Министерство транспорта Российской Федерации  
Федеральное агентство железнодорожного транспорта

Федеральное государственное бюджетное образовательное учреждение  
 высшего образования  
«Дальневосточный государственный университет путей сообщения»

Кафедра «Информационные технологии и системы»

Курсовая работа

Создание приложения, работающего с базой данных, реализованной в СУБД PostgreSQL, для обеспечения деятельности предприятия

дисциплина «Языки управления данными»

КР 09.03.04. 07.01.БО221ПИН

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.К. Журин

*(подпись, дата)*

Руководитель

доцент, к.т.н. Р.А. Ешенко

*(подпись, дата)*

Хабаровск 2023

Вариант 21

Разработать прикладное программное обеспечение для ведения реестра имущества университетского городка. В состав имущества входит несколько зданий. В зданиях располагаются аудитории, кафедры, лаборатории, вычислительные центры, деканаты и т. д. Любое помещение университета относится к какому-либо подразделению. Все движимое имущество, находящееся в помещении, состоит на балансе материально ответственного лица. Каждая аудитория закреплена за определённой кафедрой университета, так же в ведении кафедр находятся и лаборатории. По истечении определённого времени имущество, находящееся в помещениях, списывается. Архив списанного имущества не ведётся.

Набор данных к варианту

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Поле** | **Тип** | **Размер** | **Описание** |
| 1 | Kadastr | **Числовой** | 2 | Регистрационный номер здания |
| 2 | BuildingName | **Текстовый** | 20 | Название здания (корпуса) университета |
| 3 | Land | **Числовой** | Авто | Площадь земельного участка |
| 4 | Address | **Текстовый** | 60 | Адрес здания |
| 5 | Year | **Числовой** | 4 | Год постройки |
| 6 | Material | **Текстовый** | 15 | Материал стен здания |
| 7 | Wear | **Числовой** | 2 | Износ в процентах |
| 8 | Flow | **Числовой** | 2 | Число этажей в здании |
| 9 | Picture | **Поле объекта OLE** | Авто | Фотография здания |
| 10 | Comment | **Поле Memo** | Авто | Дополнительные сведения по зданию |
| 11 | HallID | **Текстовый** | 5 | Номер аудитории |
| 12 | Square | **Числовой** | Авто | Площадь аудитории |
| 13 | Windows | **Числовой** | 1 | Количество окон |
| 14 | Heating | **Числовой** | 3 | Число элементов в батареях отопления |
| 15 | Target | **Текстовый** | 15 | Назначение (лекционная, кафедра и т. д.) |
| 16 | Department | **Текстовый** | 15 | Принадлежность к кафедре (подразделение) |
| 17 | Chief | **Текстовый** | 30 | Материально ответственный за аудиторию |
| 18 | DepartmentID | **Числовой** | 3 | Идентификатор кафедры |
| 19 | DepartmentName | **Текстовый** | 15 | Название кафедры |
| 20 | Boss | **Текстовый** | 40 | Заведующий кафедрой |
| 21 | Phone | **Текстовый** | 10 | Телефон кафедры |
| 22 | OfficeDean | **Текстовый** | 30 | Принадлежность кафедры  к деканату |
| 23 | ChiefID | **Числовой** | 3 | Идентификатор материально ответственного лица |
| 24 | AddressChief | **Текстовый** | 60 | Домашний адрес материально ответственного лица |
| 25 | Experience | **Числовой** | 4 | Год его начала работы в университете |
| 26 | UnitID | **Числовой** | 3 | Идентификатор единицы имущества |
| 27 | UnitName | **Текстовый** | 30 | Название единицы  имущества |
| 28 | DateStart | **Дата/время** | Авто | Дата постановки на учет |
| 29 | Cost | **Денежный** | 15 | Стоимость единицы  имущества |
| 30 | CostYear | **Числовой** | 4 | Год переоценки |
| 31 | CostAfter | **Денежный** | 15 | Стоимость после переоценки |
| 32 | Period | **Числовой** | 4 | Срок службы единицы имущества |

Содержание

[Введение 7](#_Toc131443139)

[1. Описание предметной области и проектирование бизнес-процессов 8](#_Toc131443140)

[2. Реализация базы данных 9](#_Toc131443141)

[2.1. Построение ER-модели 9](#_Toc131443142)

[2.2. Преобразование ER-модели в реляционную 11](#_Toc131443143)

[2.3. Построение ERD-модели 13](#_Toc131443144)

[2.4. Создание базы данных на основе ERD-модели 13](#_Toc131443145)

[3. Описание процесса разработки приложения (руководство разработчика) 16](#_Toc131443146)

[3.1. Подключение к Базе данных 16](#_Toc131443147)

[3.2. Отображение полученных результатов 19](#_Toc131443148)

[3.3. Редактирование данных таблицы 21](#_Toc131443149)

[3.4. Реализация запросов 27](#_Toc131443150)

[3.5. Авторизация в БД 30](#_Toc131443151)

[4. Описание интерфейса (руководство пользователя) 33](#_Toc131443152)

[4.1. Знакомство с приложением 33](#_Toc131443153)

[4.2. Работа с приложением 33](#_Toc131443154)

[4.2.1 Авторизация 33](#_Toc131443155)

[4.2.2 Работа с таблицами 33](#_Toc131443156)

[4.2.3 Работа с запросами 35](#_Toc131443157)

[4.2.4 Завершение работы с программой 35](#_Toc131443158)

[Вывод 36](#_Toc131443159)

[Список использованной литературы 37](#_Toc131443160)

Введение

Отдел имущественного обеспечения занимается распределением собственности по университету. Отдел устанавливает какие зданию принадлежат аудитории, закрепляют их за подразделением. Подразделение назначает материально ответственного за аудиторию и распределяет имущество между аудиториями.

Материально ответственные в состоянии просматривать всё своё имущество или имущество конкретной аудитории, искать имущество, нуждающееся в переоценке или списании, а также полную стоимость имущества.

# Описание предметной области и проектирование бизнес-процессов

Программное обеспечение для ведения имущества студенческого городка. В состав городка входят здания, которые характеризуются площадью, названием, адресом, материалом, кадастровым номером, комментарием, числом этажей, годом постройки и фотографией.

В зданиях располагаются аудитории, описываемые площадью, материально ответственным, числом окон и батарей, номером, типом, названием здания (кадастром), кафедрой.

Любое помещение соотносится со своим структурным подразделением, основными параметрами которого являются ФИО и номер телефона заведующего, принадлежностью к директорату.

Непосредственно имущество характеризуется названием, датой поставки, стоимостью за одну единицу, датой следующей переоценки, стоимостью после последней переоценки, сроком службы, числом, износом и номером аудитории, в котором имущество установлено.

Материально ответственный, закреплённый за аудиторией, описывается годом начала работы, ФИО и адресом проживания.

Во время работы с базой данных пользователи заполняют карточки зданий студенческого городка. Затем, к каждому зданию прикрепляются соответствующие аудитории. Другой сотрудник университета создаёт карточки всех кафедр и заполняет их, а затем распределяет аудитории между кафедрами университета.

Кафедры описывают всё имущество в вверенных им аудиториях и вносят о них запись в базу данных.

Материально ответственные в состоянии следить за каждым имуществом в конкретной аудитории или за имуществом, приписанным к ним.

# Реализация базы данных

## Построение ER-модели

Все поля варианта были выделены в следующие сущности: здания, аудитории, кафедры, имущество, материально ответственные. Для полей «адрес», «материал», «город», «деканат» были созданы отдельные сущности-справочники.

Сущность «**Здание**» имеет в себе следующие атрибуты: Кадастр, название здания, площадь здания, город, улица, номер дома, год строительства, материал, этажи, комментарий, фотография. Для атрибутов «улица», «материал», «город» использован соответствующий справочник.

Сущность «**Аудитории**» в своём составе собрала атрибуты: Номер аудитории, название здания, площадь, количество окон, количество элементов в батареях отопления, кафедра, назначение, материально ответственный. Данная Сущность имеет связь с сущностью «Здание» М:1.

Сущность «**Кафедра**» будет иметь связь М:1 с сущностью «Аудитории» и следующие атрибуты: Идентификатор кафедры, название кафедры, заведующий кафедры, телефон кафедры, деканат. Для атрибута «Деканат» использован справочник.

