Краевое государственное автономное

профессиональное образовательное учреждение

«Пермский авиационный техникум им. А.Д. Швецова»

Дневник

производственной (Преддипломной) практики

специальность 09.02.07 Информационные системы и программирование

Студент Неволин Евгений Павлович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Группа ИСП-20-4 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Место практики КГАПОУ «Авиатехникум»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Начало практики 15.04.2024 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Окончание практики 10.05.2024 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики от предприятия Ощепков Александр Олегович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики от техникума Аристова Елена Геннадьевна \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2024**

# Содержание

[Введение 3](#_Toc165891077)

[1. Анализ предметной области 4](#_Toc165891078)

[2. Техническое задание 8](#_Toc165891079)

[3. Моделирование предметной области 13](#_Toc165891080)

[3.1 Обзор и обоснование выбора нотаций моделирования 13](#_Toc165891081)

[3.2 Описание разработанных моделей 16](#_Toc165891082)

[4. Реализация разработки 21](#_Toc165891083)

[4.1 Программные средства реализации проекта 21](#_Toc165891084)

[4.2 Структура информационной системы 29](#_Toc165891085)

[4.3 Описание программных модулей в соответствии с требованиями к проекту 38](#_Toc165891086)

[5. Тестирование разработанного продукта 73](#_Toc165891087)

[5.1. Анализ и выбор методов тестирования 73](#_Toc165891088)

[5.2. Результаты тестирования 73](#_Toc165891089)

[6. Экономическое обоснование разработки 73](#_Toc165891090)

[7. Внедрение и установка информационной системы 73](#_Toc165891091)

[8. Сопровождение информационной системы 73](#_Toc165891092)

[9. Настройка безопасности информационной системы 73](#_Toc165891093)

[Заключение 74](#_Toc165891094)

[Приложение А 74](#_Toc165891095)

[Приложение Б 75](#_Toc165891096)

[Приложение В 76](#_Toc165891097)

[Приложение Г 77](#_Toc165891098)

[Приложение Д 79](#_Toc165891099)

[Приложение Е 81](#_Toc165891100)

# Введение

В современном мире информационные технологии становятся неотъемлемой частью жизни организаций, упрощая и оптимизируя процессы в различных сферах деятельности. В контексте организации и проведения мероприятий, эффективное использование информационных систем становится ключевым элементом успешной работы.

Данная область активно развивается, требуя современных решений для управления и сопровождения мероприятий, обеспечения коммуникации и взаимодействия с участниками, а также анализа и управления данными.

В рамках дипломного проекта рассматривается разработка информационной системы для организации и сопровождения мероприятий.

Целью дипломного проекта является разработка информационной системы для организации и сопровождения мероприятий.

Для достижения поставленной цели, необходимо выполнить ряд следующих задач:

* проанализировать предметную область;
* сформулировать техническое задание разрабатываемой системы;
* спроектировать информационную систему;
* выбрать подходящие программные решения для разработки;
* разработать информационную систему;
* проанализировать методы тестирования;
* произвести тестирование разработанного продукта;
* выполнить экономические расчеты для разработки;
* внедрить и сопроводить программный продукт;
* выполнить настройку безопасности информационной системы.

Прежде, чем приступать к моделированию разрабатываемой системы, требуется произвести анализ предметной области.

# Анализ предметной области

Анализ предметной области организации и сопровождения мероприятий позволяет выявить основные потребности, проблемы и вызовы, с которыми сталкиваются организаторы мероприятий, а также определить функциональные требования к информационной системе.

Проблемы организаторов мероприятий:

* сложности в планировании: Организаторы мероприятий часто сталкиваются с проблемами в планировании событий, распределение ресурсов и определение приоритетов;
* коммуникация и координация: Эффективная коммуникация между организаторами и участниками мероприятий играет ключевую роль в успехе события;
* управление участниками: Организаторы должны управлять регистрацией участников, обработкой заявок и общением с участниками.

Для успешной разработки информационной системы необходимо провести сравнительный анализ с аналогичными программными продуктами. Этот анализ позволит выявить основные проблемы и вызовы, с которыми сталкиваются пользователи существующих систем, а также определить преимущества и недостатки конкурентов. Изучение аналогов поможет выявить особенности и уникальные возможности, которые могут быть внедрены в разрабатываемую систему. Такой подход способствует более точному определению требований к системе и формированию стратегии ее разработки.

В ходе поиска систем с схожей предметной областью в сети интернет, были обнаружены два наиболее популярных на сегодняшний день решения задач по организации мероприятий, а именно: Codeforces и HackerRank

Codeforces — это платформа для проведения соревнований по программированию, а также обучения и практики алгоритмическому мышлению.

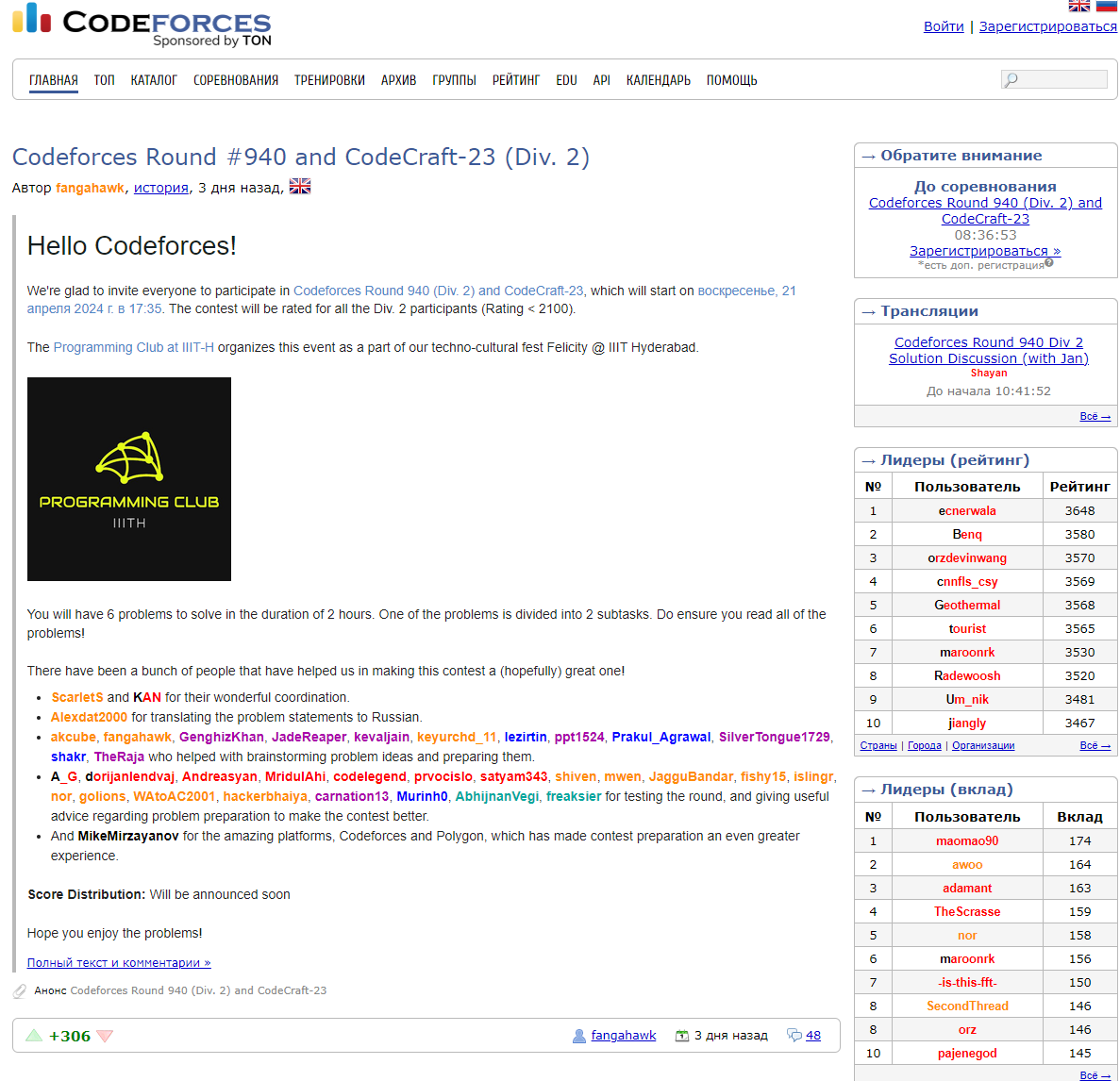


Рисунок 1 - Codeforces.

Преимущества платформы Codeforces:

* Широкий выбор задач: Платформа предлагает огромное количество задач разной сложности, что позволяет участникам выбирать задания согласно своему уровню подготовки.
* Сообщество: Codeforces имеет активное сообщество программистов, где участники могут обсуждать задачи, делиться опытом и советами.
* Регулярные соревнования: Платформа регулярно организует соревнования, что помогает участникам поддерживать мотивацию и развиваться в программировании.

Недостатки платформы Codeforces:

* Высокая сложность задач: Некоторые задачи могут быть сложными для новичков в программировании, что может быть вызовом для начинающих.
* Отсутствие обратной связи: в некоторых случаях участники могут испытывать затруднения с пониманием условий задач или причин неудачи, поскольку на Codeforces часто отсутствует детальная обратная связь к решениям задач.

HackerRank — это онлайн-платформа, которая предоставляет широкий спектр задач по программированию, алгоритмам, структурам данных и другим темам, а также возможности для обучения и практики навыков программирования.

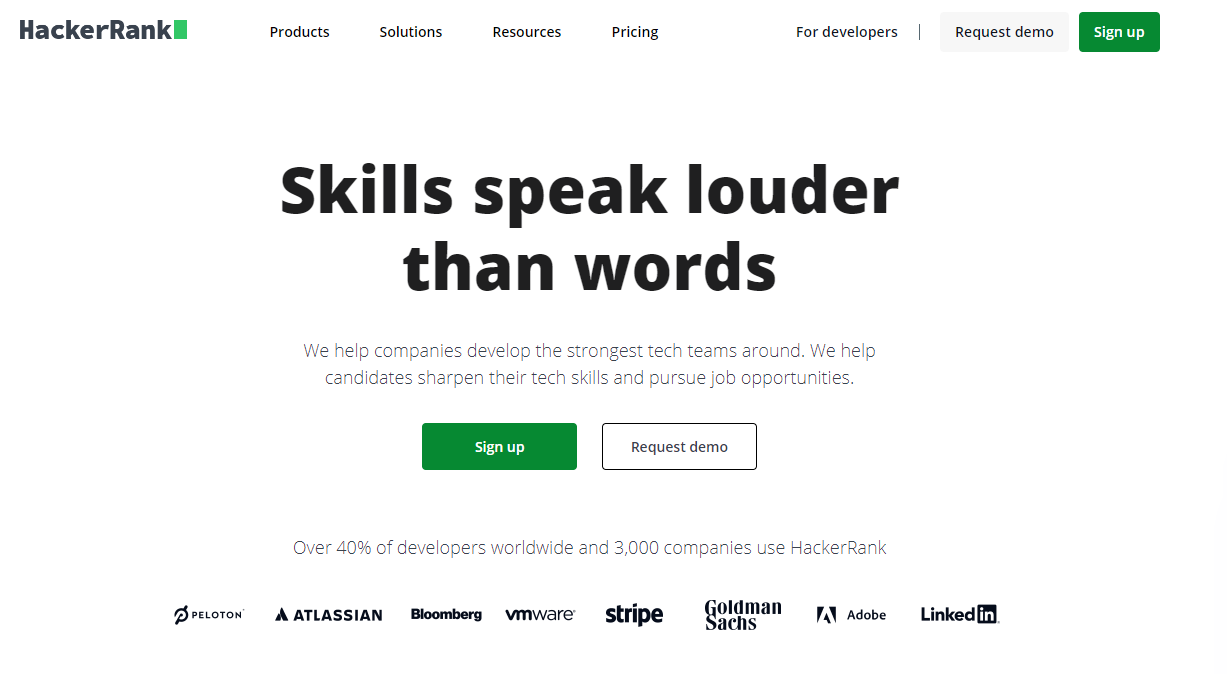


Рисунок 2 - HackerRank

Преимущества платформы HackerRank:

* Широкий выбор задач: HackerRank предлагает большое количество задач по различным темам программирования, что позволяет участникам выбирать задания согласно своему уровню подготовки и интересам.
* Образовательные ресурсы: HackerRank предлагает обширную библиотеку учебных материалов и статей по различным темам программирования.

Недостатки платформы HackerRank:

* Высокая сложность некоторых задач: Некоторые задачи на HackerRank могут быть сложными для новичков в программировании, что может создать трудности для начинающих.
* Не всегда детализированная обратная связь: в некоторых случаях обратная связь к решениям задач на HackerRank может быть недостаточно детализированной или отсутствовать.

Среди общих недостатков данных систем наиболее важным является проблема с обратной связью, что не позволяет участникам получать своевременные ответы на их вопросы и приводит к проблемам во время проведения мероприятий.

В результате анализа предметной области были выявлены основные задачи, которые должна выполнять разрабатываемая система:

* предоставление широкого функционала настройки проводимых мероприятий;
* реализация возможности обратной связи участников и организаторов;
* предоставление организатором удобного инструментария управления участниками мероприятий.

Прежде, чем приступать к моделированию системы, требуется сформулировать техническое задание разрабатываемого программного продукта.

# 2. Техническое задание

Тип продукта – информационная система, включающая в себя мобильное и веб приложения.

1. Основные требования к интерфейсу:

* функциональность (соответствие задачам пользователя): Интерфейс должен обеспечивать выполнение основных задач пользователей без лишних сложностей.
* понятность и логичность: Интерфейс должен быть интуитивно понятным, чтобы пользователи могли легко понять, как использовать различные функции приложения. Логика работы элементов управления должна быть последовательной и предсказуемой.
* обеспечение высокой скорости работы пользователя: Приложение должно быстро реагировать на действия пользователя и обеспечивать быструю загрузку данных и переходов между различными экранами.
* адаптивность: Интерфейс должен корректно отображаться на различных устройствах и в разных ориентациях экрана, обеспечивая при этом одинаковую функциональность и удобство использования.
* наличие призыва к действию там, где это необходимо: Пользователь должен понимать, какие действия можно выполнить на каждом экране приложения, и быть мотивированным к их выполнению;
* названия элементов управления должны четко отражать их функции и быть понятными для пользователей;
* элементы управления должны быть логически сгруппированы и четко разделены на экране, чтобы пользователи могли легко найти нужные функции;

1. Карта мобильного приложения

Карта приложения представлена в приложении А.

Приложение встречает пользователя окном загрузки.

После пользователь указывает номер телефона для авторизации и попадает на страницу ввода кода.

После ввода кода пользователь попадает на главную страницу.

На главной странице предоставляется возможность перехода к диалогам, списку событий и личной информации.

В диалогах пользователь может открыть нужный ему чат и просмотреть сообщения или отправить новое.

В списке событий предоставляется возможность просмотра событий и перейти к одному из них.

Страница личной информации предоставляет возможность просмотра и изменения информации о пользователе.

При переходе на страницу события пользователю отображается информация о нем, в том числе мероприятия и подразделения, к просмотру информации о которых можно перейти.

На странице мероприятия отображается информация о его расписании.

На странице подразделения отображается информация о нем.

1. Карта веб приложения

Карта веб приложения представлена в приложении Б.

1. Функциональные требования для веб сервиса:

Требования к аутентификации:

* возможность зарегистрироваться в системе;
* авторизация пользователей производится по логину и паролю;
* в качестве логина выступает адрес электронной почты, который должен быть уникален для каждого пользователя;
* регистрация организаторов модерируется администратором.

Требования к организаторам событий:

* регистрация организаторов производится с помощью создания организации;
* после создания организации возможно добавление ее участников;
* организаторы имеют возможность создавать события, настраивать информацию о них, добавлять и удалять участников и публиковать новости о проводимых ими событиях.

Требования к событиям:

* организатор имеет возможность выбора наличия подразделений;
* автоматическое формирование новости при создании события;
* возможность отказа от публикации автоматически созданной новости;
* возможность выбора публичности события;
* возможность добавления участников события;
* возможность указания сроков проведения;
* просмотр всех событий в виде списка без авторизации;
* просмотр подробной информации о каждом из событий;
* возможность подачи заявки на участие.

Требования к подразделениям в событиях:

* возможность выбора сроков проведения;
* возможность настройки расписания мероприятий для каждого из подразделений;
* возможность добавления участников;
* возможность указания места проведения на карте;
* при вводе адреса проведения высвечиваются подсказки.

Требования к расписанию событий:

* возможность выбора частоты проведения, одноразовое или повторяющееся;
* возможность выбора дат проведения;
* возможность указания дней недели, в которое проводится мероприятие;
* возможность настройки расписания для всех подразделений сразу;
* возможность указания длительности и места проведения мероприятия.

Требования к новостям:

* просмотр возможен для конкретного события или всех новостей в системе;
* возможно добавление подзаголовков с указанием их названия, текста и изображения.

Требования к администрированию системы:

* возможности модерирования данных администратором;
* возможность блокировки пользователей с указанием срока, через который он будет разблокирован;
* заявки на регистрацию организаторов событий обрабатываются администратором системы;
* возможность удаление из системы организаций, новостей и событий из системы с указанием причины и уведомлением пользователей по электронной почте.

Требования к заявкам на участие в событиях:

* пользователи веб сервиса должны иметь возможность составлять заявки на участие в событии;
* требуется реализовать возможность добавления множества участников в одну заявку;
* если добавляемый пользователь уже существует в системе, его имя должно предлагаться для выбора составителю заявки;
* организатор события должен иметь возможность принять и отклонить заявку;
* возможность отклонения заявки должна быть реализована как для всех участников сразу, так и для каждого по отдельности;
* при обработке заявки должна иметься возможность добавления комментария;
* по итогам обработки заявки ее составителю должно отправляться сообщение на электронную почту.

Наличие интерфейса для передачи данных на мобильные устройства по сети интернет.

Возможность постоянного соединения пользователей для реализации обратной связи

Проверка введенных данных:

* все данные, введенные пользователем, проходят валидацию;
* при ошибке прохождения валидации отображаются ошибки в соответствующих полях ввода информации.

