|  |
| --- |
| **МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ** |
| **ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ ИМЕНИ К.Г. РАЗУМОВСКОГО (ПЕРВЫЙ КАЗАЧИЙ УНИВЕРСИТЕТ)»**  **(ФГБОУ ВО «МГУТУ ИМ. К.Г. РАЗУМОВСКОГО (ПКУ)»)** |
|  |
| **УНИВЕРСИТЕТСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ** |

|  |  |
| --- | --- |
|  | ДОПУСКАЕТСЯ К ЗАЩИТЕ |
|  | Председатель ПЦК специальности 09.02.07 Информационные системы и программирование |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Е.В. Вернер |
|  | «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г. |

|  |
| --- |
| **Дипломный проект** |
|  |
| на тему: Разработка информационной системы по геймификации и управлению задачами (на примере приложения «Todoist» |
|  |
| студента группы 090207-9о-21/2  специальности 09.02.07 Информационные системы и программирование |
| Хохлова Дмитрия Алексеевича |

|  |  |  |
| --- | --- | --- |
| Студент |  | Д.А. Хохлов |
| Руководитель |  | А. А. Марков |

|  |  |  |
| --- | --- | --- |
| Дата защиты «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2025 г. | | |
| Оценка:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | |
| Председатель ГЭК |  | П.Р. Сафиканов |

Москва

2025

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc200101471)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 6](#_Toc200101472)

[1.1 Исследование предметной области 6](#_Toc200101473)

[1.1.1 Терминология в предметной области 6](#_Toc200101474)

[1.1.2 Автоматизация процессов 6](#_Toc200101475)

[1.1.3 Анализ схожих информационные системы 8](#_Toc200101476)

[1.1.4 Обоснование актуальности 15](#_Toc200101477)

[1.2 Анализ и выбор инструментальных средств 15](#_Toc200101478)

[1.2.1 Критерии выбора 15](#_Toc200101479)

[1.2.2 Выбранные инструменты 16](#_Toc200101480)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 18](#_Toc200101481)

[2.1 Спецификация программного изделия 18](#_Toc200101482)

[2.1.1 Постановка задачи 18](#_Toc200101483)

[2.1.2 Наименование программы 18](#_Toc200101484)

[2.1.3 Функциональное назначение 18](#_Toc200101485)

[2.1.4 Требования к функциональным характеристикам 19](#_Toc200101486)

[2.1.5 Требования к составу и параметрам технических средств 19](#_Toc200101487)

[2.2 Проектирование программного изделия 19](#_Toc200101488)

[2.2.1 Диаграмма вариантов использования 19](#_Toc200101489)

[2.2.2 Архитектура информационной системы 20](#_Toc200101490)

[2.2.3 Защита информационной системы 22](#_Toc200101491)

[2.2.4 Описание используемых библиотек 23](#_Toc200101492)

[2.3 Разработка программного изделия 25](#_Toc200101493)

[2.3.1 Описание разработки 25](#_Toc200101494)

[2.3.2 Трудности и решения 25](#_Toc200101495)

[2.3.3 Выполняемые функции 25](#_Toc200101496)

[2.3.4 Перспективы усовершенствования информационной системы 25](#_Toc200101497)

[2.4 Тестирование программного изделия 25](#_Toc200101498)

[ЗАКЛЮЧЕНИЕ 26](#_Toc200101499)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 29](#_Toc200101500)

[ПРИЛОЖЕНИЕ А 30](#_Toc200101501)

# ВВЕДЕНИЕ

Дипломный проект (далее ДП) посвящен разработке информационной системы по геймификации и управлению задачами (на примере приложения «Todoist»).

Внедрение геймификации в информационные системы становится все более популярным стратегическим подходом в современном бизнесе. Геймификация представляет собой использование игровых элементов и механик в рабочей среде для повышения мотивации, вовлеченности и участия пользователей. Она предлагает новые возможности для создания стимулирующей и привлекательной рабочей среды, способствующей достижению лучших результатов.

Геймификация является относительно новым понятием, поэтому необходимо определить его и описать основные принципы. Геймификация базируется на использовании игровых элементов, таких как цели, правила, рейтинги, достижения и награды, для создания стимулирующей и привлекательной рабочей среды. Она основывается на теории мотивации, психологии игр и дизайна игр.

Актуальность темы обусловлена растущей потребностью бизнеса в инновационных методах управления персоналом, повышения продуктивности и лояльности сотрудников, а также в усилении вовлеченности пользователей в цифровых сервисах.

Объектом дипломного проекта является: Управления задачами.

Предмет дипломного проекта: Информационная система по управлению задачами.

Целью данного дипломного проекта является разработка удобной, интуитивно понятной и увлекательной информационной системы для эффективного управления и распределения задач с использованием элементов геймификации.

Исходные данные: функционал приложения «Todoist», документация разработчика приложений, работающих на С#, документация разработчика ботов для Telegram.

В задачи данного дипломного проекта входит:

* анализ предметной области;
* рассмотрения аналогов;
* проектирование архитектуры информационной системы;
* создание интерфейса;
* реализация основных функций.

Теоретическая значимость:

* cоздание подходов к использованию to-do листов как инструмента личной эффективности.

Практическая значимость:

* использование геймификации в тайм-менеджменте мотивирует пользователя выполнять задачи за счет игровых элементов (баллы, награды), что повышает вовлеченность и дисциплинирует в следовании намеченным целям.

Круг рассматриваемых проблем:

* изучение предметной области;
* создание информационной системы;
* геймификация управления задачами.

Структура ДП состоит из введения, двух глав (теоретическая, практическая), заключения, списка используемых источников, а также приложения с программным кодом.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## Исследование предметной области

## Терминология в предметной области

To-Do List– инструмент для фиксации задач, которые необходимо выполнить. В современных приложениях (например, Todoist) туду-листы включают функции напоминаний, категоризации и интеграции с другими сервисами.

Управление задачами – процесс планирования, организации и контроля выполнения задач, направленный на достижение поставленных целей. В цифровых системах оно реализуется через создание списков дел, установку сроков, приоритезацию и делегирование.

Геймификация– применение игровых механик (баллы, награды) в неигровых контекстах для повышения мотивации пользователей. В системах управления задачами (как Todoist) она может выражаться в виде достижений за выполненные задачи или рейтинговой системы."

Продуктивность – показатель эффективности выполнения задач, часто измеряемый соотношением результата к затраченному времени. Приложения вроде Todoist помогают повышать продуктивность за счёт структурирования работы и снижения когнитивной нагрузки.

Информационная система **(**далее **ИС)** – это программно-аппаратный комплекс, предназначенный для сбора, хранения, обработки, поиска и передачи данных с целью поддержки управленческих и бизнес-процессов.

