

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий
Кафедра Информационных систем и технологий
Специальность 1–40 01 01 «Программное обеспечение информационных технологий»
Специализация 1–40 01 01 10 «Программное обеспечение информационных технологий
(программирование интернет–приложений)»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту на тему:**

Web-приложение для психологического центра «Clean Brain»

Выполнил студент Карebo Никита Сергеевич
(Ф.И.О.)

Руководитель проекта ст.преп. Дубовик М.В.
(учен. степень, звание, должность, подпись, Ф.И.О.)

И.о. заведующего кафедрой ст.преп. Блинова Е.А.
(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой _____

Содержание

Введение	3
1 Постановка задачи	4
1.1 Обзор аналогов	4
1.1.1 INSIDE.....	4
1.1.2 Grow Up.....	5
1.1.3 Nora.....	6
1.2 Формирование требований	7
2 Проектирование web-приложения	9
2.1 Архитектура приложения.....	9
2.2 Проектирование базы данных	11
3 Разработка web-приложения.....	14
3.1 Разработка клиентской части web-приложения	14
3.2 Разработка серверной части web-приложения.....	18
4 Тестирование web-приложения.....	24
5 Руководство пользователя	30
5.1 Руководство гостя	30
5.2 Руководство пользователя	33
5.3 Руководство администратора	36
5.4 Установка приложения.....	40
Заключение	301
Список используемых источников.....	42
Приложение А	43
Приложение Б.....	44

Введение

Трудно представить общество, да и человека в целом, без эмоций. Эмоции – это вся наша жизнь. Отрицательные они или положительные. Но эмоции – есть эмоции. И это нормально. Это естественная часть нашего существования. Они отражают наше внутреннее состояние, помогают выражать четче и ярче наши мысли и желания. И, конечно, они имеют влияние на наше общее благополучие. Поэтому для каждого из нас абсолютно естественно стремиться разделить с кем-то свои эмоции и свои переживания.

Но, к сожалению, часто случается, что поиск квалифицированного специалиста становится довольно долгим и трудоемким процессом. В наше время, когда психологическое здоровье занимает всё больший приоритет в жизни человека, возникает необходимость в инновационных решениях, которые могут сделать доступ к качественной психологической помощи более удобным и эффективным.

Данное приложение «Clean Brain» как раз и предоставляет клиентам психологического центра такую возможность. Поиск специалистов становится более комфортным и, что главное, понятным и простым. Клиенты имеют возможность электронной записи на консультацию, ознакомившись при этом с расписанием врачей и выбрав для себя самое удобное время. Если необходимо, запись к врачу можно легко отменить либо перенести. Приложение также открывает доступ для просмотра пользователями профилей психологов, где каждый заинтересованный может ознакомиться с предоставляемыми услугами.

Кроме прочего, приложение становится хорошим решением и для администраторов центра, позволяя им эффективно управлять базой данных, контролировать и регулировать расписание специалистов.

Таким образом, приложение для психологического центра «Clean Brain» – это надежный помощник, как для клиентов, так и для администраторов. Ведь комфортный поиск специалистов и упрощенное управление рабочим процессом – дважды востребованный критерий.

Основная цель данного курсового проекта заключается в разработке программного средства для обеспечения функциональности психологического центра. Для достижения цели необходимо выполнить следующие задачи:

- рассмотреть и проанализировать существующие аналоги;
- спроектировать web-приложение;
- разработать web-приложение;
- протестировать web-приложение;
- описать руководство пользователя.

В пояснительной записке содержится подробное описание реализации каждой из поставленных задач.

1 Постановка задачи

В данной главе проведем анализ основных конкурентов, что является важным этапом для разработки. Этот шаг позволит выявить сильные и слабые стороны существующих решений на рынке. После анализа перейдем к формированию требований приложения. Этот процесс включает определение основных функций и характеристик, необходимых для успешной конкуренции и удовлетворения потребностей пользователей.

1.1 Обзор аналогов

Одним из важных критериев выбора стратегии разработки приложения является анализ конкурентов. Если создать приложение типичное с другими, это не поспособствует расширению клиентской базы. Анализ конкуренции и реализация требований, востребованных пользователями, позволит приложению стать удобнее и в то же время эффективнее.

1.1.1 INSIDE

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

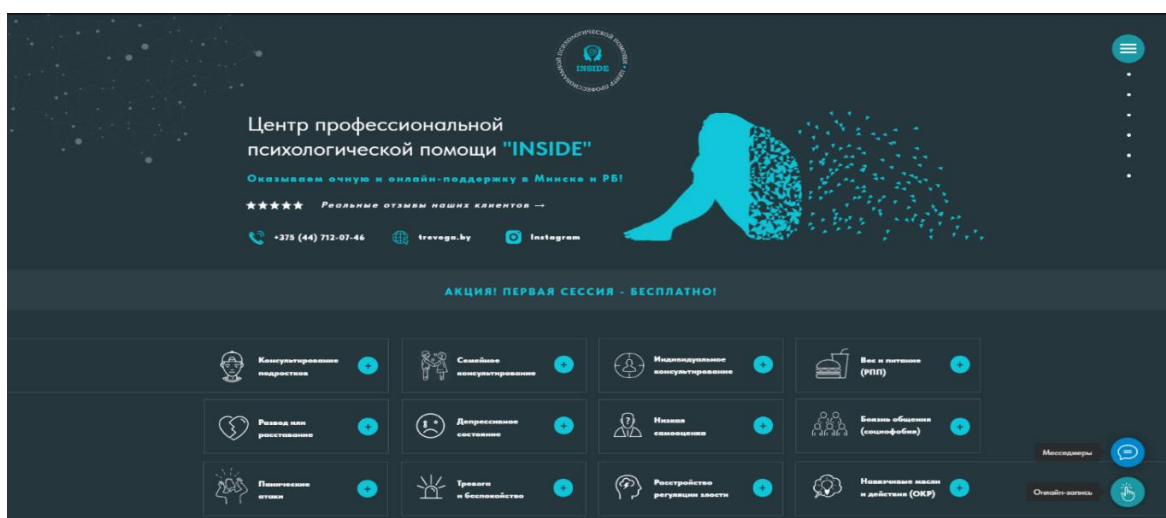


Рисунок 1.1 – Сайт «INSIDE»

Данный сайт имеет достаточно грамотный и красивый интерфейс, предоставляет удобную навигационную панель. Также есть возможность регистрации аккаунта через социальные сети Facebook, ВКонтакте, Одноклассники. 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.

Также имеется вкладка «Новости», где можно просмотреть последние достижения и события в области психологии. Как было замечено, его обновляют нечасто, и данная вкладка не несет для пациента никакой нужной информации, лишь создает визуальный шум, что зачастую только может сбивать с толку.

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.

1.1.2 Grow Up

Следующий сайт – «Grow Up», который также является популярным среди пользователей. На рисунке 1.2 представлен интерфейс сайта.

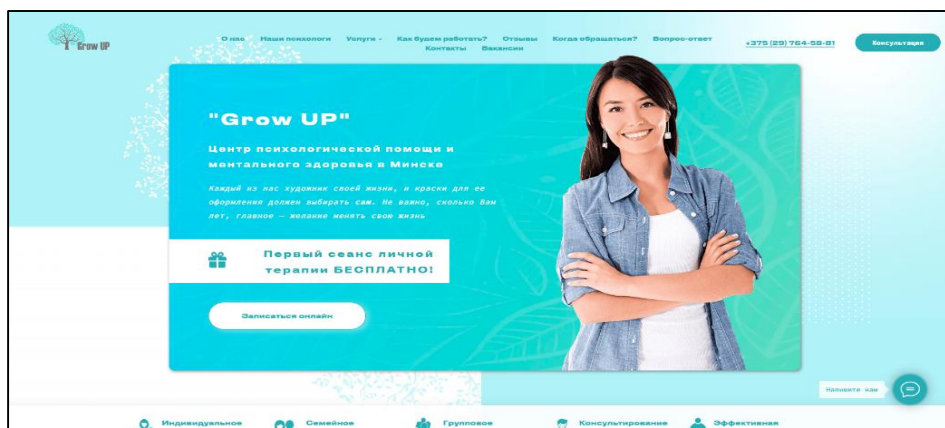


Рисунок 1.3 – Сайт «Grow Up»

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.

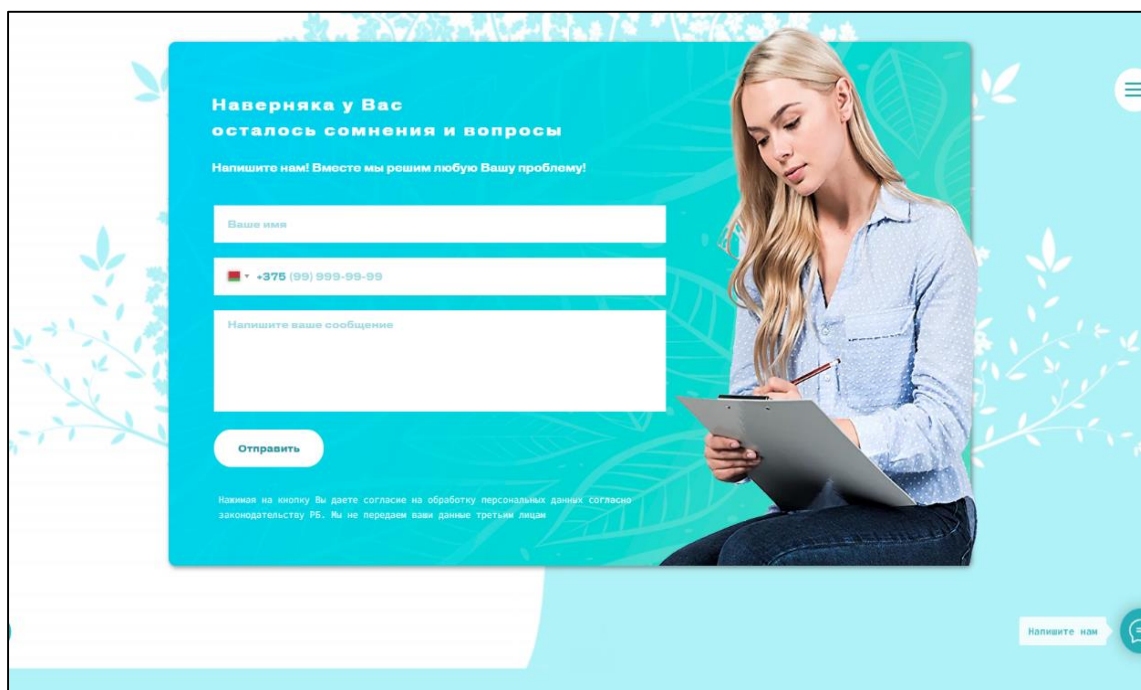


Рисунок 1.4 – Бронирование сеанса на сайте «Grow Up»

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. Также стоит отметить, что отсутствие отзывов от клиентов уменьшает доверие к ресурсу и может привести к снижению его популярности и упущению потенциальных клиентов.

1.1.3 Nora

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

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.

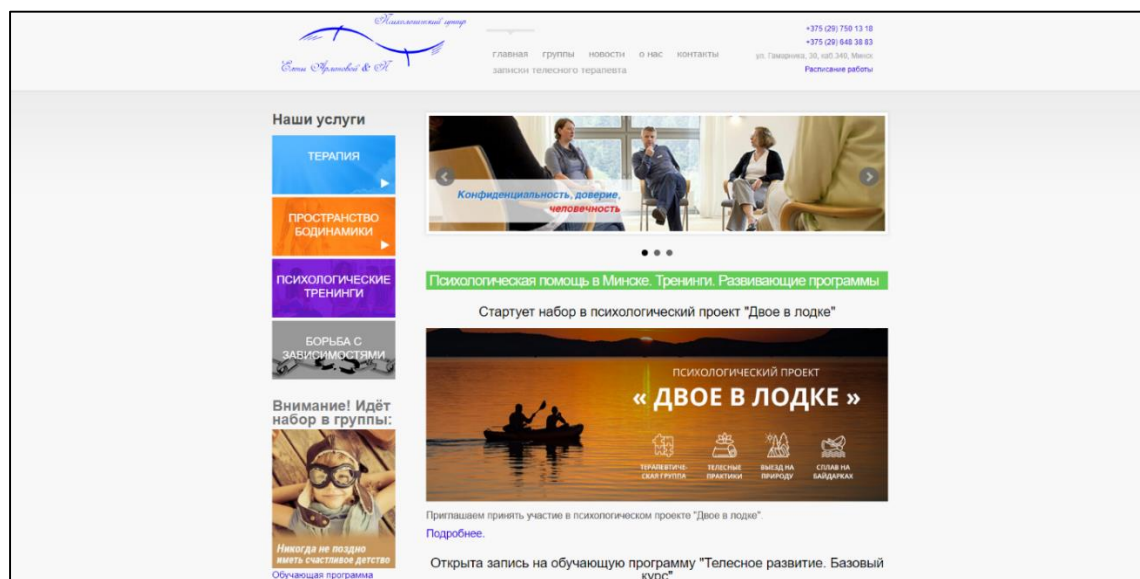


Рисунок 1.5 – Сайт «Nora»

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. Есть возможность позвонить лечащему врачу или написать ему в любой доступной соцсети.

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

1.2 Формирование требований

Web-приложение «Clean Brain» должно реализовать три основные категории функциональности. Первой категорией является функционал, который предоставляет ряд возможностей и инструментов для обычных пользователей:

1. Регистрация и профиль: клиенты могут создать свою учетную запись, заполнив необходимую информацию о себе.

2. Просмотр и выбор специалистов: клиенты имеют возможность просматривать профили психологов, узнать об их квалификации, специализации. Они могут выбрать психолога, соответствующего их потребностям и предпочтениям.

3. Выбор услуги: клиенты могут выбрать желаемую услугу, а также просмотреть подробную информацию о желаемой услуге.