1. Функциональные требования для мобильного приложения:

Аутентификация:

* авторизация производиться по номеру телефона и коду, получаемому в SMS;
* хранение данных пользователей ограничивается их идентификационным токеном.

Предоставление информации:

* пользователь не имеет возможности получения информации, к которой он не имеет доступа;
* расписание отображается как для всех событий, в которых участвует пользователь, так и для каждого по отдельности;
* требуется реализация возможности просмотра краткой информации в виде списка и подробной по каждой отдельной записи;
* вся информация должна быть получена с сервера, работающего с базой данных.

Обратная связь:

* приложение должно реализовывать возможность обратной связи;
* при получении нового сообщения, пользователю должно выводиться уведомление;
* должен иметься список диалогов, в которые пользователь сможет зайти и написать новое сообщение.

1. Все данные приложения хранятся в структурированном виде под управлением реляционной системы управления базами данных (СУБД).

# 3. Моделирование предметной области

## 3.1 Обзор и обоснование выбора нотаций моделирования

* все данные, введенные пользователем, проходят валидацию;
* при ошибке прохождения валидации отображаются ошибки в соответствующих полях ввода информации;
* организатор должен иметь возможность собственноручно добавлять участников события;
* требуется наличие возможности создания новостей, связанных с событием.

Для проектирования разрабатываемых информационных систем используются диаграммы, представляющие структуру программного обеспечения или системы. Они также показывают различные уровни абстракции и реализации. Они используются, чтобы помочь визуализировать различные структуры, составляющие систему, например, базу данных или приложение. Они показывают иерархию компонентов или модулей и то, как они связаны и взаимодействуют между собой. Эти инструменты обеспечивают руководство работы и гарантируют, что все части системы функционируют так, как задумано по отношению ко всем остальным частям [2].

Существует множество нотаций, используемых при описании бизнес-процессов и проектировании информационных систем, такие как:

* IDEF0;
* eEPC;
* UML.

IDEF0 - нотация графического моделирования, используемая для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающих эти функции. Стандарт IDEF0 утвержден в США в 1993 как Федеральный стандарт обработки информации. В России находится в статусе руководящего документа с 2000 года и в настоящее время в качестве стандарта не утвержден. Тем не менее методология IDEF0 является одним из популярных подходов для описания бизнес-процессов. К ее особенностям можно отнести:

* использование контекстной диаграммы;
* поддержка декомпозиции;
* доминирование;
* выделение 4 типов стрелок [3].

Плюсы IDEF:

* блок-схема, с использованием нотации IDEF, всегда «заточена» под лист А4. Удобно распечатывать;
* программ, поддерживающих IDEF, много.

Минусы:

* использовать модели, построенные в IDEF, сложно;
* построенную модель трудно анализировать;
* ограничения по количеству отображаемых в схеме процессов;
* правила описания и чтения бизнес-процессов неудобны;
* программы, поддерживающие IDEF, устарели вместе с ней [4].

ARIS eEPC расшифровывается следующим образом - Extended Event Driven Process Chain – расширенная нотация описания цепочки процесса, управляемого событиями. Нотация разработана специалистами компании IDS Scheer AG, в частности профессором Шелером [5].

Плюсы нотации eEPC:

* логика построения легка и понятна;
* многие ПО позволяют моделировать в eEPC;
* удобно изучать и анализировать;
* можно увидеть события, которые управляют развитием процессов;
* большое количество возможностей для моделирования любого процесса.

Минусы нотации eEPC:

* невозможно определить, как происходит взаимодействие между участниками процесса;
* события нельзя отличить по типу;
* ориентация на сложные и комплексные программные решения [4].

UML - унифицированный язык моделирования был разработан с целью обеспечить единый визуальный язык с богатой семантикой и развернутым синтаксисом для проектирования и внедрения программных систем со сложной структурой и комплексным поведением. Стоит отметить, что UML применяется не только в разработке программ, но и в других сферах, например, в схематизации потоков производственных процессов.

UML напоминает стандарты, используемые в других отраслях, и поддерживает диаграммы нескольких типов. В целом, диаграммы UML описывают границы, структуру и поведение как всей системы, так и отдельных объектов в ее составе.

UML не является языком программирования, однако на базе диаграмм UML можно сгенерировать код на разных языках, и для этого существует ряд специальных инструментов. Зато с объектно-ориентированным анализом и дизайном унифицированный язык моделирования связан напрямую [6].

Преимущества UML:

* упрощает сложности;
* сохраняет возможности открытого общения;
* автоматизирует производство программного обеспечения и процессов;
* помогает решить постоянные проблемы с архитектурой;
* улучшает качество работы;
* сокращает затраты и время выхода на рынок [4].

При выборе нотации для моделирования информационных систем следует учитывать ряд факторов, таких как доступность инструментов, простота использования, поддержка стандартов и специфика проектируемой системы. Исходя из этого, для проектирования системы было решено использовать UML по следующим причинам:

* универсальность и стандартизация: UML является широко используемым и стандартизированным языком моделирования, который поддерживается множеством инструментов и методологий разработки. Это обеспечивает единый подход к моделированию систем и облегчает взаимопонимание между участниками проекта;
* разнообразие диаграмм: UML включает в себя различные типы диаграмм, такие как диаграммы классов, последовательности, состояний и другие, что позволяет описать различные аспекты проектируемой системы на разных уровнях абстракции;
* поддержка объектно-ориентированного анализа и проектирования: UML специально разработан для моделирования объектно-ориентированных систем, что делает его подходящим инструментом для проектирования современных информационных систем.

В целом, UML является удобным и эффективным инструментом для моделирования информационных систем, который обладает широкими возможностями и поддерживается множеством инструментов и ресурсов.

## 3.2 Описание разработанных моделей

Для успешного проектирования системы важно создать основные диаграммы, которые обеспечат максимальное понимание её структуры и функциональности. Создание следующих ключевых диаграмм поможет достичь этой цели:

* ERD (Entity-Relationship Diagram, диаграмма сущностей и связей) поможет визуализировать основные сущности системы и их взаимосвязи;
* диаграмма классов представит архитектуру системы, описывая классы, их атрибуты и методы;
* диаграмма вариантов использования позволит идентифицировать основные сценарии использования системы, выявив её ключевые функциональные возможности;

Дополнительные диаграммы помогут глубже погрузиться в конкретные аспекты системы и проанализировать их более подробно:

* диаграмма последовательности покажет последовательность взаимодействия между различными объектами системы в рамках определенного сценария;
* диаграмма деятельности будет полезна для представления последовательности действий или процессов в системе, включая их условия и ветвления;

После определения основных аспектов проектирования системы, наступило время визуализировать эту информацию с помощью соответствующих диаграмм. Далее приведены подробные описания каждой из них.

ERD, разработанная с использованием нотации "лапки вороны" представлена в приложении A. На диаграмме изображены основные сущности и их атрибуты, а также связи между ними, представленные в виде "лапок вороны". Эта диаграмма помогает визуализировать структуру базы данных и ее взаимосвязи, что облегчает понимание структуры данных и их взаимосвязей для разработчиков и аналитиков данных.

Для проектирования системы используется диаграмма классов UML, которая представляет собой структуру классов в системе и их взаимосвязи. В приложении Б показана диаграмма классов, где каждый класс представлен прямоугольником с указанием его имени, атрибутов и методов. Связи между классами отображаются линиями, обозначающими типы отношений, такие как ассоциация, наследование и агрегация.

В приложении В представлена диаграмма вариантов использования, на которой показаны пользователи системы и различные варианты использования, которые они могут выполнять. Актеры обычно изображаются в виде пиктограмм, а варианты использования - в виде эллипсов с их названиями. Линии связи между актерами и вариантами использования указывают на взаимодействие между ними. Функционал веб сервиса показан на рисунке В.1, а рисунок В.2 отображает варианты использования мобильного приложения.

Диаграмма последовательности, показанная на рисунке , отображает процесс получения информации о событии. На этой диаграмме изображены последовательные шаги выполнения операций и взаимодействия между пользователем, мобильным приложением, сервером и базой данных. Она позволяет визуализировать порядок выполнения операций и передачи сообщений между объектами в процессе работы системы, что помогает понять логику выполнения функциональности и выявить потенциальные проблемы или улучшения.

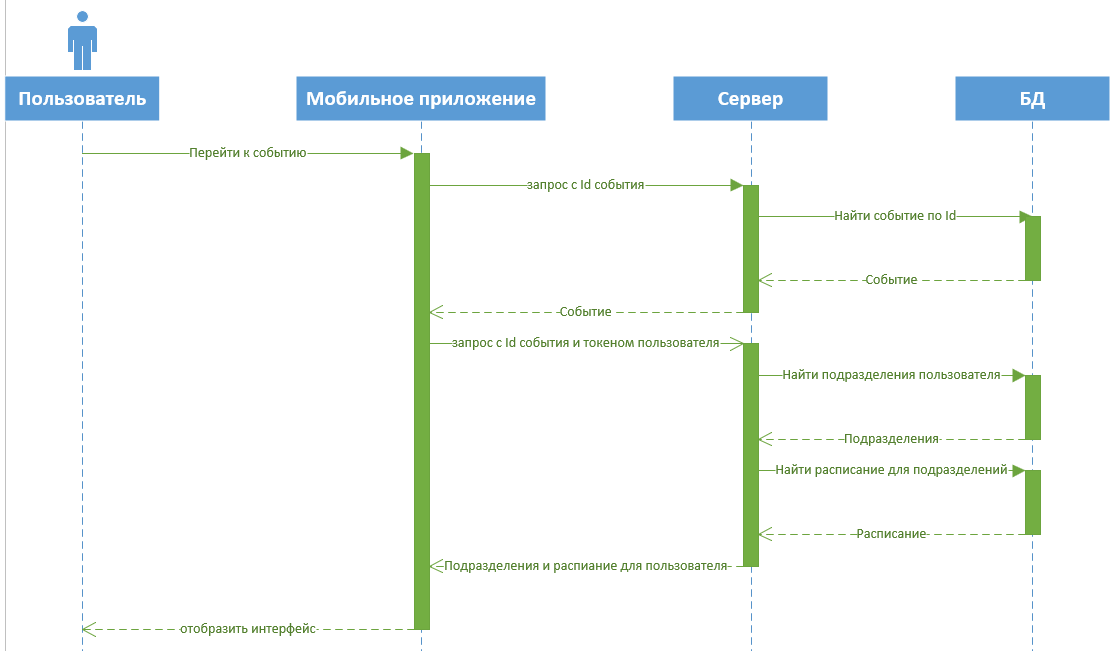


Рисунок 3 - Диаграмма последовательности

Диаграмма деятельности позволяет визуализировать и анализировать процессы и последовательность действий в системе, что помогает понять и оптимизировать её функционирование. Созданная диаграмма деятельности для прецедента регистрации организации показана на рисунке :

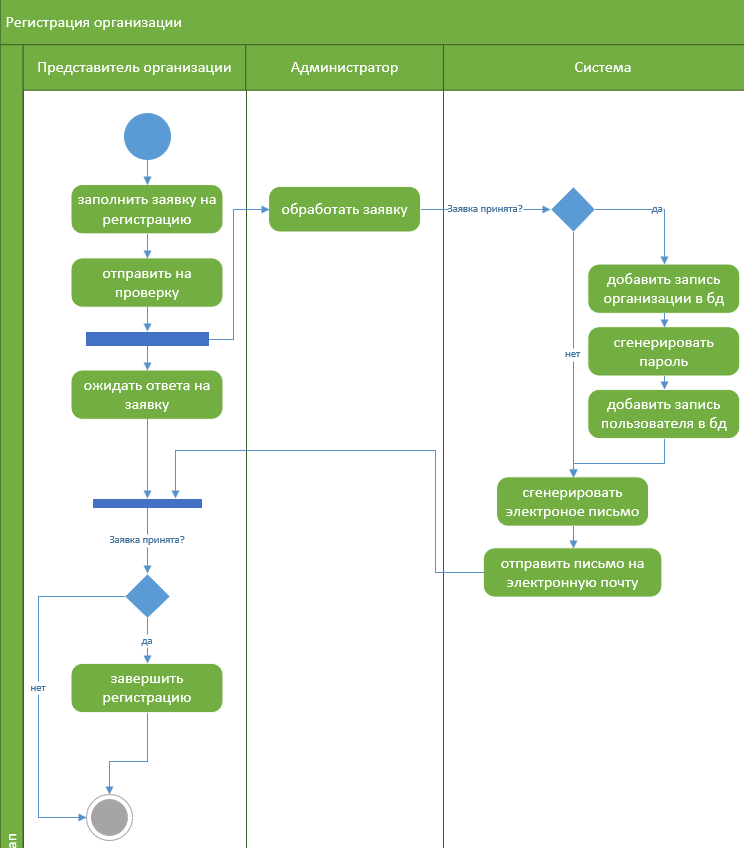


Рисунок 4 - Диаграмма деятельности для регистрации организации

# 4. Реализация разработки

## 4.1 Программные средства реализации проекта

Поскольку информационная система состоит из мобильного приложения и веб сервера, потребовалось выбрать соответствующее программное обеспечение для разработки каждого из двух приложений.

Для разработки веб-сервиса и API необходимо выбрать подходящую платформу разработки HTTP-сервера и соответствующий язык программирования. Среди основных платформ можно выделить:

* ASP.NET Core;
* Node.js;
* Django.

ASP.NET Core является кроссплатформенной, высокопроизводительной средой с открытым исходным кодом для создания современных облачных приложений, подключенных к Интернету.

ASP.NET Core позволяет выполнять следующие задачи:

* создавать веб-приложения и службы, приложения Интернета вещей и серверные части для мобильных приложений;
* использовать избранные средства разработки в Windows, macOS и Linux;
* выполнять развертывания в облаке или локальной среде.

ASP.NET Core предоставляет следующие преимущества:

* единое решение для создания пользовательского веб-интерфейса и веб-API;
* Blazor позволяет использовать в браузере язык C# вместе с JavaScript. совместное использование серверной и клиентской логик приложений, написанных с помощью .NET;
* возможность разработки и запуска в ОС Windows, macOS и Linux;
* открытый исходный код и ориентация на сообщество;
* интеграция современных клиентских платформ и рабочих процессов разработки;
* облачная система конфигурации на основе среды [11].

Node.js — это платформа с открытым исходным кодом для работы с языком JavaScript, построенная на движке Chrome V8. Она позволяет писать серверный код для веб-приложений и динамических веб-страниц, а также программ командной строки. В основе платформы — событийно-управляемая модель с неблокирующими операциями ввода-вывода, что делает ее эффективной и легкой.

Преимущества:

* высокая скорость. JavaScript-код, который выполняется в среде Node.js, может быть в несколько раз быстрее, чем написанный на языках вроде Ruby или Python;
* универсальность и гибкость. В Node.js выполняется код, который написан на JavaScript. Это означает, что frontend-разработчики, которые уже используют JavaScript в браузере, могут писать и клиентский, и серверный код на привычном языке программирования, не изучая инструмент с нуля;
* большое количество модулей и библиотек. Экосистема Node.js стремительно развивается благодаря менеджеру пакетов NPM. Он содержит более 500 000 модулей и библиотек open-source, которые находятся в свободном доступе. Также постоянно появляются новые;
* работа на движке Chrome V8. Node.js работает на JavaScript-движке V8 от Google. V8 — движок JavaScript с открытым исходным кодом, распространяемый по лицензии BSD. Он применяется в браузерах на основе Chromium [12].

Django — это специальный инструмент, который помогает разработчикам создавать веб-сайты и приложения. Он называется фреймворком, потому что предоставляет готовые инструменты и правила для работы с веб-сайтами.

Данный фреймворк нужен для того, чтобы упростить процесс разработки веб-сайта. Вместо того, чтобы создавать каждую часть сайта с нуля, разработчики могут использовать его, чтобы получить уже готовые компоненты и функциональность.

Django имеет много полезных функций, которые помогают создавать веб-сайты:

* позволяет создавать красивые и интерактивные страницы, которые пользователи могут видеть в своих браузерах;
* облегчает работу с базой данных, где хранится информация о веб-сайте, такая как список товаров или информация о пользователях;
* помогает обрабатывать запросы от пользователей и отправлять им нужную информацию;
* имеет мощный административный интерфейс, который позволяет администраторам управлять содержимым веб-сайта без необходимости знать программирование [13].

В ходе сравнения платформ создания веб сервера, выбор пал на ASP.NET Core из-за его простоты, удобства, широкого функционала и быстрой работы.

Разработка на ASP.NET Core осуществляется на языке программирования C# с использованием интегрированной среды разработки Visual Studio.

Для разработки мобильного приложения необходимо выбрать язык программирования и IDE. Среди наиболее популярных языков программирования для создания мобильных приложений можно выделить следующие:

* Java — это мощный язык программирования и платформа вычислений, которые играют ключевую роль в разработке сервисов и приложений. Преимущества Java включают межплатформенную совместимость, объектно-ориентированное программирование, обширные библиотеки и большую экосистему [14];
* C# — современный объектно-ориентированный язык программирования, который используется для создания различных приложений в среде .NET. Для разработки мобильных приложений на C# часто используется платформа Xamarin, обладающая единым стеком технологий для разработки на всех платформах и хорошей совместимостью с оборудованием [15].;
* C++ — это мощный язык программирования, который предоставляет широкие возможности, включая объектно-ориентированное программирование и шаблоны. Преимущества использования C++ для разработки мобильных приложений включают простую установку, эффективную среду сборки и унифицированный процесс отладки [16].

В результате сравнения выбор был сделан в пользу Java из-за ее высокой адаптированности под платформу Android, которую рекомендует и поддерживает Google. Для разработки приложений на Java используется IDE Android Studio, также разработанная Google.

Для хранения информации в системе требовалась база данных, для работы с которой требуется система управления базами данных (СУБД) и инструмент визуального проектирования БД.

Среди наиболее популярных СУБД можно выделить:

* MySql;
* MsSql;
* PostgreSql.

MySQL — свободная реляционная система управления базами данных (СУБД). Под словом «свободная» подразумевается ее бесплатность, под «реляционная» – работа с базами данных, основанных на двумерных таблицах. Система выпущена в 1995 году, её разработка активно продолжается.

