**Харківський національний економічний університет**

**імені Семена Кузнеця**

**ЗВІТ**

**З ВИКОНАННЯ Самостійної роботи**

**за дисципліною: *“Інформаційні системи та інтернет технології”***

**Варіант № 24**

**Виконав: студент факультету Інформаційних технологій**

**3 курсу, спец. Кібербезпека,**

**групи 6.04.125.010.21.2**

**Щербаков Олексій Віталійович**

**Перевірила:**

**Солодовник Ганна Валеріївна**

**ХНЕУ ім. С. Кузнеця**

**2024**

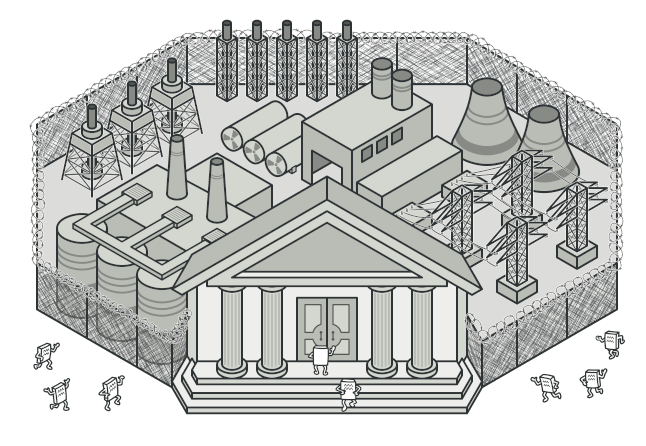
**Завдання:**

1. Надати опис патерну відповідно до номеру за журналом: Фасад (Facade)
2. Знайти на теренах Інтернет приклад коду аутентифікації користувачів та надати пояснення до цього коду

**Опис патерну «Фасад»:**

## Суть патерна

**Фасад** — це структурний патерн проектування, який надає простий інтерфейс до складної системи класів, бібліотеки або фреймворку.

****

## Проблема

Вашому коду доводиться працювати з великою кількістю об’єктів певної складної бібліотеки чи фреймворка. Ви повинні самостійно ініціалізувати ці об’єкти, стежити за правильним порядком залежностей тощо.

В результаті бізнес-логіка ваших класів тісно переплітається з деталями реалізації сторонніх класів. Такий код досить складно розуміти та підтримувати.

## Рішення

Фасад — це простий інтерфейс для роботи зі складною підсистемою, яка містить безліч класів. Фасад може бути спрощеним відображенням системи, що не має 100% тієї функціональності, якої можна було б досягти, використовуючи складну підсистему безпосередньо. Разом з тим, він надає саме ті «фічі», які потрібні клієнтові, і приховує все інше.

Фасад корисний у тому випадку, якщо ви використовуєте якусь складну бібліотеку з безліччю рухомих частин, з яких вам потрібна тільки частина.

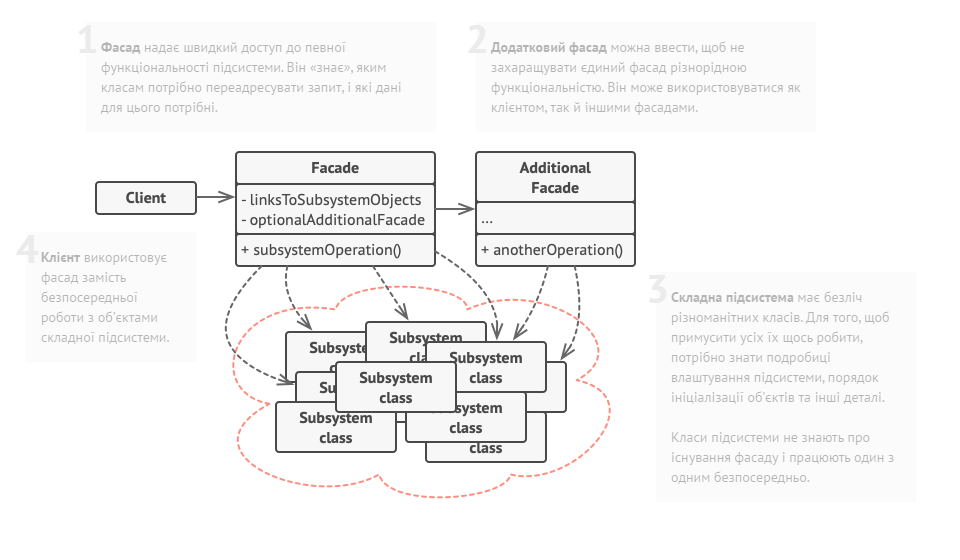
Наприклад, програма, що заливає в соціальні мережі відео з кошенятками, може використовувати професійну бібліотеку для стискання відео, але все, що потрібно клієнтському коду цієї програми, — це простий метод encode(filename, format). Створивши клас з таким методом, ви реалізуєте свій перший фасад.

