Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра автоматизації, комп'ютерних наук і технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти – бакалавр
за освітньо-професійною програмою
«Комп'ютерні науки»
зі спеціальності
122 – Комп'ютерні науки

тема роботи:

«Розробка та розгортання веб-сайту школи вивчення іноземних мов з використанням хмарної платформи Google Cloud»

Виконав студент гр. КН-22-1 _____ Лут О. Ю.

Керівник _____ Харламенко В.Ю.

Нормоконтроль _____ Маринич І. А.

Завідувач кафедри _____ Рубан С. А.

Кривий Ріг – 2026

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет: *інформаційних технологій*

Кафедра: *автоматизації, комп'ютерних наук і технологій*

Ступінь вищої освіти: *Бакалавр*

Спеціальність: *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ

Зав. кафедрою: *к.т.н. Рубан С.А.*

_____

« *26* » *січня* 2026 р.

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра

студентові групи КН-22-1 *Луту Олександру Юрійовичу*

1. **Тема кваліфікаційної роботи:** *«Розробка та розгортання веб-сайту школи вивчення іноземних мов з використанням хмарної платформи Google Cloud»*

затверджено наказом по університету № *173с* від *30.03.2026 р.*

2. Термін здачі кваліфікаційної роботи: *10.06.2026 р.*

3. Склад кваліфікаційної роботи: *Пояснювальна записка обсягом 76 с., презентація у Microsoft PowerPoint (15 слайдів) в електронному та друкованому вигляді*

4. Консультанти кваліфікаційної роботи:

Розділ 1-2 доц. Харламенко В.Ю.

Нормоконтроль доц. Маринич І. А.

5. Календарний план:

№	Етапи роботи	Термін виконання
1	*Вступ*	*01.04.26*
2	*Розділ 1*	*05.04.26*
3	*Розділ 2*	*01.05.26*
4	*Висновки*	*25.05.26*
5	*Оформлення кваліфікаційної роботи*	*28.05.26*
6	*Підготовка презентації та графічного матеріалу*	*20.05.26*
7	*Підготовка доповіді до захисту*	*10.06.26*

6. Дата видачі завдання: *28.01.2026 р.*

Керівник _____ / Харламенко В.Ю./

7. Запевнення: *Я, Лут Олександр Юрійович, запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.*

Із чинним Положенням про академічну доброчесність Криворізького національного університету ознайомлений.

Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі умисних порушень робота не допускається до захисту або оцінюється незадовільно.

Студент _____ / Лут О. Ю./

АНОТАЦІЯ

Лут О. Ю. Розробка та розгортання веб-сайту школи вивчення іноземних мов з використанням хмарної платформи Google Cloud : кваліфікаційна робота бакалавра : 122 – Комп’ютерні науки. Кривий Ріг. Криворізький національний університет, 2026. 75 с.

Актуальність теми полягає в тому, що цифровізація освітнього середовища та стрімкий розвиток EdTech-сегменту зумовлюють необхідність використання гнучких і масштабованих хмарних рішень. Застосування low-code підходу в межах платформи Google Cloud дозволяє зменшити інфраструктурні витрати та скоротити час розгортання веб-сервісів, що є важливим для сучасних освітніх установ.

Метою роботи є проєктування, розробка та розгортання веб-орієнтованої інформаційної системи школи іноземних мов на базі хмарної платформи Google Cloud, що включає реалізацію інтерактивного користувацького інтерфейсу, автоматизацію обробки клієнтських заявок засобами Google Apps Script та забезпечення мережевої безпеки з використанням Cloudflare.

У першому розділі розглянуто особливості EdTech-сегменту, проведено порівняльний аналіз існуючих веб-рішень мовних шкіл та обґрунтовано вибір хмарного стеку. Сформульовано постановку задачі на розробку системи.

У другому розділі розглядається процес логічного та фізичного проєктування системи. Побудовано UML-діаграми, що відображають основні бізнес-процеси та алгоритми обробки клієнтських заявок. Описано реалізацію фронтенд-частини за допомогою Google Sites та бекенд-логіки на основі Google Apps Script. Окрему увагу приділено налаштуванню DNS-інфраструктури та забезпеченню захищеного з’єднання із застосуванням SSL-шифрування засобами Cloudflare.

Ключові слова:

GOOGLE CLOUD, GOOGLE SITES, APPS SCRIPT, CLOUDFLARE, LOW-CODE, ВЕБ-САЙТ, АВТОМАТИЗАЦІЯ

ЗМІСТ

ВСТУП

Для мовної школи сайт — це вже не просто сторінка в інтернеті. Це основний спосіб залучити клієнта і перший контакт між потенційним учнем та адміністрацією. Успішність такого закладу напряму залежить від того, наскільки зручно, швидко та надійно організована його цифрова взаємодія з аудиторією.

Традиційні методи розробки програмних продуктів часто супроводжуються високим порогом входу та значними експлуатаційними витратами. Для суб’єктів малого та середнього бізнесу критично важливим є пошук балансу між функціональністю та економічною доцільністю. Концепція Low-code розробки в межах екосистеми Google Cloud пропонує інструментарій, що дозволяє проєктувати інтерфейси з інтегрованою логікою обробки даних без необхідності утримання дорогої серверної інфраструктури. Це дозволяє нівелювати технічні бар’єри при створенні веб-рішень.

Використання Google Sites як фронтенд-платформи дозволяє зосередитися на проєктуванні користувацького досвіду (UX), тоді як застосування Google Apps Script забезпечує виконання логіки на стороні сервера. Впровадження кастомних програмних вставок (Embed-коду) розширює стандартні можливості хмарних конструкторів, наближаючи їх за функціоналом до повноцінних веб-застосунків.

Фінальний етап — публікація сайту в мережі. Це передбачає налаштування зв'язку між хмарним хостингом і власним доменним іменем. Особлива увага приділяється делегуванню DNS-керування та реалізації алгоритмів переадресації домену на цільові адреси, що забезпечує зручність та стабільність користувацької навігації.

Актуальність теми визначається потребою малого освітнього бізнесу у доступних і надійних веб-рішеннях в умовах зростаючої конкуренції на ринку онлайн-послуг.

Метою роботи є проєктування, розробка та розгортання веб-орієнтованої інформаційної системи школи іноземних мов на базі хмарної платформи Google Cloud, що включає реалізацію інтерактивного користувацького інтерфейсу,

автоматизацію обробки клієнтських заявок засобами Google Apps Script та забезпечення мережевої безпеки з використанням Cloudflare.
Для досягнення поставленої мети визначено та вирішено такі завдання:

– Проаналізувати специфіку цифровізації бізнес-процесів у сфері EdTech та визначити роль веб-орієнтованих систем у залученні клієнтів.

– Провести порівняльну характеристику Low-code та High-code підходів для обґрунтування вибору хмарного стеку технологій Google Cloud.

– Розробити логічну структуру системи через побудову UML-діаграм.

– Спроєктувати архітектуру веб-ресурсу та реалізувати його візуальну частину на базі сервісу Google Sites із використанням кастомних компонентів.

– Розробити функціональні модулі та сценарії на мові Google Apps Script для автоматизації обробки форм зворотного зв’язку та взаємодії з базою даних.

– Сконфігурувати параметри розгортання системи з використанням Cloudflare для забезпечення SSL-шифрування та керування доменним іменем.

– Здійснити верифікацію системи шляхом аналізу показників продуктивності з використанням сервісу Google Lighthouse.

Об’єктом розробки є процес створення та розгортання веб-орієнтованих систем для інформаційного супроводу суб’єктів освітньої діяльності.

Предметом дослідження є інструменти хмарної платформи Google Cloud, методи автоматизації на базі Google Apps Script та технології забезпечення безпеки веб-ресурсів.

РОЗДІЛ 1
АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ ТА АВТОМАТИЗАЦІЇ ІНФОРМАЦІЙНИХ СИСТЕМ МАЛОГО БІЗНЕСУ

1.1 Особливості цифровізації бізнес-процесів у сфері EdTech

Сектор освітніх технологій (EdTech) переживає активні зміни: Дослідники підкреслюють, що цифровізація освіти є першочерговим завданням розвитку цифрового суспільства в Україні, а впровадження цифрових інструментів суттєво автоматизує адміністративні процеси освітніх закладів [1]. Класичні методи адміністрування поступаються місцем інтегрованим цифровим екосистемам. Впровадження інноваційних рішень у діяльність мовних шкіл зумовлене необхідністю масштабування послуг без пропорційного збільшення операційних витрат. Для суб'єктів малого підприємництва цифровізація виступає не лише засобом оптимізації, а й ключовим інструментом зменшення відставання у технологіях з великими корпоративними гравцями.

Ефективність цифрової трансформації безпосередньо залежить від того, наскільки грамотно побудована інформаційна система — зокрема, чи забезпечує вона цілісність даних на всіх етапах роботи зі клієнтом. Першим кроком автоматизації є об'єднання всіх каналів залучення клієнтів з єдиною системою обробки їхніх запитів.

Враховуючи обмеженість ресурсів малого бізнесу, особливої актуальності набуває застосування хмарних технологій, що дозволяють перенести капітальні витрати на обладнання у площину гнучких операційних рішень. У межах цього розділу розглянуто роль веб-орієнтованих систем, проаналізовано стек хмарних інструментів та визначено технічні бар'єри, що виникають при їх інтеграції в освітній процес.

1.1.1 Роль веб-орієнтованих інформаційних систем в управлінні освітніми центрами

У сфері EdTech мовні школи потребують інструментів, що забезпечують постійний контакт із клієнтом. Кожна дія відвідувача на сайті — від перегляду курсів до заповнення форми — запускає певний ланцюжок операцій у системі. Процеси залучення клієнтів у ніші вивчення іноземних мов характеризуються високим рівнем конкуренції та складністю конверсійної воронки. Маркетинг освітніх послуг вимагає від закладів освіти інноваційного підходу до цифрової взаємодії з клієнтом, а веб-ресурс стає основним інструментом залучення аудиторії [2].

Ефективність функціонування школи прямо залежить від швидкості обробки вхідного ліда. Веб-ресурс є першою точкою контакту у маркетинговій воронці: від того, наскільки зручно він побудований, залежить чи залишиться відвідувач чи одразу закриє вкладку. Автоматизація збору даних дозволяє уникнути фрагментації інформації, що є типовою проблемою при використанні розрізнених месенджерів або паперових журналів. Структурну схему взаємодії веб-системи з бізнес-процесами мовної школи наведено на рисунку 1.1.

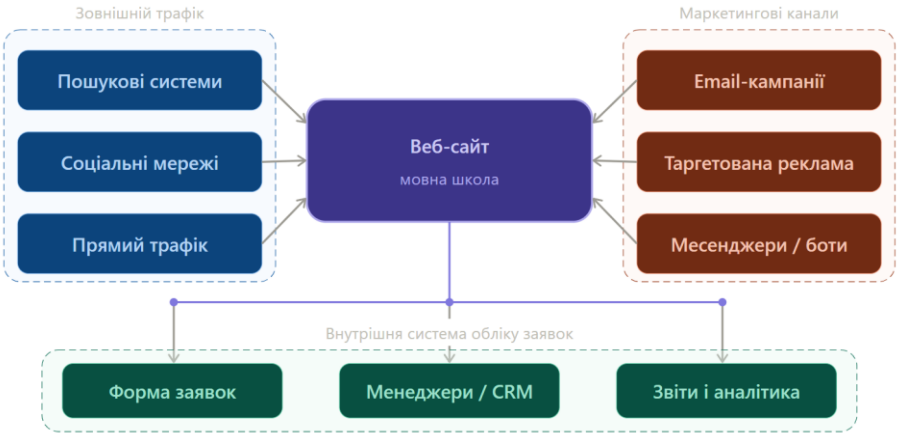

Рисунок 1.1 – Структурна схема взаємодії веб-системи з бізнес-процесами мовної школи

Централізація даних у межах веб-системи забезпечує прозорість управління. Аналіз вхідного трафіку дає змогу виявити пікові години активності та найбільш

затребувані навчальні курси, а функціонал самостійного вибору курсів розвантажує адміністративний персонал.

Інформаційна архітектура сайту мовної школи також повинна враховувати специфіку мобільного трафіку: адаптивність інтерфейсів та швидкість завантаження сторінок є критичними факторами ранжування у пошукових системах. Не менш важливим є забезпечення безпеки передачі персональних даних – веб-орієнтована ІС має гарантувати цілісність інформації.

1.1.2 Огляд сучасних хмарних рішень для автоматизації

Дослідження підтверджують, що саме пандемія COVID-19 дала значний поштовх широкому впровадженню цифрових технологій в освіті, а хмарні сервіси стали невід'ємним елементом організації освітнього процесу [3]. Вибір технологічного стеку визначається двома факторами: функціональністю та сукупною вартістю впровадження і підтримки (TCO). Ринок хмарних послуг пропонує три основні моделі надання ресурсів: SaaS (готове програмне забезпечення за підпискою), PaaS (платформа для виконання коду) та IaaS (оренда інфраструктури). Для малого бізнесу найбільш раціональним є гібридний підхід, що поєднує простоту SaaS-конструкторів із гнучкістю PaaS-середовищ. Google Cloud є одним із популярних рішень у цьому сегменті завдяки безкоштовним лімітам та безшовній інтеграції між сервісами Workspace.

