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. № 77-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*.*

*Иванова А.А.*

Пров*.*

*.*

*Блинова Е.А.*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*Смелов В.В.*

Реферат

*Лит.*

*Листов*

*1*

11111111, 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*.*

*Иванова А.А.*

Провер*.*

*.*

*Блинова Е.А.*

Н. контр.

*Николайчук А.Н.*

Утв.

*Смелов В.В.*

Abstract

*Лит.*

*Листов*

*1*

11111111, 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*.*

*Иванова А.А.*

Провер*.*

*.*

*Блинова Е.А*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*СмеловВ.В.*

Содержание

*Лит.*

*Листов*

*2*

11111111, 2023

[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*.*

*Иванова А.А.*

Провер*.*

*.*

*Блинова Е.А*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*СмеловВ.В.*

Введение

*Лит.*

*Листов*

*1*

11111111, 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*.*

*Иванова А.А.*

Провер*.*

*.*

*Блинова Е.А*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*СмеловВ.В.*

*Лит.*

*Листов*

*11*

11111111, 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).

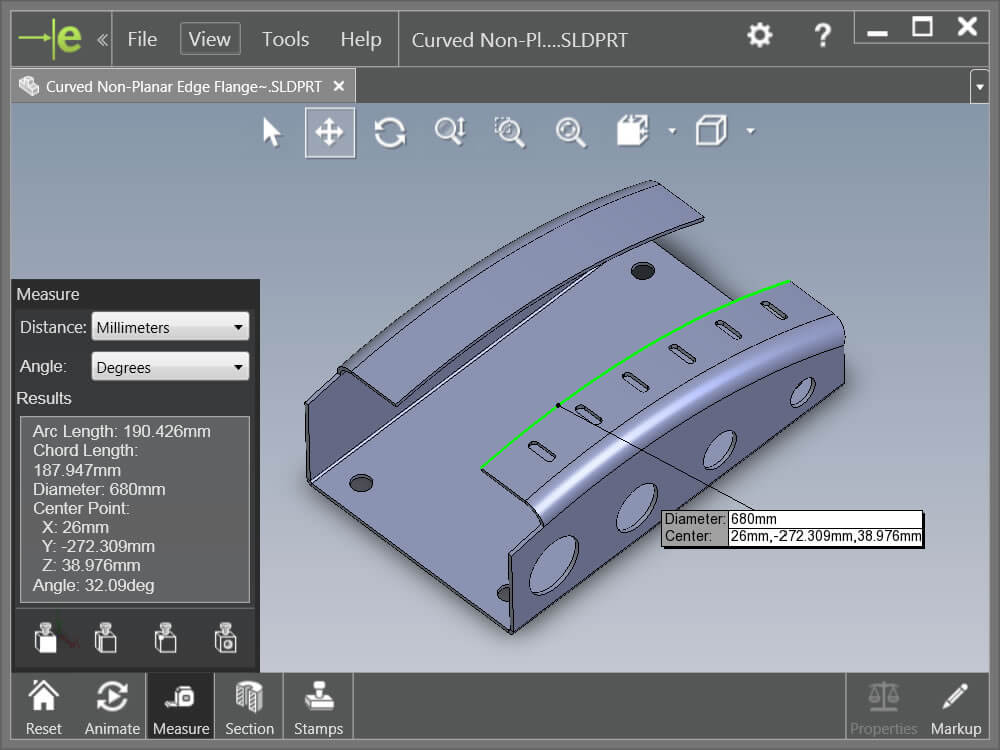


Рисунок 1.5 – Приложение eDrawings Viewer

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*.*

*Иванова А.А.*

Провер*.*

*.*

*Блинова Е.А*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*СмеловВ.В.*

*Лит.*

*Листов*

*7*

11111111, 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 Выбор технологий и библиотек

### 2.3.1 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.

### 2.6.2 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.

### 2.6.3 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.

1.6.4 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.

### 2.6.5 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.

2.7 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*.*

*Иванова А.А.*

Провер*.*

*.*

*Блинова Е.А*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*СмеловВ.В.*

*Лит.*

*Листов*

*11*

11111111, 2023

*3 Проектирование  
программного модуля*

Основной задачей дипломного проекта является создание программного модуля, который разрабатывается как часть платформы и сможет использоваться для облегчения процесса управления тарифными планами, также программный модуль, обеспечивает поддержку международных доменов на платформе *Domain.by*.

Перед началом разработки программного модуля необходимо определить цели и задачи, а также продумать все варианты его использования. Для рассмотрения вариантов использования системы, необходимо построить диаграмму прецедентов.

На платформе предусмотрен функционал проверки входа пользователей и предоставление прав в зависимости от их роли, которых всего 2: администратор и авторизованный пользователь. Для данного дипломного проекта была разработана диаграмма вариантов использования, на которой изображен функционал разрабатываемого модуля. Диаграмма вариантов использования представлена в приложении А настоящей записки.

Авторизованный пользователь в рамках разрабатываемого модуля, может регистрировать международный домен в сети *Internet*, редактировать *NS* адреса для доменов, оплачивать услуги по регистрации доменов и получать сертификат о регистрации домена, а также продлевать регистрацию домена.

Администратор в свою очередь имеет возможность управлять всеми тарифными планами (удалять, создавать и редактировать тарифные планы), осуществлять регистрацию международного домена в сети Internet для пользователей, осуществлять передачу доменов от пользователя к пользователю платформы, редактировать *NS* записи для доменов каждого пользователя.

Также необходимо создать утилиту, в которой содержатся все команды разработанного сервиса для работы с *API OpenProvider* для команды системных администраторов для проверки работы команд и связи с сервером [19].

3.1 Архитектура системы

Архитектура информационной системы – это структура системы, которая включает программные компоненты, видимые снаружи свойства этих компонентов, а также отношения между компонентами информационной системы [20].

Для разрабатываемой системы была выбрана одна из самых распространенных архитектур – трехуровневая архитектура, которая является разновидностью многоуровневой архитектуры. Многоуровневая архитектура – это клиент-серверная архитектура, в которой функции представления, обработки и

хранения данных максимально независимы друг от друга, что облегчает масштабирование системы. К преимуществам трехуровневой архитектуры можно отнести:

* масштабируемость программного средства;
* простота конфигурации, за счет того, что можно изменять конфигурацию одного уровня, а не всего программного средства;
* простота тестирования, за счет того, что программное средство разделено на три уровня, можно провести тесты отдельно для каждого элемента, при этом, не отвлекаясь на другой.

В проектируемой архитектуре системы были выделены следующие уровни: уровень преставления, уровень бизнес-логики и уровень доступа к данным.

Уровень представления (англ. *Presentation layer*): это тот уровень, с которым непосредственно взаимодействует пользователь. Этот уровень включает компоненты пользовательского интерфейса, механизм получения ввода от пользователя. Применительно к разрабатываемой системе на данном уровне расположены представления и все те компоненты, которые составляют пользовательский интерфейс, а также модели представлений, контроллеры, объекты контекста запроса.

Уровень бизнес-логики (англ. *Business layer*): содержит набор компонентов, которые отвечают за обработку полученных от уровня представлений данных, реализует всю необходимую логику приложения, все вычисления, взаимодействует с базой данных и передает уровню представления результат обработки.

Уровень доступа к данным (англ. *Data Access layer*): хранит модели, описывающие используемые сущности. Также здесь размещаются специфичные классы для работы с разными технологиями доступа к данным, применительно к разрабатываемой системе – это класс контекста данных *Entity Framework*. Здесь также хранятся репозитории, через которые уровень бизнес-логики взаимодействует с базой данных.

Архитектурная схема спроектированного модуля представлена в приложении Б настоящей записки.

3.2 Проектирование серверной части

Серверная часть приложения включает в себя уровень доступа к данным, уровень бизнес-логики и часть уровня представления.

На уровне доступа к данным содержится механизм, который обеспечивает работу с базой данных на уровне классов приложения. Этот механизм обеспечивается *ORM*-инструментом *Entity Framework Core*. Уровень доступа к данным также содержит описание моделей сущностей всех таблиц базы данных, которые в дальнейшем будут использоваться сервисами уровня бизнес-логики приложения.

Уровень бизнес-логики содержит сервисы, которые отвечают за всю необходимую для разрабатываемого модуля логику, а именно:

* создание услуги для регистрации международного домена;
* оплата услуги регистрации международного домена;
* регистрация международного домена *OpenProvider*;
* сохранение данных о международном домене в БД;
* регистрация пользователя в *OpenProvider*;
* обновление данных пользователя в *OpenProvider*;
* сохранение данных пользователя в БД;
* поиск информации о доступности регистрации международного домена;
* поиск похожих доступных доменных имен;
* создание тарифного плана;
* редакция тарифного плана;
* удаление тарифного плана;
* изменение *NS* записи для доменов;
* передача домена от пользователя к пользователю;
* конструирование сертификата о регистрации домена.

Уровень представления серверной части представлен контроллерами, которые принимают запросы клиента, обрабатывают их, используя сервисы уровня бизнес-логики, и возвращают ответы на сторону клиента.

3.3 Проектирование клиентской части

Клиентская часть приложения включает в себя уровень представления, на котором используются механизмы генерации *HTML*-разметки, передаваемой пользователю. Для генерации разметки используется описанная в предыдущем разделе технология *Razor Pages* фреймфорка *ASP.NET Core*. Также для асинхронного взаимодействия между клиентской частью и серверной используется технология *AJAX*.

Взаимодействие между пользователем и приложением происходит при помощи *HTTP*-запросов. В основном используются два метода *HTTP*-протокола:

– *GET* – запрос на получение *HTML*-разметки либо данных для их обработки на разметки с помощью *RazorPages*;

– *POST* – запрос на добавление или изменение данных, например, создания тарифного плана или его обновление.

При разработке клиентской части помимо упомянутых выше языка разметки *HTML*, технологии *Razor Pages*, использовался язык программирования *JavaScript*, также использовался язык описания внешнего вида страницы *SCSS* и библиотека*,* для удобного управления структурой *DOM* и создания *AJAX*-запросов.

3.4 Проектирование базы данных

В процессе проектирования системы была использована база данных проекта, которую в последствии необходимо было модифицировать. Всего для функционирования модуля успеваемости было затронуто 27 таблиц, из них 2 – основополагающие для всей системы, частью которой является разрабатываемый модуль. Мню было модифицировано 3 таблицы. Для самого же модуля было спроектировано 3 таблицы.

Для полноты картины, ниже будут описаны все 27 таблиц, начиная с основополагающих. Логическая схема базы данных представлена в приложении В настоящей записки. Таблицы, которые используются в приложении, но являются второстепенными, будут описаны кратко.

Таблица «*Clients*» является самой главной таблицей проекта, в ней содержатся ключи ко всем таблицам, содержащие данные клиентов в разных таблицах. Описание ее полей, которые были использованы в модуле приведено в таблице 3.1. В данной таблице было создано дополнительное поле под названием «*OpenProviderHandle*» для связи с созданной таблицей «*OpenProviderCustomers*».

Таблица 3.1 – Структура таблицы «*Clients*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ClientId | Bigint | Идентификатор записи, первичный ключ |
| UserId | Bigint | Идентификатор пользователя, внешний ключ |
| Country | Nvarchar | Cтрана проживания пользователя |
| ClientType | Nvarchar | Тип клиента |
| IsReseller | Bool | Является ли пользователь ресейлером |
| OpenProviderHandle | Nvarchar | Идентификатор id пользователя в системе OpenProvider, внешний ключ (поле разрешает нулевые значения) |

Таблица «*Users*» хранит информацию для авторизации на платформе. Описание ее полей приведено в таблице 3.2.

Таблица 3.2 – Структура таблицы «*Users*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| UserId | Bigint | Идентификатор записи, первичный ключ |
| Login | Int | Логин пользователя |
| Password | Nvarchar | Хэш пароля пользователя |
| Email | Nvarchar | Почта пользователя |
| LastVisitDate | Datetime | Дата последней авторизации в сети |
| IsActive | Bool | Авторизован ли пользователь |
| RegistrationDate | Datetime | Дата регистрации |

Была создана таблица «*OpenProviderCustomers*» которая хранит информацию, введенную пользователем для регистрации в *OpenProvider*. Описание ее полей приведено в таблице 3.3.

Таблица 3.3 – Структура таблицы «*OpenProviderCustomers*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Handle | Nvarchar | Идентификатор записи, первичный ключ |
| CompanyName | Nvarchar | Название компании (для юридических лиц, поле разрешает нулевые значение) |
| Email | Nvarchar | Почта пользователя |
| State | Nvarchar | Страна |
| Country | Nvarchar | Код страны |
| Number | Nvarchar | Номер дома с корпусом |
| Street | Nvarchar | Улица |
| City | Nvarchar | Город |

Окончание таблицы 3.3

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Zipcode | Nvarchar | Почтовый индекс |
| FirstName | Nvarchar | Имя |
| LastName | Nvarchar | Фамилия |
| PhoneAreaCode | Nvarchar | Код оператора или области телефона |
| PhoneCountryCode | Nvarchar | Код страны телфона |
| PhoneSubscriberNumber | Bigint | Номер телефона |

Таблица «*Services*» представляет таблицу, в которой хранится вся информация о предоставленных услугах пользователю. Описание ее полей, которые были задействованы в разработке модуля приведено в таблице 3.4.

Таблица 3.4 – Структура таблицы «*Services*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ServiceId | Bigint | Идентификатор записи, первичный ключ |
| ClientId | Bigint | Идентификатор клиента, которому предоставляется услуга, вторичный ключ |
| TriffPlanId | Bigint | Идентификатор тарифного плана, вторичный ключ |
| ServiceTypeId | Bigint | Тип услуги, вторичный ключ |
| ServiceOrderDate | Datetime | Дата создания услуги |
| ExpirationDate | Datetime | Дата окончания предоставления услуги |
| Login | Nvarchar | Логин пользователя, создавшего услугу |
| ServicePaymentStatus | Nvarchar | Cтатуc оплаты |

Таблица «*TariffPlans*» хранит информацию о всех тарифных планах, предоставляемых платформой. Описание полей приведено в таблице 3.5.

Таблица 3.5 – Структура таблицы «*TariffPlans*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| TariffPlanId | Bigint | Идентификатор записи, первичный ключ |
| ServiceTypeId | Bigint | Идентификатор типа сервиса, внешний ключ |
| DomainZoneId | Bigint | Идентификатор доменной зоны, внешний ключ |
| IsActive | Bool | Является ли тарифный план доступным для предоставления его пользоваетелям |
| UnicName | Nvarchar | Уникальное имя |
| PeriodUnitId | Bigint | Идентификатор периода, внешний ключ |
| TariffPlanTarificationTyepId | Bigint | Тип тарификации тарифного плана |
| OrderIndex | Bigint | Порядковый номер тарифного плана для отображения в таблице заказа |

Таблица «*TariffPlanTarificationTypes*» представляет собой таблицу, которая содержит все типы тарификации тарифного плана. Эта таблица необходима, так как компания предоставляет разные услуги и некоторые из них оплачиваются единжды, а некоторые один раз в некоторое количество времени. Описание ее полей приведено в таблице 3.6.

Таблица 3.6 – Структура таблицы «*TariffPlanTarificationTypes*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| TariffPlanTarificationTypeId | Int | Идентификатор записи, первичный ключ |
| Value | Nvarchar | Название типа тарификации тарифного плана |

Таблица «*DomainZones*» является таблицей-описанием тарифного плана типа «*Domain*». Таблица была модифицирована, в нее были добавлены поля «*DomainEngineId*» и «*DomainGroupId*». Описание ее полей приведено в таблице 3.7.

Таблица 3.7 – Структура таблицы «*DomainZones*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| DomainZoneId | Int | Идентификатор записи, первичный ключ |
| Name | Nvarchar | Название доменной зоны |
| Label | Nvarchar | Название тарифного плана |
| IsActive | Bool | Активна ли доменная зона |
| DomainEngineId | Int | Идентификатор владельца доменной зоны |
| DomainGroupId | Int | Идентификатор группы доменной зоны |

Таблица «*Domains*» хранит информацию о зарегистрированных доменах на платформе *Domain.by*. Описание ее полей приведено в таблице 3.8.

Таблица 3.8 – Структура таблицы «*Domains*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| DomainId | Bigint | Идентификатор записи, первичный ключ |
| ServiceId | Bigint | Идентификатор услуги, вторичный ключ |
| DomainZoneId | Bigint | Идентификатор доменной зоны, вторичный ключ |
| WhoIsInfoId | Int | Идентификатор информации для сервиса WhoIs |
| Name | Nvarchar | Доменное имя |
| RegistantId | Bigint | Идентификатор пользователя, который зарегистрировал домен |
| HasRecords | Bool | Есть ли у домена иные NS записи |
| HasRedirects | Bool | Есть ли у домена переадресация на иные ресурсы |

Таблица «*DomainGroups*», таблица, которую необходимо создать в проекте, для реализации функций интерфейса, основная суть которых – разделение доменов по группам, она содержит информацию о типах доменных групп в проекте. Описание ее полей приведено в таблице 3.9.

Таблица 3.9 – Структура таблицы «*DomainGroups*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| DomainGroupId | Int | Идентификатор записи, первичный ключ |
| Name | Nvarchar | Название доменной группы |

Так как в проект добавляется еще один реестр доменов, необходимо создать таблица «*DomainEngines*» хранит информацию о владельцах доменной зоны. Для разделения логики по контексту в зависимости от того, на какой сервер необходимо обращаться в случае произведения различных действий над доменом. Описание ее полей приведено в таблице 3.10.

Таблица 3.10 – Структура таблицы «*DomainEngines*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| DomainEngineId | Int | Идентификатор записи, первичный ключ |
| Name | Int | Название владельца доменной зоны |

Таблица «*TariffPlanCosts*» содержит информацию о стоимости тарифного плана. Описание ее полей приведено в таблице 3.11.

Таблица 3.11 – Структура таблицы «*TariffPlanCosts*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| TariffPlanCostId | Bigint | Идентификатор записи, первичный ключ |
| TariffPlanId | Bigint | Идентификатор подгруппы тарифного плпнп, внешний ключ |
| CostId | Bigint | Идентификатор стоимости, вторичный ключ |
| OrderActionId | Int | Тип оплаты |

Таблица «*Costs*» является таблицей содержащей стоимости тарифного плана. Описание полей приведено в таблице 3.12.

Таблица 3.12 – Структура таблицы «*Costs*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| CostId | Bigint | Идентификатор записи, первичный ключ |
| CurrencyId | Int | Валюта |
| Value | Decimal | Стоимость |

Таблица «*Currencies*» хранит информацию об валютах, доступных в приложении. Описание ее полей приведено в таблице 3.13.

Таблица 3.13 – Структура таблицы «*Currencies*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| CurrnecyId | Int | Идентификатор записи, первичный ключ |
| CurrencyName | Nvarchar | Название валюты |
| IsArchive | Bool | Является ли архивированной |

Таблица «*TariffPlanDurations*» является таблицей, в которой хранится вся информация о периодах предоставления тарифного плана. Описание ее полей приведено в таблице 3.14.

Таблица 3.14 – Структура таблицы «*TariffPlanDurations*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| TariffPlanDurationId | Bigint | Идентификатор записи, первичный ключ |
| TariffPlanId | Bigint | Идентификатор тарифного плана, вторичный ключ |
| Discount | Int | Размер скидки в процентах к конкретному периоду |

Окончание таблицы 3.14

|  |  |  |
| --- | --- | --- |
| PeriodId | Bigint | Идентификатор периода, вторичный ключ |
| TarifficationType | Nvarchar | Тип тарификаци периода |

Таблица «*Periods*» содержит размер периода тарифного плана. Описание ее полей приведено в таблице 3.15.

Таблица 3.15 – Структура таблицы «*Periods*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| PeriodId | Bigint | Идентификатор записи, первичный ключ |
| Value | Int | Размер периода |
| PeriodUnit | Nvarchar | Тип периода (год, день, месяц) |

Следующие таблицы являются второстепенными и используются для управления тарифными планами. Все таблицы разделены по типам, так как каждый тарифный план имеет свою информацию для отображения.

Таблица «*GlobalSignSslTariffPlans*» хранит информацию о тарифных планах типа *GlobalSignSsl*. Описание ее полей приведено в таблице 3.16.

Таблица 3.16 – Структура таблицы «*GlobalSignSslTariffPlans*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| GlobalSigntariffPlanId | Bigint | Идентификатор записи, первичный ключ |
| TariffPlanId | Bigint | Идентификатор тарифного плана, вторичный |
| GlobalSignSslTypeId | Bigint | Идентификатор типа ssl ключа, внешний ключ |
| GlobalSignSslSubTypeId | Bigint | Идентификатор подтипа ssl ключа, внешний ключ |
| JuricalOnly | Bool | Является ли только для юридических лиц |

Таблица «*GlobalSignSslTypes*» хранит все типы *ssl* ключей. Описание ее полей приведено в таблице 3.17.

Таблица 3.17 – Структура таблицы «*GlobalSignSslTypes*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| GlobalSignSslTypeId | Int | Идентификатор записи, первичный ключ |
| Name | Nvarchar | Название |

Таблица «*GlobalSignSslSubTypes»* содержит все подтипы *ssl* ключей.Описание ее полей приведено в таблице 3.18.

