Практическое занятие №1

План курса:

1. Практические занятия:
   1. Блок практических занятий - разработка сайта на джанго.
   2. Блок практических занятий Докер.
2. Лабораторные работы:
   1. Модель данных на Django ORM.
   2. Адресация и контроллеры.
   3. Формы и шаблоны.

# Django web framework

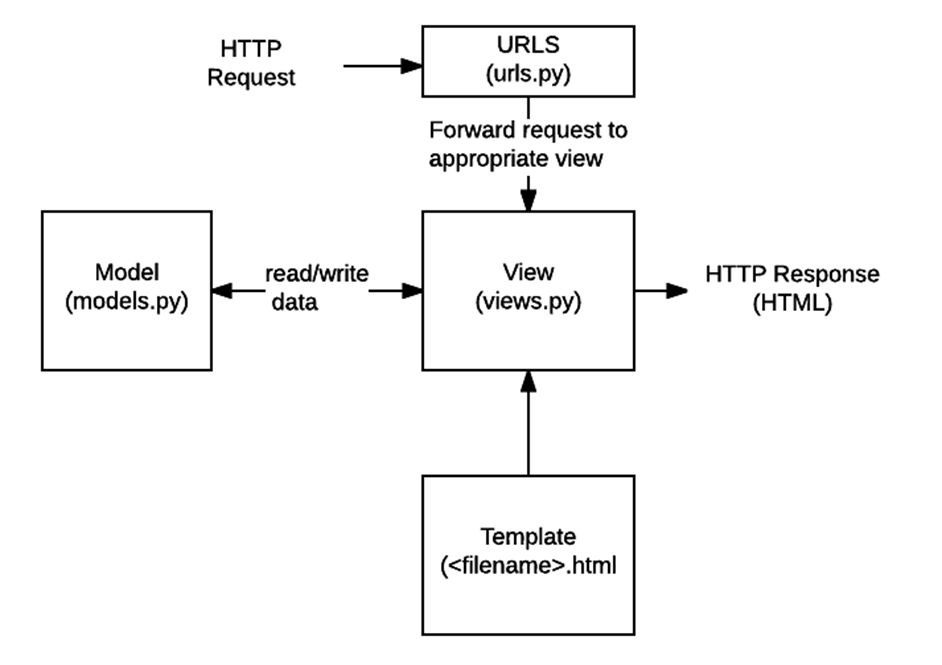
## Задача 1: установить Django Web Framework.

<https://drive.google.com/open?id=1fsNaCm2MCxVletRLdGzVrd6-lPePn2Xh> - мануал по установке Django Web Framework. Установить нужно любым доступным способом (через командную строку или PyCharm Professional Edition)

Джанго проект назвать “django\_project\_фамилия”.

Джанго приложение назвать “project\_first\_app”.