4. Просмотр текущих назначенных сеансов: если у клиента есть назначенные сеансы с психологом, он может всегда просмотреть либо же отредактировать текущие сеансы.

5. Просмотр и добавление отзывов: клиент всегда может просмотреть добавленные другими пользователями приложения отзывы, а также оставить свой.

6. Просмотр информации о психологическом центре: пользователь всегда может подробно изучить информацию о психологическом центре.

Второй категорией является функционал, реализованный для администраторов. Данный функционал включает в себя:

1. Управление психологами: администратор имеет возможность добавлять новых психологов в систему, редактировать их профили и управлять их доступностью для клиентов. Он также может просматривать и контролировать информацию о квалификации, расписании психологов.

2. Управление услугами: администратор может добавлять новые услуги, а также их редактировать. Эта функция позволяет администратору гибко управлять списком доступных услуг, обновлять их описание, параметры и стоимость в соответствии с текущими требованиями и потребностями центра.

3. Управление отзывами: администратору предоставляется возможность просматривать все отзывы, а также если какие-то отзывы не соответствуют правилам или содержат неприемлемый контент, их можно будет удалить. Последней категорией является функционал, предоставляемый для гостей, который поможет получить представление о психологическом центре и принять решение о дальнейшем взаимодействии. Данный функционал включает в себя:

1. Просмотр информации о психологах: гость может получить информацию о психологах, работающих в психологическом центре. Он может прочитать их профили, узнать о квалификации, специализации и опыте каждого психолога.

2. Просмотр услуг и описаний: гость может ознакомиться с различными услугами, предлагаемыми психологическим центром, и получить детальное описание каждой из них.

3. Просмотр информации о психологическом центре: гость может получить доступ к общей информации о психологическом центре, такой как его миссия, ценности, основные услуги и преимущества.

Исходя из вышесказанного определим основные задачи для web-приложения психологического центра «Clean Brain»:

- обеспечение возможности регистрации и авторизации;
- наличие подтверждения почты при регистрации или авторизации;
- поддержка ролей администратора, пользователя и гостя;
- обеспечение предоставления информации о психологах и услугах;
- обеспечение предоставления информации о центре, его миссии;
- обеспечение редактирования профиля пользователя;
- обеспечение просмотра и управления текущими сеансами;
- обеспечение возможности оставлять и удалять отзывы;
- обеспечение поиска услуг или психологов по различным критериям;
- обеспечение управления списком психологов и их профилями;
- обеспечение управления услугами, включая добавление и редактирование;
- наличие оповещения клиента и психолога при оформлении услуги на почту.

2 Проектирование web-приложения

2.1 Архитектура приложения

Проект построен по принципам клиент-серверной архитектуры, где взаимодействие между компонентами обеспечивается через сеть. В роли сервера выступает приложение, основанное на Node.js, которое обрабатывает запросы, выполняет бизнес-логику и предоставляет доступ к данным. На другом конце – клиентское приложение, построенное с использованием React, создающее пользовательский интерфейс и отправляющее запросы на сервер для получения информации или выполнения операций.

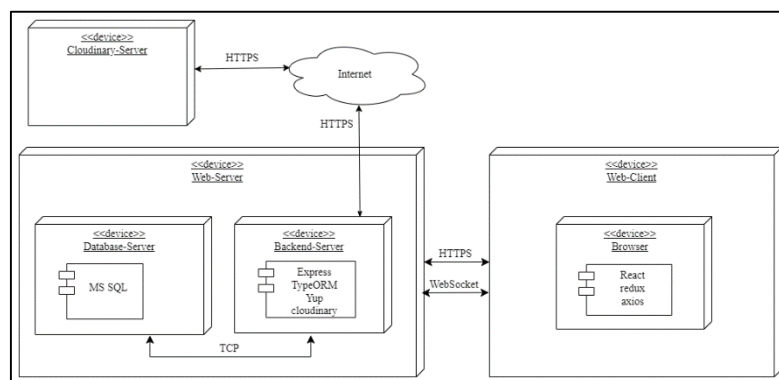


Рисунок 2.1 – Схема развёртывания

Клиент-серверная архитектура широко применяется в сфере разработки приложений. Она обеспечивает четкое разделение функций между серверной и клиентской частями приложения, облегчая его поддержку и масштабирование.

На рисунке 2.2 изображена диаграмма последовательности для авторизации пользователя в приложении «Clean Brain».



Рисунок 2.2 – Диаграмма последовательности для авторизации пользователя

Данная диаграмма последовательности наглядно иллюстрирует процесс авторизации пользователя и вывода результата. Она демонстрирует

последовательность шагов, которые происходят при попытке пользователя войти в систему психологического центра.

На рисунке 2.3 изображена диаграмма вариантов использования для приложения психологического центра «Clean Brain».

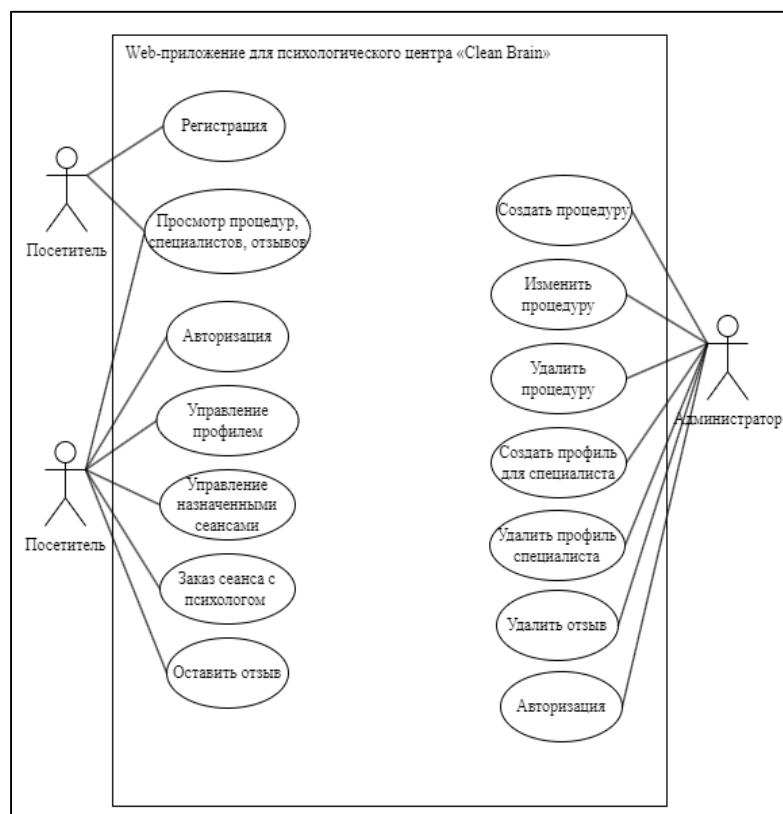


Рисунок 2.3 – Диаграмма вариантов использования

Данная диаграмма вариантов использования наглядно демонстрирует функциональность приложения «Clean Brain» для различных ролей пользователей, таких как администратор, гость и авторизованный пользователь. Она позволяет легко понять, какие функции доступны каждому из этих пользователей и как они могут взаимодействовать с приложением.

Авторизованные пользователи могут управлять персональным профилем: заполнить необходимую информацию, такую как имя, фамилия, логин, и пароль. Также они могут просматривать профили психологов, узнать их квалификацию, специализацию, опыт работы, и выбрать наиболее подходящего специалиста. Помимо этого, клиенты имеют возможность выбирать услуги, предоставляемые центром, просматривать подробную информацию о каждой из них. Они могут видеть свои назначенные сеансы с психологами и вносить изменения при необходимости. Кроме того, пользователи могут просматривать отзывы о психологическом центре, оставленные другими пользователями, и добавлять свои.

Администраторы имеют возможность управления базой психологов, включая добавление новых профилей, редактирование существующих. Они имеют возможность управлять услугами, добавляя новые или изменяя существующие, просматривать и удалять отзывы, если это необходимо.

Гости имеют возможность получить общую информацию о психологическом центре без необходимости регистрации.

Проект построен по принципам клиент-серверной архитектуры, где взаимодействие между компонентами обеспечивается через сеть.

2.2 Проектирование базы данных

Данное приложение использует базу данных с названием «Psychological_Center» и работает на платформе Microsoft SQL Server 2022 [1]. Это мощное и надежное решение для хранения, управления и обработки данных, которое обеспечивает эффективное взаимодействие с базой данных для данного психологического центра.

Чтобы составить визуальную взаимосвязанную структуру нашей базы данных, нам необходимо продумать, какая информация будет храниться в этих таблицах, после этого создать связи с помощью первичных и внешних ключей.

Логическая схема базы данных представлена на рисунке 2.5.

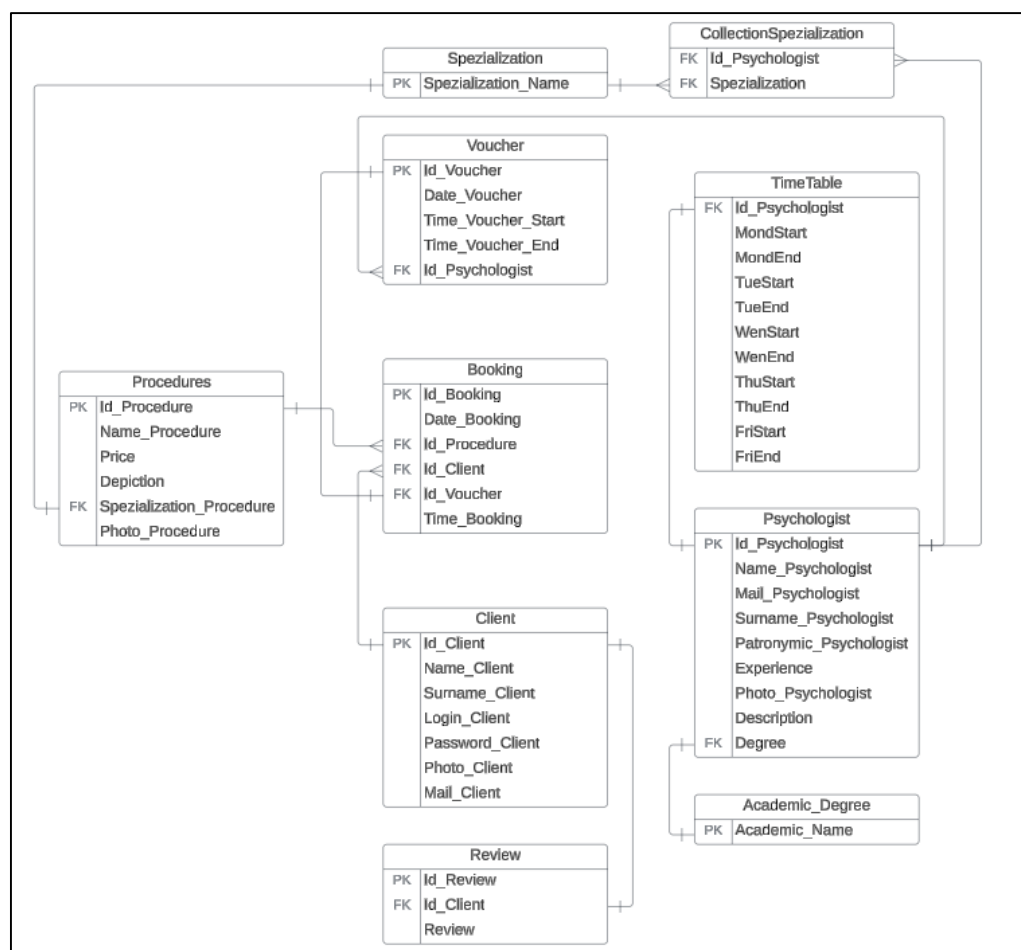


Рисунок 2.5 – Логическая схема базы данных

Таблица «Client» состоит из 7 столбцов, где поле «Id_Client» является первичным ключом. «Id_client» – id клиента, поле «Name_Client» хранит имя пользователя, «Surname_Client» – фамилию, «Login_Client» – логин клиента, «Password_Client» – пароль, «Photo_Client» – изображение, а поле «Mail_client»

хранит в себе почту клиента, на которую будут отправляться уведомления.

Таблица «Psychologist» содержит информацию о психологах. Она связана с таблицей «Academic_Degree» через поле «Degree».

В данной таблице поле «Id_Psychologist» является первичным ключом, а также хранит в себе id психолога. «Name_Psychologist» содержит имя психолога, «Surname_Psychologist» – фамилия психолога, «Mail_Psychologist» – почта психолога, «Patronymic_Psychologist» – отчество, «Degree» – ученная степень психолога, «Experience» – опыт, «Photo_Psychologist» – фотография психолога, «Description» – описание психолога.

Таблица «Spezialization» содержит информацию о специализациях.

В данной таблице поле «Spezialization_Name» является первичным ключом и содержит название специализации.

Таблица «Academic_Degree» содержит информацию о ученых степенях.

В данной таблице поле «Academic_Name» является первичным ключом и содержит название ученой степени.

Таблица «Voucher» содержит информацию о талонах. Она связана с таблицей «Psychologist» через поле «Id_Psychologist».

В данной таблице поле «Id_Voucher» является первичным ключом и содержит id талона, «Date_Voucher» – дата талона, «Time_Voucher_Start» – время начала приема, «Time_Voucher_End» – время окончания приема, «Id_Psychologist» – id психолога, «Ordered» – поле, оповещающее, что талон забронирован или нет.

Таблица «Procedures» содержит информацию о процедурах. Она связана с таблицей «Specialization» через поле «Specialization_Procedure».

В данной таблице поле «Id_Procedure» является первичным ключом, а также оно содержит id процедуры, «Name_procedure» – название процедуры, «Depiction» – описание, «Price» – цена процедуры, «Photo_Procedure» – фотография процедуры, «Spezialization_Procedure» – специализация.

Таблица «Timetable» содержит информацию о расписании психолога. Она связана с таблицей «Psychologist» через поле «Id_Psychologist».

