doi: 10.32620/oikit.2025.106.0

УДК 37.018.43:004.4:33

Т. Л. Столяренко

Архітектура інформаційної системи підтримки дистанційного навчання програмуванню на Python в глобальному середовищі економічної освіти

Харківський національний економічний університет імені Семена Кузнеця

У статті запропоновано архітектуру інформаційної системи (ІС), розроблену для подолання обмежень традиційних систем управління навчанням (LMS). Запропонована архітектура базується на мікросервісному підході, що забезпечує високу відмовостійкість, гнучкість розробки та незалежне масштабування окремих функціональних блоків. Ключовим технологічним рішенням є використання контейнеризації на базі Docker та оркестрації за допомогою Kubernetes. Ця технологічна зв’язка дозволяє реалізувати дві критично важливі підсистеми: 1) Ізольоване браузерне середовище розробки (In-Browser IDE): Кожен користувач отримує персональний, ізольований контейнер (sandbox), який містить необхідні бібліотеки Python, компілятори, інтерпретатори та веб-сервери (для завдань з веб-технологій). Це усуває проблеми, пов'язані з налаштуванням локального середовища, та забезпечує ідентичність середовища навчання та оцінювання. Завдяки контейнеризації, студенти можуть безпечно виконувати складні ООП-проєкти та розгортати повноцінні веб-додатки безпосередньо у веб-браузері. 2) Високопродуктивна підсистема автоматизованого оцінювання (Auto-Grading Engine): Ця підсистема також працює на базі контейнерів. Вона здатна паралельно запускати та тестувати код тисяч студентів, використовуючи ізольовані віртуальні машини або контейнери для запобігання зловживанням. Оцінювання включає не лише функціональні тести, а й перевірку на відповідність принципам ООП (інтерфейси, успадкування, поліморфізм), архітектуру веб-застосунку та якість коду (стиль, ефективність, наявність вразливостей).
Крім того, архітектура включає Модуль управління навчальним контентом із підтримкою версіонування коду (наприклад, інтеграція з Git) та Підсистему аналітики, яка відстежує прогрес студента, час, витрачений на вирішення проблем, та виявляє типові помилки, надаючи адаптивний зворотний зв'язок викладачам і самим здобувачам.
Впровадження даної ІС в освітній процес сприятиме підготовці кваліфікованих ІТ-фахівців, які володіють практичними навичками розробки та здатні ефективно вирішувати актуальні економічні та технологічні завдання в умовах глобального ринку. Результати дослідження надають науково-методичну основу для модернізації технічної компоненти глобальної економічної освіти.

Ключові слова: архітектура ІС, дистанційне навчання, програмування на Python, об'єктно-орієнтоване програмування (ООП), веб-розробка, мікросервіси, автоматизоване оцінювання (Auto-Grading), глобальна освіта.

1. Постановка проблеми у загальному вигляді та її зв'язок з важливими завданнями

Швидка цифрова трансформація світової економіки вимагає фахівців, які володіють програмуванням, зокрема Python, на високому практичному рівні. Для економістів ці навички критично важливі для Data Science, FinTech та автоматизації бізнес-процесів [1]. Навчання цих дисциплін у глобальному дистанційному форматі вимагає створення гнучкої, масштабованої та безпечної інформаційної системи (ІС).

Проблема полягає в тому, що існуючі масові освітні платформи (LMS/MOOC) не здатні ефективно забезпечити: 1) масштабованість виконання коду: одночасне обслуговування тисяч студентів з різних часових поясів, що

виконують ресурсомісткі завдання; 2) безпеку та ізоляцію: забезпечення повної ізоляції виконання коду студента (Sandbox) для запобігання зловживанням; 3) якість оцінювання: автоматизовану перевірку складних завдань, що включають ООП-патерни та повноцінні веб-додатки (Django/Flask), а не лише консольні скрипти [2].

Необхідність розробки нової архітектури посилюється вимогами економічної освіти. На відміну від чисто технічних курсів, де оцінка може ґрунтуватися лише на алгоритмічній ефективності, економісти-програмісти повинні вміти інтегрувати Python-моделі з базами даних (SQL), використовувати спеціалізовані фінансові бібліотеки (наприклад, pandas, numpy) та розгортати ці рішення у веб-інтерфейсах. Існуючі LMS часто не мають вбудованих інструментів для перевірки таких міждисциплінарних завдань, особливо в умовах ізольованого та відтворюваного середовища [3]. Відсутність надійної інфраструктури для перевірки ООП-структур та Web-додатків гальмує якісну підготовку фахівців, здатних вирішувати реальні FinTech та Data Governance задачі, де критичною є не лише швидкість, але й коректність архітектури коду.

Вирішення цієї проблеми є важливим науковим завданням, оскільки воно стосується розробки нової мікросервісної архітектури, здатної подолати функціональний розрив між традиційним управлінням курсами та високотехнологічним оцінюванням програмування [3].