## Автоматизация процессов

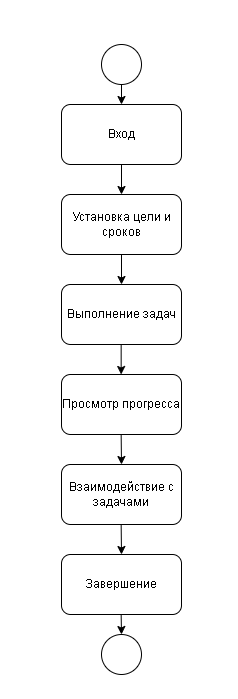
ИС направлена на автоматизацию управления задачами, и повышение продуктивности выполнения этих задач с помощью геймификации.

Ключевые процессы автоматизации:

* создание, редактирование, категоризация и удаление задач;
* отслеживание статусов, сроков;
* cбор и обработка статистики по выполнению задач;
* **мотивационные механики;**
* **социальное взаимодействие.**

ИС позволит пользователям удобно распределять задачи, отслеживать прогресс и повышать эффективность работы за счет игровых элементов. Система автоматизирует рутинные процессы, предоставляет аналитику продуктивности и поддерживает вовлеченность через мотивационные механизмы. Это создаст удобную и стимулирующую среду для управления личными и командными задачами.

Данный процесс представлен в виде схемы на Рисунке 1.



1. Бизнес-процесс

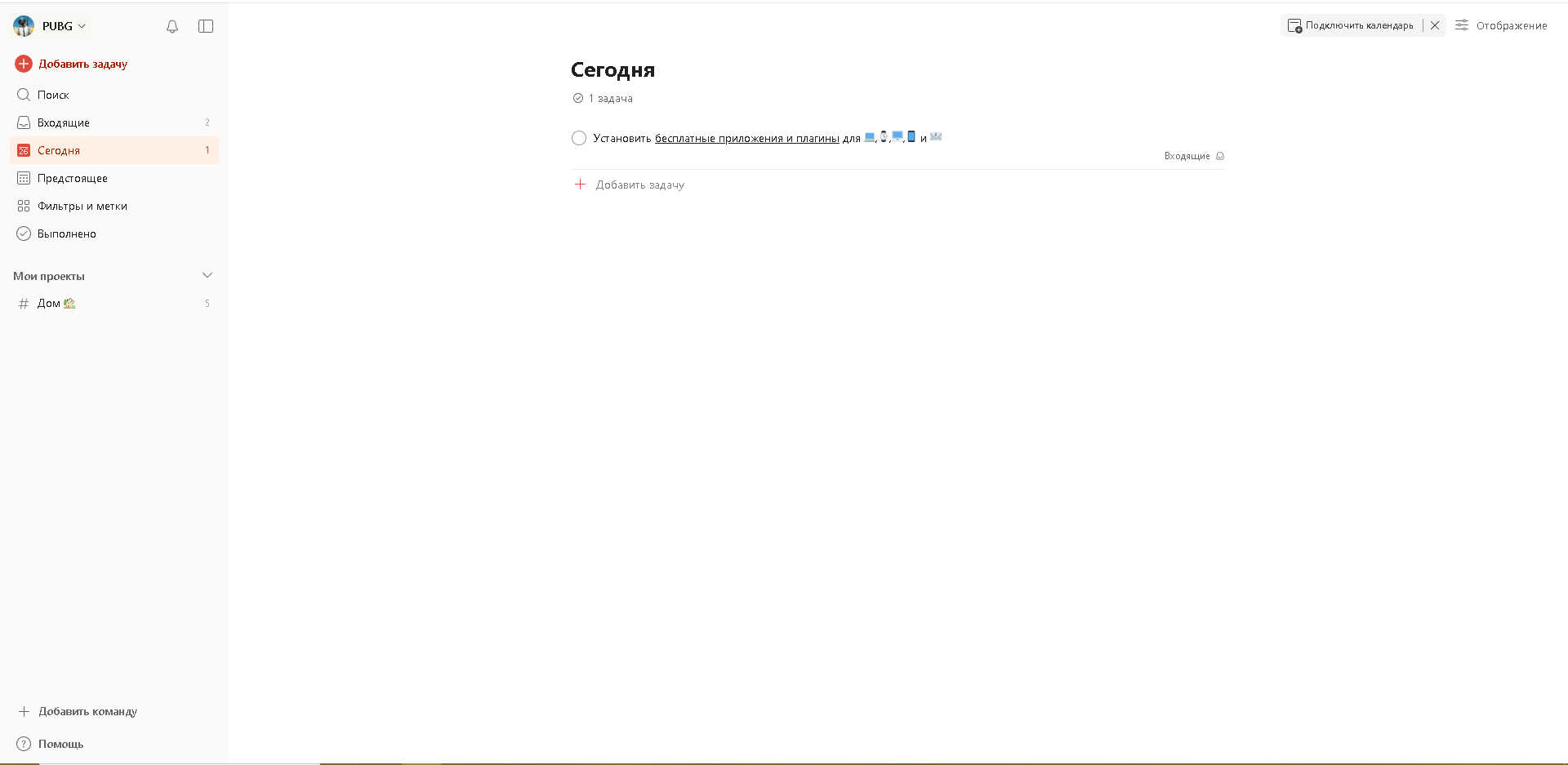
## Анализ схожих информационные системы

В данный момент времени существует множество информационных систем с уклоном на To-Do List. Для анализа были выбраны самые популярные и востребованные информационные системы.

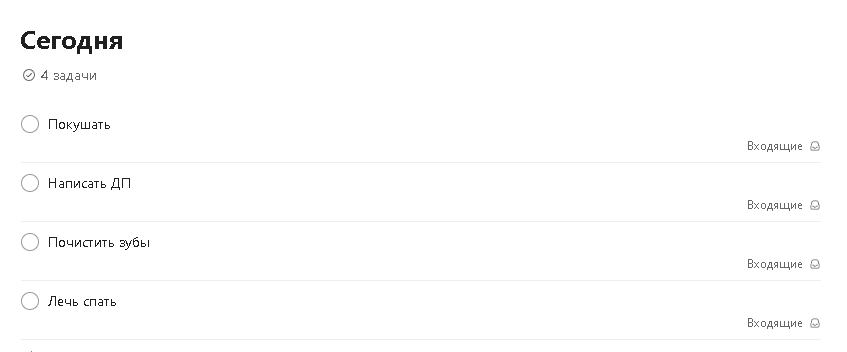
Todoist – это **инструмент для управления задачами и проектами**. Он предлагает множество функций, которые помогают пользователям организовывать свою работу и личные дела.

