МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

|  |  |  |
| --- | --- | --- |
| Специальность | 1-40 05 01 | «Информационные системы и технологии» |
| Специализация | 1-40 05 01-03 | «Информационные системы и технологии (издательско-полиграфический комплекс)» |

**Пояснительная записка   
к дипломному проекту**

«Веб-приложение для создания иллюстраций в дополненной реальности»

|  |  |  |
| --- | --- | --- |
| Дипломник |  | Дежемесов М. С. |
|  |  |  |
| Руководитель работы |  | Скребель А. С., ст. преподаватель |
|  |  |  |
| Заведующий кафедрой |  | Смелов В. В., к. т. н., доцент |
|  |  |  |
| Консультанты |  | Скребель А. С., ст. преподаватель |
|  |  |  |
|  |  | Семёнова Л. С., к. экон. н., преподаватель-стажер |
|  |  |  |
| Нормоконтроллер |  | Николайчук А. Н.,  преподаватель-стажер |

|  |  |
| --- | --- |
| Дипломный проект защищен с оценкой |  |

|  |  |  |
| --- | --- | --- |
| Председатель ГЭК |  | В. К. Дюбков, к. т. н., доцент |

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

|  |  |  |
| --- | --- | --- |
| Специальность | 1-40 05 01 | «Информационные системы и технологии» |
| Специализация | 1-40 05 01-03 | «Информационные системы и технологии  (издательско-полиграфический комплекс)» |

УТВЕРЖДАЮ:

Зав. кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

**З А Д А Н И Е**

на дипломный проект

*Дежемесов Макар Сергеевич*

(фамилия, имя, отчество)

1. Тема дипломного проекта: *Веб-приложение для создания иллюстраций в дополненной реальности*

2. Тема утверждена приказом по университету от *17.03. 2023 г. № 155-с*

3. Срок сдачи студентом законченного проекта *05.06. 2023 г.*

4. Исходные данные к проекту.

4.1. Назначение: *приложение для создания иллюстраций в дополненной реальности, корректировка добавления текста и пояснений в дополненной реальности, возможность добавления пояснений в дополненную реальность, поиск иллюстраций других пользователей.*

*4.2.*Основные функциональные возможности: *авторизация и регистрация пользователей, редактирование профиля пользователя, создание иллюстраций в дополненной реальности, просмотр иллюстраций других пользователей, экспортирование иллюстраций, размещение уже существующих иллюстраций в дополненной реальности, возможность настройки инструментов для создания иллюстраций.*

4.3. Пользовательские роли*: Администратор (администрирование учетных записей, модерирование контента); Пользователь (создание иллюстраций, просмотр чужих иллюстраций, публикация новых иллюстраций).*

4.4. Целевая аудитория: *инженеры и люди, работающие в образовании*, *художники и креативные личности, а также люди, ведущие различные социальные сети.*

4.5. Программная платформа: *Unity 2022, C# 9.0, Firebase Firestore, Firebase Authentication, Firebase Analytics, UI Toolkit, Zenject, AR Foundation, Unity MVVM Toolkit.*

5. Содержание расчетно-пояснительной записки.

5.1. *Реферат.*

5.2. *Введение.*

5.3. *Постановка задачи и анализ аналогичных решений*

5.4*. Выбор технологий для разработки.*

5.4. *Проектирование программного модуля.*

5.5. *Реализация программного модуля.*

5.6*. Тестирование программного модуля.*

5.7*. Руководство пользователя.*

5.8. *Технико-экономическое обоснование проекта.*

5.9. *Заключение.*

5.10. *Список использованных источников*

5.11. *Приложение А. Наименование приложения*

5.12. *Приложение Б. Наименование приложения*

5.13. *Приложение В. Наименование приложения*

5.14. *Приложение Г. Наименование приложения*

5.15. *Приложение Д. Наименование приложения*

6. Перечень графического материла:

6.1. Архитектурная схема приложения.

6.2. Логическая схема базы данных.

6.3. Блок-схема алгоритма заказа домена.

6.4. Диаграмма вариантов использования.

7. Консультанты по проекту (работе, с указанием относящихся к ним разделов проекта)

|  |  |
| --- | --- |
| Раздел | Консультант |
|
| *Постановка задачи и анализ анологичных решений*  *Проектирование модуля*  *Реализация модуля*  *Тестирование модуля*  *Руководство программиста* | *А.С. Скребель* |
| *Экономический раздел* | *Л.С Семенова* |

7. Дата выдачи задания \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

8. Календарный план

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Этапы дипломного проектирования | Срок выполнения | Примечание |
|  | *Постановка задачи и анализ аналогичных решений* |  |  |
|  | *Проектирование интернет-сервиса.* |  |  |
|  | *Реализация интернет-сервиса.* |  |  |
|  | *Тестирование интернет-сервиса.* |  |  |
|  | *Технико-экономическое обоснование проекта.* |  |  |
|  | *Рецензирование* |  |  |

Дипломник\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Реферат**

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*0*

*БГТУ 00.00.ПЗ*

Разраб*.*

*Дежемесов М.С.*

Пров*.*

*.*

*Скребель А.С*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*Смелов В.В.*

Реферат

*Лит.*

*Листов*

*1*

74218155, 2023

Пояснительная записка содержит 107 страниц, 76 рисунка, 33 таблицы, 22 источника литературы, 5 приложений.

ВЕБ-ПРИЛОЖЕНИЕ ДЛЯ СОЗДАНИЯ ИЛЛЮСТРАЦИЙ В ДОПОЛНЕННОЙ РЕАЛЬНОСТИ, UNITY 2022, C# 9.0, FIREBASE FIRESTORE, FIREBASE AUTHENTICATION, FIREBASE ANALYTICS, UI TOOLKIT, ZENJECT, AR FOUNDATION, UNITY MVVM TOOLKIT, ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ RIDER

Целью данного дипломного проекта является разработка веб-приложения для создания иллюстраций в дополненной реальности.

Пояснительная записка состоит из введения, семи разделов и заключения.

В первом разделе дипломного проекта приведен аналитический обзор предметной области и постановка задачи по теме дипломного проекта.

Во втором разделе описан выбор технологий для разработки приложения.

В третьем разделе описан процесс проектирования приложения.

В четвертом разделе описан процесс разработки и программной реализации приложения.

В пятом разделе приведено руководство пользователя.

В шестом разделе описано тестирование программного средства.

В седьмом разделе приводятся экономические обоснования разработанного программного модуля.

В заключении представлены итоги дипломного проекта и задачи, которые были решены в ходе разработки программного средства.

Графическая часть представлена в объеме СКОЛЬКО листа А1.

**Abstract**

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*1*

*БГТУ 00.00.ПЗ*

Разраб*.*

*Дежемесов М.С.*

Провер*.*

*.*

*Скребель А.С*

Н. контр.

*Николайчук А.Н.*

Утв.

*Смелов В.В.*

Abstract

*Лит.*

*Листов*

*1*

74218155, 2023

The explanatory note of the diploma project contains 107 pages of explanatory note, 76 illustrations, 33 tables, 22 sources used, 5 appendices.

WEB-APPLICATION FOR CREATING ILLUSTRATIONS IN AUGMENTED REALITY, C#, UNITY 2022, C# 9.0, FIREBASE FIRESTORE, FIREBASE AUTHENTICATION, FIREBASE ANALYTICS, UI TOOLKIT, ZENJECT, AR FOUNDATION, UNITY MVVM TOOLKIT, RIDER INTEGRATED DEVELOPMENT ENVIRONMENT

The purpose of this diploma project is to develop a web-application for creating illustrations in augmented reality.

The explanatory note consists of an introduction, seven sections and a conclusion.

The first section of the diploma project provides an analytical overview of the subject area and a statement of the problem on the topic of the diploma project.

The second section describes the choice of technologies for application development.

The third section describes the application design process.

The fourth section describes the process of application development and software implementation.

The fifth section contains the user's guide.

The sixth section describes the testing of the software.

The seventh section provides the economic justification of the developed software module.

In conclusion, the results of the graduation project and the tasks that were solved during the development of the software are presented.

The graphic part is presented in the volume of HOW MANY sheets A1.

Содержание

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*2*

*БГТУ 00.00.ПЗ*

Разраб*.*

*Дежемесов М.С.*

Провер*.*

*.*

*Скребель А.С*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*СмеловВ.В.*

Содержание

*Лит.*

*Листов*

*2*

74218155, 2023

74218155

[Содержание 5](#_Toc135070806)

[Введение 7](#_Toc135070807)

[1 Постановка задачи и анализ аналогичных решений 8](#_Toc135070808)

[1.1 Анализ и сравнительный обзор аналогов 8](#_Toc135070809)

[1.1.1 Hoster 8](#_Toc135070810)

[1.1.2 HostFly 11](#_Toc135070811)

[1.1.3 ActiveCloud 14](#_Toc135070812)

[1.2 Описание проекта Domain.by 15](#_Toc135070813)

[1.3 Постановка задачи 16](#_Toc135070814)

[1.4 Выводы по разделу 17](#_Toc135070815)

[2 Выбор технологий для разработки 18](#_Toc135070816)

[2.1 Выбор языка программирования 18](#_Toc135070817)

[2.2 Выбор средств разработки 18](#_Toc135070818)

[2.3 Сервер IIS Express 20](#_Toc135070819)

[2.4 База данных MS SQL Server 21](#_Toc135070820)

[2.6 Выбор технологий и библиотек 21](#_Toc135070821)

[2.6.1 Технология Entity Framework 21](#_Toc135070822)

[2.6.2 Технология Linq2SQL 22](#_Toc135070823)

[2.6.3 Библиотека Fluent Validation 23](#_Toc135070824)

[1.6.4 Технология Razor Pages 23](#_Toc135070825)

[2.6.5 Технология Bootstrap 24](#_Toc135070826)

[2.7 Выводы по разделу 24](#_Toc135070827)

[3 Проектирование программного модуля 25](#_Toc135070828)

[3.1 Архитектура системы 25](#_Toc135070829)

[3.2 Проектирование серверной части 26](#_Toc135070830)

[3.3 Проектирование клиентской части 27](#_Toc135070831)

[3.4 Проектирование базы данных 27](#_Toc135070832)

[3.5 Выводы по разделу 35](#_Toc135070833)

[4 Реализация программного модуля 36](#_Toc135070834)

[4.1 Задачи программного модуля 36](#_Toc135070835)

[4.2 Анализ и рефакторинг уже существующего кода проекта 36](#_Toc135070836)

[4.3 Файловая структура 37](#_Toc135070837)

[4.4 Уровень доступа к данным 38](#_Toc135070838)

[4.5 Уровень бизнес-логики 40](#_Toc135070839)

[4.6 Уровень представления 45](#_Toc135070840)

[4.7 Выводы по разделу 51](#_Toc135070841)

[5 Тестирование программного модуля 55](#_Toc135070842)

[5.1 Ручное тестирование 55](#_Toc135070843)

[5.2 Тестирование валидации 57](#_Toc135070844)

[5.3 Автоматическое тестирование 60](#_Toc135070845)

[5.4 Выводы по разделу 61](#_Toc135070846)

[6 Руководство пользователя 62](#_Toc135070847)

[6.1 Общие сведения 62](#_Toc135070848)

[6.2 Роль «Пользователь» 62](#_Toc135070849)

[6.3 Роль «Администратор» 67](#_Toc135070850)

[6.6 Выводы по разделу 77](#_Toc135070851)

[7 Технико-экономическое обоснование проекта 78](#_Toc135070852)

[7.1 Общая характеристика разрабатываемого программного средства 78](#_Toc135070853)

[7.2 Исходные данные для проведения расчетов и маркетинговый анализ 78](#_Toc135070854)

[7.3 Обоснование цены программного средства 79](#_Toc135070855)

[7.3.1 Расчёт затрат рабочего времени на разработку программного средства 79](#_Toc135070856)

[7.3.2 Расчёт основной заработной платы 80](#_Toc135070857)

[7.3.2 Расчёт дополнительной заработной платы 80](#_Toc135070858)

[7.3.3 Отчисления в Фонд социальной защиты населения и Белгосстрах 81](#_Toc135070859)

[7.3.4 Расчёт суммы прочих прямых затрат 81](#_Toc135070860)

[7.3.4 Расчёт суммы накладных расходов 81](#_Toc135070861)

[7.3.4 Сумма расходов на разработку программного средства 82](#_Toc135070862)

[7.3.5 Расходы на реализацию 82](#_Toc135070863)

[7.3.5 Расчет полной себестоимости 82](#_Toc135070864)

[7.3.6 Определение цены, оценка эффективности 83](#_Toc135070865)

[7.4 Вывод по разделу 84](#_Toc135070866)

[Заключение ПЕРЕПИСАТЬ 86](#_Toc135070867)

[Список использованных источников ДОБАВИТЬ ИСТОЧНИКОВ 87](#_Toc135070868)

[ПРИЛОЖЕНИЕ А 88](#_Toc135070869)

[ПРИЛОЖЕНИЕ Б 89](#_Toc135070870)

[ПРИЛОЖЕНИЕ В 90](#_Toc135070871)

[ПРИЛОЖЕНИЕ Г 91](#_Toc135070872)

[ПРИЛОЖЕНИЕ Д 92](#_Toc135070873)

[ПРИЛОЖЕНИЕ Е 103](#_Toc135070874)

[ПРИЛОЖЕНИЕ Ж 105](#_Toc135070875)

Введение

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*2*

*БГТУ 00.00.ПЗ*

Разраб*.*

*Дежемесов М.С.*

Провер*.*

*.*

*Скребель А.С*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*СмеловВ.В.*

Введение

*Лит.*

*Листов*

*1*

74218155, 2023

Концепция дополненной реальности (AR) получила широкое распространение в последние годы, став частью повседневной жизни людей. Многие компании и проекты используют AR для улучшения пользовательских интерфейсов, повышения вовлеченности пользователя и создания новых возможностей для обучения, развлечения и творчества. В этой области появляются новые технологии и инструменты, которые позволяют разработчикам создавать уникальные приложения и продукты для пользователей.

Целью дипломного проекта является разработка веб-приложения для создания иллюстраций в дополненной реальности. Оно будет предоставлять пользователям возможность создавать свои собственные иллюстрации, используя различные инструменты и функции, доступные в приложении. Пользователи смогут использовать камеру своего устройства для визуализации виртуальных объектов, наложенных на реальный мир, и добавлять к ним собственные элементы.

Приложение будет иметь широкий спектр применений, включая использование в обучении, развлечении, дизайне и инженерии. Оно может быть полезно для дизайнеров, художников и творческих людей, которые хотят создавать уникальные иллюстрации в AR, а также для обычных пользователей, которые просто хотят попробовать что-то новое и интересное.

Кроме того, приложение будет полезным для инженеров и других профессионалов, которые работают с технической документацией и нуждаются в возможности создавать пометки в реальности. Приложение поможет им улучшить качество и точность работы, а также повысить производительность и эффективность.

Целью данного проекта является создание инновационного приложения, которое будет позволять пользователям создавать уникальные иллюстрации в дополненной реальности. В рамках работы была разработана платформа со спектром функций и инструментов для создания иллюстраций.

Для достижения цели проекта сформулированы следующие задачи:

* выполнение обзора аналогичных решений;
* проектирование веб-приложения;
* разработка веб-приложения;
* проведение тестирования веб-приложения;
* составление руководства пользователя;
* расчет экономической составляющей проекта.

1 Постановка задачи и анализ аналогичных решений

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*2*

*БГТУ 01.00.ПЗ*

Разраб*.*

*Дежемесов М.С.*

Провер*.*

*.*

*Блинова Е.А*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*СмеловВ.В.*

*Лит.*

*Листов*

*11*

74218155, 2023

*1 Постановка задачи и анализ аналогичных решений*

1.1 Анализ и сравнительный обзор аналогов

Для разработки актуального веб приложения необходимо ознакомиться с уже существующими аналогами. Это необходимо для того, чтобы понять, в каком функционале нуждаются пользователи, как он устроен, а также выделить как сильные, так и слабые стороны приложения.

На сегодняшний день существует достаточно мало веб-приложений, позволяющих создавать иллюстрации в дополненной реальности. Рассмотрим некоторые из аналогов и выявим сильнее и слабые стороны. Для обзора аналогов выл выбран следующий список веб-приложений:

– Tilt Brush [1];

– Adobe Medium [2];

– eDrawings Viewer [3].

### 1.1.1 Приложение Tilt brush

Tilt Brush является приложением для виртуальной реальности, разработанное Google, которое позволяет пользователям создавать 3D-иллюстрации в дополненной реальности. Приложение доступно на платформах Oculus Rift, HTC Vive, Valve Index, а также на Oculus Quest в режиме Link.

В этом приложении можно создавать трехмерные иллюстрации. Пользователи могут использовать контроллеры VR, чтобы рисовать объекты в 3D-пространстве. Приложение поддерживает различные типы кистей и цветов, а также позволяет пользователям изменять размер и форму объектов (рисунок 1.1).



Рисунок 1.1 – Скриншот с главной страницы компании *Hoster.by*

Так же, данное приложение позволяет пользователю использовать музыку и звуковые эффекты в свои 3D-картинки. Это может создавать уникальные эффекты и дополнительную атмосферу в виртуальном пространстве. Для этого пользователь должен выбрать специальный инструмент, который называется «Waveform». Этот инструмент позволяет создавать 3д иллюстрации, которые будут реагировать на ритм играющей музыки, что позволяет создавать уникальные эффекты и добавляет динамичности различным иллюстрациям.

В приложении так же имеется интеграция с другими приложениями дополненной реальности. Tilt Brush может использоваться в сочетании с другими приложениями дополненной реальности, такими как Google Blocks, что позволяет создавать более сложные и детализированные трехмерные модели (рисунок 1.2).



Рисунок 1.2 – Интеграция Tilt Brush и Google Blocks

Кроме того, Tilt Brush поддерживает экспорт и импорт трехмерных моделей в различных форматах, таких как OBJ и FBX, что позволяет пользователям использовать свои модели в других приложениях, где можно вносить дополнительные модификации, либо использовать созданные модели в проектах напрямую. Так же, приложение Tilt Brush позволяет импортировать существующие трехмерные модели.

Tilt Brush предоставляет возможность пользователям обмениваться своими 3D-картинками с другими пользователями, что позволяет создавать сообщества и делиться своими творениями.

В целом, Tilt Brush предоставляет широкий спектр инструментов и возможностей для создания 3D-картинок в VR. Это полезный инструмент для профессиональных художников и дизайнеров, а также для тех, кто интересуется VR и хочет попробовать свои силы в создании 3D-контента.

Из основных минусов можно выделить требование к пользователю иметь шлем для дополненной реальности, а также достаточно мощный ПК, что сильно снижает потенциальное количество пользователей.

### Приложение Adobe Medium

Следующим приложением для сравнения является Adobe Medium. Данное приложение используется для создания виртуальных объектов и сред. Оно было разработано для использования с устройствами виртуальной реальности, такими как Oculus Rift, Oculus Quest и HTC Vive. Приложение позволяет создавать 3D-модели, которые могут быть использованы в приложениях виртуальной реальности, играх, фильмах и других проектах (рисунок 1.3).



Рисунок 1.3 – Приложение Adobe Medium

Функционал приложения Adobe Medium включает в себя множество инструментов для создания и редактирования трехмерных моделей. Так же, приложение имеет инструменты для моделирования, которые позволяют создавать различные формы объектов, настраивать разнообразные поверхности, а также настраивать параметры текстур и освещения.

Кроме инструментов для трехмерных моделей, данное приложение имеет в себе инструменты для рисования, которые позволяют создавать трехмерные рисунки, аналогичные рисованию в двухмерном пространстве.

В приложение так же имеются инструменты для создания скульптур, которые позволяют создавать детализированные трехмерные модели, добавляя детали и выдавливая формы из различных материалов. Панель для выбора инструментов изображена на рисунке 1.4.



Рисунок 1.4 – Приложение Adobe Medium

Так же, в приложении имеются инструменты для работы с цветом, которые позволяют настраивать цвета и оттенки объектов, создавать текстуры и наносить рисунки, а также инструменты для работы с масками и слоями, которые позволяют создавать сложные многослойные объекты, настраивать прозрачность и смешивать цвета.

Кроме того, Adobe Medium имеет широкий спектр настроек, которые позволяют пользователю управлять качеством изображения и производительностью приложения. В приложении есть также функционал социального взаимодействия, позволяющий пользователям делиться своими проектами и работать вместе над созданием сцен виртуальной реальности.

В целом, Adobe Medium – это мощный инструмент для создания 3D-моделей и сцен в виртуальной реальности. Он предлагает широкий спектр инструментов и настроек, которые позволяют пользователю создавать качественные 3D-модели, а также работать вместе с другими пользователей и делиться своими проектами.

Как и в прошлом приложении, основной минусом Adobe Medium является использование шлема виртуальной реальности и доступ к мощному ПК.

### 1.1.3 Приложение eDrawings Viewer

Первые два приложения используются только для творческих целей. Следующее же приложение было разработано в основном для инженеров и архитекторов. eDrawings Viewer является бесплатным приложением для просмотра и обмена 2D и 3D моделями. Оно позволяет пользователям открывать и просматривать файлы в различных форматах, включая DWG, DXF, SolidWorks, CATIA, Pro/ENGINEER и многие другие (рисунок 1.5).

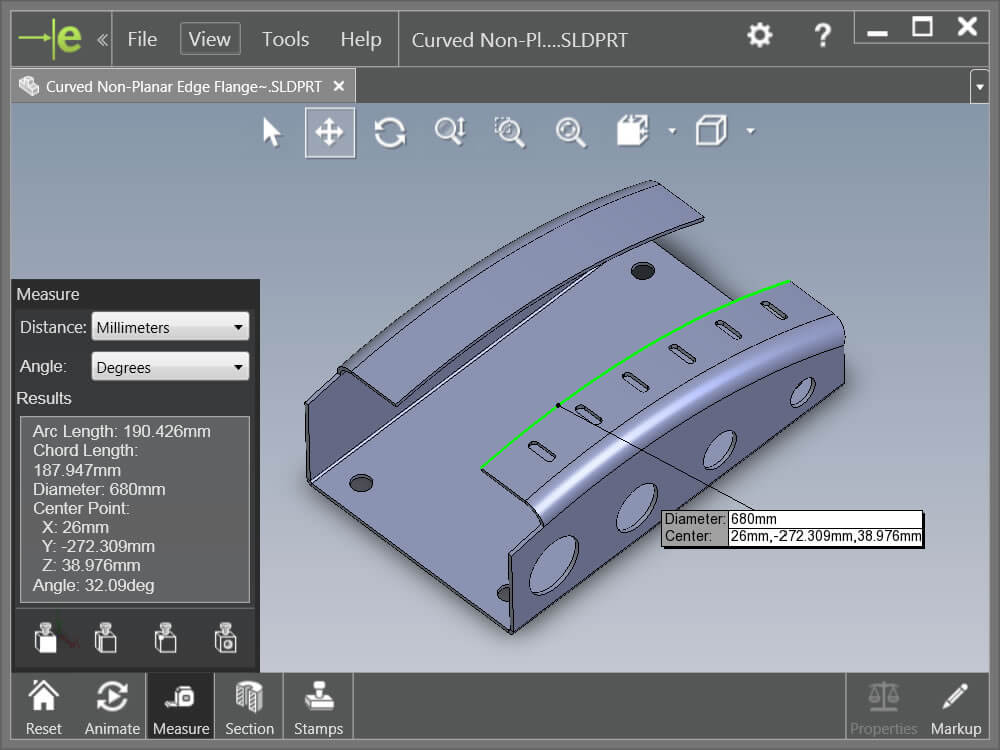


Рисунок 1.5 – Приложение eDrawings Viewer

Приложение позволяет пользователю просматривать и масштабировать двухмерные и трехмерные модели, приближать и отдалять, поворачивать их вокруг всех осей, а также устанавливать различные точки обзора и камеры, что позволяет лучше рассмотреть трехмерную модель, а также различные мелкие детали, которые пользователь мог упустить при просмотре с экрана монитора.

Так же, в приложении можно найти различные инструменты измерения: eDrawings Viewer имеет инструменты для измерения размеров и расстояний в двухмерных и трехмерных моделях, что позволяет пользователям более точно оценивать размеры и соотношения в модели.

Кроме того, eDrawings Viewer позволяет пользователям добавлять комментарии, маркеры, стрелки и другие элементы на модели, что упрощает коммуникацию и обмен информацией между участниками проекта.

В приложении так же пользователи могут экспортировать созданные модели в различных форматах, таких как BMP, JPEG, PNG, TIF, STL и другие, что позволяет использовать их в других приложениях и проектах. Кроме того, eDrawings Viewer также имеет функцию облачного хранилища, которая позволяет пользователям загружать и синхронизировать свои модели в облаке, а так же обмениваться ими с другими участниками проекта.

Пользователи могут использовать приложение на устройствах, поддерживающих AR, таких как смартфоны и планшеты с операционной системой iOS или Android. Для этого нужно скачать специальное приложение eDrawings AR и установить его на свое устройство (рисунок 1.6).



Рисунок 1.6 – Приложение eDrawings Viewer

В целом, eDrawings Viewer – это мощное и удобное приложение для просмотра и обмена 2D и 3D моделями, которое может быть использовано в различных областях, включая проектирование, машиностроение, архитектуру, промышленное производство и другие области.

В отличие от прошлых аналогов, приложение eDrawings Viewer может быть запущено как в виртуальной, так и в дополненной реальности. Из недостатков можно выделить то, что eDrawings Viewer не позволяет пользователю как-то модифицировать изображение, и имеет в себе только функционал просмотра моделей.

### 1.2 Анализ прототипов

Сравнив прототипы приложений, в случае первых двух приложений, можно отметить схожесть в их функциональности и интерфейсе. В каждом рассмотренном аналоге есть свои достоинства и недостатки. Наиболее важным является простота интерфейса и информативность. Из функциональных достоинств можно отметить возможность создания иллюстраций различными инструментами, экспортирование созданных иллюстраций, а также простые в управлении визуальные инструменты.

Из основных минусов всех вышеперечисленных приложений можно выделить требование к доступу к шлему дополненной реальности, а также к производительному персональному компьютеру, который сможет выполнять приложение и отображать изображение в VR шлем.

Оптимальным решением разработки собственного продукта является выбор отдельных возможностей каждого из конкурентов с последующими устранениями их недостатков, а также добавлением собственных возможностей и правил.

### 1.3 Постановка задачи

После анализа прототипов, выявления их сильных и слабых сторон, был сформулирован список функционала, который должен присутствовать в приложении:

* авторизация и регистрация пользователей;
* редактирование профиля пользователя;
* создание иллюстраций в дополненной реальности;
* просмотр иллюстраций других пользователей;
* экспортирование иллюстраций;
* размещение уже существующих иллюстраций в дополненной реальности;
* возможность настройки инструментов для создания иллюстраций.

Приложение будет иметь широкий спектр применений, включая использование в инженерии, обучении, развлечении и дизайне.

Целью данного проекта является создание инновационного приложения, которое будет позволять пользователям создавать уникальные иллюстрации в дополненной реальности. В рамках работы была разработана платформа со спектром функций и инструментов для создания иллюстраций.

### 1.4 Выводы по разделу

Данный раздел содержит информацию о аналогах разрабатываемого программного средства, был приведен кратки обзор к каждому из них, рассмотрены их возможности и недостатки, выделены плюсы и минусы.

В итоге был сделан вывод, что в приложении важен выбор различных настроек кистей, фигур, и прочих визуальных инструментов, а также набор инструментов для пользователей, которые будут использовать приложение в инженерных целях.

Из основных минусов всех вышеперечисленных приложений можно выделить требование к доступу к шлему дополненной реальности, а также к производительному персональному компьютеру, который сможет выполнять приложение и отображать изображение в VR шлем.

В результате было принято решение на создание собственной разработки в рамках дипломного проекта, которая призвана добавить функционал создания иллюстраций в дополненной реальности. Системные требования для запуска приложений в дополненной реальности достаточно низкие, что позволяет куда большему количеству пользователей использовать приложение.

Обзор аналогов, анализ предметной области помогли в определении используемых технологий. Для разработки серверной части и базы данных приложения была выбраны технологии Firebase. Клиентская часть использует Unity.

2 Выбор технологий для разработки

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*2*

*БГТУ 02.00.ПЗ*

Разраб*.*

*Дежемесов М.С.*

Провер*.*

*.*

*Скребель А.С*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*СмеловВ.В.*

*Лит.*

*Листов*

*7*

74218155, 2023

*2 Выбор технологий для разработки*

В данном подразделе будет обоснован выбор языка программирования, среды разработки и платформы, которые будут использованы в разработке приложения.

2.1 Выбор языка программирования

Существует большое количество языков программирования, с помощью которых можно разработать клиент-серверное приложение. Для разработки модуля был выбран язык программирования *C#,* так как весь проект *Domain.by* разрабатывался на нем [5].

Язык *С#* является высокоуровневым объектно-ориентированным языком программирования, поэтому он подходит для быстрого конструирования различных компонентов – от высокоуровневой бизнес-логики до системных приложений, использующих низкоуровневый код. Также следует отметить, что при помощи *С#* можно создавать различные веб-приложения. С помощью простых встроенных конструкций языка существует возможность переводить созданные компоненты в веб-сервисы, к которым можно будет обращаться из сети Интернет, используя любой язык на любой операционной системе. Также можно создать полноценные веб-приложения, которые сразу предоставляют интерфейс пользователю. Удобные методы для разработки веб-приложений позволяют программистам, владеющим навыками объектно-ориентированного программирования, быстро освоиться в разработке веб-приложений.

2.2 Выбор средств разработки

Unity – это кроссплатформенная среда разработки приложений и игр, которая позволяет создавать игры для мобильных устройств, компьютеров, игровых консолей, виртуальной и дополненной реальности и других платформ. В качестве языка для написания скриптов, Unity использует C# версии 9.0 на базе Mono [5].

Платформа Unity является популярным инструментом для разработки приложений, используемым многими крупными компаниями в игровой и развлекательной отраслях, а также для образовательных и научных целей. Unity имеет активное сообщество разработчиков и множество онлайн-ресурсов, которые помогают новичкам начать работу с платформой и продвинутым разработчикам улучшить свои навыки. Он также имеет набор инструментов для создания интерактивных GUI, поэтому разработчики могут создавать пользовательские элементы управления и взаимодействовать с пользователями на многих платформах.

