**Конспект**

|  |  |
| --- | --- |
| Автор: | Горюнов Евгений С. |
| Город | Томск |
| mail: | ges1987@list.ru |
| Курс: | Fullstack разработчик на Python (SkillFactory |
| Год обучения: | 2021.08 – 2022г |
|  |  |

Оглавление

[Команды для консоли PyCharm 2](#_Toc92806938)

[Создать виртуальное окружение (имя окруж – venv) 2](#_Toc92806939)

[Активация виртуально окружения 2](#_Toc92806940)

[Просмотр установленных приложений в вирт окруж 2](#_Toc92806941)

[Установка фрейворка Django 2](#_Toc92806942)

[Создать проект Django (имя – project) 2](#_Toc92806943)

[Переход в папку project 2](#_Toc92806944)

[Переход на каталог выше 2](#_Toc92806945)

[Запуск сервера PyCharm 2](#_Toc92806946)

[Создание миграций 3](#_Toc92806947)

[Провести миграции 3](#_Toc92806948)

[Создание администратора (первого пользователя) 3](#_Toc92806949)

[Библиотеки для Django SkillFactory 3](#_Toc92806950)

[Сигналы 5](#_Toc92806951)

# Команды для консоли PyCharm

Все команды для windows среды

## Создать виртуальное окружение (имя окруж – venv)

python -m venv venv

## Активация виртуально окружения

venv\scripts\activate

## Просмотр установленных приложений в вирт окруж

pip list

## Установка фрейворка Django

pip install django

## Создать проект Django (имя – project)

python -m django-admin startproject project

## Переход в папку project

cd project

переход по конкретному пути

cd project\news

## Переход на каталог выше

cd ..

## Запуск сервера PyCharm

py manage.py runserver

http://127.0.0.1:8000/ - порт запуска сервера

для запуска сервера на другом порту, для работы нескольких проектов одновременно

py manage.py runserver 7000

## Создание миграций

python manage.py makemigrations

## Провести миграции

py manage.py migrate

## Создание администратора (первого пользователя)

python manage.py createsuperuser

## Библиотеки для Django SkillFactory

django-filter

django-allauth

def post(self, request, \*args, \*\*kwargs):

post -

\*args -

\*\*kwargs –

request-

self-

def post-

# Сигналы

## notify\_managers

читать тут более подробно

<https://russianblogs.com/article/4399514519/>

Данный метод нужен для того, чтоб начать выполнять код после какого-либо события, например сохранение в БД информации (в нашем примере), ее изменение и так далее. Сигналов множество, на любой случай

Импортируем модули

def notify\_managers\_ appointment (sender, instance, created, \*\*kwargs):

subject = f'{instance.client\_name} {instance.date.strftime("%d %m %Y")}'

print("МЕТОД НУЖНЫЙ СРАБОТАЛ!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")

mail\_managers(

subject=subject,

message=instance.message,

)

# коннектим наш сигнал к функции обработчику и указываем, к какой именно модели после сохранения привязать функцию

post\_save.connect(notify\_managers\_appointment, sender=Appointment)