В данной таблице поле «Id_Psychologist» является первичным ключом, а также содержит id психолога, «MondStart» – с какого времени психолог начинает работать в понедельник, «MondEnd» – во сколько психолог заканчивает работать в понедельник, «TueStart» – с какого времени психолог начинает работать во вторник, «TueEnd» – во сколько психолог заканчивает работать во вторник, «WenStart» – с какого времени психолог начинает работать в среду, «WenEnd» – во сколько психолог заканчивает работать в среду, «ThuStart» – с какого времени психолог начинает работать в четверг, «ThuEnd» – во сколько психолог заканчивает работать в четверг, «FriStart» – с какого времени психолог начинает работать в пятницу, «FriEnd» – во сколько психолог заканчивает работать в пятницу.

Таблица «Booking» содержит информацию о талонах. Таблица «Booking» содержит информацию о заказах. Она связана с таблицей «Client» через поле «Id_Client», с таблицей «Voucher» – через поле «Id_Voucher», с таблицей «Procedure» – через поле «Id_Procedure».

В данной таблице поле «ID_Booking» является первичным ключом, а также

содержит id заказа, «Date_Booking» – дата заказа, «Id_Procedure» – id процедуры, «Id_Client» – id клиента, «Id_Voucher» – id талона, «Time_Booking» – время заказа.

Таблица «Review» содержит информацию о отзывах. Она связана с таблицей «Client» через поле «Id_Client». В данной таблице поле «Id_Review» является первичным ключом, а также содержит id отзыва, поле «Id_Client» содержит id пользователя, «Review» – сам отзыв.

Таблица «Review» содержит информацию об отзывах. Она связана с таблицей «Client» через поле «Id_Client». В данной таблице поле «Id_Review» является первичным ключом, а также содержит id отзыва, поле «Id_Client» содержит id пользователя, «Review» – сам отзыв.

Таблица «CollectionSpezialization» содержит информацию о специализациях всех психологов. Она связана с таблицей «Psychologist» через поле «Id_Psychologist» и с таблицей «Spezialization» через поле «Spezialization». Поле «Psychologist» содержит id психолога, поле «Spezialization» содержит название специализации.

База данных обладает высокой степенью структурированности, что обеспечивает легкость в организации и доступе к данным. Каждая таблица имеет четко определенные поля, отражающие определенные аспекты работы психологического центра, что облегчает поиск и анализ необходимой информации.

Еще одним преимуществом является ее модульность. База данных разделена на логические части, каждая из которых отвечает за определенный аспект работы. Это позволяет и администраторам системы легко масштабировать и поддерживать базу данных, а также вносить изменения в определенные ее части.

Также стоит отметить гибкость базы данных в отношении адаптации к изменяющимся потребностям центра.

Например, при необходимости добавления новых направлений для психологов или услуг, необходимо просто добавить соответствующие записи в соответствующие таблицы.

3 Разработка web-приложения

3.1 Разработка клиентской части web-приложения

В разработке клиентской части приложения использовался фреймворк React JS [2]. С его помощью создание интерактивных пользовательских интерфейсов стало простым и приятным.

Декларативный подход React позволяет описать, как компоненты приложения выглядят в разных состояниях, и автоматически обновлять интерфейс при изменении данных. Это делает код более предсказуемым и упрощает отладку.

Еще одно преимущество React заключается в том, что логика компонентов написана на JavaScript, что позволяет передавать данные по всему приложению с легкостью и хранить состояние вне DOM. Это обеспечивает более эффективное управление состоянием приложения и повышает его масштабируемость.

Использование React JS не только облегчило разработку интерфейса, но и позволило создать более интуитивно понятный и удобный пользовательский опыт для пользователей.

При входе в приложение неавторизованный пользователь получает доступ к ряду ключевых страниц.

В частности, он может зарегистрироваться или авторизоваться, ознакомиться с главной страницей, на которой представлена общая информация о психологическом центре, прочитать отзывы других пользователей, изучить профили всех специалистов, а также ознакомиться с перечнем доступных услуг.

После входа в приложение пользователь, авторизованный в системе, получает расширенные функции.

В частности, он может взаимодействовать со всеми страницами в приложении, доступными неавторизованным пользователям, за исключением страниц регистрации и авторизации.

Кроме того, авторизованный пользователь имеет возможность редактировать свой профиль, включая изменение личной информации и фотографии. Он также может управлять своим расписанием сеансов, просматривать текущие, а также отменять их при необходимости.

Кроме того, авторизованный пользователь имеет возможность оставлять отзывы о работе психологического центра, делиться своим опытом и впечатлениями с другими пользователями, что помогает улучшить общий опыт пользования приложением.

Для администратора предоставляется расширенный доступ ко всем страницам, доступным обычным пользователям. Кроме того, администратору доступны специальные страницы управления приложением.

Это позволяет ему эффективно управлять профилями психологов, включая добавление, редактирование и удаление их профилей.

Также администратор может управлять услугами, внося изменения в их описания и параметры, и контролировать отзывы пользователей, имея возможность просматривать и удалять их по необходимости.

Для обеспечения маршрутизации между страницами используется компонент

Navigation, представлен на листинге 3.1.

```
const Navigation = () =>{
  const {isUserLoggedIn} = useContext(AuthContext);
  return (
    <BrowserRouter>
      <Routes>
        {isUserLoggedIn ? (
          <>
            <Route path="profile" element={<ProfilePage/>}/>
            <Route path="session" element={<SessionPage/>}/>
          </>): (
            <>
              <Route path="login" element={<LoginPage/>}/>
              <Route path="register" element={<RegPage/>}/>
            </>)}
        <Route path="home" element={<HomePage/>}/>
        <Route path="psychologist"
element={<PsychologistPage/>}/>
        <Route path="procedure" element={<ProcedurePage/>}/>
        <Route path="review" element={<ReviewPage/>}/>
        <Route path="*" element={<Navigate to={isUserLoggedIn
? "home" : "login"} replace/>}/>
      </Routes>
    </BrowserRouter>
  )
}
```

Листинг 3.1 – Содержимое файла Navigation.js

В зависимости от статуса аутентификации пользователя, компонент отображает определенные маршруты. Если пользователь авторизован, отображаются маршруты для страниц профиля и сеансов. В противном случае, если пользователь не авторизован, отображаются маршруты для страницы входа и регистрации. Всегда доступны маршруты для главной страницы, страниц психологов, процедур и отзывов. Если пользователь пытается получить доступ к несуществующему маршруту, его перенаправляют на главную страницу или страницу входа в зависимости от его статуса аутентификации.

В приложении применяется механизм аутентификации на основе JSON Web Token (JWT). Этот метод обеспечивает безопасную проверку подлинности пользователей и управление доступом к ресурсам приложения. Он включает два вида токенов: access и refresh.

Access токен используется для подтверждения аутентификации пользователя при каждом запросе к серверу. Он содержит информацию, идентифицирующую пользователя и определяющую его права доступа к различным функциям и данным в приложении. Access токен хранится локально на клиентской стороне в хранилище приложения.

Refresh токен, в свою очередь, служит для обновления access токена без необходимости повторной аутентификации пользователя. Он сохраняется в cookie браузера и используется для продления срока действия access токена. Когда access

токен истекает или становится недействительным, refresh токен передается на сервер для генерации нового access токена. Такой процесс позволяет пользователю продолжать использовать приложение без необходимости повторного ввода учетных данных.

Фрагмент кода, предоставляющий сервис управления JWT токенами в приложении, представлен на листинге 3.2.

```
const inMemoryJWTService = () =>{
  let inMemoryJWT = null;
  let refreshTokenId = null;
  const refreshToken = (expiration) =>{
    const timeoutTrigger = expiration - 10000;
    refreshTokenId = setTimeout(() =>{
      AuthClient.post("/refresh").then((res)=>{
        const {accessToken, accessTokenExpiration} =
res.data;
        setToken(accessToken, accessTokenExpiration);
      }).catch((error)=>console.error(error))
    }, timeoutTrigger);}
  const abortRefreshToken = () =>{
    if(refreshTokenId){
      clearInterval(refreshTokenId);
    }
  }
  const getToken = () => inMemoryJWT;
  const setToken = (token, tokenExpiration) =>{
    inMemoryJWT = token;
    refreshToken(tokenExpiration)};
  const deleteToken = () => {
    inMemoryJWT = null;
    abortRefreshToken();
  }
  return { getToken, setToken, deleteToken };
};
```

Листинг 3.2 – Содержимое файла inMemoryJWTService.js

Для осуществления запросов в приложении используется библиотека Axios [3]. Библиотека Axios отличается своей простотой использования благодаря интуитивно понятному API. Она предоставляет чистый и лаконичный синтаксис для отправки HTTP запросов, что делает код более читаемым и легким для поддержки.

Одним из основных преимуществ Axios является его поддержка управления асинхронными операциями. Благодаря использованию промисов, Axios позволяет делать HTTP запросы асинхронно и эффективно управлять временными задержками при получении ответов от сервера. Это особенно важно в современных веб-приложениях, где асинхронные запросы широко используются для получения данных с сервера без блокировки пользовательского интерфейса.

Кроме того, Axios обеспечивает кросс-браузерную совместимость и поддержку множества платформ, что делает его идеальным выбором для разработки веб-приложений, работающих на различных устройствах и в различных браузерах.

Пример использования библиотеки представлен на листинге 3.3.

```
const Client = axios.create({
  baseURL: `${config.API_URL}/client`,
});

export const handleProfileFetchProtected = (data) => {
  return Client.post("/update", {Name_Client: data.profileName,
Surname_Client: data.profileSurname, Photo_Client:
data.profilePhoto})
}

export const handleBookingFetchProtected = (data) => {
  return Client.post("/booking", data)
}
```

Листинг 3.3 – Пример использования библиотеки Axios

В процессе отправки защищенного запроса к серверу, используя механизм интерсепторов, который автоматически добавляет токен аутентификации в заголовок запроса перед его отправкой, представлено на листинге 3.4.

```
Client.interceptors.request.use((config) => {
  const accessToken = inMemoryJWTService.getToken();
  if (accessToken) {
    config.headers["Authorization"] = `Bearer ${accessToken}`;
  }
  return config;
}, (error) => {
  return Promise.reject(error);
});
```

Листинг 3.4 – Пример использования интерсептора

В приложении используется асинхронный Redux с помощью `createAsyncThunk` из Redux Toolkit.

Этот подход предоставляет дополнительные возможности для управления жизненным циклом асинхронных операций в Redux.

`createAsyncThunk` позволяет определить асинхронное действие всего в одной функции. Эта функция-обработчик выполняет асинхронную операцию, например, отправку запроса к серверу, и возвращает ее результат.

Преимущества асинхронного Redux с `createAsyncThunk` включают упрощенный синтаксис, встроенную обработку состояний, таких как «pending», «fulfilled» и «rejected» для управления состоянием загрузки и обработки ошибок, улучшенную обработку ошибок с помощью метода `rejectWithValue`, а также гибкость и масштабируемость для управления асинхронными операциями в приложении. Этот подход делает код более читаемым, гибким и эффективным для разработки асинхронных функций в Redux.

Пример использования `createAsyncThunk` представлен на листинге 3.5.

```
export const getAllProcedures = createAsyncThunk(
  'procedures/getAllProcedures',
  async (_, {rejectWithValue}) => {

    try {
      const res = await getAllProceduresFetch();
      return res.data.allProcedures;
    }
    catch (error) {
      return rejectWithValue(error.message);
    }
  });
```

Листинг 3.5 – Пример использования функции `createAsyncThunk`

Из листинга видно, что используется асинхронное действие `getAllProcedures`, созданное с помощью `createAsyncThunk`, для получения списка процедур. При выполнении этого действия, запрос отправляется к серверу. По завершении запроса, список процедур сохраняется в состоянии приложения. Состояние управляется с помощью `listProceduresSlice`, который обновляет `status` в зависимости от состояния выполнения запроса и обрабатывает ошибки, если таковые возникают.

3.2 Разработка серверной части web-приложения

Серверная часть приложения построена с использованием Node.js и фреймворка Express [4]. В ее архитектуре применяется парадигма `route – controller – service`. Маршруты представляют собой определение конечных точек и их маршрутизацию. Они устанавливают URL-адреса и типы запросов для каждого обрабатываемого ресурса и перенаправляют запросы к контроллерам.

Пример реализации маршрутизатора представлен на листинге 3.6.

```
const router = Router();
router.post("/sign-in-send-code",
  AuthController.sendEmailCodeSignIn);
router.post("/sign-in", AuthValidator.signIn,
  AuthController.signIn);
router.post("/sign-up-send-code",
  AuthController.sendEmailCodeSignUp);
router.post("/sign-up", AuthValidator.signUp,
  AuthController.signUp);
router.post("/logout", AuthValidator.logout, AuthController.logout);
router.post("/refresh", AuthValidator.refresh,
  AuthController.refresh);
export default router;
```

Листинг 3.6 – Пример реализации маршрутизатора

В маршрутизаторе используется валидатор, который позволяет выполнить

проверку валидности входящего запроса перед передачей его контроллеру.

Пример реализации валидатора представлен на листинге 3.7.

```
class AuthValidator {
  static async signIn(req, res, next) {
    return validateRequest(req, res, next, signInSchema);
  }
  static async signUp(req, res, next) {
    return validateRequest(req, res, next, signUpSchema);
  }
  static async logOut(req, res, next) {
    return validateRequest(req, res, next, logoutSchema);
  }
  static async refresh(req, res, next) {
    return validateRequest(req, res, next);
  }
}
```

Листинг 3.7 – Пример реализации валидатора

В процессе валидации запросов используется библиотека Yup. Сначала определяется схема валидации, которая содержит правила для проверки данных.