Unity использует 2 бэкенда для написания скриптов – Mono и IL2CPP. IL2CPP (Intermediate Language To C++) – это компилятор, который используется в Unity для преобразования байт-кода .NET Framework и C# скриптов в нативный код C++, который может быть выполнен на разных платформах. IL2CPP является альтернативой компиляции Just-In-Time (JIT), которую использует стандартный механизм компиляции в .NET Framework. Одним из главных преимуществ IL2CPP является то, что он позволяет улучшить произ-водительность игр и приложений, уменьшив время загрузки и увеличив количество кадров в секунду (FPS). Также IL2CPP позволяет разработчикам экспортировать код из Unity в другие среды разработки, такие как Xcode для iOS или Android Studio для Android. В целом, использование IL2CPP в Unity мо-жет помочь улучшить производительность игр и приложений, особенно на мобильных устройствах и других платформах с ограниченными ресурсами.

В Unity так же реализован свой компилятор под названием Burst, который позволяет ускорить выполнение кода на центральном процессоре. Он работает как дополнение к компилятору C# между этапом JIT компиляции и выполнением кода. Burst использует оптимизированный механизм генерации кода для процессора, что уменьшает нагрузку и значительно повышает производительность приложения.

Burst предлагает мощные оптимизации для ускорения кода C#, такие, как межпоточное размещение памяти, оптимизация циклов, слияние циклов, SIMD (Single Instruction, Multiple Data) и т.д. Благодаря этим оптимизациям, работа приложения может быть ускорена до 10 раз.

Unity так же имеет в себе систему для упрощения работы с многопоточностью, которая называется Unity Jobs System. Она представляет собой набор инструментов для управления задачами, позволяя выполнению кода на множестве потоков и ядер центрального процессора. Это повышает производительность приложения и уменьшает нагрузку на CPU.

Unity Jobs System использует два типа задач: Jobs и Functions. Jobs – это задачи, которые могут быть выполнены параллельно на нескольких потоках процессора, а Functions – это обычные задачи, которые могут быть выполнены только на одном рабочем потоке процессора.

С помощью Jobs System, можно распределять задания на множество потоков, используя метод Thread-Local Data (TLD). TLD позволяет сохранять данные в локальных переменных и кэшах, что ускоряет их выполнение. Кроме того, Jobs System использует Job Scheduling, что позволяет эффективно планировать исполнение задач на потоках.

Unity Jobs System также содержит набор инструментов для мониторинга выполнения задач, что помогает разрабатывать более эффективный код. Программист может использовать профилировщик Unity Profiler для анализа задач и проверки использования ресурсов и времени выполнения.

Кроме того, Unity имеет широкие возможности работы с графикой, что позволяет создавать качественные и привлекательные AR-сцены. Unity также имеет широкий набор ресурсов, таких как текстуры, модели и эффекты, которые могут быть использованы в приложениях с дополненной реальностью.

В целом, Unity предоставляет мощный и гибкий инструментарий для создания высококачественных AR-приложений, и поэтому является одним из лучших выборов для разработки AR-приложений.

2.3 Firebase

Firebase – это облачная платформа, созданная для упрощения разработки мобильных и многоуровневых приложений. Она была приобретена Google в 2014 году и включает в себя все инструменты, необходимые для построения полноценных приложений.

Firebase использует технологии облачных вычислений и распределенных баз данных, что позволяет увеличить отказоустойчивость, масштабируемость и стабильность приложения. Благодаря этому разработчики могут быть уверены в надежности и стабильности своего продукта.

Одной из ключевых особенностей Firebase является возможность работать в реальном времени, что позволяет своевременно обновлять информацию на всех устройствах. Благодаря этому пользователи получают наиболее актуальную информацию и мгновенный доступ к важным данным.

Firebase также позволяет интегрировать в приложение облачное хранение, что значительно упрощает разработку и позволяет сократить время на написание собственного механизма хранения данных. Благодаря этому приложения могут быть более масштабируемыми, быстрыми и стабильными.

Один из самых популярных сервисов данного сервиса является Firestore. Это облачная NoSQL база данных, позволяющая сохранять и получать данные в режиме реального времени. Firestore работает на человеческих структурах данных, что позволяет сохранять и обрабатывать данные в формате JSON. База данных Firestore высокопроизводительна и отлично масштабируется, позволяет использовать мультипользовательскую блокировку и обеспечивает безопасный доступ к данным.

Еще один популярный сервис – это Firebase Authentication. Он предоставляет функционал аутентификации и авторизации для приложения. Firebase Authentication позволяет регистрировать пользователей с помощью email, пароля, Google, и других систем аутентификации. Кроме того, данный сервис предоставляет пользователю встроенный функционал для сброса и восстановления пароля.

Firebase облегчает разработку кроссплатформенных приложений и позволяет разрабатывать приложения с быстро изменяющимися данными, создавать мобильные приложения в режиме онлайн, администрировать пользователей и многое другое. Поэтому Firebase является отличным выбором для приложений, разработанных на платформе Unity.

2.4 Выбор технологий и библиотек

### 2.4.1 Технология Unity UI Toolkit

Unity UI Toolkit – это новый инструмент для разработки пользовательских интерфейсов для игр и приложений, доступный с версии Unity 2021.1. UI Toolkit представляет собой редизайн Unity UI, который предлагает более гибкие функциональные возможности для UI разработки [7].

UI Toolkit предлагает больший контроль над дизайном элементов интерфейса и лучшую производительность. Он использует новый движок рендера, который обеспечивает улучшенную работу с текстом, шрифтом и цветом. Он также позволяет быстро и легко настраивать элементы интерфейса, такие как кнопки, текстовые поля, слайдеры и т. д.

Основные характеристики Unity UI Toolkit включают гибкую сетку размещения, улучшенную настройку стилей и тем, расширенные возможности анимации и поддержку многоязыковых приложений. С помощью Unity UI Toolkit можно создавать и редактировать пользовательские элементы интерфейса с помощью языка разметки UXML, который очень схож с XML. Кроме того, разработчик может настраивать внешнйи вид элементов при помощи USS (Unity Style Sheets), который очень похож на CSS. USS позволяет создавать и изменять стили элементов интерфейса, такие как цвета, шрифты, размеры и положение.

UI Toolkit также обладает программным интерфейсом, позволяющим разработчикам легко и просто создавать пользовательские элементы интерфейса. UI Toolkit поддерживает популярные инструменты, такие как Visual Studio и Rider.

Наконец, Unity UI Toolkit предоставляет возможность легко переносить пользовательские интерфейсы между различными платформами, что делает его идеальным инструментом для создания игр и приложений, которые работают на различных устройствах и платформах.

### 2.4.2 Unity AR Foundation

Unity AR Foundation – это набор инструментов для разработки приложений дополненной реальности (AR) на Unity. Он предоставляет универсальное API, которое позволяет создавать AR-приложения, которые могут работать на различных устройствах и платформах, таких как iOS, Android, ARCore и ARKit [8].

AR Foundation обеспечивает интеграцию различных AR-технологий, таких как механика определения местоположения и отслеживание поверхностей, и предоставляет единое API, которое позволяет разработчикам создавать AR-приложения для широкого спектра устройств.

К основному функционалу AR Foundation относятся:

* Трекинг объектов в реальном времени, используя машинное зрение и дополненную реальность
* Распознавание плоскостей и поверхностей, таких как столы или полы, для установки объектов AR
* Работа с графическими элементами и пользовательским интерфейсом (UI)
* Взаимодействие со средой AR посредством жестов и движений
* Работа с видео и звуком для создания более реалистичной среды AR

AR Foundation обеспечивает простой доступ к функциям ARKit, ARCore и Vuforia в Unity, что позволяет разработчикам сосредоточиться на создании AR-сцен и объектов, не углубляясь в технические детали. Так же, AR Foundation поддерживает работу с Android, iOS и другими операционными системами.

Кроме того, Unity AR Foundation дает разработчикам возможность настраивать и настраивать уровень масштабируемости и интерактивности, что обеспечивает гибкость и многофункциональность AR-приложений.

AR Foundation предоставляет разработчикам универсальный набор инструментов и API для создания AR-приложений на Unity. Он облегчает процесс разработки и позволяет разработчикам сосредоточиться на создании креативных и инновационных AR-приложений для различных устройств и платформ.

### 2.4.3 Фреймворк Zenject

Zenject – это библиотека внедрения зависимостей (DI) для Unity. Она предоставляет мощный и гибкий механизм для управления зависимостями в проектах, что облегчает разработку масштабируемых и легко поддерживаемых приложений [9].

Dependency Injection (DI) – это паттерн проектирования, который позволяет управлять зависимостями между объектами в приложении. Он заключается в том, что объекты не создают свои зависимости самостоятельно, а получают их отвне, через внешние источники, такие как контейнеры DI или фабрики.

DI позволяет создавать более гибкую и масштабируемую архитектуру приложения. Он упрощает тестирование и отладку, так как каждая зависимость может быть заменена на имитацию или заглушку для удобного тестирования. Кроме того, DI позволяет лучше разделять ответственности между объектами, уменьшая связность и увеличивая модульность кода.

DI может быть реализован различными способами, например, через конструкторы, свойства или методы объектов, которым нужны зависимости. Важно, чтобы все зависимости были определены внешними источниками и передавались в объекты с помощью механизма DI.

Общим принципом DI является инверсия управления (Inversion of Control, IoC), где контроль за созданием объектов переносится из классов во внешние источники, которые управляют созданием объектов и их зависимостей.

Zenject предоставляет простой и интуитивно понятный API для внедрения зависимостей. Он позволяет легко настроить и управлять зависимостями в проекте.

Так же, Zenject обеспечивает гибкость и масштабируемость вашего проекта, что позволяет легко добавлять, изменять и удалять зависимости без необходимости изменения большого количества кода.

Фреймворк Zenject поддерживает не только Mono несколько платформ и языков, включая Mono и IL2CPP. Кроме того, Zenject предоставляет механизм для создания и тестирования классов с внедрением зависимостей, что делает тестирование вашего проекта более эффективным и удобным.

Unity Zenject также предоставляет множество других функций, таких как автоматическое создание объектов, жизненный цикл объектов, события жизненного цикла и другие, которые помогают упростить процесс разработки и обеспечивают более эффективную работу в вашем проекте Unity.

2.4.4 Unity MVVM Toolkit

MVVM (Model-View-ViewModel) – это шаблон проектирования, который используется для создания пользовательского интерфейса (UI) в приложении. Он основывается на идеях разделения интерфейса пользователя и логики приложения на два отдельных компонента, а ViewModel обеспечивает связь между ними [10].

Модель (Model) – это состояние данных, с которыми работает приложение. Представление (View) представляет пользовательский интерфейс, который отображает информацию из модели данных или принимает ввод пользователя. Модель представления (ViewModel) является прослойкой между представлением и моделью данных, которая обрабатывает логику и обновляет данные модели представления, а также обрабатывает действия пользователя и взаимодействует с моделью данных. MVVM позволяет разделить логику приложения между различными компонентами, что упрощает разработку и позволяет создавать более гибкие и масштабируемые приложения. Кроме того, использование данного паттерна позволяет легко тестировать каждый из компонентов приложения отдельно, что улучшает качество и надежность кода.

Паттерн MVVM является одним из наиболее популярных архитектурных паттернов в разработке приложений для различных платформ, включая Web, мобильные устройства и настольные ПК. Он обеспечивает множество преимуществ в разработке ПО, в том числе улучшение качества кода, повышение производительности и увеличение гибкости и масштабируемости приложения.

В отличие от простых приложений с пользовательским интерфейсом, которые обрабатывают события напрямую в коде, MVVM предлагает разделить пользовательский интерфейс и логику обработки событий.

Совместное использование Unity UI Toolkit и MVVM позволяет разделить логику приложения и пользовательский интерфейс, улучшить поддержку кода и упростить тестирование. Unity UI Toolkit может использоваться для создания пользовательского интерфейса, в то время как MVVM может быть использован для разделения логики и данных. Модель представления может обрабатывать взаимодействие пользователя с пользовательским интерфейсом.

Применение MVVM позволяет лучше организовать код проекта, повысить его читабельность и упростить поддержку проекта, поскольку логика обработки событий находится в ViewModel, а не в пользовательском интерфейсе. Это также позволяет легче тестировать код, поскольку ViewModel может быть легко тестирована без необходимости отображения пользовательского интерфейса.

Кроме того, привязки данных (Data Binding) в MVVM позволяют автоматически отображать изменения данных в пользовательском интерфейсе без необходимости вручную обновлять элементы пользовательского интерфейса. Это упрощает процесс отображения данных и уменьшает количество ошибок, связанных с ручным обновлением пользовательского интерфейса.

### 2.4.5 Библиотека DOTween

Unity DOTween – это библиотека анимации для Unity, которая позволяет легко и интуитивно создавать сложные анимации для приложений и игр. Она использует концепцию твининга, которая является более наглядной и понятной, чем использование корутин или асинхронных функций.

DOTween обладает широким спектром возможностей для анимации, включая перемещение, вращение, масштабирование, изменение прозрачности, цвета и т.д. Она также может быть использована для анимации UI элементов и системы частиц.

Основными преимуществами DOTween являются его простота использования и гибкость. DOTween также обладает функционалом для выполнения серий анимаций и создания сложных анимационных составляющих.

Кроме того, DOTween обеспечивает быструю и плавную работу, благодаря оптимизации производительности. Это позволяет программистам создавать высококачественные анимации, не теряя при этом производительность приложения.

DOTween является проектом с открытым исходным кодом, и бесплатно распространяется на платформе Unity. У нее так же имеется хорошая документация, в которой описываются все возможности и примеры использования библиотеки. DOTween – это один из лучших инструментов для создания анимации в Unity, и ее использование может значительно упростить работу программиста при создании высококачественной анимации в их проектах.

2. Выводы по разделу

В результате обзора технических средств и анализа поставленной задачи для реализации дипломного проекта была выбрана платформа *Unity 2022*, так как данная платформа предоставляет весь нужный функционал для создания приложения.

В качестве сервера была выбрана платформа Firebase. В основу работы с базой данных была выбрана технология *Firebase Firestore*.

Основой архитектуры проекта был выбран паттерн проектирования *MVVM*, совмещенный с DI фреймворком Zenject. При разработке представлений будут использованы технологии *Unity UI Toolkit* и *DOTWeen*, позволяющие создавать гибкий, понятный и расширяемый пользовательский интерфейс.

3 Проектирование программного модуля

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*2*

*БГТУ 03.00.ПЗ*

Разраб*.*

*Дежемесов М.С.*

Провер*.*

*.*

*Скребель А.С*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*СмеловВ.В.*

*Лит.*

*Листов*

*11*

74218155, 2023

*3 Проектирование  
программного модуля*

Хорошая архитектура – один из важных факторов удачного проекта. Она делает процесс разработки и сопровождения программы более эффективным. Программу с хорошей архитектурой проще расширять и изменять, а также тестировать, отлаживать и снизить порог понимания для новых разработчиков.

Хорошая архитектура должна следовать следующим критериям:

– эффективность системы. В первую очередь программа должна решать поставленные задачи и выполнять их в различных условиях. Под этот критерий можно отнести такие характеристики, как надежность, безопасность, способность справляться с увеличением нагрузок, отказоустойчивость, производительность;

– гибкость системы. Любое приложение приходится менять со временем – изменяются поставленные задачи, добавляются новые, находятся проблемы или уязвимости в программном средстве. Чем быстрее и удобнее можно внести изменения в существующую функциональность, тем меньше затрат и ошибок это вызовет, следовательно, система является более гибкой и конкурентоспособной;

– расширяемость системы. Возможность добавления в систему новых сущностей и функций, не нарушая ее основной структуры – залог хорошей расширяемости. На начальном этапе разработки в систему имеет смысл закладывать только основной и самый необходимый набор функций. Но при этом архитектура должна позволять легко наращивать дополнительную функциональность по мере необходимости, причем так, чтобы внесение наиболее вероятных изменений требовало наименьших затрат;

– тестируемость. Код, который легче тестировать, будет содержать меньше ошибок и надежнее работать;

– возможность повторного использования кода. Систему желательно проек­тировать так, чтобы ее фрагменты можно было бы повторно использовать при написании других систем.

3.1 Проектирование структурной схемы приложения

Архитектура приложения представляется собой системную структуру и элементы дизайна, из которых состоит приложение. Она также включает в себя методы, процессы и компоненты, используемые во время разработки приложения.

Физическое представление программной системы не может быть полным, если отсутствует информация о том, на какой платформе, а также на каких   
вычислительных средствах она реализована. Для визуализации элементов и компонентов программы была разработана структурная схема приложения, представленная на рисунке 3.1.

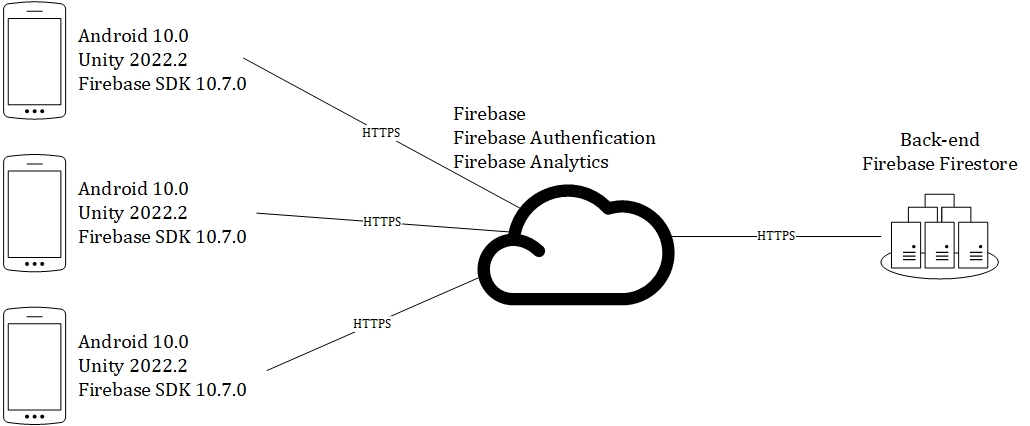


Рисунок 3.1 – Архитектура приложения

Данная диаграмма более детально описывает архитектуру программного средства, указывая какие технологии и библиотеки, в каких модулях используются. В главной роли, по обработке всех данных, а также в формировании и передаче их по удаленным каналам отвечает облачный сервис Firebase. Клиент посылает запросы на сервер по HTTP протоколу версии 1.1 через Firebase SDK версии 10.7.0.

Все процессы, отвечающие за хранение данных возложены на Firebase Firestore. Firestore это гибкая, масштабируемая база данных в реальном времени, предоставляемая Firebase. Она позволяет разработчикам создавать и хранить данные для своих приложений, а также обеспечивает возможность моментальной синхронизации изменений между клиентом и сервером. Firestore поддерживает сложные запросы и индексацию данных, что обеспечивает быстрый доступ к большим объемам информации.

3.2 Проектирование диаграммы вариантов использования

Чтобы начать разработку необходимо составить ряд требований, которым приложение должно соответствовать. Должны быть определены цели и задачи, продуманы все необходимые варианты использования, а также определены роли, используемые в приложении.

Приложение, разрабатываемое в данном дипломном проектировании, должно соответствовать требованиям безопасности, поэтому должна быть организована система защиты от несанкционированного доступа.

Каждый пользователь, прежде чем попасть в систему должен в ней аутентифицироваться. Для программного средства предусмотрены две роли: пользователь и админи­стратор. Для каждой из ролей определен свой ряд обязанностей и вариантов исполь­зования приложения.

Администратор имеет возможность просматривать аналитику.

Для роли пользователя определены следующие возможности:

– создание иллюстраций;

– публикация иллюстраций;

– экспорт иллюстраций;

– сохранение иллюстраций;

– выбор инструмента рисования;

– настройка параметров инструмента;

– настройка цвета инструмента;

– настройка материала инструмента;

– просмотр всех иллюстраций;

– фильтрация иллюстраций;

– просмотре превью иллюстраций;

– размещение иллюстраций в реальности.

Диаграмма вариантов использования (диаграмма прецедентов) в UML – диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать разрабатываемую систему на концептуальном уровне.

Основное назначение диаграммы – описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему.

Для данного дипломного проекта была разработана диаграмма вариантов использования, на которой изображены потребности пользователей системы и функциональные возможности разрабатываемого программного средства. С ее использованием будет проходить дальнейшая разработка веб-приложения. Диаграмма представлена на рисунке 3.2.



Рисунок 3.2 – Диаграмма вариантов использования

В результате была разработана диаграмма вариантов использования, на которой изображен функционал разрабатываемого программного средства. С использованием диаграммы будет проходить дальнейшая разработка приложения.

3.3 Проектирование базы данных

Прежде чем приступить к разработке программного средства необходимо спроектировать базу данных. На основе сформулированных требований к разрабатываемому приложению определяется состав информации, которая должна хранится в базе данных, а также выявляются взаимосвязи данных, размещаемых в БД. Состав и взаимосвязи данных отображаются моделью данных предметной области, которая является основой любой базы данных.

В качестве базы данных для приложения был выбран облачный сервис Firebase Firestore. Firebase Firestore – это гибкая, расширяемая облачная база данных от Google, которая используется в качестве серверного хранилища данных для веб-приложений, а также приложений для мобильных устройств. Firestore обеспечивает быстрый и масштабируемый доступ к данным, а также поддерживает синхронизацию данных в режиме реального времени [11].

Сервис Firestore использует NoSQL базу данных (Not Only SQL, не только SQL) – это семейство СУБД, которые отличаются от реляционных баз данных тем, что они не используют традиционные таблицы, столбцы и строки для хранения данных. Вместо этого они используют различные модели хранения данных, такие как ключ-значение, документоориентированные, графовые и другие.

Одной из причин появления NoSQL баз данных было желание обеспечить более высокую производительность и масштабируемость веб-приложений. Реляционные базы данных не всегда могут эффективно обрабатывать большие объемы данных или обеспечивать достаточную производительность при высокой нагрузке.

Firestore представляет собой документ-ориентированную базу данных, которая хранит данные в виде коллекций документов. Каждый документ может содержать несколько полей, каждое из которых представляет собой пару ключ-значение. Документы могут быть организованы в коллекции, которые в дальнейшем могут быть организованы в различные подколлекции.

Firestore имеет различные SDK для разных языков программирования и платформ, включая Java, Swift, Kotlin, JavaScript, Node.js, Unity и другие. Для интеграции Firestore в приложение используется SDK (Software Development Kit). Firebase Unity SDK предоставляет готовые модули для использования этих функций в приложении, что позволяет разработчику быстро и легко создавать высокопроизводительные приложения на Unity, которые могут масштабироваться по мере необходимости. Firebase Unity SDK также обеспечивает надежную интеграцию с другими модулями Unity, такими как Unity Analytics и Unity Ads, чтобы помочь разработчикам создавать более эффективные и доходные приложения.

Для лучшего понимания структуры разрабатываемой базы данных была создана логическая схема, представленная на рисунке 3.3

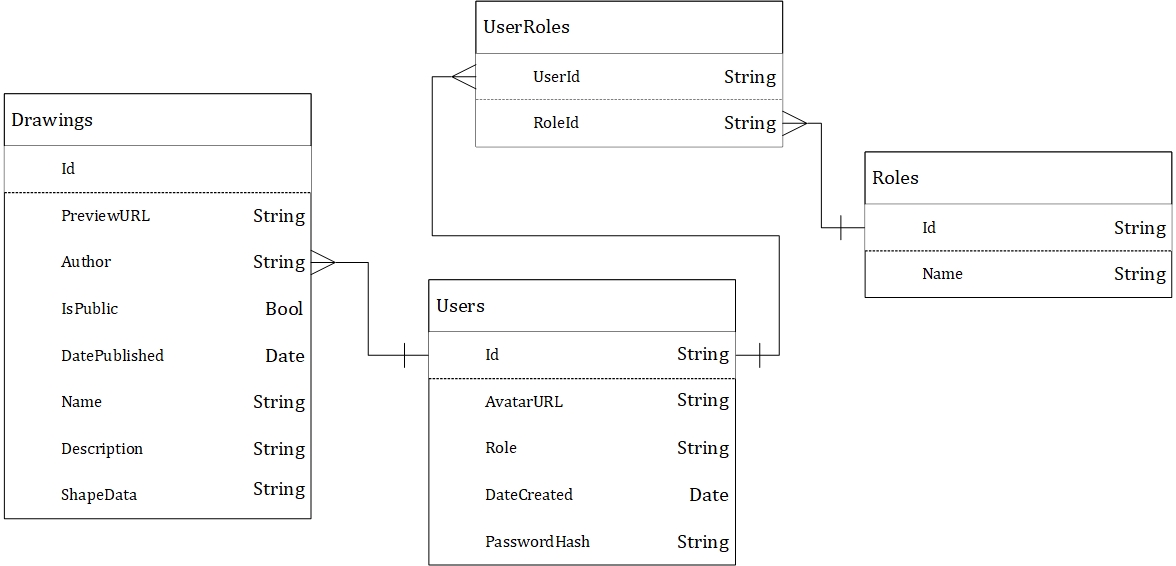


Рисунок 3.3 – Логическая схема базы данных

Для реализации функционала приложения будет создано 4 коллекции. Далее приведена структура и описание всех коллекций.

Коллекция «Users» содержит в себе информацию о пользователях. Структура коллекции представлена в таблице 3.1

Таблица 3.1 – Описание коллекции «Users»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id | String | Идентификатор пользователя |
| AvatarURL | String | Ссылка на аватар пользователя |
| Role | String | Роль пользователя |
| DateCreated | Date | Дата создания |
| PasswordHash | String | Хеш пароля пользователя |

Коллекция «Drawings» содержит в себе информацию об иллюстрациях пользователей. Структура коллекции представлена в таблице 3.2

Таблица 3.2 – Описание коллекции «Drawings»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id | String | Идентификатор иллюстрации |
| Author | String | Автор иллюстрации |
| IsPublic | Bool | Публична ли иллюстрация |
| DatePublished | Date | Дата создания |
| Name | String | Название иллюстрации |
| Description | String | Описание иллюстрации |
| ShapeData | String | Сериализованная иллюстрация |

Коллекция «UserRoles» содержит в себе информацию о ролях, выданных пользователям. Структура коллекции представлена в таблице 3.3

Таблица 3.3 – Описание коллекции «UserRoles»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| UserId | String | Идентификатор пользователя |
| RoleId | String | Идентификатор роли |

Коллекция «Roles» содержит в себе информацию о ролях. Структура коллекции представлена в таблице 3.4

Таблица 3.4 – Описание коллекции «Roles»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id | String | Идентификатор роли |
| Name | String | Название роли |

Коллекция «Roles» содержит в себе информацию о ролях. Структура коллекции представлена в таблице 3.4

Для хранения превью иллюстраций используется сервис Firebase Storage. Firebase Storage – это облачный сервис хранения файлов, который предоставляет интеграцию с платформой Firebase. Он позволяет разработчикам легко хранить, загружать и загружать файлы, такие как изображения, видео и аудиофайлы, используя высокопроизводительное хранилище в облаке.

Firebase Storage предоставляет надежное, масштабируемое и безопасное хранение файлов в облаке Google. Он автоматически масштабируется в зависимости от размера и количества загружаемых файлов, тем самым гарантируя высокую доступность данных и производительность.

Firebase Storage также позволяет контролировать доступ к файлам, например, установить права доступа на чтение и запись для определенных пользователей или групп пользователей. Это обеспечивает безопасность ваших данных и помогает предотвратить несанкционированный доступ к важным файлам.

Firebase Storage предоставляет различные способы загрузки и скачивания файлов, в том числе через веб-интерфейс, консоль Firebase и API.

3.4 Выводы по разделу

В данном разделе описан процесс разработки диаграммы вариантов использования, в которой был представлен основной функционал разрабатываемого приложения, а также описан процесс выбора архитектуры для разрабатываемого программного модуля. Помимо этого, дано краткое описание технологии базы данных, которая используется в приложении, а также разработана модель базы данных. Представлены и описаны коллекции документов БД, а также их особенности.

4 Реализация программного модуля

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*2*

*БГТУ 04.00.ПЗ*

Разраб*.*

*Дежемесов М.С.*

Провер*.*

*.*

*Скребель А.С*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*СмеловВ.В.*

*Лит.*

*Листов*

*21*

74218155, 2023

*4 Реализация программного модуля*

4.1 Задачи программного модуля

Главной задачей данного дипломного проекта является создание приложения для создания иллюстраций в дополненной реальности, который в свою очередь подразумевает работу с модулем дополненной реальности, а также большим количеством UI компонентов. Исходя из анализа аналогов были поставлены задачи по созданию программных компонентов для работы клиентов приложения, а также задача по созданию интуитивно понятного пользовательского интерфейса. Функционально веб-приложение должно выполнять следующие задачи.

Основной задачей администратора является просмотр аналитики приложения.

Задачи обычного пользователя и администратора:

* Авторизация и регистрация пользователя;
* Редактирование профиля пользователя;
* Создание иллюстрации в дополненной реальности;
* Просмотр иллюстраций других пользователей;
* Экспортирование иллюстраций;
* Размещение уже существующих иллюстраций в дополненной реальности;
* Возможность настройки инструментов для создания иллюстраций;

Исходя из перечисленных задач для пользователя можно приступить к реализации веб-приложения.

4.2 Установка модулей и библиотек

Для установки модулей в проект на Unity используется собственный package manager, который является ответвлением node package manager. Для приобретения модулей используется веб-интерфейс Unity Asset store, который содержит огромную библиотеку готовых решений, которые могут сократить время и усилия, необходимые для создания игры или приложения. Часто эти решения создаются и продавцами Unity Asset Store, которые могут зарабатывать за счет продаж своих материалов. Эта платформа также позволяет разработчикам Unity публиковать свои собственные материалы и получать доход от продаж. Asset Store предоставляет высокоэффективный способ распространения контента и продажи своих материалов для других разработчиков.