## Аналогія з життя

## 

## Коли ви телефонуєте до магазину і робите замовлення, співробітник служби підтримки є вашим фасадом до всіх служб і відділів магазину. Він надає вам спрощений інтерфейс до системи створення замовлення, платіжної системи та відділу доставки.

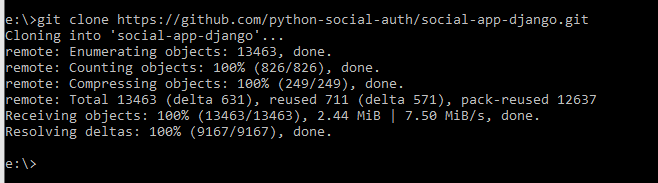
## Структура

****

**Приклад коду автентифікації користувачів та надати пояснення до цього коду**

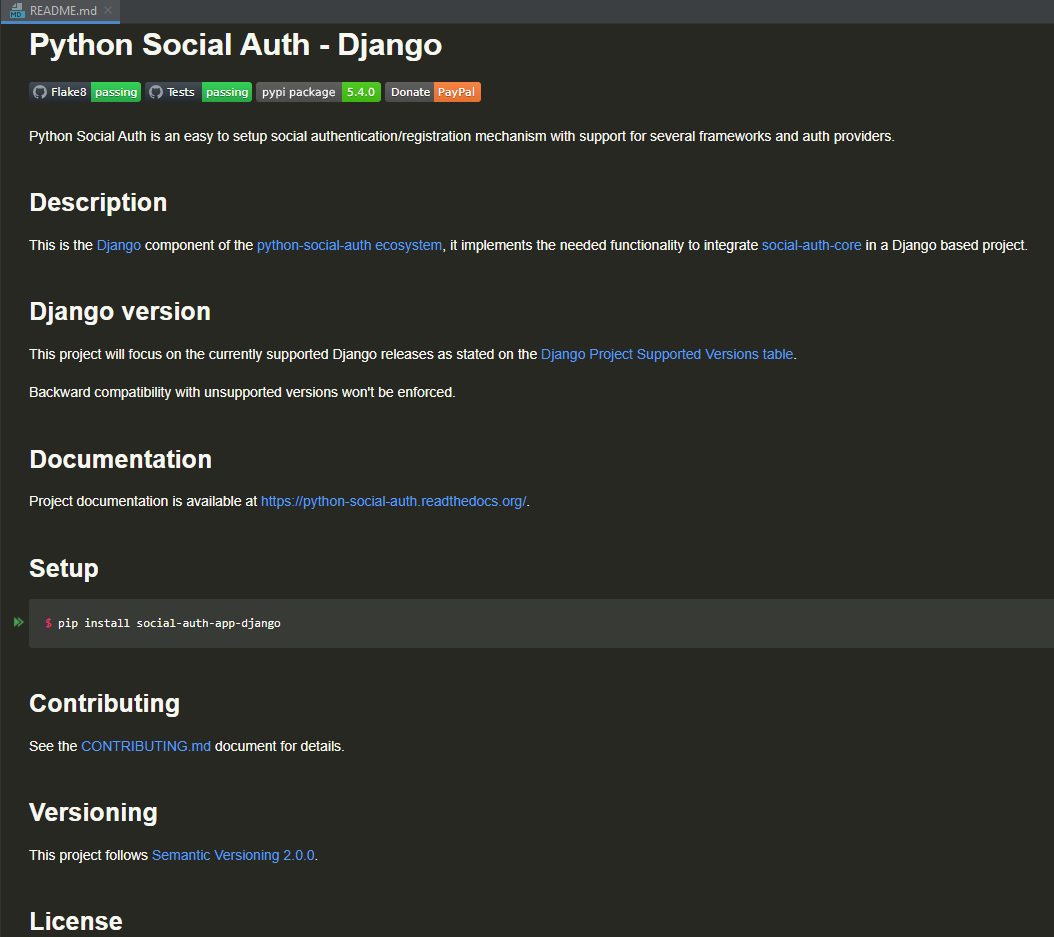
На гітхабі я знайшов даний репозиторій <https://github.com/python-social-auth/social-app-django> та буду аналізувати код зданого репозиторію

