

# Проект сайта ветеринарной клиники



# Описание проекта

Данный проект сайта предназначен для отображения информации о ветеринарной клинике - 'Clinic'. На сайте пользователю предоставляется информация о сотрудниках, оказываемых услугах, контактные данные, информация о ценах на услуги. Также на сайте есть возможность онлайн-записи на прием и оставления отзывов.

# Описание проекта

На главной странице сайта выводится навигация по сайту в виде основного меню, которая находится в верхней части экрана.



***Наша клиника!***

Главная

Услуги

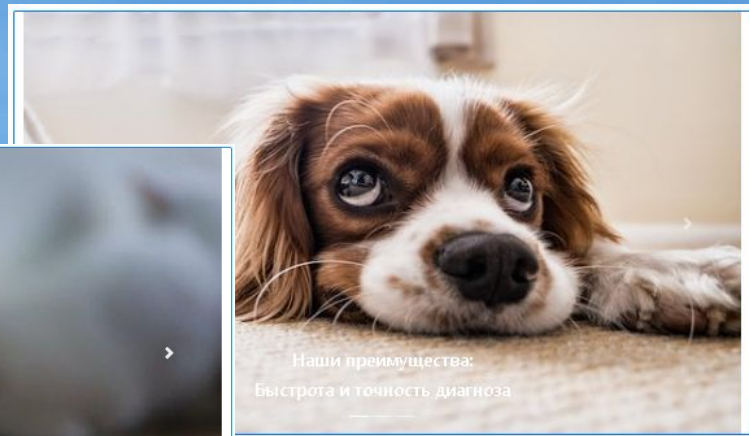
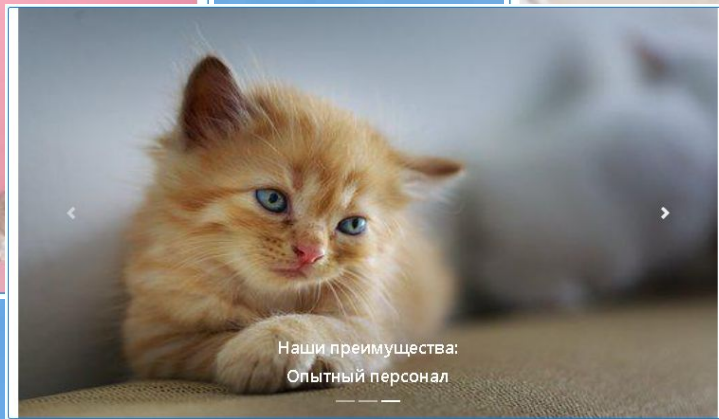
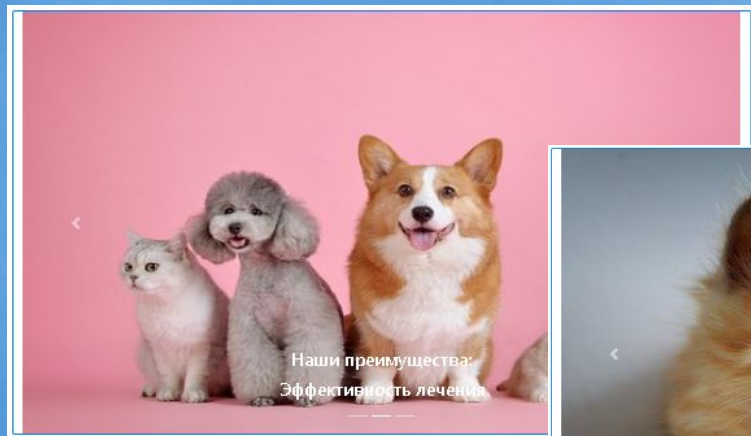
Цены

Наши специалисты

Контакты

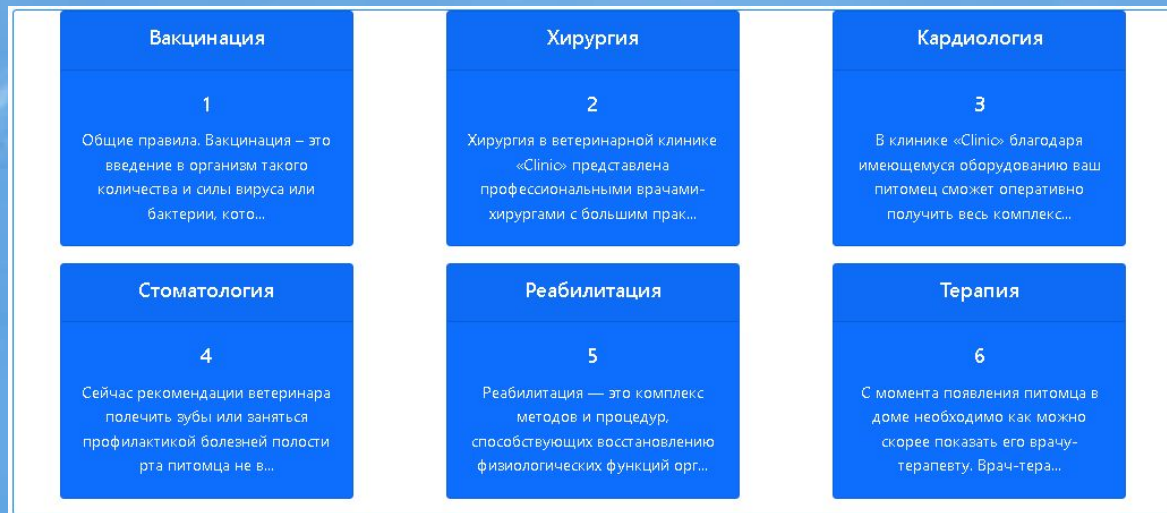
# Описание проекта

Далее по странице выводим 'карусель' из трех фотографий с описанием преимуществ клиники.



# Описание проекта

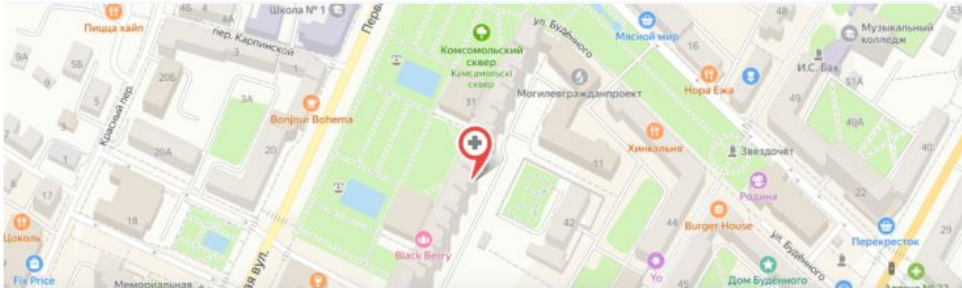
После следует информационный блок из услуг, которые оказывает клиника. Данный блок оформлен в виде блока карточек с кликабельными названиями.



# Описание проекта

Следующий блок главной страницы представлен фотографией карты с меткой нахождения клиники, адресом и контактами.

Как нас найти? Мы расположены в центре Могилева, нас нельзя не заметить.



**Контакты**

Прием у любого из наших специалистов возможен по предварительной записи

- Адрес: г. Могилев ул. Первомайская, д. 31
- Телефоны: + 375 (222) 76-76-76 + 375 (44) 702-33-33 + 375 (29) 702-33-44
- 
- 
- 
- 

Время работы  
Пн-Пт: с 8:00 до 17:00  
Сб: с 9:00 до 17:00  
Вс: с 9:00 до 17:00


[Записаться на прием](#)

- Адрес
- г. Могилев
- ул. Первомайская, д. 31



# Описание проекта

На странице 'Услуги' сайта выводится информация (не полная) с описанием оказываемых услуг. Также здесь пользователю предоставлены кнопки для получения более детальной информации по услуге.

[Главная](#)[Услуги](#)[Цены](#)[Наши специалисты](#)[Контакты](#)

[Онлайн-запись на прием](#)

### Наши сервисы:

#### Вакцинация

Общие правила. Вакцинация – это введение в организм такого количества и силы вируса или бактерии, которые неспособны спровоцировать заболевание, но заставляют организм вырабатывать защитные антитела к этим заболеваниям (при условии, что на момент вакцинации животное является клинически здоровым). ЗАЧЕМ НУЖНА ВАКЦИНАЦИЯ? Это поможет избежать заражения заболеваниями или, как минимум, снизить проявления тех заболеваний, против которых проводилась вакцинация. Прививки обычно делаются от наиболее распространенных, опасных и сложных в диагностике и лечении заболеваний. И даже если животное не покидает пределы дома, оно рискует заразиться вирусами, которые могут быть принесены в дом на одежде или обуви хозяев. Вакцинированный питомец безопасен для здоровья хозяина и его семьи. Особенно страшны для человека лептоспироз и бешенство. И если первая болезнь крайне неприятная, но излечимая, правда, с массой осложнений, то бешенство не лечится! Именно поэтому вакцинация домашних животных от бешенства является обязательной по законодательству РБ. В год в РБ регистрируется около 400 случаев бешенства у животных со ...

[Подробнее](#)


#### Кардиология

В клинике «Clinic» благодаря имеющемуся оборудованию ваш питомец сможет оперативно получить весь комплекс исследований, необходимых при подозрении на патологию сердца. у нас работают специализированные врачи-кардиологи, которые в отличие от врачей широкого профиля «заточены» под решение задач, связанных именно с сердечно-сосудистой системой и сердцем. Для получения достоверных результатов УЗИ сердца мы используем ультразвуковую систему экспертного класса «GE Vivid E9», специально разработанную для более детального сканирования сердечно-сосудистой системы с визуализацией в 4D. За один сердечный цикл данный УЗИ аппарат позволяет выстроить детальную визуализацию с высокой частотой кадров, за счет высокого качества изображения и широкого поля обзора, не только сердечного клапана, но и желудочка. Данное оборудование позволяет врачу видеть больше и точнее нарушения в сосудах и в сердце в целом, чем при обследовании на других системах, что несомненно важно при постановке точного диагноза. ПРОЯВЛЕНИЯ КАРДИОЗАБОЛЕВАНИЙ Помните, чем раньше болезнь уйдёт выявиться, тем больше шансов у пациента. Крайне важно при проявлении симптомов сердечной патологии, ...

[Подробнее](#)

# Описание проекта

При нажатии на кнопку 'Подробнее' в секции интересующего сервиса, пользователь переходит на страницу с более подробным его описанием.



Главная Услуги Цены Наши специалисты Контакты

**Наша клиника!**

[Онлайн-запись на прием](#)

## Подробнее о сервисе Вакцинация

### Вакцинация

Общие правила. Вакцинация – это введение в организм такого количества и силы вируса или бактерии, которые неспособны спровоцировать заболевание, но заставляют организм вырабатывать защитные антитела к этим заболеваниям (при условии, что на момент вакцинации животное является клинически здоровым). ЗАЧЕМ НУЖНА ВАКЦИНАЦИЯ? Это поможет избежать заражения заболеваниями или, как минимум, снизить проявления тех заболеваний, против которых проводилась вакцинация. Прививки обычно делаются от наиболее распространённых, опасных и сложных в диагностике и лечении заболеваний. И даже если животное не покидает пределы дома, оно рискует заразиться вирусами, которые могут быть принесены в дом на одежде или обуви хозяев. Вакцинированный питомец безопасен для здоровья хозяина и его семьи. Особенно страшны для человека лептоспироз и бешенство. И если первая болезнь крайне неприятная, но излечимая, правда, с массой осложнений, то бешенство не лечится! Именно поэтому вакцинация домашних животных от бешенства является обязательной по законодательству РФ. В год в РФ регистрируется около 400 случаев бешенства у животных со смертельным исходом. Согласно Постановлению Министерства здравоохранения и Министерства сельского хозяйства и продовольствия РФ 30.05.2000 N 28/10, а также Постановлению Совета Министров РФ 04.06.2001 N 834 хозяева домашних кошек и собак обязаны проводить их вакцинацию против бешенства. В случае уклонения предусмотрен штраф от 2 до 10 базовых величин. Перед вакцинацией ветврач полностью осмотрит животное (кожный покров, слизистые оболочки, глаза, уши), пропальпирует брюшную полость, прослушает грудную клетку, измерит температуру, соберёт сведения о поведении питомца, чтобы убедиться, что он здоров. Ведь если животное нездорово (вирусная или бактериальная инфекция, аллергические реакции и т.п.), банальная прививка может привести к самым неожиданным последствиям. Поэтому идеально вакцинироваться после диспансеризации, в ходе которой, кроме осмотра, будут сделаны анализы крови и мочи, проведены УЗИ и рентген, при необходимости кожные исследования. ЧТО ЕЩЁ НУЖНО ЗНАТЬ О ВАКЦИНАЦИИ? Самок, от которых планируют получать приплод, рекомендуется вакцинировать за 2-3 месяца до случки. Ввиду того что наибольшее количество антител в организме сохраняется первые 6-7 месяцев, новорожденные получат с маминым молоком их максимальное количество. Помните, животные, не прошедшие полный курс вакцинации, не защищены от болезней в полной мере и не должны гулять на улице и контактировать с другими животными. И даже после повторной вакцинации с прогулками не следует спешить, так как иммунитет на вакцины вырабатывается не сразу. Поэтому первый выгул должен быть не ранее чем через 10-14 дней после повторной противовирусной вакцинации. Если вы планируете выезд с животным за пределы страны, то вакцинация должна проводиться минимум за месяц до выезда. Кроме того животное должно пройти процедуру идентификации, а именно – чипирования. Причём, исходя из последних изменений в законодательстве, процедура чипирования должна быть выполнена в первую очередь, а уже после неё все остальные процедуры, в том числе и вакцинации.