*Файловая архитектура организации джанго (стандартная):*



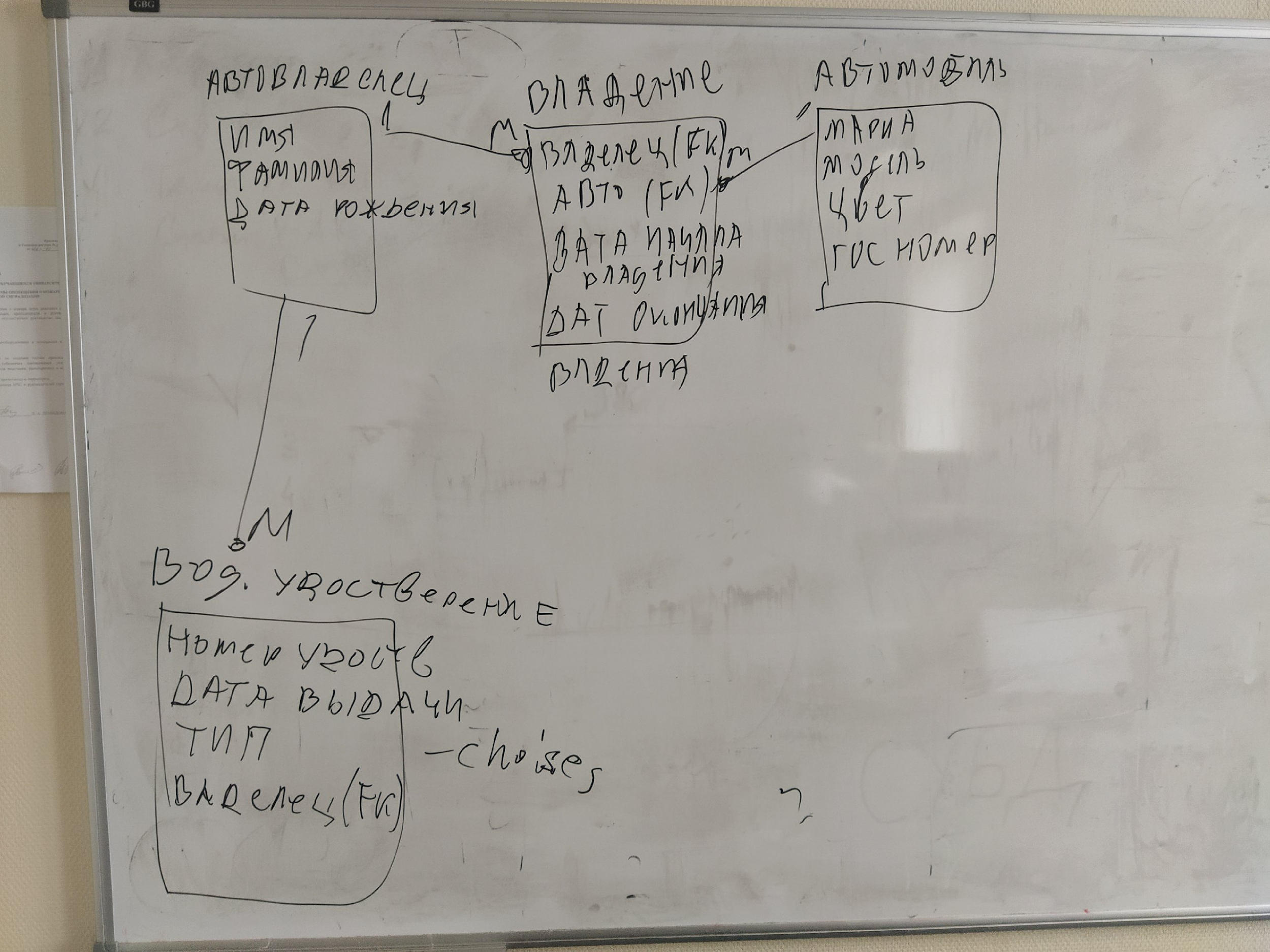
* **URLs**: Хотя можно обрабатывать запросы с каждого URL-адреса с помощью одной функции, гораздо удобнее писать отдельную функцию для обработки каждого ресурса. URL-mapper используется для перенаправления HTTP-запросов в соответствующее представление на основе URL-адреса запроса. URL-mapper также может извлекать данные из URL-адреса в соответствии с заданным шаблоном и передавать их в соответствующую функцию в виде аргументов.
* **View:** Представление (view) - это функция обработчика запросов, которая получает HTTP-запросы и возвращает ответы. View имеет доступ к данным через модели (необходимым для удовлетворения запросов и делегирования ответа в шаблоны).
* **Models:** Модели представляют собой объекты Python, которые определяют структуру данных приложения и предоставляют механизмы для управления (добавления, изменения, удаления) и выполнения запросов в базу данных.
* **Templates:** Template (шаблон) - это текстовый файл определяющий структуру или разметку страницы (например HTML страницы), с полями для подстановки используемыми для представления актуального содержимого. View может динамически создавать HTML страницы, используя HTML шаблоны и заполняя их данными из модели (model). Шаблон может быть использован для определения структуры файлов любых типов, не обязательно HTML.

## Создание модели Django

Оффициальная документация <https://docs.djangoproject.com/en/3.0/topics/db/models/>

Мануал по созданию модели данных Django ORM от Марины Михайловны <https://drive.google.com/file/d/1lmH_EM_-bB_YHxolgk10HZ8x50G8twMb/view?usp=sharing>

Необходимо создать модель с соответствии с моделью данных.



Не забывайте о внешних ключах, полях выбора, правильном оформлении ассоциативной сущности в модели.

Типы данных в Django ORM <https://metanit.com/python/django/5.2.php>

* 1. **Применение модели данных к базе данных в SQLite.**

Создать миграции и применить их к базе данных. Django использует миграции для переноса изменений в моделях (добавление поля, удаление модели и т.д.) на структуру базы данных.

|  |
| --- |
| python manage.py makemigrations #создает миграции на основе имеющейся модели данных. Созданные миграции доступны в файле миграций, их можно изменить вручную. Просмотрите их содержание в папке migrations вашего приложения. python manage.py migrate #применяет миграции к базе данных. |

Скачать DBbrowser для sqlite и открыть созданную базу.

## Создание админ панели для разработанной модели данных

* 1. Необходимо заполнить таблицы данными через адмнику.
     1. Регистрируем владельца авто в админке. Админка позволяет работать с объектами базы данных (операции создания, удаления, редактирования). Заходим в файл admin.py в папке вашего приложения (\*\_app) и регистрируем владельца автомобиля следующими командами:

|  |
| --- |
| from .models import Название владельца в модели данных #то, что указано в кавычках - это какие либо параметры, которые Вам нужно ввести самостоятельно, в соответствии с Вашим проектом. В Конкретном случае, Вам нужно указать название таблицы модели данных, которую необходимо отразить в админке. admin.site.register(Название владельца в модели данных) |

Документация: <https://docs.djangoproject.com/en/3.0/ref/contrib/admin/>

*Важно!: не забываете, что структура Вашего проекта содержит настройки проекта (папка, в которой находится файл settings.py) и папки приложений (папки, в которых находятся файлы views.py, models.py и тд)*

* + 1. Регистрируем оставшиеся таблицы в админке.
  1. Создать суперпользователя комадной:

|  |
| --- |
| python manage.py createsuperuser |

П.С. Когда вводится пароль, но символов не видно - они просто скрыты

* 1. Запустить сервер командой:

|  |
| --- |
| python manage.py runserver |

* 1. Зайти в админку по адресу (127.0.0.1:8000/admin/) и добавить двух владельцев автомобилей, 4 автомобиля. Свяжите каждого владельца минимум с тремя автомобилями, так, чтобы не было пересечений по датам владения и продажи.

