МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 01 01 Программное обеспечение информационных технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОЙ РАБОТЕ НА ТЕМУ:**

«Реализация базы данных приложения психологического центра с применением технологии репликации данных между серверами СУБД»

Выполнил студент Каребо Н.С.

(Ф.И.О.)

Руководитель работы асс. Нистюк Ольга Александровна

(учен. степень, звание, должность, Ф.И.О., подпись)

И.о. зав. кафедрой ст. преп. Блинова Е.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовая работа защищена с оценкой

Минск 2023

**Содержание**

[Введение 3](#_Toc153971528)

[1 Постановка задачи 4](#_Toc153971529)

[2 Проектирование базы данных 5](#_Toc153971530)

[2.1 Обобщенная структура управления приложением 5](#_Toc153971531)

[2.2 Проектирование модели базы данных 6](#_Toc153971532)

[3 Разработка объектов базы данных 11](#_Toc153971533)

[3.1 Создание таблиц 11](#_Toc153971534)

[3.2 Создание ролей для разграничения доступа 11](#_Toc153971535)

[3.3 Создание пакетов и процедур для работы с базой данных 11](#_Toc153971536)

[4 Описание процедур экспорта и импорта 15](#_Toc153971537)

[4.1 Описание процедуры экспорта 15](#_Toc153971538)

[4.2 Описание процедуры импорт 16](#_Toc153971539)

[5 Тестирование производительности 17](#_Toc153971540)

[5.1 Оптимизирование запросов 17](#_Toc153971541)

[5.2 Проверка выполнения хранимых процедур 19](#_Toc153971542)

[6 Описание и применение технологии в БД 21](#_Toc153971543)

[6.1 Репликация данных 21](#_Toc153971544)

[6.2 Мультимедийные типы данных 22](#_Toc153971545)

[6.3 Технология шифрования 24](#_Toc153971546)

[7 Руководство пользователя 26](#_Toc153971547)

[7.1 Руководство по использованию администраторам 27](#_Toc153971548)

[7.2 Руководство по использованию пользователем 31](#_Toc153971549)

[7.3 Руководство по использованию гостем 36](#_Toc153971550)

[Заключение 38](#_Toc153971551)

[Cписок используемых источников 39](#_Toc153971552)

[Приложение А 40](#_Toc153971553)

[Приложение Б 45](#_Toc153971554)

[Приложение В 50](#_Toc153971555)

# **Введение**

В настоящее время ни для кого не секрет, что самой важным для любого человека является – эмоции. Чувства и эмоции тесно связаны с нашими внутренними качествами, они просто являются отражением того, что происходит внутри каждого человека. Людям всегда хочется рассказать о своих переживаниях или проблемах. По этой причине люди проводят огромное количество времени в поисках нужного психолога, который им подойдёт.

База данных является ключевым элементом успешного функционирования психологического центра, обеспечивая эффективное управление и хранение разнообразной информации. В проекте используется реляционная база данных, которая строится на модели данных, основанной на связях между таблицами. Для реализации этой структуры была выбрана СУБД Oracle, известная своей высокой производительностью, масштабируемостью и надежностью.

Так же было необходимо разработать приложение для демонстрации работы базы данных, взаимодействия с ней. Приложение было написано на языке программирования C# с графическим интерфейсом, выполненным с помощью WPF. Для взаимодействия с сервером базы данных Oracle использовался пакет Oracle Managed Data Access для .NET.

Любой желающий человек может самостоятельно создать или войти в учётную запись. Это дает возможность подробно ознакомиться со всеми психологами учреждения и процедурами, которые они предоставляют, а также забронировать сеанс на определенное число, выбрав удобную для себя дату и время.

Для каждого авторизированного пользователя предоставляется возможность просмотреть свою персональную страницу, управляя её составляющим. В любой момент времени пользователь может управлять и изменять забронированные сеансы, а также прочитать краткую информации о личном психологе.

Одним из важных достоинств приложения является предоставления удобного функционала для администратора, который в свою очередь имеет возможность вносить изменения в базу данных, в частности созданием страниц для специалистов и добавлении процедур.

Если посетитель не хочет иметь собственный аккаунт, или зашёл в ознакомительных целях, приложение предоставляет возможность войти в режиме гостя, который в свою очередь предоставляет ограниченный функционал.

# **1 Постановка задачи**

Для проектирования инфраструктуры базы данных (БД) психологического центра необходимо учесть различные аспекты, начиная с определения ролей пользователей. Роли, такие как администратор, клиент и гость, должны быть четко выделены с учетом их функциональных обязанностей и уровня доступа к данным.

Создание соответствующих таблиц в базе данных является ключевым этапом. Таблицы, такие как пользователи, психологи, процедуры и другие, будут формировать основу для хранения разнообразной информации, предоставляя структурированное хранение данных.

Далее необходимо реализовать триггеры, процедуры и функции. Триггеры будут использоваться для автоматизации действий при изменении данных, в то время как процедуры и функции выполнят более сложные задачи.

Для обеспечения удобного и безопасного взаимодействия с данными, необходимо использовать современные технологии репликации базы данных, мультимедийные типы данных, а также хеширование паролей.

Разработка WPF-приложения станет интерфейсом для взаимодействия с базой данных. Здесь важно обеспечить интуитивно понятный пользовательский интерфейс, который позволит пользователям легко взаимодействовать с данными и выполнять необходимые операции.

Оптимизация запросов с использованием индексов является важным аспектом для повышения производительности системы. Установка индексов на ключевые поля таблиц позволит ускорить выполнение запросов и обеспечить эффективное использование ресурсов.

В завершение, проведение тестов отказоустойчивости базы данных и приложения станет неотъемлемой частью процесса. Тестирование позволит оценить эффективность запросов, выявить возможные узкие места и определить необходимость оптимизации для обеспечения стабильной и высокопроизводительной работы базы данных психологического центра.

# **2 Проектирование базы данных**

Проектирование базы данных включает в себя несколько важных этапов, в том числе разработку обобщенной структуры управления приложением и проектирование модели базы данных.

На этом этапе определяются структура системы, взаимосвязи между ее компонентами и общие принципы функционирования, что обеспечивает ясное представление о том, как будут взаимодействовать база данных и приложение, что является критически важным для успешной реализации проекта.

## **2.1** **Обобщенная структура управления приложением**

В процессе анализа и разработки архитектуры проекта предполагается использование UML-диаграмм для более наглядного представления структуры, взаимодействия компонентов и логики приложения.

Диаграмма вариантов использования приложения представлена на рисунке 2.1.

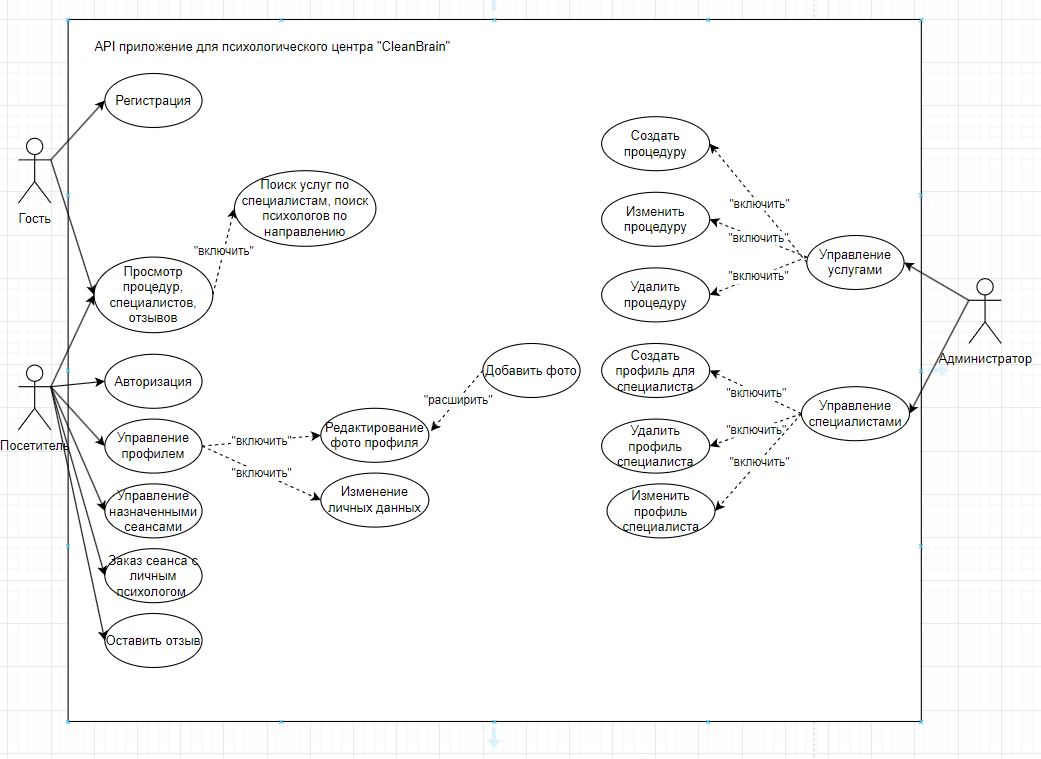


Рисунок 2.1 – Диаграмма использования приложения для клиента

Диаграмма вариантов использования в приложении "Clean Brain" является важным инструментом, иллюстрирующим взаимодействие различных пользовательских ролей.

Она облегчает коммуникацию, помогает пользователям понять, как каждая роль взаимодействует с системой.

Это визуальное представление также упрощает обучение новых пользователей, демонстрируя, как эффективно использовать приложение в соответствии с их ролями.

Она позволяет легко понять, какие функции доступны каждому из этих пользователей и как они могут взаимодействовать с приложением.

На рисунке 2.2 изображена диаграмма последовательности для авторизации пользователя.

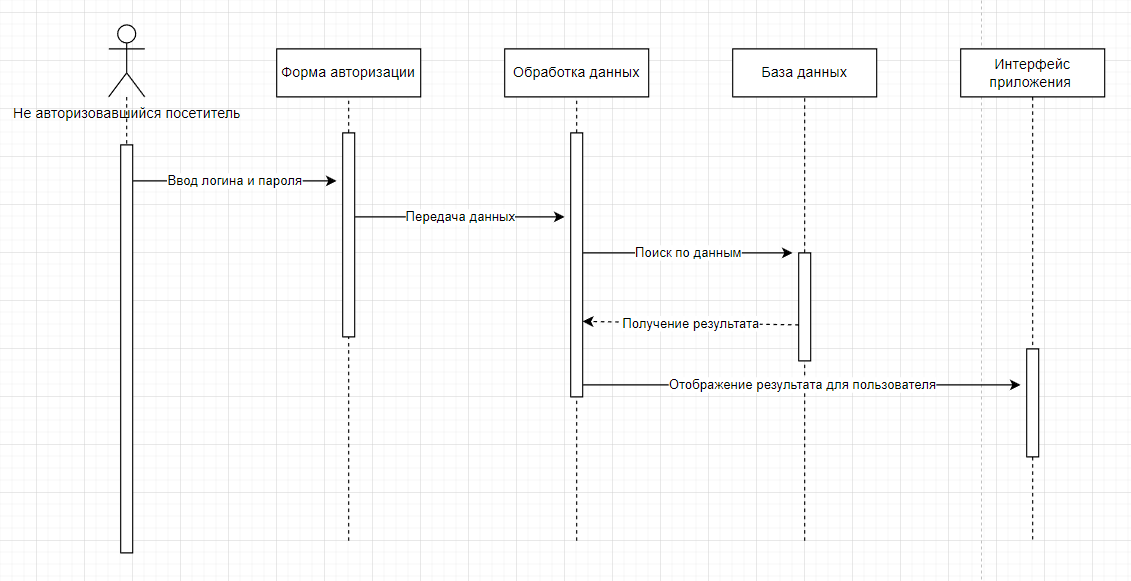


Рисунок 2.2 - Диаграмма последовательности для авторизации пользователя

Диаграмма последовательности для авторизации пользователя обобщённо показывает взаимодействие между клиентским приложением и сервером в процессе проверки учетных данных.

Пользователь инициирует процесс, отправляя свои учетные данные на сервер. Сервер, в свою очередь, осуществляет проверку данных. В случае успешной аутентификации сервер возвращает подтверждение, что пользователь является действительным.

## **2.2 Проектирование модели базы данных**

Следующим этапом выполнения курсовой работы будет создание логически взаимосвязанных таблиц. Чтобы составить визуальную взаимосвязанную структуры нашей базы данных, нам необходимо продумать, какая информация будет храниться в этих таблицах, после этого создать связи с помощью первичных и внешних ключей.

База данных состоит из 10 таблиц, перечисленных ниже (рисунок 2.3).

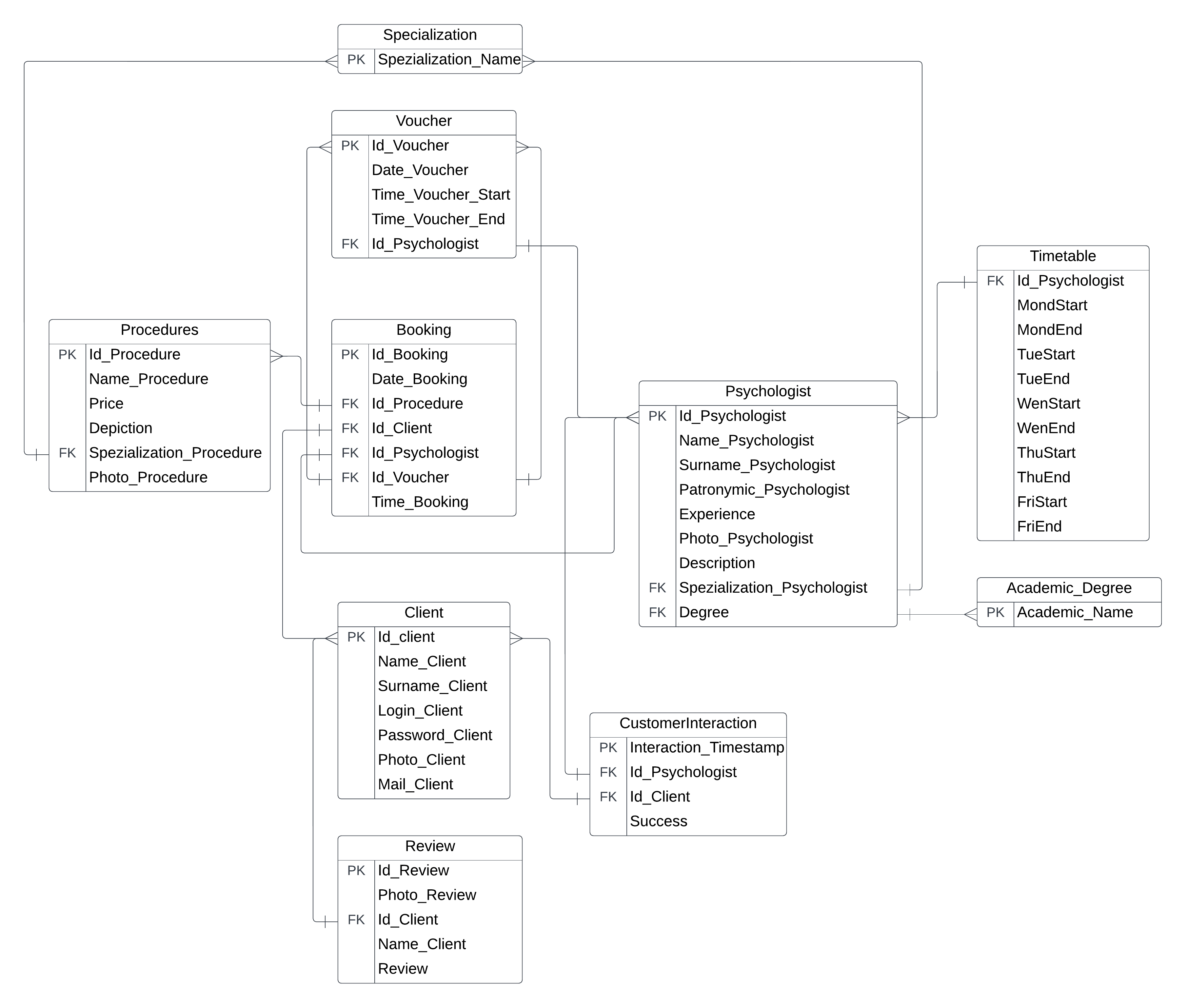


Рисунок 2.3 – Взаимосвязь таблиц данных

Таблица «Client» содержит информацию о клиентах. Описание ее полей приведено в таблице 2.1.

Таблица 2.1 – Структура таблицы «Client»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_client | number | Идентификатор клиента |
| Name\_Client | nvarchar2 | Имя |
| Surname\_Client | nvarchar2 | Фамилия |
| Login\_Client | nvarchar2 | Логин |
| Password\_Client | nvarchar2 | Пароль |
| Photo\_Client | blob | Фото клиента |
| Mail\_Client | nvarchar2 | Почта клиента |

Таблица «Psychologist» содержит информацию о психологах, включая имя, фамилию, отчество, фото психолога и так далее.