В Todoist можно создавать задачи, устанавливать сроки, добавлять метки и приоритеты, а также делегировать задачи другим пользователям. Приложение поддерживает создание подзадач и проектов, что позволяет разбивать большие задачи на более мелкие и управляемые части.

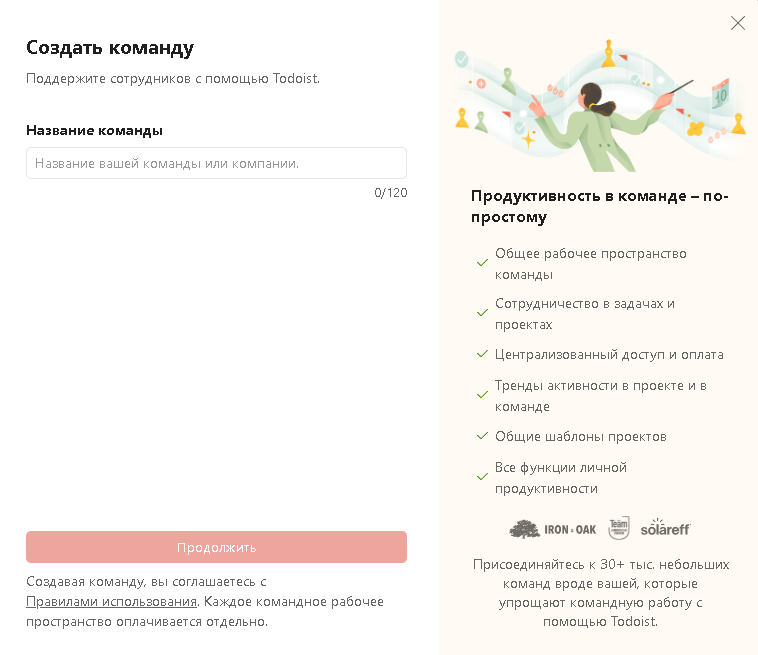
Данная информационная система изображена на рисунках 2 – 5.



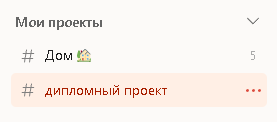
1. Начальная страница Todoist



1. Список дел в Todoist



1. Возможность добавления команды в Todoist

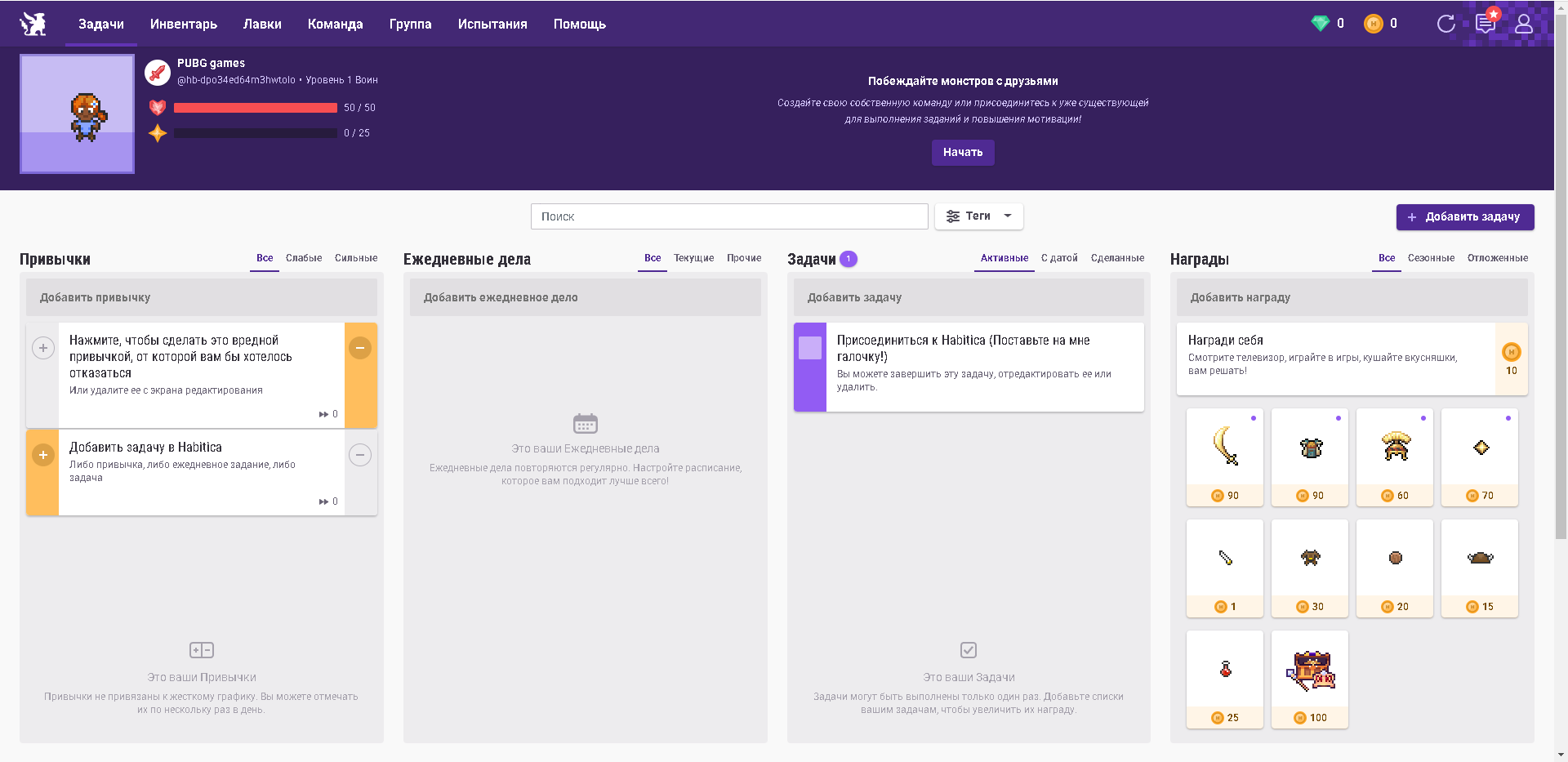


1. Возможность деления задач на проекты в Todoist

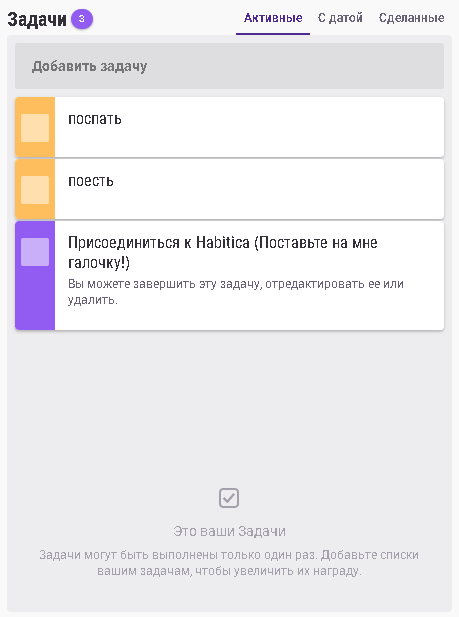
**Habitica** — это **информационная система для привычек и производительности, которое превращает жизнь пользователя в ролевую игру**

Она позволяет пользователям создавать список задач, которые они хотят выполнить, например, учёба или уборка, а затем вознаграждает их очками опыта или золотом за выполнение этих задач.