Порівняльний аналіз трьох основних хмарних моделей для освітніх проєктів наведено у таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз хмарних моделей для освітніх проєктів

Характеристика	SaaS	IaaS	PaaS / Гібрид
Вартість старту	Низька (передплата)	Середня (оренда)	Нульова (Free Tier)
Складність налаштування	Мінімальна	Дуже висока	Середня (Low-code + JS)
Гнучкість дизайну	Обмежена шаблоном	Повна	Гнучка (Embed-кодування)

Продовження табл. 1.1

Автоматизація бекенду	Закрита вендором	Потребує написання API	Повна (через Apps Script)
Безпека	На боці провайдера	На боці розробника	Гарантована Google
Масштабованість	Обмежена тарифом	Ручне керування ресурсами	Автоматична (Cloud-native)

Google Sites як фронтенд-платформа дозволяє реалізувати професійний інтерфейс у стислі терміни. На відміну від традиційних CMS, вона не потребує оновлення плагінів або моніторингу безпеки сервера. Оскільки стандартних засобів візуального редактора недостатньо для реалізації специфічної бізнес-логіки, архітектура системи доповнюється кастомними компонентами на базі HTML/JS та серверним середовищем Google Apps Script (GAS).

GAS – це JavaScript-runtime середовище, що виконується безпосередньо в хмарі Google. Воно дозволяє створювати API-ендпоінти для обробки POST-запитів з веб-сайту та взаємодіє з Google Sheets як із реляційним сховищем даних. Такий підхід усуває необхідність у налаштуванні повноцінних СУБД для малих обсягів інформації.

Важливою складовою екосистеми є мережева інфраструктура. Cloudflare – зовнішній DNS-провайдер та CDN-сервіс, який забезпечує розширений контроль над DNS-записами та безпекою. Використання Cloudflare дозволяє реалізувати повноцінну роботу з доменом другого рівня та SSL-шифрування в режимі Full (Strict), що є критичним для SEO-просування та формування довіри до бренду школи. Інтеграція CDN прискорює доставку контенту до кінцевого користувача незалежно від його географічного розташування.

Описаний стек реалізує serverless-підхід: розробнику не потрібно керувати серверами, що усуває ризик простою через інфраструктурні проблеми і знімає потребу у DevOps-спеціалісті в штаті.

1.1.3 Аналіз існуючих веб-рішень та порівняння платформ

Попри широкий вибір готових рішень, впровадження інформаційних систем у малих мовних школах завжди супроводжується необхідністю пошуку балансу між простотою, функціональністю та вартістю підтримки. Перед визначенням остаточних вимог до власної системи необхідно проаналізувати реально функціонуючі веб-ресурси освітніх закладів [8].

Мета аналізу — зрозуміти, що вже є на ринку, виявити слабкі місця наявних рішень і на цій основі обґрунтувати вибір технологічного стеку для власної розробки. Для порівняння розглянуто три характерні веб-рішення: два, що базуються на традиційній розробці (High-code), та одне рішення на базі візуального конструктора (Low-code).

Аналіз High-code реалізацій

Використання традиційної розробки дозволяє створювати максимально персоналізовані системи, проте супроводжується високим технічним порогом входу та значними витратами:

Міжнародна мережа «Speak Up» (https://speak-up.com.ua/). Даний ресурс є комплексною екосистемою, розгорнутою на стеку Django + HTMX із власним серверним бекендом. Така архітектура обслуговує розгалужену мережу з 13 філій в Україні, підтримує особисті кабінети студентів та складну систему динамічного запису (Рисунок 1.2).

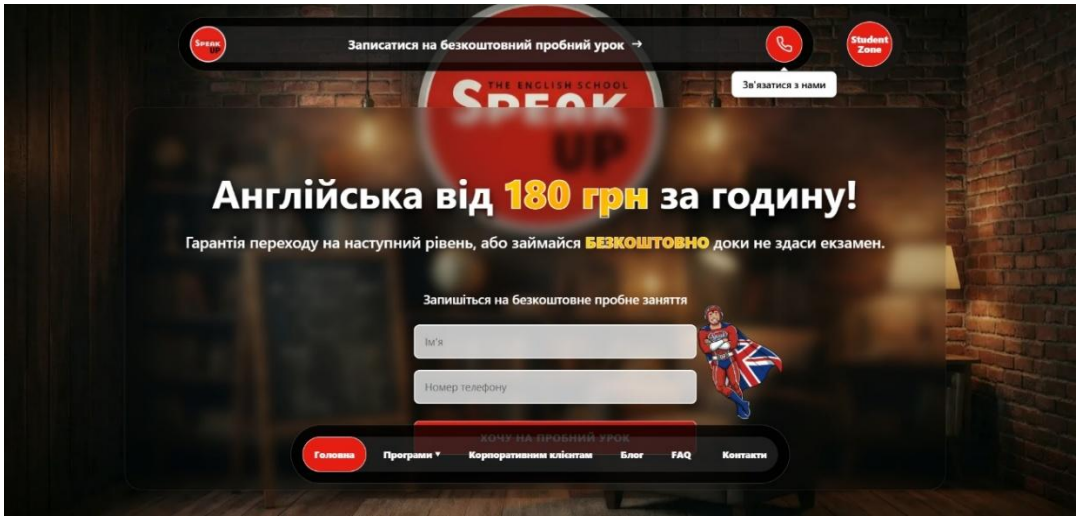

Рисунок 1.2 – Веб-інтерфейс системи Speak Up [10]

Таке рішення є виправданим для мережі національного масштабу, проте воно потребує тривалої розробки (від 4 до 6 місяців) та постійної технічної підтримки (DevOps). Для локальної школи капітальні витрати на таку інфраструктуру роблять розробку економічно недоцільною.

Школа «Wow English» (https://wow-english.ua/). Сайт мережі шкіл для дітей побудовано з використанням системи керування контентом WordPress та візуального конструктора Elementor. Це типова конфігурація для середнього EdTech-бізнесу, яка дозволяє забезпечити розгалужену структуру сторінок та ведення блогу (Рисунок 1.3).

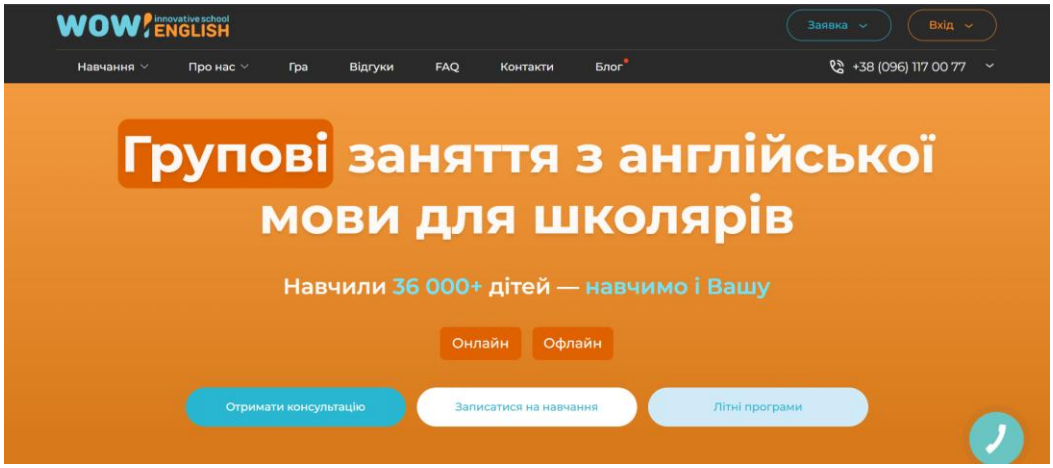

Рисунок 1.3 – Веб-інтерфейс системи Wow English [11]

Вона забезпечує швидший старт порівняно з кастомним бекендом, проте вимагає регулярного оновлення плагінів для підтримки безпеки, платного хостингу та обов'язкового залучення розробника при необхідності структурних змін.

Аналіз Low-code реалізацій (комерційні конструктори)

На противагу традиційній розробці, SaaS-платформи пропонують мінімальний час виходу на ринок, але мають жорсткі архітектурні обмеження.

Школа «LinguaDream» (https://linguadreamvn.wixsite.com/linguadream). Ресурс школи іноземних мов з Вінниці розгорнуто на базі популярного ізраїльського конструктора Wix (Рисунок 1.4). Платформа дозволяє запустити адаптивний сайт за 1–2 тижні.

Рисунок 1.4 – Веб-інтерфейс системи LinguaDream [12]

Однак аналіз виявив критичні для професійного проєкту обмеження: відсутність повного написання власної серверної логіки та неможливість автоматизованої обробки заявок без втручання менеджера. Крім того, на безкоштовному тарифі школа змушена використовувати субдомен (wixsite.com) та відображати рекламу платформи, що негативно впливає на SEO та довіру клієнтів. Перехід же на повноцінний преміум-тариф вимагає щомісячної підписки (19 дол./міс.), що формує постійні операційні витрати.

Висновок про доцільність витрати ресурсів

Аналіз показує чіткий розрив: High-code дає потрібну функціональність, але для малої школи це надто дорого — і у розробці, і у підтримці. Водночас закриті Low-code конструктори забезпечують швидкий дизайн, але не дозволяють інтегрувати серверну автоматизацію без суттєвих переплат. Спираючись на це, можна стверджувати, що витрата фінансових та часових ресурсів на класичну (High-code) розробку або на підписку комерційних конструкторів для малої мовної школи є недоцільною. Оптимальним інженерним рішенням є застосування вільного гібридного стеку на базі екосистеми Google Cloud (Google Sites для візуальної частини та Google Apps Script для серверної логіки). Такий підхід позбавлений обмежень класичного Low-code, оскільки дозволяє безкоштовно обробляти дані на стороні сервера, забезпечуючи високу рентабельність проєкту.

1.2 Підходи до розробки програмного забезпечення

Обґрунтувавши вибір хмарного стеку, необхідно розглянути методологічну основу проєктування — підходи до розробки ПЗ, що безпосередньо визначили чому для реалізації веб-ресурсу мовної школи обрано саме Low-code інструменти, а не традиційний шлях написання коду з нуля.

1.2.1 Еволюція методів розробки: від традиційного програмування до Low-code платформ

Розробка програмного забезпечення пройшла довгий шлях — від написання коду на рівні машинних команд до сучасних декларативних підходів, де розробник описує «що» потрібно, а не «як» це зробити. Початковий етап галузі базувався на імперативній парадигмі, де розробник маніпулював апаратними ресурсами на рівні регістрів. Виникнення мов програмування третього покоління (3GL) – C та Pascal – дозволило структурувати код через процедури та функції, однак поріг входу залишався досить високим. Так звана «програмна криза» 1960-х наочно показала: складність програм зростає швидше, ніж здатність розробників впоратись з нею засобами низькорівневого коду. Саме ця закономірність — складність зростає швидше ніж можливості розробника — є прямим обґрунтуванням відмови від High-code підходу для проєкту мовної школи з обмеженими ресурсами.

Об'єктно-орієнтоване програмування (ООП) зробило наступний крок, запропонувавши інкапсуляцію логіки в багаторазові об'єкти. Проте навіть у сучасних High-code фреймворках значна частка часу витрачається на написання «boilerplate-коду» – типових конструкцій для маршрутизації, автентифікації та підключення до баз даних. У сфері EdTech це обмеження проявилось особливо гостро: поки команда розробляє кастомне рішення, потреби ринку вже змінились (Рисунок 1.5).

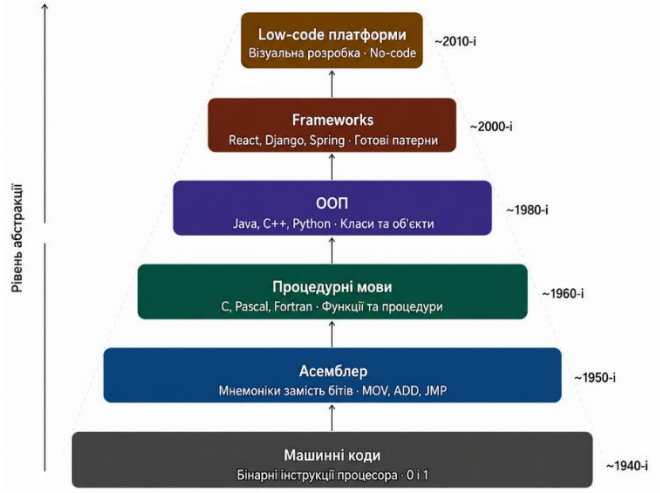

Рисунок 1.5 – Схема еволюції рівнів абстракції в розробці програмного забезпечення

Декларативний підхід, закладений у Low-code системи, базується на концепції візуального моделювання. Замість покрокового опису «як зробити», розробник визначає кінцевий стан «що отримати». Це стало можливим завдяки автоматизованій генерації вихідного коду на основі графічних метаданих. Системи четвертого покоління (4GL) та сучасні PaaS-рішення дозволяють абстрагуватися від керування інфраструктурою, зосередившись на бізнес-логіці.