Значения идентификатора психолога должны быть уникальны.

Описание полей приведено в таблице 2.2.

Таблица 2.2 – Структура таблицы «Psychologist»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_Psychologist | number | Идентификатор психолога |
| Name\_Psychologist | nvarchar2 | Имя |
| Surname\_Psychologist | nvarchar2 | Фамилия |
| Patronymic\_Psychologist | nvarchar2 | Отчество |
| Experience | number | Опыт работы |
| Photo\_Psychologist | blob | Фото психолога |
| Description | nvarchar2 | Описание |
| Spezialization\_Psychologist | nvarchar2 | Специализация |
| Degree | nvarchar2 | Учёная степень |

Таблица «Specialization» содержит названия специализации.

Описание ее полей приведено в таблице 2.3.

Таблица 2.3 – Структура таблицы «Specialization»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Spezialization\_Name | nvarchar2 | Название специализации |

Таблица «Academic\_Degree» содержит название учёных степеней.

Описание ее полей приведено в таблице 2.4.

Таблица 2.4 – Структура таблицы «Academic\_Degree»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Academic\_Name | nvarchar2 | Название учёной степени |

Таблица «Voucher» содержит информацию о талончиках.

Описание ее полей приведено в таблице 2.5.

Таблица 2.5 – Структура таблицы «Voucher»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_Voucher | number | Идентификатор талончика |
| Date\_Voucher | date | Дата талончика |
| Time\_Voucher\_Start | timestamp | Начало сеанса |
| Time\_Voucher\_End | timestamp | Конец сеанса |
| Id\_Psychologist | number | Идентификатор психолога |

Таблица «Procedures» содержит информацию о процедурах, включая название, цену и описание.

Так же значения идентификатора процедуры, название должны быть строго уникальны.

Описание ее полей приведено в таблице 2.6.

Таблица 2.6 – Структура таблицы «Procedures»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_Procedure | number | Идентификатор процедуры |
| Name\_Procedure | nvarchar2 | Название процедуры |
| Price | number | Стоимость |
| Depiction | nvarchar2 | Описание |
| Spezialization\_Procedure | nvarchar2 | Специализация |
| Photo\_Procedure | blob | Фото процедуры |

Таблица «Timetable» содержит информацию о графике работы психолога. Описание ее полей приведено в таблице 2.7.

Таблица 2.7 – Структура таблицы «Timetable»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_Psychologist | number | Идентификатор психолога |
| MondStart | timestamp | Начало работы в понедельник |
| MondEnd | timestamp | Конец работы в понедельник |
| TueStart | timestamp | Начало работы во вторник |
| TueEnd | timestamp | Конец работы во вторник |
| WenStart | timestamp | Начало работы в среду |
| WenEnd | timestamp | Конец работы в среду |
| ThuStart | timestamp | Начало работы в четверг |
| ThuEnd | timestamp | Конец работы в четверг |
| FriStart | timestamp | Начало работы в пятницу |
| FriEnd | timestamp | Конец работы в пятницу |

Таблица «Booking» содержит информацию об оформлении сеанса. Описание ее полей приведено в таблице 2.8.

Таблица 2.8 – Структура таблицы «Booking»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_Booking | number | Идентификатор сеанса |
| Date\_Booking | date | Дата оформления сеанса |
| Id\_Procedure | number | Идентификатор процедуры |
| Id\_Client | number | Идентификатор клиента |
| Id\_Psychologist | number | Идентификатор психолога |
| Id\_Voucher | number | Идентификатор талончика |
| Time\_Booking | timestamp | Время оформления сенса |

Таблица «Review» содержит основную информацию об отзывах.

Описание ее полей приведено в таблице 2.9.

Таблица 2.9 – Структура таблицы «Review»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_Review | number | Идентификатор отзыва |
| Photo\_Review | blob | Фото отзыва |
| Id\_Client | number | Идентификатор клиента |
| Name\_Client | nvarchar2 | Имя клиента |
| Review | nvarchar2 | Отзыв |

Таблица «CustomerInteraction» содержит основную информацию об взаимодействиях клиента с психологом. Описание ее полей приведено в таблице 2.9.

Таблица 2.9 – Структура таблицы «CustomerInteraction»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_Psychologist | number | Идентификатор психолога |
| Id\_Client | number | Идентификатор клиента |
| Interaction\_Timestamp | timestamp | Время взаимодействия |
| Success | nvarchar2 | Успешность |

Данные таблицы, служат графическим представлением логической схемы базы данных. В этих таблицах отражены взаимосвязи и структура данных, которые обеспечивают организацию и хранение информации в базе данных.

3 Разработка объектов базы данных

## **3.1 Создание таблиц**

Все таблицы создаются в отдельном файле DBCW.sql, для дальнейшей возможности запускать скрипт в целом файле в новой базе данных. Таблица представляет собой структурированное представление данных в базе данных, организованное в виде матрицы из столбцов и строк. Каждый столбец таблицы имеет уникальное имя и тип данных, определяющий характер хранимой информации. Создание этих таблиц осуществляется с использованием команд блока PL/SQL и записывается в отдельном файле DBCW.sql для возможности последующего запуска скрипта в новой базе данных.

Таблицы, включаемые в базу данных, составляют "Client", "Psychologist", "Specialization", "Academic\_Degree", "Voucher", "Procedures", "Timetable", "Booking", "Review" и "CustomerInteraction". Каждая из них предназначена для хранения различных аспектов информации, связанных с клиентами, психологами, специализацией, ученой степенью, талончиками, процедурами, расписанием, сеансами, отзывами и взаимодействием с клиентами соответственно.

## **3.2 Создание ролей для разграничения доступа**

Для правильной работы десктоп приложения с базой данных необходимо разграничение ролей. Для выполнения данной задачи в СУБД Oracle предусмотрены встроенные функции, такие как пользователи, роли и привилегии.

В данном случае были созданы три роли и им были выданы привилегии, также было создано три пользователя и назначены им роли (листинг 3.1).

create role RLAdmin;

grant execute on AdminPackage to RLAdmin;

create user AdminUser IDENTIFIED by Qwerty12345;

grant RLAdmin to AdminUser

create role RLClient;

grant execute on CLientPackage to RLClient;

create user ClientUser IDENTIFIED by Qwerty12345;

grant RLClient to ClientUser;

create role RLGuest;

grant execute on GuestPackage to RLGuest;

create user GuestUser IDENTIFIED by Qwerty12345;

grant RLGuest to GuestUser;

Листинг 3.1 – Создание пользователей и выдача им привилегий

## **3.3 Создание пакетов и процедур для работы с базой данных**

После установки ролей и создания всех необходимых таблиц начинаем работу по созданию процедур для управления данными. В контексте Oracle PL/SQL, пакет представляет собой объект схемы, который объединяет логически связанные типы, элементы и подпрограммы. Обычно пакеты состоят из спецификации и тела, при этом иногда тело может быть необязательным. Хранимая процедура (stored procedure) – это программа, которая выполняет определенные действия с данными в базе данных и хранится непосредственно в базе данных.

Для логического разделения процедур были разработаны три пакета. Первый - GuestPackage, предназначенный для гостей, второй - ClientPackage, предоставляющий процедуры для работы с клиентскими данными, третий - AdminPackage, содержащий процедуры для администратора базы данных.

Заголовок пакета, содержащий процедуры для работы гостя представлен в листинге 3.2. Тело пакета расположено в приложении А [[1]](#_Приложение_А).

create or replace package GuestPackage as

PROCEDURE GetClientById(p\_Id\_Client IN NUMBER,p\_Cursor OUT SYS\_REFCURSOR);

PROCEDURE GetAllClients(p\_Cursor OUT SYS\_REFCURSOR);

PROCEDURE GetAllReviews(p\_Cursor OUT SYS\_REFCURSOR);

PROCEDURE GetTimeTableByPsychologistId(p\_Id\_Psychologist IN NUMBER, p\_Cursor OUT SYS\_REFCURSOR);

PROCEDURE GetSpecializationByName(p\_SpecializationName IN VARCHAR2, p\_Cursor OUT SYS\_REFCURSOR);

PROCEDURE GetDegreeByName(p\_Degree\_Name IN VARCHAR2, p\_Cursor OUT SYS\_REFCURSOR);

PROCEDURE GetProcedureById(p\_Procedure\_Id IN NUMBER, p\_Cursor OUT SYS\_REFCURSOR);

PROCEDURE GetAllProcedures(p\_Cursor OUT SYS\_REFCURSOR);

PROCEDURE GetPsychologistById(p\_Id\_Psychologist IN NUMBER, p\_Cursor OUT SYS\_REFCURSOR);

PROCEDURE GetAllPsychologists(p\_Cursor OUT SYS\_REFCURSOR);

END GuestPackage;

/

Листинг 3.2 – Заголовок пакета GuestPackage

Пакет включает процедуры GetClientById, GetSpecializationByName, GetDegreeByName, GetPsychologistById, которые служат для получения информации конкретного клиента, специализации, учёной степени и психолога. Процедуры GetAllClients, GetAllReviews, GetAllProcedures, GetAllPsychologists предоставляют полный список клиентов, отзывов, процедур и психологов.

Заголовок пакета, содержащий процедуры для клиента представлен в листинге 3.3. Тело пакета расположено в приложении Б [[2]](#_Приложение_Б).

create or replace package CLientPackage as

PROCEDURE GetClientById( p\_Id\_Client IN NUMBER, p\_Cursor OUT SYS\_REFCURSOR);

PROCEDURE CreateReview(p\_Photo\_Review IN BLOB, p\_Name\_Client IN VARCHAR2, p\_Id\_Client IN NUMBER, p\_Review\_Text IN VARCHAR2, p\_Result OUT NUMBER);

PROCEDURE DeleteReviewById(p\_Id IN NUMBER, p\_Success OUT NUMBER);

PROCEDURE UpdateReview(p\_Review\_Id IN NUMBER, p\_Photo\_Review IN BLOB, p\_Name\_Client IN VARCHAR2, p\_Review\_Text IN VARCHAR2, p\_Result OUT NUMBER);

PROCEDURE GetAllReviews(p\_Cursor OUT SYS\_REFCURSOR);

PROCEDURE GetTimeTableByPsychologistId(p\_Id\_Psychologist IN NUMBER, p\_Cursor OUT SYS\_REFCURSOR);

PROCEDURE GetSpecializationByName(p\_SpecializationName IN VARCHAR2, p\_Cursor OUT SYS\_REFCURSOR);

PROCEDURE GetDegreeByName(p\_Degree\_Name IN VARCHAR2, p\_Cursor OUT SYS\_REFCURSOR);

PROCEDURE GetProcedureById(p\_Procedure\_Id IN NUMBER, p\_Cursor OUT SYS\_REFCURSOR);

PROCEDURE GetAllProcedures(p\_Cursor OUT SYS\_REFCURSOR);

PROCEDURE GetVoucherById(p\_Id\_Voucher IN NUMBER, p\_Cursor OUT SYS\_REFCURSOR);

PROCEDURE GetAllVouchers(p\_Cursor OUT SYS\_REFCURSOR);

PROCEDURE CreateBooking(p\_Date\_Booking IN DATE, p\_Id\_Procedure IN NUMBER, p\_Id\_Client IN NUMBER, p\_Id\_Psychologist IN NUMBER, p\_Id\_Voucher IN NUMBER, p\_Time\_Booking IN TIMESTAMP, p\_Result OUT NUMBER);

PROCEDURE GetBookingById(p\_BookingId IN NUMBER, p\_Cursor OUT SYS\_REFCURSOR);

PROCEDURE GetAllBookings(p\_Cursor OUT SYS\_REFCURSOR);

PROCEDURE DeleteBooking(p\_BookingId IN NUMBER, p\_Success OUT NUMBER);

PROCEDURE DeleteBookingByIdPsy(p\_Id\_Psychologist IN NUMBER, p\_Success OUT NUMBER);

PROCEDURE GetPsychologistById(p\_Id\_Psychologist IN NUMBER, p\_Cursor OUT SYS\_REFCURSOR);

PROCEDURE GetAllPsychologists(p\_Cursor OUT SYS\_REFCURSOR);

END ClientPackage;

/

Листинг 3.3 – Заголовок пакета ClientPackage

Данный пакет содержит все необходимые функции для получения информации о психологах, процедурах и отзывах. Пользователь с помощью процедур CreateBooking, CreateReview может бронировать сеансы, а также создать отзыв. Так же в пакете присутствуют процедуры для удаления, изменения информации о сеансах и отзывах.

Часть заголовка пакета, содержащего процедуры для работы администратора базы данных представлен в листинге 3.4.

create or replace package AdminPackage as

PROCEDURE InsertNewClient(p\_Name\_Client IN VARCHAR2, p\_Surname\_Client IN VARCHAR2, p\_Login\_Client IN VARCHAR2, p\_Password\_Client IN VARCHAR2, p\_Photo\_Client IN BLOB, p\_Mail\_Client IN VARCHAR2, p\_Client\_Id OUT NUMBER);

PROCEDURE UpdateClientInfo(p\_Id\_Client IN NUMBER, p\_Name\_Client IN VARCHAR2, p\_Surname\_Client IN VARCHAR2, p\_Photo\_Client IN BLOB, p\_Mail\_Client IN VARCHAR2, p\_Result OUT NUMBER);

PROCEDURE DeleteReviewById(p\_Id IN NUMBER, p\_Success OUT NUMBER);

PROCEDURE GetAllReviews(p\_Cursor OUT SYS\_REFCURSOR);

PROCEDURE CreateTimeTable(p\_Id\_Psychologist IN NUMBER, p\_MondStart IN TIMESTAMP, p\_MondEnd IN TIMESTAMP, p\_TueStart IN TIMESTAMP, p\_TueEnd IN TIMESTAMP, p\_WenStart IN TIMESTAMP, p\_WenEnd IN TIMESTAMP, p\_ThuStart IN TIMESTAMP, p\_ThuEnd IN TIMESTAMP, p\_FriStart IN TIMESTAMP, p\_FriEnd IN TIMESTAMP, p\_Result OUT NUMBER);

PROCEDURE DeleteTimeTableById(p\_Id\_Psychologist IN NUMBER, p\_Result OUT NUMBER);

PROCEDURE AddSpecialization(p\_Spezialization\_Name IN VARCHAR2, p\_Result OUT NUMBER);

PROCEDURE AddDegree(p\_Degree\_Name IN VARCHAR2, p\_Result OUT NUMBER);

PROCEDURE AddProcedure(p\_Name\_Procedure IN NVARCHAR2, p\_Price IN FLOAT, p\_Depiction IN NVARCHAR2, p\_Spezialization\_Procedure IN NVARCHAR2, p\_Photo\_Procedure IN BLOB, p\_Result OUT NUMBER);

PROCEDURE DeleteProcedure(p\_Id\_Procedure IN NUMBER, p\_Result OUT NUMBER);

PROCEDURE AddPsychologist(p\_Name\_Psychologist IN NVARCHAR2, p\_Surname\_Psychologist IN NVARCHAR2, p\_Patronymic\_Psychologist IN NVARCHAR2, p\_Experience IN NUMBER, p\_Photo\_Psychologist IN BLOB, p\_Description IN NVARCHAR2, p\_Specialization\_Psychologist IN NVARCHAR2, p\_Degree IN NVARCHAR2, p\_Result OUT NUMBER);

PROCEDURE DeletePsychologist(p\_Id\_Psychologist IN NUMBER, p\_Result OUT NUMBER);

END AdminPackage;

/

Листинг 3.4 – Заголовок пакета AdminPackage

Тело пакета администратора расположено в приложении В [[3]](#_Приложение_В).

Данный пакет содержит процедуры, предоставляющие возможность добавления и удаления психологов, процедур, специализаций, учёных степеней, а также графиков работы психологов. С помощью процедур InsertNewClient, UpdateClientInfo администратор может добавлять клиентов, а также изменять их информацию.

# **4 Описание процедур экспорта и импорта**

Для комфортного администрирования базы данных в ней необходимо реализовать возможность экспорта и импорта данных в удобный для администратора формат. В нашем случае это формат json;

## **4.1 Описание процедуры экспорта**

Для доступа к папке из СУБД Oracle нужно должен быть создан UTL\_DIR. Объект UTL\_DIR является логической ссылкой в базе данных на каталог файловой системы сервера, где установлена база данных Oracle. Имена объектов UTL\_DIR уникальны внутри всей базы данных. (листинг 4.1).

CREATE OR REPLACE DIRECTORY UTL\_DIR AS 'C:/JSON';

Листинг 4.1 – Создание объекта UTL\_DIR

Далее необходимо разработать процедуру ExportToJSON, которая представлена на листинге 4.2.