У MySQL есть ряд преимуществ:

* высокая скорость работы;
* поддержка практически всех CMS;
* бесплатная лицензия;
* надежная и простая система безопасности;
* поддержка нескольких типов таблиц: MyISAM, InnoDB;
* плагины, позволяющие упростить и настроить работу под себя;
* в одной таблице может содержаться несколько миллионов записей;
* MySQL универсальна и применяется при разработке веб-сайтов, веб-приложений и корпоративных баз данных начального уровня [8].

Microsoft SQL Server (MSSQL) – это система управления реляционными базами данных, используемая для хранения и извлечения данных из других программных приложений.

Microsoft разработала это программное обеспечение для управления информацией на нескольких компьютерах в одной сети. Используя язык программирования SQL, SQL Server может выполнять аналитику и обработку транзакций, а также работу с информацией.

Обработка транзакций отвечает за определение ошибок и целостность данных. Таким образом, благодаря ей выполнение процесса либо успешно завершится, либо не произойдёт и вовсе.

Преимущества SQL Server:

* процесс установки: установка Microsoft SQL Server проще, чем установка любой другой СУБД, поскольку для неё не требуется специальный набор инструментов. Кроме того, SQL Server обновляется автоматически.
* хранилище: вы можете управлять данными между устройствами, что экономит время и существенно упрощает работу с БД;
* безопасность: MSSQL обеспечивает безопасность и целостность данных. Структура таблицы со связанными функциями гарантирует, что даже случайное повреждение данных почти невозможно;
* восстановление данных: SQL Server имеет функции резервного копирования, упрощающие восстановление данных. Это может свести к минимуму риск потери данных, независимо от того, что происходит с вашим сервером [9].

PostgreSQL — это объектно-реляционная система управления базами данных (ORDBMS), наиболее развитая из открытых СУБД в мире. Имеет открытый исходный код и является альтернативой коммерческим базам данных.

Ранние версии системы были основаны на старой программе POSTGRES University, созданной университетом Беркли: так появилось название PostgreSQL. Существуют сокращения PSQL и PgSQL — они тоже обозначают PostgreSQL.

Преимущества PostgreSQL:

* PostgreSQL — объектно-реляционная СУБД. Это значит, что она поддерживает и объектный, и реляционный подход;
* поддержка множества типов данных;
* работа с большими объемами. В большинстве СУБД, рассчитанных на средние и небольшие проекты, есть ограничения по объему базы и количеству записей в ней. В PostgreSQL ограничений нет;
* поддержка сложных запросов. PostgreSQL работает со сложными, составными запросами;
* написание функций на нескольких языках. В PostgreSQL можно писать собственные функции — пользовательские блоки кода, которые выполняют те или иные действия;
* модификация SQL, которая используется в PostgreSQL, называется PL/pgSQL. Это процедурное расширение, которое поддерживает сложные вычисления и дополняет «классический» SQL новыми возможностями;
* возможность расширения. Разработчик может написать для СУБД собственные типы и их преобразования, операции и функции, ограничения и индексы, собственный процедурный язык для запросов;
* высокая мощность и широкая функциональность. PostgreSQL — возможно, единственная бесплатная СУБД с открытым исходным кодом, которая рассчитана на работу с объемными и сложными проектами [10].

В результате сравнения выбор пал на MSSQL по причинам простоты установки, эффективности управления данными, быстроты работы и безопасности.

В роли интегрированной среды разработки и управления базой данных MSSQL была выбрана SQL Server Management Studio, разработанная Microsoft для работы с их СУБД.

В результате сравнения существующего программного обеспечения, использующегося для разработки программных решений, был составлен список программ, требующихся для создания информационной системы. В него входят:

* Visual Studio - интегрированная среда разработки (IDE) от Microsoft, обеспечивающая широкие возможности для создания приложений на платформе Windows, включая поддержку различных языков программирования и инструментов разработки;
* Microsoft SQL Server - сервер баз данных, предоставляющий мощные средства хранения и обработки данных для информационной системы;
* Microsoft SQL Server Management Studio - инструмент для администрирования и управления базами данных MSSQL, облегчающий работу с данными и обеспечивающий мониторинг сервера;
* Android Studio - среда разработки для создания приложений под операционную систему Android, позволяющая разрабатывать высококачественные мобильные приложения с использованием языка программирования Java и других технологий.

Этот набор программного обеспечения обеспечивает все необходимые инструменты для успешной разработки информационной системы, включая создание и администрирование баз данных, разработку прикладного программного обеспечения и создание мобильных приложений.

## 4.2 Структура информационной системы

Перед тем, как приступить к разработке информационной системы, необходимо провести тщательный анализ ее структуры и функциональности. Это важный этап, на котором определяются ключевые аспекты проекта и выделяются основные логические модули, отвечающие за конкретные задачи и функции.

Модуль — это логически завершенный блок программного кода, который выполняет определенную функцию или решает конкретную задачу. Для обеспечения эффективного взаимодействия между модулями применяются специальные методы и механизмы, которые обеспечивают стабильность работы системы и не допускают изменения параметров и функциональности без необходимости. [[РЕСУРС](https://gb.ru/blog/modulnoe-programmirovanie/)].

Разрабатываемая информационная система, состоит из двух основных приложений: мобильного и веб-сервиса.

Каждое из этих приложений включает в себя ряд модулей, которые выполняют определенные функции и обеспечивают удобство использования для конечных пользователей.

Веб сервис выполняет роль приложения для обработки информации, а именно, ее добавления, изменения и удаления. Поэтому наиболее объемный модуль веб сервиса – работа с информацией о событиях. Его интерфейс можно увидеть на рисунке :

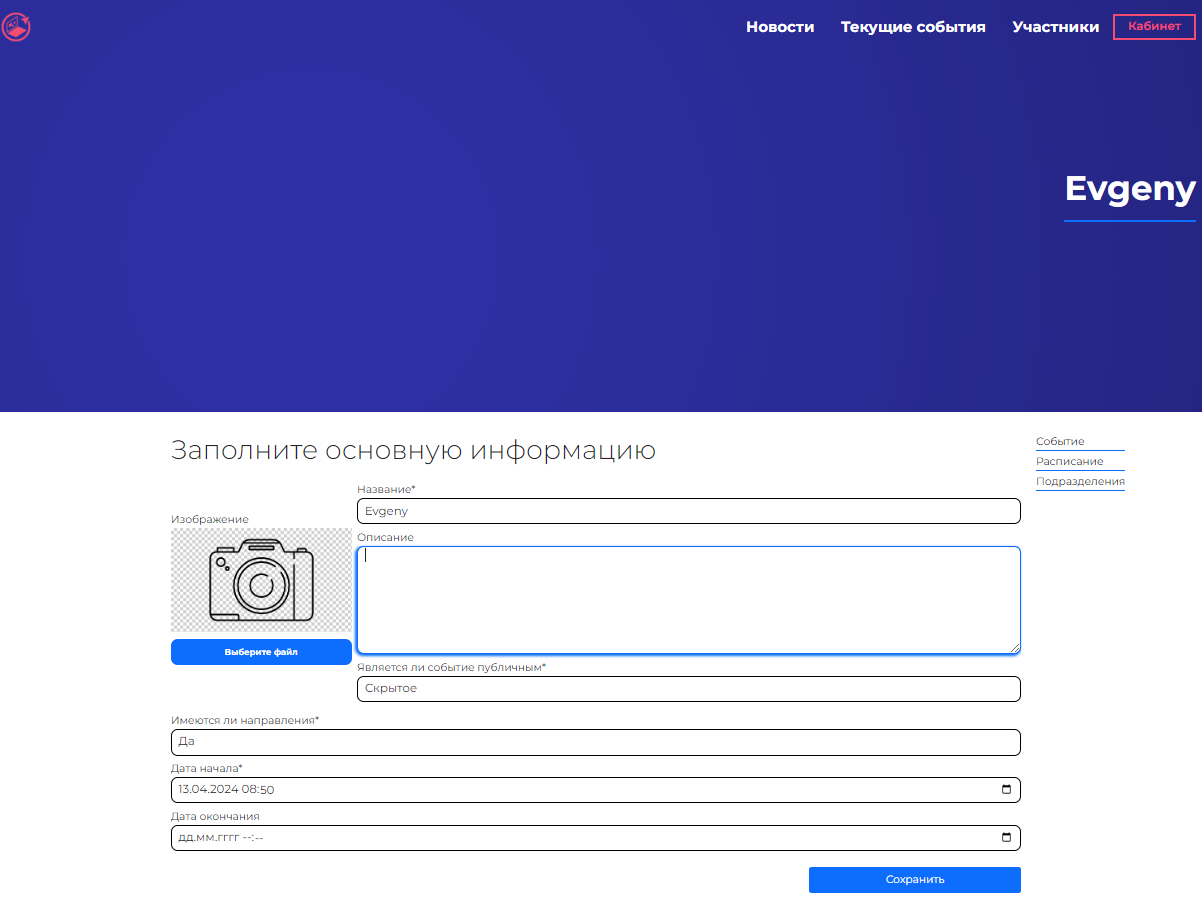


Рисунок 5 - Модуль работы с информацией о событиях

Модуль предоставляет организаторам инструментарий для управления информацией. Он позволяет создавать новые события, редактировать их параметры, управлять расписанием мероприятий, а также настраивать информацию о подразделениях и управлять списком участников. Благодаря этому модулю организаторы имеют полный контроль над организацией и проведением событий, что существенно упрощает процесс управления и обеспечивает эффективную работу системы.

Второй модуль веб-сервиса, посвященный работе с заявками на участие в событиях, предоставляет пользователям возможность легко создавать, заполнять и отправлять их. Организаторы событий, в свою очередь, получают инструменты для эффективной модерации списка участников. Интерфейс этого модуля, представлен на рисунке :

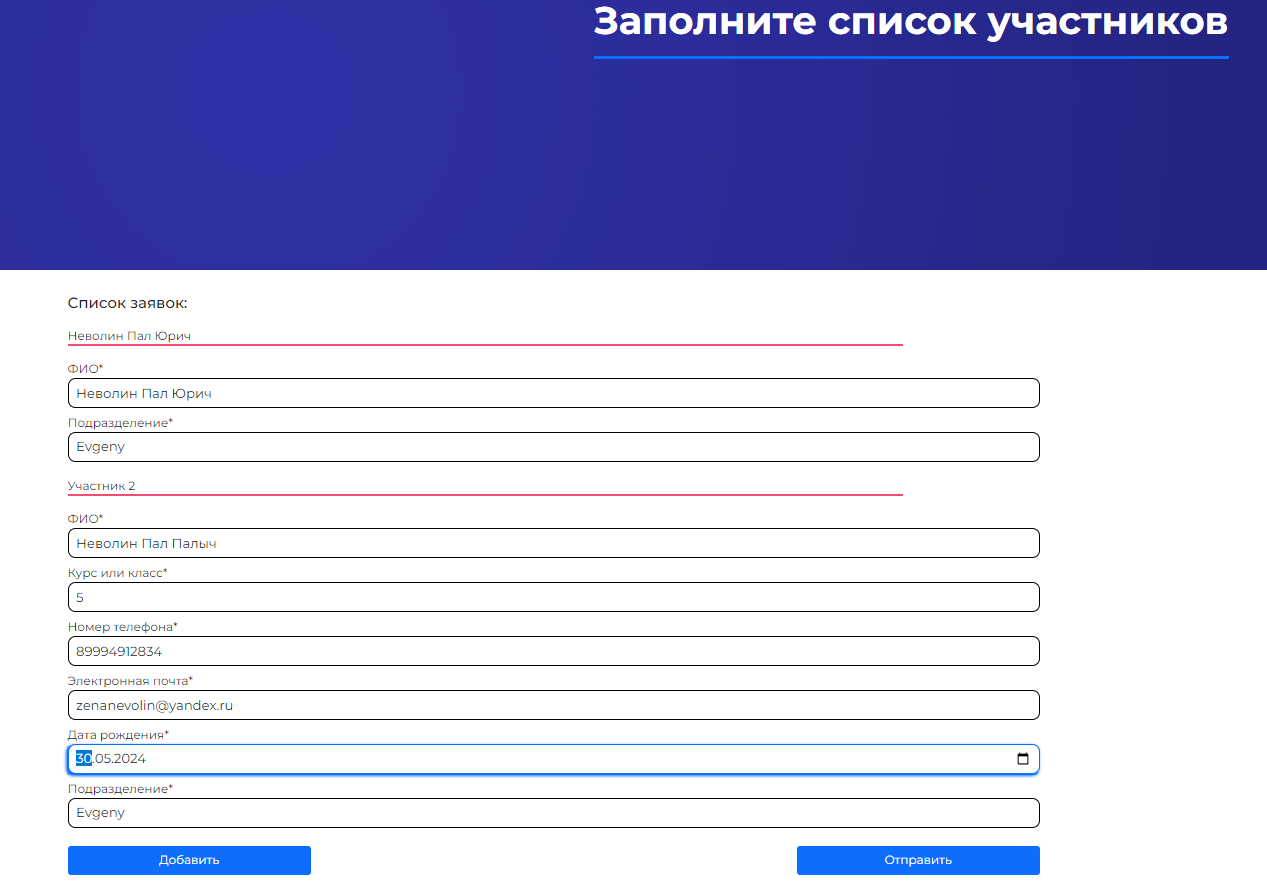


Рисунок 6 - Модуль работы с заявками на участие в событиях

После того, как пользователь веб-сервиса заполняет список участников, организаторы, связанные с событием, получают возможность принимать или отклонять заявки. Они могут принимать решения как по всем участникам сразу, так и по каждому индивидуально. Этот процесс обеспечивает прозрачность и контроль над списком участников, что важно для эффективной организации и проведения событий.

Третий модуль веб-сервиса - администрирование системы. Модуль доступен исключительно администраторам и предоставляет широкий функционал для эффективного управления сервисом. Интерфейс модуля администрирования, изображенный на рисунке :

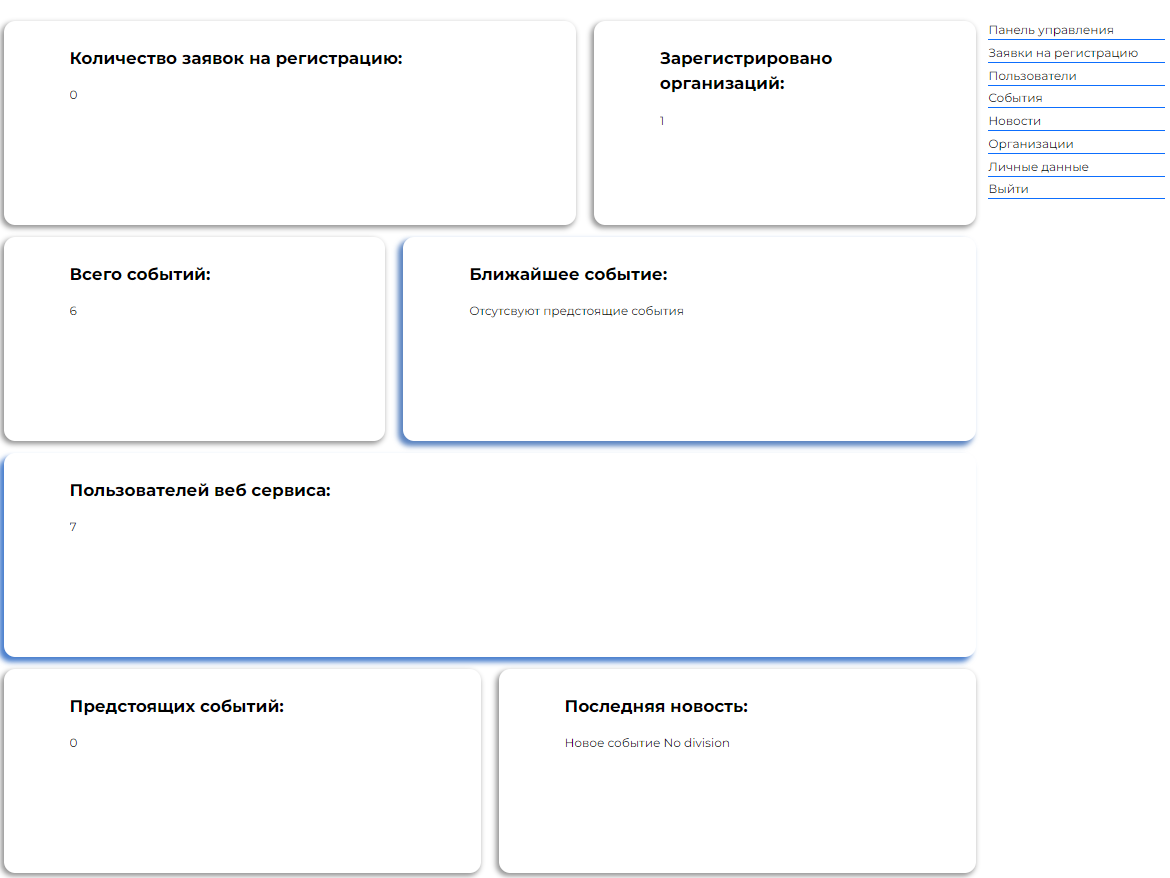


Рисунок 7 - Модуль администрирования

Основные функции этого модуля включают в себя возможность блокировки пользователей, что является важным механизмом защиты от несанкционированного доступа и поддержания безопасности системы. Модерация заявок на регистрацию обеспечивает контроль за доступом пользователей к функциям сервиса и позволяет избегать публикации нежелательного контента.

Кроме того, модуль администрирования предоставляет возможность удаления организаций, событий и новостей, что позволяет администраторам эффективно управлять контентом и поддерживать актуальность информации в системе. Эти функции играют важную роль в обеспечении качественного функционирования веб-сервиса и удовлетворении потребностей пользователей.

