

Содержание

Введение	5
1 Постановка задачи и обзор аналогичных решений	6
1.1 Постановка задачи	6
1.2 Обзор аналогов	6
1.2.1 Интернет-ресурс «INSIDE»	6
1.2.2 Интернет-ресурс «Grow Up»	8
1.2.3 Интернет-ресурс «Nora»	9
1.3 Выводы по разделу	10
2 Проектирование web-приложения	11
2.1 Функциональность web-приложения	11
2.2 Проектирование базы данных	14
2.3 Архитектура web-приложения	19
2.4 Выводы по разделу	20
3 Реализация web-приложения	21
3.1 Программная платформа Node.js	21
3.2 Система управления базами данных PostgreSQL	21
3.3 Object-Relational Mapping (ORM) Prisma	21
3.4 Программные библиотеки	27
3.5 Сторонние сервисы Stripe, Cloudinary, OpenAI	28
3.6 Применение паттернов Unit of Work и Repository	28
3.7 Структура серверной части	29
3.8 Реализация функций для гостя	31
3.8.1 Регистрация	31
3.8.2 Просмотр процедур, психологов, отзывов	32
3.8.3 Поиск услуг и психологов	34
3.9 Реализация функций для пользователя	35
3.9.1 Добавить отзыв	35
3.9.2 Оформление услуги	35
3.9.3 Удалить отзыв	36
3.9.4 Аутентифицироваться	37
3.9.5 Изменение информации в профиле	37
3.9.6 Отмена услуг	38
3.9.7 Просмотр процедур, психологов, отзывов	39
3.9.8 Поиск услуг и психологов	40
3.10 Реализация функций для психолога	41
3.10.1 Аутентифицироваться	41
3.10.2 Создание талонов	42
3.10.3 Изменение информации в профиле	42
3.10.4 Отмена услуг	43
3.11 Реализация функций для администратора	43
3.11.1 Удаление отзывов	43
3.11.2 Создание услуги	44
3.11.3 Аутентифицироваться	45

3.11.4 Удаление услуги	45
3.11.5 Создание талонов.....	46
3.11.6 Создание профиля для психолога	46
3.11.7 Удаление профиля для психолога.....	47
3.11.8 Изменение информации о психологе, процедуре	48
3.12 Структура клиентской части	49
3.13 Выводы по разделу	52
4 Тестирование web-приложения.....	53
4.1 Функциональное тестирование	53
4.2 Нагрузочное тестирование	59
4.3 Выводы по разделу	59
5 Руководство пользователя	60
5.1 Руководство гостя.....	60
5.1.1 Регистрация	61
5.1.2 Просмотр процедур, психологов, отзывов.....	62
5.1.3 Поиск услуг и психологов	65
5.2 Руководство пользователя	66
5.2.1 Аутентификация	66
5.2.2 Добавить отзыв	67
5.2.3 Удалить отзыв	68
5.2.4 Оформление услуги	69
5.2.5 Отмена услуг	70
5.2.6 Изменение информации в профиле	71
5.2.7 Просмотр процедур, психологов, отзывов.....	72
5.2.8 Поиск услуг и психологов	74
5.3 Руководство психолога	75
5.3.1 Аутентификация	75
5.3.2 Создание талонов.....	76
5.3.3 Изменение информации в профиле	77
5.3.4 Отмена услуг	77
5.4 Руководство администратора	78
5.4.1 Аутентифицироваться	78
5.4.2 Удаление отзывов	79
5.4.3 Создание услуги.....	80
5.4.4 Удаление услуги	81
5.4.5 Создание талонов.....	82
5.4.6 Создание профиля для психолога	83
5.4.7 Удаление профиля для психолога.....	84
5.4.8 Изменение информации о психологе, процедуре	85
5.5 Выводы по разделу	87
Заключение	88
Список используемых источников.....	89
Приложение А	90
Приложение Б.....	93
Приложение В	95

Введение

Веб-приложение – это клиент-серверное приложение, в котором клиент взаимодействует с сервером через веб-браузер. Логика работы приложения разделена между сервером и клиентом, а обмен информацией осуществляется через интернет-соединение.

Цель курсового проекта – упростить процесс оказания психологической помощи для людей, нуждающихся в ней, с помощью web-приложения «Clean Brain». Web-приложение должно поддерживать несколько ролей: гость, пользователь, психолог и администратор. Основные функции должны включать предоставление пользователям удобного доступа к информации об услугах, возможность записи на консультации, а также ознакомление с отзывами других клиентов.

Целевая аудитория web-приложения «Clean Brain» включает людей, ищущих психологическую помощь, а именно подростков, взрослых и пожилых людей, столкнувшихся с различными психологическими трудностями. Также web-приложение ориентировано на психологов, которые хотят предоставлять услуги психологического центра клиентам.

Для достижения указанной цели поставлены следующие задачи:

1. Спроектировать архитектуру приложения, обеспечивающую поддержку всех заявленных функций (рассмотрено в Разделе 2).
2. Разработать приложение по спроектированной архитектуре (рассмотрено в Разделе 3).
3. Организовать тестирование и отладку приложения для обеспечения его стабильной работы (освещено в Разделе 4).

В качестве программной платформы выбрана платформа Node.js. характеризуется событийно-ориентированной архитектурой и моделью однопоточного выполнения, что облегчит разработку web-приложения.

1 Постановка задачи и обзор аналогичных решений

1.1 Постановка задачи

Задачи web-приложения включают реализацию поддержки ролей администратора, пользователя, психолога и гостя. Гость должен иметь возможность регистрироваться, просматривать информацию о психологах, услугах и отзывах, а также осуществлять поиск услуг и специалистов. Пользователь должен уметь аутентифицироваться, взаимодействовать с информацией о психологах, услугах и отзывах, искать специалистов и услуги, оформлять и отменять записи, изменять информацию в профиле, добавлять и удалять отзывы. Психологу необходимо предоставлять функции аутентификации, изменения информации в профиле, создания талонов и отмены услуг. Администратор должен обладать правами аутентификации, удаления отзывов, создания и удаления услуг, изменения информации о специалистах и услугах, создания талонов, а также управления профилями психологов.

1.2 Обзор аналогов

1.2.1 Интернет-ресурс «INSIDE»

На сегодняшний день очень популярным альтернативным решением является интернет-ресурс «INSIDE» [1], представлено на рисунке 1.1.

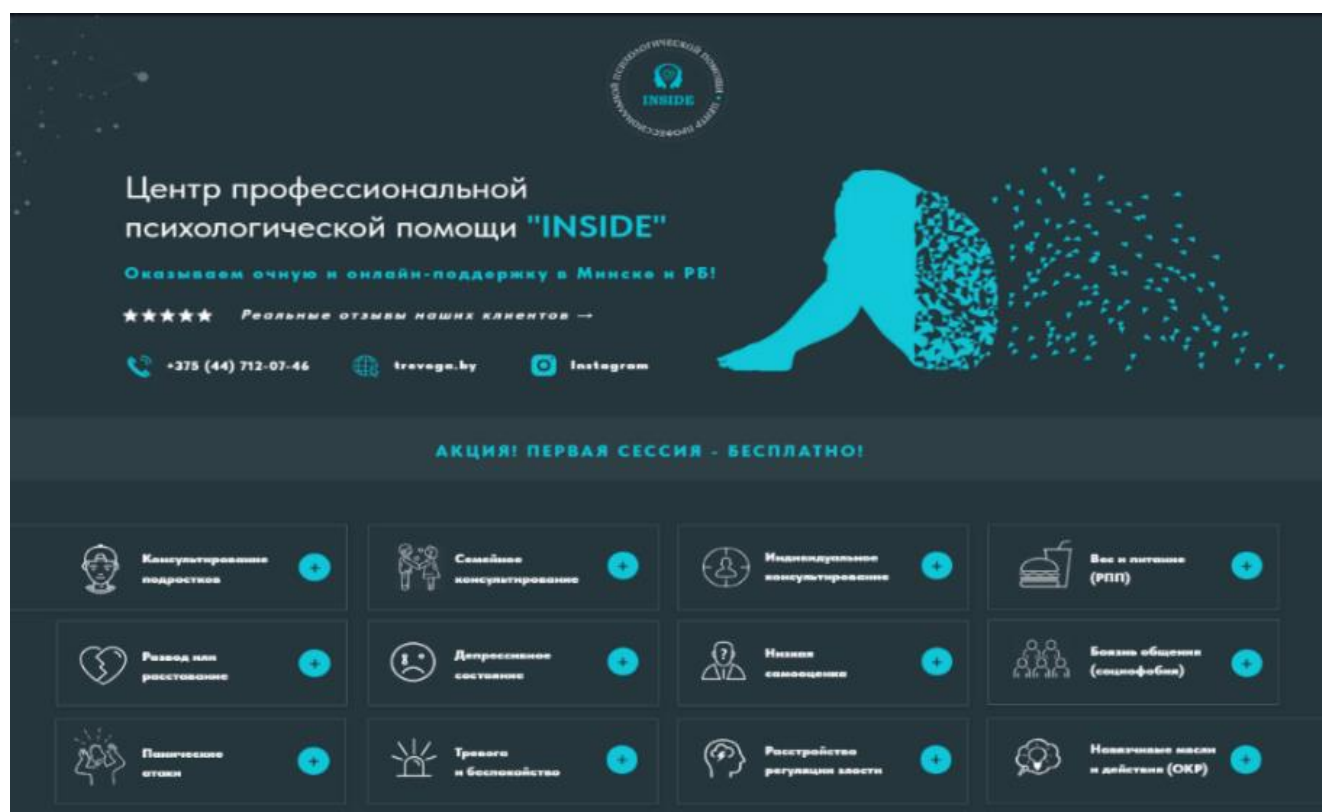


Рисунок 1.1 – Сайт «INSIDE»

Данный сайт имеет достаточно грамотный и красивый интерфейс, предоставляет удобную навигационную панель. Присутствуют отзывы от пользователей данного сайта, что положительно влияет на выбор именно этого интернет-ресурса.

При выборе направления сайт предоставляет информацию о возможной причине возникновения данной проблемы, а также возможные решения в виде предоставления формы для регистрации заказа и краткой информации о будущих этапах консультирования.

На сайте имеется пункт «О нас», который предоставляет краткую информацию об организации, а также возможность для связи при возникновении каких-либо проблем.

Рисунок 1.2 – Бронирование сеанса на сайте «INSIDE»

Говоря о недостатках сайта, при бронировании сеанса мы встречаемся с формой, где нас просят ввести свои данные. Нет возможности выбрать определенного специалиста, а лишь только его направление.

Можно лишь указать «желаемую» дату и время посещения, которые могут и не совпасть с реальным графиком нужного вам специалиста. Специалист не имеет возможности войти в личный кабинет.

Доступны только уведомления, которые отправляются на электронную почту или в мессенджеры.

Нет возможности создать аккаунт, следить за назначенными сеансами, нет информации для психологов. Присутствует поиск услуг и психологов.

1.2.2 Интернет-ресурс «Grow Up»

Следующий сайт – «Grow Up» [2], который также является популярным среди пользователей.

На рисунке 1.2 представлен интерфейс сайта.

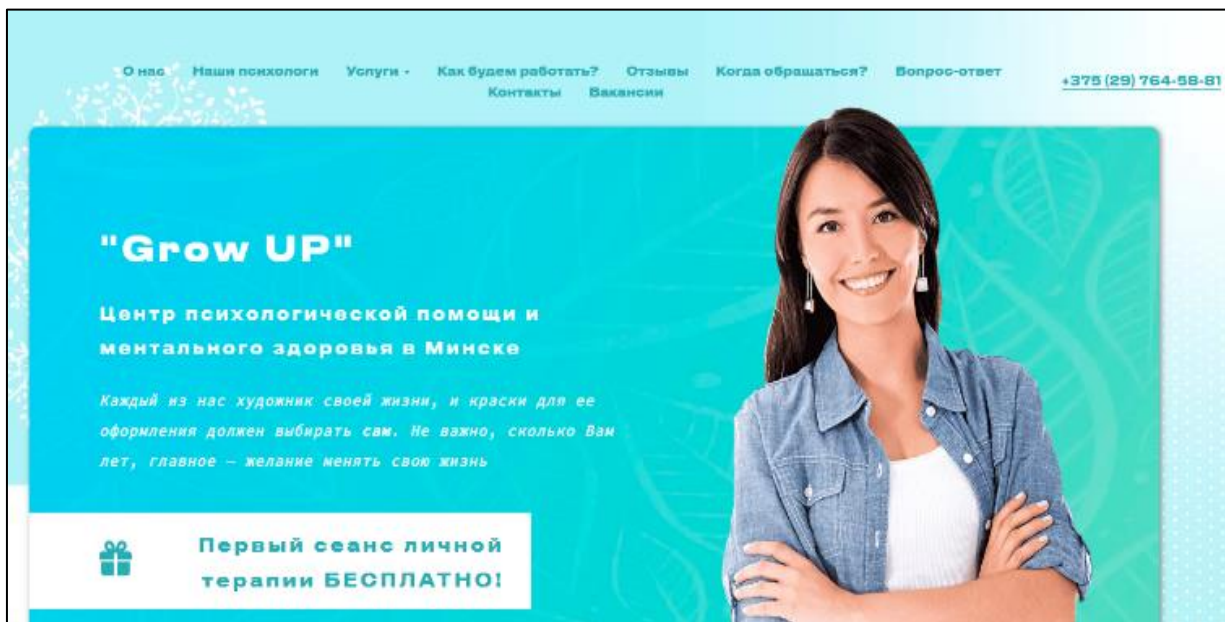


Рисунок 1.3 – Сайт «Grow Up»

Сайт предлагает пользователям простой в понимании и освоении интерфейс, что делает его привлекательнее при выборе среди пользователей. Представляет довольно краткое и достаточное описание специалистов с указанием их времени работы. Предлагаются еженедельные скидочные акции при бронировании сеанса в определенном направлении.

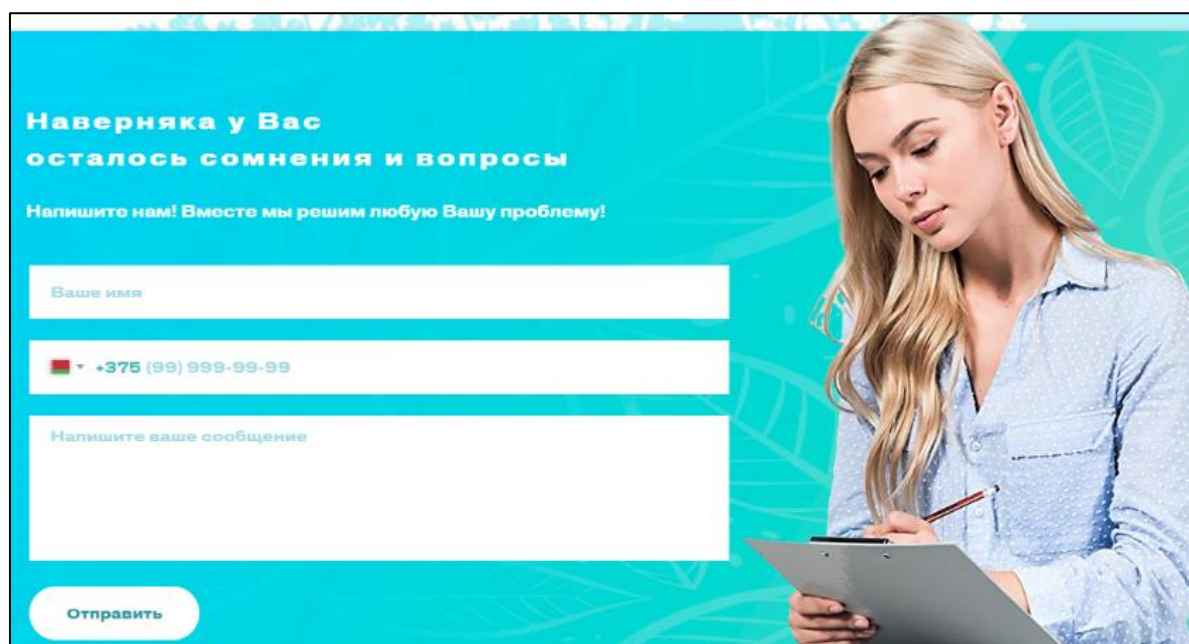


Рисунок 1.4 – Бронирование сеанса на сайте «Grow Up»

При бронировании сеанса мы встречаемся с формой для заполнения, в которой нужно указать телефон и имя клиента. Данная форма не предоставляет возможности выбрать конкретную дату, специалиста и даже желаемое направление. Уточнить данные критерии возможно, дождавшись звонка от консультанта. Нет возможности просмотреть текущие сеансы, если только вы снова не свяжетесь с администрацией данной организации. Все вышеперечисленные проблемы говорят о лишней трате свободного времени клиента и неэффективном обслуживании.

Из недостатков также можно выделить ограниченные возможности для взаимодействия с пользователями, отсутствие персонализации контента и ограниченный функционал, что может снижать привлекательность сайта для пользователей. На этом сайте отсутствует возможность создания аккаунта для обычных пользователей, а также нет информации, предназначенной для психологов. Можно добавить отзыв, искать услуги по различным направлениям.

1.2.3 Интернет-ресурс «Nora»

Еще одним не менее популярным сайтом об оказании помощи в области психологии является интернет-ресурс «Nora» [3].

Анализируемый сайт встречает нас довольно простым дизайном, что в достаточной мере отталкивает клиентов в приобретении услуг от данной организации. Имеет краткое и интуитивно понятное навигационное меню, с возможностью перейти к интересующей вас теме или узнать подробнее об предоставляемых возможностях организации.

Дизайн главной страницы сайта представлен на рисунке 1.5.

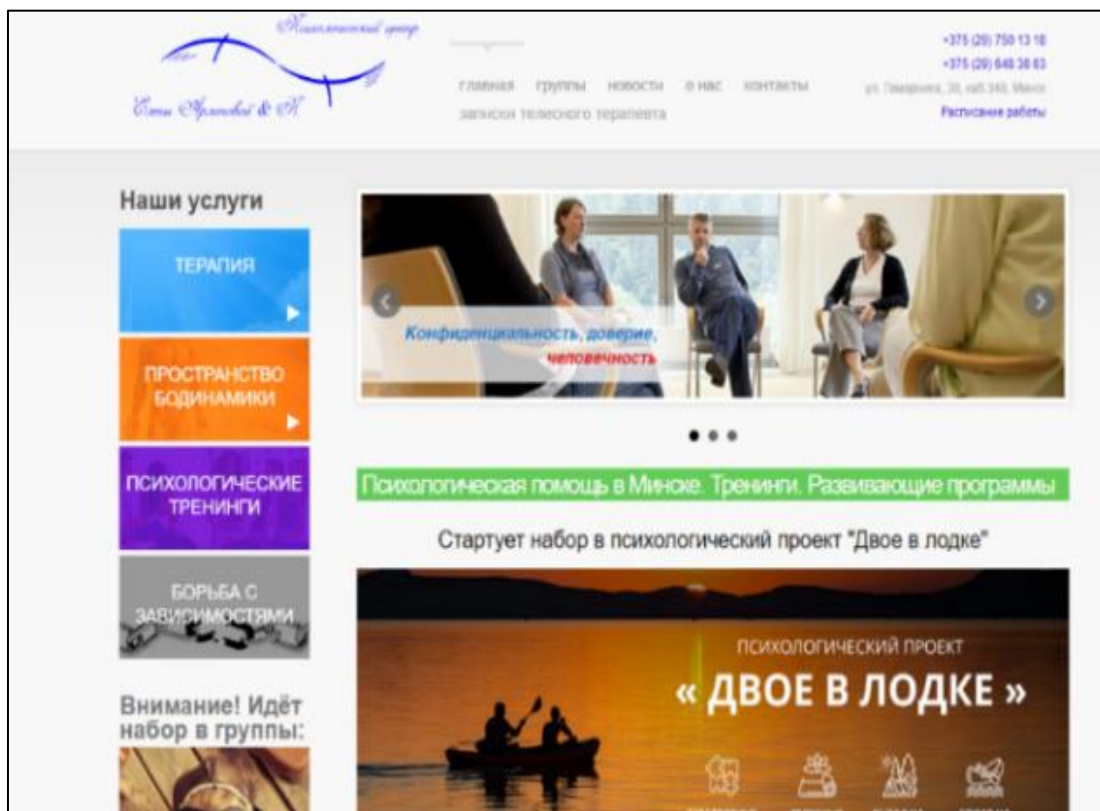


Рисунок 1.5 – Сайт «Nora»

Также на главной странице сайта присутствуют новости о недавно открытых проектах, которые предлагают клиенту принять участие в их развитии. Присутствует возможность выбора направления, в котором указывается ряд специалистов в данной области с краткой информацией.

Сайт не дает возможность создать аккаунт или заполнить форму для бронирования сеанса, все это происходит только при личном звонке специалисту.

Нет никакой информации для психологов, нет возможности просматривать запланированные сеансы.

1.3 Выводы по разделу

1. Поставленные задачи требуют разработки web-приложения с поддержкой четырех ролей: администратора, пользователя, психолога и гостя. Гостю необходимо обеспечить базовый функционал для регистрации, поиска специалистов и услуг, а также просмотра информации о психологах, услугах и отзывах. Пользователи должны иметь возможность аутентифицироваться, оформлять и отменять услуги, редактировать свой профиль, а также добавлять и удалять отзывы. Для психологов требуется функционал редактирования профиля, создания и отмены талонов. Администратор должен иметь доступ к управлению услугами, профилями специалистов, отзывами и созданию талонов.

2. Анализ существующих решений в области предоставления психологических услуг выявил как сильные стороны, так и недостатки сайтов-конкурентов. Сайты, такие как «INSIDE», «Grow Up» и «Nora», предоставляют базовый функционал, включая информацию о специалистах, направлениях работы и услуги, но ограничены в возможностях взаимодействия с пользователями. Среди ключевых недостатков отмечается отсутствие персонализации, невозможность выбора конкретного специалиста и даты, а также отсутствие личных кабинетов для пользователей и специалистов.

2 Проектирование web-приложения

2.1 Функциональность web-приложения

Функциональные возможности web-приложения представлены в диаграмме вариантов использования, представленной на рисунке 2.1.

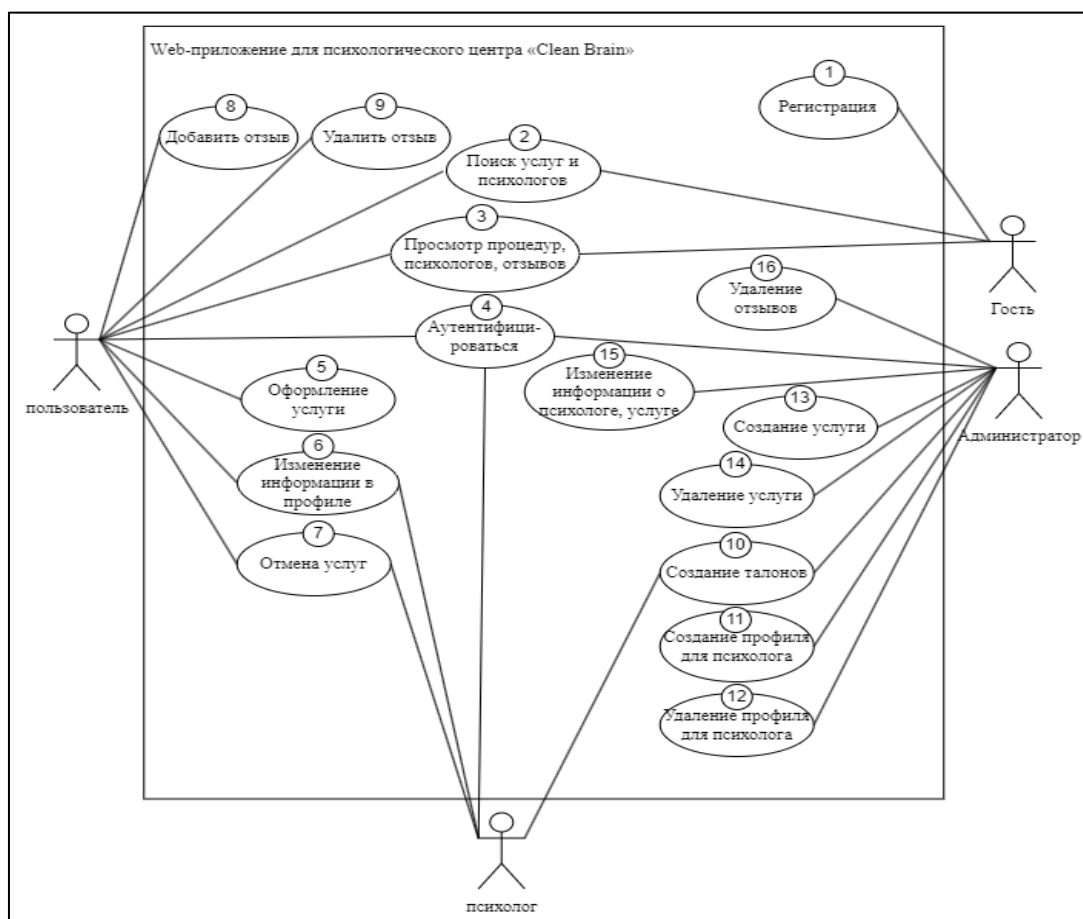


Рисунок 2.1 – Диаграмма вариантов использования web-приложения

Данная диаграмма вариантов использования наглядно демонстрирует функциональность web-приложения «Clean Brain» для различных ролей пользователей, таких как администратор, гость, психолог и пользователь. Перечень ролей и их назначение приведены в таблице 2.1.

Таблица 2.1 – Назначение ролей пользователей в web-приложении

Роль	Назначение
Гость	Просмотр информации о психологах, услугах и психологическом центре, регистрация и аутентификация.
Пользователь	Просмотр и выбор специалистов, услуг, просмотр запланированных сеансов, добавление отзывов, редактирование профиля.
Психолог	Изменение данных в личном профиле, создание талонов для записи, просмотр запланированных записей с клиентами.

Продолжение таблицы 2.1

Роль	Назначение
Администратор	Создание, удаление профилей психологов, услуг, отзывов.

Функционал гостя представлен в таблице 2.2.

Таблица 2.2 – Функционал гостя

Номер	Вариант использования	Пояснение
1	Регистрация	Гость может создать учетную запись, заполнив необходимые данные, чтобы получить доступ к полному функционалу web-приложения.
2	Поиск услуг и психологов	Гость может искать услуги и специалистов по различным критериям, таким как специализация, цена, название.
3	Просмотр процедур, психологов, отзывов	Гость может ознакомиться с описанием доступных процедур, профилями психологов и отзывами других пользователей.

Далее рассмотрим функционал, предназначенный для пользователей, представлен в таблице 2.3.

Таблица 2.3 – Функционал пользователя

Номер	Вариант использования	Пояснение
4	Аутентифицироваться	Процесс входа пользователя в систему с использованием учетных данных, таких как логин и пароль.
5	Оформление услуги	Позволяет пользователю выбрать нужную услугу, выбрать подходящего психолога, указать дату и время сеанса.
6	Изменение информации в профиле	Редактировать личные данные, включая фотографию, фамилию и имя.
7	Отмена услуги	Позволяет пользователю отменить заранее запланированный сеанс с психологом.
8	Добавить отзыв	Пользователь может добавить отзыв о психологическом центре.

Продолжение таблицы 2.3

Номер	Вариант использования	Пояснение
9	Удалить отзыв	Пользователь может удалить свой отзыв о психологическом центре, если он решит изменить мнение.
2	Поиск услуг и психологов	Гость может искать услуги и специалистов по различным критериям, таким как специализация, цена, название.
3	Просмотр процедур, психологов, отзывов	Пользователь может просматривать доступные услуги, изучать профили психологов с информацией о квалификации и опыте, а также читать отзывы других клиентов.

Функционал психолога представлен в таблице 2.4.

Таблица 2.4 – Функционал психолога

Номер	Вариант использования	Пояснение
4	Аутентифицироваться	Процесс входа психолога в систему с использованием учетных данных, таких как логин и пароль.
6	Изменение информации в профиле	Психолог может редактировать свой профиль, включая данные о квалификации, специализации и расписании.
7	Отмена услуг	Психолог может отменять запланированные.
10	Создание талонов	Психолог может добавлять талончики для записи клиентов.

Функционал администратора представлен в таблице 2.5.

Таблица 2.5 – Функционал администратора

Номер	Вариант использования	Пояснение
4	Аутентифицироваться	Процесс входа администратора в систему с использованием учетных данных, таких как логин и пароль.
10	Создание талонов	Администратор может создавать талончики для записи клиентов на сеансы с психологами.
11	Создание профиля для психолога	Администратор может создавать новые профили психологов, вводить информацию о квалификации, специализации и графике работы.

