ІНТЕГРАЦІЯ ХМАРНИХ IDE У ПРАКТИКУ ДИСТАНЦІЙНОГО НАВЧАННЯ ПРОГРАМУВАННЯ

Якименко Артем Олександрович
здобувач другого рівня вищої освіти, спеціальність Комп'ютерні науки
Тернопільський національний педагогічний університет імені Володимира Гнатюка
yakymenko_ao@fizmat.tnpu.edu.ua
Вовкодав Олександр Валерійович
кандидат технічних наук, доцент кафедри інформатики та методики її навчання
Тернопільський національний педагогічний університет імені Володимира Гнатюка
o.vovkodav@tnpu.edu.ua

Пандемія COVID-19 прискорила цифрову трансформацію освіти та актуалізувала проблему ефективного дистанційного навчання програмування. Традиційний підхід з локальним встановленням інтегрованих середовищ розробки (IDE) виявився неефективним у віддаленому форматі через технічні бар'єри: несумісність операційних систем, відсутність прав адміністратора, складність налаштування залежностей для розробки, обмежені ресурси пристроїв студентів.

Хмарні IDE (Cloud-based Integrated Development Environments) – інноваційне рішення, що переносить середовище розробки у браузер, забезпечуючи доступ з будь-якого пристрою без локального встановлення, тому обґрунтування педагогічної ефективності та методика інтеграції хмарних IDE у процес дистанційного навчання програмування є основною метою дослідження.

Теоретичне обґрунтування важливості даної проблематики можна почати з педагогічної концепції. Теорія конструктивізму розглядає навчання як активний процес побудови знань через практичну діяльність. Програмування є класичним прикладом конструктивістського навчання: студент створює працюючі програми, експериментує, отримує миттєвий зворотний зв'язок від системи.

Концепція зони найближчого розвитку (Zone of Proximal Development) Виготського особливо актуальна для хмарних IDE, які виступають «інтелектуальними інструментами», що розширюють можливості студента: автодоповнення коду, інтегроване налагодження, підказки в реальному часі.

Модель TPACK (Technological Pedagogical Content Knowledge) М. Кохлера та П. Мішри описує необхідність гармонійної інтеграції трьох типів знань [1]:

- Technological Knowledge – володіння хмарними IDE;
- Pedagogical Knowledge – методика навчання програмування;
- Content Knowledge – знання мов програмування та алгоритмів.

Теорія когнітивного навантаження (Cognitive Load Theory, J. Sweller) пояснює ефективність хмарних IDE, а саме як зменшення зовнішнього когнітивного навантаження (налаштування середовища), що дозволяє спрямувати когнітивні ресурси на внутрішнє навантаження (засвоєння алгоритмів і синтаксису) [2].

Таблиця 1

Еволюція IDE: від локальних до хмарних

Покоління	Період	Характеристика	Приклади
1	1980-2000	Локальні IDE, монолітна архітектура	Turbo Pascal, Borland C++
2	2000-2010	Модульні IDE з плагінами	Eclipse, NetBeans, Visual

			Studio
3	2010-2020	Легкі редактори з розширеннями	VS Code, Sublime Text, Atom
4	2020-теперішній час	Хмарні IDE з веббраузером	GitHub Codespaces, Gitpod, Replit

Аналіз сучасних платформ хмарних IDE:

GitHub Codespaces – хмарна версія Visual Studio Code, інтегрована в екосистему GitHub [3]. Можна використовувати для курсів з Git/GitHub, командних проєктів, CI/CD практик. Його основні переваги: повна інтеграція з GitHub (репозиторії, Pull Requests, Issues), підтримка DevContainers (опис середовища як коду), доступ до всіх розширень VS Code, потужні обчислювальні ресурси (до 32 GB RAM). Щодо обмежень: платна модель після безкоштовного ліміту (60 годин/місяць), потрібен акаунт GitHub, залежність від інтернет-з'єднання.

Gitpod – автоматизована платформа, яка створює готове середовище з будь-якого Git-репозиторію [4]. Рекомендовано для самостійної роботи студентів, швидкого старту проєктів. Перевагами даної платформи є: конфігурація через.gitpod.yml (Infrastructure as Code), підтримка GitHub, GitLab, Bitbucket, швидке розгортання (10-20 секунд), режим prebuilds (попередньо зібрані середовища). Наступний перелік характеризує як обмеження: 50 годин/місяць у безкоштовному плані, менша кількість розширень порівняно з VS Code Desktop.