После приобретения необходимых модулей, их можно добавить в проект через Unity Package Manager (UPM). UPM изображен на рисунке 4.1

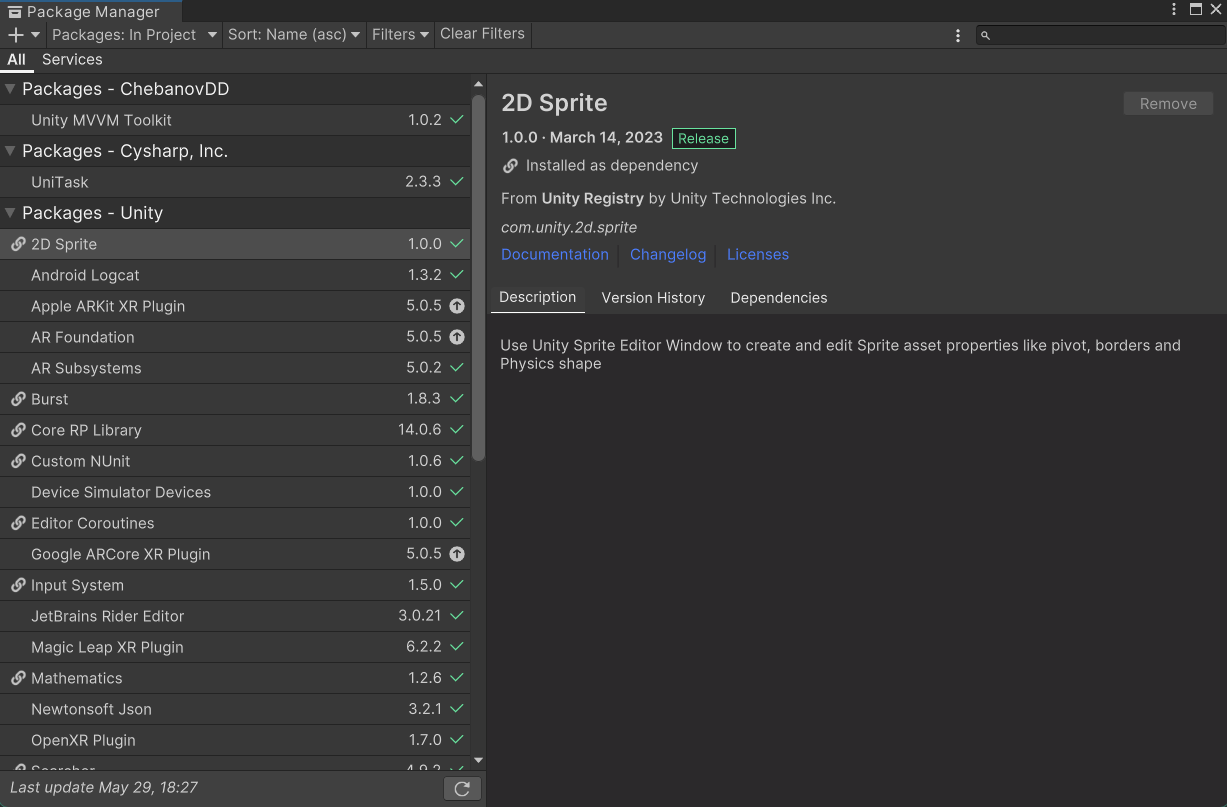


Рисунок 4.1 – Окно Unity Package Manager

Но не все модули, которые используется в проекте, можно найти в Unity Asset Store. Как правило, проекты с открытым исходным кодом используют альтернативную площадку для хостинга Unity модулей, которая называется Open Unity Package Manager (Open UPM).

Open UPM был создан как open-sourсe альтернатива официальному Unity Package Manager. Этот менеджер пакетов предоставляет более широкие возможности и сервисы, чем Unity Package Manager, такие как возможность публикации и установки пакетов без авторизации, более гибкие настройки версий пакета, удобный поиск и мониторинг изменений пакетов.

Open UPM также основан на открытом исходном коде, что позволяет разработчикам без препятствий управлять и распространять свои пакеты. Он поддерживает интеграцию с другими средствами и API, такими как GitHub или GitLab. Это делает Open UPM очень легким и масштабируемым менеджером пакетов для Unity. Open UPM предназначен для удобства и облегчения процесса работы с зависимостями в Unity, что позволяет разработчикам легко использовать сторонние решения в своих проектах и сохранять время на разработке.

Для добавления модулей из Open UPM необходимо добавить новый scope в настройках менеджера пакетов проекта. Окно настроек изображено на рисунке 4.2.

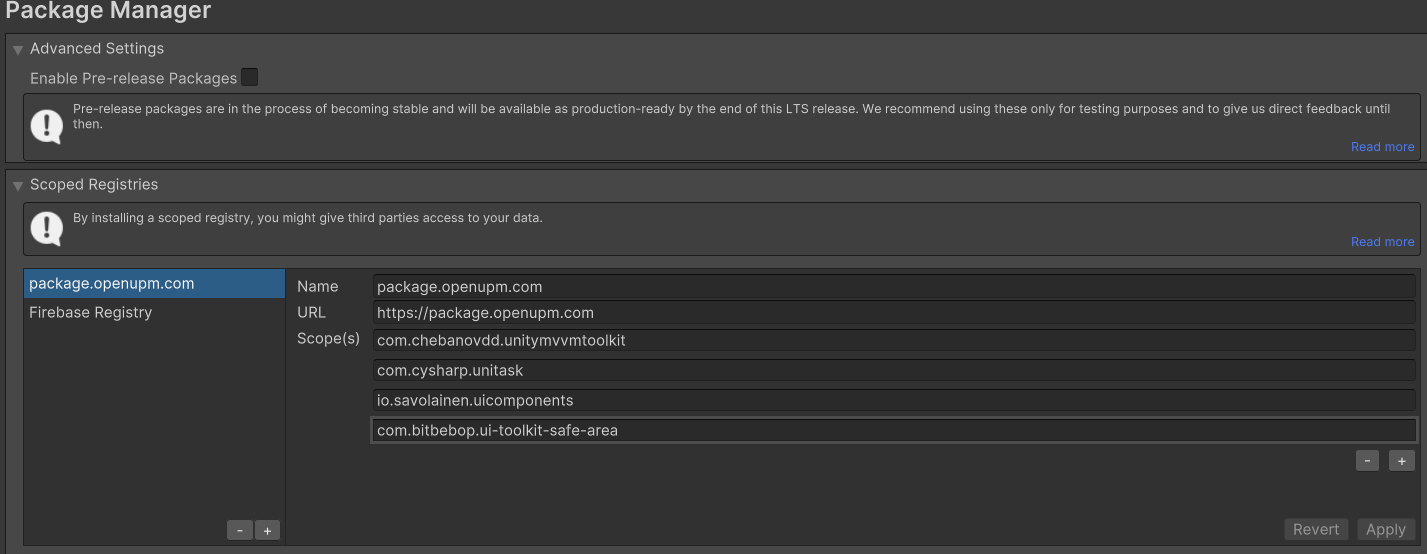


Рисунок 4.3 – Окно настроек Package Manager

Firebase SDK, в свою очередь, не находится ни в официальном Asset Store, ни в Open Unity Package Manager, и его нужно устанавливать с официального сайта Firebase. Unity позволяет устанавливать сторонние модули с расширением *.unitypackage*. Для этого необходимо просто перенести их в окно проекта

Firebase использует сторонние нативные модули, которые не включены в Unity. Для того, чтоб разработчику не устанавливать их вручную, разработчики Firebase создали утилиту External Dependency Manager (EDM). EDM предоставляет возможность управлять версиями Firebase SDK и его зависимостей прямо из Unity, что позволяет создавать приложения со свежей и актуальной версией Firebase.

Данная утилита так же позволяет использовать Firebase SDK в формате модулей, что означает, что вы можете определить, какие модули Firebase SDK должны быть установлены, чтобы сократить размер и время загрузки вашего приложения Unity. Это также помогает оптимизировать использование памяти и помогает избежать потенциальных конфликтов с другими плагинами и библиотеками, используемыми вашим приложением.

Кроме того, EDM обеспечивает автоматический поиск и установку зависимостей Firebase SDK, таких как Google Play Services и Android Support Library, которые могут используются в приложениях на Android. Это помогает облегчить процесс разработки, в особенности для тех, кто не знаком с установкой и управлением нативными зависимостями.

4.3 Выбор архитектуры проекта

Перед началом разработки приложения необходимо выбрать приемлемую архитектуру проекта, которая будет подходить для перечисленных целей приложения, предоставлять весь доступный функционал для работы.

Главной проблемой при разработке приложения на Unity является отсутствие контроля очереди загрузки компонентов приложения, а также отсутствие какой-либо входной точки приложения, из-за чего в коде появляется множество так называемых «костылей», что сильно ухудшает не только читабельность и расширяемость приложения, но в некоторых случаях даже производительность.

Для решения этой проблемы, в проекте используется Dependency Injection, в связке с паттерном State Machine и динамической подгрузкой сцен в зависимости от текущего состояния приложения.

State Machine – это паттерн проектирования, который позволяет управлять состоянием объекта в зависимости от внешних событий и условий. Он представляет собой объект, который может находиться в одном из нескольких состояний и переходить между ними в зависимости от действий пользователя или других внутренних или внешних факторов. Использование State Machine в разработке позволяет сделать код более структурированным, легко читаемым и поддерживаемым. Кроме того, State Machine упрощает тестирование и отладку приложений, так как логика переходов между состояниями объекта является более явной и прозрачной.

С другой стороны, использование Dependency Injection позволяет снизить связность кода и улучшить его модульность. Внедрение зависимостей в объекты позволяет избежать жестких связей между компонентами, что делает код более прозрачным, упрощает его понимание и упрощает разработку. Так же, DI позволяет использовать макеты объектов для тестирования и уменьшает риск ошибок, связанных с их созданием непосредственно в классах.

Кроме того, использование DI позволяет упростить расширение кода, так как изменение конкретной зависимости не требует изменения кода всех объектов, которые ее используют. Dependency Injection так же позволяет отделить зависимости от кода и делает их более переносимыми и повторно используемыми.

Так как большая часть приложения состоит из пользовательского интерфейса, так же необходимо определить архитектуру, которая будет использоваться для разработки пользовательского интерфейса. Для этого был выбран паттерн MVVM. Визуализация данного паттерна представлена на рисунке 4.4.

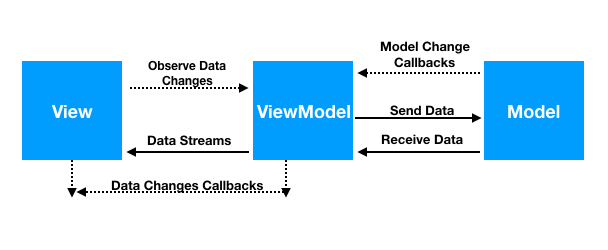


Рисунок 4.4 – Паттерн MVVM

Данный паттерн имеет большое количество плюсов при разработке. Он разделяет логику приложения на три части: модель (Model), представление (View) и модель представления (ViewModel), что повышает читаемость и расширяемость.

Так же, MVVM уменьшает зависимости между компонентами, позволяя отделить представление от модели. Это, в свою очередь, позволяет разработчикам использовать модульное тестирование. ViewModel предоставляет обёртку для работы с данными в модели и представлении. Это упрощает работу с данными и контролирует их состояние.

Кроме того, MVVM ориентирован на использование перерисовки, которая происходит только тогда, когда происходят изменения в модели представления.

Данный паттерн так же обеспечивает высокий уровень контроля за процессом разработки, позволяя быстро отлаживать код и мониторить состояние приложения.

4.3 Файловая структура

Перед началом разработки программного модуля мной была продумана файловая структура в соответствии с архитектурой проекта и паттерном MVVM, позволяющая быстро ориентироваться в коде, вносить изменения и дописывать новый функционал. Основная структура проекта изображена на рисунке 4.5.

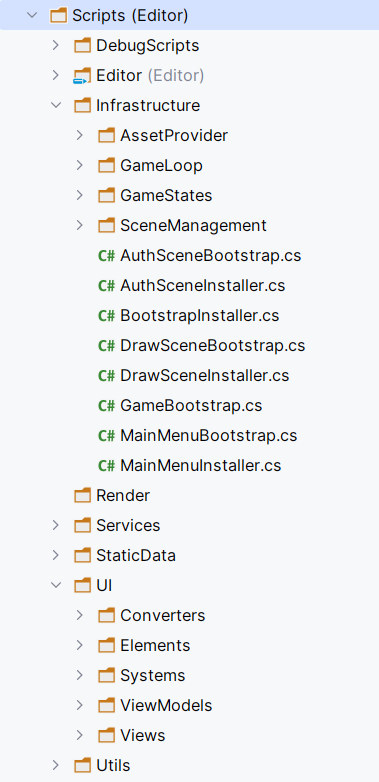


Рисунок 4.5 – Структура проекта

В Infrastructure располагаются основные архитектурные компоненты приложения, такие как состояния, управление сценами, загрузка ассетов и т.д.

Services содержит в себе основные сервисы, которые используются в проекте. К примеру, Auth Service отвечает за управление пользовательским аккаунтом, а также валидирует пользовательскую аутентификацию, Draw Service отвечает за создание новых фигур на сцене, а Input отвечает за пользовательский ввод.

В Static Data располагаются файлы, которые используются для создания статических данных, которые можно редактировать без обновления кода. Это сделано через Unity Scriptable Objecs.

Scriptable Objects (скриптовые объекты) – это специальный тип объектов в Unity, которые могут быть использованы для хранения данных и параметров игровых объектов. Scriptable Objects могут быть созданы как обычные скрипты в Unity и могут быть использованы в любом месте проекта, где требуются определенные данные. Они могут быть созданы, сохранены и управляться в Unity средствами редактора, их можно легко настроить и изменить, а также создать множество экземпляров с различными значениями параметров. Доступ к Scriptable Objects можно получать из различных объектов и даже других Scriptable Objects, что позволяет создать дополнительные взаимосвязи в данных. Пример Scriptable object можно увидеть на рисунке 4.6.

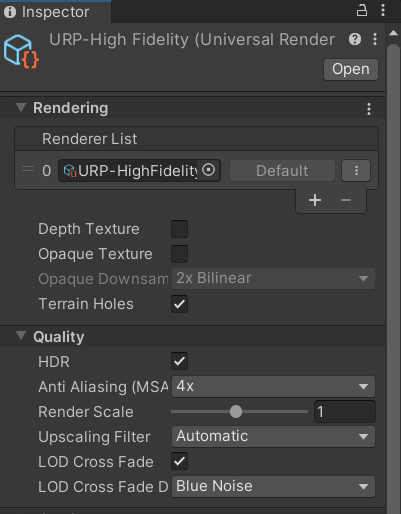


Рисунок 4.6 – Пример Scriptable Object

В папке UI находится вся логика, связанная с пользовательским интерфейсом. Здесь расположены представления (Views), и модели представлений (View Models), а также конвертеры данных, которые используются при связывании данных для автоматизированной конвертации одного типа данных в другой.

4.4 Динамическая загрузка сцен

Как говорилось выше, для контроля состояния систем приложений, используется механизм сцен, в совокупности с паттерном машины состояний.

Сцены являются одним из основных компонентов приложений, разрабатываемых в Unity. Каждая сцена представляет собой независимое окружение, где располагаются все игровые объекты, связанные с этой сценой.

Сцены позволяют разделять игровые объекты на логически связанные группы, упрощая навигацию и управление проектом. Разбиение проекта на сцены позволяет концентрироваться на конкретных функциях и задачах и, при нужде, подменять их прямо во время исполнения приложения. Сцены могут быть настроены на различные уровни детализации графики.

Разбиение проекта на сцены так же помогает быстро настроить тестовые сценарии и проверить работоспособность приложения в тестовом окружении. В Unity легко управлять сценами с помощью встроенных инструментов и настроек, что позволяет быстро переключаться между различными сценами и адаптировать приложение к разным устройствам и форматам экранов.

DI фреймворк Zenject так же позволяет устанавливать в контейнер зависимости, которые необходимы только в данный момент на конкретной сцене, и при смене сцены автоматически отгружать их, что позволяет разрабатывать более гибкую систему зависимостей объектов. Пример установки зависимостей для сцены можно увидеть на листинге 4.1

|  |
| --- |
| public class AuthSceneInstaller : MonoInstaller  {  [SerializeField] private GameObject \_registerView;  [SerializeField] private GameObject \_singInView;  public override void InstallBindings()  {  Container.BindInterfacesAndSelfTo<RegisterViewModel>().AsSingle();  Container.BindFactory<RegisterView, RegisterView.Factory>()  .FromComponentInNewPrefab(\_registerView);    Container.BindInterfacesAndSelfTo<SignInViewModel>().AsSingle();  Container.BindFactory<SignInView, SignInView.Factory>()  .FromComponentInNewPrefab(\_singInView);    Container.BindInterfacesTo<ViewStack>().AsSingle();  Container.BindInterfacesTo<AuthViewProvider>().AsSingle();  Container.BindFactory<AuthState, AuthState.Factory>();  Container.BindInterfacesAndSelfTo<AuthSceneBootstrap>().AsSingle().NonLazy();  }  } |

Листинг 4.1 – Класс установки зависимостей сцены

Входной точкой приложения является класс BootstrapInstaller.cs. Здесь происходит установка зависимостей, которые будут использоваться во всём проекте, а также переход в состояние Bootstrap, в котором происходит инициализация базовых систем приложения по порядку, что позволяет одной системе использовать другую. Так, в этом состоянии происходит загрузка статических данных, настроенных из редактора, происходит инициализация сервиса аутентификации, и, в зависимости от успешности этого процесса, переход в состояние авторизации пользователя, либо в главное меню. Исходный код данного состояния можно увидеть на листинге 4.2

|  |
| --- |
| public class BootstrapState : IEnterState  {  private readonly ISceneLoader \_sceneLoader;  private readonly IStaticDataService \_staticData;  private readonly IAuthSystem \_auth;  public BootstrapState(ISceneLoader sceneLoader, IStaticDataService staticData, IAuthSystem auth)  {  \_sceneLoader = sceneLoader;  \_staticData = staticData;  \_auth = auth;  }  public async void OnEnter()  {  Application.targetFrameRate = 300;  await \_staticData.Load();  await \_auth.Init();  await \_sceneLoader.LoadScene(\_auth.IsSignedIn ? SceneIndex.MainMenu : SceneIndex.Auth);  }  public class Factory : PlaceholderFactory<BootstrapState>  {  }  } |

Листинг 4.2 – Класс BootstrapInstaller

Таким образом, разработчик может легко переходить из одного состояние в другое, что позволяет иметь большой контроль над текущим состоянием систем и приложения, а также просто добавлять функционал в приложение.

4.5 Создание иллюстраций

В основе создания иллюстраций находится AR Foundation и компонент Unity Line Renderer, который предоставляет простой API для создания графических элементов. С помощью Unity Line Renderer можно настроить ширину линии, ее цвет, количество сегментов и другие параметры, что позволяет создавать различные эффекты. Так же, Line Renderer имеет низкий уровень нагрузки на систему, что позволяет создавать множество линий без потери производительности.

Для создания фигур используются Scriptable Objects, которые сообщают компоненту LineRenderer позиции для отрисовки линий. Это позволяет настроить отображение прямо в редакторе Unity, настроить доступные пользователю формы, а также упрощает процесс добавления новых рисунков. Код для создания куба расположен в листинге 4.3

|  |
| --- |
| [CreateAssetMenu(menuName = "Shapes/Cube")]  public class Cube : Shape  {  public override void OnDrawMove(IShapeContainer container, Vector3 position)  {  container.Clear();  DrawRectangle(container, position);  }  private static void DrawRectangle(IShapeContainer container, Vector3 endPosition)  {  var radius = Vector3.Distance(Vector3.zero, endPosition);  const float startAngle = Mathf.PI / 4f;  for (var i = 0f; i < 4f; i++)  {  var angle = i / 4f \* Mathf.PI \* 2f + startAngle;  var pointPos = new Vector3(Mathf.Cos(angle), Mathf.Sin(angle)) \* radius;  container.AppendPosition(pointPos);  }  }  } |

Листинг 4.3 – Класс Cube

Для того, чтоб созданные пользователем линии были синхронизированы с окружающим миром, сцену необходимо настроить для Unity AR Foundation.

На сцене должен быть расположен объект с компонентом ARSession. Этот компонент управляет жизненным циклом и параметрами конфигурации для сессии в дополненной реальности. Так же, на сцене должен быть компонент XR Origin, и создана камера, которая будет отображать вывод камеры устройства, а также изображение, которое создается в приложение.

Для отслеживания дополненной реальности используются компоненты AR Anchor Manager и AR Plane Manager. Плоскости в дополненной реальности – это определенные плоскости в реальном мире, на которых отображаются виртуальные объекты. Это могут быть столы, полы, стены, столешницы и другие горизонтальные или вертикальные поверхности.

Определение плоскостей в AR осуществляется с помощью камеры мобильного устройства или другого устройства для AR, которые анализируют окружающую обстановку и ищут признаки поверхностей, такие как текстуры, цвета, границы и т.д. В результате программа определяет положение, форму и размер плоскостей. В приложении, плоскости используются для создания интерактивных виртуальных объектов, которые могут быть размещены на конкретных поверхностях в реальном мире.

4.6 Разработка пользовательского интерфейса

Пользовательский интерфейс разработан при помощи технологии Unity UI Toolkit с применением сторонней библиотеки, позволяющей использовать MVVM. UI Toolkit содержит в себе визуальный редактор как элементов, так и стилей, что позволяет разработчику быстро добавлять новые элементы, и создавать адаптивные дизайны. Расположение элементов сохраняется в UI Document с расширением *.uxml.* Разработчик так же может использовать uxml напрямую для разработки. Пример uxml файла показан в листинге 4.4

|  |
| --- |
| <ui:UXML xmlns:ui="UnityEngine.UIElements" xmlns:uie="UnityEditor.UIElements" xsi="http://www.w3.org/2001/XMLSchema-instance" engine="UnityEngine.UIElements" editor="UnityEditor.UIElements" noNamespaceSchemaLocation="../../../../UIElementsSchema/UIElements.xsd" editor-extension-mode="False">  <ui:Template name="BackButton" src="project://database/Assets/Resources/UI/Common/BackButton.uxml?fileID=9197481963319205126&amp;guid=06b7cf2ba4e21dc448246807f9a635d6&amp;type=3#BackButton" />  <Style src="project://database/Assets/Resources/UI/Common.uss?fileID=7433441132597879392&amp;guid=73b8dad78532204418ede5177ae46c9a&amp;type=3#Common" />  <ArPaint.UI.Elements.SafeAreaView>  <ui:VisualElement name="root" class="view-root" style="flex-grow: 1; background-color: rgba(0, 0, 0, 0); margin-bottom: 10px;">  <ui:Instance template="BackButton" name="BackButton" />  <ui:VisualElement style="flex-grow: 1; background-color: rgba(0, 0, 0, 0); flex-direction: column-reverse;">  <UnityMvvmToolkit.UITK.BindableUIElements.BindableButton command="PlaceCommand" text="Place" class="default-button" />  </ui:VisualElement>  </ui:VisualElement>  </ArPaint.UI.Elements.SafeAreaView>  </ui:UXML> |

Листинг 4.4 – Файл DrawingPlacerView.uxml

В UI Toolkit так же имеется способ добавления стилей элементам, который называется USS. USS (Unity Style Sheets) – это язык стилей, который используется в Unity для создания стилей пользовательских элементов интерфейса (UI) для приложений и игр. USS – ключевая компонента Unity UI Toolkit.

USS позволяет разработчикам создавать стили для компонентов UI, таких как формы, редакторы текста, кнопки и т.д. Он основан на языке CSS (Cascading Style Sheets), который используется для веб-разработки.

К преимуществам USS можно отнести:

– упрощение создание UI: USS помогает разработчикам создавать пользовательские интерфейсы без необходимости создания элементов UI с нуля;

– снижение времени разработки: USS позволяет создавать стили с помощью одного файла, что ускоряет процесс разработки UI;

– увеличение согласованности: использование USS помогает поддерживать единый стиль внешнего вида UI по всему приложению.

В проекте используется несколько файлов стилей. К примеру, часть файла стиля приложения приведен в листинге 4.5.

|  |
| --- |
| :root {  --text-color: rgb(255, 255, 255);  --primary-color: rgb(47, 105, 255);  --bg-opacity: 0.15;  --default-height: 60px;  --bg-color: rgb(20, 20, 22);  --danger-color: rgb(239, 70, 111);  --neutral: rgb(255, 255, 255); }  .unity-text-element {  -unity-font: url('project://database/Assets/UI%20Toolkit/Fonts/Poppins-Regular.ttf?fileID=12800000&guid=c5973a18b44adae479ce54f5c571fd17&type=3#Poppins-Regular');  color: var(--text-color); }  .caption-label {  -unity-font-style: normal;  font-size: 24px;  white-space: normal;  margin-left: 0;  margin-right: 0;  margin-top: 0;  margin-bottom: 0;  padding-left: 0;  padding-right: 0;  padding-top: 0;  padding-bottom: 0; } .shape-select-button {  flex-grow: 1;  flex-direction: column-reverse; } |

Листинг 4.5 – Файл Common.uss

UI Toolkit так же позволяет пользователю создавать свои элементы при помощи C# и UXML. Элементы, созданные с помощью UXML можно использовать повторно, но к ним нельзя добавить дополнительную логику. Создание элемента на C# требует куда большего количества написанного кода, но позволяет разработчику добавлять свою логику элементам.

Каждый элемент в UI Toolkit должен наследовать базовый класс *Visual Element*. В отличие от обычных компонентов *MonoBehaviour* в Unity, этот класс не использует никакую систему сериализации, а также не содержит сторонний жизненный цикл, вроде методов *Start*, *Awake* и *Update*. Вместо этого, UI Toolkit использует систему событий, которая очень похожа на систему событий в веб-разработке.

События UI Toolkit генерируются при различных действиях пользователя, таких как щелчок на кнопке, перемещение курсора мыши, ввод текста и других. Разработчики могут подписываться на эти события, чтобы изменять поведение UI элементов в зависимости от действий пользователя.

Основные типы событий UI Toolkit:

– MouseEvents: события, связанные с действиями мыши, такие как MouseEnter, MouseLeave, MouseDown, MouseUp и т.д.;

– ClickEvents: события, связанные с нажатием кнопок, такие как Click, DoubleClick, DoubleClickInterval;

– KeyEvents: события, связанные с нажатием клавиш на клавиатуре, такие как KeyDown и KeyUp;

– FocusEvents: события, связанные с фокусировкой элементов UI, такие как FocusIn, FocusOut и т.д.;

– GeometryEvents: события, которые вызываются при перестроении иерархии.

Разработчики так же могут настраивать опции настройки элементов. Для этого, в классе *VisualElement* имеется подкласс *UxmlTraits*. В нём, разработчик может указать, какие поля для настройки есть у элемента, и как они меняют элемент.

Кроме того, разработчик может настраивать метаинформацию элемента. Для этого существует подкласс UxmlFactory, который занимается созданием элемента, а также генерацией информации о нем. Здесь, разработчик может поменять путь к элементу, его название, а также называние сборки, и даже модифицировать процесс создания элемента. Класс элемента слайдера показан на листинге 4.6.

|  |
| --- |
| public class FillableSlider : Slider  {  public readonly string progressBarClass = $"{ussClassName}\_\_progress";  private readonly VisualElement \_progressBar;  private readonly VisualElement \_dragger;    public FillableSlider() : base()  {  \_progressBar = new VisualElement();  \_progressBar.usageHints = UsageHints.DynamicTransform;  \_progressBar.name = "progress";  \_progressBar.AddToClassList(progressBarClass);  \_progressBar.style.height = Length.Percent(100f);  \_dragger = this.Q<VisualElement>(className: draggerUssClassName);    var tracker = this.Q<VisualElement>(className: trackerUssClassName);  tracker.Add(\_progressBar);      \_dragger.RegisterCallback<GeometryChangedEvent>(\_ => UpdateFill());  this.RegisterValueChangedCallback(\_ => UpdateFill());  UpdateFill();  }  private void UpdateFill()  {  \_progressBar.style.width = \_dragger.transform.position.x + \_dragger.contentRect.width \* 0.5f;  }    public new class UxmlTraits : Slider.UxmlTraits  {  }    public new class UxmlFactory : UxmlFactory<FillableSlider, UxmlTraits>  {  }  } |

Листинг 4.6 – Файл FillableSlider.cs

Так же после того, как был создан свой элемент с пользовательской логикой, для использования его совместно с MVVM, необходимо добавить логику обработки данных. Для этого, необходимо унаследовать базовый элемент, добавить входные данные, и реализовать интерфейс *IBindableElement.* Исходный код элемента *BindableSlider* расположен на листинге 4.7.

|  |
| --- |
| public class BindableSvgImage : Image, IBindableElement {  private PropertyBindingData \_imageBindingData;  private IProperty<VectorImage> \_imageProperty;    private string ImageBindingPath { get; set; }    public void SetBindingContext(IBindingContext context, IObjectProvider objectProvider)  {  \_imageBindingData ??= ImageBindingPath.ToPropertyBindingData();  \_imageProperty = objectProvider.RentProperty<VectorImage>(context, \_imageBindingData);  \_imageProperty.ValueChanged += OnImageChanged;   SetImage(\_imageProperty.Value);  }  public void ResetBindingContext(IObjectProvider objectProvider)  {  if (\_imageProperty == null)  return;  \_imageProperty.ValueChanged -= OnImageChanged;  objectProvider.ReturnProperty(\_imageProperty);  \_imageProperty = null;  }  private void OnImageChanged(object sender, VectorImage image)  {  SetImage(image);  }  public new class UxmlTraits : Image.UxmlTraits  {  private readonly UxmlStringAttributeDescription \_imageBindingPath = new()  { name = "image-binding-path", defaultValue = string.Empty };   public override void Init(VisualElement visualElement, IUxmlAttributes bag, CreationContext context)  {  base.Init(visualElement, bag, context);  var view = (BindableSvgImage)visualElement;  view.ImageBindingPath = \_imageBindingPath.GetValueFromBag(bag, context);  }  } |

Листинг 4.7 – Файл FillableSlider.cs

Здесь создается свойство *IProperty*, которое содержит в себе данные, которые располагаются в модели представления. Элемент может как модифицировать эти данные, так и подписываться на их модификацию.

4.7 Разработка стека представлений

Стек представлений (View Stack) – это концепция, используемая в MVVM (Model-View-ViewModel) архитектуре, которая помогает управлять и показывать представления в пользовательском интерфейсе.

Данный подход представляет собой стек, или коллекцию, представлений, которые могут быть отображены пользователю. Он может быть использован для управления навигацией в приложении и отображения и скрытия экранов пользовательского интерфейса. Обычно стек представлений реализуется в виде коллекции, которая хранит представления в порядке их отображения. Стек представлений в MVVM обеспечивает гибкость и удобство управления представлениями приложения, что облегчает навигацию и управление экранами пользовательского интерфейса.

