УДК 519.688

DOI 10.31651/2076-5886-2022-1-73-83

PACS 02.60.-x, 02.60.Pn

ІГНАТЕНКО Ігор Олександрович
Студент Черкаського національного університету ім. Б. Хмельницького
e-mail: igignt@gmail.com

СЕРДЮК Олександр Анатолійович,
кандидат економічних наук, доцент, доцент кафедри прикладної математики та інформатики Черкаського національного університету імені БогданаХмельницького
e-mail: serdyuk@ukr.net
ORCID 0000-0002-3919-4661

ДІДКОВСЬКИЙ Руслан Михайлович,
доктор технічних наук, доцент, доцент кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: didkovskyirm@vu.cdu.edu.ua
ORCID 0000-0002-5166-7564

РОЗРОБКА ОСВІТНЬОГО ПОРТАЛУ ДЛЯ ПЕРЕВІРКИ КОМП'ЮТЕРНИХ ПРОГРАМ, НАПИСАНИХ СТУДЕНТАМИ

У роботі розглянуто методи та засоби створення повноцінного вебсайту з базою даних, клієнтською та серверною частиною. Наразі онлайн навчання широко використовується в навчальних закладах та у сфері IT. Coursera, EDX, Udemy, Udacity є провідними ресурсами онлайн навчання у світі. Варто також згадати український проект масових відкритих онлайн курсів Prometheus та освітню платформу Stepik, які є дуже популярними в Україні. На даних платформах використовують різні засоби перевірки домашнього завдання. Для контролю теоретичних знань застосовують традиційні методи, переважно тестові задачі. Якщо це стосується коду, то перевірка, а саме реалізація цього процесу є складним, ресурсомістким процесом та водночас цікавим.

Метою роботи є створення вебсервісу для розміщення та перевірки практичних завдань, а саме завдань, де потрібно реалізувати певну програму. Також метою мого проекту є розробка алгоритму перевірки коду, який буде реалізовано шляхом прогону деяких вхідних даних та порівняння їх з попередньо заданими вихідними даними.

Отримане програмне забезпечення може бути використане будь-де, тому що створене воно у вигляді програмного пакета, який можна з легкістю розгорнути на сервері чи персональній машині. Безпосередньо програма може бути використана для перевірки робіт студентів в моєму навчальному закладі.

Ключові слова: *веб додаток, мова розмітки сайтів html, стилі css, фреймворк flask, мова програмування python, база даних.*

Вступ

Що, взагалі таке тестування? Тестування - це дослідження, призначений для виявлення інформації про якість роботи продукту відносно контексту, в якому він має використовуватись. Техніка тестування також включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою оцінки. [1]

Моя задача полягає в тому, щоб написати додаток, що буде перевіряти невеликі програми на правильність їх роботи. Що полягає під словом “правильність”? В моєму випадку, це позитивне проходження тестів та коректне завершення роботи програми. Сам тест реалізує прогін деяких вхідних даних через програму, та порівняння вихідних даних з попередньо підготовленими правильними вихідними даними.

Реалізувати у вигляді графічного інтерфейсу чи вебсайту? На мою думку, краще використати другий варіант. Ось декілька причин чому:

– легкий доступ до вебдодатку. В цьому випадку користувачу потрібен лише комп'ютер з браузером і з'єднання з інтернетом;
– оновлення додатку. Для того, щоб оновити вебдодаток, його необхідно оновити його тільки на сервері та всі користувачі відразу можуть працювати з новою версією;
– вебдодатки більш універсальні та практичні для кінцевого користувача. Вам достатньо буде встановити вебдодаток на сервер, що працює під будь-якою сучасної ОС, і ви зможете користуватися ним через інтернет на будь-якому комп'ютері або мобільному пристрої;

Дивлячись на ринок, можна сказати, що вебдодатки давно обігнали звичайні графічні програми. Вони легші в розробці, розгортанні на сервері, вони гнучкіші, та більш доступні. Тому я не бачу сенсу в GUI, принаймні при реалізації мого програмного забезпечення.

Стек технологій

Перше що потрібно обрати для створення будь-якого додатку, це мова програмування. Я зупинився на Python 3.8, тому що вона має досить цікаві фреймворки для створення веб сервісів, які варто дослідити. Основні характеристики:

– інтерпретована мова програмування високого рівня;
– динамічна типізація;
– лаконічний синтаксис;
– мобільність програм;
– популярність, і як наслідок - велика спільнота розробників.