Пример схемы представлен на листинге 3.8.

```
export const signUpSchema = Yup.object({
  body: Yup.object({
    email: Yup.string().required().matches(/^[a-zA-Z0-9._%+-]{1,20}@[a-zA-Z0-9.-]{1,10}\.[a-zA-Z]{2,3}$/),
    login: Yup.string().required().matches(/^[a-zA-Z0-9]{1,13}$/),
    password: Yup.string().required().matches(/^[a-zA-Z0-9]{4,13}$/)
  })
});
```

Листинг 3.8 – Пример схемы

После определения схемы валидации, она применяется к входящим данным, например, к телу запроса, где данные запроса проверяются на соответствие заданным правилам. Если данные соответствуют схеме валидации, управление передается следующему middleware или контроллеру для обработки запроса.

Функция, проверяющая валидность данных по схеме, представлена на листинге 3.9.

```
export default async (req, res, next, schema) => {
  try {
    if (schema) {
      await schema.validate(req);
    }
    return next();
  } catch ({ path, errors }) {
    return ErrorUtils.catchError(
      res, new Unprocessable(JSON.stringify({ path, errors })))
  }
};
```

Листинг 3.9 – Содержимое файла ValidateRequest.js

После прохождения через маршрутизаторы, запрос передается

соответствующему контроллеру. Основная задача контроллера состоит в том, чтобы принять полученный запрос и выполнить необходимые действия в соответствии с логикой приложения. Действия контроллера могут включать в себя обработку данных, взаимодействие с сервисами приложения, а также отправку ответа клиенту.

Одним из важных аспектов работы контроллеров является обработка ошибок и исключений. Контроллеры могут возвращать соответствующие статусы и сообщения об ошибках в случае возникновения проблем при выполнении запроса.

Фрагмент реализации кода контроллера представлен на листинге 3.10.

```
class AuthController {
  static async signIn(req, res) {
    const { login, password, Id_Auth, Code_Client } = req.body;
    const { fingerprint } = req;
    try {const { accessToken, refreshToken, accessTokenExpiration,
user } = await AuthService.signIn( 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);}}
  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);
      return res.status(200).json({ accessToken,
accessTokenExpiration, outputUser });} catch (err) {
      return ErrorsUtils.catchError(res, err);}}
```

Листинг 3.10 – Фрагмент реализации контроллера

Для более детального ознакомления с реализацией контроллера, можно перейти в приложение А. Контроллер использует сервисы для выполнения конкретных действий, например, таких как вход пользователя в систему, регистрация нового пользователя и выход пользователя из системы.

Использование сервисов позволяет разделить логику приложения на более мелкие и переиспользуемые компоненты, что делает код более структурированным, понятным и легко поддерживаемым. Они также обеспечивают более высокий уровень абстракции и инкапсуляции, что способствует повторному использованию кода и улучшению его качества.

Фрагмент реализации сервиса представлен на листинге 3.11.

```
class ClientService{
    static async
    updateClientInfo(Name_Client,Surname_Client,Photo_Client, user){
        user.Name_Client = Name_Client;
        user.Surname_Client = Surname_Client;
        const myCloud = await
        cloudinary.v2.uploader.upload(Photo_Client)
        user.Photo_Client = myCloud.secure_url;
        await
        StaticClass.unit.clientRepository.update(user.Id_client,user);
        const newUser = await
        StaticClass.unit.clientRepository.findById(user.Id_client);
        return newUser;
    }
}
```

Листинг 3.11 – Фрагмент реализации сервиса

Сервисы используют UnitOfWork для взаимодействия с репозиториями. UnitOfWork представляет собой паттерн проектирования, который позволяет создавать единый контекст работы с данными.

Вместо того чтобы сервисы напрямую обращались к репозиториям данных, они используют UnitOfWork для выполнения операций с данными. Данный подход создает централизованный контроль над всеми изменениями и транзакциями в приложении.

Это позволяет сделать код более модульным и управляемым, так как UnitOfWork управляет всеми операциями с данными в рамках единого контекста. Такой подход способствует более эффективному управлению транзакциями и обеспечивает целостность данных, а также делает код легко поддерживаемым и расширяемым.

Реализация UnitOfWork представлена на листинге 3.12.

```
class UnitOfWork {
    constructor() {
        this.typeORM = connectionSource;
        this.clientRepository = new
        ClientRepository(this.typeORM.getRepository('Client'));
        this.psychologistRepository = new
        PsychologistRepository(this.typeORM.getRepository('Psychologist'));
        this.proceduresRepository = new
        ProceduresRepository(this.typeORM.getRepository('Procedures'));
        this.bookingRepository = new
        BookingRepository(this.typeORM.getRepository('Booking'));
    }
}
```

Листинг 3.12 – Содержимое файла UnitOfWork.js

UnitOfWork содержит в себе репозитории, которые используют для

взаимодействия с данными typeORM. Репозиторий – это шаблон проектирования, который служит для разделения логики бизнес-уровня и доступа к данным в приложении. Основная цель репозитория состоит в том, чтобы создать абстракцию над сложностью работы с данными. Он позволяет работать с данными независимо от их конкретного источника и предоставляет удобный интерфейс для выполнения операций CRUD с данными, представлено на листинге 3.13.

```
class VoucherRepository {
  constructor(repoVoucher, repoBooking) {
    this.repoVoucher = repoVoucher;
    this.repoBooking = repoBooking;
  }
  async create(data) {
    return this.repoVoucher.save(data);
  }
  async delete(id) {
    return this.repoVoucher.delete({
      Id_Voucher: id,
    });
  }
}
```

Листинг 3.13 – Фрагмент реализации репозитория

Для более детального ознакомления с реализацией репозитория можно перейти в приложение Б.

Приложение взаимодействует с клиентами, используя библиотеку Socket.IO, чтобы оповещать их о предстоящих сеансах с психологом.

Реализация сокета и обработчиков событий представлена на листинге 3.14.

```
const io = new Server(server, {
  cors: {origin: process.env.CLIENT_URL, methods: ["GET", "POST"],
}
});
io.use(TokenService.socketCheckAccess)
io.on('connection', (socket) => {
  socket.on('get-user-data', () => {
    SocketService.addUser(socket.user.Id_client, socket.id);
  });
  socket.on('disconnect', () => {
    SocketService.deleteUser(socket.id);
  }
});
});
setTimerNotifyUsers(io);
```

Листинг 3.14 – Реализация сокета и обработчиков событий

Из листинга видно, что сервер использует middleware для проверки доступа через access token в методе socketCheckAccess.

Реализация метода `socketCheckAccess` представлена на листинге 3.15.

```
static async socketCheckAccess(socket, next) {
  const authHeader = socket.handshake.headers.authorization;
  const token = authHeader?.split(" ").?.[1];
  try {
    if (!token) {
      throw new Error("Пользователь неавторизирован");
    }
    socket.user = await TokenService.verifyAccessToken(token);
    next();
  } catch (error) {
    console.error("Ошибка при проверке доступа:", error.message);
    socket.emit('access-error', { message: error.message });
    socket.disconnect(true);
  }
}
```

Листинг 3.15 – Реализация метода `socketCheckAccess`

Метод `setTimerNotifyUsers` используется для создания таймера, который периодически вызывает функцию `checkAndNotify` из класса `SocketService`.

Реализация метода `checkAndNotify` представлена на листинге 3.16.

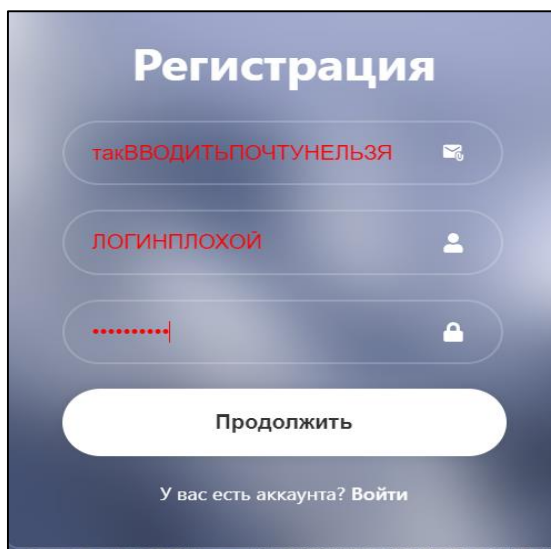
```
static async checkAndNotify(io){
  const currentDate = new Date();
  const currentTime = currentDate.getTime();
  for(const user of StaticClass.userSockets){
    const bookingsUser = await
    StaticClass.unit.bookingRepository.findByClientId(user.userId)
    for(const booking of bookingsUser){
      const voucher = booking.Voucher;
      if(voucher){ const voucherStartDateTime = new
      Date(voucher.Date_Voucher);
      voucherStartDateTime.setHours(voucher.Time_Voucher_Start.getHours()
      - 1, voucher.Time_Voucher_Start.getMinutes(),
      voucher.Time_Voucher_Start.getSeconds());
      if ( voucherStartDateTime.getTime() < currentTime) {
        if(user.socketId && io.sockets.sockets.get(user.socketId)){
          io.to(user.socketId).emit('booking-alert', {
            ID_Booking: booking.ID_Booking,
            Name_Procedure: booking.Procedures.Name_Procedure,
            Time_Start: `${voucher.Time_Voucher_Start.getHours():00`
          });
        }
      }
    }
  }
}
```

Листинг 3.16 – Реализация метода `checkAndNotify`

Данный метод производит проверку разницы текущего времени с временем записи и за час до сеанса генерирует соответствующее уведомление.

4 Тестирование web-приложения

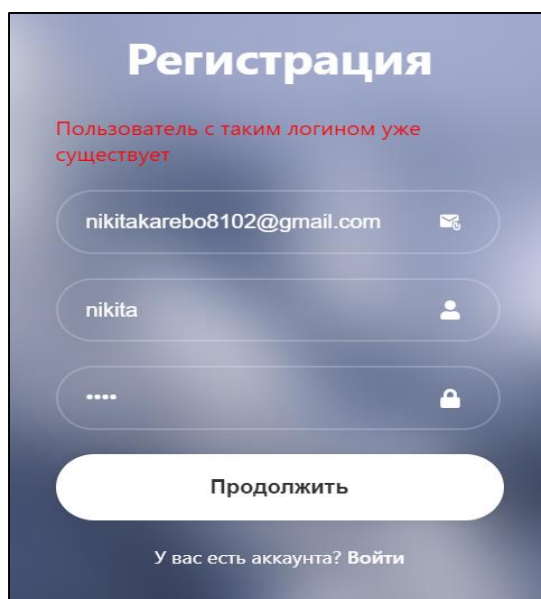
В ходе заполнения формы регистрации может возникнуть ситуация, когда пользователь вводит некорректные данные. В таких случаях текст поля становится красным, чтобы подчеркнуть ошибку и обратить внимание на необходимость коррекции, представлено на рисунке 4.1.



The screenshot shows a registration form titled "Регистрация" on a dark blue background. It contains three input fields: an email field with the text "такВВОДИТЬПОЧТУНЕЛЬЗЯ" in red, a username field with "ЛОГИНПЛОХОЙ" in red, and a password field with red dots. Each field has a corresponding icon (envelope, person, and lock respectively). Below the fields is a white "Продолжить" button and a link "У вас есть аккаунта? Войти".

Рисунок 4.1 – Неверно введенные данные

При попытке регистрации с уже зарегистрированным логином или адресом электронной почты приложение выведет соответствующее сообщение об ошибке, представлено на рисунке 4.2.



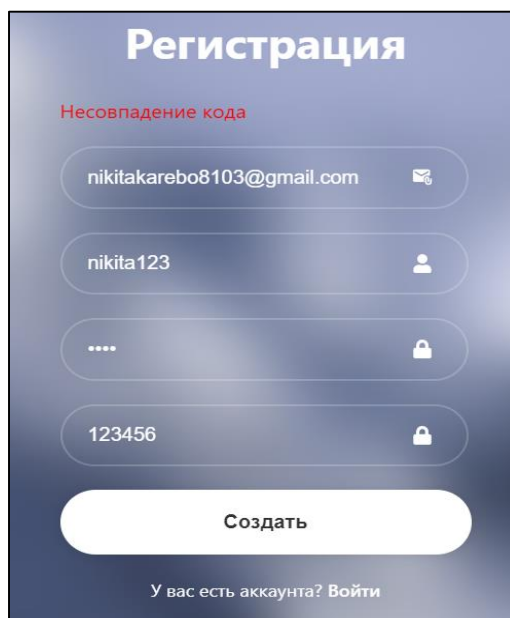
The screenshot shows the same registration form as in Figure 4.1, but with an error message at the top: "Пользователь с таким логином уже существует" in red. The email field now contains "nikitakarebo8102@gmail.com", the username field contains "nikita", and the password field contains four red dots. The "Продолжить" button and the "У вас есть аккаунта? Войти" link are still present at the bottom.

Рисунок 4.2 – Введенный логин зарегистрирован

При правильно заполненной форме регистрации на указанный адрес

электронной почты отправляется код подтверждения. Этот код необходимо будет ввести в соответствующее поле формы для завершения регистрации.

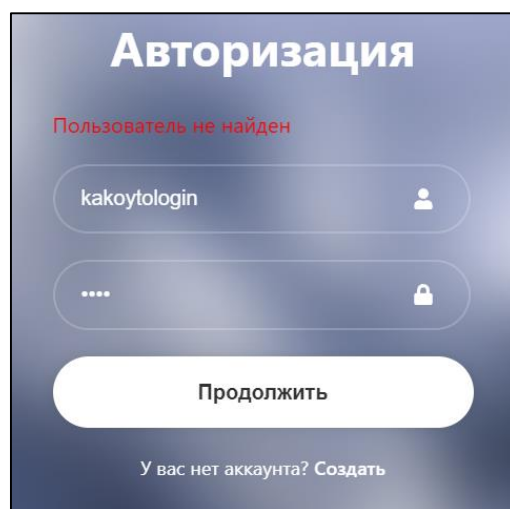
Если пользователь вводит неправильный код подтверждения, приложение выведет сообщение об ошибке, указывая на необходимость повторного ввода верного кода для завершения процесса регистрации, представлено на рисунке 4.3.