Сучасні Low-code платформи, зокрема Google Sites, інтегрують принципи компонентної розробки: кожен елемент інтерфейсу є прекомпільованим модулем із заздалегідь визначеними властивостями. Гібридні моделі, де візуальне конструювання доповнюється серверними сценаріями Google Apps Script, усувають розрив між швидкістю розгортання та гнучкістю кастомізації – що є ключовим для динамічних освітніх проєктів.

1.2.2 Порівняння Low-code та High-code підходів

Маючи уявлення про еволюцію методів розробки, розглянемо детально, чому саме Low-code підхід є оптимальним для реалізації поставленого завдання.

Традиційне програмування (High-code) надає розробнику повний контроль над архітектурою, дозволяючи оптимізувати продуктивність на рівні алгоритмів та мережевих протоколів. Це критично для систем з мільйонними аудиторіями, проте

для малого бізнесу High-code часто породжує надлишковий «технічний борг»: кожна написана функція потребує регулярного оновлення залежностей, патчів безпеки та рефакторингу.

Low-code парадигма переносить фокус з написання коду на конфігурування системи. Платформи Google Cloud забезпечують стандартизоване середовище виконання. Проте Low-code має обмеження у вигляді «жорстких рамок» платформи, що потребує використання API для розширення функціоналу за їх межі.

Детальне порівняння техніко-архітектурних характеристик обох підходів наведено у таблиці 1.2.

Таблиця 1.2 – Техніко-архітектурне порівняння High-code та Low-code підходів

Параметр аналізу	High-code рішення	Low-code рішення
Управління станом	Ручне програмування логіки станів	Автоматизоване на рівні компонентів
Робота з базами даних	ORM, SQL-запити, міграції	Нативна інтеграція (Sheets API / GAS)
Frontend-рендерінг	Клієнтський/Серверний (React, Next.js)	Візуальна компоновка (Drag-and-Drop)
Обробка помилок	Кастомна логіка (Try-Catch блокування)	Стандартизовані системні сповіщення
Розгортання	CI/CD пайплайни, Docker-контейнери	Миттєва публікація в один клік
Інтеграційна здатність	Повна (через будь-які протоколи)	Висока (через Fetch API та REST)

Аналіз продуктивності показує, що Low-code скорочує час виходу на ринок (Time-to-Market) на 50–70% [7; 9]. Це зумовлено відсутністю потреби у верстці «з нуля» та наявністю готових механізмів автентифікації. У High-code підході до 30% часу витрачається на написання тестів та налагодження сумісності бібліотек – процеси, приховані від розробника у хмарному стеку Google Cloud.

Важливим аспектом є «витік абстракції» (Abstraction Leakage): у Low-code системах виникають моменти, коли візуальних засобів недостатньо. Саме тут гібридний підхід демонструє свою перевагу – він поєднує простоту Google Sites для каркасу інтерфейсу з потужністю Google Apps Script для серверної обробки.

Результатом є система, що є легшою у підтримці, ніж повністю кастомний код, але значно гнучкішою за типовий конструктор сайтів.

High-code виправданий тоді, коли продукт справді унікальний. Для мовної школи зі стандартними процесами — форма заявки, опис курсів, контакти — це просто зайві гроші і час. Для освітнього центру з типовими бізнес-процесами (лідогенерація, зворотний зв'язок) High-code призводить до зайвих витрат. Low-code дозволяє інвестувати ресурси у контент та клієнтський сервіс, а не в інженерію інфраструктури.

1.2.3 Особливості життєвого циклу розробки ПЗ (SDLC, RAD)

Оскільки для реалізації проєкту обрано Low-code підхід, методологія розробки також має відповідати принципам гнучкості та ітеративності. Саме тому розглядається методологія RAD (Rapid Application Development) як альтернатива класичній каскадній моделі.

Традиційна каскадна модель (Waterfall) передбачає лінійну структуру, де кожен наступний етап не може початися без завершення попереднього. Для EdTech-проєктів така модель є ризикованою через «ефект запізнілого фідбеку»: замовник бачить готовий продукт лише наприкінці циклу, коли внесення змін є критично дорогим (Рисунок 1.6) [13].

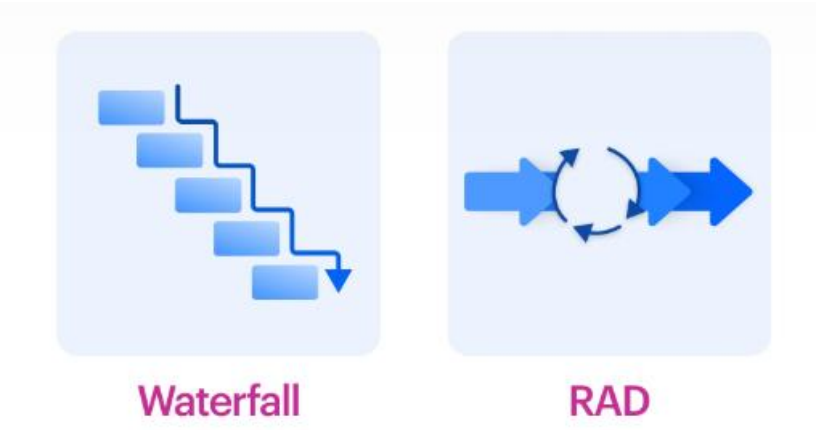

Рисунок 1.6 – Порівняння циклів Waterfall та RAD [13]

Методологія RAD базується на ітеративному прототипуванні та активному залученні замовника до тестування проміжних версій. Вона ідеально доповнює Low-code підхід, оскільки візуальні інструменти дозволяють створювати робочі моделі інтерфейсу за лічені години. Основні фази RAD у межах розробки сайту мовної школи включають:

– Планування та аналіз вимог: формування переліку необхідних модулів (форми заявок, опис курсів, тощо).

– Дизайн та прототипування: створення структури сайту безпосередньо в Google Sites, що дозволяє замовнику оцінити UX/UI на ранніх етапах.

– Ітеративна розробка: додавання бекенд-логіки через Google Apps Script за циклом «розробка – демонстрація – корекція».

– Впровадження: фінальна публікація, налаштування домену та передача системи в експлуатацію.

Циклічність RAD дозволяє виявляти невідповідності вимогам на етапі дизайну, а не після написання тисяч рядків коду. Якщо школа впроваджує новий формат навчання, адміністратор може самостійно відкоригувати структуру сторінки без залучення програміста. Такий підхід перетворює інформаційну систему на живу екосистему, що еволюціонує разом із бізнесом.

Поєднання методології RAD з інструментами Google Cloud дозволяє отримати робочий продукт у стислі терміни з мінімальними витратами — що для малого бізнесу і є найважливішим показником. Відмова від тривалих етапів підготовки документації на користь функціональних прототипів дозволяє скоротити час від ідеї до першої отриманої заявки клієнта, що є критичним для виживання малого бізнесу в умовах високої конкуренції. Таким чином, теоретичний огляд еволюції методів розробки та аналіз методології RAD слугують безпосереднім підґрунтям для вибору інструментарію у практичній частині: Google Sites як середовище візуального конструювання інтерфейсу та Google Apps Script як середовище швидкої реалізації серверної логіки — є типовими представниками Low-code/RAD підходу, описаного у цьому розділі.

1.3 Аналіз предметної області та постановка задачі

На основі проведеного технологічного аналізу та порівняння існуючих веб-рішень, необхідно перейти до детального розгляду об’єкта розробки — діяльності школи іноземних мов. Це дозволить сформувати чіткі технічні вимоги до системи та визначити пріоритетні напрямки автоматизації для малого бізнесу.

1.3.1 Опис діяльності мовної школи

Об’єктом дослідження є процеси інформаційного супроводу та залучення клієнтів у межах діяльності приватної школи вивчення іноземних мов. Сучасна мовна школа функціонує в умовах високої конкуренції, де швидкість реакції на запит потенційного клієнта є критичним фактором успіху.

Основні бізнес-процеси школи включають:

– Інформування клієнтів: надання актуальних даних про методики навчання, вартість курсів та кваліфікацію викладачів.

– Лідогенерація: збір контактних даних потенційних учнів через різні канали комунікації.

– Обробка заявок: класифікація запитів за рівнями знань та обраними мовами, а також подальший зв'язок адміністратора з клієнтом.

– Адміністрування бази даних: ведення обліку звернень та формування графіків навчання.

Окремої уваги заслуговує специфіка комунікації з клієнтом у нішевому освітньому бізнесі. На відміну від масових онлайн-платформ, мовна школа працює переважно з локальною аудиторією, для якої критично важливим є відчуття довіри до закладу ще до першого дзвінка. Веб-ресурс у цьому контексті виконує функцію «цифрового офісу»: він має відповідати на основні питання клієнта — які мови, для кого, скільки коштує і як записатись — без необхідності телефонувати адміністратору. Відсутність такої системи або її неналежна якість прямо впливає на

конверсію: потенційний учень просто обирає школу конкурента з зручнішим сайтом.

На сьогоднішній день у багатьох малих освітніх центрах спостерігається проблема фрагментації даних: запити надходять у різні месенджери, а їх облік ведеться вручну. Це призводить до втрати потенційних клієнтів та низької операційної ефективності. Таким чином, виникає гостра потреба у створенні єдиної веб-орієнтованої точки входу, яка б автоматично структурувала вхідний трафік та зберігала його у хмарному сховищі.

1.3.2 Постановка задачі на розробку

Метою роботи є проєктування, розробка та розгортання веб-орієнтованої інформаційної системи школи іноземних мов на базі хмарної платформи Google Cloud, що включає реалізацію інтерактивного користувацького інтерфейсу, автоматизацію обробки клієнтських заявок засобами Google Apps Script та забезпечення мережевої безпеки з використанням Cloudflare. Згідно з визначеною метою, виділено п’ять основних задач розробки:

1. Проєктування архітектури системи: розробка структурної схеми взаємодії компонентів та побудова UML-діаграм (Use Case, Sequence, Component) для опису логіки системи.

2. Розробка інтерфейсу (Frontend): створення адаптивного веб-сайту на базі платформи Google Sites, що включатиме кастомні компоненти для збору даних, реалізовані засобами HTML/CSS.

3. Реалізація серверної логіки (Backend): написання сценаріїв на мові Google Apps Script для обробки POST-запитів з веб-форм та їх інтеграції з хмарною базою даних у Google Sheets.

4. Конфігурація мережевої інфраструктури: делегування керування DNS-записами сервісу Cloudflare, налаштування SSL-шифрування та оптимізація швидкості доставки контенту через CDN.

5. Верифікація та оцінка: тестування працездатності системи, аналіз показників продуктивності за допомогою Google Lighthouse та підтвердження економічної ефективності обраного стеку.

До розробленої системи висуваються такі технічні вимоги:

– функціонування без постійних операційних витрат у межах безкоштовних тарифних лімітів обраних платформ;

– коректна робота форми зворотного зв'язку з автоматичним збереженням даних без участі адміністратора;

– доступність сайту за доменним ім'ям із підтримкою HTTPS-шифрування;

– можливість самостійного редагування контенту нетехнічним персоналом без залучення розробника

Об’єктом розробки є процес створення та розгортання веб-орієнтованих систем для інформаційного супроводу суб’єктів освітньої діяльності.

Предметом дослідження виступають інструменти платформи Google Cloud, методи автоматизації на базі Apps Script та технології мережевого захисту веб-ресурсів.

Висновки до розділу :

У першому розділі було проведено аналіз специфіки EdTech-сегменту та обґрунтовано вибір хмарного стеку Google Cloud як найбільш рентабельного рішення для малого бізнесу. Огляд сервісів Speak Up, Wow English та LinguaDream виявив недоцільність кастомної розробки або закритих SaaS-платформ через їхню високу вартість та обмежену гнучкість. За результатами аналізу сформульовано технічне завдання на створення гібридної системи на базі Google Sites та Apps Script, що стало базою для подальшого проєктування у другому розділі.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ МОВНОЇ ШКОЛИ

2.1 Архітектурні особливості та безпека платформи Google Sites

Google Sites є хмарною платформою для створення веб-сайтів, що входить до екосистеми Google Workspace. На відміну від традиційних CMS, вона реалізує концепцію serverless-хостингу: розробник не взаємодіє з серверною інфраструктурою безпосередньо — усі обчислювальні ресурси надаються та масштабуються Google автоматично. У межах цього пункту розглядаються архітектурні характеристики платформи, що є основою для побудови веб-ресурсу мовної школи Mother Tongue School.

2.1.1 Функціональні можливості платформи

Google Sites надає розробнику WYSIWYG-редактор з компонентною моделлю побудови інтерфейсу. Інтерфейс редактора поділено на три функціональні панелі: «Вставити» (додавання контентних блоків), «Сторінки» (управління ієрархічною структурою сайту) та «Теми» (налаштування глобального дизайну).

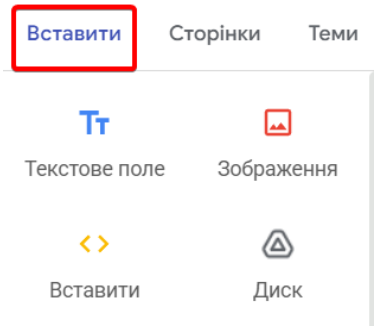

Рисунок 2.1 – Інтерфейс редактора Google Sites: панель «Вставити»

Панель «Вставити» містить такі типи блоків: текстові поля з підтримкою форматування, зображення та медіа-файли з Google Drive, вставка HTML/JavaScript через механізм Embed, блоки контенту з різними варіантами компоновки (одна

колонка, дві, три, сітка), кнопки з налаштуванням кольору та посилання, карусель зображень, розділювачі, зміст (Table of Contents), вставка YouTube-відео.