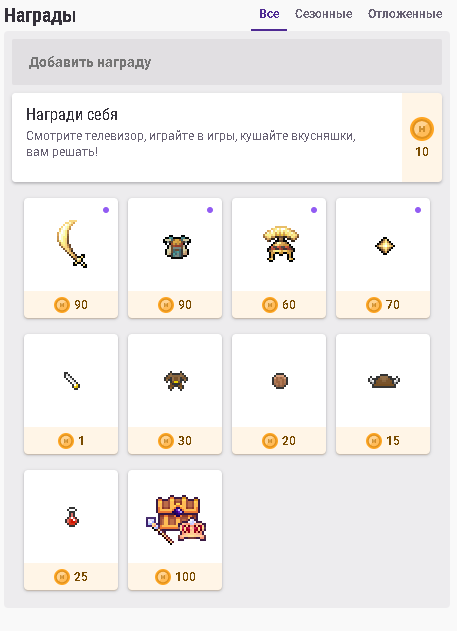
Данная информационная система изображена на рисунках 6 – 8.



1. Начальная страница **Habitica**

****

1. Список дел в **Habitica**



1. Список наград в **Habiticaэ**

После изучения данных информационных систем были выявлены плюсы и минусы каждой системы. Плюсы и минусы можно рассмотреть в таблице 1.

1. Сравнение информационных систем

|  |  |  |
| --- | --- | --- |
| **Категория** | **Todoist** | **Habitica** |
| Плюсы | * **Профессиональный интерфейс** — чёткий, быстрый, без лишнего; * **Г**ибко**сть** — проекты, метки, фильтры, шаблоны; * **Интеграции** (Google Календарь, Slack и др.); * **Надёжность** — работает без багов; * **Коллаборация** — удобно для команд. | * **Мотивация через игру** — прокачка персонажа, квесты; * **Вовлекает** — делает рутину веселее; * **Социальность** — групповые квесты с друзьями; * **Привычки** — удобно трекать регулярные дела; * **Бесплатно** — базовая версия полноценна. |
| Минусы | * **Можно использовать только через сайт;** * **Скучноват** — нет «вау»-мотивации; * **Геймификация слабая** (только Karma); * **Для привычек не идеален** — нет глубокой аналитики. | * **Нужно устанавливать приложение;** * **Выглядит непрофессионально** — пиксельный дизайн; * **Сложные проекты неудобны** — нет нормальных подзадач; * **Баговат** — мобильное приложение глючит; * **Мало интеграций** — только базовые; * **Отвлекает** — можно увлечься игрой, а не делами. |
| Для кого? | * Кому **важна эффективность**, а не мотивация; * Для **работы/учебы;** * Если нужны **сложные проекты.** | * Тем, кому **скучно** вести списки дел; * Для **привычек и личных целей;** * Если любите **игры и соревнование.** |

В Информационной системе, разработанной входе ДП взаимодействие с пользователем будет проходить с помощью мессенджера «Telegram». Благодаря кроссплатформенности, пользователи смогут получать уведомления и работать с системой как с компьютера, так и со смартфона, без необходимости установки дополнительного ПО. «Telegram» обеспечивает быструю доставку сообщений, поддержку ботов для автоматизации задач, а также высокий уровень безопасности.

Так же преимуществом данной информационной системы является легкая в понимание геймификация. Возможность создавать команды и соревноваться между собой. Приятный интерфейс для пользователей, который побуждает заниматься тайм-манежем.

## Обоснование актуальности

Актуальность темы обусловлена растущей потребностью бизнеса в инновационных методах управления персоналом, повышения продуктивности и лояльности сотрудников, а также в усилении вовлеченности пользователей в цифровых сервисах.

Сегодня все больше людей ищут способы не просто планировать дела, а делать это с удовольствием. Обычные списки задач быстро надоедают, а приложения вроде Todoist хоть и удобные, но слишком сухие и деловые. С другой стороны, геймифицированные сервисы, такие как Habitica, добавляют азарта, но часто выглядят слишком по-детски или отвлекают от реальных дел.

## Анализ и выбор инструментальных средств

## Критерии выбора

Основными критерия выбора для разработки информационной системы являются:

* функциональные возможности;
* безопасность системы;
* возможность создания удобного для пользователя интерфейса;
* масштабируемость;
* доступность в регионе.

## Выбранные инструменты

Анализ инструментов для разработки информационной системы представлен в таблице 2.

1. Выбор инструментов

|  |  |  |
| --- | --- | --- |
| **Инструменты** | **Плюсы** | **Минусы** |
| C# | Высокая производительность, хорошая поддержка OOP, отличная интеграция с Windows | Меньшая кроссплатформенность, заточенность под Windows |
| Java | Кроссплатформенность, широкое сообщество, используется в Android | Медленнее запускается, требует больше памяти |
| Visual Studio | Мощная IDE, отличная отладка, поддержка .NET | Тяжеловесность, привязанность к Windows |
| Eclipse | Быстрая, открытая IDE, хороша для Java | Меньше функционала из коробки по сравнению с VS |
| PostgreSQL | Надежность, поддержка расширений, хорошо работает с большими данными | Сложность настройки, не самая простая кривая обучения |
| MySQL | Простота, широкая поддержка хостингов, быстрее на малых объемах данных | Меньше возможностей масштабирования, не такие строгие типы |
| pgAdmin4 | Удобный графический интерфейс для PostgreSQL, продвинутый функционал | Более сложный интерфейс для новичков |

Продолжение таблицы 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| phpMyAdmin | Простой интерфейс для управления MySQL | | | Ограниченный функционал по сравнению с pgAdmin4 |
| Docker | Контейнеризация, переносимость, легкость в развертывании | | | Не всегда нужен весь функционал, требуется знание Dockerfile |
| VirtualBox | | Полноценная виртуализация, поддержка разных ОС | Большая нагрузка на систему, медленнее запускается | | |
| Git | | Распределённая система контроля версий, гибкость | Сложность для новичков, требует понимания ветвления | | |
| SVN | | Централизованная система контроля версий, проще концептуально | Менее гибкая, плохо масштабируется на большие проекты | | |
| GitHub | | Интеграция с Git, поддержка CI/CD, популярность | Ограничения в приватных репозиториях без оплаты | | |
| Bitbucket | | Интеграция с Git, приватные репозитории бесплатно | Меньшая популярность по сравнению с GitHub | | |

Исходя из данного анализа под критерии инструментов для разработки информационной системы подходят:

* C#;
* Visual Studio;
* PostgresSQL;
* PgAdmin4;
* Docker;
* Git;
* Github.

