МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Программной инженерии

Специальность 6-05-0612-01 Программная инженерия

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТ:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»

Тема Программное средство «Платформа по поиску репетиторов»

Исполнитель

студент (ка) 2 курса группы 9 Дубина Артём Александрович

(Ф.И.О.)

Руководитель работы асс. Ромыш А.С. (учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Смелов В.В.

(подпись)

Минск 2025

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра программной инженерии

Утверждаю

Заведующий кафедрой ПИ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В.Смелов

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025г.

**ЗАДАНИЕ**

**к курсовому проектированию**

**по дисциплине** "Объектно-ориентированные технологии программирования и стандарты проектирования"

|  |  |
| --- | --- |
| Специальность: 6-05-0612-01 Программная инженерия  Студент: Дубина Артём Александрович | Группа:\_\_9\_\_ |
| **Тема: Программное средство «Поиск репетиторов»** | |

**1. Срок сдачи студентом законченной работы**: "21 мая 2025 г."

**2. Исходные данные к проекту:**

**2.1**. Функционально ПС поддерживает:

* Функции администратора:
* Регистрация и авторизация.
* Управление пользователями (блокировка, изменение данных).
* Модерация рейтингов и отзывов.
* Функции клиента:
  + Регистрация и авторизация.
  + Просмотр личного кабинета.
  + Поиск репетиторов по предметам, рейтингу, цене.
  + Просмотр профиля репетитора.
  + Оставление отзывов и рейтингов.
  + Запись на занятие при помощи заполнения формы.
* Функции клиента-репетитора:
  + Регистрация и авторизация.
  + Личный кабинет с расписанием занятий.
  + Добавление информации о себе (опыт, предметы, стоимость).
  + Просмотр заявок учеников и отклик на них.

**2.2.** При выполнении курсового проекта необходимо использовать принципы проектирования ООП. Приложение разрабатывается под ОС Windows и представляет собой настольное приложение (desktop). Отображение, бизнес логика должны быть максимально независимы друг от друга для возможности расширения. Диаграммы вариантов использования, классов реализации задачи, взаимодействия разработать на основе UML. Язык разработки проекта – C#. Управление программой должно быть интуитивно понятным и удобным. При разработке использовать несколько наиболее подходящих шаблонов проектирования ПО.

**3. Содержание расчетно-пояснительной записки**

* Введение
* Постановка задачи и обзор литературы
* Проектирование архитектуры проекта
* Разработка функциональной модели и модели данных ПС
* Тестирование
* Заключение
* Список используемых источников
* Приложения

**4. Форма представления выполненной курсовой работы:**

* + Теоретическая часть курсового проекта должны быть представлены в формате docx. Оформление записки должно быть согласно выданным правилам.
  + Листинги программы представляются в приложении.
  + Пояснительную записку, листинги, проект (инсталляцию проекта) необходимо загрузить диск, указанный преподавателем.

***Календарный план***

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 12.03.2025 |  |
| 2 | Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования | 19.03.2025 |  |
| 3 | Анализ и проектирование архитектуры приложения (построение диаграмм, проектирование бизнес-слоя, представления и данных) | 31.03.2025 |  |
| 4 | Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса | 09.04.2025 |  |
| 5 | Кодирование программного средства | 23.04.2025 |  |
| 6 | Тестирования и отладка программного средства | 07.05.2025 |  |
| 7 | Оформление пояснительной записки | 14.05.2025 |  |
| 9 | Защита проекта | 21.05.2025 |  |

**5. Дата выдачи задания \_\_\_\_**05.03.2025**\_\_\_\_**

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *асс. Ромыш А.С.*

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Дубина А.А.*

(дата и подпись студента)

**Оглавление**

[Введение 5](#_Toc198744561)

[1. Аналитический обзор прототипов и литературных источников 6](#_Toc198744562)

[1.1 Анализ прототипов 6](#_Toc198744563)

[1.1.1 Интернет-ресурс «profi-bel.by» 6](#_Toc198744564)

[1.1.2. Интернет-ресурс «repetit-center.by» 7](#_Toc198744565)

[1.1.3. Интернет-ресурс «repetit.ru» 8](#_Toc198744566)

[1.1.4. Интернет-ресурс «protutor.ru» 9](#_Toc198744567)

[1.2 Постановка задачи 10](#_Toc198744568)

[2. Анализ требований к программному средству и разработка функциональных требований 11](#_Toc198744569)

[2.1 Определение требований к программному средству 11](#_Toc198744570)

[2.2 Описание средств разработки 11](#_Toc198744571)

[2.3 Описание функциональности программного средства 12](#_Toc198744572)

[2.4 Спецификация функциональных требований 12](#_Toc198744573)

[3. Проектирование программного средства 14](#_Toc198744574)

[3.1 Архитектура проектируемого программного средства 14](#_Toc198744575)

[3.2 Проектирование структуры базы данных 15](#_Toc198744576)

[4 Реализация программного средства 17](#_Toc198744577)

[4.1 Реализация паттерна MVVM 17](#_Toc198744578)

[4.2 Классы для работы с базой данных 18](#_Toc198744579)

[5 Тестирование, проверка работоспособности приложения и анализ результатов 19](#_Toc198744580)

[5.1 Тестирование регистрации и авторизации 19](#_Toc198744581)

[5.2 Тестирование всей работы приложение 21](#_Toc198744582)

[6 Руководство по использованию 23](#_Toc198744583)

[6.1 Регистрация и авторизация 23](#_Toc198744584)

[6.2 Пользовательская часть 24](#_Toc198744585)

[6.3 Администрирование 27](#_Toc198744586)

[ЗАКЛЮЧЕНИЕ 29](#_Toc198744587)

[Список использованных источников 30](#_Toc198744588)

[ПРИЛОЖЕНИЯ А 31](#_Toc198744589)

[ПРИЛОЖЕНИЕ Б 32](#_Toc198744590)

[ПРИЛОЖЕНИЕ В 33](#_Toc198744591)

# **Введение**

В современном мире цифровые технологии активно проникают во все сферы жизни, включая образование. Одним из ключевых аспектов обучения является поиск квалифицированных репетиторов, что зачастую становится сложной задачей как для учеников, так и для преподавателей. Решение этой проблемы требует автоматизации процесса подбора репетиторов с учётом индивидуальных потребностей пользователей.

Данный курсовой проект направлен на создание программного средства для поиска репетиторов, обеспечивающего удобный интерфейс, возможность фильтрации по различным критериям (предмет, стоимость, рейтинг) и обратную связь между пользователями. Приложение позволит студентам и школьникам быстрее находить подходящих преподавателей, а репетиторам — расширять клиентскую базу.

С помощью этого программного средства пользователи смогут находить квалифицированных репетиторов, выбирать удобное для них время занятий, а также фильтровать преподавателей по предмету, стоимости и рейтингу. Будет возможность онлайн-записи на занятия, чтобы пользователь мог связаться с репетитором и договориться о проведении уроков в удобном формате.

Для успешной реализации курсового проекта необходимо:

* провести анализ соответствующей литературы;
* ознакомиться с прототипами аналогов по данной теме;
* определить функциональные требования;
* продумать структуру базы данных;
* продумать структуру проекта;
* реализовать программное средство;
* протестировать программное средство;
* написать руководство пользователя.

В процессе выполнения курсового проекта будут применены принципы и подходы объектно-ориентированного программирования (ООП), а также будет использована база данных MS SQL Server и технология Windows Presentation Foundation (WPF).

# **Аналитический обзор прототипов и литературных источников**

## **Анализ прототипов**

Курсовой проект представляет из себя разработку приложения для поиска репетиторов. В этой сфере существует множество аналогичных решений, которые предоставляют пользователям возможность выбирать и записываться на занятия. В ходе выполнения курсового проекта были проанализированы цели и задачи, а также рассмотрены аналогичные примеры их решений. На основании анализа всех достоинств и недостатков данных альтернативных решений были сформулированы требования к данному программному средству.

## **Интернет-ресурс «profi-bel.by»**

Сайт «profi-bel.by» [1] представляет поиск специалистов по разным областям в Минске. Интерфейс у сайта удобный и интуитивно понятный. Как минус можно выделить сильную нагрузку на пользователя из-за огромного спектра услуг. Интерфейс сайта представлен на рисунке 1.1.

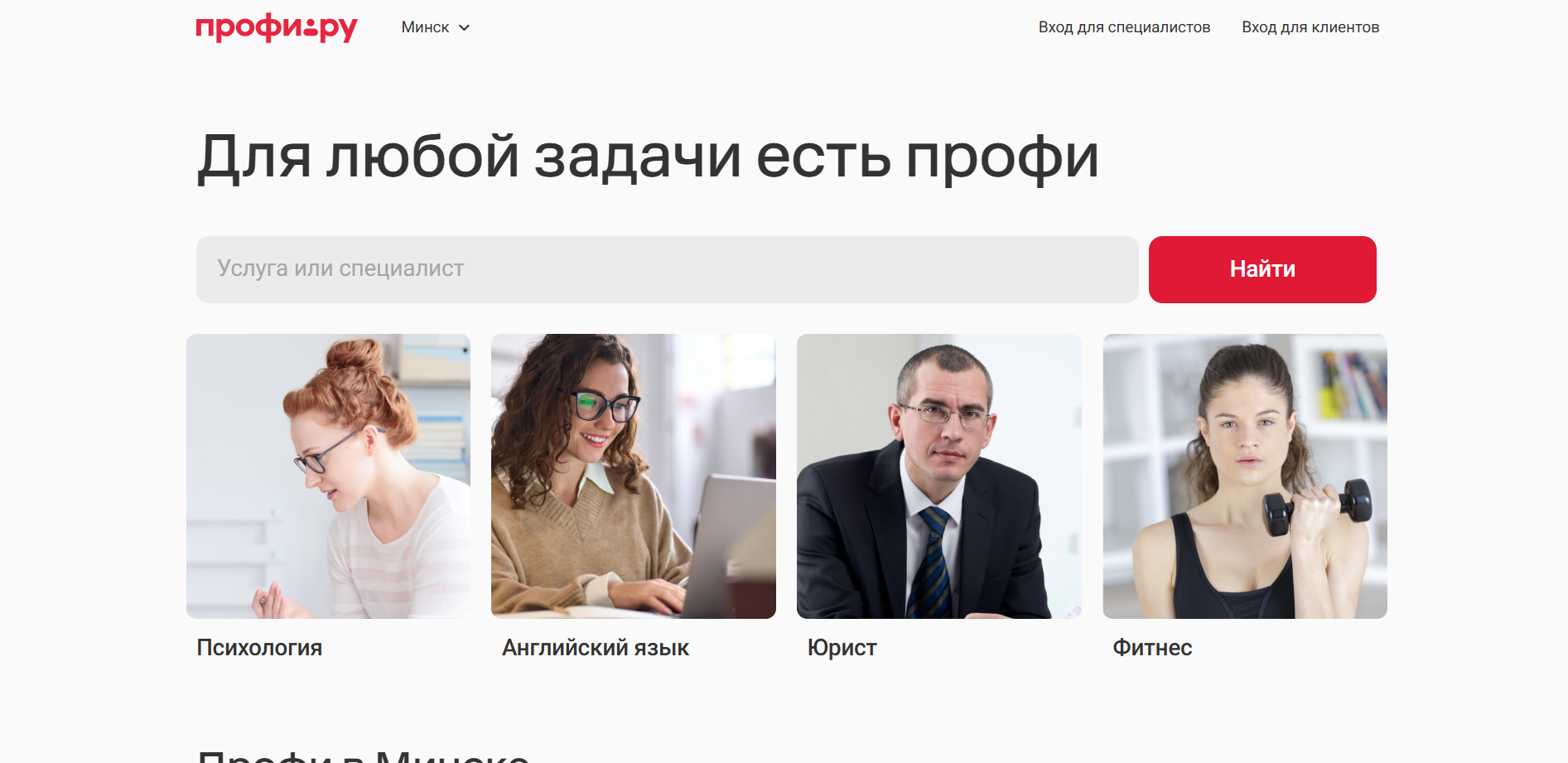


Рисунок 1.1 – Интерфейс сайта «profi-bel.by»

На этом сайте пользователь имеет возможность найти специалистов в различных областях, включая репетиторов, мастеров по ремонту, тренеров и других профессионалов. Пользователи могут ознакомиться с профилями специалистов, содержащими информацию об их опыте, услугах и контактных данных. Сайт также предлагает систему отзывов, позволяющую оценить качество работы специалистов на основе мнений других клиентов. ​

Главной целью сайта является соединение клиентов с профессионалами, предоставляя удобный инструмент для поиска и выбора подходящего специалиста. Пользователь может оставить заявку, указав свои требования и предпочтения, после чего специалисты откликаются на нее, предлагая свои услуги. Однако процесс записи на услуги осуществляется через прямой контакт со специалистом, и на сайте отсутствует возможность онлайн-бронирования или оплаты услуг, что может быть неудобством для некоторых пользователей. ​

Положительной стороной сайта является наличие системы отзывов, позволяющей клиентам делиться своим опытом взаимодействия со специалистами. Однако пользователи не могут оставлять отзывы или задавать вопросы непосредственно на сайте без предварительного взаимодействия со специалистом через платформу. ​

Сайт предоставляет информацию о местоположении специалистов и их контактные данные, что облегчает процесс связи.

