

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

Направленность Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

Бизнес-проект «Система для исторических квест-экскурсий».
Мобильное приложение для проведения квестов с применением
технологии дополненной реальности

Студент _____ Кудашов Александр Сергеевич
(подпись, дата) (ФИО полностью)

Руководитель _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой _____ Козлов А.Ю.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2024

Экспертное заключение
о согласовании темы ВКР в форме стартапа

Экспертная комиссия в составе: декан ФВТ д.т.н., профессор Л.Р. Фионова, доцент кафедры ВТ, к.э.н., И.В. Акифьев, главный разработчик ООО «Ай-теко ВЦ», к.т.н. М.П. Синев, рассмотрев стартап-проект «Система для исторических квест-экскурсий» Кудашова Александра Сергеевича подтверждает его актуальность, практическую значимость и возможность коммерциализации.

Решение экспертной комиссии: выпускная квалификационная работа на тему «Бизнес-проект «Система для исторических квест-экскурсий». Мобильное приложение для проведения квестов с применением технологии дополненной реальности» Кудашова Александра Сергеевича может выполняться в форме стартапа (бизнес-проекта).

Председатель ЭК

д.т.н., профессор

декан ФВТ

Л.Р. Фионова

Члены ЭК

к.э.н., доцент кафедры ВТ

И.В. Акифьев

к.т.н., главный разработчик

ООО «Ай-теко ВЦ»

М.П. Синев

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ А.Ю. Козлов
«25» декабря 2023 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ В ФОРМЕ
СТАРТАПА

1. Студент Кудашов Александр Сергеевич гр. 20ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>
2. Тема работы Бизнес-проект «Система для исторических квест-экскурсий». Мобильное приложение для проведения квестов с применением технологии дополненной реальности
Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20__ г. _____
2. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Такташкин Д.В.
3. Общая часть ВКР:
Подготовка компьютерной презентации бизнес-проекта (стартапа).
4. Индивидуальная часть ВКР:
4.1. Задание на работу (назначение разработки, исходные данные и т.п.)
Назначение: программное средство предназначено для прохождения исторических экскурсий в формате квестов
Основные функции:
1) Прохождение квестовых заданий
2) Получение исторических справок
3) Получение дополнительной информации в дополненной реальности
4) Получение рейтинга игроков
5) Управление аккаунтом
- Технологии разработки: ООП
Язык программирования: C#
Инструментальные средства разработки: Microsoft Visual Studio 2022 Community Edition, Unity 2022.3.2f1

Среда исполнения: Android 10.0 и выше, Linux, Docker

4.2. Перечень подлежащих разработке вопросов

1. Анализ предметной области и требований к системе. Постановка задачи

2. Проектирование программных средств

3. Реализация программных средств

4. Тестирование программных средств

5. Оформление пояснительной записки

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Разработка бизнес-плана проекта	10%	08.04.2024 – 10.04.2024	
2	Анализ предметной области и требований к системе	10%	11.04.2024 – 13.04.2024	
3	Проектирование программных средств	20%	15.04.2024 – 25.04.2024	
4	Реализация программных средств	30%	26.04.2024 – 12.05.2024	
5	Тестирование программных средств	10%	13.05.2024 – 22.05.2024	
6	Оформление пояснительной записки	20%	23.05.2024 – 03.06.2024	

Дата выдачи задания «21» декабря 2023 г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 170 листов, 41 рисунок, 26 таблиц, 48 использованных источников, 5 приложений.

БИЗНЕС-ПРОЕКТ, ТУРИСТИЧЕСКОЕ МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, АВТОМАТИЗИРОВАННЫЕ КВЕСТ-ЭКСКУРСИИ, ДОПОЛНЕННАЯ РЕАЛЬНОСТЬ, АРХИТЕКТУРА НА ОСНОВЕ СЕРВИСОВ, АРХИТЕКТУРЫ MVx-СЕМЕЙСТВА, POSTGRESQL, ASP.NET CORE, C#

Объект исследования: программные средства организации туристического досуга с геймифицированным подходом.

Целью выпускной квалификационной работы как стартапа является разработка бизнес-плана проекта «Система для исторических квест-экскурсий» и начало его реализации.

Целью индивидуальной части выпускной квалификационной работы как стартапа является разработка туристического мобильного приложения для прохождения исторических квест-экскурсий с применением технологии дополненной реальности.

Технология разработки: ООП.

Среда исполнения: Android 10.0 и выше (клиентское мобильное приложение), Linux, Docker (серверная часть).

Область применения: индивидуальный туризм.

Разработка серверной части и клиентского мобильного приложения проводилась на языке программирования C# в среде разработки Microsoft Visual Studio 2022 Community Edition и игровом движке Unity соответственно. Разработка производилась в операционной системе Windows 11.

					ПГУ 09.03.04 – 08BP201.11 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Кудашов А.С.			Бизнес-проект «Система для исторических квест-экскурсий». Мобильное приложение для проведения квестов с применением технологии дополненной реальности. Пояснительная записка	Лит.	Лист	Листов
Провер.		Такташкин Д.В.					5	170
Реценз.						Группа 20ВП1		
Н. Контр.		Такташкин Д.В.						
Утверд.		Козлов А.Ю.						

Содержание

Введение.....	9
1 Бизнес-проект «Система для исторических квест-экскурсий Questie».....	11
1.1 Постановка цели и задач бизнес-проекта	11
1.2 Описание инновационной идеи	11
1.3 Описание продукта	14
1.4 Анализ рынка.....	16
1.5 Маркетинговый план	17
1.6 Финансовый план	19
1.7 Достигнутые результаты	20
2 Мобильное приложение для проведения квестов с применением технологии дополненной реальности	22
2.1 Аналитическая часть.....	22
2.1.1 Анализ предметной области	22
2.1.2 Сравнительный анализ программных продуктов-аналогов	23
2.1.2.1 Анализ Ubego	24
2.1.2.2 Анализ Questo.....	25
2.1.2.3 Анализ Quest Quiz.....	26
2.1.2.4 Результат проведенного анализа	27
2.1.3 Анализ требований к программным средствам	28
2.1.3.1 Анализ функциональных требований.....	28
2.1.3.2 Анализ нефункциональных требований.....	31
2.1.4 Постановка цели и задач	32
2.2 Проектирование программных средств для проведения квестов с применением технологии дополненной реальности	34
2.2.1 Проектирование архитектуры программных средств.....	34
2.2.1.1 Проектирование архитектуры серверной части	35
2.2.1.1.1 Выбор архитектуры серверной части	35
2.2.1.1.2 Разбиение серверной части на сервисы.....	40

2.2.1.1.3 Проектирование последовательности выполнения запросов	43
2.2.1.1.4 Проектирование основной логики обработки запросов	45
2.2.1.2 Проектирование архитектуры клиентского мобильного приложения.....	47
2.2.2 Проектирование логической модели базы данных	53
2.2.3 Проектирование пользовательского интерфейса клиентского мобильного приложения	55
2.2.4 Планирование разработки и оценка бюджета.....	57
2.2.5.1 Организация производства	57
2.2.5.2 Календарный план разработки	58
2.2.5.3 Расчет себестоимости разработки программного продукта	60
2.3 Реализация программных средств для проведения квестов с применением технологии дополненной реальности	63
2.3.1 Реализация пользовательского интерфейса клиентского мобильного приложения	73
2.3.2 Реализация архитектуры программных средств.....	63
2.3.2.1 Реализация архитектуры клиентского мобильного приложения.....	63
2.3.2.2 Реализация архитектуры серверной части	63
2.3.3 Реализация базы данных	71
2.3.4 Тестирование и оценка качества кода	76
2.3.4.1 Функциональное тестирование	76
2.3.4.2 Структурное тестирование	79
2.3.4.3 Модульное тестирование	82
2.3.4.4 Метрики качества кода.....	83
Заключение	87
Список использованных источников	90
Приложение А (справочное) Глоссарий	96

Приложение Б (рекомендуемое) Результаты бизнес-проекта	99
Приложение В (рекомендуемое) Исходный код программного обеспечения.	105
Приложение Г (рекомендуемое) Функциональное тестирование	148
Приложение Д (рекомендуемое) Исходный код модульного тестирования ...	159

Введение

Развитие туризма, как и большинства других отраслей, активно цифровизируется [1, 2], что позволяет внедрять новые технологии, направленные на улучшение качества услуг и персонализацию путешествий.

Бизнес-проект «Система для исторических квест-экскурсий» предназначен для предоставления туристам возможности проведения досуга в геймифицированной форме с познавательным сопровождением, что должно помочь им в знакомстве с новыми городами, повышении уровня исторических знаний, планировании и организации путешествий, а также способствовать развитию отечественного туризма.

Актуальность бизнес-продукта обоснована следующими фактами:

- Правительство России ведет активную политику по поддержке и развитию отечественного туризма;
- по данным ВЦИОМ 96% россиян считают важным знание истории своей страны и своего региона;
- турпоток по городам России за три квартала 2023 года по данным Росстата составил 134 955 161 поездок, что на 16,48% больше, чем за тот же период 2022 года;
- политическая обстановка привела к ограниченному выезду россиян за границу.

Происходит переориентация потока на внутренний туристический рынок. Около пяти миллионов туристов (примерно 150 млрд рублей в год в денежном эквиваленте) аккумулируются внутри страны, что влечет за собой необходимость развития и повышения качества туризма в России.

Требуется развивать туризм в регионах и привлекать внимание к менее популярным направлениям, создавать туристические маршруты. Разрабатываемый продукт сможет разгрузить туристические потоки в крупных городах. Необходимость внедрения технологии дополненной реальности в проект объясняется увеличением благодаря ей уровня вовлеченности

пользователей в экскурсионный процесс и повышением заинтересованности. Так, по результатам анализа, проведенного webinar.ru, внедрение технологии AR увеличивает вовлеченность пользователей в процесс взаимодействия с приложениями в среднем на 40%. Кроме того, по данным агентства Gamification Now! использование в приложениях подхода геймификации увеличивает вовлеченность пользователей на 30%.

Целевой аудиторией системы для исторических квест-экскурсий станут индивидуальные туристы, в основном в возрасте от 18 до 50 лет.

Основным источником привлечения клиентов должна стать целевая реклама в поисковых системах, социальных сетях, а также реклама у блогеров-путешественников.

Определения, используемые в данной выпускной квалификационной работе, в приложении А.

1 Бизнес-проект «Система для исторических квест-экскурсий Questie»

1.1 Постановка цели и задач бизнес-проекта

Сформулированной в ходе анализа задания целью данной выпускной квалификационной работы как стартапа является разработка бизнес-плана проекта «Система для исторических квест-экскурсий».

Для достижения поставленной цели необходимо выполнить следующие задачи:

- выполнить детализированное описание инновационной идеи;
- сформировать описание предлагаемого продукта и его технико-экономические характеристики;
- провести анализ рынка и спрогнозировать развитие отрасли;
- разработать маркетинговый план, описывающий ценообразование продукта и способ его продажи;
- разработать финансовый план, содержащий этапы развития бизнес-проекта и описывающий ожидаемые финансовые показатели на каждом из этапов;
- найти партнеров проекта и привлечь финансирование на первый этап разработки.

1.2 Описание инновационной идеи

Экскурсионная деятельность является основой формирования туристского продукта. Современный рынок туристской индустрии требует внедрения инновационных технологий при организации экскурсионной деятельности. Высокий уровень конкуренции среди туристских компаний обуславливает необходимость постоянного развития, отслеживания тенденций потребительского спроса и внедрения новых форматов [1]. Одним из новых

форматов, набирающих популярность среди путешественников, является квест-экскурсия.

Квест-экскурсия – это услуга по организации посещения специально отобранных объектов экскурсионного показа, предполагающая наличие сюжетной линии и препятствий в форме различных задач, головоломок, игр, разгадывая которые участники знакомятся и изучают конкретные объекты посредством использования имеющихся знаний, наблюдения и общения с другими субъектами [2].

Квест-экскурсия является комбинацией квеста и экскурсии, включающей в себя элементы обоих понятий, приведенные на рисунке 1.

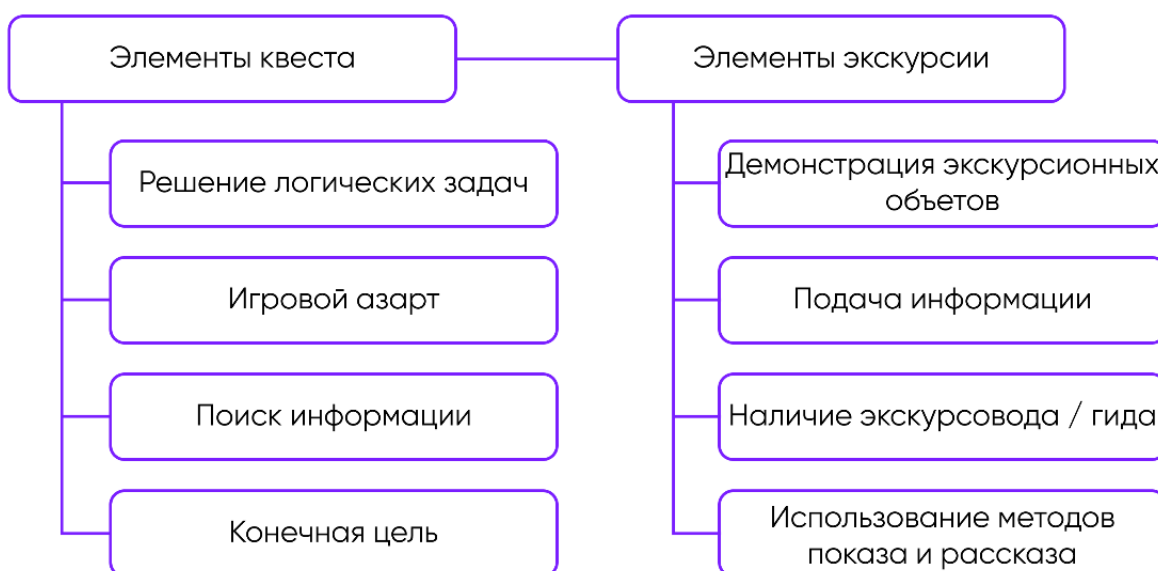


Рисунок 1 – Составляющие квест-экскурсии

Развитие цифровизации приводит к появлению различных мобильных приложений, покрывающих все сферы жизни человека. Туризм не стал исключением. Так, описанные выше квест-экскурсии переносятся в мобильные приложения, становясь удобным и интересным дополнением для путешествий, не требующим непосредственного контакта с гидом или экскурсоводом.

Повышает интерес пользователей к продукту также и использование технологии дополненной реальности (AR). Согласно исследованию компании

Neuro-Insight Study, дополненная реальность почти втрое увеличивает уровень когнитивной активности мозга во время обучения, по сравнению с аналогичным опытом, не связанным с AR. При этом, формируются устойчивые связи между объектом изучения (на котором размещена AR метка) и связанным с ним контентом (полученном в AR режиме с этого объекта) [3].

Разрабатываемая система должна предоставлять возможность прохождения квест-экскурсий с применением технологии дополненной реальности, а также обеспечивать необходимые административные функции, такие как управление пользователями, конструирование квестов и отслеживание статистики. Наиболее трудоемкая задача при разработке такой системы – наполнение контентом, самой сложной частью которого является создание 3D-объектов для генерации AR-таргетов. Этот процесс необходимо максимально автоматизировать для снижения стоимости продукта. Для этого планируется создать и применить комбинированный метод построения 3D-моделей, основанный на нейросетевом анализе и позволяющий использовать меньшее количество исходных данных.

Исходя из этого, можно сделать вывод, что система должна состоять из трех компонентов:

- мобильное приложение для проведения квестов с применением технологии дополненной реальности;
- приложение администратора (web-приложение для конструирования квестов и анализа статистики);
- нейросетевой программный модуль для генерации 3D-моделей.

Предлагается создать экскурсионные маршруты на основе квестов с элементами дополненной реальности в формате приложения для мобильных устройств. Приложение должно позволять пользователям выбирать квесты в городе своего расположения, получать баллы за выполнение заданий и доступ к историческим справкам, следить за достижениями друзей. Уникальность

проекта заключается в использовании элементов дополненной реальности на станциях квестов.

Приложение администратора должно упрощать работу администраторов системы и создателей маршрутов квест-экскурсий, предоставляя им возможность конструировать квест-экскурсии, отслеживать статистику по квестам и пользователям, управлять пользователями системы.

Инновационная идея реализуется в несколько этапов. Первым этапом является исследование алгоритмов генерации 3D-моделей по изображениям, разработка программного модуля генерации 3D-моделей по изображениям, исследование зависимости качества сгенерированных 3D-моделей от входных параметров модуля генерации, а также исследование эффективности различных моделей баз данных в зависимости от объема выборки. Вторым этапом является разработка клиентского мобильного приложения с применением технологии дополненной реальности и приложения администратора. Далее будет проводиться апробация минимального жизнеспособного продукта и его вывод на рынок.

Приобретение аппаратных средств и неэксклюзивных прав на программное обеспечение не потребовало привлечения дополнительных финансов и было осуществлено из средств гранта УМНИК.

Ключевым фактором успешной реализации плана является научно-техническое превосходство, состоящее в результатах, достигнутых в ходе НИОКР в рамках реализации проекта УМНИК. Основными рисками реализации плана являются нехватка финансирования на реализацию маркетинговых кампаний, возможность возникновения несовместимости с некоторыми устройствами пользователей, не обладающими поддержкой ARCore (в случае с операционной системой Android), неудовлетворенность пользователей контентом и быстрое изменение их предпочтений.

1.3 Описание продукта

Описание продукта приведено в таблице 1.

Таблица 1 – Описание продукта

Характеристика	Описание
Наименование продукта	Программный комплекс для разработки и проведения городских квест-экскурсий с применением технологии дополненной реальности
Назначение и область применения	Предназначен для организации научно-развлекательного досуга групп людей в сфере самостоятельного туризма [4] путем проведения квест-экскурсий по туристическим маршрутам и историческим, культурным и природным достопримечательностям
Перспективы выпуска на рынке	Целевым рынком является рынок самостоятельного туризма. Потенциальные покупатели – туристы в возрасте от 18 до 50 лет, путешествующие по городам России (около 50 млн человек, из которых лишь 25 млн – платежеспособная аудитория); туристические агентства, заказывающие уникальные квесты под требования бизнеса. Способ поставки – мобильное приложение, устанавливаемое на мобильные телефоны пользователей и являющееся конечным продуктом.
Возможность замещения	В настоящее время подавляющая доля рынка приходится на отечественные приложения для проведения автоматизированных городских квест-экскурсий. Продукты иностранных компаний представлены лишь в нескольких городах. Замещение отечественных аналогов возможно за счет более качественной реализации программного обеспечения, внедрения инновационного функционала, а также лучшей проработанности сценариев квестов.
Технико-экономические характеристики	Разрабатываемая система должна представлять собой мобильное приложение (ПО) под управлением операционной системы Android версии 10.0 и новее. Телефон должен обладать не менее 350 Мб свободной памяти в хранилище и 2 Гб оперативной памяти. Обязательно наличие камеры, поддержка AR Core и доступ к сети-интернет. Модуль генерации 3D-моделей по изображениям реализован с применением языка программирования Python 3, фреймворка машинного обучения PyTorch и библиотек NumPy для обработки многомерных массивов, PyOpenGL, trimesh для визуализации. Компьютер для запуска модуля генерации 3D-моделей (для администратора системы) должен обладать следующими рекомендуемыми техническими характеристиками: объем оперативной памяти: 64 Гб, объем памяти видеокарты: 24 Гб, объем свободной памяти в хранилище: 100 Гб. Оптимальное количество фотографий для генерации 3D-моделей – 50 шт., что позволяет производить генерацию в среднем за 5 минут на видеокарте RTX 3090 Ti.
Конкурентоспособность продукта	Основана на научно-техническом превосходстве, состоящем в использовании комбинированного метода построения 3D-моделей, используемых для построения AR-сцен, основанного на нейросетевом анализе. Данный метода позволяет сделать создание AR-контента менее трудозатратным и недорогим, а также обеспечивает возможность распознавания 3D-таргетов со всех сторон.

Продолжение таблицы 1

Характеристика	Описание
Потенциал повышения конкурентоспособности	Возможно дальнейшее проведение НИОКР с целью совершенствования методов работы над созданием AR-контента. Кроме того, повышению конкурентоспособности может способствовать качественная проработка сценариев квест-экскурсий.
Необходимость лицензирования	Отсутствует необходимость в приобретении сторонних лицензий. Права на РИД защищены.
Степень готовности	К началу выполнения выпускной квалификационной работы как стартапа командой проекта разработан комбинированный метод построения 3D-моделей, используемых для построения AR-сцен, основанный на нейросетевом анализе; разработано программное средство, реализующее данный метод; выполнено прототипирование пользовательского интерфейса мобильного приложения.
Безопасность и экологичность	Продукт не несет и не может нести угроз окружающей среде.

В данном разделе был описан разрабатываемый продукт и его основные характеристики.

1.4 Анализ рынка

Анализ рынка приведен в таблице 2.

Таблица 2 – Анализ рынка

Характеристика	Описание
Объем рынка	На основании данных Яндекс.Вордстат российский рынок экскурсий составляет около 15 млрд рублей в год. Оценка объема рынка по модели TAM/SAM/SOM. TAM – 15,3 млрд. рублей. SAM – 219.9 млн. рублей. SOM – 1,4 млн. рублей.
Темпы роста рынка	Рынок отечественного туризма в последние годы активно растет. Политическая обстановка привела к ограниченному выезду россиян за границу. Происходит переориентация потока на внутренний туристический рынок. Около 5 миллионов туристов аккумулируются внутри страны. Турпоток по городам России за три квартала 2023 года по данным Росстата составил 134 955 161 поездок, что на 16,48% больше, чем за тот же период 2022 года.
Производственные возможности	Производство продукта требует лишь разовых первоначальных затрат на разработку. Дальнейшие производственные работы сводятся к наращиванию базы квестов с целью масштабирования, а также к доработке и внедрению нового функционала с целью поддержания и увеличения уровня интереса аудитории.
Характер отрасли	Туристическая отрасль имеет сезонный характер, но в долгосрочной динамике стабильно развивается.

Продолжение таблицы 2

Характеристика	Описание
Динамика продаж за последние 5 лет	На динамику продаж за последние пять лет существенным образом повлияли такие факторы как пандемия, повлекшая за собой резкое падение отрасли, а также политическая ситуация, способствовавшая перенаправлению потока отечественных туристов на внутренний рынок. По результатам более стабильного 2023 года рост продаж отрасли составил по разным оценкам от 10% до 15%. В 2024 году предполагается рост продаж еще на 15%.
Прогнозы развития отрасли	Согласно Стратегии развития туризма от 20.09.2019 [5] к 2035 году планируется: увеличить количество внутренних туристских поездок на одного жителя более чем в два раза; увеличить экспорт туристских услуг более чем в три раза; увеличить инвестиции в сферу туризма в три раза. В 2020 году показатель количества туристов оценивался в 45,22 млн. человек. К 2025 году эту цифру планируется увеличить в 1,8 раза, до 83,78 млн. За следующие пять лет этот показатель должен вырасти еще более чем в полтора раза, до 140,29 млн. человек. Наконец, к 2035 году число туристов в России должно составить почти 144 млн человек, треть из которых – иностранцы [6].
Основные и потенциальные конкуренты	Основные конкуренты: Ubego – лидер рынка отечественных сервисов автоматизированных городских квестов; Quest Quiz – развивающийся сервис, автоматизирующий проведение квизов и викторин; SurpriseMe – развивающийся сервис, автоматизирующий проведение квизов; Urban Quest – развивающийся сервис, автоматизирующий проведение квизов. Потенциальные конкуренты: аудиогиды, например, izi.TRAVEL (более 1 млн. скачиваний), Sputnik8, YouTravel.Me, GetYourGuide, WeGoTrip, TravelRu и другие.
Уровень рентабельности отрасли	По оценкам рентабельности смежных направлений отрасли чистая рентабельность должна составить от 9% до 11% [7], что позволяет сделать вывод о высокой рентабельности проекта.

Таким образом, был проанализирован рынок сбыта продукта, выявлены прямые и косвенные конкуренты и сделаны прогнозы развития отрасли до 2035 года. Выявлено, что отрасль имеет высокую рентабельность и постоянно развивается.

1.5 Маркетинговый план

Для продвижения разрабатываемой системы для исторических квест-экскурсий с применением технологии дополненной реальности на туристическом рынке, привлечения и удержания целевой аудитории был разработан маркетинговый план, приведенный в таблице 3.

Таблица 3 – Маркетинговый план

Характеристика	Описание
Конечные потребители	Туристы в возрасте от 18 до 50 лет, путешествующие по городам России (около 50 млн чел., из которых лишь 25 млн – платежеспособная аудитория); туристические агентства, заказывающие уникальные квесты под требования бизнеса.
Уровень удовлетворения спроса	Представленные в данный момент на рынке продукты предоставляют квест-экскурсии лишь в 50 городах России, не перекрывая всю географию страны. В большинстве городов присутствует лишь по 1 – 2 квест-экскурсии.
Особенности сегмента рынка	Конкурентное преимущество продукта зависит от характеристик и разнообразия возможностей программного обеспечения для прохождения квестов. Также большое значение имеет качество проработанности сюжетов, что влияет на повторной использование продукта при выборе других квестов. Отрасль имеет сезонный характер.
Конкурентные преимущества товара (услуги) конкурентов	Ubego – большое количество квестов в разных городах; Quest Quiz – разнообразные форматы квестов; SurpriseMe – разнообразные форматы заданий, предложение для бизнеса.
Планируемая доля рынка	За 3-ий этап (с 05.2025 по 04.2026) реализации проекта планируется занять 0,64% рынка. За 4-ый этап (с 05.2026 по 04.2027) реализации проекта планируется занять 9,58% рынка. За 5-ый этап (с 05.2026 по 07.2028) реализации проекта планируется занять 31,96% рынка.
Патентная ситуация	Разработанное программное обеспечение будет зарегистрировано в качестве программы для ЭВМ в Федеральном институте промышленной собственности (ФИПС) с целью предотвращения прямого копирования. Копирование внутренних алгоритмов работы системы будет затруднительно выявить.
Обоснование цены на продукт	На начальном этапе развития проекта предполагается установить цену на квест-экскурсии ниже цен аналогичных продуктов, продающихся за 800 – 2600 рублей за квест. Во-первых, это даст дополнительное конкурентное преимущество. Во-вторых, снизит недовольство клиентов от возможных программных сбоев на стадии вывода первой версии продукта на рынок. С учетом необходимости покрытия операционных расходов, предполагаемая средняя цена может составлять 450 рублей за квест и увеличиваться на дистанции в процессе улучшения предлагаемого продукта.
Организация сбыта	Предполагается коммерциализировать продукт в двух направлениях. Основное направление коммерциализации – b2c: продажа доступа к платным квест-маршрутам пользователям мобильного приложения. По направлению b2b предполагается продажа рекламы в приложении компаниям из сферы услуг, расположенным на туристических маршрутах, а также сотрудничество с туристическими агентствами: создание эксклюзивных квестов под требования бизнеса.

Таким образом, был разработан маркетинговый план для программного комплекса для разработки и проведения городских квест-экскурсий с применением технологии дополненной реальности. Описана целевая аудитория продукта и особенности организации сбыта.

1.6 Финансовый план

Финансовый план приведен в таблице 4.

Таблица 4 – Финансовый план

Этап	Описание работ
Первый этап реализации (с 08.2023 по 05.2024)	Август 2023 – ноябрь 2023: разработка комбинированного метода построения 3D-моделей, используемых для построения AR-сцен, основанного на нейросетевом анализе и программного средства, реализующего данный метод. Декабрь 2023 – май 2024: разработка первой версии мобильного приложения для проведения исторических квест-экскурсии, а также административной панели. Июнь 2024: разработка первых туристических маршрутов по городу Пензе. Июль 2024: апробация минимального жизнеспособного продукта в некоммерческом формате на фокусных группах.
Второй этап реализации (с 08.2024 по 04.2025)	Август 2024: завершение обязательств по грантовой программе УМНИК. Сентябрь 2024 – декабрь 2024: исправление и доработка начального функционала в соответствии с результатами апробации минимального жизнеспособного. Январь 2025: регистрация юридического лица, поиск сценаристов исторических квестов (общая сумма около 100000 рублей). Февраль 2025: разработка туристических маршрутов и сценариев исторических квестов (60000 рублей). Март 2025: пробная маркетинговая компания (e-mail рассылка, целевая реклама в Интернет) (50000 рублей); разработка туристических маршрутов и сценариев исторических квестов (60000 рублей). Апрель 2025: подготовка заявки на конкурс Старт-1.
Третий этап реализации (с 05.2025 по 04.2026)	Май 2025 – апрель 2026: настройка маркетинговой компании (70000 рублей в месяц); разработка туристических маршрутов и сценариев исторических квестов (60000 рублей); первые продажи (предполагается 3000 продаж на 1,4 млн. рублей).
Четвертый этап реализации (с 05.2026 по 04.2027)	Май 2026: получение средств по программе Старт-1 (до 1 млн. рублей) Май 2026 – апрель 2027: работы над улучшением функционала системы; настройка маркетинговой компании; разработка туристических маршрутов и сценариев исторических квестов; увеличение среднего чека продажи; продажи (предполагается 30000 продаж на 21 млн. рублей).
Пятый этап реализации (с 05.2026 по 07.2028)	Май 2027: получение средств на реализацию второго этапа программы Старт-1. Май 2027 – июль 2028: поиск оптимальной модели реализации продукта, перевод исполнителей в штат; выход на самоокупаемость; поиск путей масштабирования бизнеса.

Таким образом, был разработан финансовый план на первые 5 лет реализации программного комплекса для разработки и проведения городских квест-экскурсий с применением технологии дополненной реальности. Выделены основные этапы разработки и продвижения продукта. Увеличение продаж запланировано за счет постепенного увеличения экскурсионных маршрутов. Финансовая модель 3 – 5 этапов плана приведена в таблице 5.

Таблица 5 – Финансовая модель 3 – 5 этапов плана развития проекта

	3 этап	4 этап	5 этап
Объем продаж, ед.	3000	30000	75000
Количество клиентов	1500	15000	25000
Средний чек, руб.	467	700	937
Выручка, тыс. руб.	1401	21000	70275
Расходы, тыс. руб., в т.ч.:	1600	15000	32775
Переменные, тыс. руб.	720	7200	12000
Постоянные, тыс. руб.	880	7800	20775
Операционная прибыль, тыс. руб.	-199	6000	37500
Маржа, %	-14,2%	25,6%	53,3%
Грант, тыс. руб.	3000	Отсутствует	Отсутствует
Инвестиции, займы, тыс. руб	Отсутствуют	Отсутствуют	Отсутствуют

Из составленной финансовой модели видно, что уже на четвертом этапе реализации проект должен выйти на самоокупаемость и начать приносить первую прибыль.

1.7 Достигнутые результаты

Реализуемому проекту одобрено грантовое финансирование в размере 500000 рублей (Договор №18446ГУ/2023 от 16.08.2023) по программе УМНИК-2022 (Пензенская область) [8].

Компания ООО «Открытые решения» выразила поддержку проекту и заявила о готовности к консультативной и информационной помощи. Кафедра «Маркетинг, коммерция и сфера обслуживания» Института экономики и управления выразила готовность к экспертной поддержке проекта в разработке содержательной части квест-экскурсий (туристических маршрутов и сюжетов для заданий квестов). Министерство культуры и туризма Пензенской области выразило заинтересованность в реализации и продвижении проекта. Все

организации выразили готовность поддержки проекта в своих информационных ресурсах. Письма поддержки приведены в приложении Б.

Была проведена работа со СМИ, в выпуске №15 (240) от 24 октября 2023 молодежной газеты Пензенской области «Деловой» была опубликована статья «По Пензе минувших эпох» о проекте системы для исторических квест-экскурсий с применением технологии дополненной реальности [9].

Пройдена преакселерационная программа «Дотком инновации» [10] и акселерационная программа Пензенского государственного университета «Делать вместе новое» при поддержке стартап-студии «Открытые инновации» [11], позволившие качественнее проработать идею проекта и повысить компетенции членов команды. Сертификаты приведены в приложении Б.

К началу реализации ВКР членами команды был разработан программный модуль генерации 3D-моделей по изображениям, проведено исследование алгоритмов генерации 3D-моделей, качества сгенерированных моделей в зависимости от количества входных изображений. Выявлено оптимальное количество входных параметров для разработанного программного модуля генерации 3D-моделей по изображениям, наилучшим образом подходящая для использования в программной системе.

2 Мобильное приложение для проведения квестов с применением технологии дополненной реальности

2.1 Аналитическая часть

2.1.1 Анализ предметной области

Организация путешествий – сложный и трудоемкий процесс, требующий комплексного подхода. Существует множество средств его автоматизации, в том числе и ряд мобильных приложений, которые можно разделить по следующим категориям:

- карты и навигаторы, помогающие не заблудиться в незнакомой местности;
- планировщики, позволяющие объединить информацию о всех запланированных на время поездки событиях и спланировать время и маршруты;
- транспортные приложения, помогающие отслеживать расписание самолетов, поездов и рейсовых автобусов;
- агрегаторы жилья (отелей, гостиниц, квартир, каучсерфинга);
- языковые приложения (переводчики, сборники разговорной лексики);
- агрегаторы мест общественного питания (как известных ресторанов, так и небольших кафе, известных только местным жителям);
- приложения о культурных событиях;
- путеводители, предоставляющие готовые туристические маршруты, информацию об исторических местах и достопримечательностях.

Именно о последней категории мобильных приложений для путешественников далее и пойдет речь более подробно.

Согласно ГОСТ Р 7.0.60-2020 под путеводителем понимается справочник, содержащий сведения о каком-либо географическом пункте или культурно-просветительном учреждении (мероприятии), расположенные в порядке, удобном для следования или осмотра [12].

На рынке мобильных приложений существует большое количество туристских путеводителей с разной направленностью: поиск ближайших музеев; предоставление перечня туристских достопримечательностей; содержание информации об интересных событиях, проходящих неподалеку и многое другое [13].

Особое место среди электронных путеводителей занимают мобильные приложения, организующие туристические маршруты в интерактивном формате квестов. По данным агентства Gamification Now! использование в приложениях подхода геймификации увеличивает вовлеченность пользователей на 30% [14].

Квест-экскурсия – это посещение достопримечательностей, связанных с той или иной тематикой, с использованием различного рода загадок и головоломок, без использования традиционных экскурсоводов [15].

Подобный подход к организации туристических маршрутов имеет ряд преимуществ, таких как вовлеченность в процесс, неограниченность времени использования, свобода перемещения, отсутствие зависимости и «давления» от экскурсовода, игровое соперничество с другими туристами, стоимость и др.

Инновационным элементом электронных исторических квест-экскурсий может стать применение технологии дополненной реальности, что по результатам анализа, проведенного сотрудниками онлайн платформы webinar.ru, увеличивает вовлеченность пользователей в процесс взаимодействия с приложениями в среднем на 40% [16], а также способно повысить конверсию продаж на 94% [17].

2.1.2 Сравнительный анализ программных продуктов-аналогов

Рассмотрим существующие на рынке мобильные приложения для проведения квест-экскурсий.

2.1.2.1 Анализ Ubego

Приложение Ubego (рисунок 2) – лидер российского рынка квест-экскурсий без экскурсоводов, имеющий более 10 тысяч скачиваний в магазине приложений Google Play и предоставляющий доступ к квестам в более чем 50 городах.

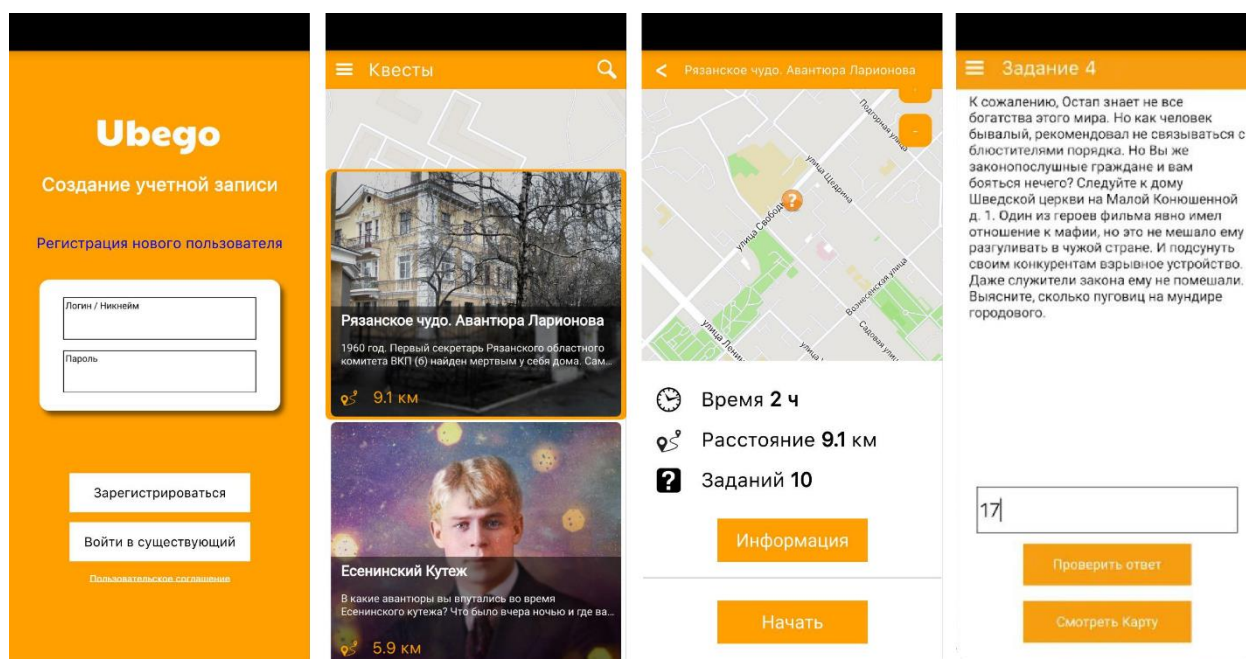


Рисунок 2 – Мобильное приложение Ubego

Преимущества:

- большое количество доступных квестов;
- возможность подключения нескольких игроков к одному квесту;
- отсутствие необходимости в сопровождающем экскурссии;
- возможность стать автором квест-экскурсии, создав ее через web-интерфейс.

Недостатки:

- нет возможности продолжить прерванный квест с места остановки после перезапуска приложения;
- однотипность заданий [18];
- орфографические и грамматические ошибки;
- фактические ошибки в исторических справках.

2.1.2.2 Анализ Questo

Questo (рисунок 3) – англоязычная румынская разработка, предоставляющая доступ к квест-экскурсиям в 153 городах по всему миру.

