**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc193731060)

[1 ХАРАКТЕРИСТИКИ ПРЕДПРИЯТИЯ 5](#_Toc193731061)

[1.1. Изучение мероприятий по охране труда и технике безопасности 6](#_Toc193731062)

[1.2. Изучение характеристик используемых аппаратных средств вычистельной техники и ПО 7](#_Toc193731063)

[1.3. Изучение методов проектирования ПО и средств его разработки 8](#_Toc193731064)

[1.4. Изучение методов оценки качества ПО и его сертификации 10](#_Toc193731065)

[1.5. Изучение принципов формирования экономического эффекта от использования современных информационных технологий, формирования себестоимости ПО, рентабельности, прибыли 11](#_Toc193731066)

[2 АНАЛИЗ ИСХОДНЫХ ДАННЫХ И РАЗРАБОТКА ТЕХНИЧЕСКОГО ЗАДАНИЯ 12](#_Toc193731067)

[2.1. Анализ исходных требований 12](#_Toc193731068)

[2.2. Анализ предметной области 14](#_Toc193731069)

[2.3. Анализ существующих прототипов и аналогов 16](#_Toc193731070)

[2.4. Выбор языка разработки 21](#_Toc193731071)

[2.5. Выбор среды разработки 25](#_Toc193731072)

[2.6. Выбор инструмента проектирования 27](#_Toc193731073)

[2.7. Разработка технического задания 28](#_Toc193731074)

[ЗАКЛЮЧЕНИЕ 29](#_Toc193731075)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 30](#_Toc193731076)

[Приложение А 31](#_Toc193731077)

[Техническое задание 31](#_Toc193731078)

# ВВЕДЕНИЕ

Настоящая работа выполнена в рамках преддипломной практики, основной целью которой является подготовка к выполнению дипломного проекта. В процессе прохождения практики проводился анализ предметной области, изучение используемых технологий, а также приобретение практических навыков, необходимых для успешной реализации проекта.

Преддипломная практика является важным этапом обучения, позволяющим на практике применить полученные теоретические знания, ознакомиться с реальными процессами разработки программного обеспечения и углубить профессиональные компетенции. В ходе практики были изучены современные подходы к проектированию, разработке и тестированию программных продуктов, что позволило сформировать основу для дальнейшей работы над дипломным проектом.

В процессе прохождения практики были выполнены следующие задачи:

* изучение структуры и особенностей организации производственного процесса предприятия;
* ознакомление с используемыми аппаратными и программными средствами, включая инструменты проектирования и тестирования;
* подбор и анализ материалов, необходимых для разработки и тестирования программного обеспечения;
* разработка технического задания;
* подготовка отчета по преддипломной практике.

Дипломный проект посвящен разработке серверной части социальной платформы для организации активных игр. В рамках работы планируется спроектировать, разработать и протестировать программное решение, обеспечивающее удобное взаимодействие пользователей с системой организации и проведения спортивных мероприятий.

# ХАРАКТЕРИСТИКИ ПРЕДПРИЯТИЯ

**Компания «Астон Софт»** была основана в **2007 году** и с тех пор зарекомендовала себя как надежный разработчик программного обеспечения и IT-услуг. Основной задачей компании стало создание высококачественных решений в области информационных технологий для различных отраслей бизнеса, от стартапов до крупных корпоративных клиентов.

В **2007 году** группа профессионалов, имеющих опыт в разработке программного обеспечения, решила объединить свои усилия для создания компании, способной предоставить высококлассные IT-решения. С самого начала компания сосредоточилась на разработке сложных и высоконагруженных систем, что позволило завоевать доверие первых клиентов.

С 2008 года компания начала активно осваивать новые технологические направления. Одним из таких стал фокус на **разработку решений на базе платформы 1С,** что позволило «Астон Софт» занять прочные позиции в области автоматизации бизнес-процессов для среднего и крупного бизнеса.

В этот период компания начала сотрудничество с рядом крупных российских и международных заказчиков, что способствовало укреплению ее репутации и расширению партнерской сети.