В целом, сайт выполняет свою основную функцию по предоставлению информации о специалистах и их услугах. Однако отсутствие возможности онлайн-бронирования и ограниченные функции взаимодействия на сайте могут быть восприняты как недостатки для пользователей, ожидающих более полного онлайн-сервиса.

## **1.1.2. Интернет-ресурс «repetit-center.by»**

​ Сайт repetit-center.by [2] предназначен для подбора репетиторов по различным предметам и направлениям. Пользователи могут изучить список преподавателей, ознакомиться с их опытом, квалификацией и специализацией. Каждый репетитор имеет личную анкету с указанием цен на занятия и контактной информации, что упрощает процесс выбора подходящего специалиста. Интерфейс сайта представлена на рисунке 1.2.

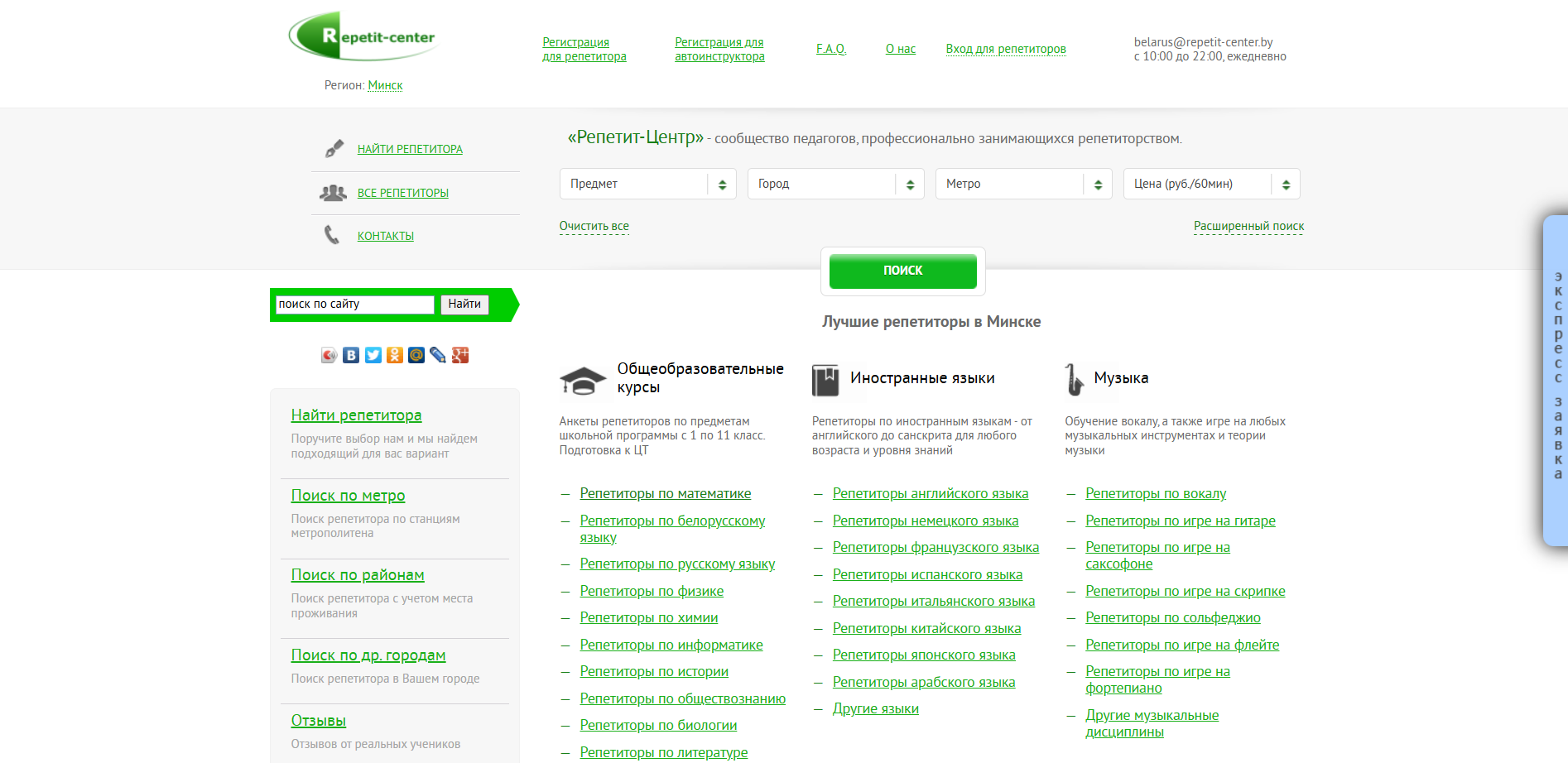


Рисунок 1.2 – Интерфейс сайта «repetit-center.by»

Основная функция сайта – помочь ученикам найти подходящего репетитора, связаться с ним и договориться о занятиях. Однако бронирование уроков и оплата происходят вне платформы, что может быть не всегда удобно. Пользователь оставляет заявку с указанием своих требований, после чего преподаватели могут связаться с ним.

Одним из преимуществ сайта является наличие отзывов о репетиторах, что помогает оценить их компетентность перед началом занятий. Однако функционал платформы не позволяет оставлять комментарии напрямую без предварительного взаимодействия с преподавателем.

Также на сайте доступна информация о местоположении преподавателей, их контактные данные, ссылки на социальные сети и способы связи. Это облегчает коммуникацию, но отсутствие онлайн-записи и личного кабинета для учеников является минусом, поскольку пользователям приходится управлять расписанием самостоятельно.

В целом, repetit-center.by – полезный ресурс для поиска репетиторов, но платформа могла бы быть более удобной, если бы включала дополнительные функции, такие как система бронирования и онлайн-оплата.

**1.1.3. Интернет-ресурс «repetit.ru»**

Основная функция сайта repetit.ru [3] – помочь ученикам найти подходящего репетитора, связаться с ним и договориться о занятиях. Платформа предоставляет удобный поиск репетиторов по различным предметам и специализациям, что позволяет быстро выбрать подходящего кандидата.

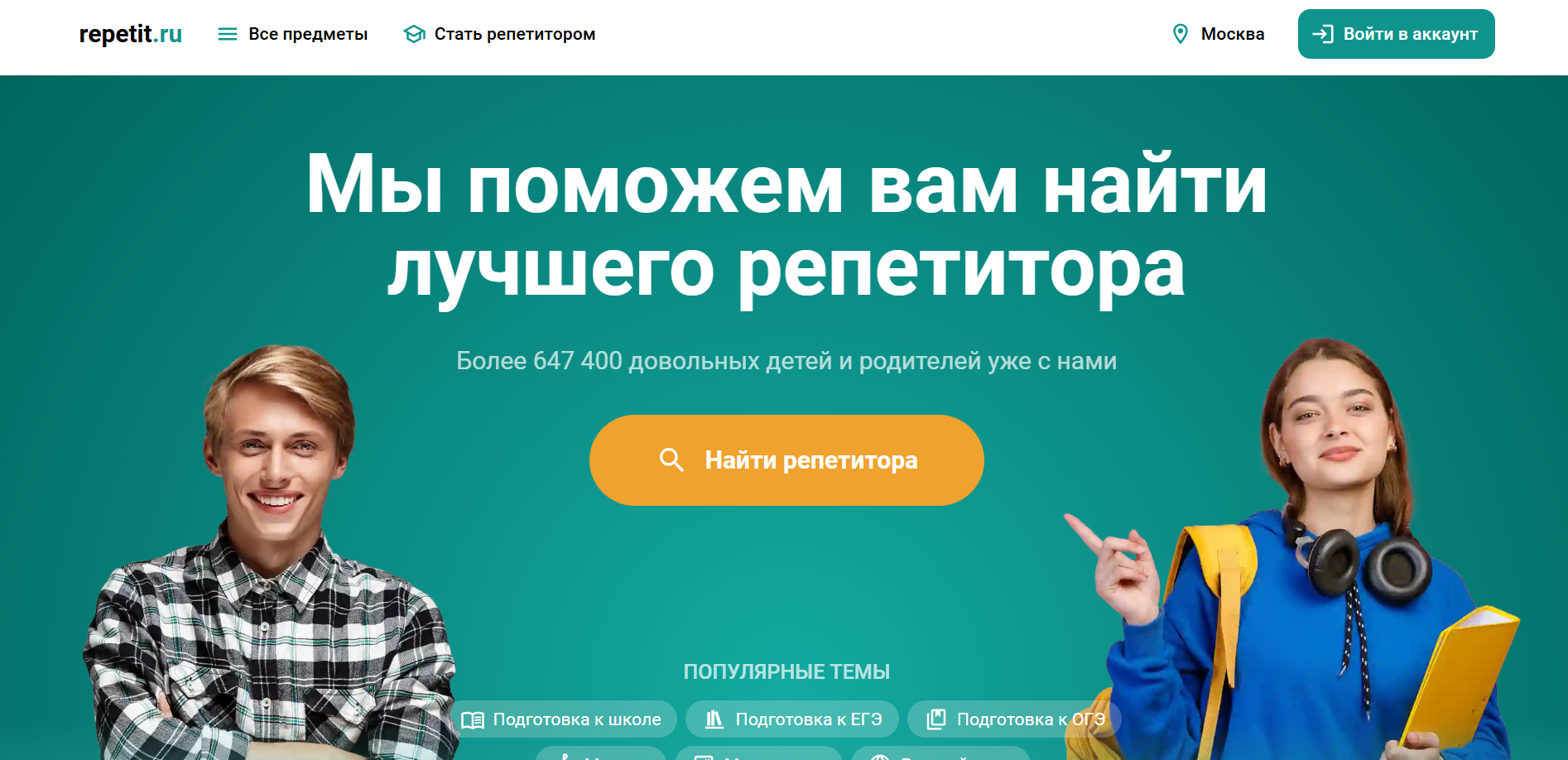


Рисунок 1.3 – Интерфейс сайта «repetit.ru»

Как и в случае с repetit-center.by, бронирование уроков и оплата происходят вне сайта, что может быть не всегда удобно для пользователей, так как процесс взаимодействия с преподавателем не интегрирован в саму платформу. Одним из ключевых преимуществ сайта является наличие отзывов о репетиторах, что помогает сформировать мнение о компетентности преподавателей. Отзывы могут быть полезны для учеников при принятии решения о выборе репетитора. Однако, как и на предыдущем сайте, функционал платформы не позволяет оставлять комментарии без предварительного взаимодействия с преподавателем, что ограничивает возможности для более детальной обратной связи.

Сайт предоставляет информацию о местоположении репетиторов, их контактные данные, а также ссылки на социальные сети, что помогает установить контакт. Однако отсутствие возможности онлайн-бронирования и личного кабинета для учеников ограничивает функциональность платформы, что усложняет управление расписанием и взаимодействие с преподавателем.

В целом, repetit.ru – полезный ресурс для поиска репетиторов, но платформа могла бы быть более удобной и эффективной, если бы добавила систему онлайн-бронирования и оплати, а также личный кабинет для учеников для удобного управления процессом обучения.

## **1.1.4. Интернет-ресурс «protutor.ru»**

Сайт protutor.ru [4] представляет собой яркий и многофункциональный сайт по поиску репетиторов онлайн. Интерфейс данного сайта представлен на рисунке 1.4.