# ПРАКТИЧЕСКАЯ ЧАСТЬ

## Спецификация программного изделия

## Постановка задачи

Разработка информационной системы по геймификации и управлению задачами (на примере приложения «Todoist»).

Исходные данные: функционал приложения «Todoist», документация разработчика приложений, работающих на С#, документация разработчика ботов для Telegram.

Постановка задачи: изучить функциональные возможности приложения «Todoist» и на их основе разработать распределённое приложение с серверной частью на С# и клиентским интерфейсом в виде Telegram-бота, позволяющее пользователям создавать и выполнять задачи с использованием системы очков, наград, клановой и рейтинговой систем для повышения мотивации и вовлечённости.

## Наименование программы

Разработка информационной системы «**TaskSlayer**».

## Функциональное назначение

Информационная система «**TaskSlayer**», предназначенная для управления задачами**,** превращения рутинного планирование в увлекательную игру. Система предназначена для повышения продуктивности пользователей за счет мотивационных механик, визуального прогресса и элементов соревнования.

## Требования к функциональным характеристикам

ИС должна работать на всех устройства поддерживающих мессенджер Telegram. Корректно отвечать на запросы пользователя.

## Требования к составу и параметрам технических средств

Минимальные характеристики телефона для полноценной работоспособности:

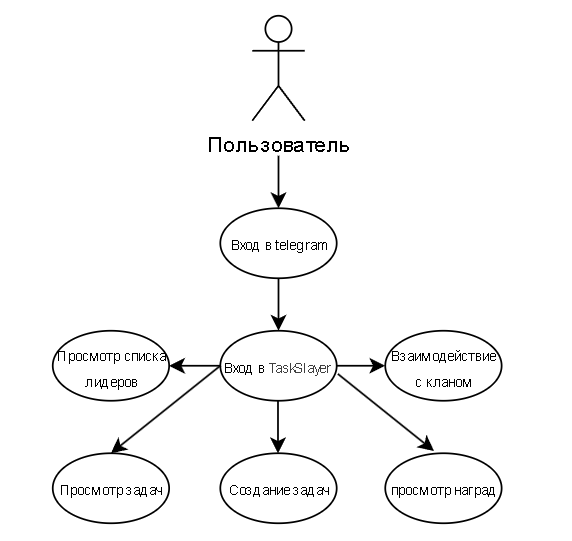
* операционная система Android версия 6.0 или выше;
* операционная система iOS версия 11.0 или выше;
* 100 МБ внутренней памяти;
* доступ к интернету.

Минимальные характеристики Персонального Компьютера для полноценной работоспособности:

* **операционная система** Microsoft Windows 7 и выше, macOS 11 (Big Sur) и выше;
* **процессор** Intel, AMD или Apple Silicon Processor;
* **оперативная память** не менее 4 ГБ;
* **Жёсткий диск** не менее 10 ГБ свободного дискового пространства;

## Проектирование программного изделия

## Диаграмма вариантов использования



1. Диаграмма вариантов

## Архитектура информационной системы

Серверная часть ИС обрабатывает запросы клиентской части (с использованием jwt токенов), организует защиту и хранение данных. Код написан на языке программирования C# с использованием таких подходов как SOLID и Domain-Driven Design (DDD). Для хранения данных был использован postgresSQL.

**SOLID** – это пять ключевых принципов объектно-ориентированного проектирования, обеспечивающих гибкость, масштабируемость и поддержку кода:

* **SRP (Single Responsibility Principle)** каждый класс должен иметь только одну причину для изменения. Один класс = одна ответственность;
* **OCP (Open/Closed Principle)** классы должны быть открыты для расширения, но закрыты для изменения. Новая функциональность добавляется через наследование/интерфейсы без модификации существующего кода;
* **LSP (Liskov Substitution Principle)** объекты подклассов должны заменять объекты родительских классов, не нарушая работу программы. наследники не должны "ломать" поведение базового класса.
* **ISP (Interface Segregation Principle)** много специализированных интерфейсов лучше одного общего. Клиенты не должны зависеть от методов, которые они не используют.
* **DIP (Dependency Inversion Principle** зависимости должны строиться на абстракциях, а не на конкретных реализациях. Модули высокого уровня не зависят от модулей низкого уровня напрямую.

**DDD** – это подход к разработке ПО, ориентированный на глубокое моделирование предметной области (Domain) для точного отражения бизнес-процессов в коде.

Клиентская часть ИС создает запросы и отправляет на серверную часть ИС. Реализует команды для пользователя и графический интерфейс. Программный код написан на C# с использование м подхода SOLID. Для взаимодействия с серверной часть используется NSwag.

NSwag — это современный и гибкий инструмент, предназначенный для автоматической генерации клиентского кода и документации API на основе спецификаций OpenAPI (ранее известного как Swagger). Он предоставляет разработчикам возможность не только создавать описания API в формате OpenAPI на основе существующего ASP.NET Core-приложения, но и генерировать типобезопасный клиентский код на различных языках программирования, включая C# и TypeScript. Это значительно упрощает интеграцию фронтенда с бэкендом, так как позволяет автоматически создавать классы и методы, отражающие структуру и логику веб-сервиса, без необходимости вручную описывать каждый запрос и ответ.

Достоинства решения:

* **высокая модульность и читаемость кода** благодаря применению SOLID и DDD;
* **гибкость архитектуры** — систему легко расширять и модифицировать;
* **надёжная система аутентификации** с использованием JWT;
* **использование PostgreSQL** как отказоустойчивой и производительной СУБД;
* **минимизация ручного труда** за счёт NSwag — генерация клиентского кода и API-документации автоматизирована;
* **поддержка типобезопасности** в клиент-серверном взаимодействии;
* **быстрое масштабирование и сопровождение** благодаря разделению по слоям и внедрению абстракций.