Сущность «**Имущество**» имеет М:1 связь с сущностью «Аудитории» и включает атрибуты: Код имущества, название имущества, дата постановки на учёт, стоимость одного экземпляра, год переоценки, стоимость после переоценки, количество, срок службы, износ.

Сущность «**Материально ответственные**» имеет связь 1:М с сущностью «Аудитории», и связь М:1 со справочниками улиц и городов. Атрибуты: Код материально ответственного, ФИО, город, улица, номер дома, номер квартиры, год начала работы.

Сущность «**Справочник материалов**» имеет связь 1:М с сущностью «Здания» и следующие атрибуты: Код материала, материал.

Сущность «**Справочник деканатов**» имеет связь 1:М с сущностью «Кафедры» и имеет следующие атрибуты: Код деканата, название.

Сущность «**Справочник улиц**» имеет связь 1:М с сущностями «Материально ответственный» и «Здания» и имеет следующие атрибуты: Код улицы, признак адреса, порядок адреса и название.

Сущность «**Справочник городов**» имеет связь 1:М с сущностями «Материально ответственный» и «Здания» и имеет следующие атрибуты: Код города, признак адреса, порядок адреса и название.

Результат построенной ER-модели представлен на рисунке Рисунок 2.1.1.

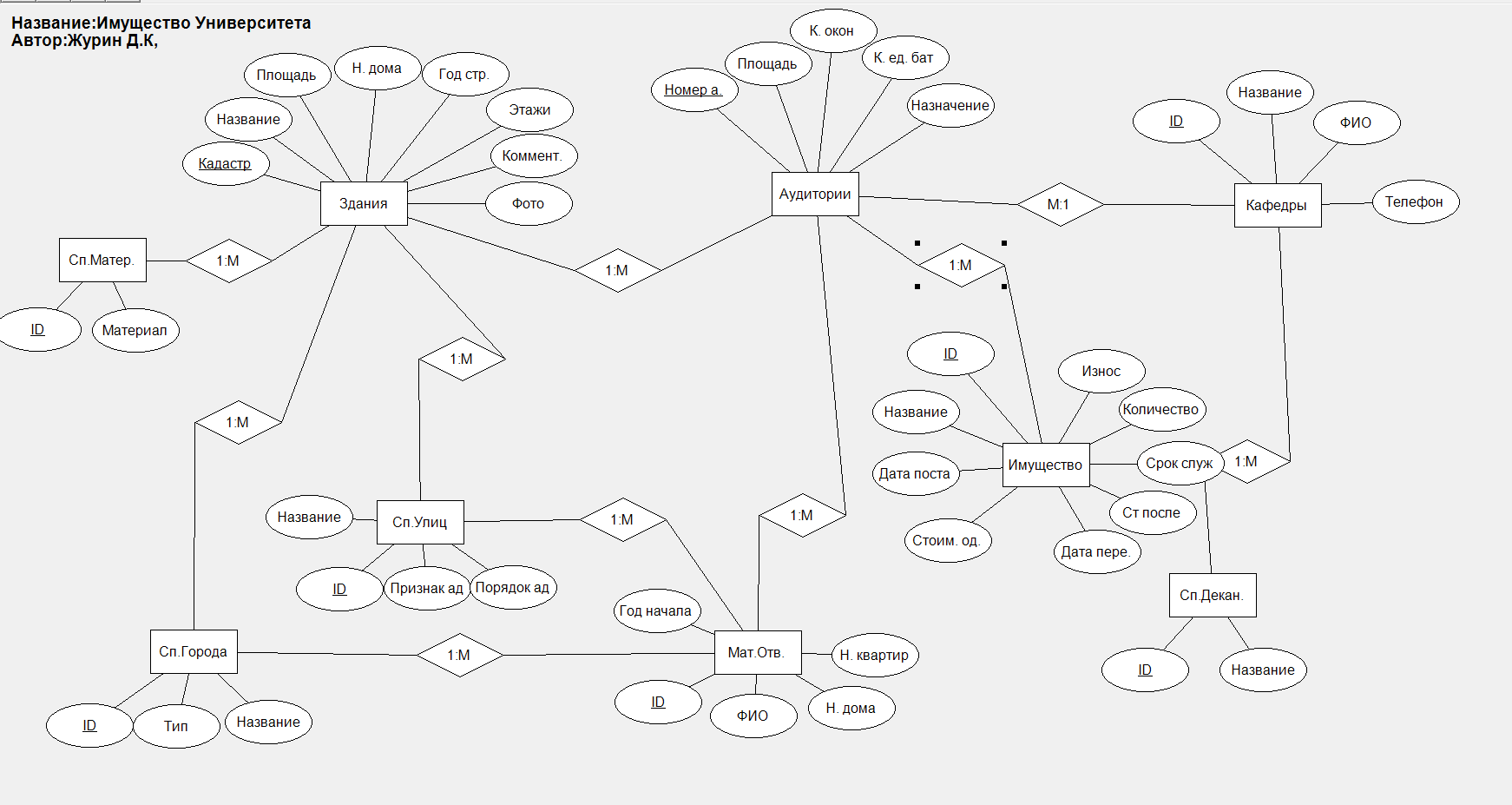


Рисунок 2.1.1 – ER-модель

После составления ER-модели было выполнено приведение до третьей нормальной формы, что подразумевает:

* каждый атрибут сущности атомарен;
* любой атрибут сущности, не входящий в состав первичного ключа, функционально полно зависит от первичного ключа;
* любой неключевой атрибут сущности функционально зависит только от первичного ключа.

Результат приведён на рисунке Рисунок 2.1.2.

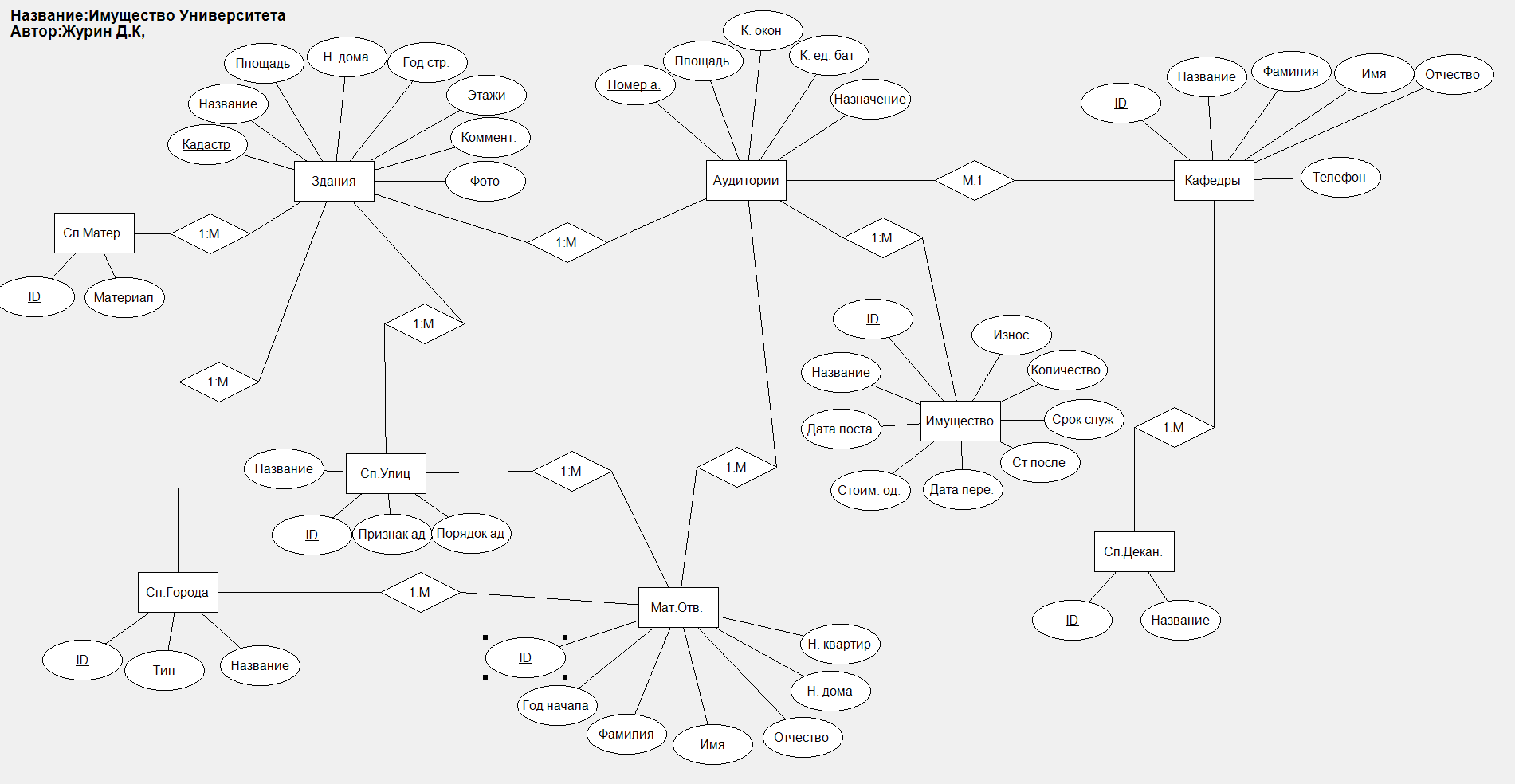


Рисунок 2.1.2 – ER-модель, приведённая к третьей нормальной форме

## Преобразование ER-модели в реляционную

Сущность преобразуется в отдельную таблицу, полями таблицы становятся все атрибуты сущности. После преобразования полученной модели получились следующие таблицы:

Таблица «**Справочник городов»**:

Cities Handbook (ID, Type, Name)

Таблица «**Справочник улиц**»**:**

Streets Handbook (ID, Address attribute, Address order, Name)

Таблица «**Справочник материала**»**:**

Material Handbook (ID, Material)

Таблица «**Здания**»:

Содержит ссылку на внешние ключи – улица, город и тип материала.

Buildings (Cadastre, Name, Square, House number, Year built, Num of floors, Comment, Photo, Material, City, Address)

Таблица «**Аудитории**»:

Содержит внешние ключи – Название здания, кафедра.

Audience (Aud num, Square, Windows num, Battery num, Type, Name of building, Materially responsible, Department)

Таблица «**Имущество**»:

Содержит внешние ключи – Аудитория, материально ответственный

Property (ID, Name, Delivery date, Cost Per One, Reprice Date, Cost after reprice, Lifetime, Amount, Depreciation, Audience)

Таблица «**Материально ответственные**»:

Содержит внешние ключи – Город, улица

Materially Responsible (ID, Start year, Second name, First name, Fathers name, Num of house, Num of flat, Address, City)

Таблица «**Справочник Деканатов**»:

Deans (ID, Name)

Таблица **Department** содержит внешний ключ – деканат

Department (ID, Name, Second name, First name, Fathers name, Deans, Phone)

Для хранения логинов и паролей была создана отдельная таблица **Login**.

Login (Login, Password)

## Построение ERD-модели

С помощью PgAdmin была составлена ERD-модель базы данных.

Также отдельно была добавлена таблица «Login» для хранения логинов и паролей пользователей.

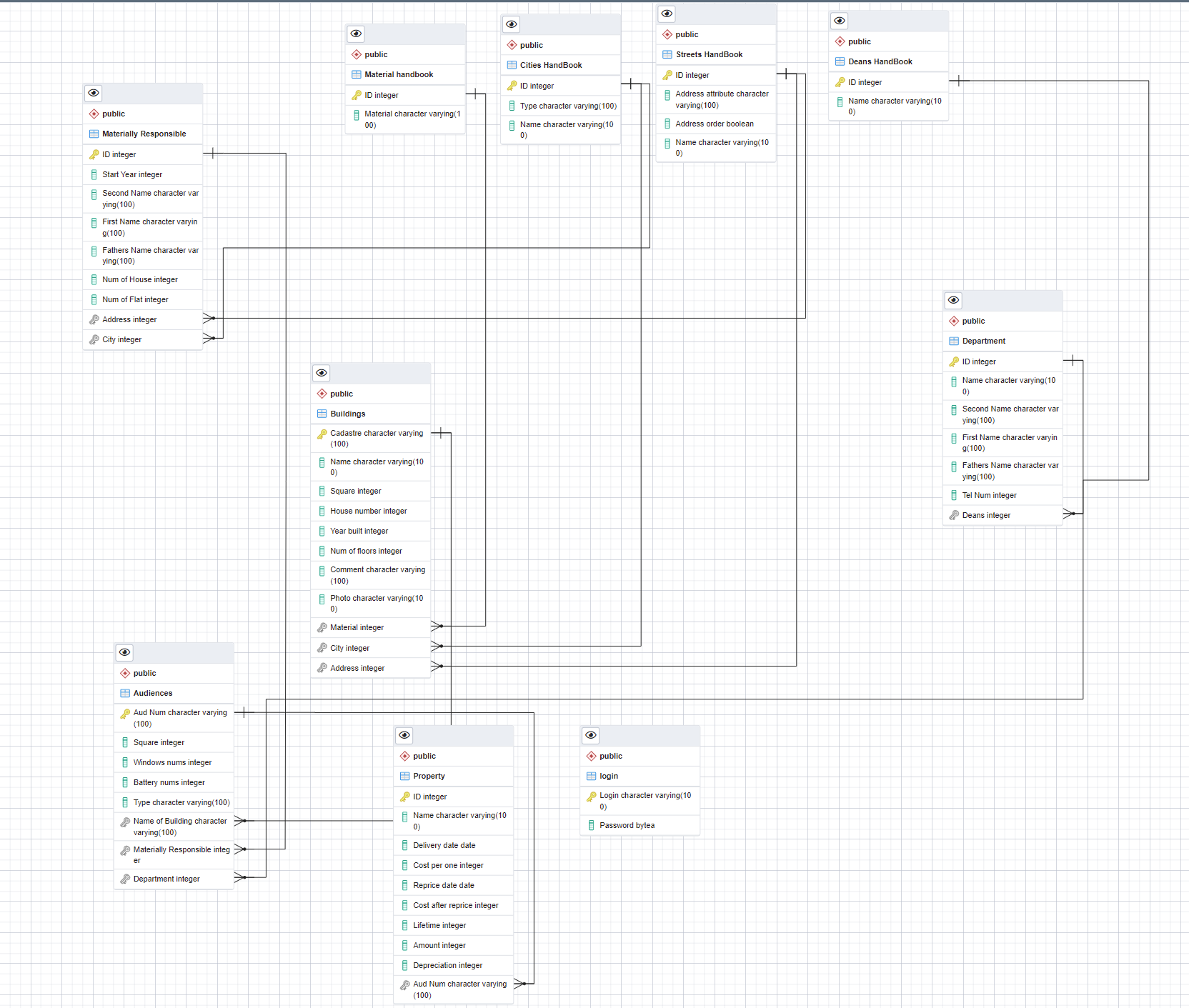


Рисунок 2.3.1 – ERD-модель базы данных

## Создание базы данных на основе ERD-модели

На основе ERD-модели был сгенерирован автоматический SQL код.

Пример кода приведён в листинге Листинг 2.4.1. В нём создаётся таблица «Здания», а также устанавливается связь 1 ко многим со справочником улиц.

Листинг 2.4.1 – Пример создания таблицы и связи

CREATE TABLE IF NOT EXISTS public.Buildings

(

Cadastre character varying(100) NOT NULL,

Name character varying(100),

Square integer NOT NULL,

House\_number integer NOT NULL,

Year\_built integer NOT NULL,

Num\_of\_floors integer NOT NULL,

Comment character varying(100),

Photo character varying(100),

Material integer NOT NULL,

City integer NOT NULL,

Address integer NOT NULL,

PRIMARY KEY (Cadastre)

);

ALTER TABLE public.Buildings

ADD FOREIGN KEY (Address)

REFERENCES public.Streets\_HandBook (ID)

ON UPDATE CASCADE;

Аналогичным образом были созданы все таблицы. Результат приведён на рисунке Рисунок 2.4.2.

Затем все таблицы заполнились с помощью SQL команды INSERT, синтаксис которой выглядит так: INSERT INTO *название таблицы (столбец 1, столбец 2…) VALUES (значение столбца 1, значение столбца 2…).* Пример приведён на рисунке Рисунок 2.4.1.