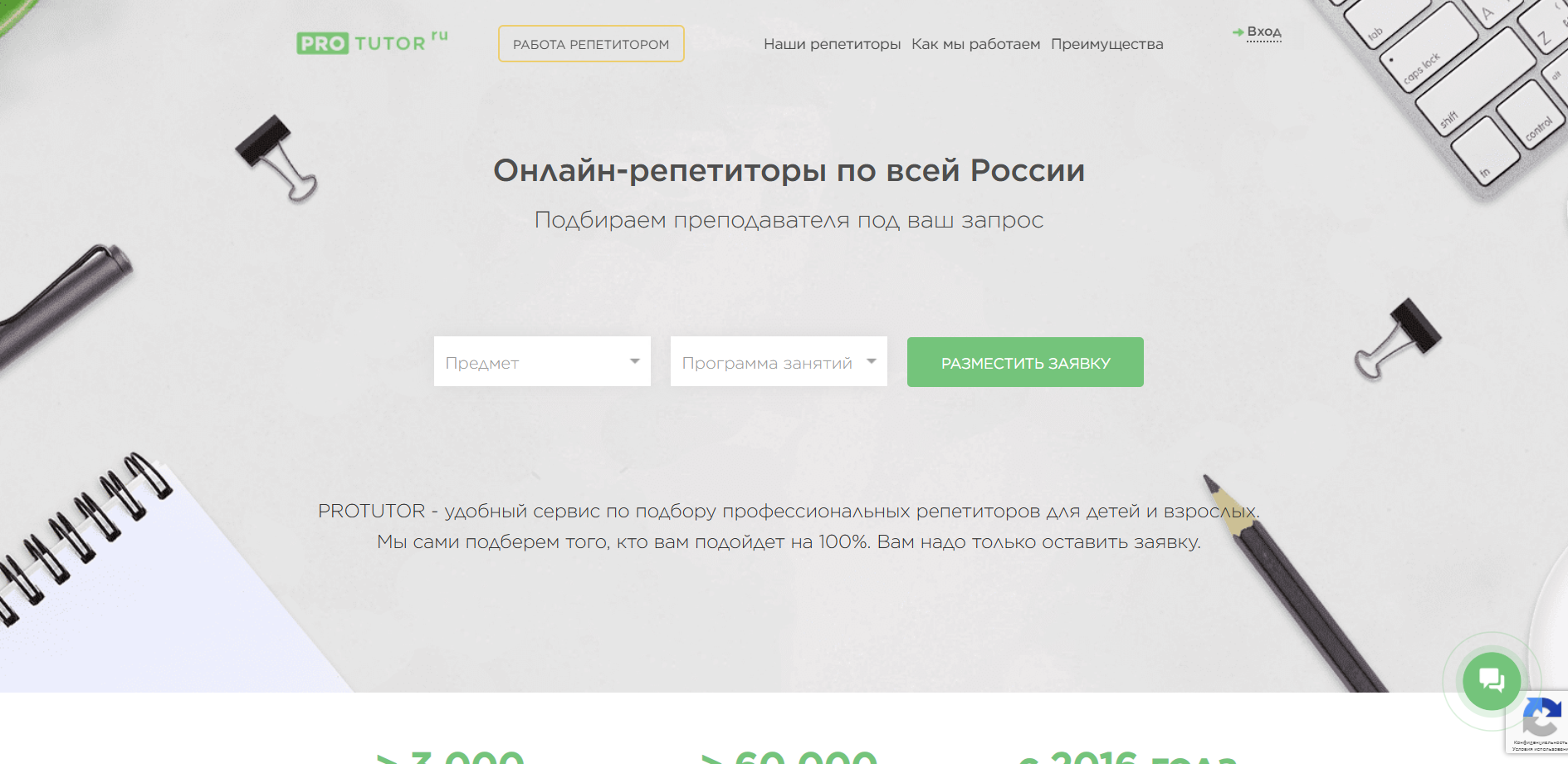


Рисунок 1.4 – Интерфейс сайта «protutor.ru»

Сайт protutor.ru ориентирован на помощь ученикам в поиске репетиторов и установлении контакта с ними. Платформа предлагает широкий выбор преподавателей по множеству предметов и специализаций, что является значительным плюсом для пользователей. Также сайт предлагает возможность фильтрации репетиторов по различным критериям, таким как стоимость, опыт работы и место проживания, что упрощает поиск подходящего кандидата.

Одним из главных преимуществ является наличие системы рейтингов и отзывов, которая помогает пользователям оценить качество работы репетитора. Однако стоит отметить, что отзывы можно оставить только после первых занятий, что ограничивает возможность получения информации о преподавателе до начала курса.

На сайте также предусмотрена возможность связи с преподавателями через личные сообщения, но отсутствие системы онлайн-бронирования и оплаты может создать неудобства для пользователей, так как все эти процессы приходится организовывать вне платформы.

Кроме того, сайт предоставляет полезную информацию о преподавателях, включая их профессиональное образование, сертификаты и опыт, что способствует более осознанному выбору репетитора. Однако, несмотря на наличие этих данных, отсутствие функционала для создания личных кабинетов для учеников и преподавателей затрудняет управление процессом обучения.

В общем, protutor.ru является хорошим ресурсом для поиска репетиторов с разнообразными специальностями и гибкими возможностями фильтрации, но добавление функций онлайн-бронирования и личных кабинетов сделало бы платформу более удобной и функциональной для пользователей.

## **1.2 Постановка задачи**

После анализа существующих прототипов и аналогичных решений были определены ключевые задачи, которые должно решать разрабатываемое программное обеспечение. Одной из центральных задач стало создание интуитивно понятного, функционального и визуально привлекательного пользовательского интерфейса. Именно через интерфейс осуществляется основное взаимодействие пользователя с системой, поэтому он играет важнейшую роль в восприятии программы.

При проектировании интерфейса особое внимание уделяется удобству использования, логичной структуре, минимальному количеству действий для выполнения повседневных операций и визуальной четкости. Интерфейс ориентирован на широкую аудиторию — как на опытных, так и на начинающих пользователей. Поэтому его элементы — кнопки, меню, поля ввода — размещаются в соответствии с привычной логикой поведения пользователя. Используются стандартные элементы управления, знакомые большинству, что ускоряет освоение и повышает доверие к системе. Визуальное оформление выдержано в едином стиле, что делает интерфейс целостным и понятным.

Таким образом, продуманный пользовательский интерфейс значительно повышает удобство и эффективность работы с программой, способствует более быстрому освоению и обеспечивает положительное восприятие всего программного продукта. Качественный интерфейс не только облегчает доступ к функционалу, но и отражает внутреннюю логику системы, обеспечивая её целостность и понятность.

1. **Анализ требований к программному средству и разработка функциональных требований**

Для того чтобы приступить к этапу разработки проекта, необходимо четко сформулировать функциональные требования к программному средству.

Анализ требований — это ключевой этап разработки программного обеспечения, направленный на выявление и формализацию потребностей пользователей и заказчиков. В ходе этого этапа осуществляется сбор, уточнение, документирование и систематизация требований, что позволяет минимизировать риск недопонимания и ошибок на последующих стадиях разработки.

* 1. **Определение требований к программному средству**

Функционально ПС должно выполнять следующие задачи:

Функции администратора:

* регистрация и авторизация.
* управление пользователями (блокировка, изменение данных).
* модерация рейтингов и отзывов.

Функции клиента:

* регистрация и авторизация.
* просмотр личного кабинета.
* поиск репетиторов по предметам, рейтингу, цене.
* просмотр профиля репетитора.
* оставление отзывов и рейтингов.
* запись на занятие при помощи заполнения формы.

Функции клиента-репетитора:

* регистрация и авторизация.
* личный кабинет с расписанием занятий.
* добавление информации о себе (опыт, предметы, стоимость).
* просмотр заявок учеников и отклик на них.
  1. **Описание средств разработки**

При разработке приложения были использованы:

* интегрированная среда разработки Microsoft Visual Studio 2022;
* программная платформа .NET Framework;
* язык программирования C#;
* расширяемый язык разметки XAML;
* технология WPF;
* технология ADO.NET;
* Microsoft SQL Server.

В качестве интерфейса прикладного программирования был выбран обширный API-интерфейс – Windows Presentation Foundation (WPF), предназначенный для создания настольных программ с графически насыщенным пользовательским интерфейсом. В основе графической технологии WPF лежит мощная инфраструктура, основанная на DirectX. Это является одним из основных отличий WPF от более ранней технологии создания пользовательских интерфейсов – Windows Forms.

Для работы с WPF использовался объектно-ориентированный язык программирования с С-подобным синтаксисом – С#, разработанный для создания приложений на платформе Microsoft .NET Framework.

В C# существует несколько подходов к работе с базами данных. Поскольку моя база данных является сравнительно небольшой, я выбрал технологию **ADO.NET**. ADO.NET предоставляет доступ к данным через SQL-запросы и соединения, обеспечивая при этом высокую производительность и прямой контроль над операциями с базой данных.

* 1. **Описание функциональности программного средства**

Описание функциональности программного средства представлено с помощью UML-диаграммы вариантов использования.

Диаграмма вариантов использования (Use Case Diagram) — это один из ключевых инструментов UML, предназначенный для визуального представления функциональности системы с точки зрения её взаимодействия с внешними участниками. Она позволяет определить, какие задачи должна выполнять система, кто с ней взаимодействует и в каких формах происходит это взаимодействие.

На Use Case диаграммах изображаются **актеры** — внешние пользователи или другие системы, представленные в виде фигурок человека, и **варианты использования** — основные функции системы, обозначаемые овалами. Связи между актёрами и вариантами использования показывают, как именно осуществляется взаимодействие между участниками и системой.

Главное преимущество таких диаграмм — наглядность. Они помогают структурировать требования на этапе проектирования, делают функциональность системы понятной для всех заинтересованных сторон, включая заказчиков, аналитиков и разработчиков. Также Use Case диаграммы облегчают переход от анализа к технической реализации.

Диаграмма вариантов использования представлена в Приложении А.

* 1. **Спецификация функциональных требований**

Для функциональности ПС необходимо создание базы данных для хранения информации приложения. Подробно о базе данных описано в главе 3.

В программном средстве «Приложение по поиску репетиторов» при запуске необходимо реализовать регистрацию и авторизацию пользователей для дальнейшего использования приложения. Для авторизации входными параметрами являются электронная почта и пароль пользователя, которые содержатся в базе данных. Для того чтобы зарегистрироваться необходимо ввести имя, электронную почту, пароль. Введенные данные, успешно прошедшие валидацию, заносятся в базу данных, а пользователь получает доступ к приложению.

При авторизации администратор может перейти в панель администратора и блокировать пользователей, изменять их данные при нажатии на соответствующие кнопки действий. Также он может модерировать отзывы и рейтинги, удаляя нежелательный контент или редактируя записи при необходимости. Ещё администратор может написать каждому из пользователей в окне сообщений.

При авторизации клиент может на странице «Главная» осуществлять поиск репетиторов по предмету, цене и рейтингу. На странице «Профиль репетитора» просматривать подробную информацию о преподавателе и перейти к записи на занятие. На странице «Запись на занятие» клиент может выбрать дату и время, оставить сообщение и отправить заявку. После проведения занятия пользователь может оставить отзыв и рейтинг. В «Личном кабинете» клиент просматривает свои заявки, отклики и может редактировать личную информацию. В окне «Сообщения» пользователь может общаться с репетиторами, с которыми у него есть заявки.

При авторизации репетитор может на странице «Личный кабинет» добавлять и редактировать информацию о себе — такие как предметы преподавания, опыт, стоимость. На странице «Расписание» преподаватель может просматривать входящие заявки от учеников и откликаться на них. В окне «Сообщения» репетитор может общаться с пользователями, с которыми у него есть заявки.

# **Проектирование программного средства**

Проектирование программного обеспечения представляет собой ключевой этап разработки, на котором формируется архитектура и структура будущей системы с учётом требований, ограничений и возможных рисков. На этом этапе принимаются решения о выборе архитектурных подходов (например, MVC, MVVM), технологий, баз данных и способах взаимодействия между компонентами системы.

Особое внимание уделяется обеспечению функциональности, производительности, безопасности и удобства использования. В рамках проектирования также разрабатываются прототипы и макеты пользовательского интерфейса, что позволяет визуализировать предполагаемое поведение системы и получить раннюю обратную связь от пользователей.

Таким образом, проектирование программного обеспечения служит фундаментом для дальнейшей разработки, обеспечивая соответствие будущего продукта ожиданиям заказчиков и пользователей, а также повышая шансы на успешную реализацию и внедрение системы.

* 1. **Архитектура проектируемого программного средства**

При создании программного средства использовался архитектурный паттерн Model-View-ViewModel (MVVM). Он был представлен архитектором программного обеспечения компании Microsoft Джоном Госсманом в 2005 году. Одной из главных целей шаблона является отделение и абстрагирование видимого пользовательского интерфейса от логики представления. MVVM состоит из трех компонентов: модель (Model), представление (View) и модель представления (ViewModel). Наглядное представление паттерна представлено на рисунке 3.1.

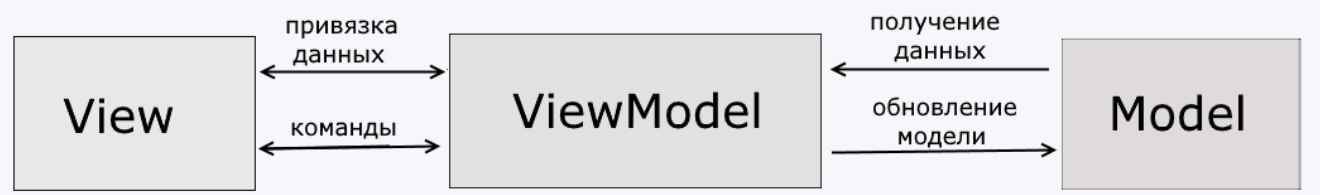


Рисунок 3.1 − Наглядное представление MVVM

Модель (Model) — это субъект, который описывает используемые в приложении данные. Это может быть что угодно, будь то обычный продукт или же сложный рецепт. Представление (View) — это графический элемент управления или набор элементов управления, отвечающих за отображение данных на экране. Модель представления (ViewModel) — это и есть та магия, стоящая за всем. Она содержит логику пользовательского интерфейса, команды, события и ссылку на модель. В MVVM модель преставления не обязана обновлять непосредственно данные представления. Благодаря мощному механизму привязки, представленному в WPF, пользовательский интерфейс обновляется, как только обновляется модель преставления. Для этого модель преставления должна реализовывать интерфейс INotifyPropertyChanged и запустить событие PropertyChanged.