Unity рекомендует кешировать представления, так как их создание во время исполнения программы требует много ресурсов. Поэтому, при переходе на определенную сцену, создаются все представления, после чего, они отключаются, тем самым перестают быть видны пользователю и обрабатывать ввод.

Для того, чтоб встраивать зависимости для объектов, которые были созданы во время исполнения программы, фреймворк Zenject требует от разработчика использовать фабрики. Для того, чтоб создавать представления, которые нужны на конкретной сцене, был создан интерфейс *IViewsProvider.* Класс, который реализует его, инкапсулирует в себе логику создания и выдачи представлений своим клиентам. Пример класса, реализующего этот интерфейс, приведен в листинге 4.8.

|  |
| --- |
| public class AuthViewProvider : ViewProviderBase {  public AuthViewProvider(RegisterView.Factory registerViewFactory, SignInView.Factory signInFactory)  {  views = new Dictionary<Type, IStackableView>  {  [typeof(RegisterView)] = registerViewFactory.Create(),  [typeof(SignInView)] = signInFactory.Create(),  };   HideAllViews();  }    protected void HideAllViews()  {  foreach (var view in views.Values)  view.Hide();  } } |

Листинг 4.8 – Файл FillableSlider.cs

Для того, чтоб управлять стеком представлений, был разработан сервис *ViewStack,* который занимается добавлением в стек новых представлений, а также их сокрытием. Кроме того, этот класс управляет получением фокуса представлений и их порядком отображения. Код данного класса показан в листинге 4.9.

|  |
| --- |
| public class ViewStack : IViewStack, IDisposable  {  private readonly IViewProvider \_viewProvider;  private readonly Stack<IStackableView> \_viewStack = new();  public IStackableView ActiveView => \_viewStack.TryPeek(out var view) ? view : null;  public ViewStack(IViewProvider viewProvider)  {  \_viewProvider = viewProvider;  }    public void PushView<TView>() where TView : IStackableView  {  (ActiveView as IViewFocusable)?.OnUnfocus();  var view = \_viewProvider.GetView<TView>();  if (view == null)  return;  \_viewStack.Push(view);    (view as ISortableView)?.SetSortOrder(\_viewStack.Count);  view.SetViewStack(this);  view.Show();  }  public void PopView()  {  if (!\_viewStack.TryPop(out var view))  return;  view.Hide();    (view as ISortableView)?.SetSortOrder(0);  (ActiveView as IViewFocusable)?.OnFocus();  }  public void Dispose()  {  \_viewStack.Clear();  }  } |

Листинг 4.9 – Файл ViewStack.cs

4.8 Firebase SDK

Для работы с сервисом Firebase в проекте используется Firebase SDK. SDK – это набор инструментов, библиотек и документации, которые помогают разработчикам создавать приложения для определенной платформы или технологии. SDK может включать в себя фреймворки, библиотеки, документацию, примеры кода и другие ресурсы, необходимые для разработки приложений.

Для работы с различными сервисами Firebase, после установки необходимых сервисов, разработчику доступно простое в использовании API. Для каждого сервиса выделен свой класс, реализующий паттерн синглтон. Так, для работы с ядром Firebase есть класс FirebaseApp с полем *DefaultInstance*. Аналогично, для сервиса с аутентификацией существует класс *FirebaseAuth*, для работы с базой данных Firestore имеется класс *FirebaseFirestore*, а для работы с сервисом Storage существует класс *FirebaseStorage*.

Так как в проекте используется Dependency Injection фреймворк Zenject, использование синглтонов напрямую недопустимо, так как создает неявные зависимости объектов, и не позволяет использовать контракты для работы с объектами, что сильно ухудшает расширяемость кода. Для решения этой проблемы, в главном установщике зависимостей все сервисы были установлены в режиме Single, что позволяет использовать их как явные зависимости других объектов. Процесс установки изображен в листинге 4.10

|  |
| --- |
| Container.Bind<FirebaseApp>().FromMethod(\_ => FirebaseApp.DefaultInstance).AsSingle();  Container.Bind<FirebaseAuth>().FromMethod(\_ => FirebaseAuth.DefaultInstance).AsSingle();  Container.Bind<FirebaseFirestore>().FromMethod(\_ => FirebaseFirestore.DefaultInstance).AsSingle();  Container.Bind<FirebaseStorage>().FromMethod(\_ => FirebaseStorage.DefaultInstance).AsSingle(); |

Листинг 4.10 – Установка зависимостей Firebase

4.8.1 Firebase Authentication

Для управление пользовательским аккаунтом был разработан сервис Auth, который является абстракцией над сервисом FirebaseAuth. FirebaseAuth предоставляет множество методов и свойств, которые позволяют управлять аутентификацией пользователей, а также получать информацию о них, такую как имя пользователя, email, идентификатор пользователя и т.д. Через него происходит регистрация пользователя, его авторизация, настройка его профиля, восстановление пароля, получение текущего профиля пользователя, а также проверка его актуальности его аутентификации.

FirebaseAuth также предоставляет методы для управления сессиями пользователей и для проверки их подлинности в приложении. Например, можно использовать метод *IsLoggedInToFirebase* для проверки, авторизован ли пользователь в приложении, или метод *User.GetToken* для получения токена аутентификации пользователя. Код функций, отвечающих за регистрацию и авторизацию пользователя представлен на листинге 4.11.

|  |
| --- |
| public async UniTask Register(string email, string username, string password)  {  var authResult = await \_auth.CreateUserWithEmailAndPasswordAsync(email, password);  var profile = new UserProfile  {  DisplayName = username  };  try  {  await authResult.User.UpdateUserProfileAsync(profile);  await authResult.User.SendEmailVerificationAsync();  }  catch  {  await authResult.User.DeleteAsync();  throw;  }  FirebaseAnalytics.LogEvent(  FirebaseAnalytics.EventSignUp, new Parameter(  FirebaseAnalytics.ParameterMethod, "mail"));  }  public async UniTask SignIn(string email, string password)  {  var credential = EmailAuthProvider.GetCredential(email, password);  await \_auth.SignInWithCredentialAsync(credential);  FirebaseAnalytics.LogEvent(  FirebaseAnalytics.EventLogin, new Parameter(  FirebaseAnalytics.ParameterMethod, "mail"));  } |

Листинг 4.11 – Функция регистрации и авторизации пользователя

Таким образом, при помощи этих функций можно просто зарегистрировать новых пользователей, а также авторизовать уже существующих.

4.8.2 Firebase Analytics

Firebase Analytics – это инструмент аналитики, который позволяет собирать и анализировать данные о пользователях и использовании приложения в Firebase. Он предоставляет широкий спектр функций, таких как отслеживание событий, отслеживание действий пользователя, определение источников трафика и т.д.

В Firebase SDK для есть класс FirebaseAnalytics, который предоставляет доступ к функциям Firebase Analytics из Unity-приложения. Этот класс позволяет отслеживать события, определять источники трафика и многое другое, что позволяет создавать более эффективные и масштабируемые приложения.

Отслеживать события можно с помощью статического метода LogEvent с указанием типа события. Здесь можно использовать как уже имеющиеся типы событий, такие как EventLogin, EventSignUp, EventPurchase и т.д., но также можно создавать свои собственные события, передавая строку с названием события. Для удобства был создан класс AnalyticsEvents, который содержит в себе названия новых событий. Код этого класса изображен в листинге 4.12

|  |
| --- |
| public static class AnalyticsEvents  {  public const string ShapeSelect = "shape\_select";  public const string ChangeUserName = "change\_username";  public const string ChangePassword = "change\_password";  public const string ViewDrawing = "view\_drawing";  public const string PublishDrawing = "publish\_drawing";  public const string UnpublishDrawing = "unpublish\_drawing";  } |

Листинг 4.12 – Класс AnalyticsEvents

Кроме того, для улучшения аналитики, разработчик может передавать параметры в события. Параметры помогают лучше понять действия пользователя, к примеру, какой предмет был куплен, либо какой рисунок был просмотрен пользователем. Параметры структурно представляют из себя словарь с видом ключ-значение. Как правило, в роли ключа выступает строка, как и в роли значения, но в значения так же можно передавать и числа типа long и double. Пример отправки события с параметрами представлен в листинге 4.13

|  |
| --- |
| public IShape Shape  {  get => \_shape;  set  {  \_shape = value;  FirebaseAnalytics.LogEvent(AnalyticsEvents.ShapeSelect,  new Parameter("shape", \_shape.GetType().Name));  }  } |

Листинг 4.13 – Отправка события

Таким образом, можно легко отслеживать пользователя, собирать аналитику, и улучшать функционал приложения.

4.8.3 Firebase Firestore и Storage

Firebase Firestore – это облачное хранилище данных NoSQL, разработанное компанией Google. Оно предоставляет возможность хранить, искать и обновлять данные в режиме реального времени. Firebase Firestore имеет ряд преимуществ по сравнению с традиционными базами данных, такими как MySQL или PostgreSQL.

Облачное хранилище данных Firebase Storage, предоставляемое компанией Google, позволяет хранить и получать доступ к данным в облачном хранилище.

Класс *FirebaseFirestore* предоставляет доступ к Firebase Firestore через API. Он позволяет создавать и обновлять документы, получать данные из коллекций, выполнять запросы и многое другое.

Для доступа к документу достаточно вызвать метод *Document*, передав в него путь к запрашиваемому документу. Документы можно объединить в коллекцию документов для более удобной архитектуры базы данных. Для доступа к коллекции, по аналогии с доступом к документу, необходимо вызвать метод *Collection,* передав в него путь к коллекции.

Коллекции не могут содержать в себе другие коллекции, как и документы не могут содержать в себе другие документы, поэтому при создании пути, путь к документу должен быть четным, а путь к коллекции – нечетным, что нужно помнить, при создании архитектуры базы данных. Однако, в документе может содержаться несколько коллекции, так же как в коллекции может содержаться несколько документов, что так же может сыграть на руку при проектировании БД.

Документ в Firebase может быть представлен классом, либо структурой. Для того, чтоб объект можно быль использовать в качестве документа, необходимо добавить ему атрибут *FirestoreData.* Затем, разработчику необходимо пометить все поля, которые будут синхронизироваться с документом атрибутом *FirestoreProperty.* Этим атрибутом могут быть помечены любые типы данных, однако, если пользователю необходимо создать ссылку на структуру, либо же класс, данный класс и его поля должны быть сериализуемыми и помечены атрибутом *System.Serializable.* Пример такого класса можно увидеть на листинге 4.14.

|  |
| --- |
| [FirestoreData]  [Serializable]  public class DrawingData  {  [FirestoreProperty]  public int Id { get; set; }      [FirestoreProperty]  public string Author { get; set; }      [FirestoreProperty]  public string AuthorName { get; set; }  [FirestoreProperty]  public DateTime CreationDate { get; set; }    [FirestoreProperty]  public string Name { get; set; }  [FirestoreProperty]  public string Description { get; set; }    [FirestoreProperty]  public List<SerializableDrawCommand> DrawCommands { get; set; }  public bool IsOwned { get; set; }  public bool IsPublished { get; set; }  public byte[] Preview { get; set; }  } |

Листинг 4.13 – Класс DrawingData

После вызова метода *Collection* либо *Document,* разработчику возвращается ссылка на часть базы данных, которую можно использовать для модификации объекта. Так, к примеру, можно получить список всех документов в коллекции, либо получить содержимое документа, а также осуществить его удаление, либо модификацию. Пример работы с базой данных показан в листинге 4.14.

|  |
| --- |
| public async UniTask Save()  {  var json = JsonConvert.SerializeObject(Drawings);  \_persistentData.SetValue(nameof(Drawings), json);  foreach (var drawing in Drawings.Where(drawing => drawing.IsPublished))  {  await \_firestore.Document($"{CollectionName}/drawing\_{drawing.Id}").SetAsync(drawing);  }  } |

Листинг 4.14 – Сохранение документа в Firestore

Так же, при помощи Firebase SDK, можно получить копию всех документов, располагающихся в коллекции, называющуюся снэпшотом, и используется только для чтения коллекции. Получить снепшот коллекции можно с помощью метода *GetSnapshotAsync.* Пример получения снепшота можно увидеть в листинге 4.15.

|  |
| --- |
| public async UniTask<List<DrawingData>> GetPublishedDrawings()  {  var drawings = new List<DrawingData>();  var query = await \_firestore.Collection(CollectionName).GetSnapshotAsync();    foreach (var document in query.Documents)  {  var data = document.ConvertTo<DrawingData>();  data.IsOwned = data.Author == \_auth.User.UserId;  drawings.Add(data);  }  return drawings;  } |

Листинг 4.15 – Получение копии документов коллекции

Здесь происходит конвертация документов Firebase в класс, который представляет собой документ, помеченный атрибутом *FirestoreData.*

Для отображения превью рисунков был использован сервис Firebase Storage, который используется для загрузки и скачивания файлов. Данный сервис был выбран, так как процедурная генерация изображения рисунка из загруженных данных имеет линейную нагрузку на устройство, поэтому при растущем количестве изображений, загрузка всех изображений рисунков станет накладной задачей. Для решения этой проблемы, после завершения процесса рисования, генерируется картинка, содержащая в себе изображение иллюстрации, которое используется в списке доступных иллюстраций.

Для работы с изображениями был написан сервис *ImageProvider*, который выступает в роли абстракции над классом FirebaseStorage, а также выполняет кеширование уже загруженных изображений.

В Firebase Storage можно создавать директории, а также добавлять Buckets, которые выступают в роли контейнеров для файлов и директорий. В проекте используется только корневой Bucket, доступ к которому можно осуществить при помощи поля RootReference. Все файлы, находящиеся в Firebase Storage, являются предками своего контейнера Bucket, поэтому для получения ссылки на них, в ссылке на контейнер, необходимо вызвать метод *Child,* в который передается путь к файлу. Затем, файл можно создать, обновить, либо же удалить. Пример работы с файлом можно увидеть в листинге 4.16

|  |
| --- |
| public async UniTask<Texture2D> LoadImage(string path, int maxDownloadSize = 512 \* 512)  {  if (\_cache.TryGetValue(path, out var cachedTexture))  return cachedTexture;    var storageReference = \_storage.GetReference(ImagePath(path));  var bytes = await storageReference.GetBytesAsync(maxDownloadSize);  var texture = bytes.ToTexture2D();  \_cache[path] = texture;  return texture;  }  public async UniTask UploadImage(string path, Texture texture)  {  var bytes = await texture.ToBytesArray(PreviewSize.x, PreviewSize.y);  await UploadImage(path, bytes);  }  public async UniTask UploadImage(string path, byte[] bytes)  {  if (bytes != null)  {  var storageReference = \_storage.RootReference.Child(ImagePath(path));  await storageReference.PutBytesAsync(bytes);  }  }  public async UniTask DeleteImage(string path)  {  var storageReference = \_storage.GetReference(ImagePath(path));  await storageReference.DeleteAsync();  \_cache.Remove(path);  } |

Листинг 4.14 – Работа с файлами в Firebase Storage

Таким образом можно просто получать и модифицировать изображения, которые находятся в облачном хранилище Firebase Storage.

4.9 Выводы по разделу

В данном разделе были описаны элементы архитектуры приложения и их реализация. Разработка приложения осуществлялась с использованием игрового движка Uniy 2022, технологией создания интерфейсов UI Toolkit, фреймворк AR Foundation и SDK Firebase. В процессе разработки были учтены особенности работы с этими фреймворками, использованы стандартные средства разработки, а также подключаемые сторонние библиотеки.

В результате выполнения дипломного проектирования было разработано приложение для создания иллюстраций в дополненной реальности. Результатом моей работы является готовый интерфейс, который отвечает всем задачам, поставленным перед мной. В отличии от рассмотренных аналогов, приложение имеет понятный и дружелюбный к пользователю интерфейс, а также, в отличие от аналогов, работает на мобильном телефоне, без нужны подключения очков виртуальной реальности и мощного ПК.

Веб-приложение было реализовано в полном объеме, весь функционал соответствует заявленному и готово к прохождению тестирования.

5 Тестирование программного модуля

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*2*

*БГТУ 05.00.ПЗ*

Разраб*.*

*Дежемесов М.С.*

Провер*.*

*.*

*Блинова Е.А*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*СмеловВ.В.*

*Лит.*

*Листов*

*7*

74218155, 2023

*5 Тестирование программного модуля*

Тестирование программного обеспечения – процесс испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и ее ожидаемым поведением на конечном наборе тестов, выбранных определенным образом, в соответствии с качеством приложения [21].

Качество приложения – это совокупность характеристик данного приложения, относящихся к его способности удовлетворять установленные и предполагаемые потребности пользователей. Оценить уровень качества приложения и выявить различные ошибки в его работе можно с помощью тестирования.

Для процесса тестирования, тестировщик часто составляет тест-кейсы. В наиболее простом случае, тест-кейс включает в себя:

– краткое описание (то, что тестируется);

– последовательность действий (шаги, выполняемые тестировщиком для проверки функциональности);

– ожидаемый результат.

Выполнив последовательность действий из тест-кейса, тестировщик получает фактический результат, который сравнивается с ожидаемым. Несовпадение фактического результата с ожидаемым означает то, что в программе обнаружен баг.

В данном разделе описано тестирование функционала веб-приложения, разработанного в рамках данного дипломного проекта.

## 5.1 Ручное тестирование

Как показывает практика тестирования программного обеспечения, наиболее корректной работы приложения можно добиться при ручном тестировании на реальных устройствах [22]. Для отображения тестируемых элементов были составлены таблицы тест-кейсов. В таблице 5.1 Представлены тесты для создания иллюстраций, их настройки, просмотра уже существующих иллюстраций, а также действий с аккаунтом пользователя.

Таблица 5.1 – Функциональные тест-кейсы

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Статус |
| Выбор инструмента рисования | Изменение текущего инструмента для рисования иллюстраций | Успешно |
| Изменение цвета инструмента | Смена цвета инструмента создания иллюстраций | Успешно |

Окончание таблицы 5.1

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Статус |
| Изменение толщины линии | Изменение толщины для текущего инструмента для рисования иллюстраций | Успешно |
| Изменение стиля кисти рисования | Смена стиля инструмента создания иллюстраций | Успешно |
| Изменения гладкости кисти | Края и углы линии изменили свою гладкость | Успешно |
| Сохранение изменений в рисунке | Переход на главный экран, превью рисунка изменилось, окно просмотра отображает измененную версию иллюстрации | Успешно |
| Просмотр информации о рисунке | Открытия окна просмотра информации о рисунке | Успешно |
| Вращение превью в меню информации рисунка | Изображение рисунка вращается и следует за вводом пользователя | Успешно |
| Создание рисунка с пустым именем | Вывод сообщения об ошибке, с просьбой ввода наименования рисунка | Успешно |
| Редактирование информации о рисунке | Переход на главный экран, название рисунка изменилось, окно просмотра отображает измененную информацию о рисунке | Успешно |
| Просмотр списка всех рисунков пользователей | Открывается меню выбора рисунка, при выборе рисунка пользователь не может редактировать рисунок | Успешно |
| Ввод невалидной почты при регистрации | Вывод окна с ошибкой и просьбой ввести правильный адрес | Успешно |
| Регистрация с несколькими пустыми полями | Вывод сообщения с просьбой заполнить все поля | Успешно |
| Ввод невалидного пароля при регистрации | Вывод ошибки и просьбой ввести действительный пароль | Успешно |
| Ввод неактуальных данных при входе | Вывод окна ошибки аутентификации | Успешно |
| Экспортирование рисунка в PNG | В галерее появится изображение рисунка с углом поворота, установленным в окне просмотра иллюстрации | Успешно |
| Удаление рисунка | Опубликованный рисунок удаляется из базы данных, удаление рисунка из локального хранилища пользователя | Успешно |
| Публикация рисунка | Создается запись рисунка в базе данных. Рисунок появляется в окне поиска рисунков | Успешно |
| Поиск рисунка по названию | Выводится список всех доступных рисунков, соответствующих поиску | Успешно |

Описанные выше тест-кейсы помогают протестировать систему прямо из интерфейса, что значительно уменьшает время на написание тестов.

5.2 Тестирование валидации

Валидация – это процесс проверки значений по определенному ранее стандарту, требованию, правилу. Существует три вида валидации:

* мгновенная – валидация происходит одновременно с пользовательским вводом;
* после потери фокуса – валидация происходит после переключения на другое поле или после нажатия на другую область;
* после отправки формы – процесс валидации начинается после нажатия кнопки отправки формы.

В разработанном приложении для создания иллюстраций в дополненной реальности в основном используется валидация после отправки формы.

Результат ввода пустого имени пользователя показан на рисунке 5.1.

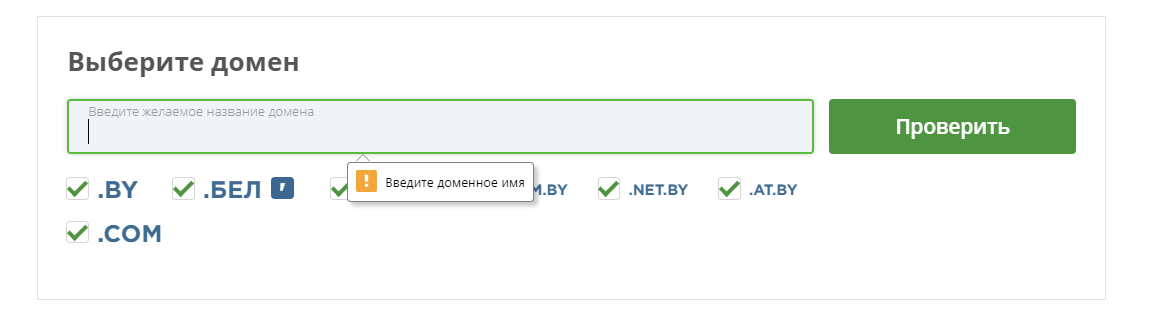


Рисунок 5.1 – Результат нажатия на кнопку «Register» при пустом имени пользователя

Также пользователи могут допускать ошибки при вводе электронной почты, например вставлять спецсимволы, которые запрещены в использовании, либо забыть написать домен. На рисунке 5.2 представлен вид страницы регистрации при введении пользователем почты неверного формата.

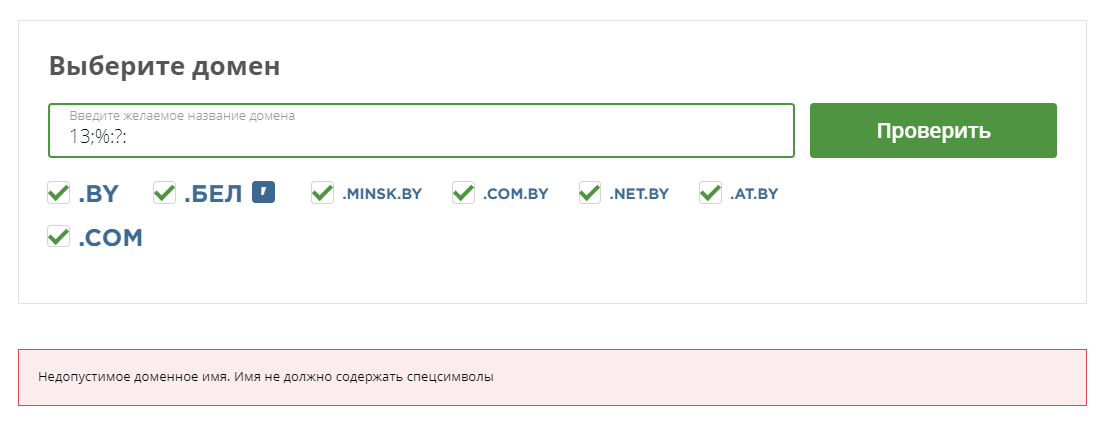


Рисунок 5.2 – Результат валидации неверного формата почты

При авторизации, пользователю необходимо ввести свой логин и пароль. Если пользователь ввел некорректные данные, будет выведено всплывающее окно с ошибкой авторизации пользователя (рисунок 5.3).

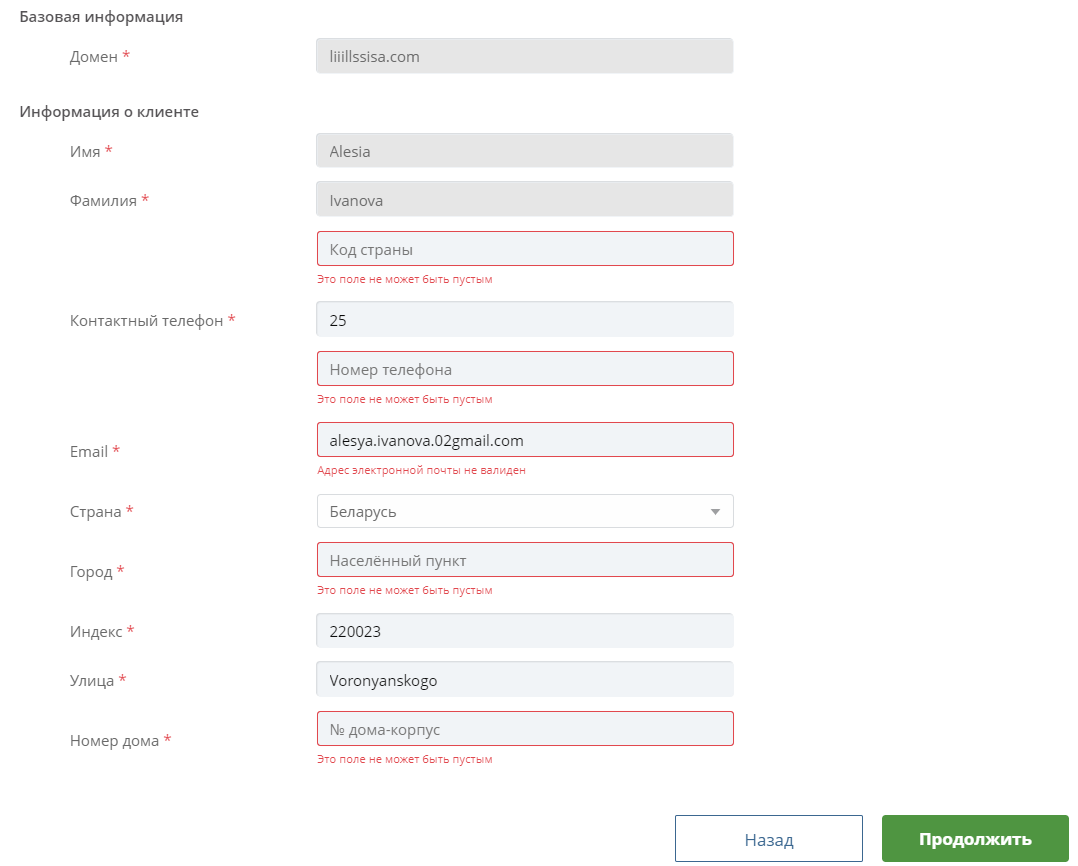


Рисунок 5.3 – Результат валидации неверных пользовательских данных

При создании нового рисунка, пользователю необходимо придумать его наименование. При попытке перехода в режим рисования без заполненного имени иллюстрации, пользователю будет выведено всплывающее окно с ошибкой. Кроме того, при создании нового рисунка, пользователю недоступны опции сохранения, удаления и публикации иллюстрации (рисунок 5.4)

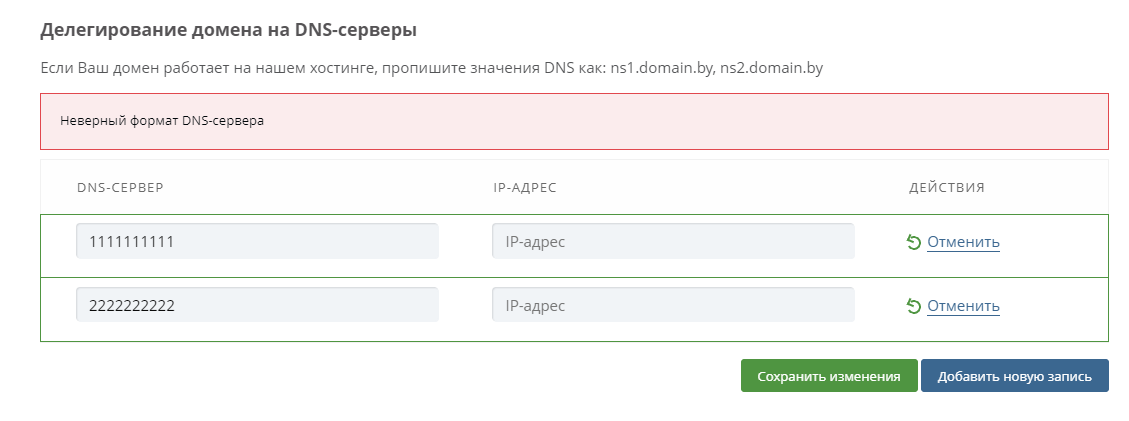


Рисунок 5.4 – Результат валидации неверных данных о рисунке

Пользователь так же не может редактировать информацию о рисунке, если он не является автором рисунка (рисунок 5.5)

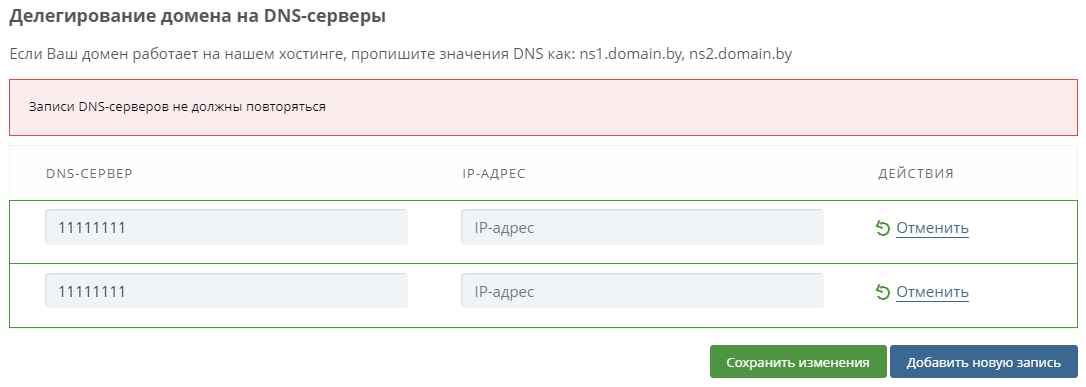


Рисунок 5.5 – Окно просмотра чужой иллюстрации

Валидация также присутствует и на странице переделегирования международного домена пользователю, который не зарегистрирован в *OpenProvider*. На рисунке 5.7 представлен результат валидации неверных данных.