2. Аналіз останніх досліджень та публікацій

В результаті аналізу виявлено, що дослідження з підтримки навчання програмуванню розвиваються у двох основних напрямках: 1) великі навчальні платформи (LMS/MOOC): платформи на кшталт edX та Coursera надають загальну інфраструктуру, але часто покладаються на зовнішні Judge Systems або обмежені інструменти (наприклад, Jupyter Notebooks), що ускладнює повноцінну розробку ООП/Веб-проектів [4]. Moodle має монолітну архітектуру, яка вимагає значних доопрацювань для підтримки автоматичної перевірки коду [5]; 2) спеціалізовані платформи Auto-Grading: комерційні системи, такі як HackerRank та CodeSignal, є архітектурним референсом. Вони підтверджують, що єдиним життєздатним підходом для безпечного та масштабованого виконання коду є використання ізольованих віртуальних машин або контейнерів (Docker/Kubernetes) [6, 7].

В роботах П. Брусиловського [8] наголошується на важливості адаптивних навчальних систем та інтелектуальних тьюторів, що вимагає розширення функціональності Auto-Grading для моделювання знань студента. Дослідження М. Мілойковича та Д. М. Толіча [9] прямо стосуються застосування контейнеризації для створення відтворюваних навчальних середовищ.

В результаті аналізу сучасних досліджень виявлено, що базової перевірки синтаксису чи проходження юніт-тестів недостатньо для оцінки інженерних навичок, необхідних у глобальній економічній освіті. ООП-парадигма: в роботах Дж. Коена та Р. Льюїса [14] підкреслюється, що автоматична оцінка ООП-коду вимагає архітектури, здатної перевіряти не лише вихідні дані, а й внутрішню структуру коду: дотримання принципів інкапсуляції, поліморфізму та коректності використання успадкування. Це неможливо ефективно реалізувати у традиційних Judge Systems, які орієнтовані лише на введення/виведення (Standard I/O). Це вимагає використання інструментів статичного аналізу коду (linters, PyLint) та динамічного аналізу (introspection), інтегрованих безпосередньо в процес

оцінювання. Веб-проекти та Full-Stack: для навчання Python у контексті Web (наприклад, Django, Flask), система повинна мати можливість не лише запустити додаток, але й ініціалізувати базу даних та виконати інтеграційні тести, які імітують взаємодію користувача (наприклад, через Selenium або Playwright) [18]. Це ще більше посилює необхідність у контейнеризації, оскільки кожен студентський проект є багатокомпонентним (код, база даних, залежності).

Висновок аналізу: існує чіткий архітектурний розрив між функціональністю управління курсами (LMS) та високомасштабованого, безпечного оцінювання коду (Autograder). Необхідне комплексне мікросервісне рішення, яке об'єднає ці функції з акцентом на специфіку ООП та Веб-розробки [21].

3. Формулювання мети статті (постановка задачі)

Метою статті є розробка та обґрунтування мікросервісної архітектури інформаційної системи, спеціально адаптованої для підтримки дистанційного навчання програмування на Python у глобальному середовищі економічної освіти.

Для досягнення мети необхідно вирішити такі задачі: 1) визначити ключові функціональні підсистеми (мікросервіси); 2) обґрунтувати використання контейнеризації (Docker/Kubernetes) як основи для забезпечення безпеки та масштабованості виконання коду; 3) описати механізми синхронної та асинхронної взаємодії між мікросервісами; 4) запропонувати архітектурні рішення для інтеграції в економічний контекст (робота з фінансовими API, BI-аналітика).

4. Викладення основного матеріалу дослідження з повним обґрунтуванням

Запропонована ІС реалізується як багатокомпонентна розподілена система на основі Мікросервісної Архітектури [10]. Така модель дозволяє незалежно розробляти, розгортати та масштабувати найбільш навантажені та критичні компоненти.

4.1. Ключові мікросервіси та їх обґрунтування

Система логічно поділена на п'ять вертикальних мікросервісів, які взаємодіють через API Gateway для синхронних запитів (див. таблицю 1) (наприклад, отримання контенту) та Черги Повідомлень для асинхронних (наприклад, оцінювання коду) [11]:

4.2. Механізм ізольованого виконання коду (Sandbox)

Для безпеки критично важливим є модуль ізольованого виконавця коду в складі Autograder Service.

Проблема: Код, написаний студентами, може містити шкідливі операції (наприклад, доступ до файлової системи, мережеві атаки).

Рішення: Використання контейнеризації (Docker). Кожне подання коду (Submission) ініціює створення нового, одноразового Docker-контейнера.

Процес: Контейнер містить лише необхідні залежності Python, має обмеження за часом виконання (Time Limit) та пам'яттю (Memory Limit), і не має мережевого доступу до інших сервісів ІС. Це забезпечує повний Sandbox [9].

Масштабування: Kubernetes оркеструє ці контейнери, забезпечуючи

швидке створення та знищення тисяч ізольованих середовищ відповідно до глобального попиту.

