МИНИСТЕРСТВО ОБРАЗОВАНИЯ КИРОВСКОЙ ОБЛАСТИ

Кировское областное государственное профессиональное образовательное

бюджетное учреждение

«Слободской колледж педагогики и социальных отношений»

**ОТЧЕТ**

**по учебной практике**

**ПМ 11. Разработка и администрирование баз данных**

**Тема: «Разработка базы данных «ХранительПРО»**

Студента:

Фокеева Никиты Петровича

Группа 22П-1

Специальность 09.02.07 Информационные системы и программирование

Руководитель практики от колледжа

\_\_\_\_\_\_/ Пентин Николай Сергеевич \_\_\_\_\_

Подпись расшифровка

2025 год

**СОДЕРЖАНИЕ**

1. Введение3

2. Анализ предметной области4

2.1 Описание системы “HranitelPRO”4

2.2. Бизнес-процессы4

2.3. Диаграмма вариантов использования 4

3. Проектирование базы данных5

3.1. Выделение сущностей и атрибутов5

3.2. ERD-диаграмма8

3.3. Нормализация базы данных 8

4. Реализация базы данных в СУБД9

4.1. Выбор СУБД9

4.2. Создание таблиц и связей9

4.3. Заполнение таблиц данными9

5. Реализация уровней доступа10

5.1. Определение ролей пользователей10

5.2. Настройка прав доступа10

6. Создание запросов и отчетов11

6.1. Примеры SQL-запросов11

6.2. Создание отчетов14

7. Резервное копирование и восстановление15

8. Заключение17

9. Приложения 18

9.1. Скрипт создания базы данных (SQL)18

9.2. Скриншоты интерфейса (авторизация, регистрация, и т.д.) 21

9.3. Ссылка на репозиторий25

**1. ВВЕДЕНИЕ**

Данный отчет описывает процесс разработки базы данных “HranitelPRO” в рамках учебной практики по ПМ 11 “Разработка и администрирование баз данных”. Целью работы является создание базы данных для системы организации пропускного режима на объектах КИИ (критической информационной инфраструктуры), включая веб-сервис для подачи заявок и взаимодействие с терминалами.

**2. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

**2.1. Описание системы “HranitelPRO”**

Система “HranitelPRO” предназначена для автоматизации процессов, связанных с организацией пропускного режима на объектах КИИ. Она включает в себя следующие компоненты:

* **Веб-сервис:** Для регистрации пользователей, подачи заявок на посещение (личных и групповых), просмотра статуса заявок.
* **Терминалы:** Для сотрудников (общего отдела, охраны, подразделений) для проверки и обработки заявок, управления доступом.
* **База данных:** Хранит информацию о пользователях, сотрудниках, подразделениях, заявках на посещение, посетителях, и т.д.

**2.2. Бизнес-процессы**

Основные бизнес-процессы, поддерживаемые системой:

* Регистрация и авторизация пользователей (включая как сотрудников, так и посетителей).
* Подача заявок на посещение (личных и групповых).
* Согласование заявок сотрудниками соответствующих подразделений.
* Утверждение заявок сотрудниками общего отдела.
* Управление пропуском на территорию сотрудниками охраны.
* Ведение журнала посещений.

**2.3. Диаграмма вариантов использования**

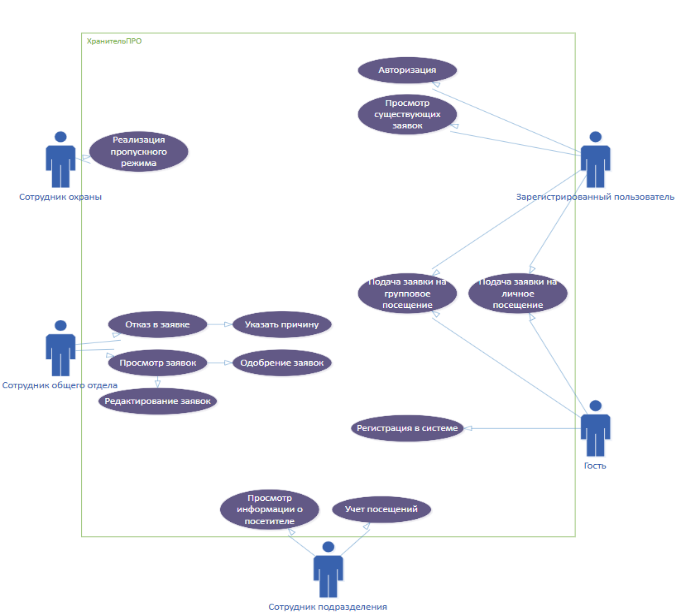


Рисунок 1 – Диаграмма вариантов использования

**3. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ**

**3.1. Выделение сущностей и атрибутов**

На основе анализа предметной области были выделены следующие сущности и их атрибуты:

* **Пользователи (Users):**
  + user\_id (INT, PRIMARY KEY, IDENTITY) - Уникальный идентификатор пользователя
  + email (VARCHAR(255), UNIQUE, NOT NULL) - Адрес электронной почты
  + password\_hash (VARCHAR(255), NOT NULL) - Хэш пароля
  + first\_name (VARCHAR(50), NOT NULL) - Имя
  + last\_name (VARCHAR(50), NOT NULL) - Фамилия
  + middle\_name (VARCHAR(50)) - Отчество
  + phone (VARCHAR(20)) - Номер телефона
  + date\_of\_birth (DATE, NOT NULL) - Дата рождения
  + purpose (VARCHAR(255), NOT NULL) - Назначение
* **Отделы (Department):**
  + department\_id (INT, PRIMARY KEY, IDENTITY) - Уникальный идентификатор отдела
  + name (VARCHAR(100), UNIQUE, NOT NULL) - Название отдела
* **Подразделения (Subdivision):**
  + subdivision\_id (INT, PRIMARY KEY, IDENTITY) - Уникальный идентификатор подразделения
  + name (VARCHAR(100), UNIQUE, NOT NULL) - Название подразделения
  + department\_id (INT, FOREIGN KEY references Department(department\_id)) - Ссылка на отдел
* **Статусы заявок (RequestStatus):**
  + status\_id (INT, PRIMARY KEY, IDENTITY) - Уникальный идентификатор статуса
  + status\_name (VARCHAR(50), NOT NULL) - Название статуса (например, “Ожидает рассмотрения”, “Согласована”, “Отклонена”)
* **Сотрудники (Employee):**
  + employee\_id (INT, PRIMARY KEY, IDENTITY) - Уникальный идентификатор сотрудника
  + subdivision\_id (INT, FOREIGN KEY references Subdivision(subdivision\_id)) - Ссылка на подразделение
  + first\_name (VARCHAR(50), NOT NULL) - Имя
  + last\_name (VARCHAR(50), NOT NULL) - Фамилия
  + middle\_name (VARCHAR(50)) - Отчество
  + employee\_code (VARCHAR(20)) - Код сотрудника
* **Заявки (Request):**
  + request\_id (INT, PRIMARY KEY, IDENTITY) - Уникальный идентификатор заявки
  + user\_id (INT, FOREIGN KEY references Users(user\_id)) - Ссылка на пользователя, подавшего заявку
  + request\_type (VARCHAR(50), NOT NULL) - Тип заявки (например, “Личная”, “Групповая”)
  + start\_date (DATE, NOT NULL) - Дата начала действия заявки
  + end\_date (DATE, NOT NULL) - Дата окончания действия заявки
  + purpose (VARCHAR(255), NOT NULL) - Цель посещения
  + subdivision\_id (INT, FOREIGN KEY references Subdivision(subdivision\_id)) - Ссылка на подразделение, к которому относится посетитель (если применимо)
  + employee\_id (INT, FOREIGN KEY references Employee(employee\_id)) - Ссылка на сотрудника, к которому идет посетитель
  + status\_id (INT, FOREIGN KEY references RequestStatus(status\_id)) - Ссылка на статус заявки
  + denial\_reason (VARCHAR(255)) - Причина отказа (если заявка отклонена)
* **Посетители (Visitor):**
  + visitor\_id (INT, PRIMARY KEY, IDENTITY) - Уникальный идентификатор посетителя
  + request\_id (INT, FOREIGN KEY references Request(request\_id)) - Ссылка на заявку
  + first\_name (VARCHAR(50), NOT NULL) - Имя
  + last\_name (VARCHAR(50), NOT NULL) - Фамилия
  + middle\_name (VARCHAR(50)) - Отчество
  + phone (VARCHAR(20)) - Номер телефона
  + email (VARCHAR(255), NOT NULL) - Адрес электронной почты
  + date\_of\_birth (DATE, NOT NULL) - Дата рождения
  + passport\_series (VARCHAR(4), NOT NULL) - Серия паспорта
  + passport\_number (VARCHAR(6), NOT NULL) - Номер паспорта
  + login (VARCHAR(50), NOT NULL) - Логин
  + password (VARCHAR(255), NOT NULL) - Пароль
  + purpose (VARCHAR(255), NOT NULL) - Назначение
* **Групповые посетители (GroupVisitor):**
  + group\_visitor\_id (INT, PRIMARY KEY, IDENTITY) - Уникальный идентификатор группового посетителя
  + request\_id (INT, FOREIGN KEY references Request(request\_id)) - Ссылка на заявку
  + order\_number (INT, NOT NULL) - Порядковый номер в группе
  + first\_name (VARCHAR(50), NOT NULL) - Имя
  + last\_name (VARCHAR(50), NOT NULL) - Фамилия
  + middle\_name (VARCHAR(50)) - Отчество
  + phone (VARCHAR(20)) - Номер телефона
  + email (VARCHAR(255), NOT NULL) - Адрес электронной почты
  + date\_of\_birth (DATE, NOT NULL) - Дата рождения
  + passport\_series (VARCHAR(4), NOT NULL) - Серия паспорта
  + passport\_number (VARCHAR(6), NOT NULL) - Номер паспорта
  + group\_name (VARCHAR(255), NOT NULL) - Название группы
  + login (VARCHAR(50), NOT NULL) - Логин
  + password (VARCHAR(255), NOT NULL) - Пароль
  + purpose (VARCHAR(255), NOT NULL) - Назначение