Четвертый модуль веб-сервиса – аутентификация – обеспечивает защиту системы от несанкционированного доступа и управление правами пользователей. Интерфейс этого модуля представлен на рисунке :

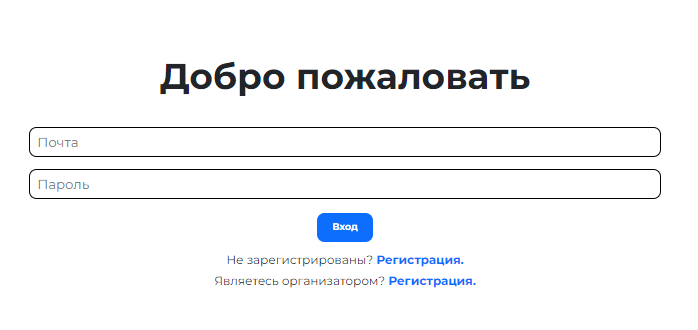


Рисунок 8 - Модуль аутентификации веб-сервиса

Модуль включает в себя следующие функции:

* авторизация по логину и паролю: позволяет пользователям войти в систему, проверяя их учётные данные;
* регистрация пользователей: обеспечивает возможность регистрации новых пользователей веб-сервиса, сохраняя связанную с ними информацию;
* регистрация организаторов: предоставляет пользователям возможность отправить заявку на добавление их организации в систему.

Этот модуль играет ключевую роль в обеспечении безопасности и функциональности веб-сервиса, обеспечивая контроль доступа и удобство использования.

Мобильное приложении, в свою очередь, имеет четыре основных модуля, каждый из которых направлен на обеспечение определенного функционала. Наиболее крупным и важным из них является модуль просмотра информации.

Модуль просиотра информации предоставляет пользователям удобный доступ к списку событий, мероприятий и подразделений, в которых они участвуют. Он обеспечивает возможность ознакомления с актуальными сведениями и расписанием событий, значительно улучшая пользовательский опыт. Внешний вид модуля отображения информации на рисунке :

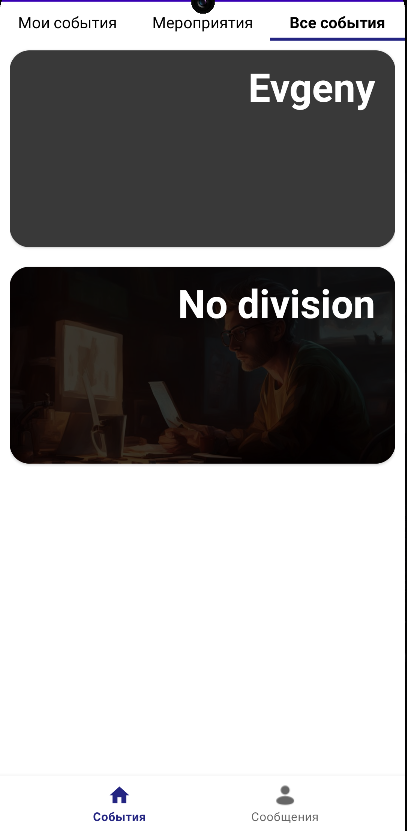


Рисунок 9 - Модуль отображения информации

Модуль работает на основе HTTP-запросов к удалённому серверу, что позволяет получать актуальные данные о событиях и мероприятиях. Пользователи могут легко просматривать информацию о проводимых событиях на платформе, а также о своих подразделениях и расписании предстоящих мероприятий.

Второй модуль мобильного приложения – авторизация пользователей по номеру телефона и присылаемому коду. Интерфейс модуля показан на рисунке :

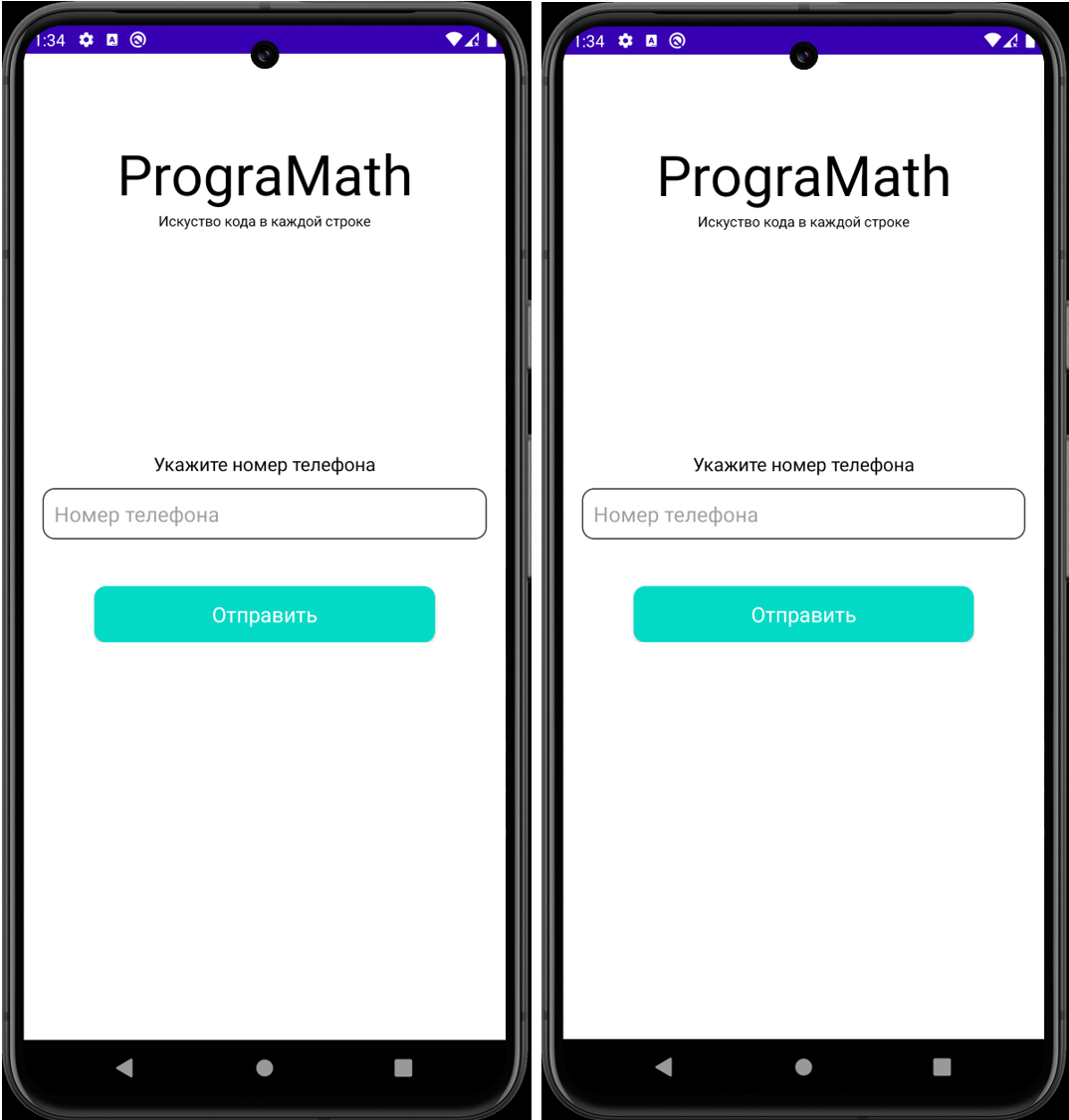


Рисунок 10 - Модуль авторизации

Модуль обеспечивает безопасный доступ пользователей к системе и контролирует их права доступа. Этот модуль необходим для защиты от несанкционированного доступа и обеспечивает разграничение прав пользователей.

Модуль авторизации осуществляет аутентификацию пользователей посредством HTTP запроса. Система не предусматривает самостоятельной регистрации пользователей в мобильном приложении.

Третий модуль – обратная связь с организаторами событий показан на рисунке :

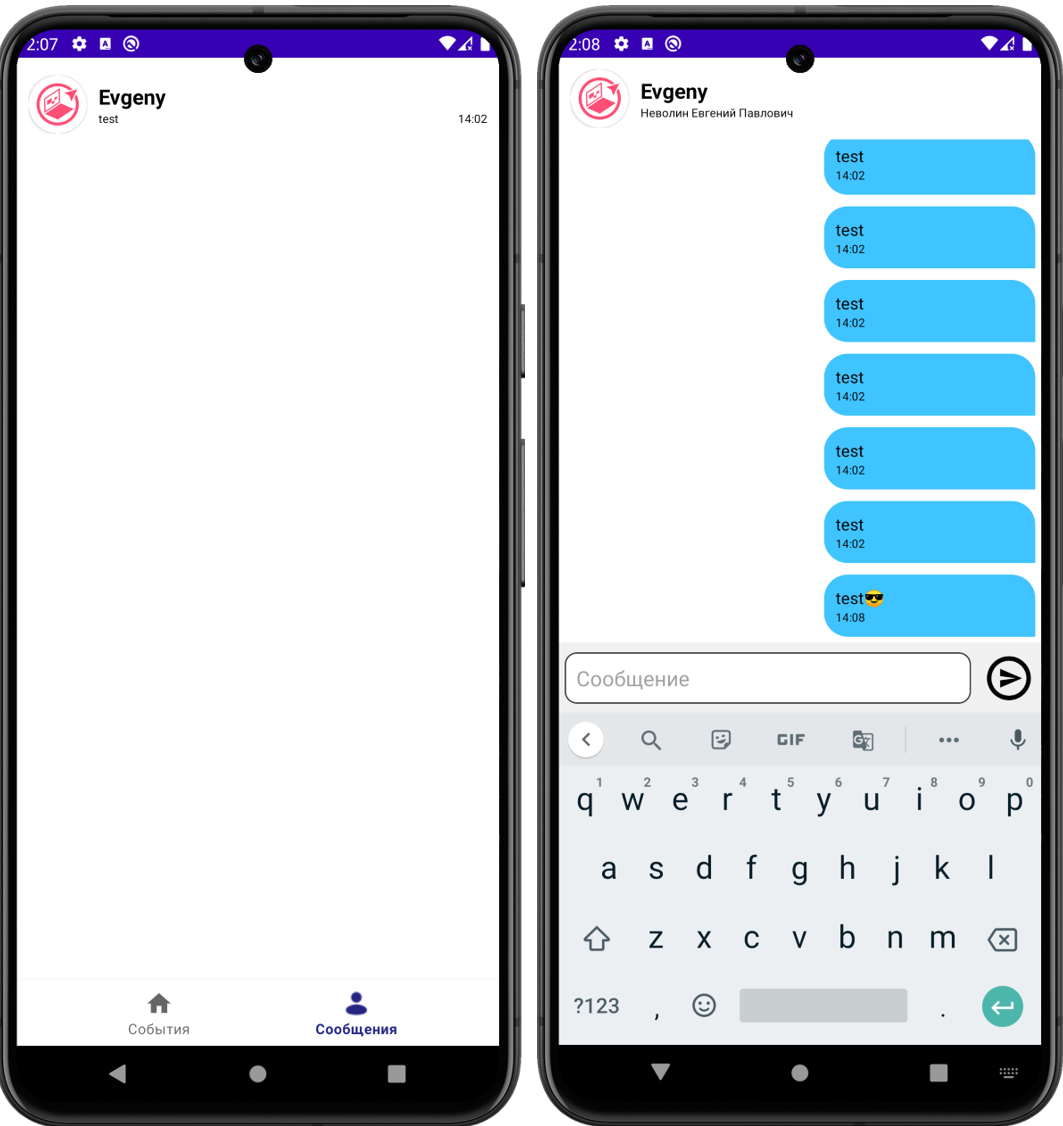


Рисунок 11 – Модуль обратной связи

модуль позволяет пользователю задать вопрос организатору события посредством отправки сообщения в чате. Обратная связь расширит интерактивность приложения и поможет участникам событий быстрее получить ответ на их вопрос. Общение происходит с использованием удаленного сервера. Сообщения отправляются адресату сразу после получения и сохраняются в базу данных.

Четвертый модуль мобильного приложения – редактирование информации о пользователе дает возможность отредактировать личную информацию, изменить фотографию, имя, дату рождения, курс и номер телефона. Его интерфейс показан на рисунке :

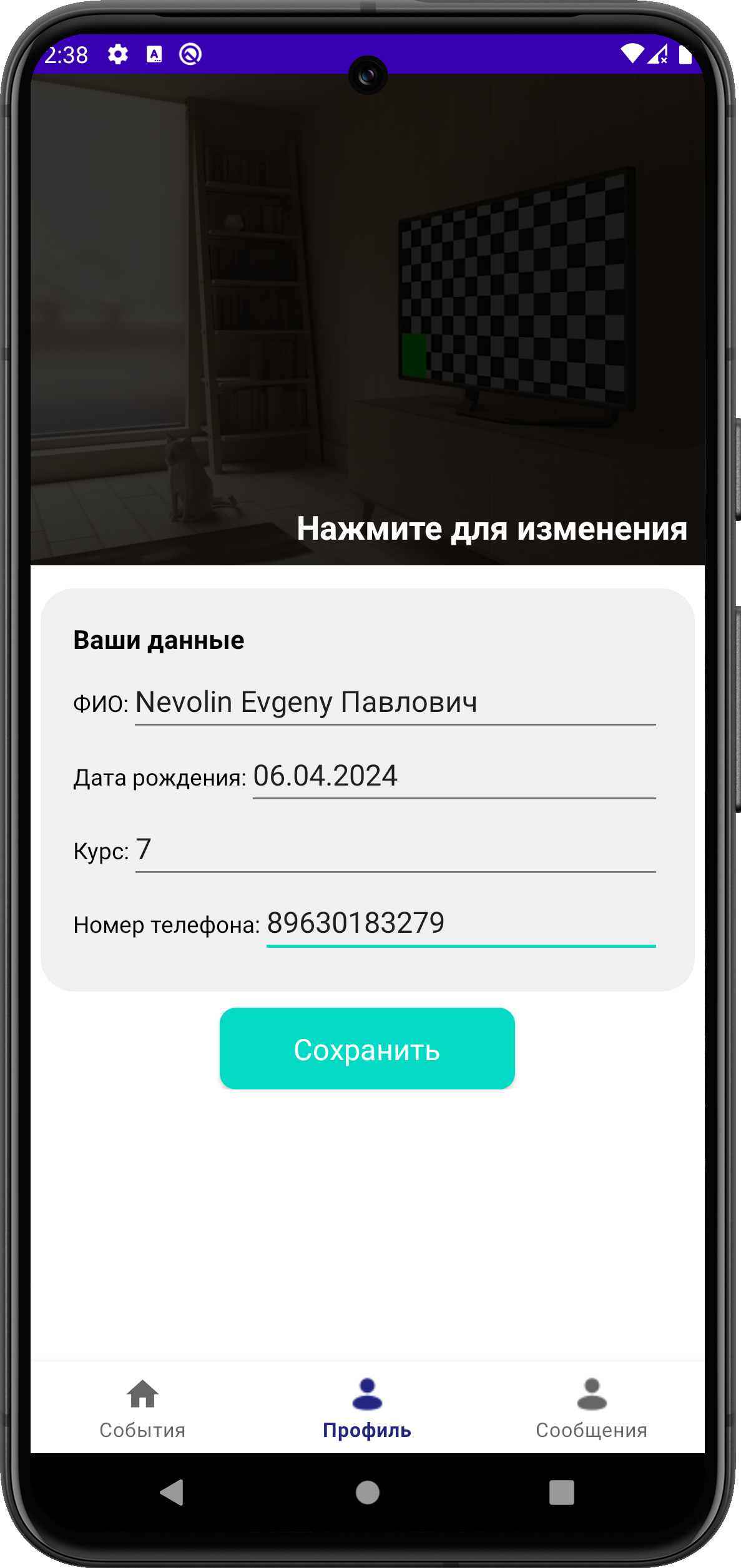


Рисунок 12 - Модуль изменения личной информации

Поскольку доступ к мобильному приложению, помимо целевой роли имеют пользователи еще двух ролей, а именно, организаторы, пользователи веб сервиса, для которых мобильное приложение не является основным инструментом в системе, данным модуль им недоступен.

## 4.3 Описание программных модулей в соответствии с требованиями

Для работы с базой информацией в разработанной системе используется фреймворк EntityFramework — это современное средство сопоставления отношений объектов, которое позволяет создавать чистый, переносимый и высокоуровневый уровень доступа к данным с помощью .NET (C#) в различных базах данных [[source](https://learn.microsoft.com/ru-ru/ef/)].

При использовании этого фреймворка требуется создание контекста, код которого показан на рисунке :



Рисунок 13 - Код контекста EntityFramework

В классе контекста представлены свойства типа DbSet, являющиеся сопоставлениями с таблицами в базе данных и хранящие в себе список записей каждой из таблиц. Помимо свойств, в контексте представлен метод OnConfiguration, выполняющий настройку фреймворка, а именно, добавление логгирования.

Поскольку веб-сервис создан с помощью фреймворка ASP NET Core -процесс работы состоял из разработки методов – обработчиков HTTP запросов, отправляемых пользователем.

Каждый метод, являющийся обработчиком, срабатывает если запрос имеет нужный для него тип и входные данные, которые указываются в виде его параметров.

Модуль работы с информацией включает в себя четыре метода, позволяющих реализовать полный функционал обработки данных в системе:

* Create – создание;
* Edit – изменение;
* Details – просмотр подробной информации;
* Index – список записей из базы данных с краткой выдержкой;
* Delete – удаление.

