**61. Новый проект**

1. **Создаем проект todo\_Project**

C:\Users\alexandr.yakushevski\Desktop\Udemy\Python\Django>django-admin startproject todo\_Project

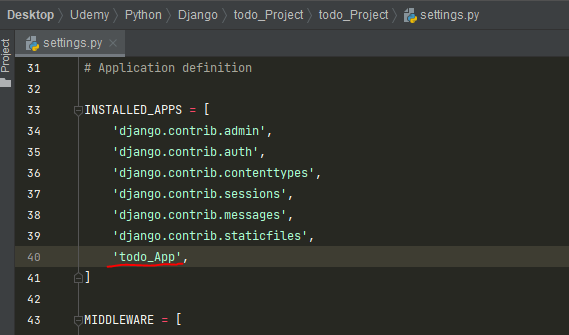
1. **Запускаем сервер**

C:\Users\alexandr.yakushevski\Desktop\Udemy\Python\Django\todo\_Project>python manage.py runserver

1. **Создаем приложение**

C:\Users\alexandr.yakushevski\Desktop\Udemy\Python\Django\todo\_Project>python manage.py startapp todo\_App

1. **В файле «todo\_Project/settings.py» в блок « INSTALLED\_APPS» добавляем наименование созданного приложения**

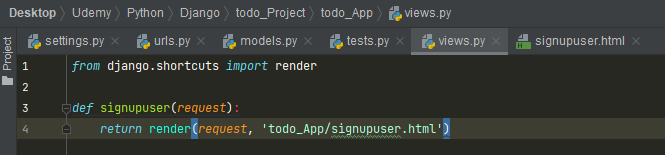


1. **Производим миграции**

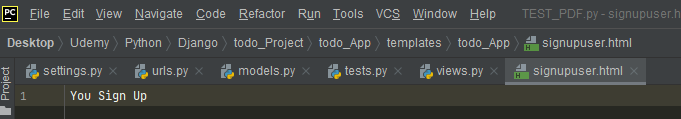
C:\Users\alexandr.yakushevski\Desktop\Udemy\Python\Django\todo\_Project>python manage.py migrate

**62. Форма регистрации**

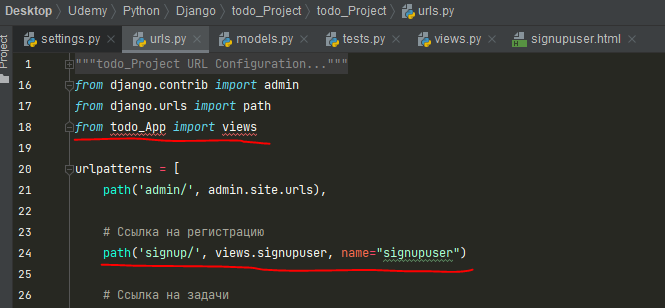
1. **В файле «todo\_App/views.py» создаем функцию для отображения страницы регистрации**



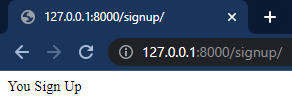
1. **Создаем страницу регистрации (создаем файл «todo\_App/templates/todo\_App/signupuser.html»)**



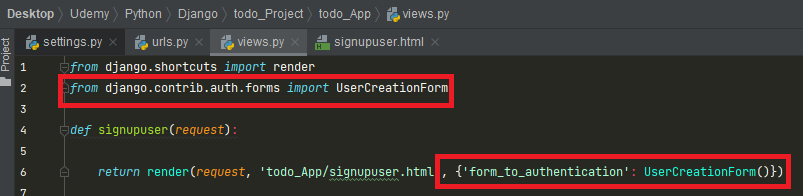
1. **Создаем ссылку на страницу сайта для регистрации в файле «todo\_Project/urls.py»**



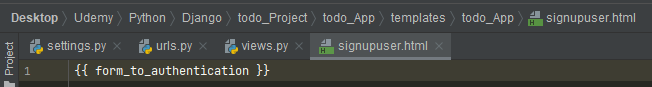
1. **При вводе адреса появляется созданная страница**



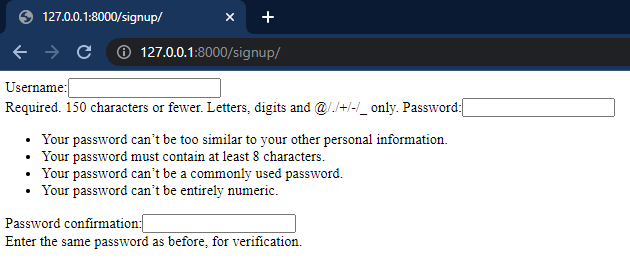
1. **Задаем форму для страницы регистрации в файле «todo\_App/views.py» (в Django есть шаблоны для создания страницы регистрации).**

****

1. **Указываем шаблон на страницы регистрации в файле «todo\_App/templates/todo\_App/signupuser.html»**



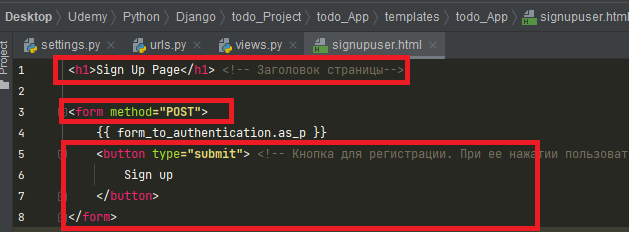
1. **Шаблон страницы регистрации Django**



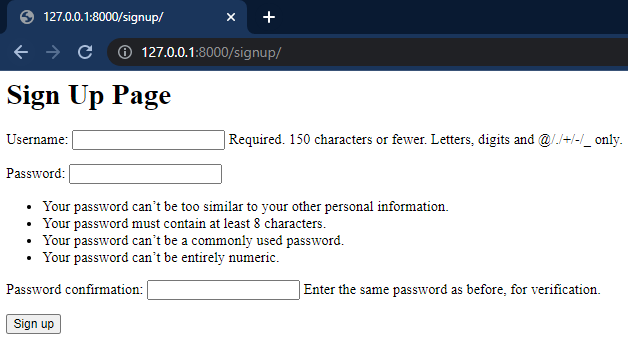
1. **Производим корректировки в отображении в файле «todo\_App/templates/todo\_App/signupuser.html»:**

- Добавляем заголовок страницы

- Добавляем кнопку, нажав которую пользовательские данные отправляются в БД

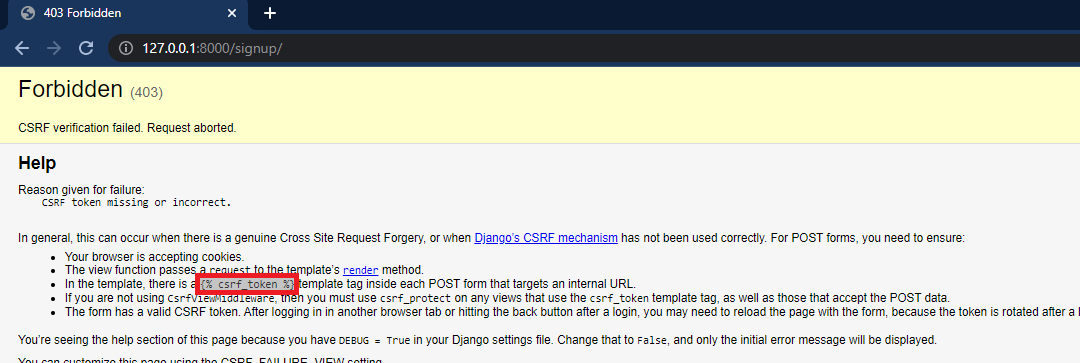


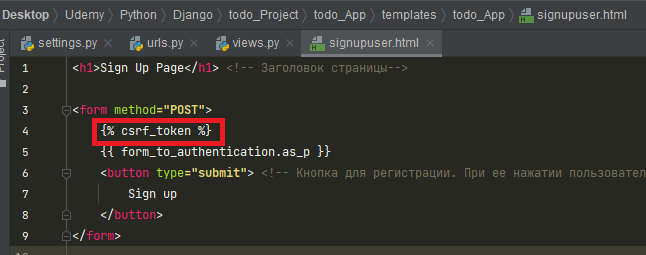
Стало:



1. **Обработка ошибки после ввода данных пользователем «CSRF token missing or incorrect»**

**Добавляем «{% csrf\_token %}» (в любое место файла т.к. это скрытой код)**

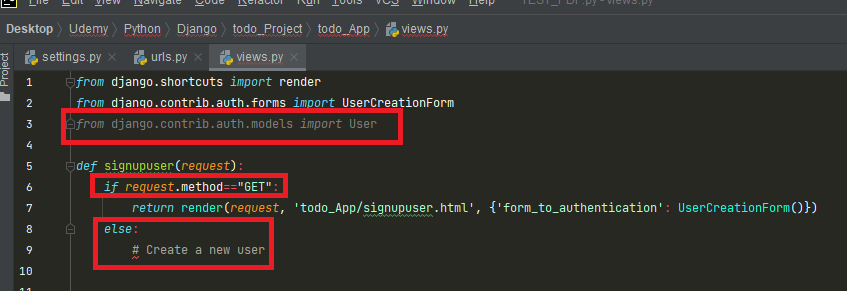
****

****

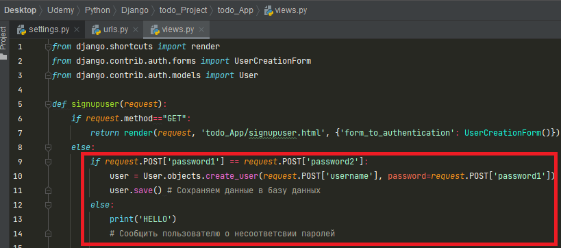
1. **method="POST" предлагает информацию на конкретный адрес, при этом оставляет информацию скрытой от основного адреса запроса, разделяя его на несколько частей. Результат запроса не отображается в ссылке**

**63. Создание пользователя**

1. **В файле «todo\_App/views.py» определяем какой из методов нужен пользователю method="POST" или method="GET" для определения пытается пользователь зайти в ЛК или только создать его.**
2. **Импортируем модель пользователя и определяем порядок действий при методах**