Таблица 3.18 – Структура таблицы «*GlobalSignSslSubTypes*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id | Int | Идентификатор записи, первичный ключ |
| Name | Nvarchar | Название |

Таблица «*SslTariffSettings*» содержит данные о тарифных планах типа *SslCertificate*. Описание ее полей приведено в таблице 3.19.

Таблица 3.19 – Структура таблицы «*SslTariffSettings*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| SslTariffPlanId | Bigint | Идентификатор записи, первичный ключ |
| TariffPlanId | Bigint | Идентификатор тарифного плана, вторичный ключ |
| SslCertificateTypeId | Int | Тип ssl сертификата |
| DomainsAmount | Int | Количество доменов, на которые распространяется сертификат |
| IsFree | Bool | Является ли бесплатным |
| HasSubDomains | Bool | Распространяется ли на поддомены |

Таблица «*SslCertificateTypes*» содержит типы *ssl* сертификатов. Описание ее полей приведено в таблице 3.20.

Таблица 3.20 – Структура таблицы «*SslCertificateTypes*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| SslCertificateTypeId | Int | Идентификатор записи, первичный ключ |
| Name | Nvarchar | Название типа сертификата |

Схема хранения информации о тарифном плане *VPS* хостинга достаточно сложная, на рисунке 3.1 представлена графическая схема связей таблиц, необходимых для управления тарифного плана типа «*VpsHosting*».

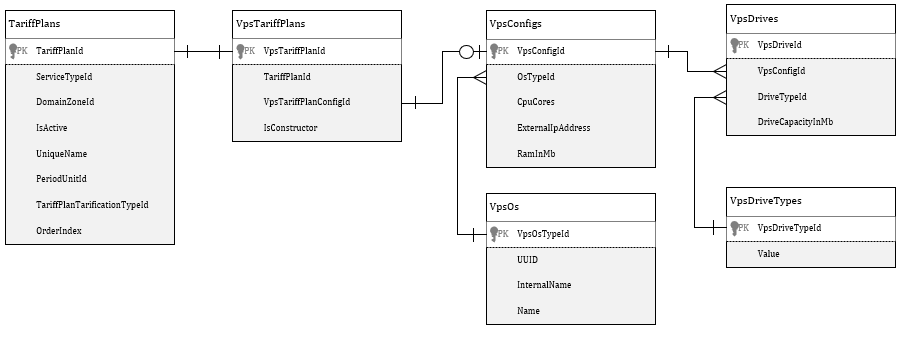


Рисунок 3.1 – Диаграмма связей таблиц, необходимых для управления тарифным планом типа «*VpsHosting*»

Таблица «*VpsTariffPlans*» содержит данные о тарифных планах типа «*VpsHosting*». Описание ее полей приведено в таблице 3.21.

Таблица 3.21 – Структура таблицы «*AttestationSubjectTeacher*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| VpsTariffPlanId | Bigint | Идентификатор записи, первичный ключ |
| TariffPlanId | Bigint | Идентификатор тарифного плана, вторичный ключ |
| VpsTariffPlanConfigId | Bigint | Идентификатор настроек тарифного плана, внешний ключ |
| IsConstructor | Bool | Является ли конструктором |

Таблица «*VpsConfigs*» содержит данные о настройках тарифного планан типа «*VpsHosting*». Описание ее полей приведено в таблице 3.22.

Таблица 3.22 – Структура таблицы «*VpsConfigs*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| VpsHostingConfigId | Bigint | Идентификатор записи, первичный ключ |
| OsTypeId | Int | Идентификатор типа операционной системы |
| CpuCores | Int | Количество ядер хостинга |
| ExternalIpAddress | Int | Количество ip адресов |
| RamInMb | Bigint | Количество оперативной памяти хостинга в мегабайтах |

Таблица «*VpsOs*» содержит данные об операционных системах *Vps* хостингов. Описание ее полей приведено в таблице 3.23.

Таблица 3.23 – Структура таблицы «*VpsOs*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| VpsOsTypeId | Int | Идентификатор записи, первичный ключ |
| UUID | Nvarchar | Идентификатор операционной системы |
| InternalName | Nvarchar | Полное название |
| Name | Nvarchar | Краткое название |

Таблица «*VpsDrives*» содержит данные о дисковых пространствах хостинга. У каждого хостинга может быть несколько дисковых пространств. Описание ее полей приведено в таблице 3.24.

Таблица 3.24 – Структура таблицы «*VpsDrives*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| VpsDriveId | Int | Идентификатор записи, первичный ключ |
| VpsConfigId | Bigint | Идентификатор настройки VPS хостинга, вторичный ключ |
| DriveTypeId | Int | Идентификатор типа диска, вторичный ключ |
| DriveCapacityInMb | Bigint | Размер дискового пространства в мегабайтах |

Таблица «*VpsDriveTypes*» содержит данные о типах дисков для *Vps* хостингов. Описание ее полей приведено в таблице 3.25.

Таблица 3.25 – Структура таблицы «*VpsDriveTypes*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| VpsDriveTypeId | Int | Идентификатор записи, первичный ключ |
| Value | Nvarchar | Тип диска |

Также в схеме есть таблицы-расширения для таблицы «*TariffPlans*» эти таблицы содержат более подробную информацию о тарифных планах. Необходимость в этих таблицах появилась из-за двух языков приложения. В этих таблицах информация хранится на двух языках. Описание полей таблицы «*TariffPlanInfos*» приведено в таблице 3.26.

Таблица 3.26 – Структура таблицы «*TariffPlanInfos*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| TariffPlanInfoId | Bigint | Идентификатор записи, первичный ключ |
| TariffPlanId | Bigint | Идентификатор тарифного плана, вторичный ключ |
| LanguageId | Int | Идентификатор языка, вторичный ключ |
| Name | Nvarchar | Название |
| Description | Nvarchar | Описание |
| ToolTipDescription | Nvarchar | Краткое описание |

Таблица «*Languages*» хранит в себе языки, на которых информация отображается на сайте. Описание полей таблицы «*TariffPlanInfos*» приведено в таблице 3.27.

Таблица 3.27 – Структура таблицы «*Languages*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| LanguageId | Int | Идентификатор записи, первичный ключ |
| Language | Nvarchar | Язык |

Описанные выше таблицы в полной мере удовлетворяют всем функциональным требованиям разрабатываемого программного модуля.

## 3.5 Выводы по разделу

В данном разделе описан процесс разработки диаграммы вариантов использования, в которой был представлен основной функционал разрабатываемого модуля приложения, а также описан процесс выбора архитектуры для разрабатываемого программного модуля. Помимо этого, дано краткое описание серверной и клиентской части приложения и разработана модель базы данных. Представлены и описаны таблицы, а также их особенности.

4 Реализация программного модуля

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*2*

*БГТУ 04.00.ПЗ*

Разраб*.*

*Иванова А.А.*

Провер*.*

*.*

*Блинова Е.А*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*СмеловВ.В.*

*Лит.*

*Листов*

*21*

11111111, 2023

*4 Реализация программного модуля*

4.1 Задачи программного модуля

Главной задачей данного дипломного проекта является создание программного модуля для управления международных доменов, который в свою очередь подразумевает работу с *legacy* кодом и его рефакторингом, а также расширение текущего функционала *Web*-приложения. Исходя из анализа существующих аналогов были поставлены задачи по созданию программных компонентов для работы администраторов и клиентов платформы *Domain.by*, а также задача по созданию интуитивно понятного пользовательского интерфейса. Функционально программный модуль должен выполнять следующие задачи в соответствии с ролями пользователей.

Задачи администратора:

* Создание тарифных планов;
* Изменение тарифных планов;
* Создание скидки для тарифного плана к конкретному периоду;
* Удаление тарифного плана;
* Просмотр тарифных планов;
* Переделегирование домена от пользователя к пользователю платформы *Domain.by*;
* Просмотр данных доменов;
* Просмотр реквизитов клиента;
* Создание заметки к домену;
* Регистрация домена для пользователя.

Задачи обычного пользователя и администратора:

* Регистрация домена;
* Выбор нужных доменных зон и доменного имени;
* Просмотр информации о недоступном домене;
* Выбор периода регистрации домена;
* Выбор похожего доменного имени и периода регистрации;
* Выбор типа оплаты услуги;
* Смена *NS* серверов;
* Получение сертификата о регистрации домена.

Исходя из перечисленных задач пользователей разных типов, можно приступить к реализации программного модуля.

4.2 Анализ и рефакторинг уже существующего кода проекта

Так как данному проекту уже более 20 лет, все взаимодействия с базой данных

были реализованы с помощью фреймворка *Ado.net*, что требовало незамедлительного исправления и переписывания кода для работы с *Entity Framework*. Также следует отметить, что код, который был разработан в то время, является статическим, что влекло за собой большое количество ошибок в работе, а также в скорости исполнения задачи.

Перед созданием программного модуля, который должен был расширить уже имеющийся функционал проекта, а также предоставить новый, необходимо было выполнить рефакторинг *legacy* кода.

На данный момент в проекте идет активное внедрение архитектурного паттерна *MVC* с использованием динамических страниц и их частей, в следствии чего статическую страницу необходимо было удалить. Под словом «статический» в данном контексте подразумевается то, что все фильтры, контекст выпадающих списков, контексты пунктов выбора прописаны непосредственно в коде, от чего необходимо избавиться и создать новый код, который весь контекст данных использует из базы данных.

При детальном рассмотрении кода, было выявлено большое количество статических методов для валидации данных, от которых необходимо избавиться. На данный момент в проекте идет процесс по внедрению *Fluent validation* для валидации всех данных, приходящих от *UI*. Не стоит забывать о том, что при работе со старым кодом при вводе не валидных данных, страница перезагружалась, не оставляя уже введенные пользователем данных.

В ходе анализа проекта было выявлено, что *legacy* код не соответствует не только современным стандартам разработки, но концепции трехуровневой архитектуры, которая реализована в большей части проекта. В результате анализа мной было решено создать заново модуль для заказа доменов, так как производить расширение уже готового кода в соответствии с концепцией трехуровневой архитектуре не предоставлялось возможным, а также создать новый модуль для управления всеми типами тарифных планов, так как прежний функционал проекта, давал пользователю возможность управления лишь четырех из восьми тарифных планов. В ходе рефакторинга *legacy* кода был произведен подобный анализ кода. Задачи и функции, которые он выполнял были полностью реализованы и дополнены, в соответствии с целями и задачами дипломного проекта, в новом модуле для управления международными доменами и тарифными планами. Также стоит отметить, что уровень доступа к данным в базе данных также был переписан с использованием более свежей технологии *Entity Framework*,что не только позволило уменьшить время обработки запросов в базу данных, но и облегчило читаемость кода разработчиком.

4.3 Файловая структура

Перед началом разработки программного модуля мной была продумана файловая структура в соответствии с архитектурой проекта и паттерном проектирования *MVC*, позволяющая быстро ориентироваться в коде, вносить изменения и дописывать новый функционал.

Для работы с *API OpenProvider* был создан отдельный проект «*OpenProviderAPI*» который в свою очередь осуществляет связь с сервером *OpenProvider*, на рисунке 4.1 представлена структура этого проекта.

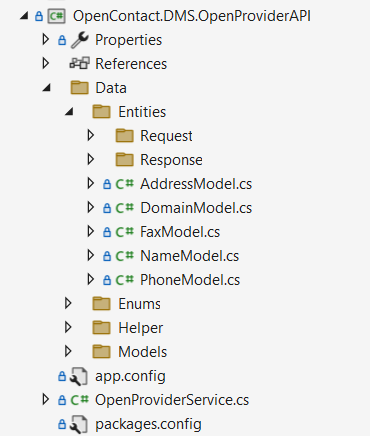


Рисунок 4.1 – Cтруктура проекта «*OpenProviderAPI»*

Основная работа над программным модулем осуществлялась в уже существующих проектах, которые реализуют трехуровневую архитектуру и выделяют следующие уровни:

* уровень доступа к данным (проект *ORM*);
* уровень бизнес-логики (проект *BusinessLogic*);
* уровень представления (проект *Web.UI*).

Стоит отметить, что программный модуль подразумевает собой не только создание новых классов, но и расширение уже имеющихся. Далее в текущем разделе будет подробно описан каждый уровень.

4.4 Уровень доступа к данным

Уровень доступа к данным представляет собой проект типа библиотеки классов с названием *ORM*. Проект использует фреймворк *.NET Core* версии 3.0. Данный проект реализован с помощью *LinqToSql*, с помощью которого в проект добавляются таблицы и редактируются уже существующие таблицы и их связи между собой. Структура уровня представлена на рисунке 4.2.

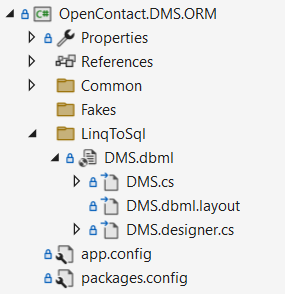


Рисунок 4.2 – Структура уровня доступа к данным

В данном проекте содержится только сама схема базы данных. В папке *LinqToSql* содержится файл *DMS.dbml*. В этом файле и содержится структурная схема базы данных. Сама же база данных находится на сервере, этот файл является графическим отображением базы данных, в него необходимо вручную добавлять все таблицы или их изменения, которые проводились на сервере. На рисунке 4.2. можно увидеть часть файла *DMS.dbml*.

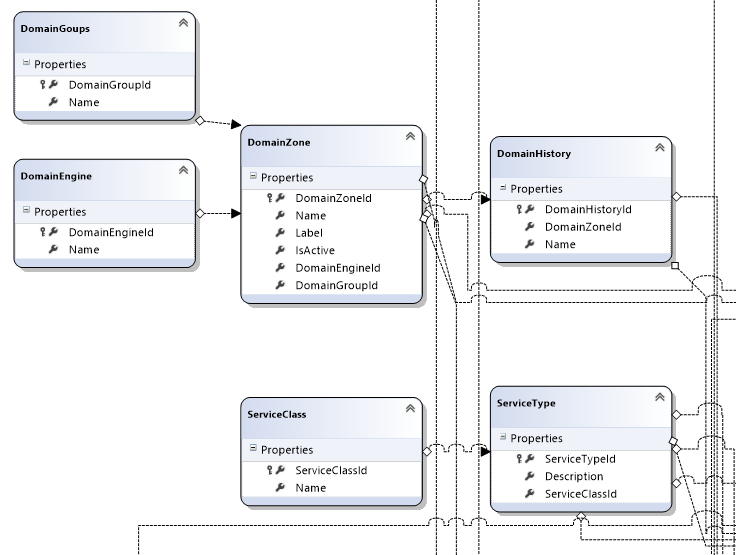


Рисунок 4.3 – Файл *DMS.dbml*

Также у этого файла есть автоматически сгенерированные классы, как видно из рисунка 4.2, в которых содержится информация о таблицах и их связях. В файле *DMS.dbml.layout* содержится та же схема как и в *DMS.dbml* только в *xml* формате со всеми координатами всех таблиц и связей, в файле *DMS.designer.cs* представлен классовый формат структуры базы данных и именно к этому файлу, в котором представлен класс *DMSDataContext* и идет обращение из уровня бизнес-логики. В этом классе описаны все таблицы, в виде вложенных классов, и их связи. Подключение к базе данный осуществляется с помощью класса *DMSDataContextFactory* в отдельном проекте *DataAccess*, так как в проекте *ORM* только сама схема базы данных. В этом классе происходит подключение к базе данных с помощью метода *Create*, который принимает дополнительный необязательный параметр *useDataLoadOptions*. Листинг этого метода показан на листинге 4.1.

|  |
| --- |
| public static DMSDataContext Create(bool useDataLoadOptions = false)  {  var dmsContext = SettingsWeb.Environment.UseTestDatabase  ? new DMSDataContext(SettingsWeb.TestDatabase.ConnectionString)  : new DMSDataContext(SettingsWeb.Database.ConnectionString);  if (useDataLoadOptions){  if (dmsContext.LoadOptions is null)  dmsContext.LoadOptions = GetDataLoadOptions();  }  return dmsContext;} |

Листинг 4.1 – Класс подключения к базе данных

Как видно из листинга 4.1 строка подключения формируется исходя из режима подключения к базе данных тестовый или обычный. Строка формируется исходя из конфигурационных данных автоматически. Параметр *useDataLoadOptions*, о котором упоминалось раннее влияет на «ленивую» подгрузку данных, для оптимизации выполнения запросов в базу данных.

4.5 Уровень бизнес-логики

В качестве уровня бизнес-логики в программном модуле используется уже созданный ранее проект *BusinessLogic* в котором осуществляется все взаимодействие проекта с различными модулями. Именно на этом уровне происходит связь с отдельными проектами, которые выполняют свои задачи. Например, проект, который осуществляет связь с сервером *OpenProvider*, *OpenProviderAPI* или, например, связь с базой данных проекта, о которой говорилось в предыдущей главе. Проект используют фреймворк *.NET Core 3.0*. Структура этого проекта показана на рисунке 4.4.

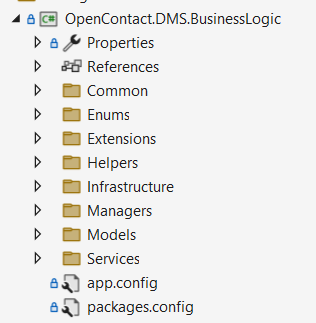


Рисунок 4.4 – Структура проекта BusinessLogic

Все классы в папке *Services* имеют одинаковую структуру, которая представлена на листинге 4.2. В этом листинге можно увидеть конструкторы и переменные класса под названием *OpenProviderCustomerService*, который был создан мной в рамках разработки программного модуля. Этот сервис отвечает за регистрацию, обновление данных клиентов на сервере *OpenProvider*, используя созданный проект *OpenProviderAPI*. Также этот сервис отвечает за сбор данных и их сохранение в таблицу *OpenProviderCustomers* базы данных проекта. Этот сервис используется при заказе международных доменов, блок-схема алгоритма заказа домена находится в приложении Г.

|  |
| --- |
| private readonly ORM.LinqToSql.DMSDataContext \_dataContext;  private readonly bool \_isUseTransactions;  private readonly User \_userChanger;  private readonly OpenProviderConfig \_openProviderConfig;  private ServiceResult<OpenProviderClientRequisiteServiceErrors> \_serviceResult;  public OpenProviderCustomerService(User userChanger, bool useTransactions, ORM.LinqToSql.DMSDataContext dataContext = null)  {  \_dataContext=dataContext  ??DMSDataContextFactory.Create(useDataLoadOptions: false);  \_userChanger = userChanger;  \_isUseTransactions = useTransactions;  \_openProviderConfig = new OpenProviderConfig();  \_openProvider = new OpenProviderService(\_openProviderConfig, CancellationToken.None);}  public OpenProviderCustomerService(bool useTransactions, ORM.LinqToSql.DMSDataContext dataContext = null)  {  \_dataContext = dataContext  ?? DMSDataContextFactory.Create(useDataLoadOptions: false);  \_isUseTransactions = useTransactions;  \_openProviderConfig = new OpenProviderConfig();  \_openProvider = new OpenProviderService(\_openProviderConfig, CancellationToken.None);  }  public OpenProviderCustomerService(ORM.LinqToSql.DMSDataContext dataContext = null)  {  \_dataContext = dataContext  ?? DMSDataContextFactory.Create(useDataLoadOptions: false);  \_openProviderConfig = new OpenProviderConfig();  \_openProvider = new OpenProviderService(\_openProviderConfig, CancellationToken.None);} |

Листинг 4.2 – Конструкторы и переменные класса *OpenProviderCustomerService*

Как говорилось ранее, этот класс отвечает за управление пользовательскими данными для *OpenProvider*. На рисунке 4.5 представлены названия всех методов класса. В классе отсутствует метод удаления пользователя, так как в этом нет необходимости и это не предусмотрено пользовательским интерфейсом.