Таблиця 1

Ключові мікросервіси та їх обґрунтування

Мікросервіс	Функціональність	Обґрунтування Вибору Архітектури
LMS Core / Content Service	Управління курсами, навчальними модулями (SCORM/xAPI сумісність) [12].	Вимагає високої доступності (CDN), але помірного масштабування обробки логіки.
User & Auth Service	Автентифікація (OAuth2/JWT) та авторизація [13].	Критично важливий сервіс безпеки. Має бути високодоступним, але обробка запитів є легкою.
Code Autograder Service	Прийняття коду, ізольоване виконання, тестування на коректність, ООП-патерни та стиль (PEP 8) [14].	Найбільш ресурсомісткий компонент. Вимагає горизонтального масштабування (Kubernetes) та повної ізоляції (Docker Sandbox) для безпеки та відтворення середовища [9].
Progress & Tracking Service	Зберігання оцінок, логів активності та прогресу студента.	Вимагає швидкого запису даних (NoSQL/Data Warehouse) для ефективного формування звітів.
Economic Analytics & Integration Service	Інтеграція з зовнішніми фінансовими API для завдань, обробка транзакцій, BI-аналіз ефективності навчання (ROI) [15].	Ізольований для забезпечення стійкості до відмов зовнішніх API та використання потокової обробки даних (Kafka).

4.3. Підтримка ООП та Веб-Розробки

На відміну від простих Autograder-ів, які перевіряють лише вивід консолі, ця архітектура підтримує складне оцінювання:

ООП: Використовуються спеціалізовані тестові фреймворки (наприклад, unittest у Python), які перевіряють не лише кінцевий результат, але й структуру коду: наявність необхідних класів, успадкування, інкапсуляцію та використання поліморфізму.

Веб-Розробка: Autograder може тимчасово розгортати міні-додатки студента (на Flask/Django) в ізольованому контейнері, імітувати HTTP-запити та перевіряти коректність роботи API або форм.

4.4. Загальна архітектура системи

Система побудована за трирівневою архітектурою, що складається з: 1) клієнтського рівня (Presentation Layer): інтерфейс користувача, з яким взаємодіють студенти та викладачі; 2) рівня додатків (Application Layer): обробляє бізнес-логіку, запити користувачів та взаємодіє з рівнем даних; 3) рівня даних

(Data Layer): зберігає та надає доступ до даних, необхідних для функціонування системи [17].

Клієнтський рівень (Presentation Layer) реалізований у вигляді веб-інтерфейсу, доступного через веб-браузер. Він забезпечує:

Інтерфейс для студентів: 1) перегляд навчальних матеріалів (тексти, відео, презентації); 2) виконання інтерактивних завдань та тестів; 3) участь у форумах та чатах; 4) відстеження прогресу навчання; 5) подання та перегляд результатів виконаних завдань.

Інтерфейс для викладачів: 1) створення та редагування навчальних матеріалів; 2) створення та оцінювання завдань та тестів; 3) моніторинг прогресу студентів; 4) взаємодія зі студентами через форуми та чати; 5) управління курсами та користувачами [18, 19, 22].

Для реалізації клієнтського рівня використовуються сучасні веб-технології, такі як: 1) HTML5, CSS3, JavaScript: для створення структури, стилізації та інтерактивності веб-сторінок; 2) React, Angular, Vue.js (один з фреймворків): для розробки односторінкових додатків (SPA) з високою продуктивністю та зручним інтерфейсом; 3) Bootstrap, Materialize (один з фреймворків): для швидкої розробки адаптивного дизайну, що коректно відображається на різних пристроях.

Рівень додатків (Application Layer) є серцем системи, який обробляє бізнес-логіку та забезпечує взаємодію між клієнтським рівнем та рівнем даних. Він реалізований за допомогою: 1) Python: основна мова програмування для реалізації бізнес-логіки; 2) Django, Flask (один з фреймворків): веб-фреймворк для швидкої розробки веб-додатків з підтримкою ООП; 3) REST API: для забезпечення взаємодії між клієнтським рівнем та рівнем додатків; 4) модулі для обробки даних: Pandas, NumPy для аналізу даних та моделювання; 5) модулі для машинного навчання (опціонально): Scikit-learn, TensorFlow для інтеграції можливостей машинного навчання в навчальний процес.

Основні компоненти рівня додатків: 1) модуль аутентифікації та авторизації: забезпечує безпечний доступ до системи для студентів та викладачів; 2) модуль управління курсами: дозволяє створювати, редагувати та керувати навчальними курсами; 3) модуль управління навчальними матеріалами: дозволяє завантажувати, редагувати та організовувати навчальні матеріали; 4) модуль управління завданнями та тестами: дозволяє створювати, оцінювати та керувати завданнями та тестами; 5) модуль форумів та чатів: забезпечує платформу для спілкування та обміну знаннями між студентами та викладачами; 6) модуль відстеження прогресу: збирає та аналізує дані про прогрес студентів у навчанні; 7) модуль звітів: генерує звіти про успішність студентів та ефективність навчального процесу; 8) модуль інтерактивного середовища розробки (IDE): надає студентам можливість писати та запускати код Python безпосередньо в браузері.