Недоліки:

– низька швидкодія, що характерно для всіх інтерпретованих мов;
– відсутність статичної типізації, що також уповільнює роботу програм.

Наступним завданням є обрання фреймворку для створення серверної частини. Існує два найпопулярніших кандидати: Flask та Django. Перший - легкий і гнучкий фреймворк, в якому не встановлено нічого зайвого. Це дозволяє вибирати модулі для вирішення конкретних завдань і встановлювати їх за потреби. Django - надає пакет «все включено»: у вас є панель адміністратора, інтерфейси баз даних, ORM, і структура каталогів для ваших додатків і проектів.

Flask частіше використовують коли потрібно більше контролю над компонентами додатку, або створення чогось нетипового. Django, якщо вас цікавить кінцевий продукт. Особливо якщо потрібно працювати з прямолінійним додатком, таким як сайт з новинами, онлайн магазин чи блог. [2] Тому я зупинився на першому варіанті - Flask.

Надалі при безпосередній розробці програмного продукту знадобляться ще декілька модулів та бібліотек для реалізації необхідних можливостей додатку. Серед яких:

– wtforms - для реалізації функціонування вебформ;
– subprocess - для запуску зовнішніх програм;
– flask_sqlalchemy - для роботи з базою даних;

– flask_admin - для управління базою даних;
– flask_bootstrap - для інтеграції з Bootstrap4;
– os, shutil - модулі для системних команд та налаштувань.

Для розробки клієнтської частини програми використаю мову розмітки HTML 5 та мову таблицю стилів CSS, та набір інструментів Bootstrap4, що містить HTML і CSS-шаблони оформлення для типографій, вебформ, кнопок, міток, блоків навігації та інших компонентів вебінтерфейсу, включаючи JavaScript-розширення.

При створенні розмітки буду використовувати шаблонизатор Jinja2. Це рушій шаблонів для мови програмування Python створений Арміном Ронакером з ліцензією BSD. На відміну від схожого рушія шаблонів у Django, Jinja використовує вирази у стилі мови Python, тому процес розмітки документу стає подібним до написання звичайного коду.

Засоби реалізації

Для розгортання та створення проекту я використаю нову розробку Microsoft - WSL. Windows Subsystem for Linux - це прошарок між ядром Windows і додатками для Linux, який дозволяє перетворювати системні виклики до ядра Linux в виклики до ядра Windows. [3] Завдяки тому, що віртуалізація практично відсутня, таке рішення працює швидше традиційної віртуалізації, де повністю емулюється комп'ютер, як це відбувається в Oracle VirtualBox і VMWare Player.

Для розробника WSL - це зручне середовище розробки. Все ж встановлення багатьох мов, компіляторів та інтерпретаторів, утиліт в Linux відбувається куди простіше.

Також при створенні я буду використовувати VS Code - зручний редактор коду з безліч розширеннями, Git система контролю версій, та Github для збереження мого додатку.

Свою програму буду писати у вигляді пакету. Це дозволяє надалі спрости запуск програми, її повторне використання, та розгортання на сервері. Файл з розширенням '.py' - це модуль. В ньому можна створити в ньому класи, функції та змінні. Якщо в директорію покласти кілька модулів і створити файл '__init__.py', створиться пакет з ім'ям даної директорії.

В наступних розділах я повністю опишу процес реалізації цього додатку та протестую його на певних даних.

Макет проекту

Файлова ієрархія проекту буде виглядати наступним чином (рис. 1):

– CodeChecker/ - основна директорія додатку
– .venv/ - директорія в якій знаходяться файли віртуального середовища Python.
– app/ - безпосередньо пакет нашого додатку
– static/ - директорія для статичних файлів, зокрема файлів стилю css
– templates/ - директорія для файлів розмітки html
– temp/ - директорія тимчасових файлів
– app.db - база даних
– config.py – код налаштувань
– run.py – програма запуску сервісу

