СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире, где цифровые технологии проникают во все сферы жизни, организация досуга и спортивной активности переживает трансформацию. Согласно исследованию (Global Wellness Institute, 2023), 84% городских жителей испытывают потребность в регулярной физической активности, но 62% из них сталкиваются с трудностями при поиске единомышленников или удобных мероприятий. Традиционные социальные сети (Facebook, Instagram) и нишевые приложения (Strava, MeetUp) не решают проблему комплексно: первые не адаптированы для спортивных событий, вторые фокусируются на узких сегментах (бег, йога, клубные виды спорта).

Особую сложность представляет организация разовых активностей, таких как футбольные матчи среди любителей, уличные квесты или тренировки воркаута. Отсутствие специализированного инструмента для планирования, геопоиска и коммуникации участников приводит к тому, что многие мероприятия остаются нереализованными. Разработка веб-платформы, объединяющей функции социальной сети, календаря событий и аналитики активности, способна стать решением этой проблемы, повысив доступность спорта и вовлеченность пользователей.

Целью данного дипломного проекта является создать многофункциональную веб-платформу для организации и участия в активных спортивных играх.

1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ И РАЗРАБОТКА ТЕХНИЧЕСКОГО ЗАДАНИЯ

1.1. Анализ исходных требований

В рамках данного дипломного проекта разрабатывается веб-приложение, предназначенное для организации и проведения активных спортивных игр, которое позволит пользователям взаимодействовать друг с другом, находить мероприятия и участвовать в спортивных активностях. Проект нацелен на создание универсальной и удобной платформы для пользователей, заинтересованных в командных и индивидуальных спортивных играх. Веб-приложение будет использоваться как для личных тренировок и участия в мероприятиях, так и для организации спортивных событий различного масштаба.

Разрабатываемая система должна обеспечивать следующие основные функции:

– создание и управление мероприятиями - веб-приложение обеспечит возможность для пользователей создавать спортивные мероприятия, указывая все необходимые параметры: дату, время, место, доступные виды спорта, количество участников и другие настройки. Мероприятия могут быть как публичными, так и приватными, доступными только для определённой группы участников. Организаторы смогут управлять списком участников, изменять параметры события и следить за ходом подготовки;

– поиск и участие в мероприятиях - для удобства пользователей будет реализована система поиска мероприятий, позволяющая находить события по различным критериям: интересам (например, футбол, бег, волейбол и т. д.), локации, датам и другим фильтрам. Возможность фильтрации и сортировки событий позволит пользователям быстро найти подходящие мероприятия. Также будет предусмотрена система уведомлений для оповещения пользователей о новых доступных событиях или изменениях в уже выбранных;

– система друзей и подписок – платформа позволит пользователям добавлять друг друга в друзья и подписываться на мероприятия, интересные им. С помощью системы подписок пользователи смогут отслеживать активность своих друзей, получать уведомления о новых мероприятиях, в которых они участвуют, а также следить за изменениями и новыми спортивными предложениями в их сети. Это создаст элемент социальной сети и позволит пользователям быть в курсе событий в спортивной сфере;

– профиль пользователя – каждый пользователь сможет создавать и редактировать свой профиль, добавлять фотографии, указывать интересы, достижения, спортивные цели и другие параметры. В профиле также будет отображаться статистика активности пользователя: количество участвованных мероприятий, достижения, пройденные дистанции, количество побед и другие

показатели. Это поможет пользователю отслеживать свой прогресс и мотивировать себя к дальнейшему улучшению;

– система социальной сети – платформа будет включать базовые элементы социальной сети, позволяющие пользователям взаимодействовать друг с другом. Пользователи смогут оставлять комментарии под мероприятиями, ставить лайки, делиться своими достижениями, фотографиями с мероприятий, а также обмениваться личными сообщениями. Это поможет создать сообщество пользователей, мотивировать их к участию в новых событиях и укрепить социальные связи;

– геолокация и рекомендации – веб-приложение будет использовать встроенную систему геолокации для того, чтобы предлагать пользователям мероприятия, подходящие по интересам и местоположению. Также будет реализована система рекомендаций, которая будет анализировать спортивные предпочтения пользователя и предлагать ему события, исходя из его активности и интересов. Возможность просмотра карты и точных местоположений мероприятий поможет пользователю более эффективно планировать участие;

– чат и уведомления – для удобства общения участников будет встроен мессенджер, позволяющий обмениваться сообщениями и обсуждать детали мероприятия до и после его проведения. Это облегчит организацию групповых тренировок и спортивных игр. Также будет внедрена система push-уведомлений и email-уведомлений, чтобы пользователь всегда был в курсе обновлений, изменений и новых событий;

– аналитика и статистика – пользователи смогут отслеживать свою активность с помощью встроенной аналитики и статистики. Платформа будет предоставлять подробные отчёты о мероприятиях, в которых они участвовали, их достижениях, выполненных тренировках и других спортивных показателях. Это может включать общую статистику по количеству сыгранных матчей, выполненных упражнений, улучшений в личных результатах, и т.д. Вся информация будет отображаться в виде графиков и диаграмм, чтобы пользователи могли легко анализировать свой прогресс и выявлять области для улучшения.

Помимо основного функционала, платформа должна обладать удобным и интуитивно понятным интерфейсом, адаптированным для веб-версии и мобильных устройств. Разрабатываемая система будет использовать микросервисную архитектуру, что обеспечит масштабируемость**,** надежность и безопасность данных.

1.2. Анализ предметной области

Активные игры — это спортивные и развлекательные мероприятия, которые направлены на физическую активность, командное взаимодействие и соревнования между участниками. Они играют важную роль в поддержании здорового образа жизни, улучшении физических кондиций, развитии коммуникационных и социально-психологических навыков, а также в формировании крепких сообществ по интересам. В условиях современного общества, когда люди часто ведут сидячий образ жизни, активные игры становятся отличным способом поддерживать хорошую физическую форму и улучшать эмоциональное состояние.

С развитием цифровых технологий многие аспекты организации активных игр и поиск участников постепенно перемещаются в онлайн-пространство. Однако, в отличие от традиционных видов спорта, таких как футбол, баскетбол или теннис, любительские активные игры часто сталкиваются с рядом проблем, которые существенно затрудняют их организацию и участие в них. Среди таких проблем можно выделить:

– нехватка игроков - найти людей, которые готовы участвовать в активных играх в определённое время и в конкретном месте, бывает довольно сложно. Особенно это актуально для неформальных мероприятий, таких как игры на свежем воздухе или командные активности;

– сложности с координацией - организация активных игр требует точного планирования, согласования времени и места, а также координации участников. Без специального инструмента это может стать большой проблемой;

– отсутствие информации - люди часто не знают о спортивных мероприятиях, которые проходят рядом с ними, и не могут быстро найти подходящие события для участия;

– ограниченные возможности общения - традиционные социальные сети не ориентированы на поиск спортивных мероприятий и игроков, что затрудняет взаимодействие и совместные тренировки.

Активные игры можно условно разделить на несколько категорий, каждая из которых имеет свои особенности и привлекает различных пользователей:

– командные спортивные игры – в эту категорию входят традиционные командные виды спорта, такие как футбол, баскетбол, волейбол, хоккей, регби и другие. Эти игры требуют слаженной работы участников, хорошей координации и заранее спланированного времени проведения. Они ориентированы на людей, которые предпочитают работать в команде и ценят соревновательный дух;

– индивидуальные спортивные активности – бег, велоспорт, плавание, воркаут и другие виды активности, которые можно выполнять как индивидуально, так и в компании. Такие игры подходят для людей, которые не всегда могут найти команду или предпочитают заниматься спортом в

одиночестве, но при этом хотят общаться с единомышленниками и делиться достижениями;

– развлекательные активные игры – лазертаг, пейнтбол, уличные квесты и другие виды активностей, которые не только развивают физическую активность, но и дарят много эмоций. Эти игры ориентированы на людей, которые хотят не только потратить калории, но и получить яркие впечатления, а также развить командные и коммуникативные навыки;

– игры на свежем воздухе – фрисби, бадминтон, петанк и другие игры, которые хорошо подходят для активного отдыха на свежем воздухе с друзьями или семьей. Эти игры не требуют специализированных площадок и могут проводиться в парках, на пляже или в любом другом открытом пространстве.

Проблемы и сложности, возникающие при организации активных игр:

– поиск единомышленников – поиск людей для участия в конкретных активных играх может быть проблемой, особенно если мероприятие должно состояться в определённый день или в конкретном месте. Часто трудно найти группу людей с похожими интересами и свободным временем.

– организация мероприятий – множество людей, организующих активные игры, сталкиваются с трудностями в планировании, координации и управлении событиями. Нет удобных инструментов для выбора площадок, определения времени и места проведения, а также для учёта всех заинтересованных участников. В результате многие мероприятия просто не организуются, поскольку процесс требует много времени и усилий;

– отсутствие информации – часто люди не знают о мероприятиях, которые проходят рядом с ними. Даже если событие организовано, его может быть трудно найти без правильных информационных каналов. Это ограничивает круг участников и затрудняет популяризацию активных игр;

– ограниченные возможности общения – традиционные социальные сети не позволяют эффективно находить спортивные мероприятия, так как они не специализированы на таких активностях. Платформы вроде Facebook или Instagram могут использоваться для обмена информацией, но они не предоставляют удобного функционала для поиска игр, регистрации и общения участников.

Основной задачей социальной платформы является создание удобного инструмента для организации и проведения активных игр и спортивных мероприятий. Приложение должно существенно упростить процесс поиска единомышленников, планирования мероприятий и взаимодействия между участниками.

Одной из ключевых функций платформы станет возможность создания и управления мероприятиями. Пользователи смогут организовывать спортивные игры, указывая такие параметры, как дата, место проведения, уровень сложности, количество участников и дополнительные условия. Это позволит избежать сложностей, связанных с поиском площадки, сбором команды и координацией участников.

Для удобного взаимодействия между пользователями будет реализован механизм поиска и участия в мероприятиях. Любой желающий сможет находить интересующие его события, фильтруя их по различным критериям: виду спорта, местоположению, уровню подготовки участников и личным предпочтениям. Это позволит каждому человеку легко присоединяться к играм и находить компанию для активного отдыха.