Панель «Сторінки» забезпечує управління ієрархічною структурою сайту. Для проєкту Mother Tongue School створено дворівневу навігаційну архітектуру: головна сторінка, розділ «Курси та ціни» з вкладеними підсторінками для кожної мови (англійська, польська, німецька) та їхніх підрівнів, розділ «Для дітей» з підрозділами, а також сторінки «Про нас» і «Публічна оферта» (Рисунок 2.2).

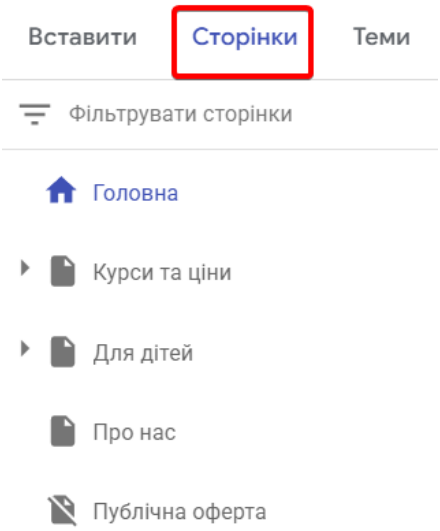

Рисунок 2.2 – Інтерфейс редактора Google Sites: панель «Сторінки»

Панель «Теми» дозволяє глобально керувати візуальним стилем: палітрою кольорів, шрифтами, стилем зображень, параметрами навігаційної панелі та компонентів.

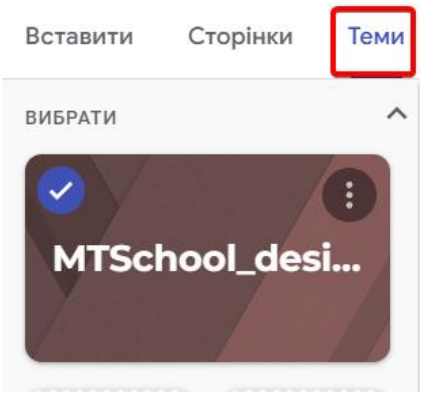

Рисунок 2.3 – Інтерфейс редактора Google Sites: панель «Теми»

Для проєкту розроблено власну кастомну тему MTSchool_design із фірмовими кольорами закладу, що забезпечує візуальну консистентність на всіх сторінках без необхідності змінювати кожен елемент окремо (Рисунок 2.3).

2.1.2 Інтеграція з іншими сервісами

Ключовою архітектурною перевагою Google Sites є нативна інтеграція з екосистемою Google Workspace. Ця інтеграція реалізується на двох рівнях: вбудована (через стандартні блоки редактора) та розширена (через механізм Embed).

Вбудована інтеграція дозволяє розміщувати на сторінках файли з Google Drive, таблиці Google Sheets, форми Google Forms, презентації Google Slides та відео з YouTube безпосередньо через відповідні блоки вставки.

Розширена інтеграція реалізується через блок «Вставити» → «Вставити код». Платформа надає два режими: вставка за URL (для підтримуваних ресурсів) та вставка HTML-коду (для кастомних компонентів). Саме цей механізм є ключовим для реалізації форми зворотного зв'язку: кастомний HTML/CSS/JavaScript-компонент вбудовується в iframe-пісочницю та взаємодіє із серверним обробником Google Apps Script через асинхронний POST-запит (Рисунок 2.4).

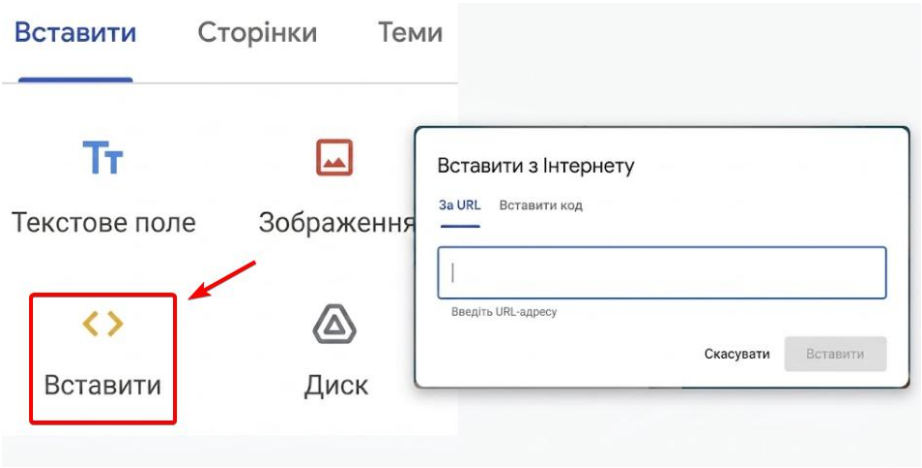

Рисунок 2.4 – Інтерфейс вставки кастомного HTML-коду в Google Sites

2.1.3 Безпека та надійність

Безпека Google Sites забезпечується на рівні хмарної інфраструктури Google і є однією з ключових переваг платформи для малого бізнесу, що не має власного системного адміністратора.

Транспортний рівень захисту реалізовано через автоматичне HTTPS-шифрування всіх з'єднань. Сайт, опублікований на sites.google.com, завжди доступний виключно за захищеним протоколом — платформа не підтримує публікацію по HTTP. У поєднанні з Cloudflare, що забезпечує SSL/TLS-шифрування в режимі Full між браузером користувача та edge-сервером, а також між Cloudflare та Google Sites, досягається наскрізне шифрування запиту.

Автентифікація та авторизація доступу до редагування сайту здійснюється через механізм Google OAuth 2.0 та IAM. Редагувати сайт можуть лише акаунти Google, яким надано відповідний дозвіл власником. Це виключає несанкціоновані зміни контенту навіть у випадку компрометації стороннього сервісу, оскільки дані сайту зберігаються в ізольованому просторі Google Drive.

На відміну від власного VPS-хостингу, де відповідальність за доступність сервісу лежить на розробнику, Google Sites не потребує моніторингу серверів, оновлення операційної системи чи патчів безпеки. За весь період функціонування сайту mtschool.org.ua зафіксовано нульовий час простою, пов'язаного з інфраструктурою платформи.

2.1.4 Обмеження платформи

Попри переваги, Google Sites має ряд архітектурних обмежень, що суттєво впливають на проєктні рішення і потребують компенсаційних механізмів.

Перше і найсуттєвіше обмеження — пісочниця (sandbox) для вбудованого коду. Будь-який HTML/JavaScript, вставлений через блок Embed, виконується в ізольованому iframe з обмеженнями браузерної політики безпеки контенту (CSP). Це унеможливлює прямий доступ до DOM батьківської сторінки та використання деяких Web API. Наслідком є те, що класичні підходи до валідації форм або динамічного оновлення контенту сторінки з iframe не працюють стандартними

методами. Вирішення полягає у використанні postMessage API для комунікації між iframe та батьківською сторінкою, або повній самодостатності кожного компонента — що і реалізовано у формі зворотного зв'язку проєкту.

Друге обмеження — неможливість роботи з кореневим доменом (naked domain) без зовнішнього DNS-сервісу. Google Sites підтримує прив'язку лише піддомену www, тоді як запит до кореневого домену (mtschool.org.ua без www) без додаткового налаштування призведе до помилки. Дане обмеження вирішується засобами Cloudflare через правило переадресації (Page Rule / Redirect Rule): запит до mtschool.org.ua автоматично перенаправляється на www.mtschool.org.ua з кодом 301.

Третє обмеження — відсутність вбудованої серверної логіки. Google Sites є виключно статичним генератором сторінок і не має власного бекенду для обробки даних форм, роботи з базою даних або відправлення email-сповіщень. Це вирішується інтеграцією з Google Apps Script як зовнішнім серверним середовищем, що детально розглядається у підрозділі 2.4.

Четверте обмеження — відсутність вбудованого управління метаданими сторінок (title, meta description) на рівні окремих сторінок, що обмежує гнучкість SEO-оптимізації. Частково ця проблема компенсується коректним налаштуванням DNS та структурою URL, що формується автоматично на основі назв сторінок у панелі «Сторінки».

Зазначені обмеження є відомими технічними характеристиками платформи і не є критичними для завдань інформаційного сайту мовної школи. Кожне з них має інженерне вирішення в межах обраного технологічного стеку, що підтверджує доцільність гібридного підходу до розробки.

2.2 Моделювання системи

Визначивши архітектурні особливості та обмеження платформи Google Sites, перейдемо до формального моделювання системи. UML-діаграми дозволяють зафіксувати поведінкові та структурні характеристики веб-ресурсу до початку

безпосередньої реалізації, виявити потенційні суперечності у вимогах та забезпечити єдине розуміння системи між розробником і замовником.

2.2.1 Побудова UML-діаграм варіантів використання

Діаграма варіантів використання (Use Case Diagram) описує функціональні вимоги до системи через взаємодію акторів із варіантами використання. Для веб-ресурсу мовної школи визначено двох акторів: Visitor (відвідувач сайту) та Administrator (адміністратор закладу) (Рисунок 2.5).

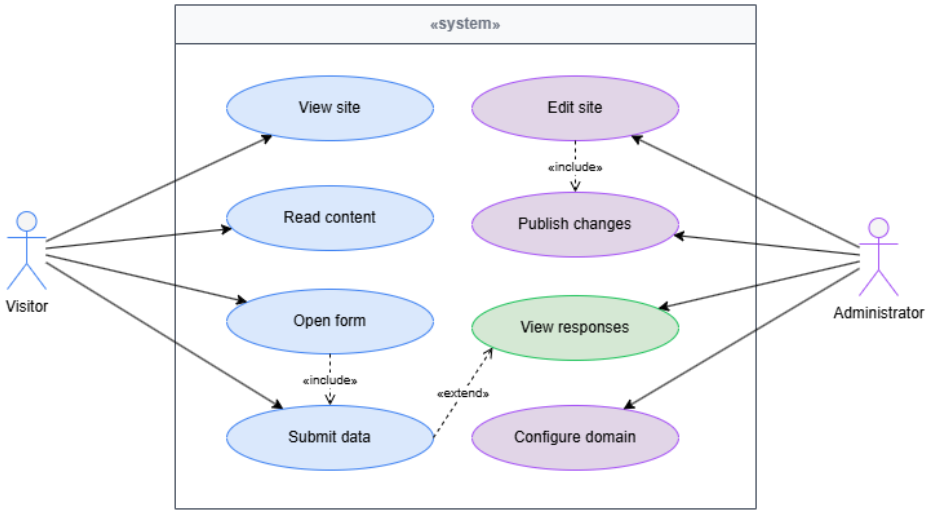

Рисунок 2.5 – Use Case Diagram веб-ресурсу Mother Tongue School

Актор Visitor має доступ до трьох варіантів використання. View site — перегляд опублікованого сайту в браузері. Read content — читання інформації про курси, розклад, викладачів та ціни. Open form — відкриття форми запиту на консультацію, що включає (відношення «include») підваріант Submit data — заповнення та відправлення форми із даними користувача.

Актор Administrator має доступ до чотирьох варіантів використання. Edit site — редагування контенту в WYSIWYG-редакторі Google Sites, яке включає (відношення «include») підваріант Publish changes — публікацію збережених змін для відвідувачів. View responses — перегляд бази даних заявок у Google Sheets. Configure domain — налаштування DNS-параметрів і кастомного домену; цей

варіант є розширенням (відношення «extend») від View responses, оскільки виконується опційно при зміні конфігурації розгортання.

Відношення «include» між Open form та Submit data відображає обов'язкову послідовність: відкриття форми завжди передбачає можливість її відправлення. Відношення «extend» між View responses та Configure domain вказує на те, що налаштування домену є адміністративною дією [15]..

2.2.2 Проєктування компонентної бази та загальної архітектури

Для повного розуміння структури системи розроблено дві взаємодоповнюючі діаграми: System Architecture Diagram, що відображає рівні інфраструктури, та Component Diagram, що показує залежності між програмними компонентами.

System Architecture Diagram організовано за чотирма горизонтальними рівнями (Рисунок 2.6).