В 2011 году «Астон Софт» начала расширять географию своей деятельности. Были открыты новые офисы разработки в разных городах СНГ, а также в **Европе**. Одним из значимых шагов стало начало активной работы с клиентами из **Западной Европы** и **США.** Это позволило компании не только расширить спектр проектов, но и развить международные продажи и привлечь высококвалифицированных специалистов.

С 2012 года компания сделала акцент на интеграции с **облачными технологиями,** а также разработке решений для **мобильных платформ.** Это стало важным шагом в направлении цифровой трансформации бизнеса и внедрения инновационных решений в области IT.

С 2014 года «Астон Софт» продолжала активно развивать свои ключевые направления: разработку корпоративных приложений, автоматизацию процессов, внедрение инновационных решений в области **Big Data, машинного обучения** и **искусственного интеллекта.** Кроме того, компания начала активно работать над улучшением внутренней инфраструктуры, оптимизируя процессы управления проектами и повысив качество обслуживания клиентов. В этот период был создан новый отдел продаж, а также усилился фокус на долгосрочные партнерства с ключевыми клиентами.

В последние годы «Астон Софт» продолжила расширять свою команду, активно привлекая специалистов в таких областях, как **анализ данных, интернет вещей (IoT)** и **кибербезопасность.** Компания также активно развивает направления, связанные с созданием **интерактивных платформ, мобильных приложений** и решений для различных отраслей, таких как **финансовый сектор, образование, здравоохранение** и **электронная коммерция.**

На сегодняшний день **«Астон Софт»** — это успешная и стабильно развивающаяся компания с международной командой, которая насчитывает более **1000 специалистов.** Присутствие компании на рынке более **15 лет** позволило ей занять лидирующие позиции в области разработки ПО и IT-услуг для бизнеса. Офисы компании расположены в **Минске, Киеве, Витебске, Полоцке**, а также во **многих других странах.** «Астон Софт» продолжает свою деятельность на рынках **Европы, Азии** и **Северной Америки,** сотрудничая с крупнейшими корпорациями и стартапами.

# Изучение мероприятий по охране труда и технике безопасности

По прибытии на место прохождения практики был проведён вводный инструктаж по охране труда, который является обязательной процедурой перед началом работы в организации. В этот же день на рабочем месте был проведён первичный инструктаж, обеспечивающий ознакомление с требованиями безопасности на конкретном рабочем месте.

Согласно установленным правилам, инструктаж по охране труда делится на несколько видов:

* вводный – проводится при приёме сотрудников и практикантов, независимо от характера их работы;
* первичный на рабочем месте – осуществляется в день прибытия нового работника или стажёра до начала его профессиональной деятельности;
* повторный – проводится не реже одного раза в шесть месяцев для закрепления знаний и контроля соблюдения требований безопасности;
* внеплановый – необходим в случае изменений в нормативных документах, обновления оборудования, а также при нарушениях техники безопасности, авариях или длительных перерывах в работе (более шести месяцев);
* целевой – требуется при выполнении разовых работ, не входящих в основные обязанности сотрудника (например, погрузочные работы, уборка территории), а также при устранении последствий аварий или чрезвычайных ситуаций.

Каждый инструктаж фиксируется в журнале регистрации инструктажей по охране труда, а его проведение контролируется ответственными лицами организации.

# Изучение характеристик используемых аппаратных средств вычистельной техники и ПО

В компании применяется современная компьютерная техника, поддерживающая различные операционные системы, включая **Windows, Linux и macOS**, что обеспечивает гибкость в работе и возможность использования различных инструментов разработки.

В ходе прохождения преддипломной практики активно изучались и использовались различные программные средства, обеспечивающие эффективную разработку, тестирование и управление проектами. Среди используемого программного обеспечения можно выделить:

**JetBrains ToolBox** – комплект всех инструментов разработки от компании JetBrains, включающий **IntelliJ IDEA** (основную среду разработки для серверного программирования на **Java**), **WebStorm** (для работы с фронтенд-частью), а также **DataGrip** (для работы с базами данных).