Кроме того, приложение будет обладать расширенным социальным функционалом. Пользователи смогут добавлять друг друга в друзья, подписываться на мероприятия, следить за активностью знакомых и общаться в встроенном чате. Это создаст внутри платформы полноценное сообщество любителей активного образа жизни и упростит взаимодействие между участниками.

Еще одной важной особенностью платформы станет интеграция с геолокацией и рекомендациями. Приложение сможет анализировать интересы пользователя и его текущее местоположение, предлагая наиболее подходящие мероприятия. Это обеспечит большую вовлеченность участников, а также поможет находить спортивные события, проходящие поблизости.

Для тех, кто хочет следить за своей активностью, будет реализован раздел с аналитикой и статистикой. Пользователи смогут просматривать историю участия в мероприятиях, отслеживать личные достижения и получать рекомендации по улучшению спортивных показателей.

Таким образом, разрабатываемая платформа объединит в себе возможности социальной сети, инструменты для организации мероприятий и сервисы для поиска спортивных игр. Она позволит любителям активного образа жизни легко находить единомышленников, координировать совместные тренировки и принимать участие в различных спортивных событиях, делая спорт более доступным и увлекательным.

1.3. Анализ существующих прототипов и аналогов

На современном рынке представлено множество мобильных приложений, предназначенных для организации и проведения спортивных мероприятий, поиска единомышленников и отслеживания активности. Однако каждое из них обладает как преимуществами, так и ограничениями, что делает актуальной разработку нового решения, сочетающего в себе функции социальной сети и платформы для организации активных игр.

В рамках анализа были рассмотрены несколько популярных мобильных приложений, таких как MeetUp, OpenSports, SportEasy, Strava и другие. Эти приложения ориентированы на различные аспекты спортивной активности, от поиска мероприятий до отслеживания личных достижений. Однако каждое из них имеет определённые ограничения, которые делают их использование неудобным

для пользователей, заинтересованных в разовых встречах, гибкости и более глубоком социальном взаимодействии.

Рассмотрим основные существующие решения:

MeetUp – это международная платформа для организации встреч по интересам, которая включает в себя различные типы мероприятий, в том числе спортивные. Пользователи могут создавать группы и находить события по интересам и локации.

Преимущества: Удобная возможность находить мероприятия по интересам, включая спортивные. Платформа широко используется в разных странах и имеет большое количество пользователей.

Ограничения: MeetUp не предоставляет специализированных инструментов для организации спортивных игр, таких как управление количеством участников, уровень подготовки и другие параметры. Также в MeetUp отсутствуют функции для отслеживания активности и спортивных достижений, что ограничивает её использование для пользователей, желающих регулярно заниматься спортом.

Интерфейс плотформы MeetUp представлен на русунке 2.1

Источник: MeetUp[1].

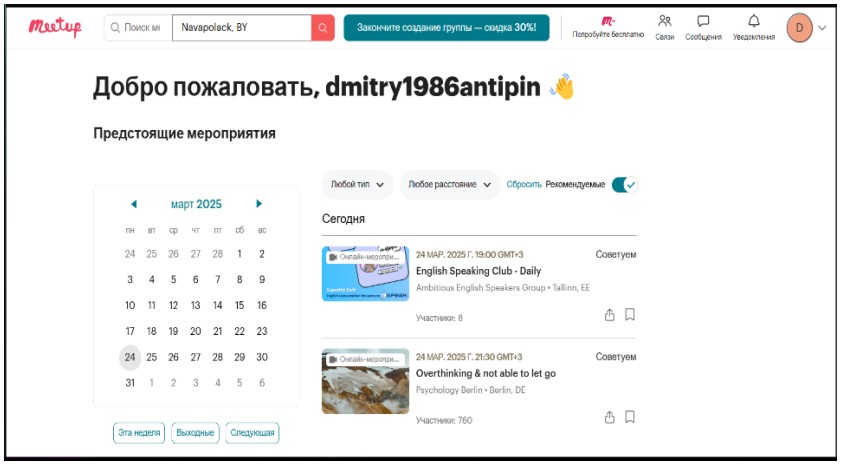


Рисунок 1.1 – Интерфейс плотформы MeetUp

OpenSports – это платформа, предназначенная для поиска и организации спортивных событий, упрощая процесс создания игр и общения между участниками. Она поддерживает различные виды спорта и мероприятия.

Преимущества: Возможность организовать спортивные события, поиска участников по интересам и геолокации. Платформа ориентирована на улучшение взаимодействия и координацию между участниками.

Ограничения: Несмотря на наличие инструментов для организации игр, OpenSports имеет ограниченную пользовательскую базу в некоторых регионах.

Это может затруднить поиск участников, особенно в менее популярных местах. Платформа не имеет полноценной социальной составляющей, что ограничивает её привлекательность для пользователей, ищущих не только спортивные мероприятия, но и возможность общения и формирования сообществ.

Интерфейс плотформы OpenSports представлен на русунке 2.2

Источник: OpenSports[2].

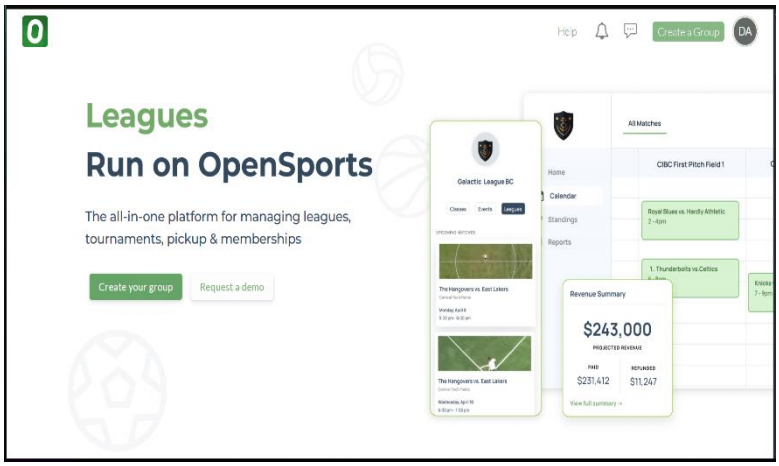


Рисунок 1.2 – Интерфейс плотформы OpenSports

SportEasy – приложение для управления спортивными командами, планирования матчей и тренировок. Оно ориентировано в первую очередь на командные виды спорта, такие как футбол, баскетбол и волейбол, и предоставляет инструменты для управления клубами и организацией матчей.

Преимущества: SportEasy обладает широкими возможностями для управления командами, включая планирование матчей, тренировок, а также коммуникацию между игроками.

Ограничения: Платформа не предоставляет удобных инструментов для поиска и организации разовых спортивных игр или встреч. Весь функционал направлен в основном на командные мероприятия и клубные активности, что ограничивает её использование для людей, желающих заниматься спортом на более гибкой основе или принимать участие в разовых играх.

Интерфейс плотформы SportEasy представлен на русунке 2.3

Источник: SportEasy[3].

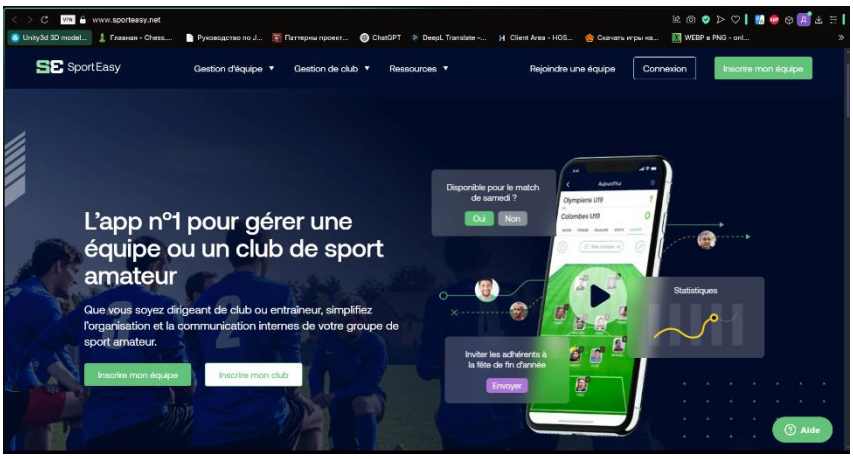


Рисунок 1.3 – Интерфейс плотформы OpenSports

Strava — популярное приложение для отслеживания индивидуальных спортивных тренировок, особенно в таких видах спорта, как бег и велоспорт. Оно также включает элементы социальной сети, позволяя пользователям делиться своими достижениями и результатами.

Преимущества: Strava известна своей возможностью отслеживания тренировок, предложением персонализированных рекомендаций и подробных статистических данных. Она поддерживает широкий спектр видов спорта и дает пользователям возможность делиться результатами с друзьями и следить за активностью других.

Ограничения: Strava не предоставляет функций для организации спортивных мероприятий или игр, что делает её неудобной для пользователей, которые хотят найти компанию для проведения спортивных игр или участие в командных мероприятиях. Платформа ориентирована в первую очередь на индивидуальные тренировки и не решает задачи организации разовых спортивных встреч.

Интерфейс плотформы Strava представлен на русунке 2.4

Источник: Strava[4].

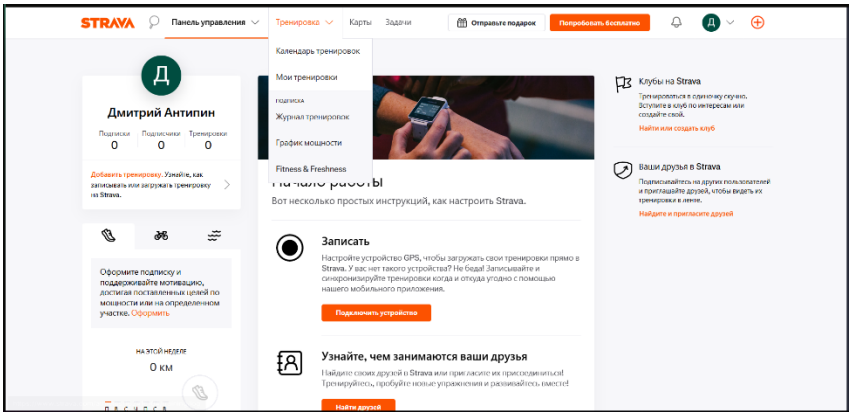


Рисунок 1.4 – Интерфейс плотформы Strava

Результаты сравнительного анализа рассматриваемых аналогов представлены в таблице 1.1.