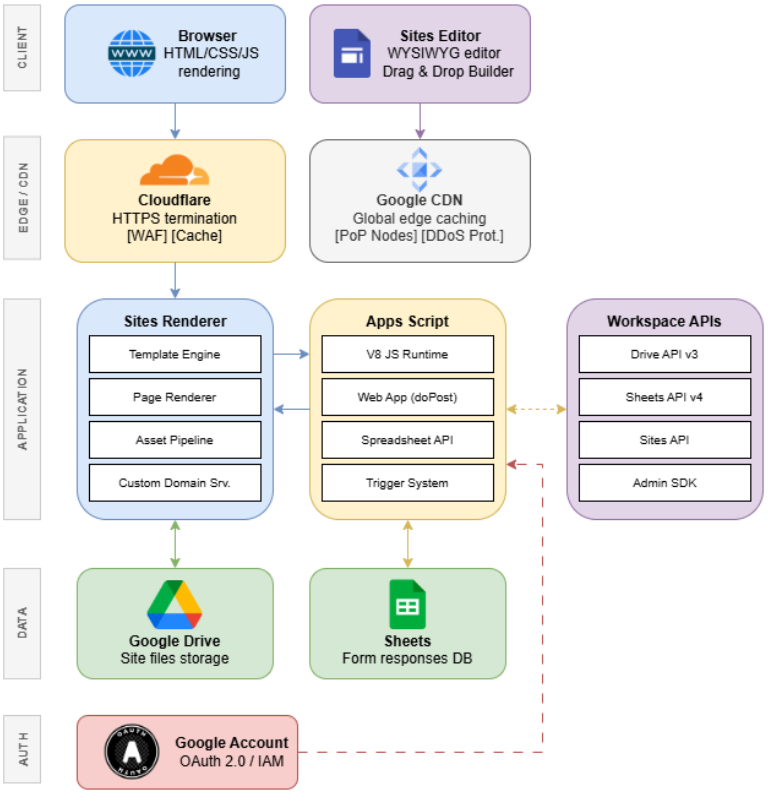

Рисунок 2.6 – System Architecture Diagram веб-ресурсу Mother Tongue School

Рівень CLIENT включає два клієнтські середовища: Browser (HTML/CSS/JS рендеринг для відвідувача) та Sites Editor (WYSIWYG-редактор для адміністратора).

Рівень EDGE/CDN містить Cloudflare (HTTPS-термінація, WAF, кеш) та Google CDN (глобальне edge-кешування, PoP-вузли, DDoS-захист).

Рівень APPLICATION складається з трьох блоків: Sites Renderer (Template Engine, Page Renderer, Asset Pipeline, Custom Domain Service), Apps Script (V8 JS Runtime, Web App doPost, Spreadsheet API, Trigger System) та Workspace APIs (Drive API v3, Sheets API v4, Sites API, Admin SDK).

Рівень DATA включає Google Drive як сховище файлів сайту та Google Sheets як базу даних заявок.

Окремо виділено рівень AUTH — Google Account з OAuth 2.0 / IAM як єдина точка автентифікації для всіх сервісів екосистеми.

Component Diagram деталізує залежності між програмними компонентами на рівні їхніх інтерфейсів (Рисунок 2.7).

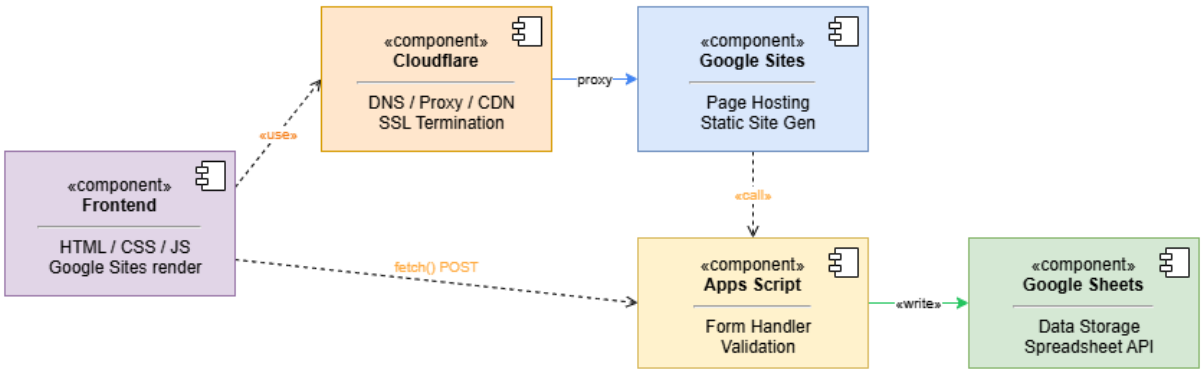

Рисунок 2.7 – Component Diagram веб-ресурсу Mother Tongue School

Компонент Frontend (HTML/CSS/JS, Google Sites render) використовує (відношення «use») компонент Cloudflare (DNS/Proxy/CDN, SSL Termination) для доступу до сайту. Cloudflare, в свою чергу, проксує запити до компонента Google Sites (Page Hosting, Static Site Gen).

Компонент Google Sites викликає (відношення «call») компонент Apps Script при обробці форм. Apps Script надсилає fetch() POST-запит безпосередньо з

Frontend та записує (відношення «write») дані до компонента Google Sheets (Data Storage, Spreadsheet API).

Така архітектура відповідає принципу розділення відповідальності: кожен компонент виконує чітко визначену функцію і взаємодіє з іншими через стандартизовані інтерфейси.

2.2.3 Моделювання динаміки обробки даних веб-форми

Sequence Diagram описує часову послідовність взаємодії між об'єктами системи при виконанні основного сценарію — подачі заявки на консультацію.

У діаграмі визначено чотири учасники: User (відвідувач), Google Sites Web-page, Apps Script Web App (doPost) та Google Sheets Database (Рисунок 2.8).

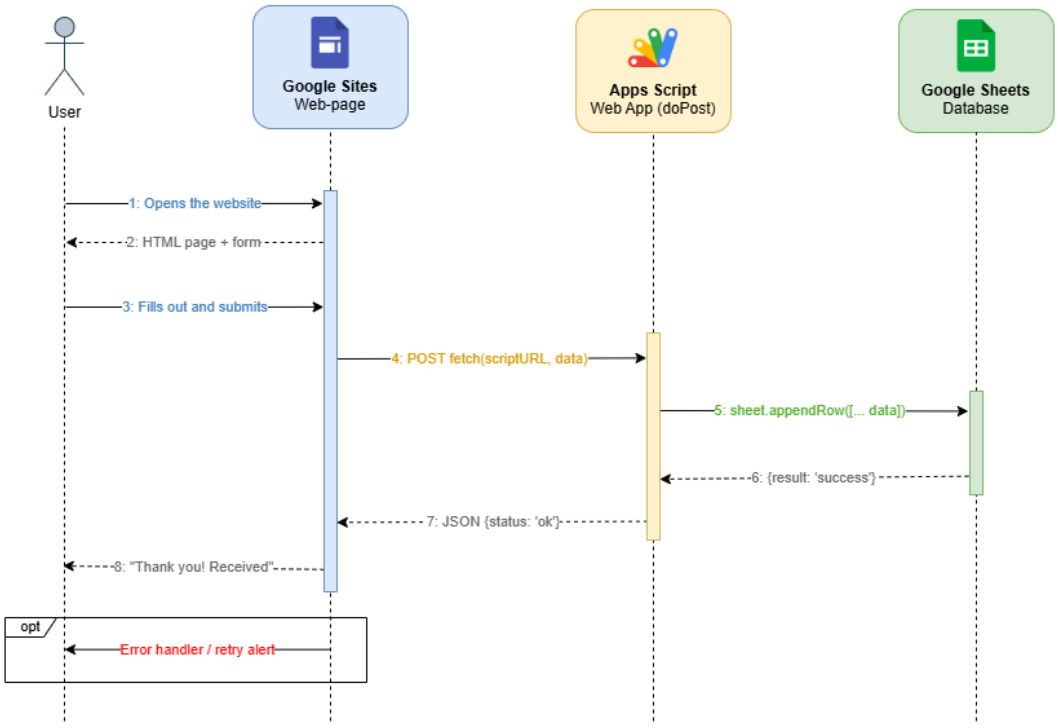

Рисунок 2.8 – Sequence Diagram обробки заявки через веб-форму

Основний потік розгортається наступним чином:

Крок 1. User ініціює HTTP-запит, відкриваючи сайт у браузері.

Крок 2. Google Sites повертає HTML-сторінку разом із вбудованою формою у вигляді iframe-компонента.

Крок 3. User заповнює поля форми та натискає кнопку відправлення.

Крок 4. JavaScript-код форми асинхронно відправляє дані методом fetch() POST на URL серверного скрипту Apps Script.

Крок 5. Apps Script отримує дані та записує новий рядок до Google Sheets.

Крок 6. Google Sheets підтверджує успішний запис і повертає результат Apps Script.

Крок 7. Apps Script повертає JSON-відповідь до Frontend (JSON {status: 'ok'}).

Крок 8. JavaScript форми відображає користувачу повідомлення про успішну відправку.

Альтернативний потік (фрейм opt) передбачає сценарій помилки: у разі відсутності відповіді або отримання HTTP-статусу помилки catch-блок JavaScript відображає Error handler / retry alert — повідомлення про помилку з пропозицією повторити спробу. Кнопка відправлення при цьому розблоковується для повторного використання.

2.2.4 Опис логіки обробки даних у системі

На основі побудованих діаграм формалізуємо повну логіку обробки даних у системі — від введення користувачем інформації до її збереження в базі даних.

Перший етап — клієнтська валідація. До відправлення POST-запиту JavaScript-код форми виконує базову перевірку введених даних. У разі виявлення помилки браузер відображає вбудоване повідомлення валідації і запит не відправляється. Це знижує навантаження на серверний обробник і покращує UX.

Другий етап — відправлення запиту. Після успішної валідації JavaScript блокує кнопку відправлення та змінює її текст на «Відправляємо...», що запобігає дублюванню заявок при повторному натисканні. Дані форми збираються об'єктом FormData та передаються методом fetch() як тіло POST-запиту на URL Web App Google Apps Script.

Третій етап — серверна обробка. Функція doPost(e) Apps Script отримує параметри запиту через об'єкт e.parameter та виконує такі операції:

1. Відкриття Google Sheets за URL через SpreadsheetApp.openByUrl();

2. Отримання цільового аркуша за назвою через getSheetByName("sheet1");

3. Додавання нового рядку з масивом значень [Name, Email, Phone, Message] методом appendRow();

4. Повернення текстової відповіді через ContentService.createTextOutput().

Четвертий етап — обробка відповіді. JavaScript-код форми у .then()-блоці перевіряє успішність запиту: відображає блок #status із повідомленням «Дякуємо! Ми зателефонуємо Вам найближчим часом.», скидає поля форми методом form.reset() та розблоковує кнопку. У .catch()-блоці (альтернативний потік) відображається повідомлення про помилку червоним кольором і кнопка отримує текст «Спробувати знову».

П'ятий етап — зберігання даних. Кожна успішна заявка фіксується окремим рядком у Google Sheets з чотирма полями: ім'я, email, телефон, повідомлення. Таблиця є повністю доступною адміністратору в режимі реального часу без необхідності входу до панелі адміністрування сайту — достатньо відкрити відповідний Google Sheets документ. Це відповідає вимозі самостійного адміністрування системи нетехнічним персоналом, визначеній у постановці задачі.

2.3 Проєктування структури та інтерфейсу системи

Визначивши архітектурні характеристики платформи та формалізувавши поведінку системи засобами UML-моделювання, перейдемо до практичного проєктування веб-ресурсу.

Даний підрозділ охоплює два взаємопов'язані аспекти: формування інформаційної архітектури сайту через побудову sitemap та розробку візуального інтерфейсу із використанням методу прототипування. Ці етапи реалізуються послідовно відповідно до методології RAD і безпосередньо передують технічній імплементації системи.

2.3.1 Розробка структури сайту

Проєктування інформаційної архітектури є першочерговим етапом розробки будь-якого веб-ресурсу. Sitemap визначає ієрархічні зв'язки між сторінками та

формує основу для навігаційної логіки. У контексті мовної школи структура сайту повинна відображати природну класифікацію освітніх послуг та відповідати ментальній моделі потенційного клієнта: відвідувач має інтуїтивно знаходити інформацію про конкретну мову, рівень знань або цільову аудиторію.

Для веб-ресурсу Mother Tongue School розроблено трирівневу ієрархічну структуру. Перший рівень утворює головна сторінка (Головна), яка є точкою входу та концентрує ключові маркетингові меседжі. Другий рівень складається з чотирьох розділів: «Курси та ціни», «Для дітей», «Про нас» та «Публічна оферта». Розділи «Курси та ціни» і «Для дітей» мають підсторінки третього рівня, що деталізують послуги за мовами та цільовими групами (Рисунок 2.9).

Рисунок 2.9 – Ієрархічна структура сторінок веб-ресурсу Mother Tongue School

Розділ «Курси та ціни» містить три підрозділи відповідно до пропонованих мов навчання: «Англійська мова», «Польська мова» та «Німецька мова». Кожен мовний підрозділ, в свою чергу, поділяється на дві спеціалізовані підсторінки: «[Мова]: рівні» — що описує доступні рівні вивчення від A1 до C1 із відповідними цінами — та «[Мова]: інше» — що містить інформацію про спеціалізовані курси та формати навчання, що виходять за межі стандартної шкали рівнів.

Розділ «Для дітей» структуровано за віковими групами та освітніми цілями: «Для дітей 7–13 років», «Для підлітків» та «НМТ та іспити». Такий підхід

відповідає принципу орієнтації на конверсійні потреби різних сегментів аудиторії: батьки молодших школярів, батьки підлітків та абітурієнти мають різні пошукові наміри (search intent) і потребують окремих точок входу.