Спочатку зроблю клон репозиторію, щоб розглядати ко було легше



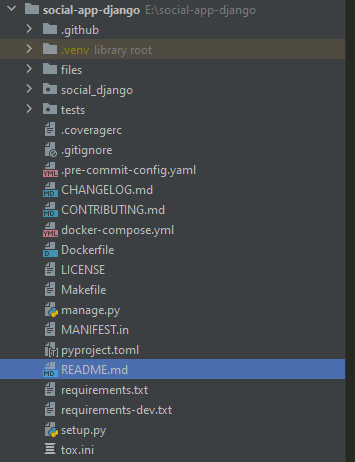
Та відкрию скопійований репозиторій у PyCharm-і

Одразу відкривається файл «README.MD»

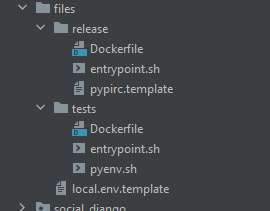


Єдине, що можна зрозуміти, це те, що для встановлення нам потрібно лише прописати команду «pip install social-auth-app-django»

Отже перейдемо до структури проекту

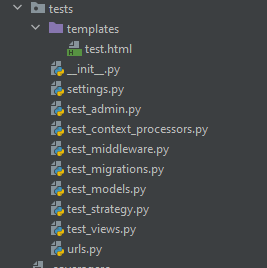


Структура проекту має папку «files»



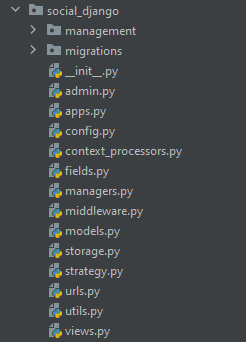
Яка всередині має ще декілька папок, але вони мають докер файли та налаштування, тому одразу закриємо її

Далі перейдемо до папки з тестами



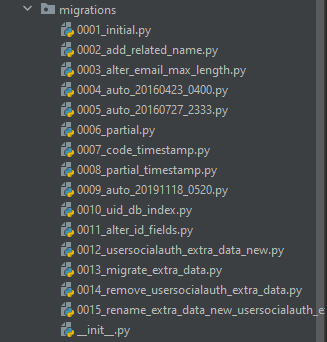
Як можна здогадатись – тут знаходяться лише тести до проекту

Тому теж закриємо її та перейдемо до папки «social\_django»