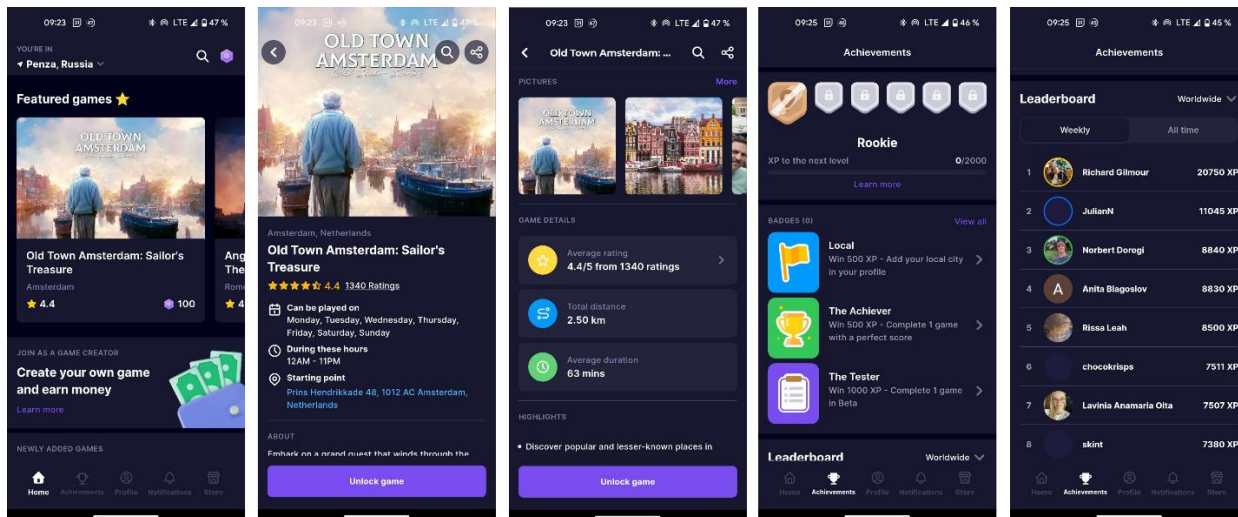


Рисунок 3 – Мобильное приложение Questo

Преимущества:

- большое количество доступных квестов;
- применение технологии дополненной реальности на ключевых точках маршрутов;
- рейтинговая таблица участников;
- наличие небольшого числа бесплатных квестов;
- возможность подключения нескольких игроков к одному квесту;
- отсутствие необходимости в сопровождающем экскурсии;
- возможность стать автором квест-экскурсии.

Недостатки:

- отсутствие квестов в России;
- отсутствие поддержки русского языка;
- сложная система оплаты квестов;
- проблемы со сканированием некоторых таргетов дополненной реальности.

2.1.2.3 Анализ Quest Quiz

Quest Quiz (рисунок 4) – развивающийся онлайн-редактор квестов, предоставляющий доступ к 28 городским квестам. Многие квесты имеют исключительно развлекательную направленность без исторического контекста.

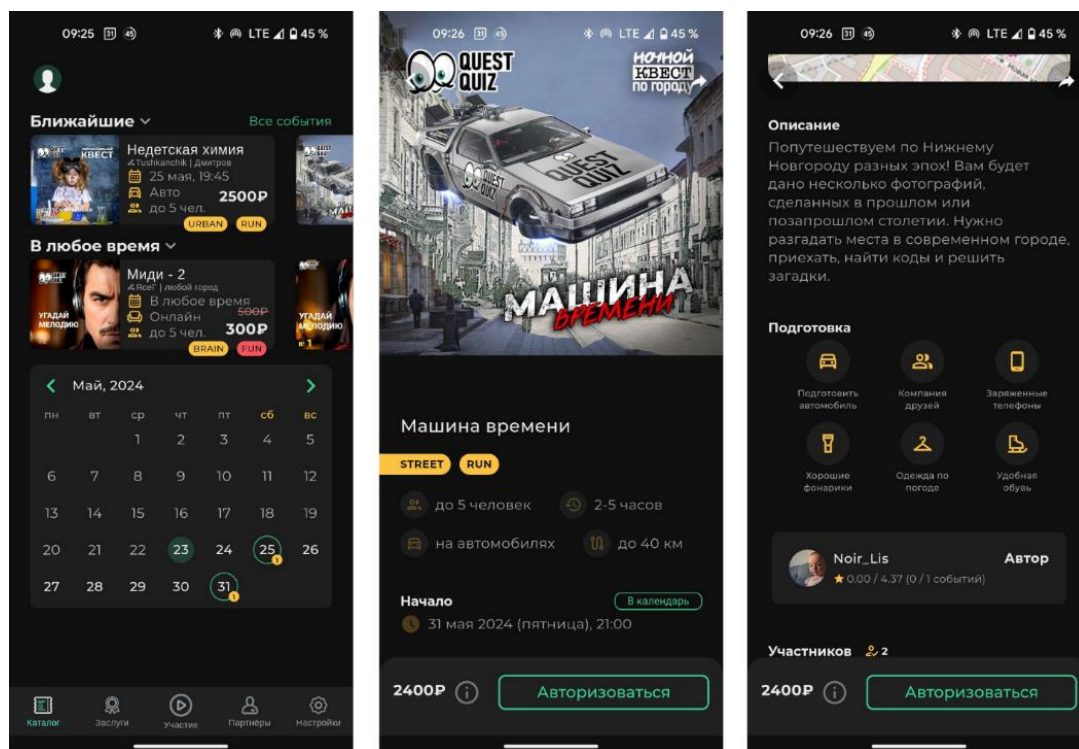


Рисунок 4 – Мобильное приложение Quest Quiz

Преимущества:

- разнообразные форматы квестов;
- возможность подключения нескольких игроков к одному квесту;
- активное развитие функционала приложения;
- возможность стать автором квест-экскурсии (качественный web-интерфейс для конструирования).

Недостатки:

- необходимость в сопровождающем-организаторе во многих квестах;
- привязка большинства квестов к конкретным датам.

2.1.2.4 Результат проведенного анализа

Для обобщения проведенного анализа аналогов была составлена сводная таблица по ключевым характеристикам систем для проведения городских квест-экскурсий (таблица 6).

Таблица 6 – Ключевые характеристики аналогичных систем для проведения городских квест-экскурсий

Характеристика	Ubego	Questo	Quest Quiz
Командная игра	Есть	Есть	Есть
Необходимость в организаторе	Нет	Нет	Есть
Проработанность сюжетов	Низкая	Средняя	Средняя
Качество технической реализации	Частые сбои	Редкие сбои в неосновном функционале	Сбои отсутствуют
Возможность стать автором квестов	Есть	Есть	Есть
Соперничество с другими игроками	Отсутствует	Рейтинговая таблица	Большинство квестов организуется для нескольких команд
Применение технологии дополненной реальности	Нет	Есть	Нет
Стоимость	От 500 до 2000 рублей	В среднем 1200 рублей	От 2200 до 2600 рублей

Из приведенной таблицы видно, что для создания конкурентно способного приложения необходимо реализовать ключевые преимущества

аналогов (современный и интуитивно-понятный пользовательский интерфейс, отсутствие технических неисправностей, возможность соперничества с другими игроками, применение технологии дополненной реальности, качество проработки сюжетов и др.), а также устранить возможность возникновения потенциальных недостатков.

2.1.3 Анализ требований к программным средствам

2.1.3.1 Анализ функциональных требований

В ходе анализа предметной области было выявлено, что разрабатываемое мобильное приложение для проведения квестов с применением технологии дополненной реальности должно предоставлять пользователям следующий функционал:

- выбирать исторические квесты;
- выполнять квестовые задания;
- получать подсказки во время прохождения квестов;
- получать карту маршрута во время прохождения квестов;
- получать исторические справки во время прохождения квестов;
- запускать сцены дополненной реальности при выполнении заданий;
- получать игровые баллы за пройденные квесты;
- просматривать рейтинг игроков;
- получать статистику текущего игрового профиля;
- управлять игровым профилем.

Доступ к функционалу мобильного приложения должны иметь только авторизованные пользователи.

Для наглядного описания поведения разрабатываемого мобильного приложения была составлена UML-диаграмма вариантов использования (рисунок 5), состоящая из актера и выполняемых им прецедентов.

Основные прецеденты вариантов использования, приведенные на диаграмме: «Авторизоваться», «Зарегистрироваться», «Выйти из аккаунта»,

Прецедент: Получить AR-сопровождение
ID: 1
<p>Краткое описание:</p> <p>Пользователь открывает экран с дополненной реальностью и видит дополнительную информацию, которая либо необходима для выполнения <u>квестового задания</u>, либо несет информационный характер.</p>
<p>Главные актеры:</p> <p>Пользователь.</p>
<p>Второстепенные актеры:</p> <p>Нет.</p>
<p>Предусловия:</p> <ol style="list-style-type: none"> 1. Пользователь авторизован в системе. 2. Пользователь начал <u>квест</u> и дошел до вопроса с AR-сопровождением.
<p>Основной поток:</p> <ol style="list-style-type: none"> 1. Пользователь нажимает на кнопку перехода к AR-сопровождению. 2. Программа открывает экран с дополненной реальностью. 3. Пользователь наводит камеру на целевой объект реального мира. 4. Программа распознает целевой объект реального мира. 5. Программа выводит около целевого объекта дополнительную информацию (текстовую и/или анимационную).
<p>Постусловия:</p> <ol style="list-style-type: none"> 1. Открыт экран с дополненной реальностью и дополнительной информацией.
<p>Альтернативные потоки:</p> <p>Нет.</p>

Рисунок 6 – Сценарий прецедента «Получить AR-сопровождение»

Описание сценария прецедента «Просмотреть статистику пользователя» приведено на рисунке 7.

Прецедент: Просмотреть статистику пользователя
ID: 2
Краткое описание: Программа выводит на экран подсказку для решения <u>квестового</u> задания и списывает за это баллы.
Главные актеры: Пользователь.
Второстепенные актеры: Нет.
Предусловия: <ol style="list-style-type: none"> 1. Пользователь авторизован в системе. 2. Пользователь начал <u>квест</u>.
Основной поток: <ol style="list-style-type: none"> 1. Пользователь нажимает на кнопку получения подсказки. 2. Система отображает подсказку. 3. Система списывает у пользователя баллы за подсказку.
Постусловия: <ol style="list-style-type: none"> 1. Пользователь получил подсказку. 2. Баллы за подсказку списаны у пользователя.
Альтернативные потоки: Нет.

Рисунок 7 – Сценарий прецедента «Получить подсказку»

В результате анализа функциональных требований были выделены варианты использования программы. Для основных вариантов использования была приведена спецификация, которая позволяет понять последовательность действий, которые выполняет пользователь при использовании функций системы.

2.1.3.2 Анализ нефункциональных требований

К разрабатываемому клиентскому мобильному приложению были выдвинуты следующие нефункциональные требования:

– поддержка работы под управлением операционной системы (ОС) Android версии 10.0 и новее;

- поддержка пакета ARCore для поддержки дополненной реальности;
- адаптивность пользовательского интерфейса;
- сопровождение переходов между экранами плавными анимациями;
- возможность масштабирования при увеличении числа пользователей в перспективе.

Целевые устройства для запуска мобильного приложения должны обладать следующими аппаратными характеристиками:

- минимум 4-ядерный процессор с частотой 1,5 ГГц (рекомендуется использовать 8-ядерные процессоры с частотой 2 ГГц и выше);
- минимум 2 Гб оперативной памяти (рекомендуется 4 Гб и выше);
- наличие не менее 350 Мб свободного места в хранилище;
- дисплей минимум с разрешением HD (рекомендуется Full HD и выше);
- наличие интернет-соединения;
- наличие камеры для поддержки дополненной реальности.

К оборудованию для развертывания серверной части были выдвинуты следующие нефункциональные требования:

- наличие установленного Docker для развёртывания приложения;
- процессор с минимальной частотой 1 ГГц;
- минимум 512 Мб оперативной памяти;
- минимальный объём дискового пространства – 6 Гб;
- наличие интернет-соединения со скоростью не менее 100 Мбит/с.

На данном этапе были разработаны функциональные и нефункциональные требования к системе, основанные на проведенном анализе предметной области.

2.1.4 Постановка цели и задач

Сформулированной в ходе анализа предметной области целью индивидуальной части данной выпускной квалификационной работы как стартапа является разработка туристического мобильного приложения для

прохождения исторических квест-экскурсий с применением технологии дополненной реальности.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- провести анализ предметной области, программных продуктов-аналогов, функциональных и нефункциональных требований;
- спроектировать архитектуру серверной части мобильного приложения и базу данных;
- спроектировать архитектуру клиентского мобильного приложения и пользовательский интерфейс;
- провести планирование разработки и оценку бюджета;
- реализовать архитектуру серверной части мобильного приложения и базу данных;
- реализовать архитектуру клиентского мобильного приложения и пользовательского интерфейса;
- выполнить тестирование и оценку качества кода разработанного программного обеспечения.

Главным назначением мобильного приложения является предоставление туристам возможности проведения досуга в геймифицированной форме с познавательным сопровождением, что должно помочь им в знакомстве с новыми городами, повышении уровня исторических знаний, планировании и организации путешествий, а также способствовать развитию отечественного туризма.

При решении поставленных задач важно учитывать привлекательность, простоту и интуитивную понятность пользовательского интерфейса, а также предусмотреть возможность масштабирования приложения для использования во многих городах.

Для повышения уровня вовлеченности пользователей в процесс прохождения квест-экскурсий необходимо применить технологию

дополненной реальности и обеспечить возможность добавления AR-сцен в задания и исторические справки на ключевых точках туристических маршрутов.

2.2 Проектирование программных средств для проведения квестов с применением технологии дополненной реальности

2.2.1 Проектирование архитектуры программных средств

Анализ предметной области и бизнес-требований позволил установить, что функционирование мобильного приложения для проведения исторических квест-экскурсий должно осуществляться в рамках программного комплекса, обеспечивающего возможность не только прохождения квестов, но и их формирования, а также поддерживающего прочие административные функции. Из этого следует, что программный комплекс должен иметь как минимум два программных компонента, оперирующих общими данными:

- мобильное приложение для проведения квестов с применением технологии дополненной реальности;
- приложение администратора.

Исходя из описанной особенности на этапе проектирования всего программного комплекса было принято решение применить трехуровневую клиент-серверную архитектуру. Данная архитектурная концепция характеризуется выделением сервера базы данных, сервера приложений и взаимодействующих с ним клиентских приложений [19], что позволяет ей обладать хорошей масштабируемостью, гибкостью, конфигурируемостью, а также иметь высокую надежность, безопасность и низкие требования к техническим характеристикам клиентских устройств [20].

Обобщенное распределение компонентов программного комплекса, организованного по принципу клиент-серверной архитектуры описано с помощью UML-диаграммы развертывания, приведенной на рисунке 8.

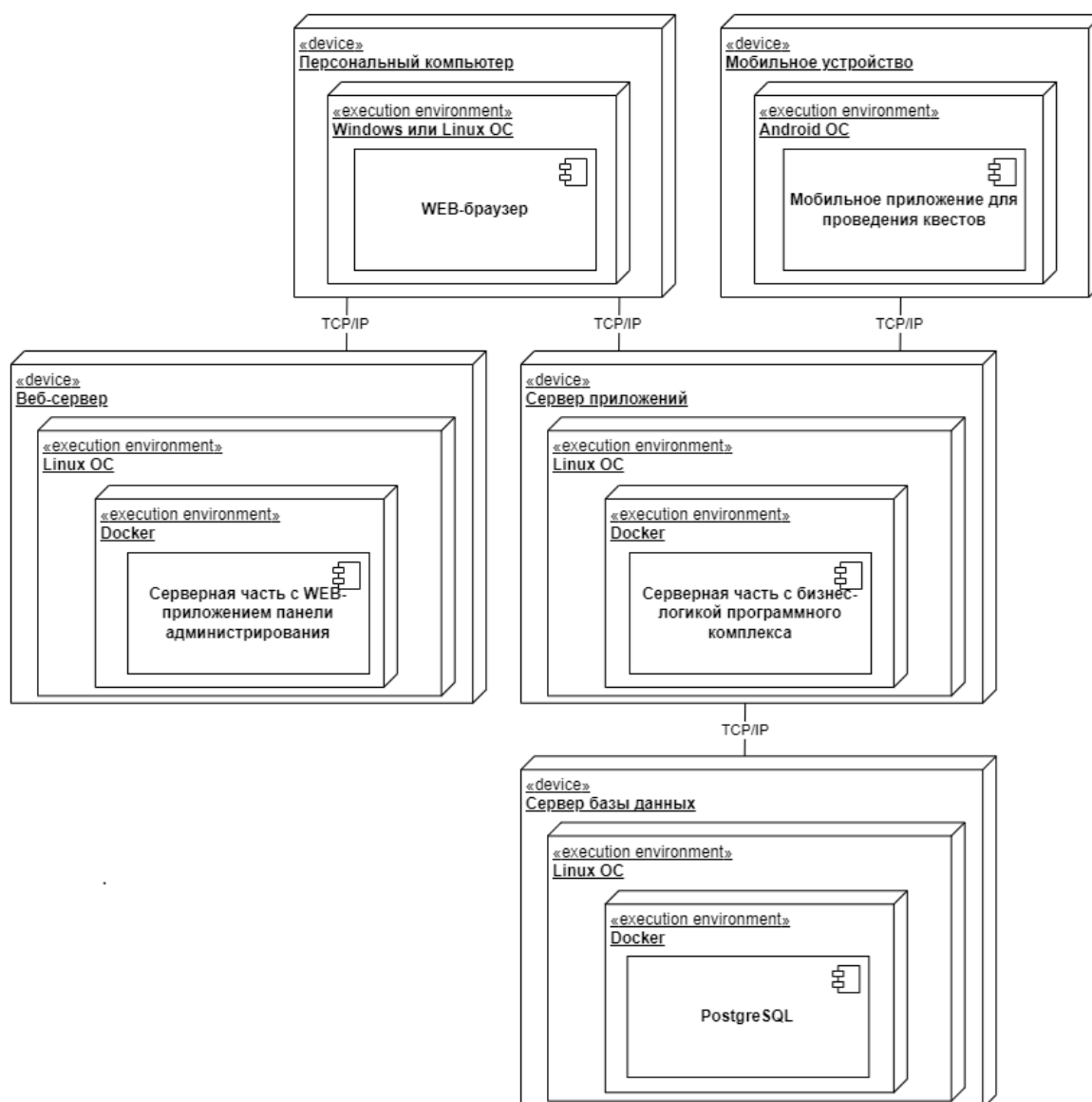


Рисунок 8 – Диаграмма развертывания программного комплекса для проведения квестов с применением технологии дополненной реальности

В рамках данной выпускной квалификационной работы производилась разработка лишь клиентского мобильного приложения и части серверного программного обеспечения (ПО), необходимой для его функционирования.

2.2.1.1 Проектирование архитектуры серверной части

2.2.1.1.1 Выбор архитектуры серверной части

Первым этапом проектирования серверной части стал выбор архитектурного подхода, на котором должно базироваться приложение. При

выборе архитектуры необходимо учесть набор факторов, обусловленных особенностями предметной области и спецификой проекта. Разработка программного комплекса с целью создания стартап-проекта предполагает возможность быстрого роста числа пользователей, что влечет за собой необходимость быстрого и легкого масштабирования. Кроме того, при развитии проекта планируется постоянное внедрение большого количества нового функционала. Также стоит учитывать устойчивость к сбоям и сложность инфраструктурного окружения программного обеспечения. Наконец, важными факторами будут являться скорость и стоимость разработки.

Правильный выбор архитектуры серверной части приложения позволит найти оптимальное соотношение между качеством программного кода и затраченными на разработку ресурсами. Для выбора архитектуры воспользуемся методами теории принятия решений. Для этого введем кодовые обозначения сравниваемых архитектур (таблица 7) и опишем критерии для их сравнения (таблица 8).

Таблица 7 – Кодовые обозначения архитектур

Архитектура	Кодовое обозначение
Монолитная архитектура	B1
Архитектура на основе сервисов	B2
Микросервисная архитектура (MSA)	B3

Таблица 8 – Описание критериев сравнения архитектур

Критерий	Код критерия	Описание
Масштабируемость	K1	Работоспособность системы при росте числа пользователей или ужесточении уровня технических требований [21]. В том числе при возможности наращивания дополнительных ресурсов без структурных изменений главных компонентов серверной части программного комплекса
Поддерживаемость	K2	Потенциал простоты последующей модификации и внедрения нового функционала
Отказоустойчивость	K3	Способность сохранять надежность при отказе отдельных компонентов или сбоях в подсистемах [22]
Производительность	K4	Скорость и эффективность выполнения запросов

Продолжение таблицы 8

Критерий	Код критерия	Описание
Развертываемость	K5	Простота и скорость выполнения действий, которые делают программную систему готовой к использованию
Тестируемость	K6	Мера того, насколько просто создать критерий проверки для системы и ее компонентов и выполнить эти тесты [23]
Общие затраты	K7	Уровень временных и экономических затрат, необходимых для разработки серверной части программного комплекса

Для выполнения сравнения качественных критериев необходимо перевести их значения в количественные показатели. Для этого введем шкалы количественных показателей для каждого из критериев (таблицы 9 – 15).

Таблица 9 – Вербально-числовая шкала для критерия K1

Критерий	Числовой эквивалент критерия
Очень высокая масштабируемость. Отсутствует зависимость между компонентами при масштабировании	1,0
Высокая масштабируемость. Присутствует зависимость между некоторыми компонентами при масштабировании	0,9
Низкая масштабируемость. Присутствует зависимость между многими компонентами при масштабировании	0,8
Очень низкая масштабируемость. Необходимо масштабировать все компоненты вместе	0,7

Таблица 10 – Вербально-числовая шкала для критерия K2

Критерий	Числовой эквивалент критерия
Очень быстрое внедрение нового и изменение существующего функционала	1,0
Быстрое внедрение нового и изменение существующего функционала	0,9
Требуется время внедрение нового и изменение существующего функционала	0,8
Очень долгое внедрение нового и изменение существующего функционала	0,7

Таблица 11 – Вербально-числовая шкала для критерия K3

Критерий	Числовой эквивалент критерия
Возникновение сбоя локально и не влияет на работоспособность системы	1,0
Возникновение сбоя локально и влияет лишь на работоспособность конкретного компонента системы	0,9

Продолжение таблицы 11

Критерий	Числовой эквивалент критерия
Возникновение сбоя локально и влияет на работоспособность компонента системы и зависящих от него компонентов. Остальная часть системы остается работоспособной	0,8
Возникновение сбоя приводит к общесистемному сбою	0,7

Таблица 12 – Вербально-числовая шкала для критерия К4

Критерий	Числовой эквивалент критерия
Очень высокая производительность	1,0
Высокая производительность	0,9
Средняя производительность	0,8
Низкая производительность	0,7

Таблица 13 – Вербально-числовая шкала для критерия К5

Критерий	Числовой эквивалент критерия
Процедура развертывания обновленных компонентов не затрагивает работоспособность системы	1,0
Процедура развертывания обновленных компонентов выполняется при остановке обновляемых компонентов	0,9
Процедура развертывания обновленных компонентов выполняется при остановке обновляемых компонентов и компонентов, зависящих от них	0,8
Процедура развертывания обновленных компонентов выполняется при полной остановке системы	0,7

Таблица 14 – Вербально-числовая шкала для критерия К6

Критерий	Числовой эквивалент критерия
Максимально полная тестируемость	1,0
Почти полная тестируемость	0,9
Частичная тестируемость	0,8
Плохо поддается тестированию	0,7

Таблица 15 – Вербально-числовая шкала для критерия К7

Критерий	Числовой эквивалент критерия
Очень низкая стоимость и длительность разработки	1,0
Низкая стоимость и длительность разработки	0,9
Высокая стоимость и длительность разработки	0,8
Очень высокая стоимость и длительность разработки	0,7

Количественные показатели критериев для каждого из вариантов архитектуры на основе введенных вербально-числовых шкал перевода приведены в таблицу 16. Оценка критериев производилась на основе

субъективного опыта и знаний, а также на основе оценок, данных архитектурам в книге «Фундаментальный подход к программной архитектуре» [21].

Таблица 16 – Количественные показатели критериев выбора архитектуры

Код критерия	Критерий	Значение критериев вариантов архитектуры		
		B1	B2	B3
K1	Масштабируемость	0,7	0,85	1,0
K2	Поддерживаемость	0,7	0,85	0,95
K3	Отказоустойчивость	0,7	0,8	0,85
K4	Производительность	0,8	0,9	0,75
K5	Развертываемость	0,7	0,8	0,9
K6	Тестируемость	0,9	0,9	0,8
K7	Общие затраты	0,95	0,9	0,7

При определении лучшего варианта архитектуры был использован метод ранжирования Борда [24]. Для этого составлена таблица рангов (таблица 17), содержащая ранги по всем критериям, зависящие от оценок всех вариантов по тому или иному критерию. Был проведен подсчет суммы рангов по формуле 1:

$$R_j = \sum_{i=1}^n r_{ij},$$

где R_j – сумма рангов j -того варианта;

r_{ij} – ранг i -того критерия j -того варианта;

n – количество критериев.

Таблица 17 – Ранжирование вариантов

Код критерия	Ранг критериев		
	B1	B2	B3
K1	3	2	1
K2	3	2	1
K3	3	2	1
K4	2	1	3
K5	3	2	1
K6	1,5	1,5	3
K7	1	2	3
Итого	16,5	12,5	13

Для определения лучшего варианта была применена формула 2:

$$R_l = \min[R_1, R_m],$$

где R_l – лучший вариант;

R_m – сумма рангов m -ого варианта;

m – Количество вариантов.

Результатом применения ранжирования вариантов по методу Борда стало расположение приоритетности выбора архитектур в следующем порядке: $B2 > B3 > B1$. Лучшей архитектурой для выбранного набора критериев является архитектура на основе сервисов под кодовым обозначением B2. Именно эта архитектура позволит соблюсти оптимальное соотношение между качеством программного кода и затраченными на разработку ресурсами с учетом особенностей проекта. Стоит отметить, что при изменении приоритетов в формировании набора критериев выбора, их анализ может привести к другому наиболее подходящему под особенности типу архитектуры. Кроме того, с учетом погрешности, разница между рангами вариантов B2 и B3 позволяет сделать выбор и в пользу микросервисной архитектуры. Однако с учетом ограниченности бюджета проекта и сроков разработки был выбран вариант архитектуры на основе сервисов.

2.2.1.1.2 Разбиение серверной части на сервисы

Типовая топология архитектуры на основе сервисов – распределенная макроуровневая структура, состоящая из отдельно развернутого пользовательского интерфейса, отдельно развернутых удаленных крупномодульных сервисов и монолитной базы данных (рисунок 9).

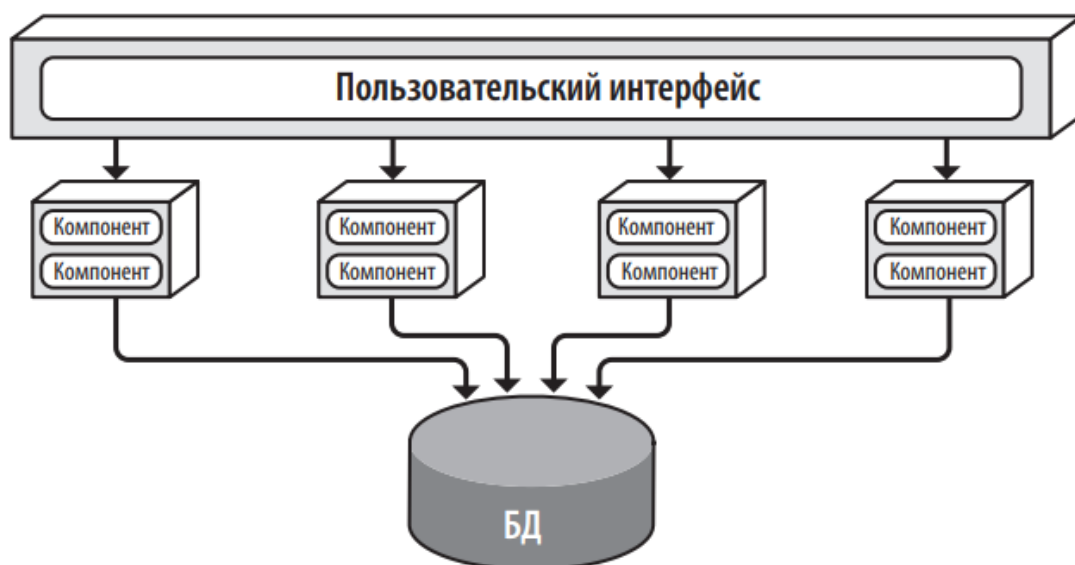


Рисунок 9 – Типовая топология архитектуры на основе сервисов

Сервисы в этом архитектурном стиле представлены, как правило, независимыми и отдельно развертываемыми крупномодульными «порциями приложения», которые обычно называют сервисами предметной области (domain services). Поскольку сервисы обычно совместно используют единую монолитную базу данных, их число в контексте приложения чаще всего варьируется от 4 до 12 единиц, то есть в среднем используется около семи сервисов [21].

При проектировании серверной части для разрабатываемого мобильного приложения для проведения квестов с применением технологии дополненной реальности было принято решение выделить четыре сервиса, наименование и назначение которых приведено в таблице 18.

Таблица 18 – Сервисы серверной части

Наименование сервиса	Назначение сервиса
migrator	Инициализация базы данных и выполнение миграций
questie_backend	Обработка запросов, связанных с квестами и квестовыми заданиями
questie_backend_authorization	Обработка запросов на генерацию JSON Web Token (JWT)
questie_backend_users	Обработка запросов, связанных с пользователями

Наглядное представление распределения компонентов программного комплекса по различным узлам, отображающее связь сервисов с базой данных и клиентским мобильным приложением приведено в виде UML-диаграммы развертывания на рисунке 10. Взаимодействие между физическими узлами должно осуществляться с помощью TCP/IP.

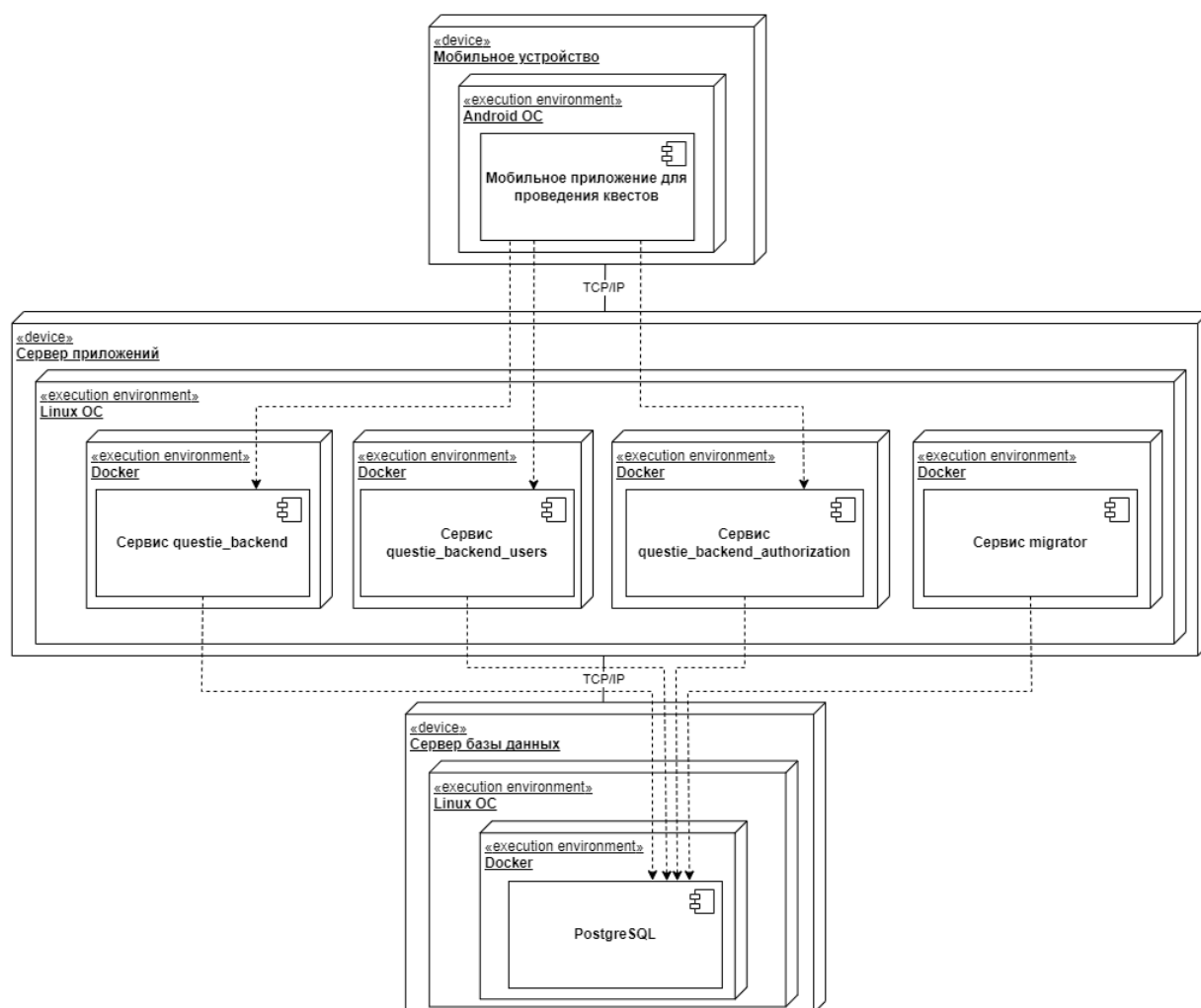


Рисунок 10 – Диаграмма развертывания части программного комплекса, необходимой для функционирования мобильного приложения

Из диаграммы видно, что все сервисы должны взаимодействовать с базой данных, однако сервис migrator не должен принимать запросы от клиентского мобильного приложения. Это обусловлено его назначением – инициализацией базы данных и выполнением миграций. При развертывании программного комплекса сервис должен активироваться после развертывания базы данных (БД) и завершать свою работу до запуска остальных сервисов.

На приведенной диаграмме все четыре сервиса расположены на одной физической машине, однако при необходимости увеличения вычислительных мощностей они могут быть разнесены по разным физическим машинам, что не вызовет затруднений благодаря использованию виртуализированной среды

выполнения контейнеров Docker. При необходимости возможен и запуск нескольких экземпляров сервисов, обрабатывающих запросы. Однако в таком случае потребуется внедрение в серверную часть балансировщика нагрузки.

Взаимодействие между клиентским мобильным приложением и сервисами должно происходить на основе стека TCP/IP с применением протокола HTTPS на прикладном уровне и применением подхода REST API для описания интерфейсов взаимодействия.

2.2.1.1.3 Проектирование последовательности выполнения запросов

Обработка поступающих к сервису запросов в ASP.NET Core происходит по принципу конвейера, состоящего из набора последовательных компонентов, называемых промежуточным или связующим программным обеспечением (middleware) [25].

Для наглядного представления последовательности обработки запросов была использована UML-диаграмма последовательности (рисунок 11), описывающая общий вид обработки запросов к сервису `questie_backend` в режиме Release.

Обработка запросов к сервису `questie_backend` начинается с работы промежуточного ПО, отвечающего за настройку CORS для обеспечения поддержки кроссдоменных запросов. После чего в случае запуска сервиса в режиме разработки вызывается настройка инструмента Swagger и генерация графического интерфейса с документацией API, поддерживающей спецификацию OpenAPI. Далее управление передается в промежуточное ПО `StaticFiles`, которое предоставляет поддержку обработки статических файлов. Затем происходит настройка поддержки сессий, аутентификации, авторизации, вызов промежуточного ПО для обработки JWT и, наконец, сопоставление URL-адресов входящих запросов и действий по их обработке. После чего управление передается в соответствующий метод контроллера, где происходит обработка запроса, не детализированная на UML-диаграмме (вызов сервисных методов,

выполняющих основную логику и при необходимости обращающихся к методам работы с данными для получения информации из базы данных).

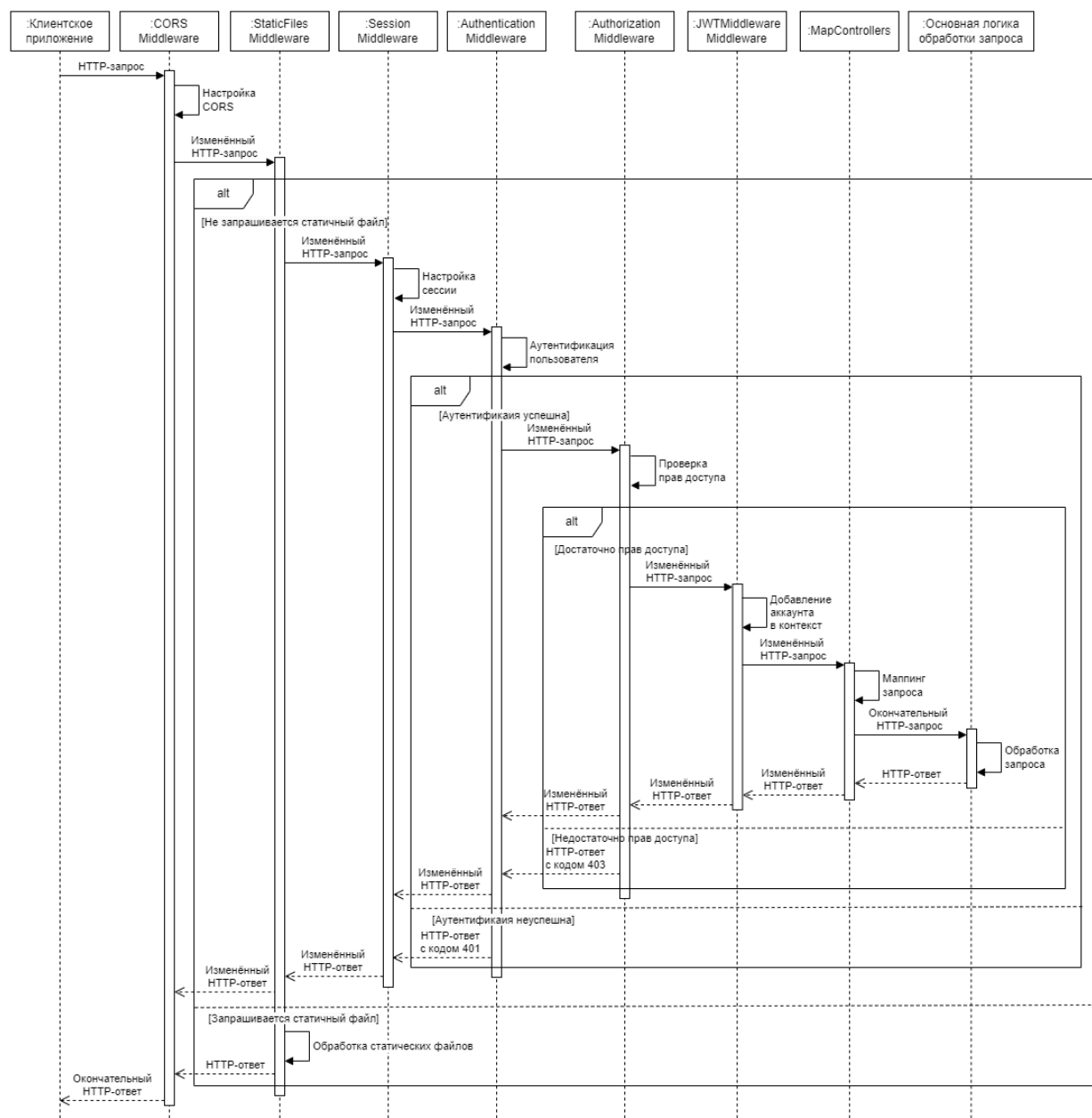


Рисунок 11 – Диаграмма последовательности запросов к сервису `questie_backend`

Обработка запроса к сервису `questie_backend_users` должна происходить аналогично, за исключением отсутствия промежуточного ПО `StaticFiles`.