## **3.2 Проектирование структуры базы данных**

Для обеспечения работоспособности программного средства была разработана база данных, что являлось обязательным условием курсового проекта. База данных была тщательно сконструирована до начала написания самого приложения. Для ее создания использовалась реляционная система управления базами данных (РСУБД) MS SQL.

Таким образом была разработана база данных CoachSearchDB, которая включает в себя таблицы Bookings, Clients, Messages, Reviews, Tutors, Users.

Таблица Booking используется для хранения и контроля данных о записях. Включает 5 столбцов: id, ClientId, TutorId, DateTime, Status.

Таблица Clients используется для хранения данных о всех клиентах. Включает 4 столбца: UsertId, Name, BookingCount, AvatarPath.

Таблица Messages используется для хранения сообщений от пользователя к пользователю. Включает 5 столбцов: Id, SenderId, ReceiverId, [Content], SentAt.

Таблица Reviews используется для хранения и управления отзывами и рейтингами. Включает 7 столбцов: Id, ClientId, TutorId, Text, Rating, CreatedAt, BookingId.

Таблица Users используется для хранения данных для авторизации разных типов пользователей. Включает 4 столбца: Id, Name, Email, Password.

Таблица Tutors используется для хранения данных репетиторов и информации о них. Включает 4 столбца: UsertId, Id, Name, AvatarPath.

Структура проекта представлена на рисунке 3.2.

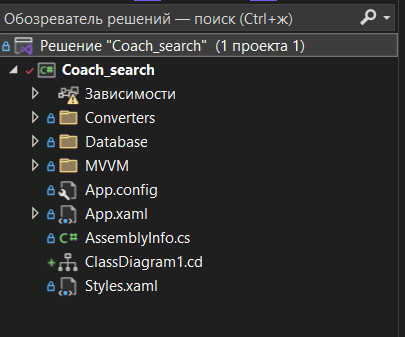


Рисунок 3.2 − Структура проекта

Папка Database хранит классы и интерфейсы, которые необходимы для подключения и работы с базой данных, так же её предварительного заполнения.

Папка Converters хранит конвертеры.

Папка Model хранит классы, описывающие используемые в проекте сущности (объекты).

Папка Resources хранит различные картинки.

Папка View содержит элементы пользовательского интерфейса, а именно окна и страницы, которые пользователь может увидеть на экране.

Папка ViewModel содержит в себе интерфейсы и классы со свойствами и командами, которые связаны механизмом привязки с объектами папки View.

Взаимосвязь между классами хорошо видна в диаграмме классов, представленной в Приложении Б.

# **4 Реализация программного средства**

В данном разделе будут рассмотрены ключевые классы, определяющие архитектуру разрабатываемого программного обеспечения. Несмотря на наличие множества вспомогательных классов, именно эти компоненты играют основополагающую роль в обеспечении корректной и устойчивой работы системы. Особое внимание уделяется классам, реализующим архитектурный паттерн MVVM, механизм команд (Command), а также обеспечивающим взаимодействие с базой данных.

## **4.1 Реализация паттерна MVVM**

Паттерн MVVM в данном проекте реализуется с использованием базового класса **BaseViewModel** (листинг 4.1), который наследует интерфейс **INotifyPropertyChanged.** Этот интерфейс служит для оповещения представления о том, что значения свойств во ViewModel изменились.

Реализация интерфейса предполагает вызов события PropertyChanged каждый раз, когда изменяется значение свойства. Благодаря этому механизм привязки данных (data binding) автоматически отслеживает изменения в состоянии объекта и обновляет соответствующие элементы пользовательского интерфейса без необходимости прямого вмешательства со стороны разработчика.

На основе **BaseViewModel** построены все классы ViewModel, использующиеся в проекте. Это обеспечивает единообразие, упрощает реализацию логики представлений и способствует чистому разделению между интерфейсом и логикой приложения.

|  |
| --- |
| public abstract class BaseViewModel : INotifyPropertyChanged{  public event PropertyChangedEventHandler PropertyChanged;  protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null)  { PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));  }  protected bool SetProperty<T>(ref T field, T value, [CallerMemberName] string propertyName = null){  if (Equals(field, value)) return false;  field = value;  OnPropertyChanged(propertyName);  return true;  } |

Листинг 4.1 – Cтруктура класса BaseViewModel

Примером реализации потомка данного класса может служить класс TutorDetailsViewModel, его реализация представлена в Приложении В.

В приложении используется паттерн Command который позволяет инкапсулировать запрос на выполнение определенного действия в виде отдельного объекта. В WPF команды представлены интерфейсом ICommand. Для создания команд нужен класс, который реализует интерфейс ICommand.

## **4.2 Классы для работы с базой данных**

В проекте реализован класс DatabaseHelper, отвечающий за взаимодействие с базой данных через ADO.NET. Он инкапсулирует логику работы с SQL-запросами, управление подключениями и безопасную обработку данных.

Фрагмент класса DatabaseHelper представлен в листинге 4.2.

public class DatabaseHelper

{

public string ConnectionString { get; } = "Server=Duba;Database=CoachSearchDb;Trusted\_Connection=True;Encrypt=False;Connect Timeout=5;";

public int AddUser(string name, string email, string passwordHash, Coach\_search.Models.UserType userType)

{

using (var connection = new SqlConnection(ConnectionString))

{

connection.Open();

string query = "INSERT INTO Users (Name, Email, Password, UserType) OUTPUT INSERTED.Id VALUES (@Name, @Email, @Password, @UserType)";

using (var command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Name", name);

command.Parameters.AddWithValue("@Email", email);

command.Parameters.AddWithValue("@Password", passwordHash);

command.Parameters.AddWithValue("@UserType", userType.ToString());

return (int)command.ExecuteScalar();

}

}

}

}

Листинг 4.2 – Фрагмент класса DatabaseHelper

Класс DatabaseHelper реализован как центральный компонент для взаимодействия с базой данных через ADO.NET. Он обеспечивает выполнение SQL-запросов, управление подключениями и безопасную обработку данных. Строка подключения ConnectionString содержит параметры для соединения с SQL Server, включая адрес сервера, имя базы данных и настройки безопасности. Подключения создаются внутри блоков using, что гарантирует корректное освобождение ресурсов даже при возникновении исключений.

Диаграмма базы данных представлена на рисунке 4.1.

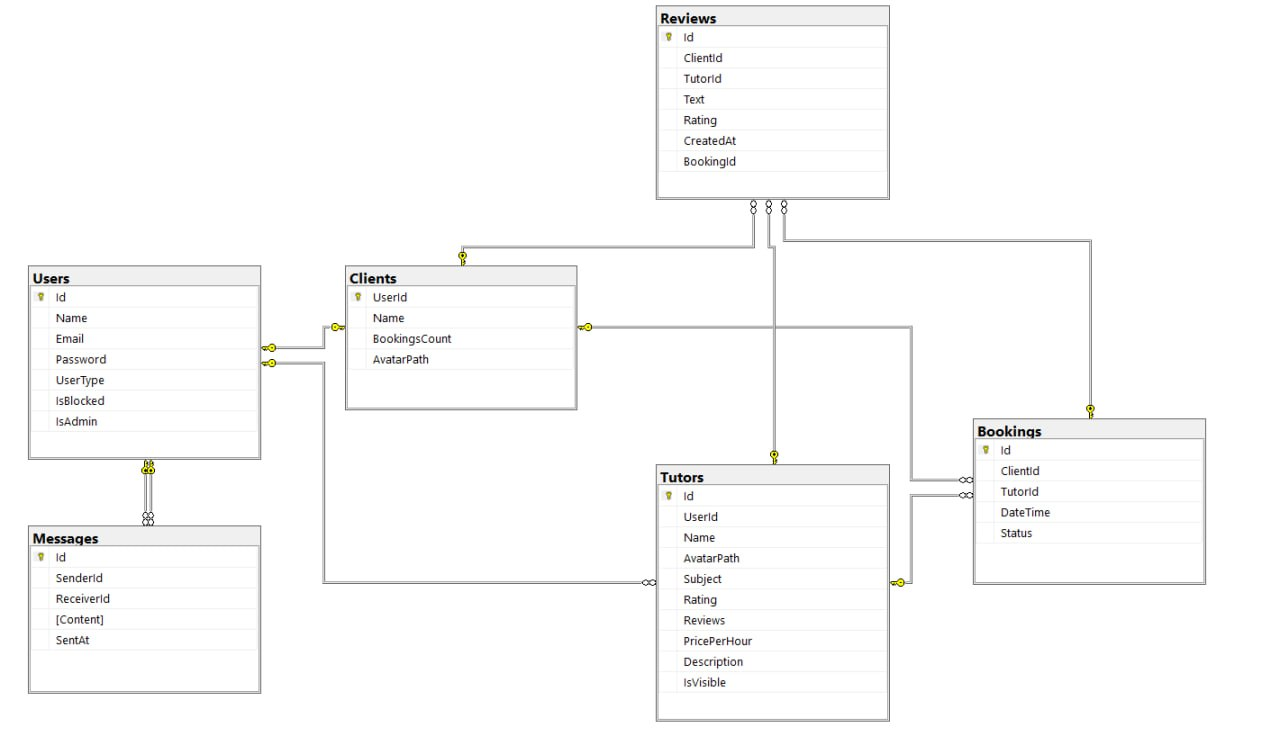


Рисунок 4.1 – Диаграмма базы данных

Реализация класса `DatabaseHelper` через ADO.NET обеспечивает безопасное и эффективное взаимодействие с базой данных за счет использования защищенной строки подключения (`ConnectionString`), включающей параметры аутентификации и шифрования. Применение блоков `using` для управления подключениями гарантирует автоматическое освобождение ресурсов, минимизируя риски утечек и повышая стабильность приложения. Централизация логики работы с БД в классе упрощает выполнение запросов и обработку данных, обеспечивая единую точку управления. Диаграмма базы данных визуализирует структуру таблиц, связей и ключей, что способствует четкому проектированию и поддержанию целостности данных. В совокупности эти подходы снижают вероятность ошибок, улучшают читаемость кода и обеспечивают масштабируемость системы.

# **5 Тестирование, проверка работоспособности приложения и анализ результатов**

Данный раздел посвящен процедуре проверки работоспособности разработанного программного решения. Основная задача тестирования заключалась в подтверждении соответствия системы критериям, установленным в техническом задании. В разделе приводится детализация проведенных испытаний, включая анализ полученных данных и оценку реакции приложения на нештатные условия работы. Акцент сделан на исследовании стабильности системы при возникновении ошибок и внештатных сценариев. Результаты тестов демонстрируют, что программный продукт полностью отвечает заявленным функциональным требованиям, обеспечивает предсказуемую работу в различных условиях и готов к внедрению в эксплуатационную среду.

## **5.1 Тестирование регистрации и авторизации**

Все ключевые поля ввода в приложении проходят обязательную валидацию на корректность заполненных данных. При ошибке пользователю будет выведено сообщение о том, что именно заполнено неправильно.

При запуске приложения открывается форма авторизации, пользователь должен ввести свой логин и пароль, либо переключиться на представление регистрации. При ошибке ввода выведется сообщение представленные на рисунках 5.1-5.5.



Рисунок 5.1 – При регистрации не введены данные

Как мы видим, окно авторизации работает корректно.

Валидация при неверном формате электронной почты:

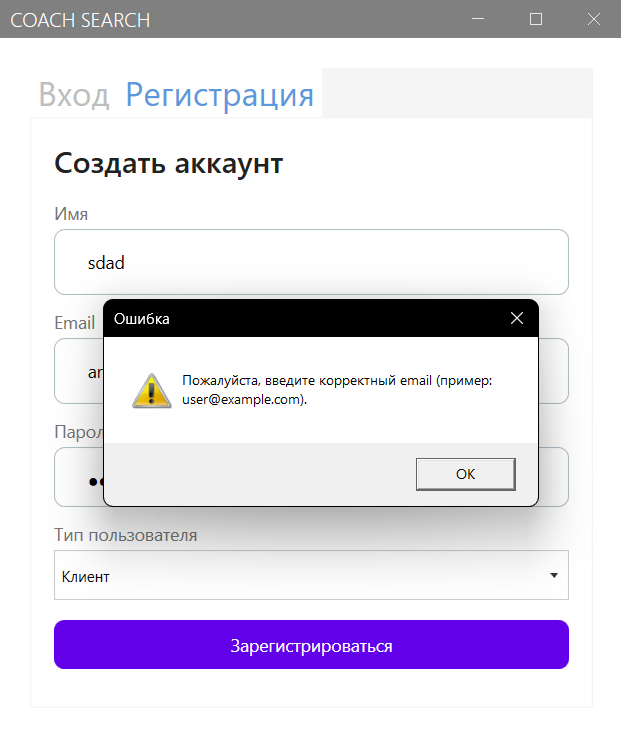


Рисунок 5.2 – При регистрации введен неверный формат почты

В этом случае система отображает сообщение об ошибке и запрашивает повторный ввод данных.