Недостатки:

* **зависимость от сторонних библиотек** (NSwag, JWT), что требует постоянного контроля версий и совместимости;
* **более высокий порог входа для разработчиков** из-за использования DDD и строгих архитектурных подходов;
* **повышенное время разработки** на начальных этапах из-за детального моделирования предметной области;
* **чувствительность к неверному управлению временными зонами** при работе с типами DateTimeOffset и PostgreSQL;
* **сложности при отладке авто-сгенерированного кода**, особенно при ошибках сериализации или конфликтах моделей.

## Защита информационной системы

Защита информационной системы осуществлена с помощью:

* взаимодействия клиента и сервера с помощью jwt токенов;
* двухфакторной авторизации в Telegram;
* проверки на подозрительную активность (50 сообщений в минуту).

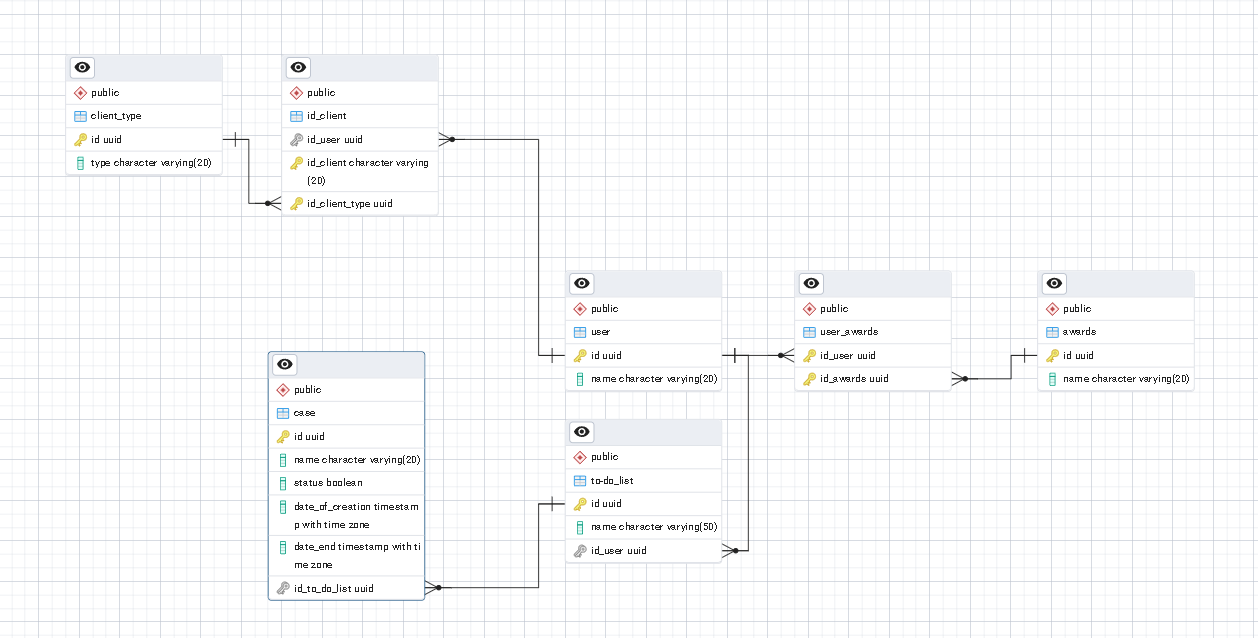
## Описание используемых библиотек

* **Microsoft.AspNetCore.Authentication.JwtBearer** middleware для ASP.NET Core, позволяющий приложению принимать OpenID Connect bearer токены;
* **Microsoft.AspNetCore.Diagnostics** middleware для обработки исключений, отображения страниц ошибок и диагностической информации в ASP.NET Core;
* **Microsoft.AspNetCore.Http.Extensions** общие методы расширения для HTTP-абстракций, заголовков, запросов/ответов и состояния сессии в ASP.NET Core;
* **Microsoft.EntityFrameworkCore** ORM (Object-Relational Mapper) для .NET, поддерживающий LINQ-запросы, отслеживание изменений, обновления и миграции схемы. Работает с PostgreSQL;
* **Microsoft.EntityFrameworkCore.Tools** инструменты Entity Framework Core для консоли диспетчера пакетов NuGet в Visual Studio;
* **Microsoft.Extensions.DependencyInjection.Abstractions** абстракции для внедрения зависимостей (Dependency Injection) в .NET;
* **Microsoft.VisualStudio.Azure.Containers.Tools.Targets** файлы для интеграции инструментов Visual Studio для работы с контейнерами;
* **Npgsql.EntityFrameworkCore.PostgreSQL** провайдер PostgreSQL для Entity Framework Core;
* **Swashbuckle.AspNetCore** инструменты для автоматической генерации документации API (Swagger) в ASP.NET Core;
* **System.IdentityModel.Tokens.Jwt** (Legacy) библиотека предоставляет функционал для создания, сериализации, валидации и обработки JSON Web Tokens (JWT):
* Microsoft.Extensions.ApiDescription.Client пакет для генерации кода и взаимодействия с описанием API в формате OpenAPI в .NET-проектах;
* Microsoft.Extensions.DependencyInjection реализация внедрения зависимостей (Dependency Injection) в проектах на .NET, обеспечивающая слабую связанность компонентов;
* Microsoft.Extensions.Hosting инфраструктура для управления временем жизни приложений, поддержка жизненного цикла приложений (консольных, веб-приложений и сервисов);
* Microsoft.Extensions.Http предоставляет HttpClientFactory, упрощающий конфигурацию, управление временем жизни и расширение HTTP-клиентов в .NET-приложениях;
* Newtonsoft.Json популярная высокопроизводительная библиотека для сериализации и десериализации объектов в JSON-формат;
* NSwag.ApiDescription.Client часть инструментария NSwag, генерирует клиентский код на основе OpenAPI-описания, облегчая интеграцию клиент-серверного взаимодействия;
* Telegram.Bot библиотека, упрощающая создание ботов Telegram на .NET, предоставляя удобные обёртки для API Telegram;
* Telegram.Bots.Extensions.Polling надстройка над Telegram.Bot для организации long polling — способа получения обновлений с сервера Telegram;
* Microsoft.VisualStudio.Azure.Containers.Tools.Targets внутренние цели MSBuild, необходимые для поддержки разработки и развертывания контейнеров в Visual Studio (предварительная версия).