****

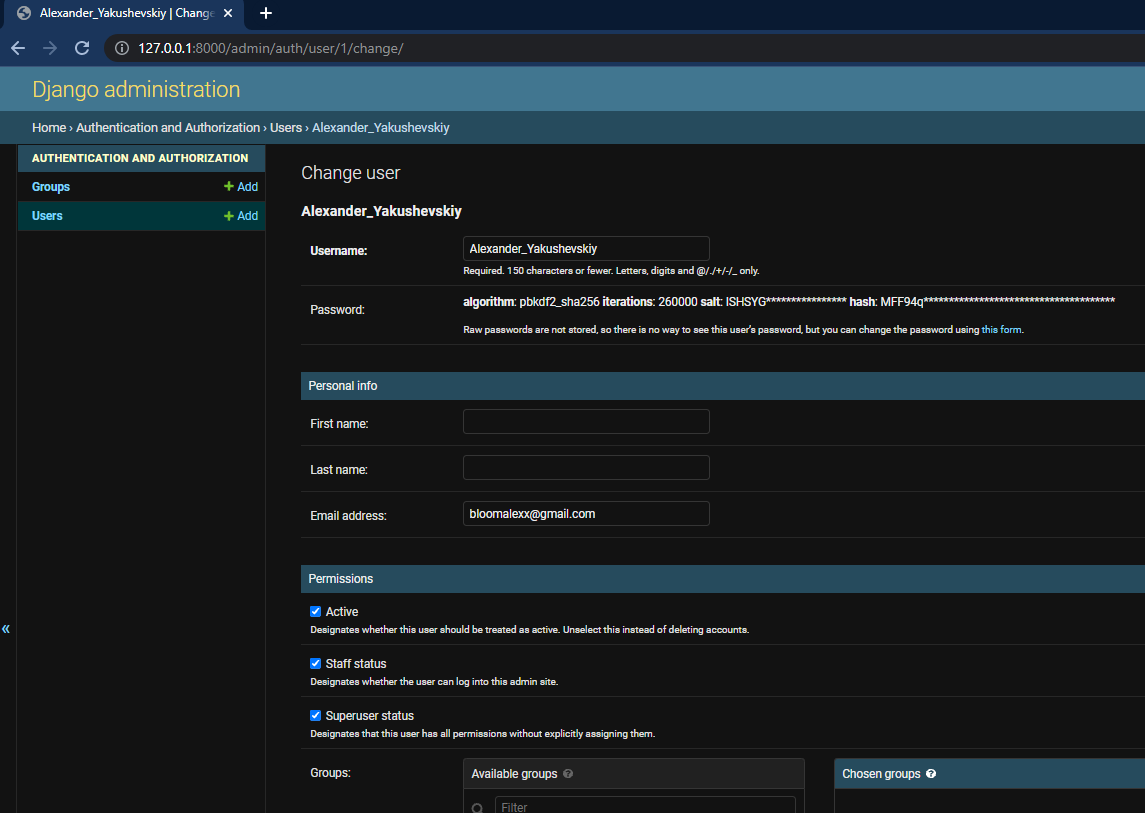
1. **Если пользователь новый, то:**

****

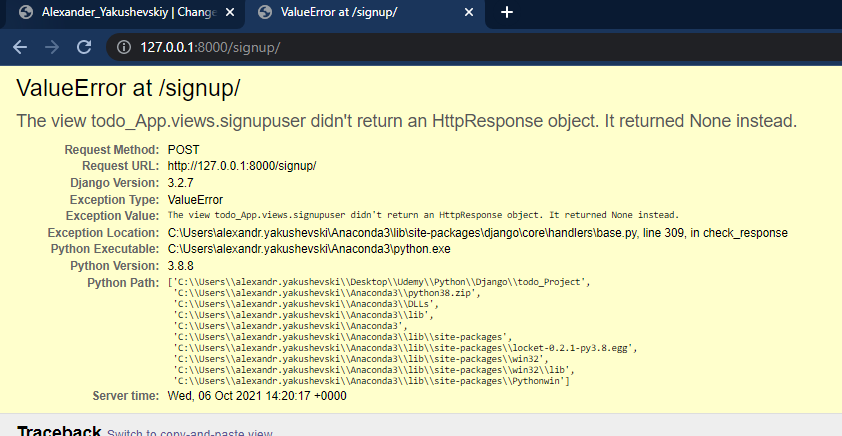
1. **Создаем SuperUser и переходим в панель администратора**

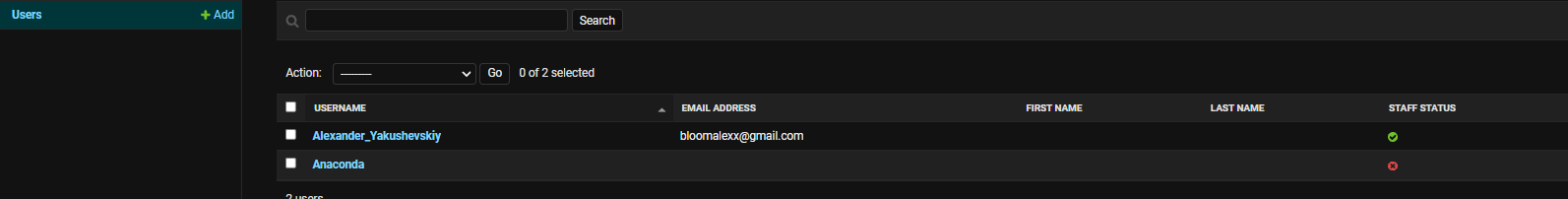
C:\Users\alexandr.yakushevski\Desktop\Udemy\Python\Django\todo\_Project>python manage.py createsuperuser

В панели администратора в группе «Users» появились данные о созданном SuperUser



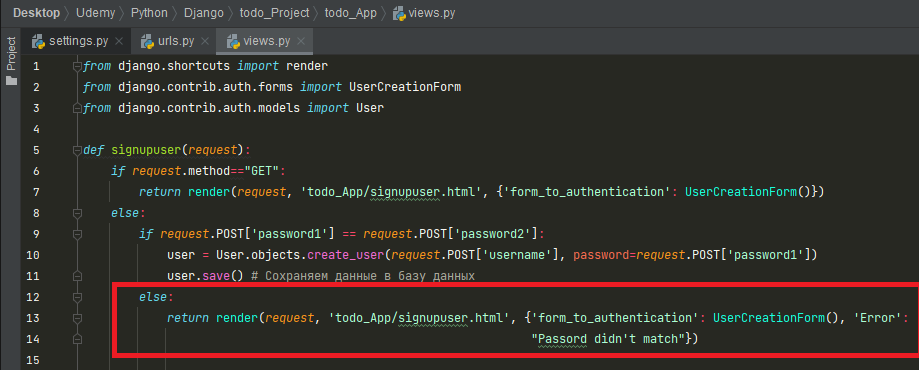
1. **При попытке создания пользователя на странице регистрации возникает ошибка «ValueError at /signup/» но в панели администратора незарегистрированный пользователь появился**



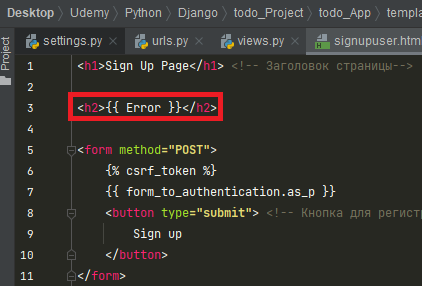


**64. Обработка ошибок**

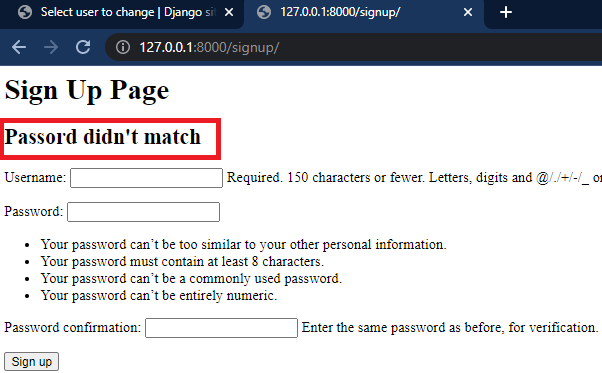
1. **Для пользовательского отображения об ошибке переходим к файлу «todo\_App/views.py» к блоку кода, где производится проверка соответствия паролей**

****

1. **Переходим в файл «todo\_App/templates/todo\_App/signupuser.html» для добавления заголовка об ошибке**

****

1. **При вводе пользователем разных паролей пользователь остается на странице регистрации, на которой отображается информация о разных паролях**

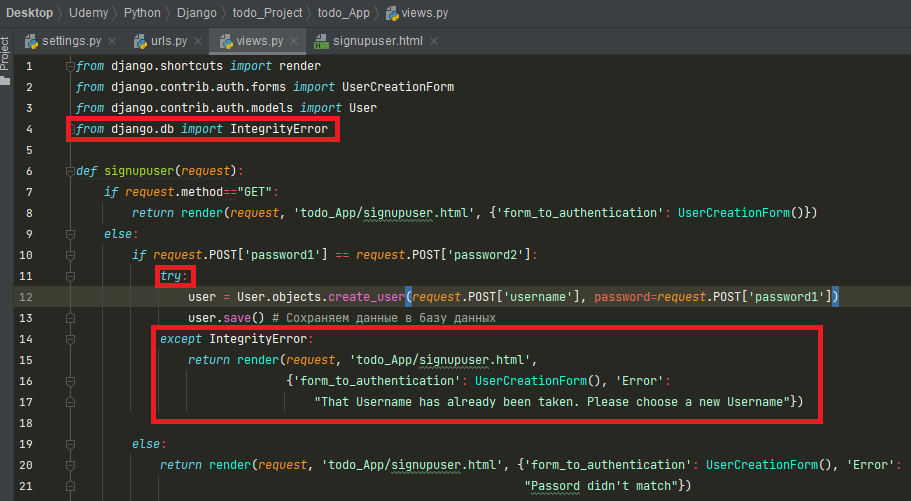
****

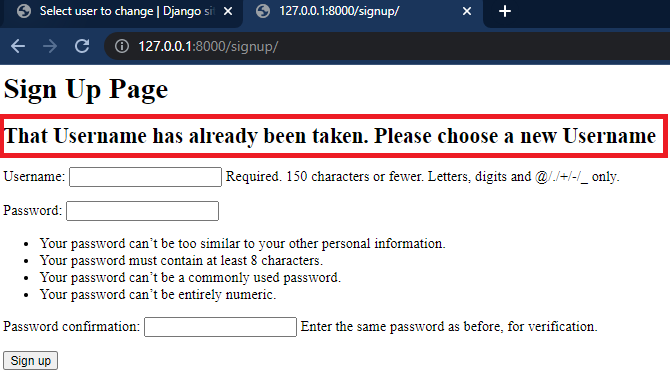
1. **В случае регистрации пользователя под уже имеющимся ником возникает ошибка «IntegrityError at /signup/». В теле этой ошибке есть параметр «Exception Type: IntegrityError»**

****

В теле этой ошибке есть параметр «Exception Type:IntegrityError».

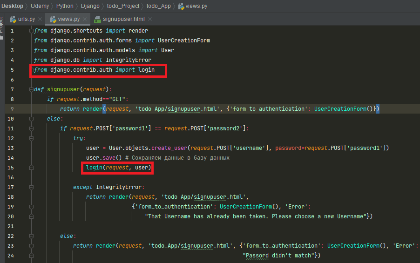
1. **Обработка этой ошибки производится в файле «todo\_App/views.py»**

****

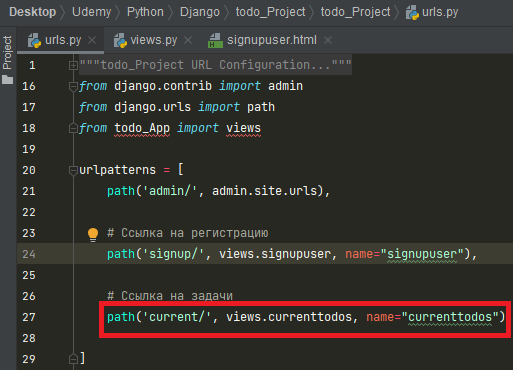
****

1. **После регистрации пользователь может проходить не в свой аккаунт, а в панель администратора. Для перевода пользователя в личный кабинет:**

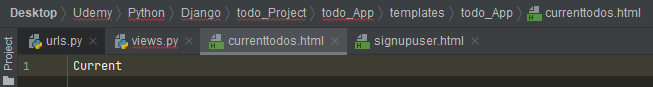
* импортируем дополнительный код в файле «todo\_App/views.py»