Таблица 1.1 – Сравнительный анализ существующих аналогов

Критерий сравнения	**MeetUp**	**OpenSports**	**SportEasy**	**Strava**
Создание мероприятий	Да	Да	Да	Нет
Поиск по локации	Да	Да	Да	Ограниченно
Командные игры	Нет	Да	Да	Нет
Индивидуальные тренировки	Нет	Нет	Нет	Да
Чат и социальные функции	Ограниченно	Ограниченно	Да	Да
Персональные рекомендации	Нет	Нет	Нет	Да
Геолокация	Да	Да	Да	Да

Анализ показал, что существующие решения либо ориентированы на командные виды спорта и клубную активность (SportEasy, OpenSports), либо на индивидуальные тренировки (Strava), либо не имеют достаточного функционала для удобной организации мероприятий (MeetUp).

Разрабатываемая социальная платформа для активных игр сочетает в себе лучшие стороны представленных решений, расширяя их возможности за счет глубокой социальной интеграции, умных рекомендаций, персонализированного поиска мероприятий и удобного взаимодействия с другими пользователями.

1.4. Выбор языка разработки

Выбор языка разработки для веб-платформы на микросервисной архитектуре является одним из ключевых этапов проектирования системы. Язык разработки определяет производительность приложения, простоту масштабирования, поддержку технологий и библиотек, а также общую эффективность разработки. На момент разработки данного дипломного проекта, для создания веб-платформы, использующей микросервисную архитектуру, наиболее популярными языками программирования являются Java, Kotlin, Python, Go и Node.js. В ходе анализа был выбран язык Java. Рассмотрим подробнее, почему этот выбор был сделан.

Java – это статически типизированный объектно-ориентированный язык программирования общего назначения. Java обладает множеством библиотек, фреймворков и инструментов, которые идеально подходят для создания масштабируемых и производительных веб-приложений и микросервисных архитектур. Для реализации микросервисов Java традиционно используется с такими фреймворками, как Spring Boot, Spring Cloud, которые упрощают разработку и обслуживание микросервисов. Java имеет огромное сообщество и отличную поддержку, что также является важным фактором выбора.

Преимущества Java:

1. Масштабируемость и производительность:

Java является одним из самых производительных языков, что особенно важно для создания высоконагруженных веб-приложений и микросервисов. Благодаря JVM (Java Virtual Machine**)** Java-программы могут эффективно управлять памятью и работать с большими объемами данных.

В сочетании с инструментами для параллельной обработки, такими как Java Concurrency API, Java позволяет строить системы, которые легко масштабируются как вертикально, так и горизонтально, что является основным требованием для микросервисной архитектуры.

2. Широкая поддержка и зрелость:

Java имеет огромное сообщество разработчиков, что позволяет легко находить решения для различных проблем и получать поддержку при разработке.

Она обладает зрелой экосистемой, а ее фреймворки, такие как Spring Boot, позволяют создавать надежные и безопасные микросервисы с минимальными усилиями.

3. Интеграция с другими системами:

Java имеет богатый набор библиотек и инструментов для интеграции с различными сервисами и компонентами системы (например, базы данных, API, системы мониторинга и т. д.).

Фреймворк Spring Cloud предоставляет мощные инструменты для работы с распределенными системами, такими как конфигурационные серверы,

балансировка нагрузки**,** сервис-ориентированные архитектуры **и** обработку отказов**.**

4. Безопасность и надежность:

Java является одним из самых безопасных языков программирования, поскольку его экосистема включает проверенные механизмы защиты от уязвимостей, таких как безопасные соединения по HTTPS, шифрование данных и механизмы для защиты от атак.

Недостатки Java:

1. Более высокая сложность в сравнении с некоторыми другими языками**:**

Для новичков Java может показаться более сложной в освоении из-за строгой типизации и обширной экосистемы, требующей более глубокого понимания.

2. Больший объем кода**:**

Код на Java часто бывает более многословным, чем в некоторых других языках, таких как Kotlin или Python. Это может привести к увеличению времени на разработку и поддержание кода.

Kotlin — это относительно новый язык программирования, полностью совместимый с Java и работающий на JVM. Kotlin является более современным языком с лаконичным синтаксисом, что уменьшает количество кода и упрощает процесс разработки.

Преимущества Kotlin:

1. Лаконичность и современность**:**

Kotlin требует написания меньшего количества кода, что делает разработку более быстрой и эффективной.

Он поддерживает многие современные возможности, такие как нулевая безопасность (null safety), функциональное программирование и расширяемые функции.

2. Полная совместимость с Java:

Kotlin полностью совместим с Java, что позволяет использовать весь существующий стек технологий, библиотек и фреймворков на Java, включая Spring Boot и другие.

Это позволяет мигрировать с Java на Kotlin поэтапно без полной переработки существующего кода.

Недостатки Kotlin:

1. Меньшая зрелость:

Несмотря на свою популярность, Kotlin является относительно новым языком по сравнению с Java, что может вызывать определенные сложности в использовании в сложных или крупных проектах.

2. Ограниченная поддержка инструментов:

Некоторые инструменты и библиотеки могут не поддерживать Kotlin на том же уровне, что и Java, что может привести к небольшим проблемам с интеграцией.

Python — это язык программирования с высокой читаемостью и простотой синтаксиса, который идеально подходит для быстрого прототипирования и разработки, особенно для малых и средних проектов. Он широко используется для разработки веб-приложений и микросервисов с помощью таких фреймворков, как Django и Flask.

Преимущества Python:

1. Простота разработки:

Python позволяет быстро разрабатывать приложения благодаря лаконичному и понятному синтаксису.

Это делает его хорошим выбором для стартапов или команд, которым нужно быстро реализовать MVP (minimum viable product).

2. Большая экосистема библиотек:

Python имеет огромное количество библиотек, что упрощает разработку, тестирование и интеграцию различных сервисов и технологий.

Недостатки Python:

1. Проблемы с производительностью:

Несмотря на свою гибкость, Python имеет ограничения по производительности в сравнении с Java, особенно при работе с большими объемами данных или высокими нагрузками.

Python не так хорошо масштабируется, как Java, и может стать узким местом в системах с высокими требованиями к скорости обработки данных.

2. Глобальная блокировка интерпретатора (GIL):

Python использует механизм GIL (Global Interpreter Lock), который ограничивает параллельное выполнение многозадачных операций, что может затруднить масштабирование системы.

Go — это язык программирования, ориентированный на создание высокопроизводительных и многозадачных приложений. Он разработан для работы с распределенными системами и микросервисами и предлагает простую модель конкуренции через горутины.

Преимущества Go:

1. Производительность:

Go компилируется в машинный код, что позволяет достигать высокой производительности при обработке большого количества запросов.

2. Легкость масштабирования:

Go идеально подходит для создания распределенных систем благодаря встроенной поддержке многозадачности и лёгкости работы с параллелизмом.

Недостатки Go:

1. Отсутствие зрелых фреймворков для микросервисов:

В отличие от Java, Go не имеет такого же богатого набора фреймворков для разработки веб-приложений и микросервисов. Это может требовать дополнительных усилий для построения архитектуры.

2. Меньшая экосистема:

Хотя Go активно развивается, его экосистема всё ещё не так велика и разнообразна, как у Java или Python, что может затруднить внедрение определённых технологий.

Node.js — это серверная платформа на основе JavaScript, которая позволяет строить высокопроизводительные асинхронные приложения. Node.js идеально подходит для приложений, работающих с большим количеством одновременных соединений и требующих асинхронной обработки.

Преимущества Node.js:

1. Асинхронная обработка запросов:

Node.js превосходно работает с I/O операциями и асинхронными запросами, что делает его идеальным для веб-приложений, где важно обрабатывать множество параллельных запросов.

2. Единая технология для клиента и сервера:

Использование JavaScript как на сервере, так и на клиенте упрощает разработку и поддержку приложения.

Недостатки Node.js:

1. Сложности при масштабировании:

Для крупных приложений Node.js может потребовать значительных усилий по поддержке и масштабированию, особенно при необходимости работы с более сложными сервисами и интеграциями.

2. Ограниченная производительность в некоторых задачах:

Хотя Node.js хорошо справляется с I/O операциями, для вычислительно интенсивных задач его производительность может уступать Java и Go.

Результаты сравнительного анализа рассматриваемых языков программирования представлены в таблице 1.2, которая была составлена на основе информации из следующих источников: Java[5], Kotlin[6], Python[7], Go[8], Node.js[9].

Таблица 1.2 – Сравнительный анализ языков программирования

Критерий сравнения	**Java**	**Kotlin**	**Python**	**Go**	**Node.js**
Производительность	Высокая	Высокая	Средняя	Высокая	Средняя
Масштабируемость	Отличная	Отличная	Средняя	Хорошая	Средняя
Поддержка микросервисов	Отличная	Отличная	Хорошая	Хорошая	Хорошая
Обширность экосистемы	Огромная	Огромная	Средняя	Средняя	Большая
Сообщество и поддержка	Отличная	Отличная	Средняя	Растущее	Огромное
Простота разработки	Средняя	Легче, чем Java	Высокая	Средняя	Высокая

Исходя из анализа всех доступных вариантов для разработки микросервисной платформы, выбор языка программирования сводится к учету нескольких факторов: зрелости языка, поддержки микросервисной архитектуры, производительности, экосистемы инструментов и личного опыта разработчика. Все эти критерии в полной мере удовлетворяет язык Java.