**3.2. ERD-диаграмма**

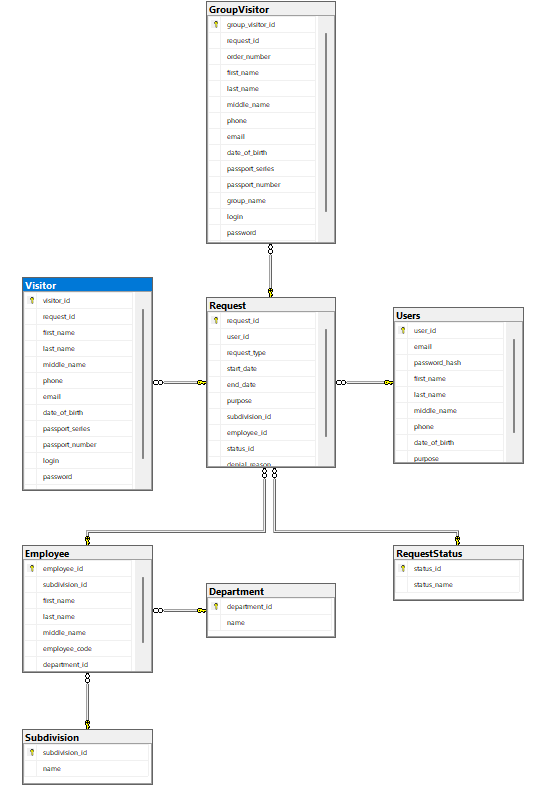


Рисунок 2 – ERD-диаграмма

**3.3. Нормализация базы данных**

База данных была нормализована до 3NF (Третья нормальная форма) с целью исключения избыточности данных и обеспечения целостности. Это подразумевает, что:

* **1NF:** Устранены повторяющиеся группы данных. Каждая таблица содержит атомарные значения.
* **2NF:** Устранены функциональные зависимости неключевых атрибутов от части составного первичного ключа. (В данной базе данных составные первичные ключи не используются, поэтому данное требование выполняется автоматически).
* **3NF:** Устранены транзитивные зависимости неключевых атрибутов от первичного ключа. Неключевые атрибуты зависят только от первичного ключа, а не от других неключевых атрибутов.

**4. РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ В СУБД**

**4.1. Выбор СУБД**

В качестве СУБД был выбран **SQL Server**, в связи с необходимостью интеграции с существующей инфраструктурой организации.

**4.2. Создание таблиц и связей**

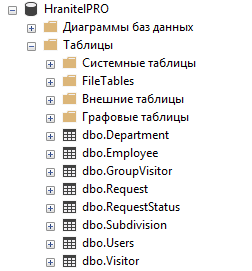


Рисунок 3 – Созданные таблицы в SQL Server

Скрипт создания базы данных приведен в Приложении 9.1. В данном скрипте определены все таблицы и связи между ними, а также указаны типы данных и ограничения для каждого атрибута.

**4.3. Заполнение таблиц данными**

После создания таблиц, они были заполнены тестовыми данными для проверки работоспособности системы. Примеры данных:

* **Users:** Записи о пользователях системы (администраторы, сотрудники, посетители).
* **Department:** Записи об отделах организации.
* **Subdivision:** Записи о подразделениях внутри отделов.
* **RequestStatus:** Записи о возможных статусах заявок (например, “Создана”, “На рассмотрении”, “Одобрена”, “Отклонена”).
* **Employee:** Записи о сотрудниках организации, с указанием их подразделений.
* **Request:** Записи о заявках на посещение, с указанием пользователя, типа заявки, дат посещения, цели, и т.д.
* **Visitor:** Записи о посетителях, подавших заявку.
* **GroupVisitor:** Записи о посетителях, подавших заявку в группе.

**5. РЕАЛИЗАЦИЯ УРОВНЕЙ ДОСТУПА**

**5.1. Определение ролей пользователей**

В программе определены следующие роли пользователей:

* **Администратор:** имеет полный доступ ко всем функциям системы, включая управление пользователями, заявками, настройками и т.д.
* **Сотрудник общего отдела:** может просматривать и утверждать/отклонять заявки на посещение.
* **Сотрудник охраны:** Может просматривать заявки на посещение и управлять доступом на территорию.
* **Пользователь (посетитель):** Может подавать заявки на посещение и просматривать их статус.

**5.2. Настройка прав доступа**

Реализация прав доступа осуществляется на уровне SQL Server с использованием ролей и разрешений. Для каждой роли создаются соответствующие представления (views) и хранимые процедуры (stored procedures), которые предоставляют доступ только к тем данным и функциям, которые необходимы для выполнения их обязанностей.