****

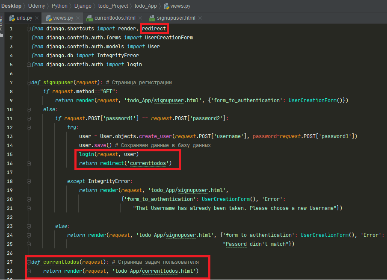
- создаем ссылку на страницу в файле «todo\_Project/urls.py»

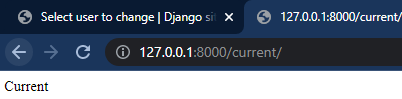
****

- создаем функцию страницу «currenttodos.html»

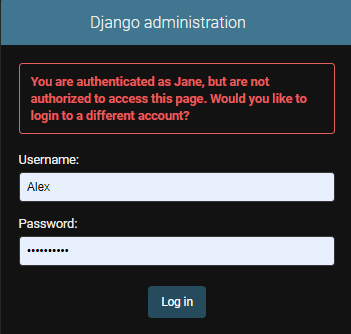


- создаем функцию «currenttodos» в файле «todo\_App/views.py» и импортируем модуль





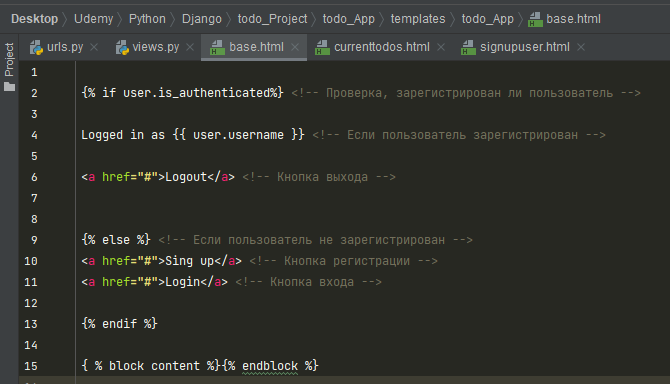
* Также доступ к панели администратора будет ограничен



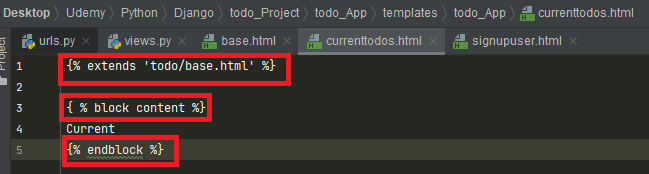
**65. Отображение статуса пользователя**

**Определяем зарегистрирован ли пользователь**

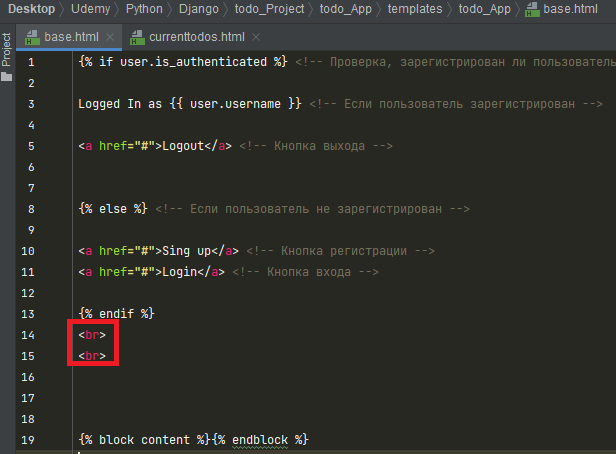
1. **Создаем новую страницу «todo\_App/templates/todo\_App/base.html» в котором проверяем зарегистрирован ли пользователь**

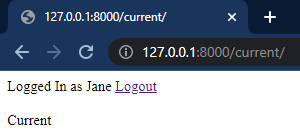


1. **Привязываем шаблон в файле «currenttodos.html»**

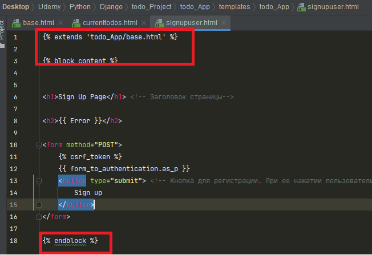
****

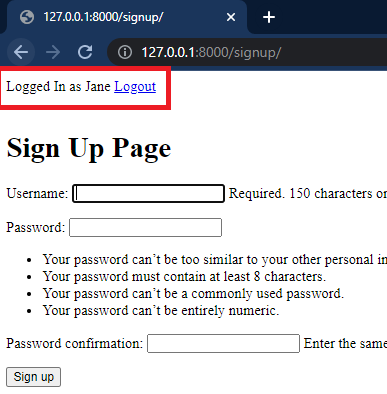
1. **Добавляем отступы на странице**

****



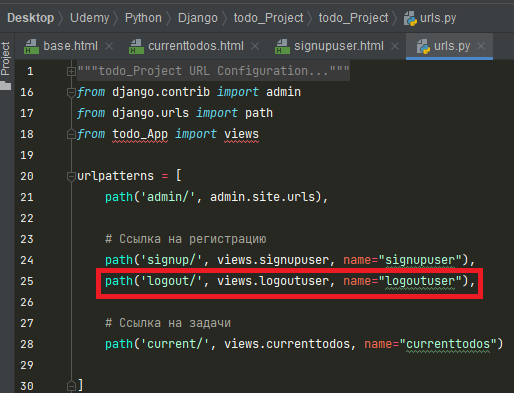
1. **То же самое производим на странице регистрации, чтобы пользователь мог видеть – вошел ли он в систему**
2. **Копируем блок кода с шаблоном из «currenttodos.html» в «todo\_App/templates/todo\_App/signupuser.html»**

****

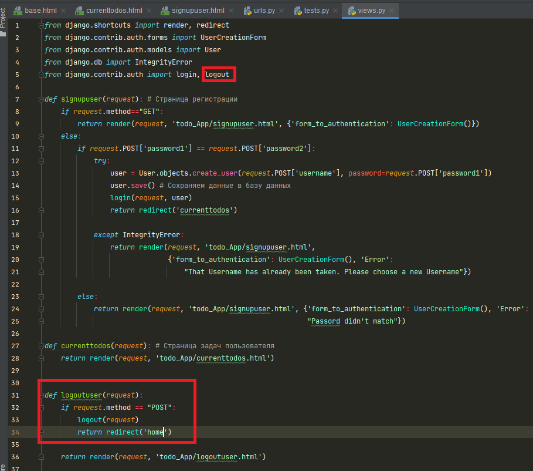


**66. Выход пользователя**

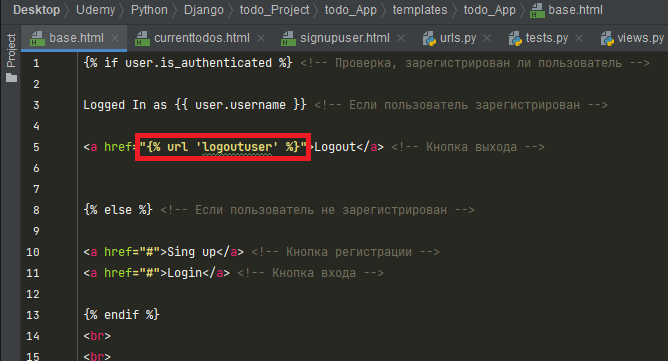
1. **Создаем новый адрес в блоке аутентификации в файле «urls.py»**

****

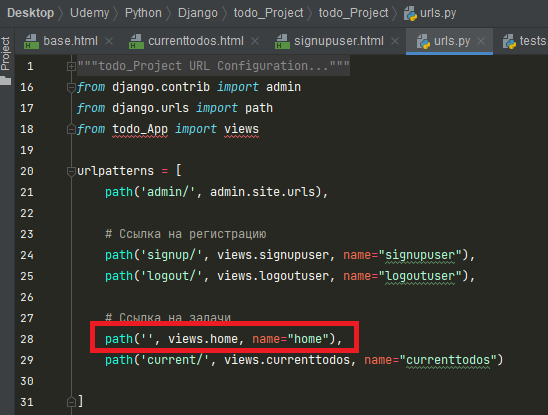
1. **Создаем функцию «logoutuser» в файле «views.py»**

****

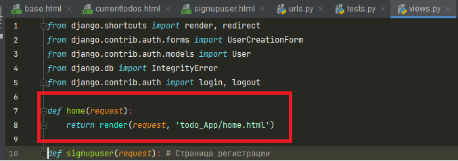
1. **Добавляем ссылку на «logoutuser» в файле «base.html»**

****

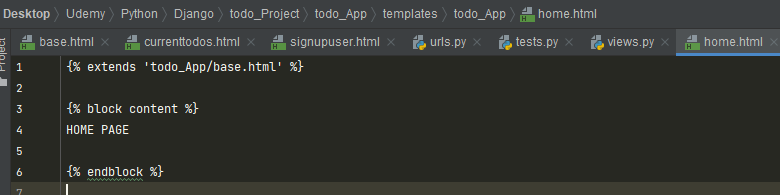
1. **Создаем домашнюю страницу**

****

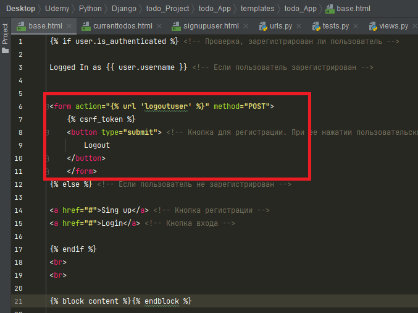
1. **В файле «views.py» создаем функцию «home»**

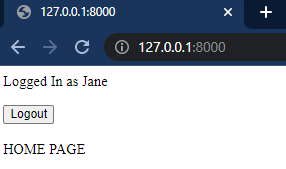
****

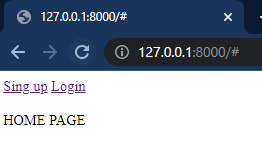
1. **Создаем страницу «todo\_App/templates/todo\_App/home.html» (можно скопировать из «currenttodos.hrml»)**



1. **В файле «base.html» даем возможность определения Post запроса, также создаем кнопку для выхода**

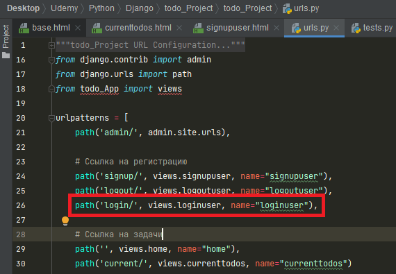
****



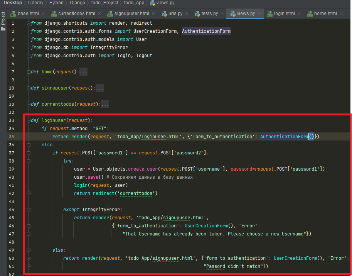


**67. Вход пользователя**

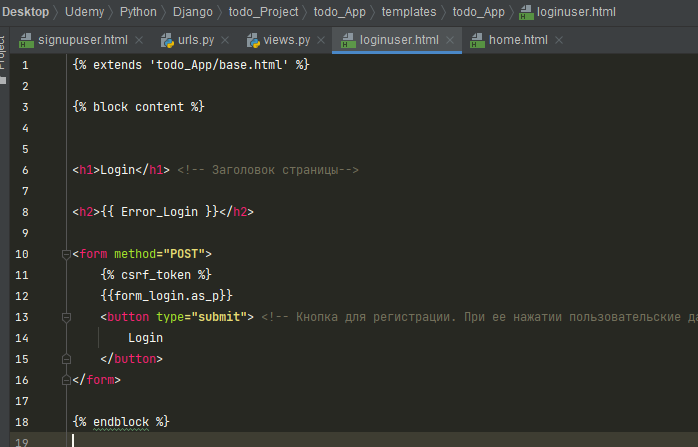
1. **Создаем ссылку на вход в файле «urls.py»**