**Jira** – одна из наиболее популярных систем управления проектами, применяемая для **планирования задач, контроля выполнения и взаимодействия между членами команды**. В рамках практики изучались возможности **Agile-подхода**, управления задачами с использованием **Kanban и Scrum**, а также ведение отчетности по задачам и проектам.

**PostgreSQL** – мощная реляционная база данных, использующаяся для хранения и обработки данных проекта. В процессе работы изучались принципы **нормализации данных, оптимизации SQL-запросов и индексации**.

**MongoDB** – документно-ориентированная нереляционная база данных, использовавшаяся для работы с динамическими структурами данных и хранения **неструктурированной информации**.

**DBeaver** – универсальный инструмент для работы с базами данных, применяемый для **администрирования, анализа и оптимизации запросов** в PostgreSQL и MongoDB.

**Postman** – инструмент для тестирования API-запросов, который использовался при проверке взаимодействия клиентской части с серверной, а также при разработке REST API.

**Docker** – система контейнеризации, позволявшая разворачивать и тестировать приложение в изолированных средах. В рамках практики изучалась работа с **Docker Compose**, настройка контейнеров для базы данных и микросервисов.

**Git** и **GitHub/GitLab** – системы контроля версий, применяемые для **ведения истории изменений, совместной работы над кодом и организации CI/CD-процессов**.

Использование этого программного обеспечения позволило не только получить практические навыки работы с современными инструментами разработки, но и понять принципы их взаимодействия в рамках микросервисной архитектуры, что является важной частью успешной реализации дипломного проекта.

# Изучение методов проектирования ПО и средств его разработки

В процессе проектирования программного обеспечения используются **гибкие методологии и современные инструменты**, позволяющие адаптироваться к динамическим изменениям требований и эффективно управлять жизненным циклом разработки.

Для проектирования системы применяются различные методы моделирования, позволяющие **визуализировать структуру, потоки данных и бизнес-логику приложения**. Основными инструментами являются:

**UML (Unified Modeling Language)** – универсальный язык моделирования, используемый для **проектирования архитектуры приложения, описания взаимодействия компонентов, последовательности выполнения операций и бизнес-процессов**. Среди применяемых диаграмм:

**Диаграммы потоков данных (DFD, Data Flow Diagrams)** – применяются для **визуализации потоков информации** между различными частями системы, что помогает выявить **узкие места** и **оптимизировать логику взаимодействия сервисов**.

**ER-моделирование (Entity-Relationship Model)** – используется при **проектировании баз данных** и позволяет **наглядно представить структуру данных, их взаимосвязи и типы отношений между сущностями**. Это особенно важно для **разработки реляционной структуры базы данных**, которая ляжет в основу социальной платформы.

**Архитектурные схемы микросервисов** – отображают взаимодействие сервисов в рамках общей системы, их API, механизмы обмена данными (REST, WebSockets, gRPC) и методы синхронизации.

Разработка ведётся **итеративными циклами (sprints)**, что соответствует гибким методологиям **Scrum и Kanban**. Такой подход позволяет **постоянно поставлять работающий функционал, тестировать его и адаптироваться к изменяющимся требованиям**.

Scrum

Использование **Scrum** включает в себя:

* **бэклог продукта** – список всех задач, необходимых для реализации функционала;
* **планирование спринтов** – выделение приоритетных задач и их распределение по разработчикам;
* **ежедневные стендапы** – короткие совещания для обсуждения статуса работы, выявления препятствий и координации команды;
* **ретроспективы** – анализ завершённого спринта для выявления успешных практик и улучшения рабочих процессов.

Kanban

Методология **Kanban** применяется для:

* **визуализации потока задач** – использование доски Kanban (в **Jira, Trello или ClickUp**) для контроля статуса задач (To Do, In Progress, Done);
* **оптимизации рабочих процессов** – уменьшение "узких мест" и регулирование нагрузки на разработчиков;
* **минимизации времени простоя** – обеспечение плавного перехода задач между этапами разработки.

Для эффективной работы команды используются современные инструменты, обеспечивающие **контроль версий, тестирование, автоматизацию развертывания и CI/CD-процессы**.