Продолжение таблицы 2.5

Номер	Вариант использования	Пояснение
12	Удаление профиля для психологов	Администратор может удалить профиль психолога.
13	Создание услуги	Администратор может добавлять новые услуги в систему, указывая описание, стоимость, название.
14	Удаление услуги	Администратор может удалить услуги из списка доступных.
15	Изменение информации о психологе, услуге	Администратор может редактировать профили психологов и информацию об услугах.
16	Удаление отзывов	Администратор может удалять отзывы, которые нарушают правила или содержат неприемлемый контент.

Таким образом для каждой роли определен набор доступных действий и возможностей.

2.2 Проектирование базы данных

Данное приложение использует базу данных с названием PsychologicalCenter. Согласно схеме вариантов использования, была разработана база данных, структура которой представлена на рисунке 2.2.

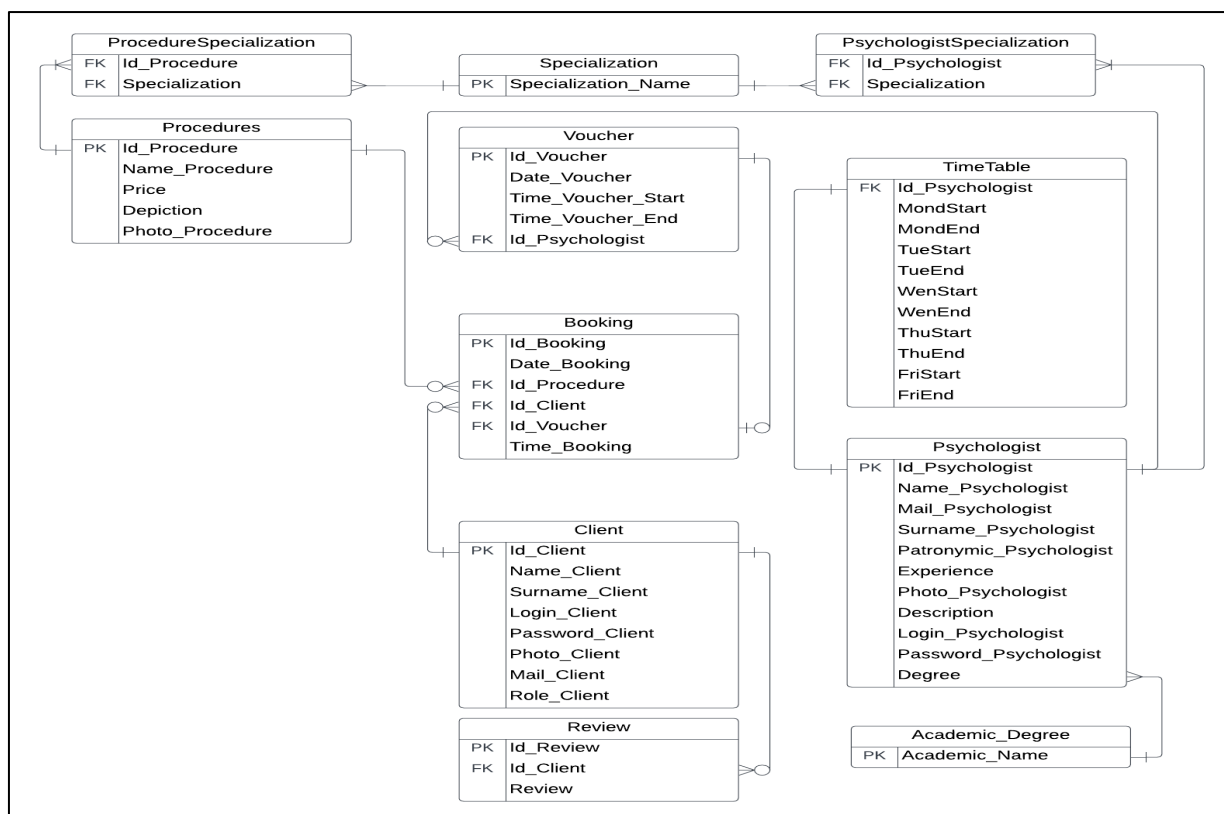


Рисунок 2.2 – Логическая схема базы данных

База данных содержит одиннадцать таблиц, назначение таблиц базы данных представлено в таблице 2.6.

Таблица 2.6 – Назначение таблиц базы данных

Таблица	Назначение
Client	Хранение информации о клиентах, включая их личные данные, логин, пароль, фото, почту и роль.
Specialization	Хранение списка специализаций, связанных с услугами и психологами.
Academic_Degree	Хранение списка академических степеней для психологов.
Psychologist	Хранение информации о психологах, включая их имя, фамилию, опыт, учёную степень, фото и описание.
Voucher	Хранение данных о талонах, включая дату, время начала и окончания, психолога и статус занятости.
Procedures	Хранение данных об услугах, включая название, цену, описание, фото.
Timetable	Хранение расписания работы психологов на каждый рабочий день недели.
Booking	Хранение данных о бронированиях услуг, включая дату, время, клиента, услугу, талон и психолога.
Review	Хранение отзывов клиентов, включая текст отзыва и связь с клиентом.
ProcedureSpecialization	Хранит информацию специализациях процедуры.
PsychologistSpecialization	Хранит информацию специализациях психолога.

Таблица Client содержит информацию о пользователях психологического центра.

Описание полей представлено в таблице 2.7.

Таблица 2.7 – Структура таблицы Client

Название	Тип данных	Описание
Id_client	uid	Идентификатор пользователя
Name_Client	varchar	Имя клиента
Surname_Client	varchar	Фамилия клиента
Login_Client	varchar	Логин пользователя
Password_Client	varchar	Пароль пользователя
Photo_Client	varchar	Фото
Mail_Client	varchar	Почта
Role_Client	varchar	Роль клиента в системе (пользователь, администратор, психолог)

Таблица Psychologist содержит информацию о психологах, включая имя, фамилию, отчество, фото психолога.

Описание полей приведено в таблице 2.8.

Таблица 2.8 – Структура таблицы Psychologist

Название	Тип данных	Описание
Id_Psychologist	uid	Идентификатор психолога
Name_Psychologist	varchar	Имя психолога
Surname_Psychologist	varchar	Фамилия психолога
Patronymic_Psychologist	varchar	Отчество психолога
Experience	smallint	Опыт работы психолога
Photo_Psychologist	varchar	Фото психолога
Description	varchar	Описание психолога
Degree	varchar	Учёная степень психолога
Login_Psychologist	varchar	Логин психолога
Password_Psychologist	varchar	Пароль психолога

Таблица Specialization содержит названия специализаций. Описание ее полей приведено в таблице 2.9.

Таблица 2.9 – Структура таблицы Specialization

Название	Тип данных	Описание
Specialization_Name	varchar	Название специализации

Таблица Academic_Degree содержит название учёных степеней. Описание ее полей приведено в таблице 2.10.

Таблица 2.10 – Структура таблицы Academic_Degree

Название	Тип данных	Описание
Academic_Name	varchar	Название учёной степени

Таблица Voucher содержит информацию о талончиках. Описание ее полей приведено в таблице 2.11.

Таблица 2.11 – Структура таблицы Voucher

Название	Тип данных	Описание
Id_Voucher	uid	Идентификатор талончика
Date_Voucher	timestamp	Дата талончика
Time_Voucher_Start	time	Начало сеанса
Time_Voucher_End	time	Конец сеанса
Id_Psychologist	uid	Идентификатор психолога

Таблица Procedures содержит информацию о процедурах, включая название, цену и описание. Описание ее полей приведено в таблице 2.12.

Таблица 2.12 – Структура таблицы Procedures

Название	Тип данных	Описание
Id_Procedure	uid	Идентификатор процедуры

Продолжение таблицы 2.12

Название	Тип данных	Описание
Name_Procedure	varchar	Название процедуры
Price	real	Стоимость процедуры
Depiction	varchar	Описание процедуры
Spezialization_Procedure	varchar	Специализация процедуры
Photo_Procedure	varchar	Фото процедуры

Таблица Timetable содержит информацию о графике работы психолога. Описание ее полей приведено в таблице 2.13.

Таблица 2.13 – Структура таблицы Timetable

Название	Тип данных	Описание
Id_Psychologist	uid	Идентификатор психолога
MondStart	time	Начало работы в понедельник
MondEnd	time	Конец работы в понедельник
TueStart	time	Начало работы во вторник
TueEnd	time	Конец работы во вторник
WenStart	time	Начало работы в среду
WenEnd	time	Конец работы в среду
ThuStart	time	Начало работы в четверг
ThuEnd	time	Конец работы в четверг
FriStart	time	Начало работы в пятницу
FriEnd	time	Конец работы в пятницу

Таблица Booking содержит информацию об оформлении сеанса. Описание ее полей приведено в таблице 2.14.

Таблица 2.14 – Структура таблицы Booking

Название	Тип данных	Описание
Id_Booking	uid	Идентификатор сеанса
Date_Booking	timestamp	Дата оформления сеанса
Id_Procedure	uid	Идентификатор процедуры
Id_Client	uid	Идентификатор клиента
Id_Psychologist	uid	Идентификатор психолога
Id_Voucher	uid	Идентификатор талончика
Time_Booking	time	Время оформления сеанса

Таблица Review содержит основную информацию об отзывах. Описание ее полей приведено в таблице 2.15.

Таблица 2.15 – Структура таблицы Review

Название	Тип данных	Описание
Id_Review	uid	Идентификатор отзыва
Id_Client	uid	Идентификатор клиента

Продолжение таблицы 2.15

Название	Тип данных	Описание
Review	varchar	Отзыв

Таблица ProcedureSpecialization содержит информацию о специализациях процедур.

Описание ее полей приведено в таблице 2.16.

Таблица 2.16 – Структура таблицы ProcedureSpecialization

Название	Тип данных	Описание
Id_Procedure	uid	Идентификатор процедуры
Specialization	varchar	Специализация процедуры

Таблица PsychologistSpecialization содержит информацию о специализациях процедур.

Описание ее полей приведено в таблице 2.17.

Таблица 2.17 – Структура таблицы PsychologistSpecialization

Название	Тип данных	Описание
Id_Psychologist	uid	Идентификатор психолога
Specialization	varchar	Специализация психолога

Назначение связей приведено в таблице 2.18.

Таблица 2.18 – Назначение связей между таблицами

Таблица источник	Связанная таблица	Тип связи
ProcedureSpecialization	Procedures	Один ко многим
ProcedureSpecialization	Specialization	Один ко многим
PsychologistSpecialization	Psychologist	Один ко многим
PsychologistSpecialization	Specialization	Один ко многим
Voucher	Psychologist	Один ко многим
Booking	Procedures	Один ко многим
Booking	Client	Один ко многим
Booking	Voucher	Один к одному
Review	Client	Один ко многим
TimeTable	Psychologist	Один к одному
Psychologist	Academic_Degree	Один ко многим

Каждая таблица имеет четко определенные поля, отражающие определенные аспекты работы психологического центра.

2.3 Архитектура web-приложения

Архитектура web-приложения представлена на рисунке 2.3.

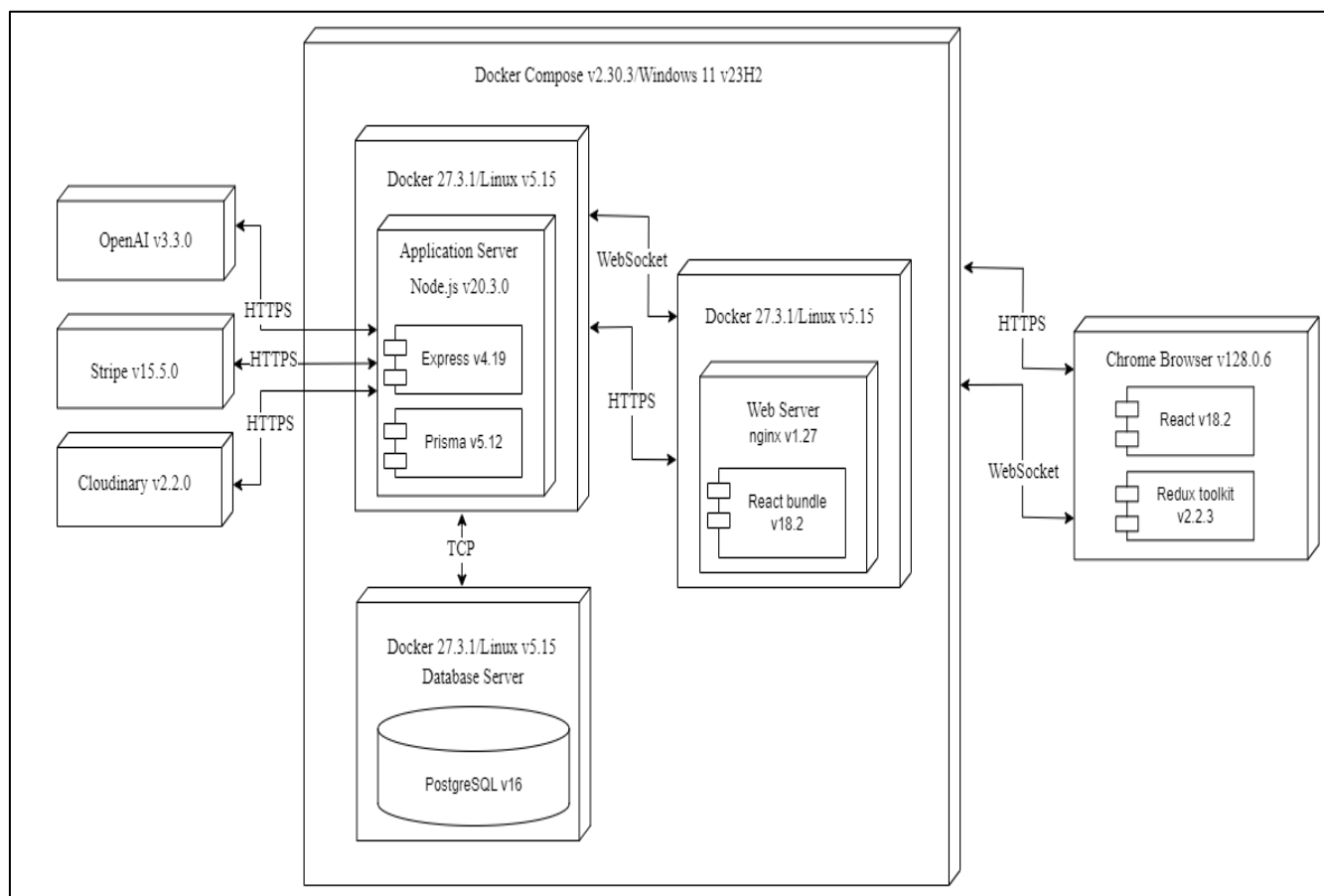


Рисунок 2.3 – Архитектура web-приложения

Пояснение назначения каждого элемента web-приложения представлено в таблице 2.19.

Таблица 2.19 – Назначение элементов архитектурной схемы web-приложения

Элемент	Назначение
Web Server (nginx)	Предоставляет доступ к статическим ресурсам фронтенд-части web-приложения.
Database Server (PostgreSQL)	Используется для хранения и предоставления доступа к данным, которые необходимы для работы web-приложения.
Application Server	Обрабатывать запросы пользователя, запрашивать данные из базы данных.
Chrome Browser	Отображать фронтэнд-часть web-приложения, отправлять запросы пользователя, отображать ответы сервера.

Продолжение таблицы 2.19

Элемент	Назначение
Stripe	Предназначенна для обработки платежей через интернет.
Cloudinary	Предназначен для хранения изображений web-приложениях.
Open AI	Предназначен для взаимодействия с пользователями с помощью ИИ.

Описание протоколов, используемых при работе web-приложений, представлено в таблице 2.20.

Таблица 2.20 – Описание используемых протоколов

Протокол	Назначение
HTTPS	Обмен данными между Web Server и Application Server, Web Server и Chrome Browser, Stripe, OpenAI, Cloudinary. Обеспечить безопасную передачу данных путём использования криптографического протокола TLS. Версия HTTP 1.1, TLS 1.2.
TCP	Обмен данными между Database Server и Application Server, а также создать виртуальное соединение между процессами.
WebSocket	Обмен данными между Web Server и Application Server, Web Server и Chrome Browser. Обеспечивать постоянное соединения используя TCP-соединение. Sec-WebSocket-Version 13.

Таким образом были рассмотрены все ключевые элементы архитектуры web-приложения.

2.4 Выводы по разделу

1. Была рассмотрена функциональность web-приложения «Clean Brain» предназначенная для всех ролей: гостя, пользователя, психолога и администратора. Гостям доступны регистрация, поиск услуг и психологов, а также просмотр информации. Пользователи могут оформлять услуги, редактировать профили и оставлять отзывы. Психологи управляют своими данными и записывают клиентов, а администраторы имеют расширенные права для создания и удаления профилей и услуг. Общее количество функций web-приложения составляет 16.

2. Рассмотрена логическая схема база данных web-приложения, которая включает 11 таблиц, каждая из которых хранит данные о пользователях, психологах, услугах и других аспектах.

3. Рассмотрена архитектура приложения. Использование современных протоколов (HTTPS, TCP, WebSocket) гарантирует безопасность и стабильность обмена данными между клиентом и сервером.

3 Реализация web-приложения

3.1 Программная платформа Node.js

Для серверной части проекта была выбрана платформа Node.js [4], которая характеризуется событийно-ориентированной архитектурой и моделью однопоточного выполнения.

В рамках проекта Node.js используется совместно с фреймворком Express.js [5], который упрощает настройку серверной части и разработку API. Express.js предоставляет гибкие инструменты для создания маршрутов и обработки запросов.

3.2 Система управления базами данных PostgreSQL

Для работы веб-приложения используется PostgreSQL [6] — мощная, надежная и масштабируемая система управления базами данных. Она поддерживает сложные SQL-запросы, транзакции с соответствием стандарту ACID, а также широкий спектр типов данных, включая JSON. Скрипт создания базы данных представлен в Приложении А.

3.3 Object-Relational Mapping (ORM) Prisma

В проекте используется ORM Prisma [7]. Prisma автоматически генерирует типизированные модели, что упрощает работу с данными и минимизирует ошибки. Она поддерживает сложные связи между таблицами, облегчает выполнение операций CRUD и ускоряет разработку благодаря интуитивно понятному API.

Сопоставление моделей, используемых в Prisma, с их реальными структурами представлено в таблице 3.1.

Таблица 3.1 – Сопоставление моделей, используемых в Prisma

Название модели	Название таблицы
Client	Client
Specialization	Specialization
Academic_Degree	Academic_Degree
Psychologist	Psychologist
Voucher	Voucher
Procedures	Procedures
Timetable	Timetable
Booking	Booking
Review	Review
ProcedureSpecialization	ProcedureSpecialization
PsychologistSpecialization	PsychologistSpecialization

Код, описывающий модель Client, приведён в листинге 3.1.

```
model Client {
  Id_client      Int    @id(map: "PK__Client__8829B11E73AEA966")
  @default(autoincrement())
  Name_Client    String  @db.VarChar
  Surname_Client String  @db.VarChar
  Login_Client   String  @db.VarChar
  Password_Client String  @db.VarChar
  Photo_Client   String?  @db.VarChar
  Mail_Client    String  @db.VarChar
  Role_Client    String  @db.VarChar
  Booking        Booking[]
  Review         Review[]
}
```

Листинг 3.1 – Модель Client

Модель Client в Prisma описывает структуру данных клиента с полями: уникальный идентификатор Id_client, Name_Client и Surname_Client для имени и фамилии, Login_Client и Password_Client для аутентификации, Photo_Client как необязательное поле для фотографии, Mail_Client для почты и Role_Client для роли пользователя. Она связана с другими сущностями через массивы Booking (бронирования) и Review (отзывы), что позволяет управлять соответствующими отношениями в базе данных.

Код, описывающий модель Specialization, приведён в листинге 3.2.

```
model Specialization {
  Spezialization_Name String @id(map:
  "PK__Speciali__4F7EC0A0C1B64DE5") @db.VarChar(60)
  ProcedureSpecialization ProcedureSpecialization[]
  PsychologistSpecialization PsychologistSpecialization[]
}
```

Листинг 3.2 – Модель Specialization

Модель Specialization в Prisma описывает специализацию с полем Spezialization_Name, которое является уникальным идентификатором (первичным ключом). Она связана с сущностями ProcedureSpecialization и PsychologistSpecialization, что позволяет управлять связями с процедурами и психологами, связанными с данной специализацией.

Код, описывающий модель Academic_Degree, приведён в листинге 3.3.

```
model Academic_Degree {
  Academic_Name String @id(map: "PK__Academic__BB1A998881085ED1")
  @db.VarChar(60)
  Psychologist Psychologist[]
}
```

Листинг 3.3 – Модель Academic_Degree

Модель Academic_Degree в Prisma описывает академическую степень с полем

Academic_Name, которое является уникальным идентификатором (первичным ключом). Модель связана с сущностью Psychologist, представляя психологов, которые имеют соответствующую академическую степень.

Код, описывающий модель Voucher, приведён в листинге 3.4.

```
model Voucher {
  Id_Voucher      Int      @id(map: "PK__Voucher__964AD8504355831B") @default(autoincrement())
  Date_Voucher    DateTime @db.Timestamp
  Time_Voucher_Start DateTime @db.Time
  Time_Voucher_End DateTime @db.Time
  Id_Psychologist Int
  Ordered         String    @db.Char(10)
  Booking         Booking[]
  Psychologist    Psychologist @relation(fields: [Id_Psychologist], references: [Id_Psychologist], onDelete: Cascade, onUpdate: NoAction, map: "FK__Voucher__Id_Psyc__5DCAEF64")
}
```

Листинг 3.4 – Модель Voucher

Модель Voucher в Prisma описывает талончик с полями: Id_Voucher (уникальный идентификатор с автогенерацией), Date_Voucher (дата по которую действует талончик), Time_Voucher_Start и Time_Voucher_End (время начала и окончания действия талончика), Id_Psychologist (идентификатор психолога), Ordered (статус талончика). Модель также включает связи с сущностями Booking (массив бронирований) и Psychologist (связь с психологом по Id_Psychologist).

Код, описывающий модель Psychologist, приведён в листинге 3.5.

```
model Psychologist {
  Id_Psychologist Int @id(map: "PK__Psycholo__282F0EC1D3E6A526") @default(autoincrement())
  Name_Psychologist String @db.VarChar
  Surname_Psychologist String @db.VarChar
  Patronymic_Psychologist String @db.VarChar
  Mail_Psychologist String @db.VarChar
  Experience Int @db.SmallInt
  Photo_Psychologist String? @db.VarChar
  Description String? @db.VarChar
  Login_Psychologist String @db.VarChar
  Password_Psychologist String @db.VarChar
  Degree String? @db.VarChar(60)
  Academic_Degree Academic_Degree? @relation(fields: [Degree], references: [Academic_Name], onUpdate: NoAction, map: "FK_Degree")
  PsychologistSpecialization PsychologistSpecialization[]
  Timetable Timetable?
  Voucher Voucher[]
}
```

Листинг 3.5 – Модель Psychologist

Модель Psychologist в Prisma включает следующие поля: Id_Psychologist (уникальный идентификатор), Name_Psychologist (имя), Surname_Psychologist

(фамилия), Patronymic_Psychologist (отчество), Mail_Psychologist (электронная почта), Experience (опыт работы), Photo_Psychologist (фотография, необязательное поле), Description (описание), Login_Psychologist (логин), Password_Psychologist (пароль), Degree (академическая степень). Связи включают Academic_Degree (ссылка на академическую степень), PsychologistSpecialization (специализации), Timetable (расписание) и Voucher (талончики).

Код, описывающий модель Review, приведён в листинге 3.6.

```
model Review {
  Id_Review Int
  @id(map: "PK__Review__7C0798D8A3A64A9E")
  @default(autoincrement())
  Id_Client Int
  Review String @db.VarChar
  Client Client @relation(fields: [Id_Client], references:
  [Id_client], onDelete: Cascade, onUpdate: NoAction, map:
  "FK__Review__Id_Clien__70DDC3D8") }
```

Листинг 3.6 – Модель Review

Модель Review в Prisma описывает отзыв с полями: Id_Review (уникальный идентификатор, первичный ключ с автогенерацией), Id_Client (идентификатор клиента), Review (текст отзыва, строка). Модель включает связь с сущностью Client, где поле Id_Client ссылается на идентификатор клиента. При удалении клиента все связанные с ним отзывы удаляются автоматически благодаря каскадному удалению (onDelete: Cascade).

Код, описывающий модель Procedures, приведён в листинге 3.7.

```
model Procedures
{
  Id_Procedure Int
  @id(map: "PK__Procedur__08F049029699B157")
  @default(autoincrement())
  Name_Procedure String @db.VarChar
  Price Float
  Depiction String @db.VarChar
  Photo_Procedure String? @db.VarChar
  Booking Booking[]
  ProcedureSpecialization ProcedureSpecialization[]
}
```

Листинг 3.7 – Модель Procedures

Модель Procedures в Prisma описывает процедуру с полями: Id_Procedure (уникальный идентификатор с автогенерацией), Name_Procedure (название процедуры), Price (стоимость процедуры), Depiction (описание процедуры), Photo_Procedure (фотография процедуры). Модель также включает связи с сущностями Booking (массив бронирований) и ProcedureSpecialization (связь с процедурами специализаций).

Код, описывающий модель Timetable, приведён в листинге 3.8.

```
model Timetable {
  Id_Psychologist Int @id(map:
"PK__Timetabl__282F0EC1C57B8539")
  MondStart DateTime? @db.Time
  MondEnd DateTime? @db.Time
  TueStart DateTime? @db.Time
  TueEnd DateTime? @db.Time
  WenStart DateTime? @db.Time
  WenEnd DateTime? @db.Time
  ThuStart DateTime? @db.Time
  ThuEnd DateTime? @db.Time
  FriStart DateTime? @db.Time
  FriEnd DateTime? @db.Time
  Psychologist Psychologist @relation(fields: [Id_Psychologist],
references: [Id_Psychologist], onDelete: Cascade, onUpdate:
NoAction, map: "FK__Timetable__Id_Ps__693CA210") }
```

Листинг 3.8 – Модель Timetable

Модель Timetable в Prisma описывает расписание психолога с полем Id_Psychologist (идентификатор психолога, первичный ключ). Для каждого дня недели (понедельник, вторник, среда, четверг, пятница) указаны время начала и окончания работы психолога (MondStart, MondEnd, TueStart, TueEnd, WenStart, WenEnd, ThuStart, ThuEnd, FriStart, FriEnd), которые хранятся в формате DateTime (тип Time). Модель включает связь с сущностью Psychologist, где поле Id_Psychologist ссылается на идентификатор психолога. Удаляя психолога, удаляется расписание (onDelete: Cascade).

Код, описывающий модель ProcedureSpecialization, приведён в листинге 3.9.

```
model ProcedureSpecialization {
  Id_Procedure Int
  Spezialization_Name String @db.VarChar(60)
  Procedures Procedures @relation(fields:
[Id_Procedure], references: [Id_Procedure], onDelete: Cascade,
onUpdate: NoAction, map: "FK_Procedures")
  Spezialization Spezialization @relation(fields:
[Spezialization_Name], references: [Spezialization_Name], onDelete:
Cascade, onUpdate: NoAction, map: "FK_Spezialization2")
  @@id([Id_Procedure, Spezialization_Name], map:
"PK__ProcedureSpecialization") }
```

Листинг 3.9 – Модель ProcedureSpecialization

Модель ProcedureSpecialization в Prisma описывает связь между процедурой и специализацией с полями: Id_Procedure (идентификатор процедуры), Spezialization_Name (название специализации). Модель включает связи с сущностями Procedures (с помощью Id_Procedure) и Spezialization (с помощью Spezialization_Name), что позволяет отображать, какие процедуры связаны с какими

специализациями. Эти два поля составляют составной первичный ключ (@@id([Id_Procedure, Spezialization_Name])).

Код, описывающий модель PsychologistSpecialization, приведён в листинге 3.10.

```
model PsychologistSpecialization {
  Id_Psychologist      Int
  Spezialization_Name String          @db.VarChar(60)
  Psychologist         Psychologist   @relation(fields:
[Id_Psychologist], references: [Id_Psychologist], onDelete: Cascade,
onUpdate: NoAction, map: "FK_Psychologist")
  Specialization       Specialization @relation(fields:
[Spezialization_Name], references: [Spezialization_Name], onDelete:
Cascade, onUpdate: NoAction, map: "FK_Spezialization")
  @@id([Id_Psychologist, Spezialization_Name], map:
"PK_PsychologistSpecialization")}
```

Листинг 3.10 – Модель PsychologistSpecialization

Модель PsychologistSpecialization в Prisma описывает связь между психологом и его специализацией с полями: Id_Psychologist (идентификатор психолога), Spezialization_Name (название специализации). Модель включает связи с сущностями Psychologist (с помощью Id_Psychologist) и Specialization (с помощью Spezialization_Name), что позволяет отображать специализацию конкретного психолога. Вместе они составляют составной первичный ключ (@@id([Id_Psychologist, Spezialization_Name])).