Код метода Create показан на рисунке :

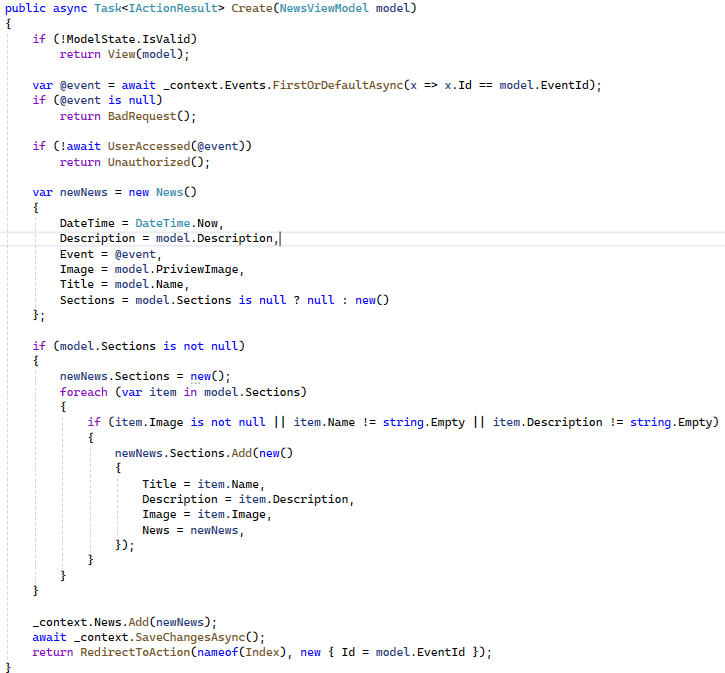


Рисунок 14 - Код метода Create

Метод Create реализует обработчик post запроса и принимает в качестве входных данных экземпляр класса NewsViewModel, валидность которого проверяется в первой строке посредством встроенных средств ASP. В случае ее непрохождения, пользователю возвращается страница заполнения данных. Свойства класса NewsViewModel показаны на рисунке :

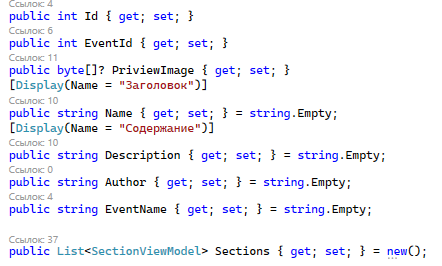


Рисунок 15 - Свойства класса NewsViewModel

После проверки валидности производится поиск события, к которому относится добавляемая новость в базе данных. В случае, если оно не найдено, пользователю в ответе возвращается ошибка, в противном же случае, создается объект новости, свойствам которого присваиваются значения соответствующих свойств входного параметра.

Если в входном параметре присутствуют секции, производится прохождение по им всем внутри цикла.

При прохождении по списку секций проверяется наличие в них каких-либо данных, поскольку ни одно из полей не является обязательным, но требуется наличие информации в хотя бы одном из них.

В случае, если какая-либо информации в секции имеется, ее копия с такими же данными добавляется в список секций добавляемой новости. После заполнения всей информации в новой новости, производится ее добавление в DBSet новостей и сохранение изменений. Последний этап выполнения метода – перенаправление пользователя на страницу Index.

Помимо добавления информации, система должна позволять ее редактировать и эту задачу выполняет метод Edit, отображенный на рисунке :

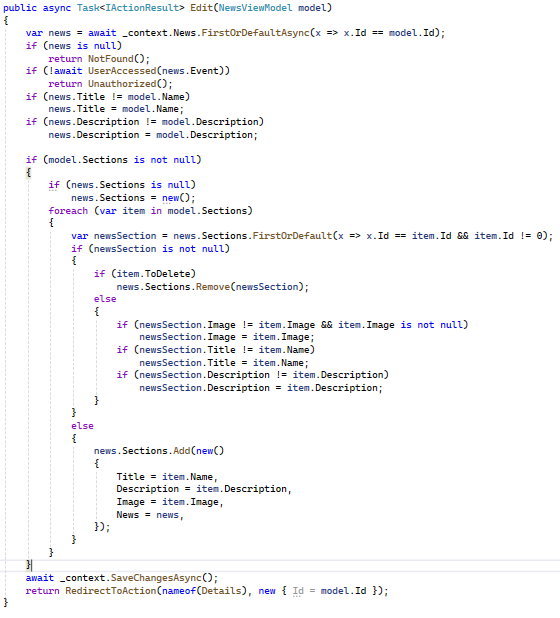


Рисунок 16 – Код метода Edit

Метод реализует обработчик post запроса и принимает в качестве входных данных экземпляр класса NewsViewModel. Изначально выполняется поиск оригинала новости по идентификатору в базе данных. Следующими этапами проверка наличия записи и доступа к ее редактированию для пользователя. Если проверки не пройдены – в ответе возвращается ошибка. После подтверждения прав редактирования сравниваются значения полей оригинальной новости и входного параметра для изменений. Для изменения значений секций производится проверка на их наличие и прохождение в цикле. Для каждой секции производится поиск ее в базе данных. Если запись имеется - производится проверка на надобность ее удаления и изменение значений свойств. В случае же, если секции нет в записи – создается новая и устанавливаются значения ее свойств.

Последним этапом изменения новости является сохранение в базу данных, после чего производится перенаправление пользователя на страницу Details.

Метод Details выполняет задачу обработки get запроса пользователя и возврат ответа в виде страницы. Его код показан на рисунке :

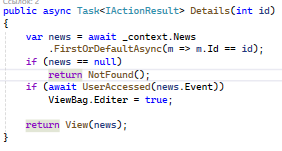


Рисунок 17 – Код метода Details

В виде параметра принимается идентификатор, по которому производится поиск новости в базе данных. Если она не была найдена – в ответе возвращается соответствующий статус. После нахождения новости, производится проверка доступа пользователя к редактированию, для чего вызывается метод UserAccessed, по прохождению которого переменной Editer присваивается значение true. В конце выполнения метода отправляется страница, в которую отправляется переменная – новость.

Помимо представления полной информации об одной записи БД, модуль предоставляет возможность просмотра в виде списка, что выполняет метод Index, код которого показан на рисунке :

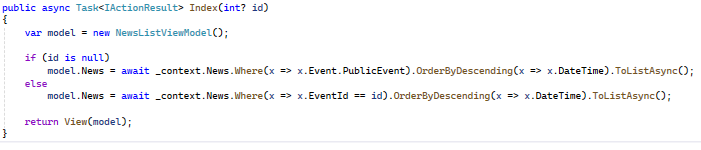


Рисунок 18- Код метода Index

Входным параметром является идентификатор события, который не обязателен для вызова. Работа метода начинается с создания модели – объекта класса NewsListViewModel. Если значение параметра отсутствует – модели приравнивается список всех новостей, отсортированный по датам создания. В противном случае, значением модели устанавливаются новости, найденных по идентификатору события. В конце выполнения метода пользователю возвращается страница списка с указанием модели.

Последний метод модуля Delete, требующийся для удаления информации. Его код показан на рисунке :

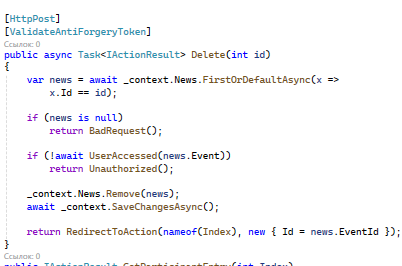


Рисунок 19 - Код метода Delete

Он реализует обработчик Post запроса. Входными данными метода является идентификатор удаляемой новости. После вызова метода производится поиск записи в базе данных. Результат поиска проверяется на наличие значения, в случае отсутствия которого возвращается ошибка.

Помимо проверки наличия записи, проверяется доступ пользователя к удалению новости. В случае отсутствия права на такое действие, возвращается ошибка.

Для изменения данных в базе данных новость удаляется из списка в контексте и производится сохранение. В конце выполнения метода возвращается переход на страницу Index.

Модуль работы с заявками на участие в событии состоит из трех методов, первый из которых – Create позволяет создавать их и отправлять организаторам. Его код показан на рисунках Д.1, Д.2 и Д.3 в приложении Д.

Метод реализует обработку Post запроса и принимает в качестве параметров идентификатор события и список участников. Идентификатор используется для нахождения в базе данных нужного события, после чего проверяется его наличие. В случае, если оно не найдено – возвращается ошибка. Если же событие существует, выполняется валидация данных, при непрохождении которой возвращается ошибка. Поскольку в заявке требуется указать отправителя, следующий шаг – поиск информации о пользователе в БД.

Затем происходит создание объекта класса EventApplication и заполнение его значениями, после чего выполняется прохождение по списку участников внутри цикла. Если пользователь уже зарегистрирован, у его должен иметься идентификатор, что и проверяется в цикле в первую очередь. В случае отсутствия идентификатора пользователя, производится проверка всех данных на их наличие и правильность введения, а в случае электронной почты и номера телефона - еще и на отсутствие пользователей с такими же данными в системе и списке участников заявки. При непрохождении которых добавляются соответствующие ошибки. Если были найдены ошибки заполнения – возвращается страница, с введенными пользователем данными и отображением ошибок.

В случае же корректности данных, ими заполняются значения свойств объекта класса ApplicationData, созданного в начале цикла и этот объект добавляется в список участников новой заявки.

Если же идентификатор пользователя указан, происходит его поиск в БД и проверка на наличии, при непрохождении которой возвращается ошибка.

В случае нахождения пользователя, его данные указываются в переменную нового участника заявки и добавляется подразделение, в котором он участвует. Последний этап выполнения метода – сохранение информации.

На рисунке показан код метода Edit – второго в модуле:



Рисунок 20 - Код метода Edit

Этот метод обрабатывает Post запрос. доступ к нему имеют только организаторы событий, ведь его функция – принятие или отклонение заявок на участие в событии. Входными данными являются идентификатор заявки, комментарий и действие (принять / отклонить). Выполнение инструкций начинается с поиска заявки в БД по идентификатору и проверка ее наличия. Если заявка найдена, проверяется доступ организатора к ее изменению и возвращается ошибка, если его нет.

Для указания статуса принятия заявки, строковое значение переводится в булево посредством сравнения действия с строкой “success”, что означает ее принятие, после чего изменение сохраняется.

После изменения заявки требуется отправить ее создателю сообщение на электронную почту, для чего создается объект класса, содержащий требующуюся информацию, выполняется рендеринг страницы в строку и отправка сообщения.

Заключительным этапом выполнения метода является добавление пользователей в систему, если заявка принята и перенаправление пользователя.

Третий метод модуля – Details позволяет просматривать подробную информацию о заявке. Ее код показан на рисунке :



Рисунок 21 - Код метода Details

Он обрабатывает Get запрос и принимает в качестве параметра идентификатор заявки.

Для получения информации о заявке, требуется найти ее в базе данных, для чего используется ее идентификатор и идентификатор пользователя, что позволяет предоставлять доступ к данным только пользователю, отправившему эту заявку.

Если заявка не была найдена, возвращается соответствующая ошибка, в ином случае выполнение метода продолжается установкой статуса, для чего используется значение свойства accepted. При его отсутствии, статус устанавливается в значение «не принята», а если значение есть, статус устанавливается в соответствии с ним. После выполнения поиска и всех проверок возвращается страница, в которую отправляется модель – найденная заявка.

Модуль администрирования выполняет задачи управления доступом пользователей и данными в системе, такими как, события, организаторы и новости.

Для создания нужного функционала были разработаны три метода, первый из которых – BlockUser выполняет задачу блокировки пользователя. Его код показан на рисунке :

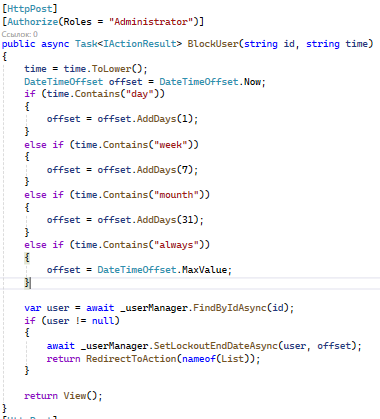


Рисунок 22- Код метода BlockUser

BlockUser доступен только Администраторам и обрабатывает Post запрос. Входными данными является идентификатор блокируемого пользователя и срок. Выполнение начинается с перевода строки строка в нижний регистр и создание переменной – периода. Следующий этап – определение значения периода, для чего проводится нахождение в строке срока значений day, week, month и always, означающие день, неделю, месяц и бессрочно соответственно. После определения срока производится поиск пользователя, его блокировка, если он найден и перенаправление.

Второй метод модуля администрирования – DeleteOrganization, выполняющий функцию удаления информации из системы. Его код показан на рисунке :

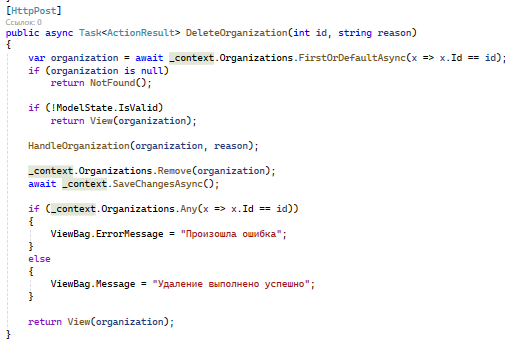


Рисунок 23- Код метода DeleteOrganization

Этот метод принимает входные данные, которые включают идентификатор удаляемой организации и причины удаления. Сначала он осуществляет поиск организации в базе данных. В случае, если запись с заданным идентификатором не найдена, возвращается ошибка, что обеспечивает надежность процесса удаления.

После успешного нахождения организации в базе данных, метод отправляет уведомление организатору об удалении с указанием причины.

Далее происходит удаление записи об организации из базы данных и проверка на выполнение удаления. Этот шаг важен для поддержания целостности и актуальности данных в системе.

В завершение метод возвращает страницу с информацией о модели.

Метод EditApplication отвечает за обработку заявок на регистрацию организаторов. Его код представлен на рисунке :

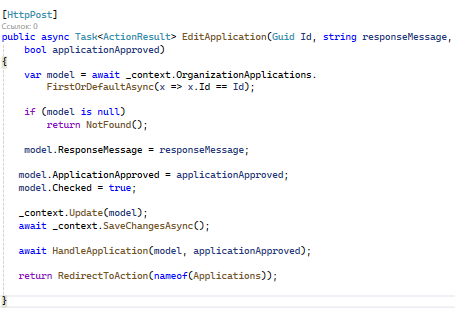


Рисунок 24 - Код метода EditApplication

Метод обрабатывает Post запрос и принимает входные данные в виде идентификатора заявки, комментария администратора и булево, означающее, принята ли заявка.

Для изменения информации о заявке, производится ее поиск в базе данных по идентификатору и проверка на наличие. Если запись в БД была найдена, производится изменение значений ее свойств и сохранение.

Последний этап выполнения работы метода – отправка электронного письма отправителю заявки и возврат перенаправления.

Последний, но не по важности модуль веб сервиса – аутентификация, включает в себя три метода, позволяющие полностью покрыть функции разделения доступа к информации, а именно:

Login – авторизация;

Registration – регистрация в системе;

OrganizationRegistration – регистрация для организаторов.

Код метода Login показан на рисунке :

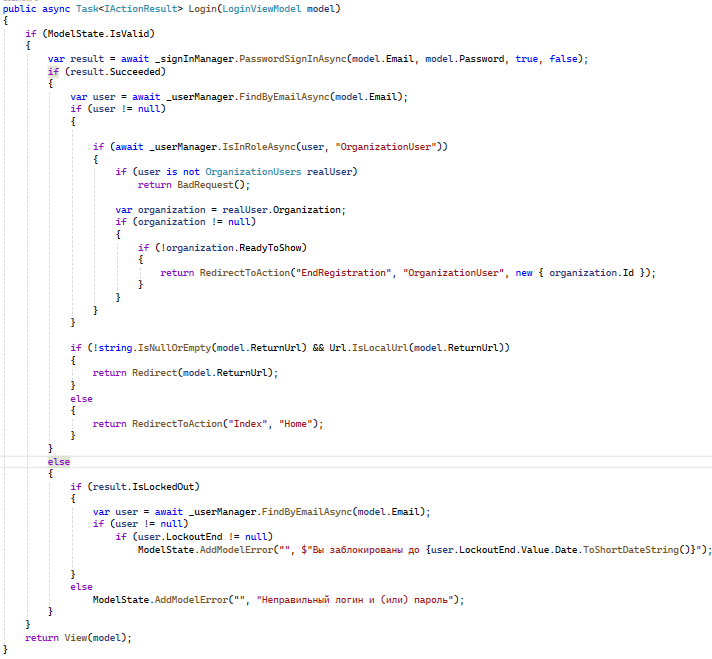


Рисунок 25 - Код метода Login

Его входными данными является объект класса LoginViewModel, по данным которого выполняется авторизация с помощью \_signInManager. Если авторизация выполнена – производится поиск полной информации о пользователе в БД. Если он является организатором события, не завершившим заполнение информации – возвращается перенаправление на страницу заполнения. В иных же случаях возвращается перенаправление на нужную страницу.

В случае, если авторизация не произведена, возвращается страница с ошибкой о неправильности данных или блокировке

Поскольку перед авторизацией пользователь должен быть зарегистрирован, был создан метод Registration, реализующий такой функционал. Его код показан на рисунке :



Рисунок 26 - Код метода Registration

Метод является обработчиком Post запроса и в роли входных данных принимает объект класса RegistrationViewModel.

Первым этапом выполнения метода является валидация модели регистрации. В случае ее непрохождения, возвращается вид с моделью, в котором будут отображены ошибки. Если же модель прошла валидацию, создается новый экземпляр класса WebUser с заполнением значений его свойств.

Перед созданием записи в БД требуется проверить наличие пользователя с такими же данными. Если проверка не пройдена, добавляется ошибка и возвращается страница, на которой она будет отображена. В ином случае производится регистрация пользователя, добавление его в соответствующую роль и авторизация с возвращением перенаправления на главную страницу.

Если регистрация не произошла, добавляются ошибки и возвращается вид, в котором они будут отображены.

Последний метод модуля – OrganizationRegistration выполняет задачу добавления заявок на регистрацию организаторов. Его код можно увидеть на рисунке :

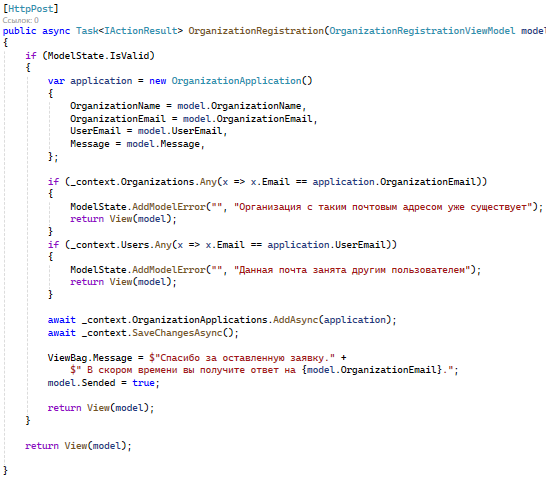


Рисунок 27 - Код метода OrganizationRegistration

Этот метод принимает в качестве входных данных объект класса OrganizationRegistrationViewModel. Сначала производится валидация входных данных. Если в процессе валидации обнаруживается ошибка, метод возвращает представление, на котором отображаются соответствующие ошибки. В противном случае создается объект заявки, и его данные заполняются.