Реалізація даної структури в середовищі Google Sites здійснюється через панель «Сторінки», де кожна підсторінка пов'язана зі своїм батьківським розділом. Платформа автоматично генерує навігаційну панель на основі визначеної ієрархії, проте для відображення підрозділів у форматі випадаючого меню (dropdown) необхідно явно задати вкладеність в ієрархії панелі «Сторінки».

Навігаційна панель сайту відображає структуру другого рівня у вигляді горизонтального меню. Пункти «Курси та ціни» та «Для дітей» мають індикатор розкриття (стрілка вниз) і реалізовані як інтерактивні dropdown-компоненти, що з'являються при наведенні курсора. Така реалізація є стандартною поведінкою навігаційних компонентів Google Sites при наявності підсторінок і не потребує кастомного коду (Рисунок 2.10).

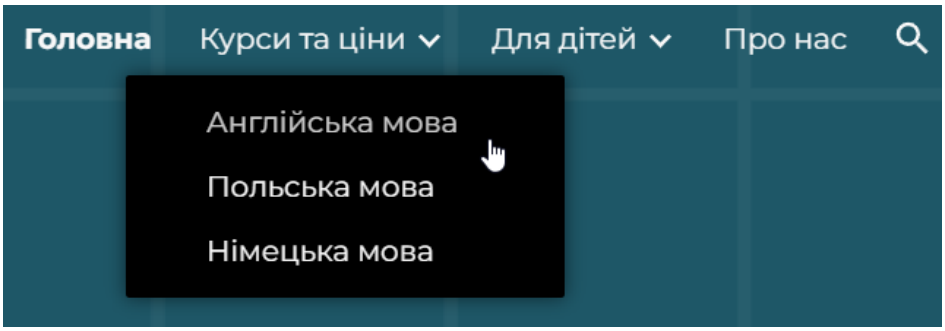

Рисунок 2.10 – Навігаційне меню сайту з розгорнутим dropdown-підменю розділу «Курси та ціни»

Важливою архітектурною особливістю є включення сторінки «Публічна оферта» до структури сайту на рівні другого рівня навігації. Незважаючи на те, що ця сторінка не є частиною основного маркетингового контенту, її наявність відповідає вимогам законодавства України щодо обов'язкового розміщення умов надання послуг, що особливо актуально для закладів освіти, що здійснюють електронну взаємодію з клієнтами.

Розроблена sitemap забезпечує максимальну глибину переходу три кліки від головної сторінки до будь-якого цільового контенту, що відповідає правилу трьох кліків — широко визнаному евристичному принципу UX-проєктування. Це мінімізує показник відмов та збільшує ймовірність конверсії відвідувача у потенційного клієнта школи.

2.3.2 Проєктування та прототипування інтерфейсу користувача

Відповідно до методології RAD, прототипування інтерфейсу передує його кінцевій реалізації та слугує засобом раннього узгодження візуальних рішень із замовником. Для проєкту Mother Tongue School прототип інтерфейсу розроблено у середовищі Figma — провідному інструменті векторного проєктування UI/UX, що дозволяє створювати деталізовані макети з точністю до пікселя та реалізовувати інтерактивні прототипи для тестування навігаційних сценаріїв.

Рисунок 2.11 – Figma-макет головної сторінки веб-ресурсу Mother Tongue School (Hero-секція)

На етапі прототипування визначено основні дизайн-токени системи, що відображають фірмовий стиль школи. Головним кольором фону обрано темно-зелений відтінок (#1E5767 та суміжні варіанти), що асоціюється з академічністю та спокоєм. Акцентний жовто-золотистий колір (#FAD791) використовується для заголовків та ключових CTA-кнопок (call-to-action). Шрифтова система базується на поєднанні чіткого шрифту для заголовків та читабельного основного тексту.

Усі ці параметри зафіксовано у кастомній темі MTSchool_design в середовищі Google Sites на всіх сторінках.

Прототип головної сторінки реалізує класичну структуру лендингу. Hero-секція містить заголовок «Школа іноземних мов», акцентний підзаголовок «MOTHER TONGUE SCHOOL», слоган, іконки соціальних мереж, контактні номери телефонів та кнопку первинного заклику до дії «Записатися на пробний урок →». Вертикальне розташування елементів відповідає принципу F-образного патерну зчитування, що підтверджено дослідженнями поведінки користувачів у веб-середовищі. Декоративні елементи у вигляді фотографій учнів по боках від центральної текстової секції додають людяності та довіри до бренду школи (Рисунок 2.12).

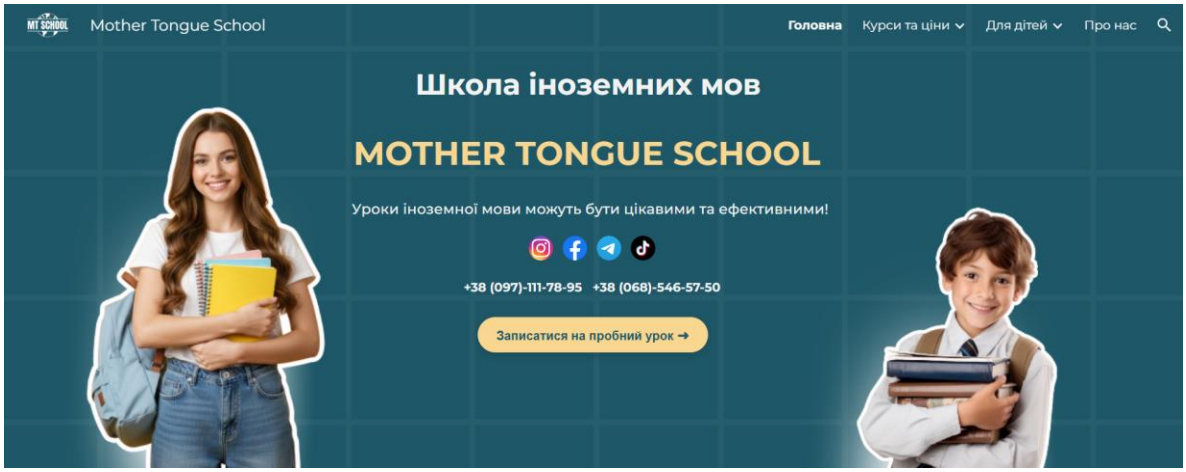

Рисунок 2.12 – Реалізована головна сторінка веб-ресурсу Mother Tongue School

Порівняння макету з реалізацією демонструє точне відтворення затвердженого прототипу: колірна схема, типографіка та компоновка елементів відповідають проєктним рішенням. Ключовою відмінністю реалізації від початкового прототипу є додавання реальних PNG-фотографій учнів із вирізаним фоном, розміщених симетрично по обидва боки від центральної секції. Це рішення прийнято на етапі ітеративного доопрацювання за результатами демонстрації замовнику.

Внутрішні сторінки курсів реалізують уніфікований шаблон структури. Верхня частина сторінки містить назву курсу, кнопки переходу до безкоштовного тесту рівня та запису на пробний урок, а також блок з контактними даними та посиланнями на соціальні мережі.

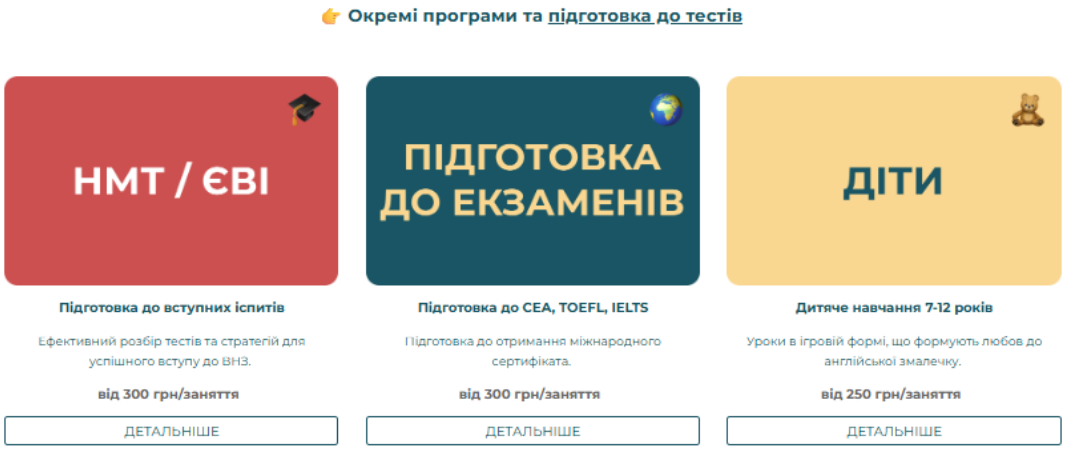

Рисунок 2.13 – Сторінка курсу «Англійська мова» з секцією програм

Форма зворотного зв'язку реалізована як кастомний HTML/CSS-компонент, вбудований через механізм Embed. Дизайн форми навмисно виконано у теплій кремово-бежевій гамі (#F5F0E8), що візуально відокремлює її від основного темно-зеленого фону сторінки та акцентує увагу відвідувача (Рисунок 2.14).

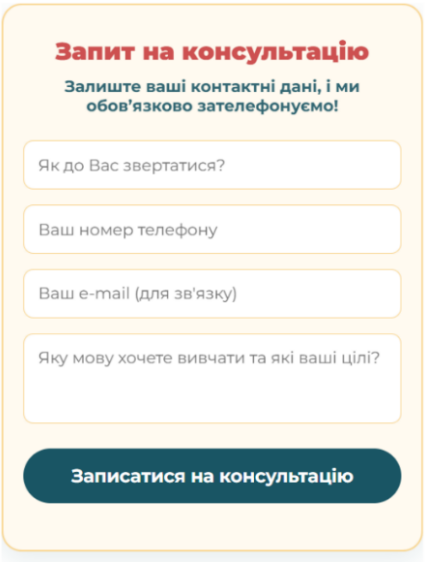

Рисунок 2.14 – Форма зворотного зв'язку «Запит на консультацію»

Форма містить чотири поля: ім'я, номер телефону, електронна пошта та текстове повідомлення з описом бажаного курсу та цілей навчання. Кнопка відправлення «Записатися на консультацію» виконана в темно-зеленому кольорі з контрастним білим текстом, що забезпечує відповідність стандартам WCAG щодо контрастності інтерфейсів.

Адаптивність інтерфейсу забезпечується вбудованими механізмами Google Sites. На мобільних пристроях горизонтальне навігаційне меню автоматично замінюється кнопкою «бургер» (≡), що розгортає вертикальне меню-drawer. Контентні блоки перебудовуються з багатоколонкової сітки у вертикальний стек, а розміри шрифтів та відступи коригуються для забезпечення зручності читання на екранах з шириною від 360 пікселів. Картки рівнів на мобільній версії відображаються послідовно у вигляді вертикального списку замість горизонтального ряду.

Таким чином, проєктування інтерфейсу реалізовано у відповідності до принципів Mobile First та Progressive Enhancement: базова функціональність доступна на будь-якому пристрої, а досвід користувача покращується в міру збільшення можливостей клієнтського середовища. Поєднання Figma-прототипування з ітеративним доопрацюванням у рамках RAD-методології забезпечило відповідність кінцевого інтерфейсу як маркетинговим вимогам замовника, так і технічним обмеженням платформи Google Sites.

2.4 Реалізація інформаційної системи

Спроєктовану інформаційну систему необхідно перевести у площину практичної реалізації. Цей підрозділ охоплює безпосередній процес побудови веб-ресурсу в середовищі Google Sites, розробку серверної логіки засобами Apps Script, налаштування інтеграції з хмарною базою даних Google Sheets та реалізацію кастомних компонентів. Послідовність викладення відповідає технологічному ланцюжку: від фронтенд-складання до бекенд-обробки та збереження даних.

2.4.1 Розробка веб-ресурсу на платформі Google Sites

Маючи затверджену sitemap та узгоджений прототип інтерфейсу, розпочато безпосереднє складання веб-ресурсу у редакторі Google Sites. Процес реалізації охоплює чотири послідовні етапи: конфігурацію глобальних параметрів сайту, налаштування кастомної теми оформлення, побудову контентних блоків на кожній сторінці та верифікацію адаптивного відображення.

Конфігурація глобальних параметрів

Першим кроком є налаштування параметрів через діалог «Налаштування» редактора. Вкладка «Зображення бренду» містить два поля завантаження: логотип, що відображається в навігаційній панелі поруч із назвою «Mother Tongue School», та значок веб-сторінки (favicon), який відображається у вкладці браузера. Обидва елементи завантажено у форматі PNG з прозорим фоном, що забезпечує коректне відображення на будь-якому кольоровому тлі навігаційної панелі. Ключовою перевагою налаштування логотипу на рівні глобальних параметрів є автоматичне відтворення на всіх сторінках сайту без необхідності ручного редагування кожного шаблону окремо (Рисунок 2.15).