Прежде всего, Java является зрелым и стабильным языком, идеально подходящим для разработки сложных и высоконагруженных распределенных систем, таких как микросервисные платформы. Язык предоставляет все необходимые инструменты для построения эффективных и масштабируемых систем, что крайне важно для успешной реализации микросервисной архитектуры. Вдобавок, с помощью таких фреймворков, как Spring Boot, Spring Cloud и Hibernate, процесс разработки, тестирования и внедрения микросервисов становится значительно проще и быстрее.

К тому же Java имеет большую и активную экосистему, которая включает в себя мощные инструменты для интеграции с различными сервисами и компонентами, а также поддержки безопасности и обработки данных. Это позволяет эффективно решать любые задачи, возникающие в процессе разработки, и ускоряет процесс разработки системы в целом.

Личный опыт работы с Java также сыграл немаловажную роль в принятии решения. За время обучения было накоплено достаточное количество знаний и навыков для работы с этим языком, что позволяет уверенно и эффективно разрабатывать системы на его основе. Поддержка Java в большинстве современных сред разработки также делает процесс разработки удобным и быстрым, что влияет на скорость и качество работы.

Таким образом, после анализа всех факторов и учитывая опыт разработки на Java, был выбран именно этот язык для реализации микросервисной архитектуры веб-платформы.

1.5. Выбор среды разработки

Для разработки микросервисной веб-платформы на языке Java были рассмотрены следующие среды разработки: IntelliJ IDEA, NetBeans, Eclipse и Visual Studio Code. Рассмотрим каждую из них более подробно:

IntelliJ IDEA – мощная интегрированная среда разработки, ориентированная на Java и другие популярные языки программирования. Она предоставляет множество инструментов для работы с микросервисами, поддерживает работу с фреймворками Spring Boot, Spring Cloud и Hibernate, а также интеграцию с системами контейнеризации, такими как Docker и Kubernetes. Среда включает в себя встроенные инструменты для тестирования, отладки и анализа производительности.

NetBeans – свободная среда разработки, поддерживающая множество языков программирования, включая Java. NetBeans хорошо подходит для разработки Java-приложений, но она не имеет такого же уровня поддержки для фреймворков, как IntelliJ IDEA, и её инструменты для работы с микросервисами и контейнерами менее развиты. Также NetBeans не обладает такой гибкостью и производительностью, как IntelliJ IDEA.

Eclipse – популярная среда разработки с открытым исходным кодом. Eclipse предоставляет широкие возможности для разработки на Java и интеграции с различными фреймворками, но она требует дополнительной настройки для работы с микросервисами и контейнерами. Для полноценной работы с такими фреймворками, как Spring Boot и Docker, необходимы дополнительные плагины, что делает использование Eclipse более трудозатратным.

Visual Studio Code (VS Code) – легкая и универсальная редактор с поддержкой множества языков программирования через расширения. VS Code идеально подходит для разработки фронтенд-части и небольших проектов, но не предоставляет такого же уровня функциональности для разработки крупных микросервисных приложений, как специализированные IDE

Сравнительный анализ рассматриваемых сред разработки представлен в таблице 1.3. Для сбора данных о функционале и особенностях различных сред разработки были использованы следующие источники: официальные сайты IntelliJ IDEA[10], NetBeans[11], Eclipse[12] и Visual Studio Code[13]. Эти ресурсы предоставляют актуальную информацию о возможностях каждой из сред и их характеристиках.

Таблица 1.3 – Сравнительный анализ сред разработки

Критерий сравнения	**IntelliJ IDEA**	**NetBeans**	**Eclipse**	**Visual Studio Code**
Оценка функционала по пятибалльной шкале	5	4	4	3
Интеграция с Git	Да	Да	Да	Да
Доступность	Условно бесплатно	Бесплатно	Бесплатно	Бесплатно
Необходимость установки плагина	Нет	Нет	Да	Да

Выбор среды разработки IntelliJ IDEA был обусловлен её возможностями для разработки микросервисных веб-платформ, поддержкой популярных фреймворков и инструментов, а также удобством работы и наличием всех необходимых функций для эффективного выполнения проекта.

1.6. Выбор инструмента проектирования

При выборе инструмента проектирования необходимо учитывать ряд требований, которые обеспечивают удобство моделирования, визуализации и документирования архитектуры системы. Корректно подобранное программное обеспечение позволяет сократить время разработки, улучшить взаимодействие между членами команды и повысить качество итогового решения.

UML – язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур. Не смотря на широкий диапазон использования данный язык был создан для определения, визуализации, проектирования и документирования в основном программных систем. Использование UML диаграмм при проектировании программного продукта позволяет определить четкую структуру будущей программы и тем самым значительно ускорить процесс написания кода в частности и упростить разработку в целом.

Используемый инструмент проектирования должен удовлетворять следующим требованиям:

– работу с UML-диаграммами – возможность создавать основные типы UML-диаграмм, включая диаграммы классов, последовательности, компонентов, развертывания и состояний;

– гибкость редактирования – настройку структуры диаграмм, изменение шрифтов, цветов, связей и других параметров визуального представления;

– поддержку моделирования баз данных – возможность проектирования ER-диаграмм и генерации SQL-кода;

– генерацию кода на основе диаграмм – создание каркаса программного кода из диаграмм классов для ускорения разработки.по возможности обладать интуитивно понятным и удобным интерфейсом;

– экспорт диаграмм – поддержку популярных графических форматов (PNG, SVG, PDF) и интеграцию с другими инструментами;

– совместную работу – облачное хранение или интеграция с системами управления проектами для совместного редактирования и комментирования моделей.

В соответствии с вышеперечисленными требования в качестве инструмента проектирования было выбрано программное обеспечение Enterprise Architect.

Enterprise Architect – это инструмент визуального моделирования и дизайна, основанный на UML. Платформа поддерживает проектирование и построение программных систем; моделирование бизнес-процессов; и предметные области, основанные на модельной индустрии. В качестве инструменты проектирования был выбран по причине полного соответствия поставленным требованиям.

1.7. Разработка технического задания

Техническое задание (ТЗ) – первичный документ на проектирование технического объекта, в данном случае дипломного проекта. Техническое задание устанавливает основное назначение разрабатываемого продукта, его технические характеристики и технико-экономические требования, предписание по выполнению необходимых стадий создания документации и ее состав, а также специальные требования. Иными словами, ТЗ является документом, позволяющим как разработчику, так и заказчику предоставить итоговый продукт и в дальнейшем выполнить проверку на соответствие предъявленным требованиям.

Оформление и составление ТЗ выполнено согласно ГОСТ 19.201-78.

Техническое задание дипломного проекта приведено в приложении А.

2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 Функциональная структура приложения

Программный продукт представляет собой многофункциональную социальную платформу для организации спортивных мероприятий и взаимодействия между пользователями. Он реализован как RESTful API, построен на Spring Boot и использует микросервисную архитектуру, что обеспечивает масштабируемость, отказоустойчивость и легкость сопровождения.

Интерфейс и внутренняя структура приложения спроектированы таким образом, чтобы обеспечить простоту, доступность и эффективность взаимодействия пользователя с системой. Основные функциональные модули представлены ниже:

Аутентификация и безопасность.

Вход в систему защищён многоступенчатой системой безопасности. При регистрации пользователю предлагается указать email, номер телефона и пароль. После проверки уникальности данных отправляется верификационный код (по email или SMS), который необходимо ввести для завершения регистрации. Система поддерживает двухфакторную аутентификацию (2FA), использование JWT-токенов и механизм обновления (refresh token), что гарантирует как безопасность, так и удобство при повторном входе. При утере доступа возможна быстрая процедура восстановления пароля через одноразовый код с ограниченным сроком действия (15 минут).

Права доступа реализованы через распределение ролей:

– USER – обычный пользователь с возможностью создавать события, участвовать в мероприятиях и вести переписку;

– ADMIN – администратор системы с расширенными правами (блокировка пользователей, модерация контента и событий).

Профиль и взаимодействие.

Каждый пользователь имеет собственный профиль, содержащий личную информацию: имя, город, любимые виды спорта и краткую статистику активности (количество созданных и посещённых мероприятий, рейтинг). Дополнительно доступна фотогалерея — можно загрузить до 50 изображений, которые автоматически сжимаются и отображаются в виде миниатюр. Это обеспечивает визуальную привлекательность профиля при сохранении оптимального использования ресурсов.

Поиск и социальное взаимодействие.

Система предоставляет удобный механизм поиска пользователей по нескольким параметрам: геолокации (город, регион), интересующему виду спорта (футбол, велоспорт, бег и т.д.), а также рейтингу активности. Пользователи могут отправлять друг другу запросы в друзья, формировать собственный круг общения и при необходимости блокировать нежелательные

контакты (черный список исключает возможность отправки сообщений и приглашений).

События и мероприятия.

Одной из ключевых функций платформы является создание и участие в спортивных мероприятиях. Пользователь может задать название, дату, вид спорта, описание события и количество участников. Приглашения на мероприятие рассылаются по email или через уникальную ссылку. После создания события автоматически создается групповой чат, позволяющий участникам координировать действия и обсуждать детали встречи.

Общение и чаты.

Платформа предлагает два типа чатов: приватные (между двумя пользователями) и групповые (до 100 участников). В чатах можно отправлять текстовые сообщения, эмодзи, а также прикреплять файлы (изображения и PDF до10МБ).

Общение реализовано в режиме реального времени с использованием WebSocket и протокола STOMP. Это обеспечивает мгновенную доставку сообщений и высокую интерактивность общения.

Уведомления.

Для поддержания активности и информирования пользователей внедрена гибкая система уведомлений.

Платформа уведомляет о важных событиях, таких как:

– Системные действия - подтверждение регистрации, смена пароля, успешная авторизация;

– Изменения в мероприятиях - напоминания, изменение даты или времени;

Файловое хранилище

Все медиафайлы хранятся в безопасном локальном хранилище. Система поддерживает хранение изображений с прикреплёнными метаданными (тип, размер, ID владельца и дата загрузки).

Доступ к файлам реализован через подписанные временные ссылки (signed URLs), а передача данных защищена с использованием протокола (TLS 1.3).