**6. СОЗДАНИЕ ЗАПРОСОВ И ОТЧЕТОВ**

**6.1. Примеры SQL-запросов**

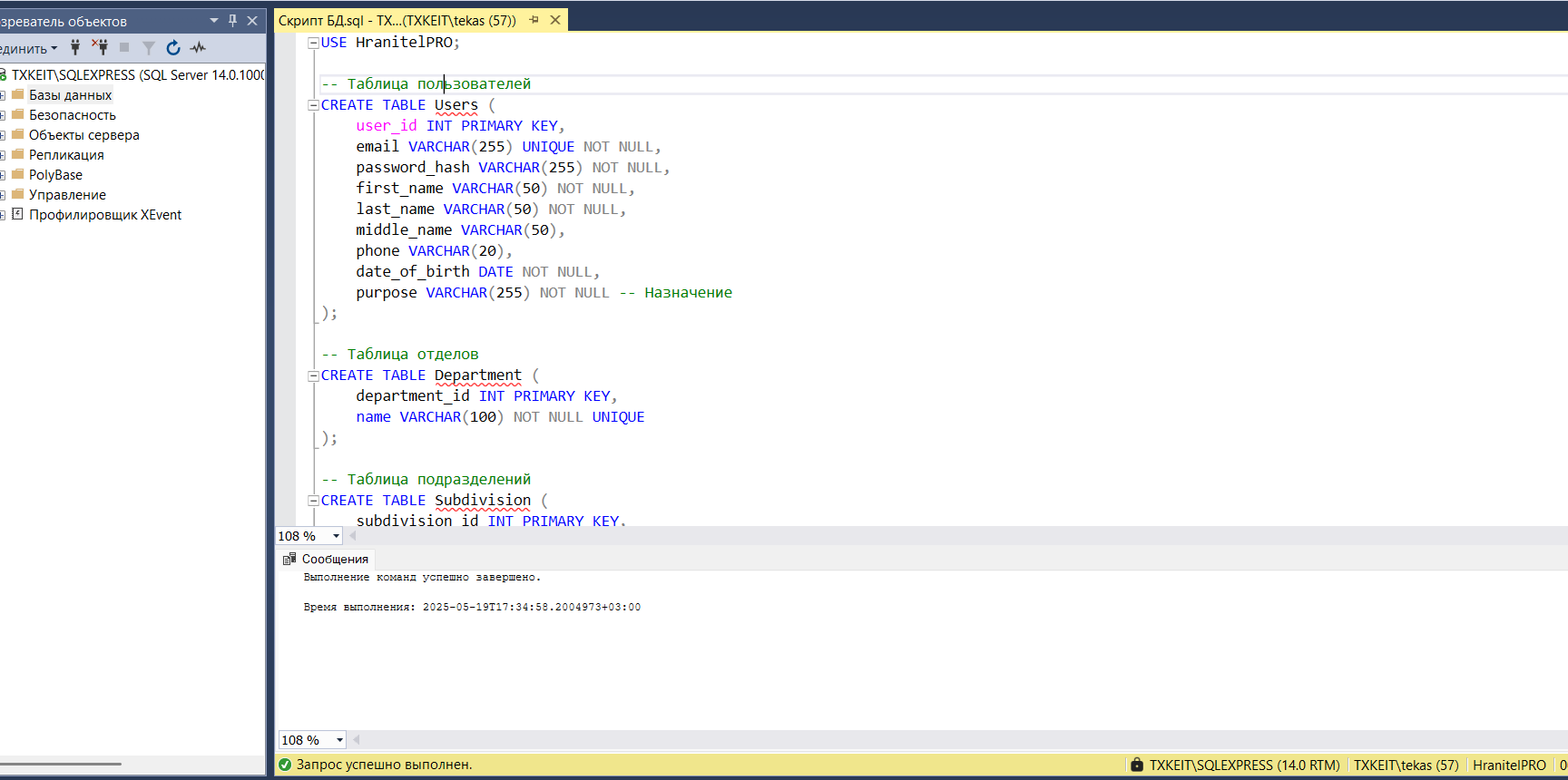
**Процедура авторизации:**

sql

USE HranitelPRO;  
**CREATE** **PROCEDURE** AuthenticateUser (  
 @email VARCHAR(255),  
 @password VARCHAR(255)  
)  
**AS**  
**BEGIN**  
 *-- Проверяем наличие пользователя с указанным email*  
 IF **EXISTS** (**SELECT** 1 **FROM** Users **WHERE** email = @email)  
 **BEGIN**  
 *-- Получаем хэш пароля из базы данных*  
 **DECLARE** @stored\_password\_hash VARCHAR(255);  
 **SELECT** @stored\_password\_hash = password\_hash **FROM** Users **WHERE** email = @email;  
  
 *-- Сравниваем введенный пароль с хэшем из базы данных*  
 IF HASHBYTES('SHA2\_512', @password + CAST(Salt **AS** VARCHAR(36))) = @stored\_password\_hash  
 **BEGIN**  
 *-- Аутентификация успешна*  
 **SELECT** 'Authentication successful' **AS** **Result**;  
 **END**  
 **ELSE**  
 **BEGIN**  
 *-- Неверный пароль*  
 **SELECT** 'Incorrect password' **AS** **Result**;  
 **END**  
 **END**  
 **ELSE**  
 **BEGIN**  
 *-- Пользователь не найден*  
 **SELECT** 'User not found' **AS** **Result**;  
 **END**  
**END**;

**Процедура регистрации:**

**CREATE** **PROCEDURE** RegisterUser (  
 @email VARCHAR(255),  
 @password VARCHAR(255),  
 @first\_name VARCHAR(50),  
 @last\_name VARCHAR(50),  
 @middle\_name VARCHAR(50),  
 @phone VARCHAR(20),  
 @date\_of\_birth DATE,  
 @purpose VARCHAR(255)  
)  
**AS**  
**BEGIN**  
 *-- Проверяем, не существует ли пользователь с таким email*  
 IF **NOT** **EXISTS** (**SELECT** 1 **FROM** Users **WHERE** email = @email)  
 **BEGIN**  
 *-- Генерируем случайную соль (salt)*  
 **DECLARE** @salt UNIQUEIDENTIFIER = NEWID();  
  