CREATE OR REPLACE PROCEDURE ExportToJSON IS

v\_file UTL\_FILE.FILE\_TYPE;

v\_row Client%ROWTYPE;

BEGIN

v\_file := UTL\_FILE.FOPEN('UTL\_DIR','Clients.json','W');

UTL\_FILE.PUT\_LINE(v\_file, '[');

FOR v\_row in (select JSON\_OBJECT(

Id\_client,Name\_Client,Surname\_Client,Login\_Client,Password\_Client,Mail\_Client) as json\_data from Client)

LOOP

UTL\_FILE.PUT\_LINE(v\_file ,v\_row.json\_data || ',');

END LOOP;

UTL\_FILE.PUT\_LINE(v\_file, ']');

UTL\_FILE.FCLOSE(v\_file);

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLCODE || ' - ' || SQLERRM);

RAISE;

END;

Листинг 4.2 – Процедура для экспорта данных

Данная процедура экспортирует данные таблицы Client в файл Clients.json

## **4.2 Описание процедуры импорт**

Процедуру импорта рассмотрим также на таблице Client. Процедура импорта представлена в листинге 4.3.

CREATE OR REPLACE PROCEDURE ExportToJSON IS

v\_file UTL\_FILE.FILE\_TYPE;

v\_row Client%ROWTYPE;

BEGIN

v\_file := UTL\_FILE.FOPEN('UTL\_DIR','Clients.json','W');

UTL\_FILE.PUT\_LINE(v\_file, '[');

FOR v\_row in (select JSON\_OBJECT(

Id\_client,Name\_Client,Surname\_Client,Login\_Client,Password\_Client,Mail\_Client) as json\_data from Client)

LOOP

UTL\_FILE.PUT\_LINE(v\_file ,v\_row.json\_data || ',');

END LOOP;

UTL\_FILE.PUT\_LINE(v\_file, ']');

UTL\_FILE.FCLOSE(v\_file);

EXCEPTION WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLCODE || ' - ' || SQLERRM);

RAISE;

END;

Листинг 4.3 – Процедура для импорта данных

В результате импорта мы получим заполненную, ранее экспортированными, данными таблицу Client.

# **5 Тестирование производительности**

## **5.1 Оптимизирование запросов**

Для проверки производительности базы данных необходимо заполнить ее большим количеством различных данных и узнать время выполнения одного запроса. Для этого с помощью анонимного блока, код которого представлен в листинге 5.1, было добавлено 100000 строк.

DECLARE

v\_Psychologist\_Id NUMBER;

v\_Client\_Id NUMBER;

v\_Timestamp TIMESTAMP(9);

v\_Success VARCHAR2(10);

BEGIN

FOR i IN 1..100000 LOOP

v\_Psychologist\_Id := 70;

v\_Client\_Id := 153;

v\_Timestamp := CURRENT\_TIMESTAMP + INTERVAL '1' SECOND \* i;

v\_Success := CASE WHEN MOD(i, 2) = 0 THEN 'ДА' ELSE 'НЕТ' END;

INSERT INTO CustomerInteraction(Id\_Psychologist, Id\_Client, Interaction\_Timestamp, Success)

VALUES (v\_Psychologist\_Id, v\_Client\_Id, v\_Timestamp, v\_Success);

IF MOD(i, 1000) = 0 THEN

COMMIT;

END IF;

END LOOP;

COMMIT;

END;

Листинг 5.1 – Заполнение таблицы

В результате таблица CustomerInteraction заполнена 100000 строка, представлено на рисунке 5.1.

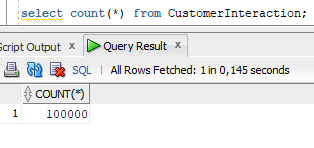


Рисунок 5.1 – Количество записей в таблице

Построим план запроса по всем элементам таблицы и посмотрим время его выполнения без индекса.

План запроса представлен на рисунке 5.2

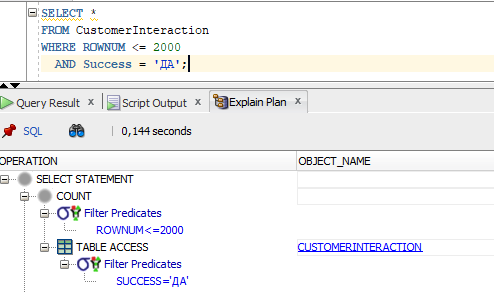


Рисунок 5.2 – План запроса без индекса

Теперь попробуем оптимизировать данный запрос с помощью индекса, созданный для данной таблицы, представлено в листинге 5.2.

CREATE INDEX idx\_CustomerInteraction\_Success

ON CustomerInteraction(Success);

Листинг 5.2 – Создание индекса

Построим план запроса по всем элементам таблицы и посмотрим время его выполнения вместе с индексом (рисунок 5.3).

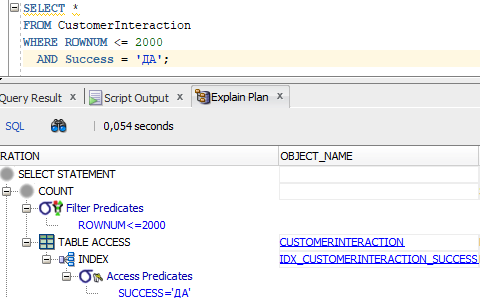


Рисунок 5.3 – План запроса с индексом

Исходя из приведенных выше результатов запросов, можно увидеть, что добавление индекса привело к уменьшению стоимости запроса примерно в 3 раза.

## **5.2 Проверка выполнения хранимых процедур**

Теперь протестируем основные хранимые процедуры таблицы Client: InsertNewClient, UpdateClientInfo, GetClientById.

При помощи процедуры InsertNewClient добавим нового пользователя – рисунок 5.4.

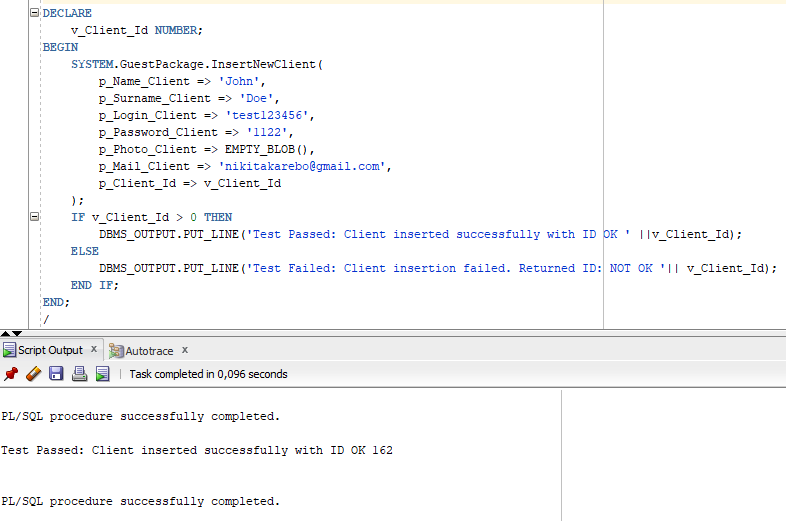


Рисунок 5.4 – Добавление нового пользователя

После успешного выполнения процедуры InsertNewClient новый клиент успешно добавлен, и это подтверждается выводом надписи "ОК".

Далее обновим информацию о клиенте с помощью процедуры UpdateClientInfo (рисунок 5.5).

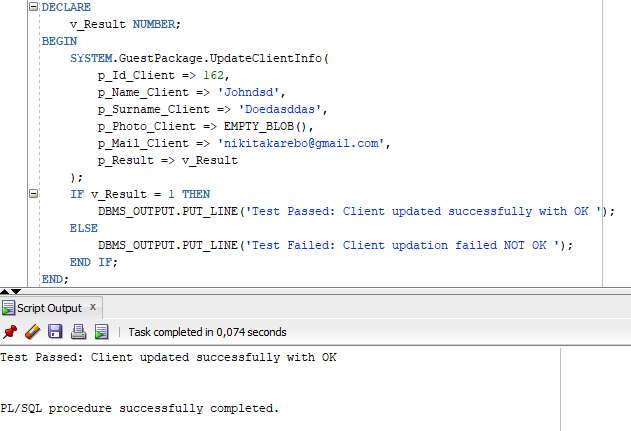


Рисунок 5.5 – Обновление информации о клиенте

Далее найдём клиента с помощью процедуры GetClientById, которая принимает в качестве параметра id клиента.

Выполнение процедуры GetClientById представлено на рисунке 5.6.

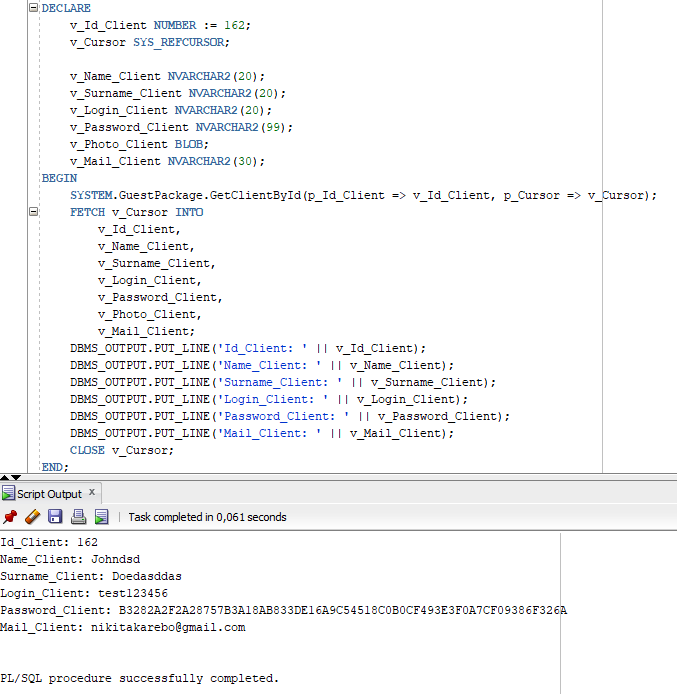


Рисунок 5.6 – Поиск информации о клиенте

После выполнения данной процедуры, выводится вся личная информация о клиенте, охватывающая различные аспекты его данных.

# **6 Описание и применение технологии в БД**

## **6.1 Репликация данных**

В процессе разработки проекта активно применялась технология репликации данных между серверами системы управления базами данных (СУБД). Концепция репликации в данном контексте заключается в том, чтобы поддерживать несколько наборов данных в состоянии взаимной согласованности.

Другими словами, изменения, внесенные в один из наборов данных (реплику), автоматически отражаются в другом (основном).

В нашем проекте выбор пал на репликацию, осуществляемую через использование триггеров.

Триггеры – это хранимые процедуры, которые автоматически выполняются при определенных изменениях в данных. Эти изменения могут включать в себя добавление, изменение или удаление записей.

Специально разработанные триггеры в данной системе могут фиксировать изменения в специальной таблице, из которой затем процесс на стороне реплики извлекает эти изменения.

Такой подход к репликации обеспечивает нам важные преимущества, такие как независимость от версий основной базы данных и реплики, а также обширные возможности по преобразованию данных в соответствии с нуждами проекта.

Среди функциональных задач, которые успешно решаются через репликацию, можно выделить поддержку резервной базы данных для обеспечения надежной защиты от потери основного набора данных.

Важно отметить, что, несмотря на высокую функциональность, подход через триггеры не лишен некоторых ограничений.

В частности, он может привести к увеличению нагрузки на основную базу данных и вызвать заметную задержку в процессе репликации. Кроме того, в связи с объемом кода в триггерах, возникает потребность в использовании специализированного программного обеспечения для их генерации, такого как "Репликация слиянием" в Microsoft SQL Server или Slony-I для PostgreSQL.

Подключение к другой базе данных осуществлялось с помощью CWCON. Скрипт создания объекта CWCON показан в листинге 6.1.

CREATE PUBLIC DATABASE LINK CWCON

CONNECT TO CommonUser

IDENTIFIED BY Qwerty1111

USING '172.24.93.152:1521/CLEANBRAINMAINPC\_PDB.be.by';

Листинг 6.1 – Скрипт создания объекта CWCON

Далее в проекте были разработаны триггеры для каждой таблицы, которые автоматически реагируют на удаление, изменение или добавление записей, осуществляя те же самые изменения на удаленном сервере.

Пример такого триггера, примененного к таблице Client, представлен в листинге 6.2.

CREATE OR REPLACE TRIGGER ClientTrigger

AFTER INSERT OR DELETE OR UPDATE

ON Client

FOR EACH ROW

DECLARE

BEGIN

IF (INSERTING) THEN

INSERT INTO Client@CWCON(Id\_client, Name\_Client, Surname\_Client, Login\_Client, Password\_Client, Mail\_Client)

VALUES(:NEW.Id\_client, :NEW.Name\_Client, :NEW.Surname\_Client, :NEW.Login\_Client, :NEW.Password\_Client, :NEW.Mail\_Client);

END IF;

IF (UPDATING) THEN

UPDATE Client@CWCON

SET Name\_Client = :NEW.Name\_Client,

Surname\_Client = :NEW.Surname\_Client,

Login\_Client = :NEW.Login\_Client,

Password\_Client = :NEW.Password\_Client,

Mail\_Client = :NEW.Mail\_Client

WHERE Id\_client = :OLD.Id\_client;

END IF;

IF (DELETING) THEN

DELETE FROM Client@CWCON WHERE Id\_client = :OLD.Id\_client;

END IF;

END;

Листинг 6.2 – Триггер для таблицы Client

Триггер использует объект CWCON для создания удаленного соединения с пользователем, обеспечивая передачу необходимых данных в процессе установки связи.

## **6.2 Мультимедийные типы данных**

Для работы с данными большого объема СУБД Oracle предоставляет типы данных BLOB, CLOB, NCLOB и BFILE. Здесь LOB означает Large OBject, или большой объект, и далее по тексту термины LOB и "большой объект" взаимозаменяемы. По сути, большой объект – это абстрактный тип для манипуляции данными большого объема внутри БД, а типы BLOB, CLOB, NCLOB и BFILE - его конкретные реализации.

В данном курсовом проекте технология мультимедийных типов данных отражена в необходимости хранить изображения, являющиеся иконками пользователей и психологов. Речь идет о таблицах Client и Psychologist, Procedures. Для хранения в поле изображений в СУБД применяется тип BLOB. Тип BLOB представляет из себя массив двоичных данных, поэтому процесс сводился к разбиению мультимедийных файлов на массив битов при загрузке их в базу данных, и к сборке изображения из массива битов на обратном пути. Рассмотрим сначала процесс преобразования изображения в тип данных, понятный Oracle (листинг 6.3).

return changeImage ?? (changeImage = new RelayCommand(obj =>

{

string path;

OpenFileDialog openFile = new OpenFileDialog();

openFile.Filter = "Image files (\*.png;\*.jpg)|\*.png;\*.jpg|All files (\*.\*)|\*.\*";

if (openFile.ShowDialog() == true)

{

path = openFile.FileName;

byte[] imageBytesBuffer;

using (FileStream fs = new FileStream(path, FileMode.Open, FileAccess.Read))

{

imageBytesBuffer = new byte[fs.Length];

fs.Read(imageBytesBuffer, 0, imageBytesBuffer.Length);

}

imageBytes = imageBytesBuffer;

Image = GetImage(imageBytesBuffer);

}

}));

Листинг 6.3 – Загрузка изображения и разбиение его на массив битов

После превращения изображения в массив битов дело остается за малым, а именно: используя драйвер Managed Data Access мы отправляем INSERT-запрос для вставки изображения в существующую строку, где с помощью подготовленного заявления передаем этот массив для вставки в поле Photo\_Client. Это можно увидеть в листинге 6.4.

OracleConnection conn = DBUtils.GetDBConnection("admin");

try

{

conn.Open();

using (OracleCommand cmd = new OracleCommand($"{GetRole()}InsertNewClient", conn))

{

cmd.CommandType = System.Data.CommandType.StoredProcedure;

cmd.Parameters.Add("p\_Name\_Client", OracleDbType.Varchar2).Value = client.Name\_Client;

cmd.Parameters.Add("p\_Surname\_Client", OracleDbType.Varchar2).Value = client.Surname\_Client;

cmd.Parameters.Add("p\_Login\_Client", OracleDbType.Varchar2).Value = client.Login\_Client;

cmd.Parameters.Add("p\_Password\_Client", OracleDbType.Varchar2).Value = client.Password\_Client;

cmd.Parameters.Add("p\_Photo\_Client", OracleDbType.Blob).Value = client.Photo\_Client;

cmd.Parameters.Add("p\_Mail\_Client", OracleDbType.Varchar2).Value = client.Mail\_Client;

OracleParameter clientIdParam = new OracleParameter("p\_Client\_Id", OracleDbType.Int32);

clientIdParam.Direction = System.Data.ParameterDirection.Output;

cmd.Parameters.Add(clientIdParam);

cmd.ExecuteNonQuery();

OracleDecimal temp = (OracleDecimal)clientIdParam.Value;

int clientId = temp.ToInt32();

conn.Close();

return clientId;

}

}

catch (Exception ex)

{

return -1;

}

Листинг 6.4 – Загрузка изображения в базу данных

Таким образом была реализована работа с изображениями в базе данных.