Система рейтинга

Каждому пользователю начисляются баллы за активность: создание мероприятий, участие в событиях, оставление отзывов. Накопленные баллы влияют на позицию в поиске и отображаются в профиле, создавая здоровую соревновательную атмосферу внутри сообщества.

2.2 Проектирование диаграммы развёртывания

Диаграмма развёртывания (Deployment Diagram) описывает физическую архитектуру программного обеспечения, а именно, как программные компоненты системы размещаются на физических или виртуальных узлах (серверы, базы данных, клиентские устройства и др.). Такая диаграмма особенно важна при проектировании распределённых и микросервисных архитектур.

Диаграмма развёртывания разработанного приложения отражает архитектуру веб-платформы, построенной на основе клиент-серверной модели. Все компоненты логически распределены между несколькими узлами, включая веб-клиент, сервер приложений, микросервисы и базы данных.

Архитектура развертывания включает следующие элементы:

1. WebClient:

- Представляет собой пользовательский интерфейс, доступный через браузер;
- Взаимодействует с веб-сервером по протоколу HTTPS;
- Запускается на устройствах конечных пользователей (ПК, смартфоны).

2. WebServer:

- ОС: Windows Server 2011;
- Веб-сервер: Apache Tomcat;
- Принимает HTTP/HTTPS-запросы от клиента и перенаправляет их в соответствующий микросервис;
- Обеспечивает маршрутизацию, безопасность и обработку запросов.

3. EurekaServer:

- Является сервисом обнаружения (Service Discovery) в экосистеме микросервисов;
- Отслеживает доступность и адреса активных микросервисов;

4. UserServiceApplication:

Микросервис, отвечающий за функциональность пользователя: чат, профиль, мероприятия и др.

Содержит:

- Контроллеры (ChatController, EventController и др.);
- Сервисы (ChatService, ProfileService и др.);
- Репозитории (UserRepository, ChatRepository и др.);
- Сущности (Entity User, Event, Chat и др.);
- База данных (UserDB), связанная через ORM/DAO-слой:

5. AuthenticationServiceApplication

Отвечает за регистрацию, авторизацию, восстановление пароля, хранение статусов и ролей.

Содержит:

- AuthenticationController;

- AuthenticationService, TokenService, UserService (auth);
- Репозитории (UserRepository, RoleRepository и др.);
- Сущности: User, Role, Status, VerificationCode;
- База данных AuthenticationDB.

Взаимодействие между компонентами:

- WebClient взаимодействует с WebServer (Tomcat) по протоколу HTTPS;
- WebServer направляет запросы к соответствующему микросервису (UserService или AuthenticationService);
- Оба микросервиса взаимодействуют с EurekaServer, чтобы регистрировать и обнаруживать сервисы;
- Каждый микросервис использует свой отдельный модуль данных (базу данных), связанный с ним через DAO/ORM слой.

Диаграмма развёртывания использования разрабатываемого приложения представлена в приложении Б.

2.3 Проектирование диаграммы компонентов

Диаграмма компонентов предназначена для визуализации архитектуры системы на высоком уровне абстракции. Она позволяет отразить структуру программных и сервисных элементов, а также связи и зависимости между ними. В контексте разрабатываемого веб-приложения — социальной платформы для организации активных игр — диаграмма компонентов играет ключевую роль в представлении микросервисной архитектуры системы.

Компоненты в данной системе представляют собой изолированные модули, каждый из которых отвечает за строго определённый набор функций. Взаимодействие между компонентами осуществляется по протоколу HTTP через REST API, а маршрутизация всех внешних запросов реализуется через централизованный компонент — API Gateway. Все микросервисы регистрируются и обнаруживаются посредством Eureka Server.

Диаграмма компонентов обеспечивает:

- наглядное представление внутренней архитектуры микросервисной системы;
- понимание распределения ответственности между сервисами;
- выявление зависимостей между модулями;
- основу для масштабирования и модульного тестирования компонентов.

В проекте используются следующие компоненты:

- AuthentificationService - управление регистрацией, аутентификацией пользователя;

– UserService – оздание профиля пользователя, редактирование и удаление мероприятий, управление участием, поддержка групповых и приватных чатов, отправка сообщений, изображений и файлов, генерация и отправка системных уведомлений пользователям (email, push);

– AnalyticsService — сбор и анализ статистики использования платформы;

– ApiGatewayService — единая точка входа, маршрутизация запросов ко всем микросервисам;

– EurekaServer — служба регистрации и обнаружения микросервисов в системе.

Каждый компонент инкапсулирует внутреннюю логику и взаимодействует с остальными через интерфейсы. Диаграмма демонстрирует зависимости между сервисами, при этом предоставляемые и требуемые интерфейсы отражаются в виде направленных связей, указывающих на клиент-серверные отношения между компонентами.

Такой подход позволяет ясно представить структуру системы, обеспечить её модульность и подготовить архитектуру к возможному масштабированию в будущем.

Диаграмма компонентов разрабатываемого приложения представлена в приложении В.

1.4 Проектирование диаграммы деятельности

Диаграммы деятельности используются для графического представления последовательности действий, выполняемых пользователями и системой в рамках определённого бизнес-процесса. Такие диаграммы позволяют наглядно отразить логику работы функциональных модулей и упрощают понимание алгоритмов взаимодействия между компонентами приложения.

В контексте разрабатываемой социальной платформы для организации активных игр диаграммы деятельности отображают ключевые процессы: регистрацию и авторизацию пользователей, создание и участие в мероприятиях, а также использование чатов и редактирование профиля.

Процесс регистрации и авторизации пользователя:

- Гость открывает страницу входа/регистрации;
- Переходит на страницу регистрации;
- Вводит email и пароль;
- Получает 6-значный код подтверждения;
- Вводит код;
- Если код верен — регистрация завершается успешно, пользователь перенаправляется на главную страницу;
- При авторизации пользователь вводит email и пароль;

– Получает 6-значный код верификации (при включенной двухфакторной аутентификации);
– Вводит код;
– Если код верен — пользователь попадает в приложение.

Главная страница приложения:
– Доступны разделы: «События», «Профиль», «Мессенджер».

Процесс создания и участия в мероприятии:
– Пользователь открывает раздел «События»;
– Нажимает кнопку «Создать своё событие?»;
– Вводит необходимые данные;
– Если данные корректны, происходит сохранение события;
– В случае ошибки — отображается сообщение об ошибке;
– Успешное событие отображается в списке активных;
– Пользователь может добавить участников;
– После успешного добавления участников создаётся групповой чат.

Процесс взаимодействия в чате:
– Пользователь открывает раздел «Мессенджер»;
– Нажимает «Создать чат?» и выбирает пользователя;
– Вводит название чата;
– Чат создаётся, пользователь может написать сообщение;
– Сообщение может содержать текст или файлы;
– Также возможна последующая редакция сообщения.

Редактирование профиля:
– Переход в раздел «Профиль»;
– Нажатие кнопки «Редактировать профиль»;
– Ввод новых данных;
– Выбор и загрузка изображения (аватар);
– Изменённые данные сохраняются, профиль обновляется.

Диаграммы деятельности отражают не только взаимодействие актёров с системой, но и логику обработки действий внутри каждого модуля. Они помогают формализовать бизнес-процессы, выявить узкие места и обеспечить основу для будущей автоматизации тестирования или оптимизации архитектуры приложения.

Диаграммы деятельности вышеуказанных процессов представлены в приложении Г.

1.5 Проектирование структуры базы данных

Для реализации микросервисной архитектуры социальной платформы для организации активных игр была разработана реляционная структура базы данных, обеспечивающая хранение и эффективную обработку информации о

пользователях, событиях, геолокациях, сообщениях и рекомендациях. В качестве системы управления базами данных выбрана PostgreSQL — надежная объектно-реляционная СУБД с поддержкой транзакций, ACID-свойств, мощного языка запросов SQL и возможностью масштабирования.

Структура базы данных проектировалась с учетом нормализации, обеспечения целостности данных, а также удобства взаимодействия с различными микросервисами системы. Для этого была спроектирована ER-диаграмма, на которой отражены основные сущности (таблицы) и связи между ними.

Логическая структура базы данных преставлена на ER-диаграмме в приложении Д.

Основные сущности

User (пользователь)

Представляет зарегистрированного пользователя платформы. Содержит основные идентификационные и контактные данные:

- user_id – уникальный идентификатор;
- phone_number, email, username, password – данные для регистрации и входа;
- status – статус активности аккаунта (активен, заблокирован и т.д.).

User_Profile (профиль пользователя)

Дополняет таблицу пользователей информацией о внешнем виде и активности:

- first_name, last_name, birth_date, gender;
- photo_url, location, bio.

User_Photos (фотографии пользователя)

Хранит список изображений, загруженных пользователем. Связана с таблицей User через внешний ключ.

Friends (друзья)

Содержит пары пользователей, находящихся в списке друзей. Реализует двустороннюю связь через поля user_id_1 и user_id_2.

События и участие

Event (мероприятие)

Основная таблица, описывающая событие (игру, встречу):

- event_id, name, description, start_time, operator_id;
- location_id, event_status_id.

Event_Participants (участники событий)

Промежуточная таблица, связывающая пользователей и события (многие ко многим):

- event_id, user_id, joined_at, is_admin.

Event_Categories

Справочник категорий событий (футбол, волейбол, настольные игры и т.д.).

Event_Statuses

Содержит возможные статусы мероприятия (запланировано, завершено, отменено).

Геолокация и инфраструктура

Locations (локации)

Хранит информацию о местах проведения мероприятий:

- location_id, address, latitude, longitude, description.

Location_Types

Типы локаций (стадион, спортзал, площадка на открытом воздухе и т.д.).

Infrastructure_Types / Units

Описание инфраструктурных объектов (душ, раздевалка) и их наличие в конкретной локации.

Чаты и сообщения

Chats

Групповые чаты между участниками событий или пользователями.

Messages

Хранит все сообщения, отправленные в чатах:

- message_id, chat_id, sender_id, text, status, timestamp.

Message_Attachments

Файлы, прикрепленные к сообщениям.