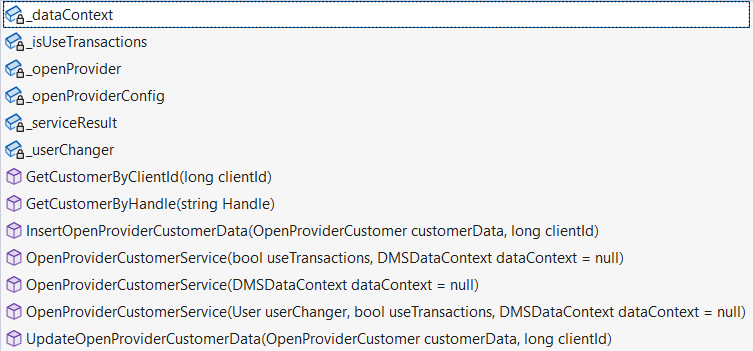


Рисунок 4.5 – Методы класса OpenProviderCustomerService

На примере метода *InsertOpenProviderCustomerData* можно понять схему работы остальных методов всех сервисов в проекте *BusinessLogic*. В листинге 4.3 представлен метод *InsertOpenProviderCustomerData* в котором происходит создание клиента в системе *OpenProvider* с помощью созданного проекта *OpenProviderAPI*, а также вставка пользовательских данных в таблицу *OpenProviderCustomerData*. Так как именно эти данные используются на интерфейсе пользователей, а также в эту таблицу сохранетя *handle* пользователя, то есть его идентификационный номер.

|  |
| --- |
| public ServiceResult<OpenProviderClientRequisiteServiceErrors> InsertOpenProviderCustomerData(OpenProviderCustomer customerData, long clientId)  {  \_serviceResult = new ServiceResult<OpenProviderClientRequisiteServiceErrors>();  try{  if (\_isUseTransactions){  using (var scope = new TransactionScope()){  InsertOperation(customerData);  \_dataContext.SubmitChanges();  scope.Complete();} }  else{  InsertOperation(customerData);  \_dataContext.SubmitChanges();}  return \_serviceResult; }  catch (Exception e){  LogWriterFactory.EventLog.Error(e);  \_serviceResult.AddError(OpenProviderClientRequisiteServiceErrors.CantInsertCustomerData);  return \_serviceResult;} |

|  |
| --- |
| void InsertOperation(OpenProviderCustomer customer){  //запрос в open provider,а затем вытяжка данных оттуда и вставка их в бд  var customerRequest = customer.ConvertToCreateRequestModel();  var createCustomerOperation = AsyncHelper.RunSync(() => \_openProvider.CreateCustomerAsync(customerRequest));  customer.Handle = createCustomerOperation.Data.Handle;  \_dataContext.OpenProviderCustomers.InsertOnSubmit(new ORM.LinqToSql.OpenProviderCustomers  {  Handle = customer.Handle, FirstName = customer.Name.FirstName,  LastName = customer.Name.LastName, City = customer.Address.City,  Country = customer.Address.Country, State = customer.Address.State,  Zipcode = customer.Address.Zipcode,  Street = customer.Address.Street, Number = customer.Address.Number,  CompanyName = customer.CompanyName, Email = customer.Email,  PhoneAreaCode = customer.Phone.AreaCode,  PhoneCountryCode = customer.Phone.CountryCode,  PhoneSubscriberNumber = customer.Phone.SubscriberNumber, });  \_dataContext.SubmitChanges();  var client = \_dataContext.Clients.Where(x => x.ClientId == clientId).FirstOrDefault();  client.OpenProviderHandle = customer.Handle;  \_dataContext.SubmitChanges();}} |

Листинг 4.3 – Метод *InsertOpenProviderCustomerData*

Как видно из листинга 4.3 у метода есть два режима работы и зависят они от параметра *\_isUseTransactions* этот параметр отвечает за откат транзакции в случае завершения одного из методов с ошибкой. Например, если создание клиента на сервере *OpenProvider* закончилось с ошибкой, данные клиента не были бы вставлены в таблицу *OpenProviderCustomers*. Все остальные классы-сервисы, которые были использованы для международных доменов были модифицированы под контекст международных доменов. Путем разделения контекста с помощью switch-case, который принимал параметр обобщения *DomainEngine* были разделены логики для работы с различными реестрами. В листинге 4.4 можно увидеть пример такой модификации. Именно таким методом и был расширен функционал уже существующих сервисов *DomainPaymentService* и *DomainService*, которые осуществляли работу с национальными доменами.

Проект *OpenProviderAPI* на который ссылается вся логика, которая связана с международными доменами был разработан мной. Данный проект осуществляет роль клиента для сервера *OpenProvider*. В этом проекте находятся асинхронные методы для манипулирования данными для управления доменами и клиентами. На рисунке 4.1 изображена структура этого проекта. В папке *Data* находятся все классы, которые используются для манипулирования данными, в этой папке находятся следующие папки: *Entities* в которой находятся модели для запросов и ответов от сервера *OpenProvider*, а также общие модели которые используются и в запросах и в ответах, папка *Enums*, содержащая все классы-обобщения, которые используются в проекте *OpenProviderAPI*, *Helper* – папка, содержащая классы-попщники, например класс с конструктором ссылок для запросов, *Models* – содержит модели класса *OpenProviderService*, например класс для настройки подключения к серверу.

Основным классом проекта *OpenProviderAPI* является класс *OpenProviderService* который выполняет все запросы на сервер *OpenProvider*. Листинг этого класса представлен в приложении Д.

Так как все методы класса OpenProviderService выполнены в асинхронном формате, а вся бизнес-логика проекта в синхронном. Возникает проблема *dead-lock*, когда несколько асинхронных потоков начинают блокировать друг друга. Для решения этой проблемы, мной был разработан класс *AsyncHelper*, который решает эту проблему. В листинге 4.4 можно увидеть этот класс.

|  |
| --- |
| public class AsyncHelper{  private static readonly TaskFactory \_myTaskFactory = new TaskFactory(CancellationToken.None, TaskCreationOptions.None, TaskContinuationOptions.None, TaskScheduler.Default);  public static TResult RunSync<TResult>(Func<Task<TResult>> func)  {  var cultureUi = CultureInfo.CurrentUICulture;  var culture = CultureInfo.CurrentCulture;  return \_myTaskFactory.StartNew(() =>  {  Thread.CurrentThread.CurrentCulture = culture;  Thread.CurrentThread.CurrentUICulture = cultureUi;  return func();  }).Unwrap().GetAwaiter().GetResult();  }  public static void RunSync(Func<Task> func)  {  var cultureUi = CultureInfo.CurrentUICulture;  var culture = CultureInfo.CurrentCulture;  \_myTaskFactory.StartNew(() =>  {  Thread.CurrentThread.CurrentCulture = culture;  Thread.CurrentThread.CurrentUICulture = cultureUi;  return func();  }).Unwrap().GetAwaiter().GetResult();}} |

Листинг 4.4 – Класс *AsyncHelper*

Также кроме задач с управлением доменами, перед мной стояли задачи по управлению тарифными планами. Как и с задачами, касающимися управлением международных доменов, вся бизнес-логика, касающаяся этих задач, осуществлялась в проекте *BusinessLogic*. Для работы с базой данных в классе *TraiffPlanService* были созданы методы для добавления, изменения и создания тарифных планов, также были созданы методы для выявления, тарифных планов, которые можно удалять или редактировать с применением *Entity framework*. Пример методов изменения, добавления и удаления представлен в листинге 4.5.

|  |
| --- |
| public ServiceResult<TariffPlanServiceErrors> UpdateDomainTariffPlan(DomainTariffPlan tariffPlan) {  \_serviceResult = new ServiceResult<TariffPlanServiceErrors>();  try  {  if (!CheckCostsForTariffPlan(tariffPlan.TariffPlanCosts)) {  throw new Exception("При изменении тарифного плана произошла ошибка: стоимости указаны в неверном формате");}  if (!CheckDurationsForTariffPlan(tariffPlan.TariffPlanDurations)){  throw new Exception("При изменении тарифного плана произошла ошибка связанная с периодами или скидками на них");}  if (\_isUseTransasctions) {  using (var scope = new TransactionScope()){  UpdateTariffPlan();  \_dataContext.SubmitChanges();  scope.Complete();}}  else{  UpdateTariffPlan();  \_dataContext.SubmitChanges();}  return \_serviceResult; }  catch (Exception e) {  LogWriterFactory.EventLog.Error(e);  \_serviceResult.AddError(TariffPlanServiceErrors.CantApplyTariffPlanForService);  return \_serviceResult; } } |

Листинг 4.5 – Пример метода изменения тарифного плана

Как видно из листинга 4.5, при любых изменениях, связанных с таблицей *TariffPlans* происходит повторная проверка данных, чтобы избежать проблем целостности и правильности введенных данных.

4.6 Уровень представления

Для уровня представления был использован проект типа веб-приложения *ASP.NET Core MVC* с именем *Web.UI*. Проект так же, как и предыдущие использует фреймворк .*NET Core 3.0* и содержит ссылки проекты уровня бизнес-логики и проекты, выполняющие роль настройщиков работы.

В проекте *Web.UI* содержатся все классы контроллеров, валидации данных, конвертеры моделей из уровня бизнес-логики в уровень представления, страницы разметки и стили для этих страниц. В процессе разработки программного модуля, я использовала следующие папки проекта: *Controllers*, *Converters*, *Extensions*, *Models*, *Views и Validation*.

Папка *Controllers* содержит в себе все классы-контроллеры проекта *Web.UI*, которые принимают запросы и обрабатывают их с помощью классов бизнес-логики, модели которых конвертируются с помощью классов папки *Converters* и далее передают ответы в виде ссылки на разметку, которая находится в папке *Views*, и модели, если она необходима на разметке. Если в контроллер пришел запрос с объектом класса из папки *Models*, она проходит валидацию с помощью классов из папки *Validation* в которой хранятся классы валидации.

Имя класса контролера строится по принципу конкатенации его имени с суффиксом *Controller*, что позволяет среде разработки с легкостью понимать, где находится нужный для ответа на запрос метод, и перенаправлять его на нужный контроллер. Метод контроллера называют экшеном (от англ. *Action*). Экшены помечаются атрибутами типа запроса и авторизации. Атрибут авторизации ограничивает доступ к методу контроллера только для авторизованных пользователей определенных ролей. Для реализации дипломного проекта, мной были разработаны два контроллера: *OrderDomainController*, *TariffPlanController* и *DomainCertificateController*. Также в процессе разработки были модифицированы контроллеры DomainController, DomainsController.

Класс под названием *OrderDomainController* отвечает за регистрацию доменов всех зон. Он обеспечивает прием-отправку данных на/из интерфейса, также контроллер выполняет работу по мелкой обработке данных, которые поступают из интерфейса или для него. В листинге 4.6 можно увидеть код одного из *action* контроллера, в котором происходит процесс проверки доступности доменов исходя из выбранного имени и доменных зон. Весь остальной код контроллера выполнен в том же направлении что и код, расположенный в приложении Е.

В *action CheckDomain* использование классов-валидаторов и библиотеки *FluentValidation* для валидации названия домена является избыточным, так как валидирование объекта типа string намного быстрее проходит без использования классов-посредников. Однако, если в *action* приходит модель из проекта *Web.UI*, она проходит валидацию с использованием классов-валидаторов, которые находятся в папки *Validation* и используют библиотеку *FluentValidation*. Пример такой валидации находятся в листинге 4.6 в котором представлена часть кода метода типа *action* в которой происходит использование классов валидации.

|  |
| --- |
| [HttpPost]  public async Task<ActionResult> OrderDomain(DomainOrderView domainOrderView)  {  var domainName = new DomainName(domainOrderView.DomainName);  var user = MasterModel.UserChanger;  var clientid = MasterModel.ClientRequisites.Client.ClientId;  var domainTariffPlan = \_tariffPlanService.GetDomainTariffPlanByZone(domainName.DomainZoneWithDot, isActiveOnly: true);  if (domainTariffPlan is null) return PartialView("~/Views/OrderDomain/\_DomainValidationError.cshtml", OrderDomainResource.UnkownError);  if (!await CheckDomainStatuses.IsDomainFree(domainTariffPlan.DomainZone.DomainEngine, domainName))  return PartialView("~/Views/OrderDomain/\_DomainValidationError.cshtml", OrderDomainResource.DomainNotFree);  if(domainTariffPlan.DomainZone.DomainEngine== BusinessLogic.Services.Billing.TariffPlans.Enums.DomainEngine.OpenProvider){  var orderDomainValidator = new DomainOrderViewValidator();  var validationResults = orderDomainValidator.Validate(domainOrderView);  ModelState.Clear();  validationResults.AddToModelState(ModelState, string.Empty);  if (!ModelState.IsValid) {  return PartialView("~/Views/OrderDomain/\_OrderDomain.cshtml", domainOrderView); };} |

Листинг 4.6 – Пример использования классов-валидации

Из листинга 4.6 видно, что модель проходит класс валидации под названием *DomainOrderViewValidator*. Листинг этого класса и класса *OpenProviderCustomerViewValidator*, который использует *DomainOrderViewValidator*, находятся в листинге 4.7.

|  |
| --- |
| public class DomainOrderViewValidator : AbstractValidator<DomainOrderView> {  public DomainOrderViewValidator() {  RuleFor(x=>int.Parse(x.PeriodInYears)).InclusiveBetween(1,10)  .WithMessage(x => OrderDomainViewValidationResourse.PeriodLessThanZero);    RuleFor(x => x.OpenProviderCustomerView).SetValidator(new OpenProviderCustomerViewValidator());}}  public class OpenProviderCustomerViewValidator : AbstractValidator<OpenProviderCustomerView> {    public OpenProviderCustomerViewValidator(){    RuleFor(x => x.FirstName).NotEmpty()  .WithMessage(x => OrderDomainViewValidationResourse.Empty);  RuleFor(x => x.LastName).NotEmpty()  .WithMessage(x => OrderDomainViewValidationResourse.Empty);  RuleFor(x => x.CountryCode).NotEmpty()  .WithMessage(x => OrderDomainViewValidationResourse.Empty)  .Matches(@"^\+\d{1,4}$")  .WithMessage(x => OrderDomainViewValidationResourse.CountryCode);  RuleFor(x => x.AreaCode).NotEmpty()  .WithMessage(x => OrderDomainViewValidationResourse.Empty)  .Matches(@"^\d{1,4}$")  .WithMessage(x => OrderDomainViewValidationResourse.AreaCode);  RuleFor(x => x.SubscriberNumber).NotEmpty()  .WithMessage(x => OrderDomainViewValidationResourse.Empty)  .Matches(@"^\d{1,10}$")  .WithMessage(x => OrderDomainViewValidationResourse.AreaCode);  RuleFor(x => x.Email).NotEmpty()  .WithMessage(x => OrderDomainViewValidationResourse.Empty)  .Matches(@"^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$")  .WithMessage(x => OrderDomainViewValidationResourse.Email);  RuleFor(x => x.Country).NotEmpty()  .WithMessage(x => OrderDomainViewValidationResourse.Empty);  RuleFor(x => x.City).NotEmpty()  .WithMessage(x => OrderDomainViewValidationResourse.Empty);  }} |

Листинг 4.7 – Классы-валидаторы, использующие библиотеку *FluentValidton*

Использование библиотеки *FluentValidton* позволяет легко валидировать любые модели данных за счет своей гибкости, так как эта библиотека дает возможность валидировать данные не только с помощью внутренних методов, но и с помощью создания собственных, также можно использовать регулярные выражения как это видно в листинге 4.7. Все условия заканчиваются методом WithMessage*()* который принимает строку, которая будет отображаться в виде текста ошибки. В данном случае метод принимает строку из словаря ресурсов в котором прописаны все тексты ошибок. На рисунке 4.6 можно увидеть словарь ресурсов с текстами ошибок.

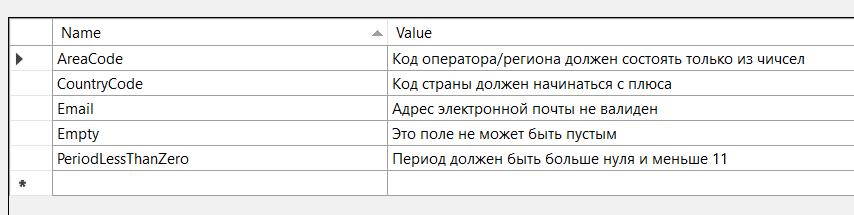


Рисунок 4.6 – Словарь ресурсов текстов ошибок валидации

Все остальные, новые, упомянутые контроллеры выше, контроллеры были сделаны по примеру контроллера *OrderDomain*, работа которого была подробно описана выше. *DomainCertificateController* также сделан по примеру контроллера *OrderDomain*, с единственным исключением, один из его action методов возвращает объект типа *File*, в который передается все необходимые данные регистрации домена и происходит генерация сертификата о регистрации домена.

Как говорилось раннее, результатом всех методов типа action, является страница с моделями, иначе говоря View. View представляет собой разметку, в моем случае это разметка с использованием RazorPages, и ViewModel – любой объект класса View. Для регистрации домена были разработаны страницы разметок с использованием RazorPages, которые были помещены в папку OrderDomain в папке Views. Для общего понимания структуры проекта, было принято решение об определенном нейминге созданных документов: родительской странице дают название контекста, а дочерние называются в соответствии с контекстом, но с символом «\_» в начале.

На рисунке 4.7 представлен скриншот, на котором видны созданные мной документы касающиеся регистрации доменов. Важно отметить, что на эти файлы разметки ссылается только контроллер *OrderDomain*.

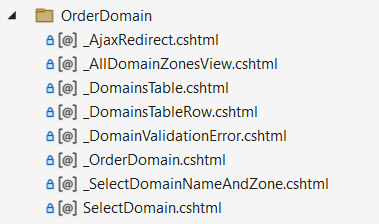


Рисунок 4.7 – Структура папки OrderDomain

На рисунке 4.8 представлен скриншот, на котором видны созданные мной файлы разметки, касающиеся управления тарифными планами. Также все представления были распределены по типу сервисов, для создания общего понимания и легкого усвоения архитектуры проекта.

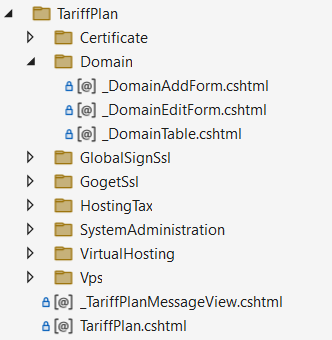


Рисунок 4.8 – Созданные представления для работы с тарифными планами

Исходя из рисунка 4.8 видно, что создана была родительская страница *TariffPlan* представляющая собой контейнер с динамическими частями страницы. В листинге 4.8 можно увидеть реализацию данной страницы.

|  |
| --- |
| @using OpenContact.DMS.Web.UI.Extensions;  @using OpenContact.DMS.Data.Enums.AdminPanel;  @using OpenContact.DMS.Resources.Web.UI.AdminPanel.Accountant;  @using OpenContact.DMS.Data.Extensions.EnumExtensions;  @using OpenContact.DMS.Data.Enums;  @using OpenContact.DMS.Resources.Web.UI.AdminPanel.TariffPlan;  @{Layout = "~/Views/AdminPanel/Shared/\_AdminPanelLayout.cshtml"; |

|  |
| --- |
| ViewBag.Title = TariffPlanResource.TariffPlanPageTitle;  var tabChange = new AjaxOptions{  HttpMethod = "POST",  InsertionMode = InsertionMode.Replace,  LoadingElementId = "service-type-change-preloader",  UpdateTargetId = "service-type-change",  OnBegin = "$('#tariff-plan-management').empty();"  };}  <p class="label">@TariffPlanResource.TariffPlanPageTitle</p>  <div class="tariff-plan-management">  <div class="bordered-wrapper green tariff-plan-table">  @using (Ajax.BeginForm("TabChange", "TariffPlan", null, tabChange, new { id = "table-load-form" })) {  <div class="tabs-container">  <div class="radio-group">  @foreach (var operation in Enum.GetValues(typeof(ServiceType)).Cast<ServiceType> ())  {  <div class="radio-container" id="radio-container">  @Html.RadioButton(nameof(ServiceType), operation, operation == ServiceType.Domain, new { @id = operation, onchange = "$(this).closest('form').submit();" })  @Html.Label(operation.ToLocalizedString(OpenContact.DMS.Resources.Common.Enums.ServiceType.ResourceManager), new { @for = operation })  </div>}  </div>  </div>}  <div id="service-type-change"></div>  @Html.Preloader("service-type-change-preloader", transparent: "transparent")  <div id="tariff-plan-management"></div>  @Html.Preloader("preloader", "middle", null, "transparent")  </div></div>  <script>$("#table-load-form").submit()</script> |

Листнг 4.8 – Реализация родительской страницы

После создания родительской страницы, мною были разработаны все дочерние страницы интерфейса.

Все остальные, модифицированные, контроллеры и файлы разметки были изменены исходя из нового контекста данных. В контроллерах, которые раннее использовали только логику для национальных доменов, была добавлена логика с использованием switch-case, для ее разделения по контексту в зависимости от реестра. Также в контроллере *DomainController*, появилось использование валидации данных, так как одна из задач этого контроллера является переделигирование домена, при переделегировании домена пользователю, который не зарегистрирован в *OpenProvider* – менеджеру необходимо ввести данные пользователя. Именно поэтому в листинге 4.7 есть разделение на валидирование общей модели *OrderDomain* и модели *OpenProviderCustomerData*.

4.7 Выводы по разделу

В данном разделе были описаны элементы архитектуры приложения и их реализация. Разработка приложения осуществлялась с использованием веб-фреймвор-ка *ASP.NET Core* *MVC*. В процессе разработки были учтены особенности работы с данным фреймворком, использованы стандартные средства разработки, а также подключаемые сторонние библиотеки.

В результате выполнения дипломного проектирования был разработан модуль для управления международных доменов и тарифных планов. Результатом моей работы является готовый интерфейс, который отвечает всем задачам, поставленным перед мной. В отличии от рассмотренных аналогов, данный модуль имеет понятный и дружелюбный к пользователю интерфейс, а также данный модуль решает некоторые проблемы, которые были в, реализованном до меня, коде.

