УДК 004.43

О. І. Вальчишен
І. В. Богач

АНАЛІЗ ПІДХОДІВ ДО РОЗРОБКИ БЕКЕНДУ ВЕБЗАСТОСУНКІВ ІЗ ВИКОРИСТАННЯМ МОВИ ПРОГРАМУВАННЯ PYTHON

Вінницький національний технічний університет

Анотація

У роботі розглянуто використання мови програмування Python для розробки серверної частини (бекенду) вебсайтів. Проаналізовано ключові переваги Python, а також популярні фреймворки, такі як Django та Flask. Оцінено їх гнучкість, швидкість розробки та відповідність сучасним вимогам до створення надійних і масштабованих вебзастосунків.

Ключові слова: Python, бекенд, веброзробка, Django, Flask, серверна частина.

Abstract

This paper examines the use of the Python programming language for backend web application development. The key advantages of Python, as well as popular frameworks such as Django and Flask, are analyzed. Their flexibility, development speed, and suitability for meeting modern requirements for building reliable and scalable web applications are evaluated.

Keywords: Python, backend, web development, Django, Flask, server-side.

Вступ

В умовах стрімкого розвитку інтернет-технологій вибір правильного інструменту для розробки серверної частини (бекенду) вебзастосунків є критично важливим етапом. Серед багатьох мов програмування Python займає одну з провідних позицій завдяки своїй виразності, читабельності коду та потужній екосистемі. Він широко використовується як для створення невеликих вебсайтів та API, так і для розробки високонавантажених корпоративних рішень.

У роботі проаналізовано ключові особливості використання Python як бекенд-мови, розглянуто підходи до створення серверної архітектури за допомогою популярних фреймворків та визначено їхні переваги для сучасних розробників.

Результати дослідження

Python — це високорівнева мова програмування загального призначення, яка завдяки своєму лаконічному синтаксису дозволяє розробникам зосередитись на реалізації бізнес-логіки проєкту, мінімізуючи час на написання шаблонного коду. Для веброзробки на Python найчастіше використовують спеціалізовані фреймворки, які беруть на себе маршрутизацію (роутинг), взаємодію з базами даних та забезпечення безпеки застосунку. [1]

Двома найпопулярнішими підходами до веброзробки на Python є використання повноцінних фреймворків (наприклад, Django) та мікрофреймворків (наприклад, Flask).

Django — це високорівневий фреймворк, який слідує принципу "включено все" (batteries-included). Він "з коробки" надає готові рішення для автентифікації користувачів, панелі адміністратора, ORM (Object-Relational Mapping) для безпечної роботи з базами даних тощо. Це робить його ідеальним вибором для великих монолітних проєктів зі складною структурою. [2]

Flask, навпаки, є легким мікрофреймворком. Він надає лише базовий набір інструментів для запуску сервера та обробки запитів, залишаючи розробнику абсолютну свободу вибору сторонніх бібліотек для роботи з даними, формами чи сесіями. Завдяки цій гнучкості, Flask чудово підходить для розробки невеликих вебзастосунків, мікросервісів або RESTful API, де потрібен максимальний контроль над кожним компонентом архітектури. [3]

Наочним прикладом простоти та виразності Python є реалізація базового маршруту на Flask. На рисунку 1 зображено фрагмент коду, який приймає запит від користувача та повертає відрендерену вебсторінку, займаючи всього кілька рядків.

```python
from flask import *
import sqlite3

app = Flask(__name__)



@app.route("/")
def main():
    return render_template("main.html")

@app.route("/products")
def products():
    return render_template("products.html")

@app.route("/about_page")
def about():
    return render_template("about_page.html")

@app.route("/contacts")
def contacts():
    return render_template("contacts.html")


@app.route("/delivery")
def delivery():
    return render_template("delivery.html")

@app.route("/julia_page")
def julia_page():
    return render_template("metalcherp/julia_page.html")

@app.route("/duna_lux_main")
def duna_lux_main():
    return render_template("metalcherp/duna_lux_main.html")
```

Рис. 1. Реалізація базової маршрутизації за допомогою фреймворку Flask

Окрім безпосереднього написання коду, розробка серверної частини передбачає проєктування логіки взаємодії між клієнтом та базою даних. На рисунку 2 наведено приклад UML-діаграми, яка ілюструє архітектуру обробки запиту сервером на базі Python.