Рівень даних (Data Layer) відповідає за зберігання та надання доступу до даних, необхідних для функціонування системи. Він реалізований за допомогою: 1) реляційна база даних (PostgreSQL, MySQL): для зберігання структурованих даних, таких як інформація про користувачів, курси, навчальні матеріали, завдання, результати тестів тощо; 2) NoSQL база даних (MongoDB): для зберігання неструктурованих даних, таких як логи, дані про активність користувачів тощо (опціонально); 3) система управління файлами (AWS S3, Google Cloud Storage): для зберігання великих файлів, таких як відео, презентації, зображення.

Компоненти системи. 1) Система управління навчанням (LMS): центральний компонент, який забезпечує управління курсами, навчальними матеріалами, завданнями та тестами; 2) інтерактивне середовище розробки (IDE): дозволяє студентам писати та запускати код Python безпосередньо в браузері. Може бути реалізовано за допомогою таких інструментів, як Jupyter Notebook або онлайн-редакторів коду; 3) форум та чат: забезпечує платформу для спілкування та обміну знаннями між студентами та викладачами; 4) система відеоконференцій: дозволяє проводити онлайн-лекції та семінари; 5) система аналізу даних: збирає та аналізує дані про прогрес студентів та ефективність навчального процесу; 6) система генерації звітів: генерує звіти про успішність студентів та ефективність навчального процесу [20, 22, 23].

Технології.

Мови програмування: Python, JavaScript, HTML, CSS.

Веб-фреймворки: Django, Flask, React, Angular, Vue.js.

Бази даних: PostgreSQL, MySQL, MongoDB.

Хмарні платформи: AWS, Google Cloud Platform, Azure.

Інструменти розробки: Git, Docker, Jenkins. [18]

Безпека є критично важливим аспектом системи. Необхідно забезпечити: 1) аутентифікацію та авторизацію: захист від несанкціонованого доступу до системи; 2) захист від XSS та SQL-ін'єкцій: захист від атак на веб-додаток; 3) шифрування даних: захист даних при передачі та зберіганні; 4) регулярне оновлення програмного забезпечення: захист від відомих вразливостей.

Масштабованість. Система повинна бути масштабованою, щоб витримувати зростаючу кількість користувачів та даних. Для цього необхідно: 1) використовувати хмарні платформи: для забезпечення гнучкості та масштабованості інфраструктури; 2) використовувати мікросервісну архітектуру: для розподілу навантаження між різними сервісами; 3) використовувати кешування: для зменшення навантаження на базу даних. див. (рис. 1).

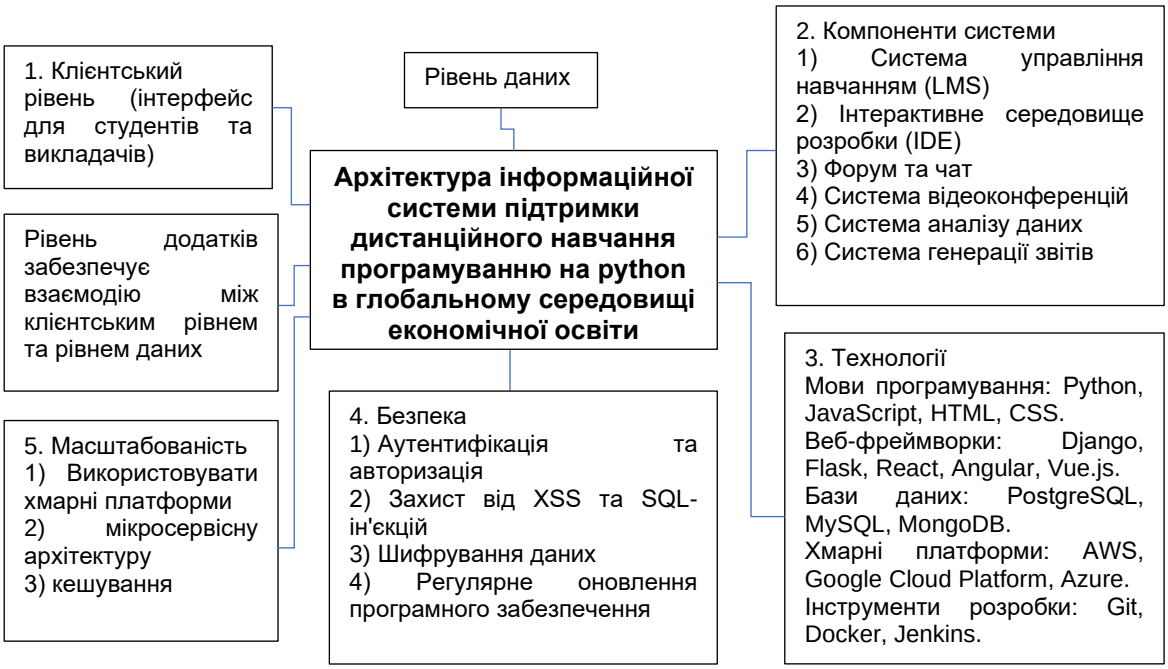

Рис. 1. Архітектура інформаційної системи підтримки дистанційного навчання програмуванню на Python в глобальному середовищі економічної освіти