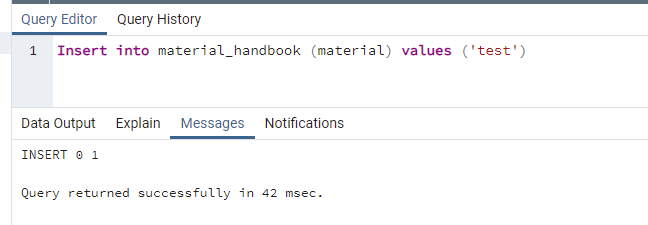


Рисунок 2.4.1 – Вставка записи в таблицу

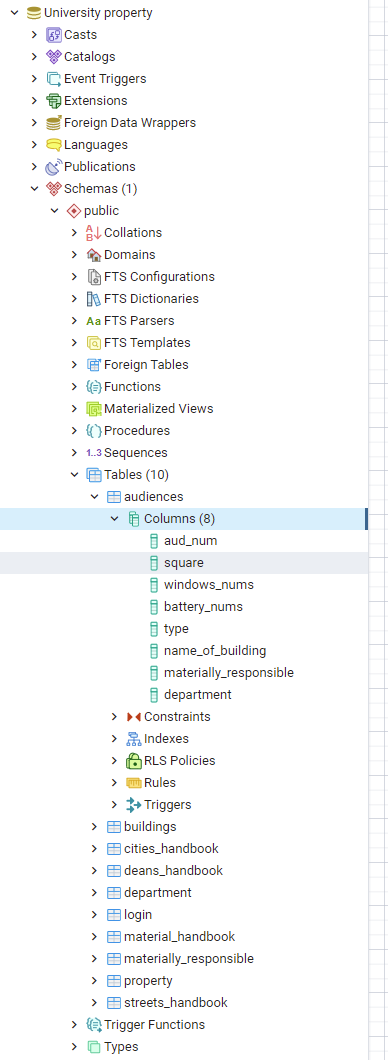


Рисунок 2.4.2 – Таблицы в PgAdmin

# Описание процесса разработки приложения (руководство разработчика)

## Подключение к Базе данных

Подключение к базе данных PostgreSQL в C# возможно с помощью библиотеки Npgsql.

Для этого была создана переменная типа NpgsqlConnection, в конструктор которой отправляется строка подключения, как показано в листинге Листинг 3.1.1.

Листинг 3.1.1 – Создание NpgsqlConnection

static readonly string connString = "Server=localhost;Port=5432;User ID=postgres;Password=123;Database=University property;";

readonly NpgsqlConnection conn = new NpgsqlConnection(connString);

После вызова метода Open у NpgsqlConnection соединение открывается и можно было работать с БД. Однако требовалось работать только в рамках блоков try catch, поскольку осуществляется работа с внешними данными, и, в случае неполадок в канале связи, возможно повреждение базы данных.

Для работы с базой потребовалось её локальное представление – DataSet. Объект DataSet содержит таблицы, которые представлены типом DataTable. Таблица, в свою очередь, состоит из столбцов и строк. Каждый столбец представляет объект DataColumn, а строка - объект DataRow.

Для быстрого считывания информации из PostgreSQL и внесения её в DataSet использовались адаптеры – NpgsqlDataAdapter.

NpgsqlDataAdapter – специальный класс, позволяющий эффективно считывать таблицы из базы данных и работать с ними в рамках DataSet.

Для инициализации адаптера потребовался экземпляр класса NpgsqlCommand, который представляет собой команду SQL. Обычно для инициализации используется SQL запрос SELECT.

Для того, чтобы адаптер заполнил DataSet применяется метод Fill. В рамках данной работы для каждой таблицы был создан свой адаптер.

В листинге Листинг 3.1.2 представлена инициализация соединения с базой данных, а также заполнение DataSet.

Для последующего отображения полученных таблиц использовался класс BindingSource – промежуточный узел, позволяющий связать таблицу с другим объектом.

Листинг 3.1.2 – Чтение из базы данных таблиц

try

{

conn.Open();

buildingAdapter = new NpgsqlDataAdapter(new NpgsqlCommand("Select cadastre AS Кадастр, name AS Название, square as Площадь, " +

"year\_built as \"Год постройки\", num\_of\_floors as \"Число этажей\", comment as Комментарий, photo as Фото, material as Материал, city as Город, address as Адрес, house\_number as \"Номер здания\"" +

" from buildings", conn));

audAdapter = new NpgsqlDataAdapter(new NpgsqlCommand("Select aud\_num as \"Номер аудитории\", square as Площадь, windows\_nums as \"Число окон\", battery\_nums as \"Число батарей\", " +

"type as Назначение, materially\_responsible as \"Материально ответственный\", department as Кафедра, name\_of\_building \"Кадастр здания\" from audiences", conn));

departmentAdapter = new NpgsqlDataAdapter(new NpgsqlCommand("Select id as ID, name as Название, second\_name as \"Фамилия заведующего\", first\_name as \"Имя заведующего\", fathers\_name as \"Отчество заведующего\", " +

"phone as Телефон, deans as \"Деканат или Директорат\" from department", conn));

citiesAdapter = new NpgsqlDataAdapter(new NpgsqlCommand("Select id as ID, type as Тип, name as Название from cities\_handbook", conn));

deansAdapter = new NpgsqlDataAdapter(new NpgsqlCommand("Select id as ID, name as Название from deans\_handbook", conn));

materialAdapter = new NpgsqlDataAdapter(new NpgsqlCommand("Select id as ID, material as Материал from material\_handbook", conn));

materialResAdapter = new NpgsqlDataAdapter(new NpgsqlCommand("Select id as ID, start\_year as \"Год начала работы\", second\_name as \"Фамилия ответственного\", first\_name as \"Имя ответственного\", fathers\_name as \"Отчество ответственного\", " +

"city as Город, address as Адрес, num\_of\_house as \"Номер дома\", num\_of\_flat as \"Номер квартиры\" from materially\_responsible", conn));

Продолжение листинга Листинг 3.1.2

propertyAdapter = new NpgsqlDataAdapter(new NpgsqlCommand("Select id as ID, name as Название, delivery\_date as \"Дата поставки\", cost\_per\_one as \"Стоимость за единицу\", " +

"reprice\_date as \"Дата переоценки\", cost\_after\_reprice as \"Стоимость после переоценки\", lifetime as \"Срок эксплуатации\", " +

"amount as Количество, depreciation as Износ, aud\_num as \"Номер аудитории\" from property", conn));

streetsAdapter = new NpgsqlDataAdapter(new NpgsqlCommand("Select id as ID, address\_attribute as Тип, address\_order as Порядок, name as Название from streets\_handbook", conn));

dataSet = new DataSet();

bs = new BindingSource();

buildingAdapter.Fill(dataSet, "buildings");

audAdapter.Fill(dataSet, "audiences");

departmentAdapter.Fill(dataSet, "department");

citiesAdapter.Fill(dataSet, "cities\_handbook");

deansAdapter.Fill(dataSet, "deans\_handbook");

materialAdapter.Fill(dataSet, "material\_handbook");

materialResAdapter.Fill(dataSet, "materially\_responsible");

propertyAdapter.Fill(dataSet, "property");

streetsAdapter.Fill(dataSet, "streets\_handbook");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

conn.Close();

}

Для дальнейшей корректной работы необходимо подготовить более удобное локальное представление словарей из базы данных.

Словари подготавливаются при загрузке главной формы, а также обновляются после каждого изменения в базе данных. Для подготовки словарей используется класс Dictionary, в котором ключом выступает строка, а значением число. Ключи отдельно хранятся в массиве из List. Словари заполняются на основе необходимых данных из соответствующих таблиц DataSet. Пример заполнения словаря материалов приведён в листинге Листинг 3.1.3. Все собранные словари объединяются в массив.

Листинг 3.1.3 – Заполнения словаря материалов

List<Dictionary<string, int>> dictionaries;

Dictionary<string, string> BuildDic;

List<string>[] keys;

**private void Make\_Dictionaries()**{

keys = new List<string>[7];

for (int i = 0; i < keys.Length; i++)

{

keys[i] = new List<string>();

}

int count = dataSet.Tables["material\_handbook"].Rows.Count;

Dictionary<string,int> materialsDic = new Dictionary<string,int>();

for (int i = 0; i < count; i++)

{

materialsDic.Add((string)dataSet.Tables["material\_handbook"].Rows[i].ItemArray[1],

(int)dataSet.Tables["material\_handbook"].Rows[i].ItemArray[0]);

keys[5].Add((string)dataSet.Tables["material\_handbook"].Rows[i].ItemArray[1]);

}

…

dictionaries = new List<Dictionary<string, int>> { streetsDic, citiesDic, deansDic , matResDic, DepDic, materialsDic };

}

## Отображение полученных результатов

Для отображения результатов была использована технология Windows Forms.

