МИНИСТЕРСТВО ОБРАЗОВАНИЯ КИРОВСКОЙ ОБЛАСТИ

Кировское областное государственное профессиональное образовательное

бюджетное учреждение

«Слободской колледж педагогики и социальных отношений»

**ОТЧЕТ**

**по учебной практике**

**ПМ 02. Разработка и администрирование баз данных**

**Тема: «Разработка ИС «Хранитель ПРО»**

Студента

Кротовой Ксении Николаевны

Группа 21П-1

Специальность 09.02.07 Информационные системы и программирование

Руководитель практики от колледжа

Калинин Арсений Олегович

\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись расшифровка

2024 год

**Содержание**

1. Выполнение анализа и предварительной обработки информации
2. Выделение объектов и атрибутов в соответствии с заданием
3. Проектирование и нормализация БД в полном соответствии с поставленной задачей и применением CASE-средств
4. Выполнение построения БД в предложенной СУБД и заполнение всех таблиц с помощью соответствующих средств
5. Реализация уровней доступа для различных категорий пользователей
6. Создание запросов и отчетов в соответствии с заданием
7. Выполнение резервного копирования БД и восстановление состояния БД на заданную дату.
8. Заключение.
9. Приложения
10. **Выполнение анализа и предварительной обработки информации**

**Предметная область:** Хранитель ПРО

Целью моей работы является создание ИС «Хранитель ПРО» для работы с посетителями, для обработки информации о посетителях сотрудниками различных отделов, для работы с заявками посетителей. Система предназначена для работы с посетителями, для обработки информации о посетителях сотрудниками различных отделов, для работы с заявками посетителей.

**Концептуальная модель базы данных (база данных в последствии работы изменялась, добавлялись новые сущности и атрибуты):**