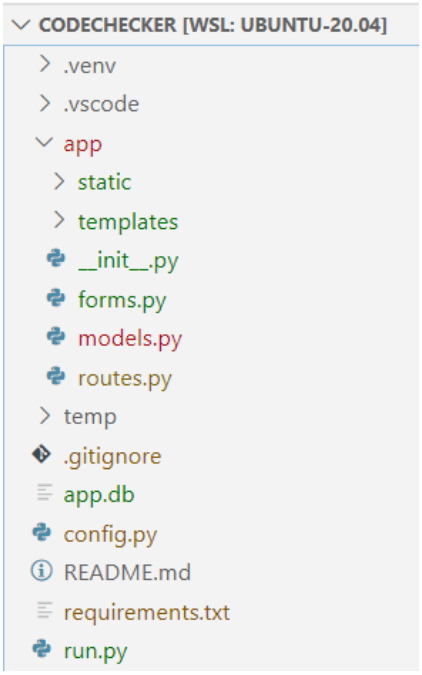

Рис. 1

Створення та стилізація вебсторінки

Наступним кроком є створення HTML сторінок та їх CSS оформлення за допомогою BootStrap4. Створю в теці templates/ 4 файли:

– base.html - базовий шаблон, що містить заголовок, навігацію, та так званий "підвал" сайту;
– main.html – головна сторінка, що містить список задач. Наслідується від сторінки base за допомогою можливостей Jinja;
– result.html - сторінка, на яку будуть передаватися результати тестування;
– task.html – сторінка з формою для вводу розв'язання певної задачі.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1, shrink-to-fit=no">
    <link rel="shortcut icon" href="../static/code.ico">
    {{ bootstrap.load_css() }}
    <title>CodeChecker</title>
</head>
```

В лістингу вище наведено "голову" сторінки base.html, де вказано основну інформацію: версія HTML, кодування, розташування файлу стилів, титулка.

```
<body>
    <div class="container">
        <header>
            <nav class="navbar navbar-default navbar-dark bg-
primary rounded mt-3 mb-3">
                <div>
```

```
                    <a class="navbar-
brand" href="{{ url_for('mainPage') }}">
                        <img src="static/code.svg" width="30" he
ight="30" hspace="10" class="d-inline-block align-
middle" alt="">
                    <strong class="text-light align-
middle">Cody</strong>
                    </a>
                </div>
                <div>
                    <a class="text-
light" href="https://github.com/Ihoriam">Github</a>
                </div>

            </nav>
        </header>
        {% block content %}
        {% endblock %}
```

В цій частині HTML розмітки файлу base.html наведена одна з цікавих можливостей Jinja - наслідування. {% block content %}{% endblock %}

Я використав оператор управління блоком, щоб визначити місце, де можна ставити похідні шаблони. Блокам присвоюється унікальне ім'я, на яке похідні шаблони можуть посилатися.

Тепер я можу написати main.html, наслідуючи його від base.html, що є хорошою практикою в програмуванні.

```
{% extends "base.html" %}
{% block content %}
<main role="main">
  <section class="jumbotron text-center mb-3">
    <div class="container">
      <h1 class="jumbotron-heading">Вітаю!</h1>
      <p class="lead text-
muted">Ви піймались! Даний сервіс створено для перевірки Вашого
домашнього завдання з програмування!</p>
      <p class="text-black-
50">Тепер не вийде аби як скинути пусту папку з домашкою на пошт
у!</p>
      <p>
        <a href="#" class="btn btn-primary my-2">До роботи</a>
      </p>
    </div>
  </section>
...
{% endblock %}
```

Цікавість цього коду полягає в використанні готових html класів, реалізованих в наборі інструментів Bootstrap4. [4] Це дозволяє економити час на створення CSS стилів власноруч.

Далі я створи ще два html документа task.html та result.html. Шаблони також підтримують керуючі оператори, їх задають в {% ...%}. [2] В даному сценарії я використав оператор умовного блока {% if flag %}.