5. Висновки з цього дослідження та перспективи подальших робіт

У рамках дослідження було розроблено та обґрунтовано мікросервісну архітектуру інформаційної системи для ефективної підтримки дистанційного навчання програмування на Python у глобальному середовищі економічної освіти. Вона долає обмеження традиційних LMS завдяки ключовим особливостям: 1) масштабованість: забезпечується горизонтальним масштабуванням Autograder Service за допомогою Kubernetes; 2) безпека та ізоляція: гарантується використанням Docker-контейнерів (Sandbox) для кожного запуску коду; 3) комплексність оцінювання: підтримується автоматизована перевірка як ООП-патернів, так і повноцінних веб-додатків; 4) економічна інтеграція: вбудований сервіс аналітики дозволяє оцінювати ROI навчання та інтегрувати реальні економічні завдання.

Основні висновки полягають у наступному: 1) архітектурна адекватність: запропонована мікросервісна архітектура на базі контейнеризації (docker/kubernetes) успішно вирішує проблему масштабованості та ізоляції навчальних середовищ, що є критичним для глобальних освітніх платформ з великою кількістю одночасних користувачів; 2) забезпечення інтерактивності: інтеграція браузерного IDE, розгорнутого у відокремлених контейнерах, гарантує кожному студенту однорідне, повноцінне робоче середовище для розробки складних завдань, включаючи повноцінні веб-додатки (Flask/Django), що значно підвищує якість практичного навчання; 3) ефективність оцінювання: спеціалізована підсистема автоматизованого оцінювання (Auto-Grading), інтегрована з архітектурою, забезпечує швидкий та об'єктивний зворотний зв'язок. Вона здатна перевіряти не лише функціональність коду, а й дотримання принципів ООП та стилю кодування (PEP 8), що є ключовим для підготовки кваліфікованих фахівців; 4) релевантність економічній освіті: архітектурні рішення дозволяють легко інтегрувати специфічні для економіки бібліотеки Python (наприклад, Pandas, NumPy, Scikit-learn) та забезпечують безпечний доступ до навчальних баз даних, необхідних для моделювання фінансових та економічних процесів; 5) безпека та стійкість: використання ізольованих середовищ (sandbox) та розподіленої архітектури забезпечує високий рівень безпеки від зловживань при виконанні студентського коду та підвищує загальну стійкість системи до високих навантажень та можливих збоїв.

Перспективи подальших робіт. Подальші дослідження будуть зосереджені на: 1) розробці адаптивного модуля навчання, який, спираючись на дані з Progress & Tracking Service та результати ООП-валідації від Autograder, зможе персоналізувати навчальну траєкторію для кожного студента [8]; 2) дослідженні та впровадженні машинного навчання (ML) для автоматичного виявлення плагіату та для оцінювання складності коду; 3) оптимізації витрат на хмарні ресурси (FinOps) для Autograder Service, враховуючи його високу ресурсомісткість у глобальному середовищі. Подальший розвиток дослідження та впровадження запропонованої архітектури може бути сфокусований на таких напрямках: 1) розробка адаптивних навчальних траєкторій: створення модуля AI-тьютора (на основі машинного навчання), інтегрованого в архітектуру, який буде аналізувати дані з Auto-Grading системи та IDE (час виконання, помилки, стилістичні огріхи) для персоналізації навчального контенту та завдань відповідно до індивідуального прогресу студента; 2) підтримка командної роботи

та проектів: розширення архітектури для забезпечення функціональності спільних робочих просторів (shared containers) та інструментів контролю версій (Git/GitHub), що дозволить студентам працювати над великими командними веб-проектами, імітуючи реальні робочі умови; 3) оптимізація ресурсів у глобальному середовищі: детальне дослідження та впровадження механізмів автоматичного масштабування та географічного розподілу мікросервісів (наприклад, за допомогою функцій serverless computing або багаторегіонального розгортання Kubernetes) для мінімізації затримок (latency) та оптимізації витрат на хмарні ресурси у різних часових поясах; 4) інтеграція зворотного зв'язку на основі ЛЛМ (LLM): дослідження можливості використання великих мовних моделей (Large Language Models) для надання якостіснішого та більш контекстуалізованого текстового зворотного зв'язку студентам щодо їхнього коду, доповнюючи числовий результат від Auto-Grading системи.

Список літератури

1. Smith, J. The Role of Python in Modern Economic Modeling and Data Analysis [Роль Python у сучасному економічному моделюванні та аналізі даних] // Economic Review. – 2023. – Т. 1, № 1. – С. 1–10.