## **6.3 Технология шифрования**

В простейшем определении шифрование представляет собой «маскировку» данных, или их преобразование, при котором данные не могут использоваться посторонними.

Для шифрования в PL/SQL используется встроенный пакет Oracle DBMS\_CRYPTO.

Была разработана функция EncryptPassword. Ее реализацию можно увидеть в листинге 6.5.

CREATE OR REPLACE FUNCTION EncryptPassword(p\_Password IN VARCHAR2)

RETURN NVARCHAR2

AS

BEGIN

IF p\_Password IS NULL

THEN

RETURN NULL;

ELSE

RETURN dbms\_crypto.hash(utl\_raw.cast\_to\_raw(TRIM(p\_Password)), dbms\_crypto.hash\_sh256);

END IF;

END;

Листинг 6.5 – Функция EncryptPassword

Эта функция позволяет зашифровать пароль пользователя. Она принимает один входной параметр: пароль пользователя, который следует зашифровать. В моем курсовом проекте для шифрования используется алгоритм AES с 128-разрядным ключом, метод сцепления CBC и механизм заполнения PKCS#5.

# **7 Руководство пользователя**

Окно регистрации содержит поля для ввода учетных данных, таких как электронная почта и пароль. Пользователь может использовать эти поля для авторизации, если у него уже есть учетная запись. После ввода данных пользователь нажимает на кнопку "Войти" или аналогичную, чтобы авторизоваться в приложении.

В случае, если пользователь не имеет учетной записи или не хочет создавать новую, он может выбрать опцию "Гость" или "Войти как гость". При выборе этой опции приложение предоставляет пользователю ограниченные возможности функционала, не требуя от него создания аккаунта.

Форма регистрации представлена на рисунке 7.1.

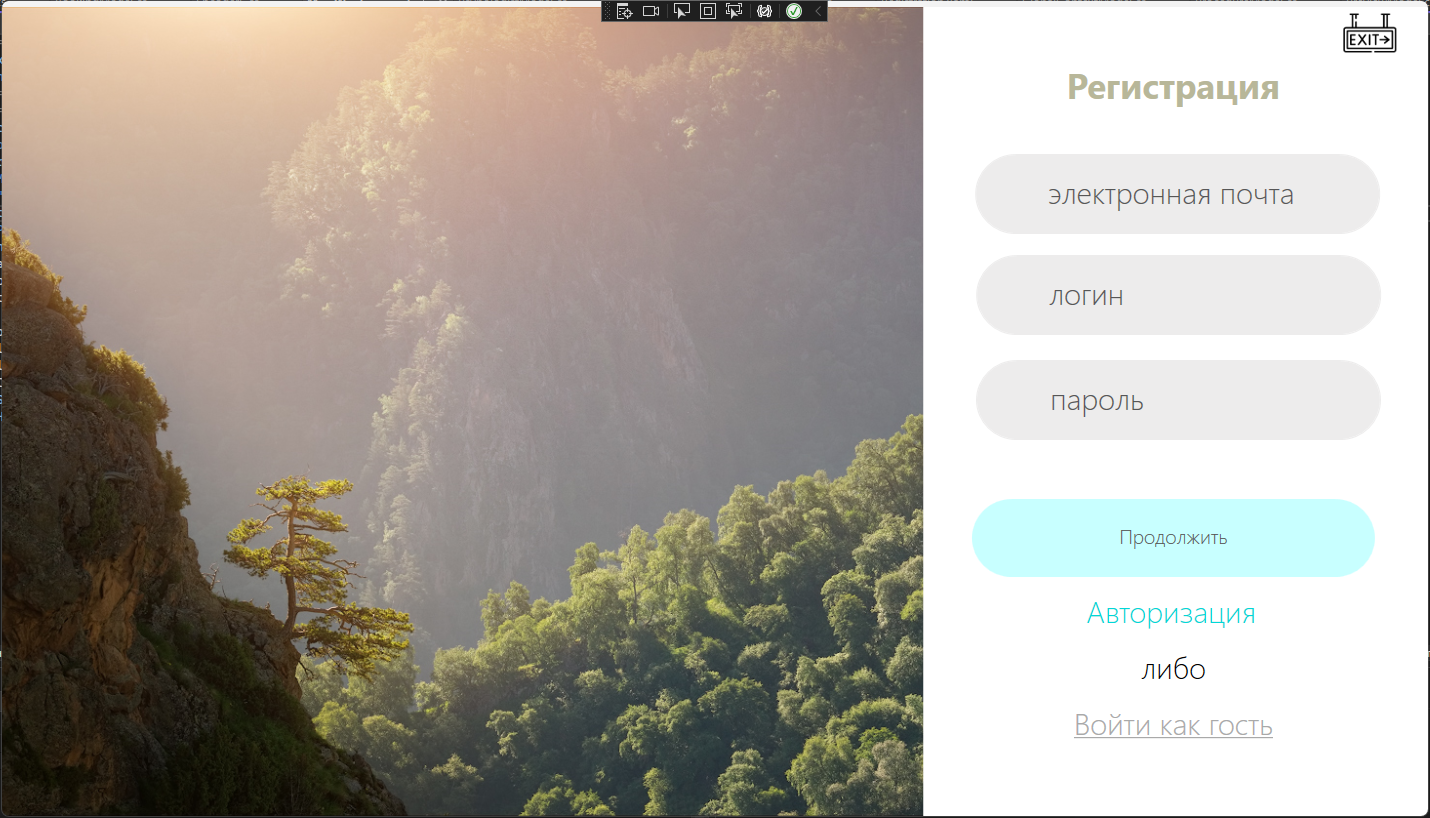


Рисунок 7.1 - Окно регистрации

Если у пользователя имеется личный аккаунт в приложении, ему всегда доступна авторизация.

При выборе опции авторизации, пользователю предлагается ввести свой логин и пароль в соответствующие поля. Пароль вводится в отдельное поле в зашифрованном виде для обеспечения безопасности.

При вводе данных пользователь может проверить являются ли они валидными, а именно будет отображаться индикатор в виде галочки, что говорит о валидности данных, а также крестик, что обозначает противоположный результат.

После заполнения полей логина и пароля пользователь нажимает на кнопку "Войти", чтобы произвести авторизацию в приложении.

Форма авторизации представлена на рисунке 7.2.

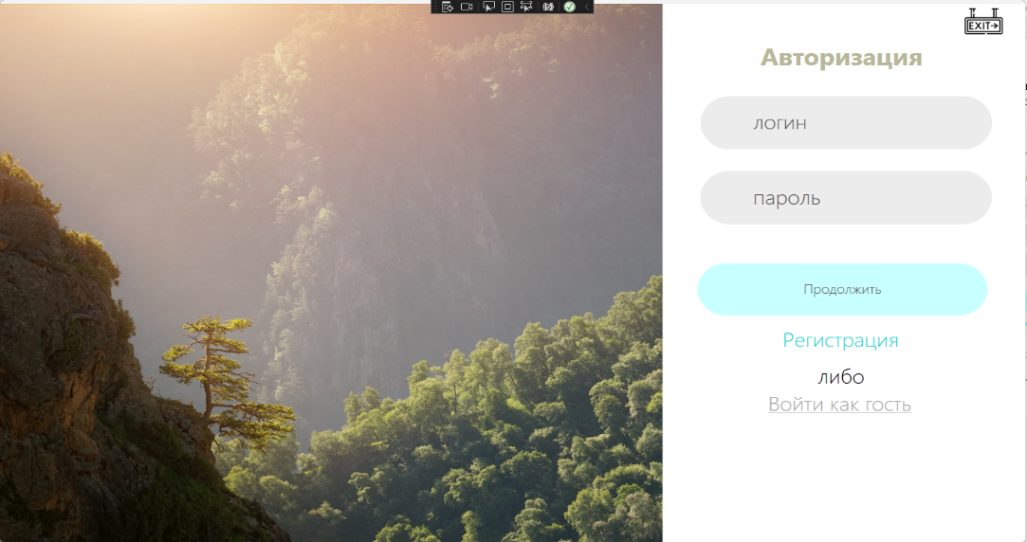


Рисунок 7.2 - Окно авторизации

Данные формы общие как для гостя, администратора и для авторизированного пользователя.

## **7.1 Руководство по использованию администраторам**

После входа в приложения администратора встречает приветствующая страница, а также навигационное меню, расположенное вертикальном положении.

Данная страница изображена на рисунке 7.3.

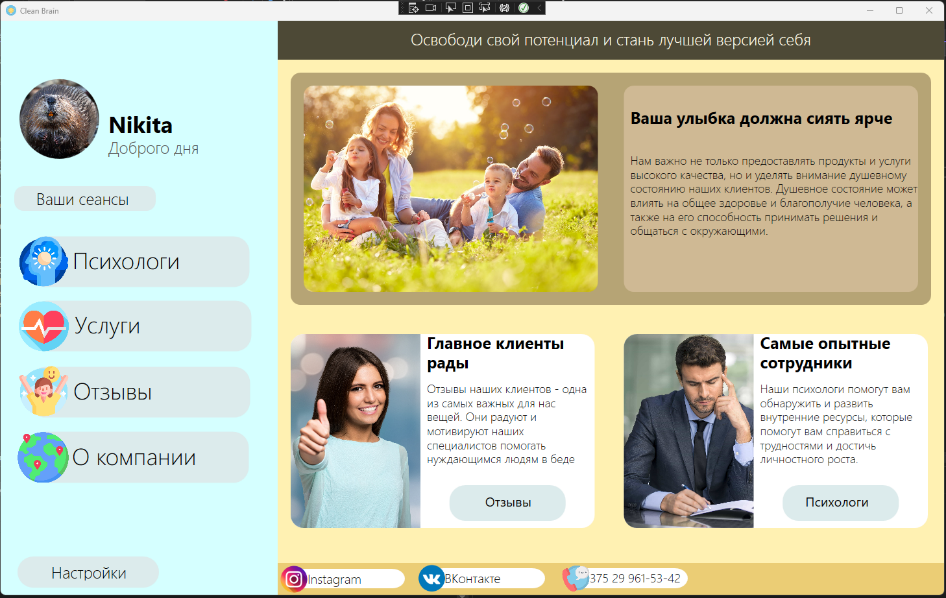


Рисунок 7.3 - Приветствующая страница

Нажав на кнопку “Психологи” мы переходим к общему перечню психологов, которых мы можем отсортировать по возможным специализациям. Так же для администратора имеется дополнительная кнопка   
“добавить”, с помощью которого можно добавить нового психолога в базу данных. Представлено на рисунке 7.4.



Рисунок 7.4 - Страница со списком психологов

При нажатии на кнопку “добавить”, администратор попадает в новое окно, с помощью которого может добавить психолога в базу данных. При добавлении и выхода из страницы с существующим психологом, талончики автоматически обновляются. Данная форма представлена на рисунке 7.5.

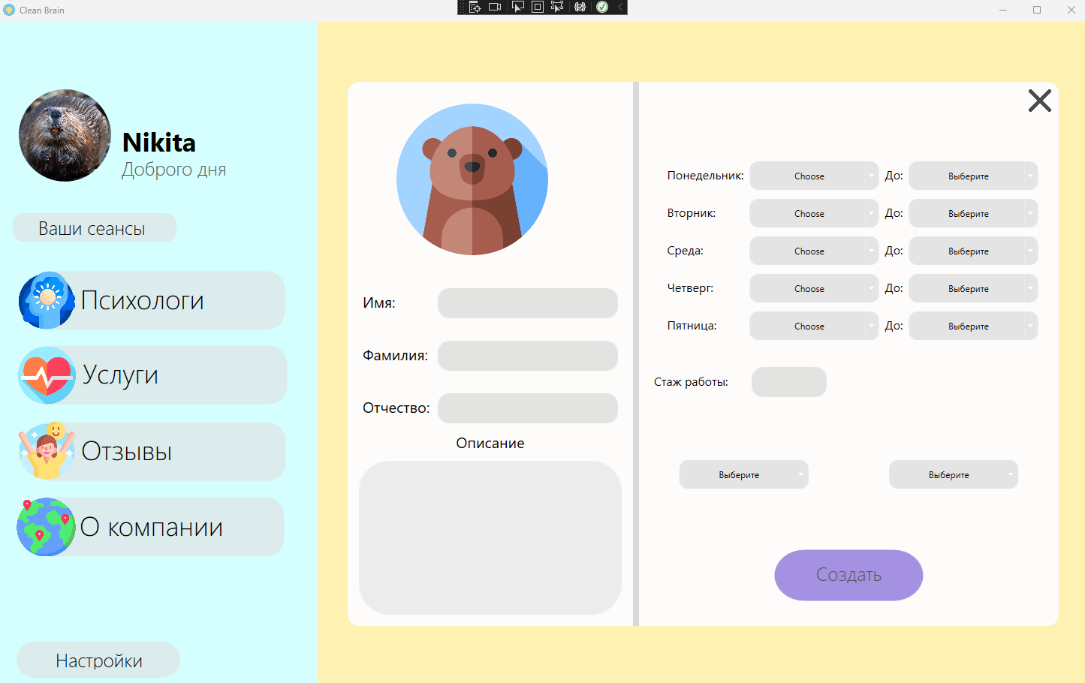


Рисунок 7.5 - Страница добавления психолога

При клике на изображение психолога из списка, администратору открывается окно с заполненными данными доступные для редактирования. Также присутствует кнопка “удалить”, которая удалит текущего психолога из базы данных.

Окно редактирования психолога представлено на рисунке 7.6.

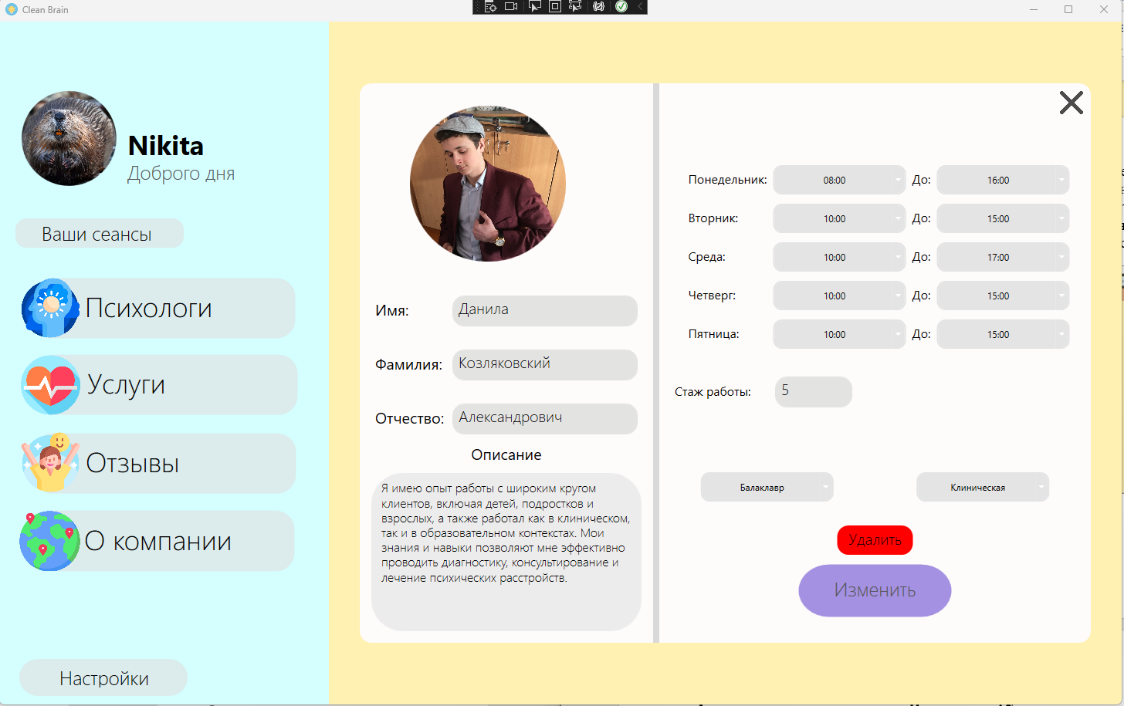


Рисунок 7.6 - Страница редактирования психолога

Далее нажав на кнопку “Услуги”, мы попадаем на страницу, где выдается перечень услуг, которые так же можно сортировать и добавлять новые в базу данных, представлено на рисунке 7.7.

