**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА» (РУТ(МИИТ))**

**Институт транспортной техники и систем управления**

**Кафедра «Управление и защита информации»**

КУРСОВОЙ ПРОЕКТ

по дисциплине

**«Основы построения защищенных баз данных»**

**на тему  
«**Изучение методов проектирования реляционных баз данных “Контора адвоката”**»**

Выполнил: ст. гр. ТКИ–511  
Меркулов Д.А.  
Проверил: Васильева М.А.

Москва 2021

**ОГЛАВЛЕНИЕ**

[ЦЕЛЬ КУРСОВОГО ПРОЕКТА 3](#_Toc89518741)

[ЗАДАНИЕ НА КУРСОВОВОЙ ПРОЕКТ 3](#_Toc89518742)

[ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ 4](#_Toc89518743)

[**Инфологическое проектирование** 4](#_Toc89518744)

[**1.1** **Анализ предметной области** 4](#_Toc89518745)

[**1.2** **Анализ информационных задач и круга пользователей системы** 6](#_Toc89518746)

[**Физическое проектирование БД** 7](#_Toc89518747)

[**2.1 Разработка классов для создание таблиц в базе данных** 7](#_Toc89518748)

[**2.2 Разработка функций на добавление данных в таблицы** 11](#_Toc89518749)

[**2.3 Разработка необходимых представлений (view) для красивого отображения данных в браузере, и вывода данных из БД.** 12](#_Toc89518750)

[**3. Демонстрация работы веб-приложение** 14](#_Toc89518751)

[ВЫВОД 17](#_Toc89518752)

[Список литературы 18](#_Toc89518753)

# ЦЕЛЬ КУРСОВОГО ПРОЕКТА

Целью курсового проекта является изучение методов и закрепление знаний в проектировании реляционных баз данных (РБД) с помощью ORM (Object-Relational Mapping, объектно-реляционное отображение, или преобразование) на выбранном языке программирования.

# ЗАДАНИЕ НА КУРСОВОВОЙ ПРОЕКТ

В данном курсовом проекте ставится задача разработать РБД в ORM по заданной теме «Контора адвоката». Проектирование РБД проводится с помощью метода «Сущность-связь». Проверка построенной модели РБД осуществляется с помощью метода нормализации отношений.

БД «Контора адвоката»

Описание предметной области. БД создается для информационного обслуживания сотрудников конторы. Контора оказывает юридические услуги (ведение дела в суде, консультация) по разным делам (гражданские, уголовные). Каждый адвокат специализируется в одной области (жилищные, семейные, убийства и т.д.). На каждое обращение в контору заводится Дело (№, содержание, адвокат, клиент).

Функционал приложения:

1. Форма записи клиентов на прием в адвокатскую контору, добавление нового клиента в таблицу БД;
2. Создание нового дела;
3. Форма для получения данных о записи клиента по его фамилии и номеру регистрации дела;
4. Вывод данных на экран.

# ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

## **Инфологическое проектирование**

## **Анализ предметной области**

База данных создаётся для информационного обслуживания клиентов и адвокатов. Контора оказывает юридические услуги (ведение дела в суде, консультация) по разным делам (гражданские, уголовные).

Выделим базовые сущности этой предметной области:

* Адвокат – номер, ФИО, номер области в которой он специализируется, занятость;
* Область специализации – номер, описание области;
* Клиенты – номер, ФИО, паспорт, номер телефона;
* Тип дела – номер, описание (уголовные, гражданские);
* Дела – номер, название, описание, дата, номер клиента, номер адвоката, номер услуги;
* Услуги – номер, тип услуги, номер области специализации, номер типа дела, цена.
* Регистрация – номер клиента, номер адвоката, номер типа услуги, номер типа дела, дата.

ER-диаграмма БД, выполненная в среде визуального построения диаграмм (Visual Paradigm, 2020) приведена на Рисунке 1. На данной ER-диаграмме присутствуют связи многие - ко - многим.