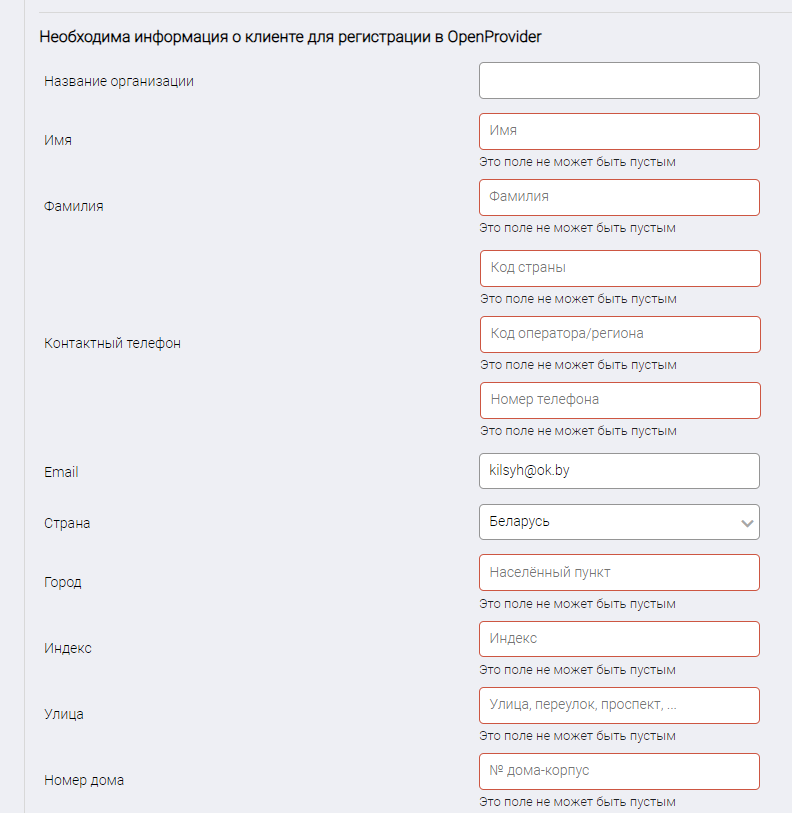


Рисунок 5.7 – Результат ввода невалидных данных пользователя

В этом же окне есть проверка на передачу домена не ресейлерам. Дело в том, что домены международного и национального уровня не могут передаваться ресейлерам.

Валидирование данных требовалось и на интерфейсе для управления тарифными планами, а именно на странице создания/редактирования тарифного плана. На рисунке 5.8 показан вид окна редакции тарифного с введенными неправильными данными.

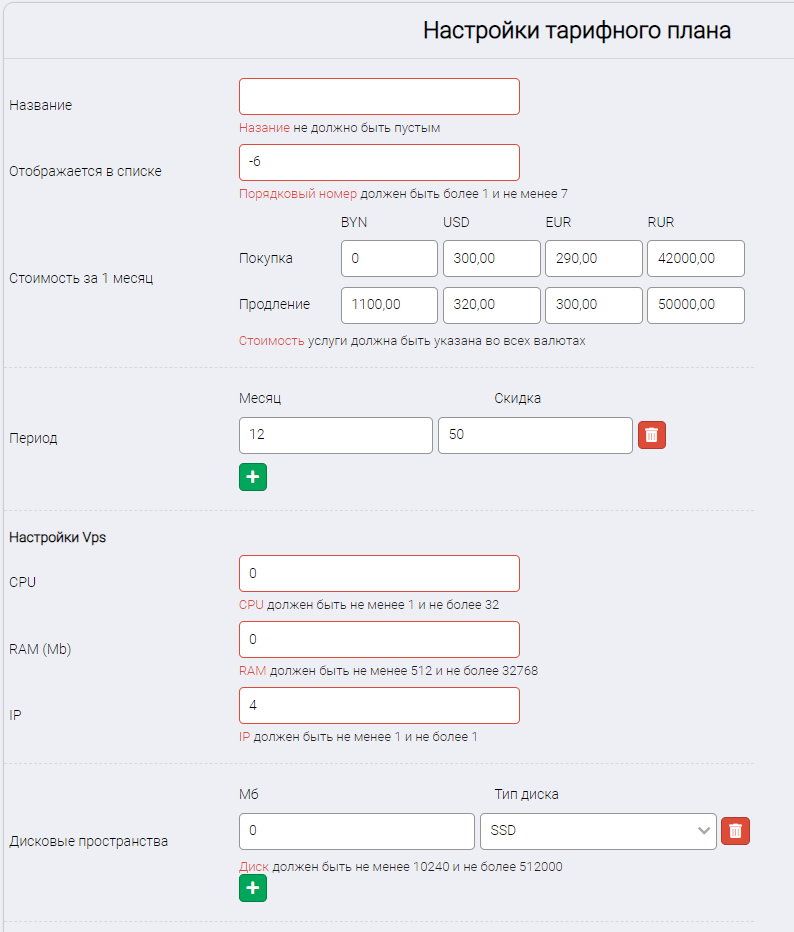


Рисунок 5.8 – Вид окна редакции после тестирования его пустыми значениями

Также необходимо протестировать окно редакции на введение отрицательных чисел в поле «Стоимость за 1 месяц», введение названия тарифного которое уже существует в поле «Название», введение не корректных данных в поля «*CPU*» «*RAM*» и «*IP*». Результат тестирования показан на рисунке 5.9.

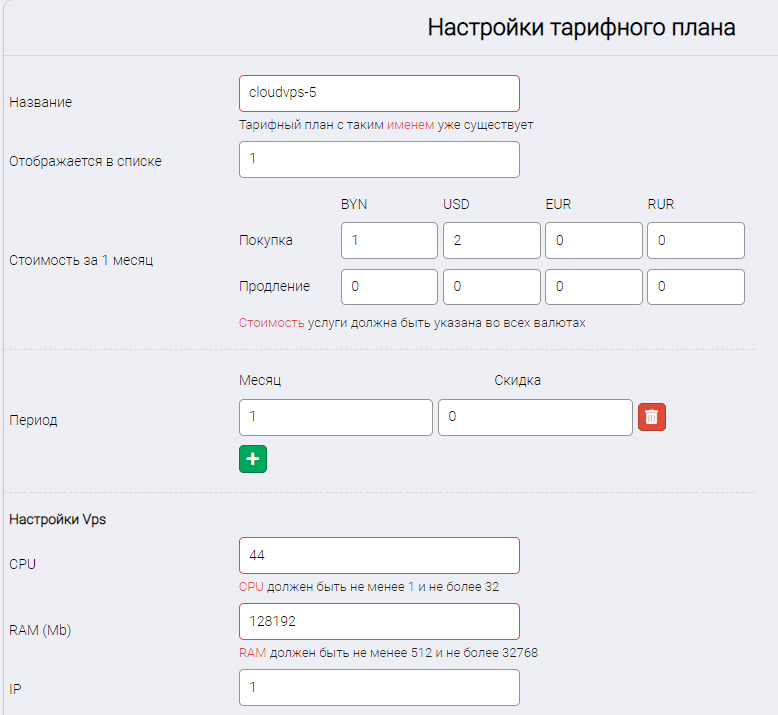


Рисунок 5.9 – Результат тестирования полей

Далее, кроме ручных тестов и тестов валидации, необходимо провести автоматическое тестирование.

5.3 Выводы по разделу

В данном разделе был описан процесс тестирования разработанного модуля управления международными доменами платформы *Domain.by*. Тестирование проводилось по предварительно написанным тест-кейсам, а его результаты помещены в таблицу 5.1. Также в процессе тестирования проводилась проверка валидации форм. Кроме ручного тестирования над задачами проводилось автоматическое тестирование. Все вышеперечисленные виды тестирования могут уверять в том, что модуль работает корректно и шанс появления непредвиденных ошибок у пользователей минимизирован.

Полученные результаты при ручном и автоматическом функциональном тестировании показали, что разработанное программное средство работает правильно и корректно.

6 Руководство пользователя

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*2*

*БГТУ 06.00.ПЗ*

Разраб*.*

*Дежемесов М.С.*

Провер*.*

*.*

*Блинова Е.А*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*СмеловВ.В.*

*Лит.*

*Листов*

*16*

74218155, 2023

*6 Руководство пользователя*

В данном разделе дипломного проектирования, будут рассмотрены основные аспекты работы с приложением. Для удобства изучения, раздел разбит на подразделы в соответствии с существующими ролями пользователей.

6.1 Общие сведения

Дипломное проектирование заключается в создании функционала для управления международных доменов и тарифных планов, реализованного в виде логического модуля. Функционал модуля разделен в зависимости от ролей пользователя: администратора и пользователя.

6.2 Роль «Пользователь»

В рамках разработанного программного модуля пользователь может выполнять следующие задачи:

* регистрирование международного домена;
* продление регистрации доменов;
* редактирование *NS* адресов для домена;
* оплата услуг по регистрации домена;
* получение сертификата о регистрации домена.

Для выполнения всех этих задач, пользователю необходимо авторизоваться на платформе Domain.by после чего пользователь окажется в личном кабинете, где он может заказывать или оплачивать услуги, а также менять их конфигурации. Главная страница личного кабинета показана на рисунке 6.1.

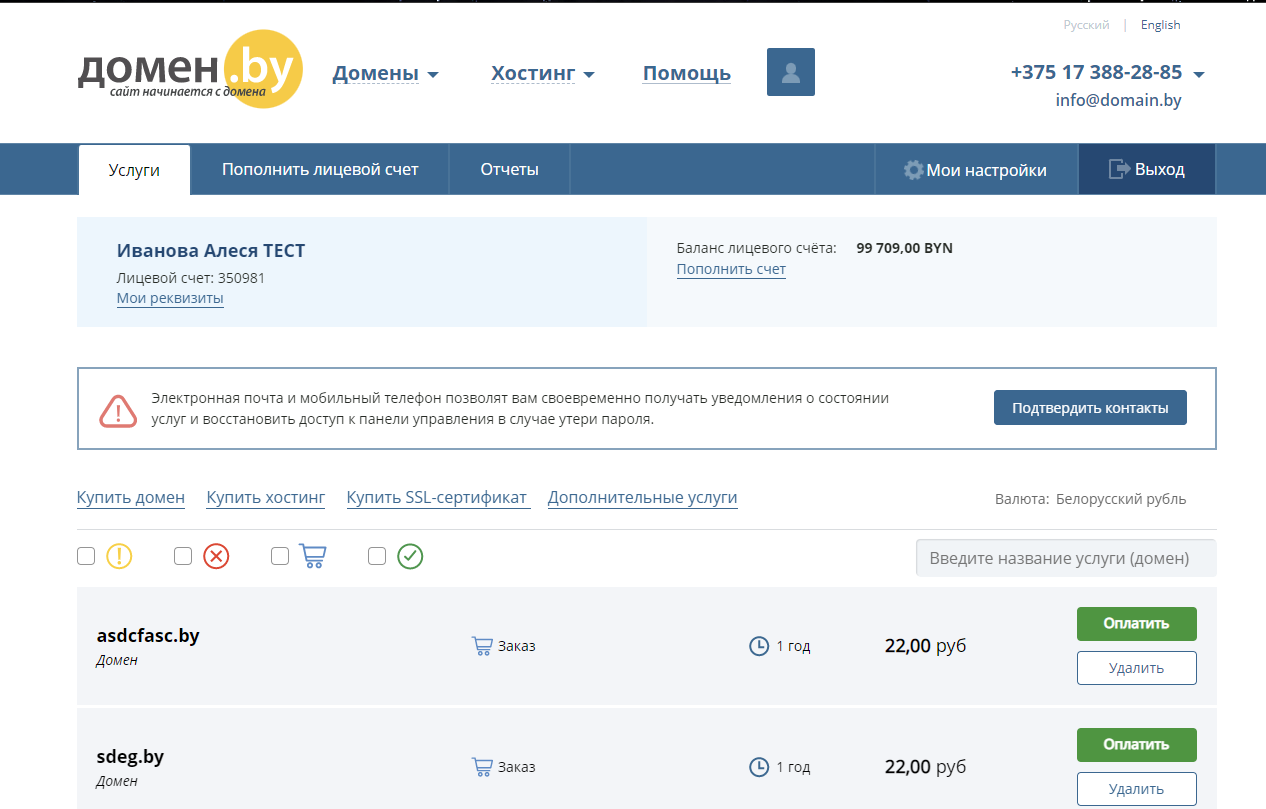


Рисунок 6.1 – Главная страница личного кабинета

Для того, чтобы оформить покупку домена, необходимо перейти на страницу покупки домена через нажатие на соответствующую кнопку: «Купить домен» в личном кабинете. Далее пользователя переадресует на страницу покупки домена, где он может выбрать нужные доменные зоны и доменное имя. Страница выбора фильтров для домена показана на рисунке 6.2

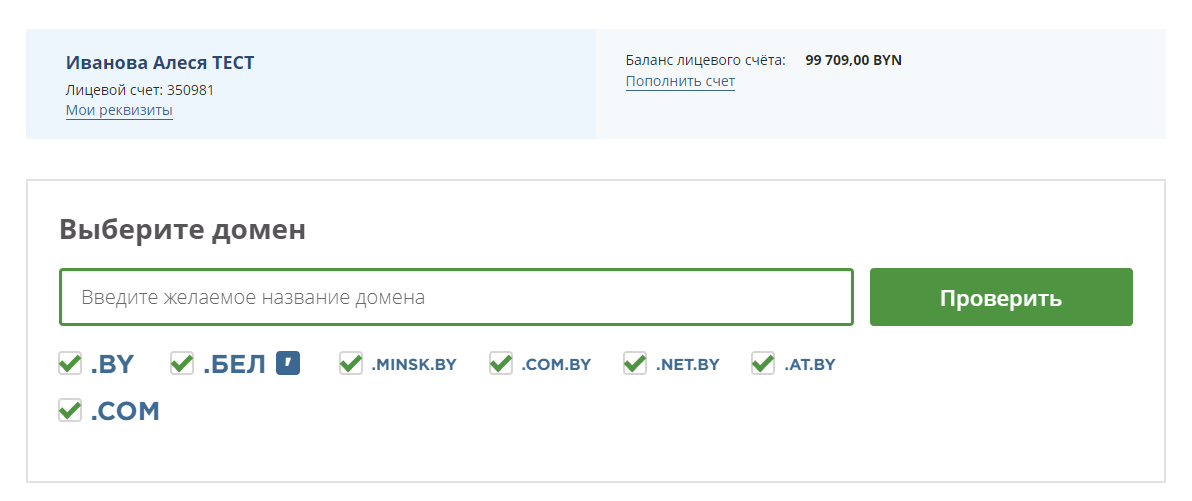


Рисунок 6.2 – Страница выбора названия домена и доменных зон для регистрации

Далее после введения имени и выбора доменных зон, необходимо нажать на кнопку «Проверить», которая переадресует пользователя на страницу с таблицей со статусами доступности доменов. Скриншот этой страницы изображен на рисунке 6.3. На этой странице происходит выбор нужного домена и его периода регистрации. Также если домен в предпочитаемой доменной зоне занят, в таблице под названием «Похожие домены», будет предложен домен с похожим названием и нужной доменной зоной.

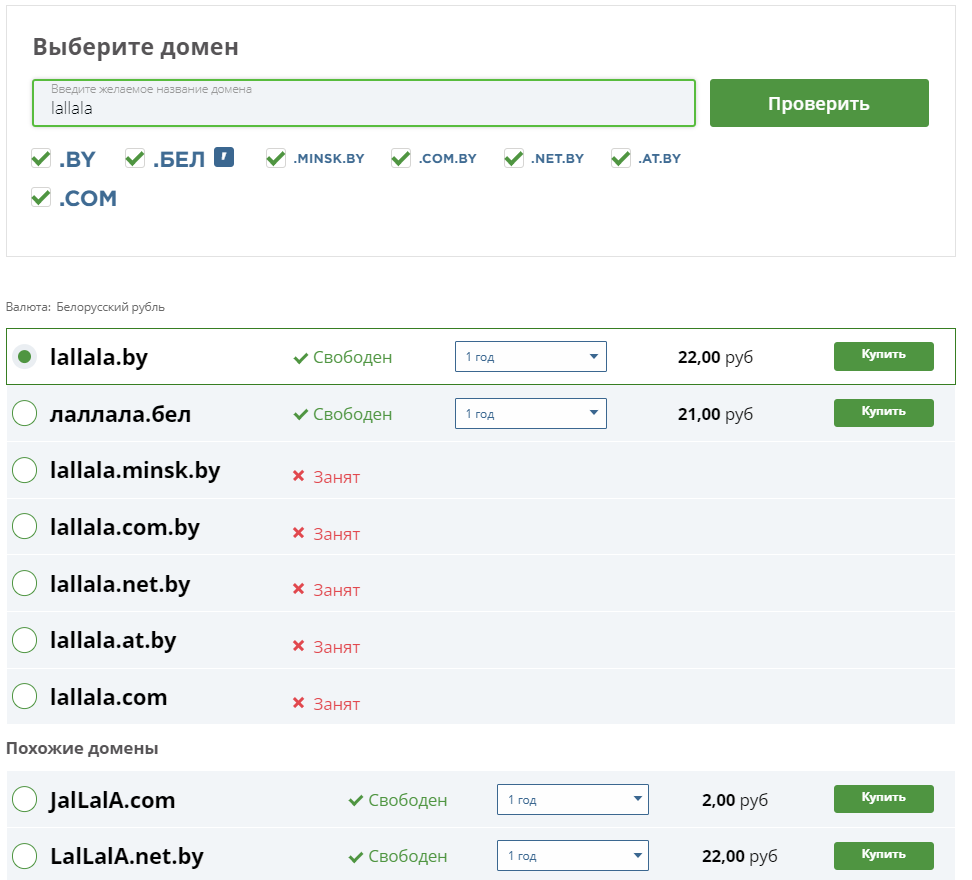


Рисунок 6.3 – Страница выбора домена и периода регистрации

Если домен в национальной зоне занят, пользователю предоставляется ссылка на сервис *WhoIs*. В этом сервисе можно выяснить кто является владельцем домена и когда у домена истекает срок регистрации. На рисунке 6.4 предоставлен скриншот страницы, на которой национальный домен не доступен для регистрации.

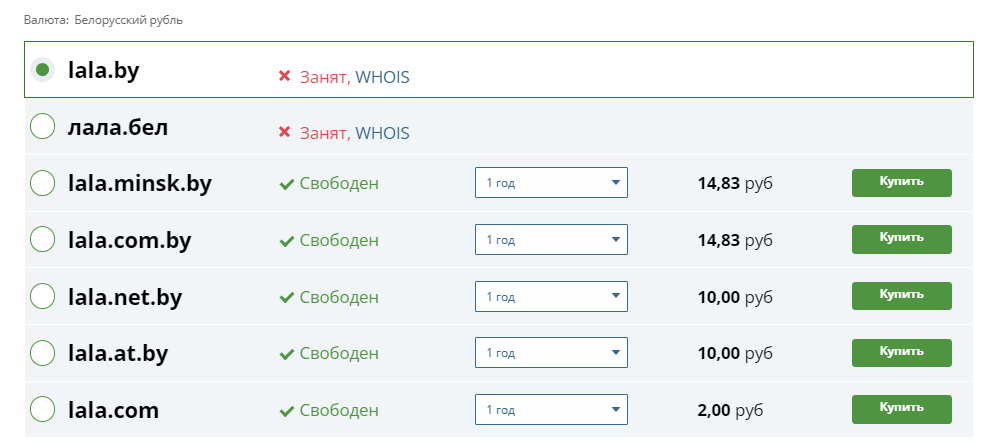


Рисунок 6.4 – Таблица доменов с недоступным для регистрации доменом национальной зоны

После выбора периода и домена, пользователю необходимо нажать на кнопку «Купить», после чего его переадресует на одну из двух страниц, изображенных на рисунках 6.5 и 6.6. Такое разделение присутствует из-за того, что пользователям, которые регистрируют домен в международной зоне от *OpenProvider* необходимо заполнить форму регистрации. На рисунке 6.5 представлен скриншот страницы при регистрации международного домена.

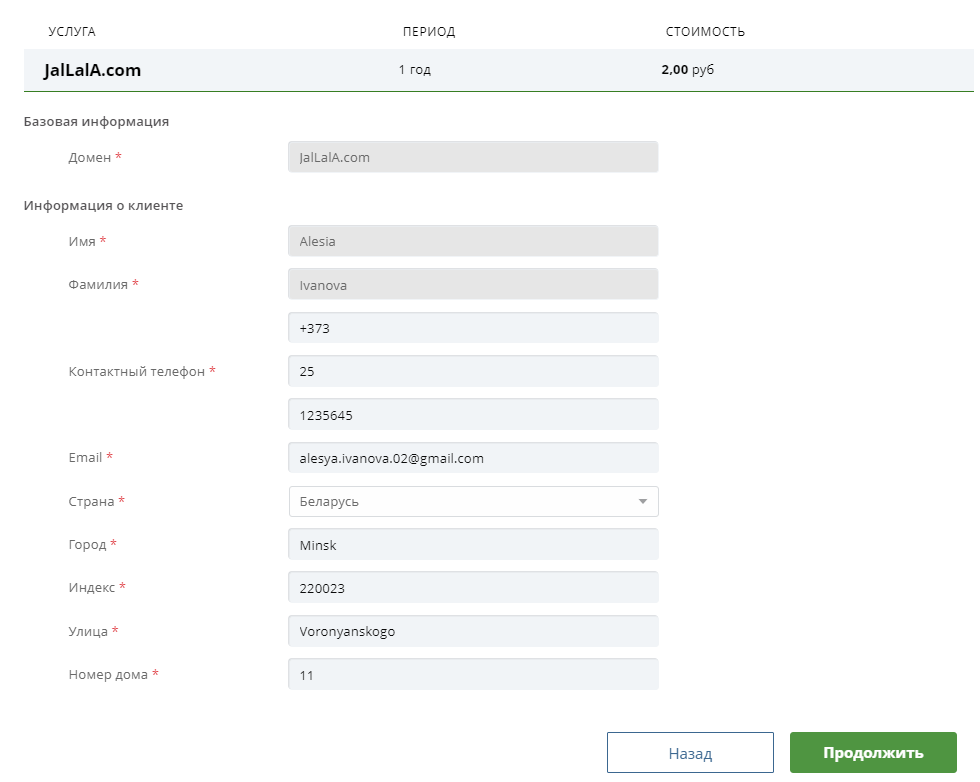


Рисунок 6.5 – Страница подтверждения и заполнения данных для регистрации в *OpenProvider*, а также подтверждение заказа услуги по регистрации международного домена

При регистрации домена из национальной зоны, пользователя переадресует на страницу, изображенную на рисунке 6.6.

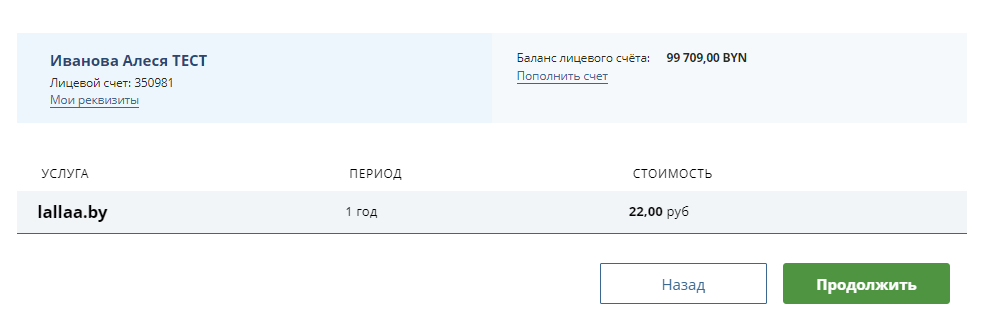


Рисунок 6.6 – страница подтверждения заказа услуги по регистрации домена международной зоны

После нажатия кнопки «Продолжить» на обеих страницах, интерфейс переадресует пользователя на страницу для оплаты услуги любым удобным для него способом. На рисунке 6.7 представлен скриншот этой страницы.

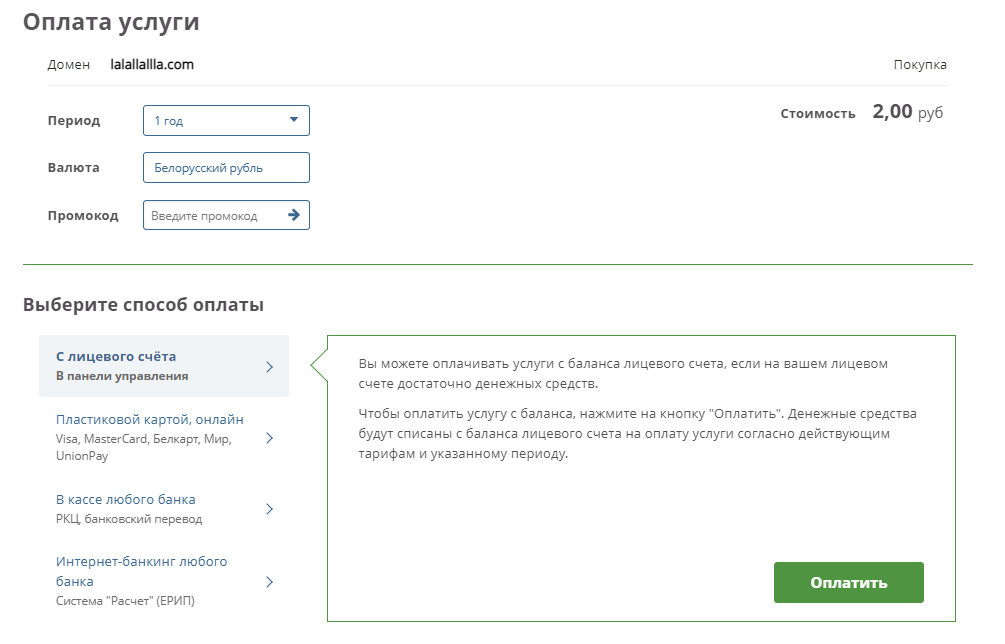


Рисунок 6.7 – Страница оплаты услуг

После оплаты услуг у пользователя на главной странице личного кабинета (рисунок 6.1) появится домен с его возможностью настройки. На рисунке 6.8 представлена строка таблицы услуг с доменом который оплачен.

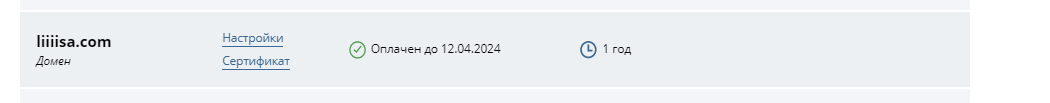


Рисунок 6.8 – Строка таблицы с оплаченным доменом

Если оплата не прошла, в таблице с услугами домен будет представлен в ином виде. На рисунке 6.9 представлен пример такого домена. Домен можно будет оплатить путем нажатия на кнопку «Оплатить» и после чего его переадресует на страницу оплаты, изображенную на рисунке 6.7. Либо пользователь может удалить услугу, путем нажатия на кнопку «Удалить».



Рисунок 6.9 – Строка таблицы услуг с неоплаченным доменом

Чтобы изменить адрес *DNS* сервера, или, иными словами, *NS* запись, необходимо нажать на кнопку «Настройки» (рисунок 6.8), в строке с оплаченным доменом. Далее интерфейс перенаправит пользователя на страницу с настройками домена, скриншот интерфейса изображен на рисунке 6.10. На этой странице можно управлять всеми настройками домена.

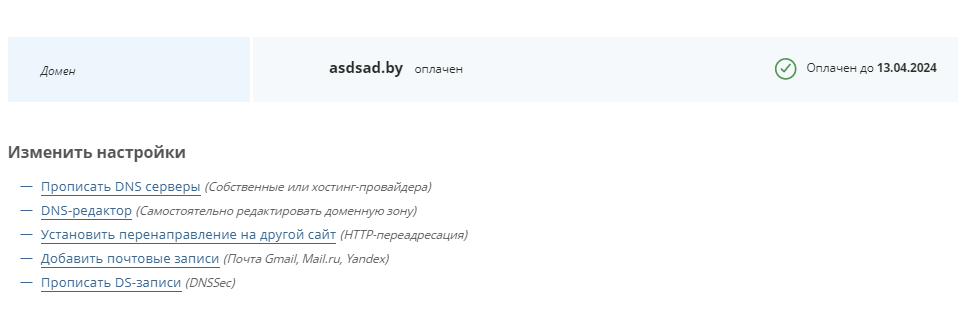


Рисунок 6.10 – Страница настроек домена

Чтобы изменить *NS* запись, пользователю необходимо нажать на кнопку «Прописать *DNS* серверы», после чего его переадресует на страницу, где можно прописать *NS* записи. Скриншот этой страницы изображен на рисунке 6.11.

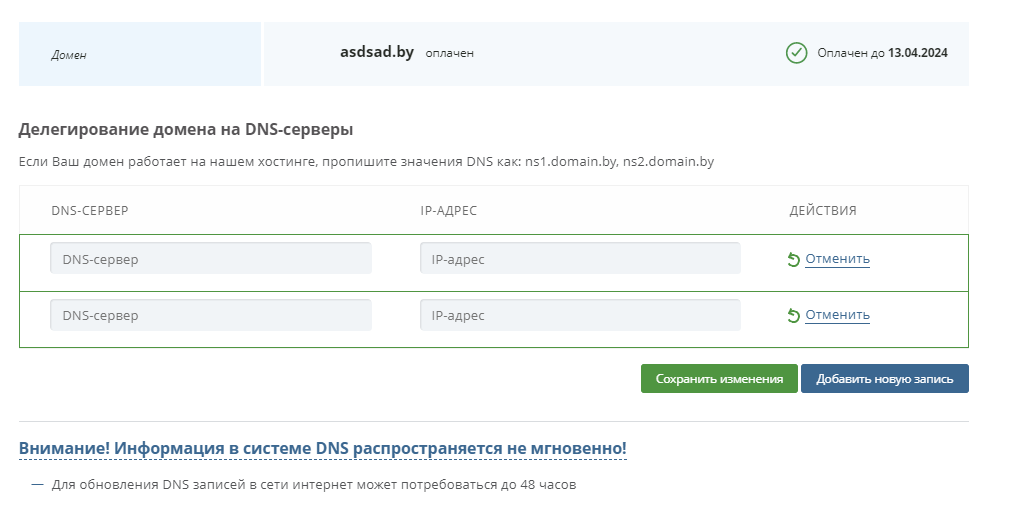


Рисунок 6.11 – Страница изменения *NS* записей домена

После того, как пользователь введет адреса серверов, ему необходимо нажать на кнопку «Сохранить изменения», если пользователю необходимо прописать более двух адресов для того, чтобы добавить еще поля для записи он может нажать на кнопку «Добавить новую запись».