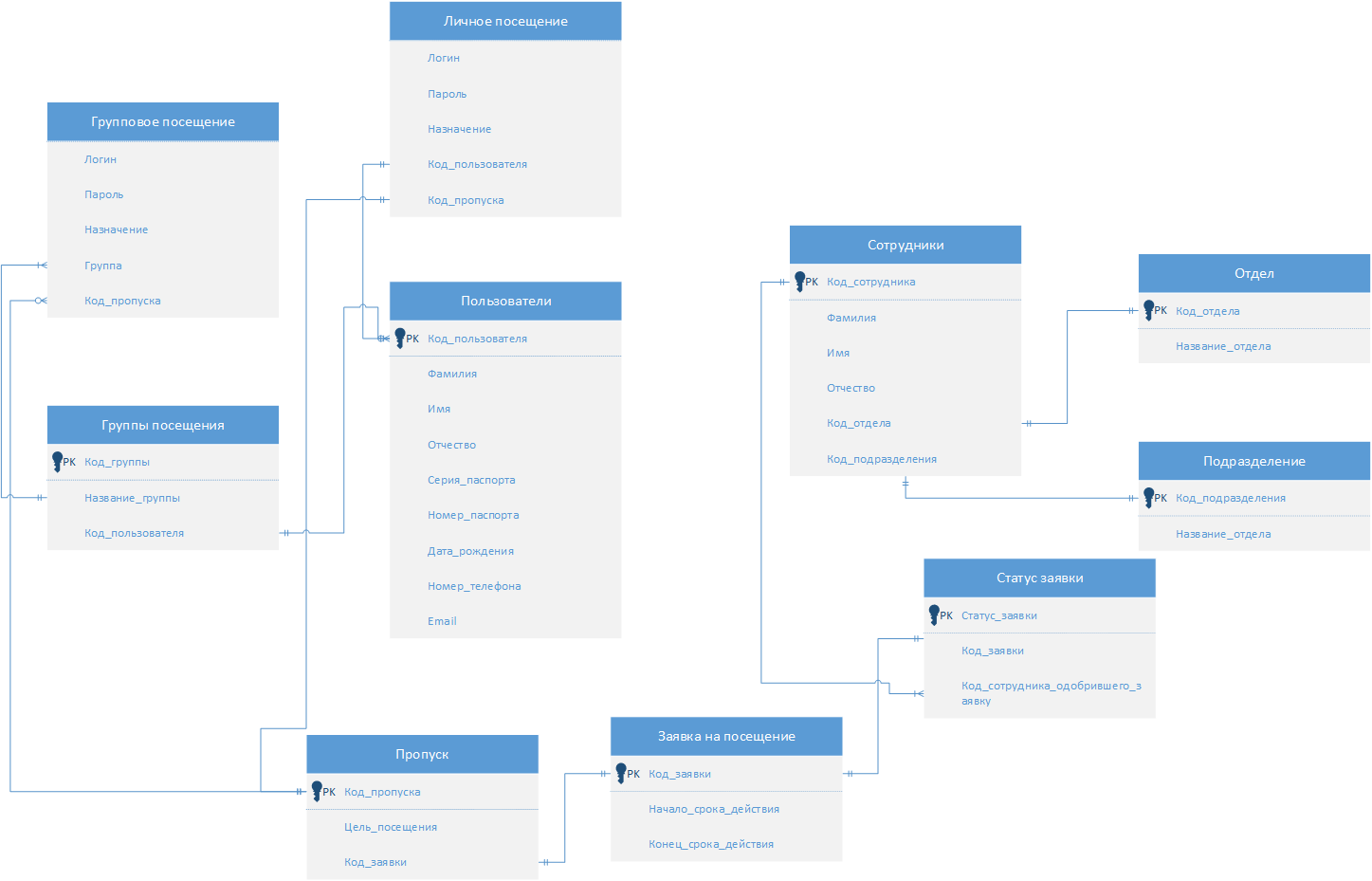


Рисунок -концептуальная модель бд

**Диаграмма вариантов использования системы «Хранитель ПРО»:**

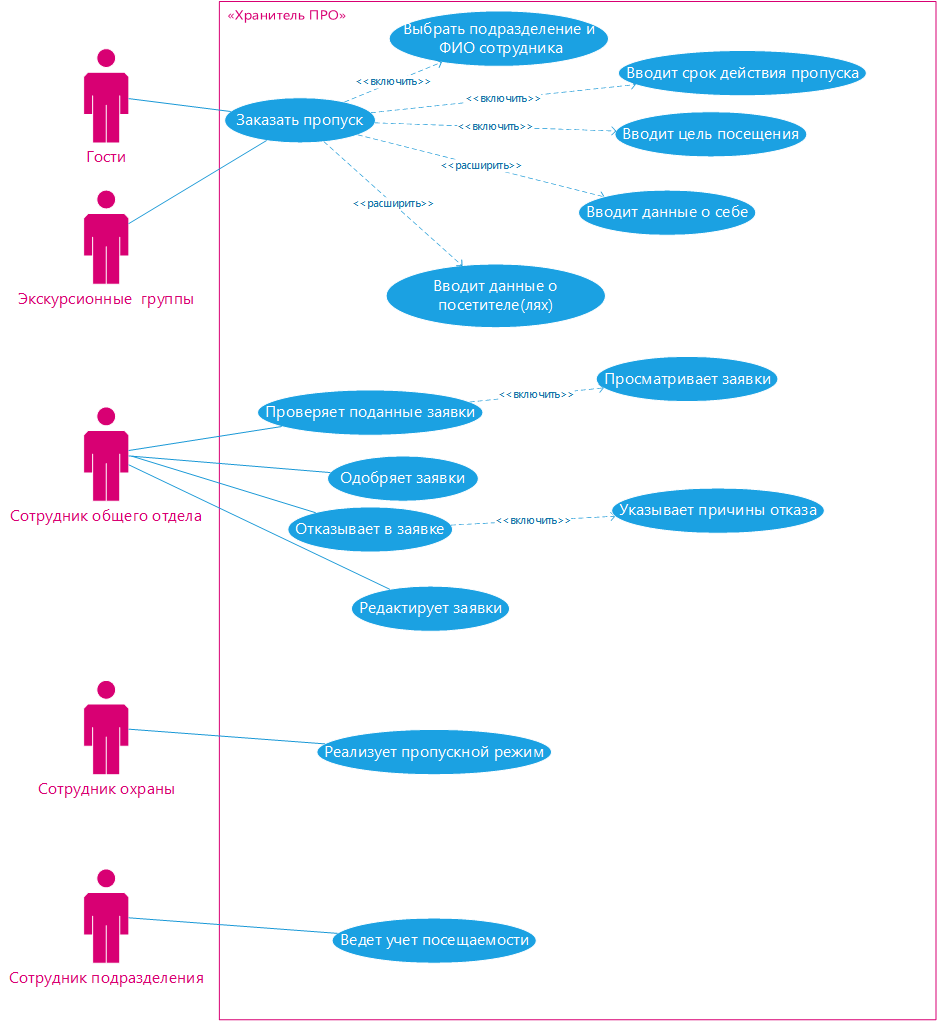


Рисунок -Варианты использования системы

1. **Выделение объектов и атрибутов в соответствии с заданием**

Для дальнейшей работы по созданию базы данных я выделила следующие сущности и атрибуты:

**Таблица Пользователи**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key | Field name | Data type | Required? | Notes |
| PK | Код\_пользователя | Int | Y | Код пользователя, обязательное поле |
|  | Фамилия | varchar(30) | Y | Фамилия (строка), обязательное поле |
|  | Имя | varchar(30) | Y | Имя (строка), обязательное поле |
|  | Отчество | varchar(30) | Y | Отчество (строка), обязательное поле |
|  | Серия\_паспорта | varchar(4) | Y | Серия паспорта (строка), обязательное поле |
|  | Номер\_паспорта | varchar(6) | Y | Номер паспорта (строка), обязательное поле |
|  | Дата\_рождения | Date | Y | Дата рождения, обязательное поле |
|  | Номер\_телефона | varchar(20) | Y | Номер телефона (строка), обязательное поле |
|  | Email | varchar(30) | Y | Электронная почта (строка), обязательное поле |
|  | Фотография | varbinary(MAX) | N |  |
|  | Документы | varbinary(MAX) | N |  |

**Таблица Сотрудники**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key | Field name | Data type | Required? | Notes |
| PK | Код\_сотрудника | Int | Y | Код сотрудника, обязательное поле |
|  | Фамилия | varchar(30) | Y | Фамилия (строка), обязательное поле |
|  | Имя | varchar(30) | Y | Имя (строка), обязательное поле |
|  | Отчество | varchar(30) | Y | Отчество (строка), обязательное поле |
| FK | Код\_отдела | Int | N | Код отдела |
| FK | Код\_подразделения | Int | N | Код подразделения |