Назад



# Описание проекта

Для упрощения навигации на странице 'Услуги' (в том числе и на других страницах) в нижней части экрана предусмотрена кнопка для перехода на предыдущую страницу.



Назад

# Описание проекта

При активации кнопки 'Цены' на панели основного меню, пользователю предоставляется информация в виде таблицы об ориентировочных ценах на услуги.

## Информация по ценам на услуги:

#	Наименование услуги	Цена [BYN]
1	<a href="#">Вакцинация</a>	от 50,00
2	<a href="#">Хирургия</a>	от 100,50
3	<a href="#">Кардиология</a>	от 70,20
4	<a href="#">Стоматология</a>	от 45,50
5	<a href="#">Реабилитация</a>	от 40,80
6	<a href="#">Терапия</a>	от 40,00

[Более подробно по стоимости уточните у специалиста](#)

# Описание проекта

На данной странице имеется возможность получить детальное описание сервиса.

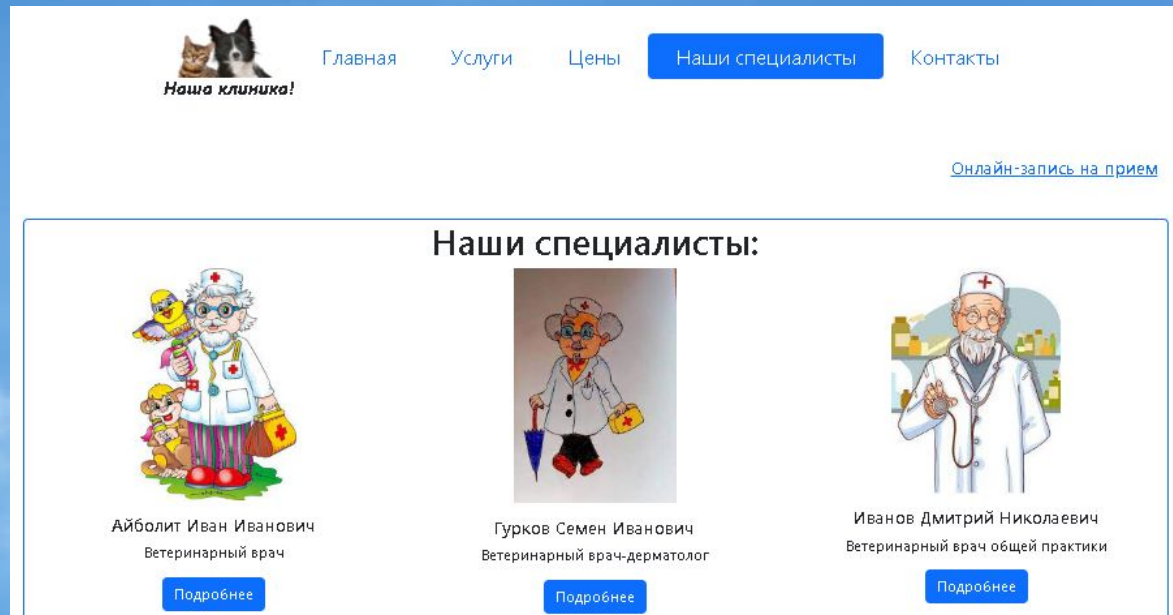
А также перейти на страницу с контактами и графиком работы.

#	Наименование услуги
1	<a href="#">Вакцинация</a>
2	<a href="#">Хирургия</a>

[Более подробно по стоимости уточните у специалиста](#)

# Описание проекта

При нажатии на кнопку 'Наши специалисты' - переходим на страницу где представлены сотрудники компании.



# Описание проекта



**Айболит Иван Иванович**

Профессия:  
Ветеринарный врач

Квалификация:  
Первая категория

Опыт работы:  
40 лет

[Назад](#)


Для предоставления  
детальной информации о  
сотрудниках  
предусмотрена кнопка  
'Подробнее'.

[Подробнее](#)



# Описание проекта

На странице 'Контакты' представлены номера телефонов компании и график работы.

[Главная](#)[Услуги](#)[Цены](#)[Наши специалисты](#)[Контакты](#)

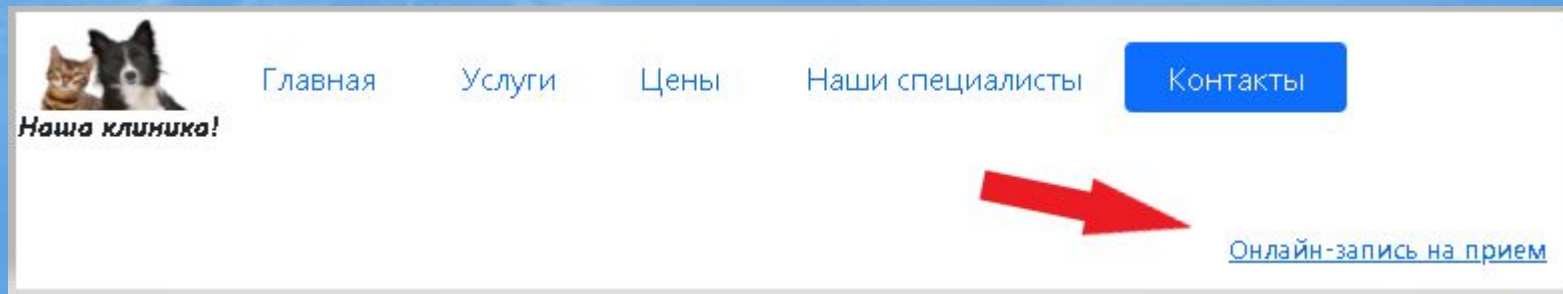
[Онлайн-запись на прием](#)

### Наши контакты:

#	Номер телефона	Время работы	Обеденный перерыв
1	+375291111111	с 8.00 до 17.00	обед с 13.00 до 14.00
2	+375292222222	с 8.00 до 17.00	обед с 13.00 до 14.00
3	+375293333333	с 8.00 до 17.00	обед с 13.00 до 14.00
4	+375294444444	с 8.00 до 17.00	обед с 13.00 до 14.00
5	+375445555555	с 8.00 до 17.00	обед с 13.00 до 14.00

# Описание проекта

В данном проекте пользователю предоставлена возможность онлайн - записи на прием.



# Описание проекта

При переходе на страницу онлайн - записи, пользователю предоставляется форма с полями которые необходимо заполнить.

**Пожалуйста заполните следующие поля:**

Введите ваше имя  
Владимир

Введите свой номер телефона  
+375291212888

Введите свой email  
Vladimirdfghan@mail.ru

Выберите категорию  
Реабилитация

Записаться

# Описание проекта

При заполнении всех полей и нажатии кнопки 'Записаться' пользователь увидит следующее сообщение.

[Онлайн-запись на прием](#)

**С Вами свяжется первый освободившийся администратор**

# Описание проекта

ООО "Clinic" УНП 770480050

[Отзывы](#)

[Просмотр регистрации](#)



На сайте пользователю  
предоставлена возможность оставить  
отзыв о компании или просмотреть  
отзывы.



# Описание проекта

На странице отзывов предусмотрена пагинация.

Причем на первой странице предоставлена форма для отзыва.

### Оставьте свой отзыв:

Ваше имя

Оставьте отзыв

Сохранить отзыв

### Отзывы:

Иван Иванович

Высококласный и отзывчивый персонал, сервис на хорошем уровне, современное оборудование. Все очень понравилось.

Александра

Очень довольна сервисом! Всем рекомендую.

Евгения

Песик остался очень довольным!!! Рекомендую.

На главную

Page 1 of 3. [Next](#)  
ООО "Clinic" УНП 770480050  
[Отзывы](#)

# Описание проекта

На следующих страницах данная форма отсутствует.

**Отзывы:**

Егор

Очень хорошая клиника!

Александр

Обратился в клинику по поводу плохого самочувствия Барсика. Очень быстро поставили диагноз и назначили лечение. Теперь Барсику очень хорошо. Рекомендую.

Виктория

Очень хороший и квалифицированный персонал. Всем рекомендую.

На главную

[Previous](#) Page 2 of 3. [Next](#)  
ООО "Clinic" УНП 770480050  
[Отзывы](#)

# Описание проекта



При переходе по ссылке 'Просмотр записи' с правами администратора, предоставляется список онлайн-записанных клиентов.

# Описание проекта

На данной странице выводится весь список онлайн-записанных клиентов.

Клиенты записанные по онлайн-записи:	
<a href="#">Весь список</a>	
<a href="#">За последний день</a>	
<a href="#">За последний месяц</a>	
<b>Владимир</b>	
Телефон: +375291212888; Email: Vladimirdfghan@mail.ru; Выбрана услуга: Реабилитация; Дата записи: 11 февраля 2023 г. 10:51;	
<b>Иван Иванович</b>	
Телефон: +375445555963; Email: FFFGFSivanIvaN@mail.ru; Выбрана услуга: Вакцинация; Дата записи: 9 февраля 2023 г. 14:58;	
<b>Семен</b>	
Телефон: +375294446588; Email: dddfhjakkksmvdvnj@mail.ru; Выбрана услуга: Хирургия; Дата записи: 9 февраля 2023 г. 14:57;	
<b>Диана</b>	
Телефон: +375445555999; Email: QWErtyREWQ.an@mail.ru; Выбрана услуга: Кардиология; Дата записи: 8 февраля 2023 г. 14:35;	
<b>Вадим</b>	
Телефон: +375293333777; Email: qwertyUYTR.an@mail.ru; Выбрана услуга: Вакцинация; Дата записи: 8 февраля 2023 г. 14:34;	
<b>Андрей</b>	
Телефон: +375291299999; Email: smvjsnmvjnmvjn@mail.ru; Выбрана услуга: Реабилитация; Дата записи: 5 февраля 2023 г. 16:54;	
<b>Николай</b>	
Телефон: +375445555599; Email: hhhhhhhhhhh.an@mail.ru; Выбрана услуга: Кардиология; Дата записи: 4 февраля 2023 г. 10:29;	

# Описание проекта

При активизации фильтра 'За последний день' - представляется список зарегистрированных за последние сутки.

## Клиенты записанные по онлайн-записи:

[Весь список](#)

[За последний день](#)

[За последний месяц](#)



**Владимир**

Телефон: +375291212888; Email: Vladimirdfghan@mail.ru; Выбрана услуга: Реабилитация; Дата записи: 11 февраля 2023 г. 10:51;

[На главную](#)



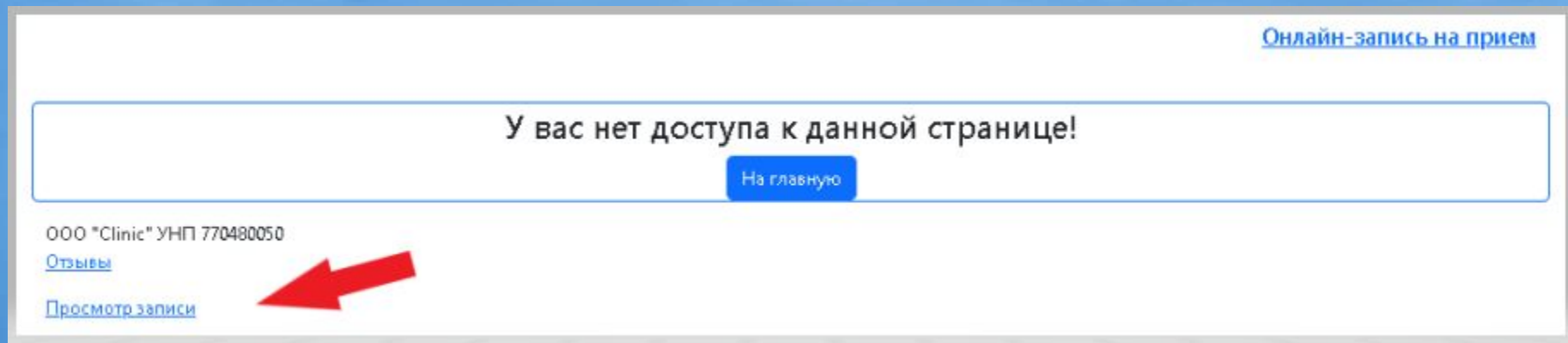
# Описание проекта

При активизации фильтра 'За последний месяц' - представляется список зарегистрированных за последний месяц.