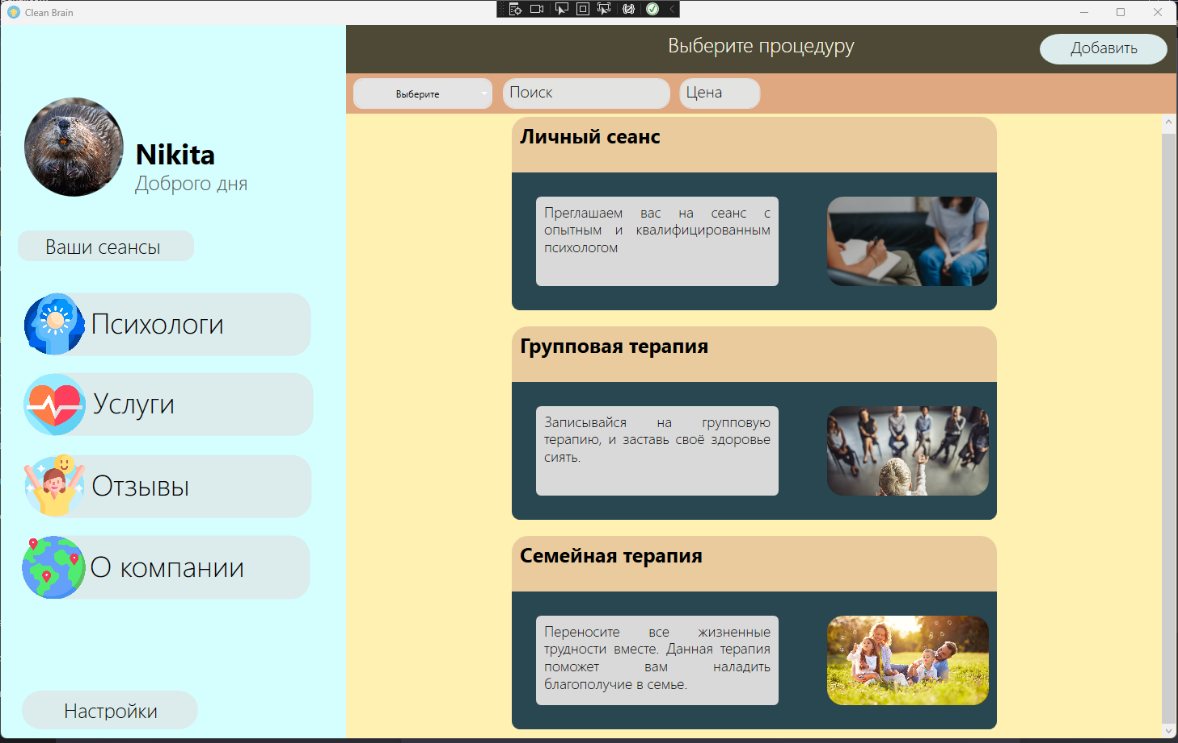


Рисунок 7.7 - Страница с перечнем услуг

Нажав на кнопку “добавить”, администратор переходит в новое окно, в которой находится форма для добавления новой услуги. В данной форме необходимо заполнить обязательные поля, для дальнейшего добавлении в базу данных, представлено на рисунке 7.8.

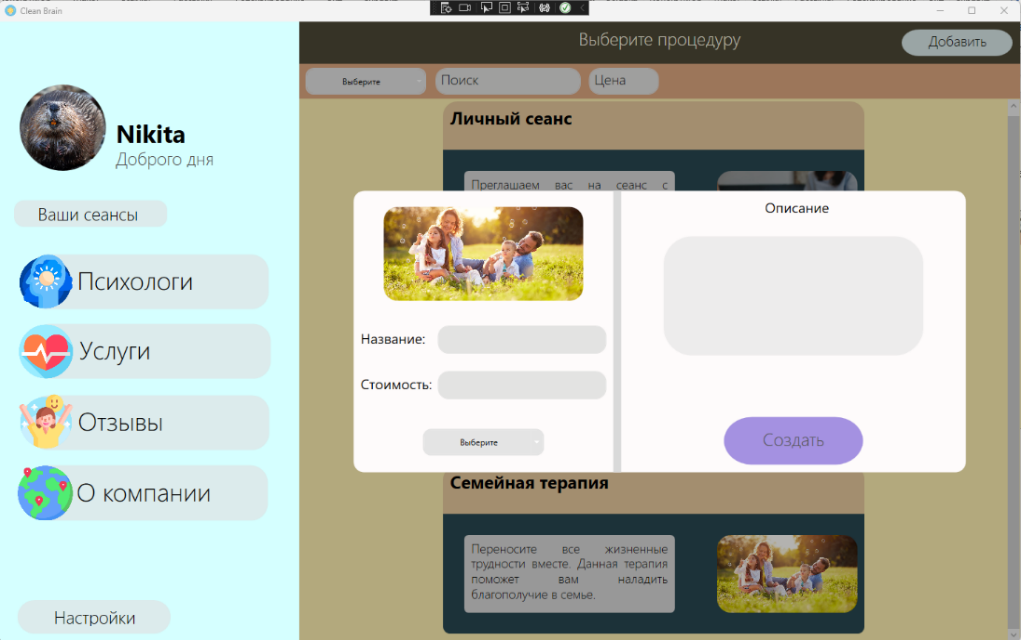


Рисунок 7.8 - Форма для добавления услуги

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.

Данная форма представлена на рисунке 7.9.

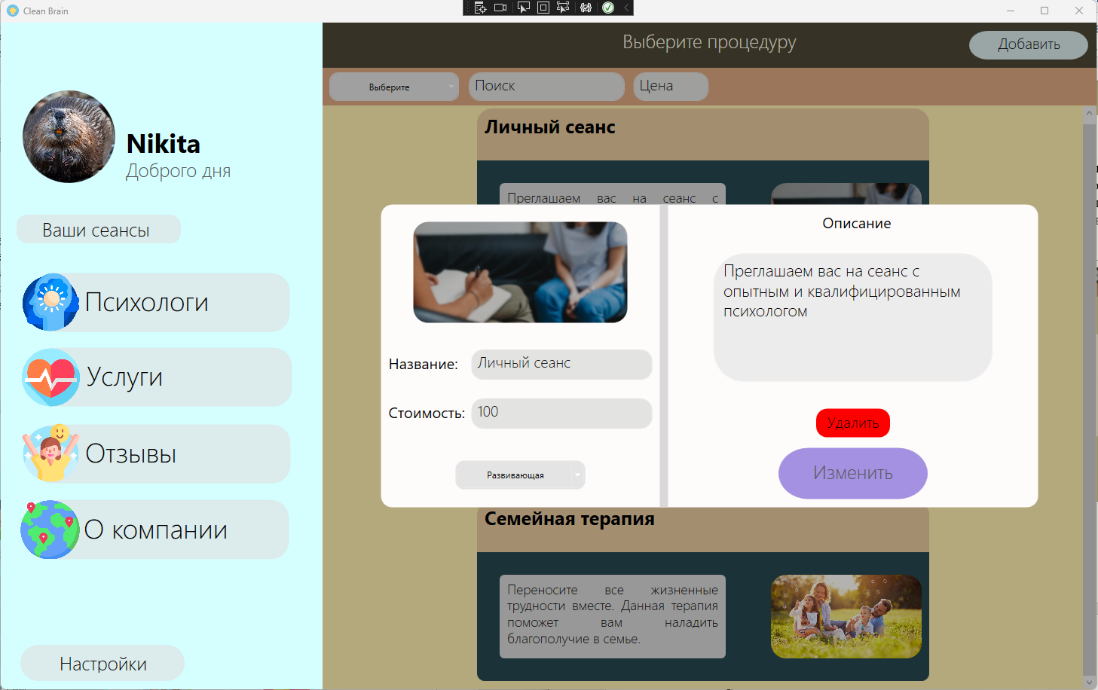


Рисунок 7.9 - Форма для изменения услуги

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

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.

Страница с отзывами представлена на рисунке 7.10.

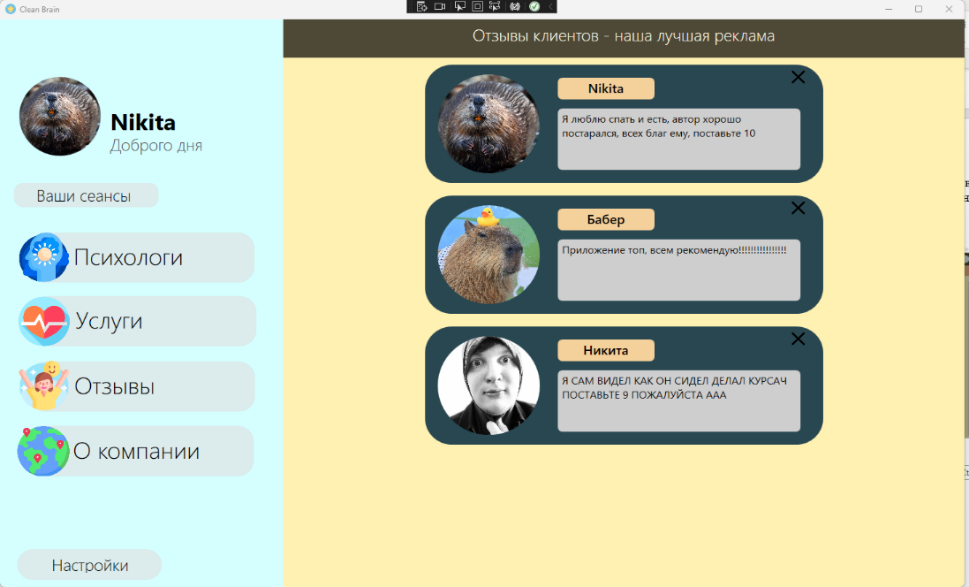


Рисунок 7.10 - Страница отзывов

## **7.2 Руководство по использованию пользователем**

После успешной авторизации, пользователя встречает приветствующая страница с навигационным меню. Данное меню обеспечивает пользователю быстрый доступ к различным функциям приложения, позволяя ему легко перемещаться по различным страницам, что представлено на рисунке 7.11.

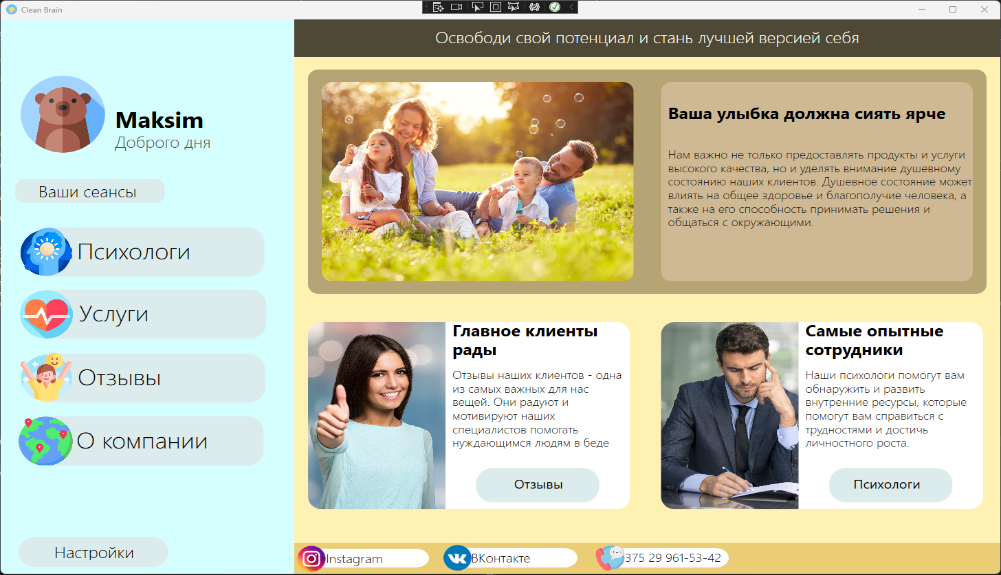


Рисунок 7.11 - Приветствующая страница

Если пользователь решит нажать на изображение аккаунта, то ему откроется новая страница с возможностью изменения личных данных, а также изображения профиля.

Если пользователь захочет выйти из данной страницы, он может кликнуть на тёмный фон.

Данная страница представлена на рисунке 7.12.

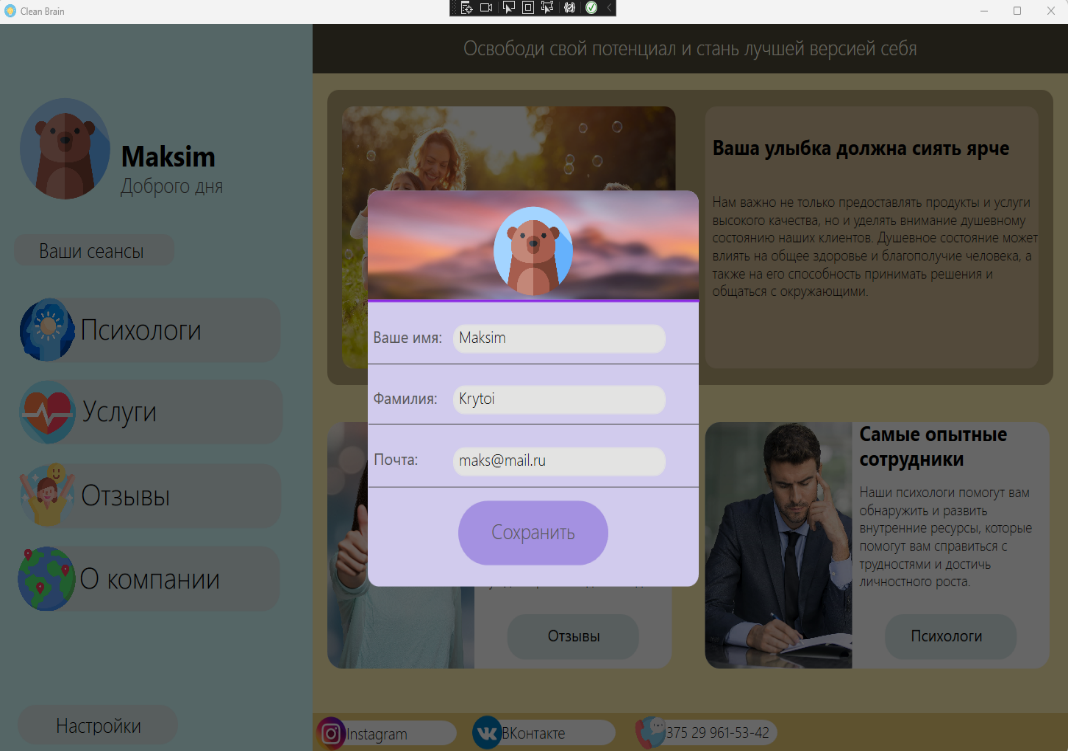


Рисунок 7.12 - Профиль пользователя

Нажав на кнопку “Психологи” мы переходим к общему перечню психологов, которых мы можем отсортировать по специализации. Как видно, кнопки для администратора скрылись. Представлено на рисунке 7.13.

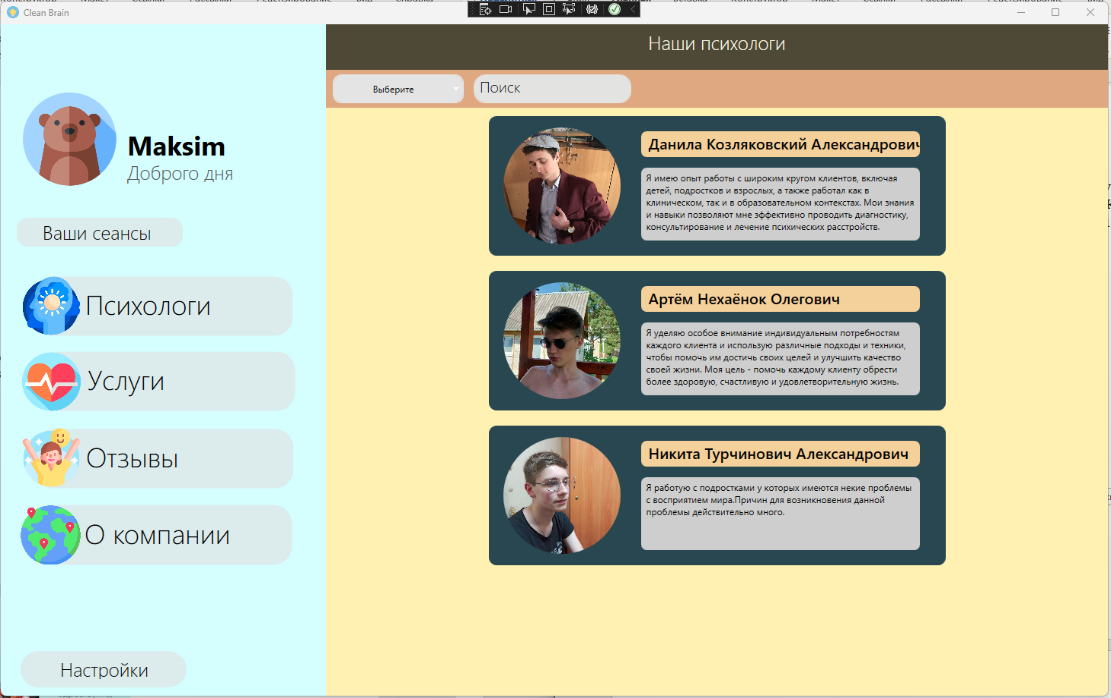


Рисунок 7.13 - Страница с перечнем психологов

При клике на изображение психолога из списка, пользователю открывается окно с заполненными данными для более детального ознакомления.