**Таблица Отдел**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key | Field name | Data type | Required? | Notes |
| PK | Код\_отдела | Int | Y | Код отдела (число), обязательное поле |
|  | Название\_отдела | varchar(30) | Y | Название отдела (строка), обязательное поле |

**Таблица Подразделение**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key | Field name | Data type | Required? | Notes |
| PK | Код\_подразделения | Int | Y | Код подразделения (число), обязательное поле |
|  | Название\_подразделения | varchar(30) | Y | Название подразделения (строка), обязательное поле |

**Таблица Заявка на посещение**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key | Field name | Data type | Required? | Notes |
| PK | Код\_заявки | Int | Y | Код заявки (число), обязательное поле |
|  | Начало\_срока\_действия | Date | N | Начало срока действия заявки |
|  | Конец\_срока\_действия | Date | N | Конец срока действия заявки |

**Таблица Статус заявки**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key | Field name | Data type | Required? | Notes |
|  | Статус\_заявки | varchar(20) | N | Статус заявки (строка), обязательное поле |
| PK | Код\_заявки | Int | Y | Код заявки (число), обязательное поле |
|  | Код\_сотрудника\_одобрившего\_заявку | Int | Y | Код сотрудника, обязательное поле |

**Таблица Пропуск**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key | Field name | Data type | Required? | Notes |
| PK | Код\_пропуска | Int | Y | Код пропуска (число), обязательное поле |
| FK | Код\_заявки | Int | Y | Код заявки (число), обязательное поле |
|  | Цель\_посещения | varchar(20) | Y | Цель посещения |
|  | Принимающий\_сотрудник | int | N | Код сотрудника |

**Таблица Группы посещения**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key | Field name | Data type | Required? | Notes |
| PK | Код\_группы | Int | Y | Код группы (число), обязательное поле |
| FK | Название\_группы | varchar(40) | N | Название группы (строка), обязательное поле |
| FK | Код\_пользователя | varchar(50) | N | Код пользователя (заполняется как список в строчку) |

**Таблица Личное посещение**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key | Field name | Data type | Required? | Notes |
| PK | Логин | varchar(20) | Y | Логин(строка), обязательное поле |
|  | Пароль | varchar(20) | Y | Пароль (строка), обязательное поле |
|  | Назначение | varchar(40) | Y | Назначение,  обязательное поле |
| FK | Код\_пользователя | Int | Y | Код пользователя,  обязательное поле |
| FK | Код\_пропуска | Int | N | Код пропуска |

**Таблица Групповое посещение**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key | Field name | Data type | Required? | Notes |
| PK | Логин | varchar(20) | Y | Логин(строка), обязательное поле |
|  | Пароль | varchar(20) | Y | Пароль (строка), обязательное поле |
|  | Назначение | varchar(40) | N | Назначение |
| FK | Код\_пропуска | Int | N | Код пропуска |
| FK | Код\_группы | Int | N | Код группы |
|  | Код\_пользователя | Int | N | Код\_пользователя |

**Таблица Черный список**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key | Field name | Data type | Required? | Notes |
| PK | Код\_записи | Int | Y | Код записи, обязательное поле |
|  | Код\_пропуска | Int | N | Код пропуска |
| FK | Код\_пользователя | Int | N | Код пользователя |
|  | Статус\_записи | varchar(30) | N | Статус\_записи |

1. **Проектирование и нормализация БД в полном соответствии с поставленной задачей и применением CASE-средств**

Проектирование и нормализацию БД я проводила в диаграмме БД, добавляя к атрибутам сущностей типы данных и ключевые атрибуты, также нормализовала связи между сущностями.

Диаграмма базы данных в SQL Server:

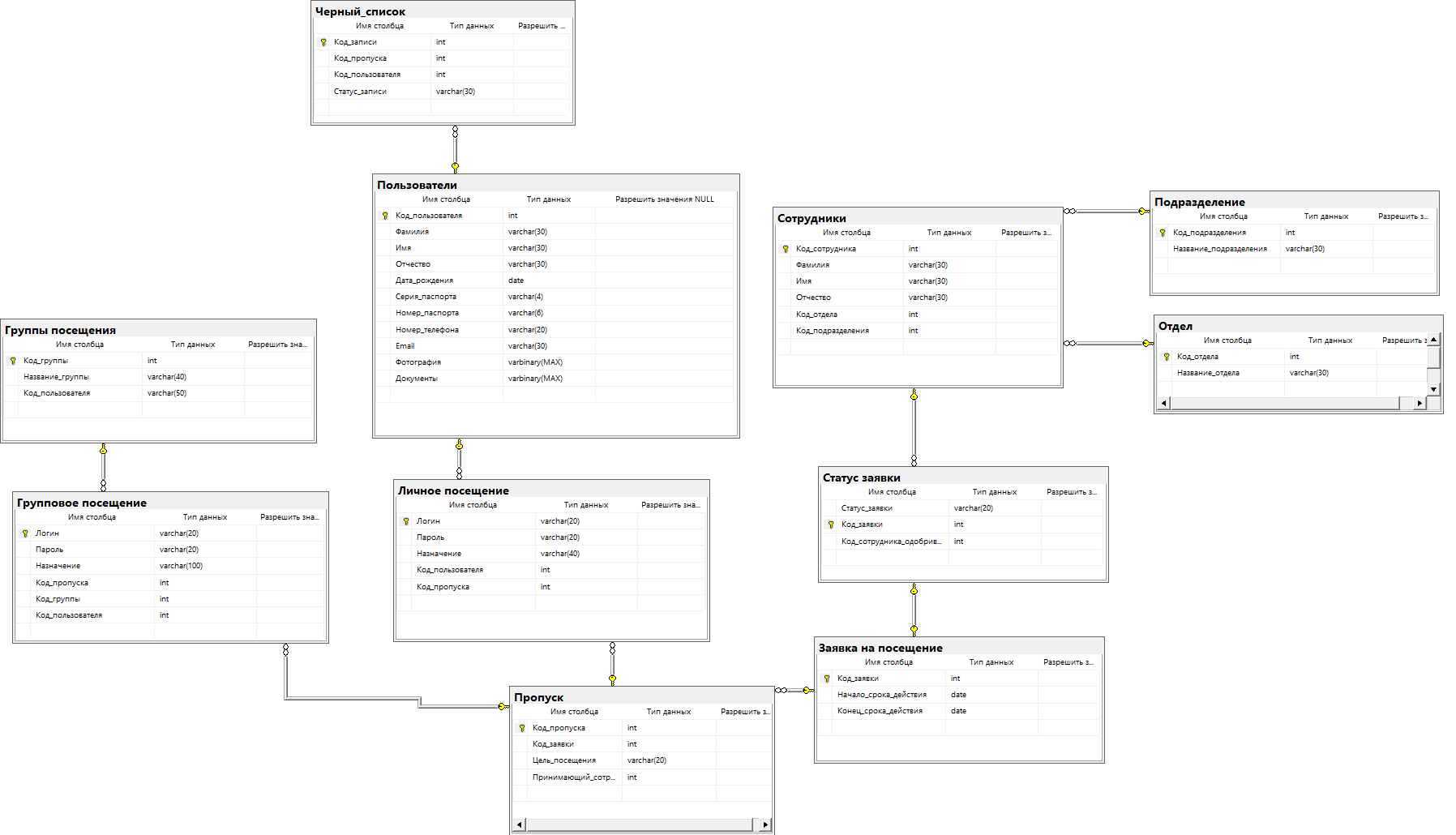


Рисунок -Диаграмма бд SQL

1. **Выполнение построения БД в предложенной СУБД и заполнение всех таблиц с помощью соответствующих средств**

Базу данных я назвала «Хранитель Про» и создала ее в приложении Microsoft SQL Server. Таблицы создавала по концептуальной модели БД, составленной мной (концептуальная модель не изменялась, но новые сущности добавлялись по мере разработки базы данных). В каждую таблицу я внесла демонстрационные данные (при помощи импорта и добавление данных вручную).

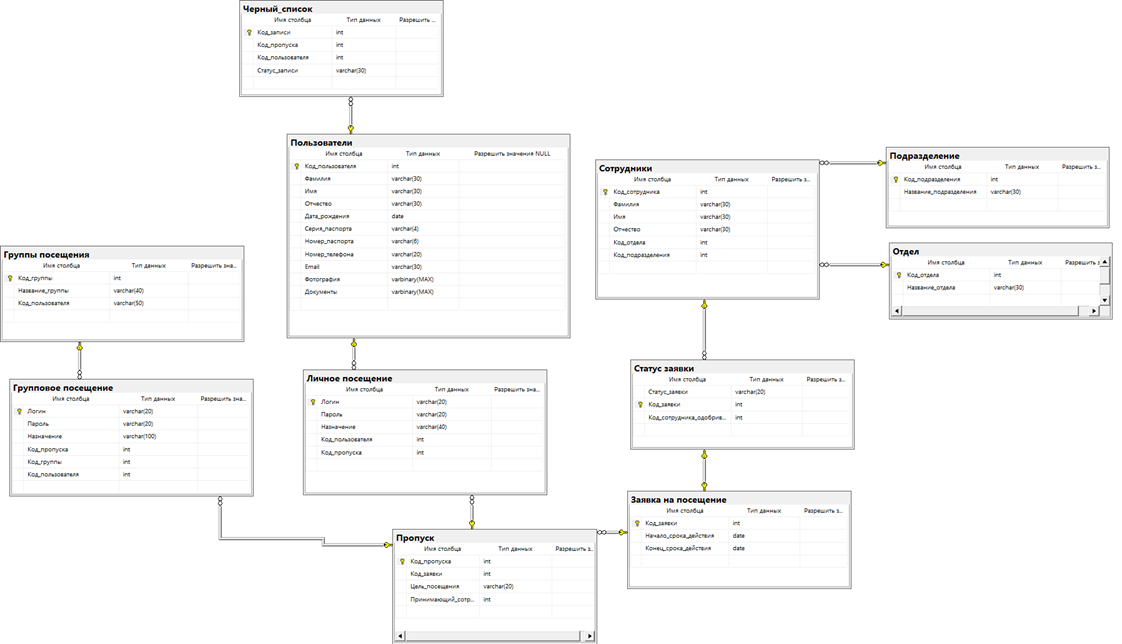


Рисунок -Диаграмма БД

Таблицы в базе данных:

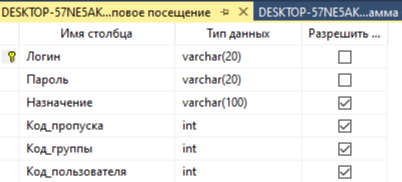


Рисунок -Групповое посещение

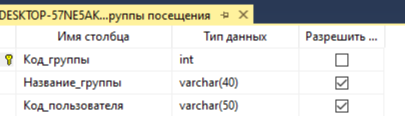


Рисунок -Группы посещения

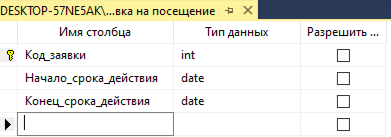


Рисунок -Заявка на посещение

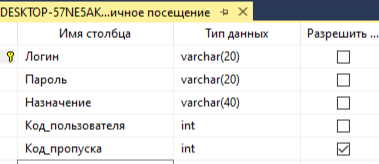


Рисунок -Личное посещение

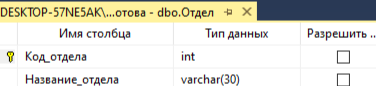


Рисунок -Отдел

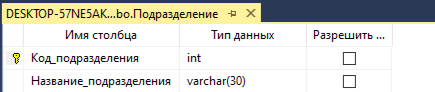


Рисунок -Подразделение

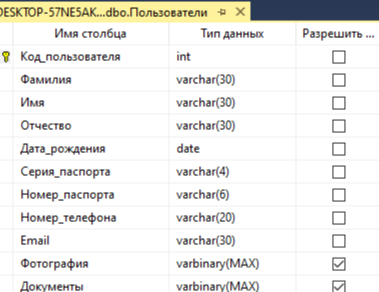


Рисунок -Пользователи

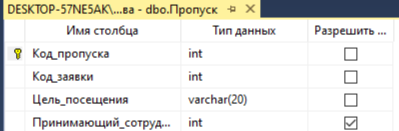


Рисунок -Пропуск

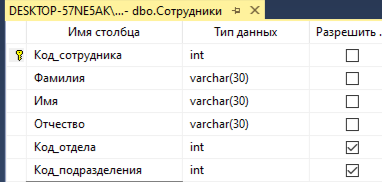


Рисунок -Сотрудники

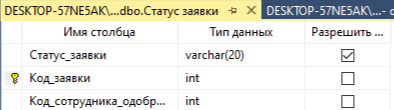


Рисунок -Статус заявки

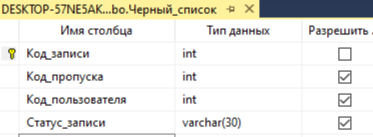


Рисунок -Черный список

Также были созданы клиентское приложение и приложение для сотрудников.

Интерфейс клиентского приложения:

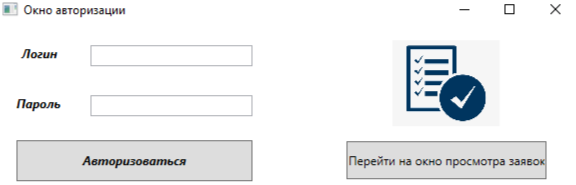


Рисунок -Окно авторизации

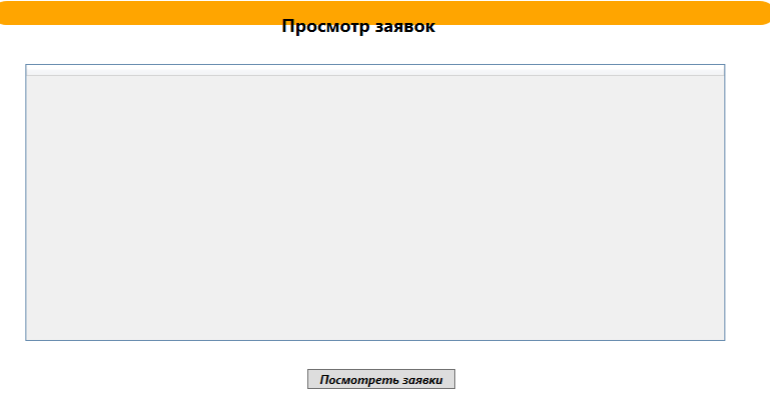


Рисунок -Окно просмотра заявок



Рисунок -Окно выбора типа посещения мероприятия

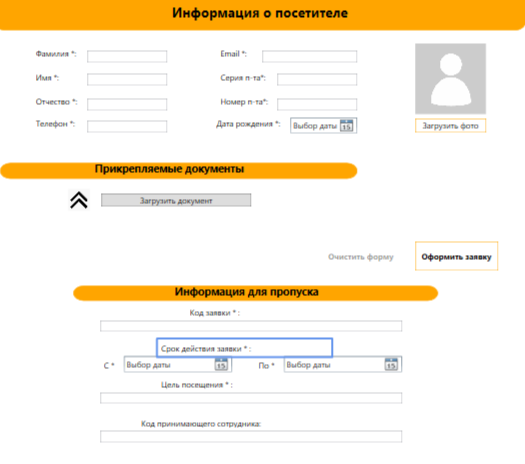


Рисунок -Окно записи личного посещения

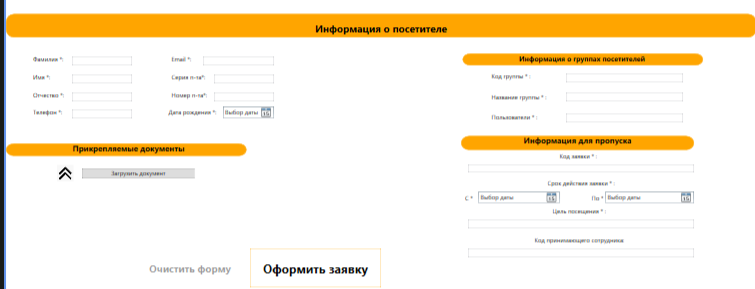


Рисунок -Окно записи на групповое посещение

Приложение сотрудников:

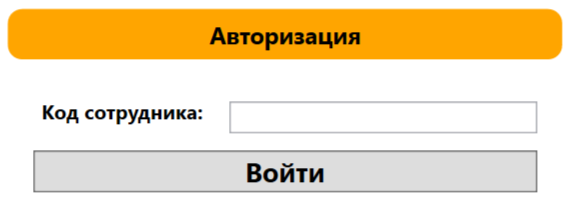


Рисунок -Окно авторизации

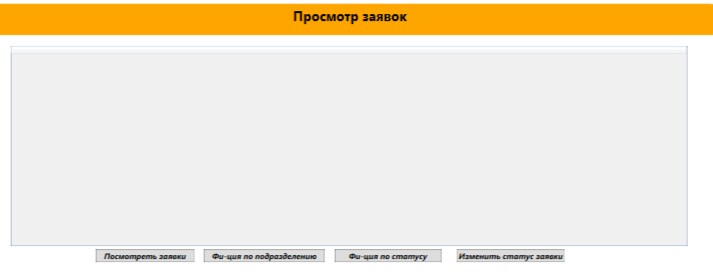


Рисунок -Окно просмотра заявок с различными функциями

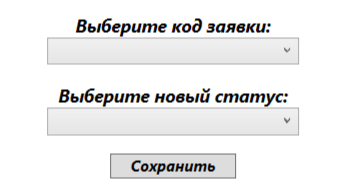


Рисунок -Окно изменения статуса заявок

1. **Реализация уровней доступа для различных категорий пользователей**

Для того чтобы можно было пользоваться данной БД, я создала пользователей через SQL запрос, которые будут ею пользоваться:

-- Создание пользователя с правами на чтение данных из таблицы Пользователи

CREATE USER ReadOnlyUser FOR LOGIN ReadOnlyLogin;

GRANT SELECT ON dbo.Пользователи TO ReadOnlyUser;

-- Создание пользователя с правами на добавление, изменение и удаление данных из таблицы Пользователи

CREATE USER ReadWriteUser FOR LOGIN ReadWriteLogin;

GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.Пользователи TO ReadWriteUser;

-- Создание пользователя с правами на чтение данных из всех таблиц

CREATE USER ReadOnlyAllUser FOR LOGIN ReadOnlyAllLogin;

GRANT SELECT ON dbo.Пользователи TO ReadOnlyAllUser;

GRANT SELECT ON dbo.[Групповое посещение] TO ReadOnlyAllUser;

GRANT SELECT ON dbo.[Личное посещение] TO ReadOnlyAllUser;

GRANT SELECT ON dbo.Сотрудники TO ReadOnlyAllUser;

-- Создание пользователя с правами на изменение данных из всех таблиц

CREATE USER ReadWriteAllUser FOR LOGIN ReadWriteAllLogin;

GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.Пользователи TO ReadWriteAllUser;

GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.[Групповое посещение] TO ReadWriteAllUser;

GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.[Личное посещение] TO ReadWriteAllUser;

GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.Сотрудники TO ReadWriteAllUser;

1. **Создание запросов и отчетов в соответствии с заданием**

По созданным мной таблицах в БД я написала следующие запросы, триггеры, хранимые процедуры, функции и представления на языке SQL:

-- Запрос на группировку сотрудников по отделам и подсчету их количества

SELECT Код\_отдела, COUNT(\*) as Количество\_сотрудников FROM dbo.Сотрудники GROUP BY Код\_отдела;

-- Запрос на сортировку пользователей по фамилии

SELECT \* FROM dbo.Пользователи ORDER BY Фамилия;

-- Запрос на добавление новой заявки на посещение

INSERT INTO dbo.[Заявка на посещение] (Код\_заявки, Начало\_срока\_действия, Конец\_срока\_действия) VALUES (1001, '2022-10-01', '2022-10-15');

-- Запрос на удаление записей из таблицы Групповое посещение, где Назначение указано как NULL

DELETE FROM dbo.[Групповое посещение] WHERE Назначение IS NULL;

-- Запрос на выбор всех пользователей и их данные

SELECT \* FROM dbo.Пользователи;

-- Запрос на выбор всех сотрудников и их данные

SELECT \* FROM dbo.Сотрудники;

-- Запрос на выбор всех посещений из таблицы Личное посещение

SELECT \* FROM dbo.[Личное посещение];

-- Запрос на выбор всех заявок на посещение и их сроки действия

SELECT Код\_заявки, Начало\_срока\_действия, Конец\_срока\_действия FROM dbo.[Заявка на посещение];

-- Запрос на выбор всех записей из таблицы Черный\_список с указанием причины

SELECT Код\_записи, Статус\_записи FROM dbo.Черный\_список;

-- Запрос для авторизации

DECLARE @Login VARCHAR(20) = 'введенный логин';