Клиенты записанные по онлайн-записи:	
<a href="#">Весь список</a>	
<a href="#">За последний день</a>	
<a href="#">За последний месяц</a>	
<b>Владимир</b>	
Телефон: +375291212888; Email: Vladimirdfghan@mail.ru; Выбрана услуга: Реабилитация; Дата записи: 11 февраля 2023 г. 10:51;	
<b>Иван Иванович</b>	
Телефон: +375445555963; Email: FFFGFSivanIvaN@mail.ru; Выбрана услуга: Вакцинация; Дата записи: 9 февраля 2023 г. 14:58;	
<b>Семен</b>	
Телефон: +375294446588; Email: dddfhjakkksnvdvnj@mail.ru; Выбрана услуга: Хирургия; Дата записи: 9 февраля 2023 г. 14:57;	
<b>Диана</b>	
Телефон: +375445555999; Email: QWErtyREWQ.an@mail.ru; Выбрана услуга: Кардиология; Дата записи: 8 февраля 2023 г. 14:35;	
<b>Вадим</b>	
Телефон: +375293333777; Email: qwertyUYTR.an@mail.ru; Выбрана услуга: Вакцинация; Дата записи: 8 февраля 2023 г. 14:34;	
<b>Андрей</b>	
Телефон: +375291299999; Email: smjzsmjnjvnkjin@mail.ru; Выбрана услуга: Реабилитация; Дата записи: 5 февраля 2023 г. 16:54;	
<b>Николай</b>	
Телефон: +375445555599; Email: hhhhhhhhhh.an@mail.ru; Выбрана услуга: Кардиология; Дата записи: 4 февраля 2023 г. 10:29;	

# Описание проекта

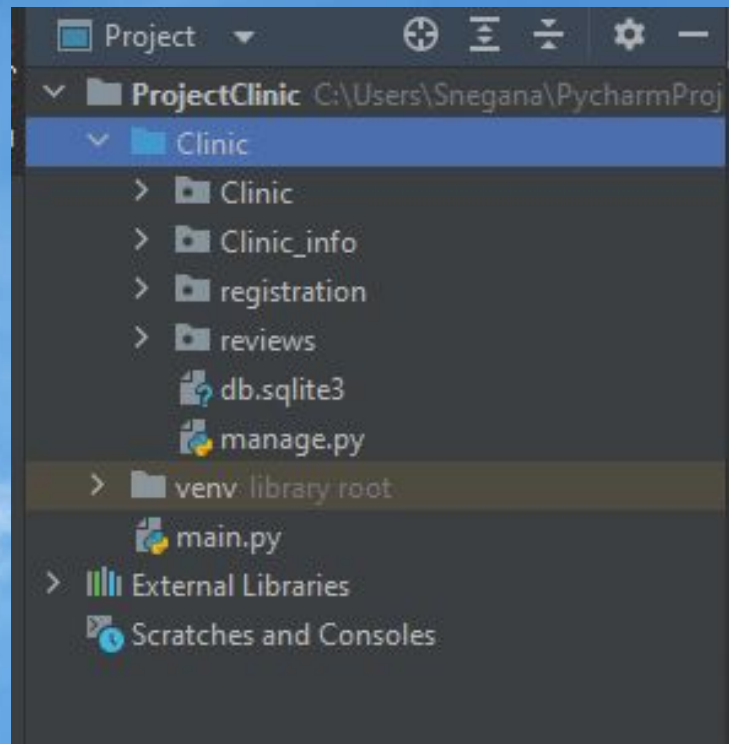
Пользователь у которого нет прав доступа к данной странице при попытке перейти по ссылке 'Просмотр записи', получит уведомление об отсутствии доступа.



# Описание структуры проекта

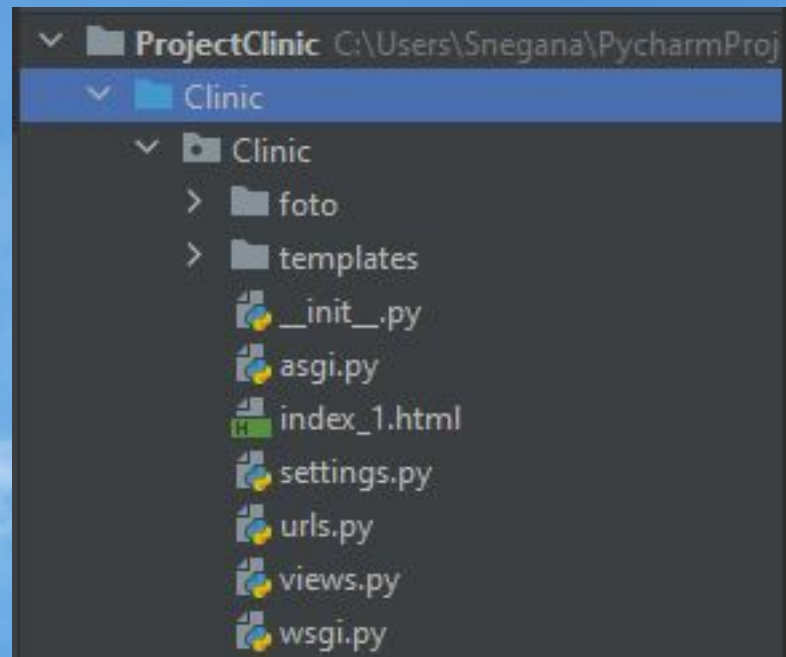
Структура проекта представлена пакетом конфигурации - Clinic, и тремя пакетами приложения:

1. Clinic\_info
2. registration
3. reviews

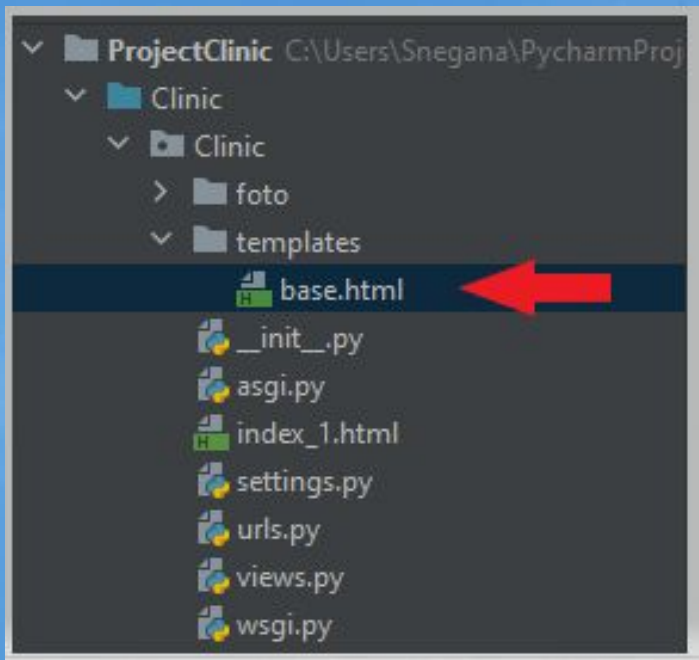


# Описание пакета конфигурации

Пакет конфигурации содержит папки и файлы, которые относятся к проекту целиком и задают его конфигурацию.



# Описание пакета конфигурации



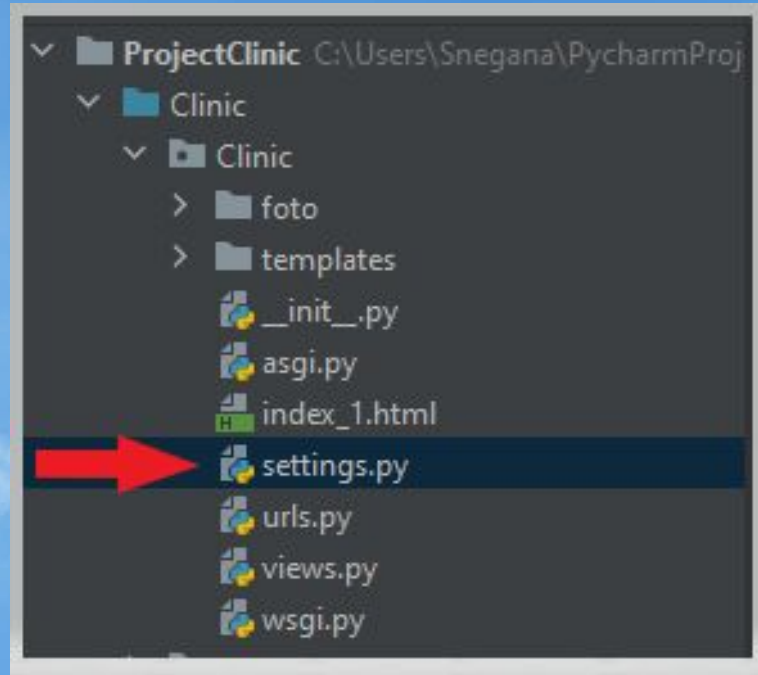
Рассмотрим более подробно некоторые элементы.

В папке templates находится файл base.html с помощью которого удастся избежать дублирования кода. Как правило в данный файл помещаются повторяющиеся на каждой странице части html кода (например 'header' и 'footer').



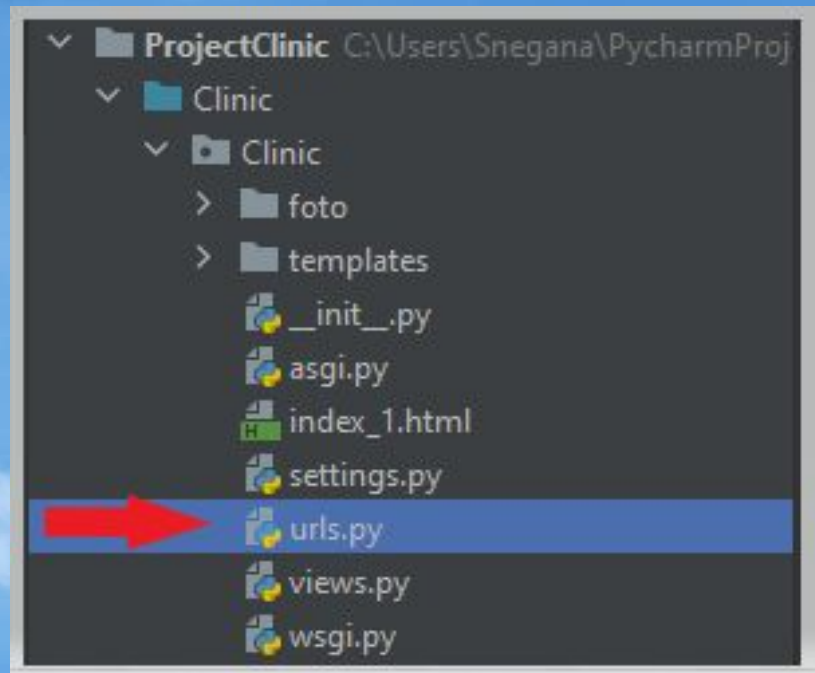
# Описание пакета конфигурации

settings.py - модуль с настройками проекта. Включает описание конфигурации базы данных проекта, пути ключевых папок, важные параметры, связанные с безопасностью и др.



# Описание пакета конфигурации

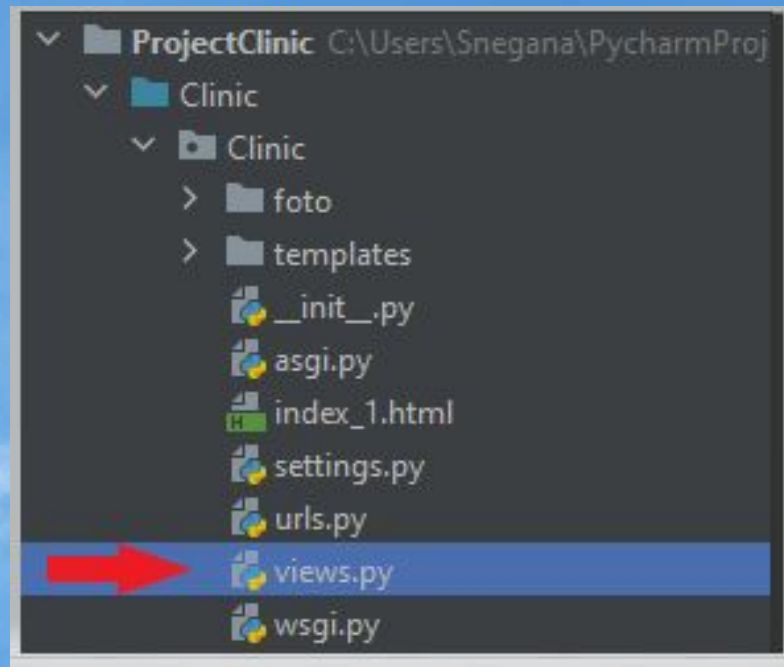
urls.py - модуль с маршрутами уровня проекта.





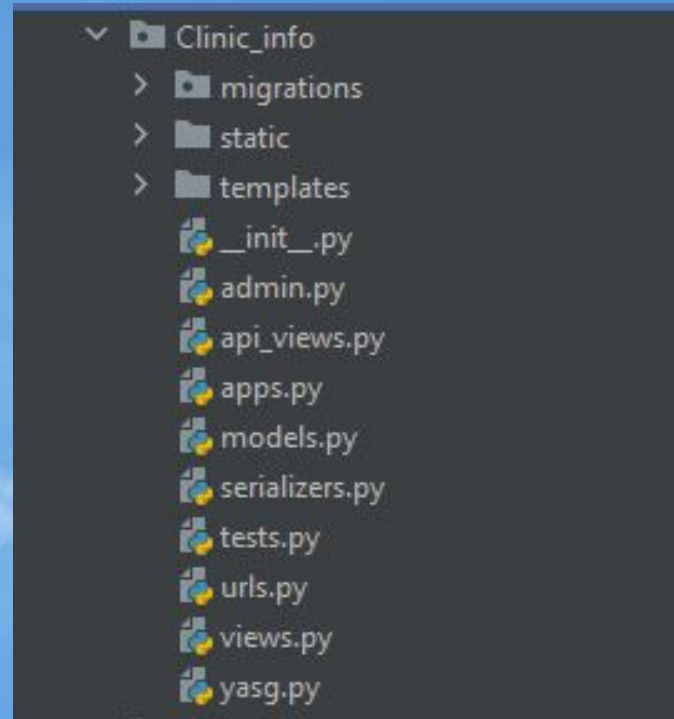
# Описание пакета конфигурации

views.py - файл, в котором находятся функции представления (код, запускаемый при обращении по интернет-адресу определенного формата и в ответ выводящий на экран определенную веб-страницу).