На форму был добавлен элемент управления menuStrip, в котором определены меню «Файл», содержащий кнопку выход, меню «Таблицы», содержащий кнопки для открытия всех таблиц, подменю «Запросы», содержащий кнопки для открытия запросов.

Для отображения непосредственно таблиц на форму был добавлен элемент DataGridView, позволяющий быстро и удобно работать с таблицами.

Для управления записями в таблице на форме были размещены 4 кнопки: «Вставить», «Изменить», «Удалить», «Обновить». Их реализация будет описана позже.

Итоговый вид формы представлен на рисунке Рисунок 3.2.1.

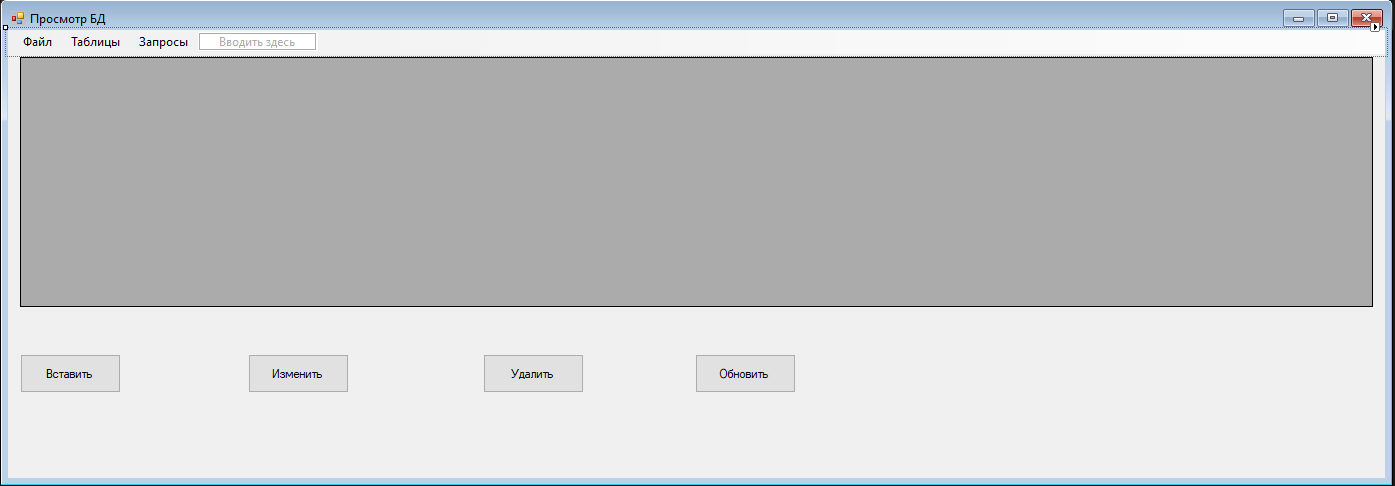


Рисунок 3.2.1 – Главная форма для просмотра таблиц и управления ими

При нажатии на соответствующую таблицу к свойству BindingSource DataSource будет подключена соответствующая таблица в DataSet, а затем к свойству DataSource DataGridView будет подключён BindingSource. Для DataGridView так же указано автоматически подобрать ширину колонок и режим выбора записей установлен на всю строку. Пример кода приведён в листинге Листинг 3.2.1, а пример отображения таблицы на рисунке Рисунок 3.2.2.

Листинг 3.2.1 – Подключение таблицы «Здания» к DataGridView

private void BuildingsToolStripMenuItem\_Click(object sender, EventArgs e)

{

currentTable = "buildings";

bs.DataSource = dataSet.Tables[currentTable];

dataGridView1.DataSource = bs;

dataGridView1.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.AllCells);

dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

}

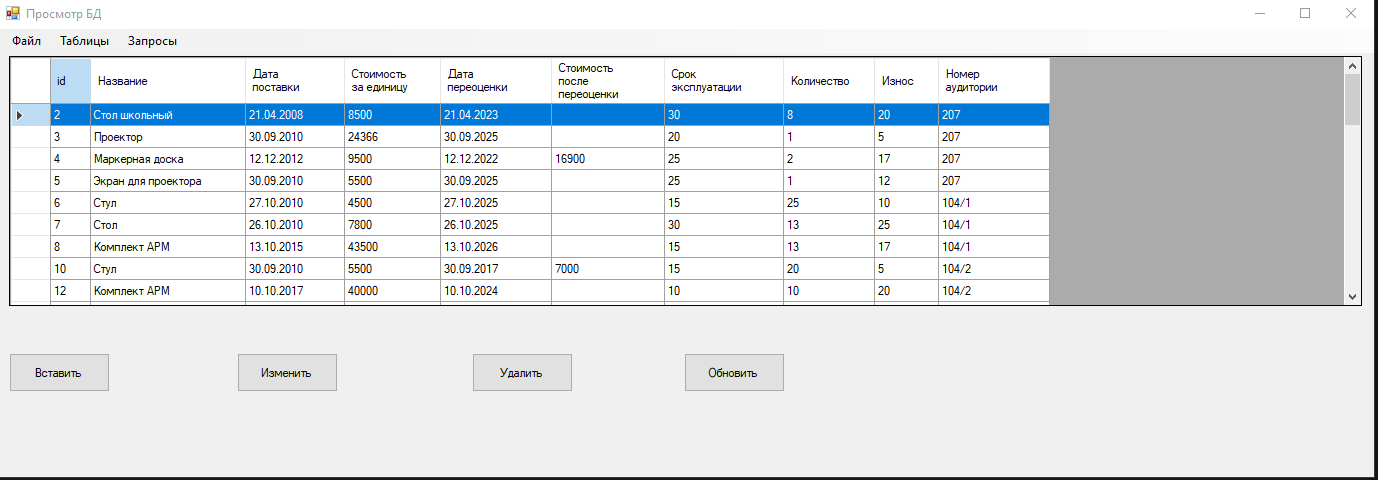


Рисунок 3.2.2 – Отображение таблицы «Имущество»

При выборе меню «Файл» и нажатии на кнопку выход будет вызван метод Close у подключения, а затем закрыта главная форма. Листинг Листинг 3.2.2 иллюстрирует данный алгоритм.

Листинг 3.2.2 – Код закрытия программы

private void ExitToolStripMenuItem\_Click(object sender, EventArgs e)

{

conn.Close();

this.Close();

}

## Редактирование данных таблицы

Для редактирования данных будут использоваться следующие кнопки «Вставить», «Обновить», «Удалить», «Изменить».

При нажатии кнопки «Вставить» будет вызвана форма соответствующей таблицы для вставки новых значений. При вызове формы будет передан параметр, указывающий с какой целью открывается форма – изменение существующей записи или добавление новой. Это позволяет иметь только одну форму для двух кнопок.

Форма имеется вид, который продемонстрирован на рисунке Рисунок 3.3.1.

Для корректной работы формы в неё были переданы BindingSource, NpgsqlConnection, строка с миссией, а также словари.

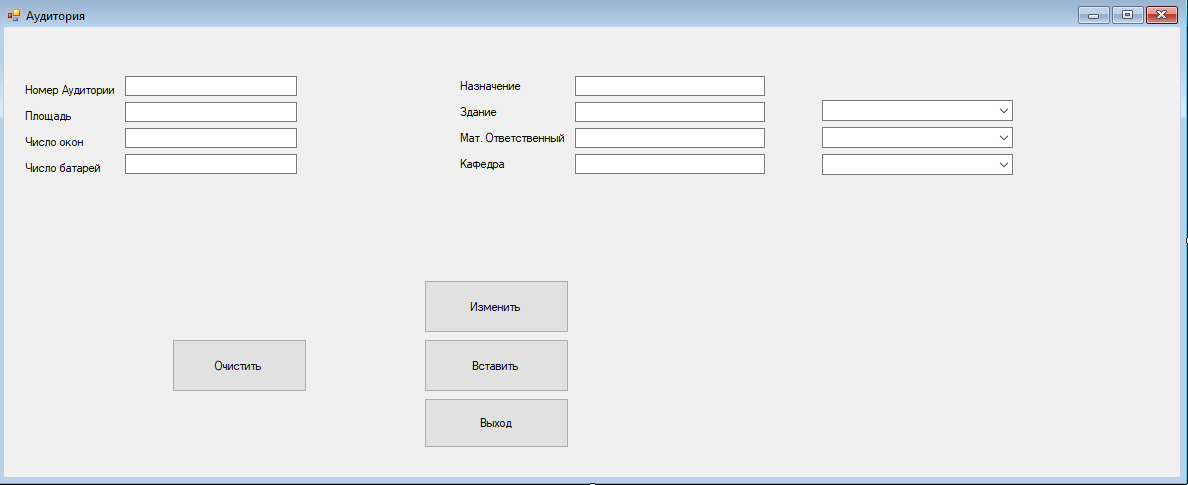


Рисунок 3.3.1 – Форма для внесения изменений или новых записей в таблицу «Аудитории»

TextBox нужны для внесения информации, кнопки для обработки соответствующий действий, а ComboBox для выбора значений с помощью подстановок.