На рисунках 4.9 и 4.10 представлен скриншот страницы «Тарифные планы» из раздела «Настройки» до моего исправления. На данном скриншоте отчетливо видно, что данные плохо структурированы, границ колонок не видно, также интерфейс блеклый и не подходит под определение «User friendly», а также не подходит под стиль общего интерфейса. ПОДНЯТЬ

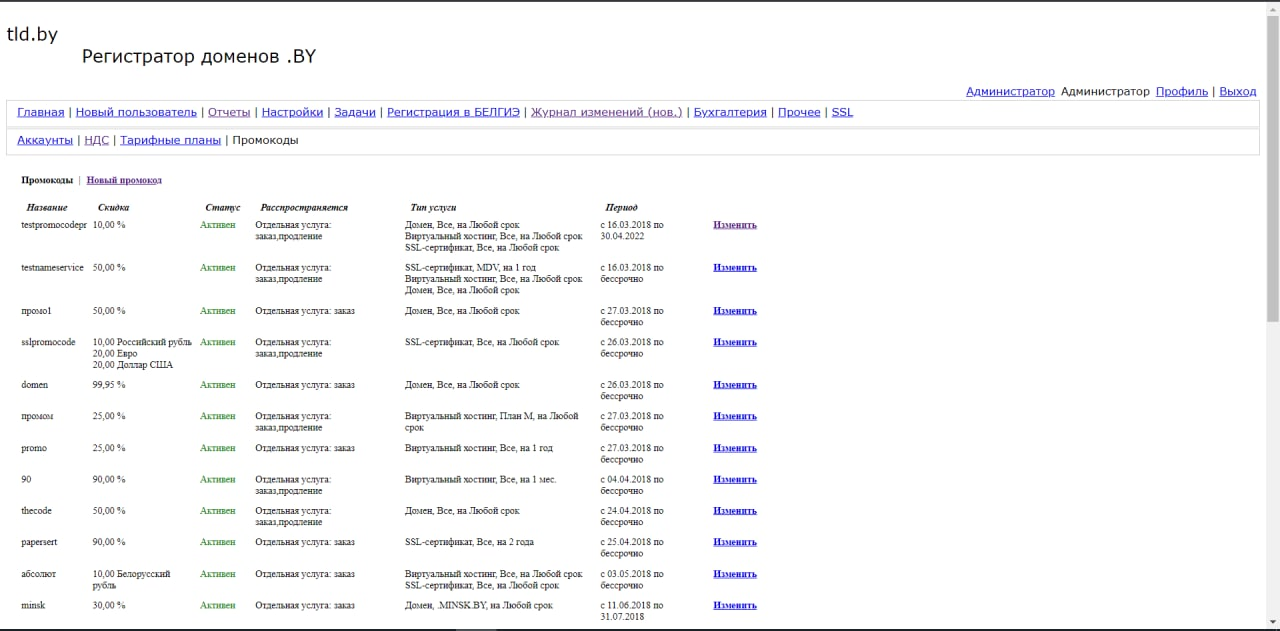


Рисунок 4.9 – Начало старой страницы «Тарифные планы» из раздела «Настройки»

Далее, чтобы изменить тарифный план или создать его необходимо нажать на кнопку «Изменить» возле нужного нам тарифного плана или нажать кнопку «Новый тарифный план» соответственно. На рисунке 4.10 можно увидеть структуру интерфейса при создании или изменении тарифного плана, при создании тарифного плана все поля сразу пустые и открыты для изменения.

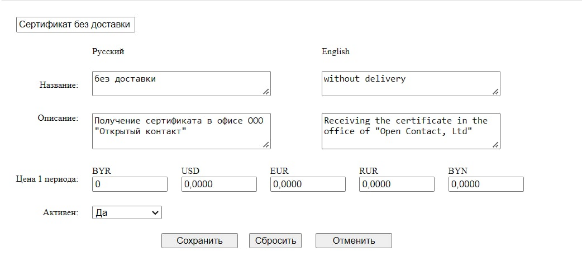


Рисунок 4.10 – Интерфейс старой страницы создания/изменения тарифного плана

Новый интерфейс, изображенный на рисунках 4.11 и 4.12, наглядно демонстрирует результаты моей работы над задачами по управлению тарифными планами.

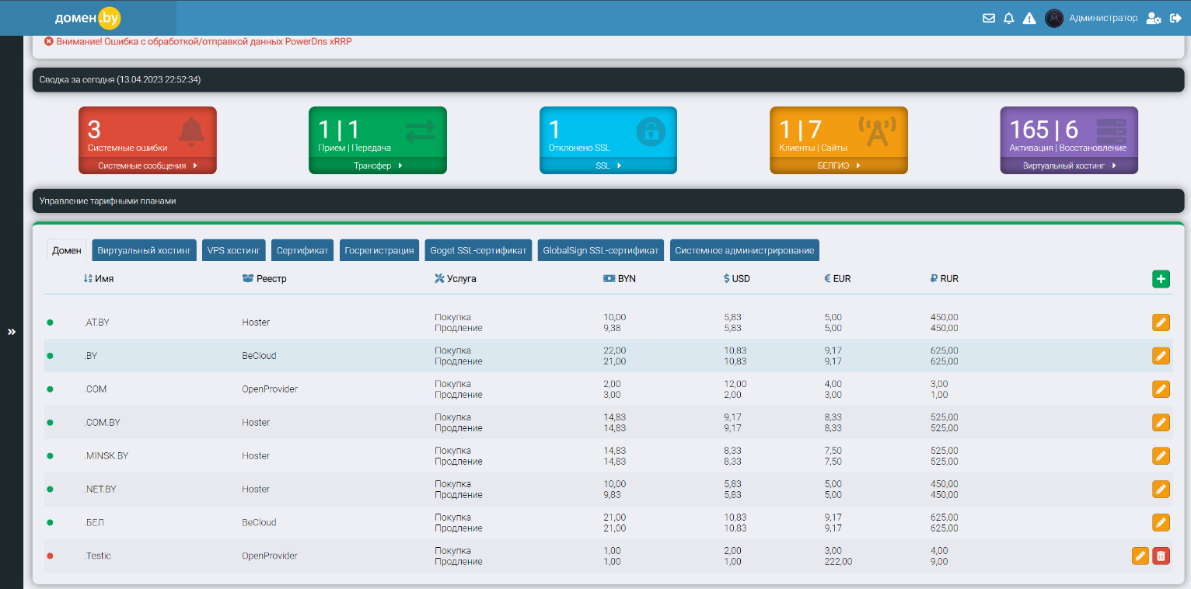


Рисунок 4.11 – Вид новой страницы «Тарифные планы»

Как видно из рисунка 4.11, не все тарифные планы можно удалить. На рисунке 4.12 представлен вид окна просмотра и редактирования тарифного плана, который нельзя изменить полностью.

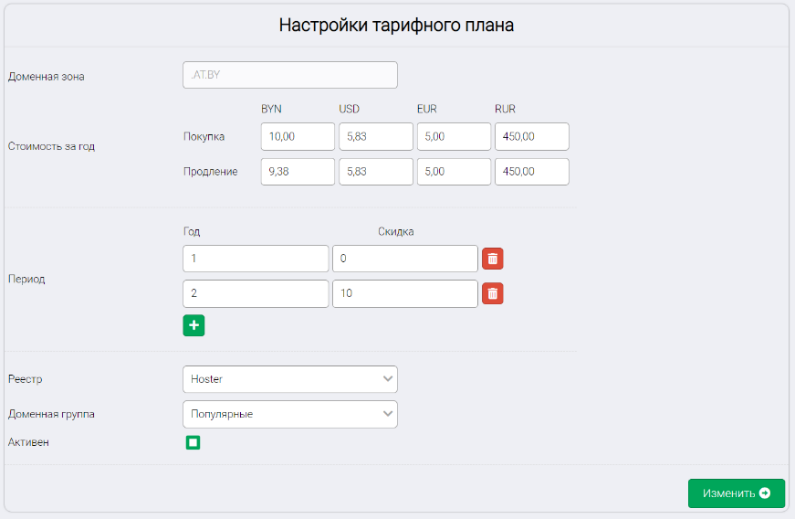


Рисунок 4.12– Вид нового окна просмотра и редактирования тарифного плана

Также мною был реализован динамический элемент добавления/удаления/изменения периода услуги и скидки на него. На рисунке 4.13 представлен этот элемент.

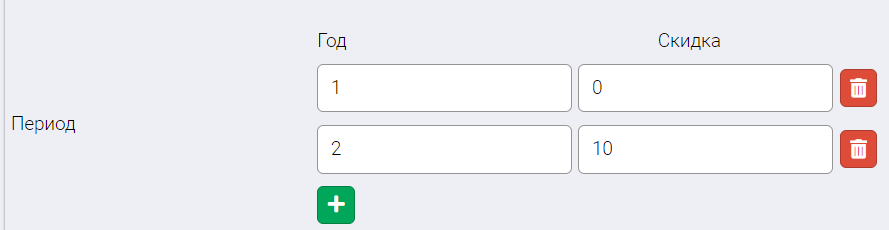


Рисунок 4.13 – Динамический элемент изменения периодов работы тарифного плана

В новом модуле для заказа международных доменов дизайн интерфейса специально не изменился, так как он был дружественный и к нему привыкли клиенты платформы, однако, на этапе подбора нужного домена интерфейс был немного изменен, так как услуга по регистрации домена уже предоставляется компанией не только на 1 и 2 года. На рисунке 4.14 можно увидеть старый интерфейс страницы подбора домена.



Рисунок 4.14 – Старый интерфейс подбора домена

В новом интерфейсе появилась возможность динамического выбора периода, также можно отметить, что предшествующий интерфейс был статическим, а новый – динамическим, что решает проблемы по периоду предоставления услуг. На рисунке 4.15 можно увидеть новые динамические элементы.

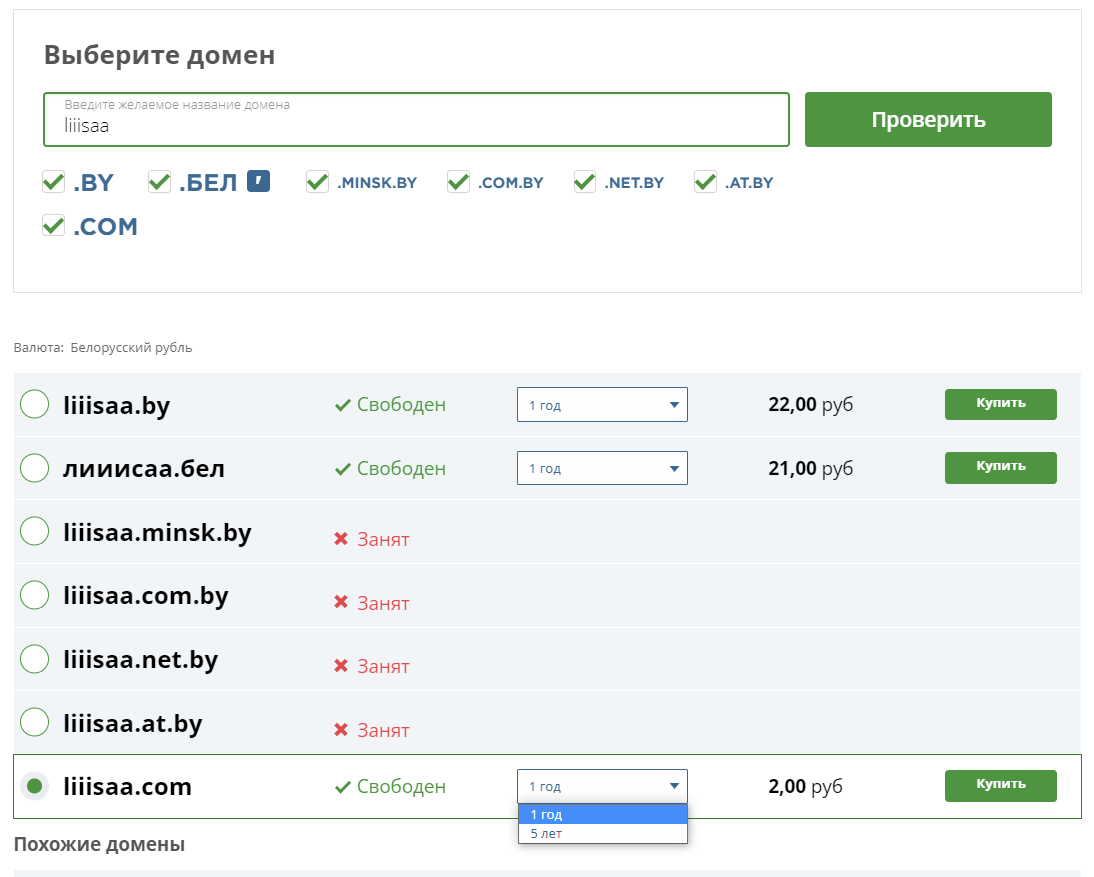


Рисунок 4.15 – Новая страница подбора домена с динамическим дизайном

Программное средство реализовано в полном объеме и готово к прохождению тестирования.

5 Тестирование программного модуля

*ФИО*

*Подпись*

*Дата*

*Лист*

*1*

*2*

*БГТУ 05.00.ПЗ*

Разраб*.*

*Иванова А.А.*

Провер*.*

*.*

*Блинова Е.А*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*СмеловВ.В.*

*Лит.*

*Листов*

*7*

11111111, 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 программного модуля, разработанного в рамках данного дипломного проекта, а также тестирование некоторого функционала системы управления учебным процессом ВУЗа, необходимого для доступа и корректной работы модуля успеваемости вышеназванной системы.

## 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 заказа домена, а также изменения *NS* адресов.

Таблица 5.1 – Функциональные тест-кейсы пользовательского интерфейса

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Статус |
| Выбор нужных доменных зон и ввод названия домена | Переход на страницу с доменами, статусами доступа, ценами услуг и их периодами предоставления | Успешно |
| Выбор иного периода предоставления услуги | Смена стоимости услуги | Успешно |

Окончание таблицы 5.1

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Статус |
| Ввод пустого доменного имени | Вывод сообщения об ошибке, связанной с пустым доменным именем | Успешно |
| Ввод невалидного доменного имени | Вывод сообщения об ошибке, связанной с вводом невалидного доменного имени | Успешно |
| Ввод символа «ʼ» через кнопку на интерфейсе | Ввод символа «ʼ» на строку для ввода доменного имени | Успешно |
| Выбор нужного домена и периода предоставления услуги | Переход на страницу, в которой при заказе международного домена будет форма для ввода данных для регистрации клиента в OpenProvider | Успешно |
| Ввод всех данных для регистрации в OpenProvider | Переход на страницу для оплаты | Успешно |
| Ввод не всех данных для регистрации в OpenProvider | Вывод сообщений об ошибке возле полей, которые не заплнены | Успешно |
| Ввод невалидных данных для полей ввода почты, региона, телефона | Вывод сообщений о невалидности данных возле полей, в которые введены неправильные данные | Успешно |
| Подтверждение оплаты на странице оплаты | Переход на страницу со списком аттестаций, созданная аттестация в конце списка | Успешно |
| Переход на страницу настроек домена и далее на страницу для ввода NS записей | Переход на страницу с формой для ввода NS записей | Успешно |
| Ввод NS записей на странице настроек домена | Обновление страницы с настройками и показ информации о новых NS записей, прикрепленных к домену | Успешно |

Также было проведено тестирование интерфейса для администраторов, результаты тестирования представлены в таблице 5.2.

Таблица 5.2 – Функциональные тест-кейсы интерфейса администраторов

|  |  |  |
| --- | --- | --- |
| Переделегирование домена другому пользователю | Открытие страницы для подтверждения делегерирования с вводом данных для регистрации клиента при переделегировании международного домена незарегистрированному пользователю в OpenProvider | Успешно |
| Отправка пустых полей и невалидных данных в форме регистрации клиента при переделегировании домена | Ввывод ошибок валидации полей рядом с полями, в которых допущены ошибки | Успешно |
| Подтверждение переделегации домена | Вывод информации об успешной переделегации домена | Успешно |

Окончание таблицы 5.2

|  |  |  |
| --- | --- | --- |
| Создание тарифного плана на странице «Тарифные планы» | Вывод окна для ввода данных нового тарифного плана | Успешно |
| Удаление тарифного плана | Вывод сообщения об успешном удалении тарифного плана | Успешно |
| Ввод невалидных данных в окне для создания/редактирования тарифного плана | Вывод сообщений об ошибке валидации данных | Успешно |
| Ввод валидных даннхы для создания/редактирования тарифного плана | Вывод сообщения об успешном создании/изменении тарифного плана | 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 разработанной системе управления доменов и тарифных планов в основном используется два вида валидации: мгновенная и после отправки формы.

Результат ввода пустого доменного имени показан на рисунке 5.1.

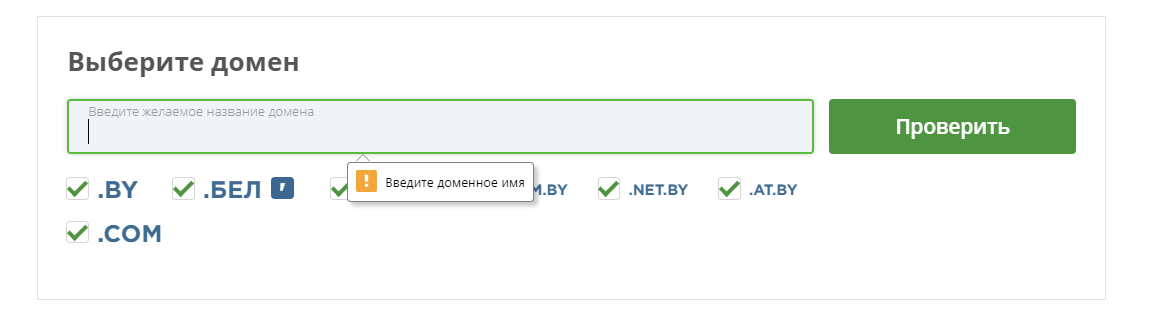


Рисунок 5.1 – Результат нажатия на кнопку «Проверить» при невведенном доменном имени

Также пользователи могут допускать ошибки при вводе доменного имени, например вставлять спецсимволы, которые запрещены в доменных именах. На рисунке 5.2 представлен вид страницы ввода доменного имени при введении неверного доменного имени.

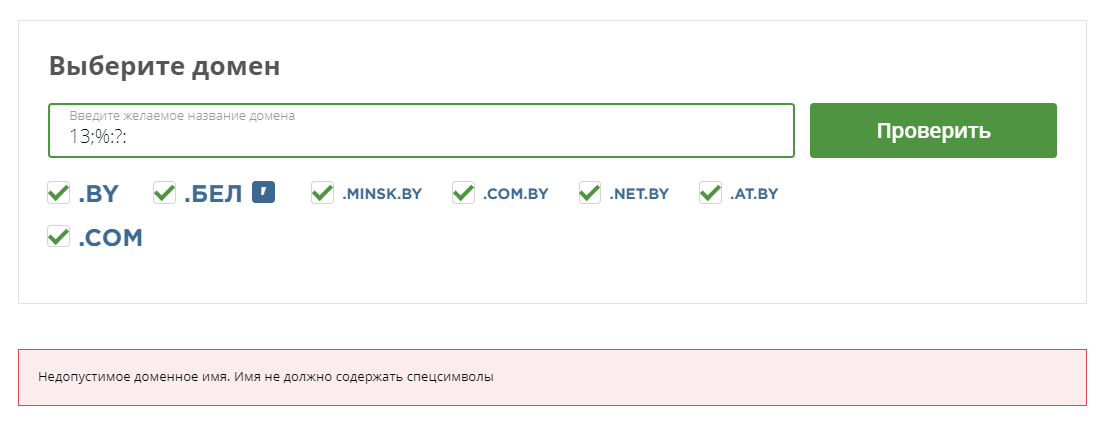


Рисунок 5.3 – Результат валидации неверного доменного имени

При заказе домена из международной доменной зоны, пользователю необходимо заполнить форму для регистрации или изменения клиентских данных в системе *OpenProvider*. На рисунке 5.4 представлена эта форма с введенными неправильными значениями после чего была нажата кнопка «Продолжить».

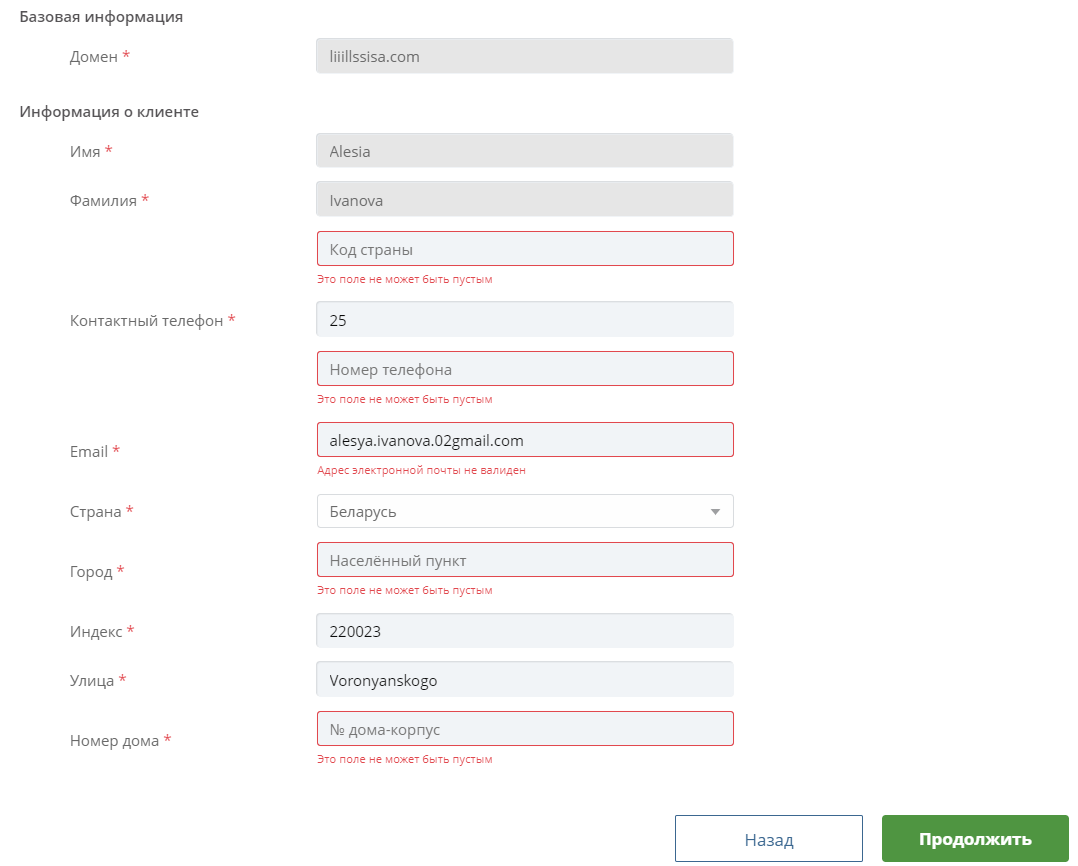


Рисунок 5.4 – Результат валидации неверных или пустых пользовательских данных

Как видно из рисунка 5.4 валидация почты есть не только на то пустое поле или нет, но и на сам фактор верности данных.