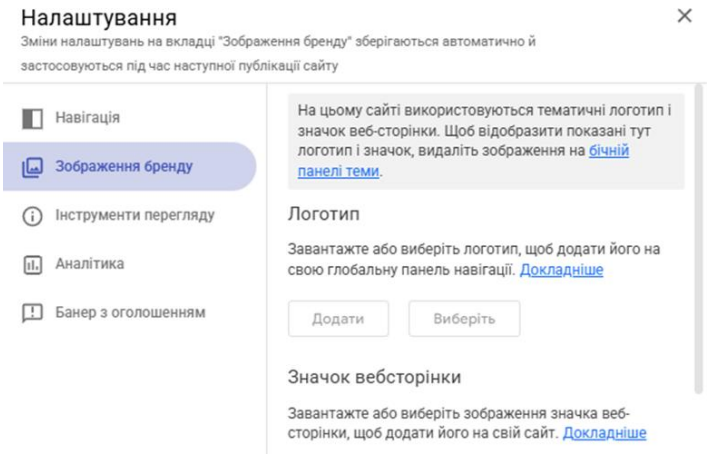

Рисунок 2.15 – Діалог налаштувань Google Sites: вкладка «Зображення бренду» з конфігурацією логотипу та favicon

Налаштування кастомної теми

Паралельно з конфігурацією глобальних параметрів визначено значення кастомної теми оформлення. Панель «Теми» надає розгорнутий редактор кольорів,

де кожному типу елементу відповідає окремий параметр: «За умовчанням» — основний акцентний колір кнопок та інтерактивних елементів; «Фон» — колір підложки секцій; «Назви та заголовки» — колір заголовних текстів; «Основний текст» — колір основного контенту. Для теми MTSchool_design встановлено такі значення: акцентний колір — темно-смарагдовий (#195564), що застосовується до навігаційних елементів та кнопок; колір фону секцій — той самий відтінок для hero-блоків і контрастний кремово-бежевий для інформаційних секцій. Шрифтову систему визначено через підключення шрифту Montserrat, що є CSS-змінною теми і забезпечує консистентне відображення кирилічних символів (Рисунок 2.16).

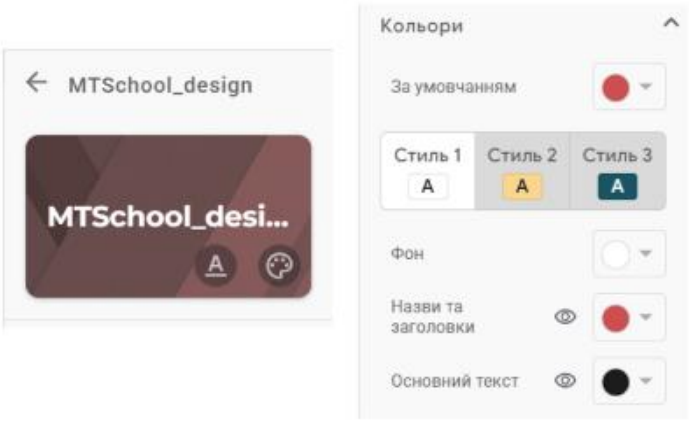

Рисунок 2.16 – Панель «Теми»: налаштування кольорів кастомної теми MTSchool_design

Принциповою перевагою конфігурації через кастомну тему є механізм глобального поширення: будь-яка зміна кольорового значення миттєво застосовується до всіх сторінок та компонентів, що використовують відповідний параметр. Це суттєво скорочує час на підтримку та редизайн системи в майбутньому.

Побудова контентних блоків

Після визначення глобального стилю розпочато наповнення сторінок. Побудову кожної сторінки здійснено за принципом «від структури до деталей»: спочатку формується загальна компоновка секцій шляхом послідовного додавання блоків-контейнерів, після чого кожен блок наповнюється конкретним контентом.

Головна сторінка організована як вертикальна послідовність тематичних секцій. Hero-секція реалізована як повноширинний банер: текстовий контент (заголовок, підзаголовок, слоган, CTA-елементи) розміщено в центральній колонці трьохколонкової сітки, тоді як бічні колонки містять декоративні PNG-зображення учнів із вирізаним фоном. Такий прийом «оживлює» статичний конструктор, надаючи йому вигляду кастомного дизайну. Нижче hero-секції послідовно розміщено блоки переваг, опису мов, форматів навчання та форму зворотного зв'язку (Рисунок 2.17).

Рисунок 2.17 – Робоча область редактора Google Sites з відкритою головною сторінкою в процесі редагування

Сторінки курсів побудовано за уніфікованим шаблоном із трьох зон. Верхня зона містить заголовок курсу та дві кнопки-CTA (перехід до безкоштовного тесту рівня та запис на пробний урок), реалізовані як кастомні HTML-компоненти. Нижня зона дублює форму зворотного зв'язку, що збільшує ймовірність конверсії відвідувача, що прочитав увесь контент сторінки. Блок контактів розміщено у правій колонці верхньої зони за принципом «завжди доступний контакт».

Застосування єдиного шаблону для всіх мовних сторінок (англійська, польська, німецька та їхні підрозділи) дозволило реалізувати шість внутрішніх сторінок курсів шляхом дублювання базової структури та заміни контенту, що знизило час розробки кожної наступної сторінки приблизно вдвічі порівняно з першою.

Верифікація адаптивного відображення

Завершальним етапом побудови кожної сторінки є перевірка адаптивного відображення в режимі попереднього перегляду редактора. Google Sites надає вбудований перемикач між трьома режимами: десктоп, планшет та смартфон, що доступний через нижню панель вікна попереднього перегляду. В режимі смартфона перевірено коректне складання всіх секцій у вертикальний стек, видимість навігаційної кнопки-гамбургера та читабельність тексту без горизонтального прокручування. Окрему увагу приділено поведінці iframe-компонентів: оскільки кастомні елементи виконуються у власній пісочниці, їх адаптивність залежить виключно від CSS-правил всередині компонента, а не від адаптивного движка Google Sites (Рисунок 2.18).

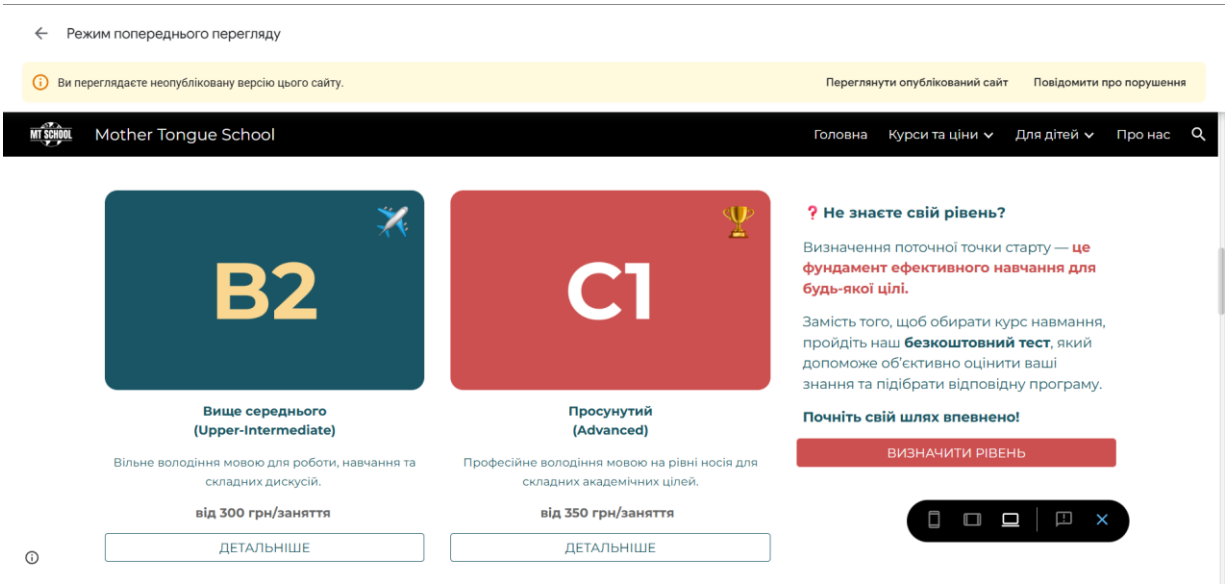

Рисунок 2.18 – Режим попереднього перегляду редактора Google Sites

Побудований таким чином веб-ресурс повністю відповідає затвердженому прототипу. Інтеграція з серверними компонентами — бекенд-логікою Apps Script та базою даних Google Sheets — описується в наступних підпунктах.

2.4.2 Розробка серверних сценаріїв автоматизації на базі GAS

Обмеження платформи Google Sites щодо відсутності власного серверного середовища, описане в підрозділі 2.1.4, вирішується підключенням Google Apps Script як зовнішнього бекенд-обробника. У межах цього підпункту розглядається структура проєкту Apps Script, логіка написаного коду та процедура розгортання скрипту як загальнодоступного веб-додатку.

Структура проєкту та глобальні змінні

Проєкт Apps Script організовано у вигляді єдиного файлу *Код.gs*. На відміну від традиційних серверних додатків, що потребують конфігурування маршрутизатора та middleware, весь необхідний функціонал реалізовано у мінімальній кількості рядків завдяки нативній інтеграції Apps Script з екосистемою Google [17].

На рівні глобальних змінних ініціалізовано з'єднання з цільовою таблицею Google Sheets. Клас *SpreadsheetApp* є вбудованим сервісом Apps Script для роботи з Google Sheets і не потребує імпорту зовнішніх бібліотек. Метод *openByUrl()* приймає повну URL-адресу документа та повертає об'єкт *Spreadsheet*, з якого методом *getSheetByName("sheet1")* отримується посилання на конкретний аркуш. Визначення цих змінних на глобальному рівні (поза функціями) дозволяє уникнути повторної ініціалізації з'єднання при кожному виклику функції-обробника.

Реалізація функції-обробника doPost(e)

Центральним елементом серверної логіки є функція *doPost(e)*, що автоматично викликається Apps Script при надходженні POST-запиту на URL розгорнутого веб-додатку. Параметр *e* є об'єктом події, що містить усі дані вхідного запиту.

```
const sheets = SpreadsheetApp.openByUrl(
  "https://docs.google.com/spreadsheets/d/1EqlVY4Y835dd...",
);
const sheet = sheets.getSheetByName("sheet1");

function doPost(e) {
  let data = e.parameter;
  sheet.appendRow([data.Name, data.Email, data.Phone, data.Message]);
  return ContentService.createTextOutput("Your message was
successfully sent!");
}
```

Рисунок 2.19 – Лістинг коду серверного обробника форми

Логіка функції реалізується у три послідовні операції

Перша — зчитування параметрів запиту через об'єкт *e.parameter*, що автоматично десеріалізує тіло POST-запиту у форматі *application/x-www-form-urlencoded* у JavaScript-об'єкт із парами «ключ–значення». Ключі відповідають атрибутам *name* HTML-полів форми (Name, Email, Phone, Message).

Друга операція — запис нового рядку в таблицю методом *appendRow()*, що приймає масив значень та автоматично додає їх до першого вільного рядку аркуша.

Третя — повернення текстової відповіді клієнту через *ContentService.createTextOutput()*, що підтверджує успішне виконання операції (Рисунок 2.20).

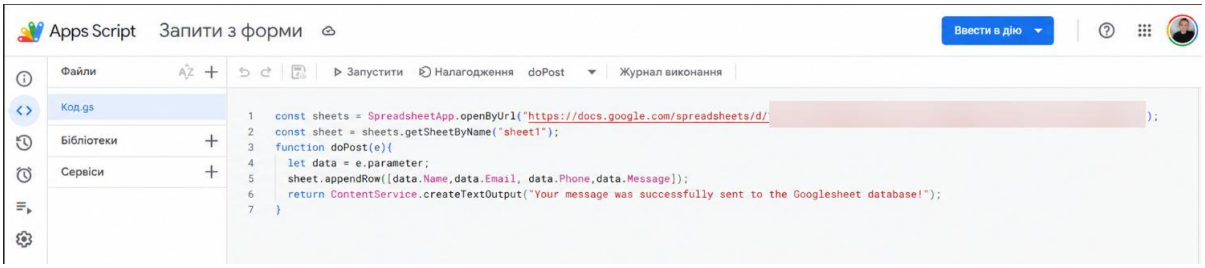

Рисунок 2.20 – Середовище розробки Google Apps Script з написаним кодом функції doPost(e)]

Розгортання скрипту як веб-додатку

Для того щоб функція *doPost(e)* стала доступною як HTTP-ендпоінт, скрипт необхідно розгорнути як веб-додаток. Процедура розгортання виконується через

меню «Ввести в дію» → «Нове введення в дію». У діалозі налаштувань визначено два критичні параметри: «Виконувати як» встановлено значення «Я (обліковий запис власника)», що надає скрипту права доступу до Google Sheets від імені власника; «Хто має доступ» встановлено значення «Усі», що дозволяє анонімним POST-запитам з форми на сайті досягати обробника без автентифікації (Рисунок 2.21).

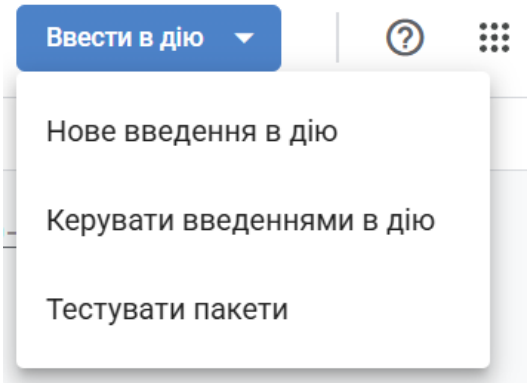

Рисунок 2.21 – Діалог розгортання Apps Script

Результатом розгортання є унікальний URL веб-додатку у форматі https://script.google.com/macros/s/{deployment_id}/exec. Саме цей URL використовується у JavaScript-коді форми як адреса для відправлення fetch()-запитів. При кожній зміні коду скрипту необхідно створювати нове розгортання — оновлення поточного призведе до зміни URL.