При загрузке формы, в зависимости от цели, по-разному будут проинициализированы элементы формы. Если был выбран режим «Изменить», то TextBox формы заполнятся в соответствии с данными из выбранной на DataGridView записи, а кнопки «Вставить» и «Очистить» станут невидимыми. Иначе поля останутся пустыми, а кнопка «Обновить» станет невидимой. Листинг Листинг 3.3.1 соответствует данному алгоритму.

Листинг 3.3.1 – Инициализация формы

private void IUAud\_Load(object sender, EventArgs e){

if (mission == "Update"){

AudBox.DataBindings.Add("Text", bs, "Номер аудитории");

SquareBox.DataBindings.Add("Text", bs, "Площадь", true, DataSourceUpdateMode.Never);

WinBox.DataBindings.Add("Text", bs, "Число окон", true, DataSourceUpdateMode.Never);

//BatteryBox.DataBindings.Add("Text", bs, "battery\_nums");

BatteryBox.DataBindings.Add("Text", bs, "Число батарей", true, DataSourceUpdateMode.Never);

typeBox.DataBindings.Add("Text", bs, "Назначение");

buildBox.DataBindings.Add("Text", bs, "Кадастр здания");

matResBox.DataBindings.Add("Text", bs, "Материально ответственный", true, DataSourceUpdateMode.Never);

depBox.DataBindings.Add("Text", bs, "Кафедра", true, DataSourceUpdateMode.Never);

button1.Enabled = false; button1.Visible = false;

button3.Enabled = false; button3.Visible = false;}

else if (mission == "Insert"){

button2.Enabled = false; button2.Visible = false; }

PrepairComboboxex();

}

Поля «Здание», «Мат. Ответственный» и «Кафедра» формы заблокированы для ручного редактирования. Их изменение осуществляется через ComboBox, которые перед использованием нужно ещё подготовить.

Метод PrepairComboboxexe добавляет значения ComboBox из словарей. Все ключи заносятся в нужные ComboBox, а при выборе значения в ComboBox автоматически заменяется текст соответствующего TextBox. Пример реализации приведён в листинге Листинг 3.3.2.

Листинг 3.3.2 – Подготовка ComboBox и внесение данных из них в TextBox

private void PrepairComboboxex()

{

comboBox1.Text = "Выберите новое значение";

comboBox2.Text = "Выберите новое значение";

comboBox3.Text = "Выберите новое значение";

comboBox1.Items.AddRange(keys[6].ToArray());

comboBox2.Items.AddRange(keys[3].ToArray());

comboBox3.Items.AddRange(keys[4].ToArray());

}

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

matResBox.Text = (dictionaries[3][comboBox2.Text]).ToString();

}

При нажатии на кнопку «Изменить» выполняется проверка на заполненность всех обязательных полей, а затем из всех TextBox считываются и парсируются в нужные типы данных значения, после чего данные значения подставляются в SQL команду UPDATE, и команда исполняется.

Отдельно стоит отметить, что в случае, если в TextBox соответствующего поля находится пустая строка, то данное значение интерпретируется как null. И именно в таком формате значение отправляется в UPDATE.

Если на каком-либо этапе работы метода произойдёт ошибка – пользователь будет уведомлён об этом. Листинг Листинг 3.3.3 иллюстрирует работу алгоритма.

Листинг 3.3.3 – Пример метода Update

private void button2\_Click(object sender, EventArgs e)

{

string command = $"UPDATE audiences SET square=@p2, windows\_nums =@p3," +

"battery\_nums =@p4, type=@p5, name\_of\_building=@p6, materially\_responsible=@p7, department=@p8 where aud\_num =@p1";

try

{

if (String.IsNullOrEmpty(AudBox.Text) || String.IsNullOrEmpty(SquareBox.Text) || String.IsNullOrEmpty(typeBox.Text)

|| String.IsNullOrEmpty(matResBox.Text) || String.IsNullOrEmpty(depBox.Text))

{

MessageBox.Show("Заполните все обязательные поля");

return;

}

int square = int.Parse(SquareBox.Text);

int? windows\_nums;

if (WinBox.Text == "" || WinBox.Text == " ")

{

windows\_nums = null;

}

else

{

windows\_nums = int.Parse(WinBox.Text);

}

int? battery\_nums;

if (BatteryBox.Text == "" || BatteryBox.Text == " ")

{

battery\_nums = null;

}

else

{

battery\_nums = int.Parse(BatteryBox.Text);

}

int materially\_responsible = int.Parse(matResBox.Text);

int department = int.Parse(depBox.Text);

NpgsqlCommand cmd = new NpgsqlCommand(command, conn);

cmd.Parameters.AddWithValue("@p1", AudBox.Text);

cmd.Parameters.AddWithValue("@p2", square);

if (windows\_nums == null)

{

cmd.Parameters.Add(new NpgsqlParameter("@p3", NpgsqlDbType.Integer));

cmd.Parameters[2].Value = DBNull.Value;

}

else

{

cmd.Parameters.AddWithValue("@p3", windows\_nums);

}

Продолжение листинга Листинг 3.3.3

if (battery\_nums == null)

{

cmd.Parameters.Add(new NpgsqlParameter("@p4", NpgsqlDbType.Integer));

cmd.Parameters[3].Value = DBNull.Value;

}

else

{

cmd.Parameters.AddWithValue("@p4", battery\_nums);

}

cmd.Parameters.AddWithValue("@p5", typeBox.Text);

if (buildBox.Text == "" || buildBox.Text == " ")

{

cmd.Parameters.Add(new NpgsqlParameter("@p6", NpgsqlDbType.Varchar));

cmd.Parameters[5].Value = DBNull.Value;

}

else

{

cmd.Parameters.AddWithValue("@p6", buildBox.Text);

}

cmd.Parameters.AddWithValue("@p7", materially\_responsible);

cmd.Parameters.AddWithValue("@p8", department);

conn.Open();

cmd.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

conn.Close();

}

}

Аналогичным образом работает метод Insert, срабатывающий при нажатии кнопки «Вставить».

При нажатии кнопки «Очистить» все TextBox на форме принимают пустое значение, а при нажатии «Выход» вызывается метод Close() у формы.

Аналогичным образом сделаны все остальные формы для редактирования таблиц.

Для удаления записей при нажатии кнопки «Удалить» будет вызываться метод, считывающий значение первого столбца выбранной на DataGridView строки, и отправляться запрос на удаление записи из таблицы. Листинг Листинг 3.3.4 демонстрирует это.