Конвейер обработки запросов к сервису `questie_backend_authorization` также лишен промежуточного ПО `Authorization` и `JWTMiddleware`.

2.2.1.1.4 Проектирование основной логики обработки запросов

Поскольку сервисы предметной области в архитектуре на основе сервисов обычно крупномодульные, каждый из них разрабатывается с использованием многоуровневого стиля архитектуры, состоящего из уровня внешнего API-интерфейса, бизнес-логики и уровня сохранения данных [21].

При проектировании сервисов серверной части программного комплекса для проведения квестов было принято решение также применить многоуровневый стиль архитектуры с техническим разбиением на слои.

Наглядное представление такого разбиения приведем в формате UML-диаграммы классов (рисунок 12) на примере классов, выполняющих обработку запросов, касающихся пользователей.

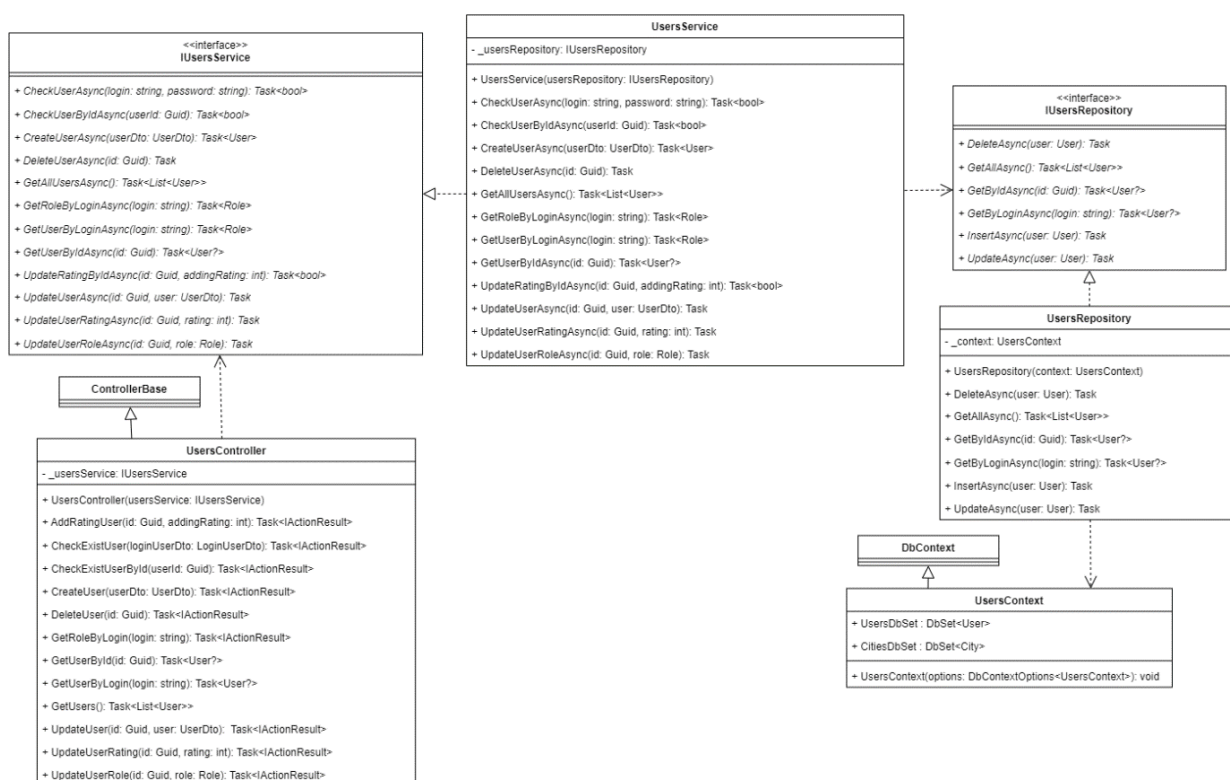


Рисунок 12 – Диаграмма классов, выполняющих обработку запросов, касающихся пользователей

Для проектирования обработки запросов, касающихся пользователей, был взят за основу шаблон проектирования «Стратегия», позволяющий определить

семейство алгоритмов, инкапсулировать каждый из них и сделать их взаимозаменяемыми [26].

Как видно из диаграммы, «Стратегия» применяется дважды. В первом случае она обеспечивает возможность замены реализации сервисных классов и включает в себя следующие элементы:

- интерфейс `IUserService`, общий для всех реализаций сервисного класса;
- класс `UserService`, реализующий интерфейс `IUserService` (подобных классов-реализаций может быть несколько);
- класс `UserController`, хранящий ссылку на объект типа `IUserService` и связанный с интерфейсом `IUserService` отношением агрегации.

Во втором случае, представленном классами `UserService`, `UsersRepository` и интерфейсом `IUsersRepository`, применение данного шаблона проектирования позволяет использовать различные реализации репозитория, что может быть актуально, например, в случае изменения ORM для взаимодействия с базой данных.

Класс `UsersContext` предназначен для определения контекста доступа к базе данных и наследуется от базового класса контекста `DbContext`.

Класс `ControllerBase` является базовым для всех контроллеров и реализован в фреймворке ASP.NET Core.

Связи с классами, описывающими сущности предметной области и объекты передачи данных (Data Transfer Object или DTO), не приведены на диаграмме для большей компактности.

Описание классов `DbContext` и `ControllerBase`, имеющих стороннюю реализацию, не приведено на диаграмме классов по той же причине.

Во избежание зависимости от низкоуровневых деталей и связи с другими конкретными классами был применен шаблон проектирования Dependency Injection или «Внедрение зависимостей» [27].

Внедрение зависимостей применяет принцип IoC (Inversion of Control или инверсия управления), чтобы гарантировать, что класс абсолютно никаким

образом не участвует в создании экземпляра зависимого класса, и тем более не управляет его жизненным циклом (lifetime), то есть временем существования объектов [28].

В описываемом примере данный шаблон проектирования применяется для получения доступа к экземплярам классов UsersService, UsersRepository и QuestsContext, имеющих длительность жизненного цикла, ограниченную одним HTTP-запросом.

2.2.1.2 Проектирование архитектуры клиентского мобильного приложения

На начальном этапе проектирования мобильного приложения был детально рассмотрен основной сценарий использования – взаимодействие с квестами авторизованного пользователя. При данной рассмотрении не учитывались внутренние особенности организации процессов серверной части и сервер воспринимался как модель «черного ящика» [29], всегда возвращающего корректный ответ. Для рассмотрения алгоритма обработки взаимодействия с квестами авторизованного пользователя была построена UML-диаграмма деятельности (рисунок 13).

После выбора квеста для прохождения из списка квестов открывается более полная информация о нем, после чего отображается карта маршрута и его описание. Затем отображается задание с возможностью выполнения нескольких действий: просмотр карты маршрута, подсказки, сопровождающей сцены дополненной реальности, а также выбор ответа на квестовое задание. После чего происходит проверка правильности ответа на стороне сервера, отображение результата и переход к исторической справке. Далее, в случае если задание не является последним, открывается следующее задание, иначе подводятся результаты прохождения квеста.

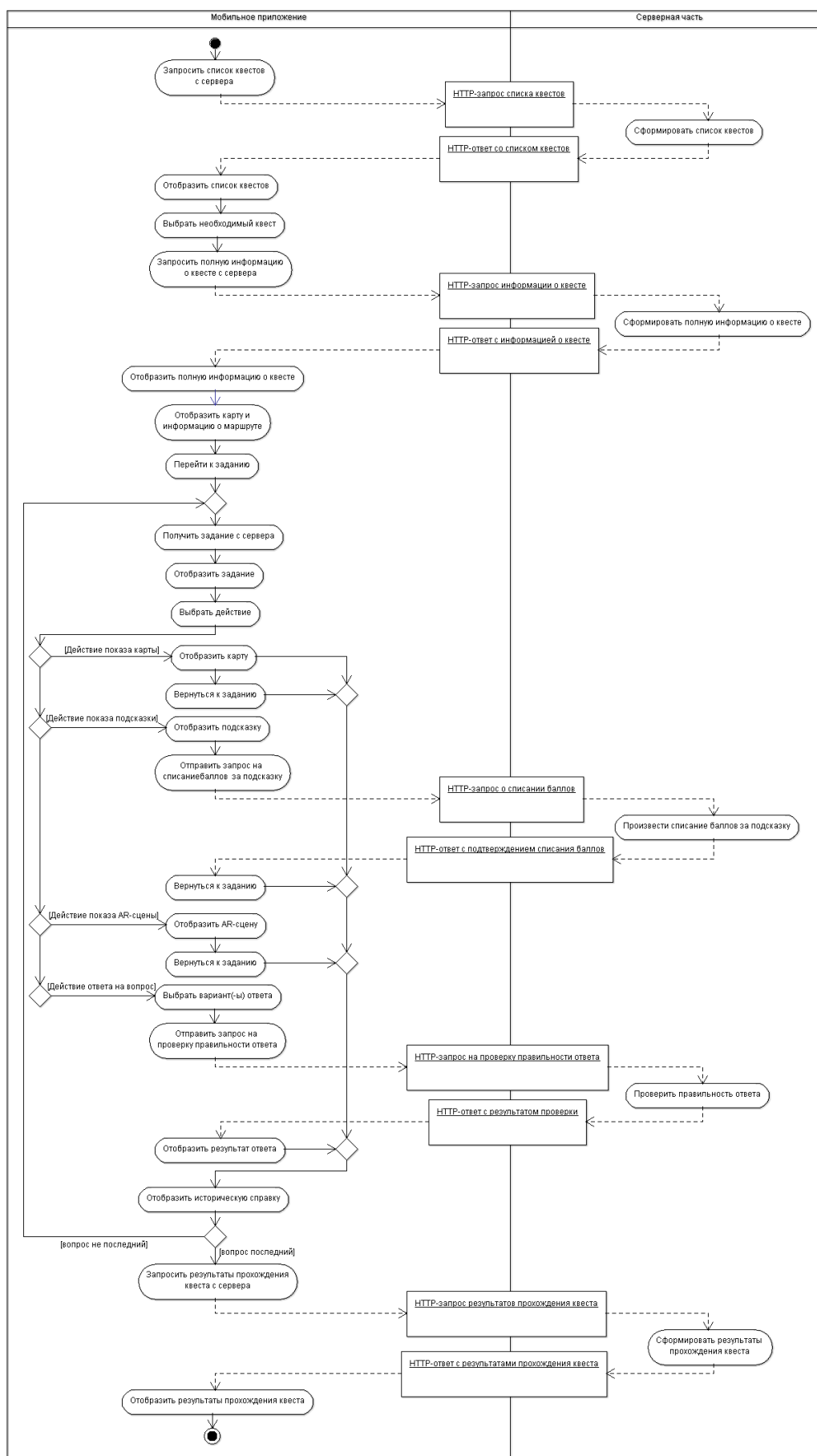


Рисунок 13 – Диаграмма деятельности взаимодействия с квестами

На следующем этапе проектирования мобильного приложения производился выбор типа архитектуры. Были рассмотрены наиболее популярные в мобильной разработке шаблоны проектирования MVx-семейства: MVC, MVP и MVVM. Агрегация изученной информации позволила выделить ключевые характеристики шаблонов, представленные в таблице 19 [30, 31].

Таблица 19 – Сравнение шаблонов проектирования MVx-семейства

Признак сравнения	MVC	MVP	MVVM
Тип отношения	Связь «многие к одному» между контроллером и представлением	Связь «один к одному» между презентером и представлением	Связь «многие к одному» между ViewModel и представлением
Ссылки	Представление не имеет ссылки на контроллер	Представление имеет ссылку на презентер	Представление имеет ссылку на View-Model
Связь представления и модели	Тесно связаны (представление зависит от модели)	Модель и представление не связаны напрямую. Каналом связи выступает презентер	Модель и представление не связаны напрямую. Шаблон основан на событиях и привязке данных
Модульное тестирование	Ограниченная поддержка	Полная поддержка. Трудности из-за тесной связи представления и презентера	Полная поддержка

Продолжение таблицы 19

Признак сравнения	MVC	MVP	MVVM
Размер проектов	Лучше подходит для небольших приложений	Подходит как для небольших, так и больших приложений	Лучше подходит для больших приложений

От шаблона проектирования MVC было решено отказаться из-за трудностей с поддержкой и расширением функционала, масштабированием приложения и проблемами с его модульным тестированием.

В основе паттернов MVP и MVVM лежит идея о дополнительном разделении слоев и вынесении бизнес-логики в отдельный для нее слой. Изначально данный подход презентовался известным американским программистом Робертом Мартином [32] для общего назначения в сфере разработки программного обеспечения, но нашел широкий отклик в более узком кругу андроид-разработчиков [33]. Использование приведенных паттернов дает хорошую распределенность между компонентами, отличную тестируемость и простоту использования [34].

Сложность интерфейса проекта не на столько высока, чтобы использовать шаблон проектирования MVVM, а связывание данных потребовало бы дополнительного кода с минимальным выигрышем по качеству разработки [35]. В свою очередь, применение MVP обеспечивает достаточный уровень тестируемости, простоты поддержки и масштабируемости приложения при оптимальном уровне трудозатрат на разработку, что позволяет сделать выбор в пользу применения данного шаблона проектирования.

На рисунке 14 представлена UML-диаграмма классов, необходимых для отображения квестового задания и взаимодействия с ним, основанная на применении шаблона проектирования MVP. Наборы классов, необходимых для

взаимодействия с другими экранами мобильного приложения имеют аналогичную структуру.

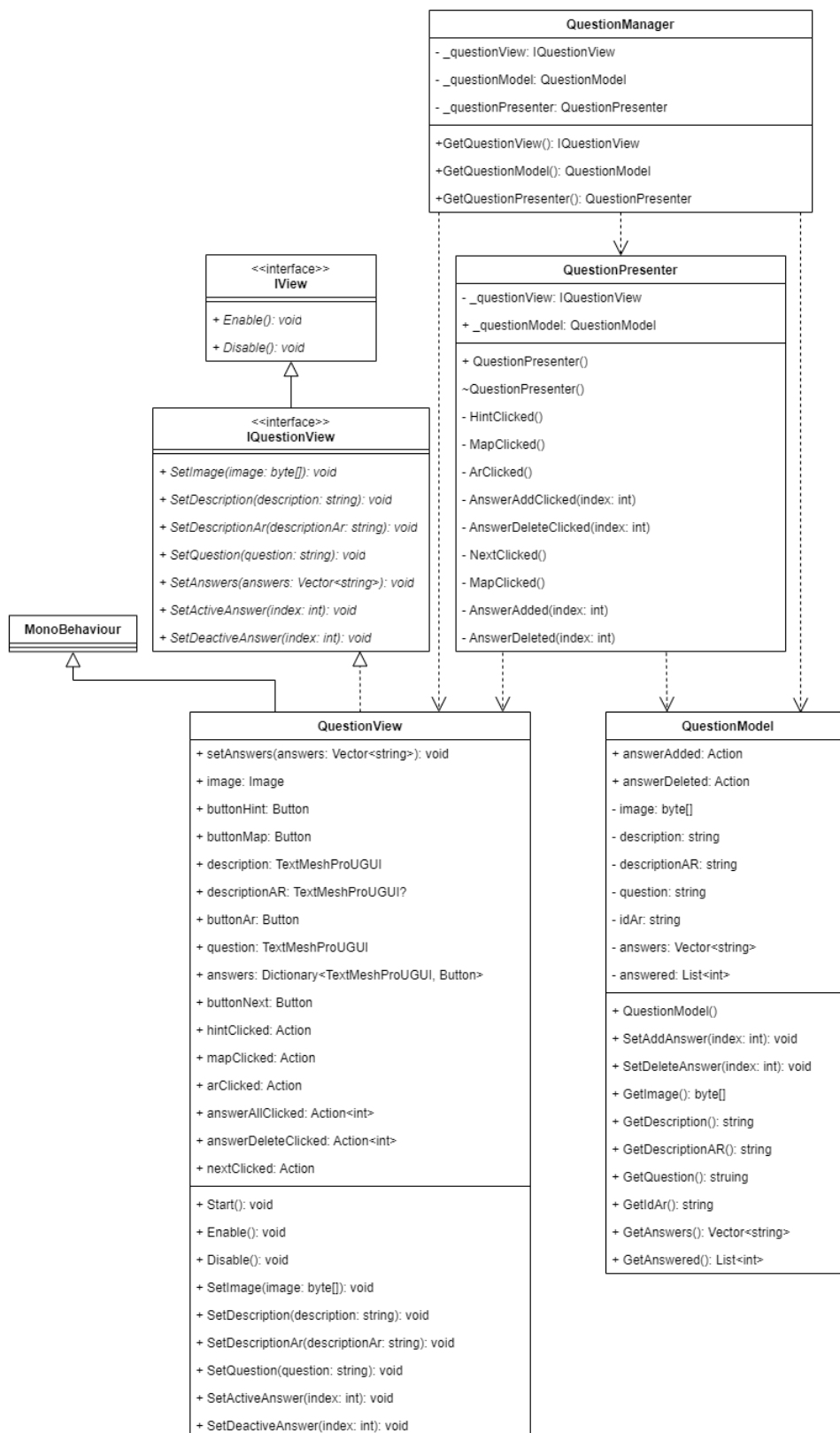


Рисунок 14 – Диаграмма классов, необходимых для взаимодействия с экраном квестового задания

Назначение классов и интерфейсов, описанных в диаграмме, приведено в таблице 20.

Таблица 20 – Назначение классов и интерфейсов, необходимых для взаимодействия с экраном квестового задания

Название	Назначение
IView	Базовый интерфейс представления. Применяется во всех экранах
IQuestionView	Интерфейс представления вопросов
QuestionView	Представление вопроса. Отвечает за взаимодействие с пользовательским интерфейсом
QuestionPresenter	Презентер вопроса. Отвечает за управление взаимодействием между данными и пользовательским интерфейсом
QuestionModel	Модель вопроса. Отвечает за взаимодействие с данными UI
QuestionManager	Класс, обеспечивающий доступ к объектам QuestionView, QuestionPresenter, QuestionModel

Класс MonoBehaviour, от которого наследуется QuestionView – это базовый класс, который должен быть унаследован каждым Unity-скриптом. Скрипт Unity, производный от MonoBehaviour, выполняет набор предопределенных функций (Awake(), Start() или OnEnable()), которые выполняются для инициализации при возникновении событий [36].

Более наглядно взаимодействие классов QuestionView, QuestionPresenter, QuestionModel представлено на UML-диаграмме коммуникации на рисунке 15.

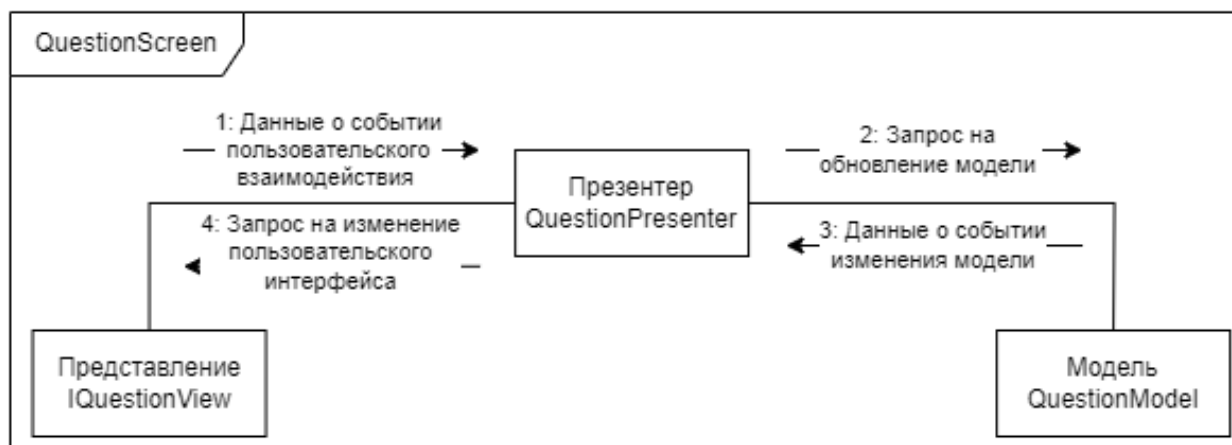


Рисунок 15 – Диаграмма коммуникации, описывающая взаимодействие с экраном квестового задания

Из диаграммы видно, что инициатором изменений выступает пользователь, совершающий некоторое действие в представлении (например,

нажимающий на кнопку). После чего информация о совершенном действии поступает в презентер, делающий запрос на изменение данных в модели. Изменение модели активирует событие, запускающее через презентер ответное изменение пользовательского интерфейса в представлении. Подобным образом устроено взаимодействие со всеми экранами мобильного приложения.

2.2.2 Проектирование логической модели базы данных

При работе с фреймворками объектно-реляционного отображения (ORM) имеется три подхода к моделированию структуры базы данных: Model-First, Database-First и Code-First [37]. Как правило, при необходимости создания базы данных с нуля используются Model-First или Code-First подход. Первый дает больше наглядности, второй позволяет экономить время разработки. При проектировании структуры базы данных серверной части для мобильного приложения для проведения квестов был выбран Code-First подход.

В Code-First работа начинается с определения модели предметной области, с использованием классов POCO (Plain Old CLR Object), не зависящих от Entity Framework. Code-First позволяет получить основную информацию о модели исключительно из описывающих ее классов. Однако существует возможность предоставить дополнительную конфигурацию для более полного описания модели или переопределить то, что изначально было создано на основе кода [38].

Таким образом, подход Code-First позволяет определять модель данных с помощью классов и свойств, а затем создавать базу данных на основе этой модели. Благодаря этому подходу разработчик может иметь большую гибкость при работе с моделью данных и управлении ее изменениями [39].

Поскольку серверная часть программного комплекса предназначена не только для функционирования мобильного приложения, разработке которого посвящена данная работа, при проектировании модели базы данных была учтена необходимость создания атрибутов сущностей, предназначенных для

функционирования web-клиента администрирования программного комплекса.
Полученная логическая модель базы данных приведена на рисунке 16.

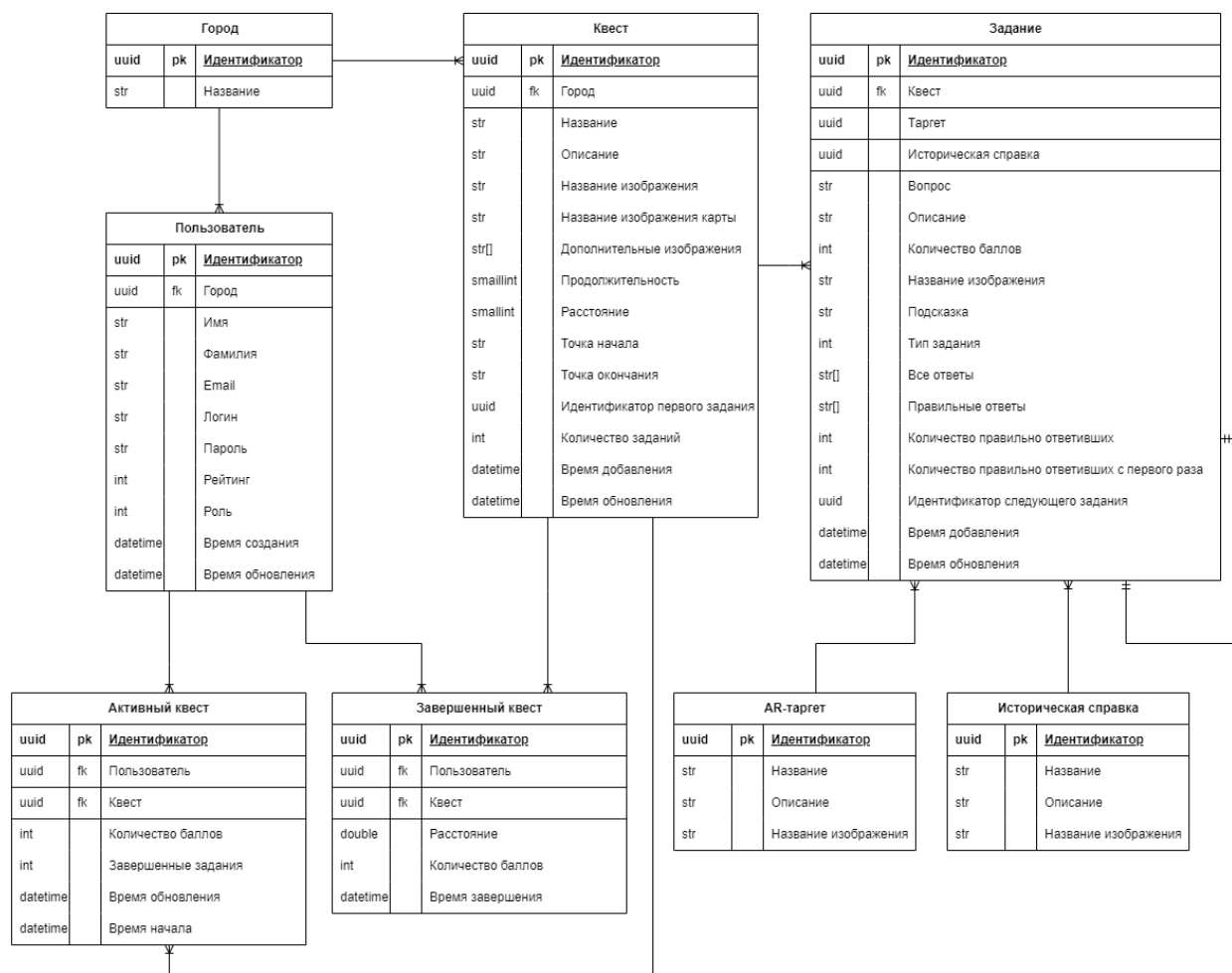


Рисунок 16 – Логическая модель базы данных

Назначение таблиц разработанной логической модели данных приведено в таблице 21.

Таблица 21 – Таблицы логического уровня модели данных

Наименование сущности	Наименование таблицы	Назначение сущности
Активные квесты	ActiveQuestsDbSet	Квесты, запущенный для прохождения конкретными пользователями, но еще не завершенные
Города	CitiesDbSet	Список доступных городов
Завершенные квесты	CompletedQuestsDbSet	Информация о пройденных пользователями квестах
Исторические справки	NotesDbSet	Информация об исторических справках
Таргеты	TargetsDbSet	Информация об AR-таргетах
Пользователи	UsersDbSet	Информация о пользователе

Продолжение таблицы 21

Наименование сущности	Наименование таблицы	Назначение сущности
Квесты	QuestsDbSet	Информация о доступных квестах
Квестовые задания	QuestionsDbSet	Информация о квестовых заданиях и их принадлежности к тем или иным квестам

Таким образом, была спроектирована логическая модель базы данных, описывающая все необходимые для функционирования приложения сущности предметной области.

2.2.3 Проектирование пользовательского интерфейса клиентского мобильного приложения

Проектирование пользовательского интерфейса разрабатываемого мобильного приложения осуществлялось в графическом редакторе для прототипирования и разработки интерфейсов Figma.

На первом этапе проектирования был разработан набор компонентов, используемых для создания прототипов экранов (рисунок 17).

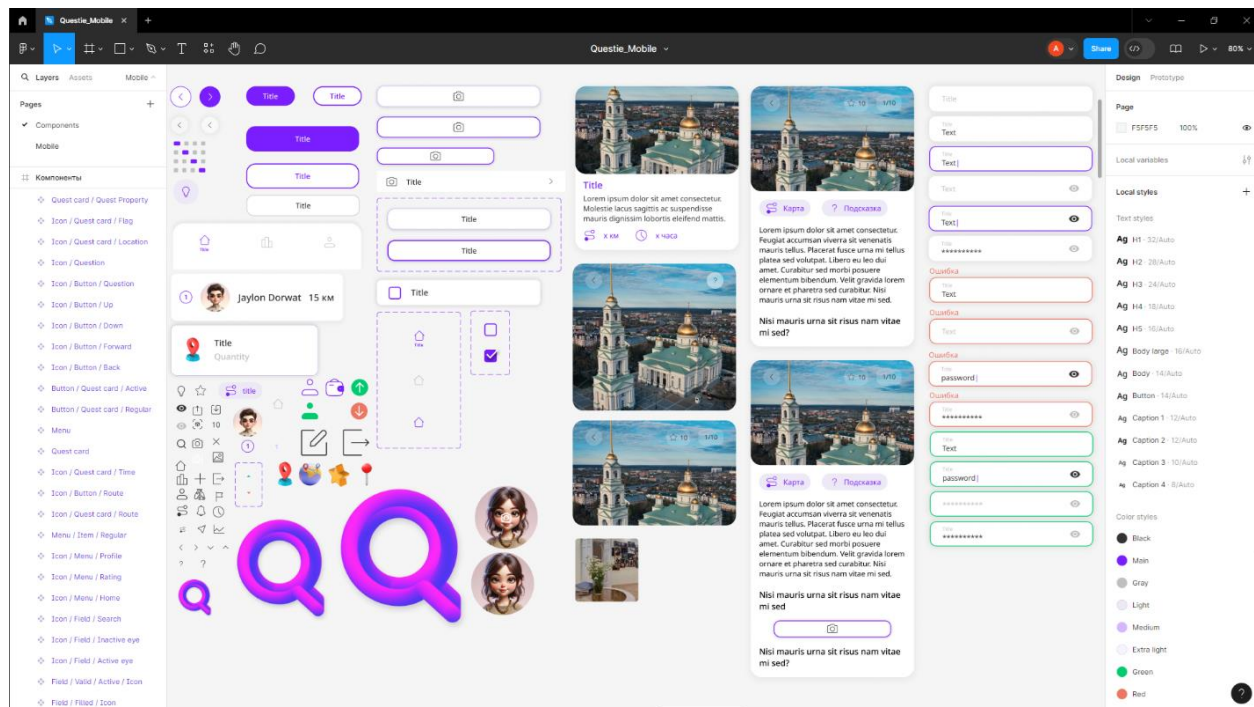


Рисунок 17 – Компоненты для проектирования пользовательского интерфейса

На основе разработанных компонентов были спроектированы макеты экранов мобильного приложения, представленные на рисунке 18.

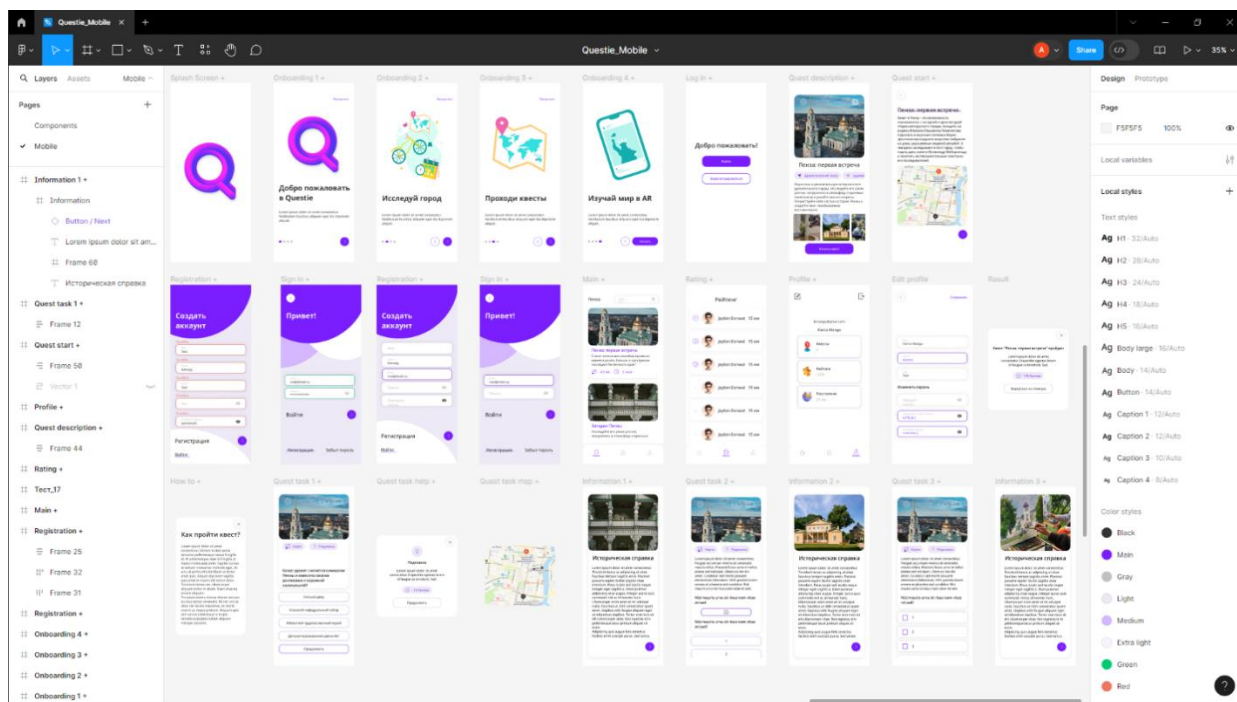


Рисунок 18 – Макеты пользовательского интерфейса мобильного приложения

Рассмотрим более детально результат проектирования экрана запуска квест-экскурсии, представленного на рисунке 19.



Рисунок 19 – Макет пользовательского интерфейса экраня запуска квест-экскурсии

Элементы макета пользовательского интерфейса экрана запуска квест-экскурсий, представленные на рисунке 19:

- 1 – кнопка возврата к списку квест-экскурсий;
- 2 – кнопка вызова окна с инструкцией о прохождении квест-экскурсий;
- 3 – главная фотография квест-экскурсии;
- 4 – надпись, название квест-экскурсии;
- 5 – карусель из тегов, соответствующих квест-экскурсии;
- 6 – надпись, описание квест-экскурсии;
- 7 – карусель из фотографий с мест квест-экскурсии;
- 8 – кнопка запуска квест-экскурсии.

Таким образом, был спроектирован пользовательский интерфейс для разрабатываемого мобильного приложения. При проектировании интерфейса были учтены требования к адаптивности, понятности и принципа дизайна.

2.2.4 Планирование разработки и оценка бюджета

2.2.5.1 Организация производства

Для успешной реализации мобильного приложения для проведения квестов необходимо применить компетенции следующих специалистов: менеджер проекта, аналитик, проектировщик, backend-разработчик, разработчик мобильных приложений, тестировщик.

Менеджер проекта должен заниматься планированием времени и денежных ресурсов, выделением этапов реализации, формированием документации и презентаций проекта, организацией и проверкой качества выполнения поставленных задач, и координацией сотрудников проекта [40].

В задачи аналитика входит анализ бизнес-процессов предметной области, выявление и формирование детальных требований к разрабатываемому продукту, макетов интерфейса, ведение сопроводительной документации по описанию технического задания, а также сущностей, взаимосвязей и процессов предметной области.

Проектировщик должен отвечать за создание архитектуры и дизайна программного обеспечения, выбор технологий и инструментов, необходимых для разработки проекта, а также за координацию работы команды разработчиков [41]. В его задачи входит определение структур данных, потоков данных, возможная оптимизация программы и рефакторинг кода, а также сопровождение программного продукта.

Обязанностями разработчиков является непосредственная реализация в программном коде решений, описанных проектировщиками для выполнения задач, поставленных аналитиками. Как правило, именно разработчики выполняет наибольший объем работ по проекту из всей команды.

Тестировщики занимаются выявлением ошибок, сбоев, некорректной работы и уязвимостей программного кода, а также формированием рекомендаций по их устранению и отчетов.

2.2.5.2 Календарный план разработки

При планировании разработки проекта и оценке бюджета было применено программное обеспечение для управления проектами Microsoft Project. Выполнение проекта было разбито на 36 задач, среди которых ключевыми являются:

- разработка бизнес-плана;
- анализ разработки программных средств;
- проектирование программных средств;
- реализация программных средств;
- тестирование программных средств;
- составление и оформление документации.

Для каждой задачи была определена ее продолжительность, даты выполнения и необходимые ресурсы. Полученное разбиение проекта на задачи приведено на рисунке 20.

	Название задачи	Длительность	Начало	Окончание	Пре	Названия ресурсов
1	Начало реализации проекта	117 дней	Пт 01.12.23	Пн 13.05.24		
2	Закупка оборудования и ПО	1 день	Пт 01.12.23	Пт 01.12.23		Компьютер[p.50 000,00];Принтер[p.8 100,00];MS Windows 11[p.2 800,00];MS Visual Studio 2022 Profes
3	Разработка бизнес-плана	14 дней	Пн 04.12.23	Чт 21.12.23		
4	Описание резюме проекта	2 дней	Пн 04.12.23	Вт 05.12.23	2	Менеджер проекта;Электронергия[16]
5	Описание инновационной идеи	2 дней	Ср 06.12.23	Чт 07.12.23	4	Менеджер проекта;Электронергия[16]
6	Описание продукции	2 дней	Пт 08.12.23	Пн 11.12.23	5	Менеджер проекта;Электронергия[16]
7	Анализ рынка	3 дней	Вт 12.12.23	Чт 14.12.23	6	Менеджер проекта;Электронергия[24]
8	Составление маркетингового плана	3 дней	Пт 15.12.23	Вт 19.12.23	7	Менеджер проекта;Бумага[p.300,00];Электронергия[24]
9	Составление финансового плана	2 дней	Ср 20.12.23	Чт 21.12.23	8	Менеджер проекта;Электронергия[16]
10	Завершение разработки бизнес-плана	0 дней	Чт 21.12.23	Чт 21.12.23	9	
11	Разработка мобильного приложения для проведения квест	102 дней	Пт 22.12.23	Пн 13.05.24		
12	Анализ разработки программных средств	10 дней	Пт 22.12.23	Чт 04.01.24		
13	Анализ предметной области	2 дней	Пт 22.12.23	Пн 25.12.23	10	Аналитик[50%];Менеджер проекта[50%];Электронергия[16]
14	Технический анализ конкурентов	2 дней	Вт 26.12.23	Ср 27.12.23	13	Аналитик[50%];Менеджер проекта[50%];Электронергия[16]
15	Постановка задачи	1 день	Чт 28.12.23	Чт 28.12.23	14	Аналитик[50%];Менеджер проекта[25%];Проектировщик[25%];Электронергия[8]
16	Анализ требований к программным средствам	3 дней	Пт 29.12.23	Вт 02.01.24	15	Аналитик[50%];Менеджер проекта[25%];Проектировщик[25%];Электронергия[24];Бумага[p.300,00]
17	Выбор технологий и средств разработки	2 дней	Ср 03.01.24	Чт 04.01.24	16	Аналитик[25%];Менеджер проекта[25%];Проектировщик[50%];Электронергия[16]
18	Завершение анализа разработки	0 дней	Чт 04.01.24	Чт 04.01.24	17	
19	Проектирование программных средств	10 дней	Пт 05.01.24	Чт 18.01.24		
20	Проектирование серверной части	5 дней	Пт 05.01.24	Чт 11.01.24	18	Проектировщик;Электронергия[40]
21	Проектирование структуры базы данных	2 дней	Пт 12.01.24	Пн 15.01.24	20	Проектировщик[75%];Аналитик[25%];Электронергия[16]
22	Проектирование клиентского мобильного приложения	3 дней	Вт 16.01.24	Чт 18.01.24	21	Проектировщик;Электронергия[24]
23	Завершение проектирования	0 дней	Чт 18.01.24	Чт 18.01.24	22	
24	Реализация программных средств	65 дней	Пт 19.01.24	Чт 18.04.24		
25	Реализация серверной части	30 дней	Пт 19.01.24	Чт 29.02.24	23	Backend-разработчик;Электронергия[240]
26	Реализация клиентского мобильного приложения	35 дней	Пт 01.03.24	Чт 18.04.24	25	Разработчик мобильных приложений;Электронергия[280]
27	Завершение реализации	0 дней	Чт 18.04.24	Чт 18.04.24	26	
28	Тестирование программных средств	10 дней	Пт 19.04.24	Чт 02.05.24		
29	Функциональное тестирование	3 дней	Пт 19.04.24	Вт 23.04.24	27	Тестировщик;Электронергия[24]
30	Оценка метрик покрытия операторов и условий	2 дней	Ср 24.04.24	Чт 25.04.24	29	Тестировщик;Электронергия[16]
31	Оценка метрик качества кода	2 дней	Пт 26.04.24	Пн 29.04.24	30	Тестировщик;Электронергия[16]
32	Модульное тестирование	3 дней	Вт 30.04.24	Чт 02.05.24	31	Тестировщик[50%];Электронергия[24];Backend-разработчик[50%]
33	Завершение тестирования	0 дней	Чт 02.05.24	Чт 02.05.24	32	Тестировщик;Электронергия[0]
34	Составление и оформление документации	7 дней	Пт 03.05.24	Пн 13.05.24	33	Менеджер проекта;Бумага[p.600,00];Электронергия[56]
35	Завершение разработки мобильного приложения для провер	0 дней	Пн 13.05.24	Пн 13.05.24	34	
36	Завершение реализации проекта	0 дней	Пн 13.05.24	Пн 13.05.24	35	

Рисунок 20 – Список задач проекта

Для визуализации описанного плана разработки была составлена диаграмма Ганта, приведенная на рисунке 21.

