МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«ИРКУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ИГУ»)

Институт математики
и информационных технологий

Отчет. «Лабораторная работа №3».

Студента 3 курса группы 02321-ДБ
направления 01.03.02 «Прикладная математика и информатика»
Саблина Михаила Александровича

Научный руководитель:
Черкашин Евгений Александрович

Оценка
_____

Иркутск – 2023

Содержание

Оглавление

План работы

Задание: на основании заданной структуры предметной области студент выполняет следующие действия:

1. Создание Django проекта для реализации сайта.
2. Создание моделей для соединения таблиц базы данных, реализованных с использованием PostgreSQL и PG Admin 4, и Django проекта.
3. Вывод данных таблиц и отчетов на сайте.

Определение модели

Больница

Платный прием пациентов проводится врачами разных специальностей (хирург, терапевт, кардиолог, офтальмолог и т.д.). Информация о враче (ФИО, специальность, сумма за прием, процент отчисления). Информация о пациенте (фамилия, имя, отчество, дата рождения, адрес).
Накапливается информация о приемах, где хранится дата приема.
Пациент оплачивает за прием некоторую сумму, которая устанавливается персонально для каждого врача. За каждый прием врачу отчисляется фиксированный процент от стоимости приема. Выходные документы:

- Список квитанций об оплате приема определенного пациента за определенный период времени в порядке убывания дат. В квитанции указывается информация о стоимости приема.
- Ведомость на оплату врачей за определенный период. Размер начисляемой врачу заработной платы за каждый прием вычисляется по формуле:
  Зарплата = Стоимость приема · Процент отчисления на зарплату.
  Из этой суммы вычитается подоходный налог, составляющий 13% от начисленной зарплаты.

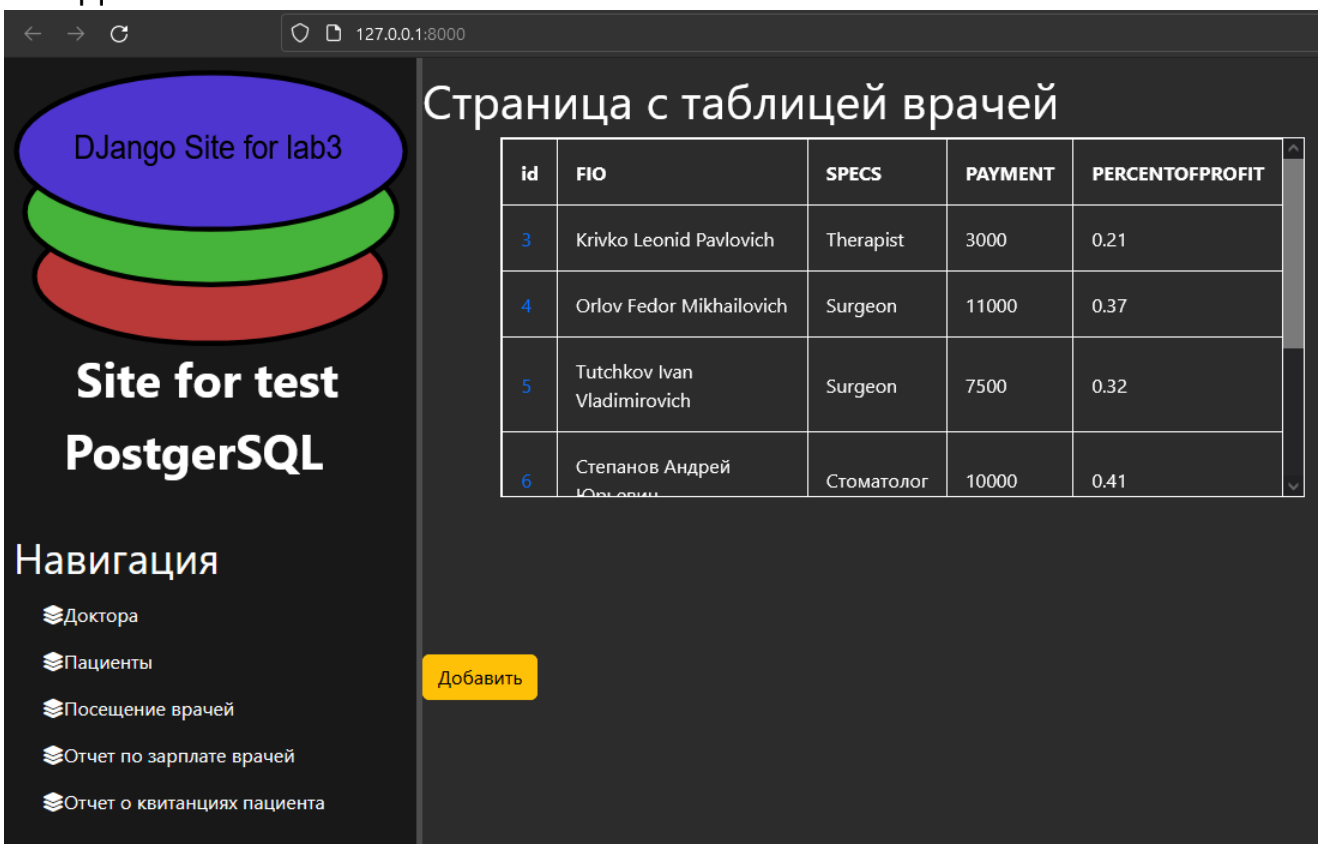
Создание Django проекта. Оформление сайта.

Создадим первую страницу и html-файл, который будет основой для остальных страниц.

```
{% load static %}
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-
scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>{% block title %}{% endblock %}</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstra
p.min.css">
    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v6.2.1/css/all.css">
    <link rel="stylesheet" href="{% static
'static/main/css/main.css'%}">
```

```
</head>
<body>
    <aside>
        <img src = "{% static 'main/img/main_img.svg'%}" alt="logo">
        <span class = "logo">Site for test PostgerSQL</span>
        <h3>Навигация</h3>
        <ul>
            <a href="{% url 'doctors' %}"><li><i class="fa-solid fa-
layer-group"></i>Доктора</li></a>
            <a href="{% url 'patients'%}"><li><i class="fa-solid fa-
layer-group"></i>Пациенты</li></a>
            <a href="{% url 'visits' %}"><li><i class="fa-solid fa-
layer-group"></i>Посещение врачей</li></a>
            <a href="{% url 'payment_rep' %}"><li><i class="fa-solid
fa-layer-group"></i>Отчет по зарплате врачей</li></a>
            <a href="{% url 'receipt_patient' %}"><li><i class="fa-
solid fa-layer-group"></i>Отчет о квитанциях пациента</li></a>
        </ul>
    </aside>
    <main>
    {% block content %}
    {% endblock %}
    </main>
</body>
</html>
```

Вид:



В проект подключен boostrap и некоторый css-код. В меню слева есть 5 кнопкок. Первая ведёт на страницу с данными таблицы врачей. Вторая ведёт на страницу с данными таблицы пациентов. Третья ведёт на страницу с данными таблицы посещения врачей. Четвертая ведёт на страницу с данными отчета о зарплате врачей. Пятая ведёт на страницу с данными отчета квитанций об оплате пациента за приемы.