Валидация на повторяющиеся имя и почту:

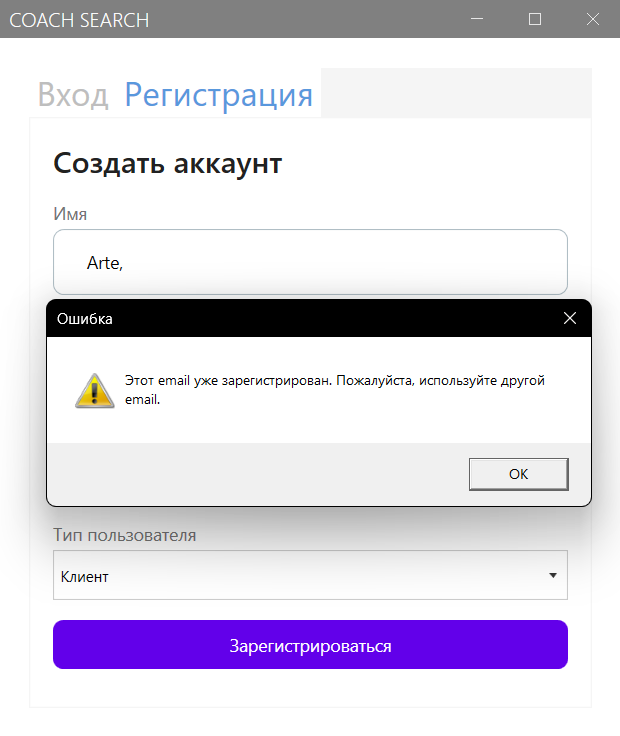


Рисунок 5.3 – При регистрации введены уже существующая почта

Как мы видим, окно регистрации работает корректно.

Теперь введем некорректные данные при авторизации

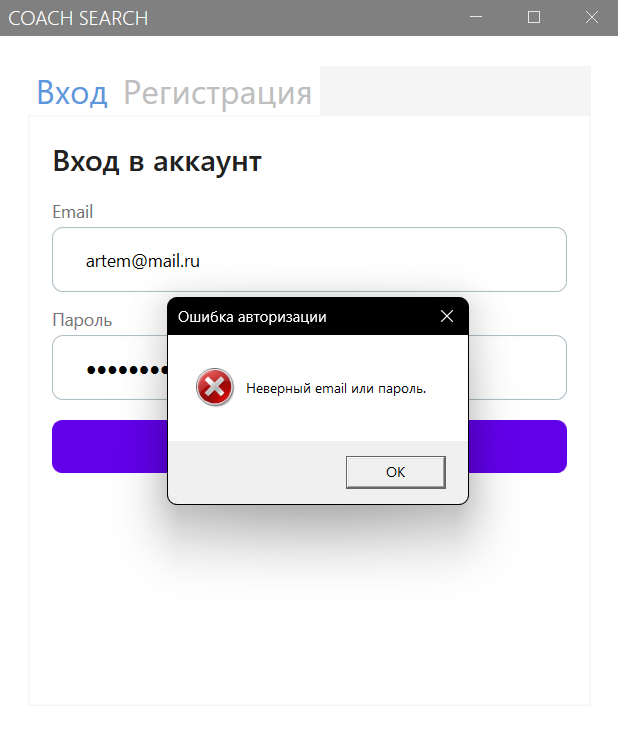


Рисунок 5.5 – При авторизации введены некорректные данные

Можно заметить, что окно авторизации работает корректно.

## **5.2 Тестирование всей работы приложение**

В данном приложении одной из главных составляющих является запись на занятие.

Допустим, на занятие уже кто-то записался и мы попробуем записаться на это же время. Результат отображен на рисунке 5.6.

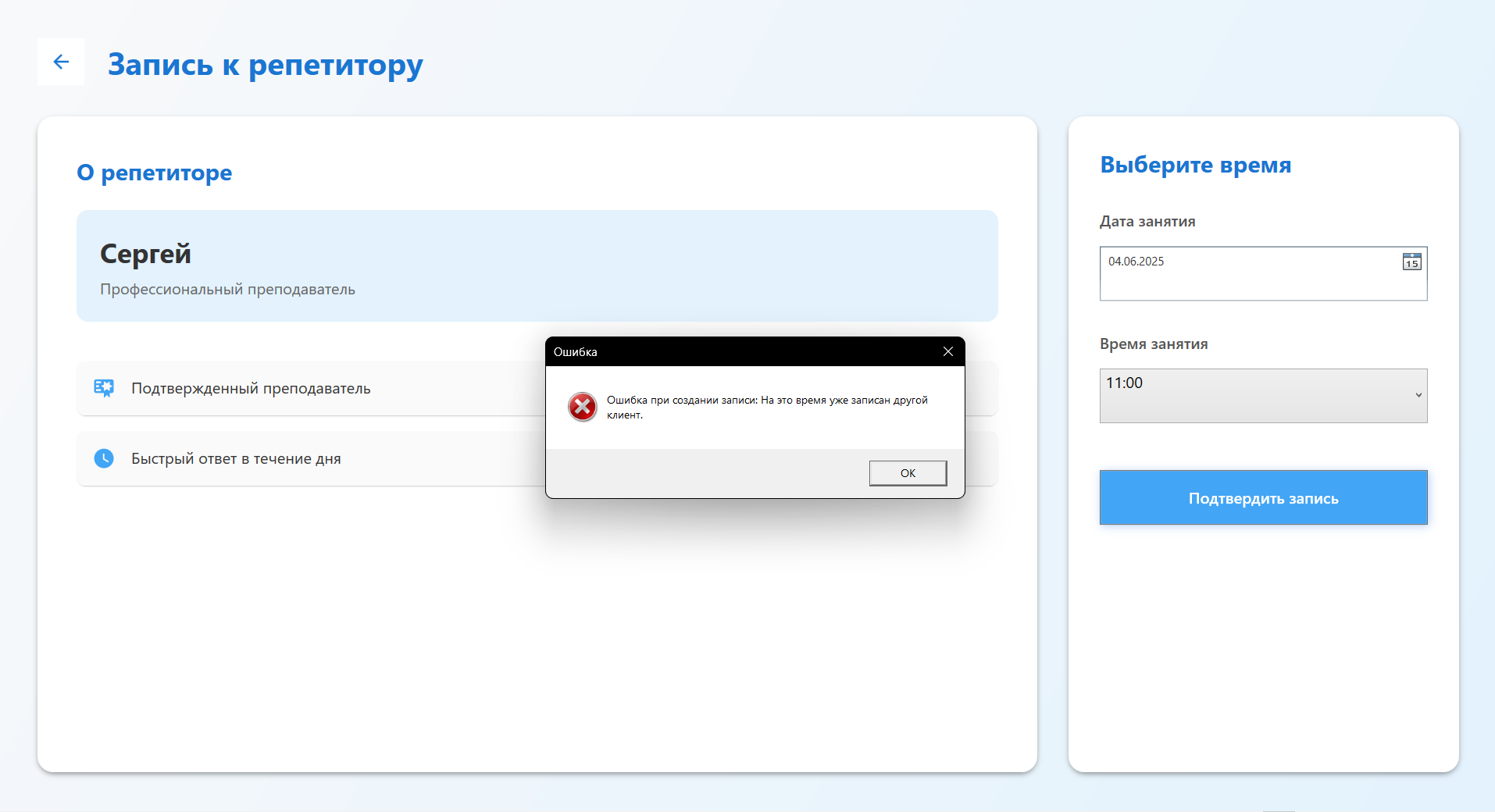


Рисунок 5.6 – Результат записи на уже занятое время

Можно заметить, что окно авторизации работает корректно.

Далее попробуем сделать профиль репетитора видимым, не введя данные о нем. Результат представлен на рисунке 5.7.

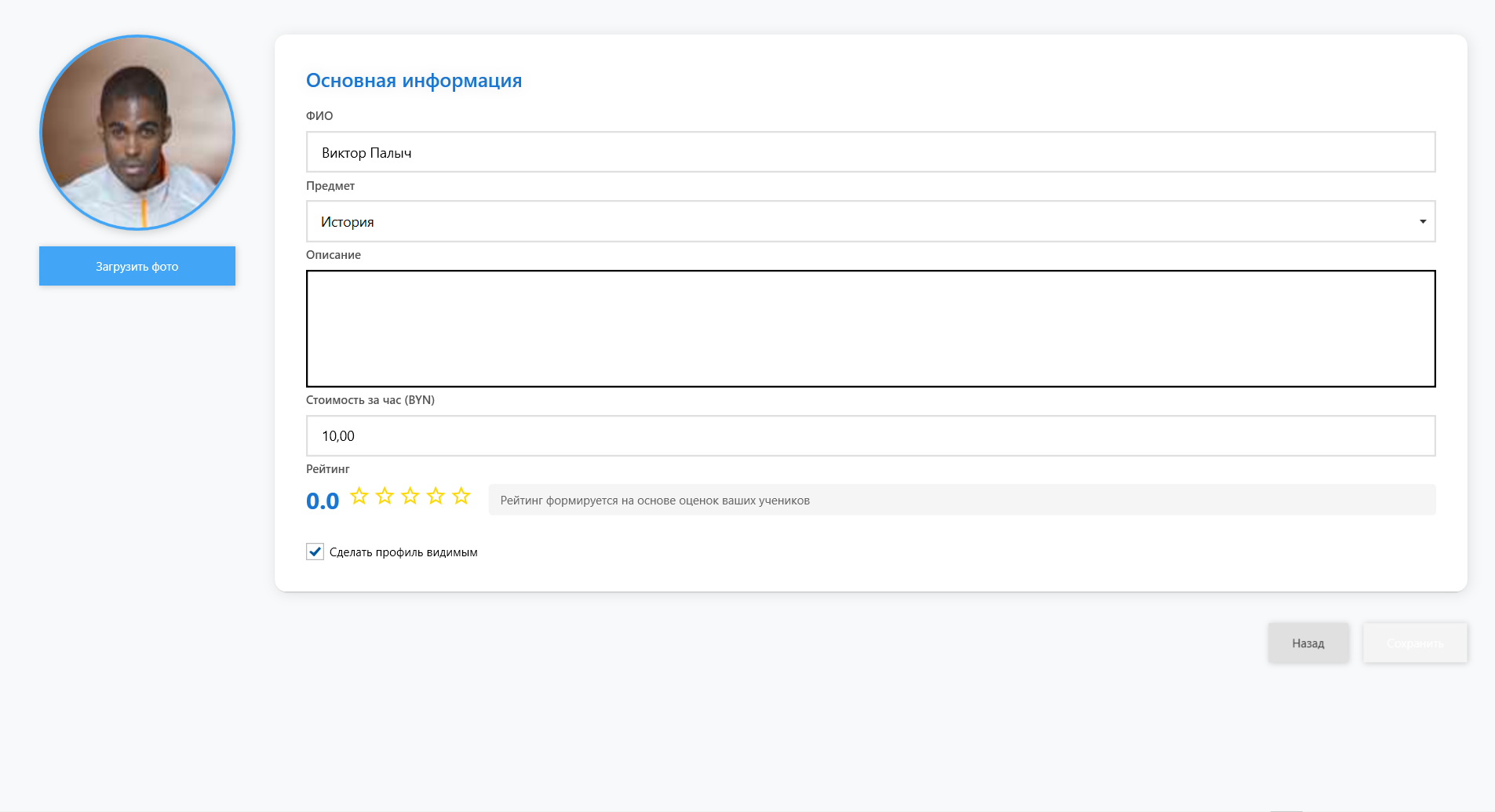


Рисунок 5.7 – Ошибка пустых полей

Проведённое тестирование подтверждает высокую надёжность и стабильность разработанного программного продукта. Система эффективно обрабатывает исключения, корректно реагируя на ошибочные или нестандартные действия пользователя, что предотвращает сбои и обеспечивает бесперебойную работу без потери данных. Это особенно важно для эксплуатации программы в реальных условиях, где ошибки ввода или непредвиденные ситуации неизбежны.

Валидация пользовательских данных реализована на высоком уровне: система успешно отсеивает некорректные данные и уведомляет пользователя о необходимости их исправления. Это повышает удобство использования и снижает вероятность ошибок, влияющих на обработку и сохранность информации. Тестирование граничных случаев и сценариев с преднамеренно некорректным поведением пользователя не выявило критических сбоев, что свидетельствует о продуманной логике и архитектуре ПО.