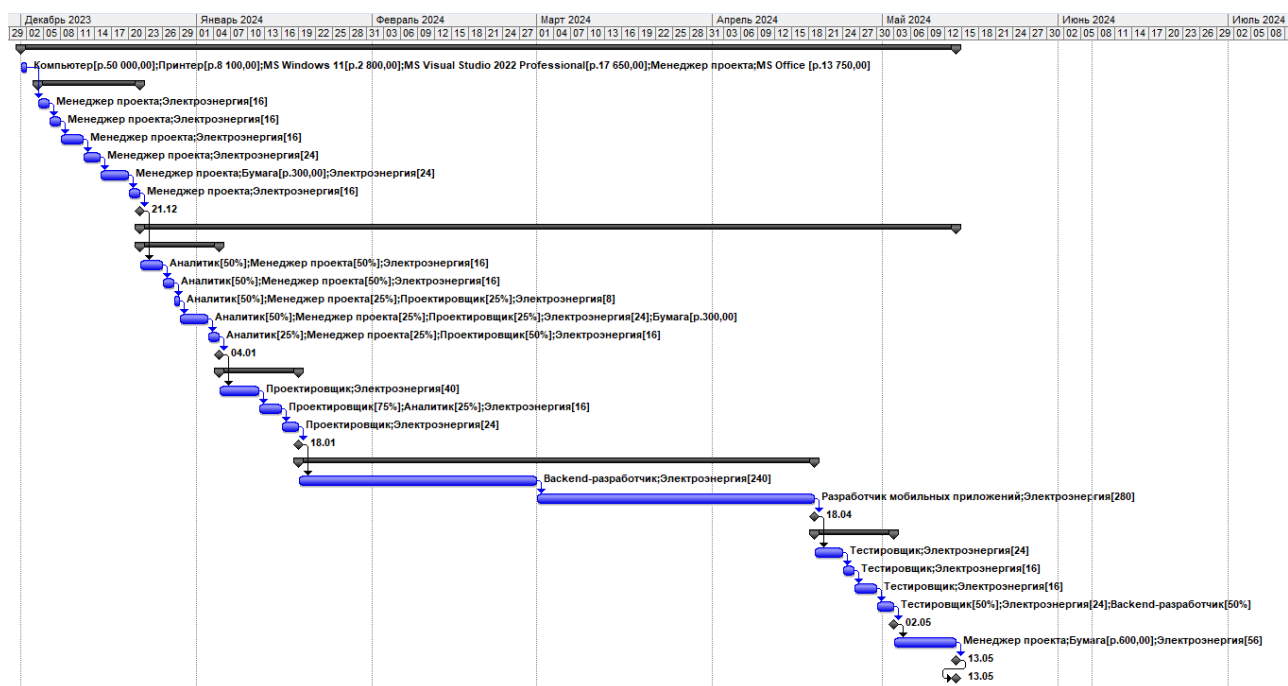


Рисунок 21 – Диаграмма Ганта

Из составленного плана проекта видно, что для его реализации потребуется 117 рабочих дней. Реализация проекта запланирована на период с 1 декабря 2023 года по 13 мая 2024 года и составляет 165 календарных дней с учетом выходных. Наиболее продолжительными работами являются реализация клиентского мобильного приложения (35 рабочих дней) и серверной части (30 рабочих дней).

2.2.5.3 Расчет себестоимости разработки программного продукта

Затраты на реализацию проекта можно разделить на несколько категорий:

- заработная плата специалистов;
- покупка оборудования;
- покупка программного обеспечения;
- прочие расходы.

Для ведения деятельности необходимо воспользоваться услугами шести специалистов, почасовой уровень заработной платы которых приведен на рисунке 22.

Название ресурса	Тип	Макс. единиц	Стандартная ставка	Ставка сверхурочных	Начисление
Менеджер проекта	Трудовой	100%	р.175,00/ч	р.300,00/ч	Пропорциональное
Аналитик	Трудовой	100%	р.150,00/ч	р.250,00/ч	Пропорциональное
Проектировщик	Трудовой	100%	р.250,00/ч	р.400,00/ч	Пропорциональное
Backend-разработчик	Трудовой	100%	р.190,00/ч	р.500,00/ч	Пропорциональное
Разработчик мобильных пр	Трудовой	100%	р.190,00/ч	р.500,00/ч	Пропорциональное
Тестирующий	Трудовой	100%	р.150,00/ч	р.380,00/ч	Пропорциональное

Рисунок 22 – Заработная плата сотрудников

Распределение задач между специалистами приведено на рисунке 23, из которого видно, что с наибольшим уровнем трудозатрат в 280 и 248 часов должны работать разработчик мобильных приложений и backend-разработчик соответственно. На третьем месте по объёму рабочего времени находится менеджер проекта, отвечающий за координацию работ сотрудников, управление ресурсами, а также за составление документации.

Менеджер проекта 201,33 ч					
И д.	Название задачи	Единицы	Трудозатраты	Начало	Окончание
4	Описание резюме проекта	100%	16 ч	Пн 04.12.23	Вт 05.12.23
5	Описание инновационной идеи	100%	16 ч	Ср 06.12.23	Чт 07.12.23
6	Описание продукции	100%	16 ч	Пт 08.12.23	Пн 11.12.23
7	Анализ рынка	100%	24 ч	Вт 12.12.23	Чт 14.12.23
8	Составление маркетингового плана	100%	24 ч	Пт 15.12.23	Вт 19.12.23
9	Составление финансового плана	100%	16 ч	Ср 20.12.23	Чт 21.12.23
34	Составление и оформление докумен	100%	56 ч	Пт 03.05.24	Пн 13.05.24
2	Закупка оборудования и ПО	100%	8 ч	Пт 01.12.23	Пт 01.12.23
13	Анализ предметной области	50%	5,33 ч	Пт 22.12.23	Пн 25.12.23
14	Технический анализ конкурентов	50%	8 ч	Вт 26.12.23	Ср 27.12.23
15	Постановка задачи	25%	2 ч	Чт 28.12.23	Чт 28.12.23
17	Выбор технологий и средств разраб	25%	4 ч	Ср 03.01.24	Чт 04.01.24
16	Анализ требований к программным с	25%	6 ч	Пт 29.12.23	Вт 02.01.24
Аналитик 40 ч					
И д.	Название задачи	Единицы	Трудозатраты	Начало	Окончание
13	Анализ предметной области	50%	8 ч	Пт 22.12.23	Пн 25.12.23
21	Проектирование структуры базы да	25%	4 ч	Пт 12.01.24	Пн 15.01.24
14	Технический анализ конкурентое	50%	8 ч	Вт 26.12.23	Ср 27.12.23
17	Выбор технологий и средств разраб	25%	4 ч	Ср 03.01.24	Чт 04.01.24
16	Анализ требований к программным с	50%	12 ч	Пт 29.12.23	Вт 02.01.24
Проектировщик 92 ч					
И д.	Название задачи	Единицы	Трудозатраты	Начало	Окончание
21	Проектирование структуры базы данных	75%	12 ч	Пт 12.01.24	Пн 15.01.24
15	Постановка задачи	25%	2 ч	Чт 28.12.23	Чт 28.12.23
20	Проектирование серверной части	100%	40 ч	Пт 05.01.24	Чт 11.01.24
22	Проектирование клиентского мобильного	100%	24 ч	Вт 16.01.24	Чт 18.01.24
17	Выбор технологий и средств разработки	50%	8 ч	Ср 03.01.24	Чт 04.01.24
16	Анализ требований к программным средст	25%	6 ч	Пт 29.12.23	Вт 02.01.24
Backend-разработчик 248 ч					
И д.	Название задачи	Единицы	Трудозатраты	Начало	Окончание
32	Модульное тестирование	50%	8 ч	Вт 30.04.24	Ср 01.05.24
25	Реализация серверной	100%	240 ч	Пт 19.01.24	Чт 29.02.24
Разработчик мобильных приложений 280 ч					
И д.	Название задачи	Единицы	Трудозатраты	Начало	Окончание
26	Реализация клиентского мобильного	100%	280 ч	Пт 01.03.24	Чт 18.04.24
Тестировщик 68 ч					
И д.	Название задачи	Единицы	Трудозатраты	Начало	Окончание
30	Оценка метрик покрытия операторо	100%	16 ч	Ср 24.04.24	Чт 25.04.24
31	Оценка метрик качества кода	100%	16 ч	Пт 26.04.24	Пн 29.04.24
32	Модульное тестирование	50%	12 ч	Вт 30.04.24	Чт 02.05.24
33	Завершение тестирования	100%	0 ч	Чт 02.05.24	Чт 02.05.24
29	Функциональное тестирование	100%	24 ч	Пт 19.04.24	Вт 23.04.24

Рисунок 23 – Распределение задач по исполнителям

Расходы на покупку персонального компьютера для проведения работ, а также на закупку лицензионного программного обеспечения компании Microsoft представлены на рисунке 24. Кроме того, для работы компьютера требуется электричество, покупаемое по ставке 4,24 рубля за 1 кВт/час.

Название ресурса	Владелец назначения	Единицы	Затраты
Компьютер			р.50 000,00
Принтер			р.8 100,00
MS Windows 11			р.2 800,00
MS Visual Studio 2022 Professional			р.17 650,00
MS Office			р.13 750,00

Рисунок 24 – Расходы на материальные ресурсы

Общий отчет по проекту, составленной с помощью Microsoft Project, приведен на рисунке 25.

Даты			
Начало:	Пт 01.12.23	Окончание:	Пн 13.05.24
Баз. начало:	Пт 01.12.23	Баз. окончание:	Пн 13.05.24
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней
Длительность			
Текущий план:	117 дней	Оставшаяся:	117 дней
Базовый план:	117 дней	Фактическая:	0 дней
Отклонение:	0 дней	Процент завершения:	0%
Трудозатраты			
Текущий план:	929,33 ч	Оставшиеся:	929,33 ч
Базовый план:	929,33 ч	Фактические:	0 ч
Отклонение:	0 ч	Процент завершения:	0%
Затраты			
Текущий план:	р.272 158,37	Оставшиеся:	р.272 158,37
Базовый план:	р.272 158,37	Фактические:	р.0,00
Отклонение:	р.0,00		
Состояние задачи		Состояние ресурса	
Еще не начатые задачи:	36	Трудовые ресурсы:	12
Выполняющиеся задачи:	0	Трудовые ресурсы с прев. дост:	0
Завершенные задачи:	0	Материальные ресурсы:	1
Итого задач:	36	Итого ресурсов:	13

Рисунок 25 – Общий отчет по проекту

Также на основе составленного плана реализации проекта был сформирован отчет о бюджете (рисунок 26).

Ид.	Название задачи	Начисление фикс. затрат	Общие затраты
2	Закупка оборудования и ПО	Пропорциональное	р.93 700,00
26	Реализация клиентского мобильного	Пропорциональное	р.54 387,20
25	Реализация серверной части	Пропорциональное	р.46 617,60
34	Составление и оформление докумен	Пропорциональное	р.10 637,44
20	Проектирование серверной части	Пропорциональное	р.10 169,60
22	Проектирование клиентского мобил	Пропорциональное	р.6 101,76
16	Анализ требований к программным	Пропорциональное	р.4 751,76
8	Составление маркетингового плана	Пропорциональное	р.4 601,76
7	Анализ рынка	Пропорциональное	р.4 301,76
29	Функциональное тестирование	Пропорциональное	р.3 701,76
21	Проектирование структуры базы дан	Пропорциональное	р.3 667,84
32	Модульное тестирование	Пропорциональное	р.3 421,76
17	Выбор технологий и средств разраб	Пропорциональное	р.3 367,84
4	Описание резюме проекта	Пропорциональное	р.2 867,84
5	Описание инновационной идеи	Пропорциональное	р.2 867,84
6	Описание продукции	Пропорциональное	р.2 867,84
9	Составление финансового плана	Пропорциональное	р.2 867,84
14	Технический анализ конкурентов	Пропорциональное	р.2 667,84
30	Оценка метрик покрытия операторо	Пропорциональное	р.2 467,84
31	Оценка метрик качества кода	Пропорциональное	р.2 467,84
13	Анализ предметной области	Пропорциональное	р.2 201,17
15	Постановка задачи	Пропорциональное	р.1 454,24
10	Завершение разработки бизнес-пла	Пропорциональное	р.0,00
18	Завершение анализа разработки	Пропорциональное	р.0,00
23	Завершение проектирования	Пропорциональное	р.0,00
27	Завершение реализации	Пропорциональное	р.0,00
33	Завершение тестирования	Пропорциональное	р.0,00
35	Завершение разработки мобильного	Пропорциональное	р.0,00
36	Завершение реализации проекта	Пропорциональное	р.0,00
			р.272 158,37

Рисунок 26 – Отчет о бюджете проекта

Из отчета о бюджете видно, что основными статьями расхода при реализации проекта будут являться закупка оборудования и программного обеспечения, а также затраты на реализацию клиентского мобильного приложения и его серверной части. Общая сумма расходов составит 272158,37 рублей.

В данном разделе был произведен выбор необходимых для применения архитектурных подходов и спроектирована архитектура мобильного приложения для проведения квестов с применением технологии дополненной реальности, а также серверной части, необходимой для его функционирования. Спроектирована структура базы данных и пользовательский интерфейс мобильного приложения.

При выполнении проектирования учитывалось, что разрабатываемые в рамках данной выпускной квалификационной работы программные средства должны функционировать в составе программного комплекса для организации и проведения квестов с применением технологии дополненной реальности. Выполнена оценка сроков и стоимости реализации проекта.

2.3 Реализация программных средств для проведения квестов с применением технологии дополненной реальности

2.3.1 Реализация архитектуры программных средств

2.3.1.1 Реализация архитектуры серверной части

Для разработки серверной части было принято решение использовать язык C# и фреймворк ASP.NET Core, что должно обеспечить высокую эффективность, гибкость и масштабируемость программного средства, а также позволить удобно и быстро вести процесс разработки за счет богатой экосистемы с широким спектром библиотек [42].

Результат реализации серверной части приложения приведен в качестве UML-диаграммы компонентов (рисунок 27) исходного кода сервиса `questie_backend`, выполняющего обработку основных запросов к серверной

части. Стоит отметить, что для уменьшения объема диаграммы на ней не указаны компоненты сущностей предметной области и DTO.

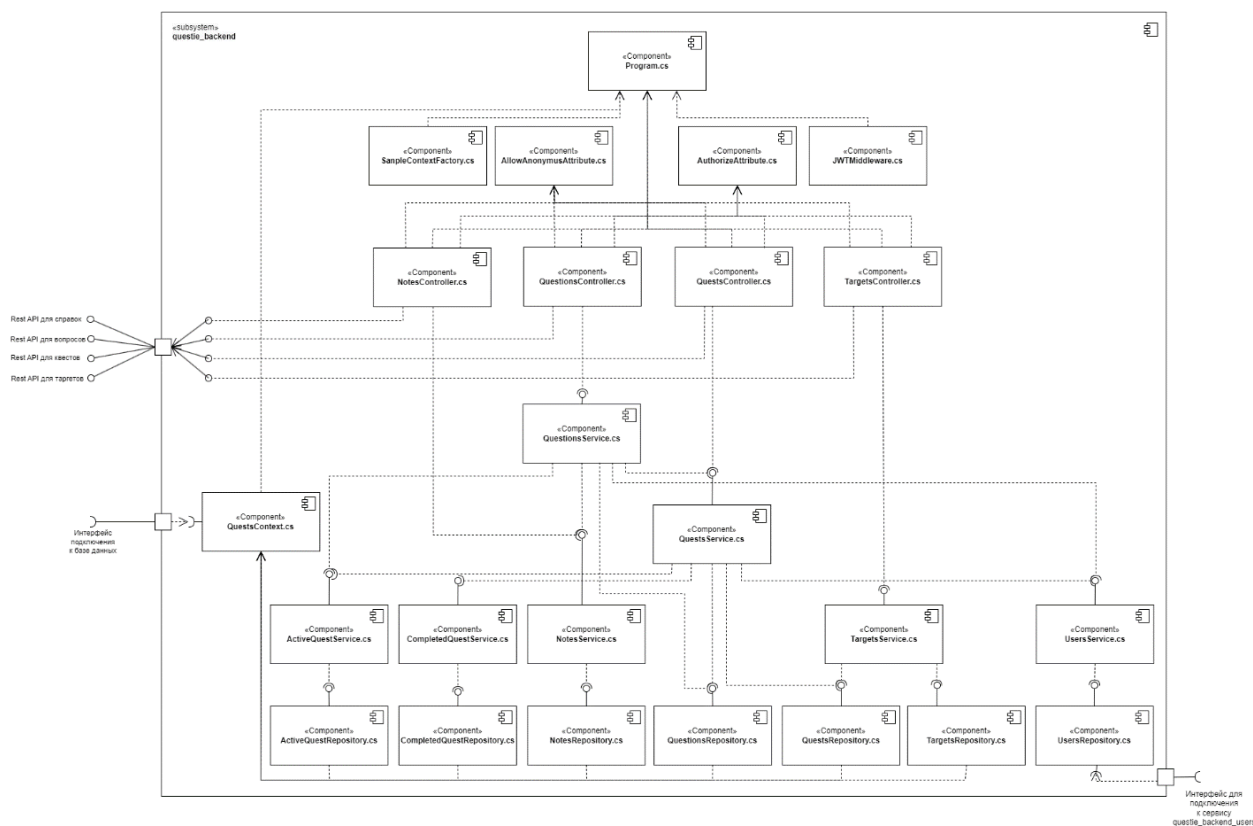


Рисунок 27 – Диаграмма компонентов сервиса questie_backend

Из диаграммы видно, что взаимодействие с сервисом questie_backend происходит через предоставляемый компонентами-контроллерами NotesController.cs, QuestionsController.cs, QuestsController.cs, TargetsController.cs интерфейс REST API. Все контроллеры предоставляют доступ к своему функционалу через единый порт.

Для функционирования контроллеры обращаются к интерфейсам, предоставляемым набором сервисных компонентов, в свою очередь также использующих интерфейсы соответствующих им компонентов-репозитория. Все компоненты-репозитории, кроме UsersRepository.cs зависят от компонента QuestContext.cs, для функционирования которого требуется обращение к внешнему интерфейсу базы данных PostgreSQL 15.3. Компонент-репозиторий

UsersRepository.cs обращается к внешнему интерфейсу для получения данных от сервиса questie_backend_users.

Все компоненты-контроллеры зависят от компонентов AllowAnonymusAttribute.cs и AuthorizeAttribute.cs, реализующих атрибуты проверки авторизации пользователей в системе.

Общая структура реализации сервисов questie_backend_authorization и questie_backend_users имеет похожий характер.

Классы, описывающие компоненты сущностей предметной области и DTO были вынесены в отдельные пакеты и разбиты следующим образом:

- common_entities – классы, описывающие перечисления и модели сущностей предметной области и применяющиеся во всех четырех сервисах серверной части приложения;

- questie_entities – классы, описывающие DTO сущностей, относящихся к квестам и применяющихся в сервисе questie_backend;

- questie_users – классы, описывающие DTO сущностей, относящихся к пользователям и применяющихся в сервисах questie_backend, questie_backend_authorization, questie_backend_users.

Описанные пакеты преобразуются в исполняемые файлы common_entities.dll, questie_entities.dll и questie_users.dll соответственно.

Сервисы questie_backend, questie_backend_authorization и questie_backend_users предоставляют для взаимодействия интерфейс REST API, описание которого соответствует спецификации OpenAPI. В качестве примера использования API на рисунке 28 приведен графический интерфейс Swagger UI, описывающий PUT-запрос на изменение исторической справки к сервису questie_backend. При формировании приведенного запроса в качестве параметра передается идентификатор обновляемой исторической справки, а обновляемые данные передются в теле запроса.

PUT /api/Notes

Parameters

Name Description

id string(\$uuid) (query) ab3d24c9-5d19-47d9-a490-5f223e9786f4

Request body multipart/form-data

Title string Пенза: первая встреча ☐ Send empty value

Description string Визит в Пензу – это возможность познакомиться с историей ☐ Send empty value

ImageFile string(\$binary) cathedral.jpg ☐ Send empty value

Execute Clear

Responses

Curl

```
curl -X 'PUT' \
  'https://localhost:5003/api/Notes?id=ab3d24c9-5d19-47d9-a490-5f223e9786f4' \
  -H 'accept: */*' \
  -H 'Content-Type: multipart/form-data' \
  -F 'Title=Пенза: первая встреча' \
  -F 'Description=Визит в Пензу – это возможность познакомиться с историей и архитектурой старинного русского города, съездить на родину Михаила Юрьевича Лермонтова, отдохнуть в ахунских сосновых борах. Цените' \
  -F 'ImageFile=cathedral.jpg;type=Image/jpeg'
```

Request URL

https://localhost:5003/api/Notes?id=ab3d24c9-5d19-47d9-a490-5f223e9786f4

Server response

Code Details

204 undocumented

Response headers

```
access-control-allow-origin: *
date: Tue, 07 May 2024 18:26:46 GMT
server: Kestrel
```

Responses

Code	Description	Links
200	Success	No links

Рисунок 28 – PUT-запрос (изменение исторической справки)

Таким образом, был разработан программный код серверной части мобильного приложения для проведения квестов с применением технологии дополненной реальности. Исходный код реализации основного функционала приведен в приложении Б. Подана заявка на получение свидетельства о государственной регистрации программы для ЭВМ «Серверное приложение управления системой для исторических квест-экскурсий с применением технологии дополненной реальности».

2.3.1.2 Реализация архитектуры клиентского мобильного приложения

Для решения поставленных задач была выбрана кроссплатформенная среда разработки игр и мобильных приложений Unity. Выбор обусловлен разнообразием и высоким уровнем развития инструментария, предоставляемого для разработки приложений с дополненной реальностью. В качестве языка программирования был выбран наиболее популярный для Unity C#, обеспечивающий высокую скорость и удобство разработки.

Получение информации для наполнения экранов было реализовано в слое моделей путем выполнения сетевых запросов к REST API серверной части с использованием класса `UnityWebRequest` пространства имен `UnityEngine.Networking`. Для работы с ответами формата json применялся встроенный API `Unity JsonUtility` [43] из пространства имен `System.IO`.

Авторизация в приложении реализована на основе JWT, получаемых от сервиса авторизации `questie_backend_authorization`. А их хранение осуществляется в хранилище `PlayerPrefs`.

Большой интерес в реализации проекта представляет применение технологии дополненной реальности. Для Unity характерно разнообразие и высокий уровень развития различных библиотек, обеспечивающих возможность решения данной задачи. При выборе средств для ее реализации необходимо учесть:

- возможность распознавания изображений;
- возможность распознавания 3D-объектов с одного ракурса;
- возможность распознавания 3D-объектов со всех сторон;
- наличие документации.

Данному набору требований наилучшим образом соответствует комплект для разработки программного обеспечения (SDK) `Vuforia`.

Для создания экрана дополненной реальности с распознаванием 3D-объекта реального мира с помощью Vuforia 10.20.3 необходимо выполнить следующие действия:

- 1) создать 3D-модель объекта реального мира;
- 2) с помощью программного обеспечения Model Target Generator, предоставляемого разработчиками Vuforia, сгенерировать таргет-модель для распознавания (рисунок 29);
- 3) установить на сцене Unity игровой объект ARCamera, необходимый для обнаружения целевого объекта реального мира;
- 4) установить на сцене Unity объект Model Target Behaviour (Script) и добавить в него сгенерированную таргет-модель для распознавания (рисунок 30);
- 5) установить объекты и скрипты, появляющиеся и выполняющиеся после обнаружения камерой объекта реального мира, соответствующего таргет-модели [44].

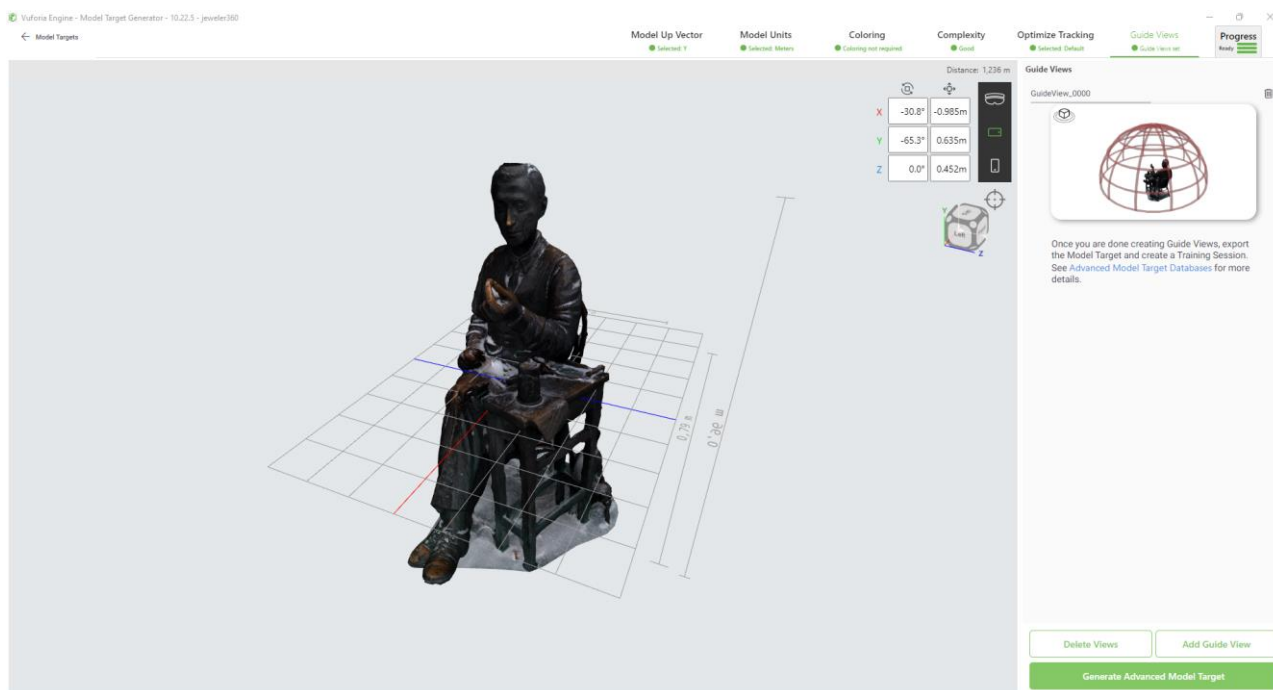


Рисунок 29 – Генерация таргета с помощью Model Target Generator

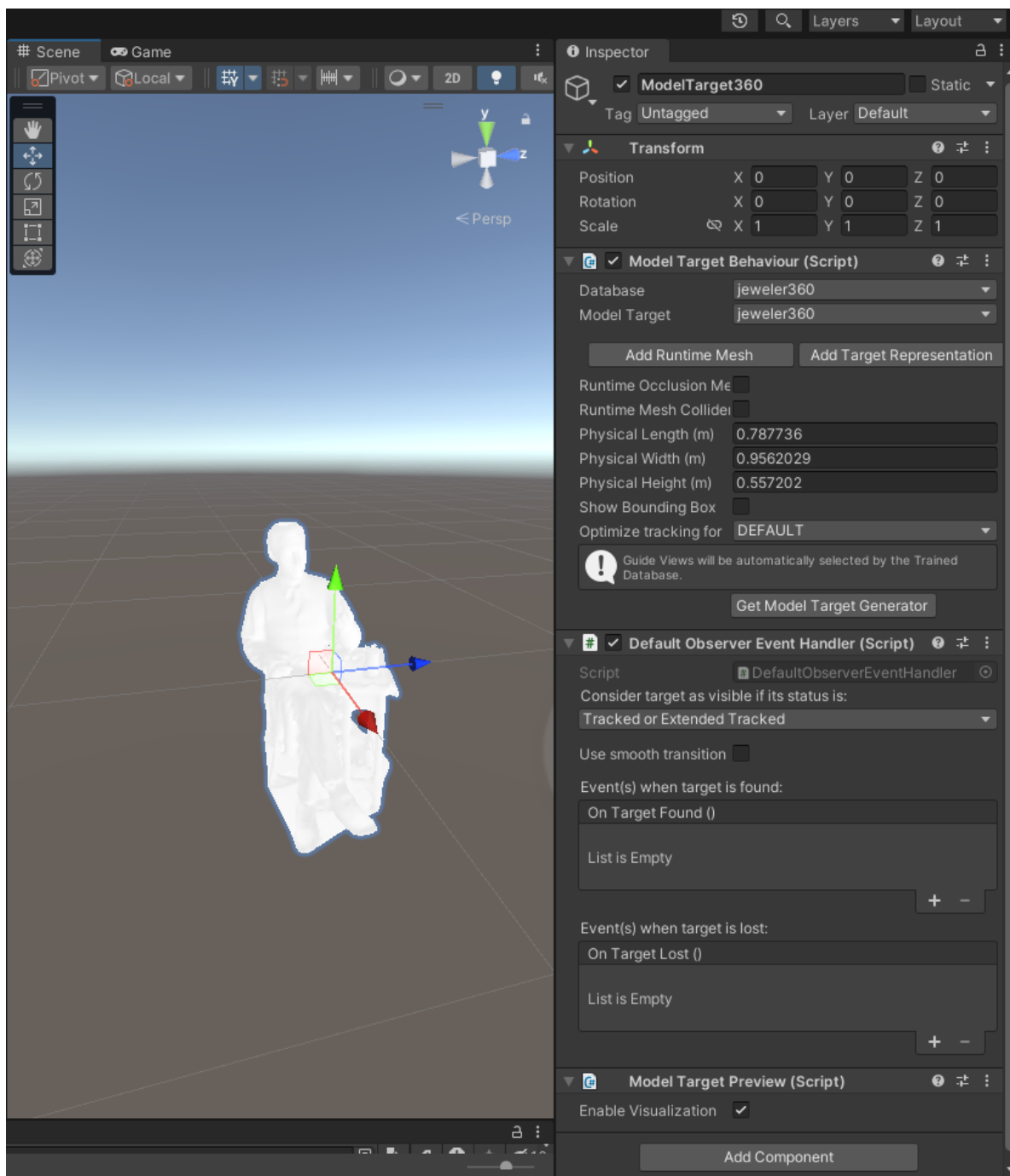


Рисунок 30 – Установка таргета на сцене Unity

Все компоненты разрабатываемого мобильного приложения были разделены на две сцены – главное меню «MenuScene» и главная сцена игры «QuestScene». Укрупненная UML-диаграмма компонентов, отражающая общее устройство мобильного приложения, приведена на рисунке 31.

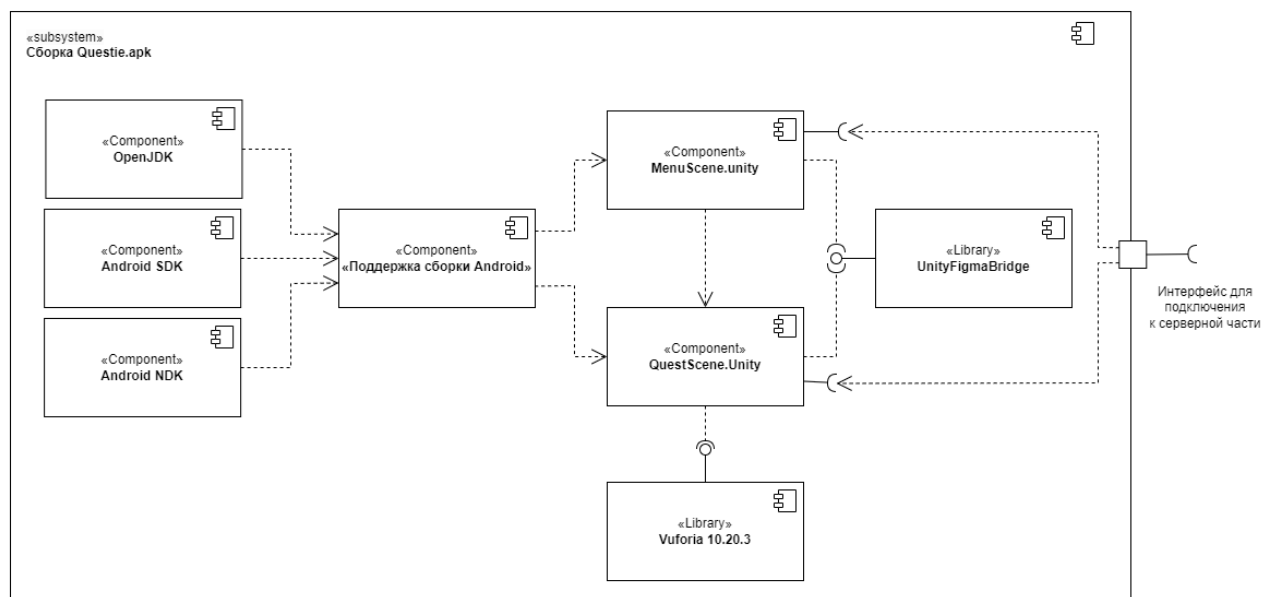


Рисунок 31 – Укрупненная диаграмма компонентов мобильного приложения

Для сборки исполняемого файла в формате «.APK» для установки приложения на устройства с ОС Android необходимо использовать модуль Unity «Поддержка сборки Android», который в свою очередь имеет следующие зависимости:

- комплект разработки программного обеспечения для Android (SDK);
- нативный комплект средств разработки (NDK);
- комплект для разработки Java (OpenJDK).

Также на диаграмме приведены зависимости компонентов сцен от используемых внешних библиотек. Кроме того, компоненты сцен обращаются к API серверной части.

Таким образом, на данном этапе выполнения выпускной квалификационной работы было реализовано клиентское мобильное приложение для проведения квестов с применением технологии дополненной реальности. Исходный код реализации основного функционала приведен в приложении Б. Планируется подача заявки на получение свидетельства о государственной регистрации программы для ЭВМ «Мобильное приложение для прохождения исторических квест-экскурсий с применением технологии дополненной реальности».

2.3.2 Реализация базы данных

Для формирования базы данных был разработан сервис migrator, проводящий создание схемы базы данных PostgreSQL из классов, описывающих сущности предметной области, на основе последовательного применения разработанного набора автоматически сгенерированных миграций. UML-диаграмма компонентов исходного кода сервиса migrator приведена на рисунке 32.

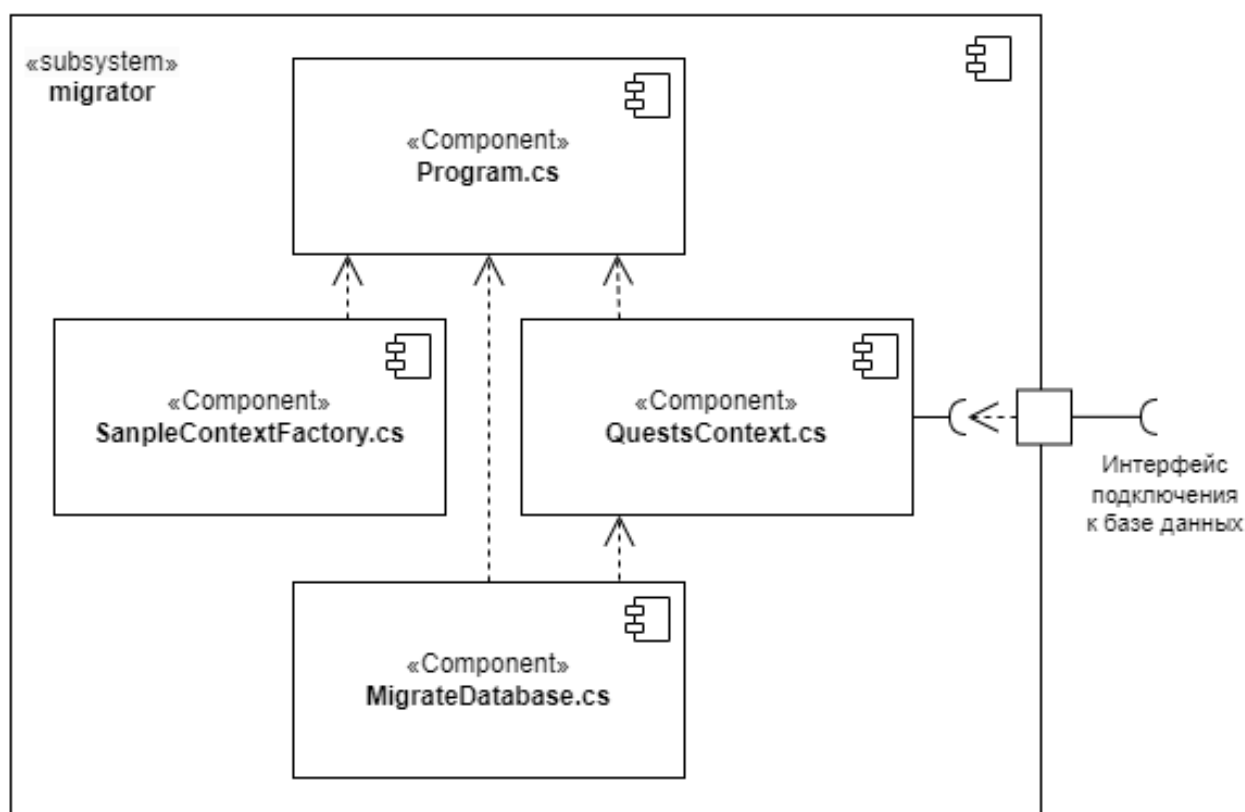


Рисунок 32 – Диаграмма компонентов сервиса migrator

Компонент QuestsContext.cs представляет собой контекст базы данных и отвечает за сопоставление программных классов и таблиц базы данных. Компонент SampleContextFactory.cs реализует шаблон проектирования «Фабрика» и отвечает за создание контекста базы данных с заданной конфигурацией подключения. Компонент MigrateDatabase.cs предназначен для запуска миграций.

