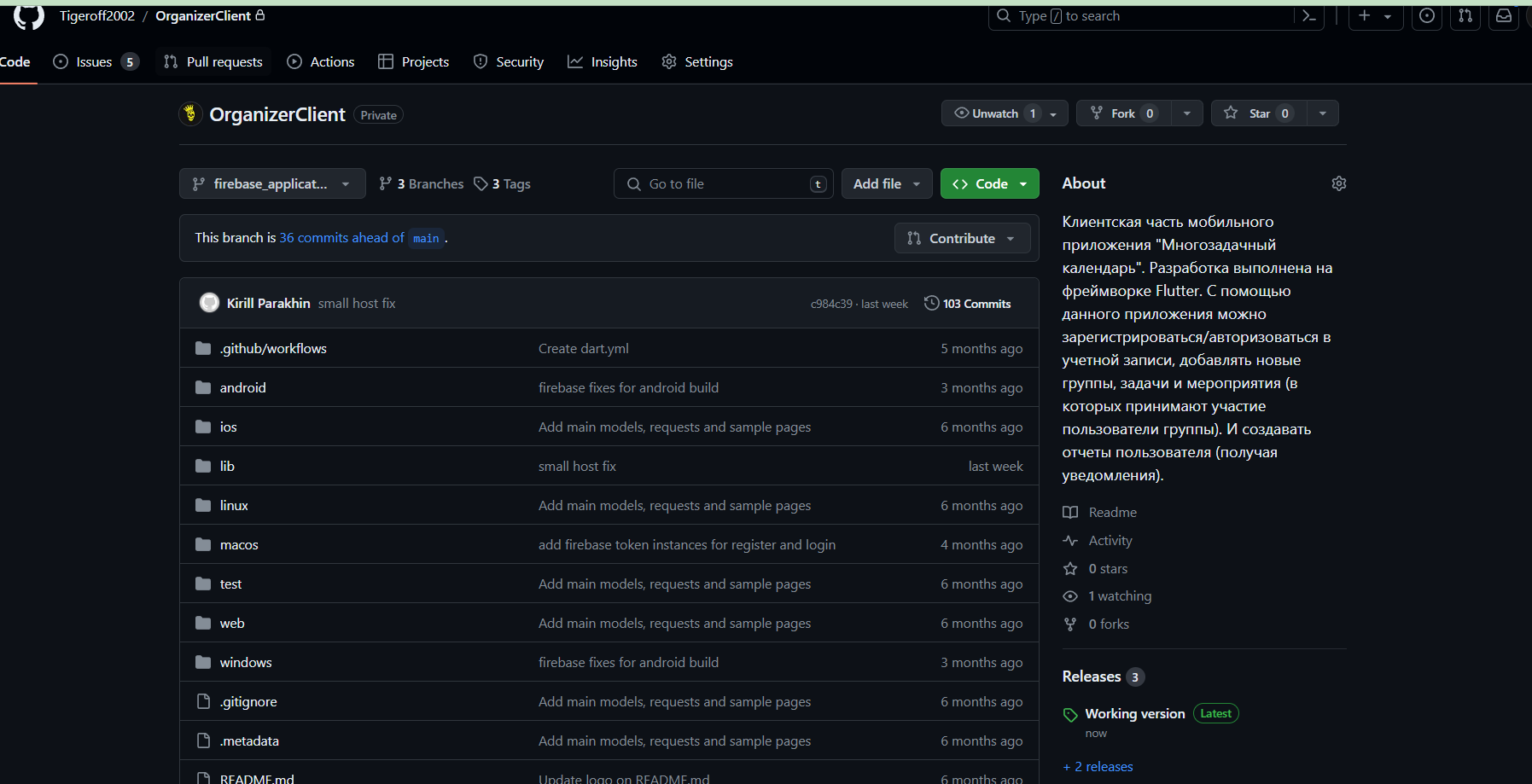


Рисунок 7.2 - Репозиторий клиентского Flutter – приложения

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 2:

Таблица 2 - Календарный план проекта

|  |  |  |  |
| --- | --- | --- | --- |
| Задача | Длит. (дней) | Начало | Окончание |
| Исследование предметной области | 9 | 20.02.2024 | 01.03.2024 |
| 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 кклиентской части | 2 | 11.03.2024 | 12.03.2024 |
| Разработка технического  задания | 2 | 13.03.2024 | 04.03.2024 |
| Проектирование и моделирование ПС | 5 | 18.03.2024 | 22.03.2024 |
| 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 8:

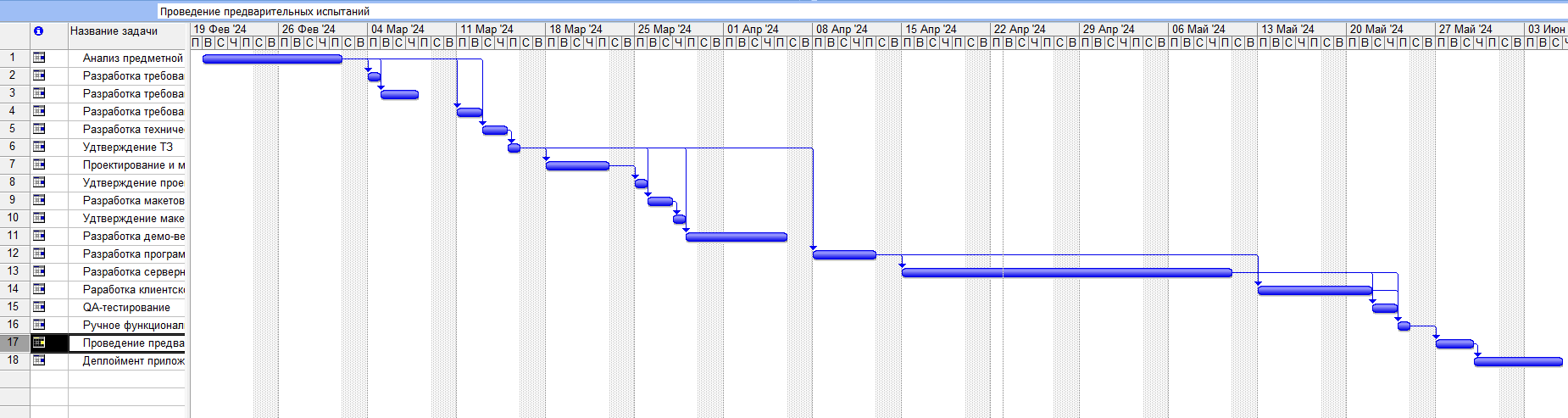


Рисунок 8 - Диаграмма Ганта

## 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 3 приведены показатели экономической эффективности процессов до и после внедрения программной системы «Мобильный органайзер»:

Таблица 3 – 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** (1)

*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.

* 1. Анализ бизнес-процессов

## Описание бизнес-процесса

Бизнес-процесс «Выпуск нового функционала программного продукта».

Кроме того, мною было выполнено задание по описанию бизнес-процесса, выполняемого сотрудниками организации, занимающейся проектированием, разработкой и выпуском программного обеспечения и остальных компонентов программно-информационной системы.

Ниже представлена модель as-is данного бизнес-процесса, выполненная в нотации IDEF0 на трех уровнях декомпозиции:

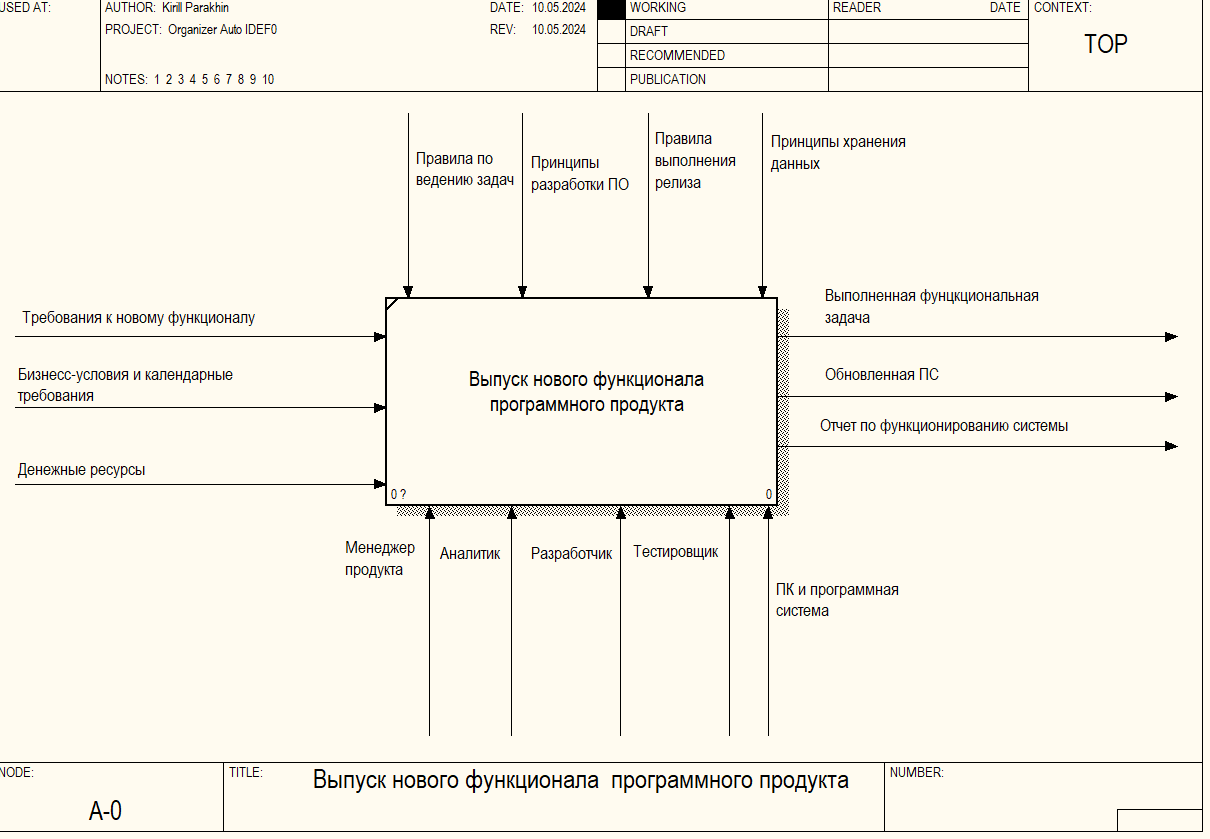


Рисунок 9.1. Модель as-is на первом уровне декомпозиции

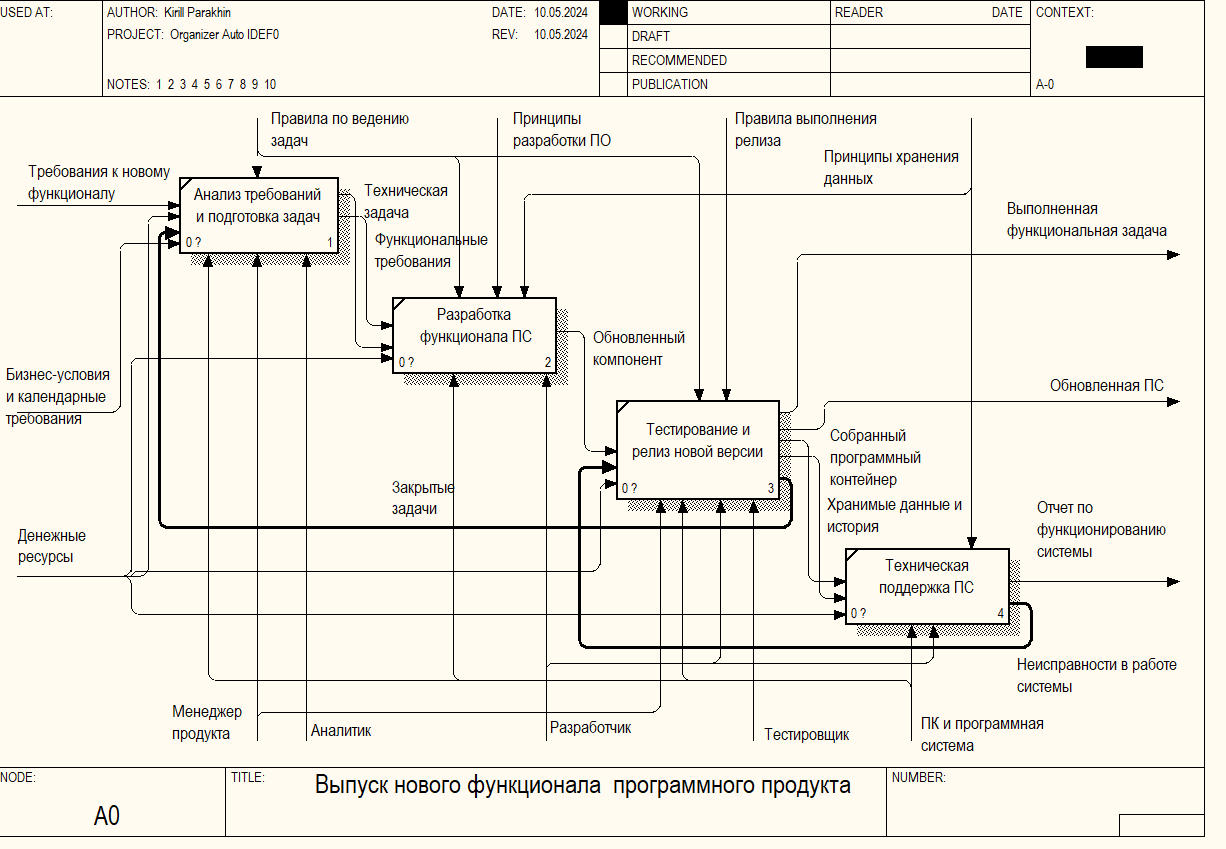


Рисунок 9.2. Модель as-is на втором уровне декомпозиции