****

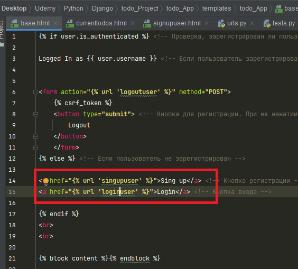
1. **Создаем функцию «loginuser» в файле «views.py»**

****

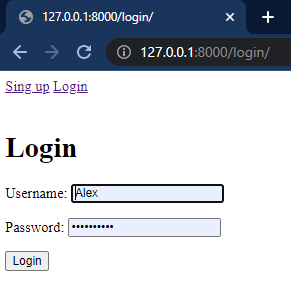
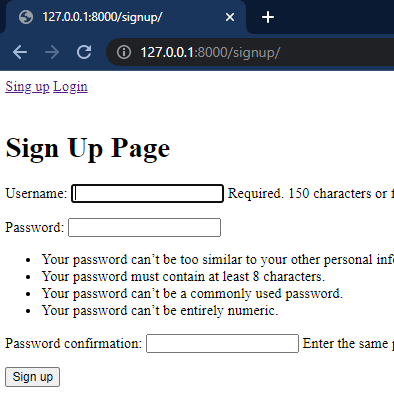
1. **Создаем страницу «loginuser.html» (можно скопировать с «signupuser.html»)**



1. **Делаем кнопки «Sing up» и «Login» в файле «base.html» кликабельными**

****

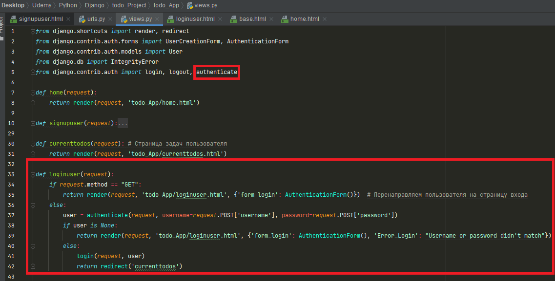
**Sign Up Login**



1. **При попытке входа возникает ошибка «MultiValueDictKeyError at /login/'password1'»**

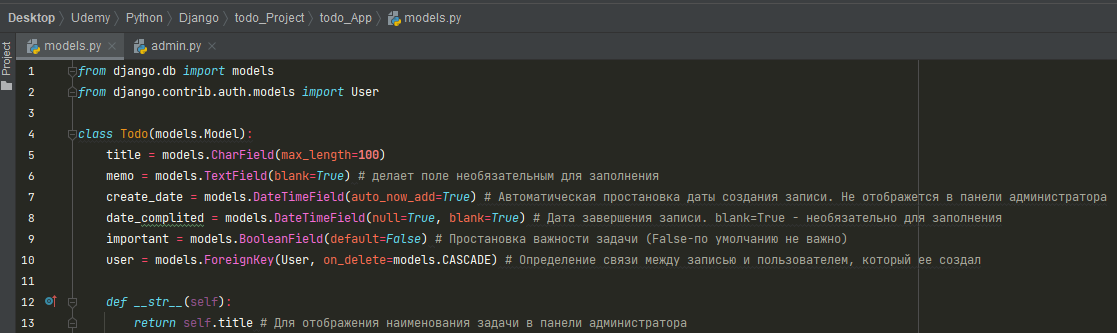
Это связано с блоком кода в файле «views.py», где производится проверка паролей в функции «loginuser»

Вместо проверки паролей мы должны производить проверку полной аутентификации пользователя с помощью функции «authenticate» в модуле Django



**68. Модель ToDo**

1. **Создаем модель TODO в файле «models.py»**



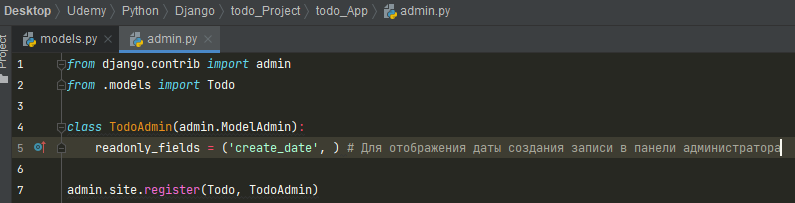
1. **Производим миграцию**

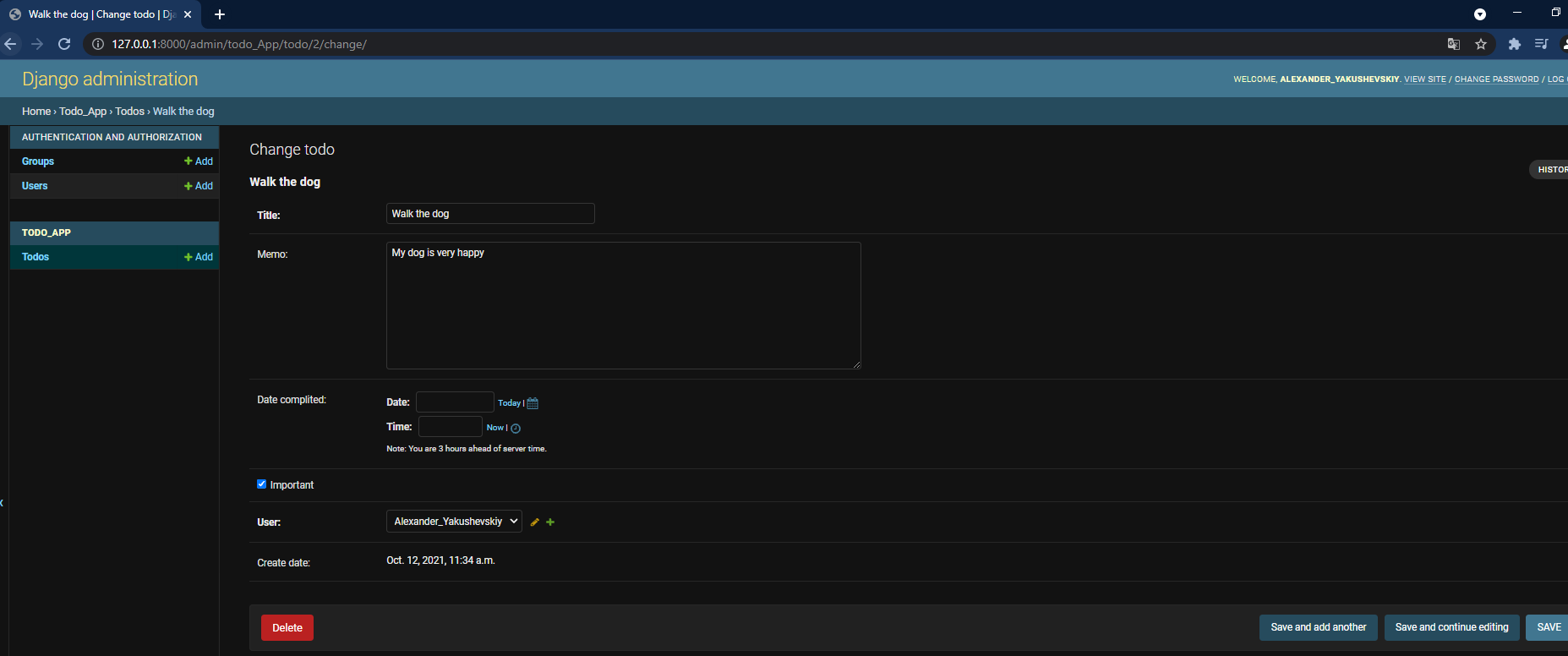
C:\Users\alexandr.yakushevski\Desktop\Udemy\Python\Django\todo\_Project>python manage.py makemigrations

1. **Выполняем команду на принятие миграции**

C:\Users\alexandr.yakushevski\Desktop\Udemy\Python\Django\todo\_Project>python manage.py migrate

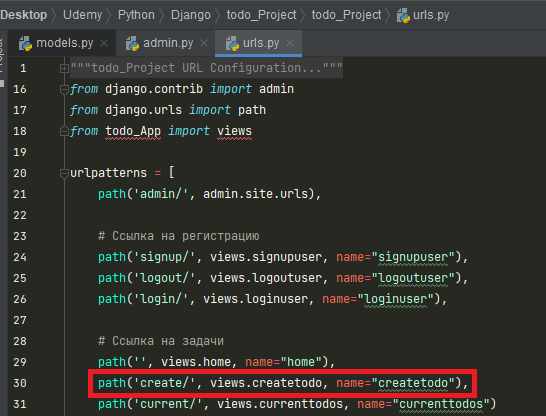
1. **Переходим к файлу «admin.py». Пишем код для отображения модели «Todo» в панели администратора**



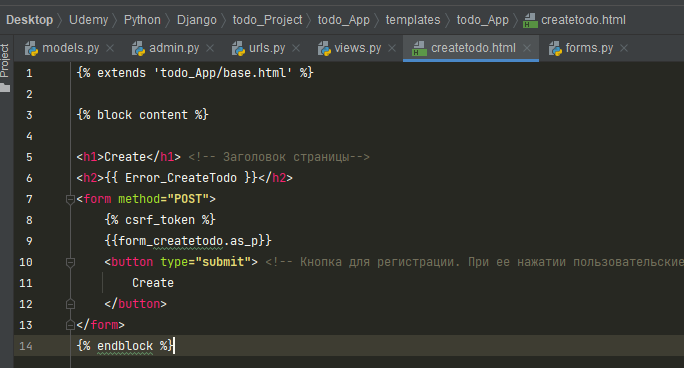


**69. Создание задач (ToDo)**

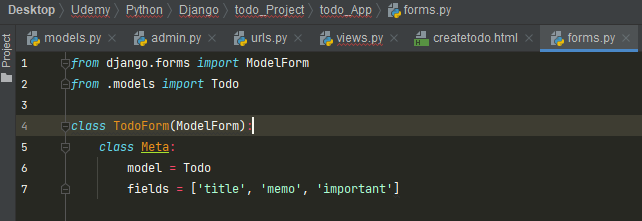
1. **Создание ссылки для личного кабинета пользователя в файле «urls.py»**

****

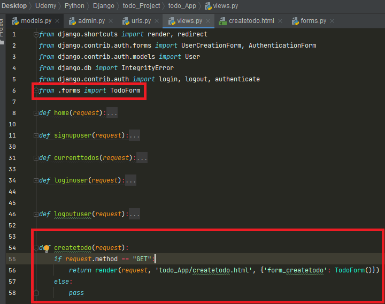
1. **Создаем файл «todo\_App/templates/todo\_App/createtodo.html»**