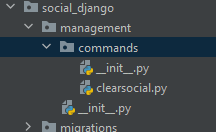


Як можна побачити тут основні файли проекту та ще дві папки

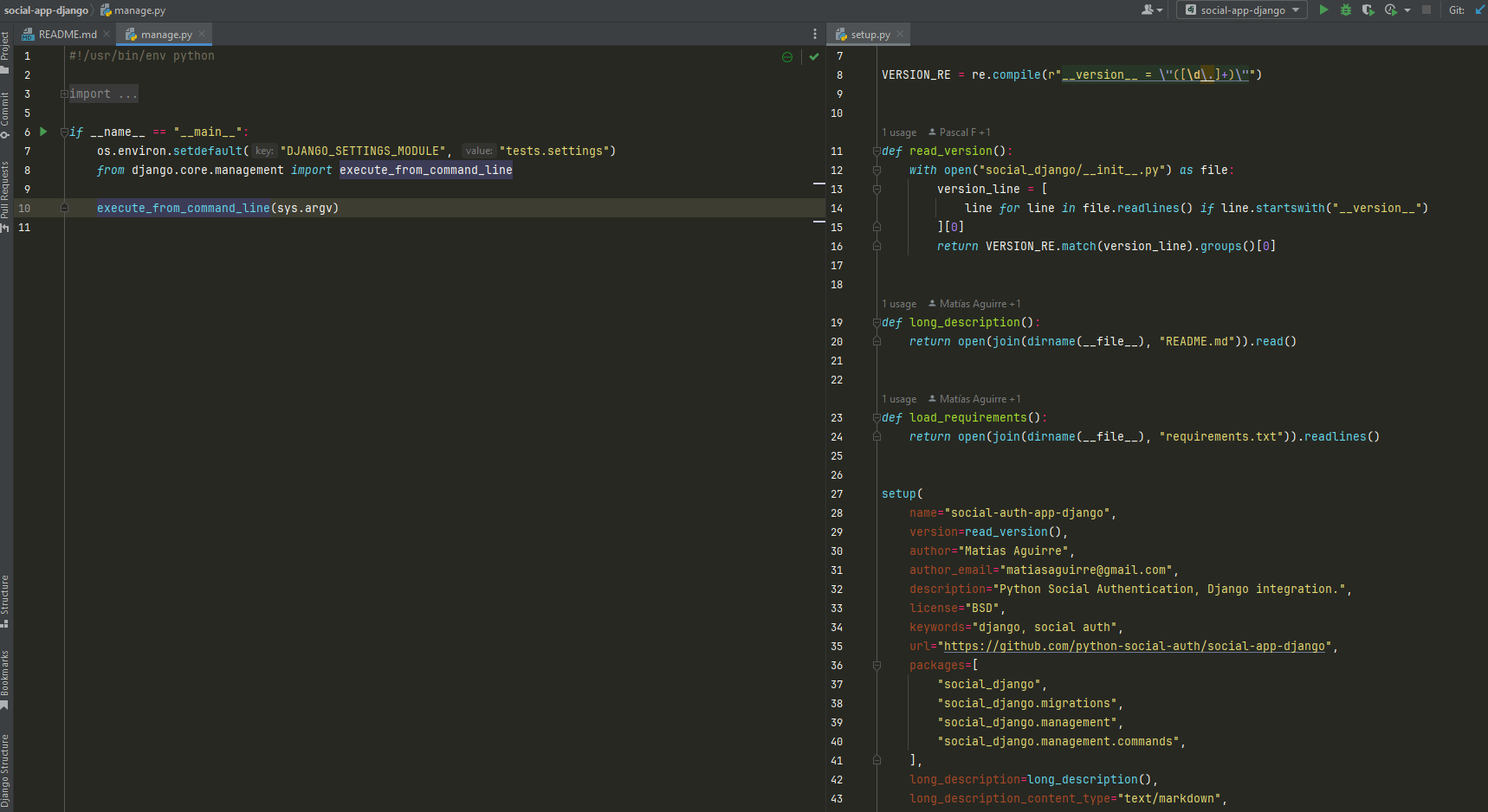
З міграціями – для міграції БД



Та «management» яка всередині має «commands»