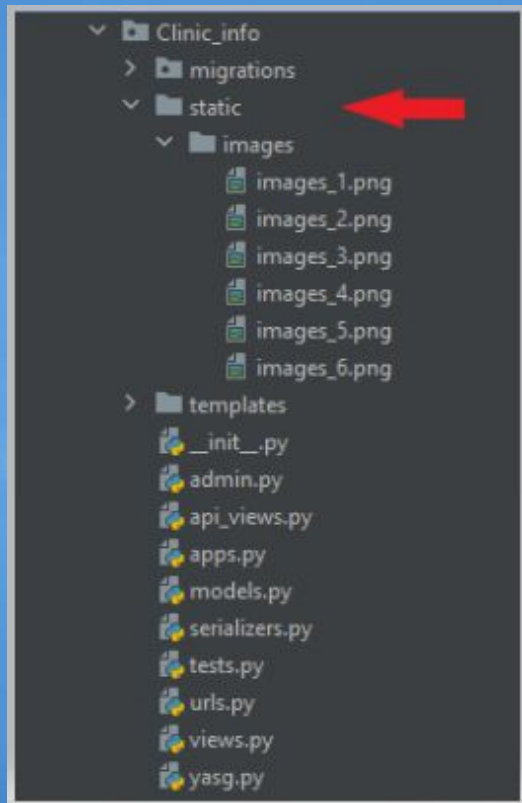


# Описание приложения Clinic\_info

Структура приложения Clinic\_info, во многом схожа со структурой пакета конфигурации, но есть и свои отличия. Далее более подробно рассмотрим работу этого приложения.

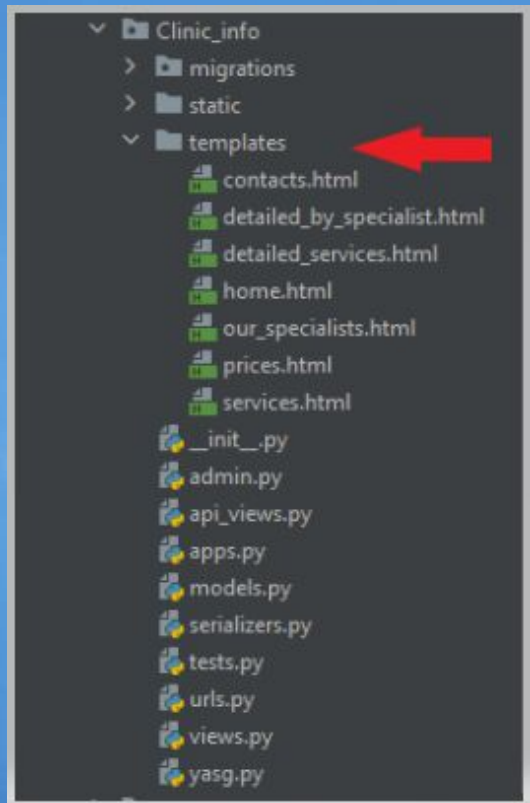


# Описание приложения Clinic\_info



В пакете приложения создана папка static, используемая для хранения статических файлов используемых в html - шаблонах.

# Описание приложения Clinic\_info



В папке templates находятся все html - файлы используемые приложением Clinic\_info.

# Описание приложения Clinic\_info

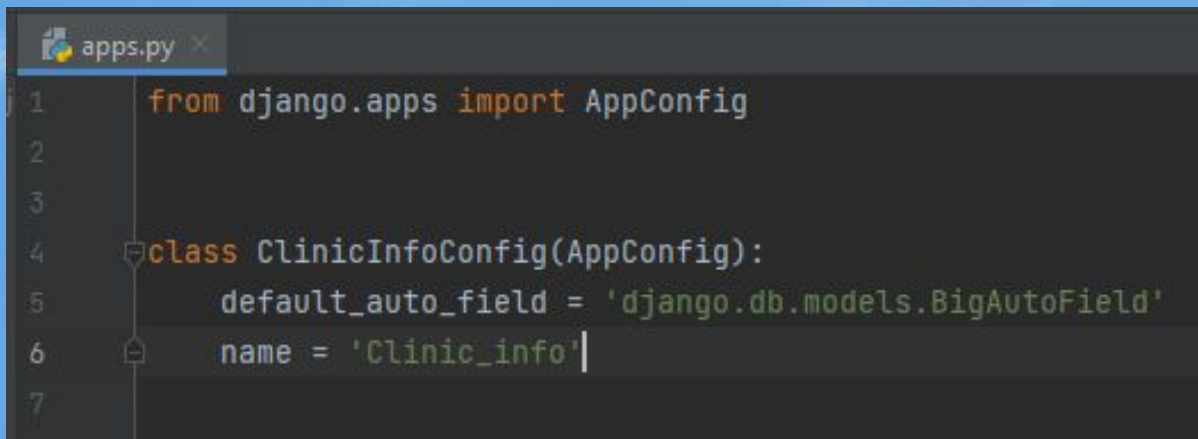
В admin.py помещен код выполняющий регистрацию моделей на административном сайте.

```
admin.py x
1  from django.contrib import admin
2
3  from .models import *
4
5
6  admin.site.register(Employee)
7  admin.site.register(Service)
8  admin.site.register(Professions)
9  admin.site.register(Client)
10 admin.site.register(Advantages)
11 admin.site.register(Contacts)
12 admin.site.register(Information)
13
```



# Описание приложения Clinic\_info

Файл apps.py пакета приложения содержит объявление конфигурационного класса, хранящего настройки приложения (полный путь к пакету приложения и настройка для автоматического создания первичного ключа в моделях).



```
apps.py x
1  from django.apps import AppConfig
2
3
4  class ClinicInfoConfig(AppConfig):
5      default_auto_field = 'django.db.models.BigAutoField'
6      name = 'Clinic_info'
7
```

# Описание приложения Clinic\_info

Чтобы приложение успешно работало, оно должно быть зарегистрировано в списке приложений INSTALLED\_APPS, который находится в параметрах проекта.

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'Clinic_info.apps.ClinicInfoConfig',  
    'drf_yasg',  
    'rest_framework',  
    'django_filters',  
    'reviews.apps.ReviewsConfig',  
    'registration.apps.RegistrationConfig',  
]
```

# Описание приложения Clinic\_info

Модель - это класс, описывающий определенную таблицу в базе данных. В файле models.py объявлены модели пакета приложения Clinic\_info. Такие как Employee:

```
models.py x
1  from django.db import models
2
3
4  class Employee(models.Model):
5      surname = models.CharField(max_length=50, verbose_name='Фамилия')
6      name = models.CharField(max_length=30, verbose_name='Имя')
7      patronymic = models.CharField(max_length=50, verbose_name='Отчество')
8      profession = models.ForeignKey('Professions', on_delete=models.PROTECT, verbose_name='Профессия')
9      qualification = models.CharField(max_length=50, verbose_name='Квалификация')
10     experience = models.CharField(max_length=20, verbose_name='Опыт')
11     image = models.ImageField(upload_to='images/', verbose_name='Фотография')
12     created_on = models.DateTimeField(auto_now_add=True)
13     last_modified = models.DateTimeField(auto_now=True)
14
15     def __str__(self):
16         return self.surname
17
18     class Meta:
19         verbose_name_plural = 'Сотрудники'
20         verbose_name = 'Сотрудника'
21         ordering = ['surname']
```

# Описание приложения Clinic\_info

## Модель таблицы Service:

```
class Service(models.Model):
    service = models.CharField(max_length=50, verbose_name='Услуга')
    description = models.TextField(verbose_name='Описание')
    price = models.DecimalField(max_digits=8, decimal_places=2, verbose_name='Цена')
    created_on = models.DateTimeField(auto_now_add=True)
    last_modified = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.service

    class Meta:
        verbose_name_plural = 'Услуги'
        verbose_name = 'Услуга'
        ordering = ['service']
```

# Описание приложения Clinic\_info

Модель таблицы Professions:

```
class Professions(models.Model):
    profession = models.CharField(max_length=50, verbose_name='Профессия')
    created_on = models.DateTimeField(auto_now_add=True)
    last_modified = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.profession

    class Meta:
        verbose_name_plural = 'Профессии'
        verbose_name = 'Профессия'
        ordering = ['profession']
```

# Описание приложения Clinic\_info

## Модель таблицы Client:

```
class Client(models.Model):
    name = models.CharField(max_length=50, verbose_name='Имя')
    telephone = models.CharField(max_length=20, verbose_name='Телефон')
    email = models.EmailField(max_length=100, verbose_name='Email')
    service = models.ForeignKey('Service', on_delete=models.PROTECT, verbose_name='Услуга')
    created_on = models.DateTimeField(auto_now_add=True)
    last_modified = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.name

    class Meta:
        verbose_name_plural = 'Клиенты'
        verbose_name = 'Клиент'
        ordering = ['service']
```

# Описание приложения Clinic\_info

Модель таблицы Advantages:

```
class Advantages(models.Model):
    advantage = models.TextField(verbose_name='Преимущество')

    def __str__(self):
        return self.advantage

    class Meta:
        verbose_name_plural = 'Преимущества'
        verbose_name = 'Преимущество'
        ordering = ['id']
```

# Описание приложения Clinic\_info

Модель таблицы Contacts:

```
class Contacts(models.Model):
    telephone = models.CharField(max_length=20, verbose_name='Телефон')

    def __str__(self):
        return self.telephone

    class Meta:
        verbose_name_plural = 'Контакты'
        verbose_name = 'Контакт'
        ordering = ['id']
```



# Описание приложения Clinic\_info

Модель таблицы Information:

```
class Information(models.Model):
    address = models.CharField(max_length=255, verbose_name='Адрес')
    working_hours = models.CharField(max_length=50, verbose_name='Время работы')

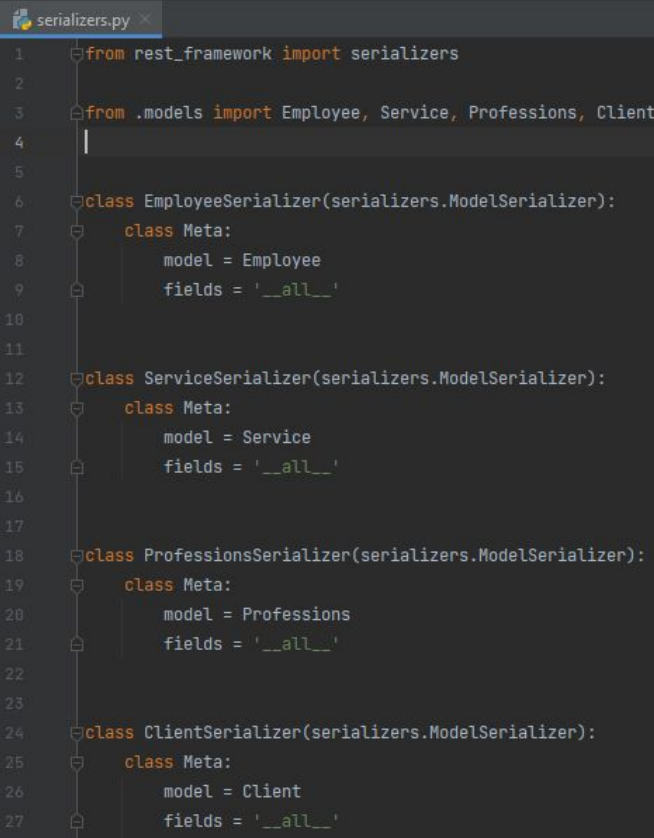
    def __str__(self):
        return self.address, self.working_hours

    class Meta:
        verbose_name_plural = 'Информация'
        verbose_name = 'Информацию'
        ordering = ['id']
```

# Описание приложения Clinic\_info

В файле `serializers.py` прописаны сериализаторы, которые позволяют преобразовать сложные данные, такие как наборы запросов `QuerySet` и объекты моделей, в типы данных Python, которые затем можно легко преобразовать в JSON, XML или другие content types.

В данном проекте при определении сериализатора используется вложенный класс `Meta` со следующими атрибутами: `model` - это django-модель которую, будет обслуживать сериализатор, `fields` - поля django-модели, для которых будут созданы соответствующие поля сериализатора.



```
serializers.py
1  from rest_framework import serializers
2
3  from .models import Employee, Service, Professions, Client
4
5
6  class EmployeeSerializer(serializers.ModelSerializer):
7      class Meta:
8          model = Employee
9          fields = '__all__'
10
11
12  class ServiceSerializer(serializers.ModelSerializer):
13      class Meta:
14          model = Service
15          fields = '__all__'
16
17
18  class ProfessionsSerializer(serializers.ModelSerializer):
19      class Meta:
20          model = Professions
21          fields = '__all__'
22
23
24  class ClientSerializer(serializers.ModelSerializer):
25      class Meta:
26          model = Client
27          fields = '__all__'
```

# Описание приложения Clinic\_info

В файле urls.py пакета приложения Clinic\_info прописана регистрация функций - представлений для API (более подробно рассмотрено далее), а также соответствующий список маршрутов в переменной urlpatterns.

```
urls.py
1 from django.urls import path, include
2 from rest_framework import routers
3
4 from .yasg import urlpatterns as swag_urls
5
6 from .views import *
7 from .api_views import *
8 from reviews.api_views import ReviewsAPIView
9
10 router = routers.DefaultRouter()
11
12 router.register(r'employee', EmployeeAPIView)
13 router.register(r'service', ServiceAPIView)
14 router.register(r'professions', ProfessionsAPIView)
15 router.register(r'client', ClientAPIView)
16 router.register(r'reviews', ReviewsAPIView)
17
18
19 urlpatterns = [
20     path('clinic/', clinic, name='clinic'),
21     path('services/', services, name='services'),
22     path('detailed_services/<int:pk>/', detailed_services,
23         name='detailed_services'),
24     path('prices/', prices, name='prices'),
25     path('contacts/', all_contacts, name='contacts'),
26     path('our_specialists/', our_specialists,
27         name='our_specialists'),
28     path('detailed_by_specialist/<int:pk>/', detailed_by_specialist,
29         name='detailed_by_specialist'),
30     path('api/', include(router.urls)),
31     path('api-auth/', include('rest_framework.urls'))
32 ]
33
34 urlpatterns += swag_urls
```

# Описание приложения Clinic\_info

В файле `views.py` прописаны функции - представления приложения `Clinic_info`.

Рассмотрим более подробно их работу. Функция `clinic` формирует `QuerySet` из объектов таблицы `Service`. Посредством метода `values()` берем из базы данных только необходимые нам поля и с помощью метода `order_by()`, сортируем по полю `'pk'`. Из полученных данных формируем словарь на который ссылается переменная `context`. Данная переменная, `request` и название `html` шаблона передаются функции `render()`, которую в свою очередь возвращает функция `clinic()`.