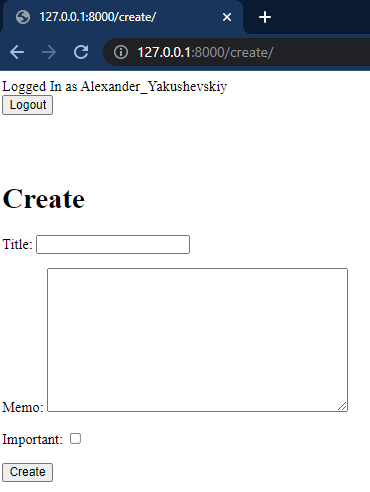


1. **Создаем форму для записей (создаем файл «todo\_App/forms.py»)**

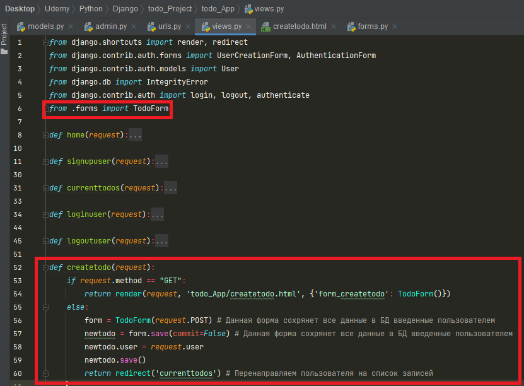


1. **В файле «views.py» создаем функцию «createtodo»**

****

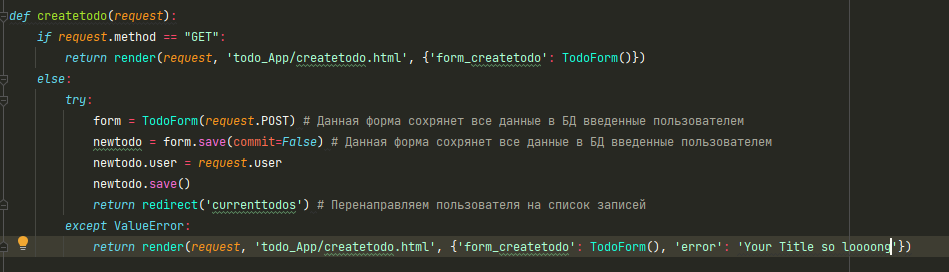


При попытке создания задачи будет ошибка. Для этого в файле «views.py» добавляем блок функции «createtodo»



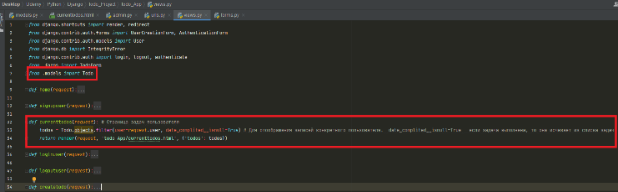
1. **В случае ввода пользователем более 100 символов в поле для наименования задачи, мы перенаправляем его на страницу с соответствующем сообщением об ошибке**



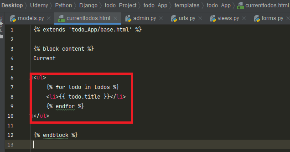


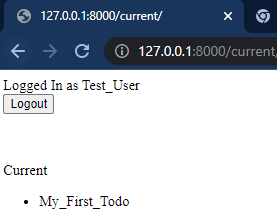
**70. Список задач**

1. **В файле «views.py» к функции «currenttodos» добавляем функционал, который бы отображал все задачи из БД**

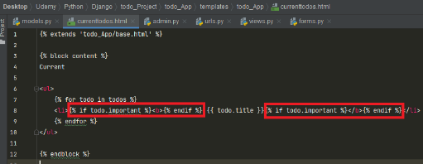
****

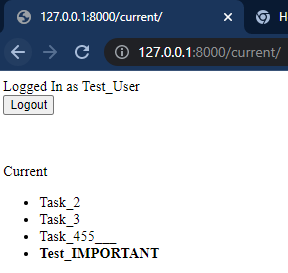
1. **Переходим к файлу «currenttodos.html»**

****

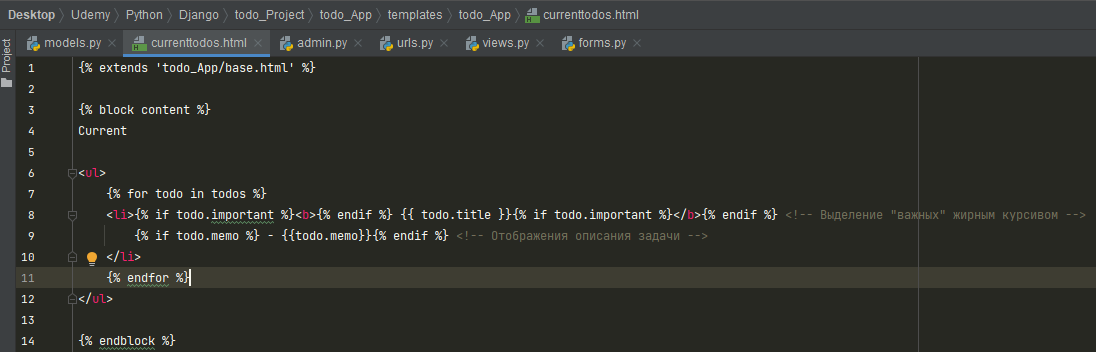


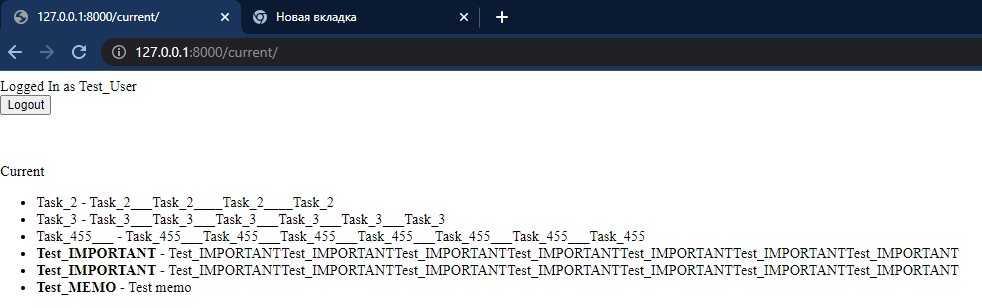
1. **Выделяем «важную» задачу в файле «currenttodos.html»**

****



**4. Отображаем часть описания задачи, если оно имеется**

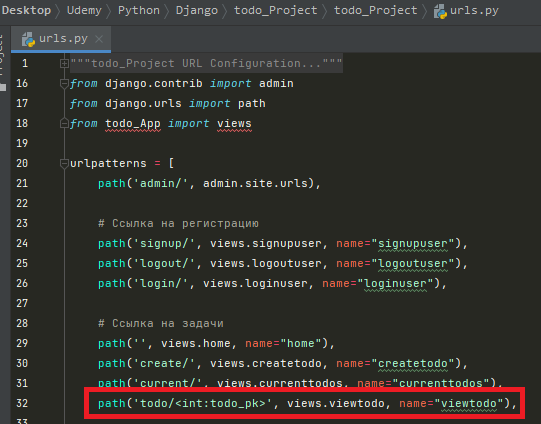




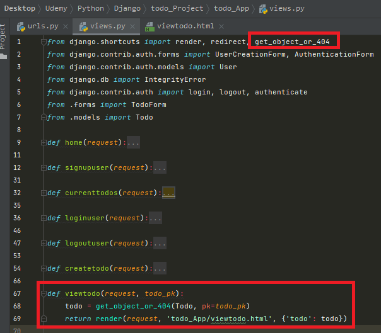
**71. Просмотр и обновление задач**

**Создаем возможность открытия записи (задачи)**

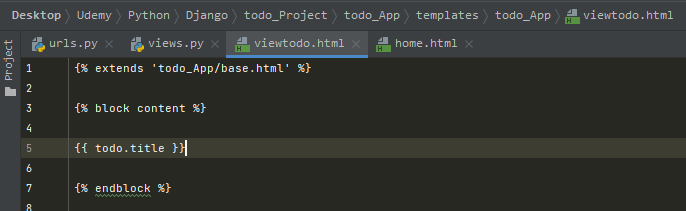
1. **Создаем новый URL**

****

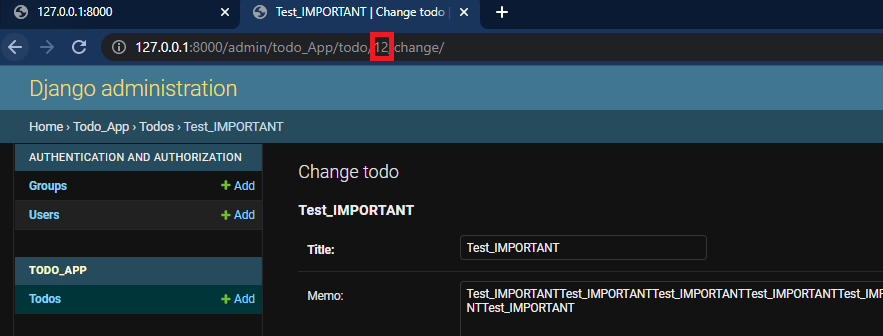
1. **Добавляем новую функцию (viewtodo) в файл «views.py»**

****

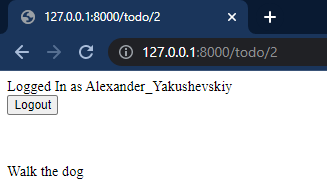
1. **Создаем html файл «todo\_App/templates/todo\_App/viewtodo.html» (можно скопировать с файла «home.html»)**



Если зайти в панель администратора и перейти к любой из задач, то в ссылке сайта будет отображаться № задачи

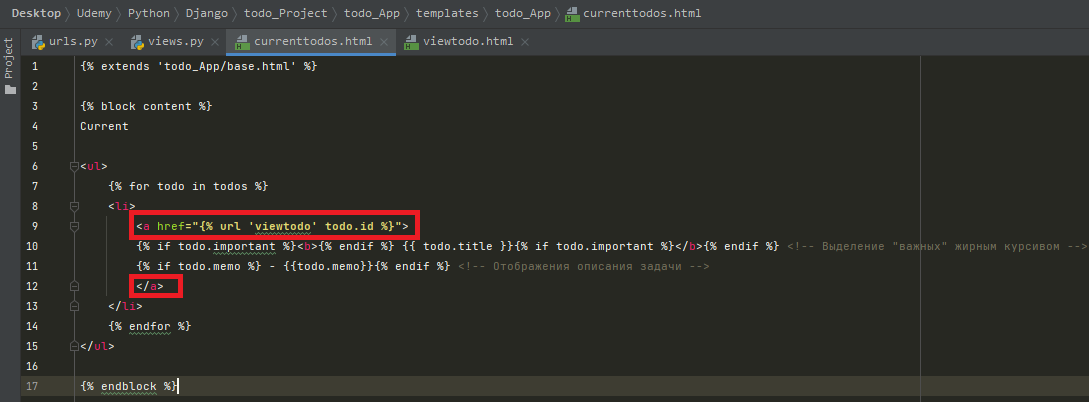


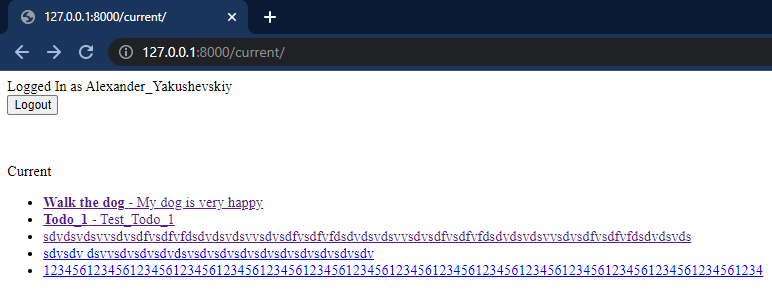
Также к задаче можно перейти через адрес сайта