 *-- Хэшируем пароль с использованием соли*  
 **DECLARE** @password\_hash VARBINARY(64) = HASHBYTES('SHA2\_512', @password + CAST(@salt **AS** VARCHAR(36)));  
  
 *-- Добавляем нового пользователя в базу данных*  
 **INSERT INTO** Users (email, password\_hash, first\_name, last\_name, middle\_name, phone, date\_of\_birth, purpose, Salt)  
 **VALUES** (@email, @password\_hash, @first\_name, @last\_name, @middle\_name, @phone, @date\_of\_birth, @purpose, @salt);  
  
 *-- Возвращаем сообщение об успешной регистрации*  
 **SELECT** 'Registration successful' **AS** **Result**;  
 **END**  
 **ELSE**  
 **BEGIN**  
 *-- Возвращаем сообщение, что пользователь с таким email уже существует*  
 **SELECT** 'User with this email already exists' **AS** **Result**;  
 **END**  
**END**;



D

Рисунок 5 – Запрос на авторизацию в SQL Server

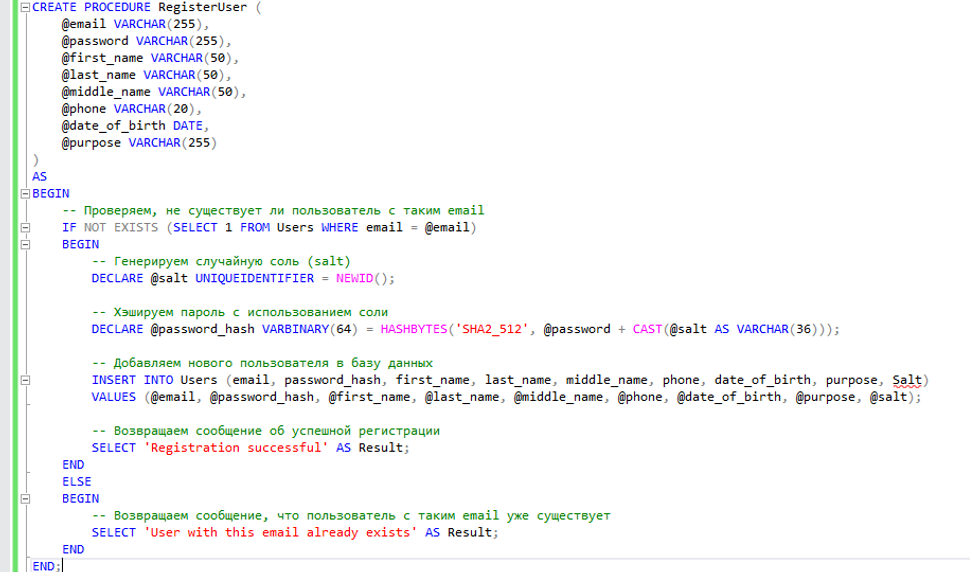
****

Рисунок 6 – Запрос на регистрацию в SQL Server

**6.2. Создание отчетов**

Примеры отчетов, которые могут быть сгенерированы:

* Список заявок на посещение за определенный период времени.
* Статистика заявок по статусам.
* Список посетителей, посетивших объект в определенную дату.
* Список сотрудников, к которым чаще всего приходят посетители.

Отчеты могут быть реализованы с использованием средств SQL Server Reporting Services (SSRS) или других инструментов для создания отчетов.

**7. РЕЗЕРВНОЕ КОПИРОВАНИЕ И ВОССТАНОВЛЕНИЕ**

Регулярное резервное копирование базы данных является важной частью администрирования. В SQL Server резервные копии могут быть созданы с использованием SQL Server Management Studio (SSMS) или с помощью T-SQL команд.