Профиль психолога представлен на рисунке 7.14.

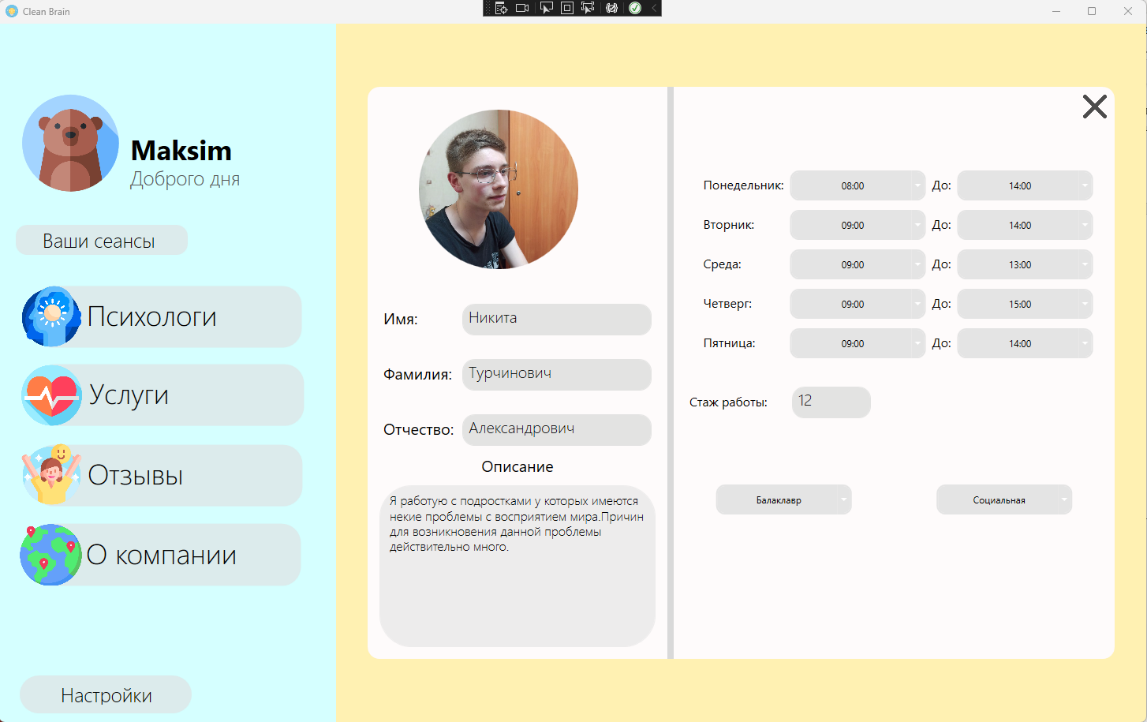


Рисунок 7.14 - Профиль психолога

Если пользователь нажмёт на карточку психолога, приложение откроет страницу с выбором услуг, которые может предоставить выбранный психолог, что представлено на рисунке 7.15.

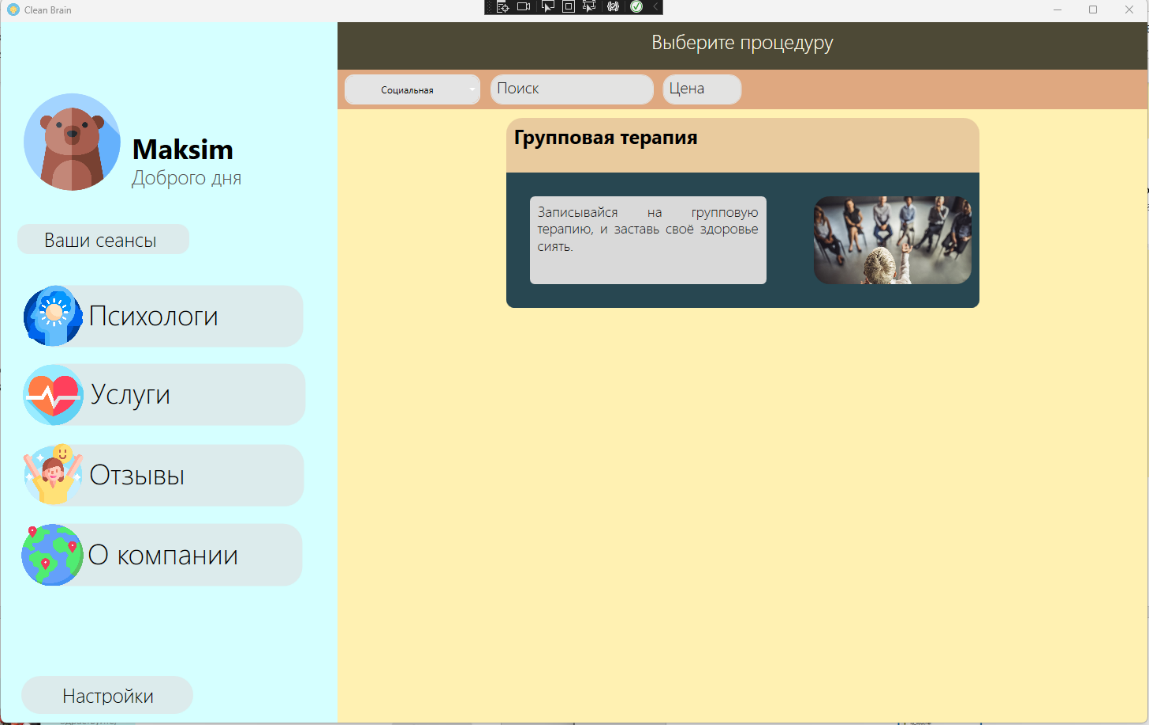


Рисунок 7.15 - Страница с услугами психолога

Далее пользователь может выбрать услугу, просто кликнув на её карточку.

После данного действия приложение предоставит форму для оформления выбранной услуги.

Данная форма представлена на рисунке 7.16.

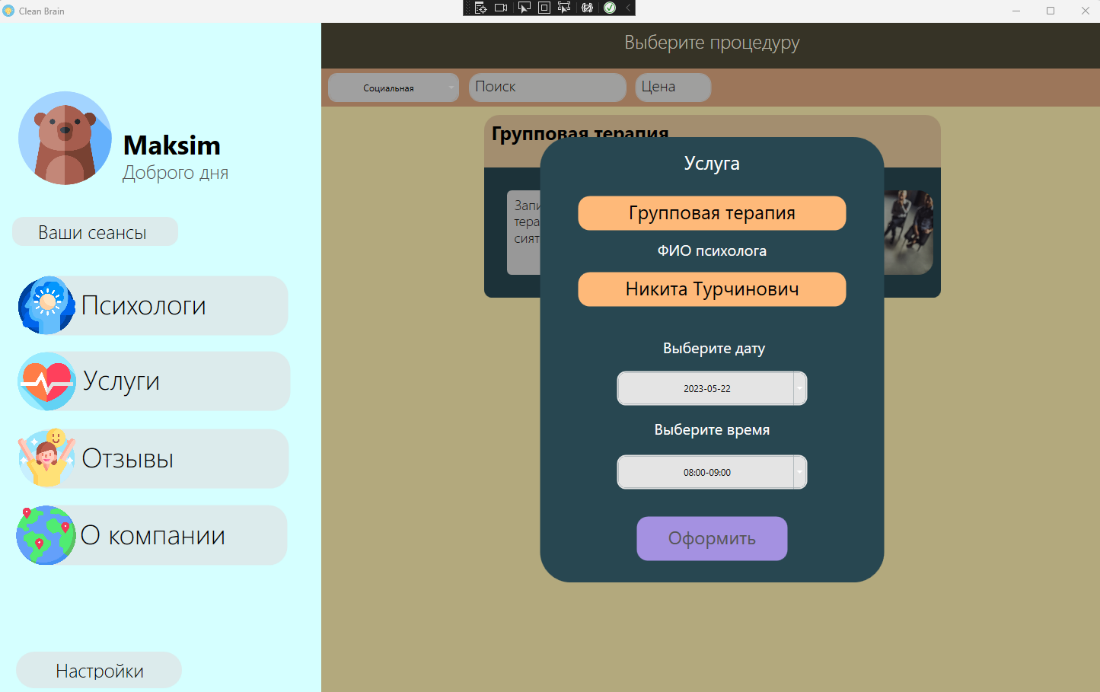


Рисунок 7.16 - Оформление выбранной услуги

После успешного оформления услуги пользователь, может просмотреть все текущие услуги просто нажав на кнопку “Ваши сеансы”. Пользователь перейдёт на список текущих заказанных услуг с возможностью их удаления, что представлено на рисунке 7.17.

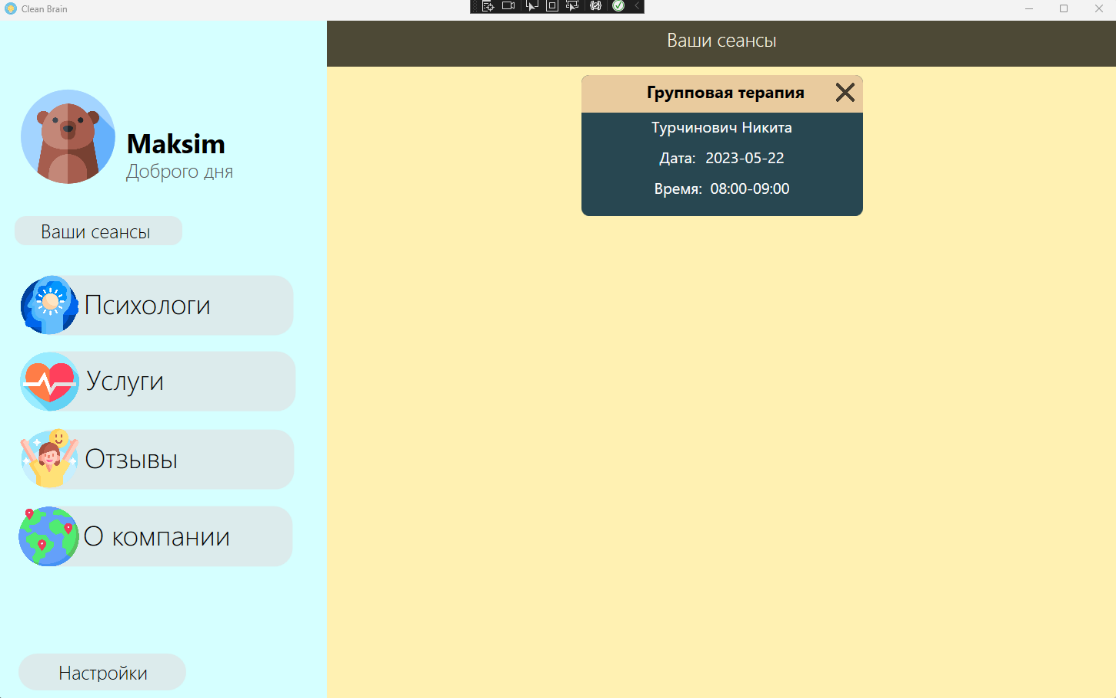


Рисунок 7.17 - Страница с текущими сеансами

Пользователь также имеет возможность нажать на кнопку "Отзывы", которая предоставит ему доступ к отзывам других пользователей. Это позволит пользователю получить дополнительную информацию и оценить качество предоставляемых услуг.

Старица с отзывами от клиентов представлена на рисунке 7.18.

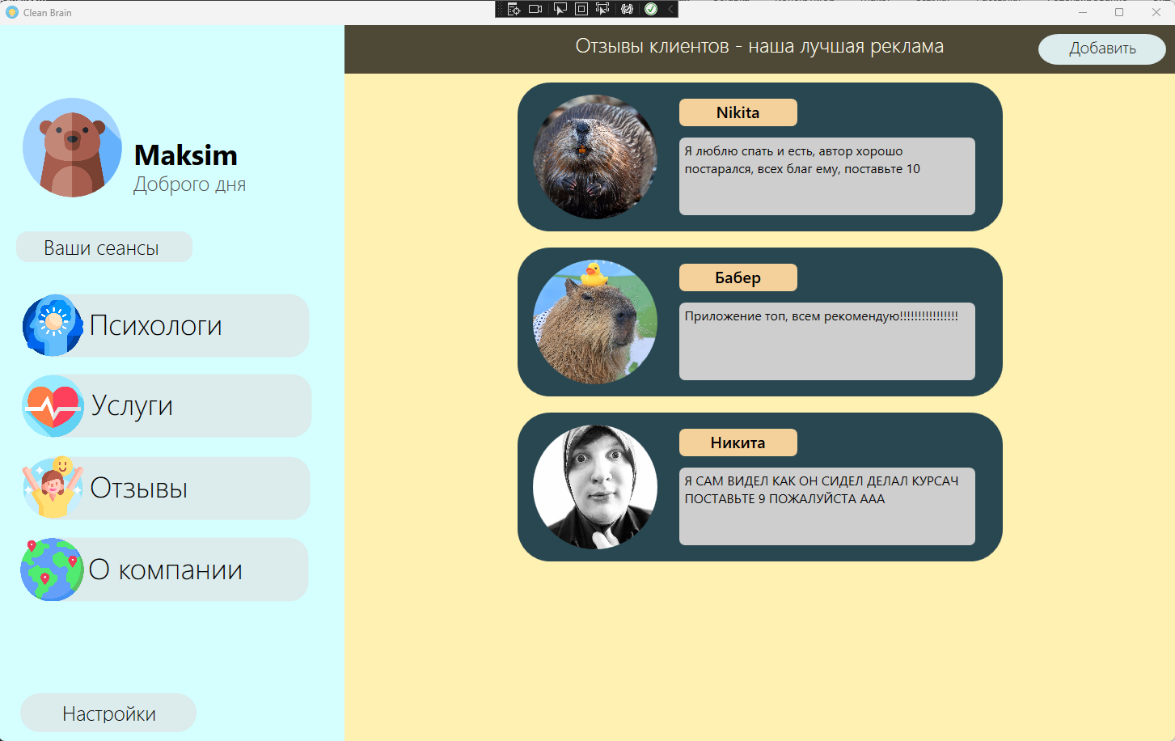


Рисунок 7.18 - Страница с отзывами

На данной странице пользователь может нажать на кнопку “добавить” и откроется форма для оформления отзыва, представленная на рисунке 7.19.

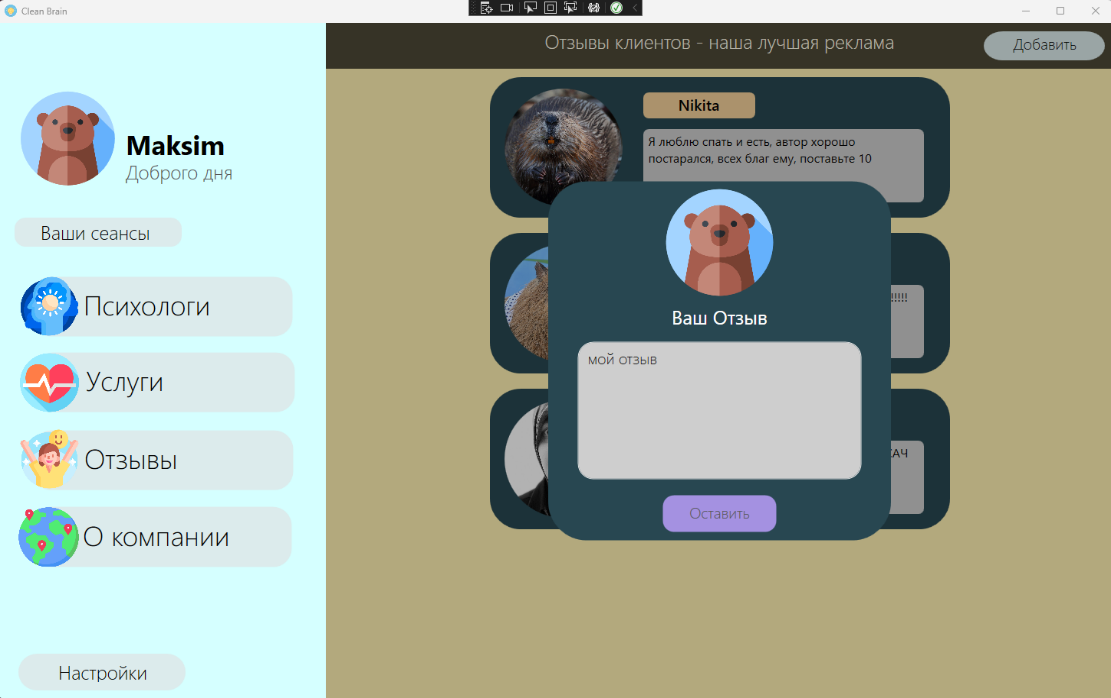


Рисунок 7.19 - Форма отзыва клиента

При заполнении формы отзыва, введенные пользователем символы проходят через процесс проверки. Это позволяет предотвратить ввод недопустимых символов.