Уведомления и рекомендации

Notifications

Содержит уведомления, направляемые пользователям системой:

- notification_id, user_id, type, message, timestamp.

Recommendations

Механизм рекомендаций событий или пользователей на основе активности:

- user_id, recommended_user_id, score.

Оптимизация и ограничения

- Индексация: Для ускорения запросов созданы индексы на ключевые поля, такие как user_id, event_id, location_id, email, username, что позволяет обеспечить высокую производительность при большом объеме данных.
- Целостность данных: Связи между таблицами реализованы через внешние ключи, что гарантирует соблюдение ссылочной целостности на уровне СУБД.
- Нормализация: Структура базы данных спроектирована в соответствии с нормальными формами (вплоть до 3NF), что исключает избыточность данных и повышает их согласованность.
- Масштабируемость: Структура позволяет легко добавлять новые таблицы и связи (например, для поддержки новых типов событий, ролей пользователей, рейтингов и т.д.).
-

ЗАКЛЮЧЕНИЕ

В процессе прохождения преддипломной практики была изучена история становления и развития компании «Астон Софт», а также её организационная структура. Кроме этого, было рассмотрена нормативная документация подразделения, информация о программном обеспечении, используемом в подразделении.

Помимо достижения вышеперечисленных целей была проведена разработка первого раздела дипломного проекта, включающая в себя анализ исходных данных и предметной области. Проведено рассмотрение и сравнительный анализ существующих прототипов и аналогов, выбор используемых языка программирования, среды разработки и инструмента проектирования. Также было разработано и оформлено техническое задание дипломного проекта.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. MeetUp [Электронный ресурс]. – Режим доступа: https://www.meetup.com – Дата обращения: 24.03.2025.
2. OpenSports [Электронный ресурс]. – Режим доступа: https://opensports.net/ – Дата обращения: 24.03.2025.
3. SportEasy [Электронный ресурс]. – Режим доступа: https://www.sporteasy.net/ – Дата обращения: 24.03.2025.
4. Strava [Электронный ресурс]. – Режим доступа: https://www.strava.com/ – Дата обращения: 24.03.2025.
5. Java [Электронный ресурс] / Википедия. – Режим доступа: https://ru.wikipedia.org/wiki/Java – Дата доступа: 11.02.2025.
6. Kotlin [Электронный ресурс] / Википедия. – Режим доступа: https://ru.wikipedia.org/wiki/Kotlin – Дата доступа: 14.02.2025.
7. Python [Электронный ресурс] / Википедия. – Режим доступа: https://ru.wikipedia.org/wiki/Python – Дата доступа: 16.02.2025.
8. Go [Электронный ресурс] / Википедия. – Режим доступа: https://ru.wikipedia.org/wiki/Go – Дата доступа: 18.02.2025
9. Node.js [Электронный ресурс] / Википедия. – Режим доступа: https://ru.wikipedia.org/wiki/Node.js – Дата доступа: 20.03.2025.
10. IntelliJ IDEA [Электронный ресурс] / Википедия. – Режим доступа: https://ru.wikipedia.org/wiki/IntelliJ_IDEA – Дата доступа: 12.03.2025
11. NetBeans [Электронный ресурс] / Википедия. – Режим доступа: https://ru.wikipedia.org/wiki/NetBeans – Дата доступа: 25.02.2025
12. Eclipse [Электронный ресурс] / Википедия. – Режим доступа: https://en.wikipedia.org/wiki/Eclipse_(software) – Дата доступа: 02.03.2025
13. Visual Studio Code [Электронный ресурс] / Википедия. – Режим доступа: https://ru.wikipedia.org/wiki/Visual_Studio_Code – Дата доступа: 23.03.2025.
14. Enterprise Architect (software) [Электронный ресурс] / Википедия. – Режим доступа: https://en.wikipedia.org/wiki/Enterprise_Architect_(software) – Дата доступа: 20.03.2025

ПРИЛОЖЕНИЕ А
(обязательное)

Техническое задание

ВВЕДЕНИЕ

Название дипломного проекта – «Разработка серверной части социальной платформы для организации активных игр».

Программа предназначена для организации и проведения активных мероприятий, объединяя пользователей, желающих участвовать в спортивных и игровых событиях. Платформа позволяет создавать и управлять мероприятиями, находить участников, получать рекомендации и анализировать активность пользователей.

Платформа будет служить инструментом для организации активных игр и спортивных мероприятий, предоставляя пользователям возможность не только участвовать, но и влиять на события, а также отслеживать свою активность и достижения.

А.1 Назначение разработки

Целью разработки является создание программного комплекса, включающего веб-платформу для автоматизированного взаимодействия пользователей при организации и участии в активных играх. Данный программный комплекс позволит пользователям легко находить интересные мероприятия, создавать собственные события, взаимодействовать с другими участниками, общаться в чате, управлять своим профилем и получать персонализированные рекомендации на основе их предпочтений и активности.

Разрабатываемая система предназначена для использования в сфере организации активного отдыха, спортивных мероприятий и командных игр. Она будет полезна как для отдельных пользователей, ищущих события для участия, так и для организаторов, заинтересованных в привлечении участников

А.2 Требования к программе или программному изделию

Разрабатываемый программный комплекс включает веб-платформу для организации и участия в активных играх. Ниже перечислены основные требования к системе, которые охватывают функциональные, технические и эксплуатационные аспекты.

А.2.1 Требования к функциональным характеристикам

Функциональные требования определяют ключевые возможности системы и её компонентов. Серверная часть социальной платформы для организации активных игр реализована с использованием микросервисной архитектуры. Каждый микросервис отвечает за свою бизнес-область и взаимодействует с другими компонентами через REST API, а также асинхронные каналы обмена сообщениями (WebSocket). Для хранения данных используется СУБД PostgreSQL, где каждый сервис работает либо со своей базой данных. Доступ к данным осуществляется с помощью JPA/Hibernate, а для преобразования между сущностями и DTO применяется MapStruct.

AuthService отвечает за процессы аутентификации и авторизации пользователей. При входе в систему пользователь передаёт свои учетные данные (email и пароль), после чего сервис проверяет их корректность и в случае успеха выдает JWT-токен, содержащий ID пользователя и его роли. Токен передаётся в заголовке Authorization при обращении к защищённым API других сервисов. Сервис также реализует:

- проверку ролей пользователя (USER, ADMIN),
- поддержку refresh-токенов для продления сессии,
- повторную авторизацию без повторного ввода пароля.

UserService реализует управление пользовательскими данными и основную социальную функциональность платформы. Он предоставляет возможности для:

- регистрации пользователей: при регистрации указываются имя, email и пароль. Пароль хешируется, а на email отправляется шестизначный код подтверждения. Пока пользователь не подтвердит регистрацию, его профиль считается неактивным. После успешного подтверждения поле verified в базе данных устанавливается в true.
- авторизации: после подтверждения аккаунта пользователь может авторизоваться, получая JWT-токен.
- ведения профиля: профиль включает имя, фамилию, аватар, интересы, виды спорта, уровень подготовки, возраст и другие параметры. Поддерживается загрузка изображений на сервер.

– дружбы и социальных связей: реализованы механизмы добавления в друзья, подтверждения и отклонения заявок, удаления из списка друзей. Информация о статусе заявок хранится в таблице User_relationships.

Помимо этого, UserService отвечает за функциональность, связанную с мероприятиями:

– создание, редактирование и удаление мероприятий: организатор указывает название, описание, место, дату, тип игры, возрастные и физические ограничения, количество участников и может загрузить баннер.

– участие в мероприятиях: пользователи могут присоединяться к мероприятиям при наличии свободных мест. Участники добавляются в таблицу event_participants, и получают уведомления об изменениях.

Также реализована система обмена сообщениями:

– личные чаты между друзьями, работающие через WebSocket или REST с polling,

– групповые чаты, создаваемые автоматически при мероприятии с несколькими участниками. Все сообщения сохраняются с указанием отправителя, времени и принадлежности к чату или мероприятию.

AnalyticsService занимается сбором, обработкой и анализом данных, генерируемых пользователями и событиями в системе. Он предоставляет возможность:

– вести учёт активности пользователей (регистрации, участие в мероприятиях, взаимодействия в чатах),

– формировать отчёты по наиболее популярным видам спорта и т.д.,

– отслеживать и визуализировать ключевые метрики системы.

Для обеспечения масштабируемости и изоляции микросервисы используют отдельные схемы или базы данных в PostgreSQL. Регистрация и обнаружение сервисов осуществляется через Eureka, а маршрутизация запросов — через API Gateway. Коммуникация между сервисами происходит преимущественно по REST.

А.2.2 Требования к надёжности

В системе предусмотрены строгие механизмы контроля как для входных, так и для выходных данных, которые передаются через API и хранятся в базе данных.

Контроль входных данных на уровне API

1. Валидация запроса:

– Каждый входящий запрос проходит проверку на уровне контроллеров. Структура запроса, наличие обязательных параметров и их типы валидируются заранее. Например, при регистрации нового пользователя API

ожидает получить JSON с полями, такими как имя пользователя, email, пароль и другие обязательные данные.

– Если параметры не соответствуют ожидаемому формату (например, неправильный email или невалидный пароль), API возвращает ошибку с кодом 400 и соответствующим сообщением.

2. Обработка лишних полей:

– API строго фильтрует лишние поля, которые не предусмотрены для данного запроса. Если в запросе присутствуют дополнительные данные, которые не ожидаются системой, они либо игнорируются, либо вызывают ошибку в зависимости от настроек сериализации.

3. Формат данных:

– Ответ API всегда возвращается в унифицированном формате JSON с полями: status ("success" или "error") и data или message (в зависимости от результата запроса).

– Все временные метки имеют формат ISO 8601 с указанием временной зоны, что позволяет поддерживать единообразие при работе с временем в разных часовых поясах.

4. Безопасность и аутентификация:

– API использует JWT-токены для аутентификации. После успешного логина пользователь получает токен, который необходимо передавать в заголовке каждого запроса.

– Все чувствительные операции доступны только авторизованным пользователям. Запросы без действительного токена блокируются с кодами ответа 401 (Unauthorized) или 403 (Forbidden).