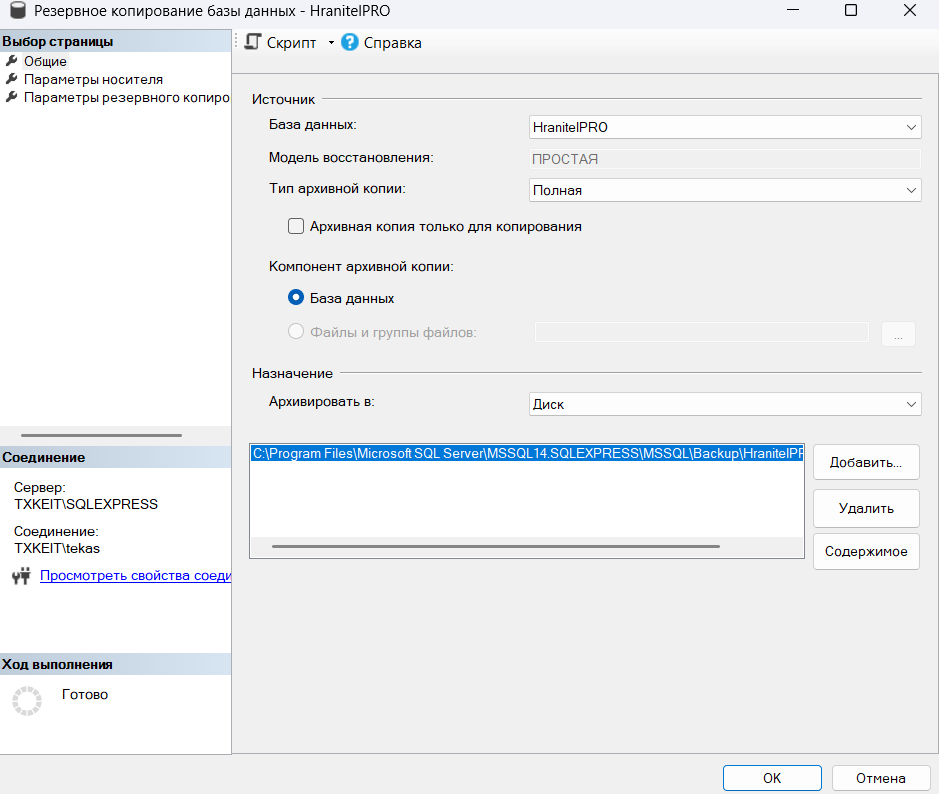


Рисунок 7 – Создание резервной копии

Восстановление базы данных из резервной копии также может быть выполнено с использованием SSMS или T-SQL. Важно регулярно проверять возможность восстановления базы данных из резервных копий.

**8. ЗАКЛЮЧЕНИЕ**

В ходе выполнения учебной практики была спроектирована и реализована база данных для модуля “HranitelPRO”, предназначенного для организации пропускного режима на объектах КИИ. Были изучены основные этапы разработки базы данных, включая анализ предметной области, проектирование структуры базы данных, нормализацию, реализацию в СУБД, создание запросов и отчетов, а также резервное копирование и восстановление.

**9. ПРИЛОЖЕНИЯ**

**9.1. Скрипт создания базы данных (SQL)**

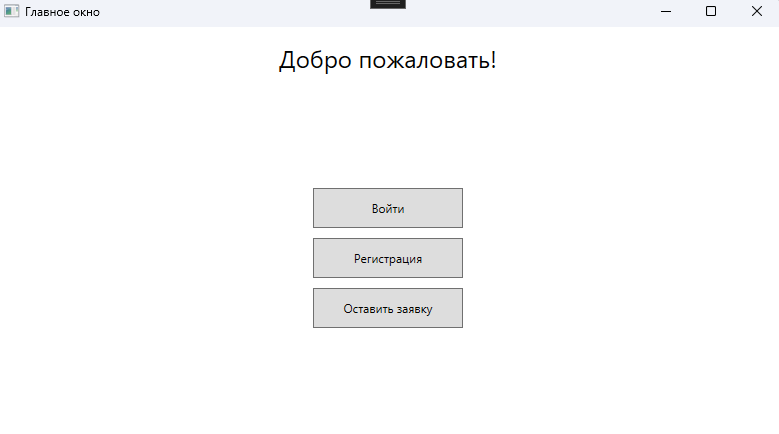
sql

*-- Создание базы данных*  
**CREATE** DATABASE HranitelPRO;  
GO  
  
USE HranitelPRO;  
  
*-- Таблица пользователей*

**CREATE TABLE** Users (  
 user\_id INT **PRIMARY KEY** **IDENTITY**(1,1),  
 email VARCHAR(255) **UNIQUE** **NOT NULL**,  
 password\_hash VARBINARY(64) **NOT NULL**, *-- Хэш пароля*  
 first\_name VARCHAR(50) **NOT NULL**,  
 last\_name VARCHAR(50) **NOT NULL**,  
 middle\_name VARCHAR(50),  
 phone VARCHAR(20),  
 date\_of\_birth DATE **NOT NULL**,  
 purpose VARCHAR(255) **NOT NULL**, *-- Назначение*  
 Salt UNIQUEIDENTIFIER **NOT NULL** *-- Соль для хэширования пароля*  
);  
  
*-- Таблица отделов*  
**CREATE TABLE** Department (  
 department\_id INT **PRIMARY KEY** **IDENTITY**(1,1),  
 name VARCHAR(100) **NOT NULL** **UNIQUE**  
);  
  
*-- Таблица подразделений*  
**CREATE TABLE** Subdivision (  
 subdivision\_id INT **PRIMARY KEY** **IDENTITY**(1,1),  
 name VARCHAR(100) **NOT NULL** **UNIQUE**,  
 department\_id INT,  
 **FOREIGN KEY** (department\_id) **REFERENCES** Department(department\_id) *-- Связь с таблицей отделов*  
);  
  
*-- Таблица статусов заявок*  
**CREATE TABLE** RequestStatus (  
 status\_id INT **PRIMARY KEY** **IDENTITY**(1,1),  
 status\_name VARCHAR(50) **NOT NULL** *-- Например, 'pending', 'approved', 'denied'*  
);  
  
*-- Таблица сотрудников*  
**CREATE TABLE** Employee (  
 employee\_id INT **PRIMARY KEY** **IDENTITY**(1,1),  
 subdivision\_id INT,  
 first\_name VARCHAR(50) **NOT NULL**,  
 last\_name VARCHAR(50) **NOT NULL**,  
 middle\_name VARCHAR(50),  
 employee\_code VARCHAR(20), *-- Код сотрудника*  
 **FOREIGN KEY** (subdivision\_id) **REFERENCES** Subdivision(subdivision\_id) *-- Связь с таблицей подразделений*  
);  
  