Replit – освітньо-орієнтована платформа з акцентом на спільну діяльність та навчання [5]. Можна використовувати для початкових курсів, інтерактивних завдань, молодших курсів. Переваги: спеціальні функції для освіти (Teams for Education), multiplayer Mode (спільне редагування коду), вбудована система автоперевірки (Unit Tests, Input/Output Tests), підтримка 50+ мов програмування, інтерактивні уроки та челенджі. Обмеження: менш потужні для складних проєктів, обмежений обсяг безкоштовного сховища.

Методику інтеграції хмарних IDE можна представити у вигляді наступної організаційно-педагогічною моделлю (рис. 1).

Рис. 1. Організаційно-педагогічна модель

Інтеграція хмарних IDE узгоджується з сучасними педагогічними теоріями (конструктивізм, TPACK, теорія когнітивного навантаження) та забезпечує ефективне середовище для дистанційного навчання програмування. Сучасні платформи (GitHub Codespaces, Gitpod, Replit) досягли високого рівня функціональності та придатні для повноцінного освітнього використання. Порівняльний аналіз показав диференціацію платформ за призначенням: Replit

оптимальний для початківців, Gitpod – для середнього рівня, GitHub Codespaces – для просунутих курсів з акцентом на DevOps. Щодо педагогічних переваг, зменшення когнітивного навантаження за рахунок автоматизації технічного налаштування, індивідуалізація через ізольовані середовища для кожного студента, спільна співпраця завдяки real-time спільному редагуванню, автентичність через використання інструментів реальної розробки (Git, CI/CD).

Список використаних джерел

1. Mishra P., Koehler M. J. Technological pedagogical content knowledge : A framework for integrating technology in teachers' knowledge. Teachers College Record. 20016. Vol. 108. № 6. P. 1017–1054. URL: https://doi.org/10.1111/j.1467-9620.2006.00684.x (дата звернення: 27.10.2025).

2. Sweller J. Cognitive load theory. In J. P. Mestre & B. H. Ross (Eds.). The psychology of learning and motivation: Cognition in education. P. 37–76. Elsevier Academic Press. URL: https://doi.org/10.1016/B978-0-12-387691-1.00002-8 (дата звернення: 27.10.2025).

3. GitHub Codespaces Documentation. URL: https://docs.github.com/codespaces (дата звернення: 27.10.2025).

4. Gitpod Documentation. URL: https://www.gitpod.io/docs (дата звернення: 27.10.2025).

5. Replit Docs. URL: https://docs.replit.com (дата звернення: 27.10.2025).

ОПТИМІЗАЦІЯ ПРОДУКТИВНОСТІ ІГОР, СТВОРЕНИХ З ВИКОРИСТАННЯМ РУШІЯ GODOT ТА C#

Якименко Карина Миколаївна
здобувач першого рівня вищої освіти, спеціальність Комп'ютерні науки
Тернопільський національний педагогічний університет імені Володимира Гнатюка
yakymenko_km@fizmat.tnpu.edu.ua
Василенко Ярослав Пилипович
викладач кафедри інформатики та методики її навчання
Тернопільський національний педагогічний університет імені Володимира Гнатюка
yava@fizmat.tnpu.edu.ua

У сучасних умовах стрімкого розвитку інформаційних технологій вебзастосунки стали основним інструментом для реалізації бізнес-процесів, надання послуг і забезпечення комунікації між користувачами та сервісами. Щодня зростають вимоги до швидкодії, масштабованості, інтерактивності та зручності вебсистем. Тому перед розробниками постає завдання не лише створювати функціональні рішення, а й забезпечувати ефективну взаємодію між клієнтською та серверною частинами програмного продукту.

Одним із найпоширеніших підходів до побудови сучасних вебсервісів є архітектура REST API, яка забезпечує обмін даними у форматі JSON між різними компонентами системи. Такий підхід дає можливість розділити логіку застосунку на незалежні частини, що підвищує гнучкість, масштабованість і зручність підтримки проєкту [1].

Серед технологій, що забезпечують ефективну розробку REST API, особливе місце посідає Express.js – мінімалістичний і гнучкий фреймворк для Node.js, який спрощує створення серверних застосунків, маршрутизацію запитів і роботу з middleware. Його перевагами є висока продуктивність, модульність і розвинена