Код, описывающий модель Booking, приведён в листинге 3.11.

```
model Booking {
  ID_Booking      Int          @id(map: "PK__Booking__4735344230D02C71")
  @default(autoincrement()) @db.SmallInt
  Date_Booking    DateTime     @db.Timestamp
  Id_Procedure    Int
  Id_Client       Int
  Id_Voucher      Int
  Time_Booking    DateTime?    @db.Time
  Client          Client       @relation(fields: [Id_Client], references:
[Id_client], onDelete: Cascade, onUpdate: NoAction, map:
"FK__Booking__Id_Clie__6D0D32F4")
  Procedures      Procedures   @relation(fields: [Id_Procedure],
references: [Id_Procedure], onDelete: Cascade, onUpdate: NoAction,
map: "FK__Booking__Id_Proc__6C190EBB")
  Voucher         Voucher      @relation(fields: [Id_Voucher],
references: [Id_Voucher], onDelete: Cascade, onUpdate: NoAction,
map: "FK__Booking__Id_Vouc__6E01572D")}
```

Листинг 3.11 – Модель Booking

Модель Booking в Prisma описывает бронирование с полями: ID_Booking (уникальный идентификатор, первичный ключ с автогенерацией), Date_Booking

(дата бронирования), Id_Procedure (идентификатор процедуры), Id_Client (идентификатор клиента), Id_Voucher (идентификатор талончика), Time_Booking (время бронирования). Модель включает связи с сущностями Client, Procedures и Voucher, где соответствующие поля ссылаются на соответствующие идентификаторы в этих моделях. При удалении клиента, процедуры или талончика связанные бронирования удаляются автоматически благодаря каскадному удалению (onDelete: Cascade).

3.4 Программные библиотеки

В процессе разработки серверной части web-приложения для обеспечения её функциональности и повышения эффективности работы системы были использованы программные библиотеки, представленные в таблице 3.2

Таблица 3.2 – Программные библиотеки серверной части

Библиотека	Версия	Назначение
prisma/client	v5.12.1	Клиентская библиотека для работы с базой данных через Prisma ORM.
bcryptjs	v2.4.3	Библиотека для хеширования паролей и других данных.
cloudinary	v2.1.0	Библиотека для интеграции с Cloudinary, сервисом для хранения и обработки изображений и видео в облаке.
jsonwebtoken	v9.2.0	Библиотека для работы с JWT (JSON Web Tokens).
openai	v3.3.0	Библиотека для интеграции с API OpenAI, используется для работы с моделями искусственного интеллекта.
pg	v8.11.3	Библиотека для работы с PostgreSQL в Node.js.
socket.io	v4.7.5	Библиотека для реализации WebSocket-соединений в Node.js.
stripe	v17.4.0	Библиотека для интеграции с платежной системой Stripe, используется для обработки платежей.

В процессе разработки клиентской части web-приложения были задействованы программные библиотеки, представленные в таблице 3.3.

Таблица 3.3 – Программные библиотеки клиентской части

Библиотека	Версия	Назначение
mui/icons-material	v5.15.14	Библиотека иконок от Material-UI, предоставляет набор иконок, которые легко интегрировать в React-приложения.

Продолжение таблицы 3.3

Библиотека	Версия	Назначение
reduxjs/toolkit	v2.2.3	Библиотека для управления состоянием в приложениях с использованием Redux.
stripe/stripe-js	v5.2.0	Библиотека для интеграции с Stripe для работы с платежами, предоставляющая клиентскую сторону для взаимодействия с API Stripe.
axios	v1.6.8	Библиотека для выполнения HTTP-запросов.
react	v18.3.1	Библиотека для создания пользовательских интерфейсов, используется для построения компонентов и управления состоянием UI
react-router-dom	v6.22.3	Библиотека для маршрутизации в React-приложениях.
socket.io-client	v4.7.5	Клиентская библиотека для работы с WebSocket-соединениями.

Программные библиотеки позволяют упростить реализацию web-приложения. Например, jsonwebtoken [8] используется для создания и проверки JWT-токенов, обеспечивая аутентификацию и авторизацию пользователей, а Socket.IO [9] — для реализации взаимодействий между клиентом и сервером, таких как чаты и уведомления.

3.5 Сторонние сервисы Stripe, Cloudinary, OpenAI

Для успешной реализации функционала web-приложения были использованы следующие сторонние сервисы:

1. Stripe [10] — интеграция с этим сервисом позволила реализовать обработку онлайн-платежей в приложении.
2. Cloudinary [11] — используется для хранения, обработки и доставки мультимедийных файлов, таких как изображения.
3. OpenAI [12] — интеграция с OpenAI API позволяет добавить функционал на основе искусственного интеллекта в приложение. В частности, с помощью OpenAI были создан чат-бот, который может взаимодействовать с пользователями, отвечать на их вопросы, давать рекомендации.

3.6 Применение паттернов Unit of Work и Repository

В web-приложении реализованы паттерны Unit of Work и Repository, что позволяет эффективно управлять бизнес-логикой и взаимодействием с базой

данных.

1. Repository [13] — паттерн, который абстрагирует логику доступа к данным и предоставляет удобный интерфейс для работы с сущностями базы данных. В web-приложении репозитории используются для организации доступа к различным моделям данных, обеспечивая единый подход к чтению и записи информации. Реализация данного паттерна находится в приложении Б.

2. Unit of Work [14] — используется для группировки репозитория и передачи одного контекста базы данных. Реализация данного паттерна находится в приложении Б.

3.7 Структура серверной части

Основные компоненты структуры серверной части включают в себя несколько ключевых элементов, которые обеспечивают эффективную работу приложения:

1. Маршрутизаторы — управляют маршрутами и направляют запросы к соответствующим контроллерам. Реализации маршрутизатора представлена в приложении В.

2. Контроллеры — обрабатывают запросы от клиента, выполняют бизнес-логику через сервисы и возвращают ответы.

3. Сервисы — содержат бизнес-логику, обрабатывают данные и взаимодействуют с репозиториями для выполнения операций. Реализация сервиса представлена в приложении В.

4. Middleware — промежуточные обработчики, используемые для валидации данных и обеспечения безопасности.

В таблице 3.4 приведён список директорий проекта разработки web-приложения и назначение файлов, хранящихся в этих директориях.

Таблица 3.4 – Директории серверной части проекта

Директория	Назначение
controllers	Содержит контроллеры, которые обрабатывают HTTP-запросы, выполняют бизнес-логику через сервисы и возвращают ответы клиенту.
repositories	Содержит файлы реализации репозитория.
routers	Содержит маршрутизаторы, которые определяют маршруты (routes) и связывают их с соответствующими контроллерами.
services	Содержит файлы, которые реализуют бизнес-логику приложения
static	Содержит статический класс, который включает в себя утилитные методы или константы, используемые по всему web-приложению.

Продолжение таблицы 3.4

Директория	Назначение
UnitOfWork	Содержит один файл, который отвечает за координацию работы с репозиториями.
utils	Содержит файлы, которые используются для обработки ошибок, взаимодействия с OpenAI API и валидации запросов.
validators	Содержит файлы, в которых определяются схемы для валидации входящих запросов.

Таблица соответствия маршрутов контроллерам и функциям в исходном коде представлена в таблице 3.5.

Таблица 3.5 – Контроллеры и функции маршрутов

Метод	Маршрут	Контроллер	Метод контроллера
POST	/auth/sign-in-send-code	AuthController	sendEmailCode-SignIn
POST	/auth/sign-in	AuthController	signIn
POST	/auth/sign-up-send-code	AuthController	sendEmailCode-SignUp
POST	/auth/sign-up	AuthController	signUp
POST	/auth/logout	AuthController	logOut
POST	/auth/refresh	AuthController	refresh
POST	/client/update	ClientController	updateClientInfo
POST	/client/booking	ClientController	createBooking-Client
GET	/client/booking	ClientController	getBookings-Client
POST	/client/reviews	ClientController	createReview-Client
DELETE	/client/reviews/:Id_Review	ClientController	deleteReview-Client
DELETE	/client/booking/:ID_Booking	ClientController	deleteBooking-Client
POST	/chat/get-answer	ChatBotController	getAnswer
GET	/academicdegree/getAll	AcademicDegree-Controller	getAllAcademic-Degree
GET	/specialization/getAll	Specialization-Controller	getAll-Specializations
GET	/reviews/	ReviewController	getAllReviews
POST	/payment/pay	PaymentController	makePayment

Продолжение таблицы 3.5

Метод	Маршрут	Контроллер	Метод контроллера
GET	/procedures/	ProcedureController	getAll-Procedures
GET	/procedures/:Id_Procedure	ProcedureController	getProcedure
GET	/psychologists/	PsychologistController	getAll-Psychologists
POST	/psychologists/update	PsychologistController	update-Psychologist-ProfileInfo
POST	/psychologists/voucher	PsychologistController	create-Psychologist-Voucher
GET	/psychologists/booking	PsychologistController	getBookings-Psychologist
GET	/psychologists/:Id_Psychologist	PsychologistController	getPsychologist
DELETE	/psychologists/booking/:ID_Booking	PsychologistController	deleteBooking-Psychologist
POST	/admin/psychologists	AdminController	create-Psychologist
POST	/admin/psychologists/:Id_Psychologist	AdminController	update-Psychologist
POST	/admin/procedures/	AdminController	createProcedure
POST	/admin/procedures/:Id_Procedure	AdminController	update-Procedure
POST	/admin/vouchers/:Id_Psychologist	AdminController	create-Psychologist-Voucher
DELETE	/admin/psychologists/:Id_Psychologist	AdminController	delete-Psychologist
DELETE	/admin/procedures/:Id_Procedure	AdminController	deleteProcedure
DELETE	/admin/reviews/:Id_Review	AdminController	deleteReview-Client

При передаче данных между клиентом и сервером используется формат JSON (JavaScript Object Notation).

3.8 Реализация функций для гостя

3.8.1 Регистрация

Для гостя доступна регистрация, которая позволяет ему создать учетную

запись в системе. Этот процесс реализован в методе `signUp` контроллера `Auth`. Реализация метода представлена на листинге 3.12.

```
static async signUp(req, res) {
    const { email, login, password, Id_Auth, Code_Client } =
req.body;
    const { fingerprint } = req;
    try {

        const { accessToken, refreshToken, accessTokenExpiration, user
} = await AuthService.signUp( email, login, password,
fingerprint, Id_Auth, Code_Client);

        res.cookie("refreshToken", refreshToken,
COOKIE_SETTINGS.REFRESH_TOKEN);

        const outputUser = {Id_client: user.Id_client, Name_Client:
user.Name_Client, Surname_Client: user.Surname_Client, Photo_Client:
user.Photo_Client, Mail_Client: user.Mail_Client, Role_Client:
user.Role_Client}

        return res.status(200).json({ accessToken,
accessTokenExpiration, outputUser });
    } catch (err) {
        return ErrorsUtils.catchError(res, err);
    }
}
```

Листинг 3.12 – Реализация метода `signUp`

Он принимает HTTP-запрос с данными из тела, включая `email`, логин, пароль, уникальный идентификатор сессии регистрации `Id_Auth` и код подтверждения `Code_Client`, который пользователь получил на свою электронную почту. Дополнительно, из запроса извлекается параметр `fingerprint`, который используется для повышения безопасности токенов.

После получения данных метод вызывает функцию `AuthService.signUp`, которая выполняет основные операции: проверяет корректность введенного кода подтверждения и соответствие идентификатора `Id_Auth`, создает учетную запись нового пользователя в базе данных и генерирует токены доступа.

В результате выполнения функции возвращаются `accessToken` и `refreshToken`, срок действия токена доступа `accessTokenExpiration`, а также объект, содержащий данные созданного пользователя.

3.8.2 Просмотр процедур, психологов, отзывов

Гостю также доступен просмотр актуальных процедур, предлагаемых психологическим центром.

Для получения информации о процедурах реализовано два метода `getProcedure`, `getAllProcedures` контроллера `Procedure`. Реализация методов

представлена на листинге 3.13.

```
static async getAllProcedures(_,res){
    try{
        const allProcedures = await ProcedureService.getAllProcedures();
        return res.status(200).json({allProcedures});
    } catch(err){
        return ErrorUtils.catchError(res, err);
    }
}
static async getProcedure(req,res){
    try{
        const Id_Procedure = parseInt(req.params.Id_Procedure,10);
        const procedure = await ProcedureService.getProcedure(Id_Procedure);
        return res.status(200).json({procedure});
    } catch(err){
        return ErrorUtils.catchError(res, err);
    }
}
```

Листинг 3.13 – Реализация методов getProcedure, getAllProcedures

Метод getAllProcedures позволяет получить список всех доступных процедур. Он вызывает сервис ProcedureService.getAllProcedures, который извлекает данные о процедурах из базы данных, и возвращает их в формате JSON. Метод getProcedure предоставляет информацию о конкретной процедуре. Он извлекает идентификатор процедуры Id_Procedure из параметров запроса, передает его в сервис ProcedureService.getProcedure для получения данных о данной процедуре и возвращает результат в формате JSON.

Для получения отзывов реализован метод getAllReviews контроллера Review. Реализация метода представлена на листинге 3.14.

```
static async getAllReviews(_,res){
    try{

        const allReviews = await ReviewService.getAllReviews();
        return res.status(200).json({allReviews});

    } catch(err){
        return ErrorUtils.catchError(res, err);
    }
}
```

Листинг 3.14 – Реализация метода getAllReviews

Этот метод обрабатывает запрос на получение всех отзывов. В блоке try он вызывает ReviewService.getAllReviews(), чтобы получить данные о всех отзывах психологического центра, и возвращает их в формате JSON.

Для получения информации о процедурах реализовано два метода getPsychologist, getAllPsychologists контроллера Psychologist. Реализация методов

представлена на листинге 3.15.

```

    static async getAllPsychologists(_, res) {
      try {
const allPsychologists = await
PsychologistService.getAllPsychologists();
        return res.status(200).json({ allPsychologists });
      } catch (err) {
        return ErrorUtils.catchError(res, err);}}
static async getPsychologist(req, res) {
  try {
    const Id_Psychologist = parseInt(req.params.Id_Psychologist, 10);
    const psychologist = await
PsychologistService.getPsychologist(Id_Psychologist);
    return res.status(200).json({ psychologist });
  } catch (err) {
    return ErrorUtils.catchError(res, err);}}

```

Листинг 3.15 – Реализация метода getAllPsychologists, getPsychologist

Метод getAllPsychologists получает список всех психологов через PsychologistService.getAllPsychologists() и возвращает их в формате JSON. Метод getPsychologist извлекает ID психолога из параметров запроса, получает данные о конкретном психологе через PsychologistService.getPsychologist() и возвращает их клиенту в формате JSON.

3.8.3 Поиск услуг и психологов

Гостю также доступен поиск услуг и психологов по различным критериям, что позволяет упростить выбор подходящих специалистов и процедур. Реализация поиска выполняется на клиентской стороне методом searchFiltered, представлено на листинге 3.16.

```

const searchFiltered = () =>{
  const checkArrayPsychologist = procedure ?
allPsychologistsProvidingProcedure : psychologists
setFilteredPsychologists(checkArrayPsychologist.filter(psycho => {
return
StaticClass.getFullNamePsychologist(psycho.Surname_Psychologist,psyc
ho.Name_Psychologist,psycho.Patronymic_Psychologist).toLowerCase().i
ncludes(inputPsychologist.toLowerCase())&&(selectedSpecification.len
gth === 0 || selectedSpecification.every(serchedSpecilization =>
StaticClass.getArraySpecializations(psycho.PsychologistSpecializatio
n).includes(serchedSpecilization))));
}));
}

```

Листинг 3.16 – Реализация метода searchFiltered

Данный метод выполняет фильтрацию как психологов, так и процедур на

основе заданных критериев. Если выбран фильтр по процедуре, то происходит поиск среди психологов, которые предоставляют эту процедуру.

Фильтрация психологов выполняется по следующим критериям: полное имя психолога, которое проверяется на соответствие введенному запросу, и специализация психолога, которая должна совпадать с выбранными пользователем специализациями.

3.9 Реализация функций для пользователя

3.9.1 Добавить отзыв

Пользователь системы психологического центра имеет возможность оставить отзыв о предоставленных услугах или работе психолога.

Логика добавления отзыва реализована в методе `createReviewClient` контроллера `Client`. Реализация данного метода представлена на листинге 3.17.

```
static async createReviewClient(req, res) {
    const {Review} = req.body;

    const authHeader = req.headers.authorization;
    const token = authHeader?.split(" ")?.[1];

    try{
        const review = await
ClientService.createReviewClient(Review, token);

        return res.status(200).json({review});
    } catch (err){

        return ErrorUtils.catchError(res, err);
    }
}
```

Листинг 3.17 – Реализация метода `createReviewClient`

Данный метод извлекает отзыв из тела запроса. Также из заголовка запроса `Authorization` извлекается токен пользователя.

После получения данных метод вызывает функцию `ClientService.createReviewClient`, передавая ей отзыв и токен.

Сервис выполняет основную логику: проверяет токен на валидность и добавляет отзыв в базу данных. Если отзыв успешно создан, метод возвращает данные добавленного отзыва в формате JSON.

3.9.2 Оформление услуги

Пользователь системы психологического центра имеет возможность оформить услугу, выбрав подходящую процедуру и талончик психолога. Логика оформления услуги реализована в методе `createBookingClient` контроллера `Client`,

представлено на листинге 3.18.

```
static async createBookingClient(req, res) {
    const {Id_Procedure, Id_Voucher} = req.body;
    const authHeader = req.headers.authorization;
    const token = authHeader?.split(" ")?.[1];
    try{
        const booking = await
        ClientService.createBookingClient(Id_Procedure, Id_Voucher, token);

        return res.status(200).json({booking});
    } catch (err){
        return ErrorUtils.catchError(res, err);
    }
}
```

Листинг 3.18 – Реализация метода createBookingClient

Из тела запроса извлекаются идентификаторы процедуры `Id_Procedure` и идентификатор талончика `Id_Voucher`, а из заголовка `Authorization` — токен пользователя. Эти данные передаются в сервис `ClientService.createBookingClient`, который обрабатывает заявку: проверяет доступность процедуры, талончика и создаёт запись бронирования в системе и возвращает её данные.

3.9.3 Удалить отзыв

Пользователь системы психологического центра имеет возможность удалить ранее оставленный отзыв.

Логика удаления отзыва реализуется в методе `deleteReviewClient` контроллера `Client`, представлено на листинге 3.19.

```
static async deleteReviewClient(req, res) {
    const authHeader = req.headers.authorization;

    const token = authHeader?.split(" ")?.[1];
    try{
        const Id_Review = parseInt(req.params.Id_Review, 10);
        await ClientService.deleteReviewClient(Id_Review, token);

        return res.sendStatus(200);
    } catch (err){
        return ErrorUtils.catchError(res, err);
    }
}
```

Листинг 3.19 – Реализация метода deleteReviewClient

Из заголовка запроса извлекается токен для проверки прав доступа, а из параметров запроса — идентификатор отзыва. Эти данные передаются в сервисный метод `ClientService.deleteReviewClient`, который выполняет проверку валидности

токена, удостоверяется, что пользователь является автором отзыва, и удаляет его из базы данных. В случае успешного выполнения сервер возвращает статус 200.

3.9.4 Аутентифицироваться

Пользователь системы психологического центра может пройти аутентификацию для получения доступа к собственному аккаунту и данным. Процесс аутентификации реализован в методе `signIn` контроллера `Auth`, представлено на листинге 3.20.

```
static async signIn(req, res) {
  const { login, password, Id_Auth, Code_Client } = req.body;
  const { fingerprint } = req;
  try {
    const { accessToken, refreshToken, accessTokenExpiration,
user, isPsychologistAccount } = await AuthService.signIn(login,
password, fingerprint, Id_Auth, Code_Client);
    res.cookie("refreshToken", refreshToken,
COOKIE_SETTINGS.REFRESH_TOKEN);
    let outputUser;
    if (isPsychologistAccount) {
      const { Password_Psychologist, Login_Psychologist,
...outputUser1 } = user;
      outputUser = outputUser1;
    } else {
      const { Password_Client, Login_Client, ...outputUser2 } =
user;
      outputUser = outputUser2;
    }

    return res.status(200).json({ accessToken,
accessTokenExpiration, outputUser });
  } catch (err) {
    return ErrorsUtils.catchError(res, err);
  }
}
```

Листинг 3.20 – Реализация метода `signIn`

Из тела запроса извлекаются данные для входа: логин, пароль, идентификатор аутентификации `Id_Auth` и код клиента `Code_Client`. Также используется отпечаток устройства `fingerprint` для дополнительной безопасности. Эти данные передаются в сервис `AuthService.signIn`, который проверяет корректность учетных данных, генерирует токены доступа `accessToken`, `refreshToken` и возвращает информацию о пользователе.

3.9.5 Изменение информации в профиле

Пользователь может редактировать свои персональные данные в профиле, указав имя, фамилию, фото. Логика изменения информации реализована в методе

updateClientInfo контроллера Client, представлено на листинге 3.21.

```
static async updateClientInfo(req, res) {
    const { Role_Client } = req.body;
    try{
        const {Name_Client, Surname_Client, Photo_Client} = req.body;
        const authHeader = req.headers.authorization;
        const token = authHeader?.split(" ")?.[1];
        const newUser = await
ClientService.updateClientInfo(Name_Client, Surname_Client, Photo_Client, token);
        return res.status(200).json({newUser});
    } catch (err){
        return ErrorUtils.catchError(res, err);
    }
}
```

Листинг 3.21 – Реализация метода updateClientInfo

Из тела запроса извлекаются новые данные: имя Name_Client, фамилия Surname_Client и фотография Photo_Client, а также токен пользователя из заголовка Authorization для проверки прав доступа. Эти данные передаются в сервис ClientService.updateClientInfo, который обновляет профиль в базе данных.

3.9.6 Отмена услуг

Так же пользователь может отменить ранее забронированную услугу. Логика отмены услуги реализована в методе deleteBookingClient контроллера Client, представлено на листинге 3.22.

```
static async deleteBookingClient(req, res) {
    const authHeader = req.headers.authorization;
    const token = authHeader?.split(" ")?.[1];
    try{
        const ID_Booking = parseInt(req.params.ID_Booking, 10);
        await
ClientService.deleteBookingClient(ID_Booking, token);

        return res.sendStatus(200);
    } catch (err){
        return ErrorUtils.catchError(res, err);
    }
}
```

Листинг 3.22 – Реализация метода deleteBookingClient

Из параметров запроса извлекается идентификатор бронирования ID_Booking, а из заголовка Authorization — токен пользователя для проверки его прав. Эти данные передаются в сервис ClientService.deleteBookingClient, который проверяет доступ пользователя и удаляет бронирование из базы данных.

3.9.7 Просмотр процедур, психологов, отзывов

Пользователю также доступен просмотр актуальных процедур, предлагаемых психологическим центром.

Для получения информации о процедурах реализовано два метода `getProcedure`, `getAllProcedures` контроллера `Procedure`. Реализация методов представлена на листинге 3.23.

```
static async getAllProcedures(_,res){
    try{
        const allProcedures = await ProcedureService.getAllProcedures();
        return res.status(200).json({allProcedures});
    } catch(err){
        return ErrorUtils.catchError(res, err);
    }
}
static async getProcedure(req,res){
    try{
        const Id_Procedure = parseInt(req.params.Id_Procedure,10);
        const procedure = await ProcedureService.getProcedure(Id_Procedure);
        return res.status(200).json({procedure});
    } catch(err){
        return ErrorUtils.catchError(res, err);
    }
}
```

Листинг 3.23 – Реализация методов `getProcedure`, `getAllProcedures`

Метод `getAllProcedures` позволяет получить список всех доступных процедур. Он вызывает сервис `ProcedureService.getAllProcedures`, который извлекает данные о процедурах из базы данных, и возвращает их в формате JSON. Метод `getProcedure` предоставляет информацию о конкретной процедуре. Он извлекает идентификатор процедуры `Id_Procedure` из параметров запроса, передает его в сервис `ProcedureService.getProcedure` для получения данных о данной процедуре и возвращает результат в формате JSON.

Для получения отзывов реализован метод `getAllReviews` контроллера `Review`. Реализация метода представлена на листинге 3.24.

```
static async getAllReviews(_,res){
    try{
        const allReviews = await ReviewService.getAllReviews();
        return res.status(200).json({allReviews});
    } catch(err){
        return ErrorUtils.catchError(res, err);
    }
}
```

Листинг 3.24 – Реализация метода `getAllReviews`

Этот метод обрабатывает запрос на получение всех отзывов. В блоке `try` он

вызывает `ReviewService.getAllReviews()`, чтобы получить данные о всех отзывах психологического центра, и возвращает их в формате JSON.

Для получения информации о процедурах реализовано два метода `getPsychologist`, `getAllPsychologists` контроллера `Psychologist`. Реализация методов представлена на листинге 3.25.

```
static async getAllPsychologists(_, res) {
  try {
    const allPsychologists = await
    PsychologistService.getAllPsychologists();
    return res.status(200).json({ allPsychologists });
  } catch (err) {
    return ErrorUtils.catchError(res, err);}}
static async getPsychologist(req, res) {
  try {
    const Id_Psychologist = parseInt(req.params.Id_Psychologist, 10);
    const psychologist = await
    PsychologistService.getPsychologist(Id_Psychologist);
    return res.status(200).json({ psychologist });
  } catch (err) {return ErrorUtils.catchError(res, err);}}
```

Листинг 3.25 – Реализация метода `getAllPsychologists`, `getPsychologist`

Метод `getAllPsychologists` получает список всех психологов через `PsychologistService.getAllPsychologists()` и возвращает их в формате JSON. Метод `getPsychologist` извлекает ID психолога из параметров запроса, получает данные о конкретном психологе через `PsychologistService.getPsychologist()` и возвращает их клиенту в формате JSON.

3.9.8 Поиск услуг и психологов

Пользователю также доступен поиск услуг и психологов по различным критериям, что позволяет упростить выбор подходящих специалистов и процедур. Реализация поиска выполняется на клиентской стороне методом `searchFiltered`, представлено на листинге 3.26.

```
const searchFiltered = () =>{
  const checkArrayPsychologist = procedure ?
  allPsychologistsProvidingProcedure : psychologists
  setFilteredPsychologists(checkArrayPsychologist.filter(psycho => {
    return
    StaticClass.getFullNamePsychologist(psycho.Surname_Psychologist,psycho.Name_Psychologist,psycho.Patronymic_Psychologist).toLowerCase().includes(inputPsychologist.toLowerCase())&&(selectedSpecification.length === 0 || selectedSpecification.every(serchedSpecilization =>
    StaticClass.getArraySpecializations(psycho.PsychologistSpecialization).includes(serchedSpecilization))));
  }));}
```

Листинг 3.26 – Реализация метода `searchFiltered`

Данный метод выполняет фильтрацию как психологов, так и процедур на основе заданных критериев. Если выбран фильтр по процедуре, то происходит поиск среди психологов, которые предоставляют эту процедуру.

Фильтрация психологов выполняется по следующим критериям: полное имя психолога, которое проверяется на соответствие введенному запросу, и специализация психолога, которая должна совпадать с выбранными пользователем специализациями.

3.10 Реализация функций для психолога

3.10.1 Аутентифицироваться

Психолог, как и пользователь, может пройти процесс аутентификации в системе психологического центра. Этот процесс реализован в методе `signIn` контроллера `Auth`, представлено на листинге 3.27.

```
static async signIn(req, res) {
  const { login, password, Id_Auth, Code_Client } = req.body;
  const { fingerprint } = req;

  try {
    const { accessToken, refreshToken, accessTokenExpiration,
    user, isPsychologistAccount } = await AuthService.signIn(login,
    password, fingerprint, Id_Auth, Code_Client);
    res.cookie("refreshToken", refreshToken,
    COOKIE_SETTINGS.REFRESH_TOKEN);
    let outputUser;

    if (isPsychologistAccount) {
      const { Password_Psychologist, Login_Psychologist,
      ...outputUser1 } = user;
      outputUser = outputUser1;
    } else {
      const { Password_Client, Login_Client, ...outputUser2 } =
      user;
      outputUser = outputUser2;
    }

    return res.status(200).json({ accessToken,
    accessTokenExpiration, outputUser });
  } catch (err) {
    return ErrorsUtils.catchError(res, err);
  }
}
```

Листинг 3.27 – Реализация метода `signIn`

Из тела запроса передаются данные: логин, пароль, идентификатор аутентификации `Id_Auth` и код клиента `Code_Client`, а также отпечаток устройства `fingerprint`.

Эти данные передаются в сервис `AuthService.signIn`, который проверяет их корректность. Метод возвращает токен доступа `accessToken`, срок его действия, токен обновления `refreshToken` и данные профиля психолога.