*-- Таблица заявок*  
**CREATE TABLE** Request (  
 request\_id INT **PRIMARY KEY** **IDENTITY**(1,1),  
 user\_id INT,  
 request\_type VARCHAR(50) **NOT NULL**,  
 start\_date DATE **NOT NULL**,  
 end\_date DATE **NOT NULL**,  
 purpose VARCHAR(255) **NOT NULL**,  
 subdivision\_id INT,  
 employee\_id INT,  
 status\_id INT,  
 denial\_reason VARCHAR(255),  
 **FOREIGN KEY** (user\_id) **REFERENCES** Users(user\_id),  
 **FOREIGN KEY** (subdivision\_id) **REFERENCES** Subdivision(subdivision\_id),  
 **FOREIGN KEY** (employee\_id) **REFERENCES** Employee(employee\_id),  
 **FOREIGN KEY** (status\_id) **REFERENCES** RequestStatus(status\_id)  
);  
  
*-- Таблица посетителей*  
**CREATE TABLE** Visitor (  
 visitor\_id INT **PRIMARY KEY** **IDENTITY**(1,1),  
 request\_id INT,  
 first\_name VARCHAR(50) **NOT NULL**,  
 last\_name VARCHAR(50) **NOT NULL**,  
 middle\_name VARCHAR(50),  
 phone VARCHAR(20),  
 email VARCHAR(255) **NOT NULL**,  
 date\_of\_birth DATE **NOT NULL**,  
 passport\_series VARCHAR(4) **NOT NULL**,  
 passport\_number VARCHAR(6) **NOT NULL**,  
 login VARCHAR(50) **NOT NULL**,  
 password VARCHAR(255) **NOT NULL**,  
 purpose VARCHAR(255) **NOT NULL**,  
 **FOREIGN KEY** (request\_id) **REFERENCES** Request(request\_id)  
);  
  
*-- Таблица групповых посетителей*  
**CREATE TABLE** GroupVisitor (  
 group\_visitor\_id INT **PRIMARY KEY** **IDENTITY**(1,1),  
 request\_id INT,  
 order\_number INT **NOT NULL**,  
 first\_name VARCHAR(50) **NOT NULL**,  
 last\_name VARCHAR(50) **NOT NULL**,  
 middle\_name VARCHAR(50),  
 phone VARCHAR(20),  
 email VARCHAR(255) **NOT NULL**,  
 date\_of\_birth DATE **NOT NULL**,  
 passport\_series VARCHAR(4) **NOT NULL**,  
 passport\_number VARCHAR(6) **NOT NULL**,  
 group\_name VARCHAR(255) **NOT NULL**,  
 login VARCHAR(50) **NOT NULL**,  
 password VARCHAR(255) **NOT NULL**,  
 purpose VARCHAR(255) **NOT NULL**,  
 **FOREIGN KEY** (request\_id) **REFERENCES** Request(request\_id)  
);