Контроль версий и совместная работа

**Git (GitHub, GitLab, Bitbucket)** – системы контроля версий, позволяющие управлять кодовой базой, обеспечивать параллельную разработку и интеграцию изменений.

**Pull Requests и Code Reviews** – используются для контроля качества кода и улучшения совместной работы команды.

CI/CD и автоматизация развертывания

**Jenkins, GitHub Actions, GitLab CI/CD** – платформы для **автоматического тестирования и деплоя кода**, что ускоряет процесс доставки обновлений.

**Docker и Kubernetes** – используются для контейнеризации сервисов, их управления и масштабирования в продакшене.

Базы данных

**PostgreSQL** – основная **реляционная база данных**, применяемая для хранения данных пользователей, мероприятий, подписок и аналитики.

**MongoDB** – **NoSQL-база**, используемая для хранения **неструктурированных данных**, например, сообщений в чатах.

**Redis** – кэширующая система для **ускорения обработки частых запросов**.

Средства тестирования

**Selenium, TestNG, JUnit** – инструменты для автоматизированного тестирования веб-приложений.

* **selenide** – библиотека для удобного тестирования UI;
* **postman** – инструмент для тестирования API-запросов;
* **allure** – система для генерации отчётов по тестам.

# Изучение методов оценки качества ПО и его сертификации

Качество программного продукта – это характеристика, определяющая степень соответствия установленных требованиям, предполагаемым ожиданиям заказчика или пользователя

Для оценки программного обеспечения существует два основных подхода: объективный и субъективный. Рассмотрим их более подробно.

Объективный метод оценивания позволяет вычислить оценку качества исходя из таких критериев, как: размер проекта, количество обнаруженных дефектов, критичность каждого дефекта. Данный метод как правило применяется в случаях, когда на проектах есть большое количество версий программного продукта. Таким образом, для наглядности на выходе получается конкретная цифра, которая покажет динамику изменения качества ПО. Кроме того, этот метод может использоваться, когда над различными модулями одного приложения работают много специалистов. Из-за этого определение качества без конкретной оценки может быть затруднено.

Субъективный метод оценки качества заключается в выставлении оценки специалистом по тестированию на основе следующих критериев: удобство использования ПО, количество новых дефектов, количество активных дефектов, количество дефектов в разрезе модулей, критичность дефектов, критичность дефектов в разрезе модулей, количество проваленных тест-кейсов в разрезе модулей, количество заблокированных тест-кейсов, сравнение с предыдущими оценками качества, сравнение с объективной оценкой качества, опыт тестировщика. Данный метод используется для проектов любого размера, сложности и продолжительности.

Сертификация программного обеспечения – это процесс подтверждения соответствия программного продукта установленным стандартам качества. В зависимости от сферы применения могут использоваться различные стандарты, такие как:

* ISO 25010 – международный стандарт оценки качества ПО (надежность, функциональность, производительность и др.);
* ГОСТ 28195-89 – стандарты сертификации ПО в России;
* IEC 62304 – стандарт для медицинского ПО;
* Common Criteria (ISO/IEC 15408) – стандарт безопасности ПО.

Сертификация может быть обязательной или добровольной. Обязательная сертификация требуется для критически важных систем, таких как медицинское, банковское или государственное ПО. Добровольная сертификация проводится для повышения доверия пользователей и конкурентоспособности продукта.

Применение всех трех методов в совокупности позволяет объективно оценить качество программного продукта и соответствие его требованиям.

# Изучение принципов формирования экономического эффекта от использования современных информационных технологий, формирования себестоимости ПО, рентабельности, прибыли

Мероприятия, связанные с внедрением современных средств вычислительной техники, требуют значительных материальных и трудовых затрат. Обязательным условием применения этих средств является получение экономической эффективности от их внедрения. Для этого требуется проведение расчетов экономической эффективности и ее обоснования.