Также валидация есть и на странице с вводом новых *NS* записей. На рисунке 5.5 можно увидеть результат валидации неверных данных.

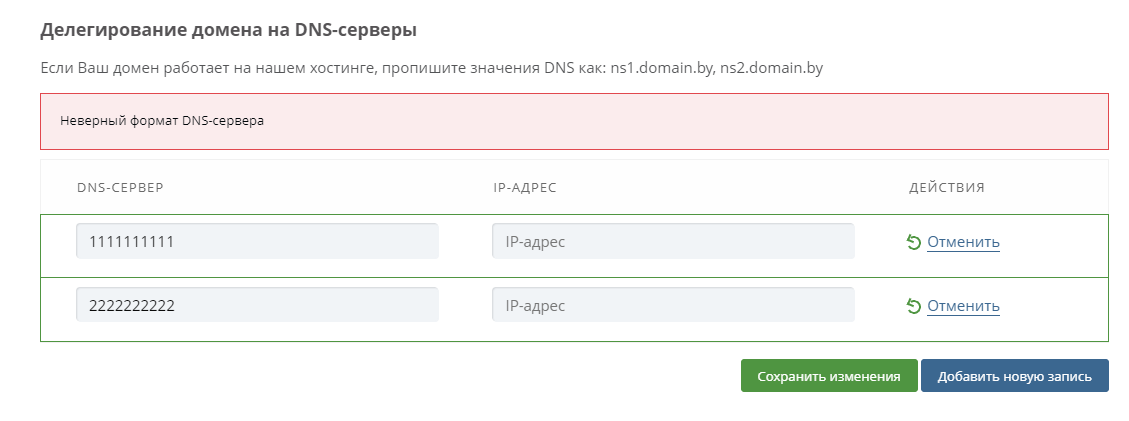


Рисунок 5.5 – Результат валидации неверных данных адресов *DNS*-сервера

Также валидация адресов *DNS*-серверов есть и при вводе одинаковых адресов, результат такой проверки предоставлен на рисунке 5.6.

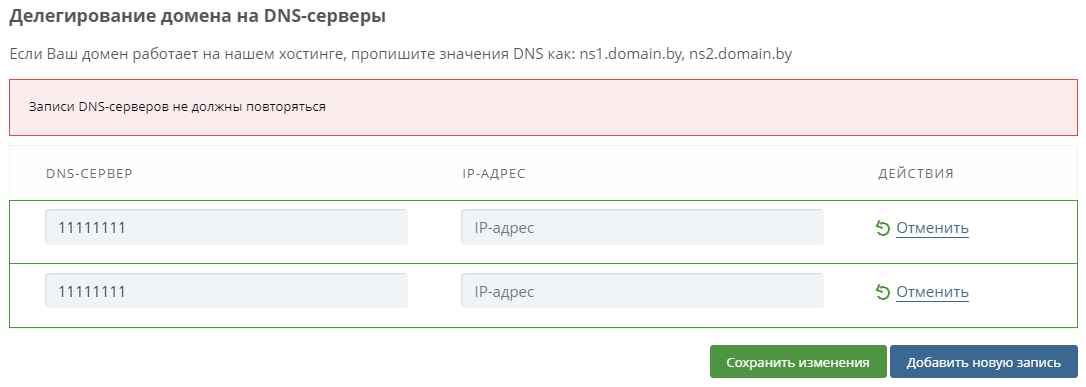


Рисунок 5.6 – Результат валидации повторяющихся адресов *DNS*-серверов

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

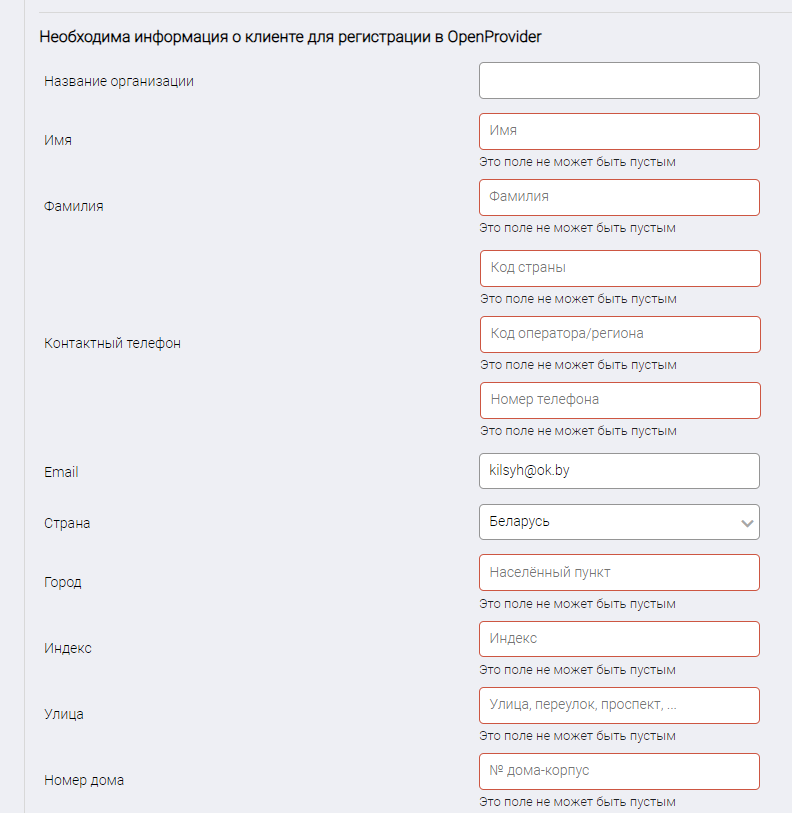


Рисунок 5.7 – Результат ввода невалидных данных пользователя

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.

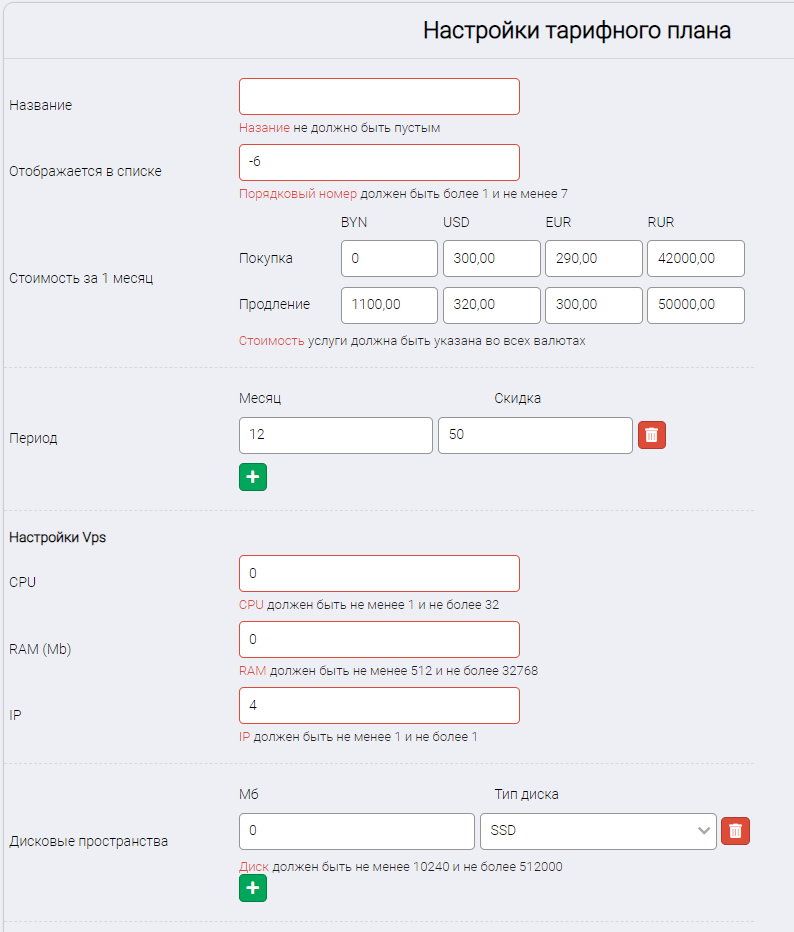


Рисунок 5.8 – Вид окна редакции после тестирования его пустыми значениями

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.

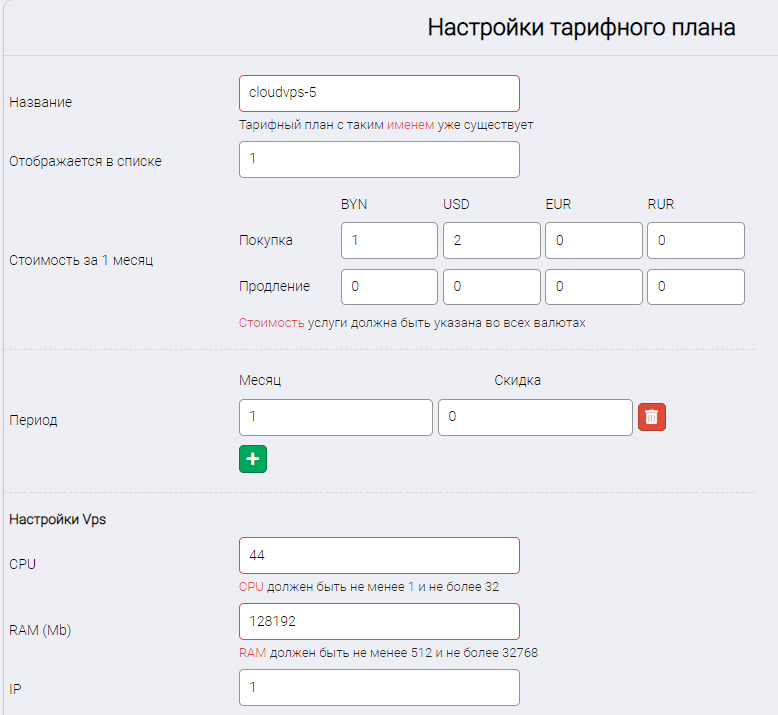


Рисунок 5.9 – Результат тестирования полей

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

5.3 Автоматическое тестирование

Автоматическим тестированием в команде занимается тестировщик, который под каждую задачу создает блок автоматических тестов. Тестировщик запускает все тесты на платформе «*Jenkins*» далее на этой платформе я могу увидеть результаты тестирования. На рисунке 5.10 изображен скриншот, из платформы «*Jenkins*» с успешно пройденными автоматическими тестами. Автоматическое тестирование тестировщиками особо важно, так как тестировщики ставят нестандартные тесты над разработанными модулями.

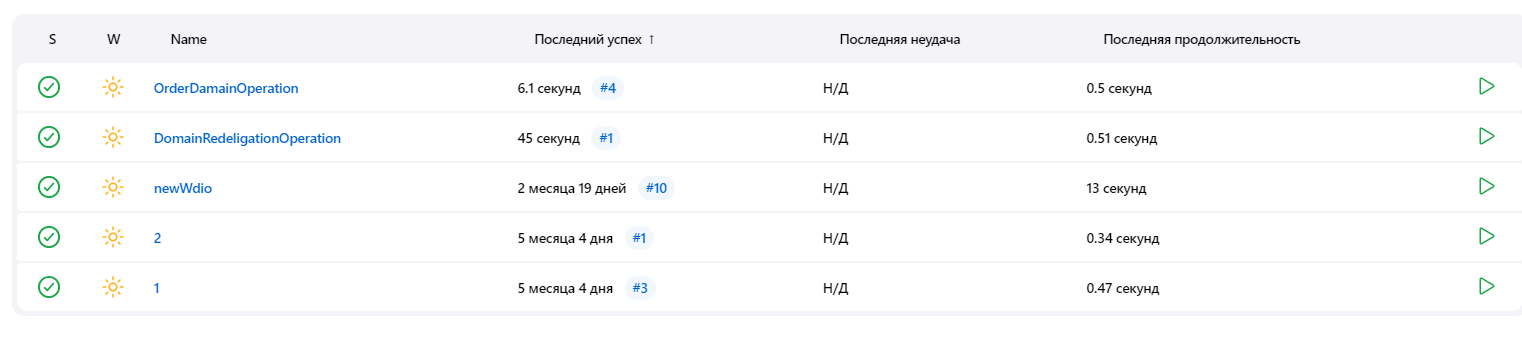


Рисунок 4.10 – Результат автоматизированного тестирования

После автоматического тестирования, мы с уверенностью можем сказать, что задача выполнена верно, а значит ее можно передавать проектному менеджеру для ее публикации.

5.4 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*.*

*Иванова А.А.*

Провер*.*

*.*

*Блинова Е.А*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*СмеловВ.В.*

*Лит.*

*Листов*

*16*

11111111, 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.

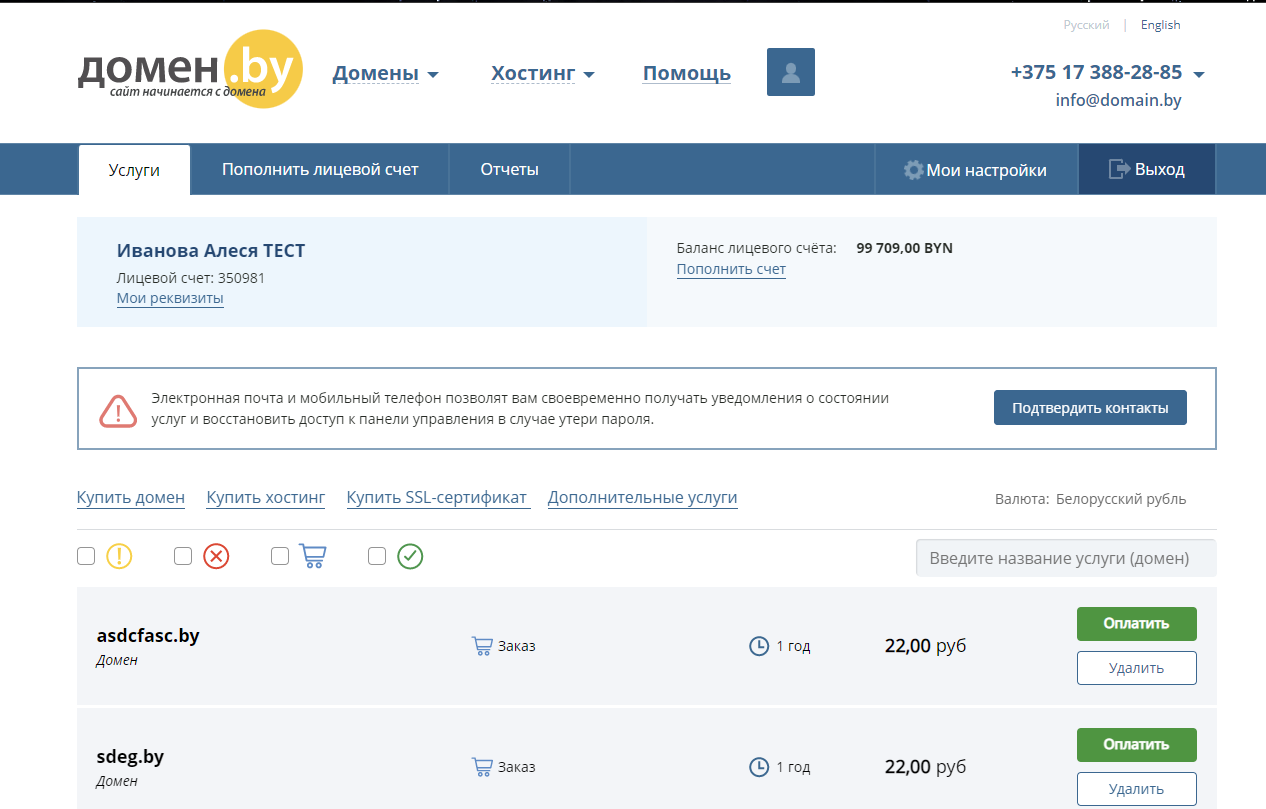


Рисунок 6.1 – Главная страница личного кабинета

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

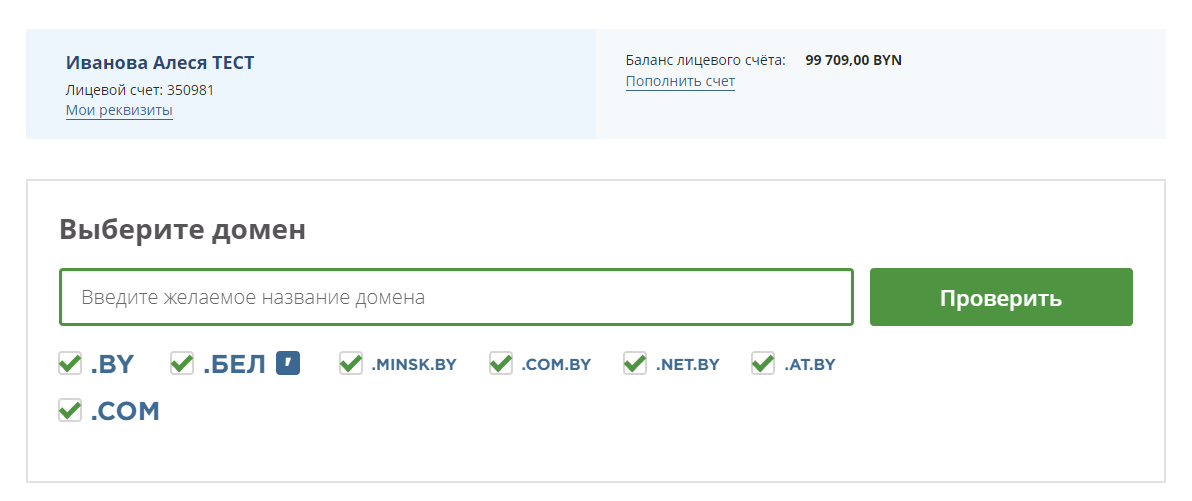


Рисунок 6.2 – Страница выбора названия домена и доменных зон для регистрации

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.

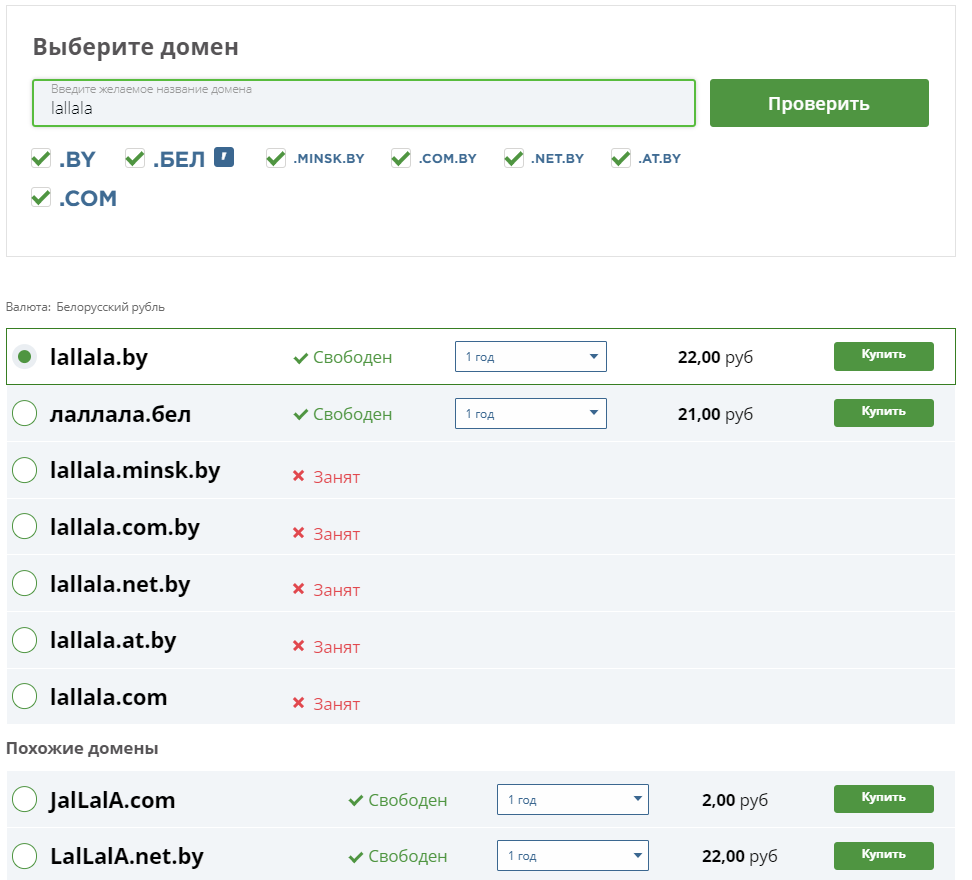


Рисунок 6.3 – Страница выбора домена и периода регистрации

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.