3.10.2 Создание талонов

Психолог может создавать талоны для записи клиентов. Этот функционал реализован в методе `createPsychologistVoucher` контроллера `Psychologist`, представлено на листинге 3.28.

```
static async createPsychologistVoucher(req, res) {
  try {
    const { Id_client } = req.user;
    await VoucherService.createVouchers(Id_client);
    res.sendStatus(200);
  } catch (err) { return ErrorUtils.catchError(res, err); }}
```

Листинг 3.28 – Реализация метода `createPsychologistVoucher`

Идентификатор психолога (`Id_client`) извлекается из запроса. Затем вызывается метод `createVouchers` сервиса `VoucherService`, который автоматически генерирует талоны на основе текущего расписания психолога.

3.10.3 Изменение информации в профиле

Психолог, так же как и пользователь, может изменять информацию в своём профиле. Логика изменения информации реализована в методе `updatePsychologistProfileInfo` контроллера `Psychologist`, представлено на листинге 3.29.

```
static async updatePsychologistProfileInfo(req, res) {
  try { const { Id_Psychologist } = req.body;
    if (req.user.Id_client == Id_Psychologist) {
      const psychologist = await
PsychologistService.updatePsychologist(req.body);
      psychologist.Role_Client = "Psychologist";
      await
TimeTableService.updateTimeTable(psychologist.Id_Psychologist, req.bo
dy.timetable);
      await
PsychologistSpecializationService.updatePsychologistSpecialization(p
sychologist.Id_Psychologist, req.body.Specialization);
      await
VoucherService.createVouchers(psychologist.Id_Psychologist);
      res.status(200).json({psychologist});
    } else throw new Forbidden("Отказано в доступе");
  } catch (err) { return ErrorUtils.catchError(res, err); }}
```

Листинг 3.29 – Реализация метода `updatePsychologistProfileInfo`

Из запроса извлекаются следующие данные: идентификатор психолога `Id_Psychologist`. В теле запроса также содержатся данные, такие как новое расписание психолога `timetable`, специализация `Specialization` и другие данные профиля психолога, включая ФИО, научную степень, опыт, описание и направления деятельности. Далее обновляются данные профиля психолога через сервис `PsychologistService.updatePsychologist`, включая расписание через `TimeTableService.updateTimeTable` и специализацию через `PsychologistSpecializationService.updatePsychologistSpecialization`. После этого, метод вызывает `VoucherService.createVouchers`, чтобы сгенерировать новые талоны, и возвращает обновлённую информацию о психологе.

3.10.4 Отмена услуг

Психолог в системе психологического центра может отменить забронированные услуги. Логика изменения информации реализована в методе `deleteBookingPsychologist` контроллера `Psychologist`, представлено на листинге 3.30.

```
static async deleteBookingPsychologist(req, res) {
    const {Id_client, Role_Client} = req.user;

    try{
        const ID_Booking = parseInt(req.params.ID_Booking, 10);

        await
PsychologistService.deleteBookingPsychologist(ID_Booking, Id_client, Role_Client);

        return res.sendStatus(200);
    } catch (err){
        return ErrorUtils.catchError(res, err);
    }
}
```

Листинг 3.30 – Реализация метода `deleteBookingPsychologist`

Из запроса извлекаются идентификатор клиента `Id_client` и роль клиента `Role_Client`, а также идентификатор бронирования `ID_Booking` из параметров запроса. После этого вызывается сервис `PsychologistService.deleteBookingPsychologist`, который выполняет удаление бронирования. Кроме того, пользователю, у которого было отменено бронирование, на почту отправляется письмо об отмене услуги психологом.

3.11 Реализация функций для администратора

3.11.1 Удаление отзывов

Администратор системы имеет возможность удалять отзывы, оставленные пользователями о психологах или услугах. Это позволяет удалять неподобающие

отзывы. Логика удаления отзыва реализована в методе `deleteReviewClient` контроллера `Admin`, представлено на листинге 3.31.

```
static async deleteReviewClient(req, res) {
    try {
        const Id_Review = parseInt(req.params.Id_Review, 10);
        await
        ReviewService.deleteReviewWithoutClient(Id_Review);

        return res.sendStatus(200);
    } catch (err) {
        return ErrorUtils.catchError(res, err);
    }
}
```

Листинг 3.31 – Реализация метода `deleteReviewClient`

Извлекается идентификатор отзыва `Id_Review` из параметров запроса. После извлечения идентификатора, вызывается сервис `ReviewService.deleteReviewWithoutClient`, который выполняет удаление отзыва из базы данных.

3.11.2 Создание услуги

Администратор в системе психологического центра имеет возможность создавать новые услуги. Логика создания отзыва реализована в методе `createProcedure` контроллера `Admin`, представлено на листинге 3.32.

```
static async createProcedure(req, res) {
    try {
        const procedure = await
        ProcedureService.createProcedure(req.body);
        await
        ProcedureSpecializationService.createProcedureSpecialization(procedure.Id_Procedure, req.body.Specialization);
        res.status(200).json({procedure});
    } catch (err) {
        return ErrorUtils.catchError(res, err);
    }
}
```

Листинг 3.32 – Реализация метода `createProcedure`

Данные о процедуре извлекаются из тела запроса, такие как название, описание, цена и массив специализаций, далее вызывается метод сервиса `ProcedureService.createProcedure`, а затем связывает её с соответствующими специализациями через сервис `ProcedureSpecializationService.createProcedureSpecialization`. После успешного создания процедуры и специализации, метод возвращает данные новой процедуры.

3.11.3 Аутентифицироваться

У администратора, так же как и у пользователя, есть процесс аутентификации. Аутентификация администратора выполняется с использованием того же метода `signIn` контроллера `Auth`, который используется для пользователей, представлено на листинге 3.33.

```
static async signIn(req, res) {
  const { login, password, Id_Auth, Code_Client } = req.body;
  const { fingerprint } = req;

  try {
    const { accessToken, refreshToken, accessTokenExpiration,
    user, isPsychologistAccount } = await AuthService.signIn(login,
    password, fingerprint, Id_Auth, Code_Client);
    res.cookie("refreshToken", refreshToken,
    COOKIE_SETTINGS.REFRESH_TOKEN);
    let outputUser;
    if (isPsychologistAccount) {
      const { Password_Psychologist, Login_Psychologist,
      ...outputUser1 } = user;

      outputUser = outputUser1;
    } else {
      const { Password_Client, Login_Client, ...outputUser2 } =
user;
      outputUser = outputUser2;
    }
    return res.status(200).json({ accessToken,
    accessTokenExpiration, outputUser });
  } catch (err) {

    return ErrorsUtils.catchError(res, err);
  }
}
```

Листинг 3.33 – Реализация метода `signIn`

Из тела запроса извлекаются данные для входа: логин, пароль, идентификатор аутентификации `Id_Auth` и код клиента `Code_Client`. Также используется отпечаток устройства `fingerprint` для дополнительной безопасности.

Эти данные передаются в сервис `AuthService.signIn`, который проверяет корректность учетных данных, генерирует токены доступа `accessToken`, `refreshToken` и возвращает информацию о пользователе.

3.11.4 Удаление услуги

Администратор системы имеет возможность удалять услуги, которые больше не актуальны или были добавлены ошибочно.

Логика удаления услуги реализована в методе `deleteProcedure` контроллера

Admin, представлена на листинге 3.34.

```
static async deleteProcedure(req, res) {
    try {
        const Id_Procedure = parseInt(req.params.Id_Procedure, 10);
        await ProcedureService.deleteProcedure(Id_Procedure);
        await
        ProcedureSpecializationService.deleteProcedureSpecialization(Id_Proc
        edure);
        res.sendStatus(200);
    } catch (err) {
        return ErrorUtils.catchError(res, err);
    }
}
```

Листинг 3.34 – Реализация метода deleteProcedure

Извлекается идентификатор услуги Id_Procedure из параметров запроса. После этого вызываются методы ProcedureService.deleteProcedure и ProcedureSpecializationService.deleteProcedureSpecialization, которые выполняют удаление услуги из базы данных.

3.11.5 Создание талонов

Администратор системы психологического центра имеет возможность создавать талоны для записи на услуги психологам. Логика создания талонов реализована в методе createPsychologistVoucher контроллера Admin, представлена на листинге 3.35.

```
static async createPsychologistVoucher(req, res) {
    try {
        const {Id_Psychologist} = req.body;
        await VoucherService.createVouchers(Id_Psychologist);
        res.sendStatus(200);
    } catch (err) {
        return ErrorUtils.catchError(res, err);
    }
}
```

Листинг 3.35 – Реализация метода createPsychologistVoucher

Идентификатор психолога Id_Psychologist, извлекаются из тела запроса. Затем вызывается сервис VoucherService.createVouchers, который генерирует талоны для указанного психолога, учитывая его расписание и доступность.

3.11.6 Создание профиля для психолога

Администратор может создать профиль для нового психолога. Это включает в себя создание учетной записи психолога, а также настройку его расписания,

специализаций и талонов для записи. Логика создания профиля реализована в методе `createPsychologist` контроллера `Admin`, представленном на листинге 3.36.

```
static async createPsychologist(req, res) {
    try {
        const psychologist = await
PsychologistService.createPsychologist(req.body);
        await
TimeTableService.createTimeTable(psychologist.Id_Psychologist, req.bo
dy.timetable);
        await
PsychologistSpecializationService.createPsychologistSpecialization(p
sychologist.Id_Psychologist, req.body.Specialization);
        await
VoucherService.createVouchers(psychologist.Id_Psychologist);
        res.status(200).json({psychologist});
    } catch (err) {
        return ErrorUtils.catchError(res, err);
    }
}
```

Листинг 3.36 – Реализация метода `createPsychologist`

Метод извлекает данные о психологе из запроса, такие данные, как фамилия, имя, отчество, научная степень, расписание, профессиональный опыт, описание и направления деятельности. Создает учетную запись через сервис `PsychologistService.createPsychologist`. Затем на основе этих данных создается расписание с помощью `TimeTableService.createTimeTable`, привязываются специализации через `PsychologistSpecializationService.createPsychologistSpecialization` и генерируются талоны для записи через `VoucherService.createVouchers`. После успешного выполнения метод возвращает информацию о созданном психологе.

3.11.7 Удаление профиля для психолога

Администратор системы имеет возможность удалить профиль психолога. Логика удаления профиля психолога реализована в методе `deletePsychologist` контроллера `Admin`, представленного на листинге 3.37.

```
static async deletePsychologist(req, res) {
    try {
        const Id_Psychologist = parseInt(req.params.Id_Psychologist, 10);
        await
PsychologistService.deletePsychologist(Id_Psychologist);
        res.sendStatus(200);
    } catch (err) {
        return ErrorUtils.catchError(res, err);
    }
}
```

Листинг 3.37 – Реализация метода `deletePsychologist`

Метод извлекает идентификатор психолога `Id_Psychologist` из параметров запроса. После этого вызывается метод сервиса `PsychologistService.deletePsychologist`, который удаляет профиль психолога из базы данных.

3.11.8 Изменение информации о психологе, процедуре

Администратор системы имеет возможность обновить информацию о психологе.

Логика изменения информации о психологе реализована в методе `updatePsychologist` контроллера `Admin`, представленном на листинге 3.38.

```
static async updatePsychologist(req, res) {
    try{

        const psychologist = await
PsychologistService.updatePsychologist(req.body);

        await
TimeTableService.updateTimeTable(psychologist.Id_Psychologist, req.bo
dy.timetable);

        await
PsychologistSpecializationService.updatePsychologistSpecialization(p
sychologist.Id_Psychologist, req.body.Specialization);

        await
VoucherService.createVouchers(psychologist.Id_Psychologist);
        res.status(200).json({psychologist});

    } catch(err) {
        return ErrorUtils.catchError(res, err);
    }
}
```

Листинг 3.38 – Реализация метода `updatePsychologist`

Метод извлекает данные психолога из тела запроса и передает их в сервис `PsychologistService.updatePsychologist` для обновления информации о психологе. Затем обновляется расписание через `TimeTableService.updateTimeTable`, специализации через `PsychologistSpecializationService.updatePsychologistSpecialization`, и генерируются новые талоны с помощью `VoucherService.createVouchers`.

После успешного выполнения операций метод возвращает обновленную информацию о психологе.

Администратор также имеет возможность обновить информацию о существующей услуге, включая ее название, описание, цену и специализации.

Логика изменения информации о услуге реализована в методе `updateProcedure` контроллера `Admin`, который отвечает за обработку запроса на изменение данных услуги.

Логика изменения информации о услуге представлена на листинге 3.39.

```
static async updateProcedure(req, res) {
  try {
    const procedure = await
ProcedureService.updateProcedure(req.body);
    await
ProcedureSpecializationService.updateProcedureSpecialization(procedure.Id_Procedure, req.body.Specialization);
    res.status(200).json({procedure});
  } catch(err) {
    return ErrorUtils.catchError(res, err);
  }
}
```

Листинг 3.39 – Реализация метода updateProcedure

Метод извлекает данные об услуге из тела запроса и передает их в сервис ProcedureService.updateProcedure для обновления информации о процедуре. Затем через сервис ProcedureSpecializationService.updateProcedureSpecialization обновляются специализации, связанные с услугой. После успешного выполнения операций метод возвращает обновленную информацию о услуге.

3.12 Структура клиентской части

Клиентская часть приложения реализована с использованием компонентного подхода. Основная логика и элементы пользовательского интерфейса размещены в директории src. Директории представлены в таблице 3.4.

Таблица 3.6 – Основные директории проекта в папке src и их назначение

Директория	Назначение
components	Включает в себя переиспользуемые React-компоненты, предназначенные для создания элементов пользовательского интерфейса web-приложения.
config	Хранит конфигурационные файлы и глобальные параметры.
context	Содержит файлы, предназначенные для управления токенами доступа пользователя
fetch	Содержит модули, которые обрабатывают запросы к серверу, обеспечивая взаимодействие клиентской части приложения с API.
services	Включает модули для взаимодействия с API внешних сервисов.
states	содержит файлы, связанные с управлением состоянием приложения с использованием Redux Toolkit.

Продолжение таблицы 3.6

Директория	Назначение
static	Содержит статический класс, который включает в себя утилитные методы или константы, используемые по всему web-приложению.
utils	содержит файлы, отвечающие за взаимодействие с сервером через WebSocket.

Таблица соответствия маршрутов и компонентов страниц представлена в таблице 3.7.

Таблица 3.7 – Маршруты и компоненты страниц

Компонент страницы	Маршрут	Роль	Назначение компонента
HomePage	/home	Администратор, психолог, гость, пользователь	Основная страница приложения, которая служит отправной точкой для навигации по другим разделам и предоставляет общую информацию о web-приложении.
PsychologistPage	/psychologist	Администратор, психолог, гость, пользователь	Отображение информации о психологах психологического центра.
ProcedurePage	/procedure	Администратор, психолог, гость, пользователь	Отображение информации обо всех услугах психологического центра.
ReviewPage	/review	Администратор, психолог, гость, пользователь	Отображение отзывов оставленных пользователями психологического центра.
ProfilePage	/profile	Психолог, пользователь	Отображение информации о пользователе или психологе.
LogPage	/login	Гость	Предоставить форму авторизации.
RegPage	/register	Гость	Предоставить форму регистрации.

Помимо маршрутов и страниц, приложение включает множество компонентов, которые обеспечивают функциональность и удобство использования клиентской части.

В таблице 3.8 представлено описание всех остальных компонентов

приложения и их назначение.

Таблица 3.8 – Описание компонентов

Компонент	Назначение
Navigation	Используется для обеспечения навигации и маршрутизации.
BodyHome	Используется для отображения информации о психологическом центре.
BodyProcedure	Используется для вывода карточек услуг, а так же предоставления возможности фильтрации.
BodyProcedureProfile	Используется для отображения информации об услуге.
BodyProfile	Используется для отображения информации о текущем пользователе.
BodyPsychologist	Используется для вывода карточек психологов, а так же предоставления возможности фильтрации.
BodyPsychologistProfile	Используется для отображения информации о психологе.
BodyReview	Используется для отображения карточек отзывов.
BodyReviewCreate	Используется для отображении формы добавления отзыва.
BodySession	Используется для вывода сеансов пользователя или психолога.
BodyBooking	Используется для отображения формы бронирования.
CardProcedure	Используется для отображения карточки услуги.
CardPsychologist	Используется для отображения карточки психолога.
CardReview	Используется для отображения карточки отзыва.
CardSession	Используется для отображения карточки сеанса.
Footer	Используется для отображения футера web-приложения.
Header	Используется для отображения хедера web-приложения.
MultiplySelect	Используется для отображения выпадающего списка.
PopUp	Используется для отображения модального окна.

Компоненты и страницы реализуют все необходимые функции для различных ролей.

3.13 Выводы по разделу

1. В web-приложении использована программная платформа Node.js с применением фреймворка Express.js. Это позволило реализовать в сервер с удобной маршрутизацией и эффективно обрабатывать HTTP-запросы.

2. Для хранения данных использовалась СУБД PostgreSQL, которая гарантировала надёжное управление данными, поддержку транзакций. Для упрощения взаимодействия с базой данных применялась ORM Prisma, которая автоматизировала создание моделей, облегчила выполнение операций с данными.

3. Рассмотрены сторонние сервисы, которые помогут расширить функциональность web-приложения, включая поддержку искусственного интеллекта, оплаты и облачного хранения данных.

4. Рассмотрена структура приложения, которая базируется на модульном подходе с использованием современных программных библиотек для клиентской и серверной частей.

5. Взаимодействие с базой данных было организовано с использованием архитектурных паттернов Unit of Work и Repository, что позволило повысить читаемость и масштабируемость кода. Это дало возможность легко добавлять новые функции и поддерживать текущий функционал.

6. Реализованы все функции для все ролей психологического центра: гостя, пользователя, психолога и администратора. Количество функций web-приложения составило 16.

4 Тестирование web-приложения

4.1 Функциональное тестирование

Для проверки корректности работы всех функций разработанного web-приложения было проведено ручное тестирование, описание и итоги которого представлены в таблице 4.1.

Таблица 4.1 – Описание тестирования функций web-приложения

Номер	Функция web-приложения	Описание тестирования	Итог тестирования функции
1	Регистрация	<p>Действие: отправить POST запрос на адрес /auth/sign-up/, указав в теле запроса логин пользователя (login) со значением «nik», адрес электронной почты (email) «nikitakar@gmail.com», пароль (password) со значением «1122», идентификатор регистрации (Id_Auth) со значением «22» и код отправленный на почту (Code_Client) со значением «353643».</p> <p>Ожидаемый результат: сервер должен вернуть ответ в формате JSON с токеном доступа, временем его истечения и данными пользователя.</p>	Успешно.
2	Поиск услуг	<p>Действие: необходимо перейти на страницу по адресу /procedure/. В секции с фильтрации, необходимо из списка выбрать направление «Клиническая», ввести цену равную «100», и в поле «Название» ввести «сказкотерапия».</p> <p>Ожидаемый результат: вывод процедур, подходящих по данным критериям.</p>	Успешно.
2	Поиск психологов	<p>Действие: необходимо перейти на страницу по адресу /psychologist/. В секции с фильтрации, необходимо из списка выбрать направление «Клиническая», в поле «ФИО» ввести «Каребо».</p> <p>Ожидаемый результат: вывод психологов, подходящих по критериям.</p>	Успешно.

Продолжение таблицы 4.1

Номер	Функция web-приложения	Описание тестирования	Итог тестирования функции
3	Просмотр психологов	Действие: отправить GET запрос на адрес /psychologists/. Ожидаемый результат: сервер должен вернуть ответ в формате JSON с перечнем всех доступных психологов.	Успешно.
3	Просмотр процедур	Действие: отправить GET запрос на адрес /procedures/. Ожидаемый результат: сервер должен вернуть ответ в формате JSON с перечнем всех доступных процедур психологического центра.	Успешно.
3	Просмотр отзывов	Действие: отправить GET запрос на адрес /reviews/. Ожидаемый результат: сервер должен вернуть ответ в формате JSON с перечнем всех отзывов пользователей	Успешно.
4	Аутентифицироваться	Действие: отправить POST запрос на адрес /auth/sign-in/, указав в теле запроса логин пользователя (email) со значением «nikitakar@gmail.com», пароль (password) со значением «1122», идентификатор аутентификации (Id_Auth) со значением «23» и код отправленный на почту (Code_Client) со значением «435313». Ожидаемый результат: сервер должен вернуть ответ в формате JSON с токеном доступа, временем его истечения и данными пользователя.	Успешно.
5	Оформление услуги	Действие: отправить POST запрос на адрес /client/booking/, указав в теле запроса идентификатор процедуры (Id_Procedure) «4», идентификатор талончика (Id_Voucher) «46». В запросе необходимо добавить заголовок «Authorization», в котором передается токен доступа пользователя. Ожидаемый результат: сервер должен вернуть в формате JSON информацию о бронировании (booking).	Успешно.

Продолжение таблицы 4.1

Номер	Функция web-приложения	Описание тестирования	Итог тестирования функции
6	Изменение информации в профиле для пользователя	<p>Действие: отправить POST запрос на адрес /client/update/, указав в теле запроса роль пользователя (Role_Client) со значением «Client», имя (Name_Client) со значением «Nik», фамилию (Surname_Client) со значением «Kar», фото клиента (Photo_Client). В запросе необходимо добавить заголовок «Authorization», в котором передается токен доступа пользователя.</p> <p>Ожидаемый результат: сервер должен вернуть в формате JSON информацию о пользователе (newUser).</p>	Успешно.
6	Изменение информации в профиле для психолога	<p>Действие: отправить POST запрос на адрес /client/update/, указав в теле запроса роль пользователя (Role_Client) со значением «Psychologist», имя (Name_Psychologist) со значением «Никита», фамилию (Surname_Psychologist) со значением «Каребо», отчество (Patronymic_Psychologist) со значением «Сергеевич», опыт (Experience) со значением «6», фото психолога (Photo_Psychologist), описание (Description) «Лучший психолог», учёную степень (Degree) «Бакалавр», расписание (timetable), специализации (Specialization) со значением «Клиническая». В запросе добавить заголовок «Authorization», в котором передается токен доступа психолога.</p> <p>Ожидаемый результат: сервер должен вернуть в формате JSON информацию о психологе (psychologist).</p>	Успешно.
7	Отмена услуг	<p>Действие: отправить DELETE запрос на адрес /client/booking/1, где «1» идентификатор бронирования, указав в запросе заголовок «Authorization», в котором передается токен доступа пользователя.</p> <p>Ожидаемый результат: сервер должен вернуть ответ со статусом «200».</p>	Успешно.

Продолжение таблицы 4.1

Номер	Функция web-приложения	Описание тестирования	Итог тестирования функции
8	Добавить отзыв	Действие: отправить POST запрос на адрес /client/reviews/, указав в теле запроса отзыв пользователя (Review) со значением «Мне понравилось». В запросе необходимо добавить заголовок «Authorization», в котором передается токен доступа. Ожидаемый результат: сервер в формате JSON должен вернуть отзыв (review).	Успешно.
9	Удалить отзыв	Действие: отправить DELETE запрос на адрес /client/reviews/1, где «1» идентификатор отзыва. В запросе необходимо добавить заголовок «Authorization», в котором передается токен доступа пользователя. Ожидаемый результат: сервер должен вернуть ответ со статусов «200».	Успешно.
10	Создание талонов для психолога	Действие: отправить POST запрос на адрес /psychologists/voucher/. В запросе необходимо добавить заголовок «Authorization», в котором передается токен доступа пользователя. Ожидаемый результат: сервер должен вернуть ответ со статусов «200».	Успешно.
10	Создание талонов для администратора	Действие: отправить POST запрос на адрес /admin/vouchers/1, где «1» идентификатор психолога. В запросе необходимо добавить заголовок «Authorization», в котором передается токен доступа пользователя. Ожидаемый результат: сервер должен вернуть ответ со статусов «200».	Успешно.

Продолжение таблицы 4.1

Номер	Функция web-приложения	Описание тестирования	Итог тестирования функции
11	Создание профиля для психолога	<p>Действие: отправить POST запрос на адрес /admin/psychologists/, указав в теле запроса логин психолога (Login_Psychologist) со значением «mik», пароль (Password_Psychologist) «1122», имя (Name_Psychologist) «Никита», фамилию (Surname_Psychologist) «Каребо», отчество (Patronymic_Psychologist) «Сергеевич», опыт (Experience) «3», почту (Mail_Psychologist) «nikkar@gmail.com», фото (Photo_Psychologist), описание (Description) «Лучший психолог», учёную степень (Degree) «Бакалавр», расписание (timetable), специализации (Specialization) «Клиническая». В запросе необходимо добавить заголовок «Authorization», в котором передается токен доступа администратора.</p> <p>Ожидаемый результат: сервер должен вернуть в формате JSON информацию о психологе (psychologist).</p>	Успешно.
12	Удаление профиля для психолога	<p>Действие: отправить DELETE запрос на адрес /admin/psychologists/1, где «1» идентификатор психолога. В запросе необходимо добавить заголовок «Authorization» с токеном доступа.</p> <p>Ожидаемый результат: сервер должен вернуть ответ со статусов «200».</p>	Успешно.
13	Создание услуги	<p>Действие: отправить POST запрос на адрес /admin/procedures/, указав в теле запроса название процедуры (Name_Procedure) со значением «Консультация», цену (Price) со значением «23», описание (Depiction) «Для семьи», фото (Photo_Procedure), специализации (Specialization) «Клиническая». В запросе добавить заголовок «Authorization», в котором передается токен доступа.</p> <p>Ожидаемый результат: сервер в формате JSON должен вернуть информацию о процедуре (procedure).</p>	Успешно.

Продолжение таблицы 4.1

Номер	Функция web-приложения	Описание тестирования	Итог тестирования функции
14	Удаление услуги	Действие: отправить DELETE запрос на адрес /admin/procedures/1, где «1» идентификатор процедуры. В запросе необходимо добавить заголовок «Authorization», в котором передается токен доступа пользователя. Ожидаемый результат: сервер должен вернуть ответ со статусов «200».	Успешно.
15	Изменение информации о психологе	Действие: отправить POST запрос на адрес /admin/psychologists/1, где «1» идентификатор психолога, также указав в теле имя (Name_Psychologist) «Сергей», фамилию (Surname_Psychologist) «Каебо», отчество (Patronymic_Psychologist) «Андреевич», опыт (Experience) «4», фото психолога (Photo_Psychologist), описание (Description) «Плохой психолог», учённую степень (Degree) «Бакалавр», расписание (timetable), специализации (Specialization) «Рзвивающая». В запросе необходимо добавить заголовок «Authorization», в котором передается токен доступа психолога. Ожидаемый результат: сервер должен вернуть в формате JSON обновлённую информацию о психологе (psychologist).	Успешно.
15	Изменение информации о услуге	Действие: отправить POST запрос на адрес /admin/procedures/1, где «1» идентификатор процедуры, указав в теле запроса название процедуры (Name_Procedure), цену (Price), описание (Depiction), фото (Photo_Procedure), специализации (Specialization). В запросе необходимо добавить заголовок «Authorization», в котором передается токен доступа пользователя. Ожидаемый результат: сервер в формате JSON должен вернуть информацию о процедуре (procedure).	Успешно.

Продолжение таблицы 4.1

Номер	Функция web-приложения	Описание тестирования	Итог тестирования функции
16	Удаление отзывов	Действие: отправить DELETE запрос на адрес /admin/reviews/1, где «1» идентификатор отзыва. В запросе необходимо добавить заголовок «Authorization», в котором передается токен доступа пользователя. Ожидаемый результат: сервер должен вернуть ответ со статусов «200».	Успешно.

Таким образом, были протестированы все ключевые функции web-приложения.

4.2 Нагрузочное тестирование

При нагрузочном тестировании использовался JMeter [15] — это инструмент с открытым исходным кодом для проведения нагрузочного тестирования различных систем, поддерживающий множество протоколов.

Были произведены тесты с разным количеством GET-запросов в секунду на адрес /psychologists сервера. Результаты тестов представлены в таблице 4.2.

Таблица 4.1 – Описание тестирования функций web-приложения

Количество запросов в секунду	Количество успешных ответов	Среднее время получения ответа в миллисекундах	Итог
100	100	1949	Успешно
500	500	4265	Успешно
1000	1000	8209	Успешно
5000	5000	33376	Успешно
10000	10000	71436	Успешно

Все тесты были пройдены, что свидетельствует о стабильности работы системы при различных уровнях нагрузки.

4.3 Выводы по разделу

1. Проведено ручное тестирование всех ключевых функций web-приложения. Корректность работы системы подтверждена соответствием фактических результатов тестирования ожидаемым. Количество тестов составило 22.

2. Нагрузочное тестирование показало, что приложение способно выдерживать значительные объёмы запросов, сохраняя работоспособность и стабильность даже под высокой нагрузкой.