Обоснование экономической эффективности применения информационной технологии позволяет определить необходимость и целесообразность затрат на создание и внедрение автоматизированной системы сбора и обработки информации; установить основные направления автоматизации обработки информации; выбрать экономически эффективные варианты технологических процессов обработки экономической информации. Экономическая эффективность автоматизированной обработки информации обеспечивается за счет следующих основных факторов: высокая скорость выполнения операций по сбору, передаче, обработке и выдаче информации, достигнутой за счет высокой производительности современных технических средств, максимального сокращения времени на выполнение отдельных операций; повышения качества выполнения экономических расчетов за счет создания единой информационной базы, установления четкого графика ее получения, устранения из нее постоянных данных и производных показателей, а также за счет ее централизованной обработки; улучшения информационного обслуживания за счет сокращения сроков разработки и получения документов.

Исходя из вышеизложенного следует сказать, что общая эффективность применения информационных технологий находится в прямой зависимости от снижения затрат на обработку информации (так называемая прямая эффективность) и от достигаемого повышения уровня информационного обслуживания (так называемая косвенная эффективность).

# АНАЛИЗ ИСХОДНЫХ ДАННЫХ И РАЗРАБОТКА ТЕХНИЧЕСКОГО ЗАДАНИЯ

# 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.

# 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.

# 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](https://www.meetup.com)[1].

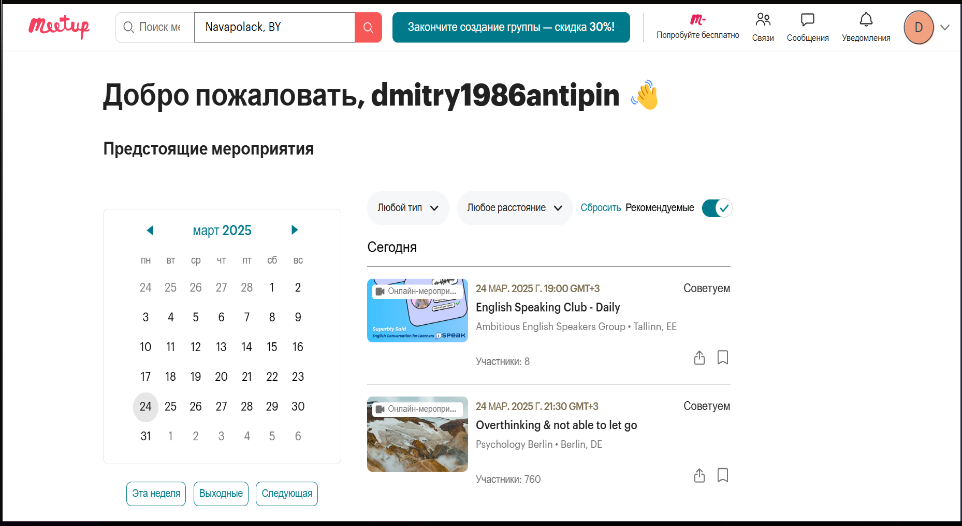


Рисунок 2.1 – Интерфейс плотформы MeetUp

**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](https://opensports.net/)[2].

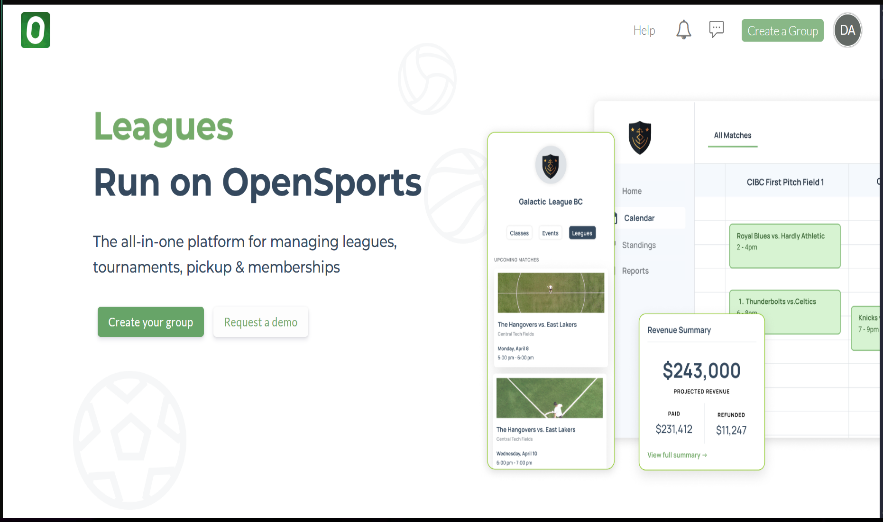


Рисунок 2.2 – Интерфейс плотформы OpenSports

**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].

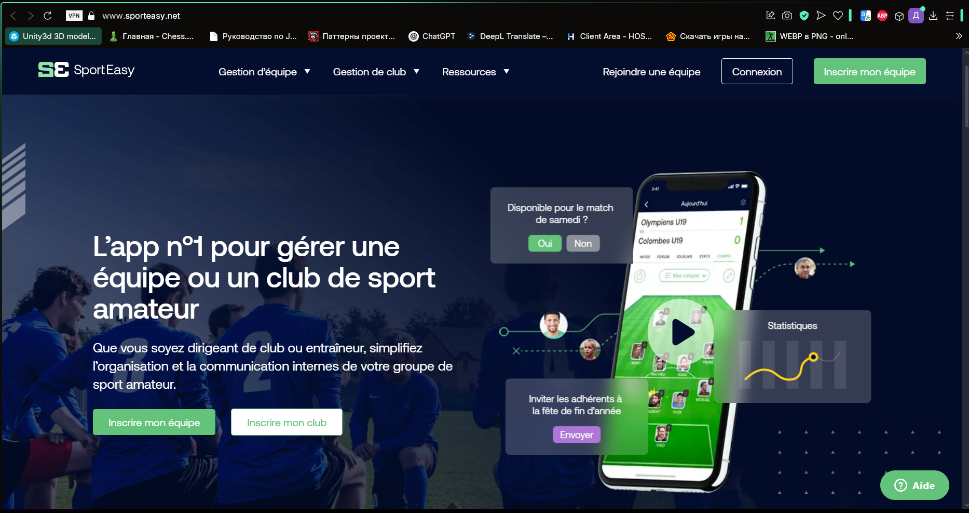


Рисунок 2.3 – Интерфейс плотформы OpenSports

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].

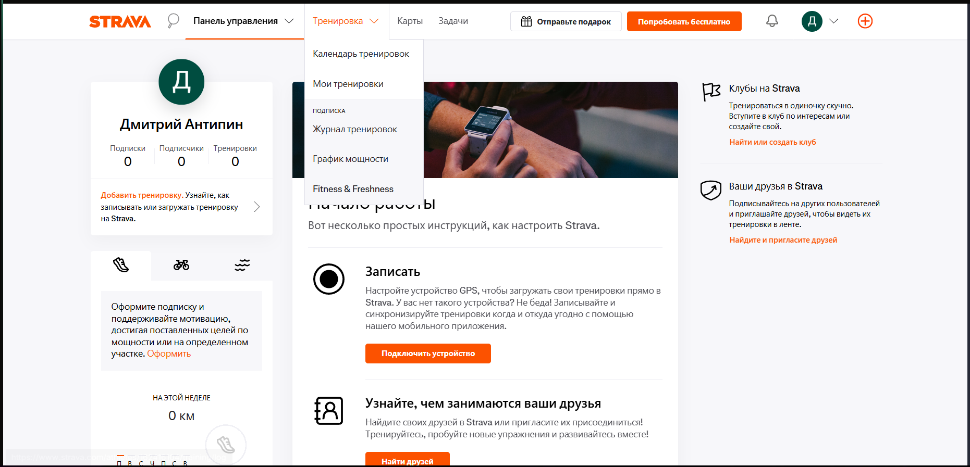


Рисунок 2.3 – Интерфейс плотформы Strava

Результаты сравнительного анализа рассматриваемых аналогов представлены в таблице 2.1.

Таблица 2.1 – 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.

# 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.

1. 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.