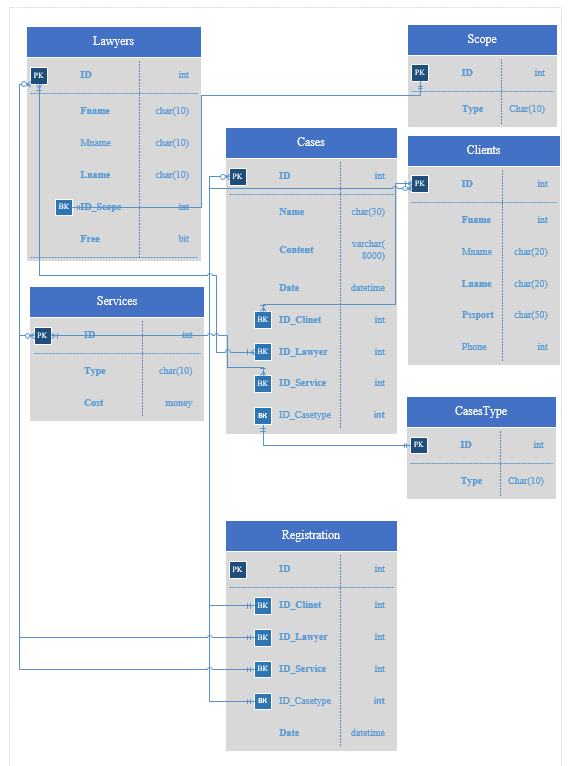


Рисунок 1 ‑ ER-диаграмма

В полученной ER-диаграмме присутствует связь многие - ко - многим, поэтому вводится специальная вспомогательная сущность, которая является соединением первичных ключей соответствующих сущностей. Таким образом разбиваются все связи типа n: m.

## **Анализ информационных задач и круга пользователей системы**

Система создаётся для обслуживания следующих групп пользователей:

* Администраторов конторы;
* Клиенты;

Определим границы информационной поддержки пользователей:

**1) Функциональные возможности:**

* ведение БД (запись, чтение, модификация, удаление в архив);
* обеспечение логической непротиворечивости БД;
* обеспечение защиты данных от несанкционированного или случайного доступа (определение прав доступа);
* реализация наиболее часто встречающихся запросов в готовом виде.

**2) Готовый функционал:**

* панель администратора, где реализовано ведение БД (запись, чтение, модификация, удаление в архив), а также предусмотрена логика непротиворечивости БД;
* для клиента – форма записи, форма проверки записи, где реализовано обеспечение защиты данных от несанкционированного или случайного доступа (определение прав доступа), а также реализация наиболее часто встречающихся запросов в готовом виде.

## **Физическое проектирование БД**

## **2.1 Разработка классов для создание таблиц в базе данных**

БД “Контора адвоката” проектируется с помощью встроенного ORM в библиотеку Django языка программирования Python.

Создание классов базы данных *«*Контора адвоката*»:*

from django.db import models

class CasesType(models.Model):

type = models.CharField(max\_length=255, unique=True, verbose\_name='Тип дела')

def \_\_str\_\_(self):

return self.type

class Meta:

verbose\_name = 'Тип дела'

verbose\_name\_plural = 'Тип дела'

ordering = ['id']

class Scope(models.Model):

type = models.CharField(max\_length=255, unique=True, verbose\_name='Cпециализация')

def \_\_str\_\_(self):

return self.type

class Meta:

verbose\_name = 'Специализация'

verbose\_name\_plural = 'Специализации'

ordering = ['id']

class Clients(models.Model):

f\_name = models.CharField(max\_length=20, verbose\_name='Имя')

m\_name = models.CharField(max\_length=20, null=True, verbose\_name='Отчество')

l\_name = models.CharField(max\_length=20, verbose\_name='Фамилия')

passport = models.CharField(max\_length=50, unique=True, verbose\_name='Серия и номер паспорта')

tel = models.IntegerField(verbose\_name='Номер телефона')

def \_\_str\_\_(self):

return f'{str(self.f\_name)} {str(self.m\_name)}'

class Meta:

verbose\_name = 'Клиент'

verbose\_name\_plural = 'Клиенты'

ordering = ['l\_name']

class Lawyers(models.Model):

f\_name = models.CharField(max\_length=20, verbose\_name='Имя')

m\_name = models.CharField(max\_length=20, null=True, verbose\_name='Отчество')

l\_name = models.CharField(max\_length=20, verbose\_name='Фамилия')