```
views.py
1  from django.shortcuts import render
2
3  from .models import *
4
5
6  def clinic(request):
7      services = Service.objects.values('pk', 'service', 'description').order_by('pk')
8      context = {'services': services}
9      return render(request, 'home.html', context)
10
```

# Описание приложения Clinic\_info

Функция `services` формирует `QuerySet` из объектов таблицы `Service`.  
Посредством метода `values()` берем из базы данных только необходимые нам поля ('pk', 'service', 'description'). Из полученных данных формируем словарь на который ссылается переменная `context`. Как и в предыдущей функции, данная переменная, `request` и название `html` шаблона передаются функции `render()`.

```
12 def services(request):
13     all_services = Service.objects.values('pk', 'service', 'description')
14     context = {'all_services': all_services}
15     return render(request, 'services.html', context)
16
```

# Описание приложения Clinic\_info

Функция `detailed_services` в свою очередь имеет два параметра, `request` и `pk`. В данной функции посредством метода `get()` берем из базы данных только один объект по значению `pk`. Из полученных данных формируем словарь на который ссылается переменная `context`. Как и ранее, данная переменная, `request` и название html шаблона передаются функции `render()`.

```
18 def detailed_services(request, pk):
19     service = Service.objects.get(pk=pk)
20     context = {'service': service}
21     return render(request, 'detailed_services.html', context)
22
```

# Описание приложения Clinic\_info

Функция `prices` формирует `QuerySet` из объектов таблицы `Service`. Посредством метода `all()` берем из базы данных все поля, которые с помощью метода `order_by()` сортируем по `'id'`. Из полученных данных формируем словарь на который ссылается переменная `context`. Как и в предыдущей функции, данная переменная, `request` и название `html` шаблона передаются функции `render()`.

```
24 def prices(request):  
25     services = Service.objects.all().order_by("id")  
26     context = {'services': services}  
27     return render(request, 'prices.html', context)  
28
```



# Описание приложения Clinic\_info

Функция `our_specialists` формирует `QuerySet` из объектов таблицы `Employee`.  
Посредством метода `all()` берем из базы данных все поля. Из полученных данных формируем словарь на который ссылается переменная `context`.  
Переменная `context`, `request` и название `html` шаблона передаются функции `render()`, которая возвращается функцией `our_specialists`.

```
30 def our_specialists(request):  
31     specialists = Employee.objects.all()  
32     context = {'specialists': specialists}  
33     return render(request, 'our_specialists.html', context)  
34
```



# Описание приложения Clinic\_info

Функция `detailed_by_specialist` как и функция `detailed_services` имеет два параметра, `request` и `pk`. В данной функции посредством метода `get()` берем из базы данных (таблица `Employee`) только один объект по значению `pk`. Из полученных данных формируем словарь на который ссылается переменная `context`. Как и ранее, данная переменная, `request` и название `html` шаблона передаются функции `render()`.

```
36 def detailed_by_specialist(request, pk):
37     specialist = Employee.objects.get(pk=pk)
38     context = {'specialist': specialist}
39     return render(request, 'detailed_by_specialist.html', context)
40
```

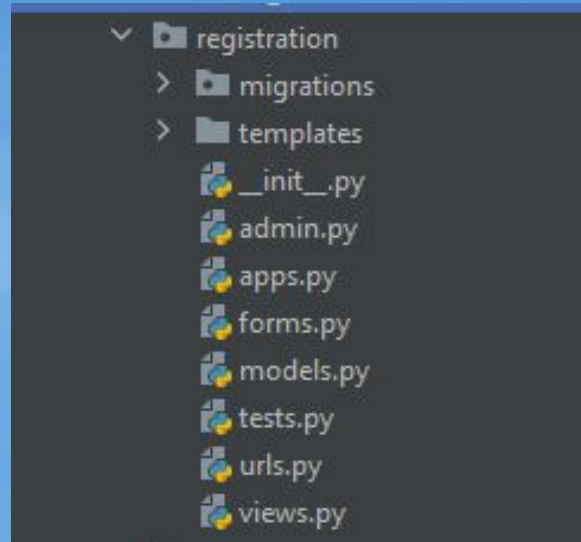
# Описание приложения Clinic\_info

Функция `all_contacts` формирует два объекта `QuerySet` из данных таблиц `Contacts` и `Information`. В данной функции использовались методы `all()`, `values()`, `order_by()`, а также строковые методы `partition()` и `rstrip()`. Из полученных данных формируем словарь на который ссылается переменная `context`. Переменная `context`, `request` и название `html` шаблона передаются функции `render()`, которая возвращается функцией `all_contacts`.

```
42 def all_contacts(request):
43     contacts = Contacts.objects.all().order_by("id")
44     information = Information.objects.values("working_hours")[0]
45     all_info = information['working_hours'].partition('(')
46     working_hours = all_info[0]
47     lunch_break = all_info[2].rstrip('(')
48     context = {'contacts': contacts,
49               'information': information,
50               'working_hours': working_hours,
51               'lunch_break': lunch_break}
52     return render(request, 'contacts.html', context)
53
```

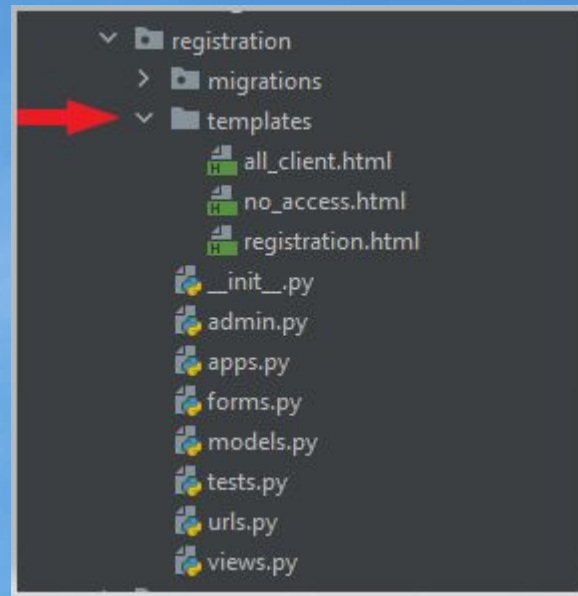
# Описание приложения registration

На данном слайде представлена структура приложения registration. Далее более подробно рассмотрена работа и особенности этого приложения.



# Описание приложения registration

Как и в приложении Clinic\_info, в папке templates находятся все html - файлы используемые приложением registration.



# Описание приложения registration

Файл apps.py пакета приложения содержит объявление конфигурационного класса.

```
apps.py x
1  from django.apps import AppConfig
2
3
4  class RegistrationConfig(AppConfig):
5      default_auto_field = 'django.db.models.BigAutoField'
6      name = 'registration'
7
```

И собственно регистрация в списке `INSTALLED_APPS`, в файле `settings.py`, который находится в пакете конфигурации.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'Clinic_info.apps.ClinicInfoConfig',
    'drf_yasg',
    'rest_framework',
    'django_filters',
    'reviews.apps.ReviewsConfig',
    'registration.apps.RegistrationConfig',
]
```

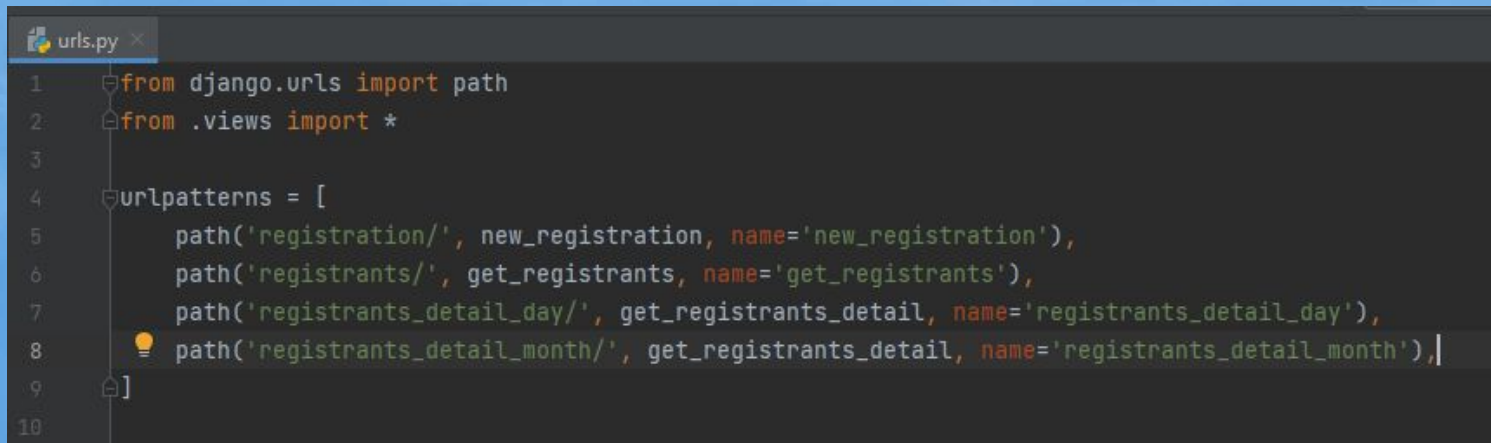
# Описание приложения registration

В файле forms.py прописан класс RegistrationForm, который определяет функциональность формы, применяемой в функции представления new\_registration (более подробно рассматривается далее).

```
forms.py
1  from django import forms
2
3  from Clinic_info.models import Service, Client
4
5
6  class RegistrationForm(forms.ModelForm):
7      def __init__(self, *args, **kwargs):
8          super().__init__(*args, **kwargs)
9          self.fields['service'].empty_label = "Категория не выбрана"
10
11      class Meta:
12          model = Client
13          fields = '__all__'
14
```

# Описание приложения registration

Собственно, в файле `urls.py` приложения `registration`, так же прописан список маршрутов, который присвоен переменной `urlpatterns`.



```
1 from django.urls import path
2 from .views import *
3
4 urlpatterns = [
5     path('registration/', new_registration, name='new_registration'),
6     path('registrants/', get_registrants, name='get_registrants'),
7     path('registrants_detail_day/', get_registrants_detail, name='registrants_detail_day'),
8     path('registrants_detail_month/', get_registrants_detail, name='registrants_detail_month'),
9 ]
```

# Описание приложения registration

Файл views.py содержит функции - представления приложения registration.

Функция new\_registration создает объект класса RegistrationForm, на который ссылается переменная form. Также здесь объявляется переменная - флаг registration. Созданные переменные, request и название html шаблона передаются функции render().

```
views.py
1  from django.shortcuts import render
2      from datetime import datetime
3      from Clinic_info.models import Client
4      from registration.forms import RegistrationForm
5
6
7  def new_registration(request):
8      form = RegistrationForm()
9      registration = 0
10
11     if request.method == 'POST':
12         form = RegistrationForm(request.POST)
13         if form.is_valid():
14             client = Client(
15                 name=form.cleaned_data["name"],
16                 telephone=form.cleaned_data["telephone"],
17                 email=form.cleaned_data["email"],
18                 service=form.cleaned_data["service"],
19             )
20             client.save()
21             registration = 1
22
23     context = {'form': form, 'registration': registration}
24     return render(request, 'registration.html', context)
```



# Описание приложения registration

Функция представления `new_registration` посредством html-шаблона выводит пользователю форму. После того как пользователь заполнит все поля и нажмет кнопку 'Записаться', функция проверяет метод запроса, если используется POST-запрос, тогда создается объект класса `RegistrationForm`, которому в качестве аргумента передается `request.POST`. После проверки на валидацию данных, создается объект класса `Client`, на который ссылается переменная `client`. Далее с помощью метода `save()` данные сохраняются в базу данных, а переменная `registration` принимает значение 1.

# Описание приложения registration

Функция `get_registrants` проверяет наличие доступа к информации. Если пользователь имеет доступ, функция формирует `QuerySet` из объектов таблицы `Client`. Посредством метода `all()` берем из базы данных все поля, которые с помощью метода `order_by()` сортируем по полю `'created_on'`. Из полученных данных формируем словарь на который ссылается переменная `context`. Переменная `context`, `request` и название `html` шаблона передаются функции `render()`.