5 Руководство пользователя

5.1 Руководство гостя

При открытии сайта гость автоматически попадает на главную страницу, которая служит визитной карточкой психологического центра. На этой странице представлена общая информация, которая помогает пользователю узнать больше о центре, его миссии и ценностях.

Главная страница представлена на рисунке 5.1.

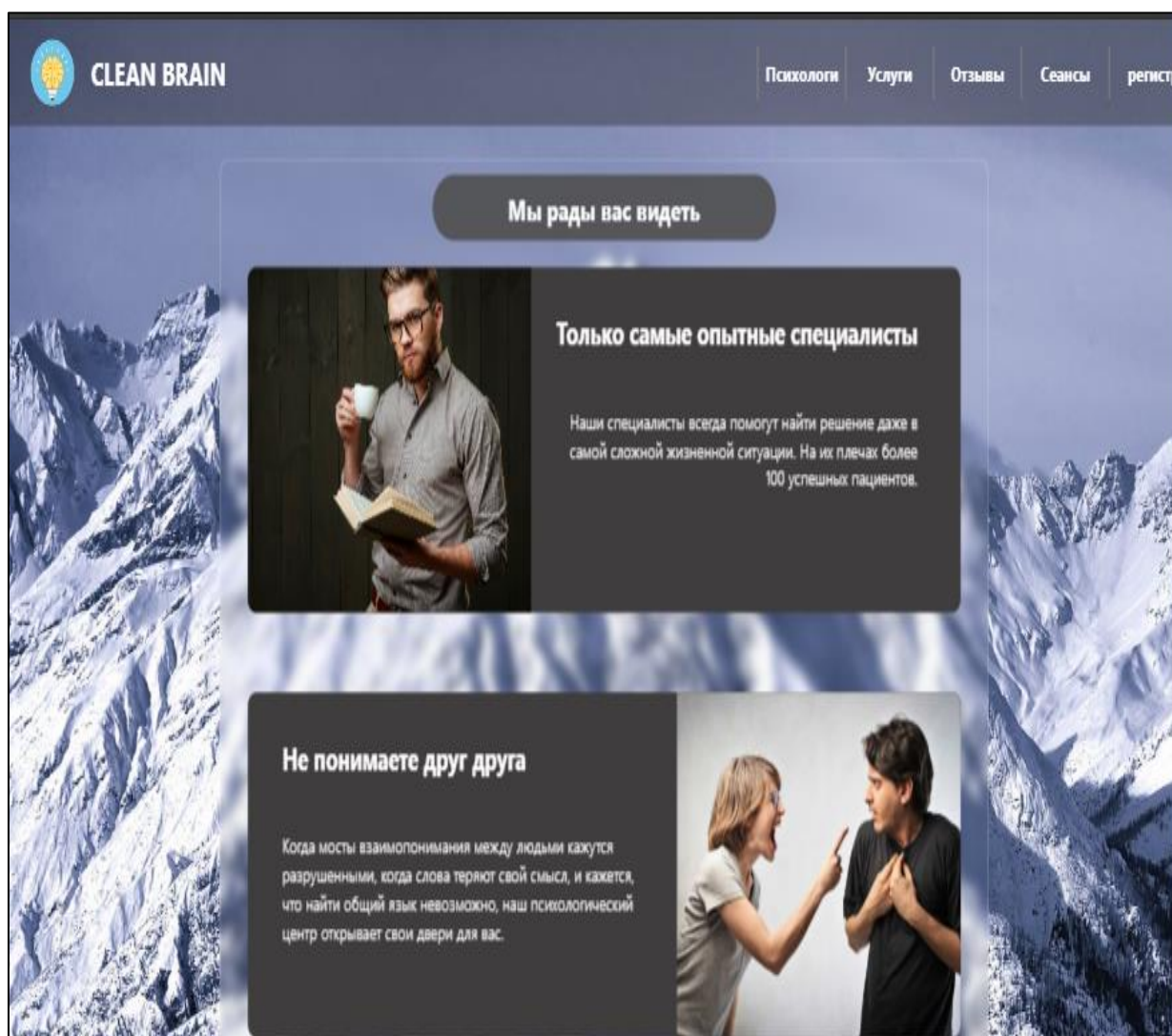


Рисунок 5.1 – Главная страница

На странице присутствует навигационное меню, которое обеспечивает удобство в перемещении по различным разделам системы. В меню представлены вкладки, такие как "Психологи", "Услуги" и "Отзывы". Вкладка "Психологи" позволяет просматривать список доступных специалистов и их профили. Раздел "Услуги" содержит перечень предлагаемых услуг с подробным описанием каждой из них. Во вкладке "Отзывы" можно ознакомиться с отзывами пользователей.

5.1.1 Регистрация

Если гость заинтересован в услугах психологического центра, он может нажать на кнопку «Регистрация», которая перенаправит его на страницу регистрации.

Форма регистрации представлена на рисунке 5.1.

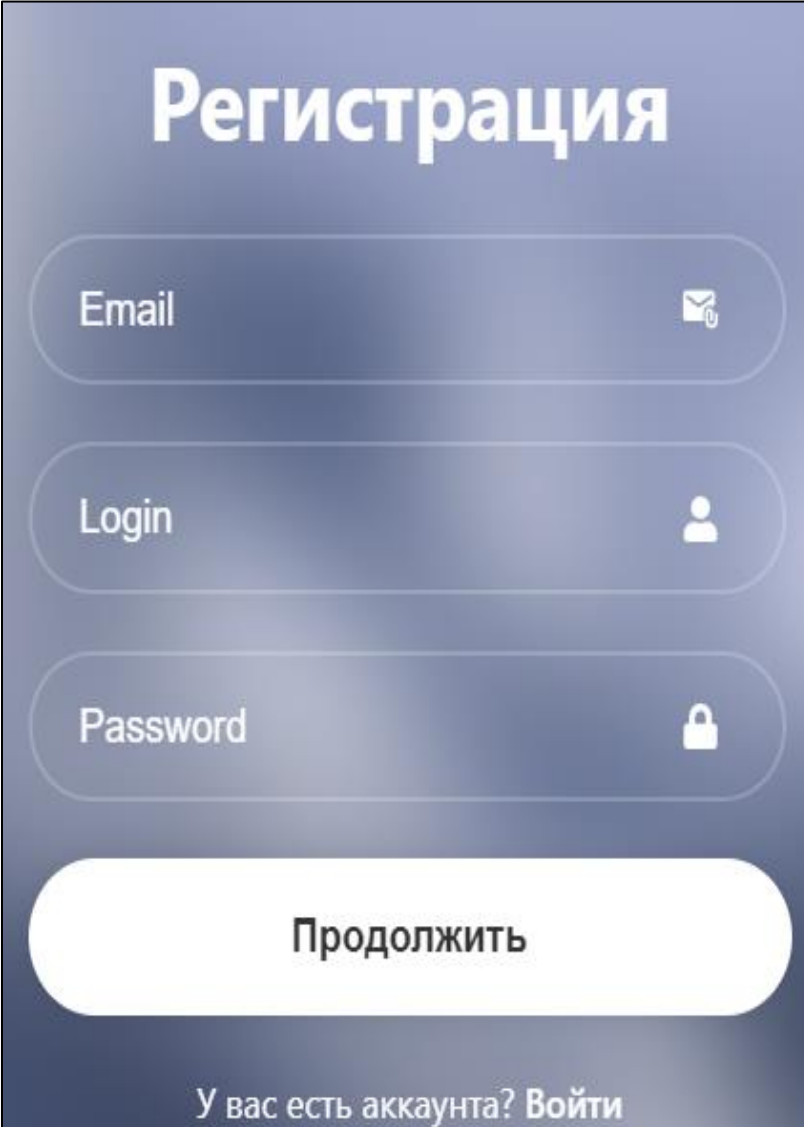
The image shows a registration form on a dark blue background. At the top, the word "Регистрация" (Registration) is written in large white letters. Below it are three input fields: "Email" with an envelope icon, "Login" with a person icon, and "Password" with a lock icon. All three fields are rounded rectangles with a light blue border. Below these fields is a large white button with the text "Продолжить" (Continue) in black. At the bottom, there is a link that says "У вас есть аккаунта? Войти" (Do you have an account? Log in) in white text.

Рисунок 5.1 – Форма регистрации

Здесь необходимо заполнить форму, указав свою электронную почту, логин и пароль. После заполнения формы, гость нажимает кнопку «Продолжить», которая инициирует процесс проверки введенных данных и отправки кода подтверждения на указанную электронную почту.

После успешного отправления кода на странице автоматически появляется дополнительное поле для его ввода. Это поле позволяет гостю подтвердить владение указанным адресом электронной почты.

Такой подход обеспечивает безопасную верификацию новых учетных записей и защиту от некорректных данных или автоматической регистрации.

Форма регистрации с полем для ввода кода, представлена на рисунке 5.2.

The image shows a registration form titled "Регистрация" (Registration) on a dark blue background. It contains four input fields, each with a corresponding icon on the right: an email field with "test123@gmail.com" and an envelope icon, a username field with "testuser" and a person icon, a password field with four dots and a lock icon, and a code field with the text "Code" and a lock icon. Below the code field is a link that says "Получить код ещё раз" (Get code again). At the bottom of the form is a large white button labeled "Создать" (Create) and a link that says "У вас есть аккаунта? Войти" (Do you have an account? Log in).

Рисунок 5.2 – Форма регистрации с полем для кода

После успешного ввода кода подтверждения, который был отправлен на указанную электронную почту, и нажатия на кнопку «Создать», будет создан личный аккаунт в системе психологического центра.

После пользователь получает доступ к полному функционалу web-приложения, включая возможность просмотра и выбора услуг, записи на консультации.

5.1.2 Просмотр процедур, психологов, отзывов

По нажатию кнопки «Психологи» на главной странице, гость перенаправляется на страницу, где отображается полный список доступных психологов с их именами, специализациями, рейтингом, отзывами и стоимостью услуг.

На странице реализованы функции поиска и фильтрации.

Поиск позволяет находить психологов по имени фамилии и отчеству, а фильтры дают возможность отобрать специалистов по специализации.

Страница со списком психологов, представлена на рисунке 5.3.

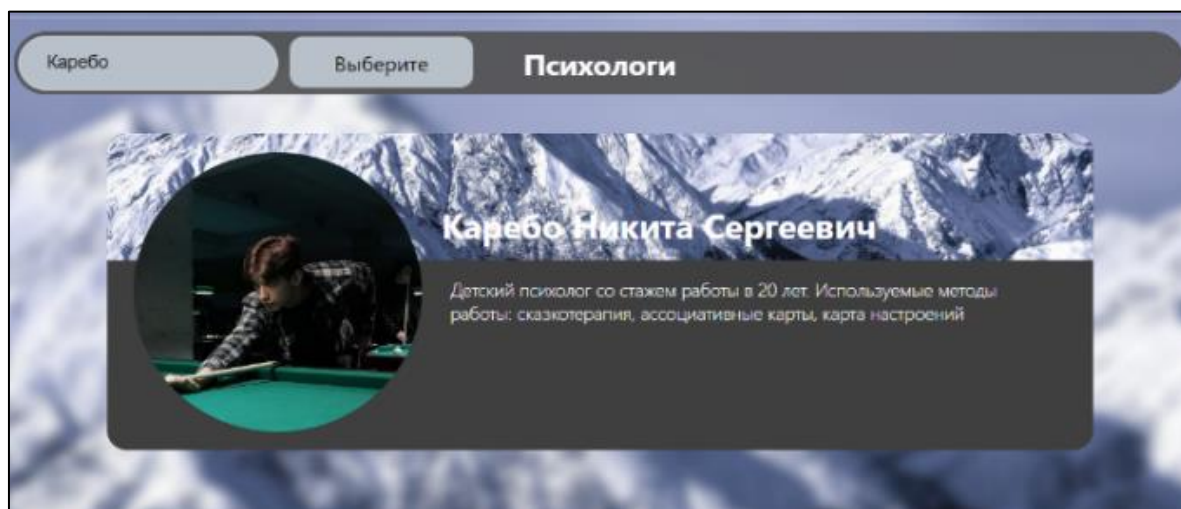


Рисунок 5.3 – Страница со списком психологов

При клике на изображение психолога, гостю открывается профиль психолога для более детального ознакомления.

Профиль психолога представлен на рисунке 5.4.

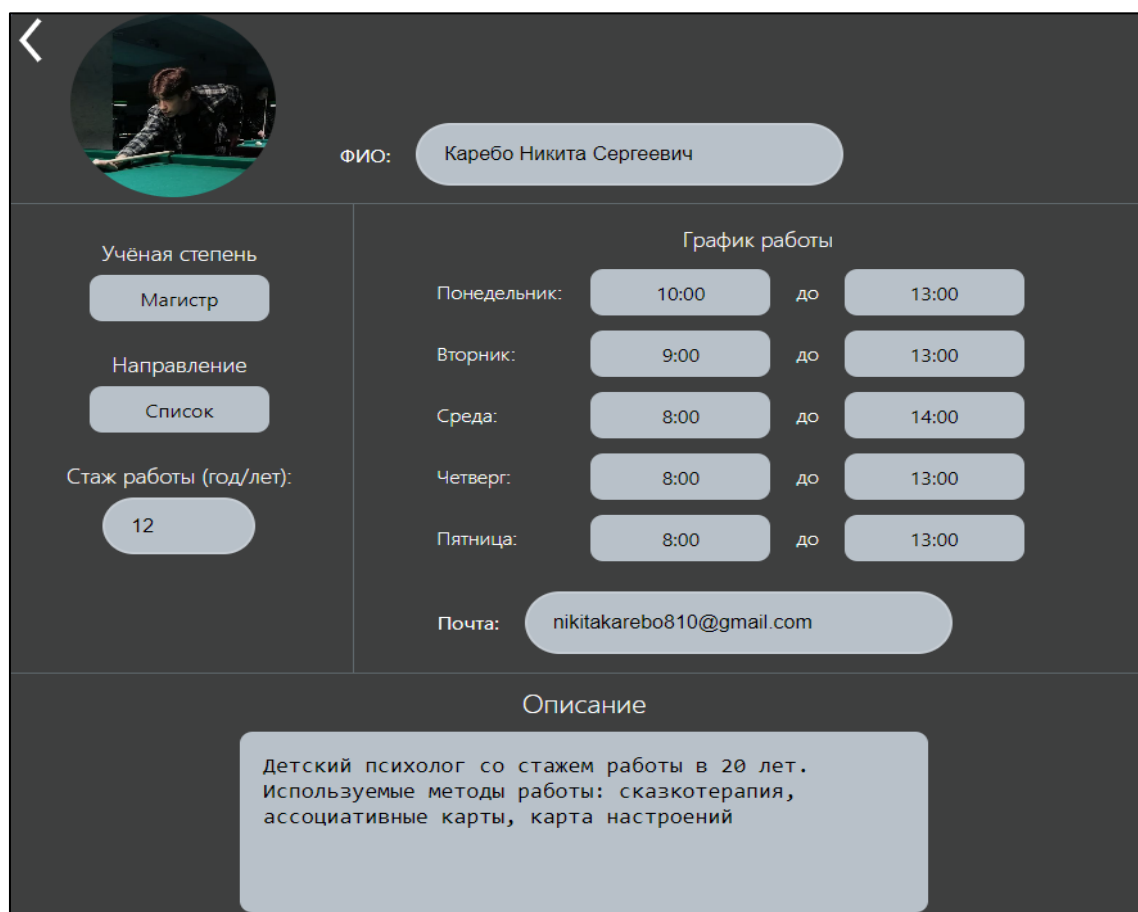


Рисунок 5.4 – Профиль психолога

На этой странице гость сможет узнать расписание психолога, почту, ознакомиться с его профессиональным стажем и прочитать подробное описание о нём.

По нажатию кнопки «Услуги» гость попадает на страницу со всеми предоставляемыми услугами. Страница с услугами психологического центра представлена на рисунке 5.5.

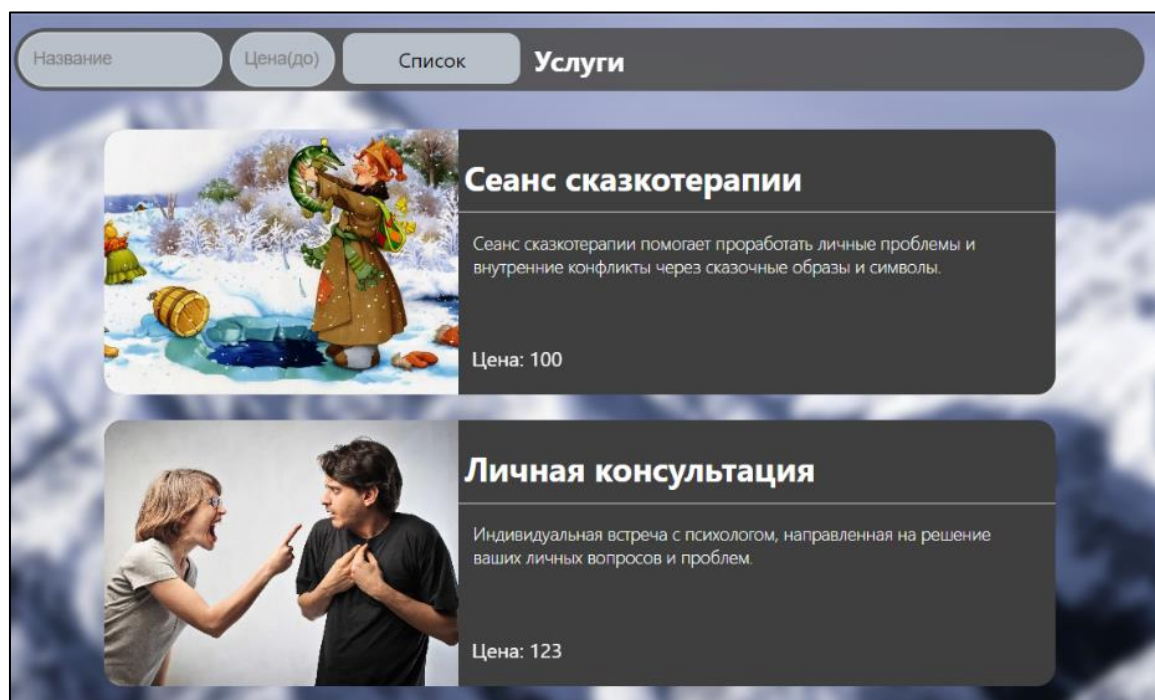


Рисунок 5.5 – Страница со списком услуг

При клике на изображение услуги гость перенаправляется на страницу с подробной информацией о данной услуге.

Представлено на рисунке 5.5.

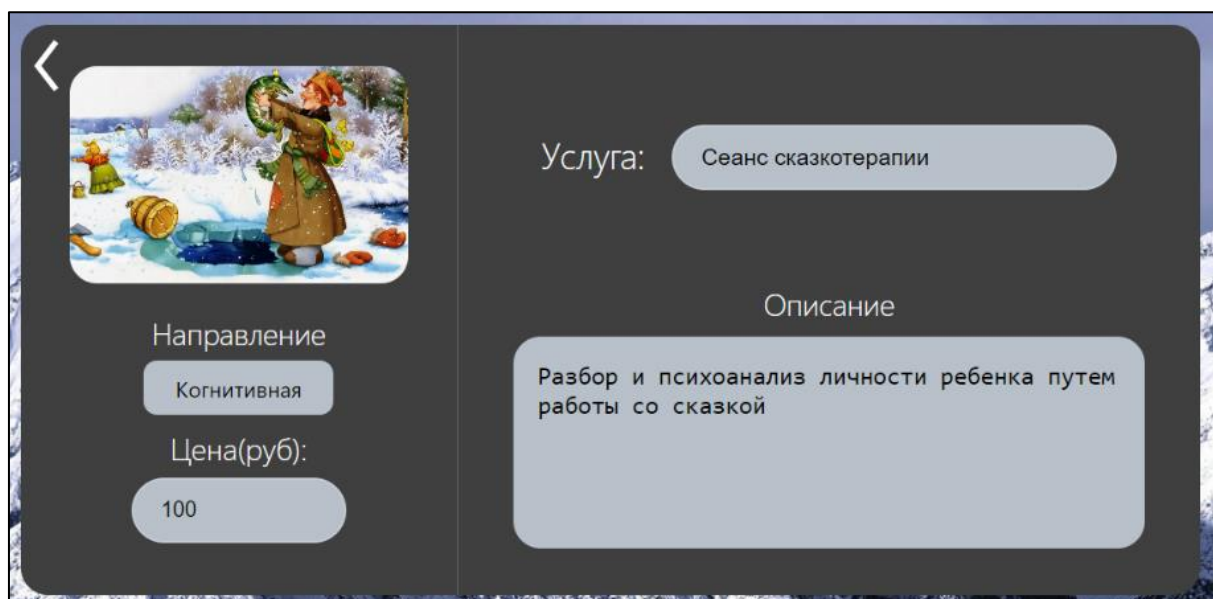


Рисунок 5.5 – Страница услуги

На данной странице гость сможет узнать о направлении услуги, её стоимости и прочитать подробное описание.

Гость также имеет возможность перейти в раздел «Отзывы», где он может просмотреть и изучить отзывы, оставленные другими пользователями.

Страница с отзывами представлена на рисунке 5.6.

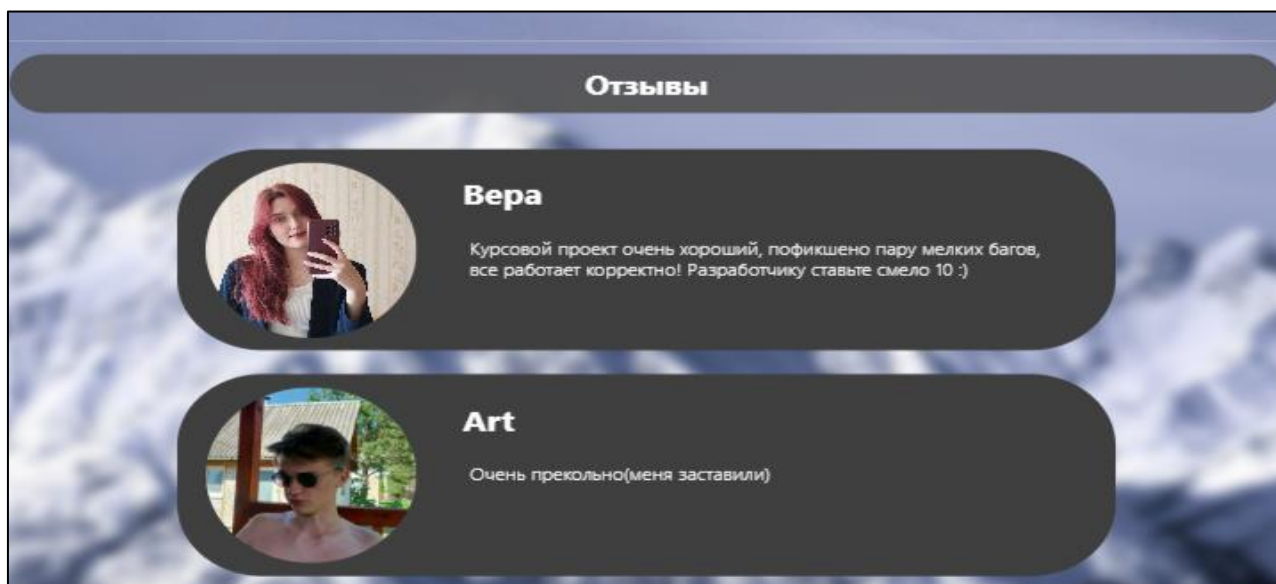


Рисунок 5.6 – Страница с отзывами

Этот раздел предоставляет гостю информацию о опыте других людей с психологическим центром, что поможет ему принять решение о выборе центра и услуг.

5.1.3 Поиск услуг и психологов

Во вкладке «Психологи» доступен поиск по ФИО и направлениям работы. Секция, содержащая эти параметры, представлена на рисунке 5.7.



Рисунок 5.7 – Секция с параметрами поиска психологов

Во вкладке «Услуги» также доступен поиск по направлениям, цене и по названию услуги. Секция, содержащая эти параметры, представлена на рисунке 5.8.

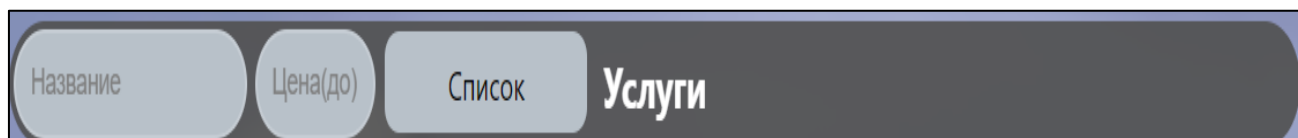


Рисунок 5.8 – Секция с параметрами поиска услуг

При изменении параметров поиска содержимое списка услуг будет автоматически обновляться, отображая результаты, соответствующие заданным критериям.

5.2 Руководство пользователя

5.2.1 Аутентификация

Если у пользователя уже есть зарегистрированный аккаунт, ему необходимо перейти на страницу авторизации.

На этой странице пользователь может ввести свой логин и пароль в соответствующие поля формы авторизации, чтобы получить доступ к личному кабинету и всем функциям приложения психологического центра.

Форма авторизации представлена на рисунке 5.9.

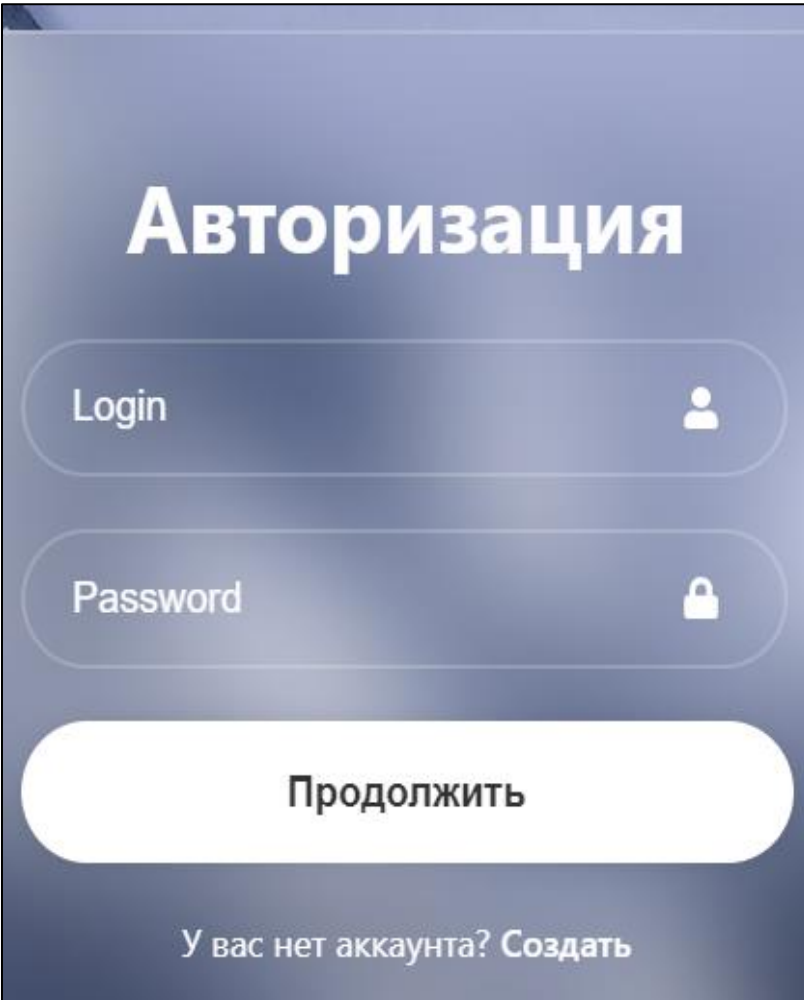
The image shows a digital form for user authentication. At the top, the word 'Авторизация' (Authorization) is displayed in a large, bold, white font against a dark blue background. Below the title, there are two input fields. The first field is labeled 'Login' in white text and has a white user icon on its right side. The second field is labeled 'Password' in white text and has a white lock icon on its right side. Below these fields is a large, white, rounded rectangular button with the text 'Продолжить' (Continue) in dark blue. At the bottom of the form, there is a link that reads 'У вас нет аккаунта? Создать' (Don't have an account? Create) in white text.

Рисунок 5.9 – Форма авторизации

После ввода логина и пароля в указанные поля и нажатия на кнопку «Продолжить», система отправляет на привязанную к аккаунту электронную почту код подтверждения.

После получения кода, пользователь должен ввести его в соответствующее поле на форме авторизации, чтобы завершить процесс аутентификации.

Форма авторизации с дополнительным полем для ввода кода представлена на рисунке 5.10.