Листинг 3.3.4 – Код удаления записи

string command = String.Empty;

switch (currentTable){

case "buildings":

command = $"DELETE FROM {currentTable} WHERE cadastre = '{dataGridView1.SelectedRows[0].Cells[0].Value}'";break;

case "audiences":

command = $"DELETE FROM {currentTable} WHERE aud\_num = '{dataGridView1.SelectedRows[0].Cells[0].Value}'";break;

case "department":

command = $"DELETE FROM {currentTable} WHERE id = {dataGridView1.SelectedRows[0].Cells[0].Value}";break;

case "cities\_handbook":

command = $"DELETE FROM {currentTable} WHERE id = {dataGridView1.SelectedRows[0].Cells[0].Value}";break;

case "deans\_handbook":

command = $"DELETE FROM {currentTable} WHERE id = {dataGridView1.SelectedRows[0].Cells[0].Value}";break;

case "material\_handbook":

command = $"DELETE FROM {currentTable} WHERE id = {dataGridView1.SelectedRows[0].Cells[0].Value}";break;

case "materially\_responsible":

command = $"DELETE FROM {currentTable} WHERE id = {dataGridView1.SelectedRows[0].Cells[0].Value}";break;

case "property":

command = $"DELETE FROM {currentTable} WHERE id = {dataGridView1.SelectedRows[0].Cells[0].Value}";break;

case "streets\_handbook":

command = $"DELETE FROM {currentTable} WHERE id = {dataGridView1.SelectedRows[0].Cells[0].Value}";break;

}

try{

conn.Open();

NpgsqlCommand cmd = new NpgsqlCommand(command, conn);

cmd.ExecuteNonQuery();

}

catch (Exception ex){

MessageBox.Show(ex.Message);

}

finally

{

conn.Close();

}

Для реализации кнопки «Обновить» был написан метод, заново подгружающий с помощью адаптеров данные из необходимой таблицы в DataSet, а затем обновляющий все словари. В листинге Листинг 3.3.5 приведён пример.

Листинг 3.3.5 – Код обновления данных

private void Refresh1(){

if (currentTable == String.Empty)

return;

dataSet.Tables[currentTable].Clear();

switch (currentTable)

{

case "buildings":

buildingAdapter.Fill(dataSet, "buildings");

break;

case "audiences":

audAdapter.Fill(dataSet, "audiences");

break;

case "department":

departmentAdapter.Fill(dataSet, "department");

break;

case "cities\_handbook":

citiesAdapter.Fill(dataSet, "cities\_handbook");

break;

case "deans\_handbook":

deansAdapter.Fill(dataSet, "deans\_handbook");

break;

Продолжение листинга Листинг 3.3.5.

case "material\_handbook":

materialAdapter.Fill(dataSet, "material\_handbook");

break;

case "materially\_responsible":

materialResAdapter.Fill(dataSet, "materially\_responsible");

break;

case "property":

propertyAdapter.Fill(dataSet, "property");

break;

case "streets\_handbook":

streetsAdapter.Fill(dataSet, "streets\_handbook");

break;

}

Make\_Dictionaries();

}

## Реализация запросов

При выборе любого из реализованных запросов вызывается форма, представленная на рисунке Рисунок 3.4.1.

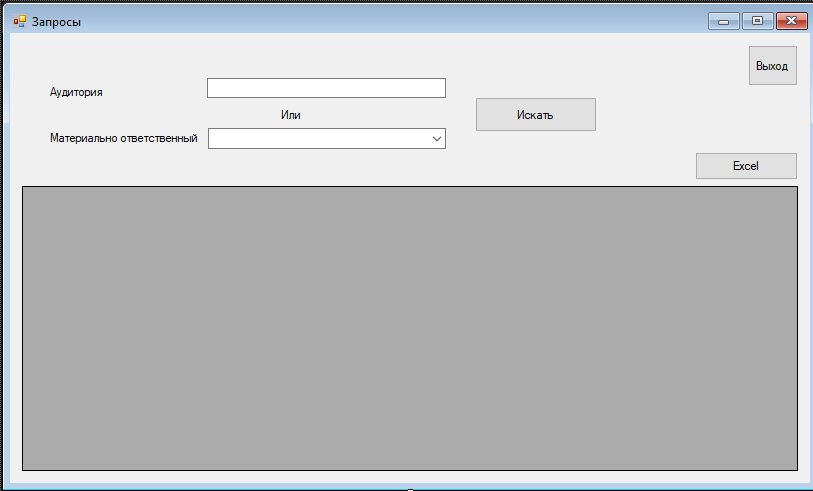


Рисунок 3.4.1 – Общая форма для запросов

Пользователь может выбрать – осуществить поиск по аудитории или по материально ответственному. Одновременно оба критерия использовать нельзя.

Для каждого из случаев формируется собственный SQL запрос на выборку данных.

Пример построения запроса для поиска аудитории и его имущества приведён в листинге Листинг 3.4.1.

Листинг 3.4.1 – Выбор SQL-запроса

if (textBox1.Text == "" || textBox1.Text == " ")

{

s1 = "Select property.id, property.name as Название, property.delivery\_date as \"Дата поставки\", property.cost\_per\_one as \"Стоимость за единицу\", property.reprice\_date as \"Дата переоценки\", " +

"property.cost\_after\_reprice as \"Стоимость после переоценки\", property.lifetime as \"Срок эксплуатации\", property.amount as \"Количество\", property.depreciation as \"Износ\", property.aud\_num as \"Номер аудитории\", materially\_responsible.second\_name as \"Фамилия мат. ответственного\", " +

"materially\_responsible.first\_name as \"Имя мат. ответственного\", materially\_responsible.fathers\_name as \"Отчество мат. ответственного\" " +

"from " +

"((property inner join audiences on property.aud\_num = audiences.aud\_num) " +

"inner join materially\_responsible on materially\_responsible=audiences.materially\_responsible) " +

$"where audiences.materially\_responsible = materially\_responsible.id and audiences.materially\_responsible = {dictionaries[3][comboBox1.Text]}"; //{dictionaries[3][comboBox1.Text]}

}

else if((comboBox1.Text == "" || comboBox1.Text == " ") && textBox1.Text != "Все")

{

s1 = "Select property.id, property.name as Название, property.delivery\_date as \"Дата поставки\", property.cost\_per\_one as \"Стоимость за единицу\", property.reprice\_date as \"Дата переоценки\", " +

"property.cost\_after\_reprice as \"Стоимость после переоценки\", property.lifetime as \"Срок эксплуатации\", property.amount as \"Количество\", property.depreciation as \"Износ\", property.aud\_num as \"Номер аудитории\", materially\_responsible.second\_name as \"Фамилия мат. ответственного\", " +

"materially\_responsible.first\_name as \"Имя мат. ответственного\", materially\_responsible.fathers\_name as \"Отчество мат. ответственного\" " +

"from " +

"((property inner join audiences on property.aud\_num = audiences.aud\_num) " +

"inner join materially\_responsible on materially\_responsible=audiences.materially\_responsible) " +

$"where audiences.aud\_num = \'{textBox1.Text}\' and audiences.materially\_responsible = materially\_responsible.id";

}

Продолжение листинга Листинг 3.4.1

else if(textBox1.Text == "Все")

{

s1 = "Select property.id, property.name as Название, property.delivery\_date as \"Дата поставки\", property.cost\_per\_one as \"Стоимость за единицу\", property.reprice\_date as \"Дата переоценки\", " +

"property.cost\_after\_reprice as \"Стоимость после переоценки\", property.lifetime as \"Срок эксплуатации\", property.amount as \"Количество\", property.depreciation as \"Износ\", property.aud\_num as \"Номер аудитории\", materially\_responsible.second\_name as \"Фамилия мат. ответственного\", " +

"materially\_responsible.first\_name as \"Имя мат. ответственного\", materially\_responsible.fathers\_name as \"Отчество мат. ответственного\" " +

"from " +

"((property inner join audiences on property.aud\_num = audiences.aud\_num) " +

"inner join materially\_responsible on materially\_responsible=audiences.materially\_responsible) " +

$"where audiences.materially\_responsible = materially\_responsible.id";

}

После загрузки данных в зависимости от критерия поиска следует избавиться от лишних данных. Для этого в коде кнопки составлен алгоритм, приведённый в листинге Листинг 3.4.2.

Листинг 3.4.2 – Код кнопки «Искать»

private void Search\_Click(object sender, EventArgs e)

{

if(dataSet.Tables.Count != 0){

dataSet.Tables.Clear();

}

string querryMatres = ChooseStringQuerry(mission);

try

{

conn.Open();

adapter = new NpgsqlDataAdapter(new NpgsqlCommand(querryMatres, conn));

adapter.Fill(dataSet,"querry");

bs.DataSource = dataSet.Tables["querry"];

dataGridView1.DataSource = bs;

if (textBox1.Text != "Все")

{

if (String.IsNullOrEmpty(comboBox1.Text) && dataSet.Tables["querry"].Rows.Count > 0)

comboBox1.Text = dataSet.Tables["querry"].Rows[0].ItemArray[dataSet.Tables["querry"].Rows[0].ItemArray.Length - 3].ToString() +

" " + dataSet.Tables["querry"].Rows[0].ItemArray[dataSet.Tables["querry"].Rows[0].ItemArray.Length - 2].ToString() +

" " + dataSet.Tables["querry"].Rows[0].ItemArray[dataSet.Tables["querry"].Rows[0].ItemArray.Length - 1].ToString();

Продолжение листинга Листинг 3.4.2

if (!String.IsNullOrEmpty(textBox1.Text))

{

dataGridView1.Columns["Фамилия мат. ответственного"].Visible = false;

dataGridView1.Columns["Имя мат. ответственного"].Visible = false;

dataGridView1.Columns["Отчество мат. ответственного"].Visible = false;

}

if (!String.IsNullOrEmpty(textBox1.Text))

dataGridView1.Columns["Номер аудитории"].Visible = false;

else

dataGridView1.Columns["Номер аудитории"].Visible = true;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

conn.Close();

}

}

При нажатии на кнопку «Excel» происходит автоматическая выгрузка результата запроса в Excel.