Таким образом, программный продукт полностью соответствует заявленным требованиям и готов к использованию в рабочей среде. Его стабильное и предсказуемое поведение при взаимодействии с пользователем создаёт надёжную основу для масштабирования и внедрения новых функций без ущерба для качества.

# **6 Руководство по использованию**

## **6.1 Регистрация и авторизация**

При запуске приложения запускается окно авторизации. Если у пользователя еще нет аккаунта, ему следует нажать на кнопку «Регистрация», которая его перенаправит на представление с формой для регистрации. Представление авторизации представлено на рисунке 6.1.

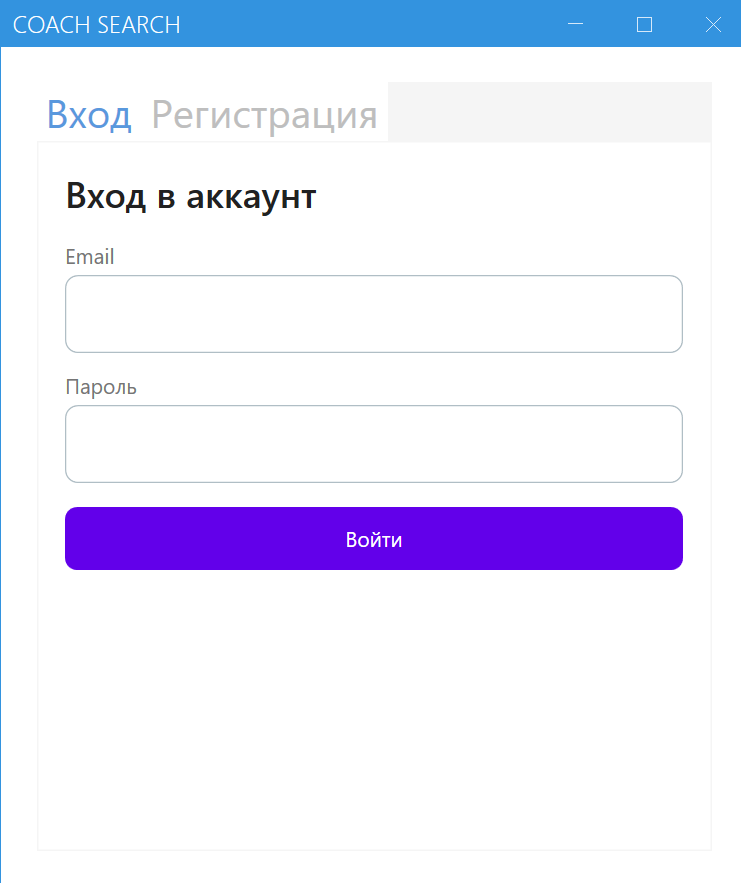


Рисунок 6.1 – Представление авторизации

Если была нажата кнопка «Регистрация», в открывшемся представлении следует ввести имя пользователя, пароль и почту, затем нажать кнопку зарегистрироваться, если данные введены корректно, то сразу же откроется главное окно. Представление регистрации изображено на рисунке 6.2.

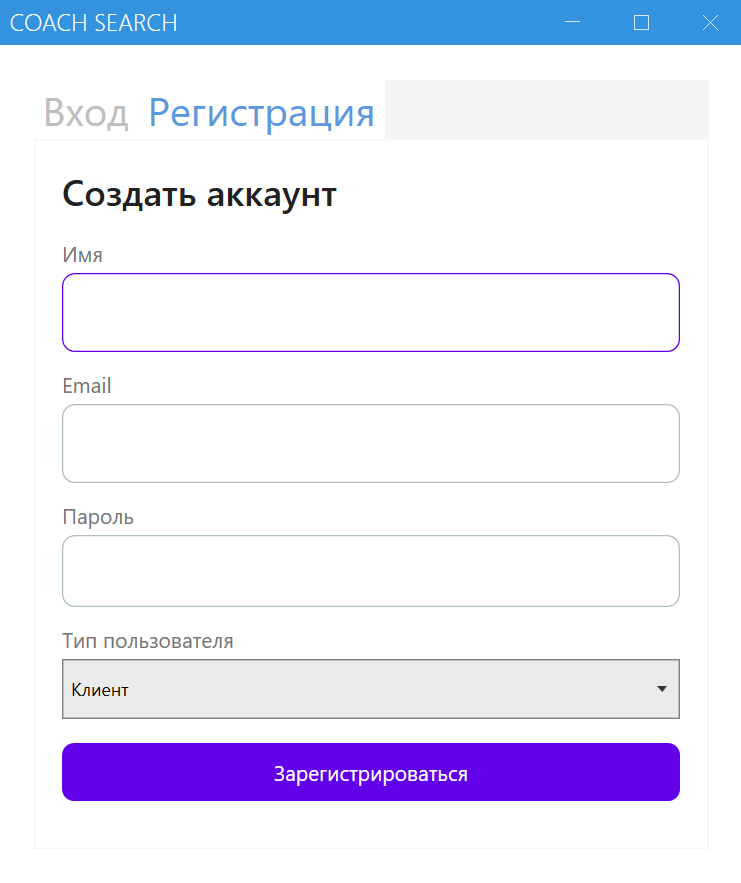


Рисунок 6.2 ‒ Представление регистрации

В случае, если кнопка была нажата ошибочно, можно нажать на кнопку «Вход» и вернуться к представлению авторизации.

## **6.2 Пользовательская часть**

При успешной авторизации пользователя приложение закроет окно регистрации и откроет главное окно пользовательского приложения с вертикальным меню и удобной фильтрацией репетиторов. На рисунке 6.3. показана главная страница.

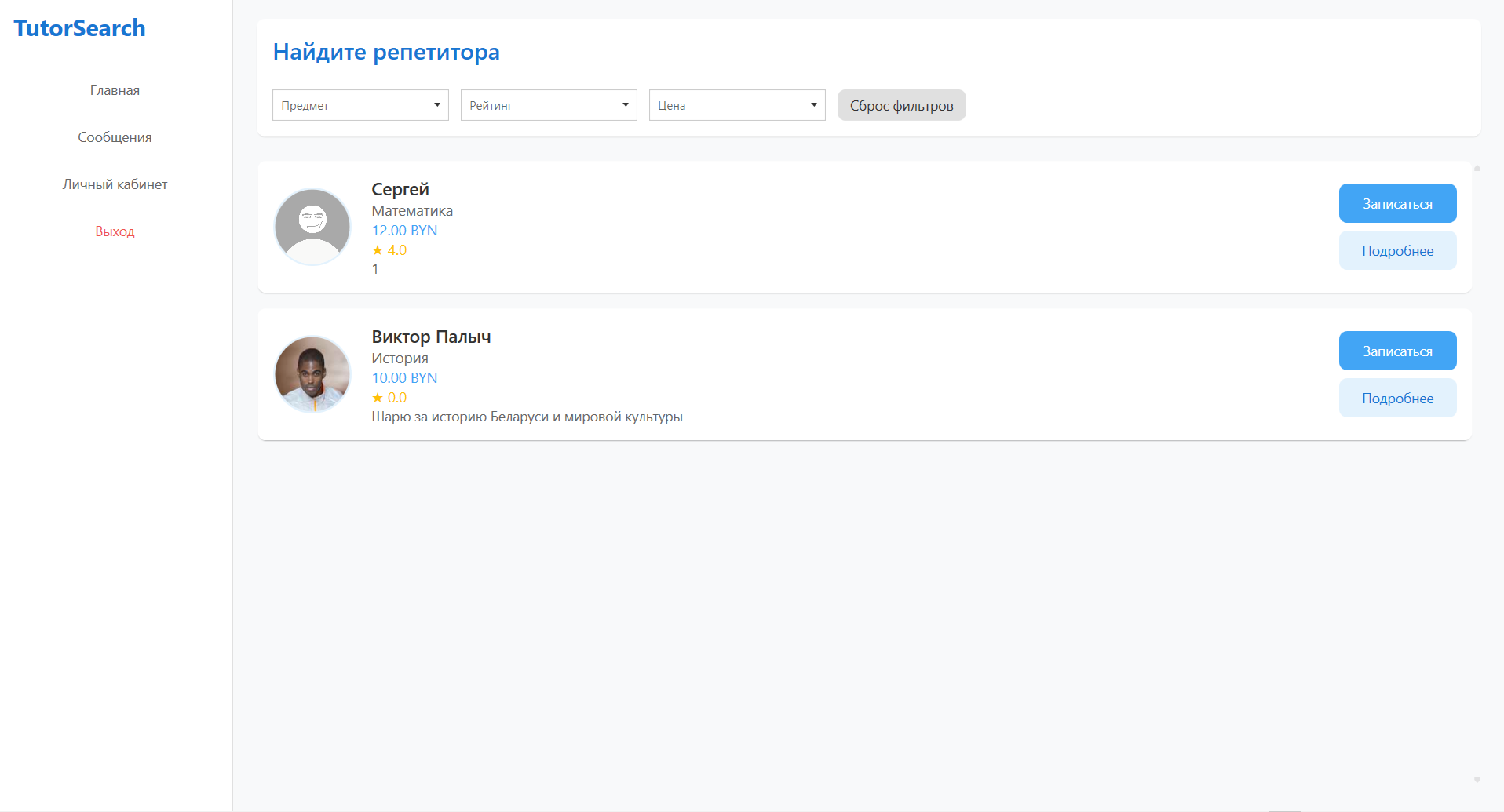


Рисунок 6.3 ‒ Главная страница

Также пользователь может ознакомиться с репетитором, перейдя на его страницу по кнопке «Подробнее». На рисунке 6.4 показана страница репетитора.

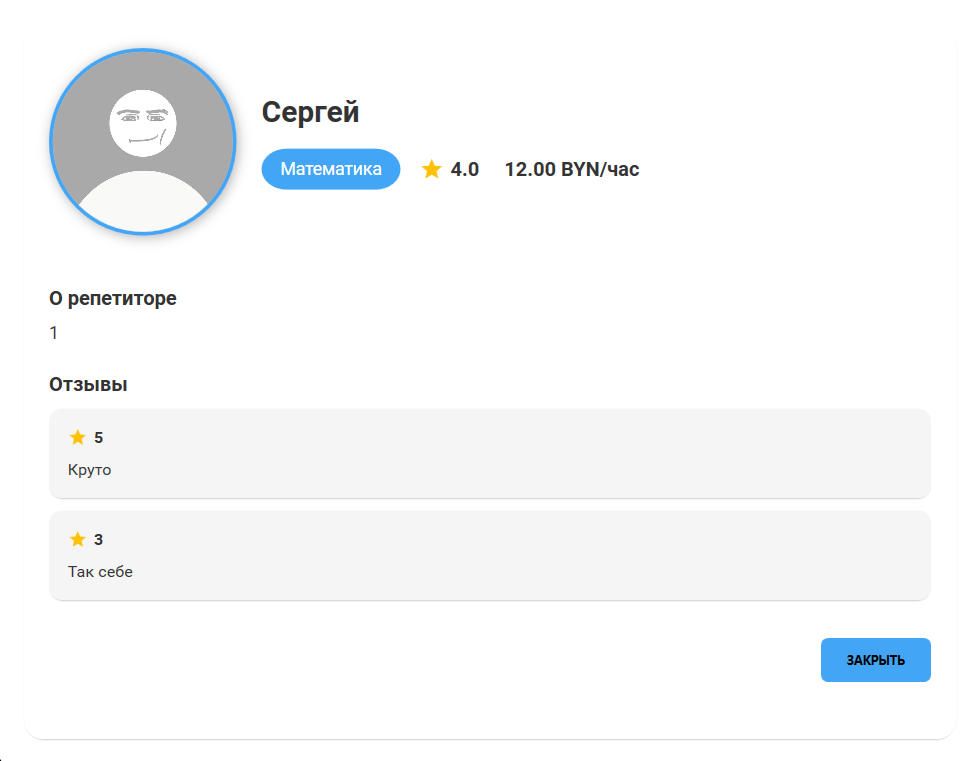


Рисунок 6.4 ‒ Страница репетитора

После пользователь может записаться на занятие с репетитором, перейдя на страницу записи по кнопке «Записаться». На рисунке 6.5 показана страница для записи.