The screenshot shows a registration form titled "Регистрация" (Registration) on a dark blue background. At the top, there is a red error message: "Несовпадение кода" (Code mismatch). Below this, there are four input fields: the first contains the email "nikitakarebo8103@gmail.com" with an envelope icon; the second contains the username "nikita123" with a person icon; the third contains four dots "...." with a lock icon; and the fourth contains the code "123456" with a lock icon. At the bottom of the form is a large white button labeled "Создать" (Create). Below the button, there is a link: "У вас есть аккаунта? Войти" (Do you have an account? Log in).

Рисунок 4.3 – Неверный код подтверждения

Далее рассмотрим возможные ошибки в форме авторизации. При вводе некорректных данных в форму, аналогично процессу регистрации, цвет текста изменится на красный. Когда пользователь вводит свои учетные данные и нажимает кнопку «Продолжить», проводится проверка введенных данных на соответствие с данными из базы данных. Если соответствующая запись не найдена, выводится соответствующая ошибка, представлено на рисунке 4.4.



The screenshot shows an authorization form titled "Авторизация" (Authorization) on a dark blue background. At the top, there is a red error message: "Пользователь не найден" (User not found). Below this, there are two input fields: the first contains the username "kakoytologin" with a person icon; the second contains four dots "...." with a lock icon. At the bottom of the form is a large white button labeled "Продолжить" (Continue). Below the button, there is a link: "У вас нет аккаунта? Создать" (Do you not have an account? Create).

Рисунок 4.3 – Пользователь не найден

После успешного входа в приложение пользователь может изменить свой

профиль, внося необходимую информацию в свои личные данные. В процессе внесения изменений, введенная информация проходит непрерывную проверку. При обнаружении неправильного ввода, текстовые поля выделяются красным цветом, представлено на рисунке 4.4.

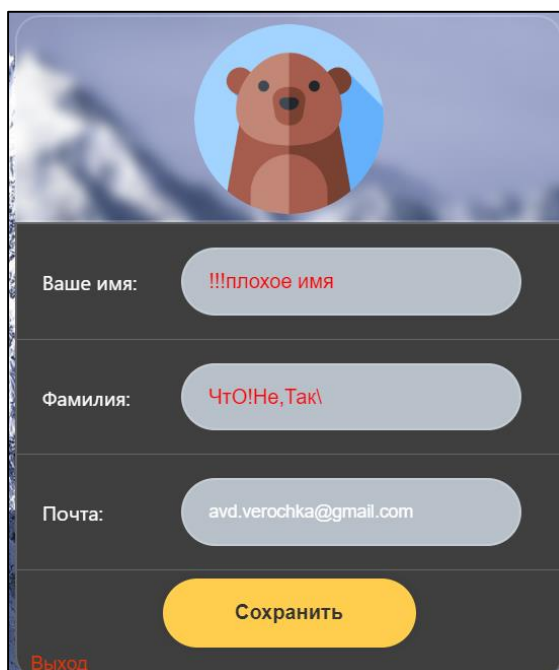


Рисунок 4.4 – Некорректные данные в профиле

Когда пользователь пытается ввести отзыв и введенный текст не соответствует требованиям, приложение изменит цвет текста на красный. Это действие является визуальным индикатором ошибки и предупреждает пользователя о том, что введенные данные не валидны, представлено на рисунке 4.5.

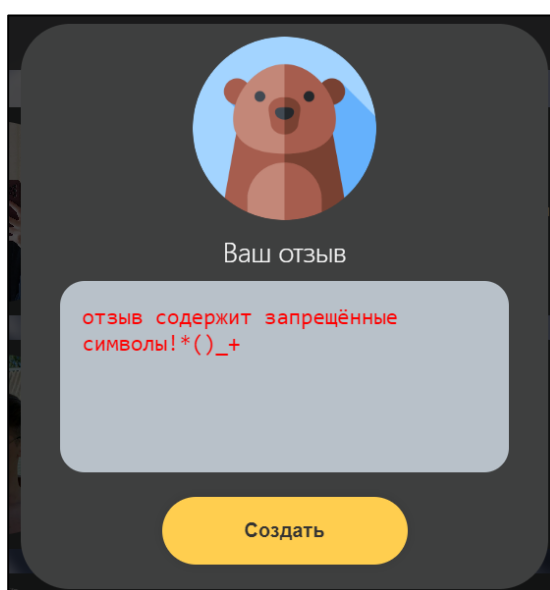


Рисунок 4.5 – Некорректный отзыв клиента

Подобные ситуации могут возникнуть и у администраторов,

поэтому рассмотрим несколько из них.

В случае, если администратор решит добавить психолога, приложение предоставит форму для заполнения. Если введенная информация не валидна, текст в полях изменит свое отображение на красный, изображено на рисунке 4.6.

The screenshot shows a registration form for a psychologist. At the top left is a back arrow and a profile picture placeholder. To the right of the profile picture is the label 'ФИО:' followed by a text input field containing 'непрвильноеФИО' in red. Below this, on the left, are three fields: 'Учёная степень' with a 'Выберите' button, 'Направление' with a 'Выберите' button, and 'Стаж работы (год/лет):' with a text input field containing '150' in red. To the right of these is a 'График работы' section with five rows for the days of the week (Понедельник through Пятница), each with two time input fields (8:00 and 13:00). Below the schedule is a 'Почта:' label followed by a text input field containing 'неправильнаяпочта' in red. At the bottom is a large text area labeled 'Описание' containing the text '! "№";%:*(((№)плохое описание|' in red. At the very bottom is a yellow 'Создать' button.

Рисунок 4.6 – Введенная информация в поля формы психолога не валидна

Когда администратор пытается создать нового психолога в приложении, возможна ситуация, когда введенные данные уже соответствуют существующему психологу в базе данных, поэтому получит соответствующую ошибку, которая изображена на рисунке 4.7.

This screenshot shows a portion of the registration form, specifically the 'Описание' (Description) field. The text input area contains the text 'плохое описание'. Below the input field, a red error message reads 'Психолог с таким ФИО уже существует'. At the bottom of the visible area is a yellow 'Создать' button.

Рисунок 4.7 – Психолог уже существует

Если администратор попытается создать психолога с почтой, которая уже задействована, получит сообщение с ошибкой.

Данная ситуация представлена на рисунке 4.8.

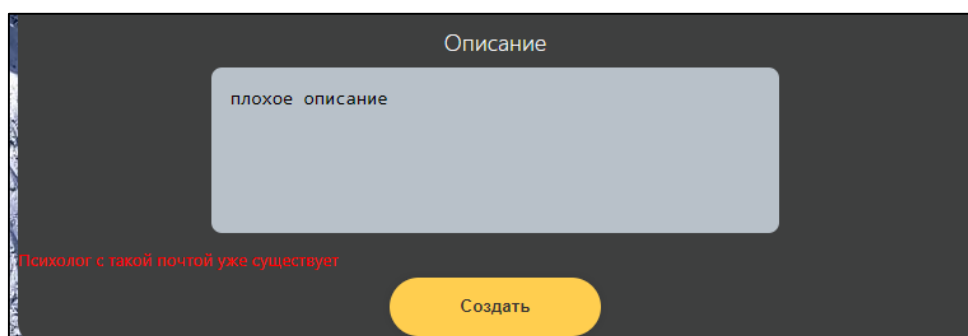


Рисунок 4.8 – Психолог с такой почтой уже существует

В случае, если администратор не выберет ученую степень или направление при заполнении формы, получит соответствующее сообщение об ошибке, представленное на рисунке 4.9.

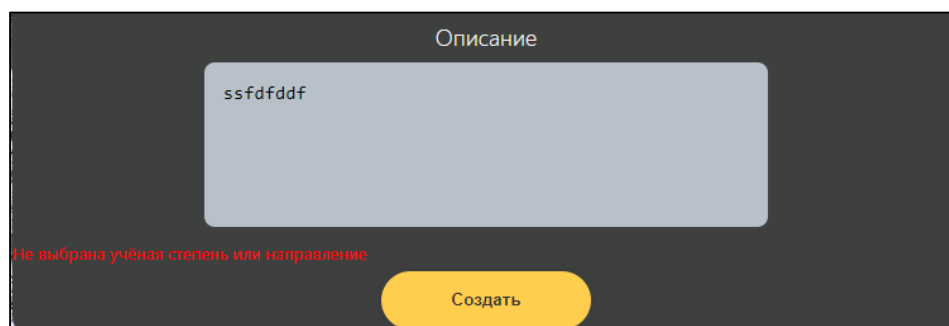


Рисунок 4.9 – Не выбрана ученая степень или направление

Схожая ситуация происходит при добавлении администратором новой услуги. При вводе некорректных данных в форму, цвет текста поля меняется на красный цвет, представлено на рисунке 4.10.

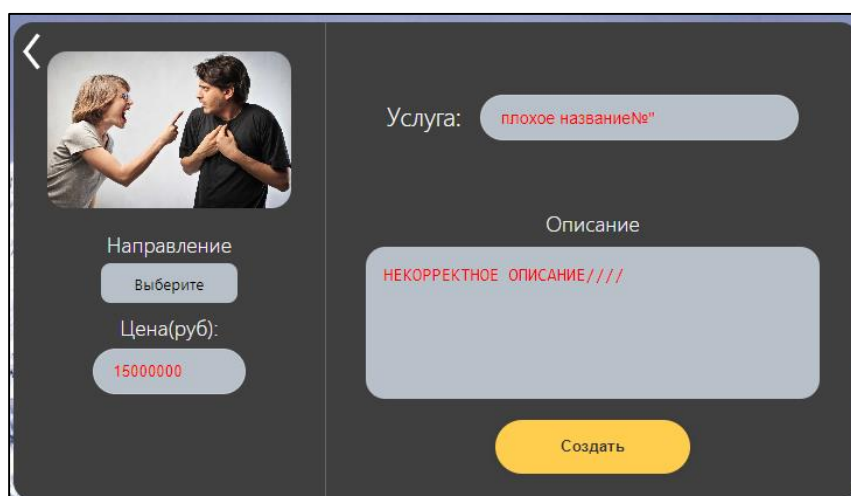


Рисунок 4.10 – Введенная информация в поля формы услуги не валидна

Если администратор попытается создать услугу с существующим названием,

приложение выведет ошибку, которая изображена на рисунке 4.11.

Рисунок 4.11 – Процедура с таким названием существует

В случае, если администратор не выберет направление при заполнении формы услуги, получит соответствующее сообщение об ошибке.

Данная ситуация представлена на рисунке 4.12.

Рисунок 4.12 – Направление услуги не выбрано

Проверка всех возможных ошибок в приложении необходима для обеспечения безопасности данных, повышения пользовательского опыта и качества приложения.

5 Руководство пользователя

5.1 Руководство гостя

При открытии веб-приложения, гость попадает на главную страницу, на которой может ознакомиться с общей информацией о психологическом центре, такой как его миссия, ценности. На странице присутствует навигационное меню, которое позволяет ему удобно перемещаться по различным страницам.

Главная страница представлена на рисунке 5.1.

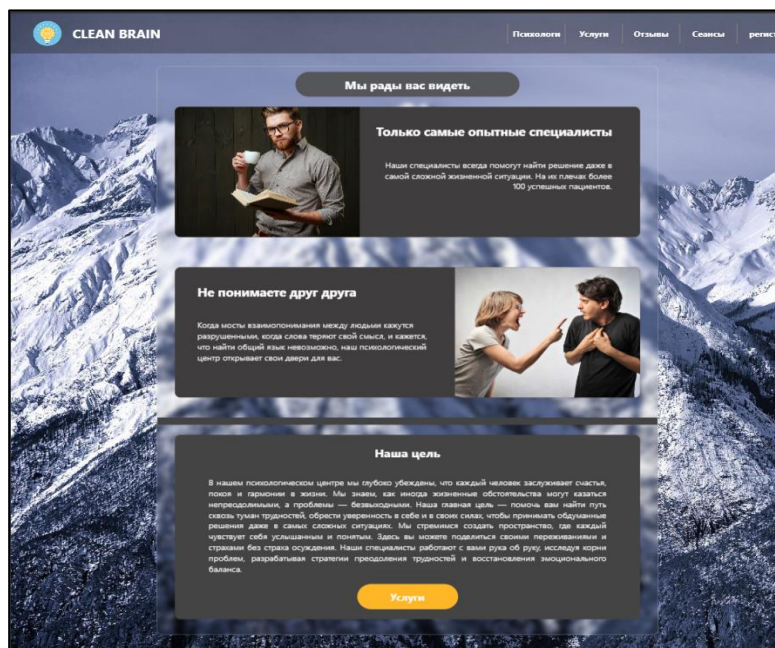


Рисунок 5.1 – Главная страница

По нажатию кнопки «Психологи» гость переходит на страницу с полным списком психологов. Здесь он может сортировать их по направлениям или использовать поиск для быстрого поиска конкретного специалиста.

Представлено на картинке 5.2.

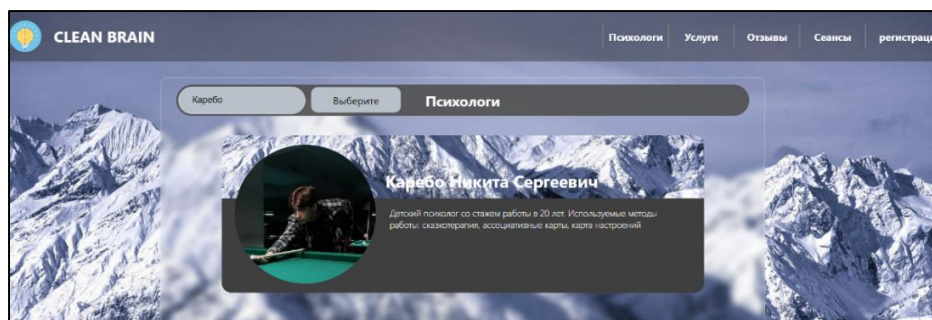


Рисунок 5.2 – Страница со списком психологов

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

Профиль психолога представлен на рисунке 5.3.