```
<div class="row bg mt-3 mb-3 rounded">
        <div class="col text-center">
          {% if flag %}
            <h3 class="text-success">Успішно виконано</h3>
            {% else %}
            <h3 class="text-danger">Ой, десь помилка</h3>
            {% endif %}
        </div>
    </div>
```

Ось так виглядає основна сторінка сайту (рис. 2).

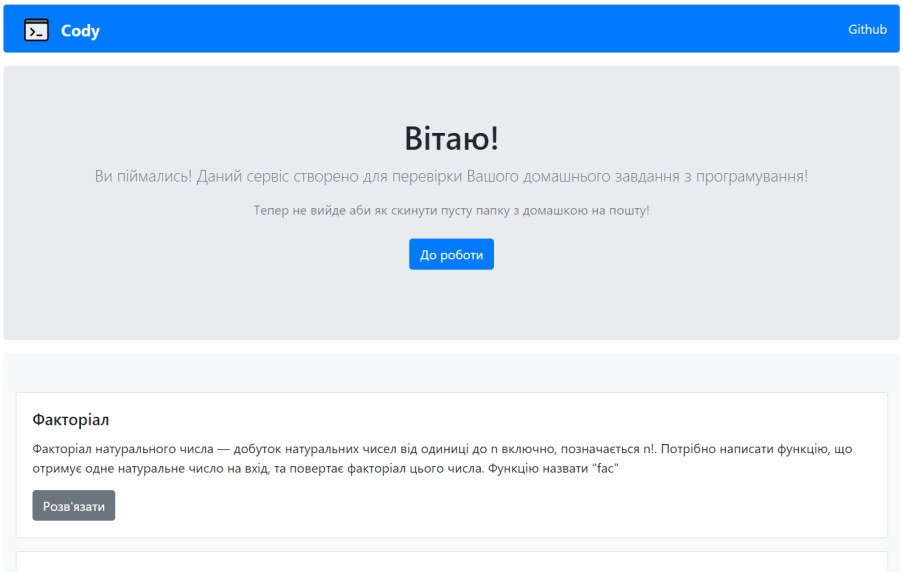

Рис. 2

Ініціалізація пакету

В директорії app розміщений файл з іменем__init__.py, в якому записано наступний код:

```
from flask import Flask
from config import Config
from flask_sqlalchemy import SQLAlchemy
from flask_admin import Admin
from flask_bootstrap import Bootstrap
app = Flask(__name__)
app.config.from_object(Config)
bootstrap = Bootstrap(app)
db = SQLAlchemy(app)
admin = Admin(app, name='cody', template_mode='bootstrap3')
from app import routes, models
app.run(debug=True)
```

Сценарій вище імпортує основні модулі для функціювання додатку. Серед яких модулі для роботи з базою даних, з формами та безпосередньо модуль Flask. А також створює екземпляр класу Flask.

Реалізація вебформ

Щоб отримувати певні дані від користувача, на сторінках використовують форми, а саме тег <form>. Щоб зв'язати HTML форму та Python код, я використаю розширення flask-wtf, яке я встановив раніше.

Створю в директорії checky/ файл з назвою forms.py:

```
from flask_wtf import FlaskForm
from wtforms import TextAreaField, SubmitField, SelectField
from flask_codemirror.fields import CodeMirrorField
from wtforms.validators import DataRequired
class CheckerForm(FlaskForm):
    codeField = TextAreaField("Ваш код", render_kw={"rows": 15,
"cols": 10}, validators=[DataRequired()])
    selectLang = SelectField('', choices=[('gcc', 'C'), ('g++',
'C++'), ('py', 'Python'),
('java', 'Java'), ('js', 'JavaScript')])
    submit = SubmitField('Перевірити')
```

В перших рядках я імпортую клас FlaskForm, та декілька функцій. Далі створюю екземпляр класу, в полях якого ініціалізую змінні, як один з типів форм. В моєму випадку це TextAreaField, тобто текстове поле, в яке користувач буде вводити дані (рис. 3).

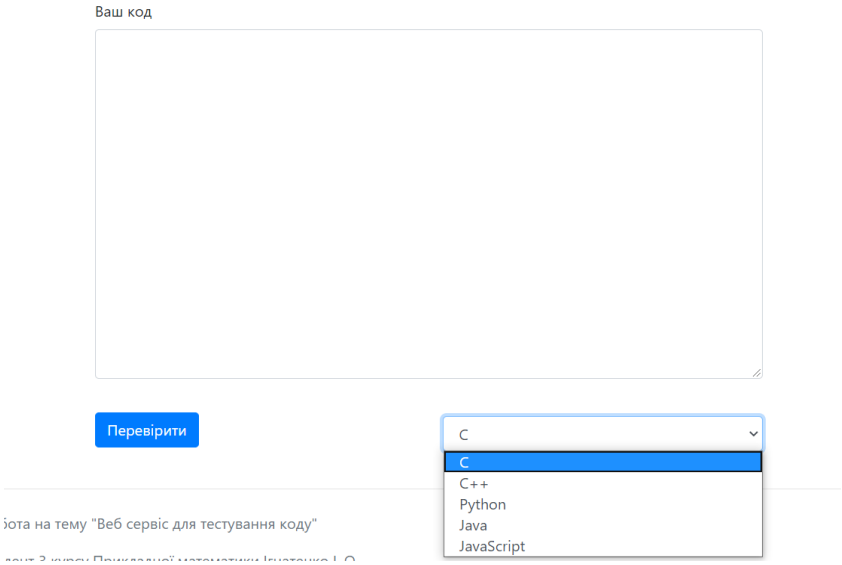

Рис. 3

Для роботи розширення flask-wtf необхідно задати секретний ключ. Для цього створю файл з налаштуваннями в теці CodeChecker/ :