**9.2. Скриншоты интерфейса (авторизация, регистрация, и т.д.)**

**1. Авторизация и регистрация**

Рисунок 9 – Основное окно программы

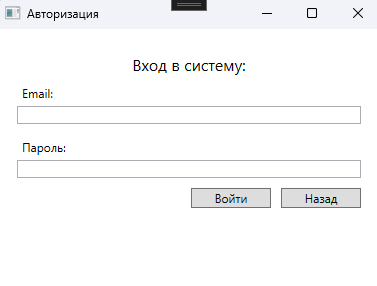


Рисунок 10 – Окно входа

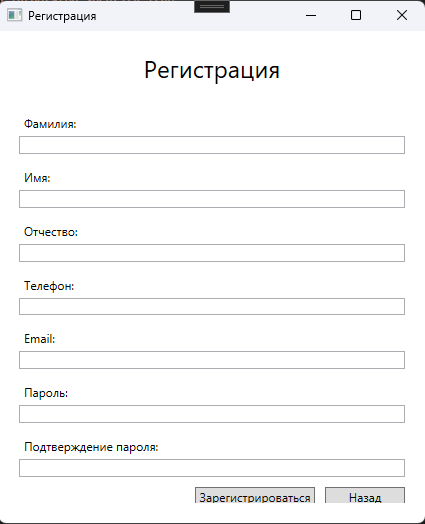


Рисунок 11 – Окно регистрации

**2. Список созданных пользователем заявок**

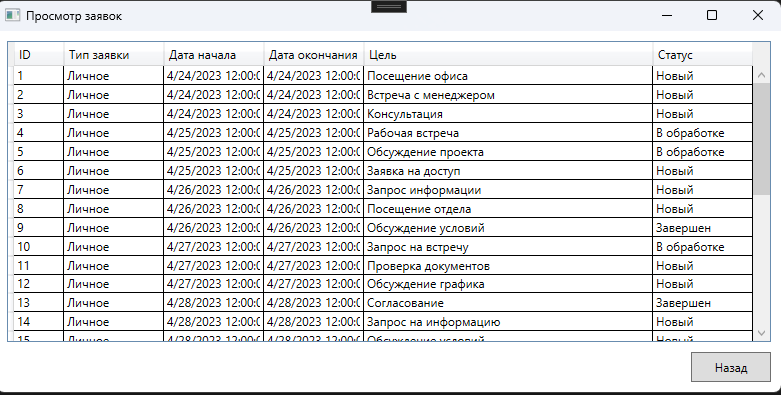
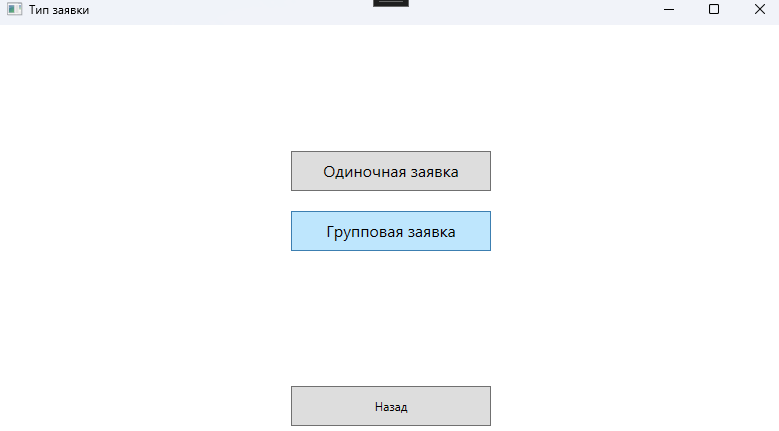


Рисунок 12 – Список заявок

**3. Создание заявок**

Рисунок 13 – Создание заявок

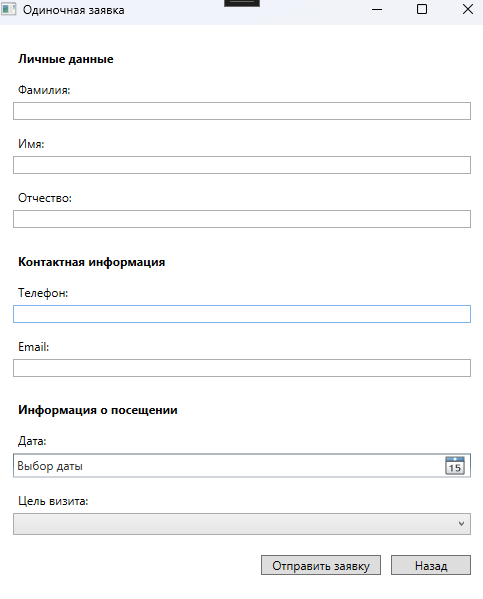
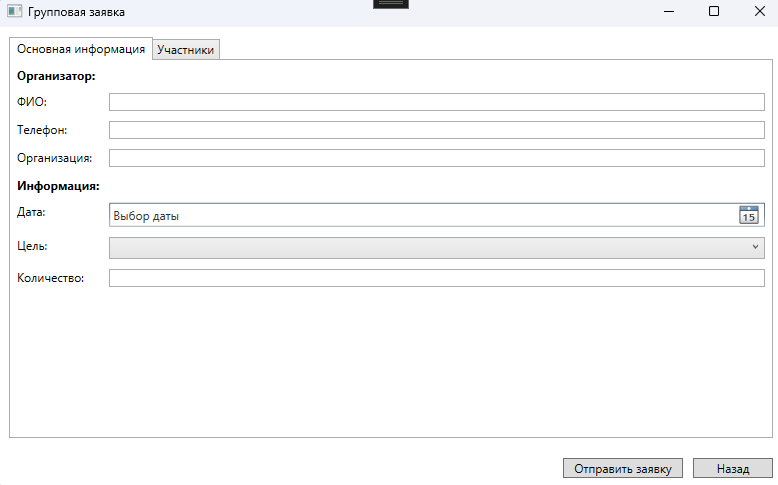


Рисунок 14 – Окно одиночной заявки

Рисунок 15 – Окно групповой заявки (основная информация)

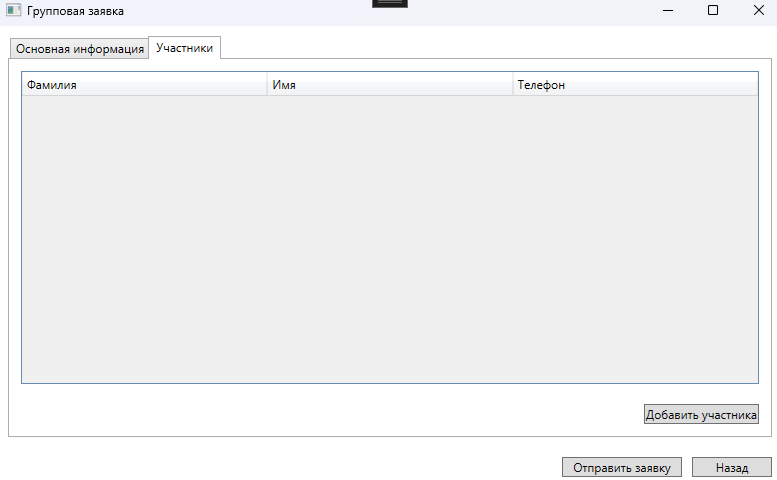


Рисунок 16 – Окно групповой заявки (участники)

**9.3. Ссылка на репозиторий**

https://github.com/Teqkillah1337/repository1