Если пользователь не проходит проверку, то посредством `html` выводим сообщение об отсутствии пр

```
26 def get_registrants(request):
27     if request.user.is_authenticated:
28         all_client = Client.objects.all().order_by('-created_on')
29         count_client = Client.objects.count()
30         context = {'all_client': all_client, 'count_client': count_client}
31         return render(request, 'all_client.html', context)
32
33     else:
34         return render(request, 'no_access.html')
35
```

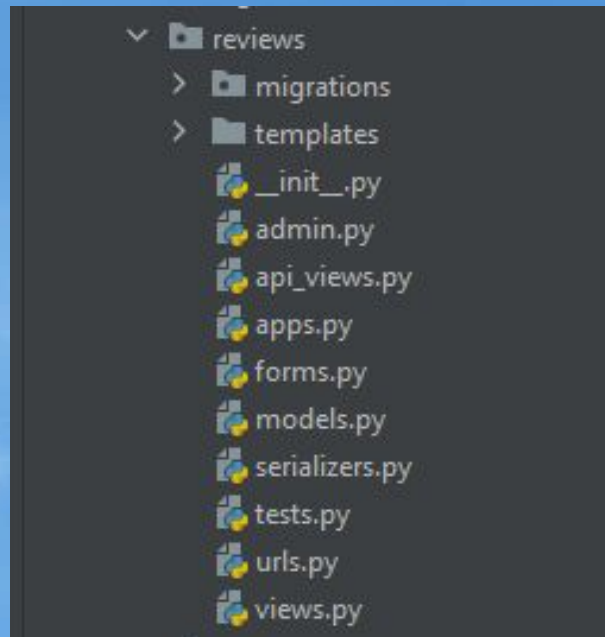
# Описание приложения registration

Функция `get_registrants_detail` формирует `QuerySet` в зависимости от значения в атрибуте `path_info`. Если значение в данном атрибуте равно - `/registrants_detail_day/`, данные берутся посредством метода `filter()` и модуля `datetime` за последний день, которые с помощью метода `order_by()` сортируем по полю `'created_on'`. В противном случае данные берутся за последний месяц. Из полученных данных формируем словарь на который ссылается переменная `context`. Переменная `context`, `request` и название `html` шаблона передаются функции `render()`.

```
37 def get_registrants_detail(request):
38     current_datetime = datetime.now()
39     if request.path_info == '/registrants_detail_day/':
40         all_client = Client.objects.filter(created_on__day=current_datetime.day).order_by('-created_on')
41     else:
42         all_client = Client.objects.filter(created_on__month=current_datetime.month).order_by('-created_on')
43
44     context = {'all_client': all_client}
45     return render(request, 'all_client.html', context)
```

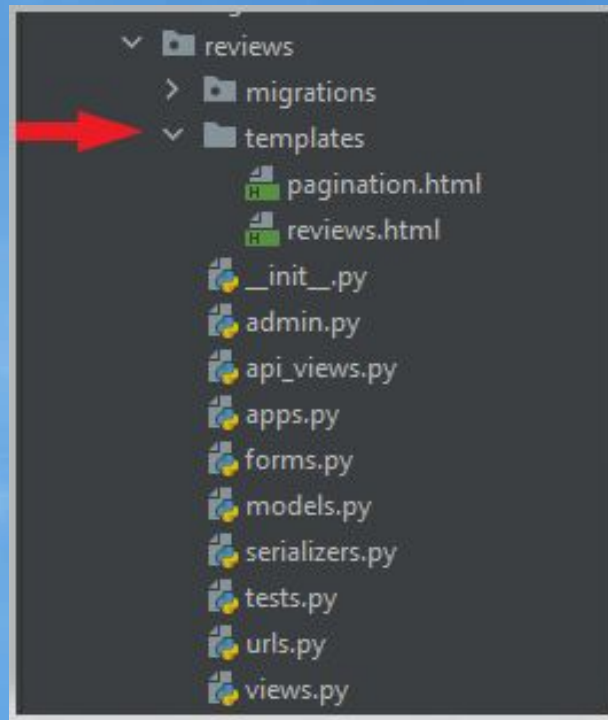
# Описание приложения reviews

На данном слайде представлена структура приложения reviews. Далее более подробно рассмотрена работа и особенности этого приложения.



# Описание приложения reviews

Данное приложение также содержит каталог `templates`, в котором размещены необходимые для работы html - файлы.



# Описание приложения reviews

В `admin.py` помещен код выполняющий регистрацию модели `Reviews` на административном сайте.

Файл `apps.py` содержит объявление конфигурационного класса `ReviewsConfig`, хранящего настройки приложения. Данное приложение также зарегистрировано в файле `settings.py` пакета конфигурации.

```
admin.py X
1  from django.contrib import admin
2
3  from .models import *
4
5  admin.site.register(Reviews)
6
```

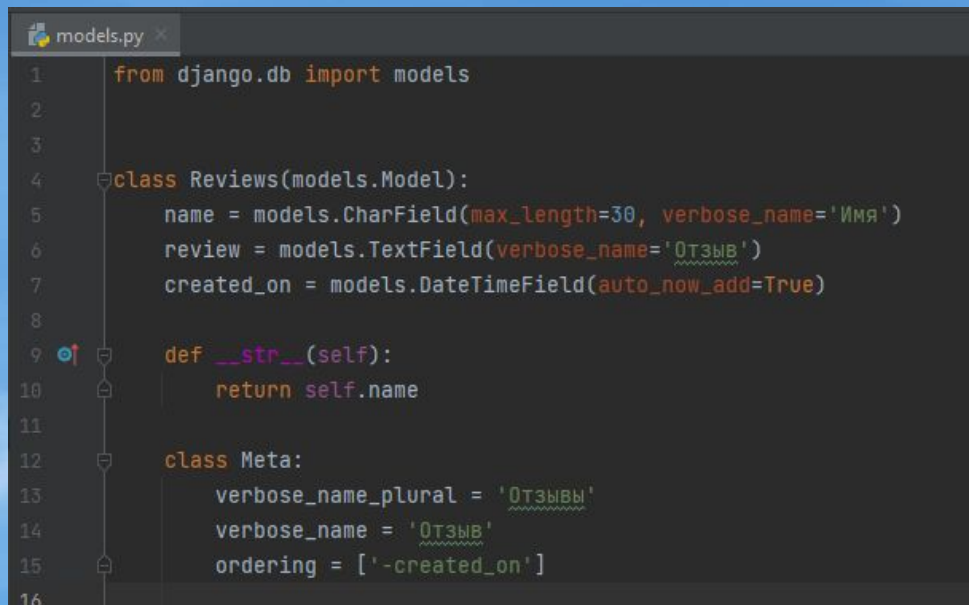
```
apps.py X
1  from django.apps import AppConfig
2
3
4  class ReviewsConfig(AppConfig):
5      default_auto_field = 'django.db.models.BigAutoField'
6      name = 'reviews'
```





# Описание приложения reviews

В файле `models.py` приложения `reviews` объявлен класс `Reviews` для описания модели. Для представления отдельного поля таблицы в модели созданы атрибуты класса, которым присвоены экземпляры класса, представляющие поле нужного типа.



```
models.py
1  from django.db import models
2
3
4  class Reviews(models.Model):
5      name = models.CharField(max_length=30, verbose_name='Имя')
6      review = models.TextField(verbose_name='Отзыв')
7      created_on = models.DateTimeField(auto_now_add=True)
8
9  def __str__(self):
10     return self.name
11
12  class Meta:
13     verbose_name_plural = 'Отзывы'
14     verbose_name = 'Отзыв'
15     ordering = ['-created_on']
16
```



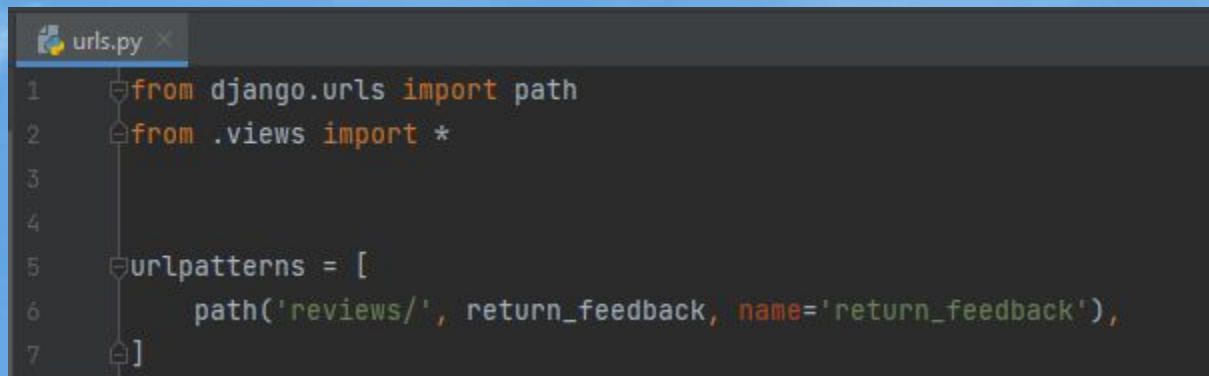
# Описание приложения reviews

В файле `serializers.py` прописан сериализатор, связанный с моделью `Reviews`. Данный сериализатор 'умеет' самостоятельно извлекать данные из модели и сохранять в ней данные, полученные от фронтенда.

```
serializers.py x
1  from rest_framework import serializers
2
3  from reviews.models import Reviews
4
5
6  class ReviewsSerializer(serializers.ModelSerializer):
7      class Meta:
8          model = Reviews
9          fields = '__all__'
10
```

# Описание приложения reviews

Файл `urls.py` приложения `reviews` содержит маршрут, связанный с функцией представления `return_feedback`.



```
urls.py x
1  from django.urls import path
2  from .views import *
3
4
5  urlpatterns = [
6      path('reviews/', return_feedback, name='return_feedback'),
7  ]
```

# Описание приложения reviews

Функция представления `add_a_reviews` возвращает словарь с формой `ReviewsForm`. После того как пользователь заполнит все поля и нажмет кнопку 'Сохранить отзыв', функция проверяет метод запроса, если используется POST-запрос, тогда создается объект класса `ReviewsForm`, которому в качестве аргумента передается `request.POST`. После проверки на валидацию данных, создается объект класса `Reviews`, на который ссылается переменная `comment`. Далее с помощью метода `save()` данные сохраняются в базу данных.

```
views.py
1  from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger
2  from django.shortcuts import render
3
4  from .forms import ReviewsForm
5  from .models import Reviews
6
7
8  def add_a_review(request):
9      form = ReviewsForm()
10     if request.method == 'POST':
11         form = ReviewsForm(request.POST)
12         if form.is_valid():
13             comment = Reviews(
14                 name=form.cleaned_data["name"],
15                 review=form.cleaned_data["review"],
16             )
17             comment.save()
18
19     context_form = {'form': form}
20
21     return context_form
22
```

# Описание приложения reviews

В функции представления `return_feedback` прописан код разбиения на страницы, на основе встроенного в Django класса (более подробно рассмотрен на следующем слайде). Так же здесь вызывается функция `add_a_reviews`, словарь которой объединяется со словарем функции `return_feedback`. Далее переменная `context`, `request` и название html шаблона передаются функции `render()`.

```
24 def return_feedback(request):
25     form = add_a_review(request)
26     reviews = Reviews.objects.all()
27     paginator = Paginator(reviews, 3)
28     page = request.GET.get('page')
29     try:
30         posts = paginator.page(page)
31     except PageNotAnInteger:
32         posts = paginator.page(1)
33     except EmptyPage:
34         posts = paginator.page(paginator.num_pages)
35
36     context = {'page': page, 'posts': posts, **form}
37
38     return render(request, 'reviews.html', context)
39
```

# Описание приложения reviews

Рассмотрим более подробно работу пагинации.

В начале мы создаем экземпляр класса пагинатор с количеством объектов, которые должны отображаться на каждой странице. Затем получаем параметр GET, который указывает номер текущей страницы. Далее получаем объекты для требуемой страницы, вызывая `page()` метод `Paginator`. Если параметр страницы не является целым числом, мы извлекаем первую страницу результатов. Если этот параметр является числом, превышающим последнюю страницу результатов, мы извлекаем последнюю страницу. Последним шагом возвращаем страницы в соответствии с указанным шаблоном.

# Описание приложения reviews

Файл pagination.html представляет собой шаблон для отображения paginator. Это отдельный файл, чтобы его можно было включить в любой шаблон, использующий разбиение на страницы.

```
1 <div class="pagination">
2   <span class="step-links">
3     {% if page.has_previous %}
4       <a href="?page={{ page.previous_page_number }}">Предыдущая</a>
5     {% endif %}
6     <span class="current">
7       Страница {{ page.number }} из {{ page.paginator.num_pages }}.
8     </span>
9     {% if page.has_next %}
10      <a href="?page={{ page.next_page_number }}">Следующая</a>
11    {% endif %}
12  </span>
13 </div>
```

Данным кодом в нижней части шаблона reviews.html мы включаем шаблон pagination.html.

```
46 <div class="container">
47   {% block content %}
48
49   {% include "pagination.html" with page=posts %}
50
51   {% endblock %}
52 </div>
```

# Административный веб-сайт Django

Django включает в свой состав полностью готовый к работе административный веб-сайт, предоставляющий доступ к любым внутренним данным, позволяющий пополнять, править и удалять их, гибко настраиваемый и исключительно удобный в использовании. Доступ к административному сайту Django имеют только суперпользователь и пользователи со статусом персонала.

# Административный веб-сайт Django

Прежде чем работать с административным сайтом, необходимо осуществить следующие подготовительные операции:

- 1) Проверить наличие маршрута (в модуле `urls.py` пакета конфигурации), который свяжет выбранный шаблонный путь (обычно используется `admin/`) со списком маршрутов, записанных в атрибуте `urls` объекта административного сайта, который хранится в переменной `site` из модуля `django.contrib.admin`:

```
path('admin/', admin.site.urls),
```



# Административный веб-сайт Django

2) Выполнить миграции.

