**Харківський національний економічний університет**

**імені Семена Кузнеця**

**ЗВІТ**

**З ВИКОНАННЯ Самостійної роботи**

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

**Варіант № 4**

**Виконав: студент факультету Інформаційних технологій**

**3 курсу, спец. Кібербезпека,**

**групи 6.04.125.010.21.2**

**Бойко Вадим Віталійович**

**Перевірила:**

**Солодовник Ганна Валеріївна**

**ХНЕУ ім. С. Кузнеця**

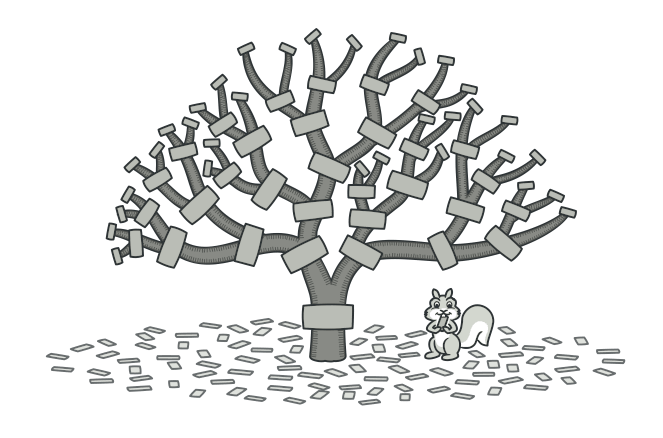
**2024**

**Завдання:**

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

**Опис патерну «Компанувальник»:**

**Компонувальник** — це структурний патерн проектування, що дає змогу згрупувати декілька об’єктів у деревоподібну структуру, а потім працювати з нею так, ніби це одиничний об’єкт.

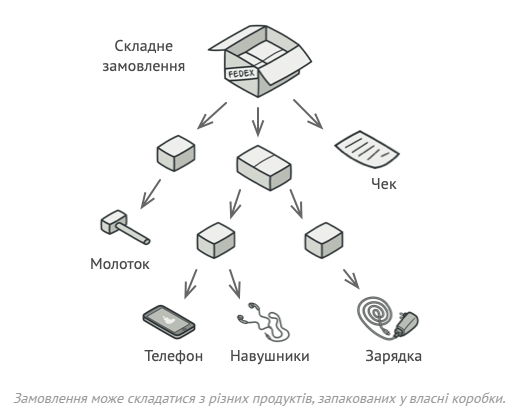


**Проблема**

Патерн Компонувальник має сенс тільки в тих випадках, коли основна модель вашої програми може бути структурована у вигляді дерева.

Наприклад, є два об’єкти — Продукт і Коробка. Коробка може містити кілька Продуктів та інших Коробок меншого розміру. Останні, в свою чергу, також містять або Продукти, або Коробки і так далі.

Тепер, припустімо, що ваші Продукти й Коробки можуть бути частиною замовлень. При цьому замовлення може містити як звичайні Продукт без пакування, так і наповнені змістом Коробки. Ваше завдання полягає в тому, щоб дізнатися вартість всього замовлення.



Якщо спробувати вирішити завдання напролом, тоді потрібно відкрити усі коробки замовлення, перебрати продукти й порахувати їхню загальну вартість. Але це занадто велика морока, оскільки типи коробок і їхній вміст можуть бути вам невідомі заздалегідь. Крім того, наперед невідомою є і кількість рівнів вкладеності коробок, тому перебрати коробки простим циклом не вийде.

## Рішення

Компонувальник пропонує розглядати Продукт і Коробку через єдиний інтерфейс зі спільним методом отримання ціни.

Продукт просто поверне свою вартість, а Коробка запитає про вартість кожного предмета всередині себе і поверне суму результатів. Якщо одним із внутрішніх предметів виявиться трохи менша коробка, вона теж буде перебирати власний вміст, і так далі, допоки не порахується вміст усіх складових частин.