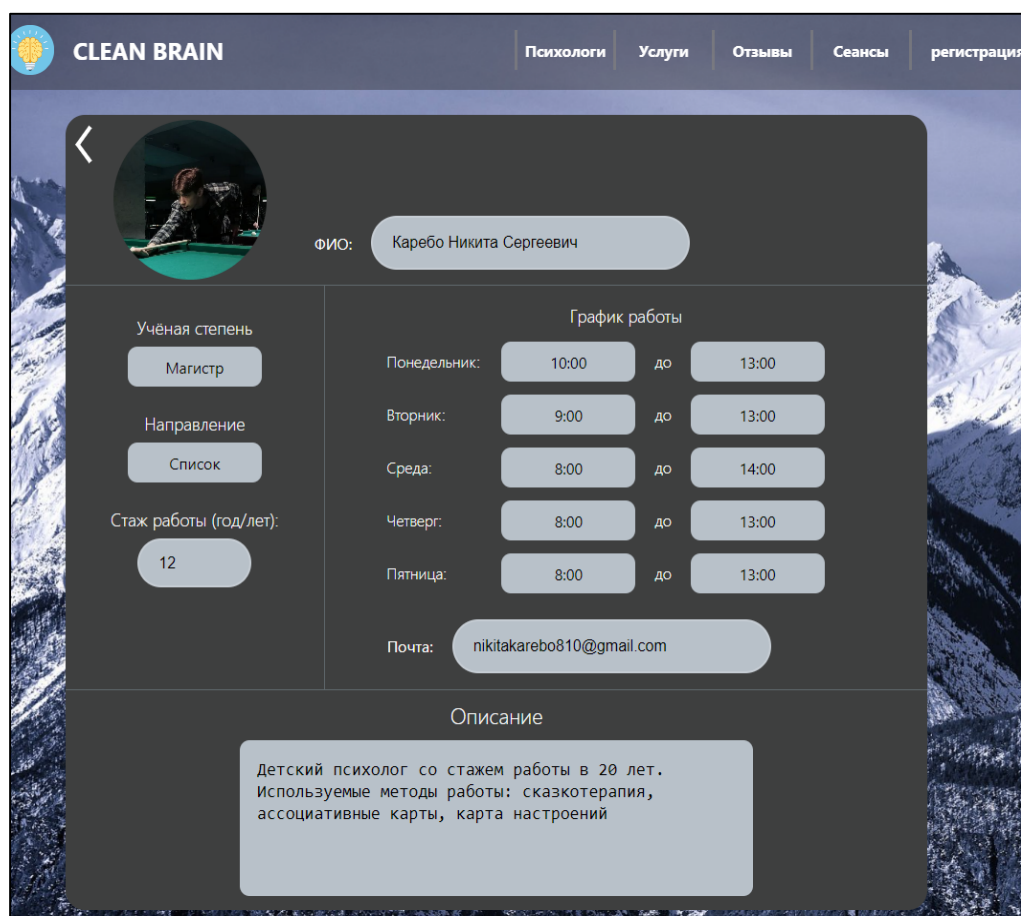


Рисунок 5.3 – Профиль психолога

По нажатию кнопки «Услуги» гость попадает на страницу со всеми предоставляемыми услугами. Здесь он может легко сортировать услуги по направлениям или цене, а также воспользоваться поиском для быстрого поиска нужной услуги. Представлено на рисунке 5.4.

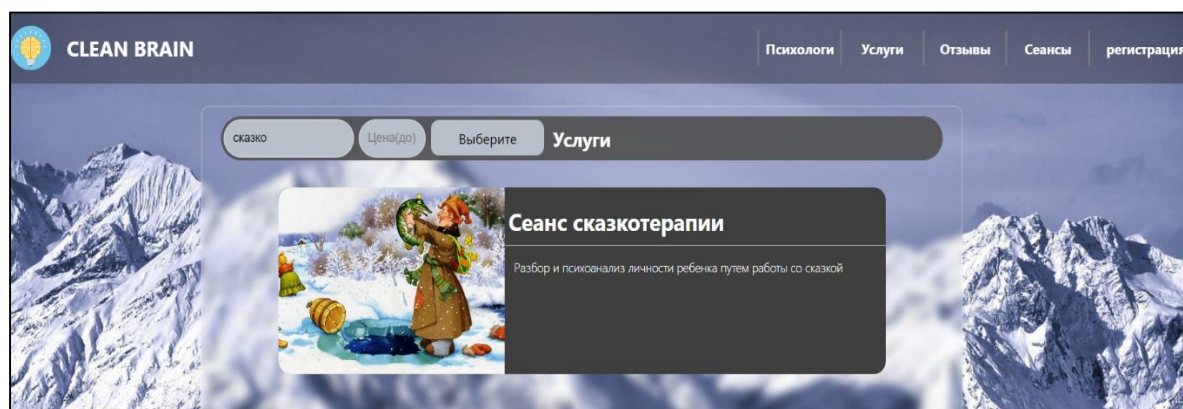


Рисунок 5.4 – Страница со списком услуг

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

Представлено на рисунке 5.5.

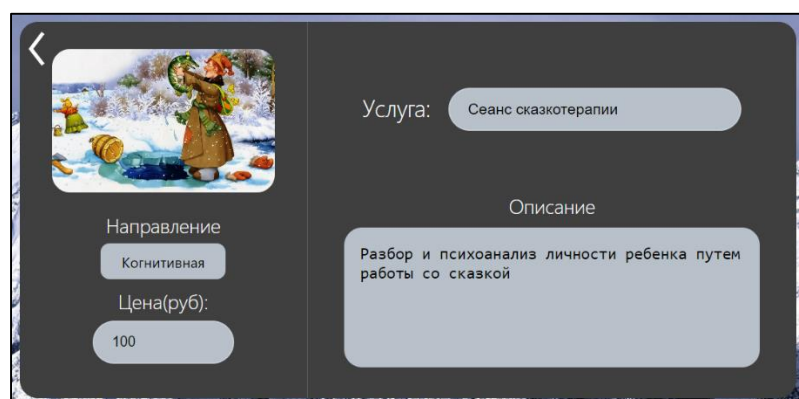


Рисунок 5.5 – Страница услуги

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.

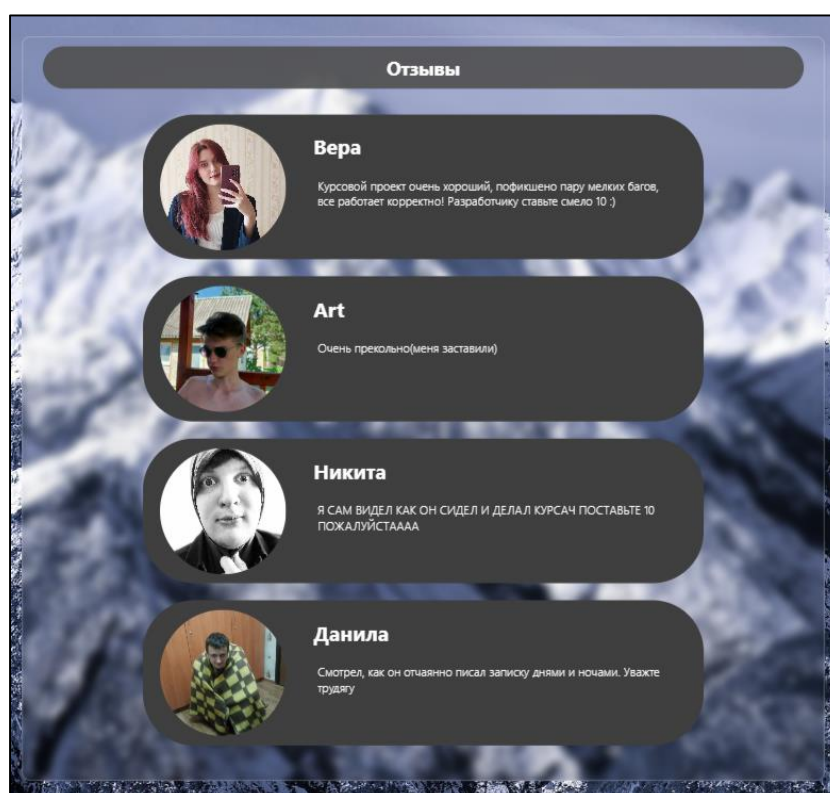


Рисунок 5.6 – Страница с отзывами

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

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

Страница регистрации представлена на рисунке 5.7.

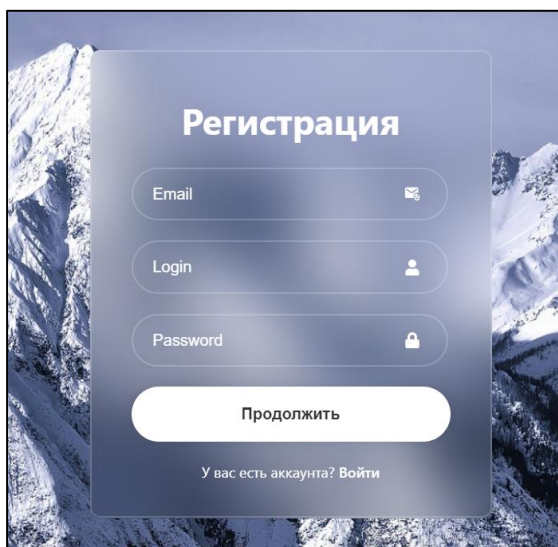
The image shows a registration form titled "Регистрация" (Registration) overlaid on a background of a snowy mountain. The form contains three input fields: "Email" with an envelope icon, "Login" with a person icon, and "Password" with a lock icon. Below these fields is a large white button labeled "Продолжить" (Continue). At the bottom of the form, there is a link that says "У вас есть аккаунта? Войти" (Do you have an account? Log in).

Рисунок 5.7 – Страница регистрации

Здесь он сможет создать учетную запись, указав свою почту, на которую придет код для подтверждения почты, логин и пароль, чтобы получить доступ ко всем возможностям и услугам психологического центра.

5.2 Руководство пользователя

Если у пользователя уже есть аккаунт, то ему необходимо перейти на страницу авторизации, где он сможет войти, используя свой логин и пароль, чтобы получить доступ к личному кабинету и другим функциям приложения, представлено на рисунке 5.8.

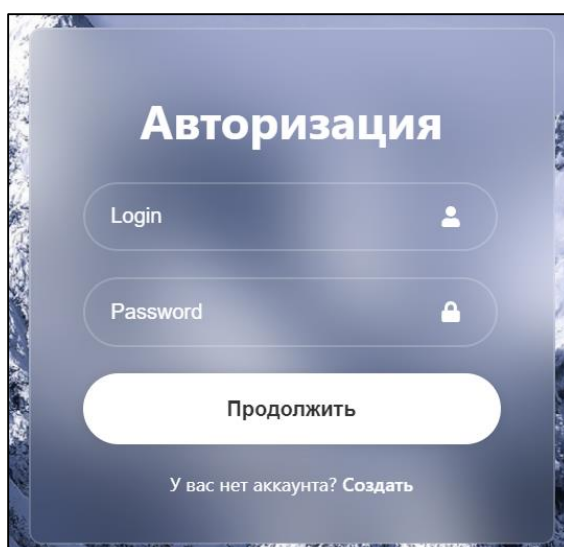
The image shows an authorization form titled "Авторизация" (Authorization) overlaid on a background of a snowy mountain. The form contains two input fields: "Login" with a person icon and "Password" with a lock icon. Below these fields is a large white button labeled "Продолжить" (Continue). At the bottom of the form, there is a link that says "У вас нет аккаунта? Создать" (Do you not have an account? Create).

Рисунок 5.8 – Страница авторизации

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

пользователь должен ввести его в соответствующее поле на форме авторизации, чтобы завершить процесс авторизации, представлено на рисунке 5.9.

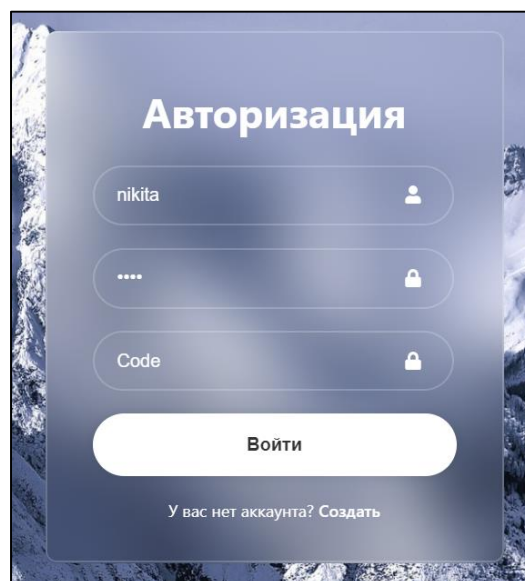


Рисунок 5.9 – Страница авторизации с полем для кода

После успешной авторизации, пользователь имеет доступ к тем же страницам, что и гость, включая страницы с информацией о психологах, услугах и отзывах. Однако, у 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 5.10.