Таким образом, с помощью фреймворка EntityFramework с применением подхода CodeFirst была реализована база данных, необходимая для функционирования разрабатываемого программного обеспечения.

2.3.3 Реализация пользовательского интерфейса клиентского мобильного приложения

Разработка мобильного приложения производилась в средах разработки Unity и Microsoft Visual Studio 2022 Community Edition с использованием языка программирования C#. Проект представляет собой совокупность исполняемых файлов, а также ресурсов (изображений, моделей и т. д.), хранящихся в его директориях.

Первым этапом реализации пользовательского интерфейса стал перенос заранее заготовленных прототипов из графического редактора для прототипирования и разработки интерфейсов Figma. Для ускорения этого процесса была использована открытая библиотека UnityFigmaBridge [45], позволяющая осуществить сопоставления элементов Figma с стандартными элементами пользовательского интерфейса Unity. Качество переноса некоторых элементов прототипов оказалось недостаточным для непосредственного использования в приложении и потребовало ручного редактирования для устранения неточностей. Однако использование данной библиотеки позволило значительно упростить и ускорить разработку пользовательского интерфейса в Unity, а также позволило беспрепятственно перенести адаптивность большинства прототипов.

Для ускорения разработки и организации переиспользования игровых объектов были сформированы и применены префабы [46]. Префабы применялись как для формирования отдельных групп пользовательских элементов (например, карточек квестов), так и для формирования экранов. Перечень префабов, описывающих различные экраны мобильного приложения приведен на рисунке 34.

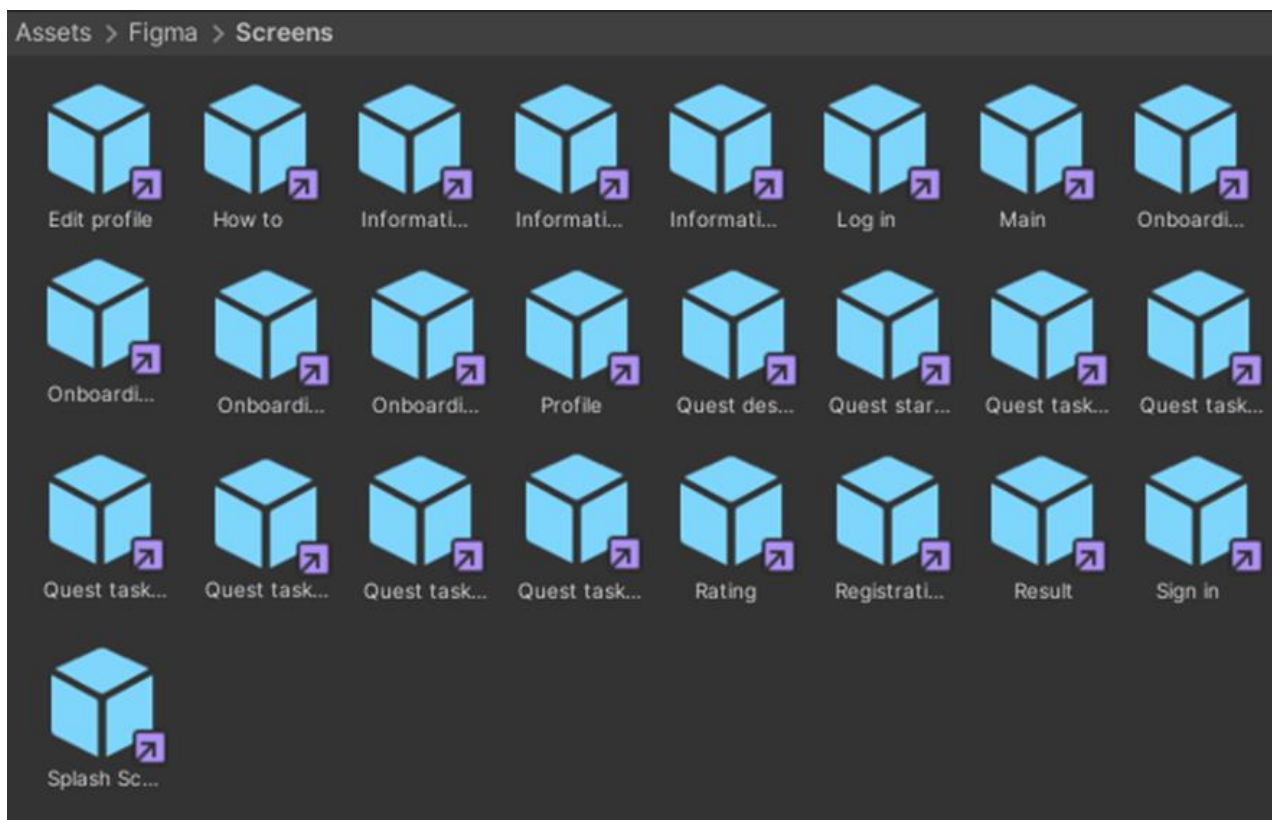


Рисунок 34 – Префабы экранов мобильного приложения

Экранами мобильного приложения, отвечающими за предоставление основного пользовательского функционала, стали:

- экран со списком квестов;
- экран с описанием квеста;
- экраны заданий различных типов;
- экран исторической справки;
- экран AR-сопровождения.

Панели данных экранов, отвечающие за вывод карточек квестов, наборов сопровождающих изображений, тегов, а также вариантов ответов были реализованы на основе компонента `ScrollRect`, что дало возможность сделать их отображение гибким и уместить любое количество элементов списков, сделав возможной их прокрутку.

Примеры данных экранов приведены на рисунке 35.

Для организации плавности смены экранов был применен компонент Animator, позволяющий настроить анимации переходов.

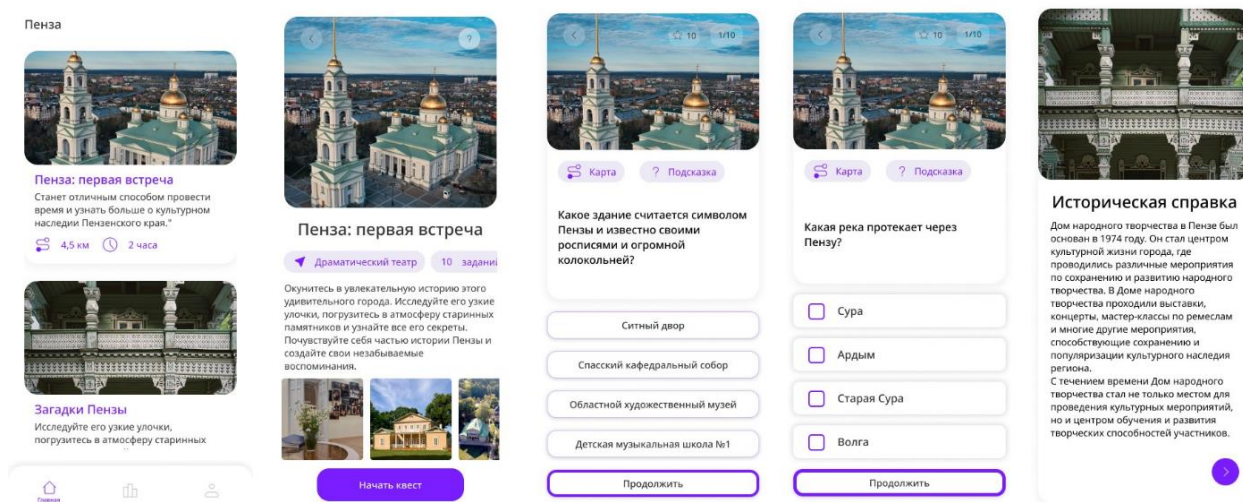


Рисунок 35 – Основные экраны мобильного приложения

Скриншоты некоторых других экранов мобильного приложения приведены на рисунке 36.

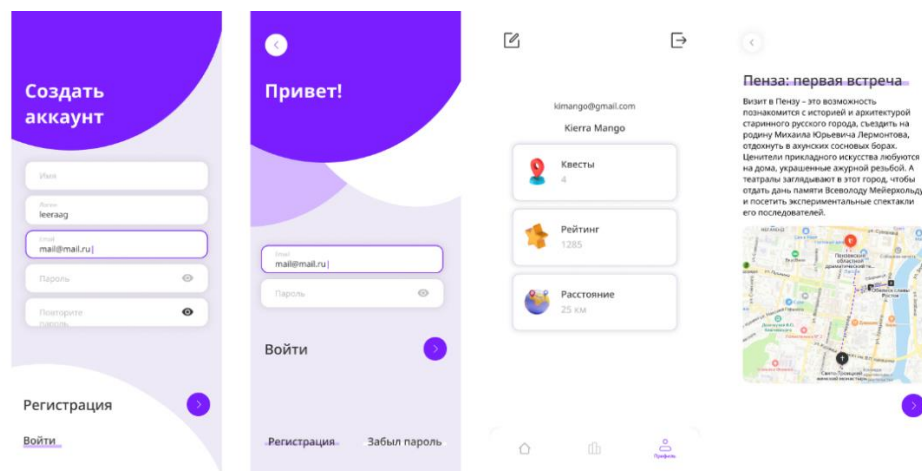


Рисунок 36 – Экраны мобильного приложения

Таким образом, на основе спроектированных прототипов был реализован пользовательский интерфейс клиентского мобильного приложения для проведения квестов с применением технологии дополненной реальности.

2.3.4 Тестирование и оценка качества кода

Для обеспечения качества разрабатываемого продукта необходимо провести тестирование. При этом для различных частей программного обеспечения должны быть выбраны наиболее подходящие методы тестирования.

2.3.4.1 Функциональное тестирование

Мобильное приложение для проведения квестов с применением технологии дополненной реальности требует ввода данных от пользователя в трёх ситуациях: авторизация пользователя, регистрация пользователя и ввод ответа на квестовое задание в свободной форме (валидируется лишь длина вводимых данных).

Было принято решение провести функциональное тестирование авторизации пользователя, как варианта использования, применяемого чаще регистрации и требующего более серьезной валидации, чем ответ на квестовое задание. Для проведения данного тестирования был выбран метод функциональных диаграмм, который помогает систематически выбирать высокорезультативные тесты и избегать неэффективного тестирования, возникающего вследствие выбора произвольного подмножества тестов [47].

Техническое задание на тестируемый участок мобильного приложения: пользователь должен иметь возможность войти в свой аккаунт. Для авторизации пользователя необходимо ввести его логин и пароль, зарегистрированные в системе. Длина логина должна составлять от 6 до 20 символов; длина пароля – от 8 до 20 символов. При этом и логин, и пароль допускают использование только строчных и заглавных букв латинского алфавита, цифр и нижнего подчеркивания в своем составе. Пароль также может содержать символы решетки, амперсанда, вопросительного и восклицательного знаков. В случае нарушения любого из требований к данным, необходимым для авторизации, должно выводиться соответствующее сообщение об ошибке.

При анализе технического задания можно выделить причины и следствия событий.

Причины событий:

- П1 – корректная длина логина;
- П2 – корректная длина пароля;
- П3 – логин состоит из корректных символов;
- П4 – пароль состоит из корректных символов;
- П5 – пользователь с заданными учетными данными зарегистрирован в системе.

Следствия событий:

- С1 – сообщение: «Некорректная длина логина»;
- С2 – сообщение: «Некорректная длина пароля»;
- С3 – сообщение: «Некорректные символы в составе логина»;
- С4 – сообщение: «Некорректные символы в составе пароля»;
- С5 – сообщение: «Неверный логин или пароль»;
- С6 – успешная авторизация. Перенаправление в аккаунт.

Выявленные причины и следствия событий были представлены в виде функциональной диаграммы (рисунок 37).

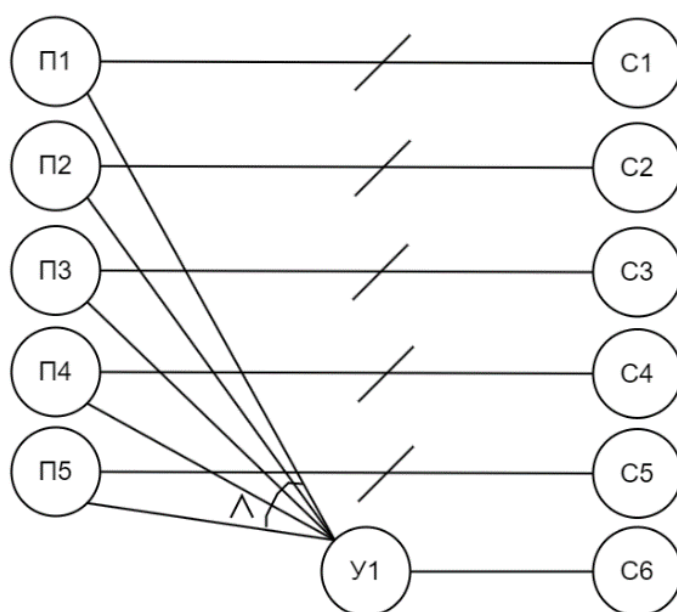


Рисунок 37 – Функциональная диаграмма тестирования авторизации

На основе функциональной диаграммы была составлена таблица решений (таблица 22), отображающая комбинации причин и соответствующие им следствия. При этом было учтено, что комбинации, при которых пользователь с заданными учетными данными зарегистрирован в системе, но не выполняется условие на корректность длины и/или состава логина и/или пароля, существовать не могут. Этот факт позволил исключить почти половину возможных комбинаций причин.

Таблица 22 – Таблица решений тестирования авторизации пользователя

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
П1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1
П2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	1
П3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1
П4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
П5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
С1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0
С2	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0
С3	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	0
С4	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
С5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
С6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

На основе таблицы решений был разработан набор тест-кейсов, по которым было проведено тестирование (таблица Г.1).

Таким образом, было проведено функциональное тестирование методом функциональных диаграмм части мобильного приложения, выполняющей авторизацию пользователя. Проведенное тестирование показало успешный результат прохождения всех составленных тест-кейсов.

2.3.4.2 Структурное тестирование

Основная логика работы программного продукта реализуется в серверной части мобильного приложения и также требует детального тестирования. Рассмотрим выполнение структурного тестирования на примере метода обновления исторической справки `UpdateNoteAsync` сервисного класса `NotesService.cs`.

На основе программного кода реализации данного метода, приведенного на рисунке 38, была составлена схема (рисунок 39), описывающая ход его выполнения.

```
1  public async Task<bool> UpdateNoteAsync(Guid id, NoteDto note)
2  {
3      Note? oldNote = await _notesRepository.GetNoteByIdAsync(id);
4      if (oldNote is null)
5      {
6          return false;
7      }
8
9      oldNote.Title = note.Title is not null
10         ? note.Title
11         : oldNote.Title;
12     oldNote.Description = oldNote.Description;
13
14     if (note.ImageFile is not null)
15     {
16         DeleteImage(oldNote.ImageName);
17         oldNote.ImageName = await SaveImage(note.ImageFile);
18     }
19
20     await _notesRepository.UpdateAsync(oldNote);
21
22     return true;
23 }
24
```

Рисунок 38 – Исходный код тестируемого метода обновления исторической справки

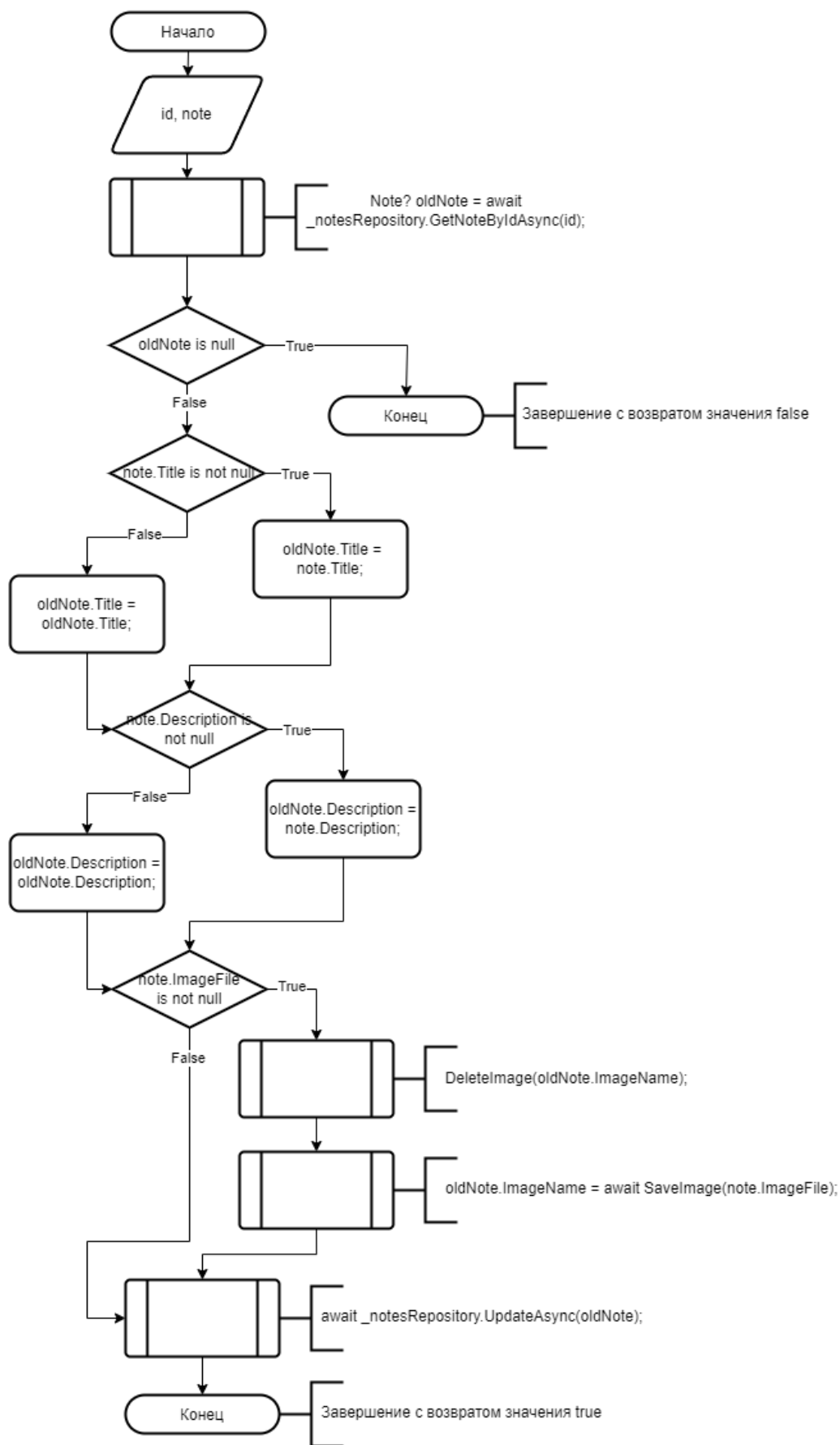


Рисунок 39 – Схема тестируемого метода обновления исторической справки

На основании схемы был составлен тест-кейс, покрывающий все операторы (таблица 23).

Таблица 23 – Тест-кейс покрытия операторов метода обновления исторической справки

Входные данные	Операторы	Условия	Ожидаемый результат
Существующий id; note, содержащий значения в свойствах Title, Description, ImageFile	1,2,6,9,11,12,13	3 (false), 4 (true), 7 (true), 10 (true)	Обновление успешно. Метод вернул значение true

Для покрытия условий были составлены три тест-кейса. Один из них повторяет тест-кейс, необходимый для покрытия операторов, два других приведены в таблице 24.

Таблица 24 – Тест-кейс покрытия условий метода обновления исторической справки

Входные данные	Операторы	Условия	Ожидаемый результат
Несуществующий id; note, содержащий значения в свойствах Title, Description, ImageFile	1,2	3 (true)	Обновление завершено неудачей. Метод вернул значение false
Существующий id; note, не содержащий значения в свойствах Title, Description, ImageFile	1,2,5,8,13	3 (false), 4 (false), 7 (false), 10 (false)	Обновление успешно. Метод вернул значение true

Поскольку все условия рассматриваемого метода не являются составными и не являются вложенными, приведенных тест-кейсов достаточно, чтобы обеспечить покрытие маршрутов. Результирующий список тест-кейсов с описаниями маршрутов, а также результат тестирования приведены в таблице 25.

Таблица 25 – Тест-кейс покрытия маршрутов метода обновления исторической справки

Входные данные	Маршрут	Ожидаемый результат	Полученный результат
Несуществующий id; note, содержащий значения в свойствах Title, Description, ImageFile	1,2,3	Обновление завершено неудачей. Метод вернул значение false	Обновление завершено неудачей. Метод вернул значение false

Продолжение таблиц 25

Входные данные	Маршрут	Ожидаемый результат	Полученный результат
Существующий id; note, не содержащий значения в свойствах Title, Description, ImageFile	1,2,3,4,5,7,8,10,13	Обновление успешно. Метод вернул значение true	Обновление успешно. Метод вернул значение true
Существующий id; note, содержащий значения в свойствах Title, Description, ImageFile	1,2,3,4,6,7,9,10,11,12,13	Обновление успешно. Метод вернул значение true	Обновление успешно. Метод вернул значение true

Таким образом, было проведено структурное тестирование метода обновления исторической справки UpdateNoteAsync сервисного класса NotesService.cs, составлены и успешно пройдены тест-кейсы, обеспечивающие покрытия операторов, условий и маршрутов.

2.3.4.3 Модульное тестирование

Для корректности работы отдельных модулей разработанного программного обеспечения было проведено модульное тестирование, которое позволило в автоматическом режиме проверить корректность работы функций программы, а также обеспечило возможность экономии времени при дальнейшем развитии проекта.

Реализация модульного тестирования проводилась с использованием фреймворков XUnit (непосредственно для разработки модульных тестов) и Moq (для имитации функциональности и создания мок-объектов). Кроме того, использовалась библиотека FluentAssertions, которая содержит набор методов расширения, позволяющих более гибко формировать блок проверок.

В качестве примера в приложении Д приведен исходный код реализации модульных тестов для класса-контроллера взаимодействия с описаниями таргетов TargetsController, а результаты успешного прохождения данных тестов приведены на рисунке 40.

Тестирование	Длительность	Сводка по группе
<ul style="list-style-type: none"> questie_backend.Tests.Controllers TestTargetsController (22) CreateTarget_BadRequestRe... CreateTarget_BadRequestRe... CreateTarget_BadRequestRe... CreateTarget_CreatedAtAction DeleteTarget_BadRequestRe... DeleteTarget_BadRequestRe... DeleteTarget_NoContentRes... DeleteTarget_NotFoundResu... GetAllTargets_ShouldReturn... GetTargetById_BadRequestR... GetTargetById_BadRequestR... GetTargetById_NotFoundRes... GetTargetById_TargetOutDto GetTargetByQuestId_BadReq... GetTargetByQuestId_BadReq... GetTargetByQuestId_NotFou... GetTargetByQuestId_TargetO... UpdateTarget_BadRequestRe... UpdateTarget_BadRequestRe... UpdateTarget_BadRequestRe... UpdateTarget_NoContentRe... UpdateTarget_NotFoundRes... 	<ul style="list-style-type: none"> 69 мс 69 мс < 1 мс < 1 мс < 1 мс 8 мс < 1 мс 1 мс < 1 мс 2 мс 2 мс < 1 мс < 1 мс < 1 мс 1 мс < 1 мс < 1 мс < 1 мс 53 мс < 1 мс < 1 мс < 1 мс < 1 мс 2 мс 	<p>TestTargetsController</p> <p>Тесты в группе: 22</p> <p>Общая длительность: 69 мс</p> <p>Результаты</p> <p>22 Пройден</p>

Рисунок 40 – Результат прохождения модульных тестов для класса-контроллера TargetsController

Таким образом, модульное тестирование показало корректность функционирования разработанного программного обеспечения.

2.3.4.4 Метрики качества кода

Качественное программное обеспечение должно не только корректно выполнять все требующиеся от него функции, но и обеспечивать возможность дальнейшей доработки и расширения, что накладывает дополнительные требования к качеству реализации программного кода. Существуют различные метрики, позволяющие получить численные характеристики свойств кода и оценить уровень его качества.

Для проверки качества реализации данной выпускной квалификационной работы было принято решение воспользоваться одной из наиболее популярных метрик – метрикой Холстеда, которая позволяет оценивать сложность разработанного программного кода на основе длины программы и её словаря.

Списки операторов и операндов, необходимый для расчета метрики Холстеда по классу-сервису ActiveQuestService, управляющему активными квестами, приведены в таблице 26.

Таблица 26 – Списки операторов и операндов класса ActiveQuestService

Операторы		Операнды	
Оператор	Количество	Операнд	Количество
!=	1	_activeQuestRepository	13
(	31	activeQuest	16
)	31	activeQuestInDto	9
,	17	activeQuestRepository	2
;	35	CompletedQuestions	1
{	19	DateTime.UtcNow	2
	6	false	8
}	19	Guid.Empty	12
++	1	null	5
+=	1	oldActiveQuest	3
=	14	questId	15
==	16	QuestId	1
ActiveQuest	2	rating	2
await	11	Rating	1
CreateActiveQuestAsync	1	StartedAt	1
DeleteAsync	2	true	4
GetActiveQuestAsync	6	UpdatedAt	1
if	10	userId	15
new	1	UserId	1
return	15		
UpdateAsync	2		

Длина программы вычисляется при помощи формулы:

$$N = N_1 + N_2 = 241 + 112 = 353,$$

где N – длина программы (общее количество операторов и операндов);

N_1 – общее количество операторов;

N_2 – общее количество операндов.

Для вычисления словаря программы применяется формула:

$$n = n_1 + n_2 = 21 + 19 = 40,$$

где n – словарь программы;

p_1 – количество уникальных операторов;

p_2 – количество уникальных операндов.

Вычислительная сложность программы по метрике Холстеда вычисляется по формуле:

$$V = N * \log_2 n = 353 * \log_2 40 = 1878,6,$$

где V – вычислительная сложность программы по метрике Холстеда

N – длина программы;

n – словарь программы.

Из расчетов видно, что длина программы составляет 353, вычисленный словарь – 40, а что вычислительная сложность программы по метрике Холстеда равна 1878,6. Допустимым показателем метрики является диапазон от 20 до 1000 для одного метода. Поскольку в рассматриваемом классе реализовано семь методов, полученное значение метрики можно считать удовлетворяющим допустимому диапазону. Текст программы можно считать компактным.

Для сервиса `questie_backend` серверной части приложения был проведен автоматический подсчет метрик, выполненный с помощью интегрированной среды разработки Microsoft Visual Studio 2022 Community Edition (рисунок 41).

Индекс удобства поддержки представляет относительную простоту обслуживания кода и имеет значения от 0 (низкий уровень поддержки) до 100.

Цикломатическая сложность (или сложность организации циклов) показывает количество потенциальных путей выполнения (чем ниже показатель, тем лучше).

Глубина наследование определяется как максимальная длина от узла до корня дерева иерархии классов. Чем больше глубина наследования, тем сложнее проследить иерархию.

Взаимозависимость классов показывает сколько классов использует один класс. С увеличением числа используемых классов уменьшается гибкость кода [48].

Иерархия	Индекс удобства поддержки	Сложность организации циклов	Глубина наследования	Взаимозависимость классов	Строки исходного кода	Строки исполняемого кода
questie_backend (Debug)	84	517	4	209	3 040	835
Program	38	3	1	72	144	56
questie_backend.Controllers	81	104	2	52	553	123
CompletedQuests	96	1	2	4	49	0
NotesController	79	12	2	26	61	15
QuestionsController	74	30	2	29	133	36
QuestsController	75	28	2	26	134	36
StatisticController	88	10	2	15	57	10
TargetsController	74	23	2	25	93	26
questie_backend.Data	81	14	2	26	34	9
QuestContext	100	13	2	10	10	0
SampleContextFactory	63	1	1	17	18	9
questie_backend.Middleware	77	15	2	45	126	29
AllowAnonymousAttribute	100	1	2	3	3	0
AuthorizeAttribute	67	8	2	23	49	11
JWTMiddleware	66	6	1	29	65	18
questie_backend.Repositories.Implementations	86	63	1	35	396	102
ActiveQuestRepository	88	6	1	15	30	8
CompletedQuestRepository	90	2	1	9	14	3
NotesRepository	90	6	1	13	35	8
QuestionsRepository	87	13	1	17	58	17
QuestRepository	89	8	1	13	41	10
StatisticRepository	73	22	1	22	162	48
TargetsRepository	90	6	1	13	35	8
questie_backend.Repositories.Interfaces	100	45	0	13	87	0
questie_backend.Services.Implementations	63	221	1	87	1 588	516
ActiveQuestService	68	23	1	9	116	40
CityService	59	3	1	18	29	12
CompletedQuestService	74	4	1	9	30	6
NotesService	68	15	1	25	129	46
QuestionsService	59	49	1	42	354	121
QuestService	59	59	1	41	412	145
StatisticService	63	42	1	16	256	51
TargetsService	67	18	1	27	138	49
UserService	55	8	1	26	97	46
questie_backend.Services.Interfaces	100	51	0	30	105	0
questie_backend.authorization.Exception	100	1	4	1	7	0

Рисунок 41 – Метрики сервиса questie_backend

Из рассчитанных метрик видно, что разработанное программное обеспечение имеет высокий индекс удобства поддержки, а также нормальные цикломатическую сложность, глубину наследования и уровень взаимозависимости классов.

Таким образом, было проведено функциональное, структурное и модульное тестирование разработанного программного обеспечения, показавшее корректность работы функционала. Также был произведен автоматический и ручной подсчет различных метрик качества кода, который показал высокий уровень удобства поддержки, гибкости и компактности разработанного программного продукта.

Заключение

В ходе выполнения выпускной квалификационной работы как стартапа был разработан бизнес-план системы для исторических квест-экскурсий с применением технологии дополненной реальности, состоящий из нескольких этапов.

1. Выполнено детализированное описание инновационной идеи, предлагаемого продукта и его технико-экономических характеристик.

2. Проведен анализ рынка и сделаны прогнозы развития отрасли. Выявлен стабильной рост динамики продаж в отрасли туризма. Установлено, что уровень чистой рентабельности отрасли высок и составляет от 9 до 11%. Прогнозы развития оптимистичны. Отрасль поддерживается различными государственными программами.

3. Разработан маркетинговый план для реализации проекта. В плане была выделена целевая аудитория продукта (главным образом, туристы в возрасте от 18 до 50 лет) и рассчитана средняя цена продажи квест-экскурсий – 450 рублей на начальном этапе выхода на рынок.

4. Разработан финансовый план для реализации проекта. План, содержащий пять этапов развития бизнес-проекта и описывает ожидаемые финансовые показатели на каждом из этапов. Составленная в результате финансовая модель показала, что на четвертом этапе реализации проект должен выйти на самоокупаемость и начать приносить первую прибыль.

5. Найдены партнеры проекта и привлечено финансирование на первый этап разработки. В результате реализации первого этапа бизнес-плана было получено грантовое финансирование по программе УМНИК в размере 500000 рублей, найдены партнеры как среди действующего бизнеса, так и среди государственных организаций. Проведено первое представление проекта в СМИ.

Кроме того, было разработано мобильное приложение для проведения квестов с применением технологии дополненной реальности. В ходе разработки было выполнено несколько этапов.

1. Проведен анализ существующих программных решений для прохождения квестов. Рассмотрены приложения Ubego, Questo и Quest Quiz. Выявлены их преимущества и недостатки. На основе анализа аналогов и требований к заданию были выявлены функциональные и нефункциональные требования к разрабатываемому мобильному приложению и сформулирована задача на разработку программных средств.

2. Спроектирована архитектура серверной части мобильного приложения и база данных. Выбрана архитектура на основе сервисов, определены сервисы, из которых состоит серверная часть, спроектирована последовательность выполнения запросов, основная логика обработки запросов, структура базы данных.

3. Спроектирована архитектура клиентского мобильного приложения и его пользовательский интерфейс. Для клиентского мобильного приложения была выбрана MVP-архитектура, описано взаимодействие программных классов, спроектированы макеты пользовательского интерфейса в графическом редакторе для прототипирования и разработки интерфейсов Figma.

4. Проведено планирование необходимых работ и оценка бюджета. Установлено, что для разработки программного обеспечения необходимо 272158,37 рубля и 117 рабочих дней.

5. Реализована серверная часть мобильного приложения и база данных. Серверная часть разрабатывалась в среде разработки Microsoft Visual Studio 2022 Community Edition с использованием языка программирования C#, фреймворка ASP.Net Core 8.0, а также базы данных PostgreSQL и может быть запущена под управлением любой ОС, поддерживающей средство контейнеризации Docker. Для взаимодействия с базой данных использовался фреймворк Entity Framework.

6. Реализована архитектура клиентского мобильного приложения и пользовательский интерфейс. Разработка мобильного приложения производилась в средах разработки Unity и Microsoft Visual Studio 2022 Community Edition с использованием языка программирования C#. Генерация таргетов для дополненной реальности из 3D-моделей проводилась с помощью программного обеспечения Model Target Generator от Vuforia. Создание 3D-моделей было выполнено из фотографий с помощью нейросетевого программного модуля, разработанного участниками команды бизнес-проекта «Система для исторических квест-экскурсий» и не рассматривается в рамках данной выпускной квалификационной работы. Разработанное мобильное приложение функционирует под управлением операционной системы Android 10.0 и выше.

7. Выполнено тестирование и оценка качества кода разработанного программного обеспечения. Успешно пройдены функциональное, структурное и модульное тестирование с помощью фреймворков XUnit и Moq. Проведены расчет и оценка метрик качества кода, показавшие соответствие разработанного программного кода критериям качества.

В результате было опубликовано 2 статьи [18], входящих в Российский индекс научного цитирования. 1 статья находится на стадии издания сборника. Подана заявка на свидетельство о государственной регистрации программы для ЭВМ «Серверное приложение управления системой для исторических квест-экскурсий с применением технологии дополненной реальности». Выполняется подготовка к подаче заявки на свидетельство о государственной регистрации программы для ЭВМ на мобильное приложение для проведения для проведения квестов с применением технологии дополненной реальности.

Список использованных источников

1. Петухина, Е.В. Методические и организационные аспекты разработки и проведения квестов в рамках экскурсионной деятельности / Е.В. Петухина, В.Г. Ден // Территория новых возможностей. – 2022. №2. – С. 105 – 115.
2. Кургина, С.О. Квест-экскурсия как инновационная форма экскурсионного продукта / С.О. Кургина, М.Г. Копцева, В.И. Суржиков // АНИ: экономика и управление. – 2017. №3 (20). – С. 231 – 234.
3. Воробьев А. Внедрение дополненной реальности (AR) в обучение. Часть #4 AR и геймификация // Argument [Электронный ресурс]. URL: <https://argument.digital/blog/vnedrenie-dopolnennoj-realnosti-ar-v-obuchenie-chas4-ar-i-gejmifikaciya/> (дата обращения: 06.12.2023).
4. Зырянова, И.С. Самостоятельный туризм: определение понятия и синонимов // Географический вестник. 2021. №1 (56). – С. 109 – 118.
5. Стратегия развития туризма в Российской Федерации на период до 2035 года: распоряжение Правительства РФ от 20.09.2019 №2129-р // Правительство России [Электронный ресурс]. URL: <http://government.ru/docs/37906/> (дата обращения: 11.12.2023).
6. Феоктистов Д. Свежий взгляд: правительство поменяло ориентиры в развитии туризма // Национальные проекты России [Электронный ресурс]. URL: <https://национальныепроекты.рф/news/svezhiy-vzglyad-pravitelstvo-pomenyalo-orientiry-v-razvitii-turizma> (дата обращения: 12.12.2023).
7. Божко, В.Е. Разработка и экономическое обоснование проекта создания мобильного приложения туристской фирмы (на примере ООО «Хэппи тур»): дипломный проект – Белгород, 2016. [Электронный ресурс]. URL: <https://nauchkor.ru/uploads/documents/5a402efc7966e104c6a3e5d1.pdf> (дата обращения: 13.12.2023).
8. Результаты региональных конкурсов по программе «УМНИК» // Фонд содействия инновациям [Электронный ресурс]. URL:

<https://fasie.ru/press/fund/umnik-region-results-2023/> (дата обращения: 15.12.2023).

9. Грабова, А. По Пензе минувших эпох // Деловой – 2023, – №15 (240), С. 10 – 11.

10. Преакселератор УМНИК // Дотком Энтерпрайз [Электронный ресурс]. URL: <https://dotcomgroup.ru/akselerator-umnik> (дата обращения: 15.12.2023).

11. Акселерационная программа Пензенского государственного университета // Институт международного сотрудничества [Электронный ресурс]. URL: https://ims.pnzgu.ru/international_cooperation_ims/projects_ims/accelator (дата обращения: 16.12.2023)

12. ГОСТ Р 7.0.60-2020. Система стандартов по информации, библиотечному и издательскому делу издания. Основные виды термины и определения. – Введ. 2020-12-01. – М.: Стандартинформ, 2020. – III, 42 с.

13. Стахова, Л.В. Мобильный туристский путеводитель как актуальный инструмент продвижения в туризме // Сервис в России и за рубежом. 2022. №2 (99). – С. 160 – 169.

14. Курылев И. Геймификация: новый уровень вашего маркетинга. Как повторить опыт «Сбера», «Якитории», Burger King и MIXIT // Mindbox [Электронный ресурс]. URL: <https://mindbox.ru/journal/experts/gejmifikaciya/> (дата обращения: 23.12.2023).

15. Копылова, А. А. Концепция экскурсионного квеста в формате мобильного приложения / А. А. Копылова, А. Ю. Кардаш, Д. Д. Строкова, О. Д. Шмулей // Туризм: технологии & тренды : материалы IV студенческой научно-практической конференции, Екатеринбург, 31 марта 2018 г. – Екатеринбург : Издательство Уральского университета, 2018. – С. 51 – 53.

16. Воробев А. AR на вебинарах – эффектный способ разбудить вашу аудиторию // Argument [Электронный ресурс]. URL: <https://argument.digital/blog/ar-na-vebinarah-effektnyy-sposob-razbudit-vashu-auditoriyu/> (дата обращения: 24.12.2023).

17. Мирзахмедов Р. AR увеличивает конверсию в продажу на 94% – глобальное исследование Snapchat за 2021 год // vc.ru [Электронный ресурс]. URL: <https://vc.ru/u/40070-rustam-mirzakhmedov/290305-ar-uvelichivaet-konversiyu-v-prodazhu-na-94-globalnoe-issledovanie-snapchat-za-2021-god> (дата обращения: 25.12.2023).

18. Агапова, В. А. Выявление функциональных особенностей мобильных приложений квест-экскурсий / В. А. Агапова, А. С. Кудашов, Е. А. Дзюба // Информационные технологии в науке и образовании. Проблемы и перспективы : сборник статей по материалам X Всероссийской научно-практической конференции, Пенза, 15 марта 2023 года. – Пенза: Пензенский государственный университет, 2023. – С. 200 – 204.