1. 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.**

1. 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.

1. **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**.

1. **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.

1. **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).

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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 2.2, которая была составлена на основе информации из следующих источников: Java[5], Kotlin[6], Python[7], Go[8], Node.js[9].

Таблица 2.2 – 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.

# 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 2.3. 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 2.3 – 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.

# 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.

# 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.

Техническое задание дипломного проекта приведено в приложении А.

# 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.

* Управление пользователями

Регистрация и авторизация пользователей через email, телефон или социальные сети.

Возможность редактирования профиля (аватар, имя, описание, предпочтения, доступность).

Управление подписками и настройками уведомлений.

Возможность поиска и добавления друзей.

* Работа с мероприятиями

Создание, редактирование и удаление мероприятий с указанием параметров (название, дата, место, описание, условия участия).

Поиск мероприятий с фильтрацией по дате, местоположению, типу игры и другим параметрам.

Возможность подписки на мероприятия и получения уведомлений об изменениях.

Просмотр детальной информации о мероприятии, включая карту местоположения и контакты организатора.

Формирование команд и распределение ролей в мероприятиях.

* Рекомендательная система

Персонализированные предложения мероприятий на основе интересов, местоположения и активности пользователя.

Использование системы рейтингов и отзывов для улучшения качества рекомендаций.

* Чат и взаимодействие пользователей

Возможность отправки личных сообщений между пользователями.

Групповые чаты для обсуждения мероприятий.

Интеграция чата с системой уведомлений.

* Аналитика и геймификация

Ведение истории посещённых мероприятий и достижений.

Отображение статистики активности пользователя.

Система рейтингов, значков и наград для повышения вовлечённости.

**А.2.2 Требования к надёжности**

Контроль входной информации

1. Проверка наличия входных данных

* все входные данные, передаваемые в API, должны проверяться на наличие обязательных параметров перед их обработкой.
* если один или несколько обязательных параметров отсутствуют, API должно возвращать соответствующий код ошибки и детализированное сообщение об ошибке.
* исключается возможность обработки пустых или некорректных запросов для предотвращения ошибок на стороне сервера.

1. Валидация формата данных

Входные данные должны соответствовать заданным форматам и типам данных:

* идентификаторы пользователей, мероприятий и сообщений должны быть валидными UUID или числовыми значениями.
* даты и временные метки должны соответствовать формату ISO 8601.
* числовые значения (например, количество участников, рейтинг) должны находиться в допустимых диапазонах.

1. Обеспечение целостности данных

* все запросы на создание, обновление и удаление данных должны проходить проверки на дублирование.
* реализация механизма **идемпотентности** для критических операций, предотвращающего повторное выполнение запроса при сбоях сети.
* использование транзакций в базе данных для предотвращения потери данных при их изменении.

Контроль выходной информации

1. **Гарантированная доставка данных**

* API должно обеспечивать надежную передачу данных клиентам (веб-платформе, мобильному приложению и боту) с использованием механизмов **повторной отправки** при неудачных попытках связи.
* реализация **очередей сообщений (Kafka, RabbitMQ)** для обеспечения гарантированной доставки данных в асинхронном режиме.
* все критически важные операции должны подтверждаться API перед их финальной фиксацией в системе.

1. Актуальность информации

* все данные, предоставляемые пользователям, должны быть **актуальными и соответствовать текущему состоянию системы**.
* использование механизмов **кэширования (Redis, Memcached)** с автоматическим обновлением данных при их изменении.
* внедрение механизма **WebSocket или SSE** для передачи актуальной информации в режиме реального времени.

1. Логирование всех операций

API должно вести **детализированный журнал всех операций**, включая:

* время и тип запроса.
* идентификатор пользователя, выполняющего операцию.
* результат выполнения запроса (успешно, ошибка).

Логи должны храниться в **централизованном хранилище** с возможностью их анализа с использованием ELK Stack (Elasticsearch, Logstash, Kibana).

**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. Клиентские устройства

Веб-версия:

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

1. 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).
  1. 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.

1. 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**.

1. 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 |