```
import os
class Config(object):
SECRET_KEY = os.environ.get('SECRET_KEY') or 'you-will-never-
guess
```

Flask і деякі його розширення використовують значення SECRET_KEY як криптографічний ключ для генерації підписів або токенів. Розширення Flask-WTF використовує його для захисту форм від вебатаки під назвою Cross-Site Request Forgery або CSRF (вимовляється як «seasurf»). [2]

База даних

Для повноцінного функціювання сайту необхідна база даних, де будуть зберігатися задачі, частини коду, тести, та ще певна інформація. Для цього створюю скрипт models.py.

```
from app import db
from app import admin
from flask_admin.contrib.sqla import ModelView

class Code (db.Model):
    id = db.Column(db.Integer(), primary_key=True)
    id_task = db.Column(db.Integer(), db.ForeignKey('task.id'))
    id_lang = db.Column(db.Integer(), db.ForeignKey('language.id
'))
    pre_code = db.Column(db.Text())
    main_code = db.Column(db.Text(), nullable=False)
    post_code = db.Column(db.Text())
    tests = db.relationship('Test', backref='code', lazy=True)
```

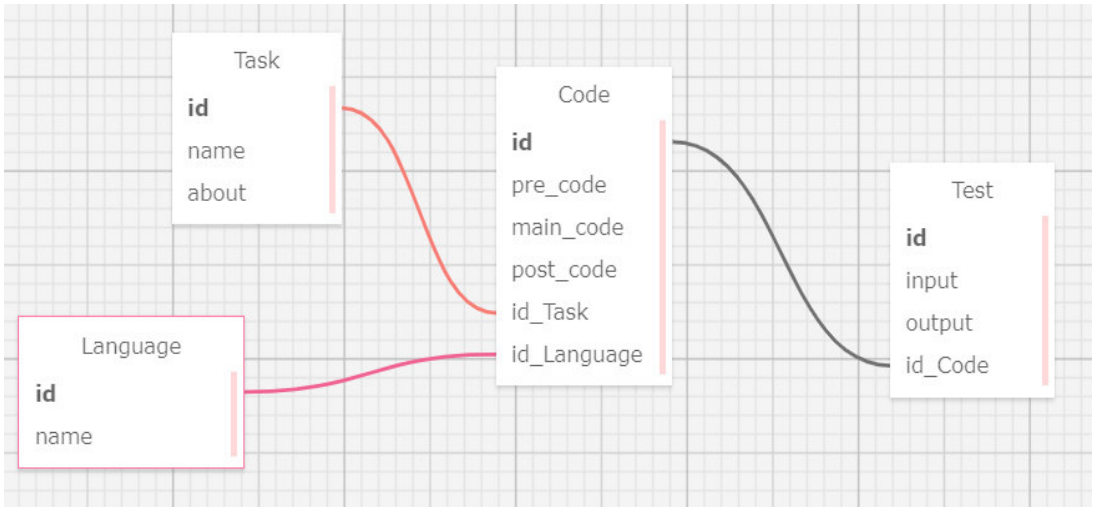

Рис. 4

В даній частині коду наведеній вище реалізована одна з таблиць бази даних. На малюнку вище продемонстрована діаграма всієї бази даних (рис. 4).

Алгоритм перевірки

В цьому розділі я буду працювати з файлом routes.py. Це основний скрипт мого вебдодатку, що містить в собі логіку URL сторінок, та алгоритм тестування коду, введеного користувачем. Алгоритм працює на таких мовах як Python, C, C++, Java та JavaScript. Нижче я наведу алгоритм перевірки для мови Python, для інших мов він схожий.