– Уровни доступа для разных пользователей (например, обычные пользователи, администраторы, модераторы) контролируются на уровне API.

5. Обработка ошибок:

– Ошибки сопровождаются кодом ответа, текстом описания и уникальным внутренним кодом ошибки, который помогает в анализе проблемы на клиентской стороне.

– Все запросы логируются с указанием идентификатора пользователя, URL, времени запроса и результата (успех или ошибка).

6. Защита от атак:

– Используются стандартные механизмы защиты от SQL-инъекций, XSS, CSRF и других угроз с помощью встроенных фильтров и дополнительных фильтров безопасности.

Контроль входных данных на уровне базы данных

1. Регистрация пользователя:

Каждое поле, передаваемое через API при регистрации нового пользователя, подвергается валидации:

– Идентификатор пользователя: должен быть уникальным и числовым или в формате UUID.

– Имя пользователя: проверяется на отсутствие специальных символов и пробелов.

– Email: проверяется через регулярное выражение и на уникальность.

– Пароль: должен соответствовать установленной политике безопасности (длина, наличие символов, цифр и т.д.).

– Идентификаторы роли и статуса: должны ссылаться на существующие записи в таблицах ролей и статусов.

2. Профиль пользователя:

– В таблице User_profiles проверяются поля first_name и last_name на соответствие требованиям (буквенные символы, ограничение по длине, отсутствие пустых значений).

– Дата рождения проверяется на соответствие формату ISO 8601 и на логику (не может быть в будущем).

3. Фото пользователя:

– В таблице User_photos проверяются URL фотографий, которые должны быть валидными интернет-адресами с расширениями .jpg или .png.

4. Активности пользователя:

– В таблице User_activities проверяется тип активности а также соответствие временной метки формату ISO и часового пояса.

5. Отношения пользователей:

– В таблице User_relationships проверяется наличие двух различных идентификаторов пользователей и их тип взаимоотношений. Недопустимы дублирующие или рекурсивные записи (пользователь не может указать себя в качестве второго участника).

6. События:

– В таблице Events проверяется валидность идентификатора создателя события, название события, дату и время начала. Дата начала события должна быть позже текущего времени.

7. Участники событий:

– В таблице event_participants проверяется, что идентификаторы события и пользователя существуют в системе. Запрещено добавление одного и того же пользователя к одному событию несколько раз.

8. Статусы событий:

– В таблице Event_statuses проверяется, что название статуса уникально и содержит только допустимые символы.

9. Чаты и сообщения:

– В таблице Chats проверяется, что чат существует, а в Messages — что отправитель является участником чата, и что текст сообщения не пустой и соответствует ограничениям по длине.

– Вложения проверяются на допустимый MIME-тип, размер и доступность по сети.

10. Рекомендации и уведомления:

– В таблицах Recommendations и Notifications проверяются ссылки на пользователей, корректность временных меток и уникальность рекомендаций.

11. Коды верификации:

– В таблице Verification_code проверяется срок действия кода, который не должен превышать допустимые интервалы (10 минут). Повторная генерация возможна только после истечения предыдущего кода.

Контроль выходных данных на уровне API и базы данных

1. Сериализация и форматирование:

– Все выходные данные проходят через слой сериализации, который фильтрует лишние поля, такие как хеши паролей и внутренние ID ролей, и форматирует данные в удобочитаемый вид.

– Для часто запрашиваемых данных может использоваться кэширование, однако оно всегда валидируется при изменениях.

2. Целостность данных:

– Каждая транзакция с данными (например, добавление участника в событие или отправка сообщения) либо завершается полностью, либо откатывается в случае ошибки. Это обеспечивается с использованием транзакционных механизмов СУБД и ORM.

3. Логирование запросов:

– Каждое обращение к API записывается с указанием времени, IP-адреса клиента, HTTP-метода, конечной точки, идентификатора пользователя и результата запроса (успех или ошибка).

– Логи хранятся централизованно и могут быть использованы для анализа поведения системы и расследования инцидентов.

Таким образом, система обеспечивает строгий контроль за всеми данными, проходящими через API и хранящимися в базе данных, гарантируя безопасность, целостность и консистентность информации на каждом этапе обработки.

А.2.3 Условия эксплуатации

Программный комплекс разрабатывается для работы в среде, обеспечивающей стабильное сетевое подключение и поддерживающей современные технологии веб- и мобильной разработки. Условия эксплуатации включают аппаратные, программные и сетевые требования, а также требования к безопасности и масштабируемости.

1. Аппаратные требования

Для работы различных компонентов системы требуются соответствующие вычислительные мощности и ресурсы.

1.1 Серверная инфраструктура

Минимальная конфигурация сервера:

– процессор: 4 ядра, 3.0 ГГц;

– оперативная память: 8 ГБ;
– хранилище: 100 ГБ SSD;
– сетевое подключение: 100 Мбит/с.

Рекомендуемая конфигурация для высокой нагрузки:

– процессор: 8–16 ядер, 3.5 ГГц;
– оперативная память: 32–64 ГБ;
– хранилище: 500 ГБ–1 ТБ SSD с возможностью резервного копирования;
– сетевое подключение: 1 Гбит/с.

1.2 Клиентские устройства

Веб-версия:

– браузеры**:** Google Chrome (последние 2 версии), Mozilla Firefox, Microsoft Edge, Safari;
– разрешение экрана: от 1280×720 и выше.

2. Программные требования

2.1 Серверное программное обеспечение

– операционная система: Ubuntu 20.04+ / Debian 11+ / Windows Server 2019+;
– СУБД: PostgreSQL 14+ / MySQL 8+;
– бэкенд: Java (Spring Boot) ;
– фронтенд: React.js с TypeScript;
– API: REST / GraphQL;
– кэширование: Redis;
– очереди сообщений: Kafka / RabbitMQ;
– логирование: ELK Stack (Elasticsearch, Logstash, Kibana).

2.2 Клиентское программное обеспечение

– браузеры: Поддержка всех современных браузеров с актуальными обновлениями;
– операционные системы: Windows, macOS, Linux для веб-версии; Android и iOS для мобильного приложения.

А.2.4 Требования к информационной и программной совместимости

Программный комплекс должен поддерживать интеграцию с внешними сервисами, обеспечивать совместимость с различными платформами и соответствовать стандартам обмена данными.

1. Информационная совместимость

– форматы данных: JSON, GraphQL, SQL (PostgreSQL), экспорт в CSV/XML/JSON;

– хранение данных: Реляционная база (PostgreSQL), медиафайлы в облачном хранилище (Amazon S3, Google Cloud Storage), история сообщений в MongoDB;

– локализация: Кодировка UTF-8, поддержка нескольких языков.

2. Программная совместимость

Операционные системы:

– веб: Windows, macOS, Linux;

– мобильное приложение: Android 8.0+ / iOS 14.0+;

– сервер: Linux (Ubuntu, Debian), Windows Server 2019+;

– браузеры: Chrome, Firefox, Edge (Chromium), Safari (последние версии).

API: Поддержка REST, GraphQL, документирование через OpenAPI (Swagger), внутренняя коммуникация через gRPC.

3. Обновляемость и масштабируемость

– CI/CD-процессы для автоматических обновлений (GitHub Actions, GitLab CI);

– горизонтальное масштабирование без остановки сервиса;

– кэширование через Redis для высокой производительности.

Эти требования гарантируют стабильную работу, гибкость и простоту интеграции с другими системами.

А.2.5 Требования к маркировке и упаковке

Требования к маркировке и упаковке не предъявляются.

А.2.6 Требования к транспортированию и хранению

Требования к транспортированию и хранению не предъявляются.

А.3 Требования к программной документации

Состав программной документации должен включать в себя:

– техническое задание, ГОСТ 19.201-78;

– описание программы, ГОСТ 19.402-2000;

– программу и методику испытаний, ГОСТ 19.301-2000.

А.4 Технико-экономические показатели

Ориентировочная экономическая эффективность не рассчитывается.