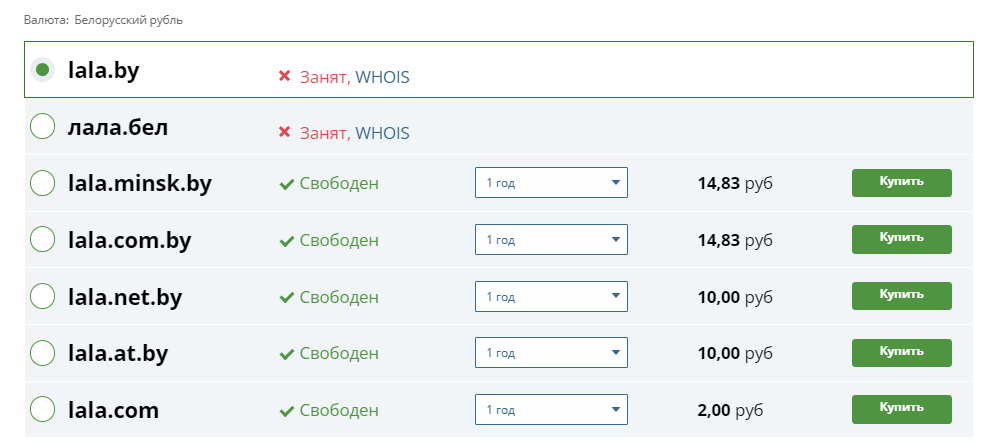


Рисунок 6.4 – Таблица доменов с недоступным для регистрации доменом национальной зоны

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.

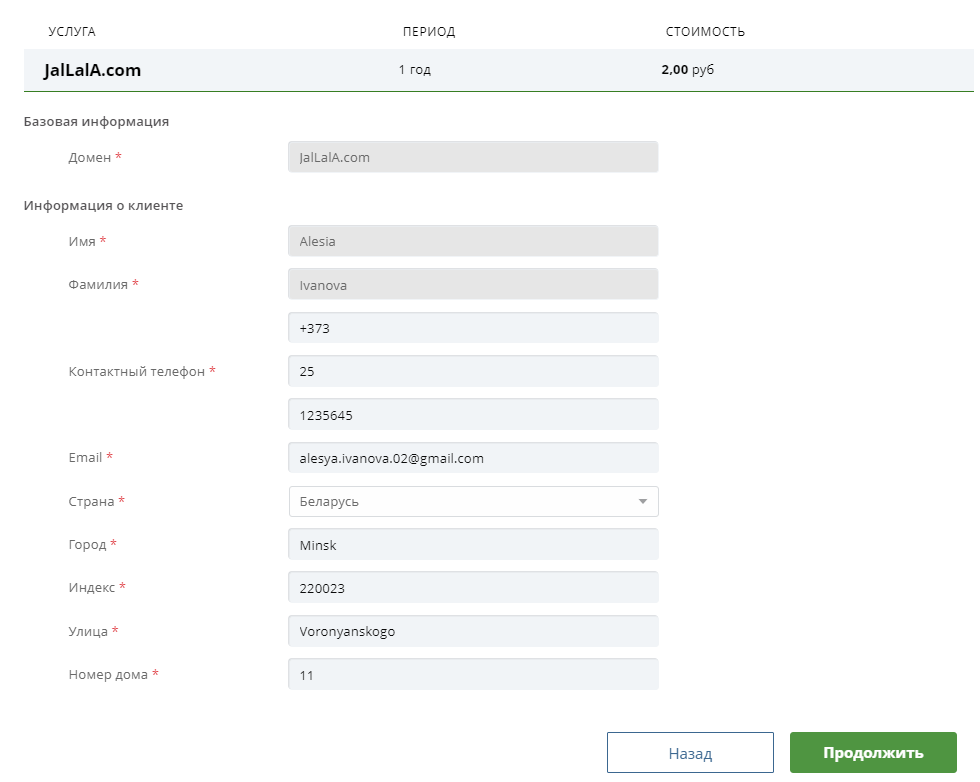


Рисунок 6.5 – Страница подтверждения и заполнения данных для регистрации в *OpenProvider*, а также подтверждение заказа услуги по регистрации международного домена

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

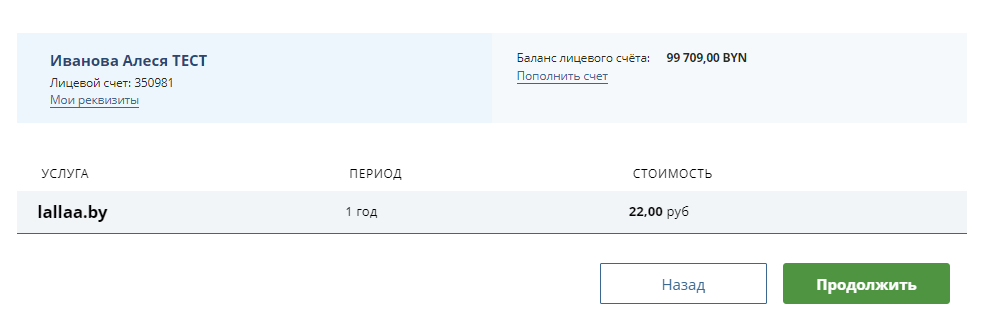


Рисунок 6.6 – страница подтверждения заказа услуги по регистрации домена международной зоны

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.

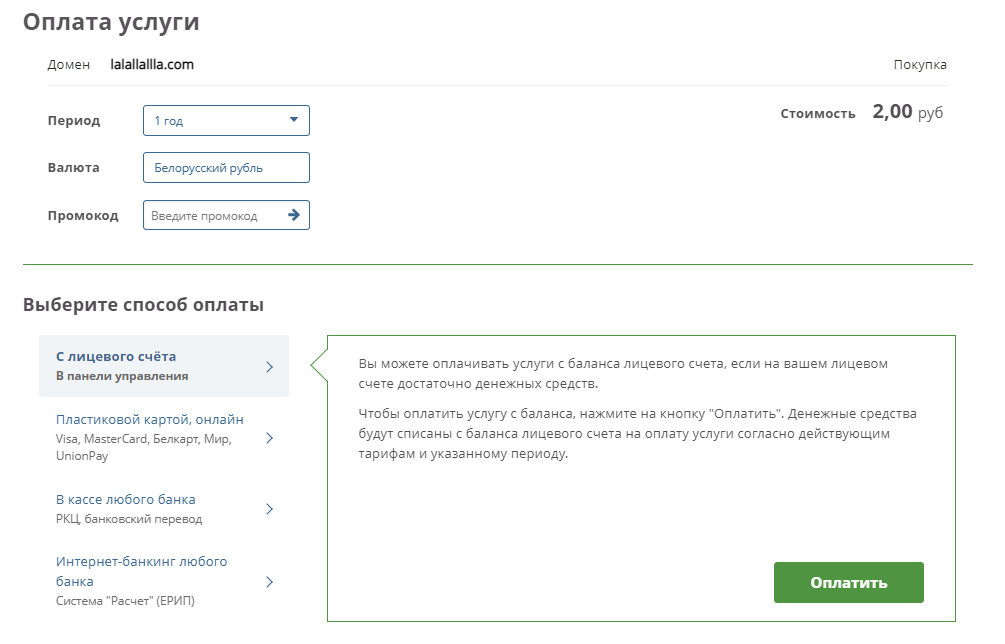


Рисунок 6.7 – Страница оплаты услуг

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.

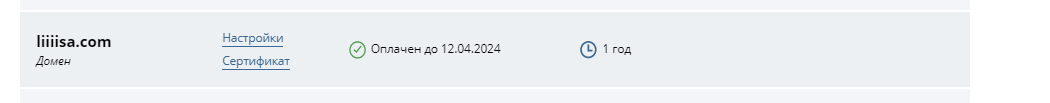


Рисунок 6.8 – Строка таблицы с оплаченным доменом

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.

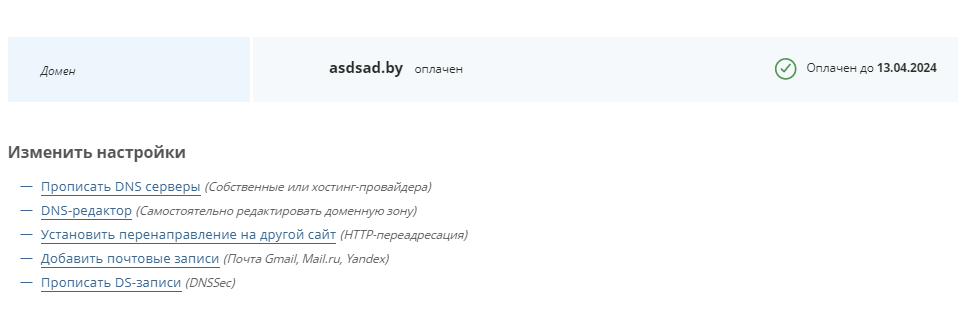


Рисунок 6.10 – Страница настроек домена

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.

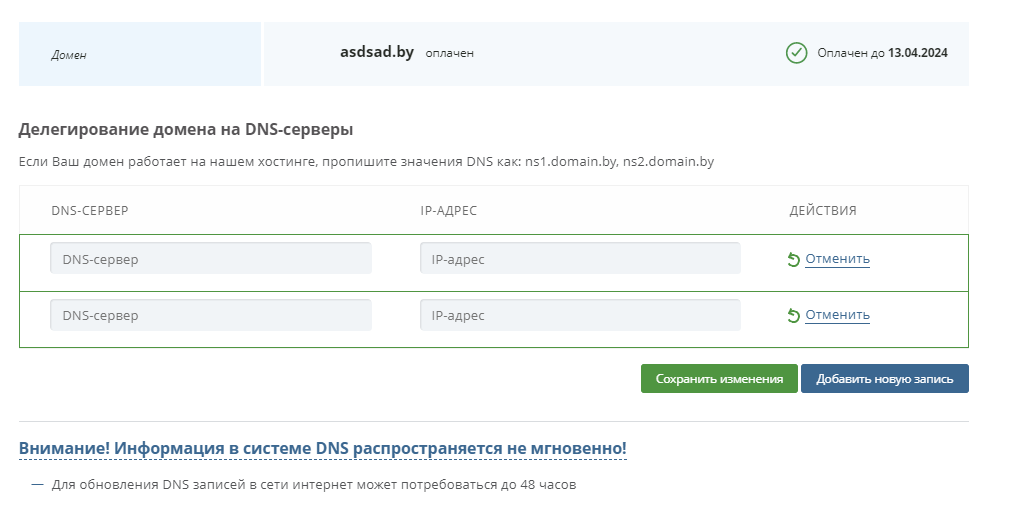


Рисунок 6.11 – Страница изменения *NS* записей домена

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».

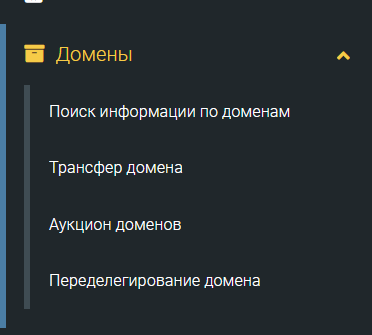


Рисунок 6.15 – Доступные операции для управления доменами

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.

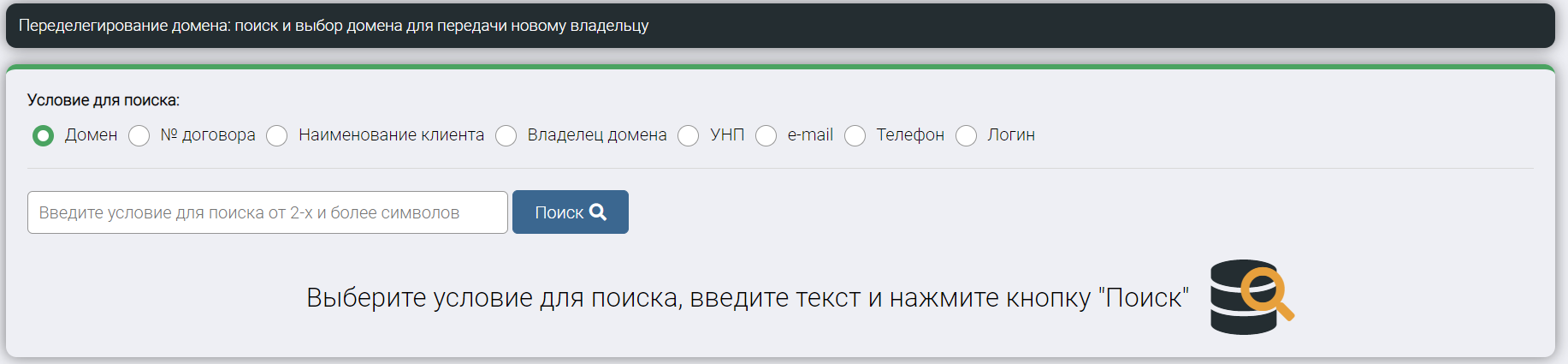


Рисунок 6.16 – Страница переделегирования домена

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.

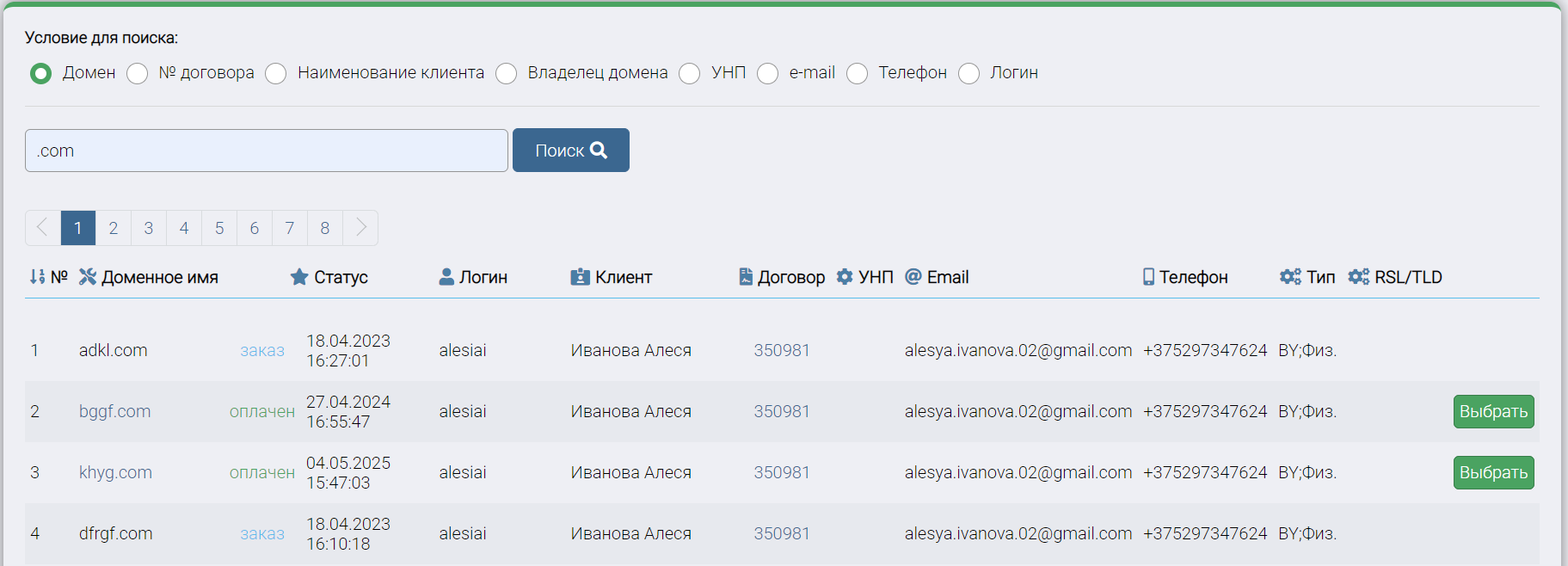


Рисунок 6.17 – Таблица доменов, удовлетворяющих условия поиска

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.

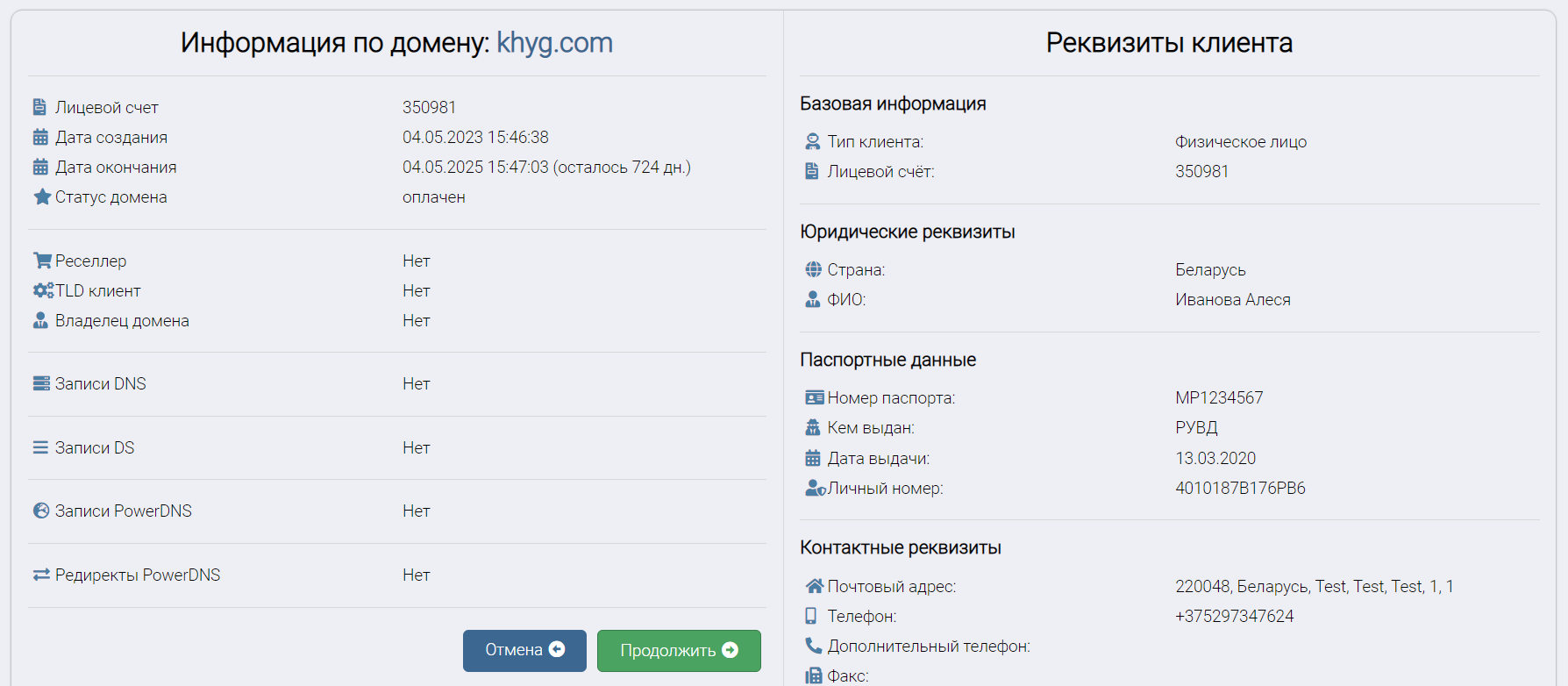


Рисунок 6.18 – Окно для просмотра информации о домене и его владельце

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.