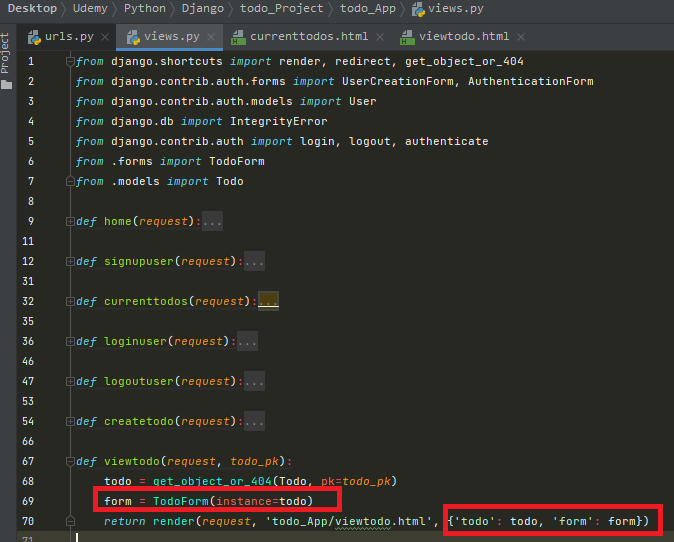
1. **Делаем записи (задачи) кликабельными**

Для этого в файле «currenttodos.html» заключаем код в тэг «<a>», указав в качестве ссылки страницу с описанием заметки



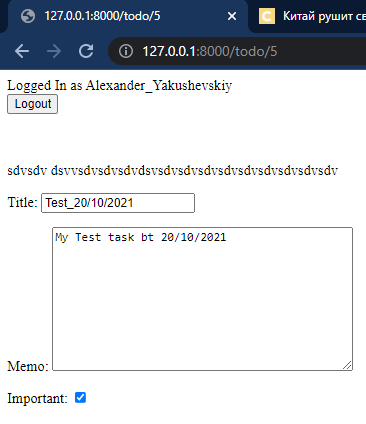
****

1. **Добавляем возможность изменения записей с помощью загрузки формы создания записей**
2. **В файле «views.py» наполняем функцию «viewtodo»**

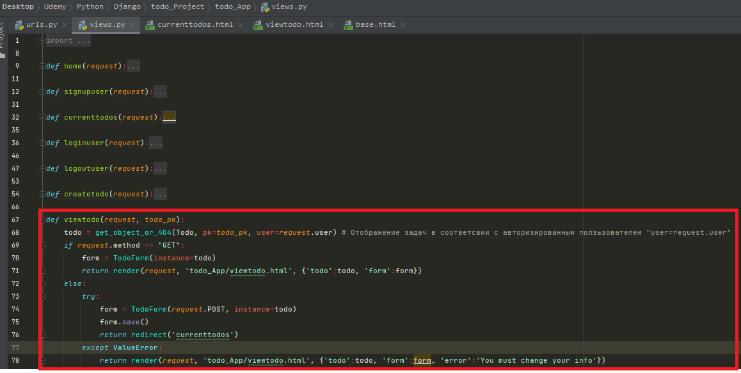
****

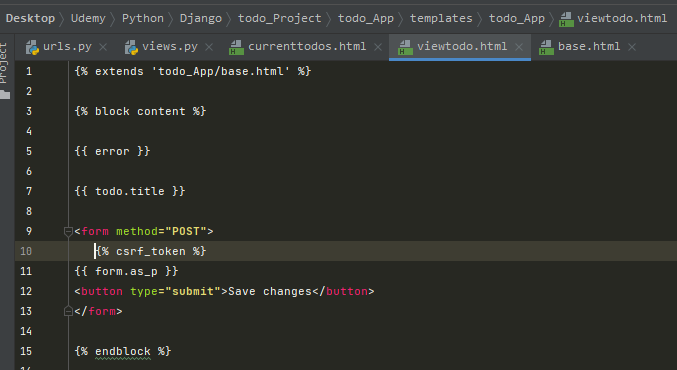
1. **В файл «viewtodo.html» добавляем отображение формы**

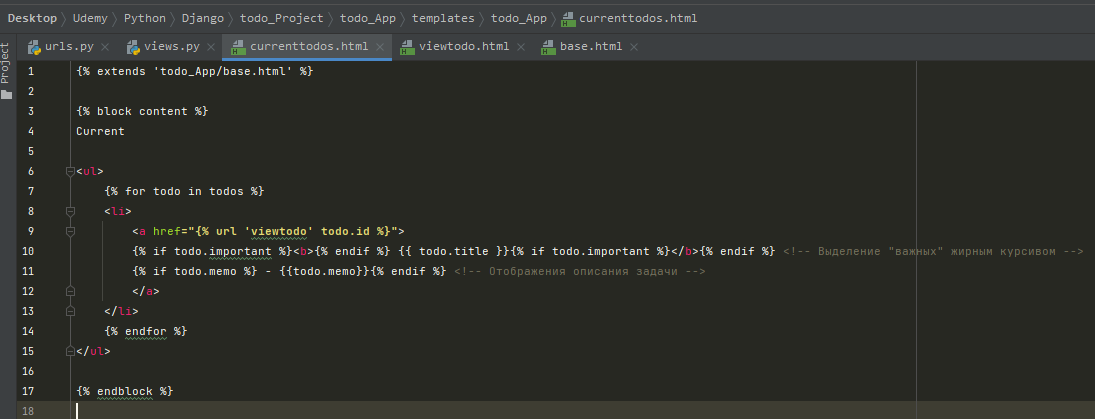
****

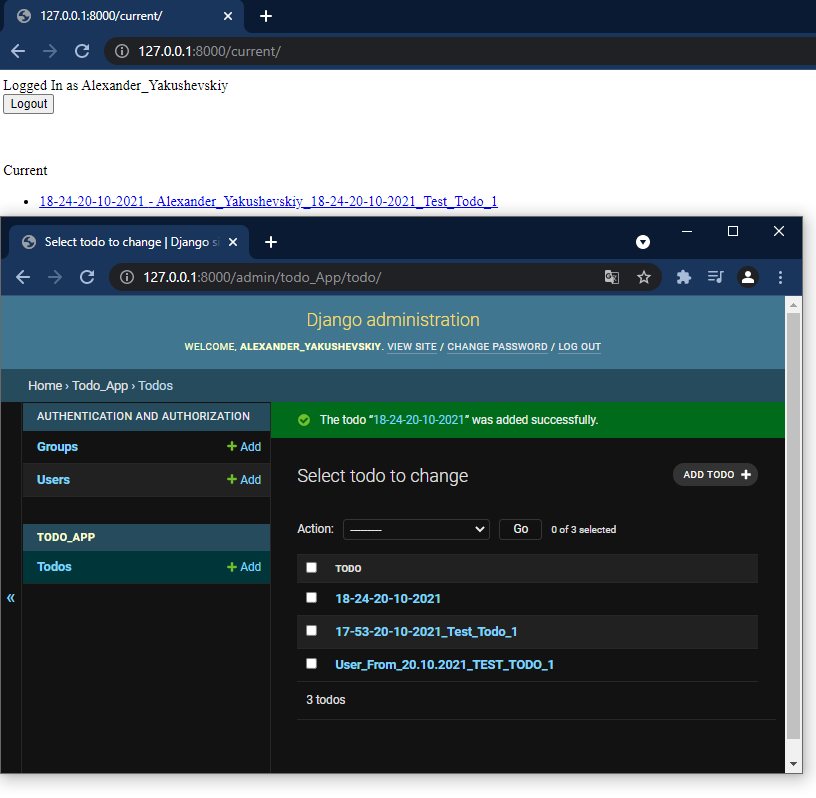


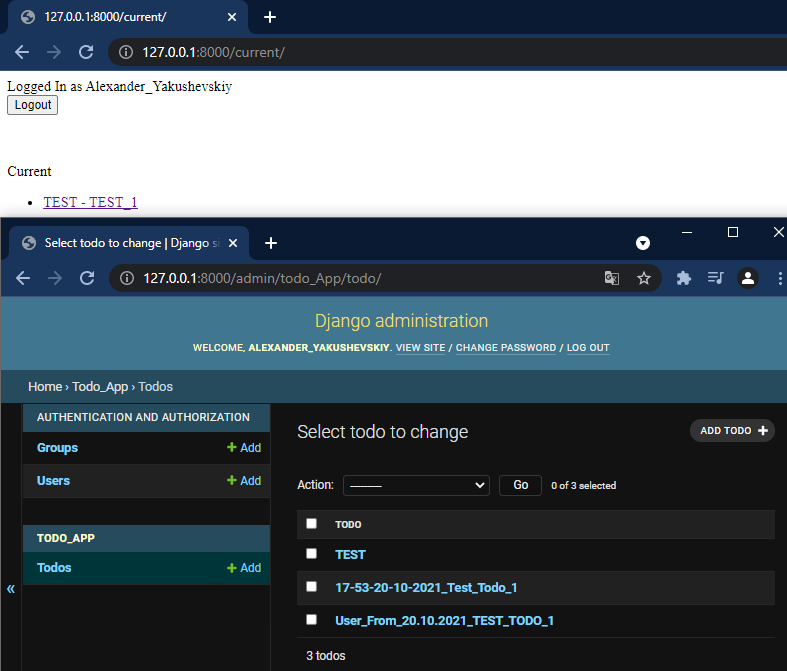
1. **Создаем возможность сохранения отредактированной записи путем осуществления отправки новых данных в базу данных с помощью POST в файле «views.py»**

****



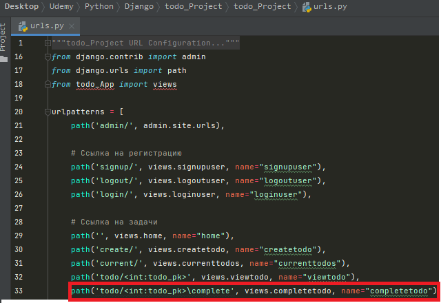




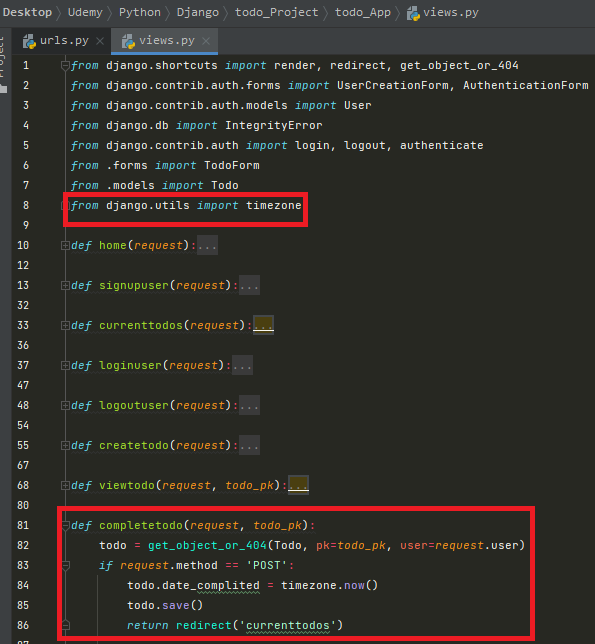


**72. Завершение и удаление задач**

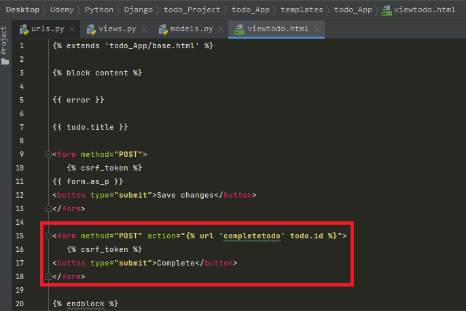
1. **Добавляем возможность «Выполнения» задачи**
2. **В файле «urls.py» создаем адрес, который будет продолжением одного из адресов**