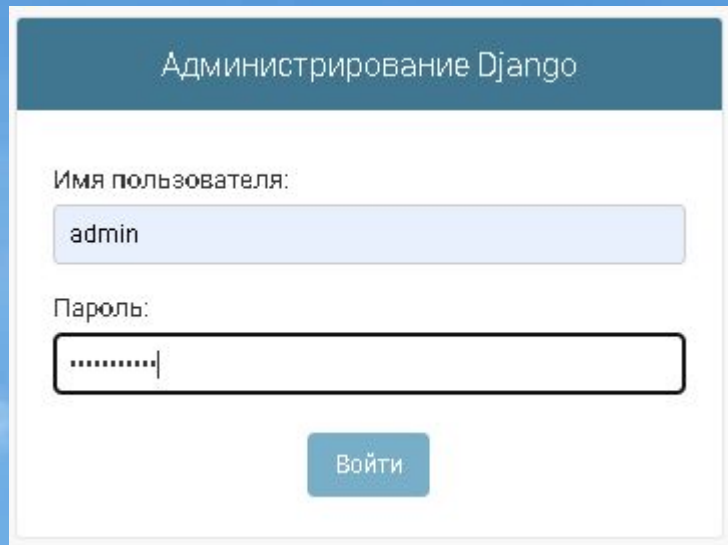
3) Создать суперпользователя.

Чтобы с данными, хранящимися в определенной модели, можно было работать посредством административного сайта, модель нужно зарегистрировать на сайте. Делается это вызовом метода `register(<модель>)` объекта административного сайта в файле `admin.py`.

```
admin.py x
1  from django.contrib import admin
2
3  from .models import *
4
5
6  admin.site.register(Employee)
7  admin.site.register(Service)
8  admin.site.register(Professions)
9  admin.site.register(Client)
10 admin.site.register(Advantages)
11 admin.site.register(Contacts)
12 admin.site.register(Information)
13
```

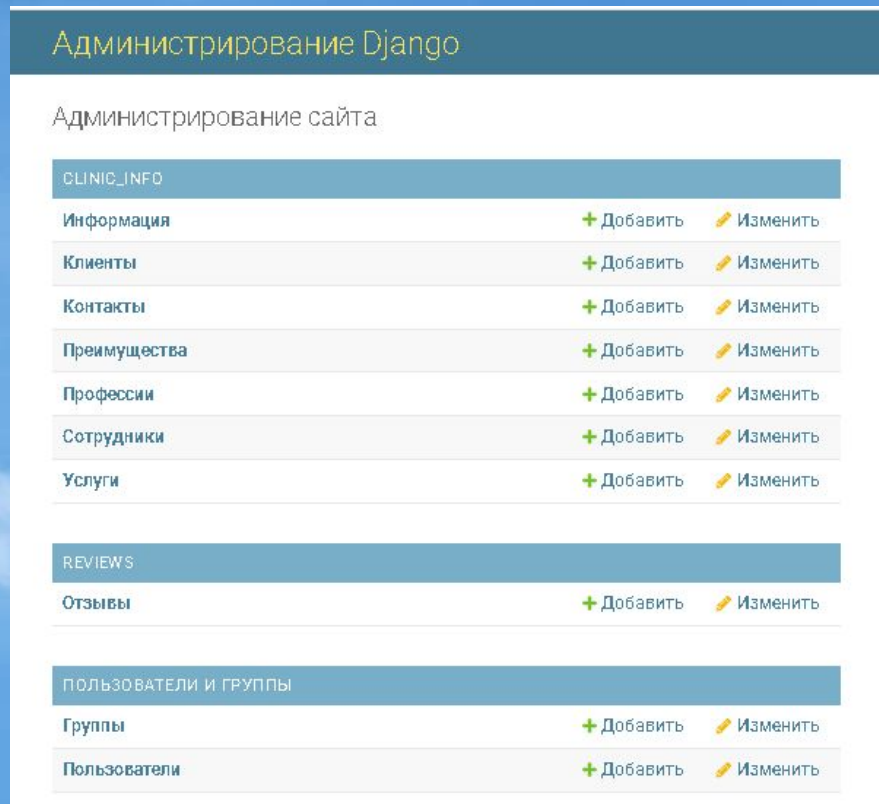
# Административный веб-сайт Django

После запуска отладочного веб-сервера и перейдя по интернет-адресу <http://127.0.0.1:8000/admin/>, будет выведена страница входа с формой, в которой нужно набрать имя и пароль, введенные при создании суперпользователя, и нажать кнопку Войти.

The image shows a screenshot of the Django Admin login interface. At the top, there is a dark teal header bar with the text "Администрирование Django" in white. Below the header, the page has a light cream background. The login form consists of two input fields: "Имя пользователя:" (Username) and "Пароль:" (Password). The username field is a light blue rounded rectangle containing the text "admin". The password field is a white rounded rectangle with a black border, containing a series of dots. Below these fields is a teal button with the text "Войти" (Login) in white.

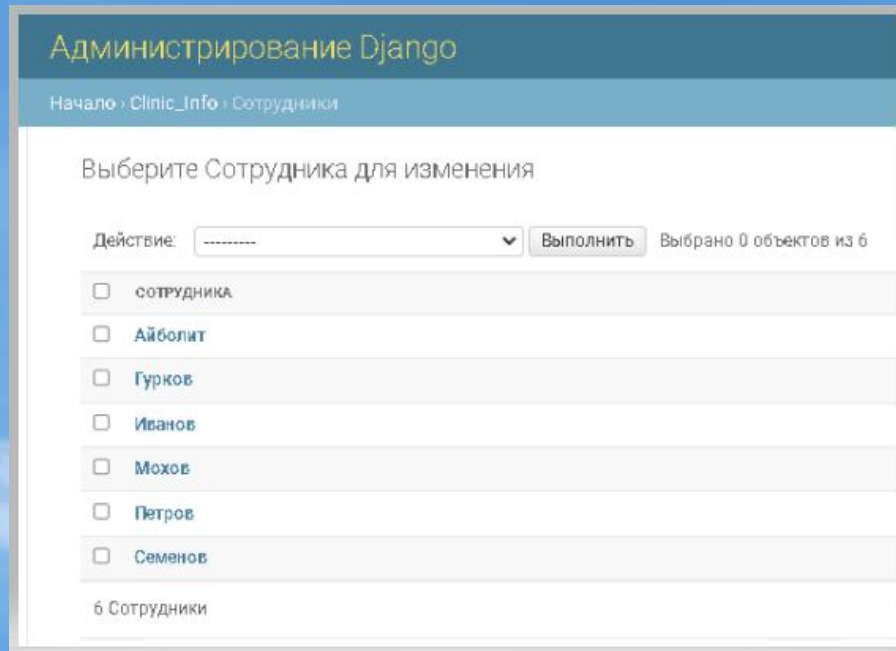
# Административный веб-сайт Django

Если имя и пароль введены без ошибок будет выведена страница со списком приложений, в котором будут присутствовать и наши зарегистрированные приложения. Каждое название модели в этом списке представляет собой гиперссылку, щелкнув на которой мы попадаем на страницу списка записей этой модели.



# Административный веб-сайт Django

Например, щелкнув на названии модели Сотрудники, мы получим страницу со списком записей, хранящихся в данной модели.



# Административный веб-сайт Django





Щелкнув на нужной записи, мы получим страницу ее правки, на которой можно изменить запись или удалить.

## Администрирование Django

[Начало](#) > [Clinic\\_Info](#) > [Сотрудники](#) > Айболит

### Изменить Сотрудника

**Айболит**

Фамилия:	<input type="text" value="Айболит"/>
Имя:	<input type="text" value="Иван"/>
Отчество:	<input type="text" value="Иванович"/>
Профессия:	<div>Ветеринарный врач</div> <div>   </div>
Квалификация:	<input type="text" value="Первая категория"/>
Опыт:	<input type="text" value="40 лет"/>
Фотография:	<div>На данный момент: <a href="#">imgaes/imgaes_1.png</a></div> <div>Изменить: <input type="button" value="Выберите файл"/> Файл не выбран</div>

# API


Многие современные веб-сайты предоставляют программные интерфейсы, предназначенные для использования сторонними программами: настольными, мобильными приложениями и другими веб-сайтами. С помощью таких интерфейсов, называемых веб-службами, сторонние программы могут получать информацию или, наоборот, заносить ее на сайт.

В Django для этих целей удобно применять библиотеку Django REST framework. Она самостоятельно извлечет данные из базы, закодирует в JSON, отправит фронтенду, получит данные от фронтенда, проведет их валидацию, занесет в базу и даже реализует разграничение доступа.

# API

Чтобы работать с библиотекой Django REST framework ее необходимо установить (pip install djangorestframework) и добавить в список зарегистрированных в проекте (INSTALLED\_APPS модуля settings.py из пакета конфигурации).

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'Clinic_info.apps.ClinicInfoConfig',  
    'drf_yasg',  
    'rest_framework',  
    'django_filters',  
    'reviews.apps.ReviewsConfig',  
    'registration.apps.RegistrationConfig',  
]
```



# API

В приложении Clinic используется маршрутизатор DefaultRouter, он дополнительно включает корневое представление API по умолчанию, которое возвращает ответ, содержащий гиперссылки на все представления списка.

У метода register() есть два обязательных аргумента: 1) prefix - Префикс URL, который будет использоваться для этого набора маршрутов. 2) viewset - Класс набора представлений.


```
urls.py
1 from django.urls import path, include
2 from rest_framework import routers
3
4 from .yasg import urlpatterns as swag_urls
5
6 from .views import *
7 from .api_views import *
8 from reviews.api_views import ReviewsAPIView
9
10 router = routers.DefaultRouter()
11
12 router.register(r'employee', EmployeeAPIView)
13 router.register(r'service', ServiceAPIView)
14 router.register(r'professions', ProfessionsAPIView)
15 router.register(r'client', ClientAPIView)
16 router.register(r'reviews', ReviewsAPIView)
17
18
```



# API

Атрибут `.urls` экземпляра маршрутизатора - это стандартный список шаблонов URL. Существует несколько различных стилей для включения этих URL. В нашем случае применяется функция Django `include`.

```
urlpatterns = [  
    path('clinic/', clinic, name='clinic'),  
    path('services/', services, name='services'),  
    path('detailed_services/<int:pk>/', detailed_services, name='detailed_services'),  
    path('prices/', prices, name='prices'),  
    path('contacts/', all_contacts, name='contacts'),  
    path('our_specialists/', our_specialists, name='our_specialists'),  
    path('detailed_by_specialist/<int:pk>/', detailed_by_specialist, name='detailed_by_specialist'),  
    path('api/', include(router.urls)),  
    path('api-auth/', include('rest_framework.urls'))  
]
```



# API

В файле `api_views.py` прописаны классы представления. Данные классы наследуются от класса `ModelViewSet` из модуля `rest_framework.viewsets`, с применением следующих атрибутов: `queryset` - набор записей, с которым будет выполняться работа и `serializer_class` - сериализатор, управляющий отправкой данных фронтенду.

Для разграничения доступа используется класс `IsAuthenticatedOrReadOnly`, который представляет полный доступ к данным только зарегистрированным пользователям, гости получают доступ лишь на чтение.

В атрибуте `filter_backends` прописаны классы, которые предоставляют интерфейс для фильтрации, поиска и сортировки данных.

Далее прописаны атрибуты, которые ссылаются на списки, здесь прописаны поля по которым будет происходить фильтрация, поиск и сортировка данных соответственно.

# API

Класс представления EmployeeAPIView:

```
api_views.py x
1 from django_filters.rest_framework import DjangoFilterBackend
2 from rest_framework import viewsets
3 from rest_framework.filters import SearchFilter, OrderingFilter
4 from rest_framework.permissions import IsAuthenticatedOrReadOnly
5 from .models import Employee, Service, Professions, Client
6 from .serializers import EmployeeSerializer, ServiceSerializer, \
7     ProfessionsSerializer, ClientSerializer
8
9
10 class EmployeeAPIView(viewsets.ModelViewSet):
11     queryset = Employee.objects.all()
12     serializer_class = EmployeeSerializer
13     permission_classes = (IsAuthenticatedOrReadOnly,)
14     filter_backends = (DjangoFilterBackend, SearchFilter, OrderingFilter,)
15     filter_set_fields = ['surname', 'profession']
16     search_fields = ['surname']
17     ordering_fields = ['surname']
```

# API

Класс представления ServiceAPIView:

```
class ServiceAPIView(viewsets.ModelViewSet):  
    queryset = Service.objects.all()  
    serializer_class = ServiceSerializer  
    permission_classes = (IsAuthenticatedOrReadOnly,)  
    filter_backends = (DjangoFilterBackend, SearchFilter, OrderingFilter,)  
    filterset_fields = ['service', 'price']  
    search_fields = ['service']  
    ordering_fields = ['service', 'price']
```

# API

Класс представления ProfessionsAPIView:

```
class ProfessionsAPIView(viewsets.ModelViewSet):  
    queryset = Professions.objects.all()  
    serializer_class = ProfessionsSerializer  
    permission_classes = (IsAuthenticatedOrReadOnly,)  
    filter_backends = (DjangoFilterBackend, SearchFilter, OrderingFilter,)  
    filterset_fields = ['profession']  
    search_fields = ['profession']  
    ordering_fields = ['profession']
```

# API

Класс представления ClientAPIView:

```
class ClientAPIView(viewsets.ModelViewSet):  
    queryset = Client.objects.all()  
    serializer_class = ClientSerializer  
    permission_classes = (IsAuthenticatedOrReadOnly,)  
    filter_backends = (DjangoFilterBackend, SearchFilter, OrderingFilter,)  
    filterset_fields = ['name', 'telephone']  
    search_fields = ['telephone']  
    ordering_fields = ['name']
```

# API

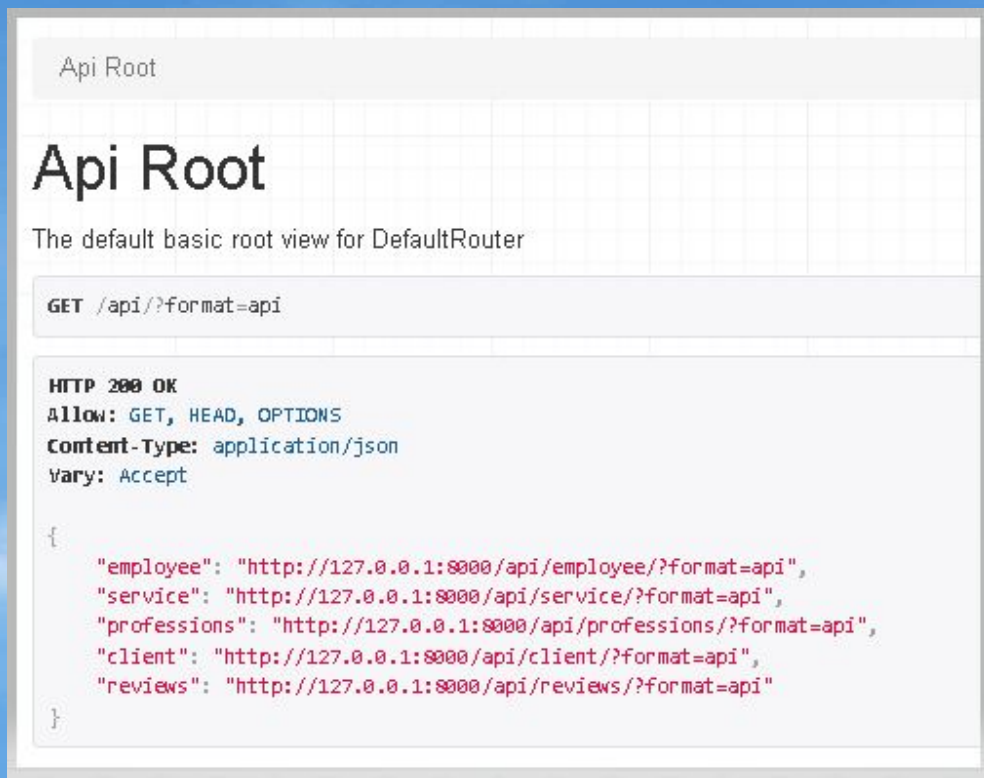
Класс представления  
ReviewsAPIView:

```
api_views.py x
1  from django_filters.rest_framework import DjangoFilterBackend
2  from rest_framework import viewsets
3  from rest_framework.filters import SearchFilter, OrderingFilter
4  from rest_framework.permissions import IsAuthenticatedOrReadOnly
5
6  from .models import Reviews
7  from .serializers import ReviewsSerializer
8
9
10 class ReviewsAPIView(viewsets.ModelViewSet):
11     queryset = Reviews.objects.all()
12     serializer_class = ReviewsSerializer
13     permission_classes = (IsAuthenticatedOrReadOnly,)
14     filter_backends = (DjangoFilterBackend, SearchFilter, OrderingFilter,)
15     filterset_fields = ['name']
16     search_fields = ['name']
17     ordering_fields = ['name', 'created_on']
18
```



# API

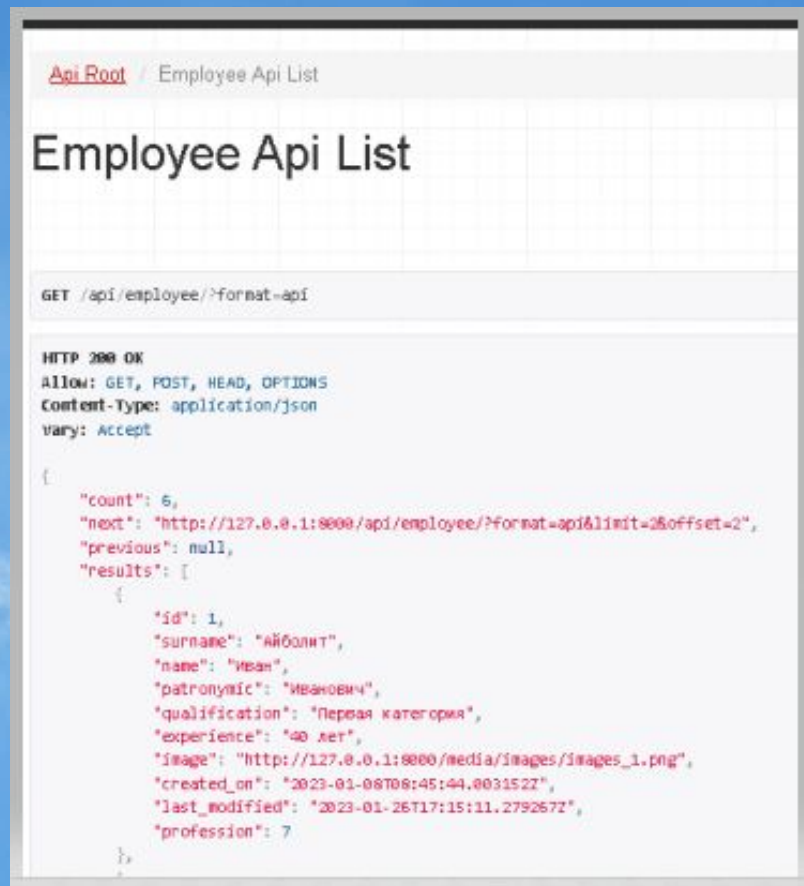
При переходе по интернет-адресу <http://127.0.0.1:8000/api/>, мы увидим результат работы маршрутизатора DefaultRouter, содержащий гиперссылки на все представления.



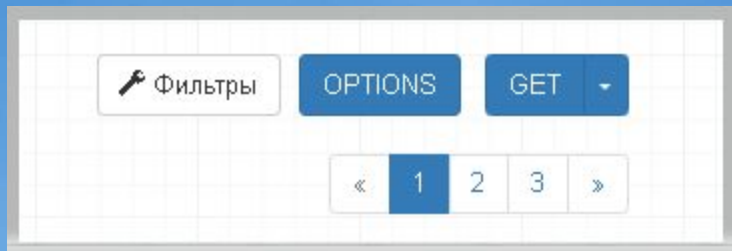


# API

Щелкнув по гиперссылке: "employee":  
"<http://127.0.0.1:8000/api/employee/>", мы получим  
все данные из таблицы Employee.



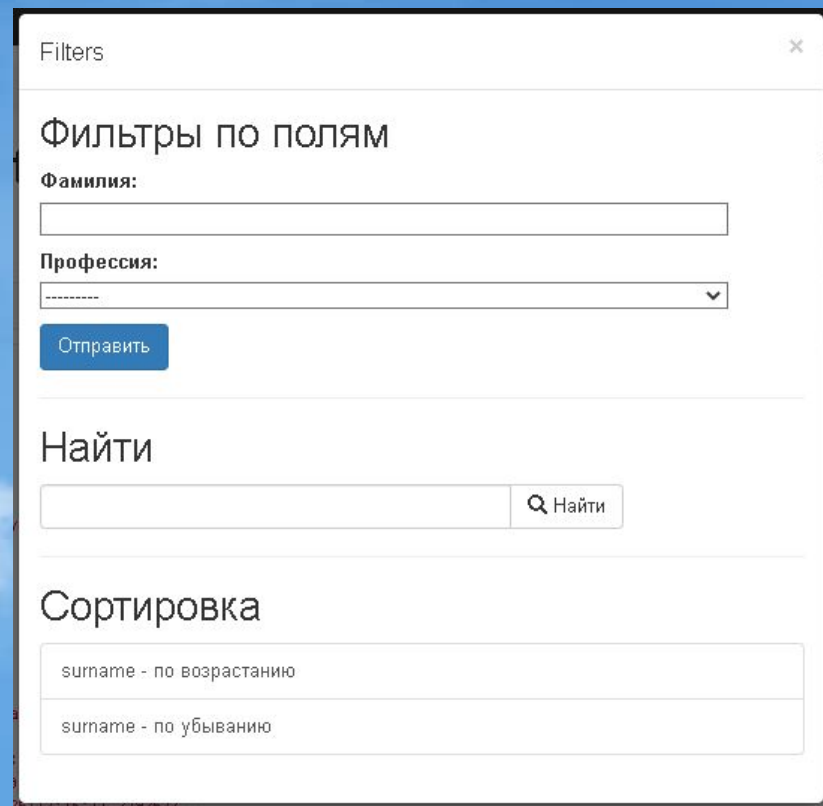
# API



Так как в наших классах представления прописаны атрибуты `filter_backends`, `filterset_fields`, `search_fields` и `ordering_fields` мы имеем доступ к фильтрам (кнопка Фильтры).

# API

В окне Filters предоставляется удобный интерфейс для фильтрации, поиска и сортировки данных.



The screenshot shows a 'Filters' window with a title bar containing the word 'Filters' and a close button. The main content area is titled 'Фильтры по полям' (Filters by fields). It contains three sections: 1. 'Фамилия:' (Surname) with a text input field. 2. 'Профессия:' (Profession) with a dropdown menu showing '-----' and a downward arrow. 3. A blue button labeled 'Отправить' (Send). Below these is a section titled 'Найти' (Find) with a search input field and a button with a magnifying glass icon and the text 'Найти'. At the bottom is a section titled 'Сортировка' (Sorting) with a list of two options: 'surname - по возрастанию' (surname - by ascending) and 'surname - по убыванию' (surname - by descending).

Filters

Фильтры по полям

Фамилия:

Профессия:

Отправить

Найти

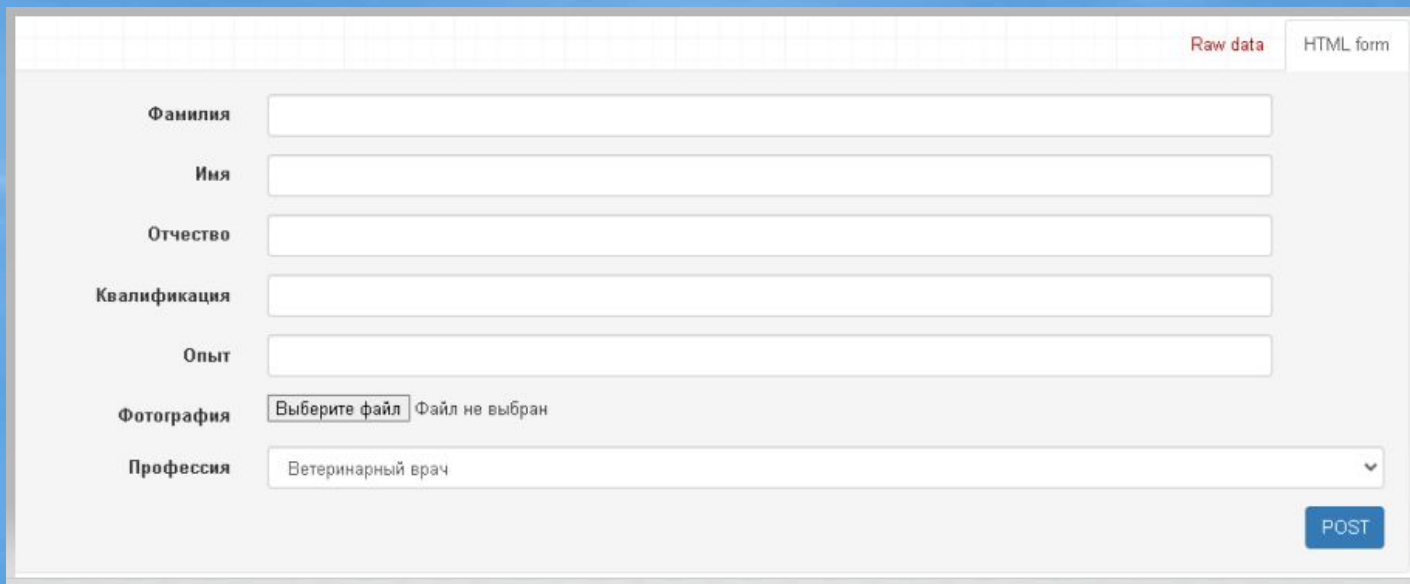
Сортировка

surname - по возрастанию

surname - по убыванию

# API

Также в авторизованных пользователей есть возможность не только просматривать данные, но и добавлять их в базу данных.



The screenshot shows a web interface for adding or editing user data. At the top, there is a grid of empty data rows. To the right of the grid are two tabs: 'Raw data' (highlighted in red) and 'HTML form'. The 'HTML form' tab is active, displaying a form with the following fields:

- Фамилия**: A text input field.
- Имя**: A text input field.
- Отчество**: A text input field.
- Квалификация**: A text input field.
- Опыт**: A text input field.
- Фотография**: A file selection area with a button labeled 'Выберите файл' and the text 'Файл не выбран'.
- Профессия**: A dropdown menu with 'Ветеринарный врач' selected.

A blue 'POST' button is located at the bottom right of the form.

# API

Перейдя по интернет-адресу <http://127.0.0.1:8000/swagger/>, мы получим доступ к набору инструментов, который позволяет автоматически описывать API на основе его кода. Документация, автоматически созданная через Swagger, облегчает понимание API для компьютеров и людей.

На основе кода или набора правил Swagger автоматически генерирует документацию в формате JSON-файла. Ее можно встроить на страницу сайта или в приложение, чтобы пользователи могли интерактивно знакомиться с документацией, можно отправлять клиентам — сгенерировать такое описание намного быстрее, чем написать с нуля.

# API

Перейдя по интернет-адресу <http://127.0.0.1:8000/redoc/> мы получаем автоматически сгенерированную документацию с помощью инструмента ReDoc.

ReDoc - это инструмент OpenAPI, созданный для справочной документации API. Он обеспечивает простое развертывание и может объединять документы в независимые HTML-файлы и др.

## Клиника v1 )

Скачать спецификацию OpenAPI :

Электронная почта : [markevich3214@tut.by](mailto:markevich3214@tut.by) | Лицензия: Лицензия BSD

Пример описания теста