DECLARE @Password VARCHAR(20) = 'введенный пароль';

DECLARE @IsSuccessful BIT;

DECLARE @UserID INT;

-- Проверяем наличие пользователя в таблице "Групповое посещение"

IF EXISTS (SELECT 1 FROM [dbo].[Групповое посещение] WHERE [Логин] = @Login AND [Пароль] = @Password)

BEGIN

SELECT @UserID = Код\_пользователя FROM [dbo].[Групповое посещение] WHERE [Логин] = @Login;

SET @IsSuccessful = 1;

END

ELSE

BEGIN

-- Если пользователь не найден в таблице "Групповое посещение", проверяем таблицу "Личное посещение"

IF EXISTS (SELECT 1 FROM [dbo].[Личное посещение] WHERE [Логин] = @Login AND [Пароль] = @Password)

BEGIN

SELECT @UserID = Код\_пользователя FROM [dbo].[Личное посещение] WHERE [Логин] = @Login;

SET @IsSuccessful = 1;

END

ELSE

BEGIN

-- Если пользователь не найден ни в одной из таблиц, устанавливаем флаг успешной авторизации в 0

SET @IsSuccessful = 0;

END

END

SELECT @IsSuccessful AS IsSuccessful, @UserID AS UserID;

-- Запрос Регистрация пользователя в таблице Пользователи

INSERT INTO [Хранитель ПРО Кротова].[dbo].[Пользователи] ([Фамилия], [Имя], [Отчество], [Дата\_рождения], [Серия\_паспорта], [Номер\_паспорта], [Номер\_телефона], [Email])

VALUES ('Иванов', 'Иван', 'Иванович', '1990-01-01', '1234', '567890', '+1234567890', 'ivanov@example.com');

-- Получение ID нового пользователя

DECLARE @UserID INT;

SET @UserID = SCOPE\_IDENTITY();

-- Регистрация пропуска пользователя в таблице Пропуск

INSERT INTO [Хранитель ПРО Кротова].[dbo].[Пропуск] ([Код\_заявки], [Цель\_посещения])

VALUES (1, 'Работа');

-- Получение ID нового пропуска

DECLARE @PassID INT;

SET @PassID = SCOPE\_IDENTITY();

-- Регистрация личного посещения пользователя в таблице Личное посещение

INSERT INTO [Хранитель ПРО Кротова].[dbo].[Личное посещение] ([Логин], [Пароль], [Назначение], [Код\_пользователя], [Код\_пропуска])

VALUES ('ivanov', 'password', 'Работа', @UserID, @PassID);

-- Создание триггера для автоматической генерации уникального логина при вставке новой записи в таблицу Пользователи

CREATE TRIGGER trg\_GenerateUniqueLogin

ON Пользователи

AFTER INSERT

AS

BEGIN

DECLARE @FirstName VARCHAR(30);

DECLARE @LastName VARCHAR(30);

DECLARE @BirthDate DATE;

DECLARE @NewLogin VARCHAR(50);

-- Получаем данные о вставленном пользователе

SELECT @FirstName = Фамилия, @LastName = Имя, @BirthDate = Дата\_рождения

FROM inserted;

-- Генерируем уникальный логин

SET @NewLogin = dbo.GENERATE\_UNIQUE\_LOGIN(@FirstName, @LastName, @BirthDate);

-- Обновляем логин в новой записи

UPDATE Пользователи

SET Логин = @NewLogin

WHERE Код\_пользователя IN (SELECT Код\_пользователя FROM inserted);

END;

GO

CREATE VIEW ViewListRequests AS

SELECT

ЗП.Код\_заявки AS Код\_заявки,

ЗП.Начало\_срока\_действия AS Начало\_срока\_действия,

ЗП.Конец\_срока\_действия AS Конец\_срока\_действия,

СЗ.Статус\_заявки AS Статус\_заявки,

СЗ.Код\_сотрудника\_одобрившего\_заявку AS Код\_сотрудника\_одобрившего\_заявку,

С.Фамилия AS Фамилия,

С.Имя AS Имя,

С.Отчество AS Отчество,

П.Цель\_посещения AS Цель\_посещения,

П.Принимающий\_сотрудник AS Принимающий\_сотрудник,

Подр.Название\_подразделения AS Название\_подразделения

FROM

[dbo].[Заявка на посещение] AS ЗП

JOIN

[dbo].[Статус заявки] AS СЗ ON ЗП.Код\_заявки = СЗ.Код\_заявки

JOIN

[dbo].[Сотрудники] AS С ON СЗ.Код\_сотрудника\_одобрившего\_заявку = С.Код\_сотрудника

JOIN

[dbo].[Пропуск] AS П ON ЗП.Код\_заявки = П.Код\_заявки

JOIN

[dbo].[Подразделение] AS Подр ON С.Код\_подразделения = Подр.Код\_подразделения;

USE [Хранитель ПРО Кротова]

GO

--хранимая процедура для сортировки заявок по подразделению и статусу

CREATE PROCEDURE FilteringRequests

@Подразделение varchar(30),

@Статус\_заявки varchar(20)

AS

BEGIN

SET NOCOUNT ON;

SELECT \*

FROM [Хранитель ПРО Кротова].[dbo].[Заявка на посещение] z

INNER JOIN [Хранитель ПРО Кротова].[dbo].[Статус заявки] pp ON z.[Код\_заявки] = pp.[Код\_заявки]