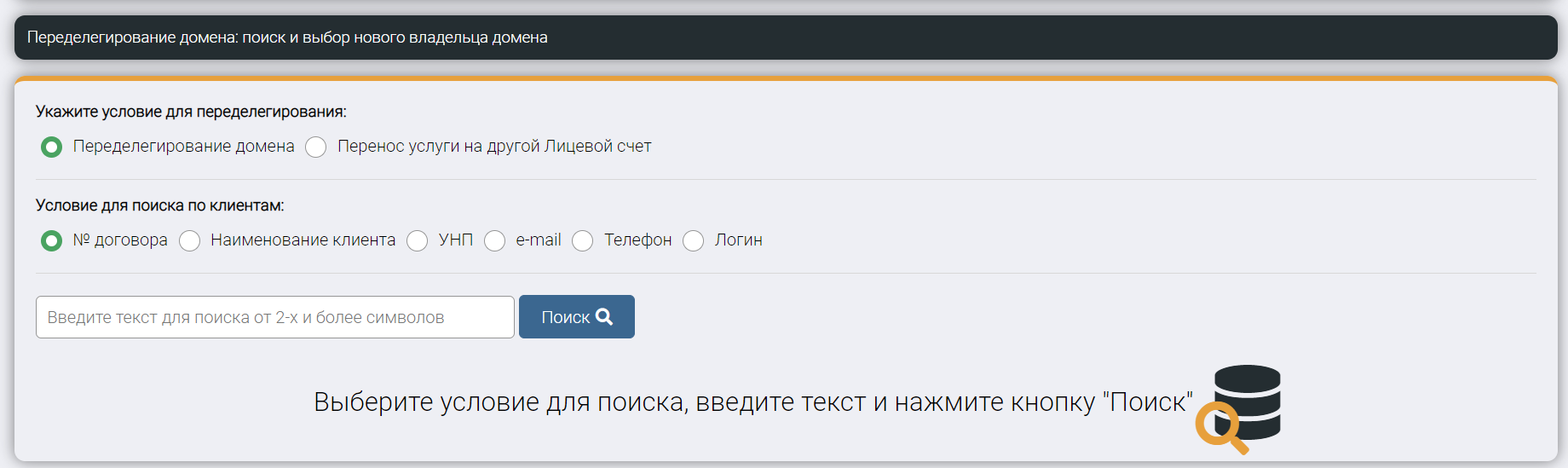


Рисунок 6.19 – Страница поиска пользователя для переделегации домена

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

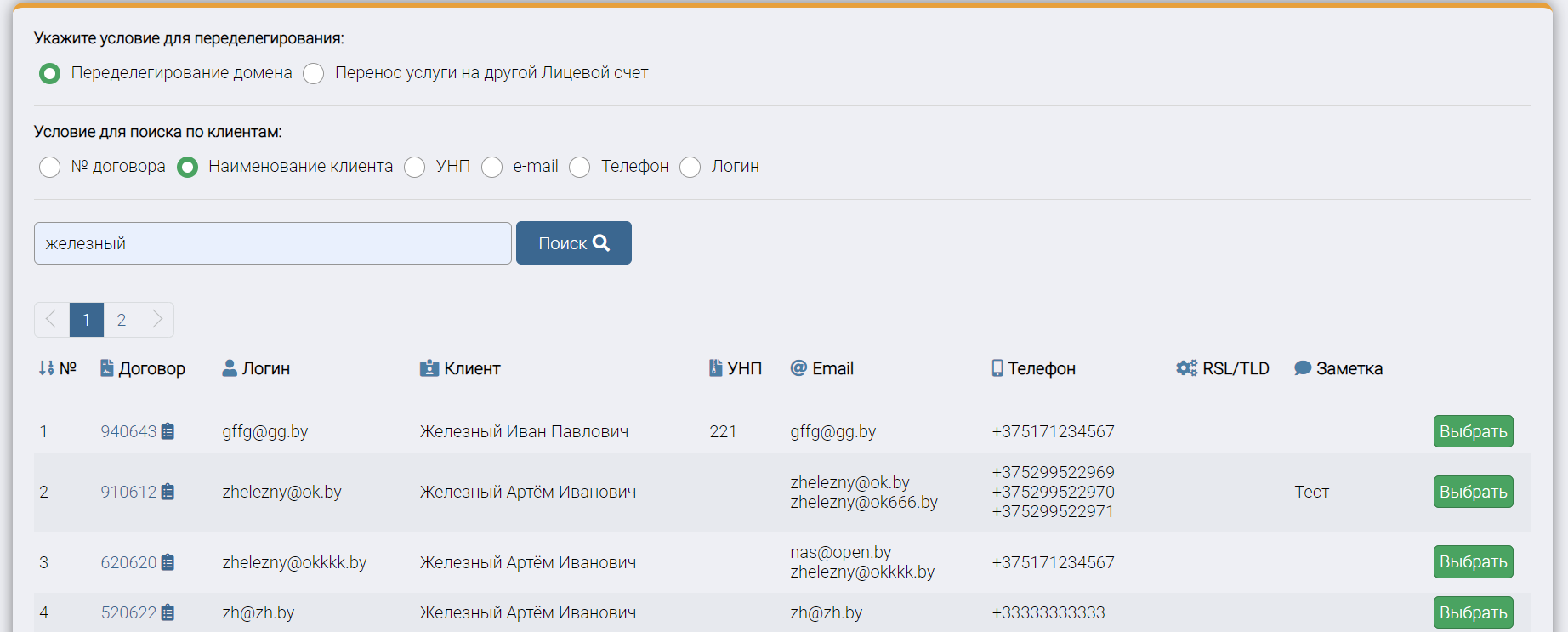


Рисунок 6.20 – Таблица пользователей платформы *Domain.by*

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.

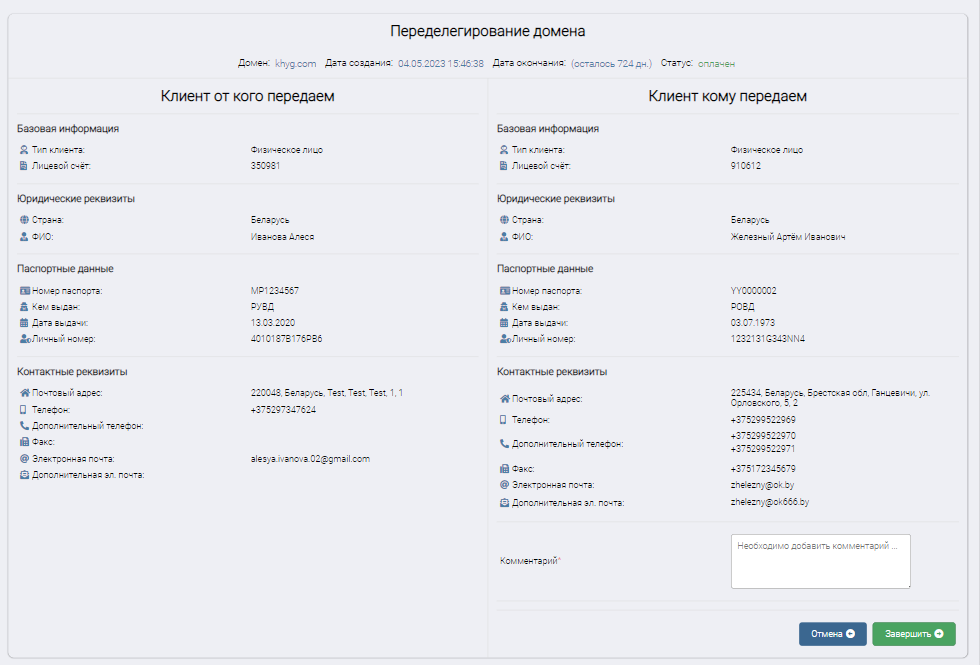


Рисунок 6.21 – Переделегирование пользователю, у которого нет необходимости в регистрации в системе *OpenProvider*

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.

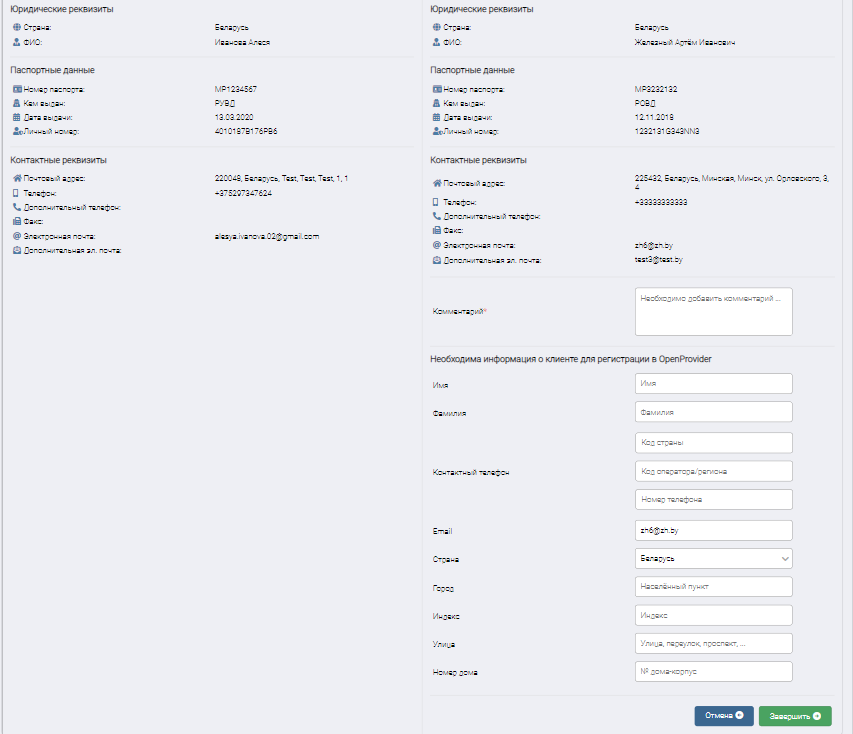


Рисунок 6.22 – страница конечного этапа переделегации с регистрационной формой для *OpenProvider*

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.

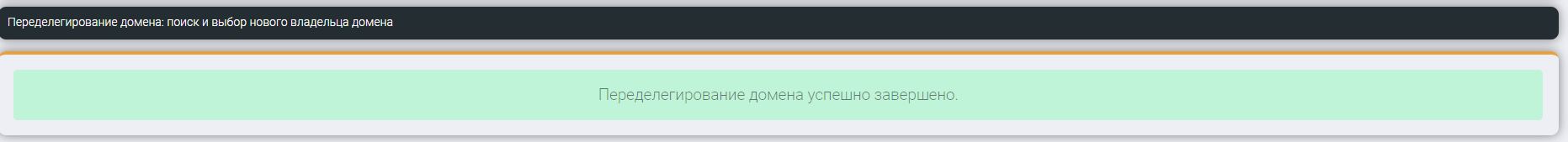


Рисунок 6.23 – Сообщение об успешной переделегации домена

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.

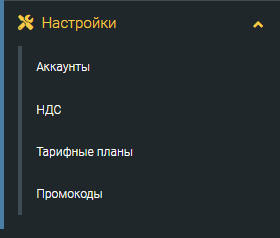


Рисунок 6.24 – Вкладка «Настройки»

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.

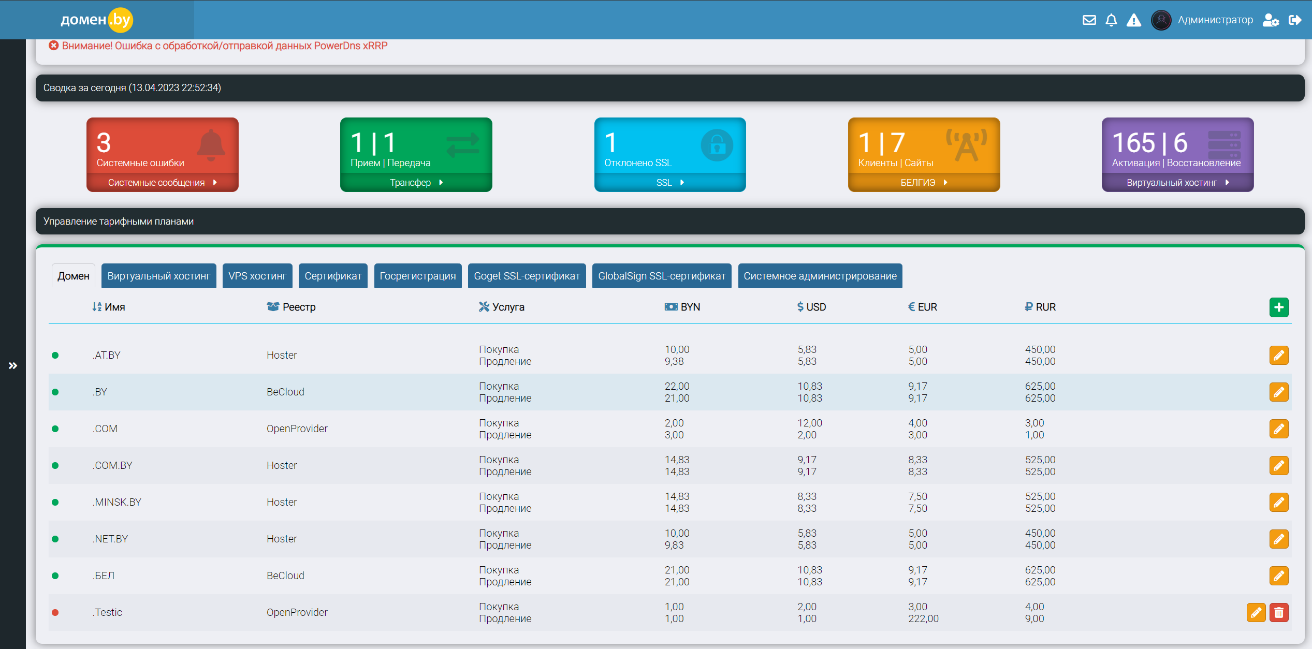


Рисунок 6.25 – Вид страницы «Тарифные планы»

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.

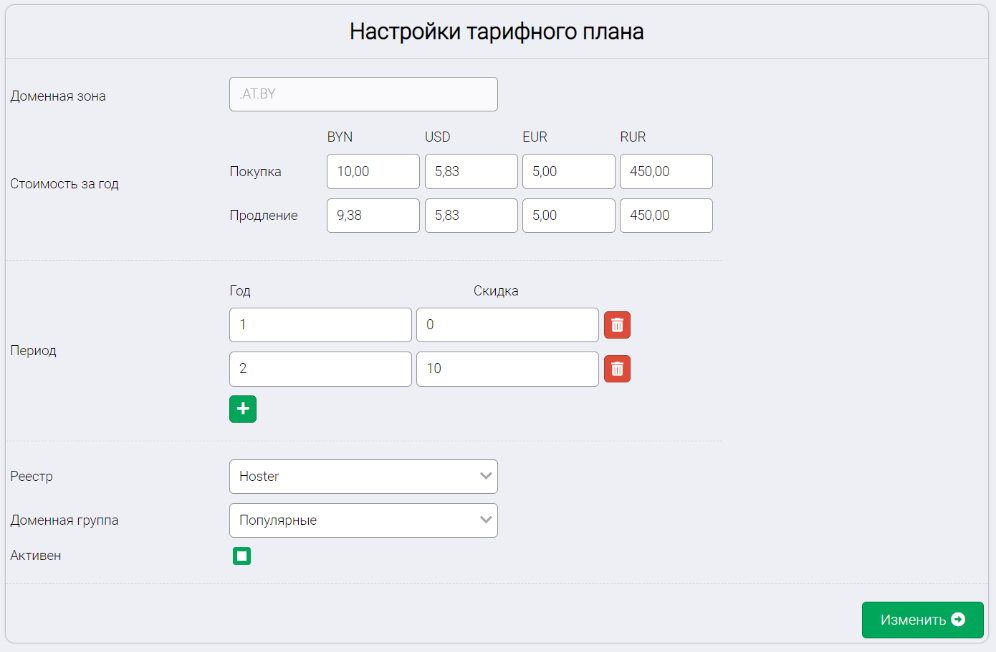


Рисунок 6.26 – Вид окна просмотра и редактирования тарифного плана, который нельзя полностью изменить

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

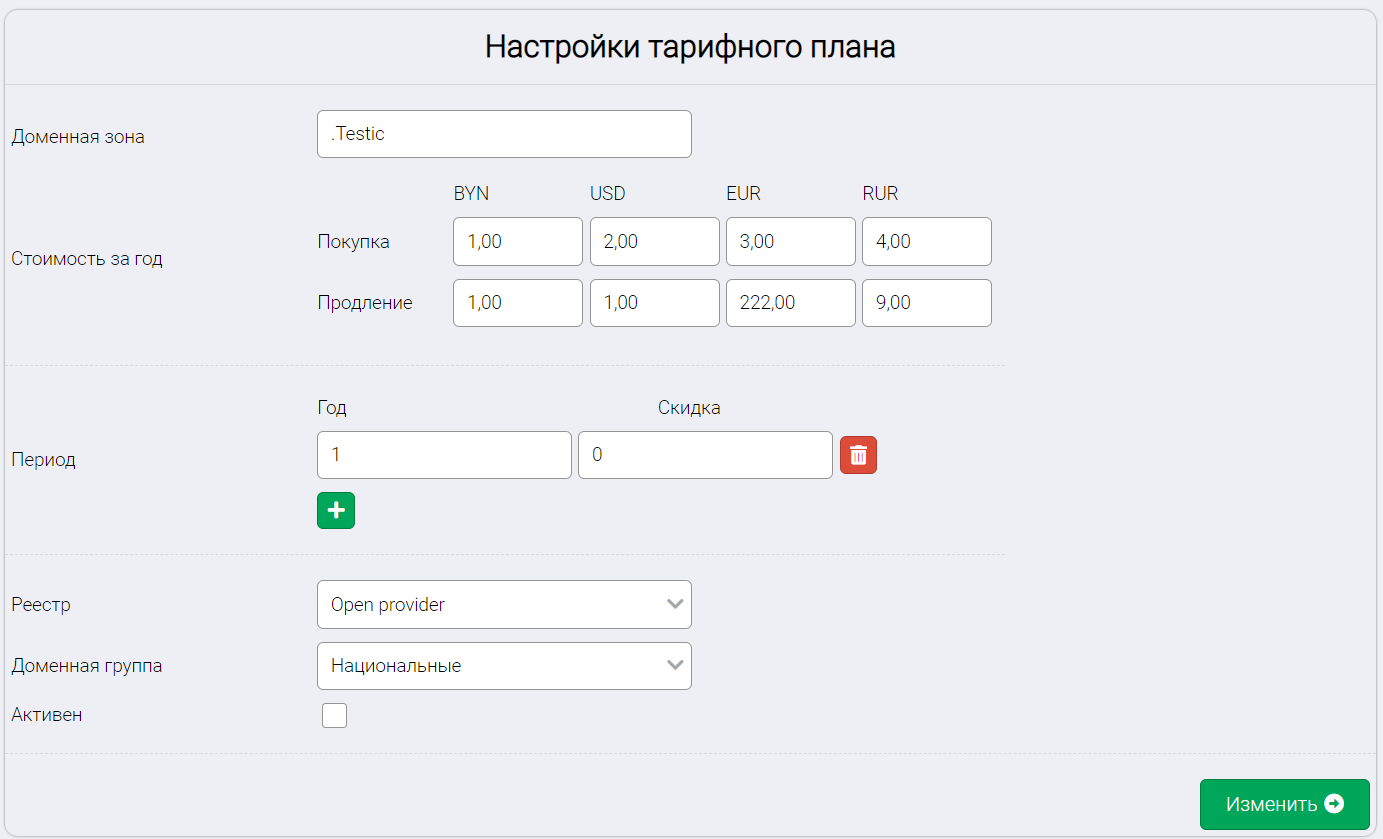


Рисунок 6.27 – Вид окна просмотра и редактирования тарифного плана, который доступен для редактирования

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

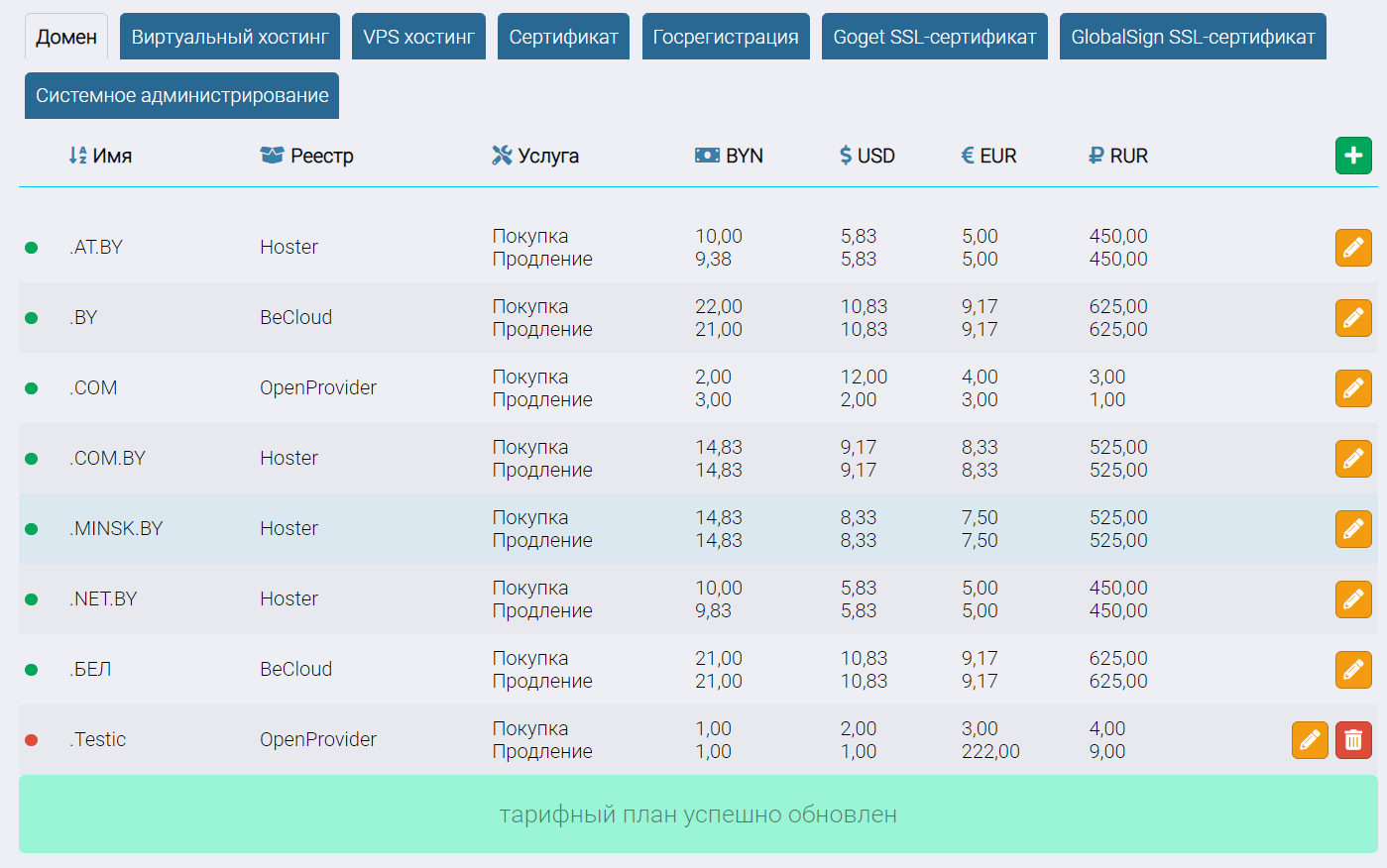


Рисунок 6.28 – Вид страницы «Тарифные планы» после сохранение настроек тарифного плана

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».