## Создание контроллеров для обработки данных

Контроллер является центральным компонентом в архитектуре MVC. Контроллер получает ввод пользователя, обрабатывает его и посылает обратно результат обработки, например, в виде представления.

* 1. Создать в файле views.py (находится в папке вашего приложения) контроллер, который выведет из базы данные о владельце автомобиля.

В качестве примера воспользоваться оффициальной документацией <https://docs.djangoproject.com/en/3.0/topics/http/views/#the-http404-exception> .

Разбор примера из официальной документации:

|  |
| --- |
| from django.http import Http404  #импортирует метод обработки ситуации, когда нет необходимых записей в бд (обработчик ошибок) from django.shortcuts import render #импортирует метод, который "запускает" созданную хтмл страницу и передает в нее указанные параметры from polls.models import Poll #импортирует таблицу Poll из модели данных models, где polls - название приложения (и папки) def detail(request, poll\_id):  try: #метод try-except - обработчик исключений  p = Poll.objects.get(pk=poll\_id) #pk - автоматически создается в джанго для любой таблицы в моделе (оно есть у любого объекта из бд), poll\_id будет передан функции при её вызове. #переменной p присваивается объект, полученный в результате выполнения запроса аналогичного "select \* from Poll where pk=poll\_id"  except Poll.DoesNotExist:  raise Http404("Poll does not exist") #исключение которое будет вызвано, если блок try вернет значение False (не будут найдены записи в таблице Poll)  return render(request, 'polls/detail.html', {'poll': p}) #данная строка рендерит хтмл страницу detail.html и передает в него объект "p", который в хтмл шаблоне будет называться "poll" |

* 1. Создать страницу html-шаблона owner.html в папке templates (создаем папку templates в корне проекта, если ее нигде нет, далее в контекстном меню папки создаем хтмл файл). Страница должна содержать отображение полей переданных из контроллера. Для обращения к этим полям используется конструкция вида (хтмл страница получит тот объект, который в пункте 3.1 имеет вид ):

|  |
| --- |
| {{название объекта переданного из контроллера.название поля из модели данных}} |

Пример:

|  |
| --- |
| <body> Имя: {{owner.first\_name}}, Фамилия: {{owner.last\_name}} <br> </body> |

## Работа с адресацией:

Мануал по работе с адресацией от Марины Михайловны <https://drive.google.com/file/d/1R9nD8b5mHPKz8LLEIW3R7QieZMhMsNMk/view?usp=sharing>

Официальная документация

<https://docs.djangoproject.com/en/3.0/topics/http/urls/>

* + 1. Создать файл urls.py в папке вашего приложения (\*\_app) (пока пустой).
    2. Импортировать файл юрлов Вашего приложения в проект (модифицируем файл urls.py в той папке, в которой хранится файл setting.py). Он должен иметь подобный код:

|  |
| --- |
| from django.contrib import admin from django.urls import path, include urlpatterns = [  path('admin/', admin.site.urls),  path('', include('example6\_app.urls')), #данная строчка импортирует в проект отдельный файл юрлов Вашего приложения (example6\_app - название Вашего приложения (название папки)), urls указывает на файл юрлов в папке приложения (указывает на тот пустой файл, который мы создали в пункте 9.а. ] |

* 1. Теперь необходимо описать в файле юрл адресов вашего приложения, созданном в пункте 4.1.1, такой юрл адрес, который сможет обратиться к контроллеру и вывести страницу, которая должна быть отрендерена контроллером.

Вам необходимо обратиться к контроллеру, который вы создали в пункте 7 и передать параметр (иденфикационный номер владельца) в адресной строке (инструкция: <https://docs.djangoproject.com/en/3.0/topics/http/urls/#example>).

|  |
| --- |
| from django.urls import path  from . import views #подключение файла контроллеров,описанного в пункте 3 urlpatterns = [  path('articles/2003/', views.special\_case\_2003), #пример вызова контроллера (функции) с именем "special\_case\_200" из файда views  path('articles/<int:year>/', views.year\_archive), #пример вызова контроллера (функции) с именем "year\_archive" из файда viewsи передачи в него переменной "year" ] |

Разбор примера из документации: в строке "path('articles/<int:year>/', views.year\_archive)," примера, по адресу "articles/1" вызывается контроллер year\_archive из папки views и ему передается параметру year значение "1" имеющие тип данных integer, .

**В итоге по адресу “127.0.0.1:8000/owner/1” мы должны получать страницу с информацией о первом владельце автомобилей.**