```
@app.route('/task/<int:id_task>', methods=['GET', 'POST'])
def taskPage(id_task):
    form = CheckerForm()
    task = Task.query.filter_by(id = id_task).first()
    if request.method == 'POST' and form.validate_on_submit():
        # print(form.selectLang.data)
        if form.selectLang.data == 'py':
            code = Code.query.filter_by(id_task=id_task, id_lang
=3).first()
```

```
            tests = Test.query.filter_by(id_code=code.id).all()
            num_tests = len(tests)
            count = 0
            with open("temp/py/sub.py", "w") as sub:
                sub.writelines([code.pre_code, '\n', form.codeFi
eld.data, '\n', code.post_code])
            for test in tests:
                # print(test.input_data)
                process = subprocess.run(['python3', 'temp/py/su
b.py', *test.input_data.split()],
                                         check=False,
                                         capture_output=True,
                                         universal_newlines=True)

                if (process.stdout) and process.stdout.strip() =
= test.output_data.strip():
                    count += 1
                    # print(count)
                else:
                    break
```

1) програма отримує з бази даних інформацію та тести на конкретно вибрану задачу;
2) оброблює форму з кодом, що заповнив користувач;
3) в циклі, програма проганяє тести через subprocess.run() [5], що запускає на фоні програму користувача з різними вхідними даними;
4) потім додаток отримує вихідні дані та звіряє їх з правильними.

```
if 'post_process' in locals():
            error = post_process.stderr
        else:
            error = process.stderr

        if count == num_tests:
            flag_valid=True
        else:
            flag_valid=False