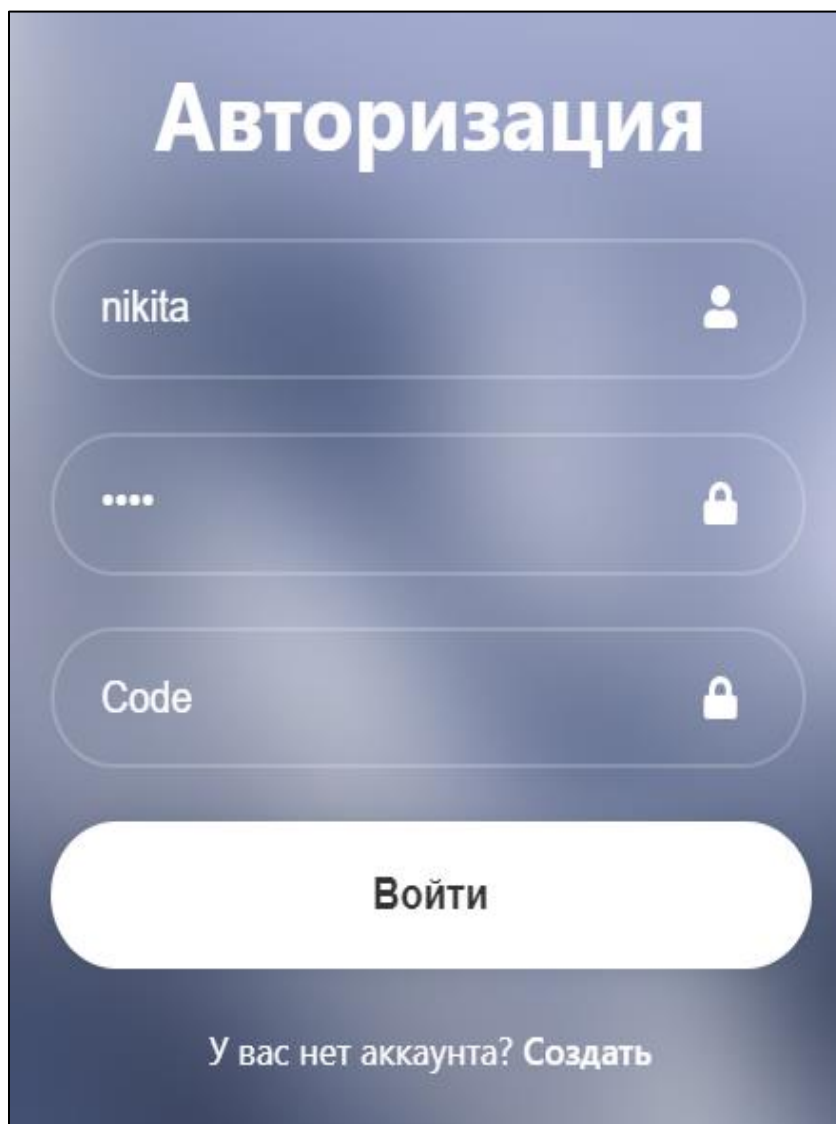


Рисунок 5.10 – Форма авторизации с полем для кода

После успешной аутентификации пользователь сможет получить доступ к тем же страницам, что и гость, включая страницы с информацией о психологах, услугах и отзывах, а также к своему профилю и текущим сеансам.

5.2.2 Добавить отзыв

В отличие от гостя, пользователь имеет возможность оставить отзыв о работе психологического центра. Для этого он должен перейти на страницу с отзывами и нажать кнопку «Создать».

После этого отобразится форма для написания отзыва, где пользователь может выразить свои впечатления и оценку работы центра.

Форма для создания отзыва представлена на рисунке 5.11.

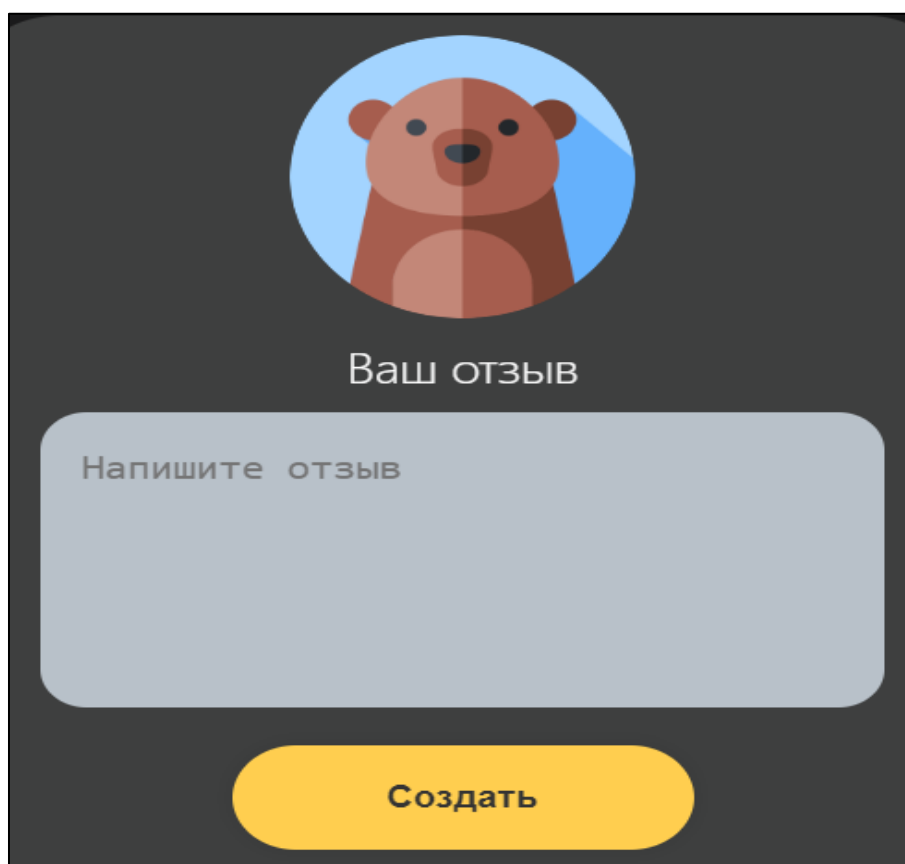
The image shows a dark gray rectangular form. At the top center is a circular profile picture of a brown bear. Below the picture, the text 'Ваш отзыв' (Your review) is centered. Underneath is a large, light gray rounded rectangular text input field with the placeholder text 'Напишите отзыв' (Write a review). At the bottom center of the form is a yellow rounded rectangular button with the text 'Создать' (Create).

Рисунок 5.11 – Форма для создания отзыва

После нажатия на кнопку «Создать», отзыв будет отображен в списке на странице с отзывами.

5.2.3 Удалить отзыв

Если пользователь захочет удалить свой отзыв по каким-либо причинам, ему нужно нажать на значок крестика рядом с отзывом, представлено на рисунке 5.12.

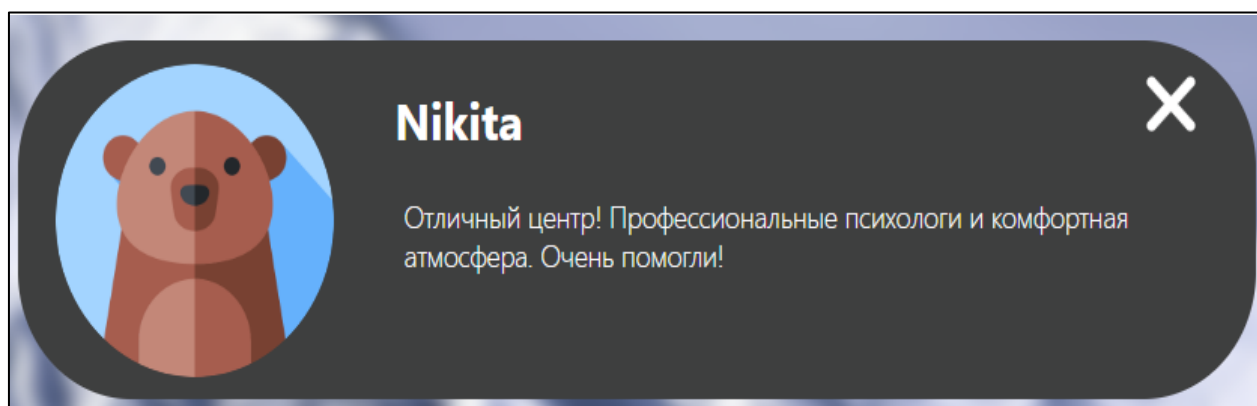


Рисунок 5.12 – Отзыв с крестиком

После нажатия отзыв будет удалён из базы данных и перестанет отображаться на странице.

5.2.4 Оформление услуги

Для оформления услуги пользователю нужно перейти на страницу с услугами и нажать кнопку «Выбрать», расположенную рядом с желаемой процедурой, представлено на рисунке 5.13.

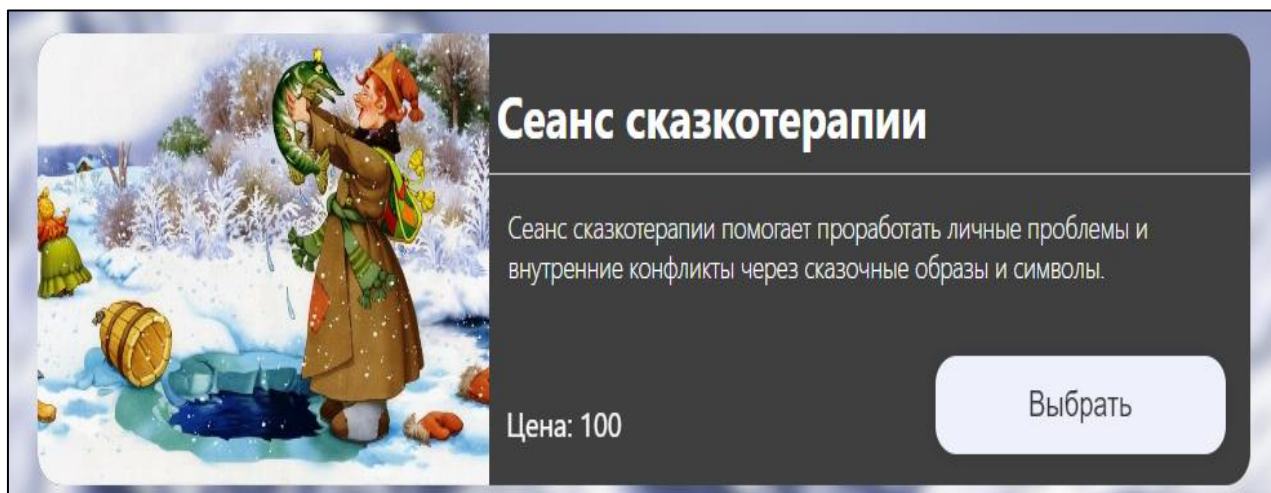


Рисунок 5.13 – Карточка услуги

После выбора процедуры пользователь будет перенаправлен на страницу с психологами, которые специализируются на данной услуге. Для записи на сеанс необходимо нажать кнопку «Записаться» напротив выбранного психолога, представлено на рисунке 5.14.

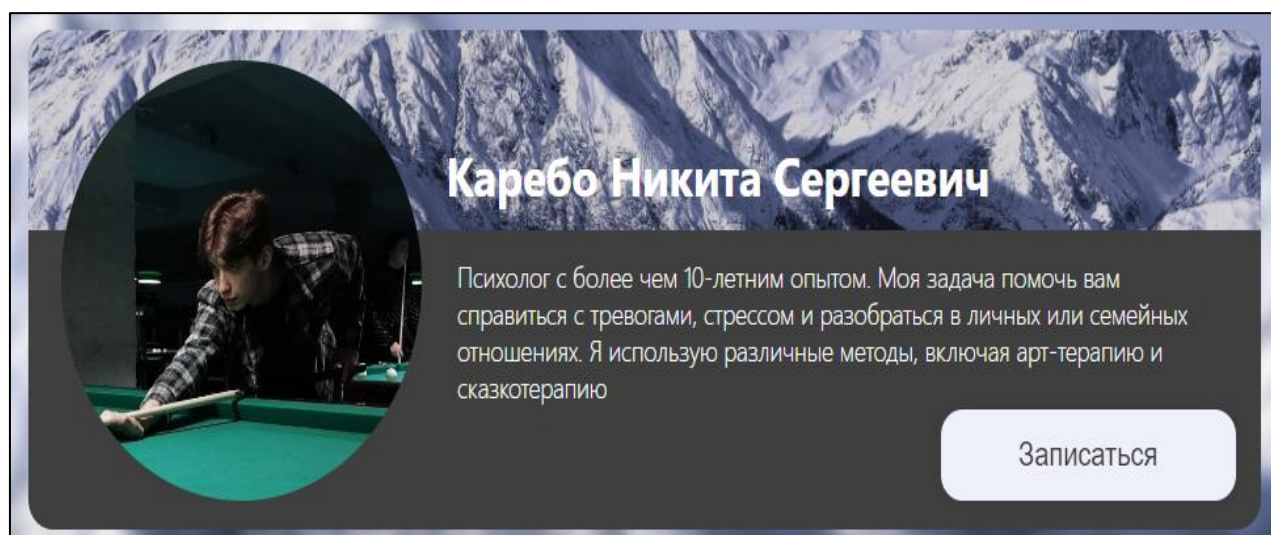


Рисунок 5.14 – Карточка психолога

После выбора психолога на странице откроется форма для оформления услуги, где пользователь сможет настроить параметры предстоящего сеанса. Эта

форма является ключевым этапом записи на услугу, так как позволяет выбрать наиболее удобные дату и время для встречи с выбранным специалистом.

В верхней части формы отображается информация о услуге, а также о психологе, чтобы пользователь мог убедиться в правильности выбора. Далее представлены свободные даты и временные интервалы, доступные в расписании специалиста. Пользователю необходимо выбрать подходящий вариант, ориентируясь на своё удобное время и доступные слоты. Форма оформления услуги представлено на рисунке 5.15.

Рисунок 5.15 – Оформление выбранной услуги

После выбора даты и времени для сеанса пользователю необходимо нажать на кнопку «Создать». После этого откроется окно для оплаты. После завершения оплаты услуга будет окончательно оформлена, и ваша запись будет подтверждена.

5.2.5 Отмена услуг

Если пользователь захочет отменить услугу по какой-либо причине, ему необходимо перейти на страницу с сеансами и нажать на крестик рядом с соответствующим сеансом.

Страница сеансов представлена на рисунке 5.16.

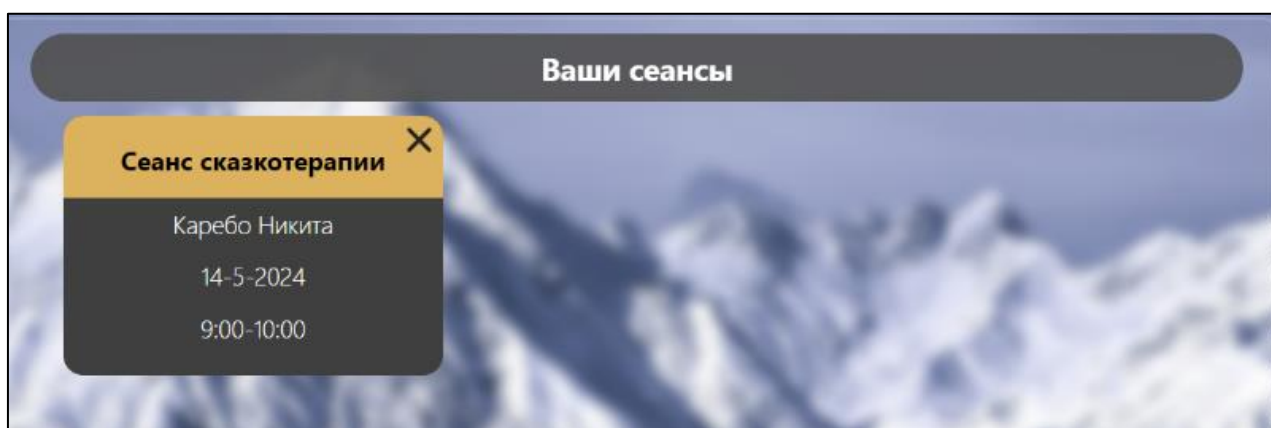


Рисунок 5.16 – Страница с текущими сеансами

После нажатия на крестик сеанс удалится из базы данных и исчезнет из списка текущих сеансов на странице.

5.2.6 Изменение информации в профиле

Пользователя также есть доступ к личному профилю, где он может просматривать и редактировать свои персональные данные, нажав на иконку профиля в навигационном меню.

Страница профиля пользователя представлена на рисунке 5.17.

The image shows a user profile form. At the top, there is a circular profile picture of a brown bear. Below the picture, there are three input fields with labels: 'Ваше имя:' (Your name), 'Фамилия:' (Surname), and 'Почта:' (Email). The 'Name' field contains the text 'Name', the 'Surname' field contains 'Surname', and the 'Email' field contains 'dankocode1@gmail.com'. At the bottom of the form, there is a yellow button with the text 'Сохранить' (Save). The background of the page is a blurred image of a snowy mountain landscape.

Рисунок 5.17 – Профиль пользователя

На данной странице пользователь может поменять своё имя и фамилию. Для сохранения изменений ему необходимо нажать на кнопку «Сохранить».

5.2.7 Просмотр процедур, психологов, отзывов

По нажатию кнопки «Психологи» на главной странице пользователь переходит на страницу с полным списком психологов. Страница с психологами представлена на рисунке 5.18.

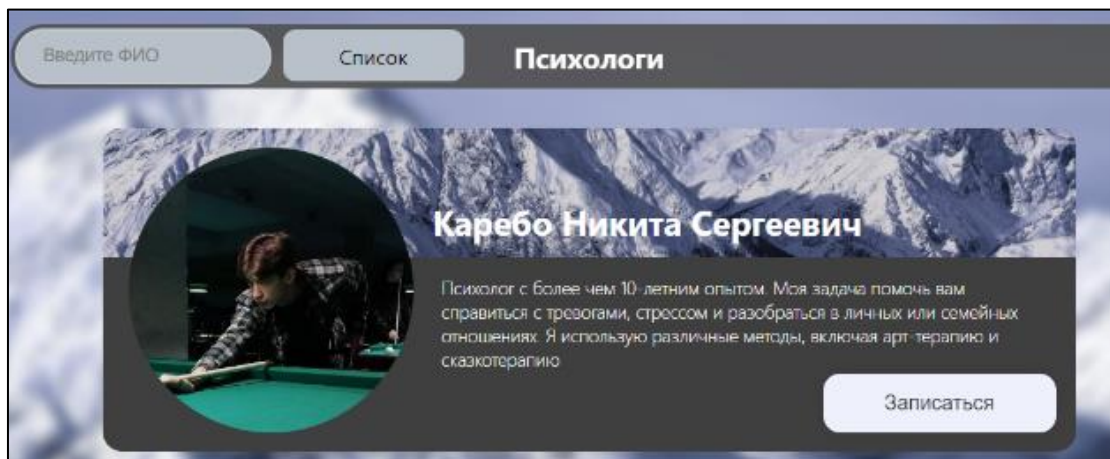


Рисунок 5.18 – Страница со списком психологов

При клике на изображение психолога, пользователю открывается профиль психолога для более детального ознакомления.

Профиль психолога представлен на рисунке 5.19.

Учёная степень		График работы			
Магистр		Понедельник:	10:00	до	13:00
Направление		Вторник:	9:00	до	13:00
Список		Среда:	8:00	до	14:00
Стаж работы (год/лет):		Четверг:	8:00	до	13:00
12		Пятница:	8:00	до	13:00
		Почта:	nikitakarebo810@gmail.com		
Описание					
Детский психолог со стажем работы в 20 лет. Используемые методы работы: сказкотерапия, ассоциативные карты, карта настроений					

Рисунок 5.19 – Профиль психолога

На этой странице пользователь сможет узнать расписание психолога, почту, ознакомиться с его профессиональным стажем и прочитать подробное описание о нём.

По нажатию кнопки «Услуги» пользователь попадает на страницу со всеми предоставляемыми услугами. Страница с услугами психологического центра представлена на рисунке 5.20.

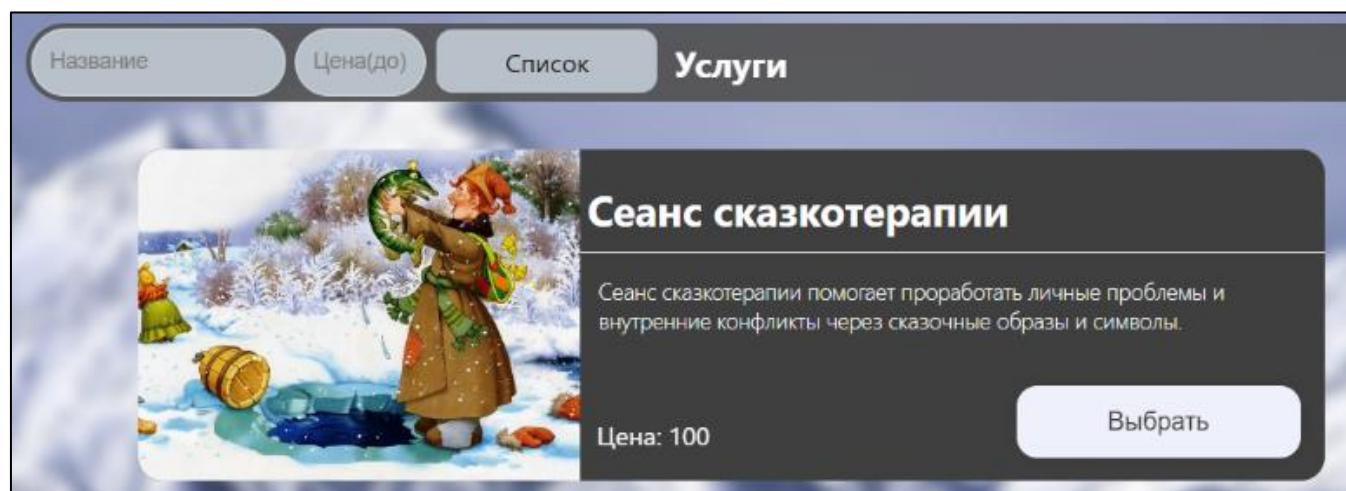


Рисунок 5.20 – Страница со списком услуг

При клике на изображение услуги пользователь перенаправляется на страницу с подробной информацией о данной услуге.

Представлено на рисунке 5.21.

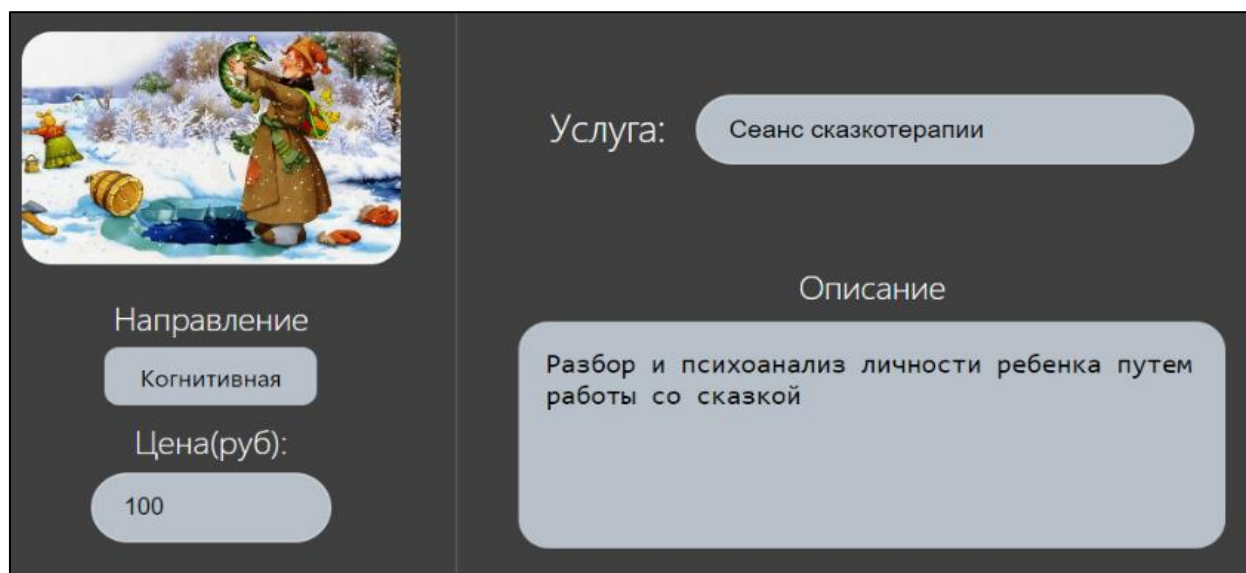


Рисунок 5.21 – Страница услуги

На данной странице пользователь сможет узнать о направлении услуги, её стоимости и прочитать подробное описание.

Пользователь также имеет возможность перейти в раздел «Отзывы», где он может просмотреть и изучить отзывы, оставленные другими пользователями.

Страница с отзывами представлена на рисунке 5.22.

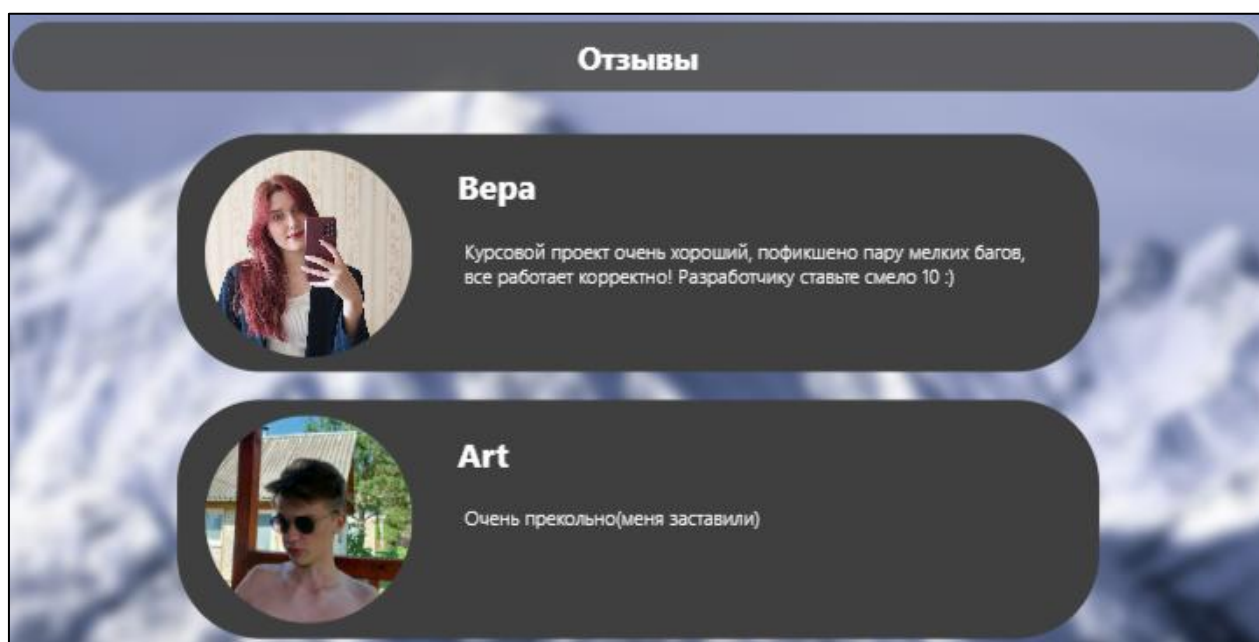


Рисунок 5.22 – Страница с отзывами

Этот раздел предоставляет пользователю информацию о опыте других людей с психологическим центром, что поможет ему принять решение о выборе центра и услуг.

5.2.8 Поиск услуг и психологов

Во вкладке «Психологи» доступен поиск по ФИО и направлениям работы. Секция, содержащая эти параметры, представлена на рисунке 5.23.



Рисунок 5.23 – Секция с параметрами поиска психологов

Во вкладке «Услуги» также доступен поиск по направлениям, цене и по названию услуги. Секция, содержащая эти параметры, представлена на рисунке 5.24.

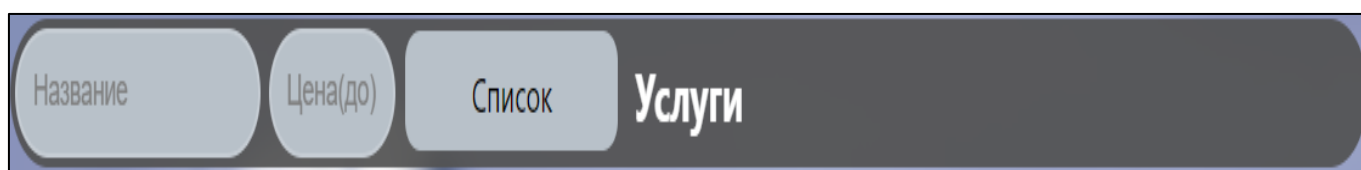


Рисунок 5.24 – Секция с параметрами поиска услуг

При изменении параметров поиска содержимое списка услуг будет автоматически обновляться, отображая результаты, соответствующие заданным критериям.