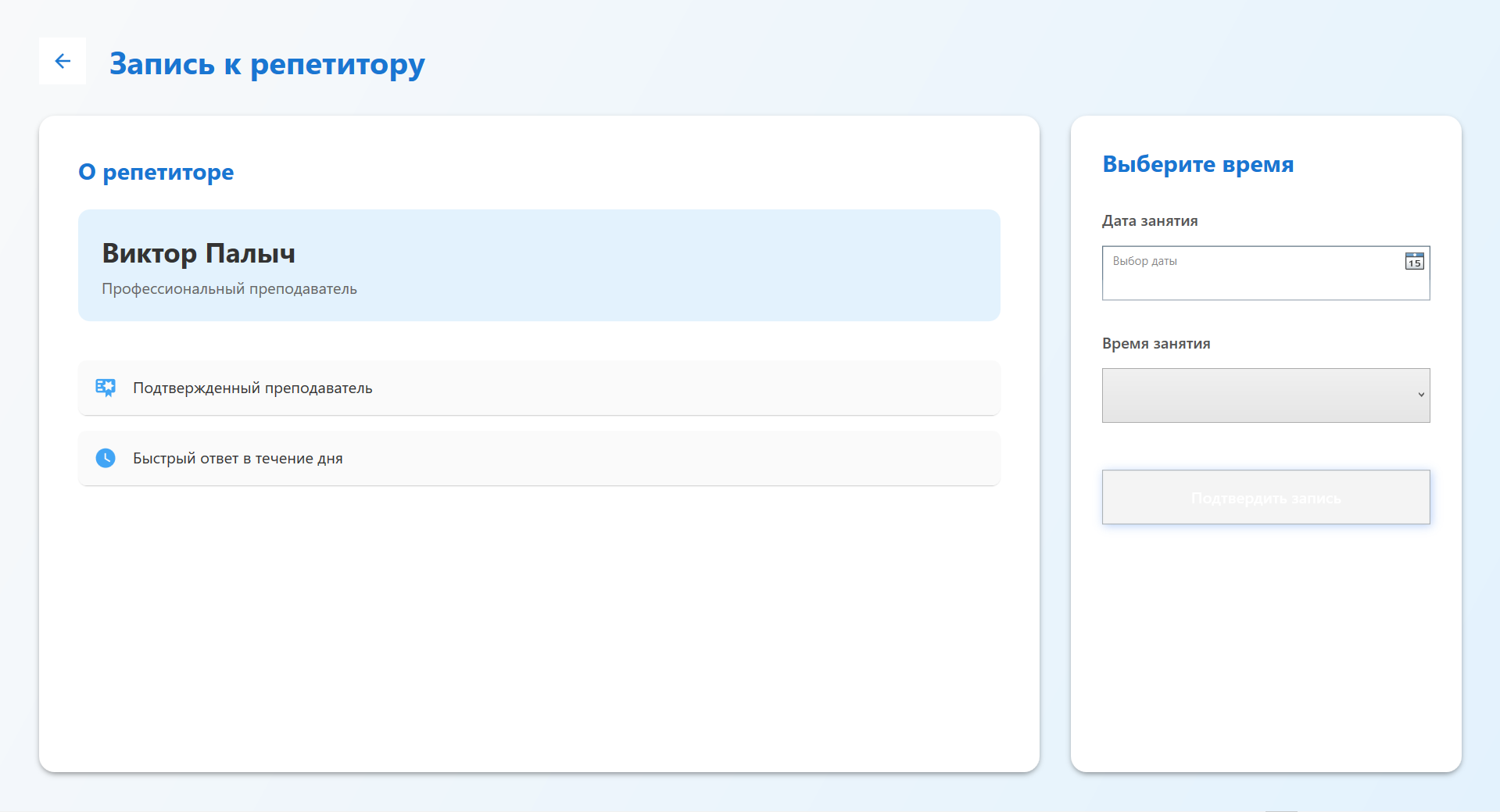


Рисунок 6.5 ‒ Страница для записи

Через левое вертикальное меню можно перейти к профилю с личной информацией, где можно просмотреть свои записи и отзывы. Страница профиля представлена на рисунке 6.6

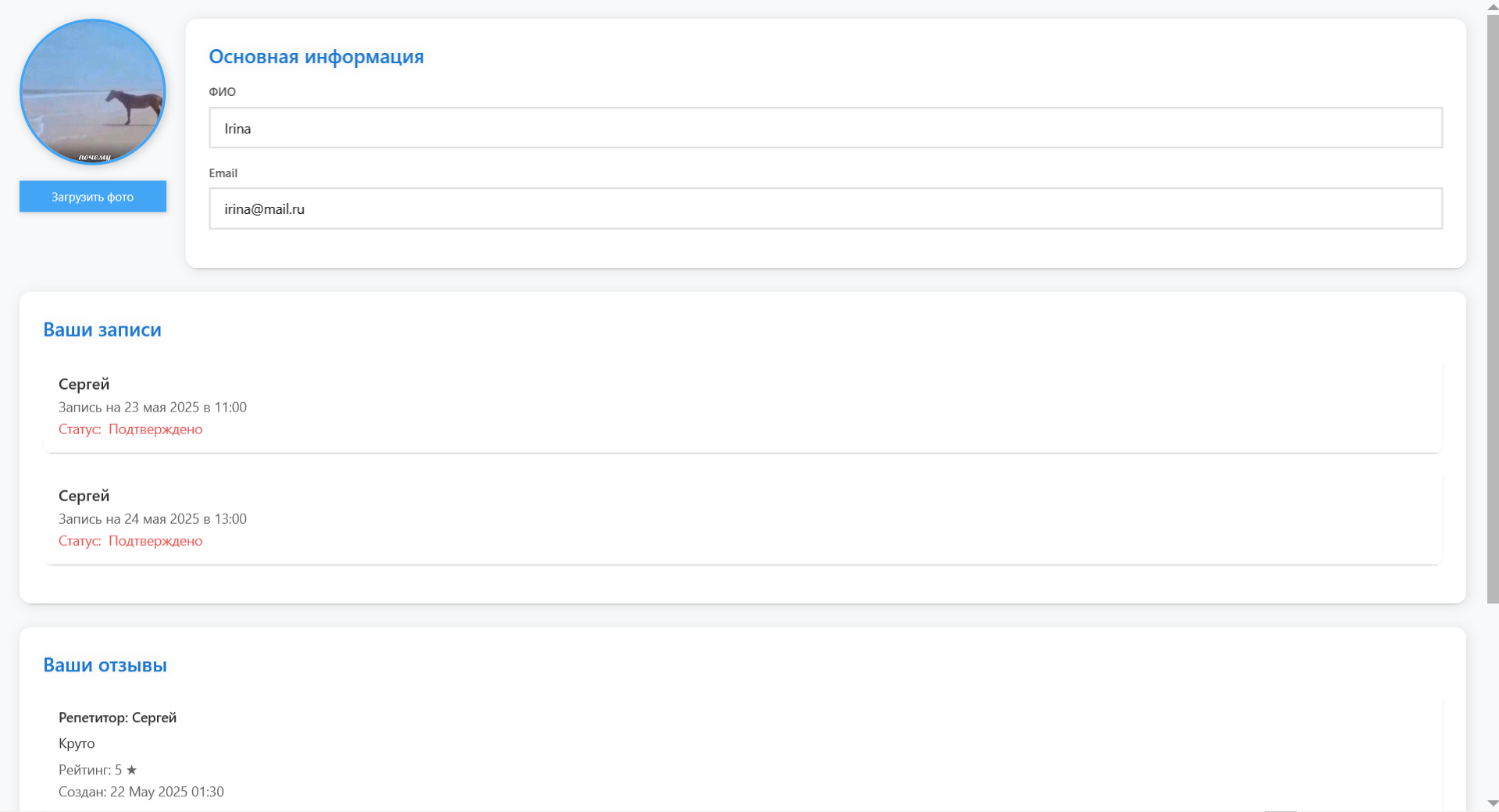


Рисунок 6.6 ‒ Страница профиля

Также из главного меню пользователь может открыть чат с преподавателем. Окно чата представлено на рисунке 6.7.

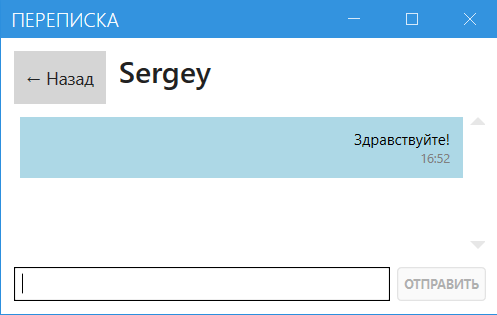


Рисунок 6.7 ‒ Окно чата

После того, как репетитор подтвердит вашу запись, вы можете оставить отзыв на него. Окно отзывов представлено на рисунке 6.8.



Рисунок 6.8 ‒ Окно отзывов

Если пользователь зайдет как репетитор, то в его вертикальном меню появится раздел с расписанием. Страница расписания представлена на рисунке 6.9.

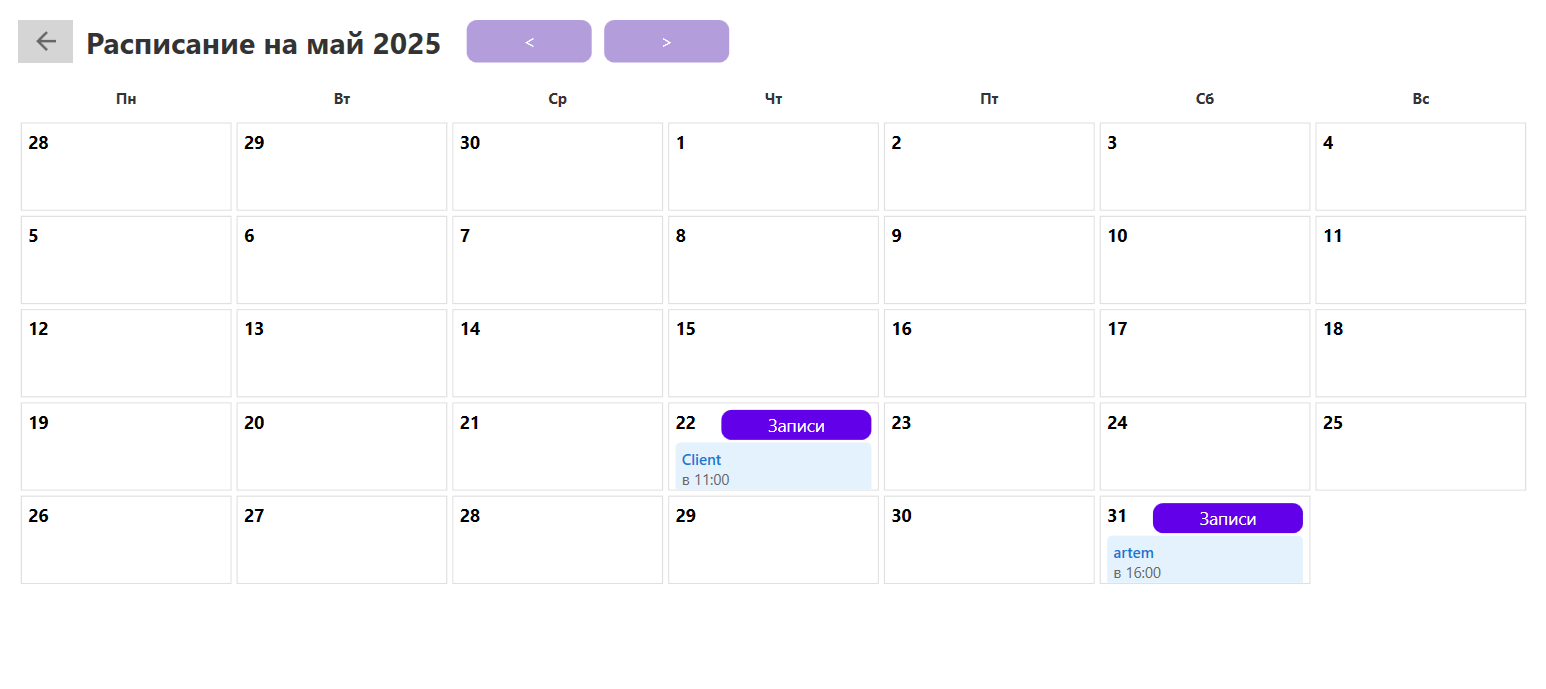


Рисунок 6.9 ‒ Расписание занятий

Из расписания преподаватель может перейти на окно записей на конкретный день. Данное окно представлено на странице 6.10.

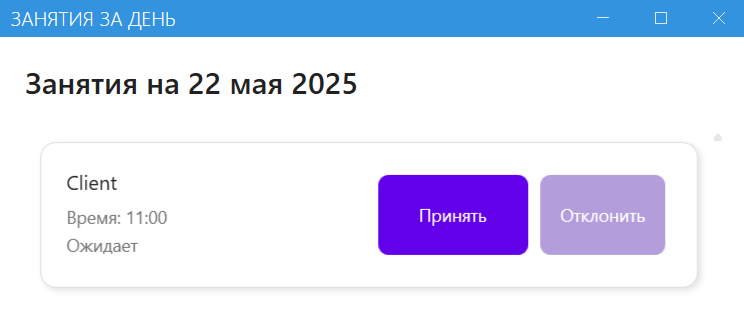


Рисунок 6.10 ‒ Записи на день

Можно заметить, что репетитор может принять или отклонить заявку на занятие.