## **7.3 Руководство по использованию гостем**

Также как и администратор, гость попадает на приветствующее окно, после чего он может выбрать интересующие страницы приложения, путям нажатия на соответствующие кнопки в навигационном меню.

Приветствующая страница гостя представлена на рисунке 7.20.

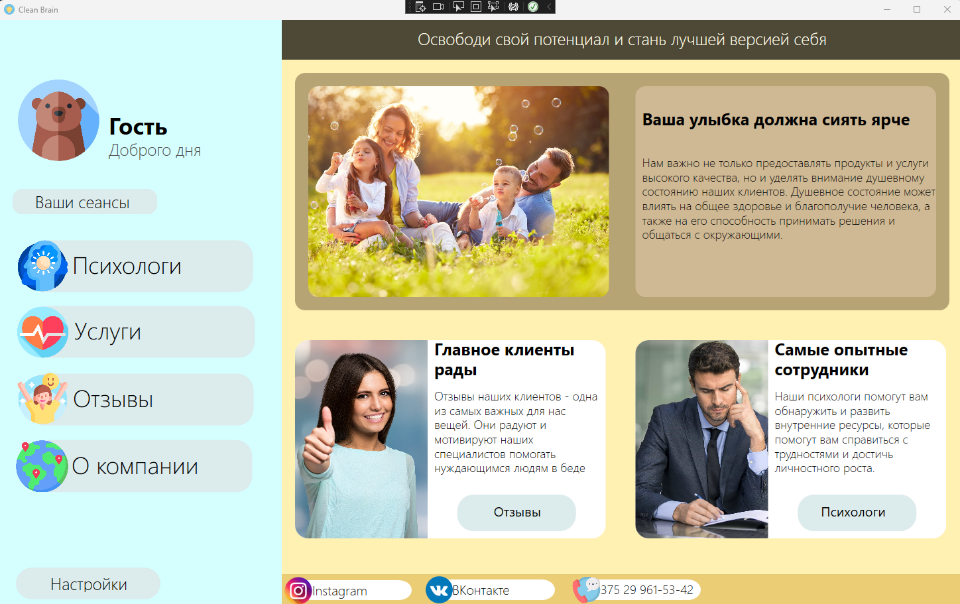


Рисунок 7.20 - Приветствующая страница

Далее гость может пройти к следующему пункту меню - "О компании". Нажав на этот пункт, гость получит доступ к информации о компании, включая ее ценности, миссию и другие важные сведения. Информация позволит гостю получить более полное представление о компании.

Данная страница представлена на рисунке 7.21.

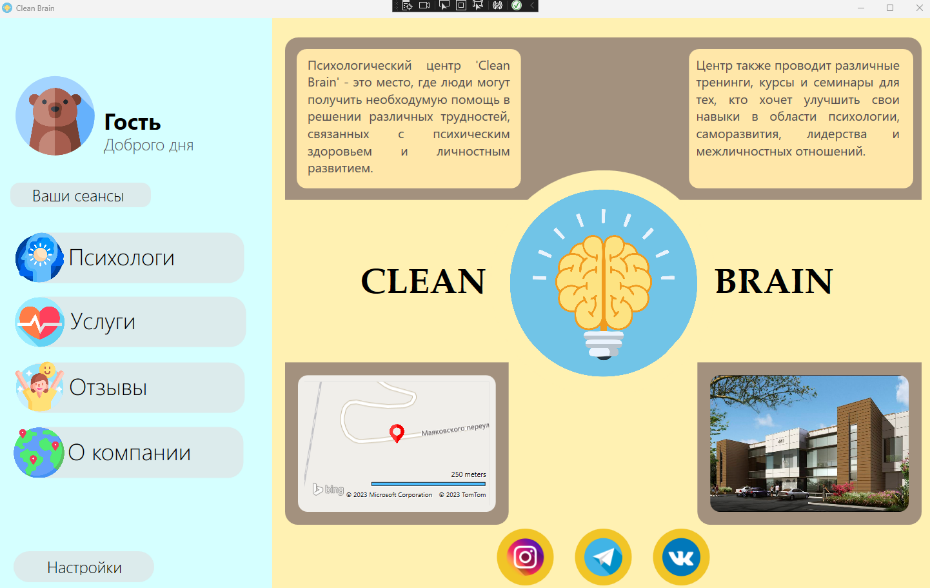


Рисунок 7.21 - Страница о компании

Гость также как и авторизированный пользователь, может просматривать список психологов, услуг, а также их страницы. Но при попытке совершить заказ, изменить профиль и нажать на кнопку “Ваши сеансы”, гость получится сообщение призывающее его зарегистрироваться.

Данное сообщение представлено на рисунке 7.22.

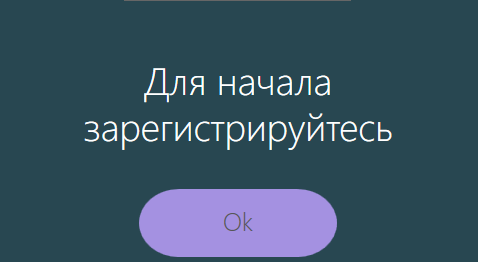


Рисунок 7.22 - Сообщение для гостя

Как можно заметить гость имеет ограниченный функционал, созданный только в ознакомительных целях.

# **Заключение**

В рамках выполнения моего курсового проекта была разработана и внедрена база данных для психологического центра на платформе Oracle. В этой базе данных я создал различные объекты, такие как таблицы для хранения информации о клиентах, психологах, процедурах, а также реализовал ограничения целостности, пользователей, триггеры и хранимые процедуры.

Основное внимание было уделено технологии репликации данных между серверами СУБД с использованием триггеров. Этот механизм обеспечивает автоматическую синхронизацию данных при их изменении, обеспечивая актуальность и согласованность в распределенной среде.

Также в проекте был реализован механизм импорта данных из JSON файлов для эффективного заполнения таблиц. При разработке базы данных я уделял особое внимание оптимизации ее производительности. Проводил тестирование на больших объемах данных (более 100 000 строк) и анализировал структуру базы данных с целью оптимизации ее работы.

Также при выполнении данного курсового проекта было разработано программное средство “Clean Brain”.

В программном средстве были реализованы следующие функции:

* регистрация пользователя в системе;
* вход пользователя в систему;
* вход гостя в систему;
* просмотр данных об психологов, услугах, отзывах и текущих сеансов;
* добавление данных о психологах и услугах;
* удаление данных о психологах и услугах;
* редактирование страниц психологов и услуг;
* бронирование талонов;
* добавление отзывов;
* регистрация новых талонов;
* изменение информации в личном кабинете;
* изменение языка приложения;
* изменение темы приложения.

В результате успешной реализации проекта была создана база данных, обеспечивающая управление всеми ключевыми аспектами психологического центра, а также приложение, позволяющая удобно бронировать сеансы с личным психологом.

# **Cписок используемых источников**

1. Язык C# и .NET Framework [Электронный ресурс] – http://professorweb.ru/my/csharp/charp\_theory/level1/infonet.php.
2. Руководство по ADO.NET и работе с базами данных [Электронный ресурс] – https://metanit.com/sharp/adonet/.
3. Документация Oracle [Электронный ресурс] / Foundation, Inc. https://docs.oracle.com/cd/B28359\_01/server.111/b31222/toc.htm
4. Официальный сайт Oracle [Электронный ресурс] / Foundation, Inc. https://www.oracle.com/database/database-vault/index.html
5. Продукты Oracle [Электронный ресурс] [Электронный ресурс] / Foundation, Inc. http://www.interface.ru/home.asp?artId=24678
6. Документы Oracle [Электронный ресурс] / Foundation, Inc. http://www.oracle.com/technetwork/database/security/database-vault-ds-12c-1898877.pdf

# **Приложение А**

CREATE OR REPLACE PACKAGE BODY GuestPackage AS

PROCEDURE GetClientById(

p\_Id\_Client IN NUMBER,

p\_Cursor OUT SYS\_REFCURSOR

)

AS

BEGIN

OPEN p\_Cursor FOR

SELECT \*

FROM Client

WHERE Id\_client = p\_Id\_Client;

END GetClientById;

PROCEDURE InsertNewClient(

p\_Name\_Client IN VARCHAR2,

p\_Surname\_Client IN VARCHAR2,

p\_Login\_Client IN VARCHAR2,

p\_Password\_Client IN VARCHAR2,

p\_Photo\_Client IN BLOB,

p\_Mail\_Client IN VARCHAR2,

p\_Client\_Id OUT NUMBER

)

IS

v\_Valid BOOLEAN := ValidateClientData(p\_Name\_Client, p\_Surname\_Client, p\_Login\_Client, p\_Password\_Client, p\_Photo\_Client, p\_Mail\_Client);

BEGIN

IF v\_Valid THEN

INSERT INTO Client (Name\_Client, Surname\_Client, Login\_Client, Password\_Client, Photo\_Client, Mail\_Client)

VALUES (p\_Name\_Client, p\_Surname\_Client, p\_Login\_Client, p\_Password\_Client, p\_Photo\_Client, p\_Mail\_Client);

COMMIT;

SELECT Id\_client

INTO p\_Client\_Id FROM Client

WHERE Login\_Client = p\_Login\_Client;

ELSE

p\_Client\_Id := -1;

END IF;

EXCEPTION

WHEN OTHERS THEN

rollback;

p\_Client\_Id := -2;

raise;

END InsertNewClient;

PROCEDURE AuthenticateClient(

p\_Login\_Client IN VARCHAR2,

p\_Password\_Client IN VARCHAR2,

p\_AuthenticationResult OUT NUMBER

)

IS

v\_ValidData BOOLEAN := ValidateAuthenticationData(p\_Login\_Client, p\_Password\_Client);

v\_Client\_Count NUMBER;

v\_Client\_id NUMBER;

pass Client.Password\_Client%Type;

BEGIN

pass := EncryptPassword(p\_Password\_Client);

IF v\_ValidData THEN

SELECT COUNT(\*)

INTO v\_Client\_Count

FROM Client

WHERE Login\_Client = p\_Login\_Client AND Password\_Client = pass;

IF v\_Client\_Count > 0 THEN

SELECT Id\_client

INTO v\_Client\_id

FROM Client

WHERE Login\_Client = p\_Login\_Client AND Password\_Client = pass;

p\_AuthenticationResult := v\_Client\_id;

ELSE

p\_AuthenticationResult := -5;

END IF;

ELSE

p\_AuthenticationResult := -20;

END IF;

EXCEPTION

WHEN OTHERS THEN

rollback;

p\_AuthenticationResult := -1;

END AuthenticateClient;

PROCEDURE UpdateClientInfo(

p\_Id\_Client IN NUMBER,

p\_Name\_Client IN VARCHAR2,

p\_Surname\_Client IN VARCHAR2,

p\_Photo\_Client IN BLOB,

p\_Mail\_Client IN VARCHAR2,

p\_Result OUT NUMBER

)

IS

v\_Valid BOOLEAN := ValidateClientUpdateData(p\_Name\_Client, p\_Surname\_Client, p\_Photo\_Client, p\_Mail\_Client);

BEGIN

IF v\_Valid THEN

UPDATE Client

SET

Name\_Client = p\_Name\_Client,

Surname\_Client = p\_Surname\_Client,

Photo\_Client = p\_Photo\_Client,

Mail\_Client = p\_Mail\_Client

WHERE Id\_Client = p\_Id\_Client;

COMMIT;

p\_Result := 1;

ELSE

p\_Result := 0;

END IF;

EXCEPTION

WHEN OTHERS THEN

rollback;

p\_Result := 0;

END UpdateClientInfo;

PROCEDURE GetAllClients(

p\_Cursor OUT SYS\_REFCURSOR

)

IS

BEGIN

OPEN p\_Cursor FOR

SELECT \*

FROM Client;

END GetAllClients;

PROCEDURE UpdateReview(

p\_Review\_Id IN NUMBER,

p\_Photo\_Review IN BLOB,

p\_Name\_Client IN VARCHAR2,

p\_Review\_Text IN VARCHAR2,

p\_Result OUT NUMBER

)

IS

v\_Valid BOOLEAN := ValidateReviewData(p\_Review\_Text);

BEGIN

IF v\_Valid THEN

UPDATE Review

SET

Photo\_Review = p\_Photo\_Review,

Name\_Client = p\_Name\_Client,

Review = p\_Review\_Text

WHERE Id\_Review = p\_Review\_Id;

IF SQL%ROWCOUNT > 0 THEN

p\_Result := 1;

COMMIT;

ELSE

p\_Result := 0;

END IF;

ELSE

p\_Result := 0;

END IF;

EXCEPTION

WHEN OTHERS THEN

rollback;

p\_Result := 0;

END UpdateReview;

PROCEDURE GetAllReviews(

p\_Cursor OUT SYS\_REFCURSOR

)

IS

BEGIN

OPEN p\_Cursor FOR

SELECT \*

FROM Review;

END GetAllReviews;

PROCEDURE GetSpecializationByName(

p\_SpecializationName IN VARCHAR2,

p\_Cursor OUT SYS\_REFCURSOR

)

IS

BEGIN

OPEN p\_Cursor FOR

SELECT \*

FROM Specialization

WHERE Spezialization\_Name = p\_SpecializationName;

END GetSpecializationByName;

PROCEDURE GetDegreeByName(

p\_Degree\_Name IN VARCHAR2, p\_Cursor OUT SYS\_REFCURSOR

)

IS

BEGIN

OPEN p\_Cursor FOR

SELECT \*

FROM Academic\_Degree

WHERE Academic\_Name = p\_Degree\_Name;

END GetDegreeByName;

PROCEDURE GetProcedureById(

p\_Procedure\_Id IN NUMBER,

p\_Cursor OUT SYS\_REFCURSOR

)

IS

BEGIN

OPEN p\_Cursor FOR

SELECT \*

FROM Procedures

WHERE Id\_Procedure = p\_Procedure\_Id;

END GetProcedureById;

PROCEDURE GetAllProcedures(

p\_Cursor OUT SYS\_REFCURSOR

)

IS

BEGIN

OPEN p\_Cursor FOR

SELECT \*

FROM Procedures;

END GetAllProcedures;

PROCEDURE GetPsychologistById(

p\_Id\_Psychologist IN NUMBER,

p\_Cursor OUT SYS\_REFCURSOR

)

AS

BEGIN

OPEN p\_Cursor FOR

SELECT \*

FROM Psychologist

WHERE Id\_Psychologist = p\_Id\_Psychologist;

END GetPsychologistById;

PROCEDURE GetAllPsychologists(

p\_Cursor OUT SYS\_REFCURSOR

)

AS

BEGIN

OPEN p\_Cursor FOR

SELECT \*

FROM Psychologist;

END GetAllPsychologists;

END GuestPackage;

# **Приложение Б**

CREATE OR REPLACE PACKAGE BODY ClientPackage AS

PROCEDURE GetClientById(

p\_Id\_Client IN NUMBER,

p\_Cursor OUT SYS\_REFCURSOR

)

AS

BEGIN

OPEN p\_Cursor FOR

SELECT \*

FROM Client

WHERE Id\_client = p\_Id\_Client;

END GetClientById;

PROCEDURE InsertNewClient(

p\_Name\_Client IN VARCHAR2,

p\_Surname\_Client IN VARCHAR2,

p\_Login\_Client IN VARCHAR2,

p\_Password\_Client IN VARCHAR2,

p\_Photo\_Client IN BLOB,

p\_Mail\_Client IN VARCHAR2,

p\_Client\_Id OUT NUMBER

)

IS

v\_Valid BOOLEAN := ValidateClientData(p\_Name\_Client, p\_Surname\_Client, p\_Login\_Client, p\_Password\_Client, p\_Photo\_Client, p\_Mail\_Client);

BEGIN

-- ,

IF v\_Valid THEN

INSERT INTO Client (Name\_Client, Surname\_Client, Login\_Client, Password\_Client, Photo\_Client, Mail\_Client)

VALUES (p\_Name\_Client, p\_Surname\_Client, p\_Login\_Client, p\_Password\_Client, p\_Photo\_Client, p\_Mail\_Client);

COMMIT; --

SELECT Id\_client

INTO p\_Client\_Id FROM Client

WHERE Login\_Client = p\_Login\_Client;

ELSE

p\_Client\_Id := -1; --

END IF;

EXCEPTION

WHEN OTHERS THEN

rollback;

p\_Client\_Id := -1;

END InsertNewClient;

PROCEDURE AuthenticateClient(

p\_Login\_Client IN VARCHAR2,

p\_Password\_Client IN VARCHAR2,

p\_AuthenticationResult OUT NUMBER

)

IS

v\_ValidData BOOLEAN := ValidateAuthenticationData(p\_Login\_Client, p\_Password\_Client);

v\_Client\_Count NUMBER;

v\_Client\_id NUMBER;

pass Client.Password\_Client%Type;

BEGIN

--

pass := EncryptPassword(p\_Password\_Client);

IF v\_ValidData THEN

SELECT COUNT(\*)

INTO v\_Client\_Count

FROM Client

WHERE Login\_Client = p\_Login\_Client AND Password\_Client = pass;