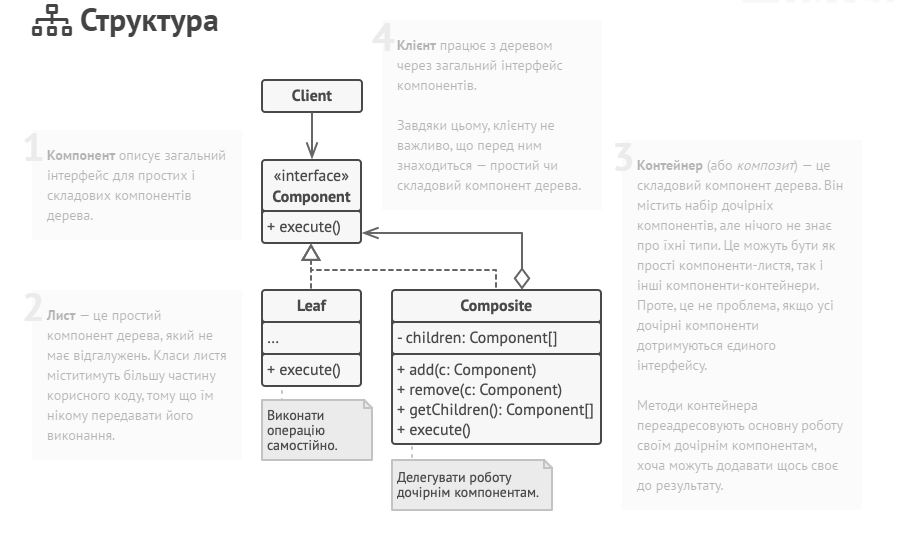


Для вас як клієнта важливим є те, що вже не потрібно нічого знати про структуру замовлень. Ви викликаєте метод отримання ціни, він повертає цифру, і ви не «тонете» в горах картону та скотчу.

## Аналогія з життя



Армії більшості країн можуть бути представлені у вигляді перевернутих дерев. На нижньому рівні у вас солдати, далі взводи, далі полки, а далі цілі армії. Накази віддаються зверху вниз структурою командування до тих пір, поки вони не доходять до конкретного солдата.



## Застосування

**Якщо вам потрібно представити деревоподібну структуру об’єктів.**

Патерн Компонувальник пропонує зберігати в складових об’єктах посилання на інші прості або складові об’єкти. Вони, у свою чергу, теж можуть зберігати свої вкладені об’єкти і так далі. У підсумку, ви можете будувати складну деревоподібну структуру даних, використовуючи всього два основних різновида об’єктів.

**Якщо клієнти повинні однаково трактувати прості та складові об’єкти.**

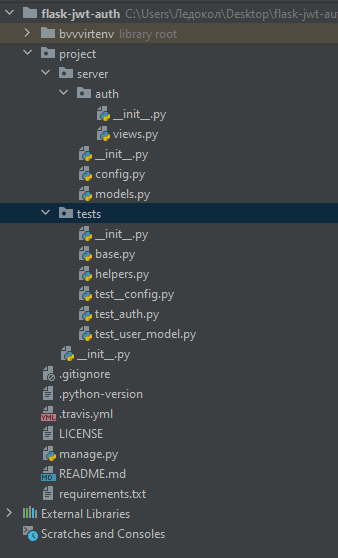
Завдяки тому, що прості та складові об’єкти реалізують спільний інтерфейс, клієнту байдуже, з яким саме об’єктом він працюватиме.

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

Для аналізу коду я обрав наступний код:   
<https://github.com/realpython/flask-jwt-auth>

Для більш зручного аналізу зроблю клон проекту

Спочатку пропоную подивитись на структуру проекту



Можна побачити, що головний файл - /manage.py та що від нього починається запуск проекту

Далі можна побачити папку /project, вона містить всю іншу частину проекту

В свою чергу папка містить ще дві /project/server – з кодом для автентифікації та /project/tests – з тестами для тестування того, як працює код

В свою чергу папка /project/server має всередині папку /project/server/auth з в’ю моделлю для автентифікації

Якщо дивитись більш детально по файлом тоді маємо наступне:

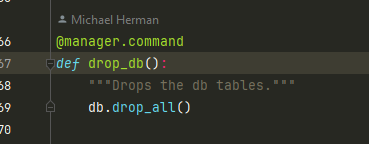
Для налаштування у файлі README.MD є інструкція