## Разработка программного изделия

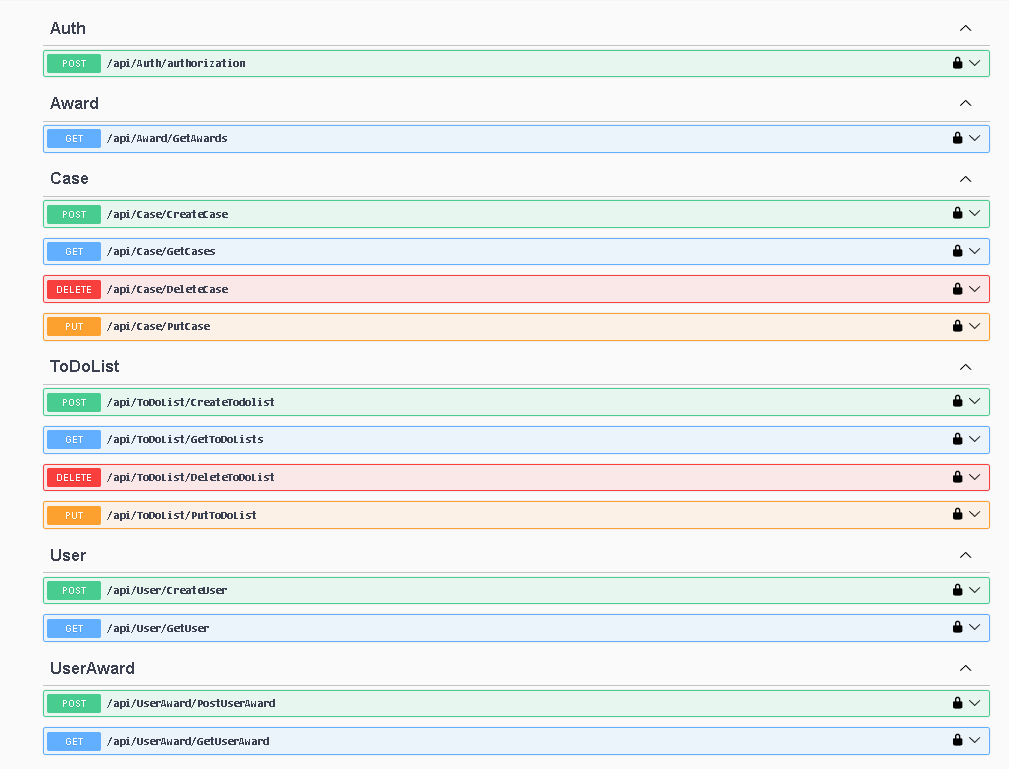
## Описание разработки

Разработка ИС началась с проектирования и реализации базы данных с помощью инструментов pgAdmin4. Диаграмма базы данных изображена на рисунке 10.



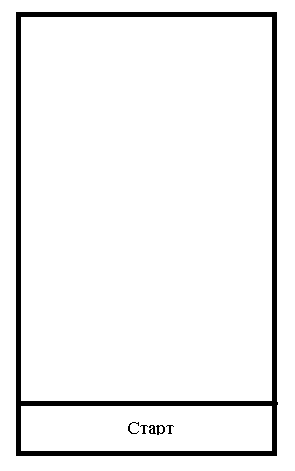
1. Диаграмма базы данных

Разработка серверной часть началась с реализации контроллеров и взаимодействия этих контроллеров с базой данных. С помощью **Microsoft.EntityFrameworkCore были сгенерированы классы базы данных для взаимодействия с PostgreSQL. Проверка работоспособности осуществлялась через Swager. Реализованные контроллеры в swager изображены на рисунке 11.**

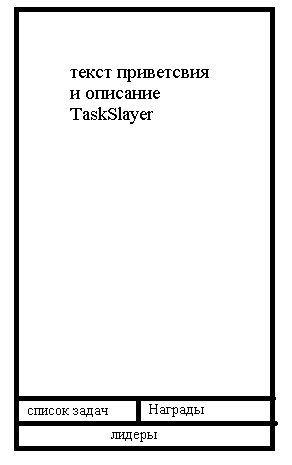
****

1. **Контроллеры серверной части в Swagger**

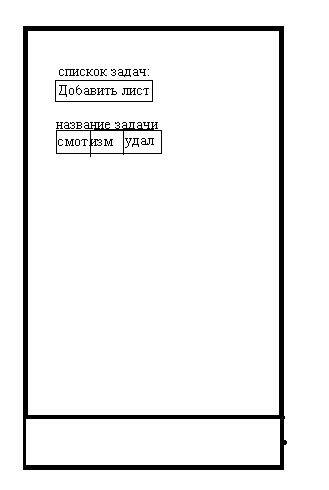
**Разработка клиентской части началась с проектирования макетов интерфейса, изображенных на рисунках.12 – 16.**

****

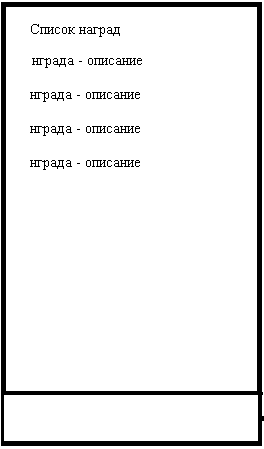
1. **Макет начального экрана**

****

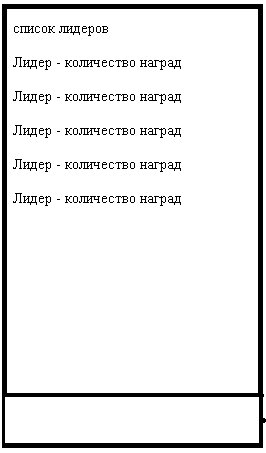
1. **Макет главного меню**

****

1. **Макет списка задач**



1. Макет списка наград



1. Макет списка лидеров

После этапа проектирования и разработки макетов пользовательского интерфейса было выполнено подключение сервиса Telegram для реализации взаимодействия с пользователями. В рамках этой задачи была произведена настройка бота с использованием официального инструмента Telegram — BotFather. С его помощью был создан уникальный бот, назначено имя, задано описание и сгенерирован токен доступа, необходимый для последующей интеграции. Это стало отправной точкой для реализации функционала автоматизированного общения с пользователями через мессенджер.

## Трудности и решения

Во время разработки ИС появлялись определённые ошибки, которые были успешно решены. Трудности и решения представлены в таблице 3.