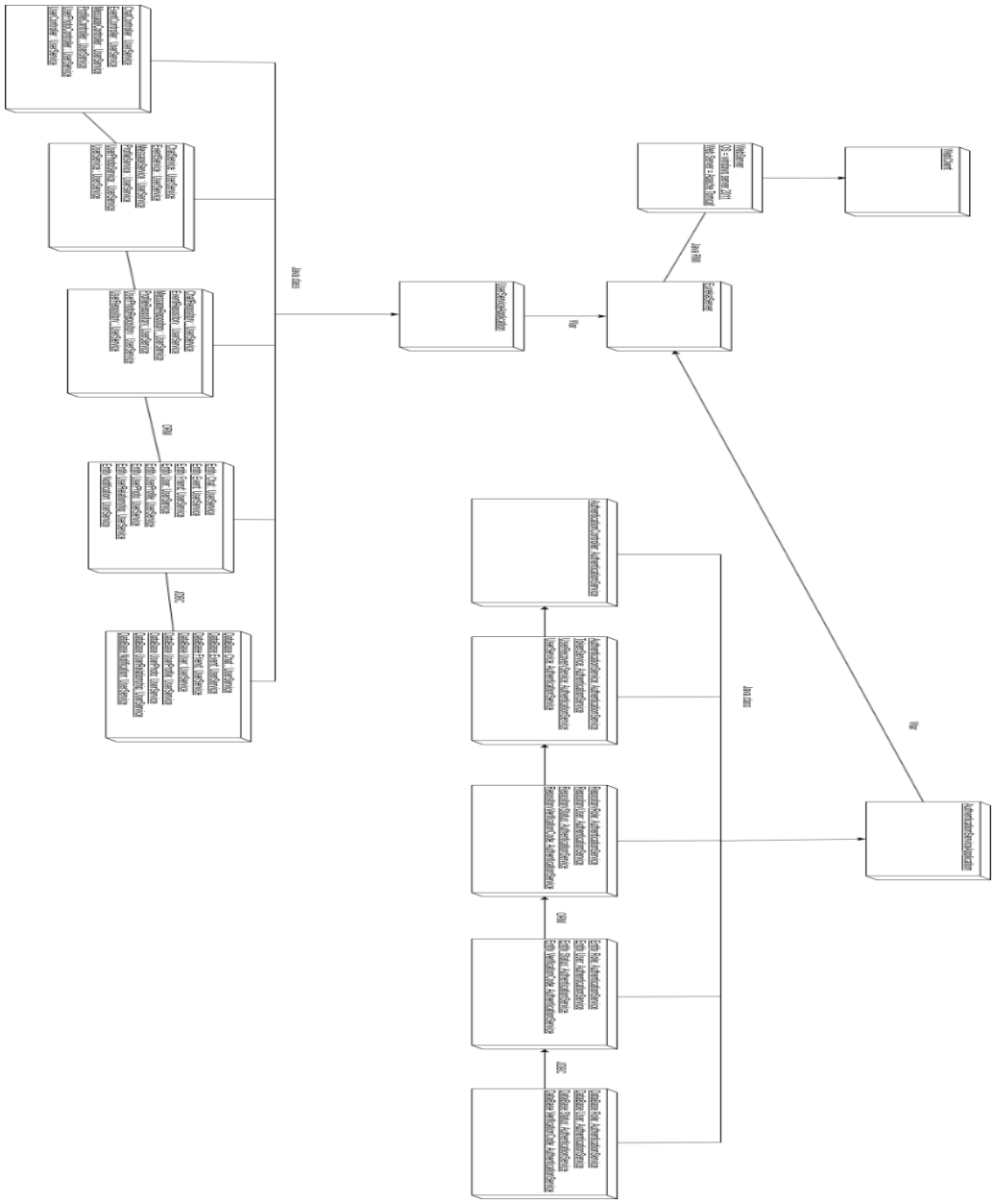
А.5 Стадии и этапы разработки

Стадии и этапы разработки программного модуля и сроки их выполнения представлены в таблице А.1.

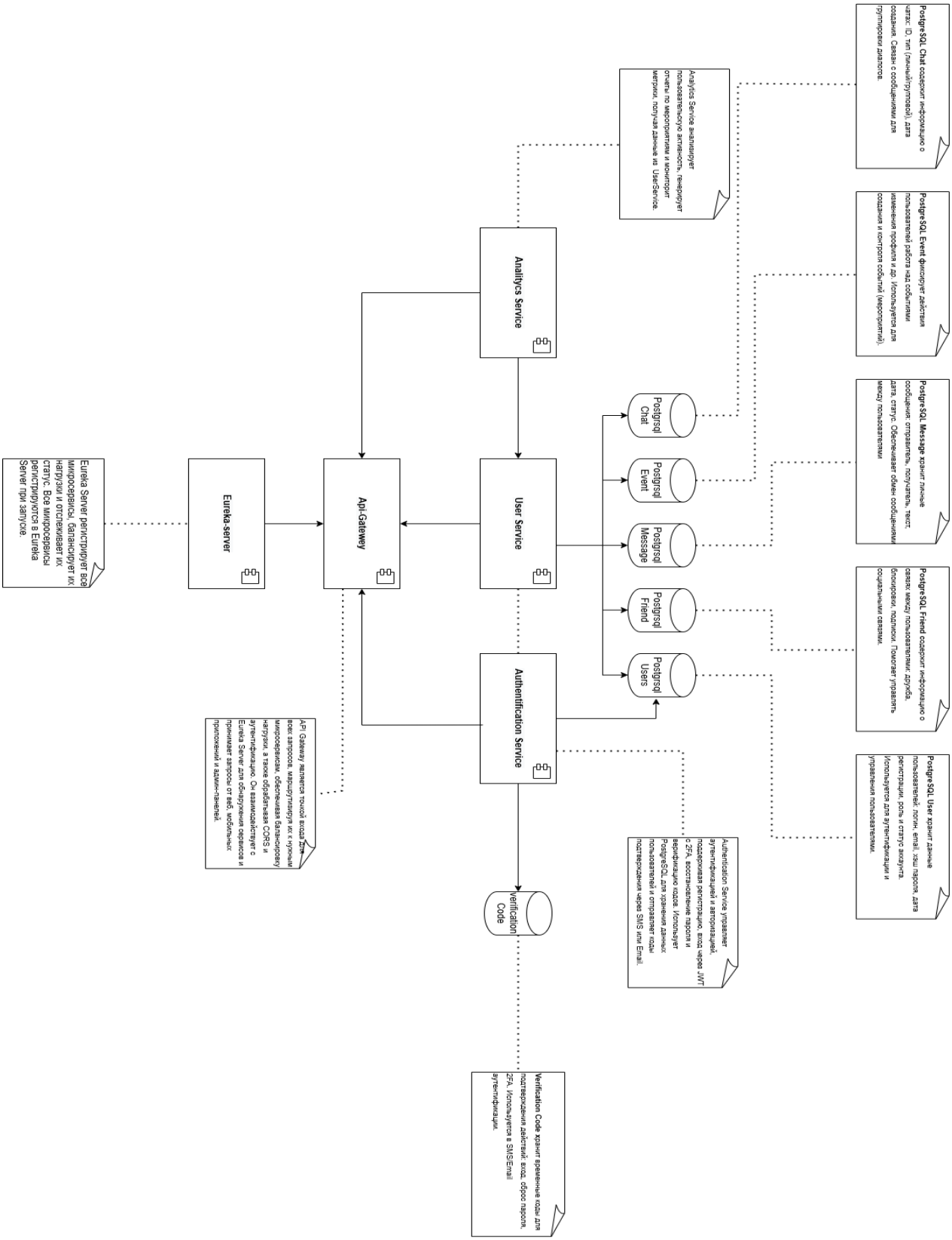
Таблица А.1 – Этапы и сроки разработки

Содержание работы	**Срок выполнения**	
Анализ и планирование	10.02.2025	13.03.2025
Проектирование	14.03.2025	21.03.2025
Реализация	27.03.2025	09.05.2025
Тестирование	10.05.2025	24.05.2025
Написание и оформление документации	25.05.2025	13.06.2025

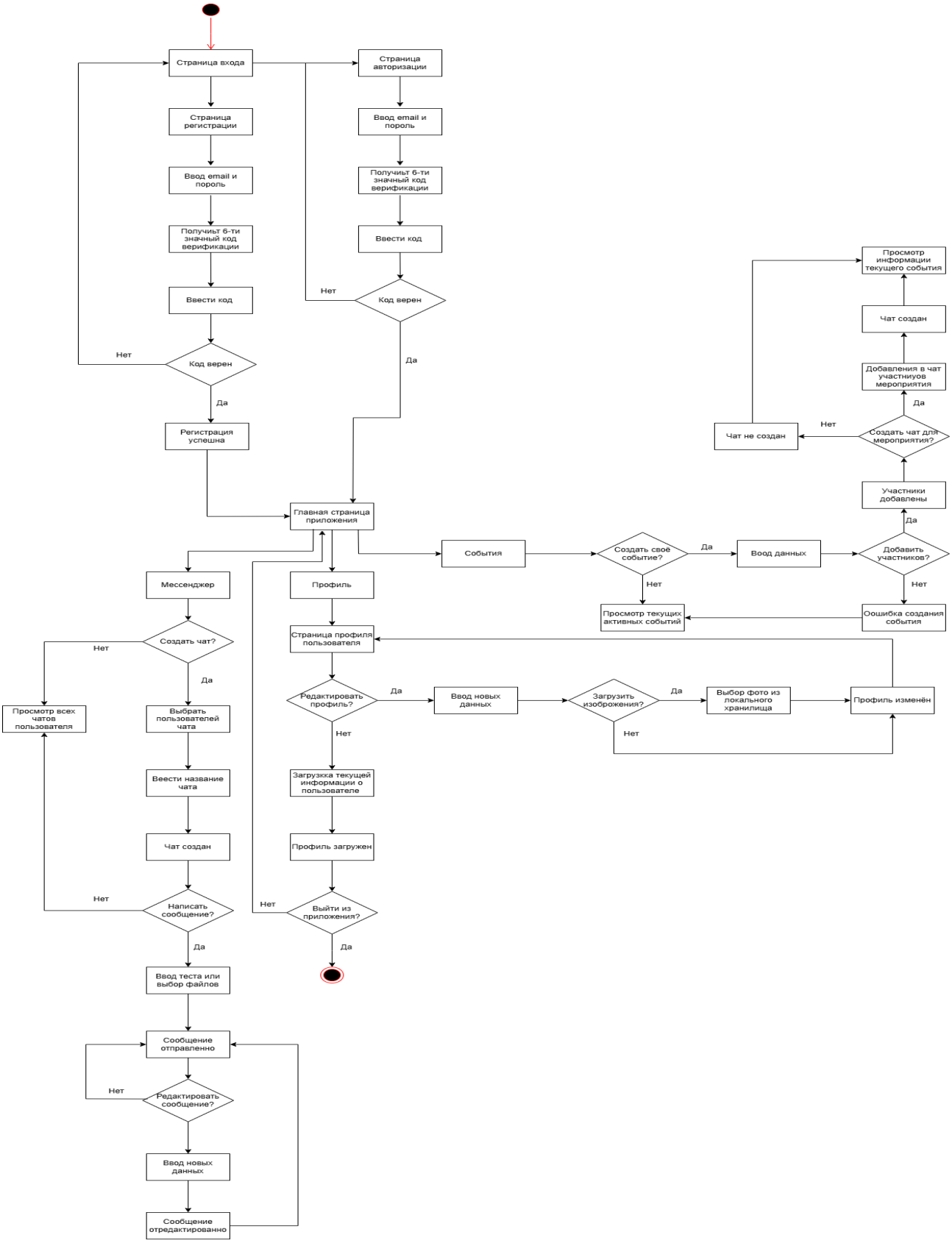
ПРИЛОЖЕНИЕ Б
(обязательное)
Диаграмма развёртывания



ПРИЛОЖЕНИЕ В
(обязательное)
Диаграмма компонентов



ПРИЛОЖЕНИЕ Г
(обязательное)
Алгоритм работы приложения



ПРИЛОЖЕНИЕ Д
(обязательное)
ER-диаграмма базы данных