Рассмотрим подключенные к проекту таблицы и их содержимое.

```
class Doctor(models.Model):
    iddoctor = models.AutoField(primary_key=True)
    fio = models.CharField(max_length=30)
    specs = models.CharField(max_length=30)
    payment = models.IntegerField()
    percentprofit = models.FloatField(blank=True, null=True)

    def __str__(self):
        return self.fio

    def get_absolute_url(self):
        return f'/{self.iddoctor}'
    class Meta:
        managed = False
        db_table = 'doctor'
class Patient(models.Model):
    idpatient = models.AutoField(primary_key=True)
    surname = models.CharField(max_length=10)
    name = models.CharField(max_length=10)
    patronymic = models.CharField(max_length=10, blank=True,
null=True)
    bdate = models.DateField(blank=True, null=True)
    address = models.CharField(max_length=30)

    def __str__(self):
        return self.surname + " " +self.name

    def get_absolute_url(self):
        return f'/{self.idpatient}'
    class Meta:
        managed = False
        db_table = 'patient'
class Visits(models.Model):
    iddoctor = models.ForeignKey(Doctor, models.DO_NOTHING,
db_column='iddoctor')
    idpatient = models.ForeignKey(Patient, models.DO_NOTHING,
db_column='idpatient')
    date_of_visit = models.DateField()
    idvisits = models.AutoField(primary_key=True)

    def __str__(self):
        return self.iddoctor + " "  + self.idpatient + " " +
self.date_of_visit
    def get_absolute_url(self):
        return f'/{self.idvisits}'
    class Meta:
        managed = False
        db_table = 'visits'
```

Вид таблиц и процедуры в pgAdmin 4

Doctor:

	iddoctor [PK] integer	fio character varying	specs character varying	payment integer	percentprofit real
1	3	Krivko Leonid Pavlovich	Therapist	3000	0.21
2	4	Orlov Fedor Mikhailovich	Surgeon	11000	0.37
3	5	Tutchkov Ivan Vladimirovich	Surgeon	7500	0.32
4	6	Степанов Андрей Юрьевич	Стоматолог	10000	0.41
5	7	Степанов Андрей Юрьевич	Стоматолог	10000	0.41
6	2	Rubcov Genadiy Adnree	Dentist	7500	0.34

Patient:

	idpatient [PK] integer	surname character varying	name character varying	patronymic character varying	bdate date	address character varying
1	2	Lobanov	Fedor	Alexandrov	2005-05-21	Irkutsk, Kirova 55a
2	3	Ryabec	Ilya	Nikitich	1997-09-17	Irkutsk, Volzskaya 55a
3	1	Sablin	Mikhail	Sanych	2002-10-31	Khomutov, Kolhoznaya 100B

Visits:

	iddoctor integer	idpatient integer	date_of_visit date	idvisits [PK] integer
1	2	2	2023-03-23	2
2	3	3	2023-03-23	3
3	2	1	2022-12-20	4
4	3	2	2023-03-13	5
5	2	3	2023-05-11	8
6	3	1	2023-03-25	9

Для того, чтобы вывести данные на сайте произведем обращение через Django к следующим процедурам:

Добавление в таблицу Docotor:

```
CREATE OR REPLACE PROCEDURE public."INSERT_DOCTOR"(
 IN dfio character varying,
 IN dspecs character varying,
 IN dpayment integer,
 IN dpercentprofit real,
 IN diddoctor integer)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
 IF EXISTS (SELECT iddoctor FROM doctor WHERE iddoctor = diddoctor)
 THEN
       RAISE EXCEPTION 'There is doctor with id %. No isertation was
performed', diddoctor;
 END IF;
 IF EXISTS (SELECT fio FROM doctor WHERE fio = dfio)
 THEN
       RAISE EXCEPTION 'There is doctor with sama name';
 END IF;
 IF diddoctor = -1
 THEN
       INSERT INTO public.doctor
             (fio, specs, payment, percentprofit, iddoctor)
       VALUES
             (dfio, dspecs, dpayment, dpercentprofit,
nextval('doctor_iddoctor_seq'));
 ELSE
       INSERT INTO public.doctor
             (fio, specs, payment, percentprofit, iddoctor)
       VALUES
             (dfio, dspecs, dpayment, dpercentprofit, diddoctor);
 END IF;
END
$BODY$;
ALTER PROCEDURE public."INSERT_DOCTOR"(character varying, character
varying, integer, real, integer)
    OWNER TO postgres;

COMMENT ON PROCEDURE public."INSERT_DOCTOR"(character varying,
character varying, integer, real, integer)
    IS 'Insert new Doctor';
```

Изменение таблицы Doctor:

```
CREATE OR REPLACE PROCEDURE public."UPDATE_DOCTOR"(
 IN mfio character varying,
 IN mspecs character varying,
 IN mpayment integer,
 IN mpercentprofit real,
 IN middoctor integer)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
-- iddoctor is the primary key, so it is not updatable
 IF EXISTS (SELECT iddoctor FROM doctor WHERE iddoctor=middoctor)
 THEN
       UPDATE public.doctor
             SET
             fio=mfio, specs=mspecs,
             payment=mpayment, percentprofit=mpercentprofit  -- ,
                                                              --
tablenumber=?, department=?
             WHERE iddoctor = middoctor;
 ELSE
       RAISE EXCEPTION 'There is no Doctor with id %.', middoctor;
 END IF;
END
$BODY$;
ALTER PROCEDURE public."UPDATE_DOCTOR"(character varying, character
varying, integer, real, integer)
    OWNER TO postgres;

COMMENT ON PROCEDURE public."UPDATE_DOCTOR"(character varying,
character varying, integer, real, integer)
    IS 'Update Employee data';
```

Удаление из таблицы Doctor:

```
CREATE OR REPLACE PROCEDURE public."DELETE_DOCTOR"(
 IN mnumber integer)
LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
 IF NOT EXISTS (SELECT iddoctor FROM doctor WHERE iddoctor = mnumber)
 THEN
       RAISE EXCEPTION 'There no doctor with id %', mnumber;
 END IF;
 IF EXISTS (SELECT iddoctor FROM visits v WHERE iddoctor = mnumber)
 THEN
       RAISE EXCEPTION 'There are visits to this doctor. Can not
delete';
 END IF;
 DELETE FROM doctor WHERE iddoctor = mnumber;
END

$BODY$;
ALTER PROCEDURE public."DELETE_DOCTOR"(integer)
    OWNER TO postgres;
```