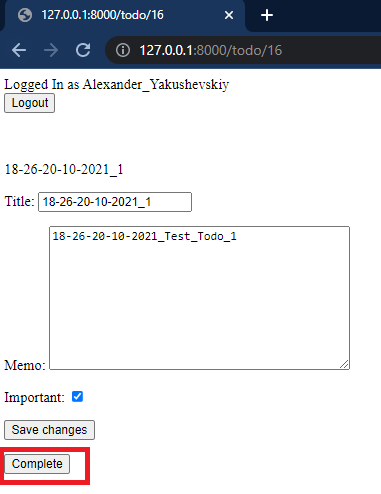
****

1. **В файле «views.py» создаем функцию «completetodo» (Todo – класс из файла «models.py», todo.** **date\_complited – метод класса Todo)**

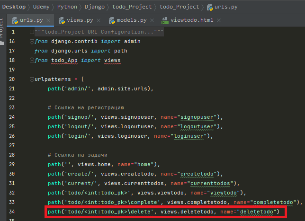
****

1. **Создаем форму (кнопку) в файле «viewtodo.html»**

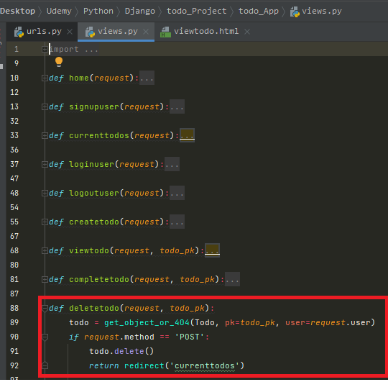
****

****

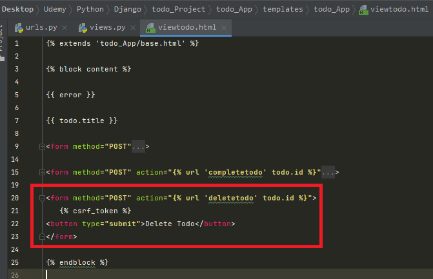
1. **Реализации возможности удаления записи (аналогичен с созданием завершения задачи, только используется не todo.save(), а todo.delete()**
2. **Создаем ссылку в файле «urls.py»**



1. **Создаем функцию «deletetodo» в файле «views.py» (УДАЛЕННЫЕ ЗАПИСИ МОЖНО ПОСМОТРЕТЬ В БАЗЕ ДАННЫХ)**