1. Трудности и решения

| **Трудность** | **Решение** |
| --- | --- |
| Неверное определение временной зоны при работе с DateTimeOffset и PostgreSQL | Приведение всех значений к UTC через ToUniversalTime() и согласование формата времени с ожидаемым API (например, yyyy-MM-ddTHH:mm:ssZ) |
| Ошибки сериализации и десериализации JSON при использовании NSwag и Newtonsoft.Json | Настройка JsonSerializerSettings, ручная корректировка моделей DTO и проверка соответствия типов полям JSON |
| Повышенная сложность архитектуры из-за применения DDD | Разделение проекта на слои (Domain, Application, Infrastructure), использование шаблонов проектирования и внедрение модульного тестирования для упрощения отладки |
| Ошибки при генерации клиентского кода через NSwag (например, при изменении структуры API) | Регулярное обновление OpenAPI-описания и перегенерация клиентов, применение версионирования API |
| Относительно высокий порог входа для новых участников команды | Подготовка внутренней документации, схем архитектуры, и проведение коротких вводных сессий |
| Долгая настройка авторизации и аутентификации с использованием JWT | Использование встроенных возможностей ASP.NET Core (JwtBearerOptions), тестирование через Postman и Swagger |
| Конфликты зависимостей сторонних библиотек | Контроль версий через NuGet, настройка PackageReferences и использование binding redirects при необходимости |
| Ошибки времени выполнения из-за авто-сгенерированного кода | Отладка через трассировку сетевых запросов (например, с помощью Fiddler, Postman или встроенного логгера), проверка сериализованных объектов в runtime |
| Необходимость поддержки типобезопасности между сервером и клиентом | Генерация строго типизированных моделей с помощью NSwag, проверка соответствия с DTO на серверной стороне |

## Перспективы усовершенствования информационной системы

Перспективы усовершенствования информационной системы заключаются в дальнейшем расширении функциональности, повышении удобства использования, производительности и отказоустойчивости, а также в обеспечении более глубокой интеграции с внешними сервисами. Одним из ключевых направлений развития является внедрение системы ролей и прав доступа, что позволит гибко разграничивать возможности пользователей в зависимости от их должностных обязанностей и уровня доверия. Это особенно важно при работе в командной среде или в условиях предприятия, где данные требуют дополнительной защиты.

Также целесообразно рассмотреть возможность внедрения системы уведомлений — как внутренних (в рамках клиентского интерфейса), так и внешних (например, через e-mail или мессенджеры, такие как Telegram). Это обеспечит своевременное информирование пользователей о наступлении сроков задач, изменениях статусов или появлении новых назначений, тем самым улучшая вовлечённость и контроль над рабочими процессами.

С точки зрения технической реализации перспективным направлением является масштабирование системы с использованием облачных технологий, таких как контейнеризация через Docker и развертывание в облаке (например, с применением Azure или AWS). Это позволит не только повысить отказоустойчивость и гибкость системы, но и упростить поддержку и развертывание новых версий.

В части пользовательского интерфейса возможно внедрение более современного фронтенд-фреймворка (например, React или Blazor WebAssembly), что обеспечит лучшую отзывчивость и интерактивность клиентской части. Также стоит рассмотреть реализацию мобильного приложения, синхронизирующегося с основной системой через API, что расширит доступность и удобство использования на различных платформах.

Кроме того, перспективным шагом станет интеграция с системами аналитики и мониторинга, такими как Prometheus или Grafana, что позволит отслеживать поведение системы, фиксировать узкие места и принимать своевременные меры для улучшения производительности. На уровне бизнес-логики возможна реализация механизма автоматической приоритизации задач или рекомендаций на основе анализа активности пользователя и статистики выполнения.

Таким образом, дальнейшее развитие информационной системы может быть направлено как на техническое усовершенствование и повышение производительности, так и на расширение бизнес-функциональности, что обеспечит её долгосрочную актуальность, адаптивность и удобство для конечного пользователя.

## Тестирование программного изделия

# ЗАКЛЮЧЕНИЕ

В результате выполнения ДП была спроектирована и разработана информационная система по геймификации и управлению задачами (на примере приложения «Todoist»).

В ходе выполнения ДП были решены следующие задачи:

* проведен анализ предметной области, в ходе которого изучены ключевые аспекты, структура и взаимосвязи исследуемой сферы. Определены основные проблемы и тенденции, что позволило сформировать базу для дальнейших решений и рекомендаций;
* рассмотрены аналоги информационных систем, проведен сравнительный анализ их функциональности, архитектуры и особенностей реализации. Выявлены ключевые преимущества и недостатки, что позволило определить оптимальные решения для проектируемой системы;
* спроектирована архитектура информационной системы, включающая модульную структуру, интерфейсы взаимодействия и ключевые компоненты. Определены технологии реализации, что обеспечивает масштабируемость, надежность и безопасность системы;
* реализован пользовательский интерфейс ИС, обеспечивающий интуитивную навигацию и удобство взаимодействия. Интерфейс адаптирован под требования пользователей и корректно отображается на различных устройствах;
* реализованы основные функции информационной системы, обеспечивающие выполнение ключевых задач предметной области. Функционал протестирован на соответствие техническим требованиям и пользовательским ожиданиям;
* выполнено тестирование системы, подтвердившее корректность работы всех модулей и соответствие требованиям. Устранены выявленные ошибки, обеспечена стабильная работа ИС в различных условиях;
* реализована защита информационной системы, соответствующая современным стандартам безопасности. Обеспечена безопасность данных и контроль несанкционированного доступа. Принятые меры гарантируют надежную работу системы в условиях потенциальных угроз.

Разработанная ИС позволяет пользователям:

* взаимодействовать с ботом в Telegram;
* создавать новые задачи;
* взаимодействовать с задачами. Удалять, редактировать, изменять статуса;
* получать награды за выполнение задачи;
* просматривать список наград;
* взаимодействовать с кланами. Создавать, просматривать, присоединяться;
* просматривать список лидеров;

Для создания ИС использовалась среда разработки Visual Studio. VS предоставляет весь спектр нужных инструментов для отладки, установки пакетов и взаимодействия с git. ИС написана на языке C#. Данный язык относится к объектно-ориентированным, что благоприятно влияет на структуру кода, обеспечивая его модульность, масштабируемость и легкость поддержки.

В процессе разработки были выявлены следующие трудности:

* подключение к удалённой базе данных;
* способ защиты данных между клиентом с сервером;
* ошибки отображения задач и списка лидеров;

В будущем ИС может быть модернизирована добавлением улучшенной защиты данных, новых фронтенд приложений, увеличением функционала.

В процессе разработки информационной системы были успешно решены поставленные задачи, что позволило создать полнофункциональное решение для геймификации управления задачами. Система интегрирует удобный пользовательский интерфейс, надежную архитектуру и современные механизмы мотивации, обеспечивая комфортное взаимодействие пользователей с их задачами. Несмотря на возникшие технические сложности, все ключевые функции были реализованы, а система готова к дальнейшему развитию и расширению функциональных возможностей.

Разработанное решение демонстрирует эффективное сочетание технологий C# и платформы Telegram, предоставляя пользователям интуитивно понятный инструмент для повышения продуктивности. В перспективе система может быть дополнена новыми функциями, улучшенными механизмами безопасности и поддержкой дополнительных платформ, что расширит ее применение и повысит удобство использования.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

# ПРИЛОЖЕНИЕ А