Для початку потрібно налаштувати змінні оточення   


Також потрібно створити бд у psql

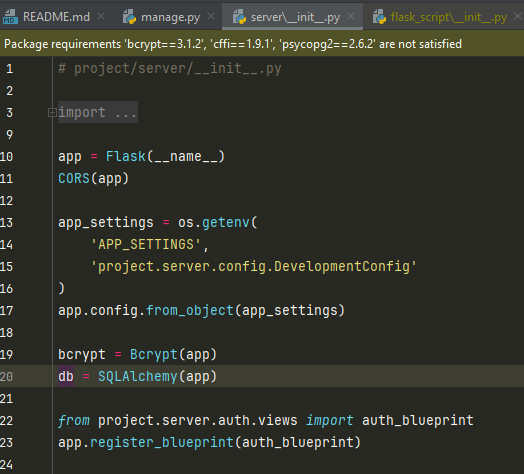
Та запустити відповідний код для налаштування   


Якщо подивитись на даний код зсередини, тоді можна побачити наступний код



Декоратор «manager.command» - надає інформацію, що дану команду можна запустити з терміналу   
  
в свою чергу db.drop\_all() це використання SQLAlchemy як систему з роботою з БД

Код з підключення до БД наступний:



Заглибнувшись більше у код можна зрозуміти, що файл /manage.py лише запускає проект, проте налаштування застосунку знаходиться у файлі /project/server/\_\_init\_\_.py

Також можна побачити, що налаштування беруться з змінних оточення

Також у кінці файлу можна побачити, що під’єднуються файли з автентифікації в’ю

В’ю знаходиться за наступним шляхом /project/server/auth/views.py

Якщо просто відкрити файл, тоді можна побачити, що файл всередині має 4 класи як окремі в’ю моделі, що вони всі зареєстровані як окремі моделі та що всі вони мають різні правила



Так для реєстрації є лише метод «POST» за шляхом /auth/register

Для входу в систему метод «POST» та шлях - /auth/login

Для отримання статусу – «GET» та шлях – /auth/status

Та для виходу «POST» та шлях - /auth/logout

Почну розглядати код з класу «RegisterAPI»

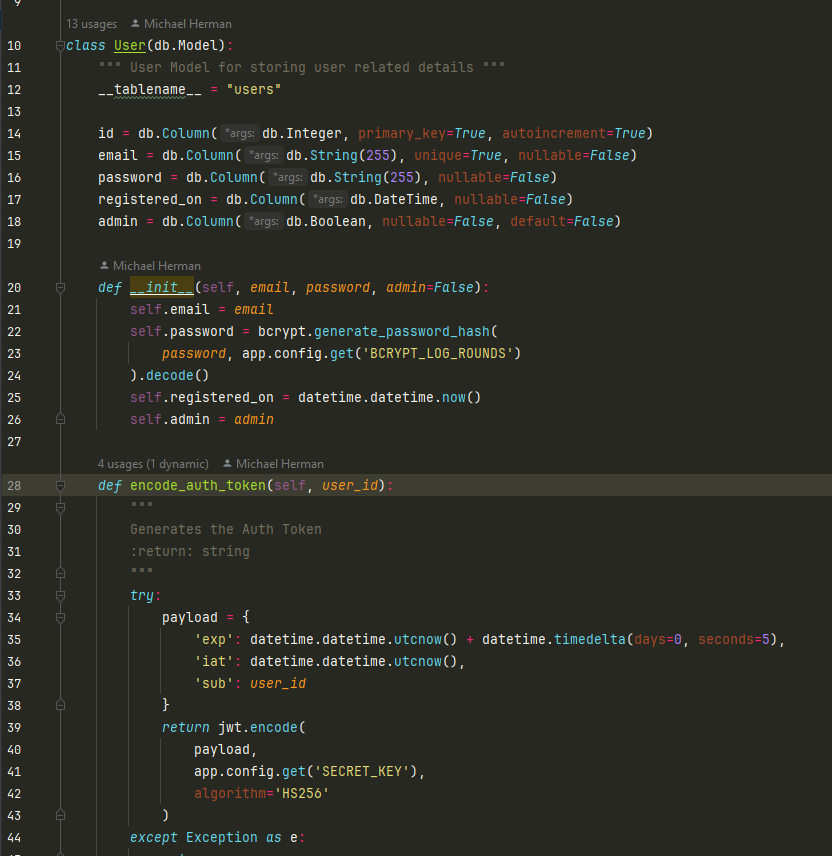


З опису одразу видно, що це просто метод, який всередині має лише код для обробки запиту типу «POST»