        return render_template('result.html',
                               code=form.codeField.data,
                               propose_code = code.main_code,
                               error=error,
                               flag=flag_valid,
                               count=count,
                               num_tests=num_tests
                               )
```

Далі програма перевіряє кількість коректно пройдених тестів, та використовую функцію render_template, що генерує сторінку result.html та передає дані до неї.

Адміністрування даних

В проекті також створена сторінка адміністратора за допомогою додатку flask-admin. В ній реалізовані 4 базові функції управління даними «створення, зчитування, зміна і видалення» CRUD [6] (рис. 5).

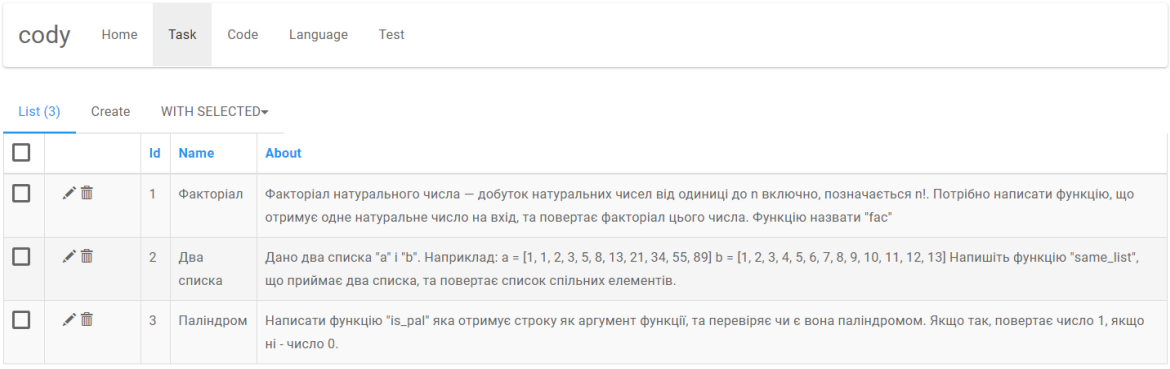

Рис. 5

Висновок

В результаті завершення цього комплексного проекту, я використав такі технології як: мова Python 3.8, фреймворк Flask, форми, база даних, wsl, шаблонизатор Jinja2, мову розмітки HTML, таблицю стилів CSS, Bootstrap4 та ще декілька модулів Python.

Поєднавши цей стек технологій, я створив цікавий додаток, з серверною та клієнтською частиною. Також я покращив свої навички в роботі з базою даних та у використанні командного термінала Linux.

На мою думку, проект цікавий як з теоретичної, так і з практичної точки зору. В майбутньому його можна використати, як основу чи як мікросервіс для більш масштабного проекту. Наприклад – для нашого кафедрального сайту, де студенти можуть дистанційно опрацьовувати матеріал. Можу сказати, що вже маю декілька ідей для його вдосконалення:

– особистий кабінет;
– підсвітка синтаксису;
– більше мов для проходження завдань;
– підвищення швидкодії за допомогою паралельного програмування;
– більш детальна система сповіщення про помилки.

Список використаної літератури:

1. Криспин Л., Грегори Д. / Гибкое тестирование. Практическое руководство для тестировщиков ПО и гибких команд - М.: Вильямс, 2010. - 464 с.
2. Miguel Grinberg / Flask Web Development: Developing Web Applications with Python 2nd Edition - Publisher(s): O'Reilly Media, Inc.
3. Microsoft blog [Електронний ресурс] // Learn About Windows Console & Windows Subsystem For Linux (WSL) - Режим доступу: https://devblogs.microsoft.com/commandline/learn-about-windows-console-and-windows-subsystem-for-linux-wsl/
4. w3schools.com // Bootstrap 4 Tutorial - Режим доступу: https://www.w3schools.com/bootstrap4/
5. Python 3 documentation // subprocess - Subprocess management - Режим доступу: https://docs.python.org/3/library/subprocess.html
6. Осипов Д. Л. / Технологии проектирования баз данных. - М.: ДМК Пресс, 2019. – 498 с.

References:

1. Crispin L., Gregory D. / Flexible testing. A practical guide for software testers and flexible teams - M .: Williams, 2010. - 464 p.
2. Miguel Grinberg / Flask Web Development: Developing Web Applications with Python 2nd Edition - Publisher(s): O'Reilly Media, Inc.

3. Microsoft blog [Електронний ресурс] // Learn About Windows Console & Windows Subsystem For Linux (WSL) - Access mode: https://devblogs.microsoft.com/commandline/learn-about-windows-console-and-windows-subsystem-for-linux-wsl/
4. w3schools.com // Bootstrap 4 Tutorial - Access mode: https://www.w3schools.com/bootstrap4/
5. Python 3 documentation // subprocess - Subprocess management - Access mode: https://docs.python.org/3/library/subprocess.html
6. Osipov DL / Database design technologies. - M .: DMK Press, 2019 .- 498 p.

IHNATENKO Ihor,
Student, Department of Informatics and Applied Mathematics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
SERDIUK Oleksandr,
Candidate of Economic Sciences, Associate Professor, Department of Informatics and Applied Mathematics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
DIDKOWSKY Ruslan,
Doctor of Technical Sciences, Associate Professor, Department of Informatics and Applied Mathematics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine

DEVELOPMENT OF AN EDUCATIONAL PORTAL FOR VERIFICATION OF COMPUTER PROGRAMS WRITTEN BY STUDENTS

The Bohdan Khmelnytsky National University of Cherkasy, student

Summary. Introduction. *Currently, online learning is widely used in schools and in the field of IT. Coursera, EDX, Udemy, Udacity are the world's leading online learning resources. It is also worth mentioning the Ukrainian project of mass open online courses Prometheus and the educational platform Stepik, which are very popular in Ukraine. These platforms use various way to check homework. Traditional methods are used to control theoretical knowledge. But when we need to verify code, this process becomes complex, resource-intensive and at the same time interesting to solve .*

Purpose. *The purpose of the course work is to create a web service for posting and testing practical tasks, namely tasks where you want to implement a program. Also, the goal of my project is to develop a method of verifying the code, which will be implemented by running some input data and comparing them with predefined output data.*

Results. *The resulting software can be used anywhere, because it is created in the form of a software package that can be easily deployed on a server or personal machine. The program can be used directly to check the work of students in my university.*

Conclusion. *As a result of completing this complex project, I used technologies such as Python, Flask framework, forms, database, wsl, Jinja2 template, HTML markup language, CSS stylesheet, Bootstrap4 and several other Python modules.*

Combining this technology stack, I created an interesting application, with a server and client part. I also improved my skills in working with the database and using the Linux command terminal.

In my opinion, the project is interesting both from a theoretical and a practical point of view. In the future, it can be used as a basis or as a micro-service for a larger project. For example - for our cathedral site, where students can remotely process the material.

Keywords: *web application, html site markup language, css styles, flask framework, python programming language, database.*

Одержано редакцією 17.12.2022 р.
Прийнято до публікації 23.06.2022 р.