5.3 Руководство психолога

5.3.1 Аутентификация

Если психолог не зашёл в аккаунт, то ему необходимо перейти на страницу авторизации, где он сможет войти, используя свой логин и пароль.

Форма авторизации представлена на рисунке 5.25.

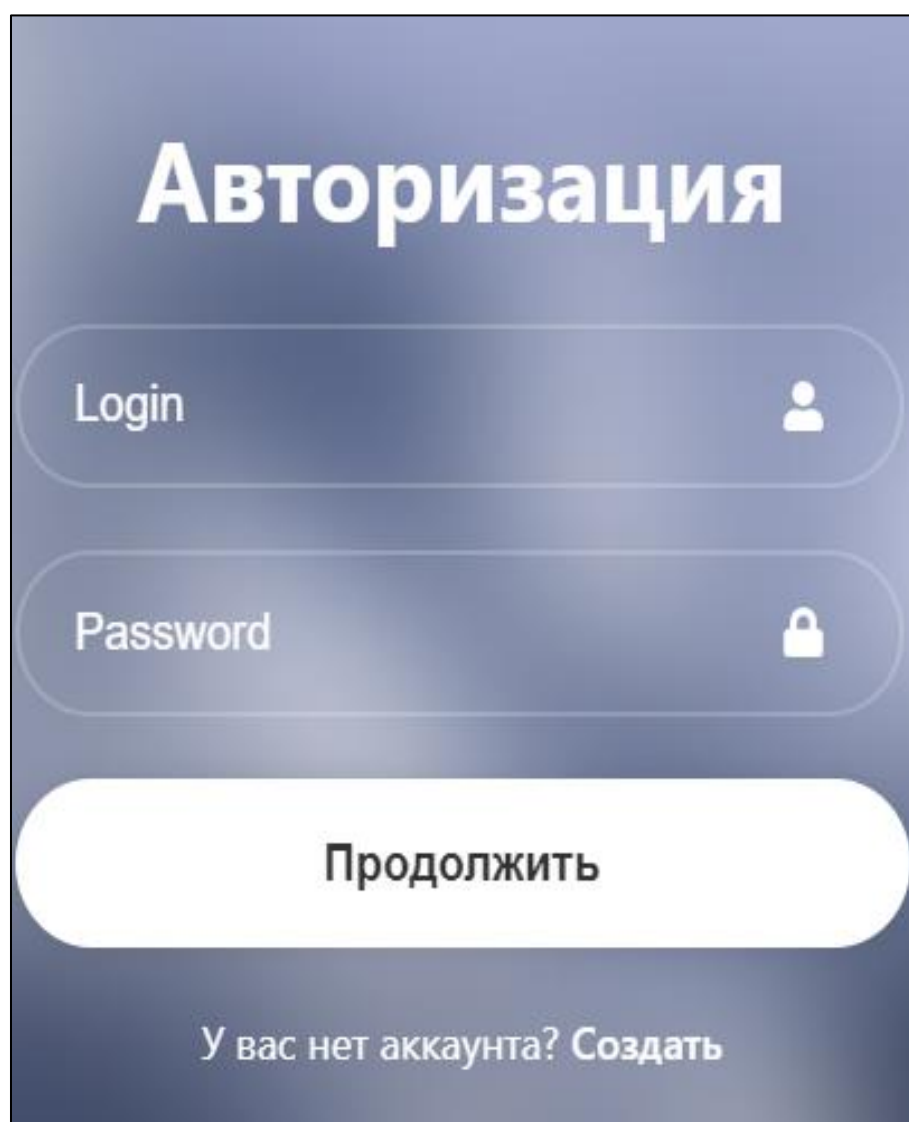
The image shows a digital form for user authentication. At the top, the word 'Авторизация' (Authorization) is displayed in a large, bold, white font against a dark blue background. Below the title, there are two input fields. The first field is labeled 'Login' in white text and has a white user icon on its right side. The second field is labeled 'Password' in white text and has a white padlock icon on its right side. Both fields have rounded corners and a light blue border. Below these fields is a large, white, rounded rectangular button with the text 'Продолжить' (Continue) in dark blue. At the bottom of the form, there is a link in white text that reads 'У вас нет аккаунта? Создать' (Don't have an account? Create).

Рисунок 5.25 – Форма авторизации

После ввода логина и пароля в указанные поля и нажатия на кнопку «Продолжить», система отправляет на привязанную к аккаунту электронную почту код подтверждения.

После получения кода, психолог должен ввести его в соответствующее поле на форме авторизации, чтобы завершить процесс аутентификации.

Форма авторизации с дополнительным полем для ввода кода представлена на рисунке 5.26.

Авторизация

nikita

....

Code

Войти

Рисунок 5.26 – Форма авторизации с полем для кода

После успешной аутентификации психолог сможет получить к своему профилю и текущим сеансам с пользователями.

5.3.2 Создание талонов

Для создания талончиков для пользователей психологу надо перейти на страницу профиля. Страница профиля психолога представлена на рисунке 5.27.

Профиль психолога

ФИО: Петров Алексей Иванович

Учёная степень: Магистр

Направление: Список

Стаж работы (год/лет): 6

График работы

Понедельник:	8:00	до	13:00
Вторник:	8:00	до	13:00
Среда:	8:00	до	13:00
Четверг:	8:00	до	13:00
Пятница:	8:00	до	13:00

Почта: nikitakarebo241@gmail.com

Описание

Я работаю с людьми, которые пережили травмы, утраты или другие сложные события. Моя задача помочь вам прожить эти моменты и найти силы двигаться дальше.

талончик Изменить

Рисунок 5.27 – Профиль психолога

Далее необходимо нажать на кнопку «Талончики», после чего автоматически, с учётом его расписания в системе, будут созданы талончики на каждый рабочий день.

5.3.3 Изменение информации в профиле

Психолог также имеет доступ к личному профилю, где он может просматривать и редактировать свои персональные данные, нажав на иконку профиля в навигационном меню.

Страница профиля психолога представлена на рисунке 5.28.

ФИО: Петров Алексей Иванович

Учёная степень: Магистр

Направление: Список

Стаж работы (год/лет): 6

График работы:

Понедельник:	8:00	до	13:00
Вторник:	8:00	до	13:00
Среда:	8:00	до	13:00
Четверг:	8:00	до	13:00
Пятница:	8:00	до	13:00

Почта: nikitakarebo241@gmail.com

Описание

Я работаю с людьми, которые пережили травмы, утраты или другие сложные события. Моя задача помочь вам прожить эти моменты и найти силы двигаться дальше.

талончик Изменить

Рисунок 5.28 – Профиль психолога

На данной странице психолог может поменять такие данные, как фамилия, имя, отчество (ФИО), научная степень, расписание, профессиональный опыт, описание и направления деятельности. Для сохранения изменений ему необходимо нажать на кнопку «Изменить».

5.3.4 Отмена услуг

Перейдя во вкладку «Сеансы», психолог получит доступ к информации о текущих активных сеансах с пользователями.

Страница сеансов представлена на рисунке 5.29.

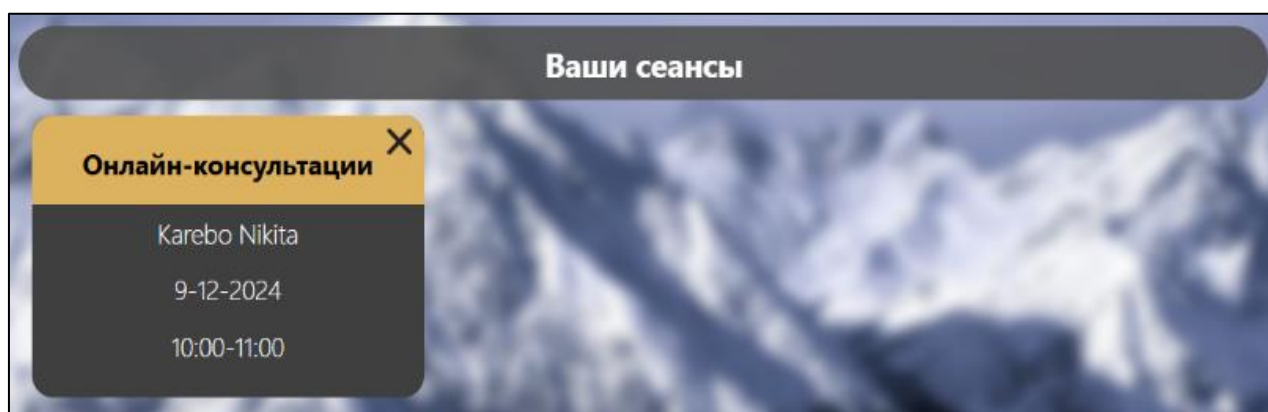


Рисунок 5.29 – Страница предстоящих сеансов

Если психологу необходимо отменить сеанс по какой-либо причине, ему просто нужно нажать на крестик рядом с соответствующим сеансом.

5.4 Руководство администратора

5.4.1 Аутентифицироваться

Если администратор не зашёл в аккаунт, то ему необходимо перейти на страницу авторизации, где он сможет войти, используя свой логин и пароль, представлено на рисунке 5.30.

The image shows a screenshot of a web application interface for user authorization. The title 'Авторизация' (Authorization) is displayed in large white letters at the top. Below the title, there are two input fields: 'Login' with a user icon and 'Password' with a lock icon. Both fields have rounded corners and a light blue background. Below these fields is a large white button with the text 'Продолжить' (Continue) in black. At the bottom, there is a link that says 'У вас нет аккаунта? Создать' (Don't have an account? Create) in white text.

Рисунок 5.30 – Форма авторизации

После заполнения данных в форму и нажатия на кнопку «Продолжить», администратору будет отправлен на почту код подтверждения. После получения

кода, администратор должен ввести его в соответствующее поле на форме авторизации, чтобы завершить процесс аутентификации, представлено на рисунке 5.31.

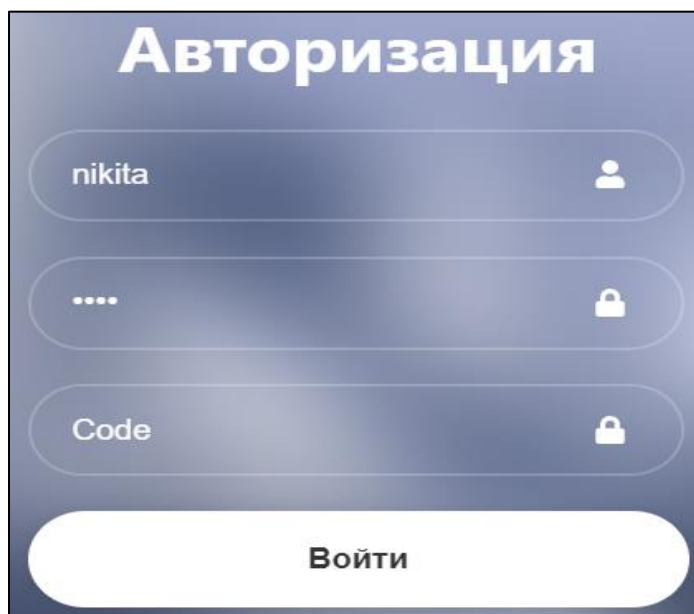
The image shows a login form titled "Авторизация" (Authorization) in a bold, white font on a dark blue background. There are three input fields: the first contains the username "nikita" with a person icon; the second contains four dots "...." with a lock icon; the third is labeled "Code" with a lock icon. At the bottom is a large white button with the text "Войти" (Login) in black.

Рисунок 5.31 – Форма авторизации с полем для кода

После успешной аутентификации администратор сможет получить доступ к тем же страницам, что и гость, включая страницы с информацией о психологах, услугах и отзывах, а также к страницам создания, редактирования психологов и услуг.

5.4.2 Удаление отзывов

Чтобы удалить отзывы пользователей, администратор должен перейти на страницу отзывов, выбрав пункт «Отзывы» в навигационной панели. Пример страницы с отзывами показан на рисунке 5.31.

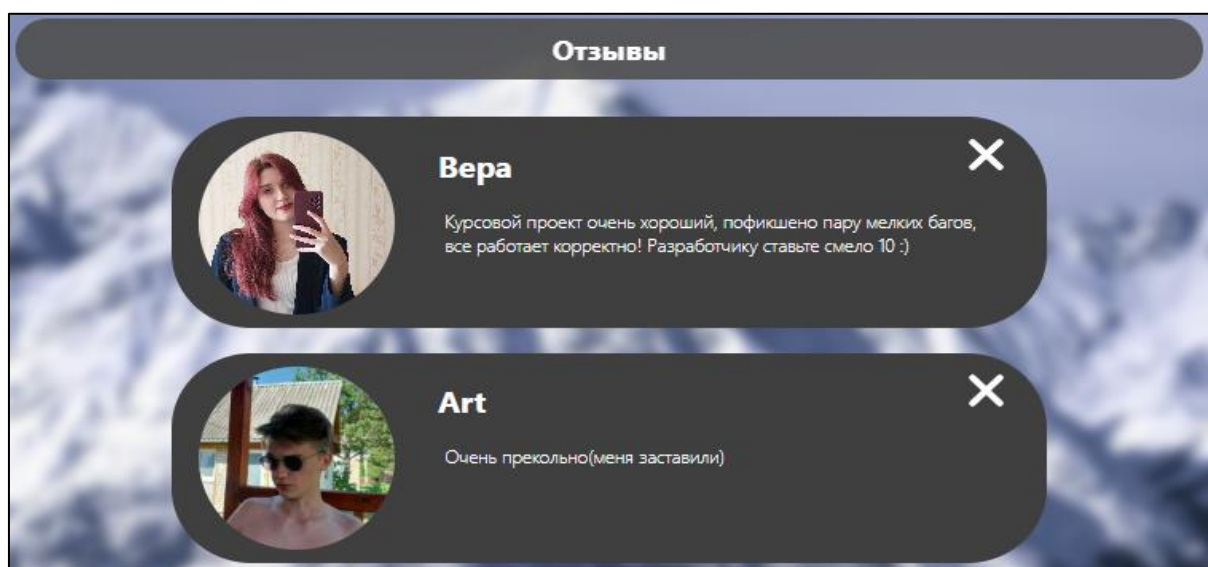


Рисунок 5.31 – Страница с отзывами для администратора

В случае, если администратор считает, что какой-либо отзыв не соответствует правилам, он может нажать на крестик около отзыва. При этом отзыв будет удален из базы данных.

5.4.3 Создание услуги

Чтобы создать услугу, администратор должен перейти на страницу со списком услуг психологического центра, выбрав пункт «Услуги» в навигационной панели. Страница со списком услуг представлена на рисунке 5.32.

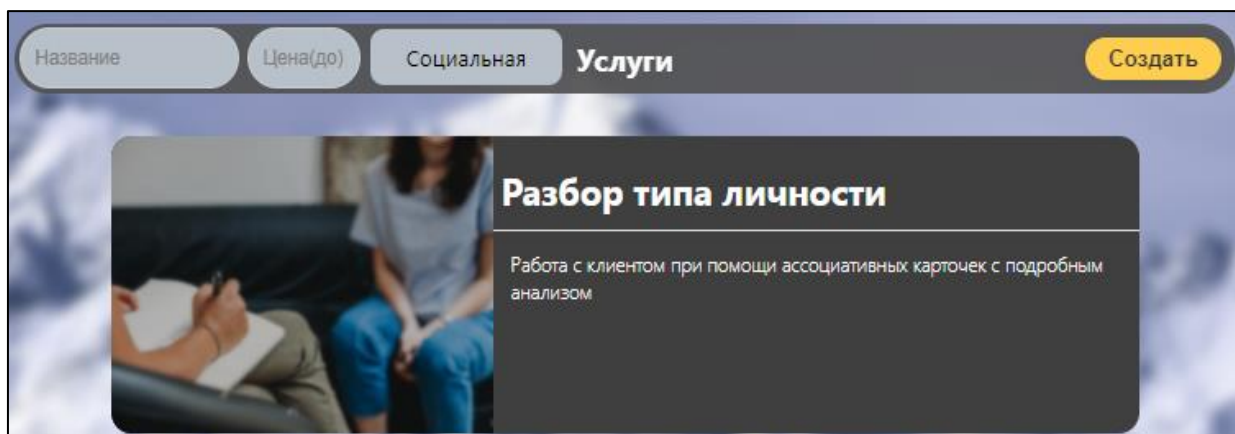


Рисунок 5.32 – Страница со списком услуг для администратора

При нажатии на кнопку «Создать» открывается форма, в которой администратор может добавить новую услугу. В этой форме администратору предоставляется возможность указать название услуги, ее цену, описание, специализацию и загрузить фотографию.

Форма для создания услуги представлена на рисунке 5.33.

Рисунок 5.33 – Форма создания услуги

После заполнения всех необходимых полей администратору следует нажать кнопку «Сохранить», чтобы добавить новую услугу.

5.4.4 Удаление услуги

Чтобы удалить услугу, администратор должен перейти на страницу со списком услуг психологического центра, выбрав пункт «Услуги» в навигационной панели. Страница со списком услуг представлена на рисунке 5.34.

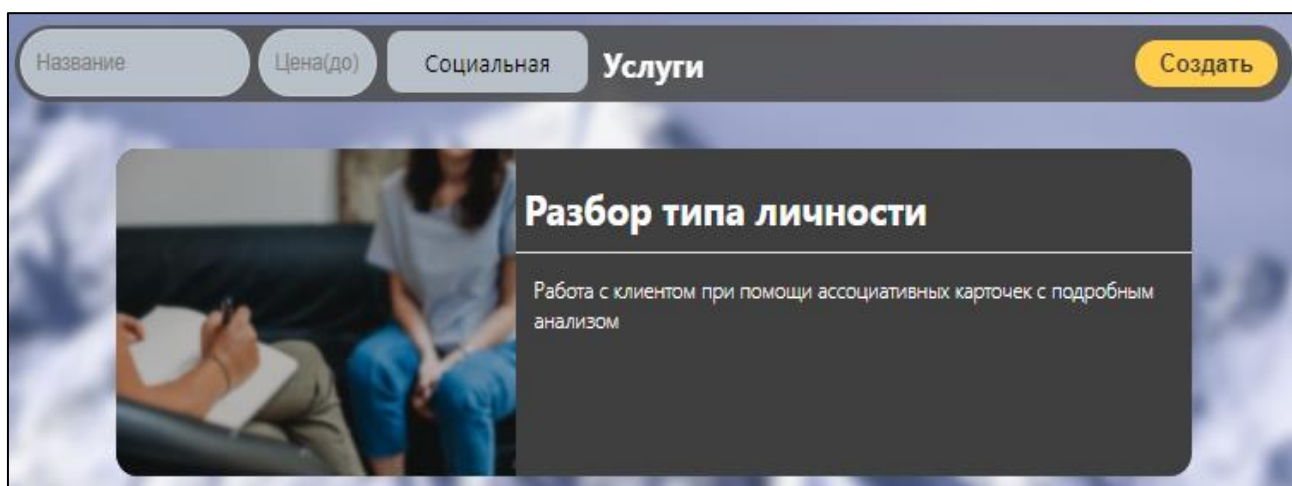


Рисунок 5.34 – Страница со списком услуг для администратора

Для удаления текущей услуги, администратор должен нажать на изображение услуги из перечня всех услуг.

Далее администратору откроется страница с данной услугой, представлено на рисунке 5.35.

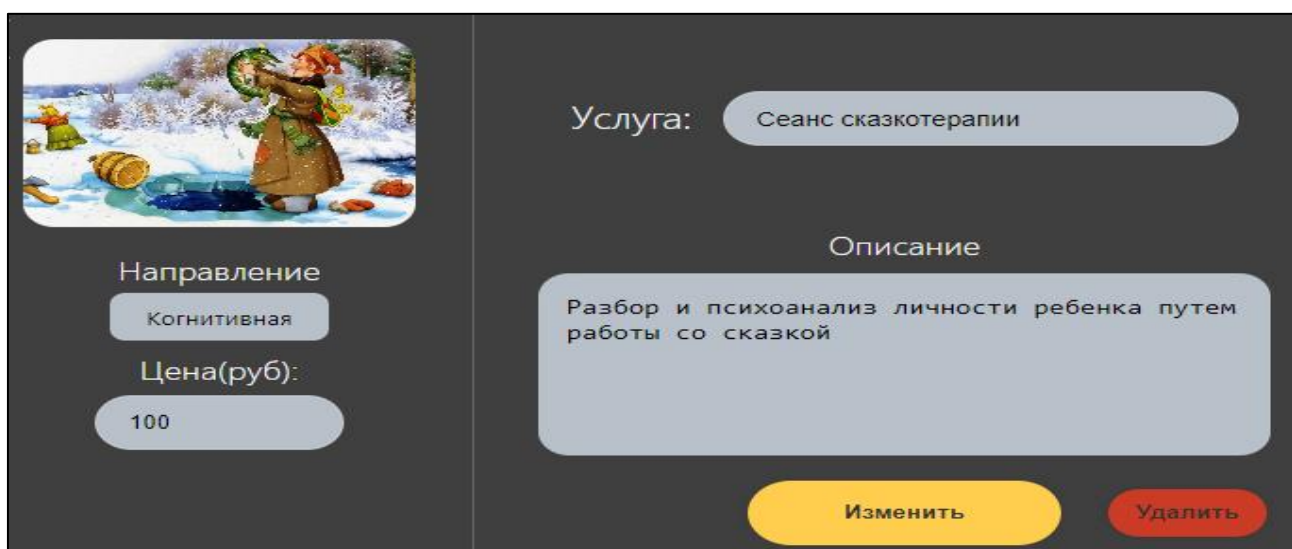


Рисунок 5.35 – Страница услуги для администратора

На странице для администратора будет доступна кнопка «Удалить» при нажатии которой услуга будет удалена из базы данных.

5.4.5 Создание талонов

Чтобы создать талончики для психолога, администратор должен перейти на страницу со списком психологов психологического центра, выбрав пункт «Психологи» в навигационной панели. Страница со списком психологов представлена на рисунке 5.36.

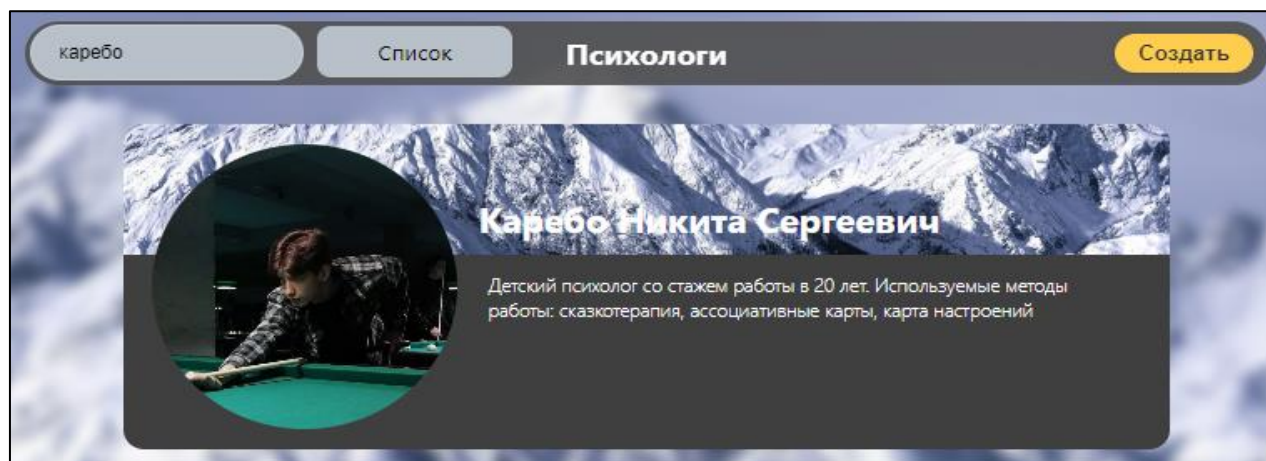


Рисунок 5.36 – Страница со списком психологов для администратора

При нажатии администратором на изображение психолога из списка психологов открывается его профиль.

Страница с профилем психолога представлена на рисунке 5.37.

Рисунок 5.37 – Профиль психолога для администратора

Для создания талончиков, администратору необходимо нажать на кнопку «Талончик». После этого талончики будут сгенерированы и добавлены в базу данных.

5.4.6 Создание профиля для психолога

Чтобы создать профиль для психолога, администратор должен перейти на страницу со списком психологов психологического центра, выбрав пункт «Психологи». Страница со списком психологов представлена на рисунке 5.38.

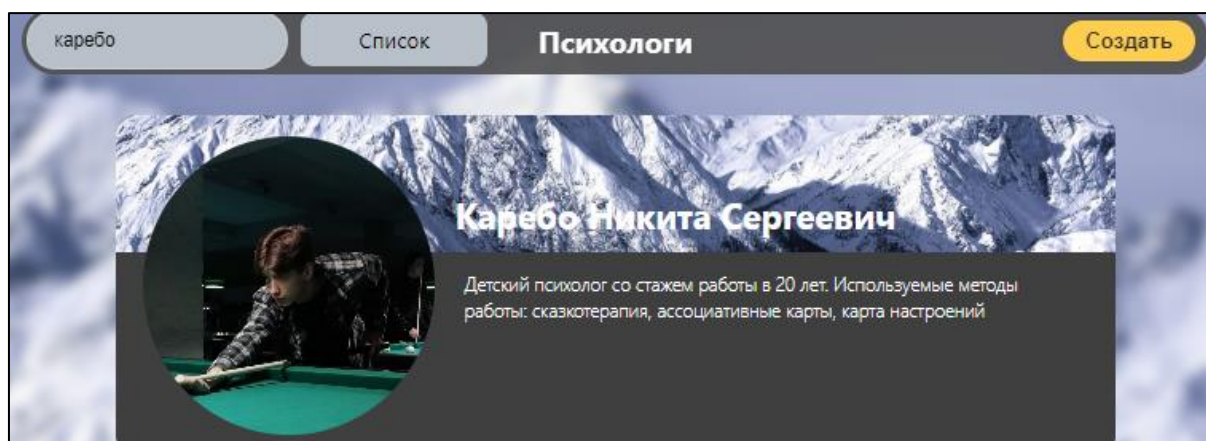


Рисунок 5.38 – Страница со списком психологов для администратора

При нажатии на кнопку «Создать» откроется форма для создания нового психолога, где администратор сможет заполнить необходимые поля.

Форма для создания психолога представлена на рисунке 5.39.

Рисунок 5.39 – Форма создания психолога

Далее администратору заполнить все доступные поля. При нажатии на кнопку «Создать» в системе будет создан аккаунт для психолога.

5.4.7 Удаление профиля для психолога

Чтобы удалить профиль психолога, администратор должен перейти на страницу со списком психологов, выбрав пункт «Психологи» в навигационной панели. Страница со списком психологов представлена на рисунке 5.40.

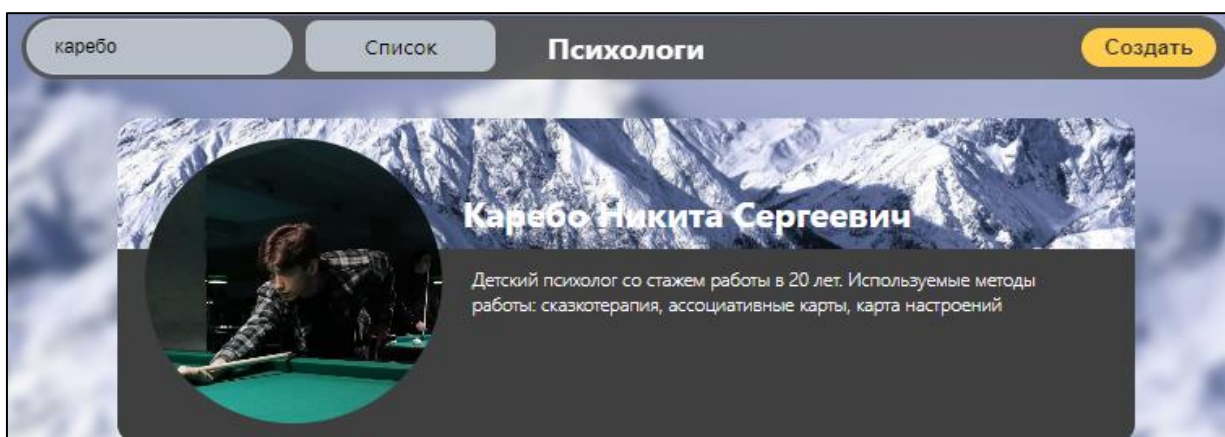


Рисунок 5.40 – Страница со списком психологов для администратора

При нажатии администратором на изображение психолога, выбранного из списка доступных психологов, открывается его профиль с детальной информацией. Страница с профилем психолога представлена на рисунке 5.41.