**6.3 Администрирование**

После успешной авторизации администратора, так же как и для пользователя, открывается главная страница. Однако в меню происходит изменение: кнопка личного кабинета отсутствует, а вместо неё отображается кнопка панели администратора, ведущая на соответствующую страницу для управления пользователями и модерации отзывов. Вертикальное меню представлено на рисунке 6.10.

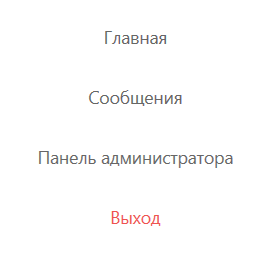


Рисунок 6.10 ‒ Вертикальное меню

Страница панели администратора представлена на рисунке 6.11.

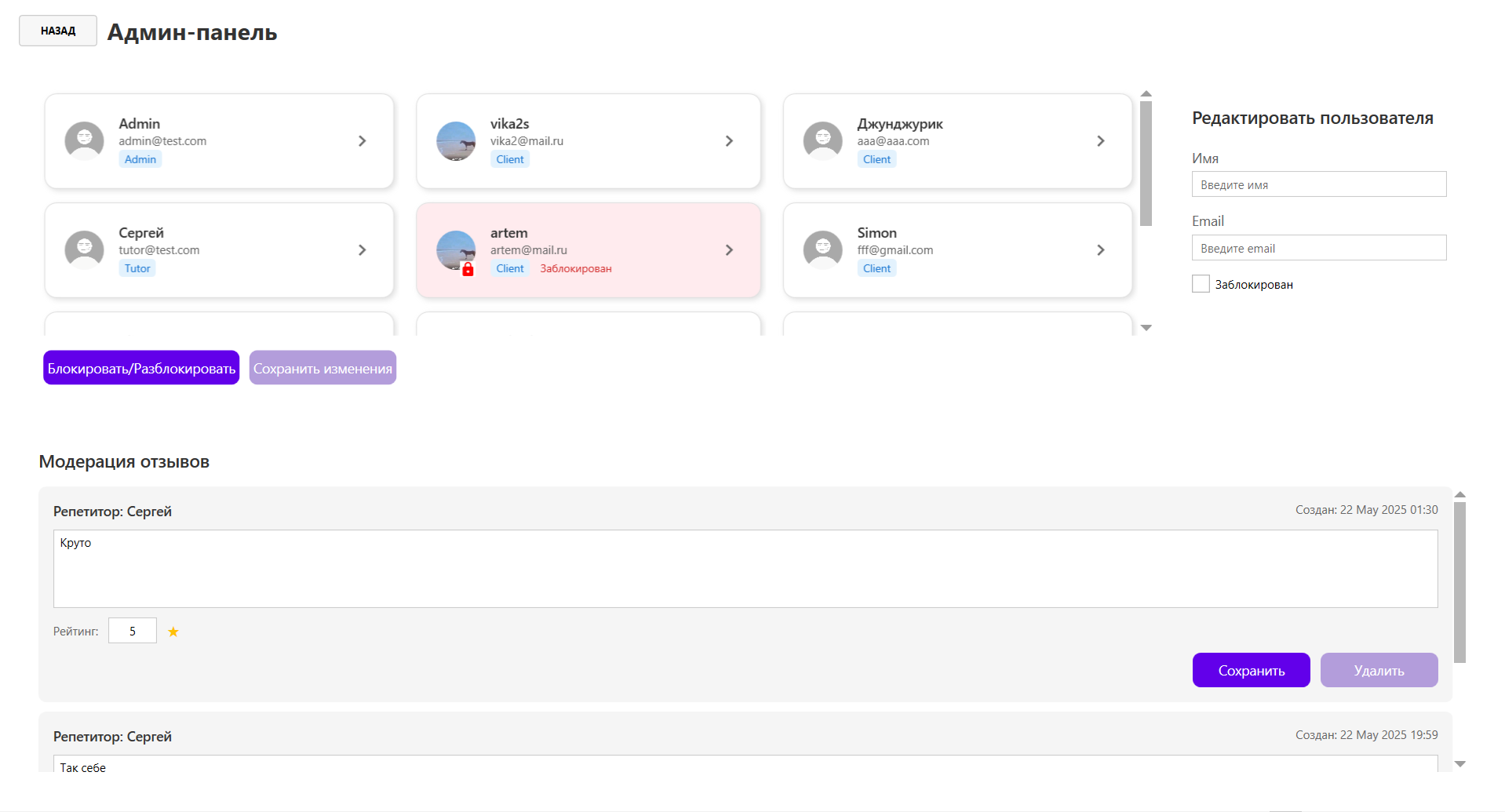


Рисунок 6.11‒ Панель администратора

Руководство по использованию разработанного программного продукта играет важную роль в обеспечении его доступности и эффективности для конечного пользователя. Оно выступает неотъемлемой частью системы, обеспечивая пользователей не только базовыми сведениями о запуске и установке программы, но и пошаговыми инструкциями по выполнению основных операций. Благодаря подробному описанию интерфейса, функций и возможных сценариев взаимодействия, руководство позволяет значительно сократить время на освоение программного обеспечения, облегчает адаптацию новых пользователей и снижает нагрузку на службу технической поддержки.

Особое внимание в руководстве уделено описанию всех ключевых элементов интерфейса, что делает процесс навигации по программе интуитивно понятным. Также подробно описаны действия, которые должен предпринять пользователь для выполнения типовых операций — от регистрации и авторизации до создания, редактирования и управления объектами внутри системы.

Таким образом, руководство пользователя выполняет функцию связующего звена между программным продуктом и конечным пользователем. Оно способствует повышению общей удовлетворённости от использования системы, минимизирует риск некорректного использования и служит надёжным источником информации при возникновении затруднений. Наличие такого документа значительно повышает качество сопровождения программного продукта и обеспечивает его успешное внедрение в повседневную деятельность.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсового проекта была разработана десктопная платформа для поиска и взаимодействия с репетиторами, объединяющая потребности учеников, преподавателей и администраторов в единой системе. На этапе проектирования проведен анализ рынка аналогичных сервисов, что позволило выявить ключевые проблемы существующих решений — недостаточную гибкость фильтрации, слабую интеграцию инструментов планирования и ограниченные возможности персонализации профилей.

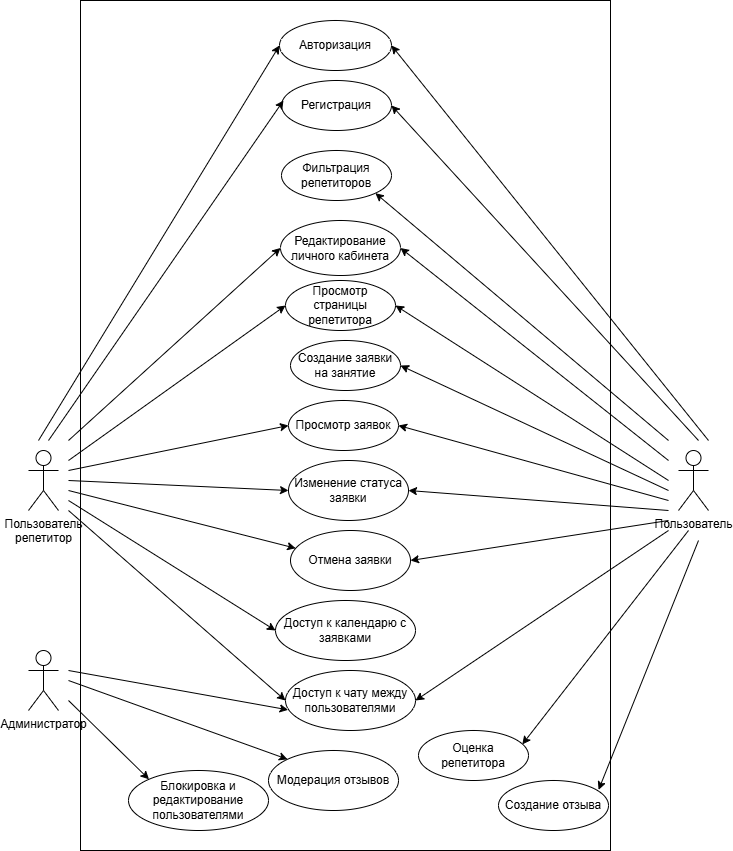
Платформа реализована с использованием технологий C#, .NET Framework и WPF, что обеспечило создание интуитивно понятного интерфейса с адаптивным дизайном. Архитектура, построенная на шаблоне MVVM, позволила четко разделить слои представления, бизнес-логики и работы с данными, повысив масштабируемость и удобство дальнейшей поддержки кода. Для хранения информации применена реляционная база данных MS SQL Server, гарантирующая целостность и безопасность данных за счет шифрования конфиденциальных сведений и реализации ролевой модели доступа.

**Список использованных источников**

1. Онлайн-ресурс «profi-bel.by» [Электронный ресурс] – Режим доступа: profi-bel.by – Дата доступа: 16.03.2025
2. Онлайн-ресурс «repetit-center.by» [Электронный ресурс] – Режим доступа: repetit-center.by – Дата доступа: 16.03.2025
3. Онлайн-ресурс «repetit.ru» [Электронный ресурс] – Режим доступа: repetit.ru – Дата доступа: 16.03.2025
4. Онлайн-ресурс «protutor.ru» [Электронный ресурс] – Режим доступа: protutor.ru – Дата доступа: 16.03.2025
5. MVVM: полное понимание (+WPF)// [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/338318/ – Дата доступа: 27.04.2025

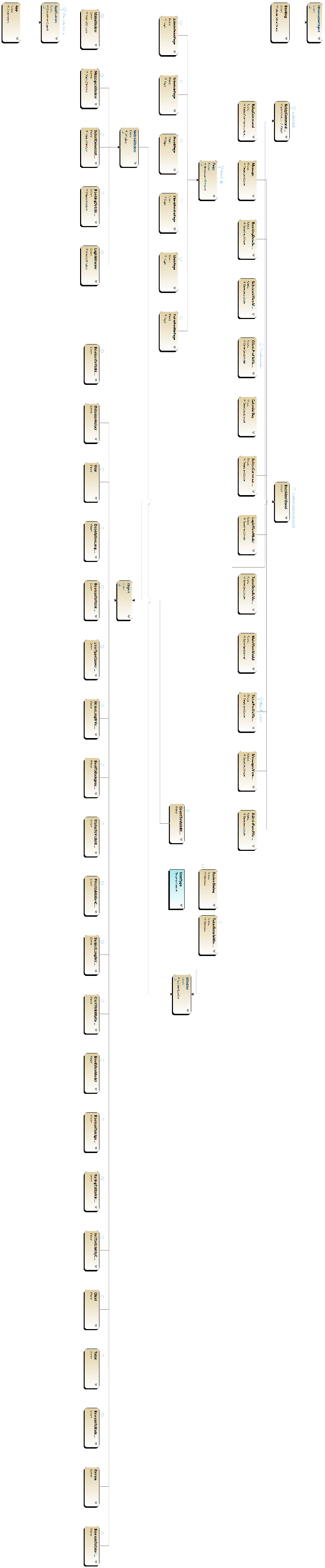
**ПРИЛОЖЕНИЯ А**

Диаграмма вариантов использования



**ПРИЛОЖЕНИЕ Б**

Диаграмма классов



**ПРИЛОЖЕНИЕ В**

|  |
| --- |
| public class TutorDetailsViewModel : BaseViewModel  {  private readonly Tutor \_tutor;  private readonly DatabaseHelper \_dbHelper;  private ObservableCollection<Review> \_reviews;  public string Name => \_tutor.Name;  public string Subject => \_tutor.Subject;  public string Description => \_tutor.Description;  public string AvatarPath => \_tutor.AvatarPath;  public double Rating => \_tutor.Rating;  public decimal PricePerHour => \_tutor.PricePerHour;  public ObservableCollection<Review> Reviews  {  get => \_reviews;  set  {  \_reviews = value;  OnPropertyChanged();  }  }  public ICommand CloseCommand { get; }  public TutorDetailsViewModel(Tutor tutor)  {  \_tutor = tutor ?? throw new ArgumentNullException(nameof(tutor));  \_dbHelper = new DatabaseHelper();  LoadReviews();  CloseCommand = new RelayCommand(CloseWindow);  }  private void LoadReviews()  {  try  {  var reviews = \_dbHelper.GetTutorReviews(\_tutor.Id);  Reviews = new ObservableCollection<Review>(reviews);  }  catch (Exception ex)  {  System.Diagnostics.Debug.WriteLine($"Error loading reviews: {ex.Message}");  Reviews = new ObservableCollection<Review>();  }  }  private void CloseWindow(object parameter)  {  var window = Application.Current.Windows.OfType<TutorDetailsWindow>().FirstOrDefault();  window?.Close();  }  } |

Листинг В – структура класса TutorDetailsViewModel