IF v\_Client\_Count > 0 THEN

SELECT Id\_client

INTO v\_Client\_id

FROM Client

WHERE Login\_Client = p\_Login\_Client AND Password\_Client = pass;

p\_AuthenticationResult := v\_Client\_id; --

ELSE

p\_AuthenticationResult := -5; --

END IF;

ELSE

p\_AuthenticationResult := -20; --

END IF;

EXCEPTION

WHEN OTHERS THEN

rollback;

p\_AuthenticationResult := -1; --

END AuthenticateClient;

PROCEDURE UpdateClientInfo(

p\_Id\_Client IN NUMBER,

p\_Name\_Client IN VARCHAR2,

p\_Surname\_Client IN VARCHAR2,

p\_Photo\_Client IN BLOB,

p\_Mail\_Client IN VARCHAR2,

p\_Result OUT NUMBER

)

IS

v\_Valid BOOLEAN := ValidateClientUpdateData(p\_Name\_Client, p\_Surname\_Client, p\_Photo\_Client, p\_Mail\_Client);

BEGIN

--

IF v\_Valid THEN

--

UPDATE Client

SET

Name\_Client = p\_Name\_Client,

Surname\_Client = p\_Surname\_Client,

Photo\_Client = p\_Photo\_Client,

Mail\_Client = p\_Mail\_Client

WHERE Id\_Client = p\_Id\_Client;

COMMIT; --

p\_Result := 1; --

ELSE

p\_Result := 0; -- ( , Id\_Client )

END IF;

EXCEPTION

WHEN OTHERS THEN

rollback;

p\_Result := 0; --

END UpdateClientInfo;

PROCEDURE GetAllClients(

p\_Cursor OUT SYS\_REFCURSOR

)

IS

BEGIN

OPEN p\_Cursor FOR

SELECT \*

FROM Client;

END GetAllClients;

PROCEDURE CreateReview(

p\_Photo\_Review in BLOB,

p\_Name\_Client IN VARCHAR2,

p\_Id\_Client IN NUMBER,

p\_Review\_Text IN VARCHAR2,

p\_Result OUT NUMBER

)

IS

v\_Valid BOOLEAN := ValidateReviewData(p\_Review\_Text);

v\_Client\_Count NUMBER;

v\_Previous\_Review\_Count NUMBER;

BEGIN

--

IF v\_Valid THEN

-- Id\_Client

SELECT COUNT(\*)

INTO v\_Client\_Count

FROM Client

WHERE Id\_Client = p\_Id\_Client;

IF v\_Client\_Count > 0 THEN

SELECT COUNT(\*)

INTO v\_Previous\_Review\_Count

FROM Review

WHERE Id\_Client = p\_Id\_Client;

IF v\_Previous\_Review\_Count = 0 THEN

-- Review

INSERT INTO Review (Photo\_Review, Name\_Client, Id\_Client,Review)

VALUES (p\_Photo\_Review, p\_Name\_Client, p\_Id\_Client, p\_Review\_Text);

COMMIT; --

p\_Result := 1; --

ELSE

p\_Result := 0;

END IF;

ELSE

p\_Result := 0; -- Id\_Client

END IF;

ELSE

p\_Result := 0; --

END IF;

EXCEPTION

WHEN OTHERS THEN

rollback;

p\_Result := 0; --

END CreateReview;

PROCEDURE DeleteReviewById(

p\_Id IN NUMBER,

p\_Success OUT NUMBER

)

IS

BEGIN

p\_Success := 0; --

DELETE FROM Review

WHERE Id\_Review = p\_Id;

IF SQL%ROWCOUNT > 0 THEN

p\_Success := 1; --

COMMIT; --

END IF;

END DeleteReviewById;

PROCEDURE UpdateReview(

p\_Review\_Id IN NUMBER,

p\_Photo\_Review IN BLOB,

p\_Name\_Client IN VARCHAR2,

p\_Review\_Text IN VARCHAR2,

p\_Result OUT NUMBER

)

IS

v\_Valid BOOLEAN := ValidateReviewData(p\_Review\_Text);

BEGIN

--

IF v\_Valid THEN

--

UPDATE Review

SET

Photo\_Review = p\_Photo\_Review,

Name\_Client = p\_Name\_Client,

Review = p\_Review\_Text

WHERE Id\_Review = p\_Review\_Id;

--

IF SQL%ROWCOUNT > 0 THEN

p\_Result := 1; --

COMMIT; --

ELSE

p\_Result := 0; -- ( , Id\_Review )

END IF;

ELSE

p\_Result := 0; --

END IF;

EXCEPTION

WHEN OTHERS THEN

rollback;

p\_Result := 0; --

END UpdateReview;

PROCEDURE GetAllReviews(

p\_Cursor OUT SYS\_REFCURSOR

)

IS

BEGIN

OPEN p\_Cursor FOR

SELECT \*

FROM Review;

END GetAllReviews;

END ClientPackage;

/

# **Приложение В**

create or replace package body cwAdmin

CREATE OR REPLACE PACKAGE BODY AdminPackage AS

PROCEDURE GetClientById(

p\_Id\_Client IN NUMBER,

p\_Cursor OUT SYS\_REFCURSOR

)

AS

BEGIN

OPEN p\_Cursor FOR

SELECT \*

FROM Client

WHERE Id\_client = p\_Id\_Client;

END GetClientById;

PROCEDURE InsertNewClient(

p\_Name\_Client IN VARCHAR2,

p\_Surname\_Client IN VARCHAR2,

p\_Login\_Client IN VARCHAR2,

p\_Password\_Client IN VARCHAR2,

p\_Photo\_Client IN BLOB,

p\_Mail\_Client IN VARCHAR2,

p\_Client\_Id OUT NUMBER

)

IS

v\_Valid BOOLEAN := ValidateClientData(p\_Name\_Client, p\_Surname\_Client, p\_Login\_Client, p\_Password\_Client, p\_Photo\_Client, p\_Mail\_Client);

BEGIN

-- ,

IF v\_Valid THEN

INSERT INTO Client (Name\_Client, Surname\_Client, Login\_Client, Password\_Client, Photo\_Client, Mail\_Client)

VALUES (p\_Name\_Client, p\_Surname\_Client, p\_Login\_Client, p\_Password\_Client, p\_Photo\_Client, p\_Mail\_Client);

COMMIT; --

SELECT Id\_client

INTO p\_Client\_Id FROM Client

WHERE Login\_Client = p\_Login\_Client;

ELSE

p\_Client\_Id := -1; --

END IF;

EXCEPTION

WHEN OTHERS THEN

rollback;

p\_Client\_Id := -1;

END InsertNewClient;

PROCEDURE AuthenticateClient(

p\_Login\_Client IN VARCHAR2,

p\_Password\_Client IN VARCHAR2,

p\_AuthenticationResult OUT NUMBER

)

IS

v\_ValidData BOOLEAN := ValidateAuthenticationData(p\_Login\_Client, p\_Password\_Client);

v\_Client\_Count NUMBER;

v\_Client\_id NUMBER;

pass Client.Password\_Client%Type;

BEGIN

--

pass := EncryptPassword(p\_Password\_Client);

IF v\_ValidData THEN

SELECT COUNT(\*)

INTO v\_Client\_Count

FROM Client

WHERE Login\_Client = p\_Login\_Client AND Password\_Client = pass;

IF v\_Client\_Count > 0 THEN

SELECT Id\_client

INTO v\_Client\_id

FROM Client

WHERE Login\_Client = p\_Login\_Client AND Password\_Client = pass;

p\_AuthenticationResult := v\_Client\_id; --

ELSE

p\_AuthenticationResult := -5; --

END IF;

ELSE

p\_AuthenticationResult := -20; --

END IF;

EXCEPTION

WHEN OTHERS THEN

p\_AuthenticationResult := -1; --

END AuthenticateClient;

PROCEDURE UpdateClientInfo(

p\_Id\_Client IN NUMBER,

p\_Name\_Client IN VARCHAR2,

p\_Surname\_Client IN VARCHAR2,

p\_Photo\_Client IN BLOB,

p\_Mail\_Client IN VARCHAR2,

p\_Result OUT NUMBER

)

IS

v\_Valid BOOLEAN := ValidateClientUpdateData(p\_Name\_Client, p\_Surname\_Client, p\_Photo\_Client, p\_Mail\_Client);

BEGIN

--

IF v\_Valid THEN

--

UPDATE Client

SET

Name\_Client = p\_Name\_Client,

Surname\_Client = p\_Surname\_Client,

Photo\_Client = p\_Photo\_Client,

Mail\_Client = p\_Mail\_Client

WHERE Id\_Client = p\_Id\_Client;

COMMIT; --

p\_Result := 1; --

ELSE

p\_Result := 0; -- ( , Id\_Client )

END IF;

EXCEPTION

WHEN OTHERS THEN

rollback;

p\_Result := 0; --

END UpdateClientInfo;

PROCEDURE GetAllClients(

p\_Cursor OUT SYS\_REFCURSOR

)

IS

BEGIN

OPEN p\_Cursor FOR

SELECT \*

FROM Client;

END GetAllClients;

--Review

PROCEDURE CreateReview(

p\_Photo\_Review in BLOB,

p\_Name\_Client IN VARCHAR2,

p\_Id\_Client IN NUMBER,

p\_Review\_Text IN VARCHAR2,

p\_Result OUT NUMBER

)

IS

v\_Valid BOOLEAN := ValidateReviewData(p\_Review\_Text);

v\_Client\_Count NUMBER;

v\_Previous\_Review\_Count NUMBER;

BEGIN

--

IF v\_Valid THEN

-- Id\_Client

SELECT COUNT(\*)

INTO v\_Client\_Count

FROM Client

WHERE Id\_Client = p\_Id\_Client;

IF v\_Client\_Count > 0 THEN

SELECT COUNT(\*)

INTO v\_Previous\_Review\_Count

FROM Review

WHERE Id\_Client = p\_Id\_Client;

IF v\_Previous\_Review\_Count = 0 THEN

-- Review

INSERT INTO Review (Photo\_Review, Name\_Client, Id\_Client, Review)

VALUES (p\_Photo\_Review, p\_Name\_Client, p\_Id\_Client, p\_Review\_Text);

COMMIT; --

p\_Result := 1; --

ELSE

p\_Result := 0;

END IF;

ELSE

p\_Result := 0; -- Id\_Client

END IF;

ELSE

p\_Result := 0; --

END IF;

EXCEPTION

WHEN OTHERS THEN

rollback;

p\_Result := 0; --

END CreateReview;

PROCEDURE DeleteReviewById(

p\_Id IN NUMBER,

p\_Success OUT NUMBER

)

IS

BEGIN

p\_Success := 0; --

DELETE FROM Review

WHERE Id\_Review = p\_Id;

IF SQL%ROWCOUNT > 0 THEN

p\_Success := 1; --

COMMIT; --

END IF;

END DeleteReviewById;

PROCEDURE UpdateReview(

p\_Review\_Id IN NUMBER,

p\_Photo\_Review IN BLOB,

p\_Name\_Client IN VARCHAR2,

p\_Review\_Text IN VARCHAR2,

p\_Result OUT NUMBER

)

IS

v\_Valid BOOLEAN := ValidateReviewData(p\_Review\_Text);

BEGIN

--

IF v\_Valid THEN

--

UPDATE Review

SET

Photo\_Review = p\_Photo\_Review,

Name\_Client = p\_Name\_Client,

Review = p\_Review\_Text

WHERE Id\_Review = p\_Review\_Id;

--

IF SQL%ROWCOUNT > 0 THEN

p\_Result := 1; --

COMMIT; --

ELSE

p\_Result := 0; -- ( , Id\_Review )

END IF;

ELSE

p\_Result := 0; --

END IF;

EXCEPTION

WHEN OTHERS THEN

rollback;

p\_Result := 0; --

END UpdateReview;

PROCEDURE GetAllReviews(

p\_Cursor OUT SYS\_REFCURSOR

)

IS

BEGIN

OPEN p\_Cursor FOR

SELECT \*

FROM Review;

END GetAllReviews;

-- Timetable

PROCEDURE GetSpecializationByName(

p\_SpecializationName IN VARCHAR2,

p\_Cursor OUT SYS\_REFCURSOR

)

IS

BEGIN

OPEN p\_Cursor FOR

SELECT \*

FROM Specialization

WHERE Spezialization\_Name = p\_SpecializationName;

END GetSpecializationByName;

--degree

PROCEDURE AddDegree(

p\_Degree\_Name IN VARCHAR2, p\_Result OUT NUMBER

)

IS

v\_Valid BOOLEAN := ValidateDegreeData(p\_Degree\_Name);

BEGIN

--

IF v\_Valid THEN

INSERT INTO Academic\_Degree (Academic\_Name)

VALUES (p\_Degree\_Name);

COMMIT; --

p\_Result := 1; --

ELSE

p\_Result := 0;

END IF;

EXCEPTION

WHEN OTHERS THEN

rollback;

p\_Result := 0; --

END AddDegree;

PROCEDURE AddPsychologist(

p\_Name\_Psychologist IN NVARCHAR2,

p\_Surname\_Psychologist IN NVARCHAR2,

p\_Patronymic\_Psychologist IN NVARCHAR2,

p\_Experience IN NUMBER,

p\_Photo\_Psychologist IN BLOB,

p\_Description IN NVARCHAR2,

p\_Specialization\_Psychologist IN NVARCHAR2,

p\_Degree IN NVARCHAR2,

p\_Result OUT NUMBER

)

AS

v\_Valid BOOLEAN := ValidatePsychologistData(

p\_Name\_Psychologist,

p\_Surname\_Psychologist,

p\_Patronymic\_Psychologist,

p\_Description

);

BEGIN

--

IF v\_Valid THEN

INSERT INTO Psychologist (

Name\_Psychologist,

Surname\_Psychologist,

Patronymic\_Psychologist,

Experience,

Photo\_Psychologist,

Description,

Spezialization\_Psychologist,

Degree

) VALUES (

p\_Name\_Psychologist,

p\_Surname\_Psychologist,

p\_Patronymic\_Psychologist,

p\_Experience,

p\_Photo\_Psychologist,

p\_Description,

p\_Specialization\_Psychologist,

p\_Degree

);

commit;

p\_Result := 1; --

ELSE

p\_Result := 0; --

END IF;

EXCEPTION

WHEN OTHERS THEN

rollback;

--

p\_Result := 0;

Raise;

END AddPsychologist;

PROCEDURE UpdatePsychologist(

p\_Id\_Psychologist IN NUMBER,

p\_Name\_Psychologist IN NVARCHAR2,

p\_Surname\_Psychologist IN NVARCHAR2,

p\_Patronymic\_Psychologist IN NVARCHAR2,

p\_Experience IN NUMBER,

p\_Photo\_Psychologist IN BLOB,

p\_Description IN NVARCHAR2,

p\_Spezialization\_Psychologist IN NVARCHAR2,

p\_Degree IN NVARCHAR2,

p\_Result OUT NUMBER

)

AS

v\_Valid BOOLEAN := ValidatePsychologistData(p\_Name\_Psychologist, p\_Surname\_Psychologist, p\_Patronymic\_Psychologist, p\_Description);

BEGIN

--

IF v\_Valid THEN

--

UPDATE Psychologist

SET

Name\_Psychologist = p\_Name\_Psychologist,

Surname\_Psychologist = p\_Surname\_Psychologist,

Patronymic\_Psychologist = p\_Patronymic\_Psychologist,

Experience = p\_Experience,

Photo\_Psychologist = p\_Photo\_Psychologist,

Description = p\_Description,

Spezialization\_Psychologist = p\_Spezialization\_Psychologist,

Degree = p\_Degree

WHERE Id\_Psychologist = p\_Id\_Psychologist;

COMMIT; --

p\_Result := 1; --

ELSE

p\_Result := 0; --

END IF;

EXCEPTION

WHEN OTHERS THEN

rollback;

p\_Result := 0; --

END UpdatePsychologist;

PROCEDURE DeletePsychologist(

p\_Id\_Psychologist IN NUMBER,

p\_Result OUT NUMBER

)

AS

BEGIN

DELETE FROM Psychologist WHERE Id\_Psychologist = p\_Id\_Psychologist;

COMMIT;

p\_Result := SQL%ROWCOUNT; --

EXCEPTION

WHEN OTHERS THEN

p\_Result := 0; --

END DeletePsychologist;

PROCEDURE GetPsychologistById(

p\_Id\_Psychologist IN NUMBER,

p\_Cursor OUT SYS\_REFCURSOR

)

AS

BEGIN

OPEN p\_Cursor FOR

SELECT \*

FROM Psychologist

WHERE Id\_Psychologist = p\_Id\_Psychologist;

END GetPsychologistById;

PROCEDURE GetAllPsychologists(

p\_Cursor OUT SYS\_REFCURSOR

)

AS

BEGIN

OPEN p\_Cursor FOR

SELECT \*

FROM Psychologist;

END GetAllPsychologists;

END AdminPackage;

/