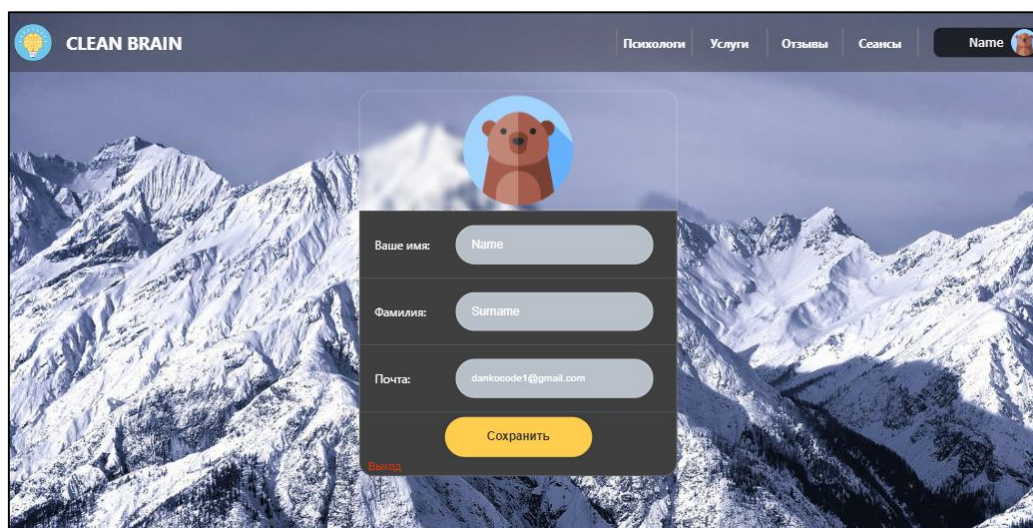
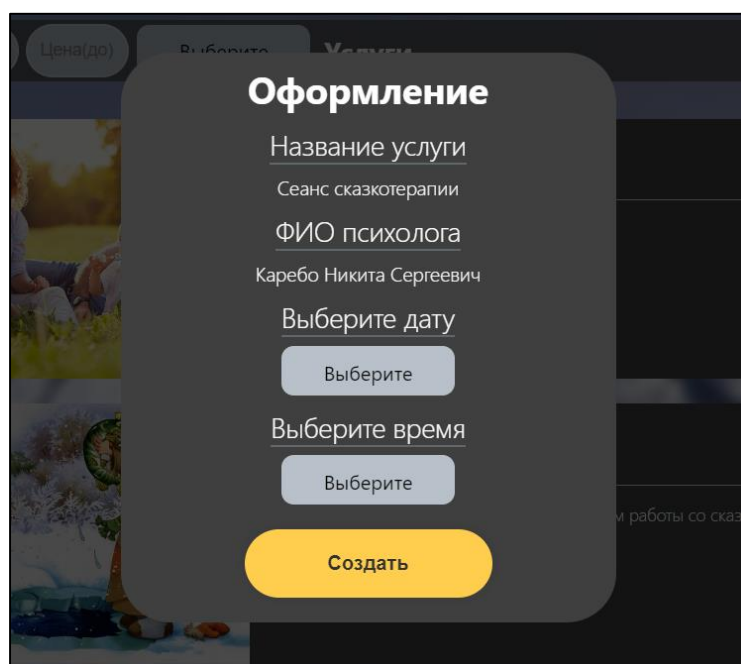


Рисунок 5.10 – Профиль пользователя

Как и у гостя, пользователь может просматривать список психологов, и при выборе конкретного психолога отображается список всех услуг, которые этот психолог предоставляет. При выборе услуги клиенту отображается форма с оформлением услуги, где он может выбрать дату и время приема.

Форма оформления представлена на рисунке 5.11.



The screenshot shows a dark-themed modal window titled 'Оформление' (Booking). It contains the following fields and elements:

- Название услуги** (Service Name): A text input field with the value 'Сеанс сказкотерапии' (Storytelling session).
- ФИО психолога** (Psychologist's Full Name): A text input field with the value 'Каребо Никита Сергеевич'.
- Выберите дату** (Select Date): A date picker button with the text 'Выберите' (Select).
- Выберите время** (Select Time): A time picker button with the text 'Выберите' (Select).
- Создать** (Create): A large yellow button at the bottom.

Рисунок 5.11 – Оформление выбранной услуги

После успешного оформления услуги пользователь может просмотреть свои текущие сеансы, нажав на кнопку «Сеансы».

Страница с сеансами пользователя представлена на рисунке 5.12.

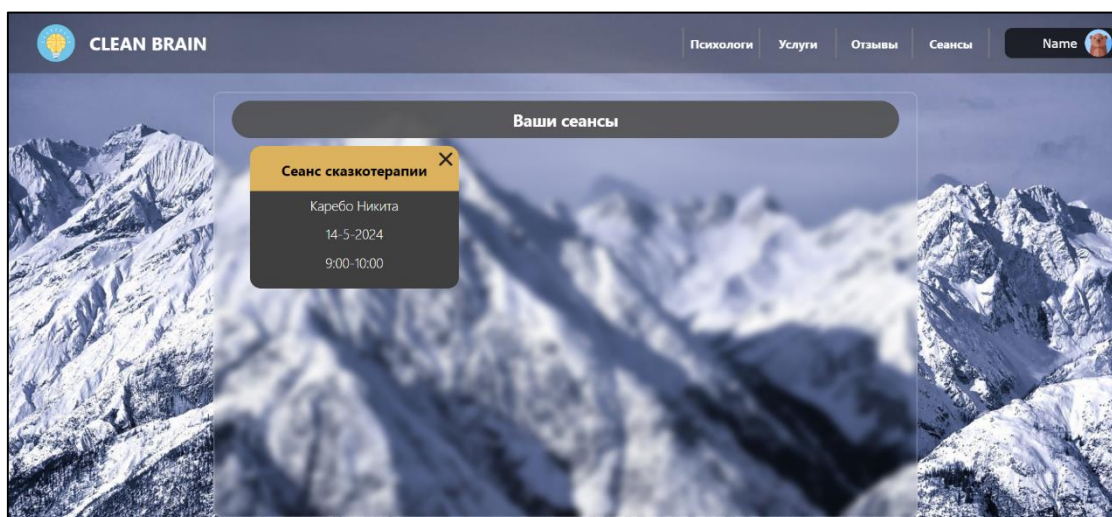


Рисунок 5.12 – Страница с текущими сеансами

Если пользователь хочет отказаться от сеанса, ему просто нужно нажать на крестик рядом с соответствующим сеансом.

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 отзыва представлена на рисунке 5.13.

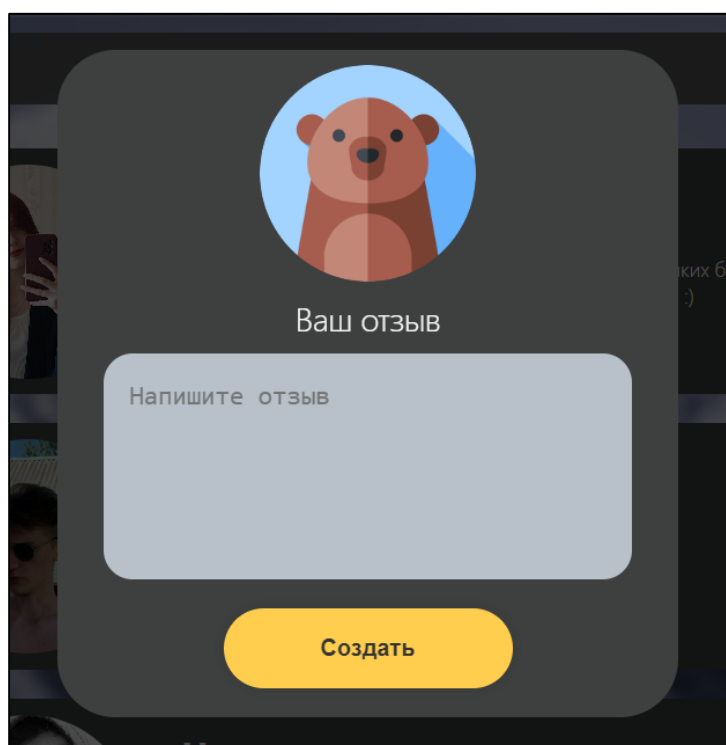


Рисунок 5.13 – Форма отзыва пользователя

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

5.3 Руководство администратора

Когда администратор зайдет в приложение и откроет страницу со всеми психологами, ему будет доступна кнопка «Создать», представлено на рисунке 5.14.

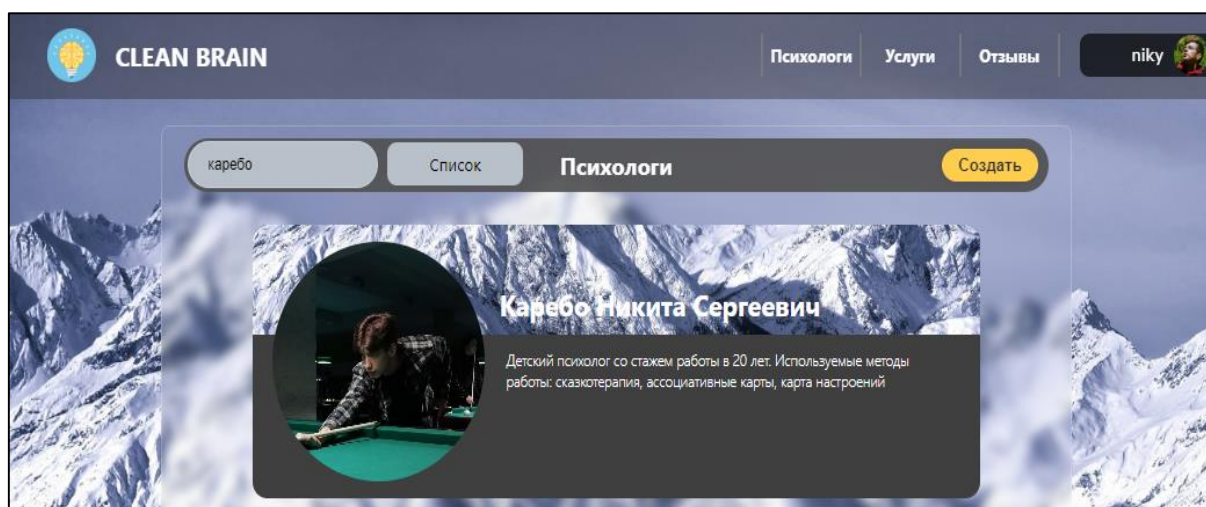


Рисунок 5.14 – Страница со списком психологов администратора

При нажатии на кнопку «Создать» откроется форма создания психолога.

Форма для создания психолога представлена на рисунке 5.15.

The screenshot shows a web application interface for 'CLEAN BRAIN'. At the top, there are navigation links: 'Психологи' (Psychologists), 'Услуги' (Services), 'Отзывы' (Reviews), and a user profile 'niky'. The main content area is a form for creating a psychologist profile. It features a circular profile picture placeholder, a text input for 'ФИО:' (FIO), and several dropdown menus for 'Ученая степень' (Academic degree), 'Направление' (Specialization), and 'Стаж работы (год/лет):' (Work experience). A 'График работы' (Work schedule) section contains a table for selecting hours for each day of the week. Below this is a text input for 'Почта:' (Email) and a larger text area for 'Описание' (Description). A yellow 'Создать' (Create) button is positioned at the bottom of the form.

Рисунок 5.15 – Форма создания психолога

При нажатии администратором на изображение психолога из списка психологов открывается его профиль. В этом профиле администратор может вносить изменения в данные психолога, а также удалять его из системы или создавать талончики для психолога.

Страница психолога для администратора представлена на рисунке 5.16.

The screenshot displays the profile of a psychologist named 'Карбо Никита Сергеевич'. The layout is similar to the creation form but with pre-filled data. The 'Ученая степень' is 'Магистр', 'Направление' is 'Список', and 'Стаж работы' is '12'. The 'График работы' table shows specific hours for each day. The 'Почта' field contains 'nikitakarebo810@gmail.com'. The 'Описание' field contains a detailed text about the psychologist's experience and methods. At the bottom of the profile card, there are three action buttons: a yellow 'талончик' (Ticket) button, a yellow 'Изменить' (Edit) button, and a red 'Удалить' (Delete) button.

Рисунок 5.16 – Профиль психолога для администратора

Если администратор зайдет на страницу со всеми услугами, ему будет доступна кнопка «Создать».

Страница со списком всех услуг, представлена на рисунке 5.17.

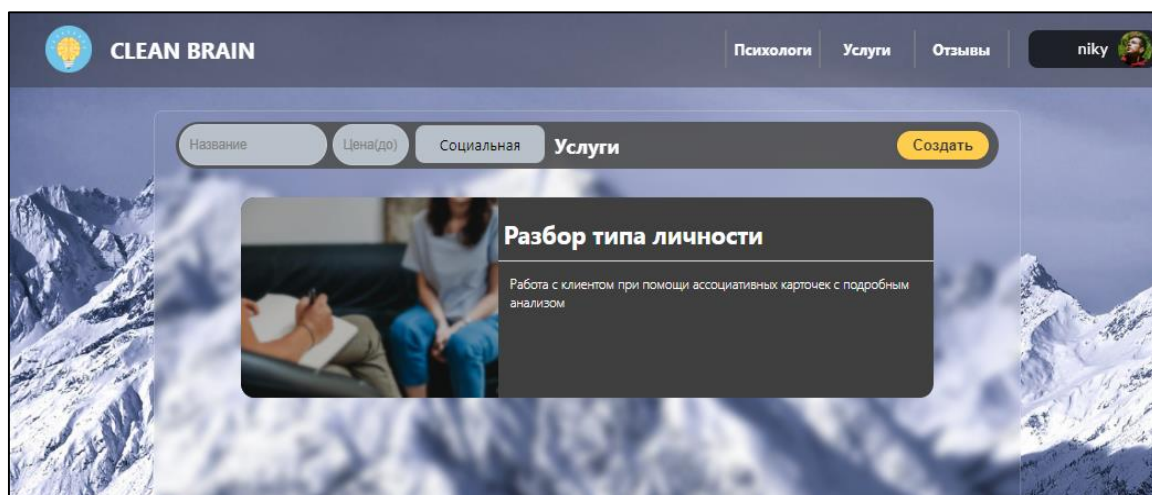


Рисунок 5.17 – Страница со списком услуг для администратора

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.

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

Форма для создания услуги представлена на рисунке 5.18.

