# Общие сведения.

## Что такое Django

**Django** — это высокоуровневый Python веб-фреймворк, который позволяет быстро создавать безопасные и поддерживаемые веб-сайты. Созданный опытными разработчиками, Django берёт на себя большую часть хлопот веб-разработки, поэтому вы можете сосредоточиться на написании своего веб-приложения без необходимости изобретать велосипед.

## Установка и настройка.

Чтобы установить Джанго в проект, после создания проекта нужно ввести в терминал команду:

$ pip install Django

Далее ввести в терминал:

$ django-admin startproject

Таким образом внутри проекта создастся джанго-проект.

## Основные команды Django

Самые важные или частые команды, которыми я пользовалась при разработке проекта.

Установка Джанго

$ pip install Django

Создание Джанго-проекта

$ django-admin startproject

Создание миграций модели

$ python manage.py makemigrations learning\_logs

Проведение миграций

$ python manage.py migrate

Создание учтённой записи администратора

$ python manage.py createsuperuser

# Руководство программиста

## Модели

А **Модель Django** — это встроенная функция, которую Django использует для создания таблиц, их полей и различных ограничений. Проще говоря, модели Django - это база данных SQL, которую используют с Django. Модели Django упрощают задачи и организуют таблицы в модели. Как правило, каждая модель сопоставляется с отдельной таблицей базы данных.

В моем проекте я создала модель Positions – база данных позиции меню.

class Position(models.Model):  
 name = models.CharField('Название', max\_length=30)  
 type = models.CharField('Тип', max\_length=30)  
 gramm = models.CharField('Граммовка порции', max\_length=10)  
 sostav = models.TextField('Описание')  
 price = models.IntegerField('Цена')  
  
 def \_\_str\_\_(self):  
 return self.name  
 class Meta:  
 verbose\_name="Позиция"  
 verbose\_name\_plural="Позиции меню"

## Представления

В Django представления (views) — это код, который управляет тем, что отображается в браузере после запроса пользователя на сервер. Views преобразуют данные из моделей и контекста запроса в HTML-страницу, которую браузер может отображать.

Для создания представления в Django, необходимо определить функцию, которая будет получать запрос и возвращать объект ответа HTTP или вызвать метод, которые Django предоставляет для создания ответа.

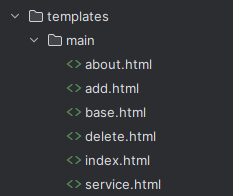
Код файла views.py в моем проекте.

def index(request):  
 positions\_all = Position.objects.all()  
 positions = PostFilter(request.GET, queryset=positions\_all)  
 return render(request, 'main/index.html', {'title': 'Главная страница сайта', 'positions': positions})  
  
  
def about(request):  
 return render(request, 'main/about.html')  
  
  
def service(request):  
 return render(request, 'main/service.html')  
  
  
def add(request):  
 error = ''  
 if request.method == 'POST':  
 form = PosForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 return redirect('home')  
 else:  
 error = 'Форма заполнена неправильно или не до конца'  
 form = PosForm()  
 context = {  
 'form': form  
 }  
 return render(request, 'main/add.html', context)  
  
  
class Delete(DeleteView):  
 model = Position  
 success\_url = '/'  
 template\_name = 'main/delete.html'  
  
  
class Update(UpdateView):  
 model = Position  
 template\_name = 'main/add.html'  
 form\_class = PosForm  
 success\_url = '/'

## Шаблоны

**Шаблоны** **Django** — **это** файлы, которые определяют, как должен выглядеть HTML-код страниц веб-приложения. Они используют специальный язык **шаблонов**, который позволяет вставлять данные из контекста приложения, использовать циклы, условные операторы и другие конструкции.

Шаблоны хранятся в папке templates находящейся в директории по адресу(в моем случае) main/templates/main.



## Работа с формами

В HTML форма представляет собой набор элементов внутри <form>...</form>, которые позволяют посетителю выполнять такие действия, как ввод текста, выбор параметров, манипулирование объектами или элементами управления и так далее, а затем отправлять эту информацию обратно на сервер.

GET и POST - единственные HTTP-методы, которые можно использовать при работе с формами.

Форма входа в систему Django возвращается с использованием POST метода, в котором браузер объединяет данные формы, кодирует их для передачи, отправляет на сервер, а затем получает обратно свой ответ.

Код в файле forms.py в моем проекте:

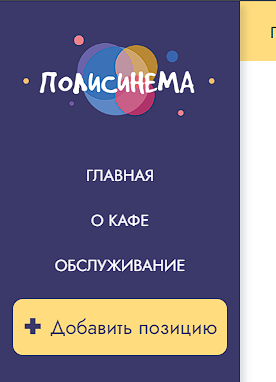
class PosForm(ModelForm):  
 class Meta:  
 model = Position  
 fields = ["name", "type", "gramm", "sostav", "price"]  
 widgets = {  
 "name": TextInput(attrs={'placeholder': 'Название','class': 'form-input'}),  
 "type": TextInput(attrs={'placeholder': 'Тип','class': 'form-input'}),  
 "gramm": TextInput(attrs={'placeholder': 'Граммы','class': 'form-input'}),  
 "sostav":Textarea(attrs={'placeholder': 'Описание','class': 'form-input'}),  
 "price": TextInput(attrs={'placeholder': 'Цена в руб.','class': 'form-input'}),  
 }

Этот код создает поля фомы, а также задает атрибуты полей.

# Руководство пользователя

## **Навигация**.

Главное меню с левой стороны сайта отвечает за навигацию по сайту.



«Главная» - начальная и главная страница.

«О кафе» - переносит на страницу с информацией о кафе.

«Осблуживание» - переносит на страницу с информацией об системе обслуживаня.

## Добавление позиции в базу данных.

|  |  |
| --- | --- |
| В левом меню выберите пункт «Добавить позицию» |  |
| Заполните поля формы. |  |
| Нажмите «Добавить» |  |
| Вас переадресует на Главную страницу, где будет отображена новая позиция. |  |

## Удаление или изменение позиции меню.

|  |  |
| --- | --- |
| Перейдите на Главную страницу |  |
| В строке с позицией нажмите кнопку «Изменить» |  |
| Измените нужные поля. |  |
| Нажмите кнопку «Добавить» |  |

|  |  |
| --- | --- |
| Перейдите на Главную страницу |  |
| В строке с позицией нажмите кнопку «Удалить» |  |
| Подтвердите удаление |  |