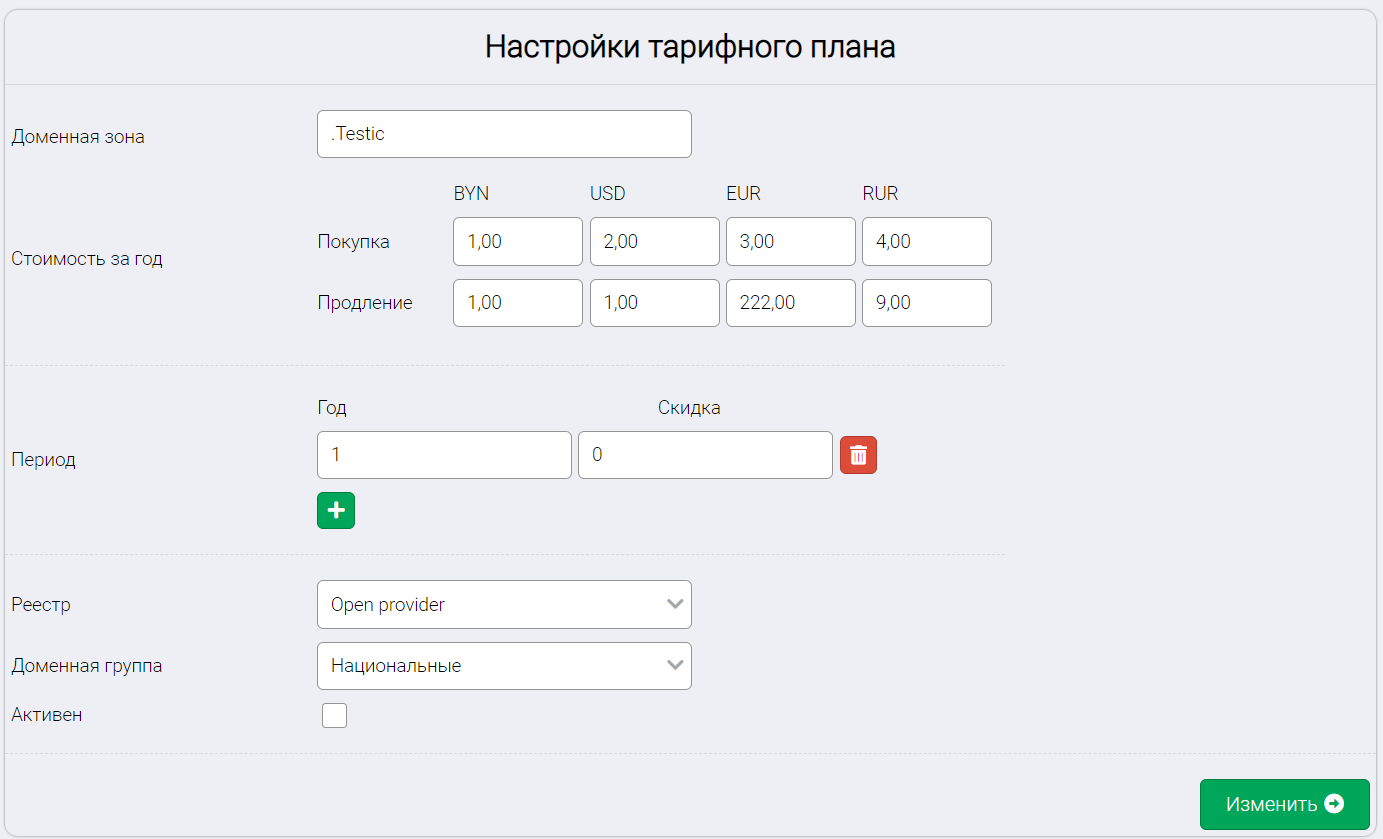


Рисунок 6.29 – Вид окна создания тарифного плана

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.

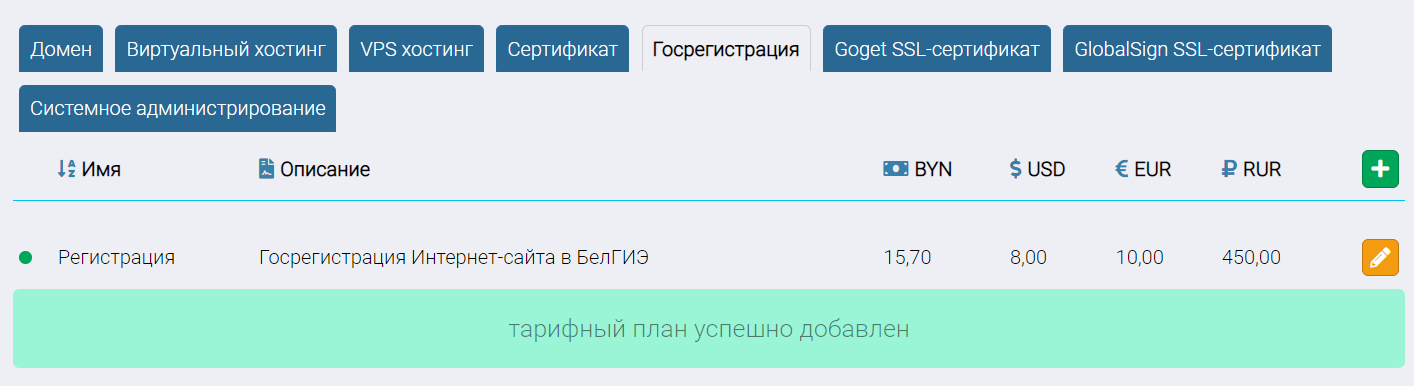


Рисунок 6.30 – Вид страницы «Тарифные планы» после создания тарифного плана

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.

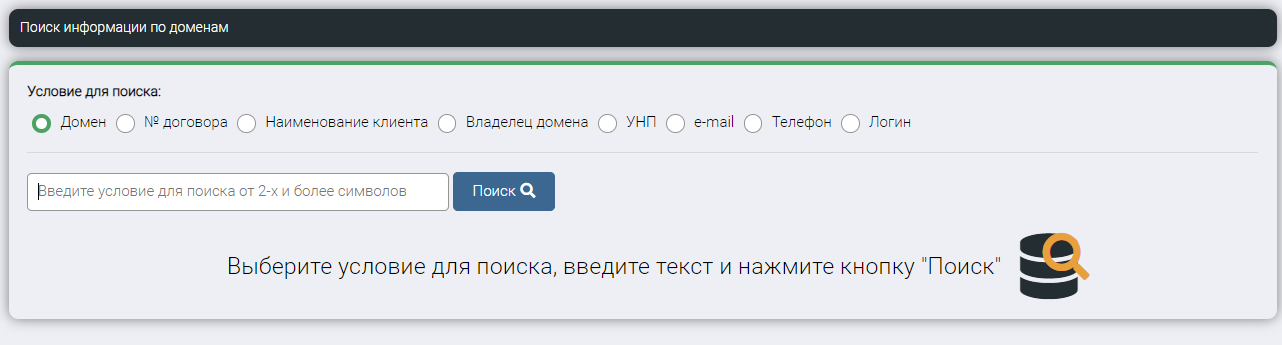


Рисунок 6.31 – Страница поиска информации по доменам

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.

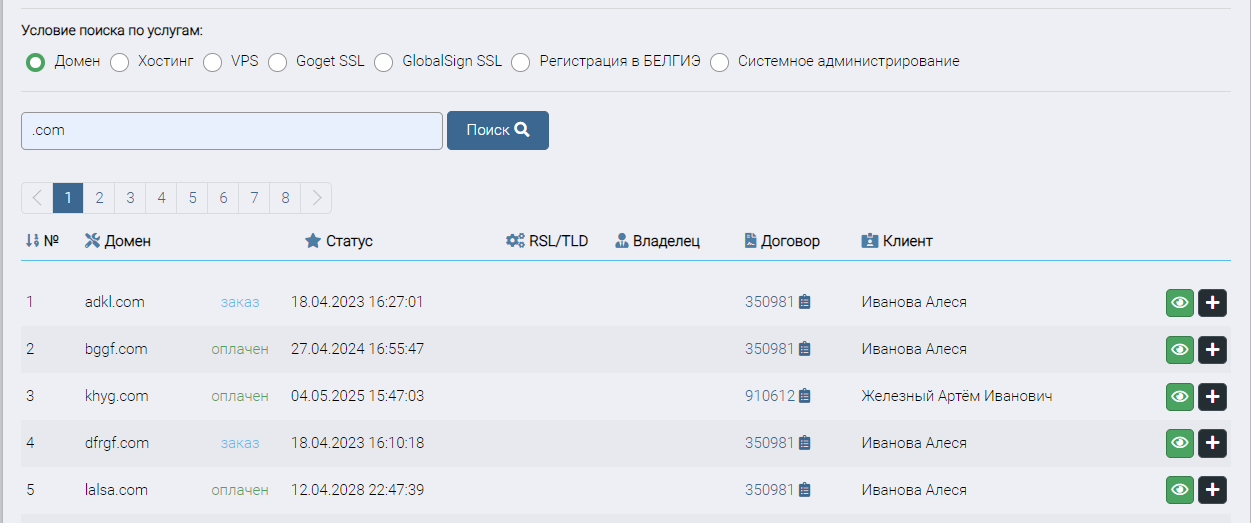


Рисунок 6.33 – Таблица доменов, удовлетворяющих критериям поиска

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

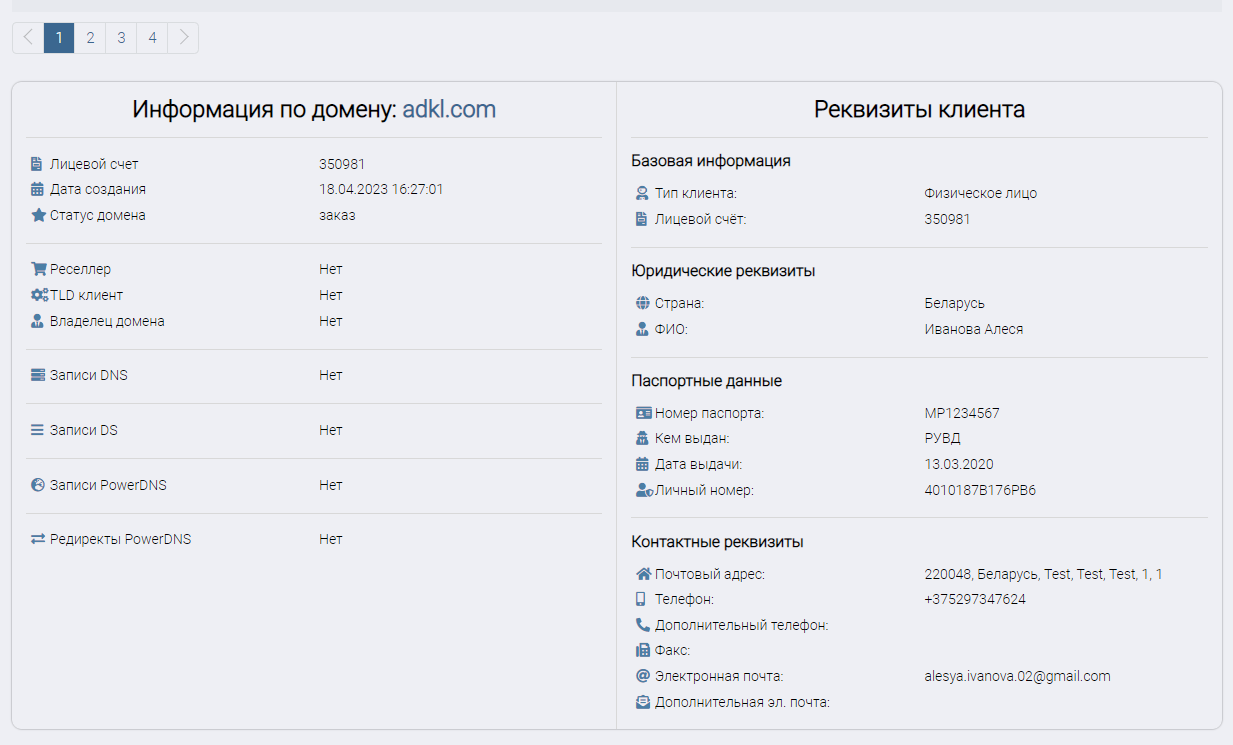


Рисунок 6.35 – Страница показа реквизитов клиента и данных домена

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.

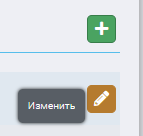


Рисунок 6.36 – Пример подсказки для кнопки, на которой присутствует только иконка

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*.*

*Иванова А.А.*

Провер*.*

*.*

*Блинова Е.А*

Консульт.

*Семёнова Л.С*

Н*.* контр*.*

*Николайчук А.Н.*

Утв*.*

*СмеловВ.В.*

*Лит.*

*Листов*

*8*

11111111, 2023

*7 Технико-экономическое обоснование проекта*

7.1 Общая характеристика разрабатываемого программного средства

При выполнении данного дипломного проект был разработан программный модуль для управления международными доменами, данный модуль предназначен для платформы *Domain.by*. Основной целью разработанного программного модуля является обеспечение пользователей интерфейсом для управления международными доменами, с помощью которого пользователь может зарегистрировать домен в сети Internet, а также при необходимости изменять его *NS* записи или получить сертификат о регистрации домена. Также, администраторы платформы, кроме основных задач пользователя могут осуществлять управление тарифных планов, предлагаемых платформой.

При выполнении данного дипломного проекта использовалась технология *ASP.NET Core 3.0* с фреймворком *MVC*. Также в данный дипломный проект была внедрена сторонняя библиотека для валидации данных *FluentValidation*.

Разработанное программное обеспечение превосходит аналогичные программные модули, которые были рассмотрены в данном дипломном проекте простым и лаконичным дизайном, имеющим достаточно «негативного пространства», простотой использования программного модуля, обеспечением пользователя возможностью не оплачивать услуги сразу при их заказе, возможностью интерфейса предлагать возможные похожие названия доменов для тех, которые доступны для регистрации, наличием выбора нужных доменных зон и предоставлением списка всех доменных зон, которые предлагает платформа для регистрации, возможностью выбора периода для предоставления услуги регистрации домена.

По результатам анализа применяемых продуктами-аналогами стратегий монетизации следует выбрать стратегию монетизации: годовая подписка на использование продукта (пользователь покупает подписку для своего сайта).

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 прочих прямых затрат, % | Нпз | 20 |
| Норматив накладных расходов, % | Нобп,обх | 10 |
| 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 7.1 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 модуля | 10 |
| 3. Разработка серверной части | 80 |
| 4. Разработка клиентской части | 40 |
| 5. Модификация базы данных | 6 |
| 6. Тестирование интерфейса | 16 |
| 7. Тестирование бизнес логики | 20 |
| 8. Написание руководства пользователя | 18 |
| Всего | 196 |

Результат (значение «всего») по данной таблице будет использовано далее для расчётов.

### 7.3.2 Расчёт основной заработной платы

Для определения величины основной заработной платы, было проведено исследование величин заработных плат для специалистов в программирования на .Net. В итоге было установлено, что средняя месячная заработная плата на позиции junior составляет 2 240 рублей, часовая ставка составляет 13,33 руб/час.

Согласно таблице 6.2, проект разрабатывался одним специалистом на протяжении 196 часов. Таким образом, основная заработная плата будет расчитываться 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.

### 7.3.2 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 руб.

### 7.3.3 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.

### 7.3.4 Расчёт суммы прочих прямых затрат

Сумма прочих затрат Спз определяется как произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив прочих затрат в целом по организации Нпз согласно формуле 7.5

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

Спз = 2 613,33 ⋅ 20 / 100 = 522,67 руб.

### 7.3.4 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 7.6

|  |  |  |
| --- | --- | --- |
|  |  | (7.6) |

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

Собп, обх = 2 613,33 ⋅ 10 / 100 = 261,33 руб.

### 7.3.4 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 7.7.

|  |  |  |
| --- | --- | --- |
|  | Ср = Соз + Сдз + Сфсзн + Сбгс + Спз + Собп, обх | (7.7) |

Ср = 2 613,33 + 392,00 +1 021,81 + 12,02 + 522,67 + 261,33 = 4 823,17 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 4 823,17 рублей.

### 7.3.5 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 7.8.

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

*= 4 823,17 ⋅ 7 / 100 = 337,62 руб.*

Все проведенные выше расчеты необходимы для вычисления полной себестоимости проекта.

### 7.3.5 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 7.9

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

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 = 4 823,17 + 337,62 = 5 160,76 руб.

Полная себестоимость составила 5 160,76 рублей.

### 7.3.6 Определение цены, оценка эффективности

Так как монетизация продукта осуществляется путем предоставления услуги продажи доменов. Необходимо определить сумму денежных поступлений и окупаемость затрат на разработку продукта, т.е. целесообразность и эффективность.

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

1. *Hoster.by* – компания, предоставляющая услуги по регистрации хостинга, доменов и *SSL*-сертификатов, а также разработку сайтов. Это компания является провайдером облачных решений и хостинга, который обслуживает белорусские сайты. Данная компания предлагает более сотни доменных зон для регистрации. Обслуживание и управление всеми доменами осуществляется через запросы в техподдержку в компанию.
2. *HostFly* – облачный хостинг-провайдер и регистратор доменных имен. *HostFly* обеспечивает поддержку хостинг-компаний не только на территории Белоруси, но и на територии США. Эта компания занимается регистрацией исключительно доменных имен зоны *.by* и .бел.
3. *ActiveCloud* является поставщиком облачных решений, ИТ-инфраструктуры и хостинга для клиентов в Беларуси и России. Компания работает исключительно для Беларуси и России. *ActiveCloud* предоставляет домены во всех национальных доменных зонах.

По результатам анализа применяемых продуктами-аналогами стратегий монетизации следует выбрать стратегию монетизации: предоставление услуги по регистрации домена на создаваемый нами продукт. 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 7.3.

Таблица 7.3 – 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* |
| *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 7.3 можно сделать вывод что наш продукт выгодно отличается от продуктов наших конкурентов и имеет выгодный баланс возможностей, дизайна и качества кода.

Расчёт прогнозного количества заказов программного средства рассчитывается исходя из среднего значения заказов по каждому аналогу. Количество заказов К аналогов при монетизации методом покупок услуги, скорректированная на оценку показателей качества, рассчитывается по формуле 7.10:

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

где К0 − количество заказов ПС конкурента;

T0 – количество лет существования приложения;

ИР – показатель рассматриваемого программного продукта;

ИК – показатель программного продукта конкурента.

*К1 = (25 000 / 17 \* 8,1) / 7,2 = 1 654,41 (заказов в год),*

*К2 = (25 000 / 18 \* 8,1) / 6,1 = 1 844,26 (заказов в год),*

*К3 = (25 000 / 20 \* 8,1) / 6,5 = 1 557,69 (заказов в год),*

*К = (1 654,41+ 1 844,26 + 1 557,69) / 3 = 1 685,46 ≈ 1 685 (заказов год).*

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:

|  |  |  |
| --- | --- | --- |
|  |  | (7.11) |

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 расчета рыночной цены рассмотренных конкурентов такого решения, если среднее количество заказов 1 685 за год, то денежные поступления от продажи подписки год составят 101 027,09 рублей.

Количество покупателей продукта необходимых для окупаемости Пп вычисляется по формуле 7.12:

|  |  |  |
| --- | --- | --- |
|  |  | (7.12) |

*= 5 160,79 / 59,94 = 86 покупателей.*

7.4 Вывод по разделу

В приложении Ж представлены результаты расчётов для основных показателей данной главы в краткой форме.

Разработка программного средства, осуществляемая одним программистом в течении 168 часов (1 месяц), при заданных условиях обойдется в 5 160,79 руб. Реализации данного программного средства будет приносить годовые денежные поступления от продажи подписки в размере 101 027,09 рублей и окупится при продаже 86 услуг в год. Необходимо написать сколько в среднем установок в год, ориентировочно количество установок 25000, отсюда делаем предположение что проект окупится.

Необходимость разработки модуля, обусловлена все большим ростом популярности услуги продажи международных доменов и ростом конкуренции компаний, предоставляющих эту услугу. Модуль позволяет продать пользователям услугу продажи и управления международных доменов.

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 |