Чтобы получить сертификат о регистрации домена, на строке с нужных доменов в таблице всех услуг (рисунок 6.8) необходимо нажать на кнопку «Сертификат». После чего сертификат о регистрации будет автоматически скачан в формате *pdf* файла на устройство пользователя. Сертификат о регистрации изображен на рисунке 6.12.



Рисунок 6.12 – Сертификат о регистрации домена

Для того, чтобы продлить регистрацию домена. За 4 дня до окончания регистрации домена, пользователь в личном кабинете на строке с нужным доменом должен нажать на появившуюся кнопку «Продлить» (рисунок 6.13).



Рисунок 6.13 – Строка таблицы услуги с доменом, доступным для продления

Далее пользователя переадресует на страницу оплаты услуги продления регистрации домена, скриншот страницы оплаты изображен на рисунке 6.7.

6.3 Роль «Администратор»

В рамках разработанного программного модуля администратор может выполнять все задачи пользователя, а также следующие задачи:

* создание тарифного плана;
* удаление тарифного плана;
* изменение тарифного плана;
* проведение процедуры переделегации домена от пользователя к пользователю платформы *Domain.by*;
* просмотр данных доменов;
* создание заметки к доменам;
* просмотр реквизитов владельца домена.

Для выполнения всех этих задач, пользователю необходимо авторизоваться как администратор на платформе *Domain.by* после чего пользователь окажется на главной странице, скриншот которой изображен на рисунке 6.14.



Рисунок 6.14 – Домашняя страница администратора

Как видно из рисунка 6.14 у администратора есть меню, в котором он может вести управление всех, необходимых ему, аспектов. Стоит отметить, что у приложения есть еще роли две «менеджер» и «бухгалтер». У этих ролей в боковом меню есть ссылки на управление только тех, аспектов, которые подразумеваются под их ролью.

Для проведения процедуры переделегирования, пользователю необходимо нажать на вкладку домены (рисунок 6.14) и из выпадающего списка, изображенного на рисунке 6.15, выбрать «Переделегирование домена».

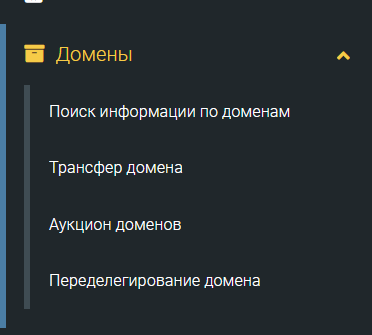


Рисунок 6.15 – Доступные операции для управления доменами

После нажатия на кнопку «Переделегирование домена», интерфейс переадресует администратора на страницу переделегации домена, изображенной на рисунке 6.16. На этой странице пользователю необходимо выбрать и ввести условие для поиска домена для переделегирования домена.

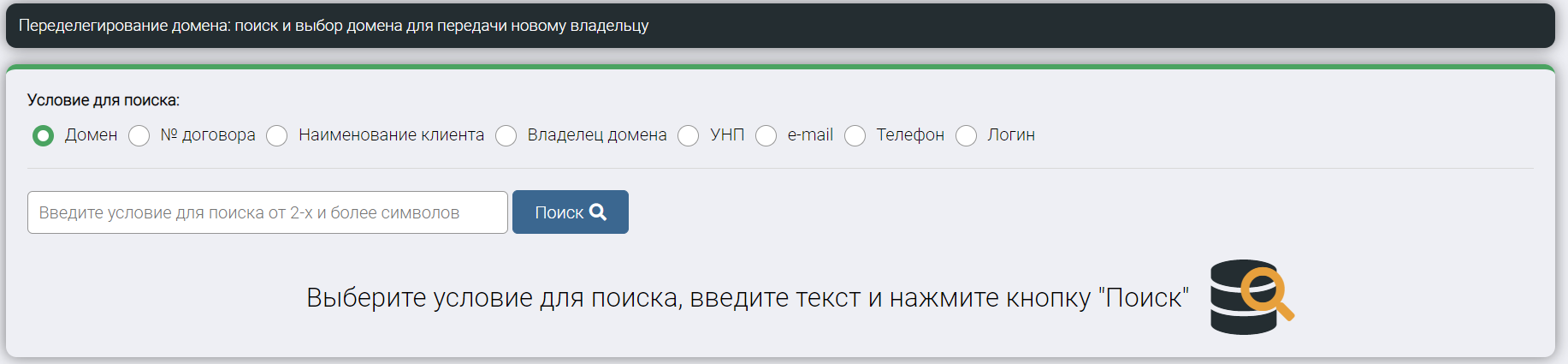


Рисунок 6.16 – Страница переделегирования домена

После ввода условий для поиска домена и нажатия на кнопку «Поиск» – перед пользователем предстает таблица с доменами, соответствующими условию поиска. На рисунке 6.17 изображена эта таблица.

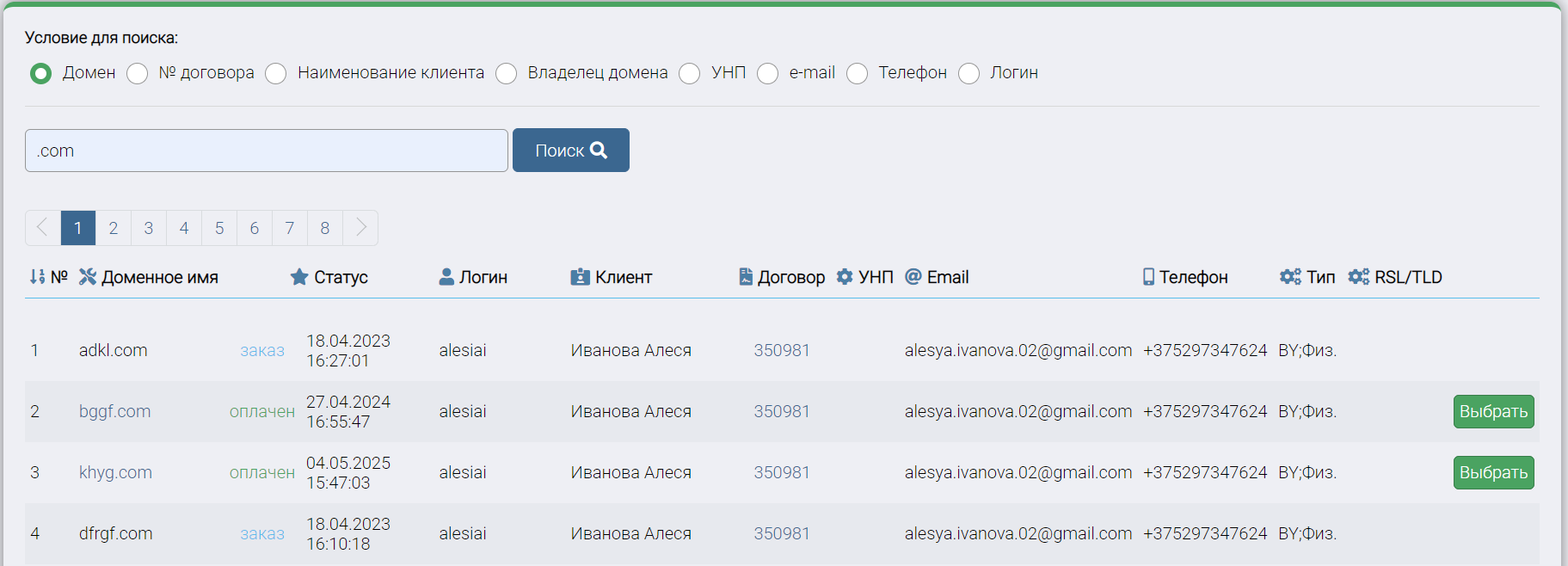


Рисунок 6.17 – Таблица доменов, удовлетворяющих условия поиска

Как видно из рисунка 6.17, домены, которые находятся в статусе «заказ» недоступны для переделегации. Чтобы выбрать домен для переделегирования, необходимо нажать кнопку «Выбрать» в строке с нужным доменом. После чего, интерфейс покажет данные домена и его владельца, скриншот этого окна изображен на рисунке 6.18.

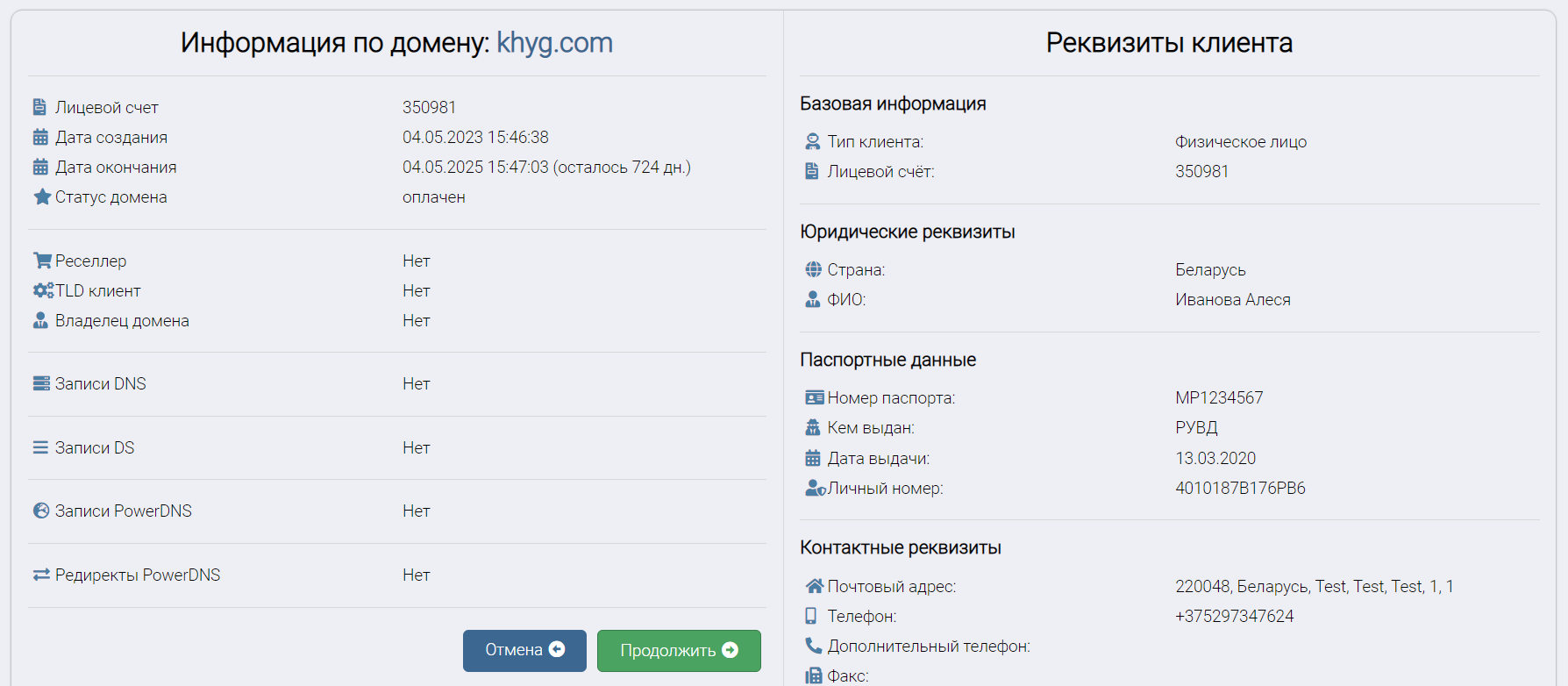


Рисунок 6.18 – Окно для просмотра информации о домене и его владельце

Для продолжения операции переделегирования необходимо нажать на кнопку «Продолжить», и далее выбрать пользователя для переделегации домена. Интерфейс для поиска пользователя показан на рисунке 6.19.

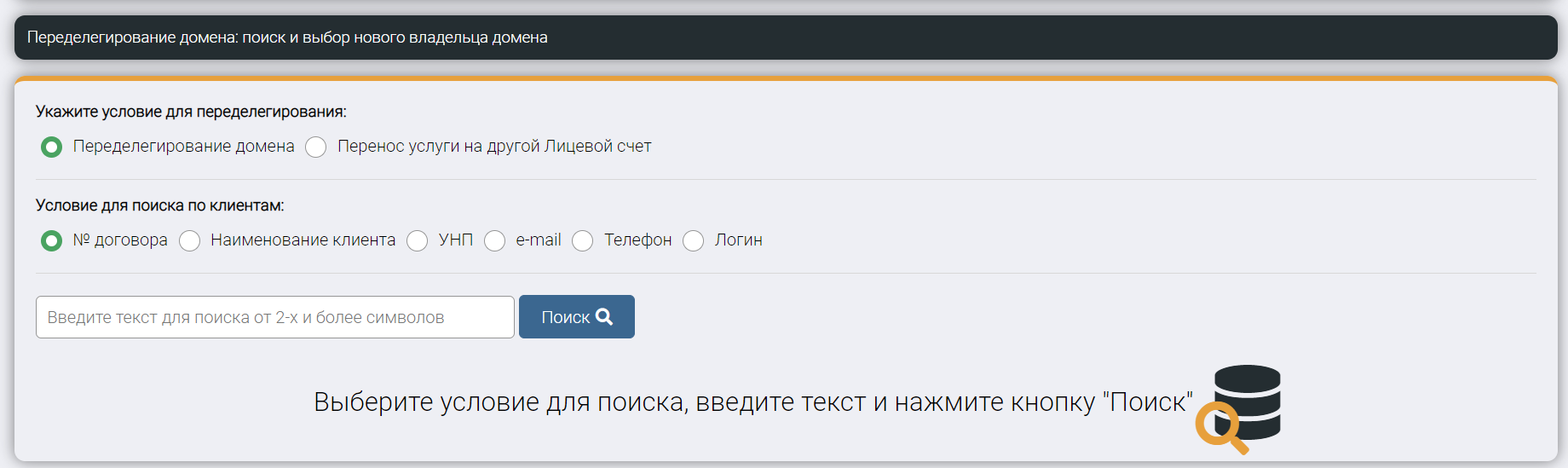


Рисунок 6.19 – Страница поиска пользователя для переделегации домена

После введения нужных данных для поиска пользователя, которому будет делегирован домен, администратору необходимо выбрать нужного из таблицы, скриншот таблицы представлен на рисунке 6.20.

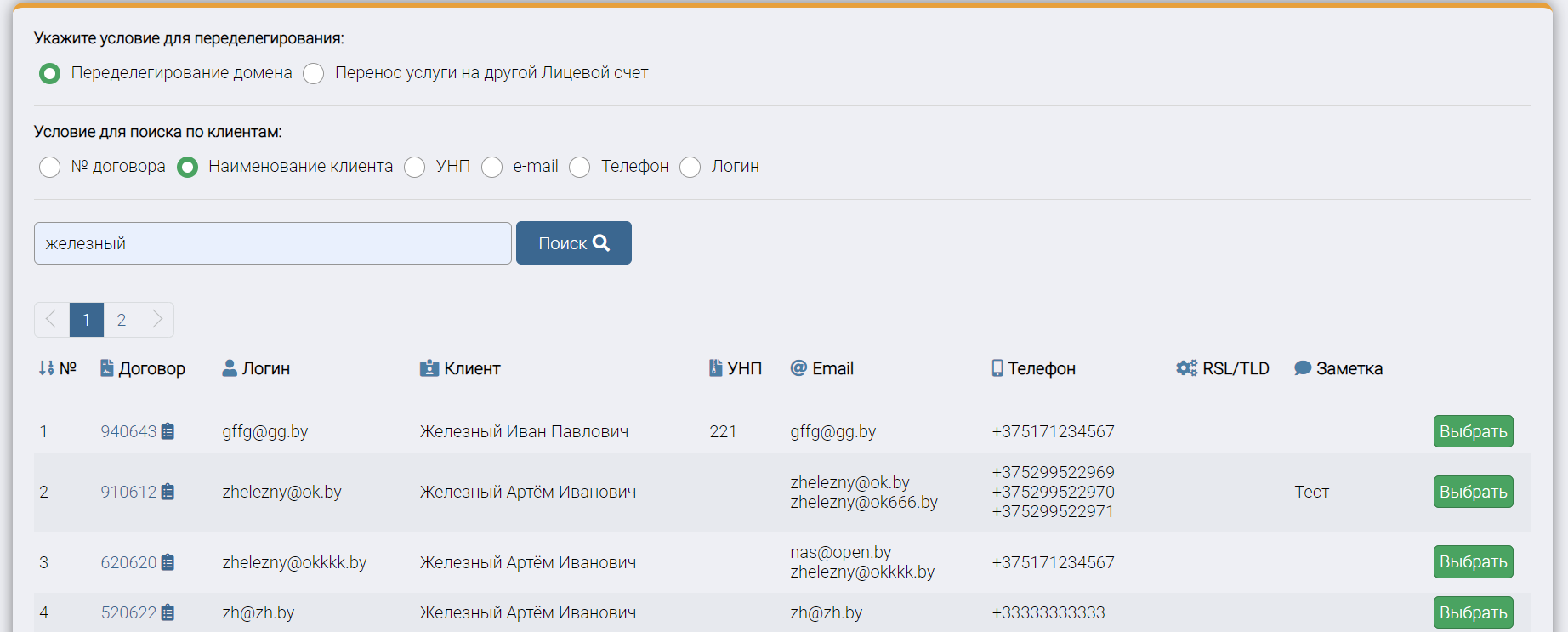


Рисунок 6.20 – Таблица пользователей платформы *Domain.by*

Далее нужно нажать на кнопку «Выбрать» рядом с нужным клиентом. После чего администратору будет представлен один из двух интерфейсов, представленных на рисунках 6.21 и 6.22. Если домен для переделегации является национальным или международным и клиент уже зарегистрирован в системе *OpenProvider* – администратор увидит интерфейс, изображенный на рисунке 6.21, если домен международный и клиент для переделегации не зарегистрирован в системе *OpenProvider* – администратор увидит интерфейс, изображенный на рисунке 6.22.

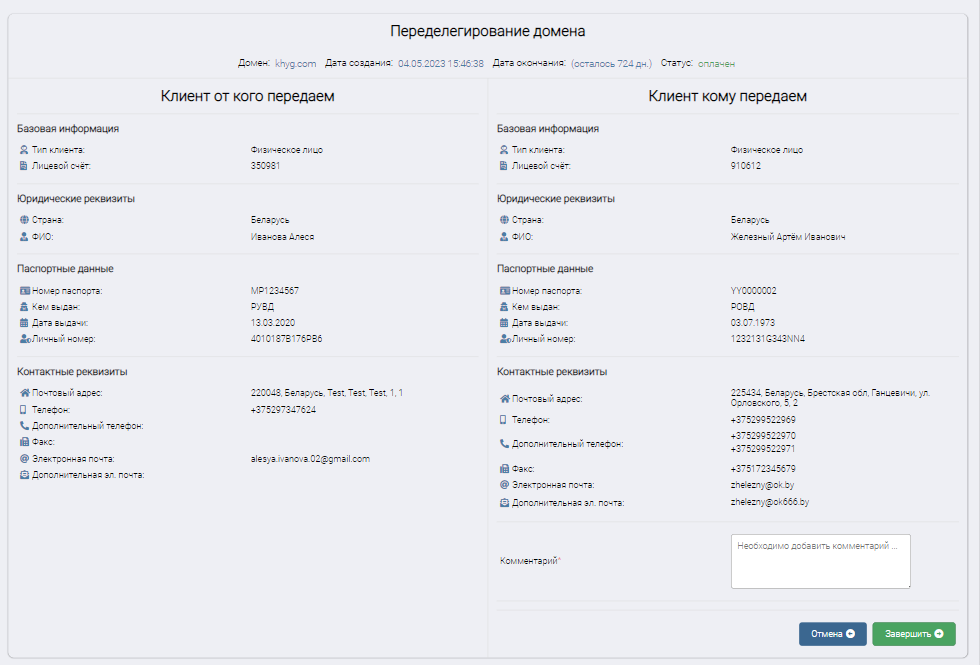


Рисунок 6.21 – Переделегирование пользователю, у которого нет необходимости в регистрации в системе *OpenProvider*

На рисунке 6.22 изображена страница переделегации пользователя с формой для регистрации клиента в системе *OpenProvider*. Появляется она только в том случае если пользователь не зарегистрирован в системе.

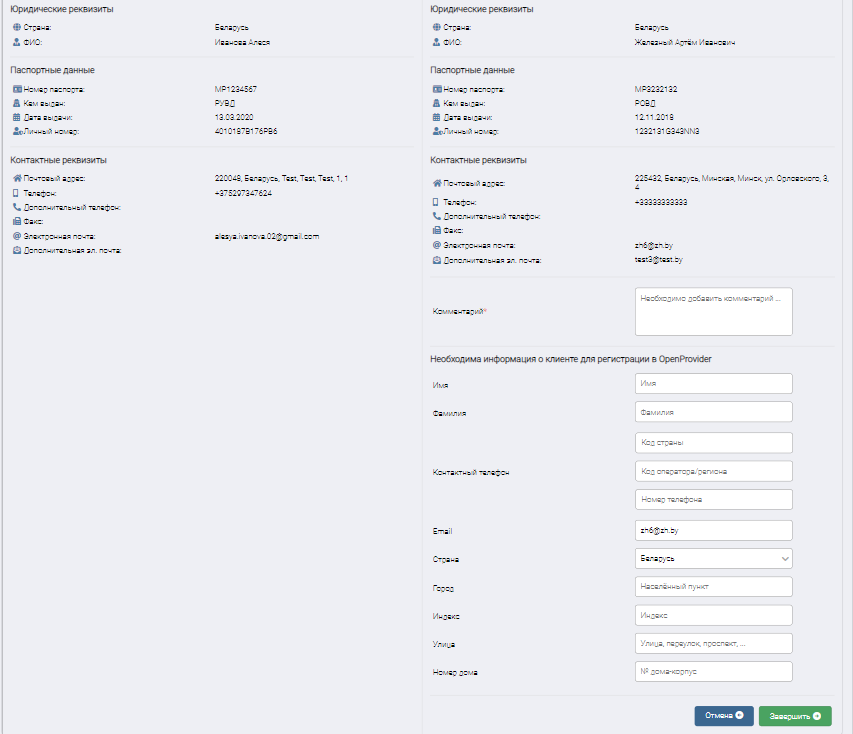


Рисунок 6.22 – страница конечного этапа переделегации с регистрационной формой для *OpenProvider*

Чтобы подтвердить тот факт, что выбран нужный пользователь для делегирования домена – нужно нажать на кнопку «Завершить», изображенную на рисунках 6.21 или 6.22. После чего, пользователю будет показано сообщение об успешной переделегации домена, изображенное на рисунке 6.23.

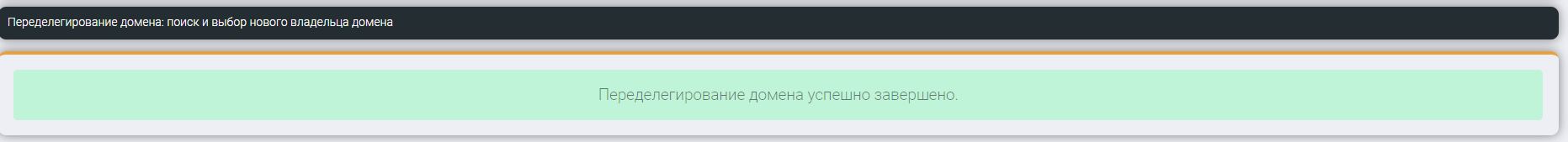


Рисунок 6.23 – Сообщение об успешной переделегации домена

Для управления тарифными планами, необходимо перейти на страницу «Тарифные планы». Чтобы это сделать, необходимо на странице, изображенной на рисунке 6.14 навести на раздел «Настройки» и выбрать пункт «Тарифные планы» (рисунок 6.24), после чего пользователь окажется на странице для управления тарифными планами.

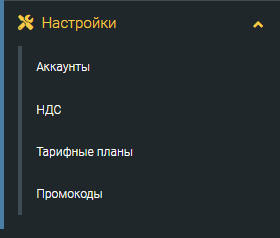


Рисунок 6.24 – Вкладка «Настройки»

На рисунке 6.25 представлена страница, которая появляется после нажатия на кнопку «Тарифные планы» в разделе «Настройки». Активные и неактивные тарифные планы выделены индикатором возле названия тарифного плана.

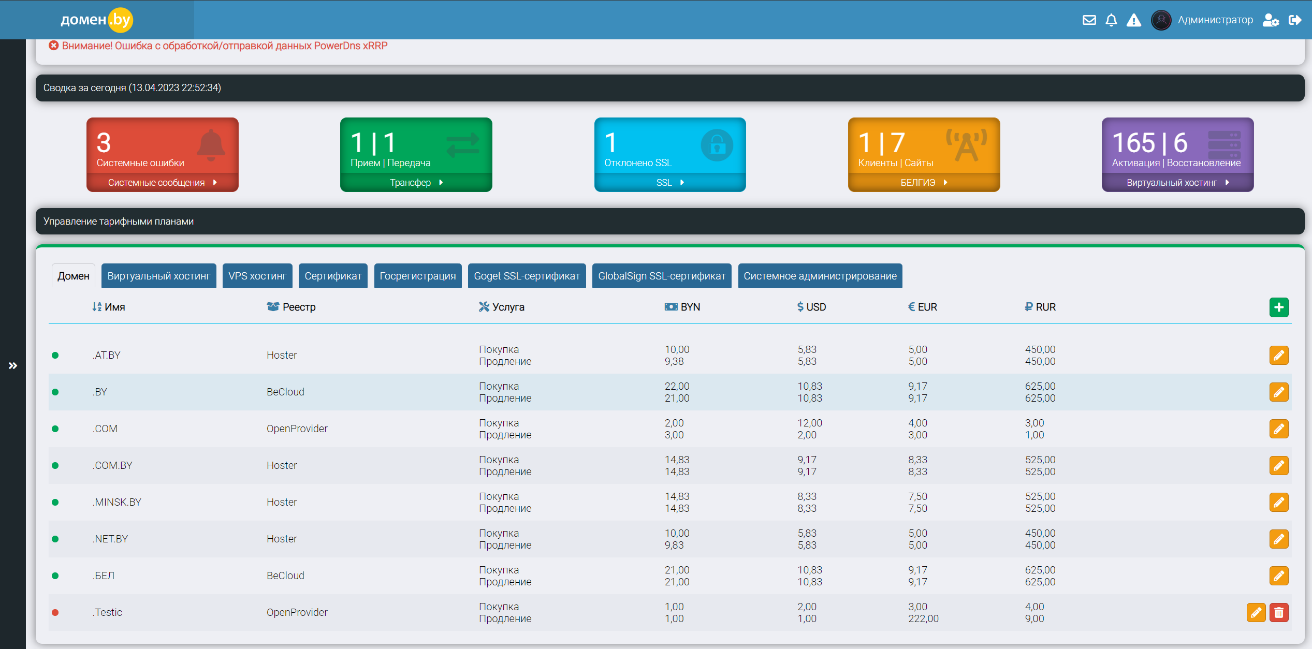


Рисунок 6.25 – Вид страницы «Тарифные планы»

Как видно из рисунка 6.25, не все тарифные планы можно удалить. На рисунке 4.26 представлен вид окна просмотра и редактирования тарифного плана, который нельзя изменить полностью. Для открытия страницы редакции тарифного плана нужно нажать на иконку с изображением карандаша возле тарифного плана.

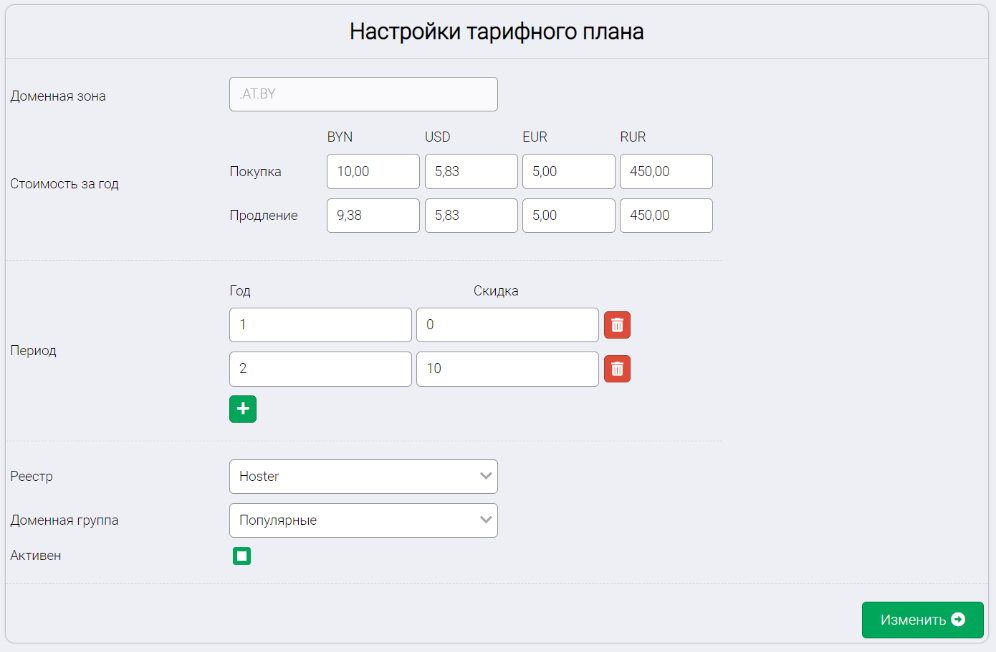


Рисунок 6.26 – Вид окна просмотра и редактирования тарифного плана, который нельзя полностью изменить

На рисунке 6.27 представлен вид окна просмотра и редактирования тарифного плана, который доступен для полного редактирования.