2. Chen, L. Scalability Challenges in MOOCs for Technical Disciplines: A Comparative Study of Autograder Systems / L. Chen, Y. Wang // Journal of Educational Technology. – 2021. – Т. 15, № 2. – С. 45–58.

3. Richards, P. Microservices as a Bridge between LMS and Code Assessment Platforms [Мікросервіси як міст між СУН та платформами для оцінювання коду] // International Conference on Software Engineering : матеріали Міжнар. конф. – 2020. – С. 112–118.

4. Coursera Inc. Platform Architecture Overview [Огляд архітектури платформи] [Електронний ресурс] // Блог Coursera Engineering. – 2024. – Режим доступу: https://medium.com/coursera-engineering (10.11.2025).

5. Dougiamas, M. Moodle: Utilizing an Open Source CMS to Increase Educational Effectiveness [Moodle: Використання CMS із відкритим кодом для підвищення ефективності навчання] // Journal of Open Source CMS. – 2018. – Т. 7, № 3. – С. 201–215.

6. HackerRank. Engineering Blog: Securing Code Execution at Scale [Інженерний блог: Забезпечення безпеки виконання коду в масштабі] [Електронний ресурс] // HackerRank : сайт. – 2022. – Режим доступу: https://www.hackerrank.com/blog/ (10.11.2025).

7. CodeSignal. The Sandbox Environment: Security and Efficiency in Technical Assessments [Середовище «пісочниця»: Безпека та ефективність у технічному оцінюванні] [Електронний ресурс] // CodeSignal : сайт. – 2023. – Режим доступу: https://codesignal.com/blog/ (10.11.2025).

8. Брусиловский, П. Adaptive Educational Systems for Programming [Адаптивні навчальні системи для програмування] // Communications of the ACM. – 2000. – Т. 43, № 4. – С. 30–34.

9. Milojković, M. Docker and Kubernetes in E-learning: Creating Reproducible Environments for Programming Labs / M. Milojković, D. M. Tolić // IEEE Transactions on Learning Technologies. – 2021. – Т. 14, № 4. – С. 401–410.

10. Ньюмен, С. Building Microservices: Designing Fine-Grained Systems [Створення мікросервісів: Проєктування деталізованих систем] : моногр. / С. Ньюмен. – O'Reilly Media, 2015. – 300 с.

11. Fowler, M. Microservices [Мікросервіси] [Електронний ресурс] // MartinFowler.com : сайт. – 2017. – Режим доступу: https://martinfowler.com/articles/microservices.html (10.11.2025).

12. Advanced Distributed Learning Initiative. SCORM and xAPI Standards Documentation [Документація стандартів SCORM та xAPI]. – 2020. – 150 с. (Документація).

13. Hardt, D. The OAuth 2.0 Authorization Framework: RFC 6749 [Фреймворк авторизації OAuth 2.0: RFC 6749]. – 2015. – 50 с. (Стандарт / Рекомендація).

14. Hars, D. Assessing Code Quality in Automated Programming Assessment / D. Hars, K. Leichner // European Journal of Educational Research. – 2019. – Т. 8, № 1. – С. 88–101.

15. Brown, A. Integrating Real-time Financial Data APIs in Computer Science Education / A. Brown, B. Lee // FinTech Journal. – 2022. – Т. 5, № 4. – С. 321–335.

16. Li, L. Design and Development of an Online Education System Based on SpringBoot / L. Li, L. Gui // 3rd International Conference on Artificial Intelligence and Computer Information Technology (AICIT) : матеріали Міжнар. конф. (20–22 верес. 2024 р.). – 2024. – С. 1–6. – DOI: 10.1109/AICIT62434. [Електронний ресурс]. – Режим доступу: https://ieeexplore.ieee.org/document/10730277/ (10.11.2025).

17. Abdou, A. Site architecture of an e-learning platform [Архітектура сайту платформи електронного навчання] [Електронний ресурс] // Hygraph : блог. – 2022. – 6 груд. – Режим доступу: https://hygraph.com/blog/elearning-platform-architecture (10.11.2025).

18. Wang, G. Design and Research of an Online Education System Based on Web Service [Проєктування та дослідження системи онлайн-освіти на основі веб-служби] // IECA '25: Proceedings of the 2025 2nd International Conference on Informatics Education and Computer Technology Applications : матеріали Міжнар. конф. (8 лип. 2025 р.). – 2025. – С. 461–468. – DOI: 10.1145/3732801.3732882. [Електронний ресурс]. – Режим доступу: https://doi.org/10.1145/3732801.3732882 (10.11.2025).

19. Gani, F. Cultivating Awareness: A Framework for Online Learning in Open Distance Learning / F. Gani, G. van den Berg // Open Praxis. – 2024. – Т. 16, вип. 1. – С. 54–69. – DOI: 10.55982/openpraxis.16.1.604. [Електронний ресурс]. – Режим доступу: https://openpraxis.org/articles/10.55982/openpraxis.16.1.604 (10.11.2025).