19. Лиманова, Н.И. Анализ эффективности клиент-серверной архитектуры / Н. И. Лиманова, И. А. Селезнев // Бюллетень науки и практики. – 2022. – Т. 8, № 7. – С. 392 – 396. – DOI 10.33619/2414-2948/80/37. – EDN AONLOB. [Электронный ресурс]. URL: <https://www.elibrary.ru/item.asp?edn=aonlob> (дата обращения: 10.04.2024).

20. Прунов А. Архитектура «клиент-сервер»: основы и принципы // Айтиспектр [Электронный ресурс]. URL: <https://itspectr.ru/arhitektura-klient-server-osnovy-i-princzipy/> (дата обращения: 08.01.2024).

21. Ричардс, М. Фундаментальный подход к программной архитектуре: паттерны, свойства, проверенные методы / Ричардс М., Форд Н. – СПб.: Питер, 2023. – 448 с.: ил. – (Серия «Для профессионалов»)

22. Рокатанский М. Инженерная надежность и отказоустойчивость распределенной системы // Хабр [Электронный ресурс]. URL: <https://habr.com/ru/companies/otus/articles/561630/> (дата обращения: 09.01.2024).

23. Сомасегар, С. Руководство Microsoft по проектированию архитектуры приложений / Сомасегар С., Гатри С., Хилл Д. – 2-е изд-е, 2009. – Корпорация Майкрософт, 529 с.

24. Бешелев, С. Д. Математико-статистические методы экспертных оценок / Бешелев С. Д., Гурвич Ф. Г. – 2-е изд., перераб. и доп. – М.: Статистика, 1980. – 263 с., ил. – (Матем. статистика для экономистов).

25. Sanctis V. Building Web APIs with ASP.NET Core. – NY: Manning Publications Co, 2023. – 821 с.

26. Эванс, Э. Предметно-ориентированное проектирование (DDD): структуризация сложных программных систем. : Пер. с англ. – М.: 000 «И.Д. Вильямс», 2011. – 448 с. : ил. – Парал. тит. англ.

27. Fowler, M. Inversion of Control Containers and the Dependency Injection pattern // martinFowler.com [Электронный ресурс]. URL: <https://www.martinfowler.com/articles/injection.html> (дата обращения: 11.01.2024).

28. Dependency Injection: Принцип. Паттерн. Контейнер // Calabonga SOFT [Электронный ресурс]. URL: <https://www.calabonga.net/blog/post/dependency-injection-principle-pattern-container> (дата обращения: 11.01.2024).

29. Семакин, И. Г. Информатика. Базовый уровень учебник для 11 класса/ И. Г. Семакин, Е. К. Хеннер, Т. Ю. Шеина. – 3-е изд. – М. : БИНОМ. Лаборатория знаний, 2014. – 224 с. : ил.

30. Difference Between MVC, MVP and MVVM Architecture Pattern in Android // GeeksforGeeks [Электронный ресурс]. URL: <https://www.geeksforgeeks.org/difference-between-mvc-mvp-and-mvvm-architecture-pattern-in-android/> (дата обращения: 17.01.2024).

31. Krzysztof K. MVC vs MVP vs MVVM: Common Android architecture patterns explained // miquido [Электронный ресурс]. URL: <https://www.miquido.com/blog/mvc-mvp-mvvm/> (дата обращения: 17.01.2024)

32. Мартин, Р. Чистая архитектура. Искусство разработки программного обеспечения. – СПб.: Питер, 2021. – 352 с.: ил. – (Серия «Библиотека программиста»).

33. Шульгин, Е.М. Архитектурный паттерн model-view-presenter в разработке под мобильную платформу Android // Форум молодых ученых. 2018.

№6 – 3 (22). – С. 771 – 784. [Электронный ресурс]. URL: <https://cyberleninka.ru/article/n/arhitekturnyy-pattern-model-view-presenter-v-razrabotke-pod-mobilnuyu-platformu-android> (дата обращения: 17.01.2024).

34. Курганова, А.Г. Процесс выбора архитектуры для мобильного приложения // StudNet. 2021. №6. – С. 386 – 397.

35. Ликсин, С.С. Сравнительный анализ шаблонов проектирования приложений с пользовательским интерфейсом / Ликсин С.С., Лукошкин П.А., Шибанов С.В. // Вестник ПензГУ. 2021. №4 (36). – С. 79 – 85.

36. What is MonoBehaviour? // codinBlack [Электронный ресурс]. URL: <https://www.codinblack.com/what-is-monobehaviour/> (дата обращения: 18.01.2024).

37. Code-First vs Model-First vs Database-First: Pros and Cons // Ryadel [Электронный ресурс]. URL: <https://www.ryadel.com/en/code-first-model-first-database-first-vs-comparison-orm-asp-net-core-entity-framework-ef-data/> (дата обращения: 15.01.2024).

38. Lerman, J. Programming Entity Framework: Code First – Gravenstein Highway North, Sebastopol, CA: O'Reilly Media, Inc. / J. Lerman, R. Miller, 2011. – 177 с.

39. Киорогло Р. Подходы code first, database first и model first в создании базы данных используя entity framework // IBN Instrument Bibliometric National [Электронный ресурс]. URL: https://ibn.idsi.md/sites/default/files/imag_file/486-490_3.pdf (дата обращения: 16.01.2024).

40. Должностная инструкция проект-менеджера // Медиа «Просто работа» [Электронный ресурс]. URL: https://prosto.rabota.ru/post/dolzhnostnaja_instruktsija_proekt_menedzhera/ (дата обращения: 19.01.2024).

41. Проектировщик // Radar4Site [Электронный ресурс]. URL: <https://radar4site.ru/blog/2842-proektirovschik.html> (дата обращения: 20.01.2024).

42. ASP.NET Core for Backend Developers // CodingDrills [Электронный ресурс]. URL: <https://www.codingdrills.com/tutorial/backend-development-tutorial/asp-net-core-backend> (дата обращения: 05.01.2024).

43. JsonUtility // Unity – Manual [Электронный ресурс]. URL: <https://docs.unity3d.com/ScriptReference/JsonUtility.html> (дата обращения: 04.04.2024).

44. Bas A. Vuforia: немного магии в нашей реальности // Хабр [Электронный ресурс]. URL: <https://habr.com/ru/articles/198862/> (дата обращения: 05.04.2024).

45. Oliver S. UnityFigmaBridge // GitHub [Электронный ресурс]. URL: <https://github.com/simonoliver/UnityFigmaBridge> (дата обращения: 19.03.2024).

46. Префабы (Prefabs) // Unity – Manual [Электронный ресурс]. URL: <https://docs.unity3d.com/ru/530/Manual/Prefabs.html> (дата обращения: 25.03.2024).

47. Бражников, С.А. Методические указания к выполнению лабораторных работ по дисциплине «Технология программирования» / Сост. С.А. Бражников, М.А. Лебедева, С.Н. Фисун. – Севастополь: Изд-во СевНТУ, 2006. – 38 с.

48. Значение метрик кода // Microsoft Learn [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/visualstudio/code-quality/code-metrics-values?view=vs-2022> (дата обращения: 30.04.2024).

Приложение А
(справочное)
Глоссарий

Android – операционная система для мобильных устройств, разработанная компанией Google.

API (англ. Application Programming Interface – программный интерфейс приложения) – это набор способов и правил, по которым различные программы общаются между собой и обмениваются данными.

CORS (Cross-Origin Resource Sharing) – это механизм, позволяющий web-странице запрашивать данные с другого домена, отличного от домена исходной страницы.

Docker – платформа контейнеризации приложений.

JSON Web Token (JWT) – это компактный, URL-безопасный способ представления токенов доступа, подлежащих передаче между двумя сторонами (RFC 7519).

Plain Old CLR Object (POCO) – простой объект, созданный в среде Common Language Runtime (CLR) в .NET Framework и свободный от наследования или атрибутов.

UI-дизайн (user interface design – дизайн пользовательского интерфейса) – это графическая часть интерфейса. Она включает в себя все элементы, с которыми взаимодействуют пользователи: кнопки, поля ввода, изображения, тексты.

UML (Unified Modeling Language) – это унифицированный язык моделирования, система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования.

Unity – кроссплатформенная среда разработки компьютерных игр.

UX-дизайн (user experience design – дизайн пользовательского опыта) – это функциональная часть интерфейса. В нее входит все, что определяет взаимодействие пользователя с сайтом или приложением.

Дополненная реальность (AR) – технология наложения цифровых объектов на предметы реального мира. Например, когда пользователь наводит камеру смартфона на реальный фасад здания и видит на нем анимацию с его историей.

Объектно-ориентированное программирование (ООП) – парадигма программирования, предполагающая составление кода из взаимодействующих друг с другом объектов, обладающих набор данных и/или методов.

Префаб – наборы игровых объектов и логики, которые используются в нескольких местах приложения Unity и изагружаются на сцену игровым движком.

Скрипт – класс-наследник от MonoBehaviour, который навешивается на внутри игровые объекты движка Unity и описывает логику их поведения.

Сцена – рабочее пространство в Unity, на котором располагаются объекты.

Фреймворк – программная платформа, состоящая из набора готовых решения и предназначенная для упрощения и ускорения разработки программных продуктов.

Шаблон проектирования (паттерн) – повторяемая архитектурная конструкция, представляющая собой продуманное решение определенного типа задач проектирования программного обеспечения.

Приложение Б
(рекомендуемое)
Результаты бизнес-проекта



**Открытые
Решения_**
разработка программного
обеспечения

> + 7 (8412) 20-06-04
info@osinit.com
www.osinit.com

> ООО «Открытые решения»
Россия, 440000 г. Пенза,
ул. Московская, д. 29
ОГРН 1095835001712
ИНН 5835081619

Исх. № 894 от «31» октября 2022 г.

**В конкурсную комиссию
программы «УМНИК»
фонда содействия развития
малых форм предприятий
в научно-технической сфере**

Настоящим письмом компания ООО «Открытые Решения» в лице Исполнительного директора Бочкарева Игоря Вячеславовича выражает поддержку проекту Александра Кудашова «Разработка системы для исторических квест-экскурсий с применением технологии дополненной реальности».

Проект представляет собой мобильное Android – приложение для городских исторических квест экскурсий, предназначенное для изучения туристами культурно-исторических достопримечательностей и географии города с повышенным уровнем вовлеченности в процесс. В ходе реализации планируется разработка backend- и frontend-части приложения, формирования списка культурно-исторических мест города, а также апробация квестов, занесение исторических справок в приложение. Компания поддерживает участие Проекта в грантовом конкурсе программы «УМНИК» и рекомендует проект к рассмотрению конкурсной комиссией.

Компания готова оказать консультативную и информационную поддержку проекту, а также экспертную поддержку в лице своих сотрудников. Компания готова обеспечить освещение проекта на своих информационных ресурсах: сайт платформы подготовки IT-специалистов – software-academy.ru, группы в социальных сетях – vk.com/osinitcom; t.me/open_solutions.

Исполнительный директор



И.В. Бочкарев

Рисунок Б.1 – Письмо поддержки ООО «Открытые решения»



МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Пензенский государственный университет»
(ФГБОУ ВО «ПГУ»)

Красная ул., д.40, г. Пенза, Россия, 440026. Тел/факс: (841-2) 66-62-85
 E-mail: cnit@pnzgu.ru; <http://www.pnzgu.ru>
 ОКПО 02069042, ОГРН 1025801440620, ИНН5837003736 КПП 583701001

Федеральное государственное бюджетное
 учреждение «Фонд содействия развитию малых
 форм предприятий в научно-технической сфере»
 (Фонд содействия инновациям)
 Генеральному директору
 Полякову Сергею Геннадьевичу

Уважаемый Сергей Геннадьевич!

Кафедра «Маркетинг, коммерция и сфера обслуживания» Института экономики и управления ФГБОУ ВО «Пензенский государственный университет» выражает заинтересованность и готовность участия в разработке проекта «Разработка системы для исторических квест-экскурсий с применением технологии дополненной реальности» в рамках программы «УМНИК» (далее Проект), а также его последующей коммерциализации.

Проект представляет собой мобильное Android-приложение для городских исторических квест-экскурсий, предназначенное для популяризации краеведения и формирования желания у туристов и местных жителей в игровой форме знакомиться с культурно-историческими достопримечательностями российских регионов. Пилотная версия мобильного приложения будет разработана для города Пенза. В перспективе возможно тиражирование и на другие населенные пункты Пензенской области и других субъектов РФ. В ходе реализации планируется разработка backend- и frontend- части приложения, формирования списка культурно-исторических мест города, а также квестов, занесение исторических справок в приложение.

Кафедра «Маркетинг, коммерция и сфера обслуживания» Института экономики и управления ФГБОУ ВО «Пензенский государственный университет» поддерживает участие Проекта в грантовом конкурсе программы «УМНИК» и рекомендует проект к рассмотрению конкурсной комиссией.

Кафедра готова оказать экспертную поддержку в лице учёных кафедры, занимающихся исследованием сферы туризма как социально-экономической системы; поддержку в лице студентов направления подготовки 43.03.02 «Туризм» (уровень бакалавриата) - в разработке содержательной части квест-экскурсии.

Кроме того, кафедра готова обеспечить информационную поддержку за счет освещения проекта на своих информационных ресурсах: сайт - https://dep_mkiso.pnzgu.ru/, группа в социальной сети ВКонтакте - <https://vk.com/mkiso>.

/ Директор «ИЭиУ»

В. М. Володин

Заведующий кафедрой «МКиСО»

Л. А. Гамидуллаева



Рисунок Б.2 – Письмо поддержки кафедры «Маркетинг, коммерция и сфера обслуживания» Института экономики и управления



**МИНИСТЕРСТВО
КУЛЬТУРЫ И ТУРИЗМА
ПЕНЗЕНСКОЙ ОБЛАСТИ**

ул. Красная, д. 71, г. Пенза, 440026
тел. (8412) 56-57-22, факс (8412) 56-63-09
e-mail: uka_po@mail.ru
<http://minkult.pnzreg.ru>

Генеральному директору
ФГБУ «Фонд содействия развитию
малых форм предприятий в научно-
технической сфере»
Полякову С.Г.
info@fasie.ru

от 14.11.2022 № 15-12/4940
На № _____ от _____

Уважаемый Сергей Геннадьевич!

Министерство культуры и туризма Пензенской области (далее – Министерство) выражает заинтересованность в реализации и продвижении проекта «Разработка системы для исторических квест-экскурсий с применением технологии дополненной реальности» в рамках программы «УМНИК».

Проект представляет собой мобильное Android-приложение для городских исторических квест-экскурсий, предназначенное для изучения туристами культурно-исторических достопримечательностей и географии города с повышенным уровнем вовлеченности в процесс. В ходе реализации планируется разработка backend- и frontend- части приложения, формирования списка культурно-исторических мест города, а также квестов, занесение исторических справок в приложение.

Общая актуальность разрабатываемого продукта обоснована следующими фактами: 2022-ой год – год культурного наследия народов России; Правительство России ведет активную политику по поддержке и развитию отечественного туризма; по данным ВЦИОМ 96% россиян считают важным знание истории своей страны и своего региона; турпоток по городам России по прогнозам на конец 2022 года составит 61 млн человек (на начало августа 2022 г. он уже составил 34 млн чел.). Необходимость внедрения технологии дополненной реальности в проект объясняется увеличением благодаря ей уровня вовлеченности пользователей в экскурсионный процесс и повышением заинтересованности.

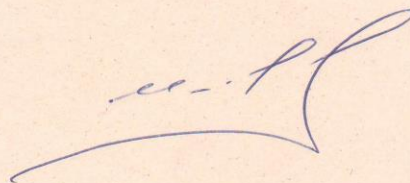
Министерство поддерживает участие Проекта в грантовом конкурсе

Рисунок Б.3 – Письмо поддержки Министерства культуры и туризма
Пензенской области (часть 1)

программы «УМНИК» и рекомендует проект к рассмотрению конкурсной комиссией.

Кроме того, Министерство готово оказать консультативную и информационную поддержку проекту, а также обеспечить освещение проекта на официальном сайте Министерства и в социальных сетях.

Врио Министра



И.Н. Мавлюдов

Барышева Екатерина Михайловна
8(8412) 56-07-69

Рисунок Б.4 – Письмо поддержки Министерства культуры и туризма
Пензенской области (часть 2)



Рисунок Б.6 – Сертификат о прохождении акселерационной программы
Пензенского государственного универси

Приложение В
(рекомендуемое)
Исходный код программного обеспечения

Листинг В.1 – docker-compose.yml

```

version: '3.4'

services:
  questie_backend:
    image: ${DOCKER_REGISTRY-}questiebackend
    build:
      context: .
      dockerfile: questie_backend/Dockerfile
    environment:
      - ASPNETCORE_ENVIRONMENT=Development
      - ASPNETCORE_HTTP_PORTS=8080
      - ASPNETCORE_HTTPS_PORTS=8081
    ports:
      - "5002:8080"
      - "5003:8081"
    volumes:
      -
      ${APPDATA}/Microsoft/UserSecrets:/home/app/.microsoft/usersecrets:ro
      - ${APPDATA}/ASP.NET/Https:/home/app/.aspnet/https:ro
    networks:
      - questienet
    depends_on:
      - db
      - migrator

  questie_backend_users:
    image: ${DOCKER_REGISTRY-}questiebackendusers
    build:
      context: .
      dockerfile: questie_backend_users/Dockerfile
    environment:
      - ASPNETCORE_ENVIRONMENT=Development
      - ASPNETCORE_HTTP_PORTS=8080
      - ASPNETCORE_HTTPS_PORTS=8081
    ports:
      - "5004:8080"
      - "5005:8081"
    volumes:
      -
      ${APPDATA}/Microsoft/UserSecrets:/home/app/.microsoft/usersecrets:ro
      - ${APPDATA}/ASP.NET/Https:/home/app/.aspnet/https:ro
      - questie-upload-images:/app/Images
    networks:
      - questienet
    depends_on:
      - db
      - migrator

  questie_backend_authorization:
    image: ${DOCKER_REGISTRY-}questiebackendauthorization
    build:
      context: .
      dockerfile: questie_backend_authorization/Dockerfile
    environment:
      - ASPNETCORE_ENVIRONMENT=Development

```

```

    - ASPNETCORE_HTTP_PORTS=8080
    - ASPNETCORE_HTTPS_PORTS=8081
  ports:
    - "5006:8080"
    - "5007:8081"
  volumes:
    -
  ${APPDATA}/Microsoft/UserSecrets:/home/app/.microsoft/usersecrets:ro
    - ${APPDATA}/ASP.NET/Https:/home/app/.aspnet/https:ro
  networks:
    - questienet
  depends_on:
    - db
    - migrator

migrator:
  image: ${DOCKER_REGISTRY-}migrator
  build:
    context: .
    dockerfile: migrator/Dockerfile
  environment:
    - ASPNETCORE_ENVIRONMENT=Development
    - ASPNETCORE_HTTP_PORTS=8080
    - ASPNETCORE_HTTPS_PORTS=8081
  ports:
    - "5008:8080"
    - "5009:8081"
  volumes:
    -
  ${APPDATA}/Microsoft/UserSecrets:/home/app/.microsoft/usersecrets:ro
    - ${APPDATA}/ASP.NET/Https:/home/app/.aspnet/https:ro
  networks:
    - questienet
  depends_on:
    - db

db:
  container_name: postgresql-db
  image: postgres:15.3-alpine3.18
  environment:
    - POSTGRES_DB=questie_db
    - POSTGRES_USER=questie_user
    - POSTGRES_PASSWORD=admin
  ports:
    - "5000:5432"
  volumes:
    - postgres-data:/var/lib/postgresql/data
  networks:
    - questienet

pgadmin:
  container_name: pgadmin
  image: dpage/pgadmin4:8.0
  environment:
    PGADMIN_DEFAULT_EMAIL: questie@yandex.ru
    PGADMIN_DEFAULT_PASSWORD: admin
    PGADMIN_LISTEN_PORT: 80

```

```

    VIRTUAL_HOST: pgadmin.local.net
    VIRTUAL_PORT: 5001
  ports:
    - 5001:80
  volumes:
    - pgadmin:/var/lib/pgadmin
  networks:
    - questienet
  depends_on:
    - db

networks:
  questienet:
    name: questie_network
    driver: bridge

volumes:
  postgres-data:
  pgadmin:
  questie-upload-images:

```

Листинг В.2 – Program.cs программы migrator

```

using Microsoft.EntityFrameworkCore;
using migrator.Data;

namespace migrator
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var builder = WebApplication.CreateBuilder(args);

            // Add services to the container.
            var configBuilder = new ConfigurationBuilder();
            // установка пути к текущему каталогу

            configBuilder.SetBasePath(Directory.GetCurrentDirectory());
            // получаем конфигурацию из файла appsettings.json
            configBuilder.AddJsonFile("appsettings.json");

            // создаем конфигурацию
            var config = configBuilder.Build();
            // получаем строку подключения
            string connectionString =
            config.GetConnectionString("DefaultConnection");

            // Add services to the container.
            builder.Services.AddDbContext<QuestsContext>(options =>
            options.UseNpgsql(connectionString));

            var app = builder.Build();

            app.MigrateDatabase();
            Environment.Exit(0);
        }
    }
}

```

```

    }
}

```

Листинг В.3 – MigrateDatabase.cs программы migrator

```

using Microsoft.EntityFrameworkCore;

namespace migrator.Data
{
    public static class MigrationManager
    {
        public static IHost MigrateDatabase(this IHost host)
        {
            using (var scope = host.Services.CreateScope())
            {
                using (var appContext =
scope.ServiceProvider.GetRequiredService<QuestsContext>())
                {
                    try
                    {
                        appContext.Database.Migrate();
                    }
                    catch (Exception ex) { }
                }
            }

            return host;
        }
    }
}

```

Листинг В.4 – Program.cs программы questie_backend

```

using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.FileProviders;
using Microsoft.IdentityModel.Tokens;
using Microsoft.OpenApi.Models;
using questie_backend.Data;
using questie_backend.Middleware;
using questie_backend.Repositories.Implimentations;
using questie_backend.Repositories.Interfaces;
using questie_backend.Services.Implimentations;
using questie_backend.Services.Interfaces;
using System.Text;

internal class Program
{
    private static void Main(string[] args)
    {
        var builder = WebApplication.CreateBuilder(args);
        ConfigurationManager configuration = builder.Configuration;

        var configBuilder = new ConfigurationBuilder();
        // установка пути к текущему каталогу
        configBuilder.SetBasePath(Directory.GetCurrentDirectory());
        // получаем конфигурацию из файла appsettings.json

```

```

        configBuilder.AddJsonFile("appsettings.json");
        // создаем конфигурацию
        var config = configBuilder.Build();
        // получаем строку подключения
        string connectionString =
config.GetConnectionString("DefaultConnection");

        // Add services to the container.
        builder.Services.AddDbContext<QuestsContext>(options =>
options.UseNpgsql(connectionString), ServiceLifetime.Transient,
ServiceLifetime.Transient);

        builder.Services.AddControllers();
        // Learn more about configuring Swagger/OpenAPI at
https://aka.ms/aspnetcore/swashbuckle
        builder.Services.AddEndpointsApiExplorer();
        builder.Services.AddSwaggerGen(opt =>
        {
            // To Enable authorization using Swagger (JWT)
            opt.AddSecurityDefinition("Bearer", new
OpenApiSecurityScheme()
            {
                Name = "Authorization",
                Type = SecuritySchemeType.ApiKey,
                Scheme = "Bearer",
                BearerFormat = "JWT",
                In = ParameterLocation.Header,
                Description = "Используется JWT авторизация.
\r\n\r\n Введите 'Bearer' [пробел] и свой токен в текстовом поле
ниже.\r\n\r\nНапример: \"Bearer 12345abcdef\"",
            });
            opt.AddSecurityRequirement(new
OpenApiSecurityRequirement
            {
                {
                    new OpenApiSecurityScheme
                    {
                        Reference = new OpenApiReference
                        {
                            Type =
ReferenceType.SecurityScheme,
                            Id = "Bearer"
                        },
                        Array.Empty<string>()
                    },
                },
            });
        });

        builder.Services.AddAuthorization();
        // Adding Authentication
        builder.Services.AddAuthentication(options =>
        {
            options.DefaultAuthenticateScheme =
JwtBearerDefaults.AuthenticationScheme;
            options.DefaultScheme =
JwtBearerDefaults.AuthenticationScheme;

```

```

    })
    // Adding Jwt Bearer
    .AddJwtBearer(options =>
    {
        options.SaveToken = true;
        options.RequireHttpsMetadata = false;
        options.TokenValidationParameters = new
TokenValidationParameters()
        {
            ValidateAudience = true,
            ValidAudience = configuration["Jwt:Audience"],
            ValidateIssuer = true,
            ValidIssuer = configuration["Jwt:Issuer"],
            ValidateIssuerSigningKey = true,

            IssuerSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(configuration["Jwt:Key"]))
        };

    });

    builder.Services.AddDistributedMemoryCache();
    builder.Services.AddSession();

    builder.Services.AddHttpContextAccessor();

    builder.Services.AddScoped<IActiveQuestRepository,
ActiveQuestRepository>();
    builder.Services.AddScoped<IActiveQuestService,
ActiveQuestService>();
    builder.Services.AddScoped<ICityService, CityService>();
    builder.Services.AddScoped<ICompletedQuestRepository,
CompletedQuestRepository>();
    builder.Services.AddScoped<ICompletedQuestService,
CompletedQuestService>();
    builder.Services.AddScoped<IQuestRepository,
QuestRepository>();
    builder.Services.AddScoped<IQuestService, QuestService>();
    builder.Services.AddScoped<IQuestionsRepository,
QuestionsRepository>();
    builder.Services.AddScoped<IQuestionsService,
QuestionsService>();
    builder.Services.AddScoped<INotesRepository,
NotesRepository>();
    builder.Services.AddScoped<INotesService, NotesService>();
    builder.Services.AddScoped<IStatisticRepository,
StatisticRepository>();
    builder.Services.AddScoped<IStatisticService,
StatisticService>();
    builder.Services.AddScoped<ITargetsRepository,
TargetsRepository>();
    builder.Services.AddScoped<ITargetsService, TargetsService>();
    builder.Services.AddScoped<IUsersService, UsersService>();

    var app = builder.Build();
    var env = app.Environment;
    app.UseCors(builder => builder

```

```

        .AllowAnyHeader()
        .AllowAnyMethod()
        .AllowAnyOrigin()
    );

    // Configure the HTTP request pipeline.
    if (app.Environment.IsDevelopment())
    {
        app.UseSwagger();
        app.UseSwaggerUI();
    }

    List<string> staticFilePaths = new List<string>()
{
    "Images/Notes",
    "Images/Quests",
    "Images/Questions",
    "Images/Targets"
};

    foreach (var path in staticFilePaths)
    {
        app.UseStaticFiles(new StaticFileOptions
        {
            FileProvider = new
PhysicalFileProvider(Path.Combine(env.ContentRootPath, path)),
            RequestPath = $"/{path}"
        });
    }

    app.UseSession();

    app.UseAuthentication();
    app.UseAuthorization();

    app.UseMiddleware<JWTMiddleware>();

    app.UseAuthorization();

    app.MapControllers();

    app.Run();
}
}

```

Листинг В.5 – QuestsContext.cs программы questie_backend

```

using Entits.Models;
using Microsoft.EntityFrameworkCore;

namespace questie_backend.Data
{
    public class QuestsContext : DbContext
    {
        public DbSet<Quest> QuestsDbSet { get; set; }
        public DbSet<ActiveQuest> ActiveQuestsDbSet { get; set; }
    }
}

```

```

        public DbSet<CompletedQuest> CompletedQuestsDbSet { get; set; }
    }

    public DbSet<Question> QuestionsDbSet { get; set; }
    public DbSet<Target> TargetsDbSet { get; set; }
    public DbSet<Note> NotesDbSet { get; set; }
    public QuestsContext(DbContextOptions<QuestsContext> options)
: base(options) { }
    }
}

```

Листинг В.6 – QuestionsController.cs программы questie_backend

```

using entitys.DTO;
using entitys.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.IdentityModel.Tokens;
using questie_backend.Services.Interfaces;

namespace questie_backend.Controllers
{
    [ApiController]
    [Produces("application/json")]
    [Route("api/[controller]")]
    public class QuestionsController : ControllerBase
    {
        private IQuestionsService _questionsService;
        public QuestionsController(IQuestionsService questionsService)
        {
            _questionsService = questionsService;
        }

        [HttpGet("questions")]
        public async Task<List<QuestionOutDto>> GetAllQuestions()
        {
            return await _questionsService.GetAllQuestionsAsync();
        }

        [HttpGet("from-quest/{questId}")]
        public async Task<List<QuestionOutDto>>
        GetQuestionsFromQuest(Guid questId)
        {
            return await
            _questionsService.GetQuestionsFromQuestAsync(questId);
        }

        [HttpGet("question/{questionId}")]
        public async Task<ActionResult<QuestionOutDto>>
        GetQuestionById(Guid questionId)
        {
            if (questionId == Guid.Empty)
            {
                return BadRequest("Идентификатор задания не может
                быть пустым");
            }

            QuestionOutDto? question = await
            _questionsService.GetQuestionByIdAsync(questionId);
        }
    }
}

```

```

        return question != null ? question : NotFound();
    }

    [HttpGet("hint/{questionId}")]
    public async Task<ActionResult<HintDto>>
GetHintByQuestionId(Guid questionId)
    {
        if (questionId == Guid.Empty)
        {
            return BadRequest("Идентификатор задания не может
быть пустым");
        }

        HintDto? hint = await
_questionsService.GetHintByQuestionIdAsync(questionId);
        return hint != null ? hint : NotFound();
    }

    [HttpPost("create")]
    public async Task<ActionResult> CreateQuestion(QuistionInDto
questionDto)
    {
        if (questionDto == null)
        {
            return BadRequest("Невозможно создать пустое
задание. Полученный объект задания равен NULL");
        }

        Question? question = await
_questionsService.CreateQuestionAsync(questionDto);

        return question != null
            ? CreatedAtAction("GetQuestionById", new {
questionId = question.Id }, question)
            : BadRequest("Не удалось создать задание из-за
некорректных данных");
    }

    [HttpPost("exist")]
    public async Task<ActionResult> CheckExistQuestionById(Guid
questionId)
    {
        bool exist = await
_questionsService.CheckQuestionAsync(questionId);
        return exist ? Ok(new { Exist = "true" })
            : Ok(new { Exist = "false" });
    }

    [HttpDelete("{questionId}")]
    public async Task<ActionResult> DeleteQuestion(Guid
questionId)
    {
        if (questionId == Guid.Empty)
        {
            return BadRequest("Идентификатор задания не может
быть пустым");
        }
    }

```

```

        return await
        _questionsService.DeleteQuestionAsync(questionId) ? NoContent() :
        NotFound();
    }

    [HttpPut("question")]
    public async Task<IActionResult> UpdateQuestion(Guid id,
        QuestionUpdatedDto questionUpdatedDto)
    {
        if (id == Guid.Empty)
        {
            return BadRequest("Идентификатор задания не может
        БЫТЬ ПУСТЫМ");
        }
        if (questionUpdatedDto is null)
        {
            return BadRequest("Обновленное задание не может
        БЫТЬ ПУСТЫМ");
        }

        return await _questionsService.UpdateQuestionAsync(id,
        questionUpdatedDto)
            ? NoContent()
            : NotFound();
    }

    [HttpPut("next")]
    public async Task<IActionResult> AddNextQuestionId(Guid id,
        Guid nextQuestionId)
    {
        if (id == Guid.Empty)
        {
            return BadRequest("Идентификатор задания не может
        БЫТЬ ПУСТЫМ");
        }
        if (nextQuestionId == Guid.Empty)
        {
            return BadRequest("Идентификатор следующего задания
        не может БЫТЬ ПУСТЫМ");
        }

        return await
        _questionsService.UpdateNextQuestionIdAsync(id, nextQuestionId)
            ? NoContent()
            : NotFound();
    }

    [HttpPost("check")]
    public async Task<ActionResult<CheckedAnswerDto?>>
        CheckQuestion(AnswerDto answer)
    {
        if (answer is null || answer.Answers.IsNullOrEmpty())
        {
            return BadRequest("Ответ не может БЫТЬ ПУСТЫМ");
        }
        if (answer.QuestionId == Guid.Empty)
    
```

```

        {
            return BadRequest("Идентификатор вопроса не может
быть пустым");
        }

        CheckedAnswerDto? checkedAnswer = await
_questionsService.GetCheckedAsync(answer);
        return checkedAnswer != null
            ? Ok(checkedException)
            : BadRequest("Не удалось проверить задание из-за
некорректных данных");
    }
}

```

Листинг В.7 – IQuestionsService.cs программы questie_backend

```

using entitys.DTO;
using entitys.Models;

namespace questie_backend.Services.Interfaces
{
    public interface IQuestionsService
    {
        Task<bool> CheckQuestionAsync(Guid questionId);
        Task<Question?> CreateQuestionAsync(QuistionInDto
questionDto);
        Task<bool> DeleteQuestionAsync(Guid questionId);
        Task<List<QuestionOutDto>> GetAllQuestionsAsync();
        Task<CheckedAnswerDto?> GetCheckedAsync(AnswerDto answer);
        Task<HintDto?> GetHintByQuestionIdAsync(Guid questionId);
        Task<QuestionOutDto?> GetQuestionByIdAsync(Guid questionId);
        Task<List<QuestionOutDto>> GetQuestionsFromQuestAsync(Guid
questId);
        Task<bool> UpdateNextQuestionIdAsync(Guid id, Guid
nextQuestionId);
        Task<bool> UpdateQuestionAsync(Guid id, QuestionUpdatedto
questionUpdatedto);
    }
}

```

Листинг В.8 – QuestionsService.cs программы questie_backend

```

using entitys.DTO;
using entitys.Enums;
using entitys.Models;
using questie_backend.Repositories.Interfaces;
using questie_backend.Services.Interfaces;
using System.Text.Json.Nodes;
using System.Text.Json;

namespace questie_backend.Services.Implimentations
{
    public class QuestionsService : IQuestionsService
    {
        private readonly IQuestionsRepository _questionsRepository;
        private readonly IActiveQuestService _activeQuestService;
        private readonly INotesService _notesService;
    }
}

```

```

        private readonly IQuestService _questService;
        private readonly IUsersService _userService;
        private readonly IWebHostEnvironment _webHostEnvironment;
        private readonly IHttpContextAccessor _httpContextAccessor;

        public QuestionsService(IQuestionsRepository
questionsRepository,
                                IActiveQuestService
activeQuestService,
                                INotesService notesService,
                                IQuestService questService,
                                IUsersService userService,
                                IWebHostEnvironment
webHostEnvironment,
                                IHttpContextAccessor
httpContextAccessor)
        {
            _questionsRepository = questionsRepository;
            _activeQuestService = activeQuestService;
            _notesService = notesService;
            _questService = questService;
            _userService = userService;
            _webHostEnvironment = webHostEnvironment;
            _httpContextAccessor = httpContextAccessor;
        }

        public async Task<bool> CheckQuestionAsync(Guid questionId)
        {
            return await
            _questionsRepository.GetQuestionByIdAsync(questionId) == null;
        }

        public async Task<bool> DeleteQuestionAsync(Guid questionId)
        {
            Question? question = await
            _questionsRepository.GetQuestionByIdAsync(questionId);
            if (question == null)
            {
                return false;
            }

            Guid? previousId = await
            _questionsRepository.GetQuestionIdWithNextIdAsync(questionId);

            if (previousId != null)
            {
                await UpdateNextQuestionIdAsync(previousId.Value,
question.NextId);
            }
            else
            {
                await
            _questService.UpdateFirstQuestionIdAsync(question.QuestId,
question.NextId);
            }

            if (question.NoteId is not null)

```

```

        {
            await
            _notesService.DeleteNoteAsync(question.NoteId.Value);
        }

        DeleteImage(question.ImageName);
        await _questionsRepository.DeleteAsync(question);
        await
        _questService.DecrementQuantityQuestionsAsync(question.QuestId);

        return true;
    }

    public async Task<List<QuestionOutDto>> GetAllQuestionsAsync()
    {
        List<Question> questions = await
        _questionsRepository.GetAllQuestionsAsync();
        List<QuestionOutDto> questionsOutDto = new
        List<QuestionOutDto>();

        foreach (var question in questions)
        {
            questionsOutDto.Add(new QuestionOutDto()
            {
                Id = question.Id,
                TextQuestion = question.TextQuestion,
                Description = question.Description,
                Rating = question.Rating,
                ImageName =
                String.Format("${_httpContextAccessor.HttpContext!.Request.Scheme}://" +
                "${_httpContextAccessor.HttpContext!.Request.Host}" +
                "${_httpContextAccessor.HttpContext!.Request.PathBase}" +
                $"/Images/Questions/{question.ImageName}"),
                Hint = question.Hint,
                Type = question.Type,
                Answers = question.Answers,
                CorrectAnswers = question.CorrectAnswers,
                AnsweredCounter = question.AnsweredCounter,
                AnsweredFirstCounter =
                question.AnsweredFirstCounter,
                NextId = question.NextId,
                QuestId = question.QuestId,
                TargetId = question.TargetId,
                NoteId = question.NoteId
            });
        }

        return questionsOutDto;
    }

    public async Task<List<QuestionOutDto>>
    GetQuestionsFromQuestAsync(Guid questId)
    {

```

```

        List<Question> questions = await
        _questionsRepository.GetQuestionsFromQuestAsync(questId);
        List<QuestionOutDto> questionsOutDto = new
        List<QuestionOutDto>();

        foreach (var question in questions)
        {
            questionsOutDto.Add(new QuestionOutDto()
            {
                Id = question.Id,
                TextQuestion = question.TextQuestion,
                Description = question.Description,
                Rating = question.Rating,
                ImageName =
                String.Format($"{_httpContextAccessor.HttpContext!.Request.Scheme}://" +
                $"{_httpContextAccessor.HttpContext!.Request.Host}" +
                $"{_httpContextAccessor.HttpContext!.Request.PathBase}" +
                $"/Images/Questions/{question.ImageName}"),
                Hint = question.Hint,
                Type = question.Type,
                Answers = question.Answers,
                CorrectAnswers = question.CorrectAnswers,
                AnsweredCounter = question.AnsweredCounter,
                AnsweredFirstCounter =
                question.AnsweredFirstCounter,
                NextId = question.NextId,
                QuestId = question.QuestId,
                TargetId = question.TargetId,
                NoteId = question.NoteId
            });
        }

        return questionsOutDto;
    }

    public async Task<HintDto?> GetHintByQuestionIdAsync(Guid
questionId)
    {
        string? hint = await
        _questionsRepository.GetHintByQuestionIdAsync(questionId);

        return hint != null ? new HintDto()
        {
            Id = questionId,
            Hint = hint
        } : null;
    }

    public async Task<QuestionOutDto?> GetQuestionByIdAsync(Guid
questionId)
    {
        Question? question = await
        _questionsRepository.GetQuestionByIdAsync(questionId);
        QuestionOutDto? questionOutDto = null;
    }