INNER JOIN [Хранитель ПРО Кротова].[dbo].[Сотрудники] s ON pp.[Код\_сотрудника\_одобрившего\_заявку] = s.[Код\_сотрудника]

INNER JOIN [Хранитель ПРО Кротова].[dbo].[Подразделение] p ON s.[Код\_подразделения] = p.[Код\_подразделения]

INNER JOIN [Хранитель ПРО Кротова].[dbo].[Статус заявки] sz ON z.[Код\_заявки] = sz.[Код\_заявки]

WHERE (@Подразделение IS NULL OR p.[Название\_подразделения] = @Подразделение)

AND (@Статус\_заявки IS NULL OR sz.[Статус\_заявки] = @Статус\_заявки)

END

GO

-- Создание функции для генерации уникального логина

CREATE FUNCTION dbo.GENERATE\_UNIQUE\_LOGIN(@FirstName VARCHAR(30), @LastName VARCHAR(30), @BirthDate DATE)

RETURNS VARCHAR(50)

AS

BEGIN

DECLARE @Login VARCHAR(50);

DECLARE @Counter INT = 1;

DECLARE @BaseLogin VARCHAR(50);

SET @BaseLogin = LOWER(LEFT(@FirstName, 1) + @LastName);

-- Проверяем, существует ли уже логин с таким именем

WHILE EXISTS (SELECT 1 FROM Пользователи WHERE Логин = @BaseLogin)

BEGIN

-- Если логин уже существует, добавляем числовой суффикс

SET @Login = @BaseLogin + CAST(@Counter AS VARCHAR(5));

SET @Counter = @Counter + 1;

END

-- Возвращаем сгенерированный уникальный логин

RETURN @Login;

END;

GO

CREATE PROCEDURE AuthenticateUser

@Login VARCHAR(20),

@Password VARCHAR(20),

@IsSuccessful BIT OUTPUT,

@UserID INT OUTPUT

AS

BEGIN

SET NOCOUNT ON;

-- Проверяем наличие пользователя в таблице "Групповое посещение"

SELECT @UserID = Код\_пользователя

FROM [dbo].[Групповое посещение]

WHERE [Логин] = @Login AND [Пароль] = @Password;

IF @UserID IS NOT NULL

BEGIN

SET @IsSuccessful = 1;

RETURN;

END

-- Если пользователь не найден в таблице "Групповое посещение", проверяем таблицу "Личное посещение"

SELECT @UserID = Код\_пользователя

FROM [dbo].[Личное посещение]

WHERE [Логин] = @Login AND [Пароль] = @Password;

IF @UserID IS NOT NULL

BEGIN

SET @IsSuccessful = 1;

RETURN;

END

-- Если пользователь не найден ни в одной из таблиц, возвращаем ошибку

SET @IsSuccessful = 0;

END;

CREATE PROCEDURE [dbo].[RegisterUser]

@LastName VARCHAR(30),

@FirstName VARCHAR(30),

@MiddleName VARCHAR(30),

@BirthDate DATE,

@PassportSeries VARCHAR(4),

@PassportNumber VARCHAR(6),

@PhoneNumber VARCHAR(20),

@Email VARCHAR(30),

@Login VARCHAR(20),

@Password VARCHAR(20),

@Purpose VARCHAR(40)

AS

BEGIN

SET NOCOUNT ON;

-- Регистрация пользователя в таблице Пользователи

INSERT INTO [Хранитель ПРО Кротова].[dbo].[Пользователи] ([Фамилия], [Имя], [Отчество], [Дата\_рождения], [Серия\_паспорта], [Номер\_паспорта], [Номер\_телефона], [Email])

VALUES (@LastName, @FirstName, @MiddleName, @BirthDate, @PassportSeries, @PassportNumber, @PhoneNumber, @Email);

-- Получение ID нового пользователя

DECLARE @UserID INT;

SET @UserID = SCOPE\_IDENTITY();

-- Регистрация пропуска пользователя в таблице Пропуск

INSERT INTO [Хранитель ПРО Кротова].[dbo].[Пропуск] ([Код\_заявки], [Цель\_посещения])

VALUES (1, @Purpose);

-- Получение ID нового пропуска

DECLARE @PassID INT;

SET @PassID = SCOPE\_IDENTITY();

-- Регистрация личного посещения пользователя в таблице Личное посещение

INSERT INTO [Хранитель ПРО Кротова].[dbo].[Личное посещение] ([Логин], [Пароль], [Назначение], [Код\_пользователя], [Код\_пропуска])

VALUES (@Login, @Password, @Purpose, @UserID, @PassID);

END;

1. **Выполнение резервного копирования БД и восстановление состояния БД на заданную дату.**

Резервное копирование и восстановление я делала через графический интерфейс MS SQL Management Studio. Файлы резервных копий прилагаются в проекте.

1. **Заключение.**

В заключении хотелось бы сказать, что работать над проектом мне действительно понравилось, я узнала и много нового и готова применять знания на практике в дальнейшем. Этот проект позволил мне развить навыки командной работы, аналитического мышления и решения проблем. Я горжусь результатами своей работы и уверена, что они принесут пользу. Благодарю за возможность принять участие в этом проекте и надеюсь на дальнейшее сотрудничество.

1. **Приложения**

Все приложения и документы доступны в проекте.