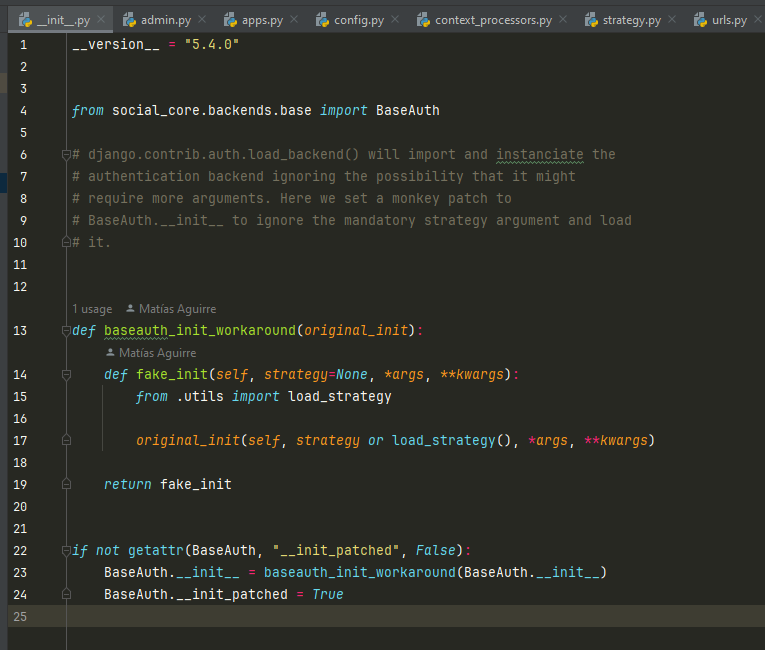
Також у головній папці є два файли – setup.py та manage.py, одже розпочну з перегляду цих файлів



Трохи подивившись на ці файли, можна зрозуміти,  
що manage.py лише запускає проект, а команду для запуску ми бачили у README.MD та файл setup.py потрібен лише для налаштування

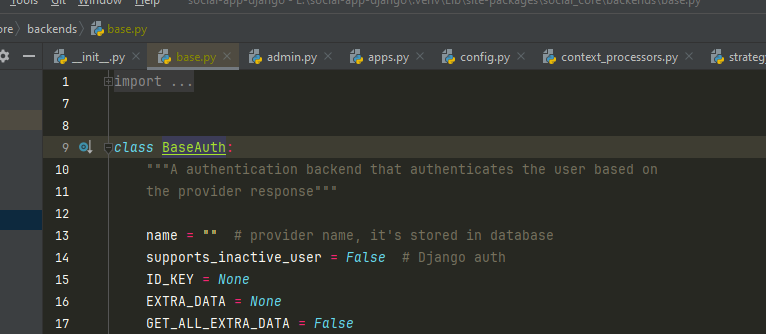
Отже відповідно до тексту вище нам потрібно переміститись до папки «social\_django», бо весь проект знаходиться там

Зазвичай код починає інііціалізуватись з файлу \_\_init\_\_.py, тому почну аналізувати з нього



Тут лише ініціалізація, тому перейдемо до класу BaseAuth

Відповідно до кольору файлу, можна зрозуміти, що він відноситься до файлів бібліотеки

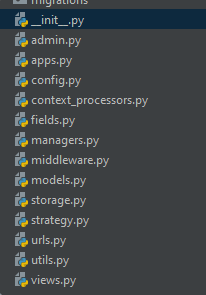


Тепер перейду до функції «load\_strategy»

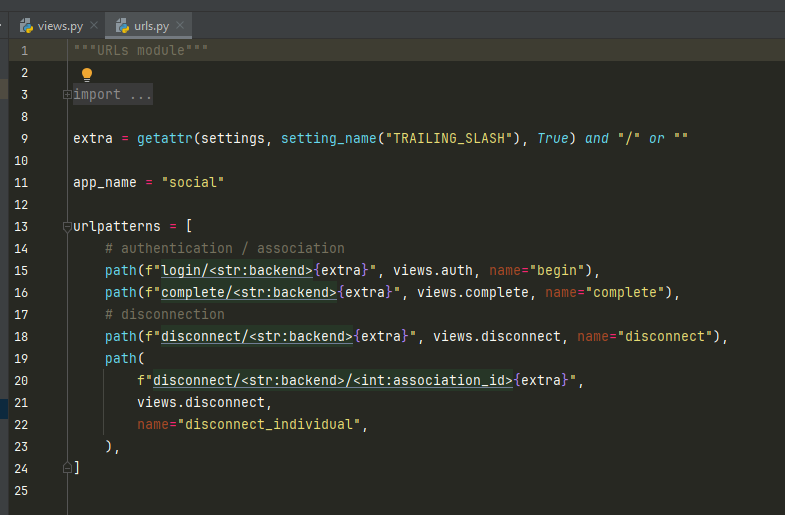


Можна зрозуміти, що те, як буде працювати код буде залежати від атрибутів при запуску