После создания заявки производится проверка наличия организаций и пользователей с аналогичными данными в базе данных. Если указанные данные уже существуют, возвращается представление с сообщением об ошибке.

В случае успешного прохождения всех проверок новая заявка добавляется в базу данных, и метод возвращает представление с сообщением о успешном создании заявки.

Поскольку информационная система включает в себя мобильное приложение, требовалось разработать модуль API, который позволит передавать информацию с сервера на телефон. Код метода, возвращающего информации о событии показан на рисунке :

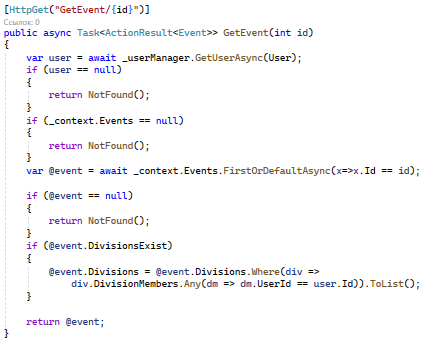


Рисунок 28 - Код метода GetEvent

Входным параметром является идентификатор события, по которому производится поиск в базе данных.

Если запись в БД не найдена – возвращается соответствующая ошибка. В случае прохождения проверки, производится поиска пользователя и проверка на его существование.

Последний этап выполнения метода – выбор подразделений, в которых участвует пользователь, если они существуют в событии и возврат события, которое автоматически преобразуется в Json строку фреймворком ASP NET Core.

Помимо API с мобильным приложением сервер взаимодействует с помощью TCP протокола, что позволяет реализовывать чат. Для выполнения этой задачи был создан модуль обратной связи, состоящий из пяти методов.

Метод Start, показанный на рисунке , выполняет задачу запуска сервера:

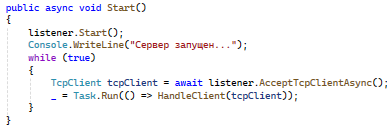


Рисунок 29 - Код метода Start

Выполняется запуск слушателя подключений и запускается бесконечный цикл, в котором выполняется подключение новых клиентов и обработка взаимодействия с ними.

Для обработки запросов пользователя используется метод ClientHandler, показанный в приложении Е на рисунках Е1 и Е2.

Входным параметром метода является TCP соединение, из которого производится получение потока и потокового читателя, из которого считывается строка, в которой должен содержаться идентификационный токен пользователя.

Если в строке есть токен, то по нему производится поиск пользователя в БД. Если пользователь найден, в список подключенных пользователей добавляется новый.

После идентификации пользователя запускается бесконечный цикл взаимодействия с клиентом, в котором производится построчное чтение сообщений и их десериализация в объект класса сообщений.

Если сообщение было преобразовано, производится получение чата, к которому оно относится с помощью метода GetChat. Когда чат получен, к нему добавляется сообщение от пользователя и изменения сохраняются в БД.

Для отправки сообщения собеседнику производится его поиск. Когда второй участник чата найден, выполняется отправка уведомлений и его поиск среди подключенных клиентов и отправка сообщения с помощью метода BroadcastMessage.

Поскольку сообщение может быть отправлено в уже существующий или новый чат, требуется как искать существующий, так и новый. Эту задачу выполняет метод GetChat, код которого показан на рисунке :

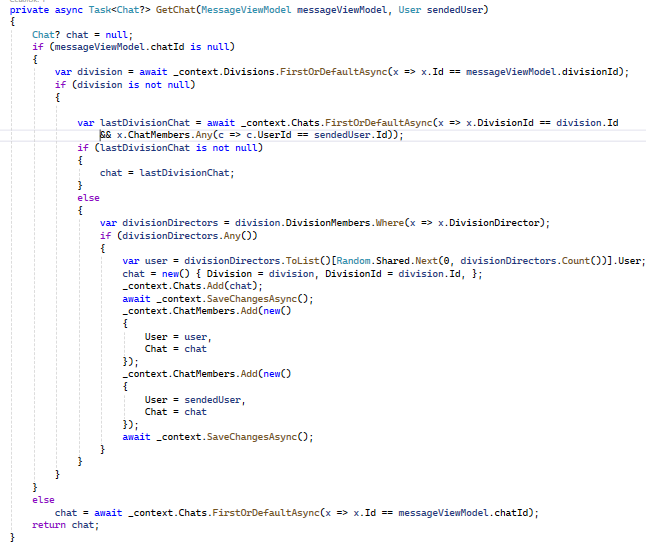


Рисунок 30 - Код метода GetChat

Входными параметрами является сообщение и пользователь, отправивший его. Если чат существует, возвращается найденный в БД. В противном случае производится поиск подразделения и последний диалог пользователя в нем.

В случае существования такого чата – возвращается он, в иной ситуации, производится получение списка ответственных в подразделении и случайный выбор собеседника среди них.

Если нужный чат все еще не был найден, создается новый, ему добавляются участники. Все это сохраняется в БД и возвращается.

Чтобы пользователи узнавали о полученных сообщениях даже при закрытом приложении, требуется реализация отправки уведомлений, которой является метод PushPopUp, показанный на рисунке :

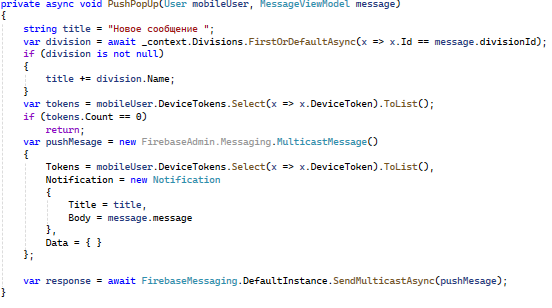


Рисунок 31 - Код метода PushPopUp

Во входных параметрах он получает адресата и сообщение. Перед отправкой сообщения выполняется создание заголовка, включающего в себя название подразделения и получение токена пользователя, по которому выполняется отправка уведомлений.

Для отправки уведомления создается объект класса MulticastMessage, в котором указываются токены, тело и заголовок. В конце выполнения метода этот объект отправляется с помощью сервиса Firebase.

Последний метод модуля используется для отправки сообщений по TCP. Его код показан на рисунке :

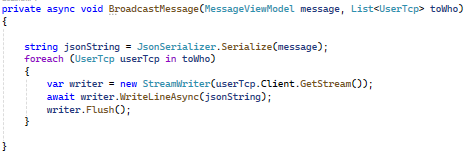


Рисунок 32 - Код метода BroadcastMessage

Входными параметрами являются сообщение и список конечных адресов. Для отправки сообщение преобразуется в строку Json и отправляется потоковым писателем по каждому из адресов в цикле.

В мобильном же приложении крупнейшим модулем является просмотр информации. Он включает в себя работу с сервером и интерфейсом приложения.

Взаимодействие с сервером производится с помощью REST API, для работы с которым в программе используется HTTP клиент и библиотека Retrofit.

Representational State Transfer (REST) в переводе — это передача состояния представления. Технология позволяет получать и модифицировать данные и состояния удаленных приложений, передавая HTTP-вызовы через интернет или любую другую сеть [[source](https://cloud.vk.com/blog/vvedenie-v-rest-api)].

HTTP —протокол передачи гипертекста. Он представляет собой список правил, по которым компьютеры обмениваются данными в интернете. HTTP умеет передавать все возможные форматы файлов — например, видео, аудио, текст. Но при этом состоит только из текста. Протокол HTTP используют ещё с 1992 года. Он очень простой, но при этом довольно функциональный. Работает он с помощью протоколов TCP/IP и использует их, чтобы передавать данные [[source](https://skillbox.ru/media/code/chto-takoe-http-i-zachem-on-nuzhen/)].

Retrofit — это библиотека, которая упрощает работу с сетевыми запросами в приложениях на Android. Например, с этим инструментом можно настроить работу приложения с погодой. Разработчик задаёт параметры запросов, которые должны поступать на сервер и забирать данные по температуре в разных городах. Всё остальное происходит автоматически. В итоге пользователь видит актуальные данные каждый раз, когда открывает приложение[[source](https://practicum.yandex.ru/blog/retrofit-na-android/)].

Для работы с библиотекой Retrofit был создан интерфейс для реализации методов, выполняющих отправку запросов на сервер. Его код отображен на рисунке :



Рисунок 33 - Интерфейс Retrofit

Помимо библиотеки для работы был создан статический метод создания интерфейса Retrofit. Код этого метода показан на рисунке :

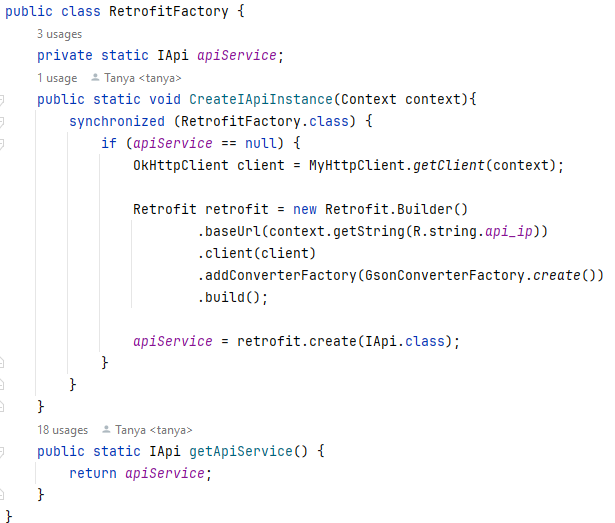


Рисунок 34 - Создание экземпляра интерфейса

Этот метод проверяет, создан ли экземпляр интерфейса. В случае, если нет – производится создание HTTP клиента, использующегося для создания объекта Retrofit и его настройки, а именно, указания базового адреса Api, клиента и конвертора JSON. Последний этап выполнения метода – присвоение интерфейсу значения в виде созданного объекта.

Последним средством, требующимся для работы с сервером является HTTP клиент, код которого показан на рисунке :



Рисунок 35 - Создание экземпляра клиента HTTP

Показанный метод выполняет проверку на наличие значения у статической переменной клиента. Если значение присутствует – метод возвращает его значение. В ином случае – создается обработчик куки файлов, реализующий функционал добавления полученных из ответа сервера файлов в последующие запросы, после чего создается объект логгера и присвоение значения статической переменной client, посредством построения объекта с указанием настроек.

После реализации возможности взаимодействия с API стало возможным получения данных с сервера. Получив эту информации можно выполнять ее отображение в интерфейсе приложения. Рисунок показывает код метода getMeasures, выполняющего отображение информации о мероприятиях, в которых пользователь принимает участие.

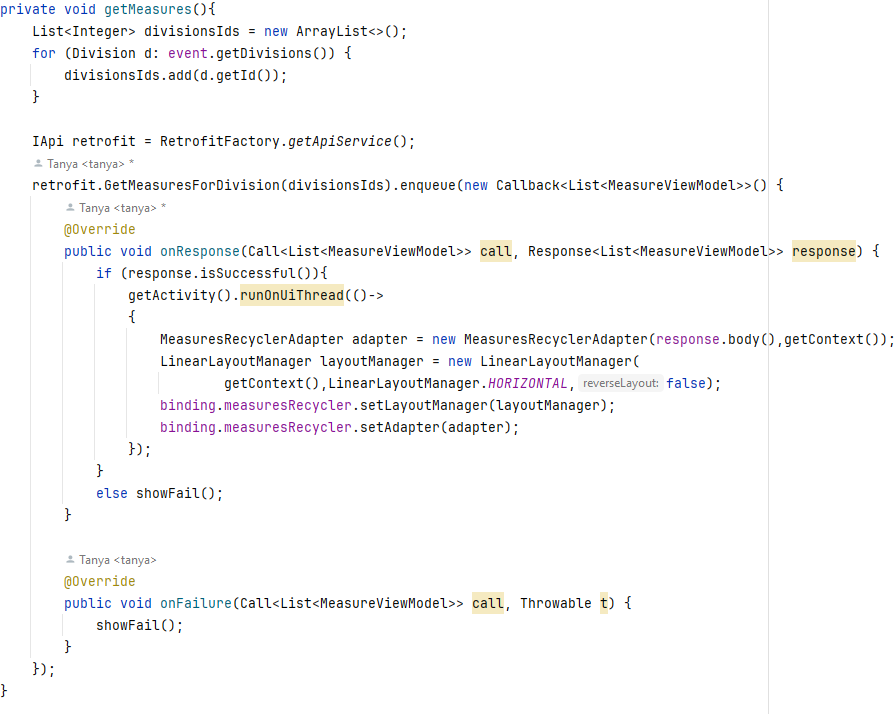


Рисунок 36 - Метод getMeasures

Поскольку обработчик запроса на сервере принимает в виде параметра список подразделений, для которых нужно получить расписание, выполнение метода начинается с создания и заполнения списка Id подразделений. После получения списка создается переменная – интерфейс Retrofit и с ее использованием выполняется запрос к серверу. В случае успешного получения запроса выполняется создание адаптера для списка мероприятий, полученных с сервера, и его использование в соответствующем RecyclerViewer элемента интерфейса. В противном случае отображается ошибка.

Модуль аутентификации мобильного приложения состоит из методов checkAuthorize, sendCode, resendCode, sendPhoneNumber, подробное описание которых представлено ниже.

Метод checkAuthorize используется для проверки авторизации пользователя. Его код отображен на рисунке :



Рисунок 37 - Метод checkAuthorize

Работа метода заключается в отправке запроса к API отправляемого с использованием HTTP клиента, который создается при запуске приложения и считывает токен пользователя, добавляя его в куки файлы, отправляемые на сервер для проверки авторизации. Если ответ от сервера положительный – открывается основная активность приложения, в ином случае пользователь остается на странице авторизации.

Сама авторизация состоит из двух этапов – отправки номера телефона и кода на сервер.

Отправку номера телефона на сервер реализует метод sendPhoneNumber, код которого показан на рисунке :

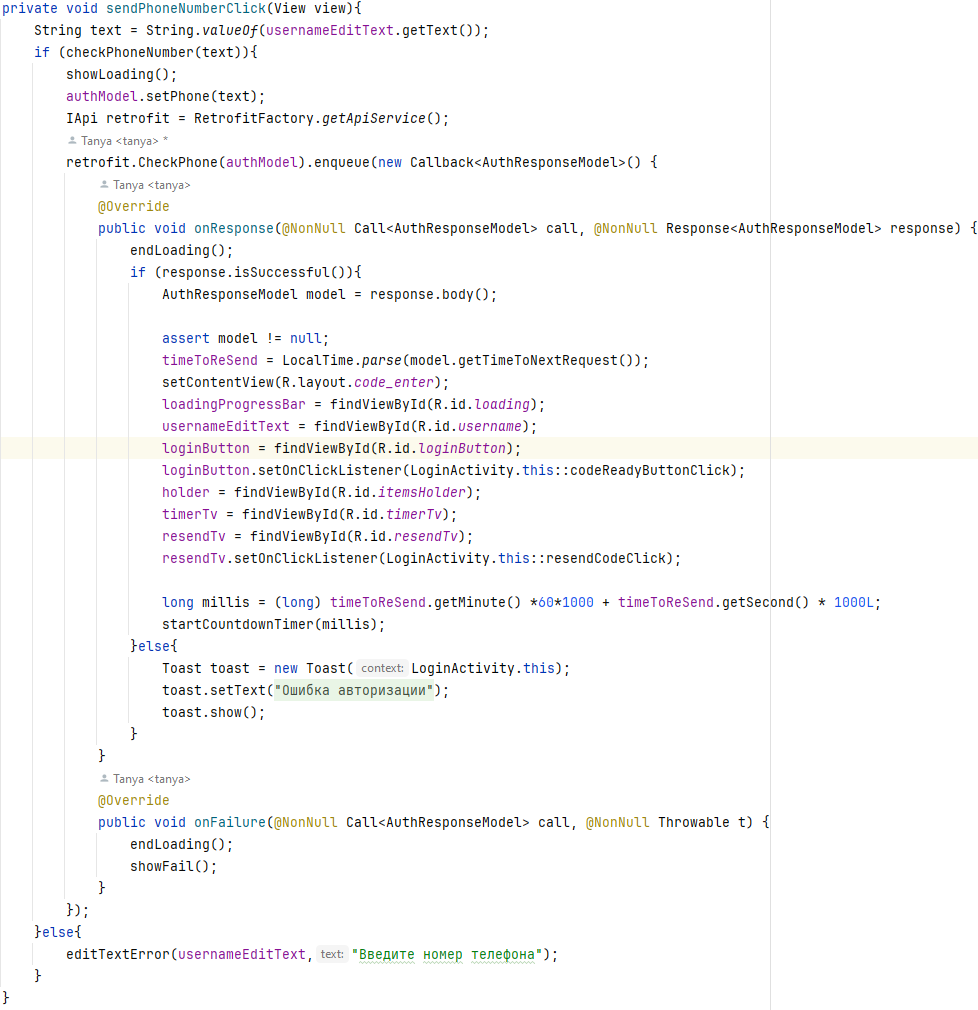


Рисунок 38 - Метод sendPhoneNumber

Метод вызывается при клике на кнопку. В первую очередь считывается введенный пользователем номер телефона и производится его проверка на корректность. Если проверка пройдена, то модели авторизации присваивается номер телефона и производится отправка запроса на сервер, в ином случае, отображается ошибка. В случае получения положительного ответа от сервера, производится смена макета страницы и запуск таймера для повторной отправки кода, а если ответ сервера отрицательный – отображается ошибка.

Интерфейс, открываемый после проверки номера телефона, предназначен для введения пользователем кода авторизации. Саму отправку кода реализует метод sendCode. Код этого метода отображен на рисунке :



Рисунок 39 - Метод sendCode

Метод вызывается при клике на кнопку отправки. После нажатия на кнопку происходит получение текста, введенного пользователя и проверка на длину кода. Если она равна пяти – в модели авторизации устанавливается код и производится отправка запроса на сервер. В ином случае, отображается ошибка. В случае получения положительного ответа от сервера, производится проверка авторизации пользователя, а если ответ сервера отрицательный – отображается ошибка.