```

```

        if (question != null)
        {
            questionOutDto = new QuestionOutDto()
            {
                Id = question.Id,
                TextQuestion = question.TextQuestion,
                Description = question.Description,
                Rating = question.Rating,
                ImageName =
String.Format("${_httpContextAccessor.HttpContext!.Request.Scheme}://" +

                "${_httpContextAccessor.HttpContext!.Request.Host}" +

                "${_httpContextAccessor.HttpContext!.Request.PathBase}" +

                $"/Images/Questions/{question.ImageName}"),
                Hint = question.Hint,
                Type = question.Type,
                Answers = question.Answers,
                CorrectAnswers = question.CorrectAnswers,
                AnsweredCounter = question.AnsweredCounter,
                AnsweredFirstCounter =
question.AnsweredFirstCounter,
                NextId = question.NextId,
                QuestId = question.QuestId,
                TargetId = question.TargetId,
                NoteId = question.NoteId
            };
        }

        return questionOutDto;
    }

    public async Task<Question?> CreateQuestionAsync(QuistionInDto
questionDto)
    {
        if (questionDto.QuestId == Guid.Empty)
        {
            return null;
        }

        if (await
_questService.GetQuestBrifByIdAsync(questionDto.QuestId) is null)
        {
            return null;
        }

        Question question = new Question();
        question.TextQuestion = questionDto.TextQuestion;
        question.Description = questionDto.Description;
        question.Rating = questionDto.Rating;
        question.ImageName = questionDto.ImgFile != null ? await
SaveImage(questionDto.ImgFile) : string.Empty;
        question.Hint = questionDto.Hint;
        question.Type = questionDto.Type;
        question.Answers = questionDto.Answers;
    }

```

```

        question.CorrectAnswers = questionDto.CorrectAnswers;
        question.AnsweredCounter = 0;
        question.AnsweredFirstCounter = 0;
        question.QuestId = questionDto.QuestId;
        question.TargetId = questionDto.TargetId;
        question.NoteId = questionDto.NoteId;
        question.CreatedAt = DateTime.UtcNow;
        question.UpdatedAt = DateTime.UtcNow;

        await
        _questionsRepository.InsertQuestionAsync(question);

        await
        _questService.IncrementQuantityQuestionsAsync(question.QuestId);

        return question;
    }

    public async Task<bool> UpdateNextQuestionIdAsync(Guid id,
    Guid nextQuestionId)
    {
        Question? oldQuestion = await
        _questionsRepository.GetQuestionByIdAsync(id);
        if (oldQuestion == null || await
        _questionsRepository.GetQuestionByIdAsync(nextQuestionId) == null)
        {
            return false;
        }

        oldQuestion.NextId = nextQuestionId;
        oldQuestion.UpdatedAt = DateTime.UtcNow;

        await _questionsRepository.UpdateAsync(oldQuestion);

        return true;
    }

    public async Task<bool> UpdateQuestionAsync(Guid id,
    QuestionUpdateDto questionUpdateDto)
    {
        Question? oldQuestion = await
        _questionsRepository.GetQuestionByIdAsync(id);
        if (oldQuestion == null)
        {
            return false;
        }

        oldQuestion.TextQuestion =
        questionUpdateDto.TextQuestion != null ? questionUpdateDto.TextQuestion :
        oldQuestion.TextQuestion;
        oldQuestion.Description = questionUpdateDto.Description
        != null ? questionUpdateDto.Description : oldQuestion.Description;
        oldQuestion.Rating = questionUpdateDto.Rating != null ?
        (int)questionUpdateDto.Rating : oldQuestion.Rating;
        oldQuestion.TextQuestion =
        questionUpdateDto.TextQuestion != null ? questionUpdateDto.TextQuestion :
        oldQuestion.TextQuestion;
    }

```

```

        if (questionUpdatedDto.ImgFile != null)
        {
            DeleteImage(oldQuestion.ImageName);
            oldQuestion.ImageName = await
SaveImage(questionUpdatedDto.ImgFile);
        }

        oldQuestion.Hint = questionUpdatedDto.Hint != null ?
questionUpdatedDto.Hint : oldQuestion.Hint;
        oldQuestion.Type = questionUpdatedDto.Type != null ?
(QuestionType)questionUpdatedDto.Type : oldQuestion.Type;
        oldQuestion.Answers = questionUpdatedDto.Answers != null
? questionUpdatedDto.Answers : oldQuestion.Answers;
        oldQuestion.CorrectAnswers =
questionUpdatedDto.CorrectAnswers != null ?
questionUpdatedDto.CorrectAnswers : oldQuestion.CorrectAnswers;
        oldQuestion.TargetId = questionUpdatedDto.TargetId !=
null ? questionUpdatedDto.TargetId : oldQuestion.TargetId;
        oldQuestion.NoteId = questionUpdatedDto.NoteId != null ?
questionUpdatedDto.NoteId : oldQuestion.NoteId;
        oldQuestion.UpdatedAt = DateTime.UtcNow;

        await _questionsRepository.UpdateAsync(oldQuestion);

        return true;
    }

    private async Task<string> SaveImage(IFormFile imageFile)
    {
        string imageName = new
String(Path.GetFileNameWithoutExtension(imageFile.FileName).Take(10).ToAr
ray()).Replace(' ', '-');
        imageName += DateTime.Now.ToString("yymmssfff") +
Path.GetExtension(imageFile.FileName);
        var imagePath =
Path.Combine(_webHostEnvironment.ContentRootPath, "Images/Questions",
imageName);

        using (var fileStream = new FileStream(imagePath,
FileMode.Create))
        {
            await imageFile.CopyToAsync(fileStream);
        }

        return imageName;
    }

    private void DeleteImage(string imageName)
    {
        var imagePath =
Path.Combine(_webHostEnvironment.ContentRootPath, "Images/Questions",
imageName);

        if (File.Exists(imagePath))
        {
            File.Delete(imagePath);
        }
    }

```

```

    }
}

public async Task<CheckedAnswerDto?> GetCheckedAsync (AnswerDto
answer)
{
    Question? question = await
    _questionsRepository.GetQuestionByIdAsync (answer.QuestionId);
    if (question is null)
    {
        return null;
    }

    string? user =
    _httpContextAccessor.HttpContext!.Session.GetString("User");
    if (user is null)
    {
        return null;
    }

    var jsonData =
    JsonSerializer.Deserialize<JsonNode>(user!)!;
    if (jsonData is null)
    {
        return null;
    }

    Guid userId = (Guid)jsonData["Id"]!;

    if (!await _userService.CheckUserExistByIdAsync (userId))
    {
        return null;
    }

    bool isCorrectly = false;

    if (answer.Answers == question.CorrectAnswers)
    {
        isCorrectly = true;
    }

    if (await
    _activeQuestService.CheckExistActiveQuestAsync (question.QuestId, userId))
    {
        if (isCorrectly)
        {
            if (!await
            _activeQuestService.AddRatingAsync (question.QuestId, userId,
            question.Rating))
            {
                return null;
            }
        }

        if (!await
        _activeQuestService.AddCompletedQuestion (question.QuestId, userId))
        {

```

```

        return null;
    }
    else
    {
        ActiveQuestInDto activeQuest = new
ActiveQuestInDto()
        {
            Rating = isCorrectly ? question.Rating : 0,
            CompletedQuestions = 1,
            QuestId = question.QuestId,
            UserId = userId
        };

        if (!await
_activeQuestService.CreateActiveQuestAsync(activeQuest))
        {
            return null;
        }

        return new CheckedAnswerDto()
        {
            QuestionId = answer.QuestionId,
            CorrectAnswers = question.CorrectAnswers,
            IsCorrectly = isCorrectly
        };
    }
}
}

```

Листинг В.9 – IQuestionsRepository.cs программы questie_backend

```

using entitys.Models;

namespace questie_backend.Repositories.Interfaces
{
    public interface IQuestionsRepository
    {
        Task DeleteAsync(Question question);
        Task<List<Question>> GetAllQuestionsAsync();
        Task<string?> GetHintByQuestionIdAsync(Guid questionId);
        Task<Question?> GetQuestionByIdAsync(Guid questionId);
        Task<Guid?> GetQuestionIdWithNextIdAsync(Guid nextId);
        Task<List<Question>> GetQuestionsFromQuestAsync(Guid questId);
        Task<Guid?> GetTargetIdByQuestionIdAsync(Guid questionId);
        Task InsertQuestionAsync(Question quest);
        Task UpdateAsync(Question oldQuestion);
    }
}

```

Листинг В.10 – QuestionsRepository.cs программы questie_backend

```

using entitys.Models;
using Microsoft.EntityFrameworkCore;
using questie_backend.Data;
using questie_backend.Repositories.Interfaces;

```

```

namespace questie_backend.Repositories.Implimentations
{
    public class QuestionsRepository : IQuestionsRepository
    {
        private readonly QuestsContext _context;
        public QuestionsRepository(QuestsContext context)
        {
            _context = context;
        }

        public async Task DeleteAsync(Question question)
        {
            _context.QuestionsDbSet.Remove(question);
            await _context.SaveChangesAsync();
        }

        public async Task<List<Question>> GetAllQuestionsAsync()
        {
            return await _context.QuestionsDbSet.ToListAsync();
        }

        public async Task<string?> GetHintByQuestionIdAsync(Guid
questionId)
        {
            Question? question = await
_context.QuestionsDbSet.FindAsync(questionId);
            return question != null ? question.Hint : null;
        }

        public async Task<Guid?> GetTargetIdByQuestionIdAsync(Guid
questionId)
        {
            Question? question = await
_context.QuestionsDbSet.FindAsync(questionId);
            return question != null ? question.TargetId : null;
        }

        public async Task<Question?> GetQuestionByIdAsync(Guid
questionId)
        {
            return await
_context.QuestionsDbSet.FindAsync(questionId);
        }

        public async Task InsertQuestionAsync(Question question)
        {
            await _context.QuestionsDbSet.AddAsync(question);
            await _context.SaveChangesAsync();
        }

        public async Task UpdateAsync(Question oldQuestion)
        {
            await _context.SaveChangesAsync();
        }

        public async Task<Guid?> GetQuestionIdWithNextIdAsync(Guid
nextId)

```

```

        {
            Question? question = await
_context.QuestionsDbSet.FirstOrDefaultAsync(q => q.NextId == nextId);
            return question != null ? question.Id : null;
        }

        public async Task<List<Question>>
GetQuestionsFromQuestAsync(Guid questId)
        {
            return await _context.QuestionsDbSet.Where(q =>
q.QuestId == questId).ToListAsync();
        }
    }
}

```

Листинг В.11 – QuestsController.cs программы questie_backend

```

using entitys.DTO;
using entitys.Models;
using Microsoft.AspNetCore.Mvc;
using questie_backend.Services.Interfaces;

namespace questie_backend.Controllers
{
    [ApiController]
    [Produces("application/json")]
    [Route("api/[controller]")]
    public class QuestsController : ControllerBase
    {
        private IQuestService _questService;
        public QuestsController(IQuestService questService)
        {
            _questService = questService;
        }

        [HttpGet("all-brief-info")]
        public async Task<List<QuestBrifOutDto>>
GetAllBriefInformation()
        {
            return await _questService.GetAllQuestsBrifAsync();
        }

        [HttpGet("all-full-info")]
        public async Task<List<QuestOutDto>> GetAllFullInformation()
        {
            return await _questService.GetAllQuestsFullAsync();
        }

        [HttpGet("brief-info/{questId}")]
        public async Task<ActionResult<QuestBrifOutDto>>
GetBriefInformationById(Guid questId)
        {
            if (questId == Guid.Empty)
            {
                return BadRequest("Идентификатор квеста не может
быть пустым");
            }
        }
    }
}

```

```

        QuestBrifOutDto? quest = await
_questService.GetQuestBrifByIdAsync(questId);
        return quest is not null ? quest : NotFound();
    }

    [HttpGet("full-info/{questId}")]
    public async Task<ActionResult<QuestOutDto>>
GetFullInformationById(Guid questId)
    {
        if (questId == Guid.Empty)
        {
            return BadRequest("Идентификатор квеста не может
быть пустым");
        }

        QuestOutDto? quest = await
_questService.GetQuestFullByIdAsync(questId);
        return quest is not null ? quest : NotFound();
    }

    [HttpGet("map/{questId}")]
    public async Task<IActionResult> GetMapImageById(Guid questId)
    {
        if (questId == Guid.Empty)
        {
            return BadRequest("Идентификатор квеста не может
быть пустым");
        }

        string? imageSrc = await
_questService.GetMapImageByQuestIdAsync(questId);

        return imageSrc is not null ? Ok(new { imageSrc }) :
NotFound();
    }

    [HttpPost("end/{questId}")]
    public async Task<ActionResult<CompletedQuestOutDto?>>
EndQuest(Guid questId)
    {
        if (questId == Guid.Empty)
        {
            return BadRequest("Идентификатор квеста не может
быть пустым");
        }

        CompletedQuestOutDto? completedQuest = await
_questService.GetCompletedQuest(questId);

        return completedQuest is not null
            ? Ok(completedQuest)
            : NotFound();
    }

    [HttpPost("create")]

```

```

        public async Task<IActionResult> CreateQuest(QuestInDto
questDto)
        {
            if (questDto is null)
            {
                return BadRequest("Невозможно создать пустой квест.
Полученный объект квеста равен NULL");
            }

            Quest quest = await
_questService.CreateQuestAsync(questDto);

            return CreatedAtAction("GetFullInformationById", new {
questId = quest.Id }, quest);
        }

        [HttpDelete("{questId}")]
        public async Task<IActionResult> DeleteQuest(Guid questId)
        {
            if (questId == Guid.Empty)
            {
                return BadRequest("Идентификатор квеста не может
быть пустым");
            }

            return await _questService.DeleteQuestAsync(questId) ?
NoContent() : NotFound();
        }

        [HttpPut("quest")]
        public async Task<IActionResult> UpdateQuest(Guid questId,
QuestUpdateDto questUpdateDto)
        {
            if (questId == Guid.Empty)
            {
                return BadRequest("Идентификатор квеста не может
быть пустым");
            }
            if (questUpdateDto is null)
            {
                return BadRequest("Обновленный квест не может быть
пустым");
            }

            return await _questService.UpdateQuestAsync(questId,
questUpdateDto)
                ? NoContent()
                : NotFound();
        }

        [HttpPut("first")]
        public async Task<IActionResult> AddFirstQuestionId(Guid
questId, Guid firstQuestionId)
        {
            if (questId == Guid.Empty)
            {

```

```

        return BadRequest("Идентификатор квеста не может
        быть пустым");
    }

    if (firstQuestionId == Guid.Empty)
    {
        return BadRequest("Идентификатор первого задания не
        может быть пустым");
    }

    return await
    _questService.UpdateFirstQuestionIdAsync(questId, firstQuestionId)
        ? NoContent()
        : NotFound();
    }
}

```

Листинг В.12 – IQuestService.cs программы questie_backend

```

using entitys.DTO;
using entitys.Models;

namespace questie_backend.Services.Interfaces
{
    public interface IQuestService
    {
        Task<Quest> CreateQuestAsync(QuestInDto questDto);
        Task<bool> DeleteQuestAsync(Guid questId);
        Task<List<QuestBrifOutDto>> GetAllQuestsBrifAsync();
        Task<List<QuestOutDto>> GetAllQuestsFullAsync();
        Task<string?> GetMapImageByQuestIdAsync(Guid questId);
        Task<QuestBrifOutDto?> GetQuestBrifByIdAsync(Guid questId);
        Task<QuestOutDto?> GetQuestFullByIdAsync(Guid questId);
        Task<bool> IncrementQuantityQuestionsAsync(Guid questId);
        Task<bool> DecrementQuantityQuestionsAsync(Guid questId);
        Task<bool> UpdateFirstQuestionIdAsync(Guid questId, Guid
firstQuestionId);
        Task<bool> UpdateQuestAsync(Guid questId, QuestUpdatedDto
questUpdatedDto);
        Task<CompletedQuestOutDto?> GetCompletedQuest(Guid questId);
    }
}

```

Листинг В.13 – QuestService.cs программы questie_backend

```

using entitys.DTO;
using entitys.Enums;
using entitys.Models;
using questie_backend.Repositories.Interfaces;
using questie_backend.Services.Interfaces;
using System.Text.Json.Nodes;
using System.Text.Json;

namespace questie_backend.Services.Implimentations
{
    public class QuestionsService : IQuestionsService

```

```

    {
        private readonly IQuestionsRepository _questionsRepository;
        private readonly IActiveQuestService _activeQuestService;
        private readonly INotesService _notesService;
        private readonly IQuestService _questService;
        private readonly IUsersService _userService;
        private readonly IWebHostEnvironment _webHostEnvironment;
        private readonly IHttpContextAccessor _httpContextAccessor;

        public QuestionsService(IQuestionsRepository
questionsRepository,
                                IActiveQuestService
activeQuestService,
                                INotesService notesService,
                                IQuestService questService,
                                IUsersService userService,
                                IWebHostEnvironment
webHostEnvironment,
                                IHttpContextAccessor
httpContextAccessor)
        {
            _questionsRepository = questionsRepository;
            _activeQuestService = activeQuestService;
            _notesService = notesService;
            _questService = questService;
            _userService = userService;
            _webHostEnvironment = webHostEnvironment;
            _httpContextAccessor = httpContextAccessor;
        }

        public async Task<bool> CheckQuestionAsync(Guid questionId)
        {
            return await
            _questionsRepository.GetQuestionByIdAsync(questionId) == null;
        }

        public async Task<bool> DeleteQuestionAsync(Guid questionId)
        {
            Question? question = await
            _questionsRepository.GetQuestionByIdAsync(questionId);
            if (question == null)
            {
                return false;
            }

            Guid? previousId = await
            _questionsRepository.GetQuestionIdWithNextIdAsync(questionId);

            if (previousId != null)
            {
                await UpdateNextQuestionIdAsync(previousId.Value,
question.NextId);
            }
            else
            {

```

```

        await
        _questService.UpdateFirstQuestionIdAsync(question.QuestId,
        question.NextId);
    }

    if (question.NoteId is not null)
    {
        await
        _notesService.DeleteNoteAsync(question.NoteId.Value);
    }

    DeleteImage(question.ImageName);
    await _questionsRepository.DeleteAsync(question);
    await
    _questService.DecrementQuantityQuestionsAsync(question.QuestId);

    return true;
}

public async Task<List<QuestionOutDto>> GetAllQuestionsAsync()
{
    List<Question> questions = await
    _questionsRepository.GetAllQuestionsAsync();
    List<QuestionOutDto> questionsOutDto = new
    List<QuestionOutDto>();

    foreach (var question in questions)
    {
        questionsOutDto.Add(new QuestionOutDto()
        {
            Id = question.Id,
            TextQuestion = question.TextQuestion,
            Description = question.Description,
            Rating = question.Rating,
            ImageName =
            String.Format($"{_httpContextAccessor.HttpContext!.Request.Scheme}://" +
            $"{_httpContextAccessor.HttpContext!.Request.Host}" +
            $"{_httpContextAccessor.HttpContext!.Request.PathBase}" +
            $"/Images/Questions/{question.ImageName}"),
            Hint = question.Hint,
            Type = question.Type,
            Answers = question.Answers,
            CorrectAnswers = question.CorrectAnswers,
            AnsweredCounter = question.AnsweredCounter,
            AnsweredFirstCounter =
            question.AnsweredFirstCounter,
            NextId = question.NextId,
            QuestId = question.QuestId,
            TargetId = question.TargetId,
            NoteId = question.NoteId
        });
    }

    return questionsOutDto;
}

```

```

    }

    public async Task<List<QuestionOutDto>>
    GetQuestionsFromQuestAsync(Guid questId)
    {
        List<Question> questions = await
        _questionsRepository.GetQuestionsFromQuestAsync(questId);
        List<QuestionOutDto> questionsOutDto = new
        List<QuestionOutDto>();

        foreach (var question in questions)
        {
            questionsOutDto.Add(new QuestionOutDto()
            {
                Id = question.Id,
                TextQuestion = question.TextQuestion,
                Description = question.Description,
                Rating = question.Rating,
                ImageName =
                String.Format($"{_httpContextAccessor.HttpContext!.Request.Scheme}://" +
                $"{_httpContextAccessor.HttpContext!.Request.Host}" +
                $"{_httpContextAccessor.HttpContext!.Request.PathBase}" +
                $"{_httpContextAccessor.HttpContext!.Request.Path}"/>{question.ImageName}"),
                Hint = question.Hint,
                Type = question.Type,
                Answers = question.Answers,
                CorrectAnswers = question.CorrectAnswers,
                AnsweredCounter = question.AnsweredCounter,
                AnsweredFirstCounter =
                question.AnsweredFirstCounter,
                NextId = question.NextId,
                QuestId = question.QuestId,
                TargetId = question.TargetId,
                NoteId = question.NoteId
            });
        }

        return questionsOutDto;
    }

    public async Task<HintDto?> GetHintByQuestionIdAsync(Guid
    questionId)
    {
        string? hint = await
        _questionsRepository.GetHintByQuestionIdAsync(questionId);

        return hint != null ? new HintDto()
        {
            Id = questionId,
            Hint = hint
        } : null;
    }

```

```

        public async Task<QuestionOutDto?> GetQuestionByIdAsync(Guid
questionId)
        {
            Question? question = await
            _questionsRepository.GetQuestionByIdAsync(questionId);
            QuestionOutDto? questionOutDto = null;

            if (question != null)
            {
                questionOutDto = new QuestionOutDto()
                {
                    Id = question.Id,
                    TextQuestion = question.TextQuestion,
                    Description = question.Description,
                    Rating = question.Rating,
                    ImageName =
String.Format($"{_httpContextAccessor.HttpContext!.Request.Scheme}://" +

                    $"{_httpContextAccessor.HttpContext!.Request.Host}" +

                    $"{_httpContextAccessor.HttpContext!.Request.PathBase}" +

                    $"/Images/Questions/{question.ImageName}"),
                    Hint = question.Hint,
                    Type = question.Type,
                    Answers = question.Answers,
                    CorrectAnswers = question.CorrectAnswers,
                    AnsweredCounter = question.AnsweredCounter,
                    AnsweredFirstCounter =
question.AnsweredFirstCounter,
                    NextId = question.NextId,
                    QuestId = question.QuestId,
                    TargetId = question.TargetId,
                    NoteId = question.NoteId
                };
            }

            return questionOutDto;
        }

        public async Task<Question?> CreateQuestionAsync(QuistionInDto
questionDto)
        {
            if (questionDto.QuestId == Guid.Empty)
            {
                return null;
            }

            if (await
            _questService.GetQuestBrifByIdAsync(questionDto.QuestId) is null)
            {
                return null;
            }

            Question question = new Question();
            question.TextQuestion = questionDto.TextQuestion;
            question.Description = questionDto.Description;

```

```

        question.Rating = questionDto.Rating;
        question.ImageName = questionDto.ImgFile != null ? await
SaveImage(questionDto.ImgFile) : string.Empty;
        question.Hint = questionDto.Hint;
        question.Type = questionDto.Type;
        question.Answers = questionDto.Answers;
        question.CorrectAnswers = questionDto.CorrectAnswers;
        question.AnsweredCounter = 0;
        question.AnsweredFirstCounter = 0;
        question.QuestId = questionDto.QuestId;
        question.TargetId = questionDto.TargetId;
        question.NoteId = questionDto.NoteId;
        question.CreatedAt = DateTime.UtcNow;
        question.UpdatedAt = DateTime.UtcNow;

        await
_questionsRepository.InsertQuestionAsync(question);

        await
_questService.IncrementQuantityQuestionsAsync(question.QuestId);

        return question;
    }

    public async Task<bool> UpdateNextQuestionIdAsync(Guid id,
Guid nextQuestionId)
    {
        Question? oldQuestion = await
_questionsRepository.GetQuestionByIdAsync(id);
        if (oldQuestion == null || await
_questionsRepository.GetQuestionByIdAsync(nextQuestionId) == null)
        {
            return false;
        }

        oldQuestion.NextId = nextQuestionId;
        oldQuestion.UpdatedAt = DateTime.UtcNow;

        await _questionsRepository.UpdateAsync(oldQuestion);

        return true;
    }

    public async Task<bool> UpdateQuestionAsync(Guid id,
QuestionUpdateDto questionUpdateDto)
    {
        Question? oldQuestion = await
_questionsRepository.GetQuestionByIdAsync(id);
        if (oldQuestion == null)
        {
            return false;
        }

        oldQuestion.TextQuestion =
questionUpdateDto.TextQuestion != null ? questionUpdateDto.TextQuestion :
oldQuestion.TextQuestion;

```

```

        oldQuestion.Description = questionUpdateDto.Description
        != null ? questionUpdateDto.Description : oldQuestion.Description;
        oldQuestion.Rating = questionUpdateDto.Rating != null ?
        (int)questionUpdateDto.Rating : oldQuestion.Rating;
        oldQuestion.TextQuestion =
        questionUpdateDto.TextQuestion != null ? questionUpdateDto.TextQuestion :
        oldQuestion.TextQuestion;

        if (questionUpdateDto.ImgFile != null)
        {
            DeleteImage(oldQuestion.ImageName);
            oldQuestion.ImageName = await
            SaveImage(questionUpdateDto.ImgFile);
        }

        oldQuestion.Hint = questionUpdateDto.Hint != null ?
        questionUpdateDto.Hint : oldQuestion.Hint;
        oldQuestion.Type = questionUpdateDto.Type != null ?
        (QuestionType)questionUpdateDto.Type : oldQuestion.Type;
        oldQuestion.Answers = questionUpdateDto.Answers != null
        ? questionUpdateDto.Answers : oldQuestion.Answers;
        oldQuestion.CorrectAnswers =
        questionUpdateDto.CorrectAnswers != null ?
        questionUpdateDto.CorrectAnswers : oldQuestion.CorrectAnswers;
        oldQuestion.TargetId = questionUpdateDto.TargetId !=
        null ? questionUpdateDto.TargetId : oldQuestion.TargetId;
        oldQuestion.NoteId = questionUpdateDto.NoteId != null ?
        questionUpdateDto.NoteId : oldQuestion.NoteId;
        oldQuestion.UpdatedAt = DateTime.UtcNow;

        await _questionsRepository.UpdateAsync(oldQuestion);

        return true;
    }

    private async Task<string> SaveImage(IFormFile imageFile)
    {
        string imageName = new
        String(Path.GetFileNameWithoutExtension(imageFile.FileName).Take(10).ToAr
        ray()).Replace(' ', '-');
        imageName += DateTime.Now.ToString("yymmssfff") +
        Path.GetExtension(imageFile.FileName);
        var imagePath =
        Path.Combine(_webHostEnvironment.ContentRootPath, "Images/Questions",
        imageName);

        using (var fileStream = new FileStream(imagePath,
        FileMode.Create))
        {
            await imageFile.CopyToAsync(fileStream);
        }

        return imageName;
    }

    private void DeleteImage(string imageName)
    {

```

```

        var imagePath =
Path.Combine(_webHostEnvironment.ContentRootPath, "Images/Questions",
imageName);

        if (File.Exists(imagePath))
        {
            File.Delete(imagePath);
        }
    }

    public async Task<CheckedAnswerDto?> GetCheckedAsync (AnswerDto
answer)
    {
        Question? question = await
_questionsRepository.GetQuestionByIdAsync (answer.QuestionId);
        if (question is null)
        {
            return null;
        }

        string? user =
_httpContextAccessor.HttpContext!.Session.GetString("User");
        if (user is null)
        {
            return null;
        }

        var jsonData =
JsonSerializer.Deserialize<JsonNode>(user!);
        if (jsonData is null)
        {
            return null;
        }

        Guid userId = (Guid)jsonData["Id"]!;

        if (!await _userService.CheckUserExistByIdAsync (userId))
        {
            return null;
        }

        bool isCorrectly = false;

        if (answer.Answers == question.CorrectAnswers)
        {
            isCorrectly = true;
        }

        if (await
_activeQuestService.CheckExistActiveQuestAsync (question.QuestId, userId))
        {
            if (isCorrectly)
            {
                if (!await
_activeQuestService.AddRatingAsync (question.QuestId, userId,
question.Rating))
                {

```


Листинг В.15 – QuestRepository.cs программы questie_backend

```
using entitys.Models;
using Microsoft.EntityFrameworkCore;
using questie_backend.Data;
using questie_backend.Repositories.Interfaces;

namespace questie_backend.Repositories.Implimentations
{
    public class QuestRepository : IQuestRepository
    {
        private readonly QuestsContext _context;
        public QuestRepository(QuestsContext context)
        {
            _context = context;
        }

        public async Task DeleteAsync(Quest quest)
        {
            _context.QuestsDbSet.Remove(quest);
            await _context.SaveChangesAsync();
        }

        public async Task<List<Quest>> GetAllQuestsAsync()
        {
            return await _context.QuestsDbSet.ToListAsync();
        }

        public async Task<string?> GetMapImageByQuestIdAsync(Guid
questId)
        {
            Quest? quest = await
_context.QuestsDbSet.FindAsync(questId);
            return quest != null ? quest.ImgMapName : null;
        }

        public async Task<Quest?> GetQuestByIdAsync(Guid questId)
        {
            return await _context.QuestsDbSet.FindAsync(questId);
        }

        public async Task InsertQuestAsync(Quest quest)
        {
            await _context.QuestsDbSet.AddAsync(quest);
            await _context.SaveChangesAsync();
        }

        public async Task UpdateAsync(Quest oldQuest)
        {
            await _context.SaveChangesAsync();
        }
    }
}
```

Листинг В.16 – UsersController.cs программы questie_backend_users

```
using entitys.DTO;
using entitys.Enums;
using entitys.Models;
using Microsoft.AspNetCore.Mvc;
```

```

using questie_backend_authorization.Exception;
using questie_backend_users.Services.Interfaces;

namespace questie_backend_users.Controllers
{
    [ApiController]
    [Produces("application/json")]
    [Route("api/[controller]")]
    public class UsersController : ControllerBase
    {
        private IUsersService _usersService;
        public UsersController(IUsersService usersService)
        {
            _usersService = usersService;
        }

        [HttpGet]
        public async Task<List<User>> GetUsers()
        {
            return await _usersService.GetAllUsersAsync();
        }

        [HttpGet("id/{id}")]
        public async Task<User?> GetUserById(Guid id)
        {
            return await _usersService.GetUserByIdAsync(id);
        }

        [HttpGet("login/{login}")]
        public async Task<User?> GetUserByLogin(string login)
        {
            return await _usersService.GetUserByLoginAsync(login);
        }

        [HttpGet("role/{login}")]
        public async Task<IActionResult> GetRoleByLogin(string login)
        {
            try
            {
                Role role = await
                _usersService.GetRoleByLoginAsync(login);
                return Ok(new { Role = role });
            }
            catch (NotFoundUserException ex)
            {
                return BadRequest(ex.Message);
            }
        }

        [HttpPost("rating")]
        public async Task<IActionResult> AddRatingUser(Guid id, int
addingRating)
        {
            if (id == Guid.Empty)
            {

```

```

        return BadRequest("Идентификатор пользователя не
может быть пустым");
    }

    return await _userService.UpdateRatingByIdAsync(id,
addingRating)
        ? NoContent()
        : NotFound();
    }

    [HttpPost("user")]
    public async Task<IActionResult> CreateUser(UserDto userDto)
    {
        User user = await
_usersService.CreateUserAsync(userDto);
        return CreatedAtAction("GetUserById", new { id = user.Id
}, user);
    }

    [HttpPost("exist")]
    public async Task<IActionResult> CheckExistUser(LoginUserDto
loginUserDto)
    {
        return await
_usersService.CheckUserAsync(loginUserDto.Login, loginUserDto.Password)
            ? Ok(new { Exist = "true" })
            : Ok(new { Exist = "false" });
    }

    [HttpPost("exist/{userId}")]
    public async Task<IActionResult> CheckExistUserById(Guid
userId)
    {
        return await _userService.CheckUserByIdAsync(userId)
            ? Ok(new { Exist = "true" })
            : Ok(new { Exist = "false" });
    }

    [HttpDelete("{id}")]
    public async Task<IActionResult> DeleteUser(Guid id)
    {
        await _userService.DeleteUserAsync(id);
        return NoContent();
    }

    [HttpPut]
    public async Task<IActionResult> UpdateUser(Guid id, UserDto
user)
    {
        await _userService.UpdateUserAsync(id, user);
        return NoContent();
    }

    [HttpPut("role")]
    public async Task<IActionResult> UpdateUserRole(Guid id, Role
role)

```

```

        {
            await _userService.UpdateUserRoleAsync(id, role);
            return NoContent();
        }

        [HttpPut("rating")]
        public async Task<IActionResult> UpdateUserRating(Guid id, int
rating)
        {
            await _userService.UpdateUserRatingAsync(id, rating);
            return NoContent();
        }
    }
}

```

Листинг В.17 – IUserService.cs программы questie_backend_users

```

using entitys.DTO;
using entitys.Enums;
using entitys.Models;

namespace questie_backend_users.Services.Interfaces
{
    public interface IUserService
    {
        public Task<List<User>> GetAllUsersAsync();
        public Task<User?> GetUserByIdAsync(Guid id);
        public Task<User?> GetUserByLoginAsync(string login);
        public Task<Role> GetRoleByLoginAsync(string login);
        public Task<User> CreateUserAsync(UserDto userDto);
        public Task<bool> CheckUserAsync(string login, string
password);
        public Task DeleteUserAsync(Guid id);
        public Task UpdateUserAsync(Guid id, UserDto user);
        public Task UpdateUserRoleAsync(Guid id, Role role);
        public Task UpdateUserRatingAsync(Guid id, int rating);
        Task<bool> UpdateRatingByIdAsync(Guid id, int addingRating);
        Task<bool> CheckUserByIdAsync(Guid userId);
    }
}

```

Листинг В.18 – UsersService.cs программы questie_backend_users

```

using entitys.DTO;
using entitys.Enums;
using entitys.Models;
using questie_backend_authorization.Exception;
using questie_backend_users.Repositories.Interfaces;
using questie_backend_users.Services.Interfaces;

namespace questie_backend_users.Services.Implementations
{
    public class UsersService : IUserService
    {
        private readonly IUserRepository _userRepository;

        public UsersService(IUserRepository userRepository)
        {

```

```

        _usersRepository = usersRepository;
    }

    public async Task<List<User>> GetAllUsersAsync()
    {
        return await _usersRepository.GetAllAsync();
    }

    public async Task<User?> GetUserByIdAsync(Guid id)
    {
        return await _usersRepository.GetByIdAsync(id);
    }

    public async Task<User?> GetUserByLoginAsync(string login)
    {
        return await _usersRepository.GetByLoginAsync(login);
    }

    public async Task<Role> GetRoleByLoginAsync(string login)
    {
        User? user = await
        _usersRepository.GetByLoginAsync(login);

        if (user == null)
        {
            throw new NotFoundUserException($"Пользователь не
найден. Отсутствующий логин: {login}");
        }

        return user.Role;
    }

    public async Task<bool> CheckUserAsync(string login, string
password)
    {
        User? user = await
        _usersRepository.GetByLoginAsync(login);

        if (user == null)
        {
            return false;
        }

        return user.Password.Equals(password);
    }

    public async Task<bool> CheckUserByIdAsync(Guid userId)
    {
        if (userId == Guid.Empty)
        {
            return false;
        }

        User? user = await
        _usersRepository.GetByIdAsync(userId);

        return user is not null;
    }

```

```

    }

    public async Task<User> CreateUserAsync(UserDto userDto)
    {
        User user = new User(userDto.Name,
                               userDto.Surname,
                               userDto.Email,
                               userDto.Login,
                               userDto.Password,
                               userDto.Role,
                               userDto.CityId);
        await _usersRepository.InsertAsync(user);
        return user;
    }

    public async Task DeleteUserAsync(Guid id)
    {
        User? user = await _usersRepository.GetByIdAsync(id);
        if (user != null)
        {
            await _usersRepository.DeleteAsync(user);
        }
    }

    public async Task UpdateUserAsync(Guid id, UserDto user)
    {
        User? oldUser = await _usersRepository.GetByIdAsync(id);
        if (oldUser != null)
        {
            oldUser.Name = user.Name;
            oldUser.Surname = user.Surname;
            oldUser.Email = user.Email;
            oldUser.Login = user.Login;
            oldUser.Password = user.Password;
            oldUser.Rating = user.Rating;
            oldUser.Role = user.Role;
            oldUser.CityId = user.CityId;
            await _usersRepository.UpdateAsync(oldUser);
        }
    }

    public async Task UpdateUserRoleAsync(Guid id, Role role)
    {
        User? oldUser = await _usersRepository.GetByIdAsync(id);
        if (oldUser != null)
        {
            oldUser.Role = role;
            await _usersRepository.UpdateAsync(oldUser);
        }
    }

    public async Task UpdateUserRatingAsync(Guid id, int rating)
    {
        User? oldUser = await _usersRepository.GetByIdAsync(id);
        if (oldUser != null)
        {
            oldUser.Rating = rating;
        }
    }

```

```

        await _usersRepository.UpdateAsync(oldUser);
    }
}

public async Task<bool> UpdateRatingByIdAsync(Guid id, int
addingRating)
{
    User? oldUser = await _usersRepository.GetByIdAsync(id);
    if (oldUser == null)
    {
        return false;
    }

    oldUser.Rating += addingRating;
    await _usersRepository.UpdateAsync(oldUser);

    return true;
}
}
}

```

Листинг В.19 – IUsersRepository.cs программы questie_backend_users

```

using entitys.Models;

namespace questie_backend_users.Repositories.Interfaces
{
    public interface IUsersRepository
    {
        Task DeleteAsync(User user);
        Task<List<User>> GetAllAsync();
        Task<User?> GetByIdAsync(Guid id);
        Task<User?> GetByLoginAsync(string login);
        Task InsertAsync(User user);
        Task UpdateAsync(User user);
    }
}

```

Листинг В.20 – UsersRepository.cs программы questie_backend_users

```

using entitys.Models;
using Microsoft.EntityFrameworkCore;
using questie_backend_users.Data;
using questie_backend_users.Repositories.Interfaces;

namespace questie_backend_users.Repositories.Implementations
{
    public class UsersRepository : IUsersRepository
    {
        private readonly UsersContext _context;
        public UsersRepository(UsersContext context)
        {
            _context = context;
        }

        public async Task<List<User>> GetAllAsync()
        {

```

```

        return await _context.UsersDbSet.Include(u =>
u.City).ToListAsync();
    }

    public async Task<User?> GetByIdAsync(Guid id)
    {
        return await _context.UsersDbSet.Include(u =>
u.City).FirstOrDefaultAsync(u => u.Id == id);
    }

    public async Task<User?> GetByLoginAsync(string login)
    {
        return await _context.UsersDbSet.Include(u =>
u.City).FirstOrDefaultAsync(u => u.Login == login);
    }

    public async Task InsertAsync(User user)
    {
        await _context.UsersDbSet.AddAsync(user);
        await _context.SaveChangesAsync();
    }

    public async Task DeleteAsync(User user)
    {
        _context.UsersDbSet.Remove(user);
        await _context.SaveChangesAsync();
    }

    public async Task UpdateAsync(User user)
    {
        user.UpdatedAt = DateTime.UtcNow;
        await _context.SaveChangesAsync();
    }
}

```

Листинг В.21 – AuthorizationController.cs программы

questie_backend_authorization

```

using entitys.Enums;
using entitys.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.IdentityModel.Tokens;
using questie_backend_authorization.Exceptions;
using questie_backend_authorization.Services.Interfaces;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;

namespace questie_backend_authorization.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    [Produces("application/json")]
    public class AuthorizationController : ControllerBase