Также создано представление для вывода на сайте отчета о количестве и стоимости посещения врача пациентом:

```
CREATE OR REPLACE FUNCTION public."RECEIPT_OF_PATIENT"(
 first_date date,
 second_date date,
 pid_of_patient integer)
    RETURNS TABLE(surname character varying, name character varying,
pdate_of_visit date, receipt_of_visit integer)
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
begin
 return query
 SELECT pt.surname,
    pt.name,
    vi.date_of_visit,
    dc.payment
       FROM patient pt,
    visits vi,
    doctor dc
       WHERE vi.idpatient = pt.idpatient AND pt.idpatient=pid_of_patient
AND vi.iddoctor = dc.iddoctor AND vi.date_of_visit <= second_date AND
vi.date_of_visit >=first_date
       ORDER BY vi.date_of_visit DESC;
end
$BODY$;

ALTER FUNCTION public."RECEIPT_OF_PATIENT"(date, date, integer)
    OWNER TO postgres;
```

Также создано представление для вывода на сайте отчета о зарплате врачей за определенный промежуток времени:

```
CREATE OR REPLACE FUNCTION public."SALARY_OF_DOCTORS"(
	first_date date,
	second_date date)
    RETURNS TABLE(fio character varying, salary_of_doctor real)
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
begin
	return query
	SELECT dc.fio,
    (count(vi.iddoctor = dc.iddoctor) * dc.payment)::real * dc.percentprofit - (0.13 * count(vi.iddoctor = 
dc.iddoctor)::numeric * dc.payment::numeric)::real AS salary
	FROM visits vi,
    doctor dc
	WHERE vi.iddoctor = dc.iddoctor AND vi.date_of_visit >= first_date  AND vi.date_of_visit <= 
second_date
	GROUP BY dc.iddoctor
	ORDER BY dc.fio;
end
$BODY$;

ALTER FUNCTION public."SALARY_OF_DOCTORS"(date, date)
    OWNER TO postgres;
```

Реализация функций на сайте. Добавление записей о врачах в таблицу.

Ниже приведен Html-код: страницы.

add_doctor.html:

```
{% extends 'main/layout.html'%}

{% block title %}AddDoctor page{% endblock %}

{% block content %}
<div class="features">
    <h1>Форма для добавления врача</h1>
    <form method="post">
        {% csrf_token %}
        <input type="hidden" name="id_Doctor" value="{{ Doctor.pk
}}">
        {{ form.iddoctor }}<br>
        {{ form.fio }}<br>
        {{ form.specs }}<br>
        {{ form.payment }}<br>
```

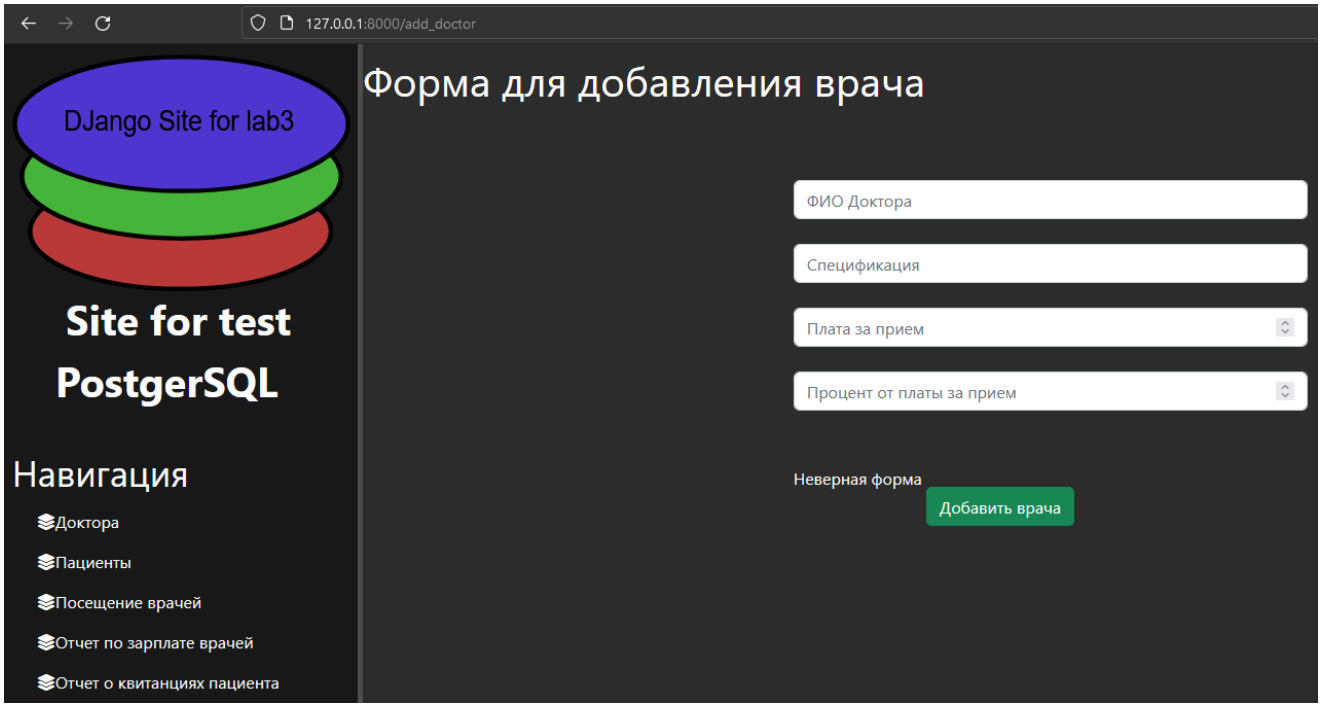
Создана модель для формы в forms.py

```
class DoctorForm(ModelForm):
    class Meta:
        model = Doctor
        fields = ['iddoctor', 'fio', 'specs', 'payment',
'percentprofit']

        widgets = {
            "fio": TextInput(attrs={
                'class': 'form-control',
                'placeholder': 'ФИО Доктора'
            }),
            "specs": TextInput(attrs={
                'class': 'form-control',
                'placeholder': 'Спецификация'
            }),
            "payment": NumberInput(attrs={
                'class': 'form-control',
                'placeholder': 'Плата за прием'
            }),
            "percentprofit": NumberInput(attrs={
                'class': 'form-control',
                'min': '0', 'max': '1', 'step': '0.01',
                'placeholder': 'Процент от платы за прием'
            })
        }
```

Функция, реализующая функционал формы и обращающаяся к процедуре add_doctor в файле views.py:

```
def add_doctor(request):
    error = ''
    if request.method == 'POST':
        fio = request.POST.get('fio')
        specs = request.POST.get('specs')
        payment = request.POST.get('payment')
        percentprofit = request.POST.get('percentprofit')
        diddoctor = -1
        with connection.cursor() as cursor:
            cursor.execute('call "INSERT_DOCTOR"(%s,%s,%s,%s,%s)',
(fio, specs, payment, percentprofit, diddoctor))
            cursor.close()
        return redirect('doctors')
    else:
        error = 'Неверная форма'

    form = DoctorForm()
    data = {
        'form': form,
        'error': error
    }
    return render(request, 'main/add_doctor.html', data)
```

Как это выглядит на сайте:



Данные файла urls.py будут приведены после рассмотрения всего функционала.