Также в модуле авторизации предусмотрен функционал повторного запроса кода авторизации по потребности пользователя. Для выполнения этой задачи используется метод resendCode, код которого показан на рисунке :



Рисунок 40- Метод resendCode

Пользователь инициирует выполнение метода, нажатием кнопки запроса повторной отправки кода. Сначала происходит отображение элемента, который показывает оставшееся время до возможности нового запроса кода, а также скрытие кнопки, на которую было совершено нажатие для вызова данного метода. После внесения изменений в пользовательский интерфейс, запускается таймер, который отсчитывает время до следующей возможности получения нового кода, а также отправляется соответствующий запрос на сервер для этой цели.

Для работы с информацией в модуле обратной связи используется ViewModel(vm), показанная на рисунках Г.1, Г.2.

Согласно документации Android, класс ViewModel выступает в роли компонента, который предназначен для хранения и управления данными, связанными с пользовательским интерфейсом и живущими в пределах жизненного цикла приложения. vm помогает отделить бизнес-логику от пользовательского интерфейса, обеспечивая сохранение данных при повороте экрана или других изменениях конфигурации устройства [[source](https://habr.com/ru/companies/otus/articles/807723/)].

Разработанная vm включает в себя переменные:

* chats – список чатов пользователя;
* newMessage – новое сообщение;
* currentChat – текущий чат, открытый в активности;
* isError –отображение ошибки.

Конструктов класса вызывает метод getChats для получения списка чатов и создает статический экземпляр класса.

Метод getChats выполняет запрос к API для получения чатов и присваивает результат переменной chats при успешном выполнении, в случае же ошибки, меняется значение переменной isError.

Методы getInstance, getChats, getNewMessage и getCurrentChat нужны для получения значений соответствующих переменных, а setCurrentChat принимает входные данные объекта ChatViewModel и присваивает его значение переменной currentChat.

Наиболее важным методом класса является addMessage, принимающий в виде входных данных объект класса MessageViewModel. Он вызывается при отправке пользователем нового сообщения и при получении сообщении от сервера. Первым этапом выполнения этого метода является проверка существования чата, в который отправляется сообщение в списке чатов. Для этого создается переменная отвечающая за то, найден ли чат, после чего выполняется цикл по всем элементам списка chats. Если чат с нужным идентификатором будет найден, в список его сообщений добавляется новое – входной параметр и прекращается выполнение цикла. В случае же, если чат найден не был, для получения информации о нем выполняется запрос к API. В случае его успешного выполнения полученный из ответа чат добавляется в список уже существующих, в противном случае значение переменной, отвечающей за ошибку, устанавливается true.

Последний метод разработанного класса updateMessageTime вызывается, когда сообщение доставлено на сервер и используется для установки времени, когда сервер его получил. Его входным параметром является объект класса MessageViewModel – сообщение, для которого требуется установить время отправки. В ходе выполнения инструкций метода производится получение списка сообщений текущего чата, среди которых происходит поиск сообщения – параметра. Когда нужное сообщение найдено, производится присвоение времени отправки.

В рамках задачи по реализации функционала работы с сообщениями, помимо получения чатов через API, требовалось обеспечить мгновенное получение сообщений адресатом сразу после их обработки на сервере. Для достижения этой цели необходимо было использовать протокол, поддерживающий постоянное соединение, в отличие от HTTP, реализующего архитектуру "запрос-ответ".

В качестве подходящего протокола был выбран TCP, который обеспечивает постоянное подключение клиента к серверу. Для работы с TCP был разработан класс TcpClient.

Для работы с протоколом TCP был разработан класс TcpClient, код которого отображен на рисунке :



Рисунок 41 - Класс TcpClient

Конструктор класса устанавливает адрес и порт сервера, с которым планируется взаимодействовать.

Метод connect выполняет подключение к серверу и присваивает значения сокету, работающему с подключением, потоковым писателю и читателю.

Метод setUserCode устанавливает значение переменной userCode – токен аутентификации пользователя выполняет вызов метод authentication, который, в свою очередь, отправляет токен на сервер с помощью sendData, принимающего строку как входной параметр и отправляет ее на сервер с помощью потокового писателя.

Метод connectWithRetries используется для переподключения к серверу в случае потери соединения. В процессе его выполнения переменным, отвечающим за показание статуса работы клиента, присваиваются значения. Следующий этап – попытки подключения раз в секунду в цикле, выполняющемся, пока не будет пройдена повторная авторизация.

После создания инструментов для работы с сервером и хранением информации было реализовано получение сообщений, код которого показан на рисунке :



Рисунок 42 - Прослушивание TCP

Метод выполняется в параллельном потоке, не блокируя работу основной системы. Используется объект класса Gson для преобразования JSON-строк в объекты сообщений. Бесконечный цикл обеспечивает постоянное получение сообщений с сервера, их конвертацию и добавление в хранилище. В целях предотвращения ошибок предусмотрена проверка наличия данных и обработка исключений, инициирующая повторное подключение к серверу.

Для отправки сообщений был написан метод onSendClick, код которого показан на рисунке :



Рисунок 43 - Код отправки сообщений

Метод вызывается нажатием пользователя на кнопку, после чего производится проверка на возможность отправки сообщения и наличие текста в соответствующем элементе интерфейса. Если все проверки пройдены, создается объект сообщения, устанавливаются значения его полей и создается асинхронная задача, поскольку Android не позволяет использовать интернет-соединение в основном потоке. Код этой задачи показан на рисунке :



Рисунок 44 - Асинхронная задача отправки сообщения

Задача вызывает метод sendMessage TCP-клиента, что позволяет отправить сообщение на сервер.

Последний этап разработки модуля обратной связи – работа с интерфейсом и именно для этого пригодится созданная ранее vm. Она позволит взаимодействовать с одинаковыми данными из фрагмента со списком чатов и активностью отдельного чата, что даст возможность отображать новые сообщения как в списке чатов и нужном чате одновременно. Так, на рисунке показан метод отслеживания изменений списка чатов и обновление интерфейса в фрагменте.



Рисунок 45 - Код отображения списка чатов

Показанный код выполняет сортировку списка чатов по времени отправки их последнего сообщения, создает адаптер и устанавливает его для соответствующий элемент интерфейса RecyclerViewer.

Код, выполняющий задачу отображения сообщений в интерфейсе открытого чата, показан на рисунке :



Рисунок 46 - Отображение сообщений открытого чата

В начале выполнения производится получение информации о чате из API по его идентификатору, если чат уже существует или же, в противном случае, по идентификатору подразделения. После получения информации о чате производится установка значения переменной, отвечающей за текущий чат в vm и добавляется отслеживание ее изменения. Так при добавлении сообщения в текущий чат, выполнится код, создающий адаптер со списком сообщений и устанавливающий его для соответствующего RecyclerViewer в интерфейсе, после чего производится прокрутка этого элемента в самый низ.

Последний модуль мобильного приложения – редактирование информации о пользователе состоит из трех методов:

* getData - получение информации;
* updateImage - выбор изображения;
* saveChanges - сохранение изменений.

Код метода получения информации отображен на рисунке :



Рисунок 47 - Код метода получения информации о пользователе

Последовательность его выполнения заключается в получение экземпляра интерфейса Retrofit с его последующим использованием для выполнения запроса получения информации о пользователе. При успешном получении ответа от сервера, интерфейс заполняется полученными данными, в противном случае производится отображение ошибки.

После разработки информационной системы требуется провести тестирование, что позволит проанализировать правильность работы программного продукта и исправить ошибки при их обнаружении.

# 5. Тестирование разработанного продукта

## 5.1. Анализ и выбор методов тестирования

Прежде, чем проводить тестирование, требуется произвести анализ методов, которыми его можно производить.

Этот процесс включает в себя оценку различных методик тестирования, их применимости к конкретному проекту, а также выбор наиболее эффективных и оптимальных вариантов. Рассмотрим основные методы тестирования и их применимость:

* модульное тестирование: Этот метод направлен на проверку отдельных модулей системы на корректность работы. Для системы организации и сопровождения мероприятий этот метод может быть менее эффективным из-за сложной взаимосвязи различных компонентов системы;
* интеграционное тестирование: Этот метод проверяет взаимодействие между различными модулями или компонентами системы. Для информационной системы организации мероприятий интеграционное тестирование будет важным для проверки взаимодействия между веб-сервисом и мобильным приложением;
* системное тестирование: Этот метод проверяет систему в целом на соответствие требованиям и ожиданиям пользователей. Для разработанной системы такое тестирование позволит проверить ее функциональность, производительность и безопасность;
* нагрузочное тестирование: Этот метод направлен на оценку производительности системы при различных нагрузках. Для информационной системы организации мероприятий важно провести нагрузочное тестирование для обеспечения ее стабильной работы во время пиковых нагрузок, например, при регистрации большого числа участников на мероприятие.

Выбор оптимальных методов тестирования зависит от особенностей проекта, его требований и целей. Комбинация различных методов тестирования позволяет достичь максимальной эффективности и обеспечить высокое качество разрабатываемой информационной системы.

Поэтому для проверки разработанной системы были выбраны следующие методы тестирования:

* интеграционное, поскольку система состоит из двух частей и передача информации между ними должна работать без каких-либо ошибок, что и позволит проверить этот метод;
* нагрузочное. Оно позволит проверить отказоустойчивость системы, ведь серверная часть предназначена для одновременной работы множества пользователей, что представляет большую нагрузку, которое приложение должно выдерживать.

# 5.2. Результаты тестирования

Поскольку интеграция системы заключается в взаимодействии мобильного и серверного приложения через API, требовалось провести проверку целостности и скорости передачи данных.

Для измерения времени между отправкой запроса и получением ответа в мобильном приложении был дописан код, показанный на рисунке :

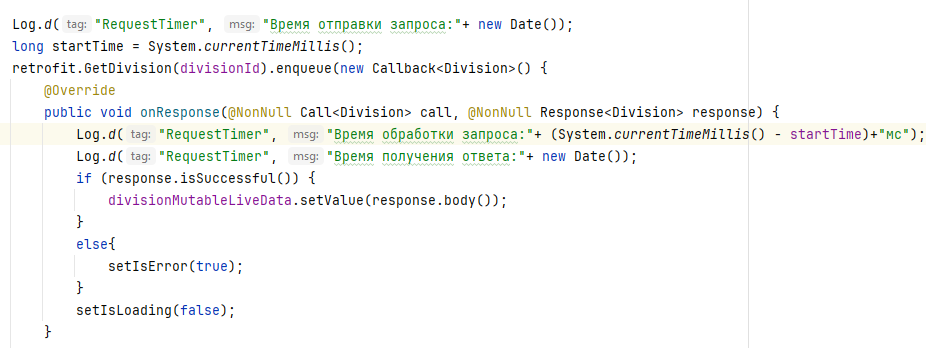


Рисунок 48 - Измененный код

Перед отправкой выполняется установка времени отправки запроса. после получения ответа в консоль выводится результат его сравнения с текущим временем.

На сервере же код, показанный на рисунке : позволяет отследить время получения запроса и отправки ответа.

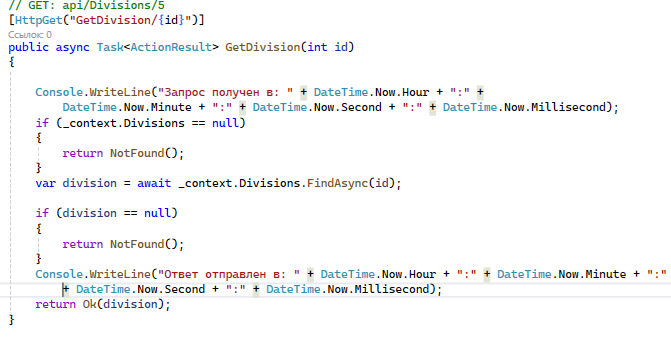


Рисунок 49 - Измененный код

Вывод времени получения запроса и отправки ответа на сервер показан на рисунке :



Рисунок 50 - Время обработки запроса

Результат же проведения тестирования в мобильном приложении показан на рисунке :



Рисунок 51 - Время получения ответа

Результаты тестирования скорости передачи данных в мобильном приложении показывают, что сервер обрабатывает запросы за менее чем сто миллисекунд, а общее время от отправки запроса до получения ответа не превышает одной секунды. Этот результат свидетельствует о высокой эффективности работы системы.

Помимо скорости передачи данных, важным критерием взаимодействия является их целостность после получения. Для тестирования этого критерия использовались стандартные средства сред разработки, использовавшиеся для создания продуктов, позволяющие просмотреть значения отправляемых и полученных данных.

Значения переменных на сервере перед их отправкой показаны на рисунке :

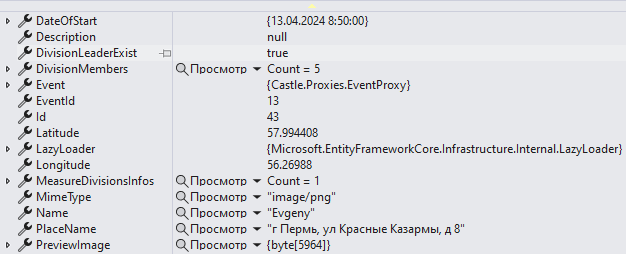


Рисунок 52 - Отправленные сервером данные

Рисунок : отображает значения после получения ответа в мобильном приложении.

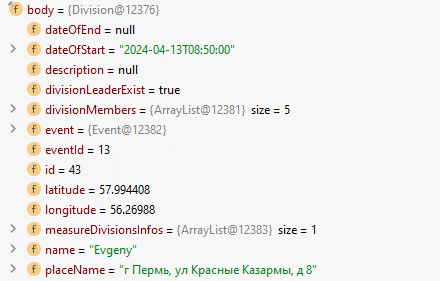


Рисунок 53 - Полученные с сервера данные

При сравнении значений отличий не было обнаружено, что позволяет сделать вывод о сохранении целостности данных после их передачи между сервером и мобильным приложением.

Поскольку обмен данными между частями информационной системы производится быстро и без потерь, выполнение интеграционного тестирование можно признать успешным.

Для проверки нагрузочного тестирования веб сервиса использовались средства интегрированной среды разработки Visual Studio, позволяющей получить подробную информацию о нагрузке на аппаратное оборудование устройства программной.

Поскольку провести тестирование с реальными пользователями не представляется возможным, была создана программа, имитирующая сто одновременных подключений и отправляющая 6 последовательных запросов раз в секунду. Ее код показан на рисунке :

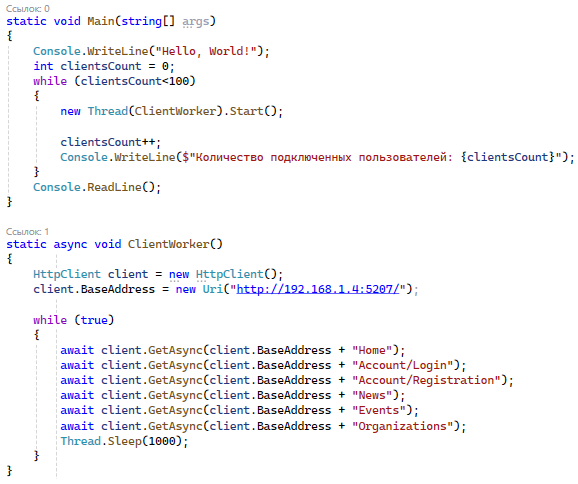


Рисунок 54 - Программа для тестирования

В ходе сеанса тестирования с множеством виртуальных пользователей был получен отчет, отображенный на рисунке :

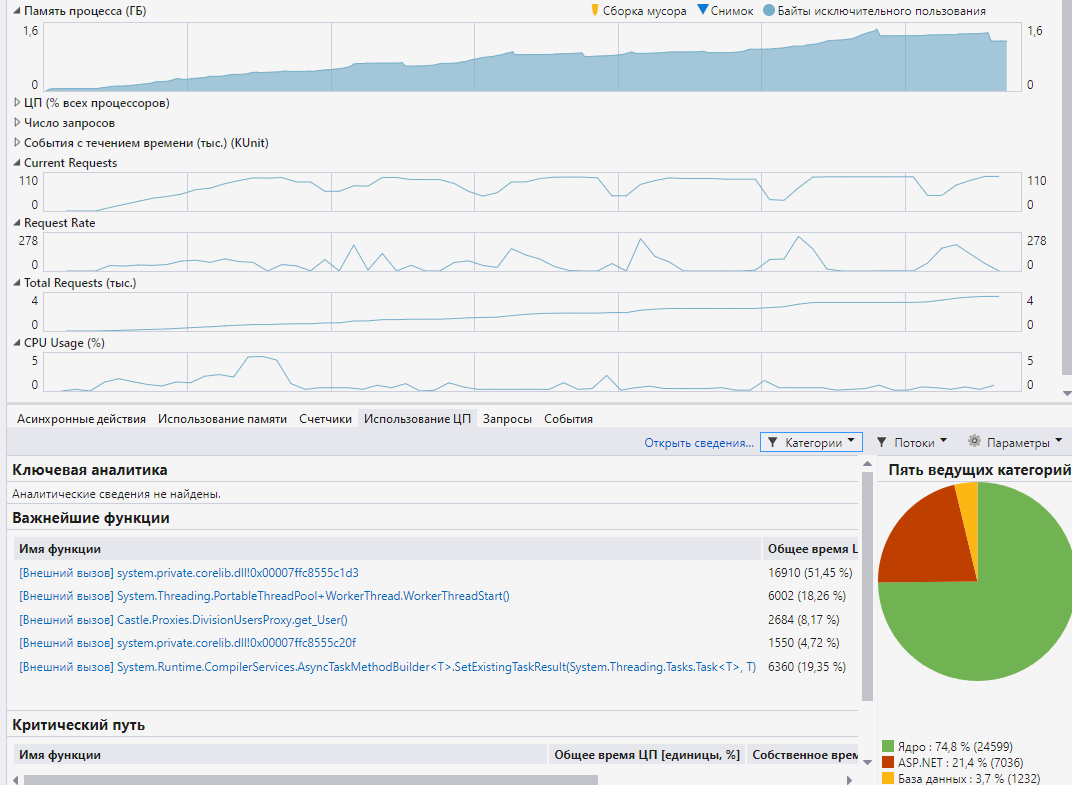


Рисунок 55 – Отчет о производительности системы

В нем можно заметить, что максимальное количество используемой памяти достигло значения в 1.6 гигабайт при 278 одновременных запросах, а использование процессора не превышало пяти процентов.