id\_scope = models.ForeignKey(Scope, on\_delete=models.PROTECT, verbose\_name='Специализация')

free = models.BooleanField(default=False, verbose\_name='Занятость')

def \_\_str\_\_(self):

return f'{str(self.f\_name)} {str(self.m\_name)}'

class Meta:

verbose\_name = 'Адвокат'

verbose\_name\_plural = 'Адвокаты'

ordering = ['l\_name']

class Services(models.Model):

type = models.CharField(max\_length=255, verbose\_name='Услуга')

cost = models.IntegerField(verbose\_name='Цена')

def \_\_str\_\_(self):

return self.type

class Meta:

verbose\_name = 'Услуга'

verbose\_name\_plural = 'Услуги'

ordering = ['id']

class Cases(models.Model):

name = models.CharField(max\_length=30, verbose\_name='Название')

content = models.TextField(blank=True, verbose\_name='Материалы дела')

date = models.DateTimeField(auto\_now\_add=True, verbose\_name='Дата создания')

id\_client = models.ForeignKey(Clients, on\_delete=models.PROTECT, verbose\_name='Клиент')

id\_lawyer = models.ForeignKey(Lawyers, on\_delete=models.PROTECT, verbose\_name='Адвокат')

id\_service = models.ForeignKey(Services, on\_delete=models.PROTECT, verbose\_name='Услуга')

id\_case\_type = models.ForeignKey(CasesType, on\_delete=models.PROTECT, verbose\_name='Тип дела')

def \_\_str\_\_(self):

return self.name

class Meta:

verbose\_name = 'Дело'

verbose\_name\_plural = 'Дела'

ordering = ['name']

class Registration(models.Model):

id\_client = models.ForeignKey(Clients, on\_delete=models.PROTECT, verbose\_name='Клиент')

id\_lawyer = models.ForeignKey(Lawyers, on\_delete=models.PROTECT, verbose\_name='Адвокат')

id\_service = models.ForeignKey(Services, on\_delete=models.PROTECT, verbose\_name='Услуга')

id\_case\_type = models.ForeignKey(CasesType, on\_delete=models.PROTECT, verbose\_name='Тип дела')

date = models.DateTimeField(verbose\_name='Дата записи')

def \_\_str\_\_(self):

return self.id\_client

class Meta:

verbose\_name = 'Регистрация'

verbose\_name\_plural = 'Регистрация'

ordering = ['date']

## **2.2 Разработка функций на добавление данных в таблицы**

def add\_client(f\_name, m\_name, l\_name, passport, tel):

client = Clients(f\_name=f\_name, m\_name=m\_name, l\_name=l\_name, passport=passport, tel=tel)

client.save()

return client

def add\_case(name, content, id\_client, id\_lawyer, id\_service, id\_case\_type):

case = Cases(name=name, content=content, id\_client=id\_client, id\_lawyer=id\_lawyer, id\_service=id\_service,

id\_case\_type=id\_case\_type)

case.save()

return case.id

def registration(id\_client, id\_lawyer, id\_service, id\_case\_type, date):

reg = Registration(id\_client=id\_client, id\_lawyer=id\_lawyer, id\_service=id\_service,

id\_case\_type=id\_case\_type, date=date)

reg.save()

return reg.id

## **2.3 Разработка необходимых представлений (view) для красивого отображения данных в браузере, и вывода данных из БД.**

def client(request):

menu = [{'title': 'Записаться', 'url': 'client', 'active': True},

{'title': 'Найти запись', 'url': 'search', 'active': False}]

if request.method == 'POST':

request\_form = request.POST

reg\_id, name, date = check\_form(request\_form)

return thanks(request, reg\_id, name, date)

else:

cases\_type = CasesType.objects.all()

scope = Scope.objects.all()

services = Services.objects.all()

context = {

'menu': menu,

'cases': cases\_type,

'scope': scope,

'services': services,

}

return render(request, 'lawyer/client.html', context=context)

def thanks(request, reg\_id, name, date):

menu = [{'title': 'Записаться', 'url': 'client', 'active': False},

{'title': 'Найти запись', 'url': 'search', 'active': False}]

context = {

'menu': menu,

'reg': reg\_id,

'name': name,

'date': date

}

return render(request, 'lawyer/thanks.html', context=context)