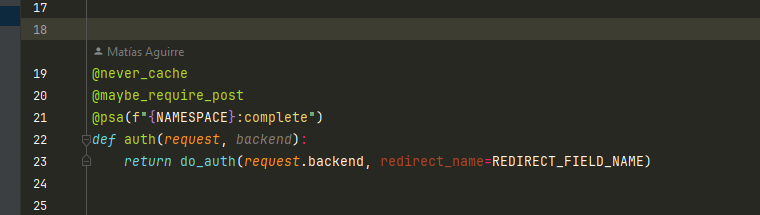
Також ми знаємо, що це застосунок на джанго, отже потрібно ще раз проаналізувати структуру файлів



У файлі «urls.py» можна зрозуміти, які шляхи є

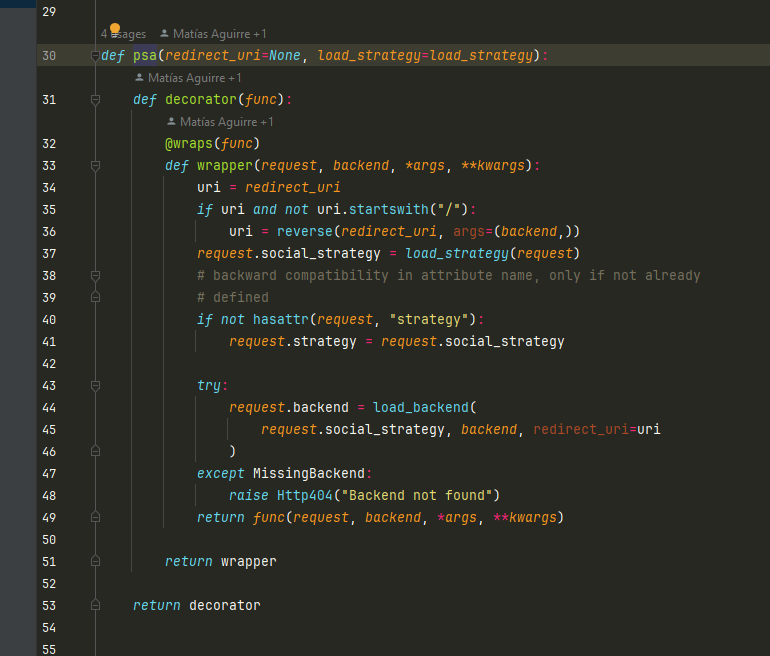


Далі можна перейти до фалу “views.py” оскільки тут може знаходитись основна частина логіки



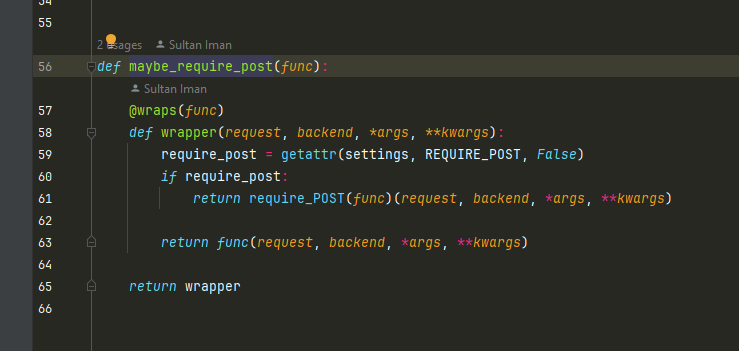
Почнемо з функції «auth»

Тут є декоратор “psa”