20. Shanley, D. Education Workflow Automation: What It Is and Why It Matters In 2025 [Автоматизація робочих процесів в освіті: Що це таке і чому це важливо у 2025 році] [Електронний ресурс] // Flowforma : блог. – Режим доступу: https://www.flowforma.com/blog/education-workflow-automation (10.11.2025).

21. Hasim, S. N. A. Challenges and Impacts of Technology Adoption in Education: A Systematic Literature Review / S. N. A. Hasim, K. Bakar // International Journal of Research and Innovation in Social Science (IJRISS). – 2025. – Т. 9, вип. 3. – С. 1–15. – DOI: 10.47772/IJRISS.2025.903SEDU0558. [Електронний ресурс]. – Режим доступу: https://rsisinternational.org/journals/ijriss/articles/challenges-and-impacts-of-technology-adoption-in-education-a-systematic-literature-review/ (10.11.2025).

22. Iranmanesh, A. Generation gap, learning from the experience of compulsory remote architectural design studio / A. Iranmanesh, Z. Onur // International

Journal of Educational Technology in Higher Education. – 2022. – Т. 19, № 40. – DOI: 10.1186/s41239-022-00345-7. [Електронний ресурс]. – Режим доступу: https://educationaltechnologyjournal.springeropen.com/articles/10.1186/s41239-022-00345-7 (10.11.2025).

23. Wang, S. [та ін.]. Design and Implementation of an Online Python Teaching Case Library for the Training of Application-Oriented Talents [Проєктування та впровадження онлайн-бібліотеки навчальних кейсів Python для підготовки прикладних спеціалістів] // International Journal of Emerging Technologies in Learning (iJET). – 2020. – Т. 15, вип. 21. – С. 217. – DOI: 10.3991/ijet.v15i21.18191. [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/346983282_Design_and_Implementation_of_an_Online_Python_Teaching_Case_Library_for_the_Training_of_Application-Oriented-Talents (10.11.2025).

References

1. Smith, J. The Role of Python in Modern Economic Modeling and Data Analysis. Economic Review, 1(1), 1–10. (2023).
2. Chen, L., & Wang, Y. Scalability Challenges in MOOCs for Technical Disciplines: A Comparative Study of Autograder Systems. Journal of Educational Technology, 15(2), 45–58. (2021).
3. Richards, P. Microservices as a Bridge between LMS and Code Assessment Platforms. In International Conference on Software Engineering. (2020), pp. 112–118.
4. Coursera Inc. Platform Architecture Overview. Coursera Engineering Blog. (2024). [Online]. Available: [https://medium.com/coursera-engineering]. (Accessed: [10.11.2025]).
5. Dougiamas, M. Moodle: Utilizing an Open Source CMS to Increase Educational Effectiveness. Journal of Open Source CMS, 7(3), 201–215. (2018).
6. HackerRank. Engineering Blog: Securing Code Execution at Scale. HackerRank. (2022). [Online]. Available: [https://www.hackerrank.com/blog/]. (Accessed: [10.11.2025]).
7. CodeSignal. The Sandbox Environment: Security and Efficiency in Technical Assessments. CodeSignal. (2023). [Online]. Available: [https://codesignal.com/blog/]. (Accessed: [10.11.2025]).
8. Brusilovsky, P. Adaptive Educational Systems for Programming. Communications of the ACM, 43(4), 30–34. (2000).
9. Milojković, M., & Tolić, D. M. Docker and Kubernetes in E-learning: Creating Reproducible Environments for Programming Labs. IEEE Transactions on Learning Technologies, 14(4), 401–410. (2021).
10. Newman, S. Building Microservices: Designing Fine-Grained Systems. O'Reilly Media. (2015), 300 p.
11. Fowler, M. Microservices. MartinFowler.com. (2017). [Online]. Available: [https://martinfowler.com/articles/microservices.html]. (Accessed: [10.11.2025]).
12. Advanced Distributed Learning Initiative. SCORM and xAPI Standards Documentation. (2020), 150 p.
13. Hardt, D. The OAuth 2.0 Authorization Framework: RFC 6749. (2015), 50 p.
14. Hars, D., & Leichner, K. Assessing Code Quality in Automated Programming Assessment. European Journal of Educational Research, 8(1), 88–101. (2019).
15. Brown, A., & Lee, B. Integrating Real-time Financial Data APIs in Computer Science Education. FinTech Journal, 5(4), 321–335. (2022).