```

```

{
    private readonly IConfiguration _configuration;
    private readonly IUserService _userService;
    public AuthorizationController(IConfiguration configuration,
    IUserService userService)
    {
        _configuration = configuration;
        _userService = userService;
    }

    /// <summary>
    /// Получение токена авторизации
    /// </summary>
    /// <param name="data">Логин и пароль</param>
    /// <returns>Токен авторизации</returns>
    [AllowAnonymous]
    [HttpPost]
    public async Task<IActionResult> Auth(LoginUser data)
    {
        try
        {
            bool isValid = await
            _userService.IsValidUserInformationAsync(data);
            if (isValid)
            {
                var tokenString = await
                GenerateJwtToken(data.Login);
                if (string.IsNullOrEmpty(tokenString))
                {
                    return BadRequest("Не удалось
                сгенерировать токен");
                }
                return Ok(new { Token = tokenString });
            }
            return BadRequest("Пожалуйста, введите
            действительное имя пользователя и пароль");
        }
        catch (Exception)
        {
            return BadRequest("Пожалуйста, введите
            действительное имя пользователя и пароль");
        }
    }

    /// <summary>
    /// Генерация JWT-токена
    /// Чутьочка документации по клеймам
    https://metanit.com/sharp/aspnet5/15.5.php
    /// </summary>
    /// <param name="userLogin"></param>
    /// <returns></returns>
    private async Task<string> GenerateJwtToken(string userLogin)
    {
        try
        {
            Role role = await
            _userService.GetUserRoleByLoginAsync(userLogin);

```

```

        var key =
Encoding.ASCII.GetBytes(_configuration["Jwt:key"]);

        var claims = new List<Claim> {
            new Claim(ClaimsIdentity.DefaultNameClaimType,
userLogin),
            new Claim(ClaimsIdentity.DefaultRoleClaimType,
role.ToString())
        };

        // создаем JWT-токен
        var jwt = new JwtSecurityToken(
            issuer: _configuration["Jwt:Issuer"],
            audience:
_configuration["Jwt:Audience"],
            claims: claims,
            expires: DateTime.UtcNow.AddHours(1),
            signingCredentials: new
SigningCredentials(new SymmetricSecurityKey(key),
SecurityAlgorithms.HmacSha256Signature));

        return new
JwtSecurityTokenHandler().WriteToken(jwt);
    }
    catch (NotFoundUserException)
    {
        return "";
    }
}
}
}

```

Приложение Г
(рекомендуемое)
Функциональное тестирование

Таблица Г.1 – Тест-кейсы тестирования авторизации пользователя и результаты их прохождения

Решение	Входные данные	Ожидаемый результат	Полученный результат
1	Логин: l@g Пароль: pas:)	Некорректная длина логина, некорректная длина пароля, некорректные символы в составе логина, некорректные символы в составе пароля	Тест пройден. Сообщения о соответствующих ошибках получены (рисунок Г.1)
2	Логин: l@gin001 Пароль: pas:)	Некорректная длина пароля, некорректные символы в составе логина, некорректные символы в составе пароля	Тест пройден. Сообщения о соответствующих ошибках получены (рисунок Г.2)
3	Логин: lg\$ Пароль: password:)	Некорректная длина логина, некорректные символы в составе логина, некорректные символы в составе пароля	Тест пройден. Сообщения о соответствующих ошибках получены (рисунок Г.3)
4	Логин: log!nFly Пароль: some_p^ssword	Некорректные символы в составе логина, некорректные символы в составе пароля	Тест пройден. Сообщения о соответствующих ошибках получены (рисунок Г.4)
5	Логин: This_is_very_very_long_login Пароль: This_is_very_very_long_password№1	Некорректная длина логина, некорректная длина пароля, некорректные символы в составе пароля	Тест пройден. Сообщения о соответствующих ошибках получены (рисунок Г.5)
6	Логин: login_1 Пароль: NoCoRrEctly%PaS	Некорректная длина пароля, некорректные символы в составе пароля	Тест пройден. Сообщения о соответствующих ошибках получены (рисунок Г.6)
7	Логин: LOG Пароль: (:password;)	Некорректная длина логина, некорректные символы в составе пароля	Тест пройден. Сообщения о соответствующих ошибках получены (рисунок Г.7)
8	Логин: Correctly_Login_1 Пароль: pswd_7+21	Некорректные символы в составе пароля	Тест пройден. Сообщения о соответствующих ошибках получены (рисунок Г.8)
9	Логин: again_(very)_log_login Пароль: short	Некорректная длина логина, некорректная длина пароля, Некорректные символы в составе логина	Тест пройден. Сообщения о соответствующих ошибках получены (рисунок Г.9)
10	Логин: some~login Пароль: pswd	Некорректная длина пароля, некорректные символы в составе логина	Тест пройден. Сообщения о соответствующих ошибках получены (рисунок Г.10)
11	Логин: log!! Пароль: password_17	Некорректная длина логина, некорректные символы в составе логина	Тест пройден. Сообщения о соответствующих ошибках получены (рисунок Г.11)
12	Логин: login!! Пароль: password_17	Некорректные символы в составе логина	Тест пройден. Сообщения о соответствующих ошибках получены (рисунок Г.12)

Продолжение таблицы Г.1

Решение	Входные данные	Ожидаемый результат	Полученный результат
13	Логин: aaaaaaaaarrrrrriiiivvvvaa Пароль: pswrd	Некорректная длина логина, некорректная длина пароля	Тест пройден. Сообщения о соответствующих ошибках получены (рисунок Г.13)
14	Логин: some_Login Пароль: aga!n_Very_Long?_pAsSwOrD	Некорректная длина пароля	Тест пройден. Сообщения о соответствующих ошибках получены (рисунок Г.14)
15	Логин: Correct1y_Log1n_but_long_again Пароль: password123	Некорректная длина логина	Тест пройден. Сообщения о соответствующих ошибках получены (рисунок Г.15)
16	Логин: user_login Пароль: user_password (пароль не соответствует логину)	Неверный логин или пароль	Тест пройден. Сообщения о соответствующих ошибках получены (рисунок Г.16)
17	Логин: user_login Пароль: user&pass#ord! (пароль соответствует логину)	Успешная авторизация. Перенаправление в аккаунт	Тест пройден. Сообщения о соответствующих ошибках получены (рисунок Г.17)

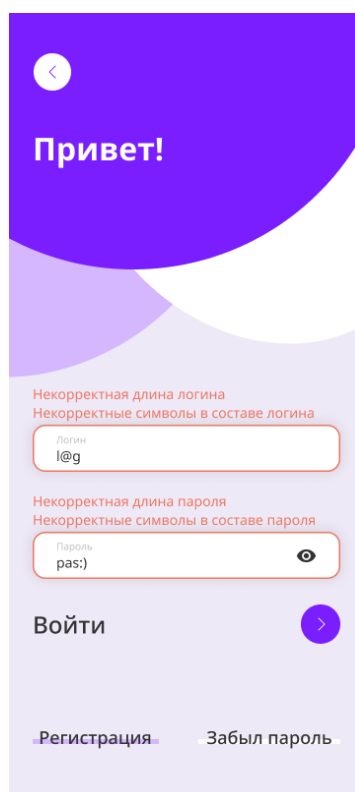


Рисунок Г.1 – Результат прохождения тест-кейса 1 для авторизации

The screenshot shows a login interface with a purple header containing a back arrow and the text "Привет!". Below the header, there are two input fields. The first field, labeled "Логин", contains the text "l@gin001" and has a red border with the error message "Некорректные символы в составе логина" above it. The second field, labeled "Пароль", contains the text "pas:)" and has a red border with the error message "Некорректная длина пароля" and "Некорректные символы в составе пароля" above it. Below the password field is a "Войти" button with a right arrow. At the bottom, there are links for "Регистрация" and "Забыл пароль".

Рисунок Г.2 – Результат прохождения тест-кейса 2 для авторизации

The screenshot shows a login interface with a purple header containing a back arrow and the text "Привет!". Below the header, there are two input fields. The first field, labeled "Логин", contains the text "lg\$" and has a red border with the error message "Некорректная длина логина" and "Некорректные символы в составе логина" above it. The second field, labeled "Пароль", contains the text "password:)" and has a red border with the error message "Некорректные символы в составе пароля" above it. Below the password field is a "Войти" button with a right arrow. At the bottom, there are links for "Регистрация" and "Забыл пароль".

Рисунок Г.3 – Результат прохождения тест-кейса 3 для авторизации

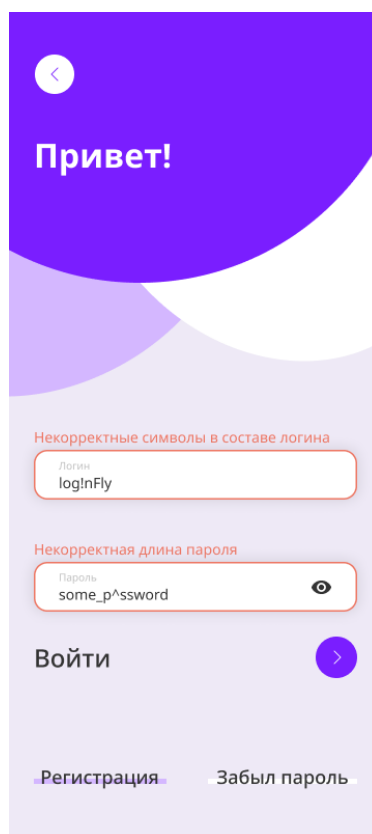


Рисунок Г.4 – Результат прохождения тест-кейса 4 для авторизации

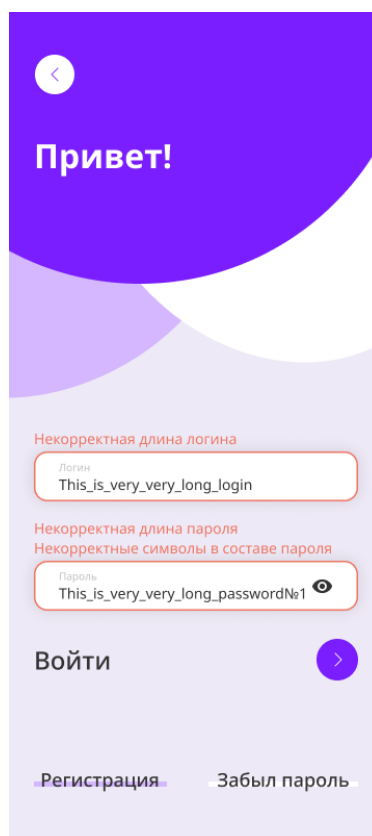
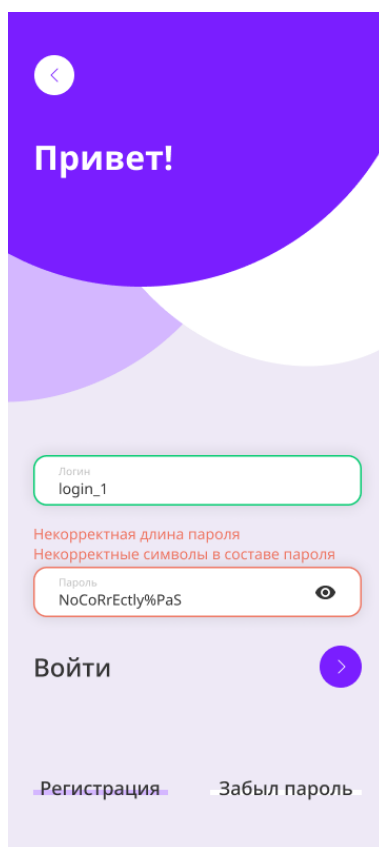
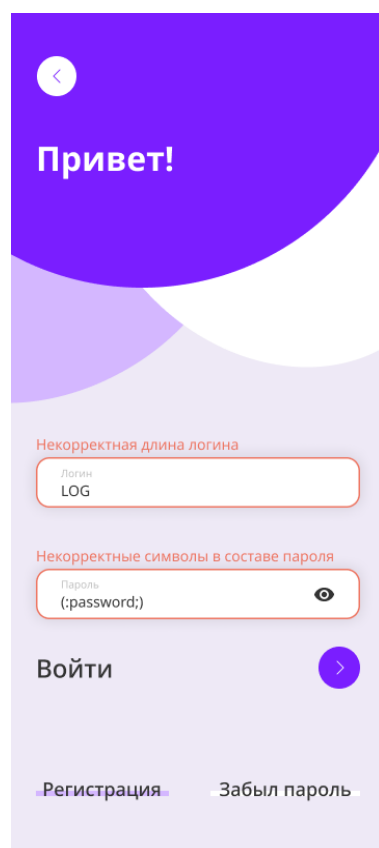


Рисунок Г.5 – Результат прохождения тест-кейса 5 для авторизации



The screenshot shows a mobile application interface for login. At the top, there is a blue header with a back arrow and the text "Привет!". Below the header, there is a login form. The "Логин" (Login) field contains the text "login_1". The "Пароль" (Password) field contains the text "NoCoRrEcTly%PaS" and has a red border, indicating an error. Above the password field, there are two lines of red text: "Некорректная длина пароля" and "Некорректные символы в составе пароля". Below the password field, there is a "Войти" (Login) button with a right arrow. At the bottom, there are two links: "Регистрация" (Registration) and "Забыл пароль" (Forgot password).

Рисунок Г.6 – Результат прохождения тест-кейса 6 для авторизации



The screenshot shows the same mobile application interface for login. The "Логин" (Login) field now contains the text "LOG" and has a red border, indicating an error. Above the login field, there is a line of red text: "Некорректная длина логина". The "Пароль" (Password) field contains the text "(:password;)" and has a red border, indicating an error. Above the password field, there is a line of red text: "Некорректные символы в составе пароля". The "Войти" (Login) button and the links "Регистрация" (Registration) and "Забыл пароль" (Forgot password) are still present at the bottom.

Рисунок Г.7 – Результат прохождения тест-кейса 7 для авторизации

A mobile application login screen with a purple header containing a back arrow and the text "Привет!". Below the header, there is a green-bordered input field for the login with the text "Логин" and "Correctly_Login_1". Below this, there is a red error message "Некорректные символы в составе пароля" above a red-bordered password input field containing "pswd_7+21" and an eye icon. At the bottom, there is a "Войти" button with a right arrow, and two links: "Регистрация" and "Забыл пароль".

Рисунок Г.8 – Результат прохождения тест-кейса 8 для авторизации

A mobile application login screen with a purple header containing a back arrow and the text "Привет!". Below the header, there is a red error message "Некорректная длина логина" and "Некорректные символы в составе логина" above a red-bordered login input field containing "Логин" and "again_(very)_log_login". Below this, there is a red error message "Некорректная длина пароля" above a red-bordered password input field containing "Пароль" and "short" with an eye icon. At the bottom, there is a "Войти" button with a right arrow, and two links: "Регистрация" and "Забыл пароль".

Рисунок Г.9 – Результат прохождения тест-кейса 9 для авторизации

The screenshot shows a mobile application login screen. At the top, there is a blue header with a back arrow icon and the text "Привет!". Below the header, there are two input fields. The first field is labeled "Логин" and contains the text "some-login". Above this field, there is a red error message: "Некорректные символы в составе логина". The second field is labeled "Пароль" and contains the text "pswd". Above this field, there is a red error message: "Некорректная длина пароля". Below the input fields, there is a blue button labeled "Войти" with a right arrow icon. At the bottom, there are two links: "Регистрация" and "Забыл пароль".

Рисунок Г.10 – Результат прохождения тест-кейса 10 для авторизации

The screenshot shows the same mobile application login screen. The "Логин" field now contains "log!!" and has two red error messages above it: "Некорректная длина логина" and "Некорректные символы в составе логина". The "Пароль" field now contains "password_17" and has a green border, indicating it is valid. The "Войти" button and the bottom links remain the same.

Рисунок Г.11 – Результат прохождения тест-кейса 11 для авторизации

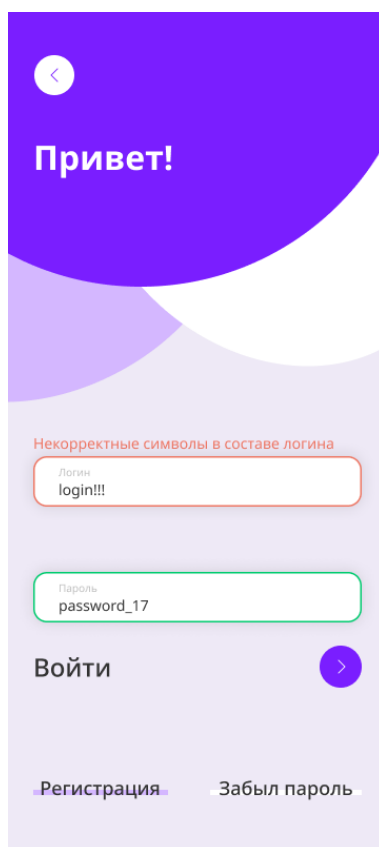


Рисунок Г.12 – Результат прохождения тест-кейса 12 для авторизации

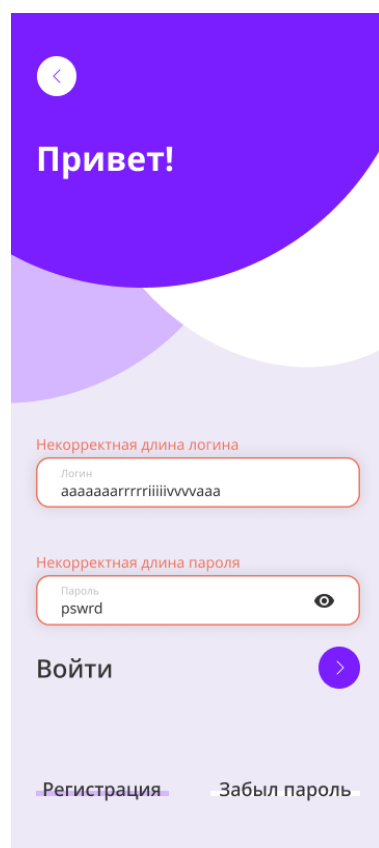
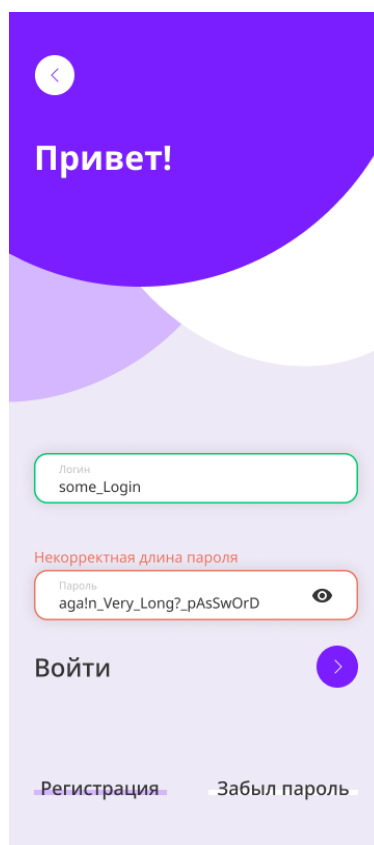
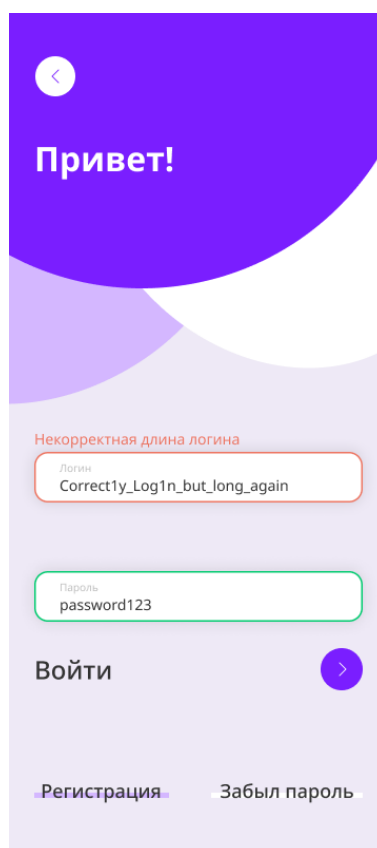


Рисунок Г.13 – Результат прохождения тест-кейса 13 для авторизации



A mobile application login screen with a purple header. The header contains a back arrow icon and the text "Привет!". Below the header, there are two input fields. The first field is labeled "Логин" and contains the text "some_Login". The second field is labeled "Пароль" and contains the text "aga!n_Very_Long?_pAsSwOrD". Above the password field, there is a red error message: "Некорректная длина пароля". Below the input fields, there is a button labeled "Войти" with a right arrow icon. At the bottom, there are two links: "Регистрация" and "Забыл пароль".

Рисунок Г.14 – Результат прохождения тест-кейса 14 для авторизации



A mobile application login screen with a purple header. The header contains a back arrow icon and the text "Привет!". Below the header, there are two input fields. The first field is labeled "Логин" and contains the text "Correct1y_Log1n_but_long_again". Above this field, there is a red error message: "Некорректная длина логина". The second field is labeled "Пароль" and contains the text "password123". Below the input fields, there is a button labeled "Войти" with a right arrow icon. At the bottom, there are two links: "Регистрация" and "Забыл пароль".

Рисунок Г.15 – Результат прохождения тест-кейса 15 для авторизации

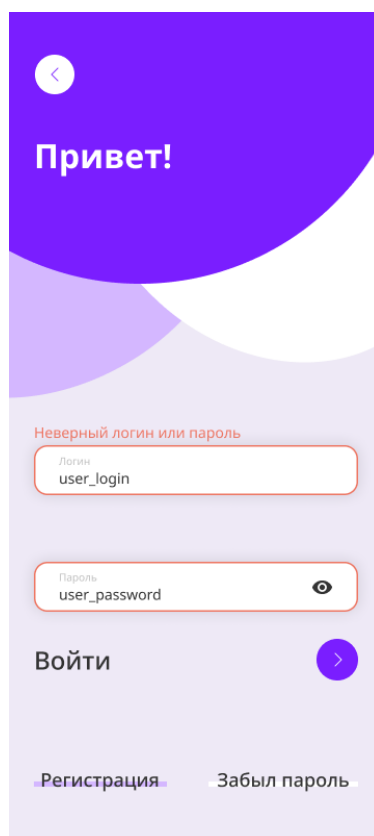


Рисунок Г.16 – Результат прохождения тест-кейса 16 для авторизации

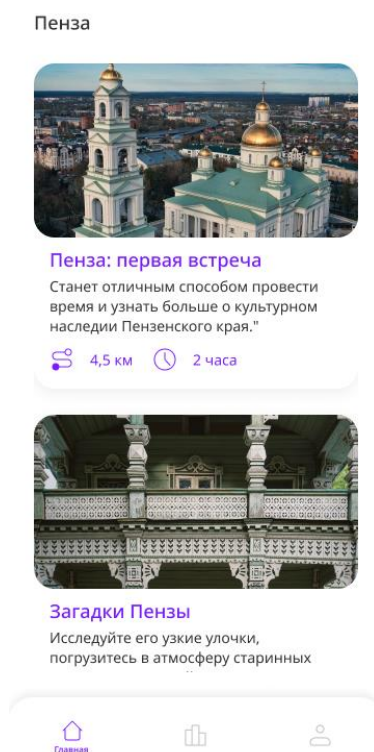


Рисунок Г.17 – Результат прохождения тест-кейса 17 для авторизации

Приложение Д
(рекомендуемое)
Исходный код модульного тестирования

Листинг Д.1 – TestTargetsController.cs

```

using entitys.DTO;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Moq;
using questie_backend.Controllers;
using questie_backend.Services.Interfaces;
using questie_backend.Tests.Mockdata;

namespace questie_backend.Tests.Controllers
{
    public class TestTargetsController
    {
        /// <summary>
        /// GetAllTargets
        /// </summary>
        [Fact]
        public async Task GetAllTargets_ShouldReturn200Status()
        {
            /// Arrange
            var targetsServiceMock = new Mock<ITargetsService>();
            targetsServiceMock.Setup(_ => _.GetAllTargetsAsync())

.ReturnsAsync(TargetsData.GetTestTargets());
            var targetsControllerMock = new
TargetsController(targetsServiceMock.Object);

            /// Act
            var result = await
targetsControllerMock.GetAllTargets();

            /// Assert
            Assert.Equal(TargetsData.GetTestTargets().Count(),
result.Count());
        }

        /// <summary>
        /// GetTargetById
        /// </summary>
        [Fact]
        public async Task GetTargetById_BadRequestResultWhenIdIsNull()
        {
            /// Arrange
            var targetsServiceMock = new Mock<ITargetsService>();
            var targetsControllerMock = new
TargetsController(targetsServiceMock.Object);

            /// Act
            var response = await
targetsControllerMock.GetTargetById(null);

            /// Assert
            Assert.IsType<BadRequestObjectResult>(response.Result);
        }

        [Fact]
    }
}

```

```

        public async Task
GetTargetById_BadRequestResultWhenIdIsEmpty()
    {
        /// Arrange
        var targetsServiceMock = new Mock<ITargetsService>();
        var targetsControllerMock = new
TargetsController(targetsServiceMock.Object);

        /// Act
        var response = await
targetsControllerMock.GetTargetById(Guid.Empty);

        /// Assert
        Assert.IsType<BadRequestObjectResult>(response.Result);
    }

    [Fact]
    public async Task
GetTargetById_NotFoundResultWhenUserNotFound()
    {
        /// Arrange
        Guid testTargetId = new Guid("6c17bd3b-a853-44ea-a193-
359a2bfe5de4");
        var targetsServiceMock = new Mock<ITargetsService>();
        targetsServiceMock.Setup(_ =>
_.GetTargetByIdAsync(testTargetId))

.ReturnsAsync(null as TargetOutDto);
        var controller = new
TargetsController(targetsServiceMock.Object);

        /// Act
        var response = await
controller.GetTargetById(testTargetId);

        /// Assert
        Assert.IsType<NotFoundResult>(response.Result);
    }

    [Fact]
    public async Task GetTargetById_TargetOutDto()
    {
        /// Arrange
        Guid testTargetId = new Guid("6c17bd3b-a853-44ea-a193-
359a2bfe5de1");
        var targetsServiceMock = new Mock<ITargetsService>();
        targetsServiceMock.Setup(_ =>
_.GetTargetByIdAsync(testTargetId))

.ReturnsAsync(TargetsData.GetTestTargets()).

        FirstOrDefault(p => p.Id == testTargetId));
        var controller = new
TargetsController(targetsServiceMock.Object);

        /// Act

```

```

        var response = await
controller.GetTargetById(testTargetId);

        /// Assert
        var target =
Assert.IsType<TargetOutDto>(response.Value);
        Assert.Equal("Target One", target.Title);
        Assert.Equal("Target Description One",
target.Description);

        Assert.Equal("https://localhost:5003/Images/Targets/target_one.jpg"
, target.ImageSrc);
    }

    /// <summary>
    /// GetTargetByQuestId
    /// </summary>
    [Fact]
    public async Task
GetTargetByQuestId_BadRequestResultWhenIdIsNull()
    {
        /// Arrange
        var targetsServiceMock = new Mock<ITargetsService>();
        var targetsControllerMock = new
TargetsController(targetsServiceMock.Object);

        /// Act
        var response = await
targetsControllerMock.GetTargetByQuestId(null);

        /// Assert
        Assert.IsType<BadRequestObjectResult>(response.Result);
    }

    [Fact]
    public async Task
GetTargetByQuestId_BadRequestResultWhenIdIsEmpty()
    {
        /// Arrange
        var targetsServiceMock = new Mock<ITargetsService>();
        var targetsControllerMock = new
TargetsController(targetsServiceMock.Object);

        /// Act
        var response = await
targetsControllerMock.GetTargetByQuestId(Guid.Empty);

        /// Assert
        Assert.IsType<BadRequestObjectResult>(response.Result);
    }

    [Fact]
    public async Task
GetTargetByQuestId_NotFoundResultWhenTargetNotFound()
    {
        /// Arrange

```

```

        Guid testQuestId = new Guid("6c17bd3b-a853-44ea-a193-
359a2bfe5de4");
        var targetsServiceMock = new Mock<ITargetsService>();
        targetsServiceMock.Setup(_ =>
_.GetTargetByQuestIdAsync(testQuestId))

.ReturnsAsync(null as TargetOutDto);
        var controller = new
TargetsController(targetsServiceMock.Object);

        /// Act
        var response = await
controller.GetTargetByQuestId(testQuestId);

        /// Assert
        Assert.IsType<NotFoundResult>(response.Result);
    }

    [Fact]
    public async Task GetTargetByQuestId_TargetOutDto()
    {
        /// Arrange
        Guid testQuestId = new Guid("6c17bd3b-a853-44ea-a193-
359a2bfe5de1");
        var targetsServiceMock = new Mock<ITargetsService>();
        targetsServiceMock.Setup(_ =>
_.GetTargetByQuestIdAsync(testQuestId))

.ReturnsAsync(TargetsData.GetTestTargets().
    FirstOrDefault(p => p.Id == testQuestId));
        var controller = new
TargetsController(targetsServiceMock.Object);

        /// Act
        var response = await
controller.GetTargetByQuestId(testQuestId);

        /// Assert
        var target =
Assert.IsType<TargetOutDto>(response.Value);
        Assert.Equal("Target One", target.Title);
        Assert.Equal("Target Description One",
target.Description);

        Assert.Equal("https://localhost:5003/Images/Targets/target_one.jpg"
, target.ImageSrc);
    }

    /// <summary>
    /// CreateTarget
    /// </summary>
    [Fact]
    public async Task CreateTarget_CreatedAtAction()
    {
        /// Arrange
        var fileMock = new Mock<IFormFile>();

```

```

        fileMock.Setup(f => f.FileName).Returns("example.jpg");
        fileMock.Setup(f => f.Length).Returns(1024); // Set file
length as needed

        TargetDto targetDto = TargetsData.GetNewTargetDto();
        targetDto.ImageFile = fileMock.Object;

        var targetsServiceMock = new Mock<ITargetsService>();
        targetsServiceMock.Setup(_ =>
            _.CreateTargetAsync(It.IsAny<TargetDto>()))
            .ReturnsAsync(TargetsData.GetNewTarget());

        var controller = new
TargetsController(targetsServiceMock.Object);

        /// Act
        var response = await controller.CreateTarget(targetDto);

        /// Assert
        var target =
Assert.IsType<CreatedAtActionResult>(response);
        targetsServiceMock.Verify(_ =>
            _.CreateTargetAsync(It.IsAny<TargetDto>()), Times.Exactly(1));
    }

    [Fact]
    public async Task
CreateTarget_BadRequestResultWhenTitleFieldIsNull()
    {
        /// Arrange
        var fileMock = new Mock<IFormFile>();
        fileMock.Setup(f => f.FileName).Returns("example.jpg");
        fileMock.Setup(f => f.Length).Returns(1024); // Set file
length as needed

        TargetDto targetDto =
TargetsData.GetNullTitleTargetDto();
        targetDto.ImageFile = fileMock.Object;

        var targetsServiceMock = new Mock<ITargetsService>();
        var controller = new
TargetsController(targetsServiceMock.Object);

        /// Act
        var response = await controller.CreateTarget(targetDto);

        /// Assert
        var target =
Assert.IsType<BadRequestObjectResult>(response);
    }

    [Fact]
    public async Task
CreateTarget_BadRequestResultWhenDescriptionFieldIsNull()
    {
        /// Arrange

```

```

        var fileMock = new Mock<IFormFile>();
        fileMock.Setup(f => f.FileName).Returns("example.jpg");
        fileMock.Setup(f => f.Length).Returns(1024); // Set file
length as needed

        TargetDto targetDto =
TargetsData.GetNullDescriptionTargetDto();
        targetDto.ImageFile = fileMock.Object;

        var targetsServiceMock = new Mock<ITargetsService>();
        var controller = new
TargetsController(targetsServiceMock.Object);

        /// Act
        var response = await controller.CreateTarget(targetDto);

        /// Assert
        var target =
Assert.IsType<BadRequestObjectResult>(response);
    }

    [Fact]
    public async Task
CreateTarget_BadRequestResultWhenFileFieldIsNull()
    {
        /// Arrange
        TargetDto targetDto = TargetsData.GetNewTargetDto();

        var targetsServiceMock = new Mock<ITargetsService>();
        var controller = new
TargetsController(targetsServiceMock.Object);

        /// Act
        var response = await controller.CreateTarget(targetDto);

        /// Assert
        var target =
Assert.IsType<BadRequestObjectResult>(response);
    }

    /// <summary>
    /// DeleteTarget
    /// </summary>
    [Fact]
    public async Task DeleteTarget_BadRequestResultWhenIdIsNull()
    {
        /// Arrange
        var targetsServiceMock = new Mock<ITargetsService>();
        var targetsControllerMock = new
TargetsController(targetsServiceMock.Object);

        /// Act
        var response = await
targetsControllerMock.DeleteTarget(null);

        /// Assert
        Assert.IsType<BadRequestObjectResult>(response);
    }

```

```

    }

    [Fact]
    public async Task DeleteTarget_BadRequestResultWhenIdIsEmpty()
    {
        /// Arrange
        var targetsServiceMock = new Mock<ITargetsService>();
        var targetsControllerMock = new
TargetsController(targetsServiceMock.Object);

        /// Act
        var response = await
targetsControllerMock.DeleteTarget(Guid.Empty);

        /// Assert
        Assert.IsType<BadRequestObjectResult>(response);
    }

    [Fact]
    public async Task
DeleteTarget_NotFoundResultWhenTargetNotFound()
    {
        /// Arrange
        Guid targetId = new Guid("6c17bd3b-a853-44ea-a193-
359a2bfe5de4");
        var targetsServiceMock = new Mock<ITargetsService>();
        targetsServiceMock.Setup(_ =>
_.DeleteTargetAsync(It.IsAny<Guid>()))
.ReturnsAsync(false);
        var controller = new
TargetsController(targetsServiceMock.Object);

        /// Act
        var response = await controller.DeleteTarget(targetId);

        /// Assert
        Assert.IsType<NotFoundResult>(response);
    }

    [Fact]
    public async Task DeleteTarget_NoContentResultWhenAllOk()
    {
        Guid targetId = new Guid("6c17bd3b-a853-44ea-a193-
359a2bfe5de1");
        var targetsServiceMock = new Mock<ITargetsService>();
        targetsServiceMock.Setup(_ =>
_.DeleteTargetAsync(It.IsAny<Guid>())) .ReturnsAsync(true);
        var controller = new
TargetsController(targetsServiceMock.Object);

        /// Act
        var response = await controller.DeleteTarget(targetId);

        /// Assert
        Assert.IsType<NoContentResult>(response);
    }

```

```

    /// <summary>
    /// UpdateTarget
    /// </summary>
    [Fact]
    public async Task UpdateTarget_BadRequestResultWhenIdIsNull()
    {
        /// Arrange
        var targetsServiceMock = new Mock<ITargetsService>();
        var targetsControllerMock = new
TargetsController(targetsServiceMock.Object);

        /// Act
        var response = await
targetsControllerMock.UpdateTarget(null, TargetsData.GetNewTargetDto());

        /// Assert
        Assert.IsType<BadRequestObjectResult>(response);
    }

    [Fact]
    public async Task UpdateTarget_BadRequestResultWhenIdIsEmpty()
    {
        /// Arrange
        var targetsServiceMock = new Mock<ITargetsService>();
        var targetsControllerMock = new
TargetsController(targetsServiceMock.Object);

        /// Act
        var response = await
targetsControllerMock.UpdateTarget(Guid.Empty,
TargetsData.GetNewTargetDto());

        /// Assert
        Assert.IsType<BadRequestObjectResult>(response);
    }

    [Fact]
    public async Task
UpdateTarget_BadRequestResultWhenTargetIsNull()
    {
        /// Arrange
        Guid targetId = new Guid("6c17bd3b-a853-44ea-a193-
359a2bfe5de1");
        var targetsServiceMock = new Mock<ITargetsService>();
        var targetsControllerMock = new
TargetsController(targetsServiceMock.Object);

        /// Act
        var response = await
targetsControllerMock.UpdateTarget(targetId, null);

        /// Assert
        Assert.IsType<BadRequestObjectResult>(response);
    }

    [Fact]

```

```

        public async Task
UpdateTarget_NotFoundResultWhenTargetNotFound()
    {
        /// Arrange
        Guid targetId = new Guid("6c17bd3b-a853-44ea-a193-
359a2bfe5de4");
        var targetsServiceMock = new Mock<ITargetsService>();
        targetsServiceMock.Setup(_ =>
_.UpdateTargetAsync(It.IsAny<Guid>(), It.IsAny<TargetDto>()))
.ReturnsAsync(false);
        var controller = new
TargetsController(targetsServiceMock.Object);

        /// Act
        var response = await controller.UpdateTarget(targetId,
TargetsData.GetNewTargetDto());

        /// Assert
        Assert.IsType<NotFoundResult>(response);
    }

    [Fact]
    public async Task UpdateTarget_NoContentResultWhenAllOk()
    {
        /// Arrange
        Guid targetId = new Guid("6c17bd3b-a853-44ea-a193-
359a2bfe5de1");
        var targetsServiceMock = new Mock<ITargetsService>();
        targetsServiceMock.Setup(_ =>
_.UpdateTargetAsync(It.IsAny<Guid>(),
It.IsAny<TargetDto>())) .ReturnsAsync(true);

        var controller = new
TargetsController(targetsServiceMock.Object);

        /// Act
        var response = await controller.UpdateTarget(targetId,
TargetsData.GetNewTargetDto());

        /// Assert
        Assert.IsType<NoContentResult>(response);
    }
}
}

```

Листинг Д.2 – TargetsData.cs

```

using entityys.DTO;
using entityys.Models;

namespace questie_backend.Tests.Mockdata
{
    public class TargetsData
    {
        public static List<TargetOutDto> GetTestTargets()
        {

```

```

        return new List<TargetOutDto>
        {
            new TargetOutDto
            {
                Id = new Guid("6c17bd3b-a853-44ea-a193-
359a2bfe5de1"),
                Title = "Target One",
                Description = "Target Description One",
                ImageSrc =
                "https://localhost:5003/Images/Targets/target_one.jpg"
            },
            new TargetOutDto
            {
                Id = new Guid("6c17bd3b-a853-44ea-a193-
359a2bfe5de2"),
                Title = "Target Second",
                Description = "Target Description Second",
                ImageSrc =
                "https://localhost:5003/Images/Targets/target_second.jpg"
            },
            new TargetOutDto
            {
                Id = new Guid("6c17bd3b-a853-44ea-a193-
359a2bfe5de3"),
                Title = "Target Third",
                Description = "Target Description Third",
                ImageSrc =
                "https://localhost:5003/Images/Targets/target_third.jpg"
            },
        };
    }

    public static TargetDto GetNullTitleTargetDto()
    {
        return new TargetDto
        {
            Title = null,
            Description = "Target Description Fourth",
            ImageFile = null,
        };
    }

    public static TargetDto GetNullDescriptionTargetDto()
    {
        return new TargetDto
        {
            Title = "Target Fourth",
            Description = null,
            ImageFile = null,
        };
    }

    public static TargetDto GetNewTargetDto()
    {
        return new TargetDto
        {
            Title = "Target Fourth",

```

```

        Description = "Target Description Fourth",
        ImageFile = null,
    };
}

public static Target GetNewTarget()
{
    return new Target
    {
        Id = new Guid("6c17bd3b-a853-44ea-a193-
359a2bfe5de4"),
        Title = "Target Fourth",
        Description = "Target Description Fourth",
        ImageName = "example.jpg"
    };
}
}
}

```