Страница с данными таблицы Doctors.

doctors.html:

```
{% extends 'main/layout.html'%}

{% block title %}Main page{% endblock %}

{% block content %}

    <div class="features">
        {% if doctors %}
            <h1>Страница с таблицей врачей</h1>
            <div class="table-responsive">
                <table className="table table-striped table-
bordered">
                    <thead>
                      <tr>
                        <th scope="col" class="sticky-column"
>id</th>
                        <th scope="col" class="sticky-
column">FIO</th>
                        <th scope="col" class="sticky-
column">SPECS</th>
                        <th scope="col" class="sticky-
column">PAYMENT</th>
                        <th scope="col" class="sticky-
column">PERCENTOFPROFIT</th>

                      </tr>
                    </thead>
                    <tbody>
                      {% for el in doctors%}
                        <tr>
                          <td><a href = "{% url 'doctor_detail'
el.iddoctor  %}"> {{ el.iddoctor }}</a></td>
                          <td> {{ el.fio}}</td>
                          <td> {{ el.specs}}</td>
                          <td> {{ el.payment}}</td>
                          <td> {{ el.percentprofit}}</td>
                        </tr>
                      {% endfor %}
                    </tbody>
                </table>
            </div>
            <a href="{% url 'add_doctor' %}"><button class="btn
btn-warning">Добавить</button> </a>

        {% else %}
            <p>Puk!</p>
        {% endif %}
    </div>

{% endblock %}
```

Функция для отображения врачей на странице:

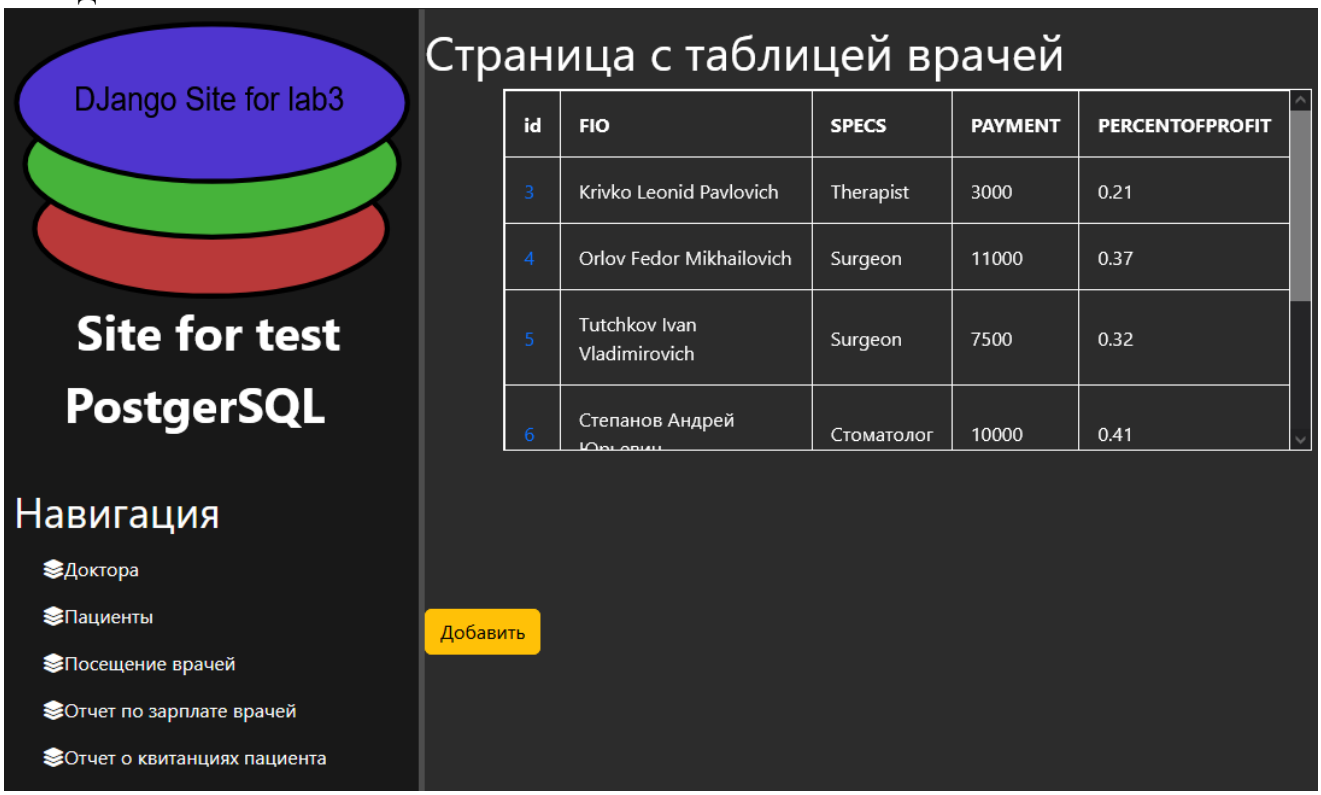
```
def doctors(request):
    get_doctor = Doctor.objects.all()
    return render(request, 'main/doctors.html', {'doctors' :
get_doctor})
```

Чтобы как-либо взаимодействовать с записью о враче из таблицы, реализованы ссылки, по которым можно перейти, если нажать на ID доктора.

Функция в DoctorDetailView для отображения каждой записи о доктрое отдельно:

```
class DoctorDetailView(DetailView):
    model = Doctor
    template_name = 'main/doctor_view.html'
```

Вид на сайте:

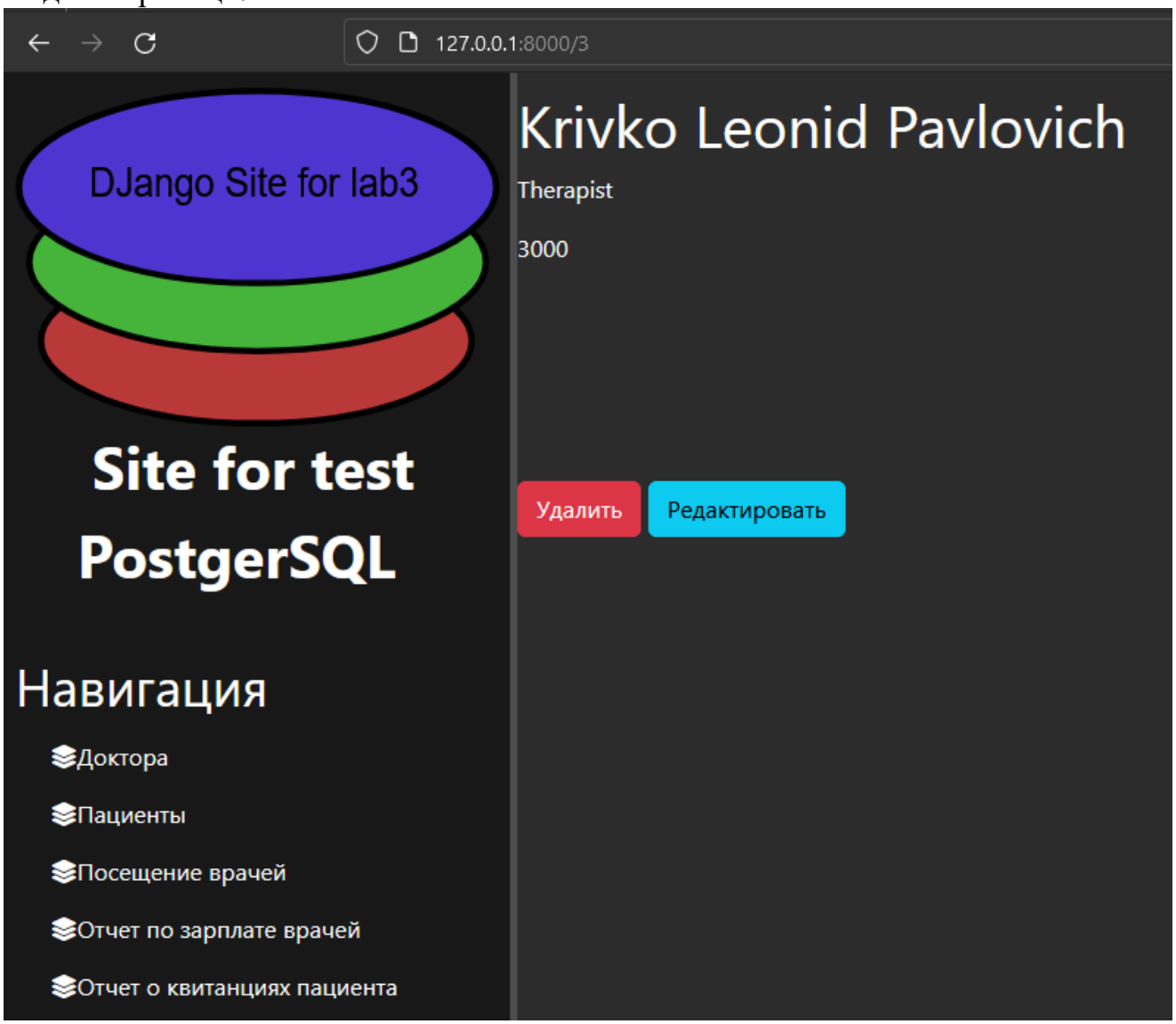


Страница для отображения каждой записи доктора отдельно:

```
{% extends 'main/layout.html'%}

{% block title %}Main page{% endblock %}

{% block content %}

    <div class="features">
        <h1>{{ doctor.fio }}</h1>
        <p>{{ doctor.specs }}</p>
        <p>{{ doctor.payment }}</p>
        <a href="{% url  'doctor_delete' doctor.iddoctor %}"><button
class="btn btn-danger">Удалить</button> </a>
        <a href="{% url  'doctor_update' doctor.iddoctor %}"><button
```

```
class="btn btn-info">Редактировать</button> </a>
    </div>

{% endblock %}
```

Вид на странице:



Теперь реализуем функционал каждой кнопки.

Редактирование и добавление в таблице doctors

add_doctor.html:

```
{% extends 'main/layout.html'%}

{% block title %}AddDoctor page{% endblock %}

{% block content %}
<div class="features">
    <h1>Форма для добавления врача</h1>
    <form method="post">
        {% csrf_token %}
        <input type="hidden" name="id_Doctor" value="{{ Doctor.pk
}}">
        {{ form.iddoctor }}<br>
        {{ form.fio }}<br>
        {{ form.specs }}<br>
        {{ form.payment }}<br>
        {{ form.percentprofit }}<br>
        <span>{{ error }}</span>

        <button class ="btn btn-success" type="submit">Добавить
врача</button>
    </form>

</div>
{% endblock %}
```

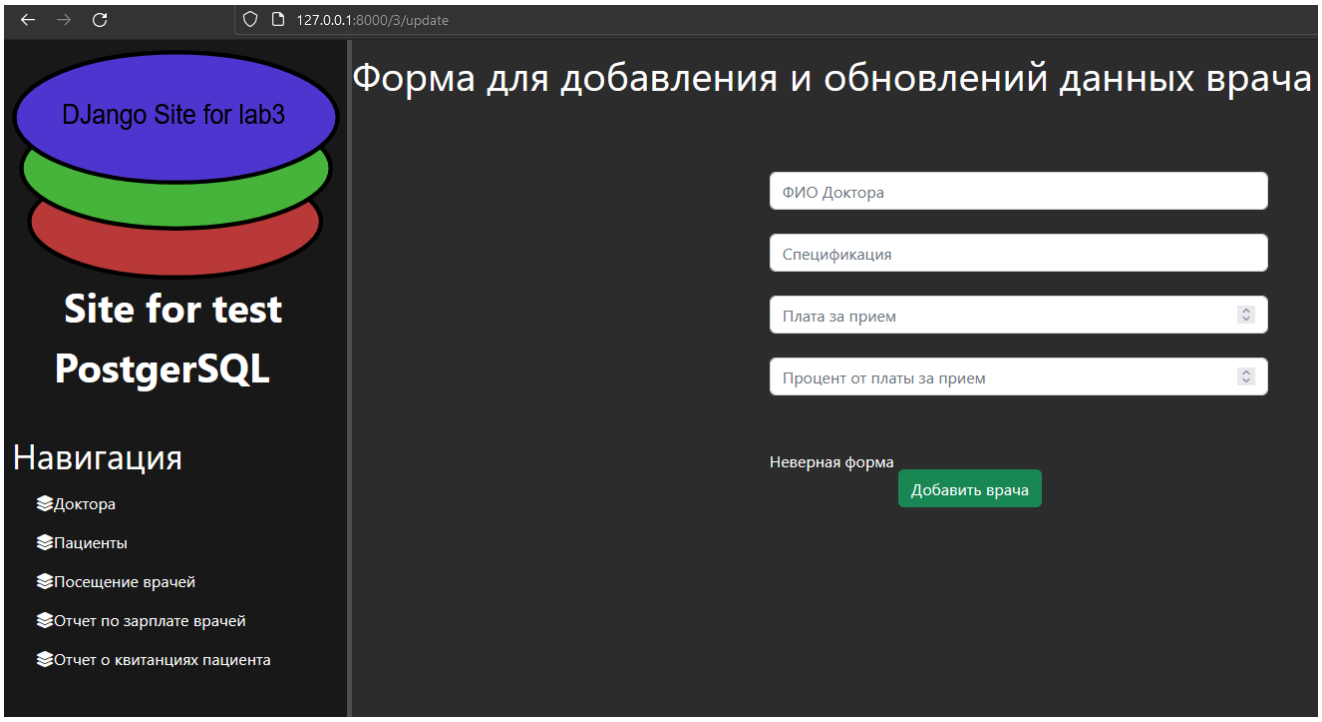
Функция добавления новой записи для врачей:

```
def add_doctor(request):
    error = ''
    if request.method == 'POST':
        fio = request.POST.get('fio')
        specs = request.POST.get('specs')
        payment = request.POST.get('payment')
        percentprofit = request.POST.get('percentprofit')
        diddoctor = -1
        with connection.cursor() as cursor:
            cursor.execute('call
"INSERT_DOCTOR"(%s,%s,%s,%s,%s)', (fio, specs, payment,
percentprofit, diddoctor))
            cursor.close()
        return redirect('doctors')
    else:
        error = 'Неверная форма'

    form = DoctorForm()
    data = {
        'form': form,
        'error': error
    }
    return render(request, 'main/add_doctor.html', data)
```