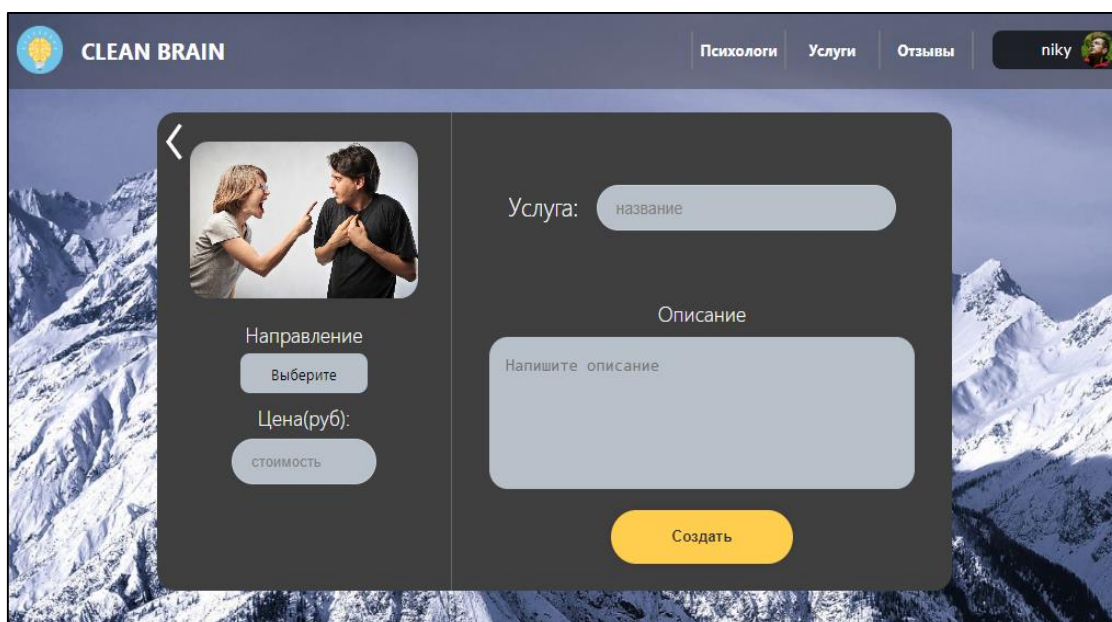


Рисунок 5.18 – Форма создания услуги

Для изменение текущей услуги, администратор должен нажать на изображение услуги из перечня всех услуг. Администратору вновь открывается форма с заполненными полями доступных для изменения. Так же на форме доступна кнопка «Удалить», которая удалит услугу из базы данных.

Страница услуги для администратора представлена на рисунке 5.19.

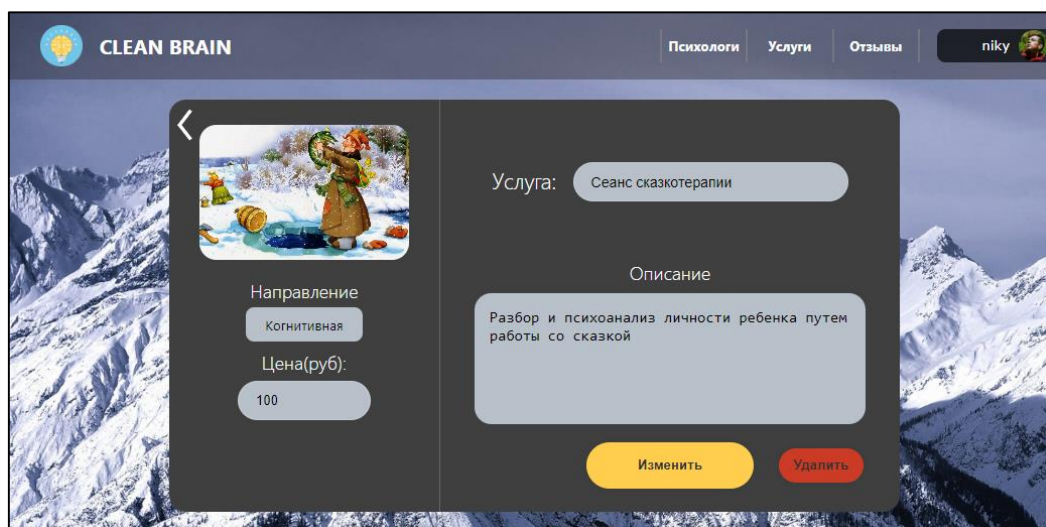


Рисунок 5.19 – Страница услуги для администратора

При выборе администратором пункта «Отзывы» в навигационном меню, открывается страница с отзывами от клиентов.

Страница с отзывами для администратора представлена на рисунке 5.20.

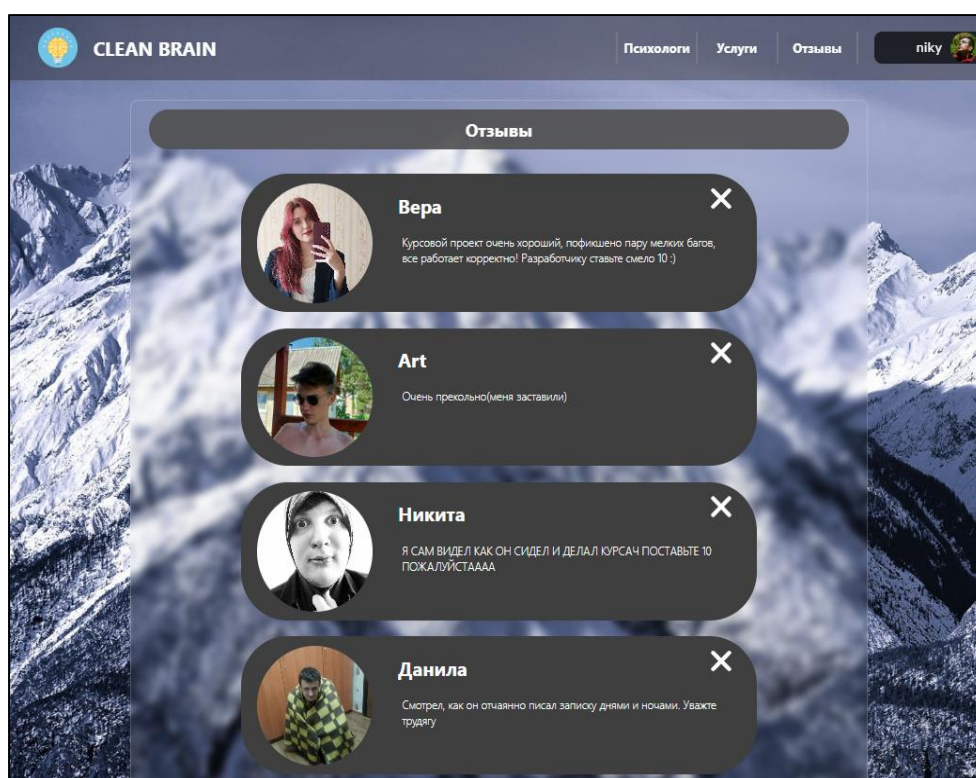


Рисунок 5.20 – Страница с отзывами для администратора

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 Установка приложения

Первый важный этап включает запуск серверной части приложения при помощи команды «`npm start server`».

Этот шаг не ограничивается просто активацией сервера, но также включает в себя необходимую подготовку к работе, установку необходимых зависимостей и настройку окружения.

Этот этап особенно критичен, поскольку сервер является неотъемлемым центральным элементом приложения, ответственным за обработку запросов, взаимодействие с базой данных и обеспечение безопасности.

После успешного запуска сервера следует второй важный шаг, который заключается в запуске клиентской части приложения с помощью команды «`npm start clean-brain`». Этот этап играет ключевую роль в предоставлении пользовательского интерфейса и обеспечении взаимодействия с сервером.

Клиентская часть приложения не только обрабатывает запросы пользователей, но и отвечает за формирование и предоставление удобного, интуитивно понятного интерфейса.

Оба эти важные этапы совместно обеспечивают полноценное функционирование приложения.

Запуск сервера гарантирует его готовность к обработке запросов, а запуск клиентской части приложения предоставляет пользователям возможность начать взаимодействие с приложением через интуитивно понятный и удобный пользовательский интерфейс.

После успешного выполнения всех указанных шагов приложение будет готово к использованию.

Пользователи смогут получить доступ к сайту, используя адрес `https://localhost:3000`, и начать активное взаимодействие с ним через его интуитивно понятный интерфейс.

Заключение

Приложение «Clean Brain», разработанное для психологического центра, представляет собой интегрированную платформу для эффективной реализации услуг по психологическому консультированию и поддержке. С его помощью клиенты могут удобно просматривать услуги, записываться на консультации к специалистам, просматривать отзывы пользователей. Приложение обеспечивает простой и интуитивно понятный пользовательский интерфейс, а также конфиденциальность данных клиентов.

Серверная часть приложения, разработанная на основе Node.js и фреймворка Express, предоставляет высокую производительность и безопасность при работе с данными. В ее структуру входят роутеры, контроллеры, сервисы и репозитории, которые позволяют эффективно управлять информацией о клиентах и их консультациях, обеспечивая надежное взаимодействие с базой данных.

С помощью библиотеки Socket.IO пользователи приложения «Clean Brain» получают мгновенные оповещения о предстоящих сеансах с психологом прямо в реальном времени. Этот функционал обеспечивает клиентам возможность подготовиться к встрече с психологом заблаговременно и эффективно.

Применение протокола HTTPS обеспечивает безопасность передачи данных между клиентами и сервером, что особенно важно при работе с конфиденциальной информацией.

Клиентская часть приложения разработана с использованием React, что обеспечивает гибкость и удобство в использовании для клиентов «Clean Brain». Модуль axios обеспечивает надежную передачу данных между клиентами и сервером и удобство в использовании.

Система управления базами данных MSSQL обеспечивает надежное хранение данных клиентов и истории их консультаций. Работа с базой данных осуществляется через typeORM, что упрощает взаимодействие с данными и обеспечивает безопасность информации.

После тщательного тестирования и анализа приложение было признано готовым к использованию в реальных условиях.

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

Список используемых источников

1 Документация по Microsoft SQL [Электронный ресурс] / Режим доступа: <https://learn.microsoft.com/ru-ru/sql/> – Дата доступа: 03.04.2024.

2 The React Handbook [Электронный ресурс] / Режим доступа: <https://medium.com/free-code-camp/the-react-handbook> – Дата доступа: 27.03.2024.

3 Axios Api Documentation [Электронный ресурс] / Режим доступа: https://axios-http.com/docs/api_intro – Дата доступа: 11.04.2024.

4 Веб-фреймворк Express [Электронный ресурс] / Режим доступа: <https://developer.mozilla.org/ru/docs/Learn/Server-side/> – Дата доступа: 10.04.2024.

Приложение А

```

class AuthController {
  static async sendEmailCodeSignUp(req, res) {
    const { Mail_Client, Login_Client } = req.body;
    try{
      const Id_Auth = await
EmailAuthenticationService.sendEmailCodeSignUp(Mail_Client, Login_Cli
ent);
      return res.status(200).json({ Id_Auth });
    }catch (err) {
      return ErrorsUtils.catchError(res, err);}}
  static async sendEmailCodeSignIn(req, res) {
    const { login, password } = req.body;
    try{
      const Id_Auth = await
EmailAuthenticationService.sendEmailCodeSignIn(login, password);
      return res.status(200).json({ Id_Auth });
    }catch (err) {
      return ErrorsUtils.catchError(res, err);}}
  static async logOut(req, res) {
    const refreshToken = req.cookies.refreshToken;
    try {
      await AuthService.logOut(refreshToken);
      res.clearCookie("refreshToken")
      return res.sendStatus(200);
    } catch (err) {
      return ErrorsUtils.catchError(res, err);
    }}
  static async refresh(req, res) {
    const { fingerprint } = req;
    const currentRefreshToken = req.cookies.refreshToken;
    try {
      const {accessToken, refreshToken, accessTokenExpiration, user}
=await AuthService.refresh({
        currentRefreshToken,
        fingerprint,});
      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 1 – Содержимое файла Auth.js

Приложение Б

```
class VoucherRepository {
  constructor(repoVoucher,repoBooking) {
    this.repoVoucher = repoVoucher;
    this.repoBooking = repoBooking;
  }
  async getAll() {
    return this.repoVoucher.find();
  }
  async deleteVoucher(id) {
    return this.repoVoucher.delete({
      Id_Voucher:id,
    });
  }
  async findById(id) {
    return this.repoVoucher.findOne({
      where: {
        Id_Voucher:id,
      },
    });
  }
  async findByPsychologistId(id) {
    return this.repoVoucher.find({
      where: {
        Id_Psychologist:id,
      },
    });
  }
  async create(data) {
    return this.repoVoucher.save(data);
  }
  async update(Id_Voucher, data) {
    return this.repoVoucher.update(Id_Voucher ,
      {
        Ordered: data.Ordered
      }
    );
  }
  async delete(id) {
    return this.repoVoucher.delete({
      Id_Voucher: id,
    });
  }
  async deleteAll() {
    return this.repoVoucher.clear();
  }
  async deleteNotOrdered(id) {
    return this.repoVoucher.delete({
      Ordered: "Нет",
    });
  }
}
```

```

        Id_Psychologist: id,
    });
}

async deleteOldVouchers(id) {
    const Vouchers = await this.repoVoucher.find({
        where: {
            Id_Psychologist: id,
            Date_Voucher: {
                lt: new Date(),
            },
        },
    });

    for (const Voucher of Vouchers) {
        if (Voucher.ordered === "Нет") {
            await this.repoVoucher.delete({
                Id_Voucher:
                    Voucher.Id_Voucher,
            });

        } else {
            await this.repoBooking.delete({
                Id_Voucher:
                    Voucher.Id_Voucher,
            });

            await this.repoVoucher.delete({

                Id_Voucher:
                    Voucher.Id_Voucher,
            });
        }
    }
}

async deleteAllById(id) {
    return this.repoVoucher.delete(
        {
            Id_Psychologist: id,
        });
}

export default VoucherRepository;

```

Листинг 2 – Содержимое файла VoucherRepository.js