График работы	
Понедельник:	10:00 до 13:00
Вторник:	9:00 до 13:00
Среда:	8:00 до 14:00
Четверг:	8:00 до 13:00
Пятница:	8:00 до 13:00

Описание

Детский психолог со стажем работы в 20 лет. Используемые методы работы: сказкотерапия, ассоциативные карты, карта настроений

Рисунок 5.41 – Профиль психолога для администратора

Для удаления профиля психолога, администратору необходимо нажать на кнопку «Удалить». После этого аккаунт психолога будет удалён из базы данных.

5.4.8 Изменение информации о психологе, процедуре

Чтобы изменить информацию для психолога, администратор должен перейти на страницу со списком психологов психологического центра, выбрав пункт «Психологи» в навигационной панели. Страница со списком психологов представлена на рисунке 5.42.

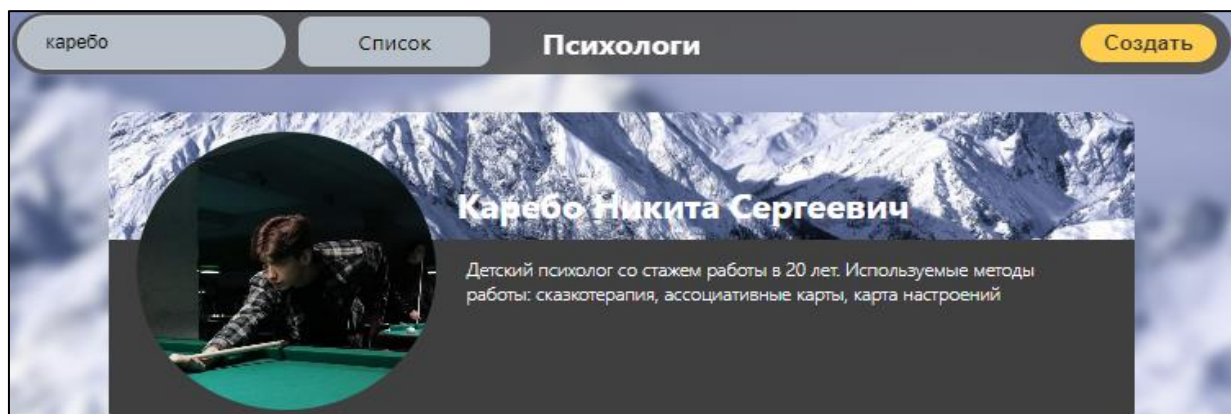


Рисунок 5.42 – Страница со списком психологов для администратора

При нажатии администратором на изображение психолога, выбранного из списка доступных психологов, открывается его профиль с детальной информацией. Страница с профилем психолога представлена на рисунке 5.43.

График работы			
Понедельник:	10:00	до	13:00
Вторник:	9:00	до	13:00
Среда:	8:00	до	14:00
Четверг:	8:00	до	13:00
Пятница:	8:00	до	13:00

Рисунок 5.43 – Профиль психолога для администратора

На данной странице администратор может поменять такие данные, как фамилия, имя, отчество (ФИО), научная степень, расписание, профессиональный опыт, описание и направления и фото. Для сохранения изменений администратору необходимо нажать на кнопку «Изменить».

Чтобы изменить информацию для услуги, администратор должен перейти на страницу со списком услуг, выбрав пункт «Услуги» в навигационной панели. Страница со списком услуг представлена на рисунке 5.44.

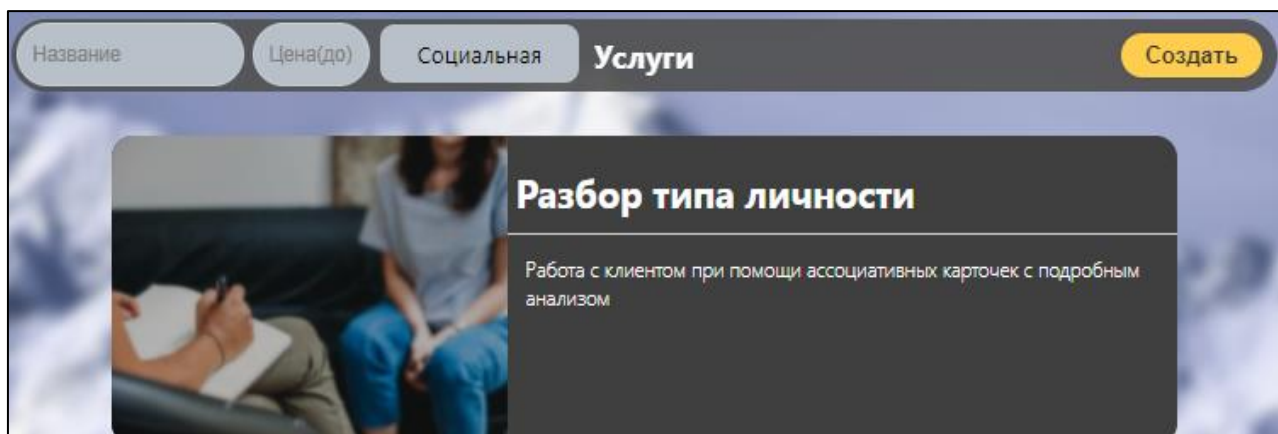


Рисунок 5.44 – Страница со списком услуг для администратора

Для изменения информации услуги, администратор должен нажать на изображение услуги из перечня всех услуг.

Далее администратору откроется страница с данной услугой, представлено на рисунке 5.45.

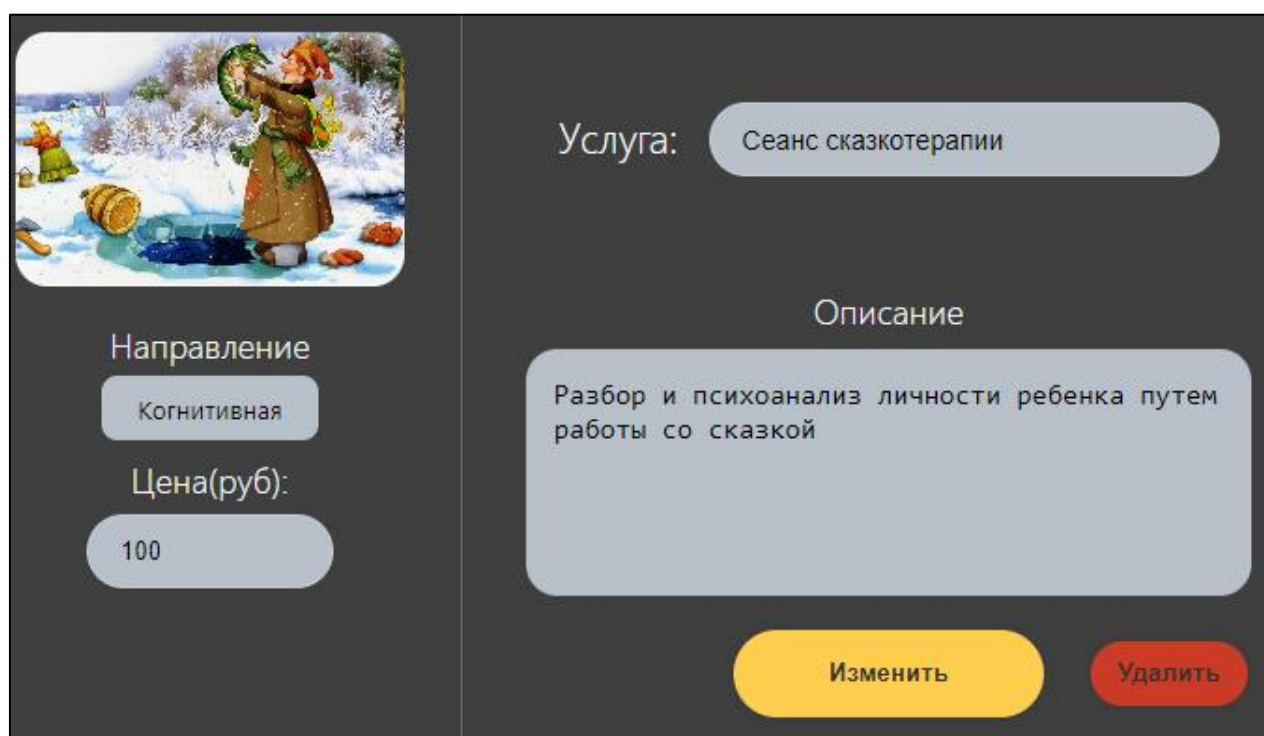


Рисунок 5.45 – Страница услуги для администратора

На странице администратор может изменить название услуги, направление, фото, цену и описание. Для сохранения изменений администратору необходимо нажать на кнопку «Изменить».

5.5 Выводы по разделу

1. Описано подробно руководство, в котором изложены действия, выполняющие пользователи разных ролей в системе: гостя, пользователя, психолога и администратора. Каждая роль имеет свои уникальные функции, предназначенные для удобства и эффективности взаимодействия с web-приложением.

2. Гости могут зарегистрироваться в системе, чтобы получить доступ к широкому спектру услуг и функций, доступных на платформе. Они могут искать услуги и психологов по различным критериям, используя фильтры для поиска психологов по специализациям, квалификации, опыту работы и другим параметрам. Гостям доступна информация о предоставляемых услугах, профилях психологов, их специализациях и квалификации, а также отзывы других пользователей, что помогает сделать выбор.

3. Пользователи могут пройти аутентификацию в системе, оформить запись на услугу, выбрав психолога, дату и время, редактировать личный профиль, добавлять и удалять отзывы, отменять запланированные услуги, а также изучать информацию о процедурах и специалистах.

4. Психологи могут войти в свою учетную запись, обновлять информацию о себе, включая квалификацию, специализации, опыт работы, расписание. Психологи могут создавать талоны для записи клиентов на консультации. В случае необходимости психологи могут отменить запись на консультацию или услугу, уведомив об этом клиента.

5. Администраторы могут создавать и удалять профили психологов, добавлять, редактировать и удалять услуги, управлять отзывами, создавать талоны для записи клиентов и редактировать данные психологов и услуг.

Заключение

1. В результате работы над проектом было разработано современное web-приложение «Clean Brain», которое предназначено для эффективной реализации услуг психологического консультирования и поддержки. Web-приложение поддерживает 4 роли пользователей: гость, пользователь, психолог и администратор. Каждая роль имеет уникальный набор прав и возможностей, что позволяет обеспечить персонализированный подход к взаимодействию с системой. Например, гости могут просматривать доступные услуги, пользователи – записываться на консультации, психологи – отслеживать запланированные сеансы, а администраторы – создавать услуги и профили психологов.

2. Web-приложение включает 16 ключевых функций, охватывающих весь спектр необходимого функционала: просмотр и запись на консультации, создание и управление отзывами, мгновенные уведомления о изменениях в расписании. Все эти функции были тщательно спроектированы и интегрированы в интерфейс, чтобы обеспечить максимальное удобство и простоту взаимодействия для различных категорий пользователей.

3. База данных web-приложения состоит из 11 таблиц, предназначенных для хранения и обработки информации о пользователях, услугах, сеансах, отзывах и других данных.

4. Архитектура web-приложения имеет несколько ключевых особенностей: серверная часть построена на Node.js с использованием фреймворка Express. Использование сервисов OpenAI, Cloudinary, Stripe является особенностью архитектуры web-приложения. В качестве клиентской части используется React, что обеспечивает масштабируемость web-приложения.

5. Общий объем программного кода web-приложения составил 8400 авторских строк. Общее количество тестов составило 22, а покрытие кода тестами web-приложения составило 100%.

В соответствии с полученным результатом работы web-приложения можно сделать вывод, что цель достигнута, а требования технического задания выполнены в полном объеме.

Список используемых источников

- 1 Центр Экстренной психологической помощи INSIDE [Электронный ресурс] / Режим доступа: <https://trevoga.by/> – Дата доступа: 03.04.2024.
- 2 Центр психологической помощи Grow Up [Электронный ресурс] / Режим доступа: <https://psiholog-minsk.by/> – Дата доступа: 27.03.2024.
- 3 Психологическая помощь Nora [Электронный ресурс] / Режим доступа: <https://www.nora.by> – Дата доступа: 11.04.2024.
- 4 Полное руководство по Node.js [Электронный ресурс] / Режим доступа: <https://nodejsdev-ru.pages.dev/guides/freecodecamp/> – Дата доступа: 10.04.2024.
- 5 Express.js 4.x API [Электронный ресурс] / Режим доступа: <https://expressjs.com/en/api.html> – Дата доступа: 10.04.2024.
- 6 Руководство по PostgreSQL [Электронный ресурс] / Режим доступа: <https://metanit.com/sql/postgresql/> – Дата доступа: 10.04.2024.
- 7 Prisma Documentation [Электронный ресурс] / Режим доступа: <https://www.prisma.io/docs/orm/overview/databases/postgresql> – Дата доступа: 10.04.2024.
- 8 jsonwebtoken [Электронный ресурс] / Режим доступа: <https://www.npmjs.com/package/jsonwebtoken> – Дата доступа: 10.04.2024.
- 9 socket.io npm [Электронный ресурс] / Режим доступа: <https://www.npmjs.com/package/socket.io> – Дата доступа: 10.04.2024.
- 10 Stripe [Электронный ресурс] / Режим доступа: <https://www.npmjs.com/package/stripe> – Дата доступа: 10.04.2024.
- 11 Cloudinary [Электронный ресурс] / Режим доступа: https://cloudinary.com/documentation/node_integration – Дата доступа: 10.04.2024.
- 12 OpenAI API [Электронный ресурс] / Режим доступа: <https://openai-docs.ru/docs/api-reference#introduction> – Дата доступа: 10.04.2024.
- 13 Repository Design Pattern [Электронный ресурс] / Режим доступа: <https://www.geeksforgeeks.org/repository-design-pattern/> – Дата доступа: 10.04.2024.
- 14 Unit of Work Pattern on a Node.js Application [Электронный ресурс] / Режим доступа: <https://dev.to/scheid/using-clean-architecture> – Дата доступа: 10.04.2024.
- 15 Apache JMeter [Электронный ресурс] / Режим доступа: <https://jmeter.apache.org/usermanual/index.html> – Дата доступа: 10.04.2024.

Приложение А

```

CREATE DATABASE PsychologicalCenter;

\c PsychologicalCenter;

CREATE TABLE Academic_Degree (
    Academic_Name VARCHAR(60) PRIMARY KEY
);

CREATE TABLE Client (
    Id_client UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    Name_Client VARCHAR NOT NULL,
    Surname_Client VARCHAR NOT NULL,
    Login_Client VARCHAR NOT NULL,
    Password_Client VARCHAR NOT NULL,
    Photo_Client VARCHAR,
    Mail_Client VARCHAR NOT NULL,
    Role_Client VARCHAR NOT NULL
);

CREATE TABLE Procedures (
    Id_Procedure UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    Name_Procedure VARCHAR NOT NULL,
    Price FLOAT NOT NULL,
    Depiction VARCHAR NOT NULL,
    Photo_Procedure VARCHAR
);

CREATE TABLE ProcedureSpecialization (
    Id_Procedure UUID NOT NULL,
    Spezialization_Name VARCHAR(60) NOT NULL,
    PRIMARY KEY (Id_Procedure, Spezialization_Name),
    FOREIGN KEY (Id_Procedure) REFERENCES Procedures(Id_Procedure)
ON DELETE CASCADE,
    FOREIGN KEY (Spezialization_Name) REFERENCES
Specialization(Spezialization_Name) ON DELETE CASCADE
);

CREATE TABLE Psychologist (
    Id_Psychologist UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    Name_Psychologist VARCHAR NOT NULL,
    Surname_Psychologist VARCHAR NOT NULL,
    Patronymic_Psychologist VARCHAR NOT NULL,
    Mail_Psychologist VARCHAR NOT NULL,
    Experience SMALLINT NOT NULL,
    Photo_Psychologist VARCHAR,
    Description VARCHAR,
    Login_Psychologist VARCHAR NOT NULL,
    Password_Psychologist VARCHAR NOT NULL,
    Degree VARCHAR(60),
    FOREIGN KEY (Degree) REFERENCES Academic_Degree(Academic_Name)
ON DELETE SET NULL

```

```

);

CREATE TABLE PsychologistSpecialization (
    Id_Psychologist UUID NOT NULL,
    Spezialisierung_Name VARCHAR(60) NOT NULL,
    PRIMARY KEY (Id_Psychologist, Spezialisierung_Name),
    FOREIGN KEY (Id_Psychologist) REFERENCES
Psychologist(Id_Psychologist) ON DELETE CASCADE,
    FOREIGN KEY (Spezialisierung_Name) REFERENCES
Specialization(Spezialisierung_Name) ON DELETE CASCADE
);

CREATE TABLE Review (
    Id_Review UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    Id_Client UUID NOT NULL,
    Review VARCHAR NOT NULL,
    FOREIGN KEY (Id_Client) REFERENCES Client(Id_client) ON DELETE
CASCADE
);

CREATE TABLE Specialization (
    Spezialisierung_Name VARCHAR(60) PRIMARY KEY
);

CREATE TABLE Timetable (
    Id_Psychologist UUID PRIMARY KEY,
    MondStart TIME,
    MondEnd TIME,
    TueStart TIME,
    TueEnd TIME,
    WenStart TIME,
    WenEnd TIME,
    ThuStart TIME,
    ThuEnd TIME,
    FriStart TIME,
    FriEnd TIME,
    FOREIGN KEY (Id_Psychologist) REFERENCES
Psychologist(Id_Psychologist) ON DELETE CASCADE
);

CREATE TABLE Voucher (
    Id_Voucher UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    Date_Voucher TIMESTAMP NOT NULL,
    Time_Voucher_Start TIME NOT NULL,
    Time_Voucher_End TIME NOT NULL,
    Id_Psychologist UUID NOT NULL,
    Ordered CHAR(10) NOT NULL,
    FOREIGN KEY (Id_Psychologist) REFERENCES
Psychologist(Id_Psychologist) ON DELETE CASCADE
);

CREATE TABLE Booking (
    ID_Booking UUID PRIMARY KEY DEFAULT gen_random_uuid(),

```

```
    Date_Booking TIMESTAMP NOT NULL,  
    Id_Procedure UUID NOT NULL,  
    Id_Client UUID NOT NULL,  
    Id_Voucher UUID NOT NULL,  
    Time_Booking TIME,  
    FOREIGN KEY (Id_Client) REFERENCES Client(Id_client) ON DELETE  
CASCADE,  
    FOREIGN KEY (Id_Procedure) REFERENCES Procedures(Id_Procedure)  
ON DELETE CASCADE,  
    FOREIGN KEY (Id_Voucher) REFERENCES Voucher(Id_Voucher) ON  
DELETE CASCADE  
);
```

Листинг – Скрипт создания базы данных

Приложение Б

```

class ClientRepository {
  constructor(prismaClient) {
    this.prisma = prismaClient;
  }
  async getAll() {
    return this.prisma.client.findMany();
  }
  async findById(Id_client) {
    return this.prisma.client.findUnique({ where: { Id_client } });
  }
  async findByLogin(Login_Client) {
    return this.prisma.client.findMany({ where: { Login_Client } });
  }
  async findByEmail(Mail_Client) {
    return this.prisma.client.findMany({ where: { Mail_Client } });
  }
  async create(data) {
    return this.prisma.client.create({ data });
  }
  async update(Id_client, user) {
    return this.prisma.client.update({
      where: { Id_client },
      data: {
        Name_Client: user.Name_Client,
        Surname_Client: user.Surname_Client,
        Photo_Client: user.Photo_Client,
      },
    });
  }
  async delete(Id_client) {
    return this.prisma.client.delete({ where: { Id_client } });
  }
}

export default ClientRepository;

```

Листинг – Реализация паттерна Repository

```

class UnitOfWork {
  constructor() {
    this.prisma = new PrismaClient();
    this.clientRepository = new ClientRepository(this.prisma);
    this.psychologistRepository = new
PsychologistRepository(this.prisma);
    this.proceduresRepository = new
ProceduresRepository(this.prisma);
    this.bookingRepository = new BookingRepository(this.prisma);
    this.degreeRepository = new DegreeRepository(this.prisma);
    this.specializationRepository = new
SpecializationRepository(this.prisma);
    this.reviewRepository = new ReviewRepository(this.prisma);
  }
}

```



```
        this.voucherRepository = new VoucherRepository(this.prisma);
        this.psychologistSpecializationRepository = new
PsychologistSpecializationRepository(this.prisma);
        this.refreshSessionRepository = new
RefreshSessionRepository(this.prisma);
        this.timeTableRepository = new
TimeTableRepository(this.prisma);
        this.emailAuthenticationRepository = new
EmailAuthenticationRepository(this.prisma);
        this.procedureSpecializationRepository = new
ProcedureSpecializationRepository(this.prisma);
    }
}

export default UnitOfWork;
```

Листинг – Реализация паттерна Unit Of Work

Приложение В

```
import { Router } from "express";
import AdminController from "../controllers/Admin.js";
import AdminValidator from "../validators/Admin.js";

const router = Router();
router.post("/psychologists", AdminValidator.createPsychologist,
AdminController.createPsychologist);
router.post("/psychologists/:Id_Psychologist",
AdminValidator.updatePsychologist,
AdminController.updatePsychologist);
router.post("/procedures/", AdminValidator.createProcedure,
AdminController.createProcedure);

router.post("/procedures/:Id_Procedure",
AdminValidator.updateProcedure, AdminController.updateProcedure);
router.post("/vouchers/:Id_Psychologist",
AdminController.createPsychologistVoucher);
router.delete("/psychologists/:Id_Psychologist",
AdminValidator.deletePsychologist ,
AdminController.deletePsychologist);
router.delete("/procedures/:Id_Procedure",
AdminValidator.deleteProcedure , AdminController.deleteProcedure);
router.delete("/reviews/:Id_Review",
AdminValidator.deleteReviewClient ,
AdminController.deleteReviewClient);

export default router;
```

Листинг – Реализация маршрутизатора Admin

```
import bcrypt from "bcryptjs";
import TokenService from "../Token.js";
import { NotFound, Forbidden, Conflict, Unauthorized } from
"../utils/Errors.js";
import { ACCESS_TOKEN_EXPIRATION } from "../constants.js";
import StaticClass from "../static/StaticClass.js";
import cloundinary from "cloundinary"
import EmailAuthenticationService from "../EmailAuthentication.js";

class AuthService {
  static async signIn( Login_Client, password, fingerprint, Id_Auth,
Code_Client) {
    let isPsychologistAccount = false;
    const userData = await
StaticClass.unit.clientRepository.findByLogin(Login_Client);
    const psychogitsData = await
StaticClass.unit.psychologistRepository.findByLogin(Login_Client);

    if(userData.length == 0) {
      if (psychogitsData.length == 0)
        throw new NotFound("Пользователь не найден");
```

```

        else
            isPsychologistAccount = true;
        }

        const passwordUser = isPsychologistAccount == true ?
psychologitsData[0].Password_Psychologist :
userData[0].Password_Client;
        const mailUser = isPsychologistAccount == true ?
psychologitsData[0].Mail_Psychologist : userData[0].Mail_Client;
        const idUser = isPsychologistAccount == true ?
psychologitsData[0].Id_Psychologist : userData[0].Id_client;
        const roleUser = isPsychologistAccount == true ? "Psychologist"
: userData[0].Role_Client;
        const loginUser = isPsychologistAccount == true ?
psychologitsData[0].Login_Psychologist : Login_Client;
        const isPasswordValid = bcrypt.compareSync(password,
passwordUser);

        if(!isPasswordValid){
            throw new Unauthorized("Неверный логин или пароль");
        }
        await EmailAuthenticationService.verifyEmailCode(Id_Auth,
Code_Client, mailUser)
        const payload = { Id_client: idUser, Role_Client: roleUser,
Login_Client: loginUser}
        const accessToken = await
TokenService.generateAccessToken(payload);
        const refreshToken = await
TokenService.generateRefreshToken(payload);
        await
StaticClass.unit.refreshSessionRepository.createRefreshSession({clie
nt_id: payload.Id_client, refresh_token: refreshToken, finger_print:
fingerprint.hash});

        const updatedPsychologitsData = {
            ...psychologitsData[0],
            Role_Client: 'Psychologist'
        };

        return {
            accessToken,
            refreshToken,
            accessTokenExpiration: ACCESS_TOKEN_EXPIRATION,
            user : isPsychologistAccount == true ? updatedPsychologitsData
: userData[0],
            isPsychologistAccount
        };
    }

    static async signUp( Mail_Client, Login_Client, password,
fingerprint, Id_Auth, Code_Client) {
        const userData = await
StaticClass.unit.clientRepository.findByLogin(Login_Client);

```

```

    const psychologistsData = await
StaticClass.unit.psychologistRepository.findByLogin(Login_Client);

    if (userData.length != 0 || psychologistsData.length != 0){
        throw new Conflict("Пользователь с таким именем уже
существует");
    }

    const userDataEmail = await
StaticClass.unit.clientRepository.findByEmail(Mail_Client);
    const psychologistsDataEmail = await
StaticClass.unit.psychologistRepository.findByEmail(Mail_Client);

    if (userDataEmail.length != 0 || psychologistsDataEmail.length){
        throw new Conflict("Пользователь с такой почтой уже
существует");
    }
    await EmailAuthenticationService.verifyEmailCode(Id_Auth,
Code_Client, Mail_Client)

    const myCloud = await
cloudinary.v2.uploader.upload("./images/bear.png")
    const Photo_Client = myCloud.secure_url;
    const Name_Client = "Name";
    const Surname_Client = "Surname";
    const Role_Client = "Client";
    const Password_Client = bcrypt.hashSync(password, 8);
    const user = await
StaticClass.unit.clientRepository.create({Name_Client, Surname_Client
, Login_Client, Password_Client, Photo_Client, Mail_Client, Role_Client
});

    const payload = { Id_client: user.Id_client, Role_Client,
Login_Client };
    const accessToken = await
TokenService.generateAccessToken(payload);
    const refreshToken = await
TokenService.generateRefreshToken(payload);
    await
StaticClass.unit.refreshSessionRepository.createRefreshSession({clie
nt_id: payload.Id_client, refresh_token: refreshToken, finger_print:
fingerprint.hash});

    return {
        accessToken,
        refreshToken,
        accessTokenExpiration: ACCESS_TOKEN_EXPIRATION,
        user
    };
}

static async logOut(refreshToken) {

```

```

    await
    StaticClass.unit.refreshSessionRepository.deleteRefreshSession(refreshToken);
  }

  static async refresh({ fingerprint, currentRefreshToken }) {
    if(!currentRefreshToken){
      throw new Unauthorized();
    }

    const refreshSession = await
    StaticClass.unit.refreshSessionRepository.getRefreshSession(currentRefreshToken);
    let isPsychologistAccount = false;

    if(refreshSession.length == 0){
      throw new Unauthorized();
    }
    if(refreshSession[0].finger_print !== fingerprint.hash){
      throw new Forbidden();
    }

    await
    StaticClass.unit.refreshSessionRepository.deleteRefreshSession(currentRefreshToken);
    let payload;
    try{
      payload = await
      TokenService.verifyRefreshToken(currentRefreshToken);
    }catch (error){
      throw new Forbidden(error);
    }
    const userData = await
    StaticClass.unit.clientRepository.findByLogin(payload.Login_Client);
    const psychologistData = await
    StaticClass.unit.psychologistRepository.findByLogin(payload.Login_Client);

    if(userData.length == 0) {
      if(psychologistData.length != 0)
        isPsychologistAccount = true;
    }

    const idUser = isPsychologistAccount == true ?
    psychologistData[0].Id_Psychologist : userData[0].Id_client;
    const roleUser = isPsychologistAccount == true ? "Psychologist" :
    userData[0].Role_Client;
    const loginUser = isPsychologistAccount == true ?
    psychologistData[0].Login_Psychologist : userData[0].Login_Client;
    const updatedPsychologitsData = {
      ...psychologistData[0],
      Role_Client: 'Psychologist'
    };
  };

```

```

    const user = isPsychologistAccount == true ?
updatedPsychologitsData : userData[0];

    const actualPayload = { Id_client : idUser, Login_Client:
loginUser, Role_Client: roleUser};

    const accessToken = await
TokenService.generateAccessToken(actualPayload);
    const refreshToken = await
TokenService.generateRefreshToken(actualPayload);

    await
StaticClass.unit.refreshSessionRepository.createRefreshSession({clie
nt_id: idUser, refresh_token: refreshToken, finger_print:
fingerprint.hash});

    return {
        accessToken,
        refreshToken,
        accessTokenExpiration: ACCESS_TOKEN_EXPIRATION,
        user
    };
}
}

export default AuthService;

```

Листинг – Реализация сервиса Auth