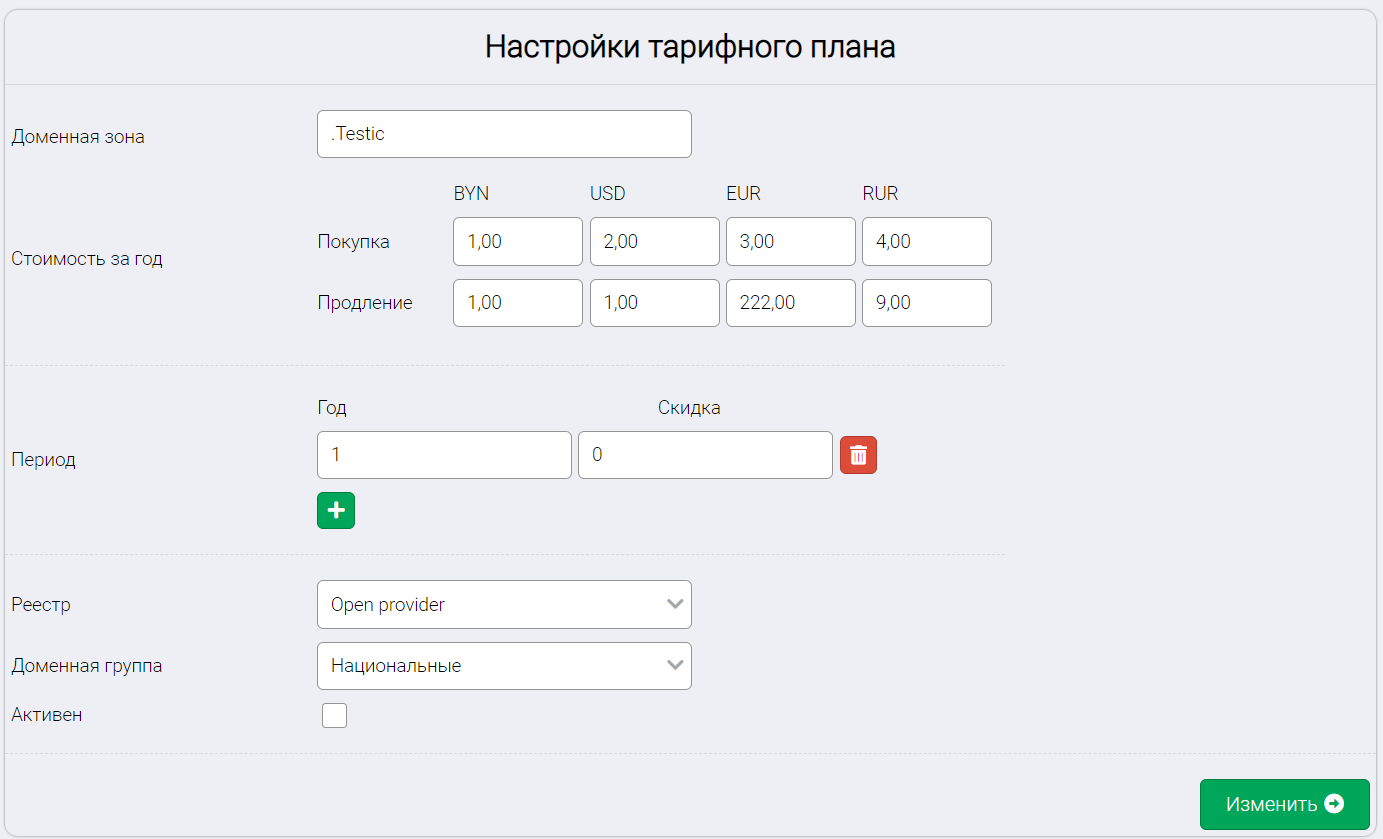


Рисунок 6.27 – Вид окна просмотра и редактирования тарифного плана, который доступен для редактирования

Чтобы сохранить настройки тарифного плана необходимо нажать на кнопку «Изменить». На рисунке 6.28 представлен вид страницы «Тарифные планы» после сохранение настроек тарифного плана.

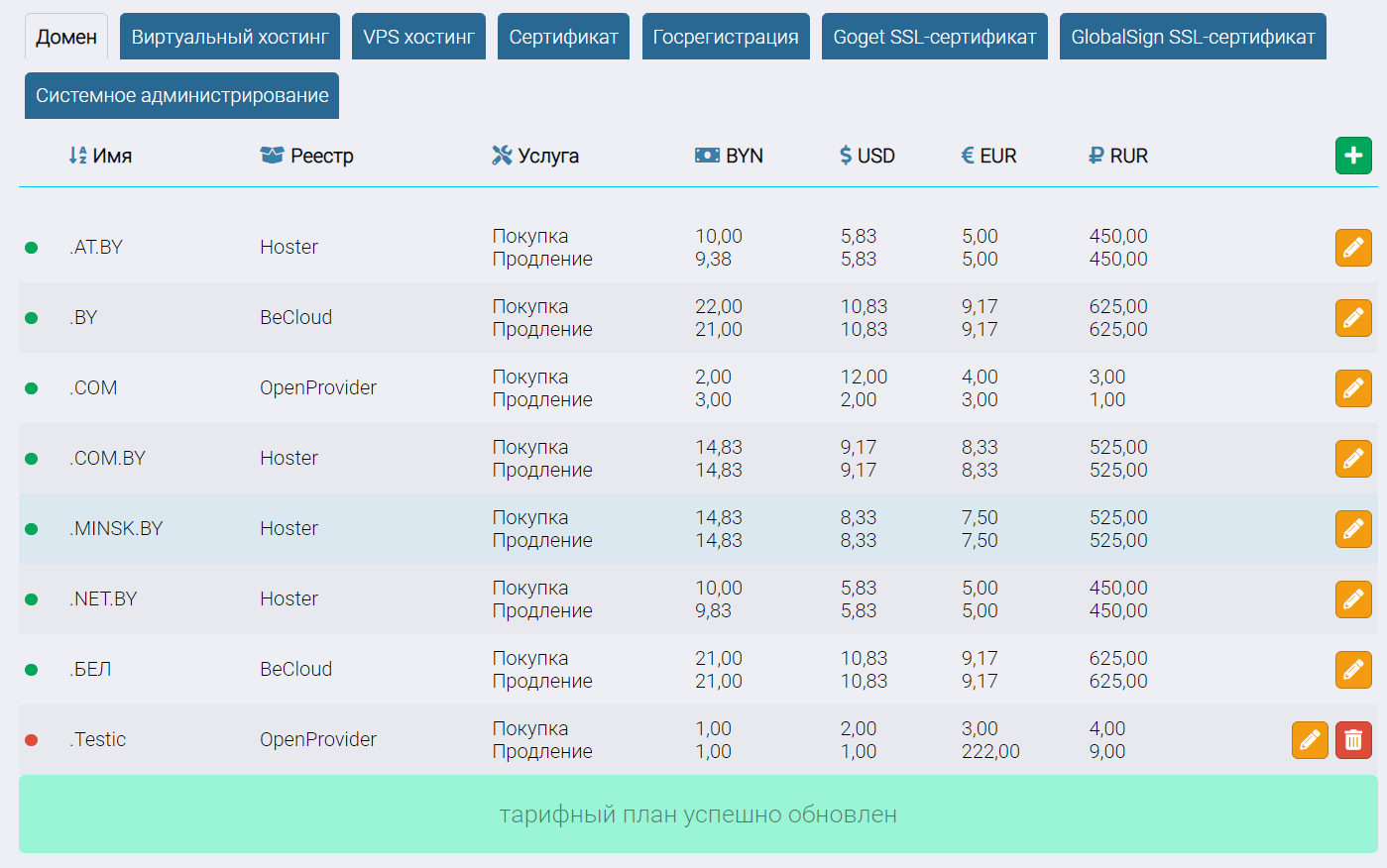


Рисунок 6.28 – Вид страницы «Тарифные планы» после сохранение настроек тарифного плана

Не все тарифные планы доступны к удалению, удалить можно только те, что ни разу не использовались.

Чтобы создать тарифный план, нужно нажать на иконку с символом «+», после чего, пользователю будет показан интерфейс, изображенный на рисунке 6.29. Важно отметить, что создание тарифного плана будет согласно типу, на странице которого была нажата кнопка для добавления тарифного плана. Чтобы сохранить тарифный план, необходимо нажать на кнопку «Изменить».

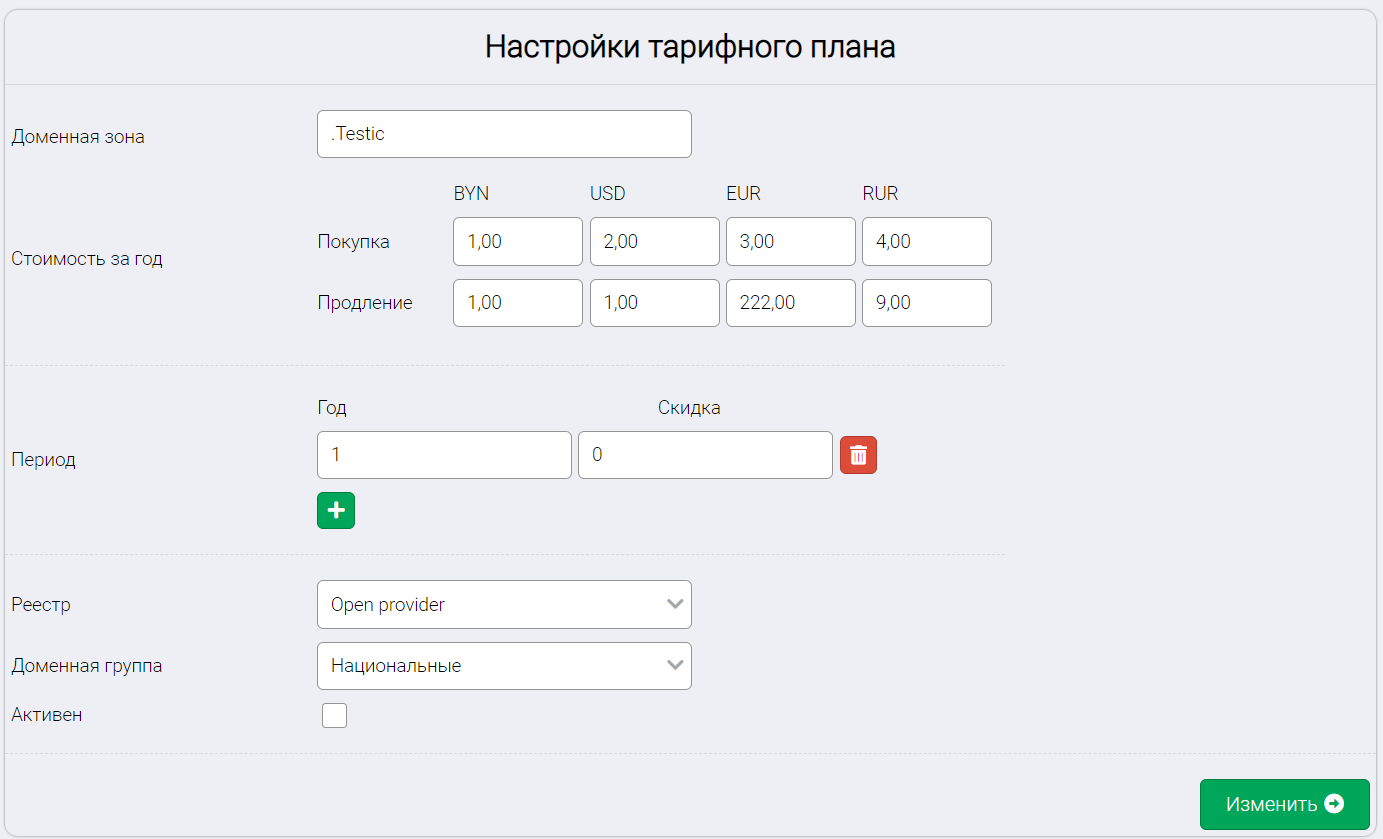


Рисунок 6.29 – Вид окна создания тарифного плана

После создания тарифного плана, вся страница не обновляется, обновляется только таблица. Вид страницы «Тарифные планы» после создания нового тарифного плана показан на рисунке 6.30.

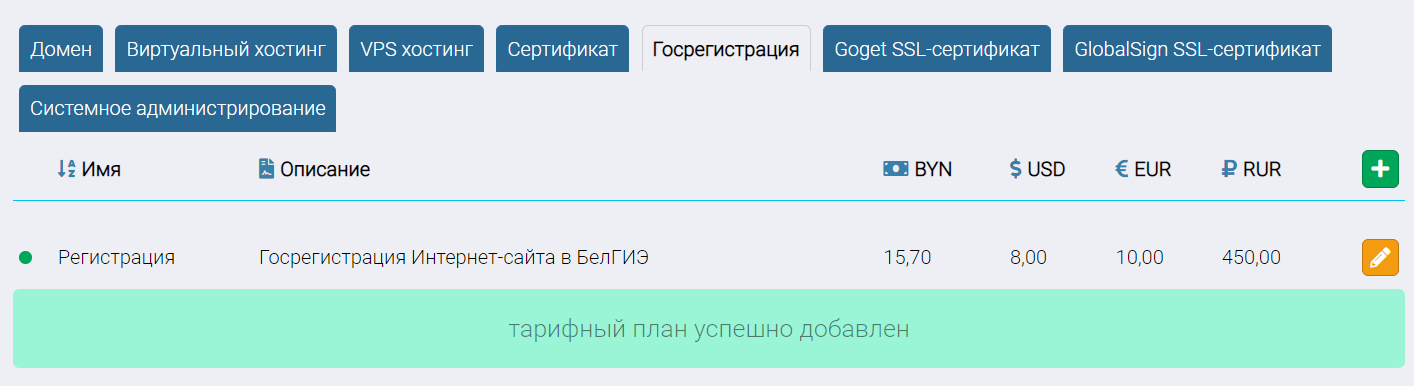


Рисунок 6.30 – Вид страницы «Тарифные планы» после создания тарифного плана

Для просмотра данных домена и реквизитов его владельца, пользователю необходимо в меню (рисунок 6.14) выбрать пункт «Домены» и в выпадающем списке выбрать пункт «Поиск информации по доменам» (рисунок 6.15). После чего интерфейс переадресует пользователя на страницу поиска нужного домена, изображенную на рисунке 6.31.

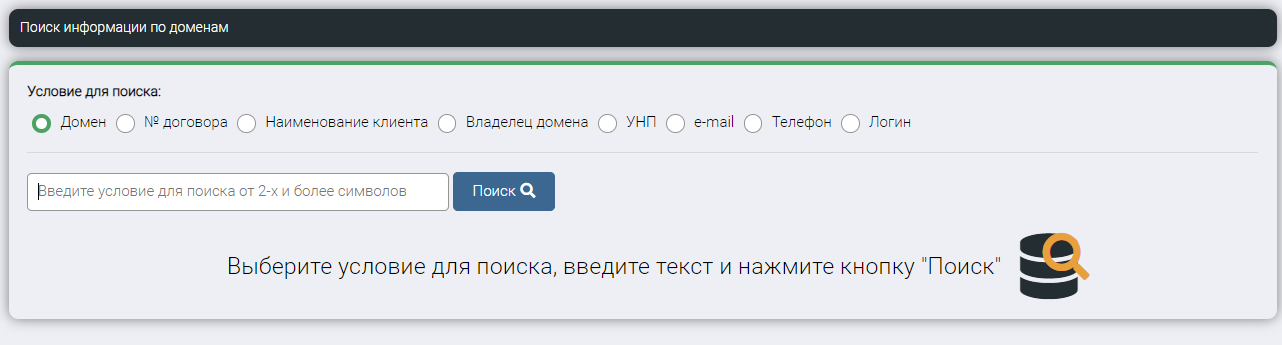


Рисунок 6.31 – Страница поиска информации по доменам

После ввода необходимых данных для поиска нужного домена и нажатия кнопки «Поиск», пользователю будет показана таблица, в которой будут все домены, которые отвечают всем критериям поиска. На рисунке 6.32 показана эта таблица, в которой только те домены, которые удовлетворяют критерии поиска.

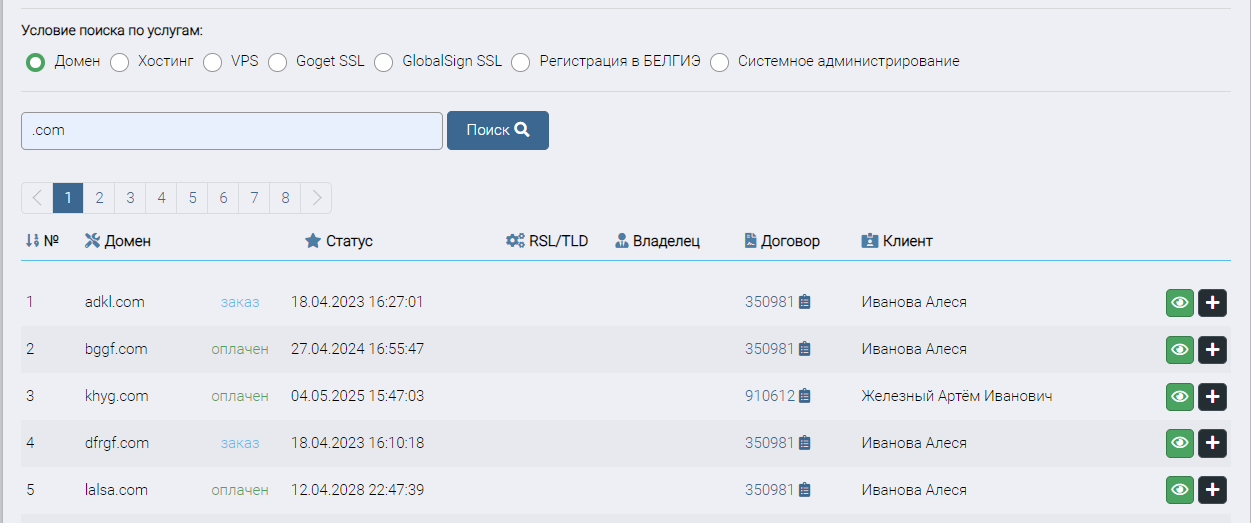


Рисунок 6.33 – Таблица доменов, удовлетворяющих критериям поиска

Чтобы создать заметку к домену, необходимо нажать на кнопку с символом «+», после нажатия откроется окно, в котором можно увидеть все заметки к домену, а также всю историю журнала звонков менеджеров к владельцу домена.



Рисунок 6.34 – Страница создания заметок к домену

Чтобы просмотреть реквизиты клиента, а также просмотреть данные домена, пользователю необходимо нажать на кнопку с изображением глаза. Интерфейс показа данных домена и его владельца показан на рисунке 6.35

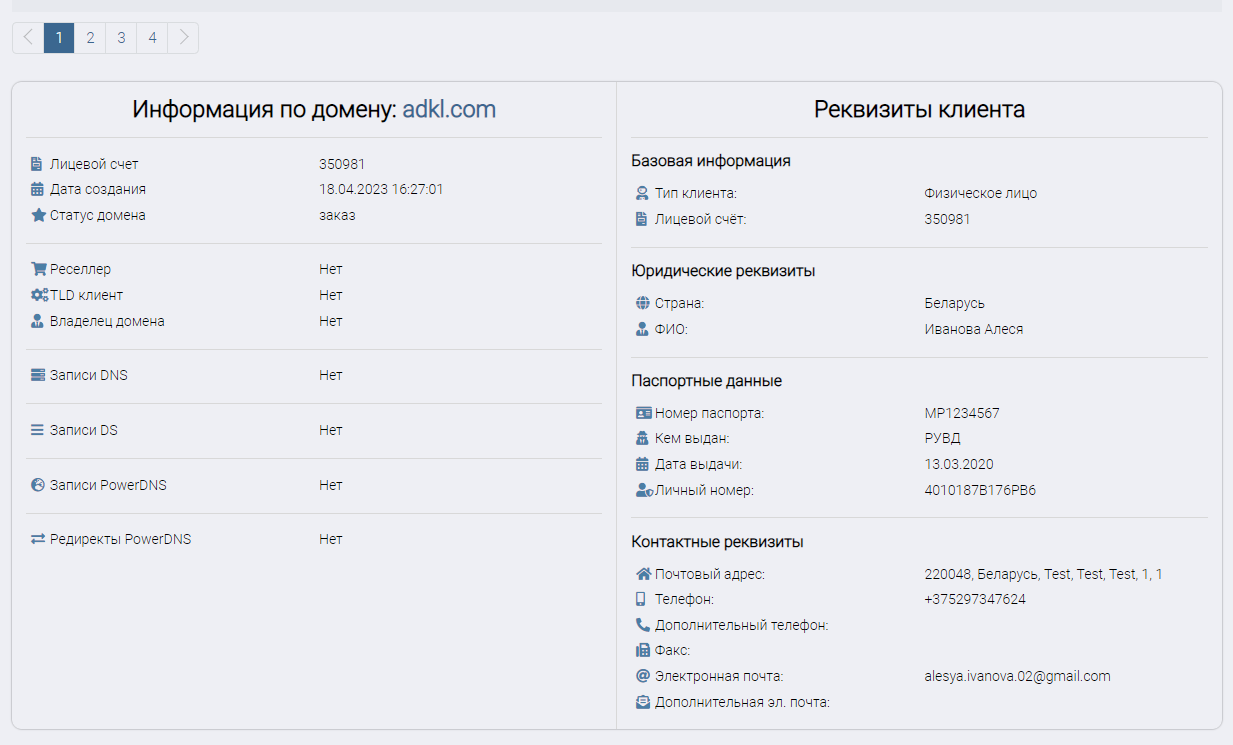


Рисунок 6.35 – Страница показа реквизитов клиента и данных домена

У всех иконок, изображенных на рисунках 6.28, 6.29, 6.33 и подобные, при наведении на них есть подсказка и информацией о том, что делает данная кнопка. Пример такой подсказки изображен на рисунке 6.36.

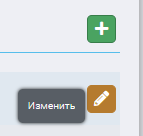


Рисунок 6.36 – Пример подсказки для кнопки, на которой присутствует только иконка

Такое решение отвечает критериям принципа построения дружественного интерфейса, а также экономит место на странице для того, чтобы была возможность отобразить больше информации. Такие кнопки размещены на всех страницах, где есть необходимость в показе большого количество информации.

6.6 Выводы по разделу

Руководство пользователя было составлено в соответствии с функционалом разработанного программного модуля управления международных доменов. В разделе представлены основные моменты работы с приложением для всех существующих в системе ролей.

7 Технико-экономическое обоснование проекта

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*2*

*БГТУ 07.00.ПЗ*

Разраб*.*

*Дежемесов М.С.*

Провер*.*

*.*

*Скребель А.С*

Консульт.

*Семёнова Л.С*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*СмеловВ.В.*

*Лит.*

*Листов*

*8*

74218155, 2023

*7 Технико-экономическое обоснование проекта*

7.1 Общая характеристика разрабатываемого программного средства

При выполнении данного дипломного проекта было разработано приложение для создания иллюстраций в дополненной реальности. Основной целью разработанного приложение является предоставление пользователям возможность создавать свои собственные пометки и иллюстрации, используя различные инструменты и функции, доступные в приложении. Пользователи смогут использовать камеру своего устройства для визуализации виртуальных объектов, наложенных на реальный мир, и добавлять к ним собственные элементы.

При выполнении данного дипломного проекта использовалась технология *Unity 2022* с использованием *MVVM* и *Unity UI Toolkit*. Также в данный дипломный проект была внедрена сторонняя библиотека для реализации dependency injection Zenject, и библиотека создания анимаций DOTween.

Разработанное программное обеспечение превосходит аналогичные программы, которые были рассмотрены в данном дипломном проекте простым и лаконичным дизайном, простотой использования программы, обеспечением пользователя возможностью использовать приложение с мобильного устройства, без использования шлема виртуальной реальности, улучшенная информативность интерфейса, возможность простого и быстрого экспортирования графики.

По результатам анализа применяемых продуктами-аналогами стратегий монетизации следует выбрать стратегию монетизации: месячная подписка на использование полной версии продукта. Месячная подписка гарантирует стабильный доход с каждого пользователя, который оформил подписку. Это может стать надежным источником дохода и способствовать более стабильной финансовой ситуации для разработчика.

7.2 Исходные данные для проведения расчетов и маркетинговый анализ

Источниками исходных данных для данных расчетов выступают действующие законы и нормативно-правовые акты. Исходные данные для расчета приведены в таблице 7.1.

Таблица 7.1 – Исходные данные для расчета

|  |  |  |
| --- | --- | --- |
| Наименование показателя | Условные обозначения | Норматив |
| Численность разработчиков, чел | Чр | 1,00 |

Окончание таблицы 7.1

|  |  |  |
| --- | --- | --- |
| Наименование показателя | Условные обозначения | Норматив |
| Норматив дополнительной заработной платы, % | Ндз | 15,00 |
| Ставка отчислений в Фонд социальной защиты населения, % | Нфсзн | 34,00 |
| Ставка отчислений в БРУСП «Белгосстрах», % | Нбгс | 0,40 |
| Норматив накладных расходов, % | Нобп,обх | 40 |
| Норматив расходов на реализацию, % | Нрр | 7 |

7.3 Обоснование цены программного средства

Широкое применение вычислительных технологий требует постоянного обновления и совершенствования ПС. Выбор эффективных проектов ПС связан с их экономической оценкой и расчетом экономического эффекта, который может определяться как для разработчика, так и для пользователя.

У разработчика экономический эффект выступает в виде чистой прибыли от реализации ПС, остающейся в распоряжении организации, а у пользователя – в виде экономии трудовых, материальных и финансовых ресурсов, получаемой за счет:

* снижения трудоемкости расчетов и алгоритмизации программирования и отладки программ;
* сокращения расходов на оплату машинного времени и других ресурсов на отладку программных средств;
* снижения расходов на материалы;
* оптимизации программных средств;
* улучшения показателей основной деятельности в результате использования передовых программных средств.

Стоимостная оценка программных средств у разработчиков предполагает определение затрат, что включает следующие статьи:

* заработная плата исполнителей – основная и дополнительная;
* отчисления в фонд социальной защиты населения;
* отчисления по обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний;
* прочие прямые затраты;
* накладные расходы;
* расходы на реализацию.

7.3.1 Расчёт затрат рабочего времени на разработку программного средства

В таблице 7.2 указаны в укрупнённом виде все работы, реально выполненные для создания, указанного в дипломной работе программного средства и количество рабочих часов, реально потраченных для выполнения этих работ. Были учтены затраты на разработку программного средства, поиск нужных компонентов платформы, куда будет интегрировано программное средство, а также тестирование и написание руководства пользователя.

Таблица 7.2 – Затраты рабочего времени на разработку ПС

|  |  |
| --- | --- |
| Содержание работ | Затраты рабочего времени, часов |
| 1. Построение диаграмм для проектирования дипломного проекта | 6 |
| 2. Поиск необходимых функций приложения | 10 |
| 3. Разработка серверной части | 20 |
| 4. Разработка клиентской части | 100 |
| 5. Модификация базы данных | 6 |
| 6. Тестирование интерфейса | 16 |
| 7. Тестирование бизнес-логики | 20 |
| 8. Написание руководства пользователя | 18 |
| Всего | 196 |

Результат расчетов затрат рабочего времени на разработку программного средства, приведенный в таблице 7.2, будет использован в дальнейших расчетах.

7.3.2 Расчёт основной заработной платы

Для определения величины основной заработной платы, было проведено исследование величин заработных плат для специалистов в программирования на Unity. В итоге было установлено, что средняя месячная заработная плата на позиции middle составляет 2 240 рублей, часовая ставка составляет 13,33 руб/час. Согласно таблице 7.2, проект разрабатывался одним специалистом на протяжении 196 часов. Таким образом, основная заработная плата будет рассчитываться по формуле 7.1.

|  |  |
| --- | --- |
| , | (7.1) |

где Соз– основная заработная плата, руб.;

Траз – время раработки (часов);

Сзп – средняя часовая ставка руб./час;

Краз – количество разработчиков, человек.

руб

В дальнейшем для других расчётов используется основная заработная плата, рассчитанная по указанной выше методике.

7.3.3 Расчёт дополнительной заработной платы

Дополнительная заработная плата на конкретное программное средство включает выплаты, предусмотренные законодательством о труде, и определяется по нормативу в процентах к основной заработной плате по формуле 7.2

|  |  |  |
| --- | --- | --- |
|  |  | (7.2) |

где Соз– основная заработная плата, руб.;

Ндз – норматив дополнительной заработной платы, %.

Cдз = 2 613,33 ⋅ 15 / 100 = 392,00 руб.

7.3.4 Отчисления в Фонд социальной защиты населения и Белгосстрах

Отчисления в Фонд социальной защиты населения и Белгосстрах (ФСЗН) определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей и вычисляются по формуле 7.3

|  |  |  |
| --- | --- | --- |
|  |  | (7.3) |

где Соз – основная заработная плата, руб.;

Сдз – дополнительная заработная плата, руб.;

Нфсзн – ставка отчислений в Фонд социальной защиты населения, %.

Отчисления в БРУСП «Белгосстрах» вычисляются по формуле 7.4

|  |  |
| --- | --- |
|  | (7.4) |

руб.

руб.

Таким образом, общие отчисления в БРУСП «Белгосстрах» составили 12,02 руб., а в фонд социальной защиты населения – 1021,81 руб.

7.3.5 Расчёт суммы прочих прямых затрат

Сумма прочих прямых затрат вычисляется на основе совокупности всех конкретных затрат, необходимых для разработки веб-приложения. Были учтены затраты на использование программной среды разработки *Jetbrains Rider* учитывая использование подписки 2 месяца во время разработки веб-приложения, также затраты на сервис *Firebase.* В таблице 7.3 представлено описание прочих прямых затрат на разработку веб-приложения.

Таблица 7.3 – Прочие прямые затраты на разработку приложения

|  |  |
| --- | --- |
| Категория затрат | Значение за |
| Среда разработки Jetbrains Rider | 227 |
| Сервис Firebase | 387,09 |
| Всего | 614,09 |

7.3.6 Расчёт суммы накладных расходов

Сумма накладных расходов Cобп, обх – произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив накладных расходов в целом по организации Нобп, обх согласно формуле 7.5

|  |  |
| --- | --- |
|  | (7.5) |

Все данные необходимые для вычисления есть, поэтому можно определить сумму накладных расходов:

Собп, обх = 2 613,33 ⋅ 40 / 100 = 1 045,33 руб.