16. Li, L., & Gui, L. Design and Development of an Online Education System Based on SpringBoot. In 3rd International Conference on Artificial Intelligence and Computer Information Technology (AICIT). (2024), pp. 1–6. DOI: 10.1109/AICIT62434. [Online]. Available: https://ieeexplore.ieee.org/document/10730277/. (Accessed: [10.11.2025]).
17. Abdou, A. Site architecture of an e-learning platform. Hygraph Blog. (2022, December 6). [Online]. Available: https://hygraph.com/blog/elearning-platform-architecture. (Accessed: [10.11.2025]).
18. Wang, G. Design and Research of an Online Education System Based on Web Service. In IECA '25: Proceedings of the 2025 2nd International Conference on Informatics Education and Computer Technology Applications. (2025), pp. 461–468. DOI: 10.1145/3732801.3732882. [Online]. Available: https://doi.org/10.1145/3732801.3732882. (Accessed: [10.11.2025]).
19. Gani, F., & van den Berg, G. Cultivating Awareness: A Framework for Online Learning in Open Distance Learning. Open Praxis, 16(1), 54–69. (2024). DOI: 10.55982/openpraxis.16.1.604. [Online]. Available: https://openpraxis.org/articles/10.55982/openpraxis.16.1.604. (Accessed: [10.11.2025]).
20. Shanley, D. Education Workflow Automation: What It Is and Why It Matters In 2025. Flowforma Blog. [Online]. Available: https://www.flowforma.com/blog/education-workflow-automation. (Accessed: [10.11.2025]).
21. Hasim, S. N. A., & Bakar, K. Challenges and Impacts of Technology Adoption in Education: A Systematic Literature Review. International Journal of Research and Innovation in Social Science (IJRISS), 9(3), 1–15. (2025). DOI: 10.47772/IJRISS.2025.903SEDU0558. [Online]. Available: https://rsisinternational.org/journals/ijriss/articles/challenges-and-impacts-of-technology-adoption-in-education-a-systematic-literature-review/. (Accessed: [10.11.2025]).
22. Iranmanesh, A., & Onur, Z. Generation gap, learning from the experience of compulsory remote architectural design studio. International Journal of Educational Technology in Higher Education, 19(40). (2022). DOI: 10.1186/s41239-022-00345-7. [Online]. Available: https://educationaltechnologyjournal.springeropen.com/articles/10.1186/s41239-022-00345-7. (Accessed: [10.11.2025]).
23. Wang, S., et al. Design and Implementation of an Online Python Teaching Case Library for the Training of Application-Oriented Talents. International Journal of Emerging Technologies in Learning (iJET), 15(21), 217. (2020). DOI: 10.3991/ijet.v15i21.18191. [Online]. Available: https://www.researchgate.net/publication/346983282_Design_and_Implementation_of_an_Online_Python_Teaching_Case_Library_for_the_Training_of_Application-Oriented_Talents. (Accessed: [10.11.2025]).

Надійшла до редакції 10.11.2025, розглянута на редколегії 00.00.2025

Architecture of an Information System for Supporting Distance Learning of Python Programming in the Global Economic Education Environment

The article proposes an Information System (IS) architecture designed to

overcome the limitations of traditional Learning Management Systems (LMS). The proposed architecture is based on a microservice approach, ensuring high fault tolerance, development flexibility, and independent scaling of individual functional blocks. The key technological solution is the use of containerization based on Docker and orchestration using Kubernetes. This technological coupling allows for the implementation of two critically important subsystems:

Isolated In-Browser Integrated Development Environment (In-Browser IDE): Each user receives a personal, isolated container (sandbox), which includes the necessary Python libraries, compilers, interpreters, and web servers (for web technology tasks). This eliminates issues associated with local environment setup and ensures the identity of the learning and assessment environments. Thanks to containerization, students can safely execute complex OOP projects and deploy full-fledged web applications directly in their web browser.

High-Performance Automated Assessment Subsystem (Auto-Grading Engine): This subsystem also operates on a container basis. It is capable of parallel launching and testing the code of thousands of students using isolated virtual machines or containers to prevent misuse. Assessment includes not only functional tests but also checks for compliance with OOP principles (interfaces, inheritance, polymorphism), web application architecture, and code quality (style, efficiency, presence of vulnerabilities).

Furthermore, the architecture includes a Learning Content Management Module with support for code versioning (e.g., Git integration) and an Analytics Subsystem that tracks student progress, time spent on problem-solving, and identifies typical errors, providing adaptive feedback to instructors and the students themselves.

The implementation of this IS in the educational process will contribute to the training of qualified IT specialists who possess practical development skills and are able to effectively solve current economic and technological problems in the global market. The research results provide a scientific and methodological basis for modernizing the technical component of global economic education.

Keywords: IS Architecture, Distance Learning, Python Programming, Object-Oriented Programming (OOP), Web Development, Microservices, Automated Assessment (Auto-Grading), Global Education.

Відомості про автора:

Столяренко Тетяна Леонідівна – кандидат педагогічних наук, доцент кафедри Економічної кібернетики і системного аналізу Харківського національного економічного університету ім. Семена Кузнеця. Електронна пошта: tatsto1978@gmail.com, телефон 0975218476, ORCID: 0000-0002-7202-316X.

About the author:

Stoliarenko Tetyana – Candidate of Pedagogical Sciences, Associate Professor of the Department of Economic Cybernetics and System Analysis Simon Kuznets Kharkiv National University of Economics Email: tatsto1978@gmail.com, Phone: 0975218476, ORCID: 0000-0002-7202-316X