def search(request):

menu = [{'title': 'Записаться', 'url': 'client', 'active': False},

{'title': 'Найти запись', 'url': 'search', 'active': True}]

case = False

if request.method == 'POST':

search\_form = request.POST

name = search\_form['name'].lower()

try:

reg = Registration.objects.get(pk=int(search\_form['num']))

if name in reg.id\_client.f\_name.lower() or name in reg.id\_client.l\_name.lower() or \

name in reg.id\_client.m\_name.lower():

case = [reg.id, reg.id\_client.f\_name + ' ' + reg.id\_client.l\_name, reg.id\_service.type,

reg.id\_case\_type.type, reg.id\_service.cost, reg.date]

except:

pass

context = {

'menu': menu,

'case': case,

}

return render(request, 'lawyer/search.html', context=context)

## **3. Демонстрация работы веб-приложение**

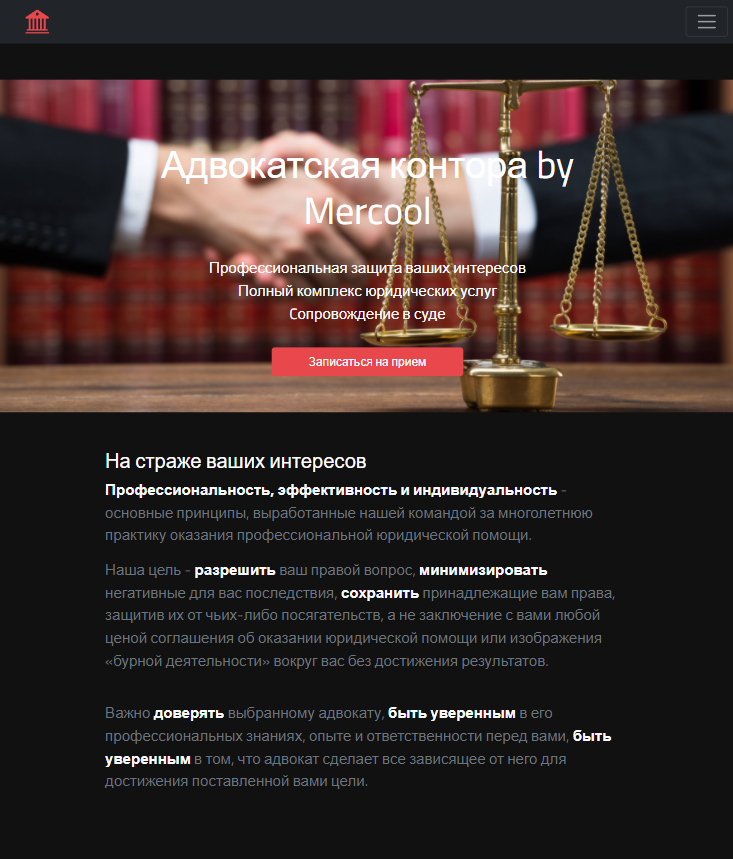


Рисунок 3.1 – Главная страница

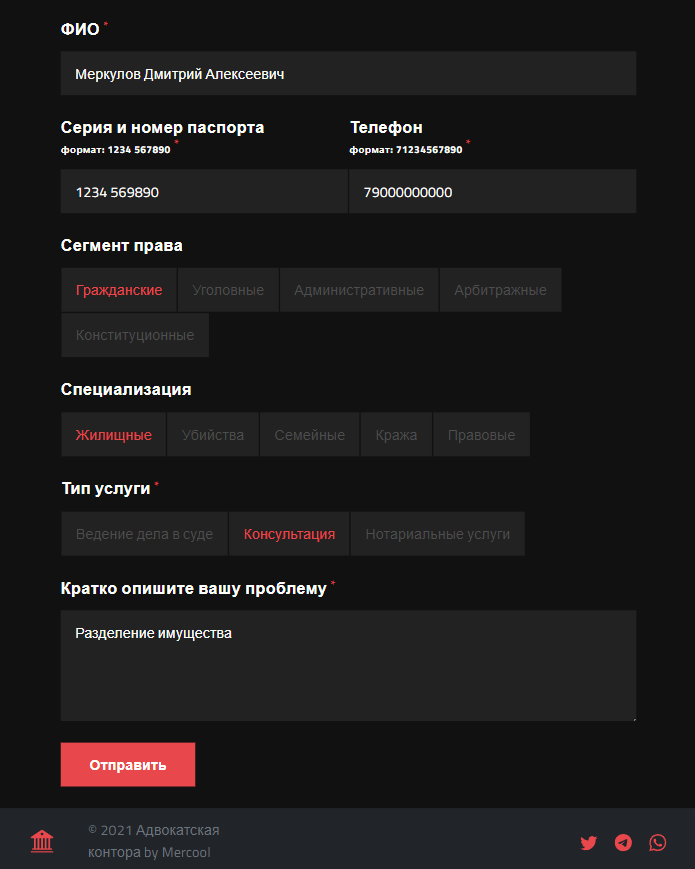


Рисунок 3.2 – Форма записи клиента

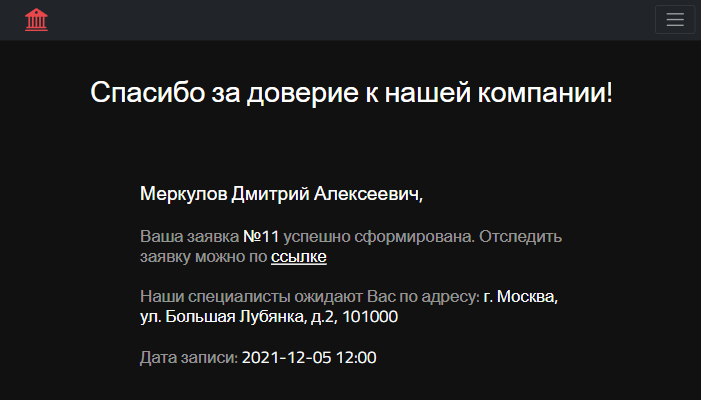


Рисунок 3.3 Подтверждение записи

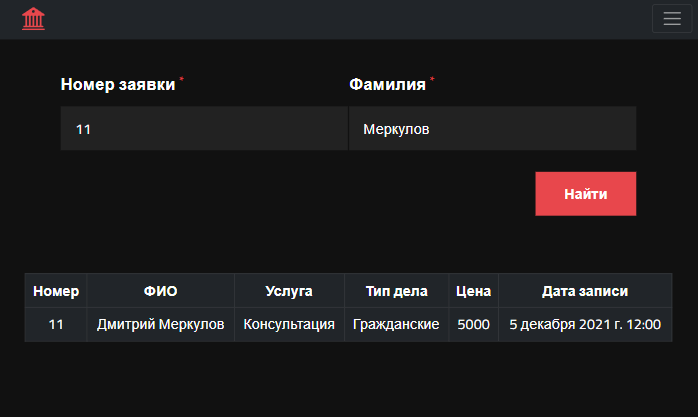


Рисунок 3.4 Поиск записи по номеру и фамилии

# ВЫВОД

Мы изучили методы и закрепили знания в проектировании реляционных баз данных (РБД) с помощью ORM (Object-Relational Mapping) с помощью библиотеки Django на объектно-ориентированном языке программирования Python, также разработали собственном, защищённое веб-приложение для записи на прием в адвокатскую контору, что значительно упрощает работу с клиентами.

# Список литературы

Visual Paradigm. (2020). *Entity Relation*. (@2020 by Visual Paradigm) Retrieved 2 2, 2020, from Visual Paradigm Online: https://online.visual-paradigm.com/app/diagrams/#diagram:proj=0&type=ERDiagram

*ГОСТ 7.32-2017. Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления.* (б.д.). Получено 24 Март 2020 г., из Официальный сайт компании «КонсультантПлюс»: http://www.consultant.ru/document/cons\_doc\_LAW\_292293/

Учебно-методическое пособие для бакалавров по направлению «Управление в технических системах» профиля «Управление и информатика в технических системах», а также специалистов по специальности «Компьютерная безопасность» специализации «Информационная безопасность объектов информатизации на базе компьютерных систем» - М.А. Васильева, Е. П. Балакина, К. М. Филипченко Изд. № 289-20

Документация Python Django: https://djangodoc.ru/3.1/