7.3.7 Сумма расходов на разработку программного средства

Сумма расходов на разработку веб-приложения Ср определяется как сумма основной и дополнительной заработных плат исполнителей на конкретное программное средство, отчислений на социальные нужды, расходов на материалы, расходов на оплату машинного времени, суммы прочих затрат и суммы накладных расходов согласно формуле 7.6.

|  |  |  |
| --- | --- | --- |
|  | Ср = Соз + Сдз + Сфсзн + Сбгс + Спз + Собп, обх, | (7.6) |

Ср = 2 613,33 + 392,00 +1 021,81 + 12,02 + 614,09 + 1 045,33 = 5 598,59 руб.

Сумма расходов на разработку веб-приложения была вычислена на основе данных, рассчитанных ранее в данном разделе. Таким образом, сумма расходов на разработку веб-приложения составила 5 598,59 рублей.

7.3.8 Расходы на реализацию

Сумма расходов на реализацию программного средства Срр определяется как произведение суммы расходов на разработку на норматив расходов на реализацию Нрр, и находится по формуле 7.7. Для привлечения клиентов для покупки подписки на полную версию приложения, используется рассылка по почте уже существующим клиентам, у которых подписка не оформлена.

|  |  |  |
| --- | --- | --- |
|  |  | (7.7) |

= 5 598,59 ⋅ 7 / 100 = 398,90 руб*.*

Все проведенные выше расчеты необходимы для вычисления полной себестоимости проекта.

7.3.9 Расчет полной себестоимости

Полная себестоимость Сп определяется как сумма двух элементов: суммы расходов на разработку Ср и суммы расходов на сопровождение и адаптацию веб-приложения Срса согласно формуле 7.8

|  |  |  |
| --- | --- | --- |
|  |  | (7.8) |

где Сп – полная себестоимость веб-приложения, руб.;

Ср – сумма расходов на разработку веб-приложения, руб.;

Срса – сумма расходов на сопровождение и адаптацию веб-приложения, руб.

Сп = 5 698,50 + 398,90 = 6 097,40 руб.

Полная себестоимость составила 6 097,40 рублей.

7.3.10 Определение цены, оценка эффективности

Так как монетизация продукта осуществляется путем предоставления подписки на полную версию приложения, необходимо определить сумму денежных поступлений и окупаемость затрат на разработку веб-приложения, т.е. целесообразность и эффективность.

Для этого необходимо рассчитать количество подписок/установок/продаж на основании следующих данных о продуктах-аналогах:

1. *Tilt brush* – приложение для виртуальной реальности, разработанное Google, которое позволяет пользователям создавать 3D-иллюстрации в дополнен-ной реальности. Приложение доступно на платформах Oculus Rift, HTC Vive, Valve Index, а также на Oculus Quest в режиме Link.
2. *Adobe medium* – приложение для создания виртуальных объектов и сред. Оно было разработано для использования с устройствами виртуальной реальности, такими как Oculus Rift, Oculus Quest и HTC Vive. Приложение позволяет создавать 3D-модели, которые могут быть использованы в приложениях виртуальной реальности, играх, фильмах и других проектах.
3. *eDrawings Viewer* является полезным приложением для просмотра и обмена 2D и 3D моделями. Оно позволяет пользователям открывать и просматривать файлы в различных форматах, включая DWG, DXF, SolidWorks, CATIA, Pro/ENGINEER и многие другие

По результатам анализа применяемых продуктами-аналогами стратегий монетизации следует выбрать стратегию монетизации: предоставление месячной подписки на полную версию приложения. Были выбраны следующие характеристики для показателей качества рассматриваемого программного продукта и программного продукта конкурента:

1. Дизайн – то, как приложение выглядит, очень важно для любого современного приложения.
2. Юзабилити – насколько приложение удобно в использовании
3. Функциональность – количество инноваций, внедряемых в приложение
4. Отсутствие багов – наличие несущественных ошибок в работе.

Расчет показателей качества базового и нового продуктов, согласно балловому методу, приводится в таблице 7.4.

Таблица 7.4 – Оценка качества программного средства

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Показатель качества | Весовой коэффициент | Разрабатываемый продукт | Продукт-аналог 1 | Продукт-аналог 2 | Продукт-аналог 3 |
| Дизайн | 0,3 | 8 | 7 | 6 | 8 |
| Юзабилити | 0,4 | 9 | 9 | 6 | 5 |

Продолжение таблицы 7.4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Функциональность | 0,2 | 6 | 4 | 6 | 7 |
| Отсутствие багов | 0,1 | 9 | 7 | 7 | 7 |
| Всего | 1 | 8,1 | 7,2 | 6,1 | 6,5 |

Исходя из таблицы 7.4 можно сделать вывод что наш продукт выгодно отличается от продуктов наших конкурентов и имеет выгодный баланс возможностей, дизайна и качества кода.

Цена подписки Ц нового продукта рассчитывается исходя из среднего значения цены по каждому аналогу. Цена Ц установки аналога, скорректированная на оценку показателей качества, рассчитывается по формуле 7.9:

|  |  |  |
| --- | --- | --- |
|  |  | (7.9) |
|  |

где Ц0 − цена программного продукта конкурента,

ИР – показатель рассматриваемого программного продукта,

ИК – показатель программного продукта конкурента.

Ц1 = (58,00 8,1) / 7,2 = 65,25 рубля

Ц2 = (44,99 8,1) / 6,1 = 59,74 рубля

Ц3 = (44,00 8,1) / 6,5 = 54,83 рубля

Ц = (65,25 + 59,74 +54,83) / 3 = 59,94 рубля

По данным компании среднее количество проданных копий полной версии приложения 5 500 за год (по данным с сентября 2020 года по декабрь 2022 года), то денежные поступления от продажи услуг год составят 329 672,91 рублей.

Количество покупателей продукта необходимых для окупаемости Пп вычисляется по формуле 7.10:

|  |  |  |
| --- | --- | --- |
|  |  | (7.10) |

*=* 6 097,40 / 59,94 = 101,72покупателей.

7.4 Вывод по разделу

В приложении З представлены результаты расчётов для основных показателей данной главы в краткой форме. Разработка программного средства, осуществляемая одним программистом в течении 196 часов (1 месяц), при заданных условиях обойдется в 6 097,40 руб. Реализации данного программного средства будет приносить годовые денежные поступления от продажи подписки в размере 329 672,91 рублей и окупится при 102 покупателях. Согласно статистике компании, среднее количество покупателей полной версии приложения в месяц составляет 482 клиента, что означает, что срок окупаемости программного средства составляет менее года, а значит, можно сделать вывод что приложение окупится в течение года.

Необходимость разработки приложения, обусловлена его уникальностью на рынке и востребованностью такого функционала на данный момент.

Заключение

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*2*

*БГТУ 00.00.ПЗ*

Разраб*.*

*Иванова А.А.*

Провер*.*

*.*

*Блинова Е.А*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*СмеловВ.В.*

*Лит.*

*Листов*

*1*

11111111, 2023

*Заключение*

В рамках работы над проектом был проведен обзор аналогичных решений, выбраны язык и платформа для разработки сервера, технологии для разработки клиентской части приложения.

Для достижения поставленных задач были спроектированы:

* логическая схема базы данных;
* диаграмма вариантов использования;
* структурная схема программного модуля;
* блок-схема алгоритма заказа домена.

Было дано описание функциональных тестов, которые были проведены на этапе тестирования веб-приложения. Для повышения гарантии качества кода приложения было написано 19 *unit*-тестов.

В пояснительной записке приведено руководство пользователя по использованию программного модуля, которое описывает основные сценарии работы, что позволяет пользователям легко ориентироваться в приложении и работать с ним.

Рассчитаны затраты на разработку представленного в дипломном проекте программного средства. На основании полученных данных можно сделать вывод, что разработанный проект является экономически выгодным.

Конечный продукт предоставляет возможность управления доменами любой доменной зоны, а также управления тарифных планов платформы.

Программное средство является законченным программным продуктом, который реализует все перечисленные возможности в полной форме. Веб-приложение поддерживает архитектуру, которая позволяет в дальнейшем быстро и безопасно расширить функционал.

Программное средство соответствует целям дипломного проекта, реализует все поставленные перед ним задачи.

Результаты дипломного проекта доложены на 73-ей научно-технической конференции учащихся, студентов и магистрантов БГТУ с публикацией материалов конференции в сборнике работ.

Список использованных источников

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*2*

*БГТУ 00.00.ПЗ*

Разраб*.*

*Иванова А.А.*

Провер*.*

*.*

*Блинова Е.А*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*СмеловВ.В.*

*Лит.*

*Листов*

*2*

11111111, 2023

*Список использованных источников*

1. Hoster.by [Электронный ресурс] // hoster.by. – Режим доступа: <https://hoster.by/>.– Дата доступа: 11.04.2023.
2. HostFly [Электронный ресурс] // hostfly.by. – Режим доступа: <https://www.hostfly.by/>. – Дата доступа: 11.04.2023.
3. ActiveCloud [Электронный ресурс] // activecloud.by. – Режим доступа: <https://www.activecloud.by/>. – Дата доступа: 12.04.2023.
4. DomainBy [Электронный ресурс] // domain.by. – Режим доступа: <https://domain.by/>. – Дата доступа: 12.04.2023.
5. Либерти Д. Язык программирования C# // Программирование на C#. – Санкт-Петербург. – 2003: Символ-Плюс. – С. 26. – 688 с. – ISBN 5-93286-038-3.
6. Введение в ASP.NET Core MVC – Краткое описание [Электронный ресурс] / metanit.com. – Режим доступа: <https://metanit.com/sharp/aspnet5/3.1.php/>. – Дата доступа: 08.04.2023.
7. Обзор .NET Core [Электронный ресурс] // learn.microsoft.com– Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/core/introduction>. – Дата доступа: 15.04.2023.
8. Декостер К. Pro NuGet (Expert's Voice in Microsoft) – Нью-Йорк. – 2012: Apress – С. 40. – 256 с. – ISBN 1-43024-191-8.
9. Microsoft Visual Stidio [Электронный ресурс] // microsoft.com. – Режим доступа: <https://visualstudio.microsoft.com/ru/>. – Дата доступа 15.04.2023.
10. Краузе Дж. Mastering Windows Server 2019 – Бирмингем, Великобритания. – 2019: Packt Publishing С. 16. – 406 с. – ISBN 978-1-78980-453-9.
11. RFC 2616 Hypertext Transfer Protocol – HTTP 1/1 [Электронный ресурс] // tools.ietf.org. – Режим доступа: <https://tools.ietf.org/html/rfc2616/>. – Дата доступа: 16.04.2023.
12. SQL Server technical documentation [Электронный ресурс] // learn.microsoft.com. – Режим доступа: <https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver16>. – Дата доступа: 16.04.2023.
13. Введение в Entity Framework Core [Электронный ресурс] // metanit.com. – Режим доступа: <https://metanit.com/sharp/entityframeworkcore/1.1.php/>. – Дата доступа: 17.04.2023.
14. LINQ to SQL [Электронный ресурс] // metanit.com. – Режим доступа: <https://metanit.com/sharp/adonet/4.1.php> – Дата доступа: 17.04.2023.
15. FluentValidation Documentation [Электронный ресурс] // docs.fluentvalidation.net – Режим доступа: <https://docs.fluentvalidation.net/en/latest/> – Дата доступа: 17.04.2023.
16. Учебник. Начало работы с Razor Pages в Asp.Net Core [Электронный ресурс] // learn.microsoft.com– Режим доступа: <https://learn.microsoft.com/ru-ru/aspnet/core/tutorials/razor-pages/razor-pages-start?view=aspnetcore-7.0&tabs=visual-studio>. – Дата доступа 17.04.2023.
17. Bootstrap (front-end framework) [Электронный ресурс] / getbootstrap.com – Режим доступа: <https://getbootstrap.com/docs/5.0/layout/grid/>. – Дата доступа: 09.04.2023.
18. Гранд К. CSS in Depth – Нью-Йорк. – 2018: Manning – С. 37. – 364 с. – ISBN 9-78161729-345-0.
19. Openprovider documentation [Электронный ресурс] // docs.openprovider.com – Режим доступа: <https://docs.openprovider.com/doc/all> – Дата доступа: 17.04.2023.
20. Академик [Электронный ресурс] / dic.academic.ru. – Режим доступа: https://dic.academic.ru/dic.nsf/ruwiki/146913/. – Дата доступа: 11.04.2023.
21. Тестирование. Фундаментальная теория [Электронный ресурс] // habr.com. – Режим дома: https://habr.com/ru/post/279535/. – Дата доступа 01.05.2023.
22. About xUnit.net [Электронный ресурс] // xunit.net. – Режим доступа: https://xunit.net/. – Дата доступа: 02.05.2023.

ПРИЛОЖЕНИЕ А

Структурная схема приложения

ПРИЛОЖЕНИЕ Б

Архитектурная схема приложения

ПРИЛОЖЕНИЕ В

Логическая схема базы данных

ПРИЛОЖЕНИЕ Г

Блок-схема алгоритма заказа домена

ПРИЛОЖЕНИЕ Д

Листинг класса *OpenProviderService*

using Microsoft.Extensions.DependencyInjection;

using OpenContact.DMS.Common.Infrastructure.Diagnostics;

using OpenContact.DMS.Configuration;

using OpenContact.DMS.OpenProviderAPI.Data.Entities;

using OpenContact.DMS.OpenProviderAPI.Data.Entities.Request;

using OpenContact.DMS.OpenProviderAPI.Data.Entities.Response;

using OpenContact.DMS.OpenProviderAPI.Data.Helper;

using OpenContact.DMS.OpenProviderAPI.Data.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net.Http;

using System.Net.Http.Headers;

using System.Net.Http.Json;

using System.Text.Json;

using System.Text.Json.Serialization;

using System.Threading;

using System.Threading.Tasks;

namespace OpenContact.DMS.OpenProviderAPI

{

public class CustomDateTimeConverter : JsonConverter<DateTime>

{

private readonly string Format;

public CustomDateTimeConverter(string format)

{

Format = format;

}

public override void Write(Utf8JsonWriter writer, DateTime date, JsonSerializerOptions options)

{

writer.WriteStringValue(date.ToString(Format));

}

public override DateTime Read(ref Utf8JsonReader reader, Type typeToConvert, JsonSerializerOptions options)

{

return DateTime.ParseExact(reader.GetString(), Format, null);

}

}

public class OpenProviderService

{

private HttpClient \_httpClient;

private string \_token;

private DateTime \_tokenDate;

private OpenProviderConfig \_config;

private Urls \_urls;

private JsonSerializerOptions \_options;

private CancellationToken \_cancellationToken;

public OpenProviderService(OpenProviderConfig config, CancellationToken cancellationToken)

{

if (new[] { config.Login, config.Password, config.Ip, config.BaseUrl }.Any(x => String.IsNullOrWhiteSpace(x)))

{

throw new ArgumentNullException(nameof(\_config.Login));

}

\_config = config;

\_urls = new Urls(\_config.BaseUrl);

\_options = new JsonSerializerOptions

{

PropertyNameCaseInsensitive = true,

};

\_options.Converters.Add(new JsonStringEnumConverter());

\_options.Converters.Add(new CustomDateTimeConverter("yyyy-MM-dd HH:mm:ss"));

var serviceProvider = new ServiceCollection().AddHttpClient().BuildServiceProvider();

var \_clientFactory = serviceProvider.GetService<IHttpClientFactory>();

\_httpClient = \_clientFactory.CreateClient("openProviderHttpClient");

\_cancellationToken = cancellationToken; }

private async Task<string> GetTokenAsync()

{

if (String.IsNullOrWhiteSpace(\_token) || \_tokenDate.AddDays(2) <= DateTime.Now)

{

var auth = new AuthRequest()

{

Ip = \_config.Ip, //"0.0.0.0"

Password = \_config.Password, // "EDbT79sDi2YzE!8"

Username = \_config.Login, //"info@domain.by"

};

var response = await \_httpClient.PostAsJsonAsync(\_urls.Auth, auth);

var data = await response.Content.ReadFromJsonAsync<AuthResponse>();

\_httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", data.Data.Token);

\_tokenDate = DateTime.Now;

\_token = data.Data.Token;

}

}

#region Domains

public async Task<DomainsCheckResponse> DomainsCheckAsync(string domainName, IEnumerable<string> domainExtentions = null, string domainExtention = null)

{

var domainZones = new List<DomainsCheckRequest.DomainsRequest>();

if (domainExtentions != null)

{

foreach (var extention in domainExtentions)

{

domainZones.Add(new DomainsCheckRequest.DomainsRequest(domainName, extention.StartsWith(".") ? extention.Substring(1) : extention));

}

}

else

{

domainZones.Add(new DomainsCheckRequest.DomainsRequest(domainName, domainExtention.StartsWith(".") ? domainExtention.Substring(1) : domainExtention));

}

var domainsCheck = new DomainsCheckRequest()

{

Domains = domainZones,

};

try

{

await GetTokenAsync();

var response = await \_httpClient.PostAsJsonAsync(\_urls.DmainsCheck, domainsCheck, \_cancellationToken);

return await TryToGetDataAsync<DomainsCheckResponse>(response);

}

catch (Exception e)

{

LogWriterFactory.EventLog.Error(e);

return null;

}

}

public async Task<SuggestNameResponse> SuggestNameAsync(string name, List<string> domainZones)

{

var suggestName = new SuggestNameRequest()

{

Name = name,

Tlds = domainZones,

};

try

{

await GetTokenAsync();

var response = await \_httpClient.PostAsJsonAsync(\_urls.SuggestName, suggestName, \_cancellationToken);

return await TryToGetDataAsync<SuggestNameResponse>(response);

}

catch (Exception e)

{

LogWriterFactory.EventLog.Error(e);

return null;

}

}

public async Task<DomainAdditionalDataResponse> GetDomainAdditionalDataAsync(string zone)

{

try

{

await GetTokenAsync();

var response = await \_httpClient.GetAsync(\_urls.DomainAdditionalData(zone), \_cancellationToken);

return await TryToGetDataAsync<DomainAdditionalDataResponse>(response);

}

catch (Exception e)

{

LogWriterFactory.EventLog.Error(e);

return null;

}

}

public async Task<RegisterDomainResponse> CreateDomainAsync(string handle, string domainName, string domainExtention, int periodInYears, List<string> dnsCollection = null)

{

var defaultDnsServers = AppSettings.Settings.DMS.Domais.DefaultNsServers.Select(x => new RegisterDomainRequest.NameServersRequest()

{

Name = x,

});

var usersDnsServers = dnsCollection?.Select(x => new RegisterDomainRequest.NameServersRequest()

{

Name = x,

});

var dnsServers = (dnsCollection?.Any() ?? false) ? usersDnsServers : defaultDnsServers;

var domain = new RegisterDomainRequest(){

AdminHandle = handle,

OwnerHandle = handle,

TechHandle = handle,

Domain = new DomainModel()

{

Name = domainName,

Extension = domainExtention,

},

Period = periodInYears.ToString(),

NameServers = dnsServers.ToList(),

};

try

{

await GetTokenAsync();

var response = await \_httpClient.PostAsJsonAsync(\_urls.CreateDomain, domain, \_cancellationToken);

return await TryToGetDataAsync<RegisterDomainResponse>(response);

}

catch (Exception e)

{

LogWriterFactory.EventLog.Error(e);

return null;

}

}

public async Task<UpdateDomainResponse> UpdateDomainDnsAsync(UpdateDomainDnsRequest dnsCollection, long openProviderDomainId)

{

try

{

await GetTokenAsync();

if (dnsCollection is null || dnsCollection.NameServers is null || !dnsCollection.NameServers.Any())

{

dnsCollection.NameServers = AppSettings.Settings.DMS.Domais.DefaultNsServers.Select(x => new UpdateDomainDnsRequest.NameServersRequest()

{

Name = x,

}).ToList();

}

var response = await \_httpClient.PutAsJsonAsync(\_urls.UpdateDomain(openProviderDomainId), dnsCollection, \_cancellationToken);

return await TryToGetDataAsync<UpdateDomainResponse>(response);

}

catch (Exception e)

{

LogWriterFactory.EventLog.Error(e);

return null;

}

}

public async Task<UpdateDomainResponse> UpdateDomainOwnerAsync(string newOwnerHandle, long openProviderDomainId)

{

try

{

await GetTokenAsync();

var updateDomainModel = new UpdateDomainOwnerRequest()

{

AdminHandle = newOwnerHandle,

OwnerHandle = newOwnerHandle,

TechHandle = newOwnerHandle,

};

var response = await \_httpClient.PutAsJsonAsync(\_urls.UpdateDomain(openProviderDomainId), updateDomainModel, \_cancellationToken);

return await TryToGetDataAsync<UpdateDomainResponse>(response);

}

catch (Exception e)

{

LogWriterFactory.EventLog.Error(e);

return null;

}

}

public async Task<GetDomainResponse> GetDomainAsync(long id)

{

try

{

await GetTokenAsync();

var response = await \_httpClient.GetAsync(\_urls.GetDomain(id), \_cancellationToken);

return await TryToGetDataAsync<GetDomainResponse>(response);

}

catch (Exception e)

{

LogWriterFactory.EventLog.Error(e);

return null;

}

}

public async Task<GetDomainAuthCodeResponse> GetDomainAuthCodeAsync(long id)

{

try

{

await GetTokenAsync();

var response = await \_httpClient.GetAsync(\_urls.GetDomainAuthCode(id), \_cancellationToken);

return await TryToGetDataAsync<GetDomainAuthCodeResponse>(response);

}

catch (Exception e)

{

LogWriterFactory.EventLog.Error(e);

return null;

}

}

public async Task<RenewOrRestoreDomainResponse> RenewDomainAsync(long domainId, int perionInYears)

{

var domain = new RenewDomainRequest()

{

Period = perionInYears.ToString(),

};

try

{

await GetTokenAsync();

var response = await \_httpClient.PostAsJsonAsync(\_urls.RenewDomain(domainId), domain, \_cancellationToken);

return await TryToGetDataAsync<RenewOrRestoreDomainResponse>(response);

}

catch (Exception e)

{

LogWriterFactory.EventLog.Error(e);

return null;

}

}

//работает только на статусах карантина

public async Task<RenewOrRestoreDomainResponse> RestoreDomainAsync(long id)

{

var domain = new RestoreDomainRequest()

{

Domain = new DomainModel

{

Extension = "com",

Name = "exampledomainhosting11",},

Id = id,

};

try

{

await GetTokenAsync();

var response = await \_httpClient.PostAsJsonAsync(\_urls.RestoreDomain(id), domain, \_cancellationToken);

return await TryToGetDataAsync<RenewOrRestoreDomainResponse>(response);

}

catch (Exception e)

{

LogWriterFactory.EventLog.Error(e);

return null;

}

}

public async Task<RenewOrRestoreDomainResponse> DeleteDomainAsync(long id)

{

try

{

await GetTokenAsync();

var response = await \_httpClient.DeleteAsync(\_urls.DeleteDomain(id, "exampledomainhosting11", "com"), \_cancellationToken);

return await TryToGetDataAsync<RenewOrRestoreDomainResponse>(response);

}

catch (Exception e)

{

LogWriterFactory.EventLog.Error(e);

return null;

}

}

#endregion Domains

#region Customers

public async Task<GetCustomerResponse> GetCustomerAsync(string handle)

{

try

{

await GetTokenAsync();

var response = await \_httpClient.GetAsync(\_urls.Customer(handle), \_cancellationToken);

return await TryToGetDataAsync<GetCustomerResponse>(response);

}

catch (Exception e)

{

LogWriterFactory.EventLog.Error(e);

return null;

}

}

public async Task<GetAllCustomersResponse> GetAllCustomersAsync()

{

try

{

await GetTokenAsync();

var response = await \_httpClient.GetAsync(\_urls.CustomerStaticLink, \_cancellationToken);

return await TryToGetDataAsync<GetAllCustomersResponse>(response);

}

catch (Exception e)

{

LogWriterFactory.EventLog.Error(e);

return null;

}

}

public async Task<CreateCustomerResponse> CreateCustomerAsync(CreateCustomerRequest customer)

{

try

{

await GetTokenAsync();

var response = await \_httpClient.PostAsJsonAsync(\_urls.CustomerStaticLink, customer, \_options, \_cancellationToken);

return await TryToGetDataAsync<CreateCustomerResponse>(response);

}

catch (Exception e)

{

LogWriterFactory.EventLog.Error(e);

return null;

}

}

public async Task<IsSuccessResponse> UpdateCustomerAsync(UpdateCustomerRequest customer, string customerHandle)

{

try

{

await GetTokenAsync();

var response = await \_httpClient.PutAsJsonAsync(\_urls.Customer(customerHandle), customer, \_options, \_cancellationToken);

return await TryToGetDataAsync<IsSuccessResponse>(response);}

catch (Exception e)

{

LogWriterFactory.EventLog.Error(e);

return null;

}

}

public async Task<IsSuccessResponse> DeleteCustomerAsync(string handle)

{

try

{

await GetTokenAsync();

var response = await \_httpClient.DeleteAsync(\_urls.Customer(handle), \_cancellationToken);

return await TryToGetDataAsync<IsSuccessResponse>(response);

}

catch (Exception e)

{

LogWriterFactory.EventLog.Error(e);

return null;

}

}

public async Task<CustomerAdditionalDataResponse> GetCustomersAdditionalDataAsync(string zone)

{

try

{

await GetTokenAsync();

var response = await \_httpClient.GetAsync(\_urls.CustomerAdditionalData(zone), \_cancellationToken);

return await TryToGetDataAsync<CustomerAdditionalDataResponse>(response);

}

catch (Exception e)

{

LogWriterFactory.EventLog.Error(e);

return null;

}

}

#endregion Customers

#region Helpers

public async Task<T> TryToGetDataAsync<T>(HttpResponseMessage response) where T : OpenProviderResponse

{

var content = await response.Content.ReadFromJsonAsync<T>(\_options);

if (response is null)

{

throw new ArgumentNullException(nameof(response));

}

if (!response.IsSuccessStatusCode && content is null)

{ throw new Exception("OpenProviderService exception: " + response.StatusCode);

}

//20001 - Verification email in progress

if (content.Code != 0 && content.Code != 1 && content.Code != 20001)

{

throw new Exception("OpenProvider exception. code: " + content.Code + " message: " + content.Desc);

}

return content;

}#endregion

}

}

ПРИЛОЖЕНИЕ Е

Листинг метода *CheckDomain* класса *OrderDomainController*

[HttpGet]

public async Task<ActionResult> CheckDomain(string domainName, Dictionary<string, bool> domainZones = null) {

if (domainZones != null && domainZones.ContainsKey("controller")) return null;

var selectedDomainZoneIds = domainZones.Where(x => x.Value == true).Select(x => x.Key).ToList().ConvertAll(int.Parse);

return await GetDomainTable(domainName, CurrencyCode.BYN, selectedDomainZoneIds.ToArray());

}

private async Task<ActionResult> GetDomainTable(string selectedDomainName, CurrencyCode selectedCurrency, int[] selectedDomainZoneIds){

string cyrilicName = "";

if (Regex.IsMatch(selectedDomainName, @"^[а-яА-Я0-9ʼ]\*$"))

{

cyrilicName = selectedDomainName;

selectedDomainName = TranslitHelper.GetTranslitRuToEng(selectedDomainName);

}

if (!Regex.IsMatch(selectedDomainName, @"^[a-zA-Z0-9]+$")){

return PartialView("~/Views/OrderDomain/\_DomainValidationError.cshtml", OrderDomainResource.LatinDomainValidationFailed);}

if (cyrilicName.StartsWith("ʼ") || cyrilicName.EndsWith("ʼ")){

return PartialView("~/Views/OrderDomain/\_DomainValidationError.cshtml", OrderDomainResource.CyrilicDomainValidationFailed); }

var listOfSelectedDomainZoneIds = selectedDomainZoneIds.ToList();

var domainTariffPlans = \_tariffPlanService.GetDomainTariffPlans(isActiveOnly: true)

.Where(x => listOfSelectedDomainZoneIds.Contains(x.DomainZone.DomainZoneId)).ToList();

var domainsStatusesInBeCloudAndHoster = getDomainStatusesForBeCloudAndHoster(selectedDomainName, \_vat, domainTariffPlans, cyrilicName, selectedCurrency).ToList();

var domainStatusesInOpenProvider = await getDomainStatusesForOpenProvider(selectedDomainName, \_vat, domainTariffPlans, listOfSelectedDomainZoneIds, selectedCurrency);

domainsStatusesInBeCloudAndHoster.AddRange(domainStatusesInOpenProvider);

var notAvaivableDomainZones = getNotAvaivableDomainZones(domainsStatusesInBeCloudAndHoster);

var suggestedDomains = await getSuggestedNamesFromOpenProvider(selectedDomainName, \_vat, domainTariffPlans, notAvaivableDomainZones, selectedCurrency);

domainsStatusesInBeCloudAndHoster.AddRange(suggestedDomains);

MasterModel.BindBalance();

var checkDomainsView = new CheckDomainsView(){

FreeDomainsView = domainsStatusesInBeCloudAndHoster,

UserCurrencies = MasterModel.Currencies.Currencies.ToDictionary(x => Enum.TryParse<CurrencyCode>(x.Value, out var value) ? value : default, x => x.Text),

IsOnlyBynCurrency = CurrencyHelper.IsOnlyByrCurrency(MasterModel.Client.ClientType),

SelectedCurrency = selectedCurrency,

SelectedDomainName = selectedDomainName,

SelectedPeriodInYears = default,

SelectedDomainZoneIds = listOfSelectedDomainZoneIds.ToArray(), };

checkDomainsView.FreeDomainsView.Where(x => x.IsFree == true).FirstOrDefault().IsSelected = true;

return PartialView("~/Views/OrderDomain/\_DomainsTable.cshtml", checkDomainsView); }

ПРИЛОЖЕНИЕ Ж

Таблица экономических показателей

|  |  |
| --- | --- |
| Наименование показателя | Значение |
| Время разработки, ч. | 168 |
| Количество разработчиков, чел. | 1 |
| Основная заработная плата, руб. | 2613,33 |
| Дополнительная заработная плата, руб. | 392,00 |
| Отчисления в Фонд социальной защиты населения и БРУСП «Белгосстрах», руб. | 1033,83 |
| Прочие прямые затраты, руб | 522,67 |
| Накладные расходы, руб | 261,33 |
| Себестоимость разработки программного средства, руб. | 4823,17 |
| Расходы на реализацию, руб. | 337,62 |
| Полная себестоимость, руб. | 5160,79 |
| Годовые денежные поступления от продажи услуги, руб. | 101 027,09 |
| Количество пользователей, необходимых для окупаемости продукта, чел. | 86 |