Остальные запросы реализованы аналогичным образом в пределах этой же формы. Выбор запроса реализован так же, как и выбор «Insert» «Update» у форм изменения данных таблицы.

## Авторизация в БД

Для контроля доступа к базе данных после инициализации главной формы, но до загрузки значений из базы данных и отображения формы, перед пользователем возникнет форма авторизации, в которой пользователю следует указать свой логин и пароль. Логин и хешированный пароль передаются в качестве аргументов SELECT запроса к базе данных к таблице login. В случае если будет найдена запись в базе данных пользователь считается прошедшим аутентификацию и допущенным к работе с БД. Изображение формы приведено на рисунке Рисунок 3.5.1. Листинг Листинг 3.5.1 является кодом кнопки «Войти».

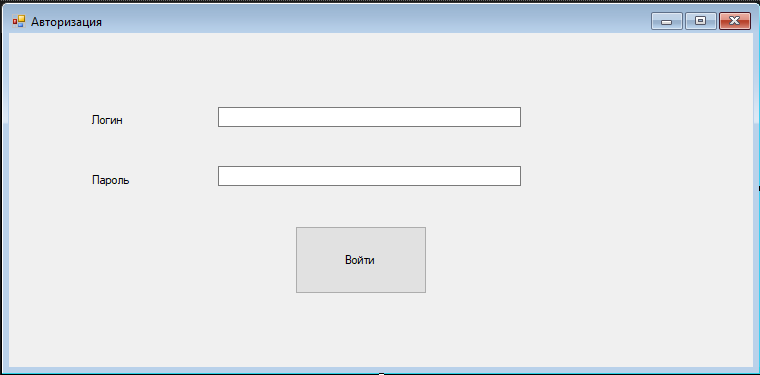


Рисунок 3.5.1 – Форма «Авторизация»

Листинг 3.5.1 – Код кнопки «Войти»

private void Button1\_Click(object sender, EventArgs e)

{

NpgsqlConnection conn = new NpgsqlConnection(connString);

NpgsqlDataAdapter adapter;

DataTable dt = new DataTable();

NpgsqlCommand cmd = new NpgsqlCommand("Select \* from login where login=@p1 and password=@p2", conn);

SHA256 hash = SHA256.Create();

byte[] password = hash.ComputeHash(Encoding.UTF8.GetBytes(textBox2.Text));

cmd.Parameters.AddWithValue("@p1", textBox1.Text);

cmd.Parameters.AddWithValue("@p2", password);

try

{

conn.Open();

adapter = new NpgsqlDataAdapter(cmd);

adapter.Fill(dt);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

conn.Close();

}

Продолжение листинга Листинг 3.5.1

if (dt.Rows.Count > 0)

{

MessageBox.Show("Авторизация пройдена успешно");

this.DialogResult = DialogResult.OK;

Close();

}

else

{

MessageBox.Show("Неверный логин или пароль.");

}

}

# Описание интерфейса (руководство пользователя)

## Знакомство с приложением

Приложение «Просмотр БД» позволяет сотрудникам университета выполнять следующие задачи:

* добавлять и изменять информацию о зданиях учебного заведения;
* добавлять и изменять информацию о аудиториях в зданиях;
* добавлять и удалять информацию об имуществе университета;
* закреплять имущество за аудиторией;
* прикреплять к аудитории материально ответственного;
* закреплять аудитории за кафедрой;
* закреплять кафедру за деканатом.

С помощью программного обеспечения можно автоматизировать процесс сопровождения имущества в университете.

## Работа с приложением

### Авторизация

При запуске программы открывается окно авторизации. Для авторизации необходимо ввести свой логин и пароль. В случае если данные будут введены корректно, пользователь попадёт в основную форму. Иначе пользователь получит уведомление о неверном логине или пароле.

### Работа с таблицами

С помощью ленты пользователь в состоянии выбрать какую таблицу он в данный момент желает просмотреть. Для того, чтобы выбрать таблицу необходимо в ленте выбрать меню «Таблицы», в меню выбрать соответствующую таблицу.

После этого на главной форме отобразится интересующая пользователя таблица.

Для того, чтобы внести новую запись пользователю необходимо нажать кнопку «Вставить». После чего перед пользователем откроется новая форма с пустыми текстовыми полями. В них необходимо внести информацию о новой записи. Поля, которые закрашены серым нельзя редактировать. Для некоторых полей предусмотрены подстановки: служебная информация отобразится в сером поле, а сам выбор пользователю необходимо совершить с помощью выпадающего списка рядом с серым полем.

Для подтверждения ввода необходимо нажать на кнопку «Вставить».

Если пользователь ввёл недействительное значение, например слово вместо числа, то при попытке создать запись возникнет исключение и пользователь будет уведомлён о некорректном значение. Аналогично, если пользователь не заполнит обязательные поля, возникнет исключение.

Для выхода из формы необходимо нажать на кнопку «Выход».

При нажатии на кнопку «Очистить» все поля будут очищены.

Для того, чтобы обновить данные уже существующей записи необходимо выбрать запись в таблице и нажать на кнопку «Изменить». После чего перед пользователем возникнет новая форма, где в полях будут подставлены все данные. Изменяя поля, можно изменить сами данные в таблице. Поля, которые закрашены серым нельзя редактировать. Для некоторый полей предусмотрены подстановки: служебная информация отобразится в сером поле, а сам выбор пользователю необходимо совершить с помощью выпадающего списка рядом с серым полем.

Для подтверждения обновления необходимо нажать на кнопку «Изменить».

Аналогично внесению новой записи, если пользователь ввёл недействительное значение, например слово вместо числа, то при попытке создать запись возникнет исключение и пользователь будет уведомлён о некорректном значение. Если пользователь не заполнит обязательные поля, возникнет исключение.

Для удаления записи пользователю необходимо в таблице выбрать запись и нажать на кнопку «Удалить».

При нажатии на кнопку «Обновить», программа обновит текущую таблицу.

### Работа с запросами

Для того, чтобы перейти к запросам пользователю необходимо на ленте выбрать меню «Запросы» и среди них нажать на интересующий запрос.

В открывшейся форме ввести либо аудиторию поиска, либо выбрать материально ответственного для поиска и нажать «Искать». Результат поиска отобразится в таблице ниже.

Для вывода результата в Excel необходимо нажать на кнопку «Excel».

Для того, чтобы выйти из формы «Запросы», необходимо нажать на кнопку «Выход»

### Завершение работы с программой

Для завершения работы с программой пользователю необходимо закрыть все второстепенные формы, затем в главной форме выбрать на ленте меню «Файл» и нажать «Выход».

Вывод

В ходе выполнения курсовой работы были выполнены следующие задачи:

* описана деятельность по контролю и сопровождению имущества университета;
* создано программное обеспечение для взаимодействия с базой данных PostgreSQL;
* была предусмотрена возможность передачи данных в MS Excel;
* был приобретён опыт по работе с базой данных.

Список использованной литературы

1. Npgsql Basic Usage. – Текст : электронный // npgsql : [сайт]. – URL: https://www.npgsql.org/doc/ (дата обращения: 01.04.2023).
2. Документация к PostgreSQL 15.2. – Текст : электронный // postgrespro : [сайт]. – URL: https://postgrespro.ru/docs/postgresql/15/index (дата обращения: 01.04.2023).
3. Моргунов, Е. П. PostgreSQL. Основы языка SQL: учеб. пособие / Е. П. Моргунов; под ред. Е. В. Рогова, П. В. Лузанова. – СПб.: БХВ-Петербург, 2018. – 336 с.: ил.