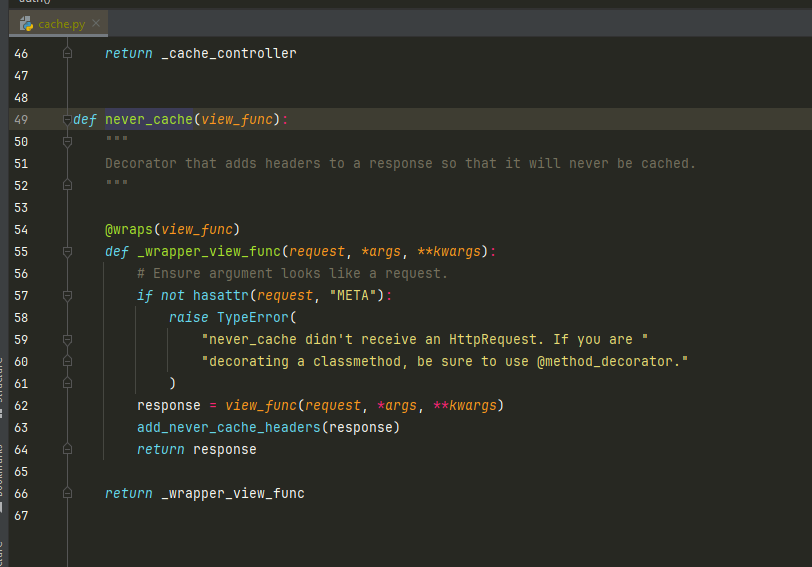
Можна зрозуміти, що це валідація «url»

Наступним виконається декоратор «maybe\_require\_post»



З назви та за кодом можна зрозуміти, що воно дозволяє, або заперечує запрос типу “POST”

Наступний «never\_cache»



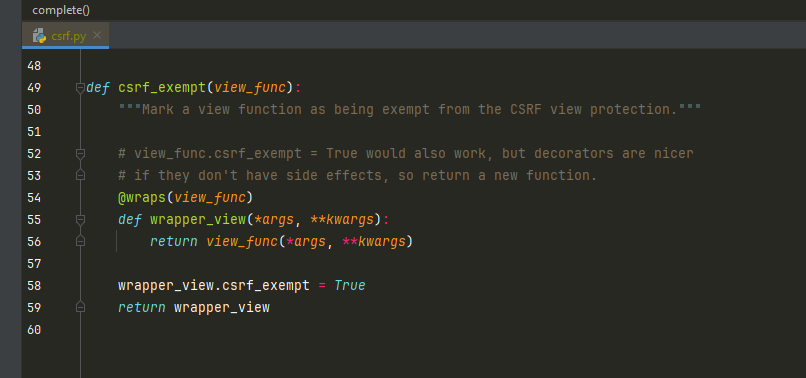
Він суто робить респонс, щоб він не був закешований

Всередині викликається функція «do\_auth», яка є частиною бібліотеки

Далі є функція «complete», яка є у в’ю

Знову декоратор з валідацією url

«csrf\_exempt»



Знову це файл бібліотеки, та основна суть у тому, щоб захистити від csrf

Та декоратор «never\_cache», про який я вже писав

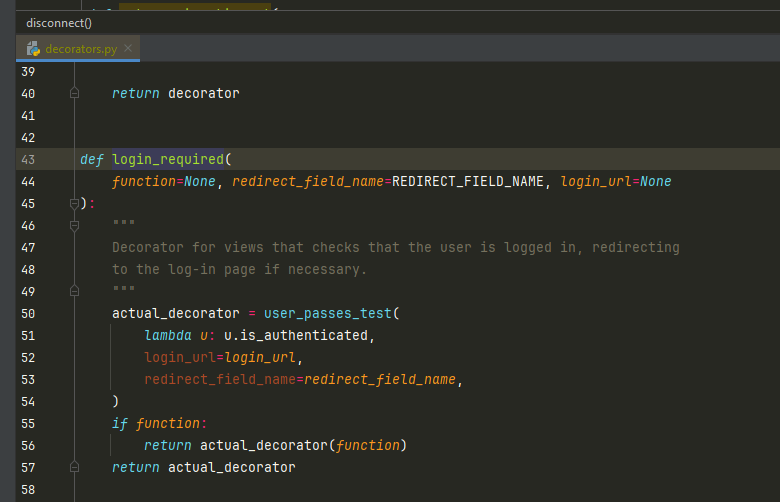
Наступна функція «disconnect»



Стосовно декораторів – захист від CSRF

“require\_POST” – перевіряє запит на тип запиту

Перевірка url



Перевірка на те, щоб користувач був авторизований

Та не зберігання кешу