Помимо этого, отчет отображает распределение нагрузки, подавляющую часть которой занимает сама платформа .NET, на что повлиять нельзя, а база данных использует лишь 3% времени работы ЦП.

Исходя из показаний, полученных из отчета, можно сделать вывод, что разработанная система без проблем справится с множеством одновременно подключенных пользователей и не откажет во время пиковой нагрузки.

# 6. Экономическое обоснование разработки

# 7. Внедрение и установка информационной системы

Минимальные требования к серверу. Процесс развёртывания.

# 8. Сопровождение информационной системы

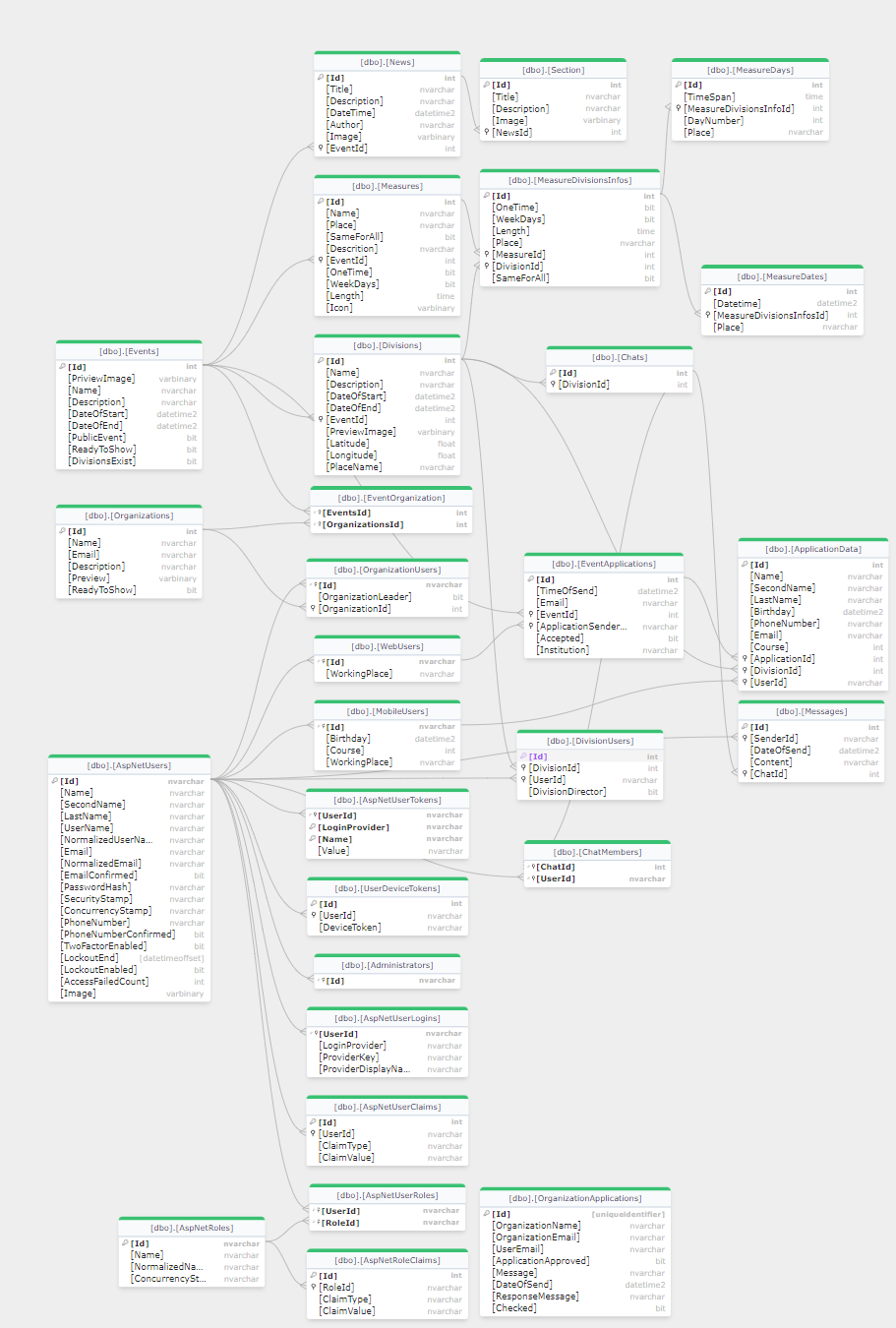
Руководство пользователя

# 9. Настройка безопасности информационной системы

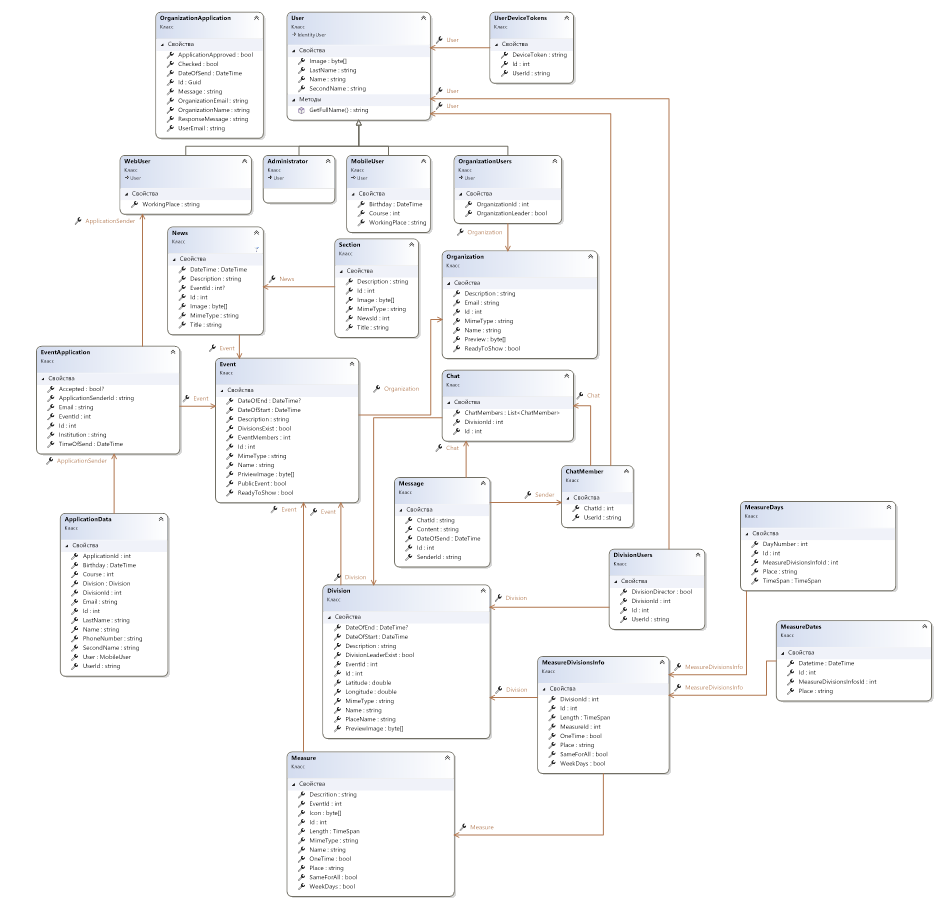
Какие действия вы приняли для того, чтобы обезопасить ИС

# Заключение

# Приложение А



# Приложение Б



# Приложение В

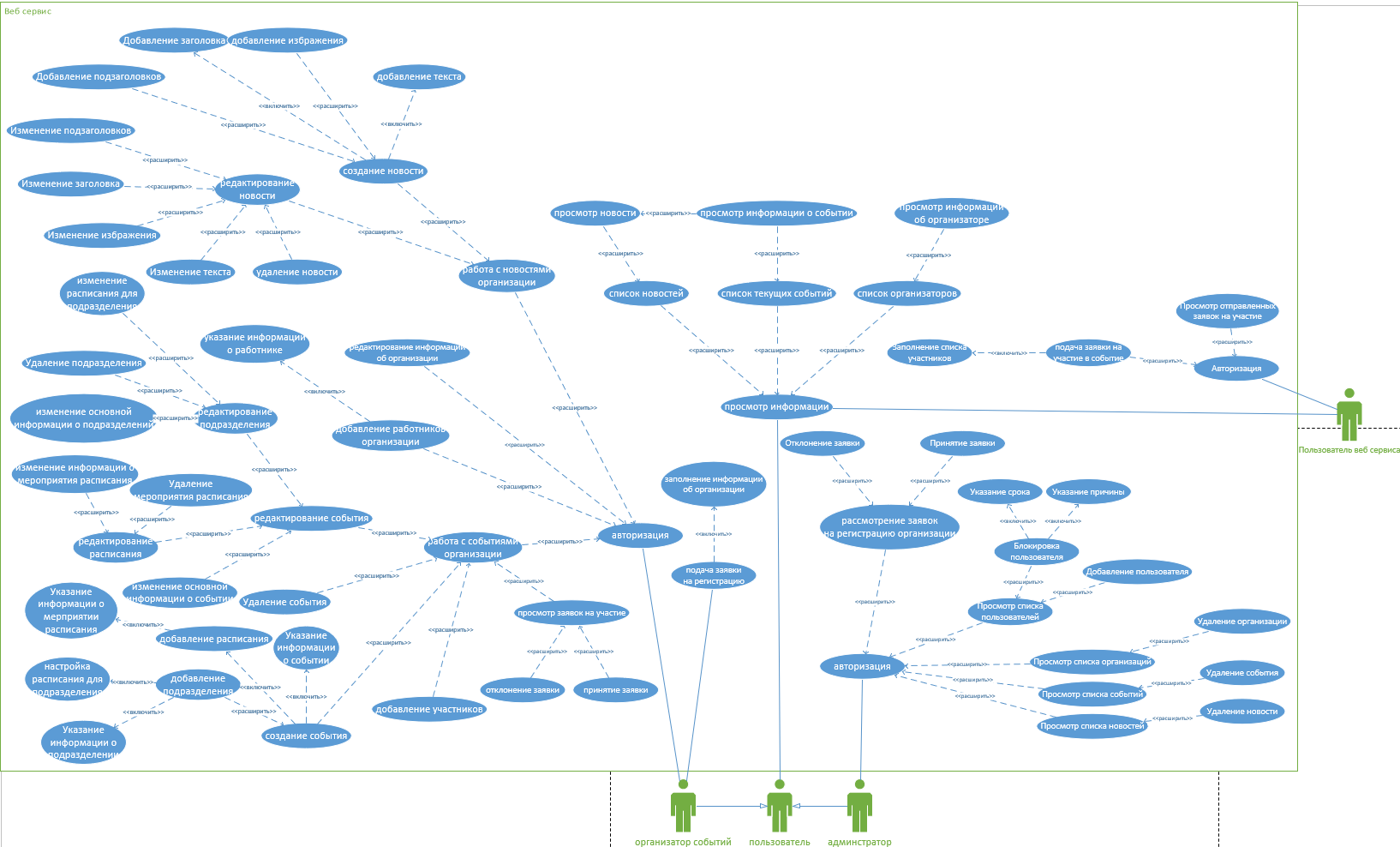


Рисунок В. 1 - Варианты использования для веб сервиса

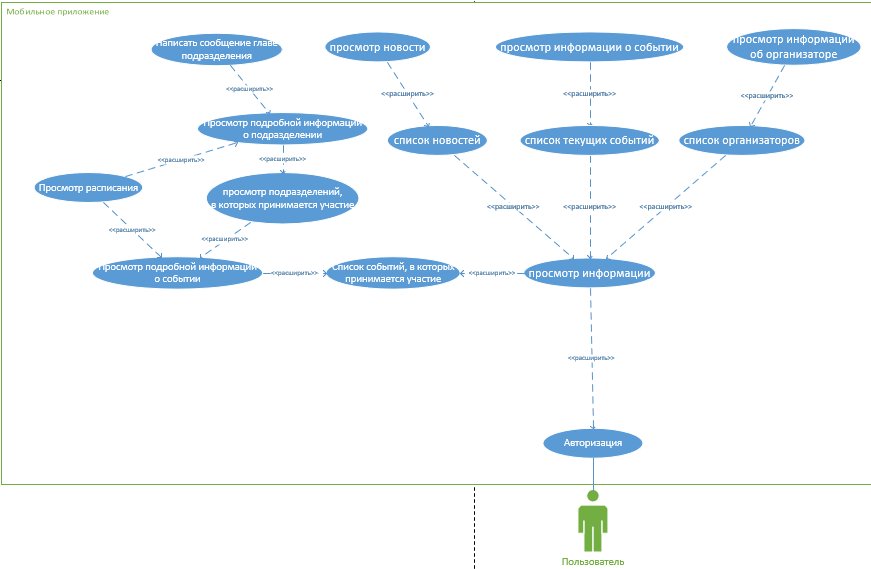


Рисунок В. 2- Варианты использования для мобильного приложения

# Приложение Г

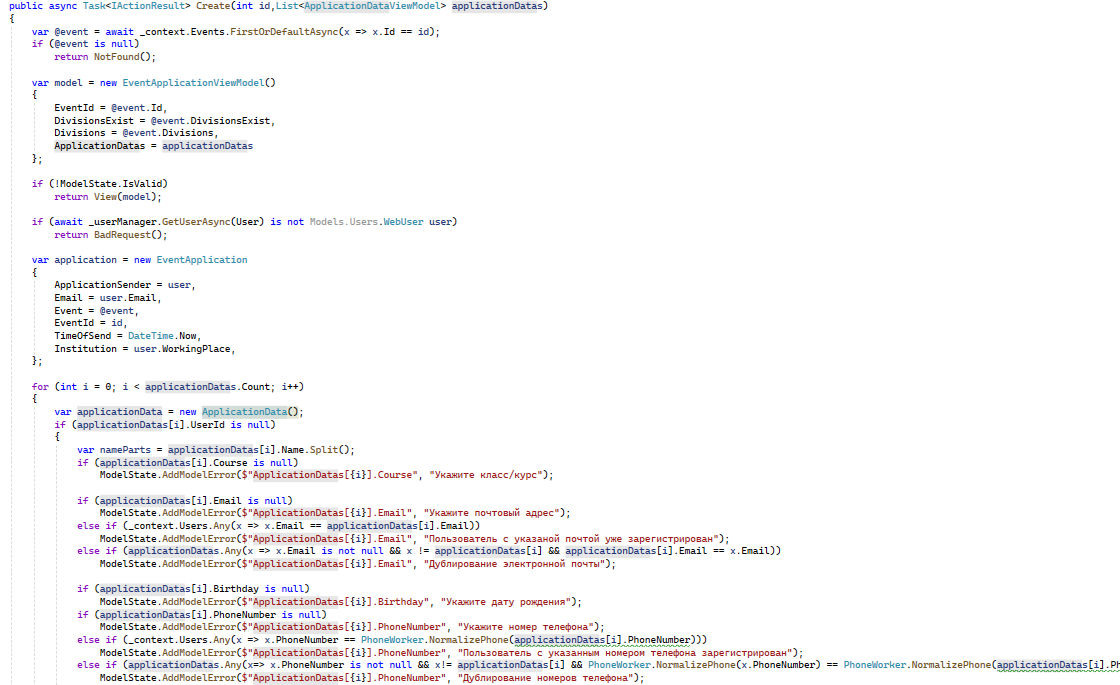


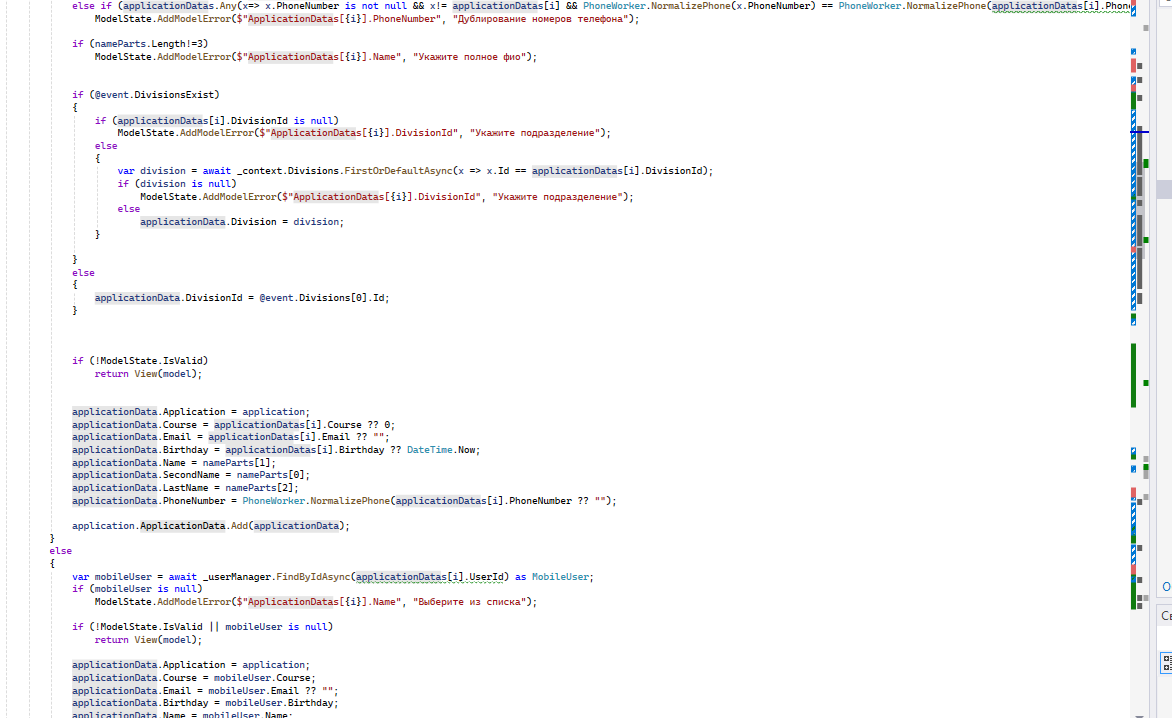
Рисунок Г. 1- ViewModel чатов



Рисунок Г. 2 - ViewModel чатов

# Приложение Д







# Приложение Е