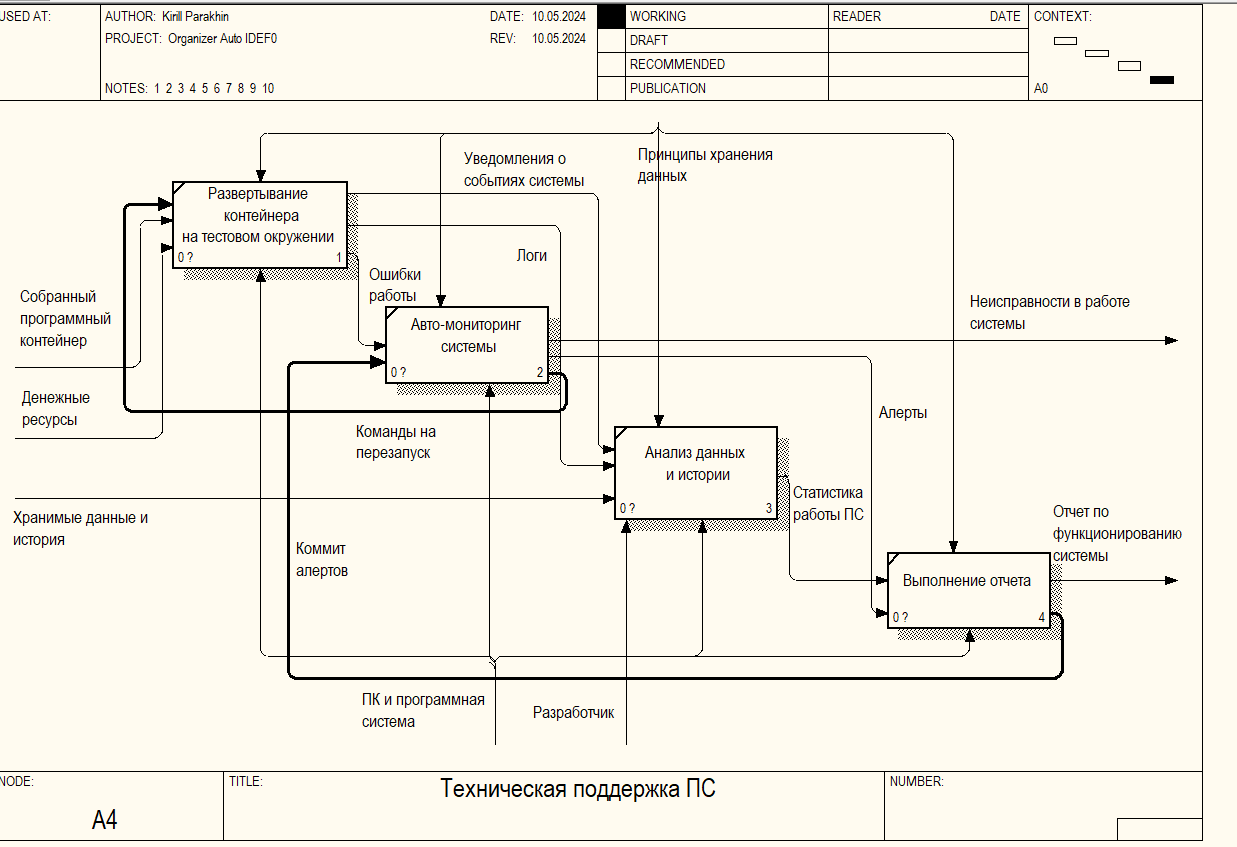


Рисунок 9.3. Модель as-is на третьем уровне декомпозиции

* 1. Жизненный цикл проекта

Жизненный цикл проекта и его фазы представлены ниже на рис. 10.1 – 10.2:

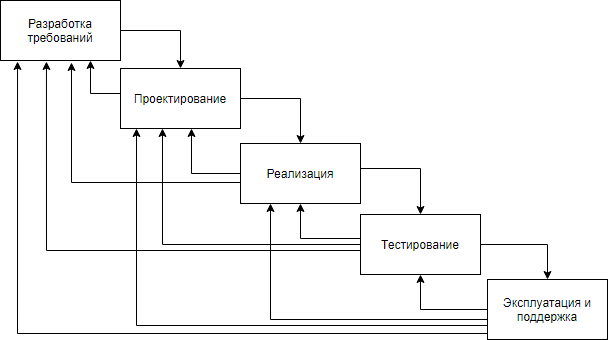


Рисунок 10.1 - Жизненный цикл проекта

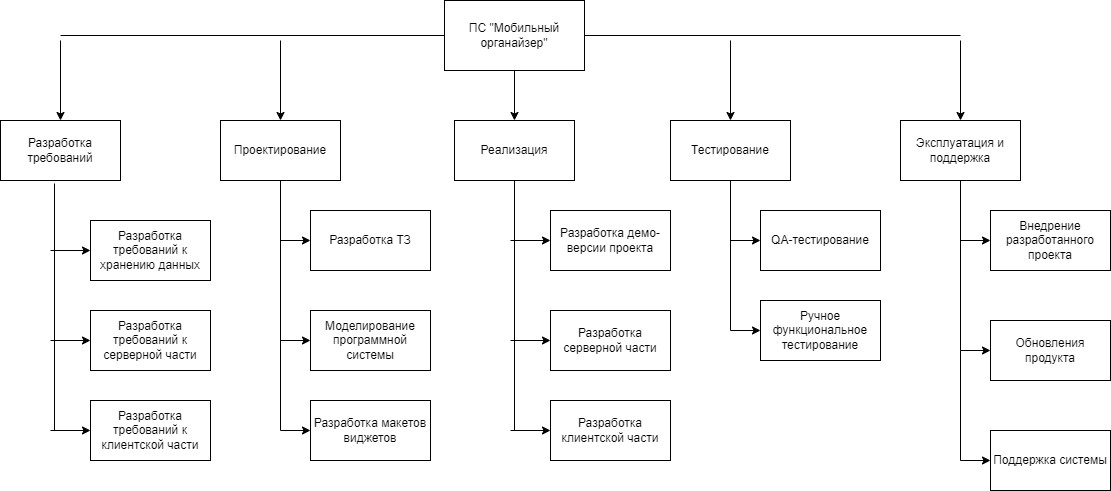


Рисунок 10.2 - Фазы жизненного цикла проекта

## 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).

Ниже в таблице 4 приведен плановые мероприятия по управлению рисками проекта:

Таблица 4 - План управления рисками проекта

|  |  |  |  |
| --- | --- | --- | --- |
| Риск | Ответственный | Дата(до) | Мероприятие |
| Утечка  персональных данных | 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. |

* 1. Подведение выводов о результате ввода системы в использование

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

а) Обеспечение автоматического ведения отчетности

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

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

б) Обеспечение оперативности получения статистики

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

Стало: любой сотрудник может получить статистику своей работы

в) Сокращение операций, выполняемых вручную

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

Стало: добавилась автоматическая система регистрации инцидентов и создания уведомлений в форме alert – сообщений,

г) Обеспечение единого синхронизированного хранилища данных

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

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

е) Обеспечение хорошего уровня анализа и аудита

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

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

Вывод

В результате прохождения производственной практики мною были получены теоретические и практические навыки по проектированию и реализации компонентов программно-информационной системы, а также аналитические навыки по описанию технической документации системы, а также навыки по моделированию бизнес-процессов – как проекта в частности, так и компании в целом.

Кроме того, мною также были изучены основные требования, стандарты и ГОСТы, применяемые сотрудниками при выполнении трудовой деятельности в IT - организациях.

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.