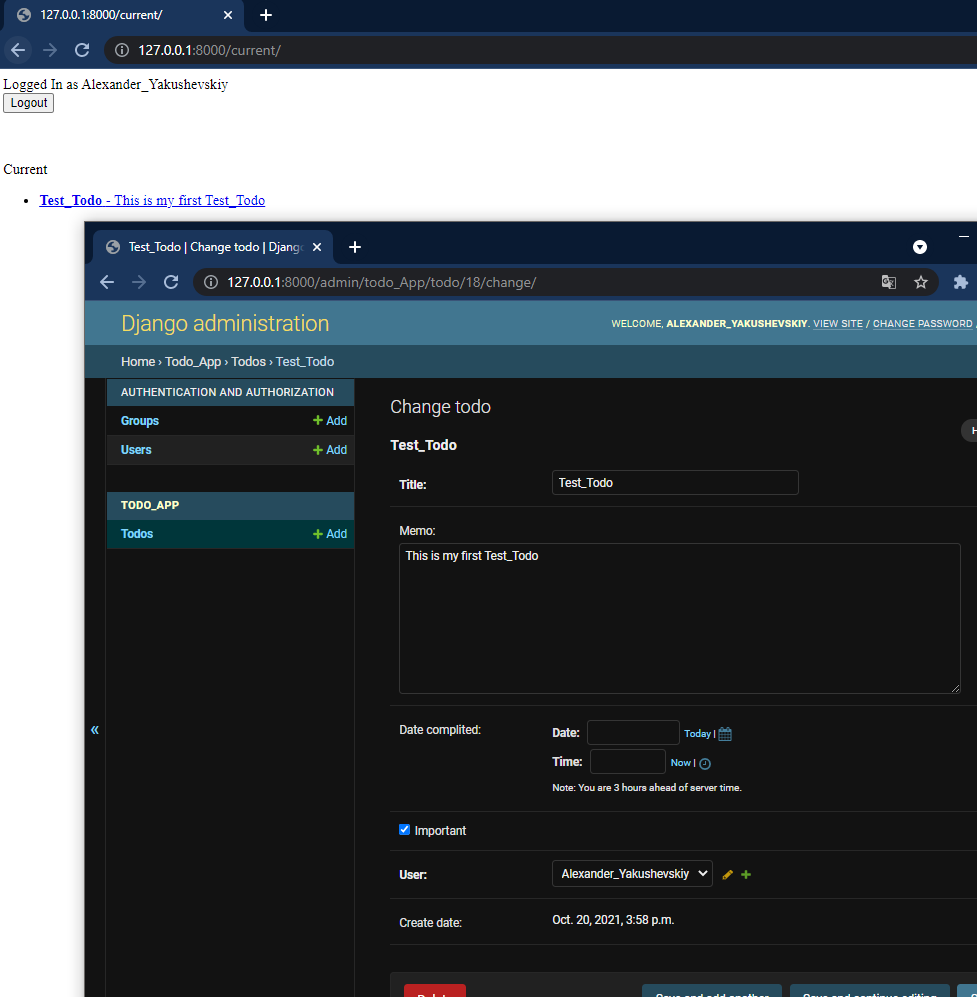
****

1. **В файле «viewtodo.html» создаем кнопку для удаления записи**

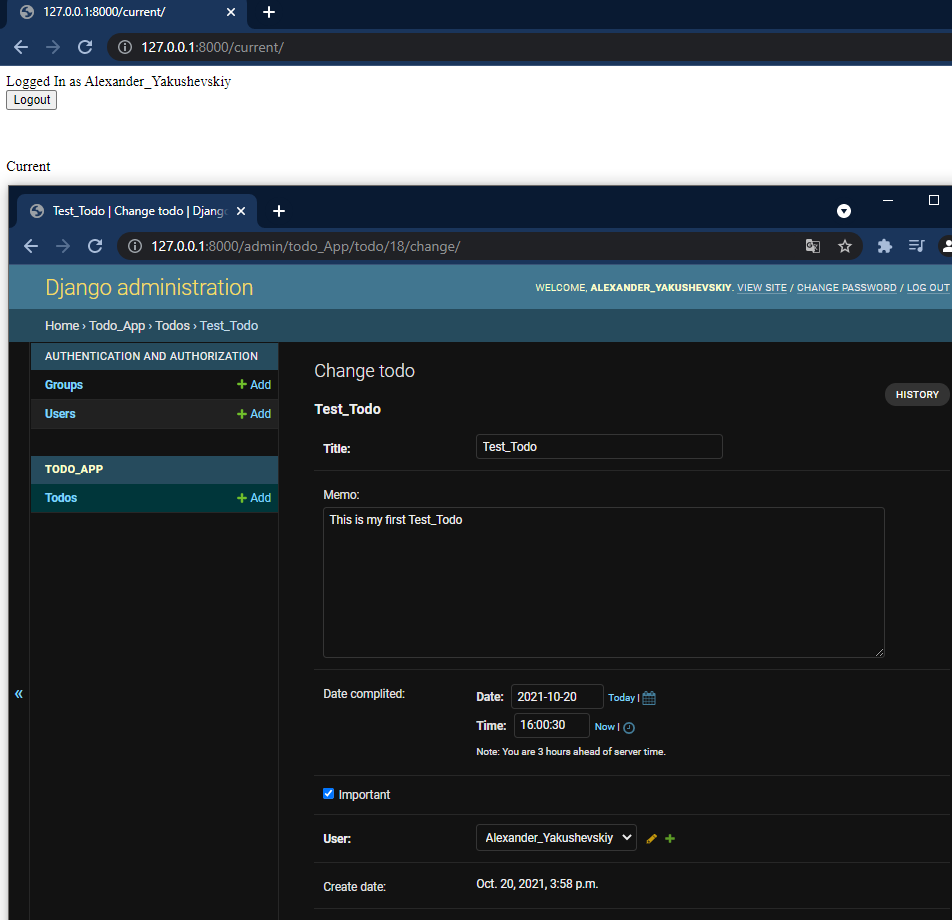
****

**Сравнение Выполненной и удаленной записи**

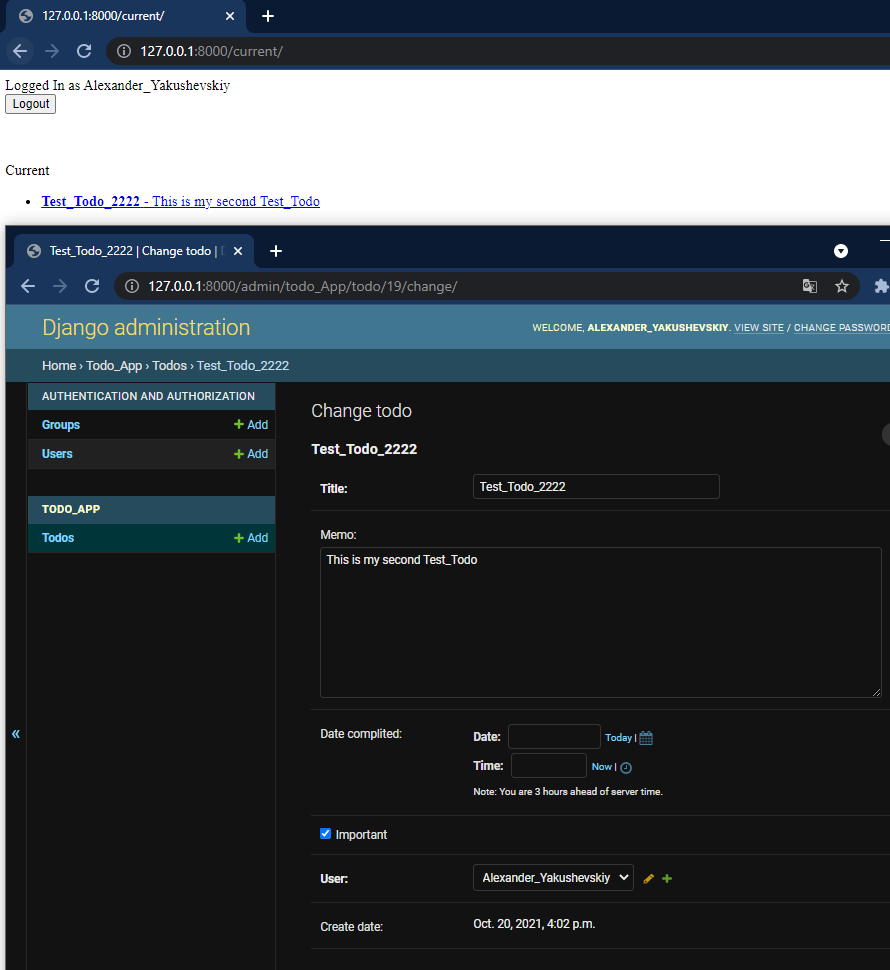
**Выполненная задача**



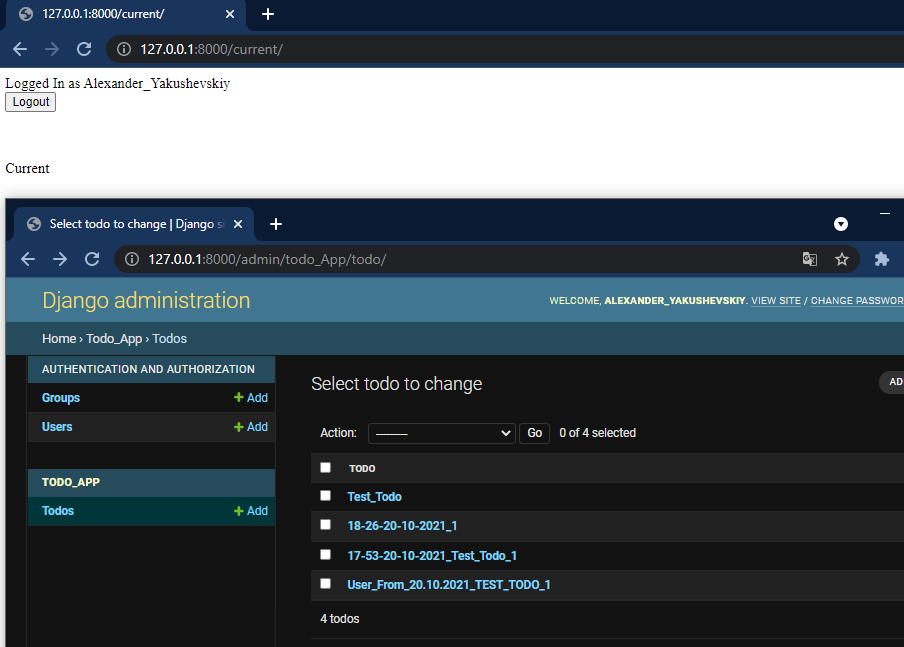
**При переходе к задаче и нажатии кнопки выполнения задача исчезает, но отображается в панели администратора**



**Удаленная задача**

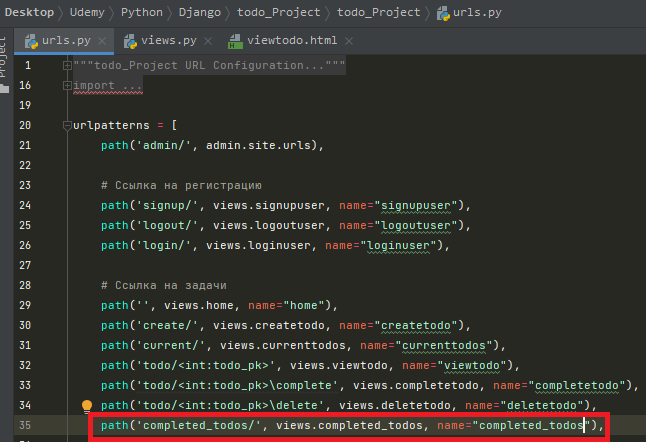


**При переходе к задаче и нажатии кнопки удаления задача исчезает, и не отображается в панели администратора, но к ней можно получить доступ через базу данных**

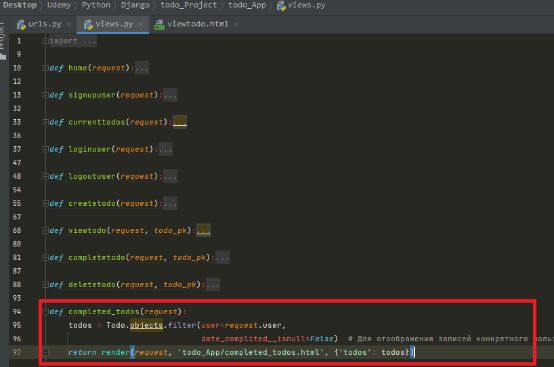


**73. Отображение завершённых задач**

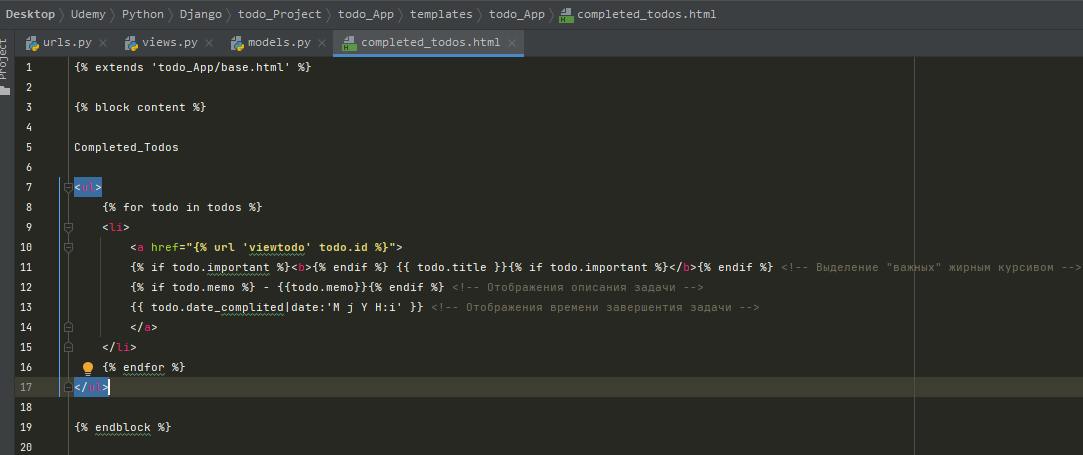
1. **Создаем новую ссылку в файле «urls.py»**

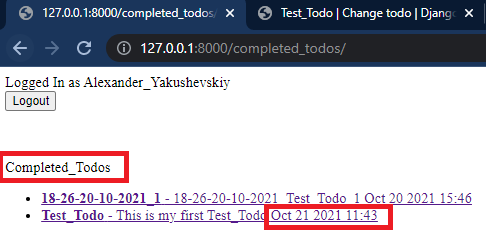
****

1. **Создаем новую функцию «completed\_todos» в файле «views.py»**

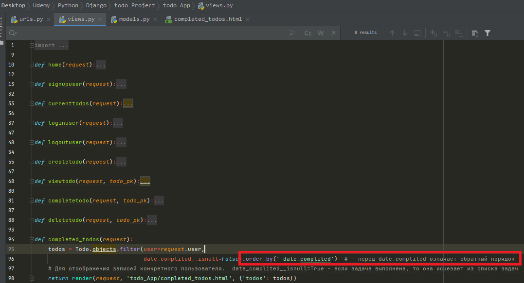
****

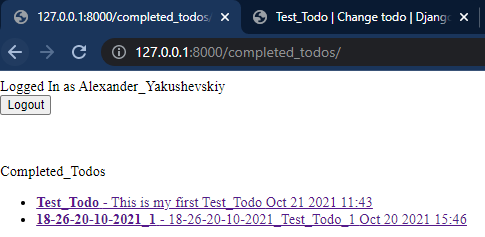
1. **Создаем html файл «completed\_todos.html» (аналогично как в файле currenttodos.html)**



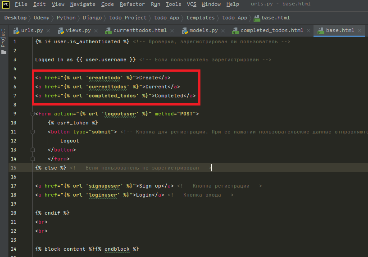
****

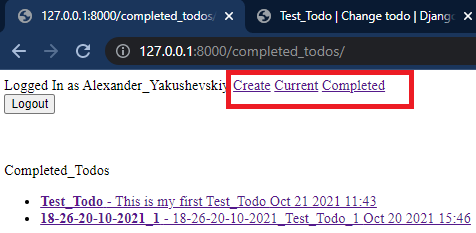
1. **Создаем возможность отображения выполненных задач в хронологическом порядке**
2. **В файле «views.py» дополняем функцию «completed\_todos»**

****

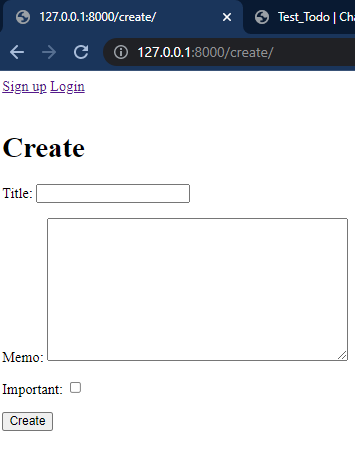


1. **Добавляем кнопки в файле «base.htlm»**

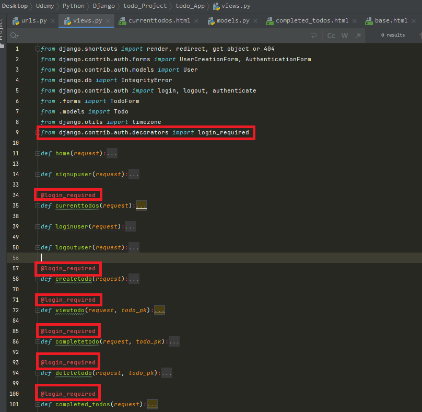
****

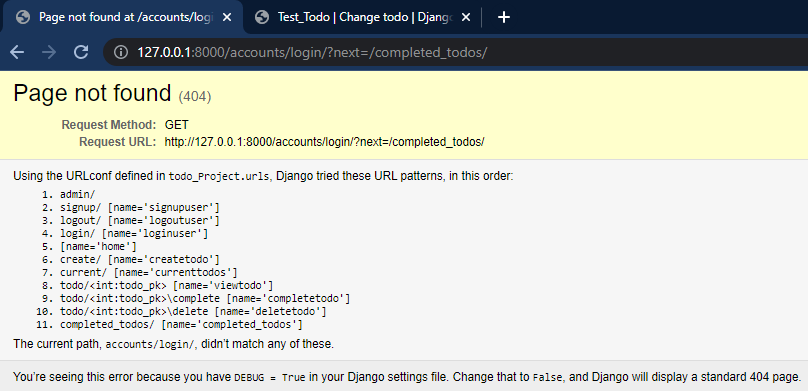
****

1. **Если НЕАВТОРИЗОВАННЫЙ пользователь в ссылке введет «\create», то он сразу попадет на страницу создания записи. Задаем правило «только авторизованный пользователь имеет доступ к страницам»**

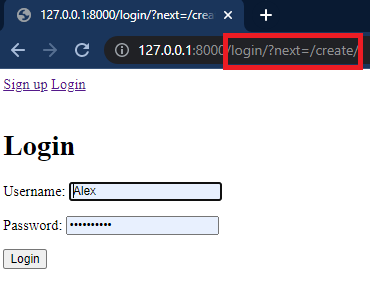
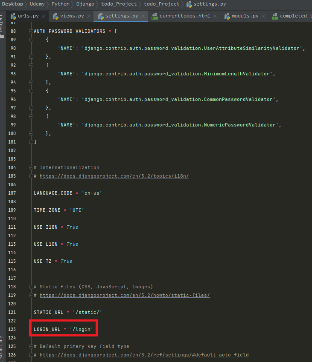


1. **Переходим к файлу «views.py» и импортируем код, который будет сообщать пользователю, что только авторизованный пользователь имеет доступ к страницам (декорируем функции) (будет отображаться ошибка 404)**

****



1. **Добавляем команду для неопределенных (неавторизированных) пользователей в файле «settings.py», которая будет перенаправлять на страницу входа**

****

**74. Стилизация**

1. **Оформление взято у автора (**<https://github.com/zappycode/django3-todowoo-project>). Возможны неточности в ссылках (автор указывает дерикторию с страницами /todo/templates, у меня /todo\_App/ templates