На 20-у рядку отримаємо дату з json пейлоаду

На 22-у намагаємось отримати користувача за емейлом

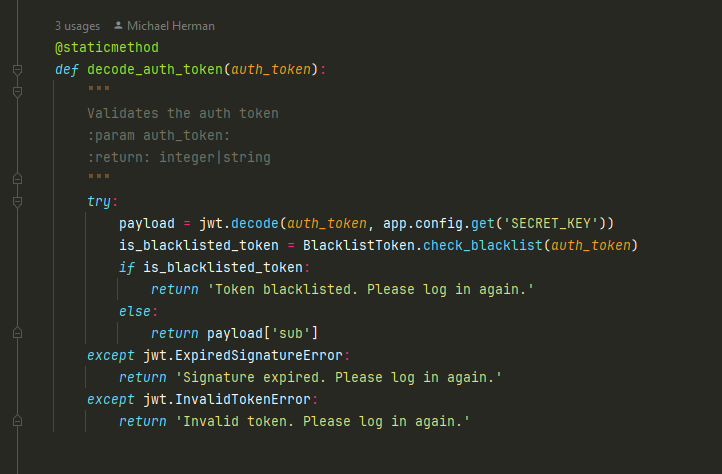
В свою чергу клас User – це модель у файлі /project/servr/models.py



Всередині цього класу можна побачити налаштування БД, ініціалізація нового користувача (рядок у БД)

Також є метод для шифрування ключа автентифікації відповідно до user\_id

Також у цьому класі є статичний метод для декодування токену



Всередині є перевірка токену на валідність та знаходження у чорному списку

В свою чергу чорний список це теж модель БД



Який всередині має лише налаштування для БД та первірку на наявність запису у таблиці

Тепер повернемось до нашої в’ю моделі

Якщо користувача буде знайдено, тоді в респонсі буде видано, що такий користувач вже існує, а якщо користувача не має, тоді буде спроба створити нового користувача, створивши користувача на 25-й строчці, відповідно отримавши дані з пейлоаду, до речі всередині класу User пароль хешується, а id надається автоматично та є ключовим полем у таблиці

Наступним кроком запис додається до таблиці на строчці 30

Та зберігається на строчці 31

Після чого буде згенеровано токен автентифікації та наданий у респонсі

Якщо в процесі виконання даного коду відбудеться помилка, тоді користувачу буде повернута помилка у відповіді від сервера

Тепер розглянемо наступний клас з логуванням користувачів



Знову це в’ю з методом, який має лише обробку запитів типу «POST»

Одразу отримаємо дані з json-у, який надішле користувач,

Наступним кроком на 63-й стрічці перевіримо, що користувач з таким емейлом існує, та якщо не існує – повертаємо помилку на 82-у рядку,  
а якщо існує, тоді перевіряємо, що хеш пароля у бд та хеш з пароля, який ми отримали у пейлоаді однакові, якщо так – генеруємо токен на 69-у рядку

Також, якщо в процесі виникла будь-яка помилка – користувачу буде повернуто помилку, та напис, що потрібно спробувати ще раз

Тепер перейдемо до класу UserAPI



Він так само має, лише обробку одного типу запиту, але це «GET» запит

Тут перевірка відбувається за допомогою хедеру, так на 98-й строчці ми намагаємось, отримати хедер, та отримати значення, якщо хедеру не має, або його тип буде не вірний – користувач отримає помилку,

Але, якщо токен існує, тоді він буде декодований та у разі його валідності – дані будуть повернуті користувачу

Та останній клас LogoutAPI



Код схожий на попередній клас, єдина різниця, що після декодування токену – створюється запис до БД, що даний токен знаходиться у чорному списку та у разі успішного запису – повертається код з успіхом, проте, якщо виникне помилка на будь якому з етапів – користувачу буде повернута помилка