Функция обновления записи о враче:

```
def update_doctor(request, pk):
    error = ''
    if request.method == 'POST':
        fio = request.POST.get('fio')
        specs = request.POST.get('specs')
        payment = request.POST.get('payment')
        percentprofit = request.POST.get('percentprofit')
        diddoctor=pk
        with connection.cursor() as cursor:
            cursor.execute('call "UPDATE_DOCTOR"(%s,%s,%s,%s,%s)',
(fio, specs, payment, percentprofit, diddoctor))
            cursor.close()
        return redirect('doctors')
    else:
        error = 'Неверная форма'
    form = DoctorForm()

    data = {
        'form': form,
        'error': error
    }

    return render(request, 'main/add_doctor.html', data)
```

Вид на сайте:



Удаление записей о врачах

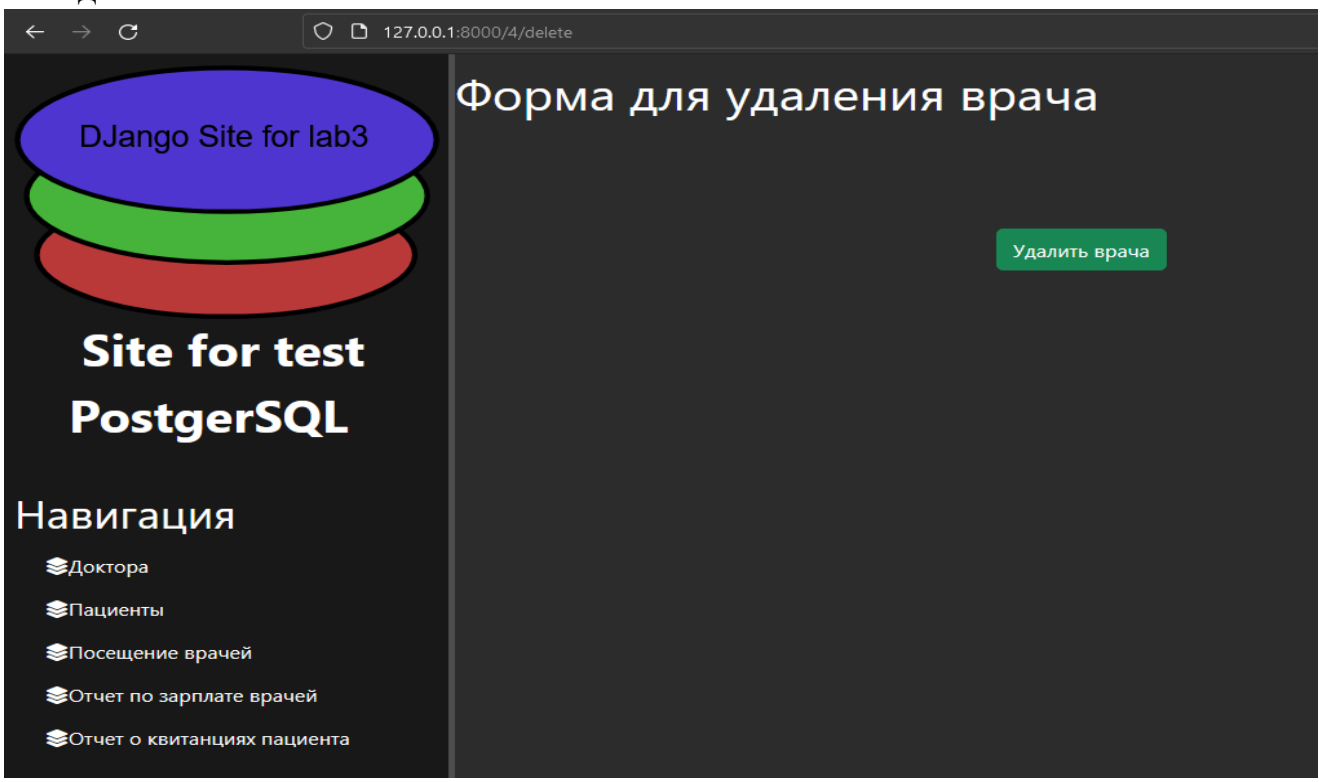
delete_doctor.html:

```
{% extends 'main/layout.html'%}

{% block title %}DeleteDoctor page{% endblock %}

{% block content %}
<div class="features">
    <h1>Форма для удаления врача</h1>
    <form method="post">
        {% csrf_token %}
        <input type="hidden" name="id_Doctor" value="{{ Doctor.pk
}}">
        <button class ="btn btn-success" type="submit">Удалить
врача</button>
    </form>

</div>
{% endblock %}
```

Views.py:

```
def doctor_delete(request, pk):
    if request.method == 'POST':
        with connection.cursor() as cursor:
            cursor.execute('call "DELETE_DOCTOR"(%s)', [pk])
            cursor.close()
        return redirect('doctors')

    return render(request, 'main/delete_doctor.html')
```

Вид на сайте:



Вывод отчетов

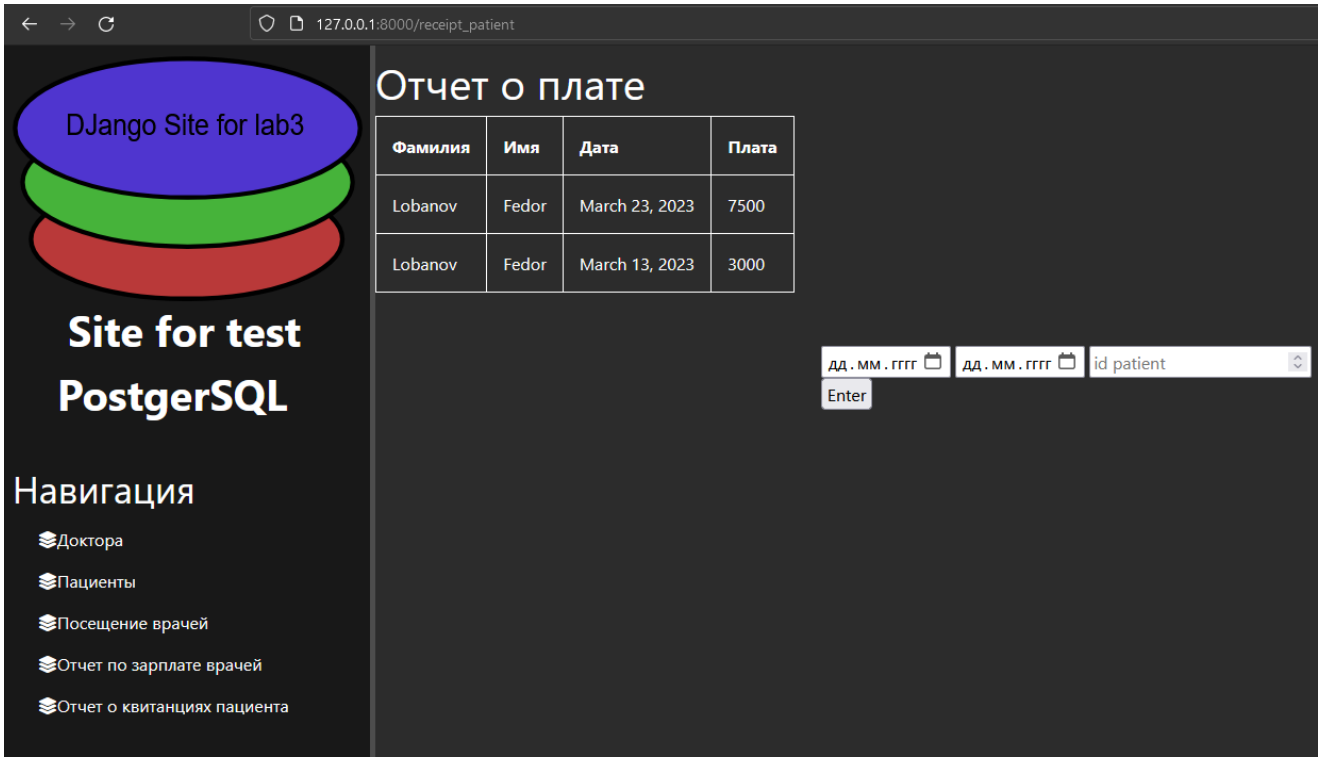
payment_rep.html:

```
{% block content %}
    <h1>Отчет {{ name }}</h1>
    <table>
        <thead>
            <tr>
            <th>Фио</th>
            <th>Зарплата</th>
            </tr>
        </thead>
        <tbody>
            {% for row in rows %}
            <tr>
            <td>{{ row.0 }}</td>
            <td>{{ row.1}}</td>
            </tr>
            {% endfor %}
        </tbody>
    </table>

    <form method="POST" action="payment_rep" >
        {% csrf_token %}
        <input type="date" name="first" placeholder="first date"
title="Enter first date">
        <input type="date" name="second" placeholder="second date"
title="Enter second date">
        <input type="submit" value="Enter">
    </form>
{% endblock %}
```

Функция для отображения отчета:

```
def payment_rep(request):

    first_date = request.POST.get('first')
    second_date = request.POST.get('second')
    with connection.cursor() as cursor:
        cursor.execute('SELECT * from "SALARY_OF_DOCTORS"(%s,%s)',
(first_date, second_date))
        rows = cursor.fetchall()
        cursor.close()
    context = {'rows': rows}
    return render(request, 'main/payment rep.html', context)
```

Вид на сайте:


127.0.0.1:8000/receipt_patient
DJango Site for lab3
Site for test
PostgerSQL
Навигация
Доктора
Пациенты
Посещение врачей
Отчет по зарплате врачей
Отчет о квитанциях пациента
Отчет о плате
Фамилия
Имя
Дата
Плата
Lobanov
Fedor
March 23, 2023
7500
Lobanov
Fedor
March 13, 2023
3000
дд . мм . гггг
дд . мм . гггг
id patient
Enter

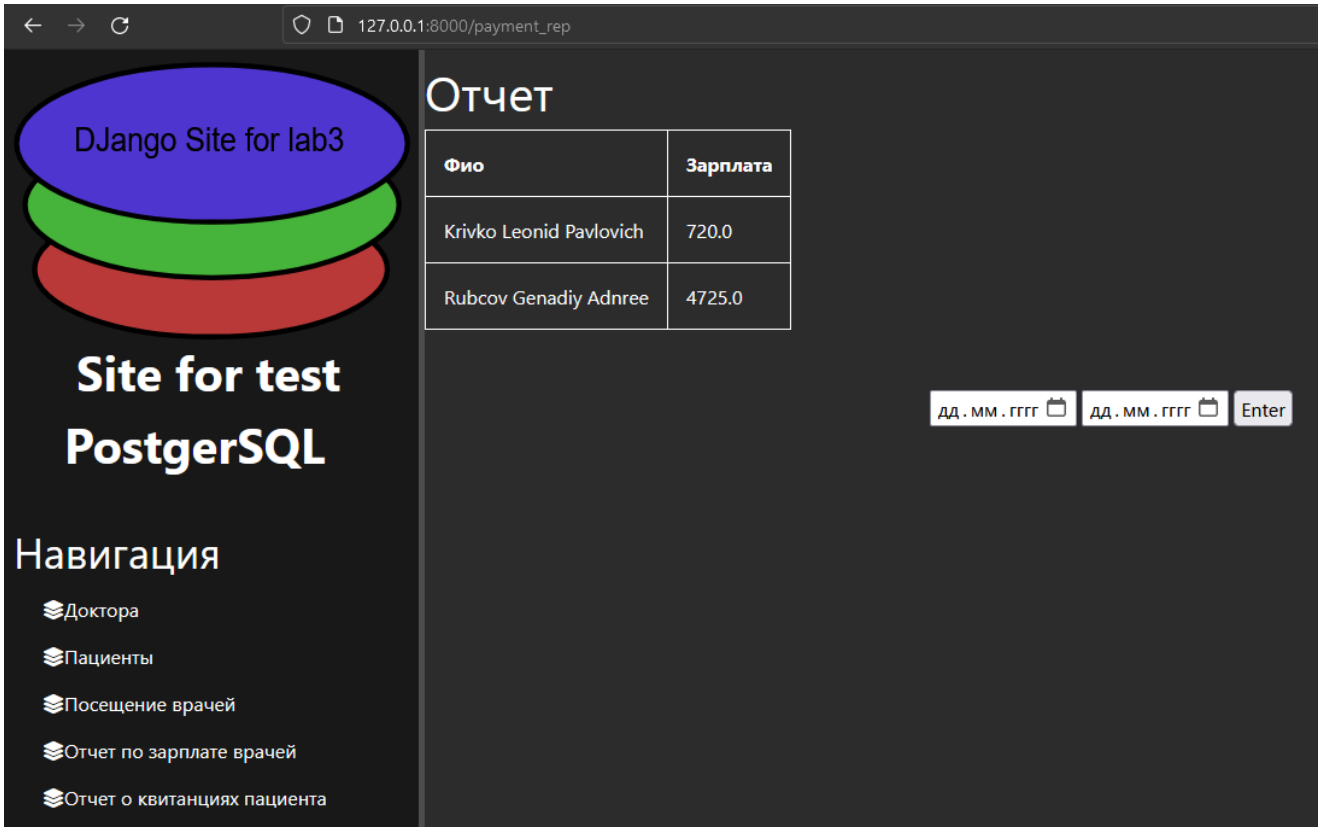
Страница для отображения зарплаты врачей за определенный промежуток:

```
{% block title %}Payment per page{% endblock %}

{% block content %}
    <h1>Отчет о плате</h1>
    <table>
        <thead>
            <tr>
                <th>Фамилия</th>
                <th>Имя</th>
                <th>Дата</th>
                <th>Плата</th>
            </tr>
        </thead>
        <tbody>
            {% for row in rows %}
                <tr>
                    <td>{{ row.0 }}</td>
                    <td>{{ row.1 }}</td>
                    <td>{{ row.2 }}</td>
                    <td>{{ row.3 }}</td>
                </tr>
            {% endfor %}
        </tbody>
    </table>

    <form method="POST" action="receipt_patient" >
        {% csrf_token %}
        <input type="date" name="first" placeholder="first date"
title="Enter first date">
        <input type="date" name="second" placeholder="second date"
title="Enter second date">
        <input type="number" name="third" placeholder="id patient"
title="Enter id patient">
        <input type="submit" value="Enter">
    </form>
{% endblock %}
```