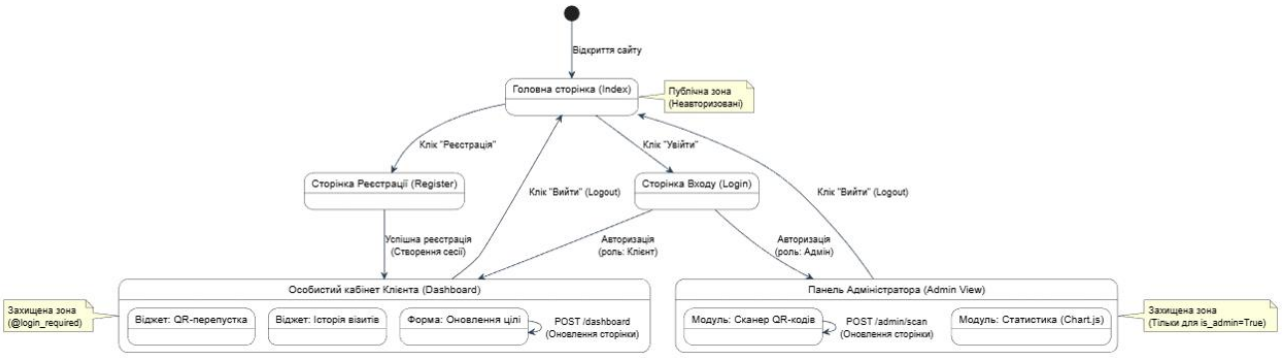

Рис. 2. UML-діаграма логіки взаємодії клієнтської та серверної частин вебзастосунку

Загалом, можна виокремити наступні переваги використання Python для розробки бекенду:

1. Висока швидкість розробки (Time-to-Market): завдяки зрозумілому синтаксису та величезній кількості готових модулів час від ідеї до готового продукту значно скорочується.

2. Велика спільнота та багата екосистема: пакетний менеджер (pip) надає доступ до тисяч бібліотек, що спрощує інтеграцію зі сторонніми сервісами, обробку платежів та роботу з алгоритмами штучного інтелекту.

3. Читабельність та підтримка коду: чистий синтаксис Python полегшує підтримку та масштабування проєкту в майбутньому, що особливо важливо при командній розробці.

4. Універсальність: мова дозволяє легко інтегрувати вебзастосунок із скриптами для аналізу даних, машинного навчання або IoT-пристроїв у межах єдиної екосистеми.

Серед недоліків традиційно виділяють дещо нижчу швидкість виконання в порівнянні з компільованими мовами (такими як Go або C++). Однак для більшості вебзастосунків вузьким місцем є не швидкість самої мови, а час відповіді бази даних або мережеві затримки, що успішно вирішується правильним проєктуванням архітектури та кешуванням.

Висновки

Python є потужним, надійним та надзвичайно гнучким інструментом для розробки бекенду вебсайтів. Вибір конкретного фреймворку залежить від масштабів та специфіки проєкту: Django чудово підходить для великих, комплексних платформ зі стандартизованою структурою, тоді як Flask є оптимальним вибором для гнучких вебзастосунків, де розробнику потрібен повний контроль над інструментарієм. Завдяки своїй продуктивності, багатій екосистемі та простоті підтримки коду, Python впевнено утримує лідерські позиції у сфері сучасної веброзробки.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Python Software Foundation. Python 3 Documentation. [Електронний ресурс] – Режим доступу: https://docs.python.org/3/ (дата звернення: 21.03.2026).
2. Django Software Foundation. Django Documentation. [Електронний ресурс] – Режим доступу: https://docs.djangoproject.com/ (дата звернення: 21.03.2026).
3. Pallets. Flask Documentation. [Електронний ресурс] – Режим доступу: https://flask.palletsprojects.com/ (дата звернення: 21.03.2026).

Вальчишен Олександр Ігорович – студент групи 1ІСТ-22б, кафедра автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м. Вінниця, e-mail: valchishensasha12@gmail.com

Богач Ілона Віталіївна – к.т.н., професор кафедри автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м. Вінниця, e-mail: ilona.bogach@gmail.com.

Valchyshen Oleksandr Ihorovych – student of 1IST-22b group, Department of Automation and Intelligent Information Technologies, Faculty of Intelligent Information Technology and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: valchishensasha12@gmail.com

Bogach Ilona Vitaliivna – PhD, Professor at the Department of Automation and Intelligent Information Technologies, Faculty of Intelligent Information Technology and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: ilona.bogach@gmail.com.