Функция для отображения отчета:

```
def receipt_patient(request):

    first_date = request.POST.get('first')
    second_date = request.POST.get('second')
    pid = request.POST.get('third')
    with connection.cursor() as cursor:
        cursor.execute('SELECT * from
"RECEIPT_OF_PATIENT"(%s,%s,%s)', (first_date, second_date, pid))
        rows = cursor.fetchall()
        cursor.close()
    context = {'rows': rows}
    return render(request, 'main/receipt_patient.html', context)
```

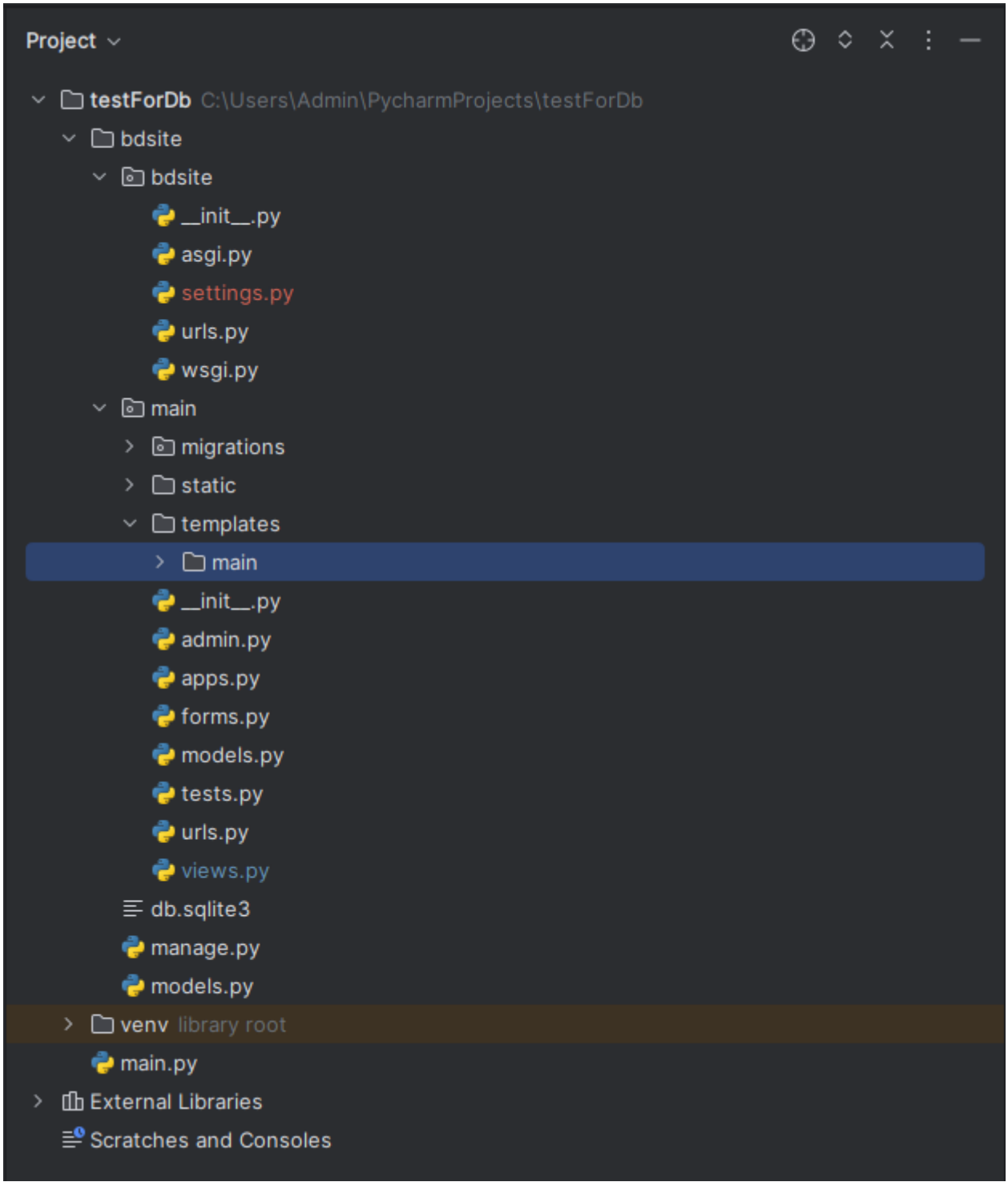
Вид на странице:


127.0.0.1:8000/payment_rep
DJango Site for lab3
Site for test
PostgerSQL
Навигация
Доктора
Пациенты
Посещение врачей
Отчет по зарплате врачей
Отчет о квитанциях пациента
Отчет
Фио
Зарплата
Krivko Leonid Pavlovich
720.0
Rubcov Genadiy Adnree
4725.0
дд . мм . гггг
дд . мм . гггг
Enter

Файл Urls.py

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.doctors, name="doctors"),
    path('patients', views.patients, name='patients'),
    path('visits', views.visits, name='visits'),
    path('add_doctor', views.add_doctor, name='add_doctor'),
    path('add_patient', views.add_patient, name='add_patient'),
    path('add_visit', views.add_visit, name='add_visit'),
    path('<int:pk>', views.DoctorDetailView.as_view(),
name='doctor_detail'),
    path('<int:pk>/update', views.update_doctor,
name='doctor_update'),
    path('<int:pk>/delete', views.doctor_delete,
name='doctor_delete'),
    path('patients/<int:pk>', views.PatientDetailView.as_view(),
name='patient_detail'),
    path('patients/<int:pk>/update', views.update_patient,
name='patient_update'),
    path('patients/<int:pk>/delete', views.patient_delete,
name='patient_delete'),
    path('visits/<int:pk>', views.VisitDetailView.as_view(),
name='visit_detail'),
    path('visits/<int:pk>/update', views.update_visit,
name='visit_update'),
    path('visits/<int:pk>/delete', views.visit_delete,
name='visit_delete'),
    path('payment_rep', views.payment_rep, name='payment_rep'),
    path('receipt_patient', views.receipt_patient,
name='receipt_patient')
]
```

Архитектура проекта



Решил не показывать код для Patient и Visits так как код очень схож с тем, что было написано для Doctor, весь код, кроме «settings.py», можно увидеть по данной ссылке: https://github.com/MishaSabre/testForDb/tree/master