Авторизація OAuth

При першому виконанні скрипт запитує дозвіл на доступ до Google Sheets через стандартний механізм Google OAuth. Адміністратор підтверджує надання прав у діалоговому вікні, що з'являється автоматично. Після одноразової авторизації скрипт виконується в автономному режимі без подальшої участі користувача.

2.4.3 Інтеграція фронтенд-частини з базою даних у Google Sheets

Розгорнутий у підпункті 2.4.2 Apps Script Web App є сполучною ланкою між клієнтською формою на сайті та хмарною базою даних. Цей підпункт описує

структуру бази даних, механізм передачі даних з форми до серверного обробника та практичну верифікацію цілісності інтеграції.

Структура бази даних у Google Sheets

Google Sheets виконує функцію реляційного сховища заявок. Цільовий аркуш sheet1 організовано у вигляді плоскої таблиці з чотирма колонками, що відповідають полям форми зворотного зв'язку. Схему бази даних наведено в таблиці 2.1.

Таблиця 2.1 – Структура аркуша бази даних заявок

Колонка	Назва поля	Тип даних	Джерело
A	Name	Рядок	input[name="Name"]
B	Email	Рядок	input[name="Email"]
C	Phone	Рядок	input[name="Phone"]
D	Message	Рядок	textarea[name="Message"]

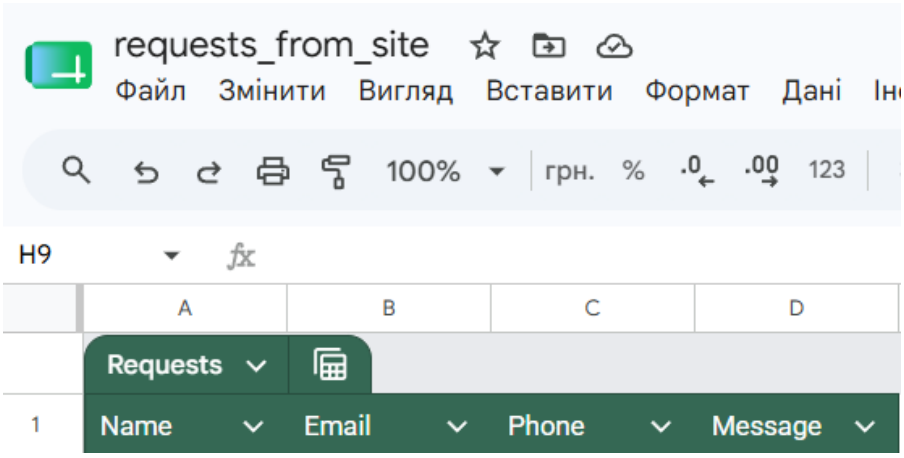

Рисунок 2.22 – Google Sheets: аркуш sheet1 з налаштованими заголовками колонок

Відповідність між атрибутами *name* HTML-полів форми та колонками таблиці є ключовим архітектурним рішенням інтеграції. Оскільки об'єкт *e.parameter* в Apps Script автоматично індексується за ключами, що збігаються з іменами полів форми, будь-яка зміна атрибута *name* в HTML без відповідного оновлення коду серверного обробника призведе до запису порожніх значень у

відповідну колонку. Тому схема іменування полів зафіксована як контракт між фронтенд- та бекенд-компонентами системи.

Реалізація клієнтської частини передачі даних

Відправлення даних з форми на URL веб-додатку Apps Script реалізовано засобами нативного Fetch API. Ключовим елементом є використання об'єкта *FormData*, що автоматично серіалізує всі поля форми у формат *multipart/form-data*. Функція *doPost(e)* Apps Script коректно опрацьовує обидва формати тіла POST-запиту — як *application/x-www-form-urlencoded*, так і *multipart/form-data* — через єдиний інтерфейс *e.parameter*.

```
const scriptURL =
  'https://script.google.com/macros/s/{id}/exec';
const form = document.getElementById('courseForm');

form.addEventListener('submit', e => {
  e.preventDefault();
  btn.disabled = true;
  btn.innerText = 'Відправляємо...';

  fetch(scriptURL, {
    method: 'POST',
    body: new FormData(form)
  })
  .then(response => { /* обробка успіху */ })
  .catch(error => { /* обробка помилки */ });
});;
}
```

Рисунок 2.23 – Лістинг клієнтського JavaScript-коду відправлення даних форми

Виклик *e.preventDefault()* перехоплює стандартну поведінку браузера при відправленні форми (перезавантаження сторінки) та передає управління JavaScript-обробнику. Блокування кнопки (*btn.disabled = true*) одразу після ініціювання запиту є механізмом захисту від дублювання заявок при повторному натисканні. Асинхронна природа *fetch()* забезпечує неблокуючу взаємодію з сервером: інтерфейс залишається відповідним під час очікування відповіді.

Верифікація інтеграції на тестових даних

Практичне підтвердження коректності інтеграції здійснено шляхом заповнення форми тестовими даними та перевірки їх появи в таблиці Google Sheets. Тестовий запис успішно з'явився у рядку аркуша sheet1 (Рисунок 2.24) одразу після відправлення форми, що підтверджує коректне виконання всього ланцюжка: HTML-форма → fetch() POST → Apps Script doPost() → Sheets appendRow().

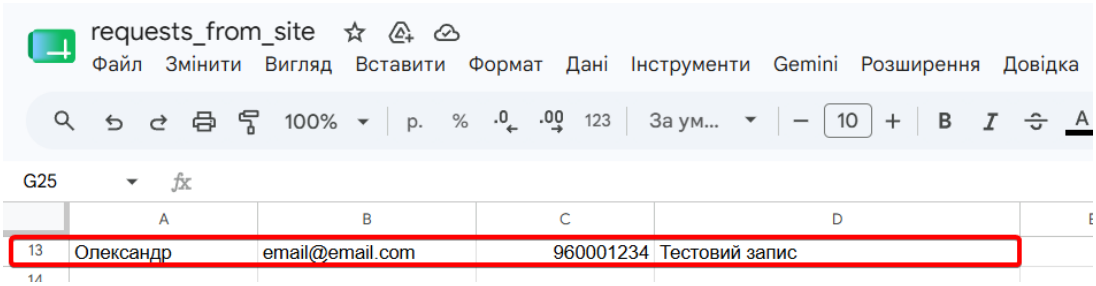

Рисунок 2.24 – Google Sheets: аркуш з тестовими даними, що надійшли після заповнення форми

Суттєвою перевагою використання Google Sheets як бази даних є нативна доступність для нетехнічного персоналу: адміністратор школи отримує нові заявки у знайомому інтерфейсі таблиці в режимі реального часу, без необхідності входу до будь-якої адміністративної панелі. Це відповідає вимозі самостійного адміністрування системи, визначеній у постановці задачі підрозділу 1.3.2.

Рисунок 2.25 – Форма зворотного зв'язку після успішного відправлення

2.4.4 Розширення функціональності засобами HTML, CSS, JavaScript

Стандартний інструментарій Google Sites, розглянутий у підрозділі 2.1.1, забезпечує базову компоновку інформаційних сторінок, проте є недостатнім для реалізації інтерактивних елементів із кастомною бізнес-логікою: форми зворотного зв'язку з асинхронною відправкою, анімованих кнопок із hover-ефектами та динамічної зміни стану інтерфейсу. Для вирішення цього завдання використано

механізм вставки HTML-коду (Embed), що дозволяє розмістити довільний HTML/CSS/JavaScript-компонент в ізольованій iframe-пісочниці [18].

Архітектура кастомних компонентів

У межах проєкту розроблено два типи кастомних HTML-компонентів: форма зворотного зв'язку та кнопка переходу до зовнішнього ресурсу. Обидва компоненти є самодостатніми одиницями: кожен містить власні стилі, розмітку та логіку, що не залежать від батьківської сторінки Google Sites. Така архітектура обумовлена sandbox-обмеженнями iframe, описаними в підрозділі 2.1.4: компонент не може звертатися до DOM батьківської сторінки, тому весь необхідний функціонал реалізовано всередині iframe [16].

Впровадження компонентів здійснюється через діалог «Вставити» → «Вставити код» у редакторі Google Sites. Після підтвердження HTML-код рендериться у вбудованому iframe, розміри якого можна коригувати безпосередньо у редакторі (Рисунок 2.26).

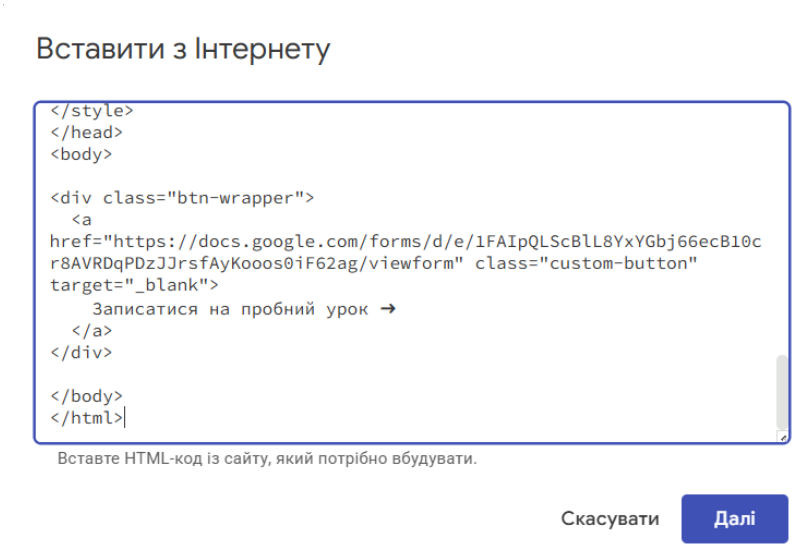

Рисунок 2.26 – Діалог «Вставити код» у Google Sites

HTML-структура та CSS-стилізація форми

Форма зворотного зв'язку організована навколо контейнера *.form-card* із заокругленими кутами (border-radius: 20px) та декоративною рамкою кольору #fad791, що візуально відокремлює форму від основного темно-зеленого фону

сторінки. Кольорова схема компонента (#fffaf0 — фон картки, #cd5050 — заголовок, #195564 — кнопка та акцент) відтворює фірмову палітру школи у теплішому, більш «запрошуючому» регістрі, що психологічно знижує бар'єр до заповнення форми.

```
.form-card {
  background-color: #fffaf0;
  border-radius: 20px;
  border: 2px solid #fad791;
  box-shadow: 0 8px 25px rgba(25, 85, 100, 0.1);
  padding: 30px 20px;
}
button {
  background-color: #195564;
  border-radius: 50px;
  transition:
    background-color 0.3s ease,
    transform 0.2s ease;
}
button:hover {
  background-color: #cd5050;
}
button:active {
  transform: scale(0.98);
}
```

Рисунок 2.27 – Лістинг CSS-стилів ключових елементів форми

CSS-властивість *transition* застосована до кнопки для плавної зміни кольору фону (0.3s ease) при наведенні курсора та масштабування (0.2s ease) при натисканні. Ці мікроанімації є важливим елементом UX: вони підтверджують користувачу, що елемент є інтерактивним. Правило *@media (max-width: 360px)* забезпечує коректне відображення на найвужчих екранах смартфонів — зменшується розмір шрифту заголовка та відступи картки.

JavaScript-логіка управління станами форми

Форма реалізує чотири візуальні стани, що забезпечують повноцінний зворотний зв'язок із користувачем:

Стан 1. «Очікування» (початковий): всі поля активні, кнопка відображає текст «Записатися на консультацію».

Стан 2. «Відправлення»: кнопка переходить у стан *disabled* та змінює текст на «Відправляємо...», що унеможливлює повторне відправлення під час обробки запиту.

Стан 3. «Успіх» *(.then()-гілка*): блок *#status* стає видимим із повідомленням про успіх поля форми скидаються методом *form.reset()*, кнопка розблоковується.

Стан 4. «Помилка» (*.catch()-гілка*): блок *#status* відображає повідомлення помилки, кнопка розблоковується зі зміненим текстом «Спробувати знову».

Кастомна кнопка-посилання

Другим кастомним компонентом є кнопка переходу на зовнішній ресурс — Google Forms для проходження безкоштовного тесту рівня. Стандартна кнопка Google Sites не підтримує hover-ефекти та обмежена стилями активної теми, тоді як кастомний компонент реалізує жовто-золотистий фон (#fad791), що змінюється на червоний (#cc5050) при наведенні з плавним переходом *translateY(-2px)* — ефектом «підняття» кнопки.

```
.custom-button {
  background-color: #fad791;
  color: #1e5767;
  border-radius: 50px;
  transition: all 0.3s ease;
}
.custom-button:hover {
  background-color: #cc5050;
  color: #f2f2f2;
  transform: translateY(-2px);
};
}
```

Рисунок 2.28 – Лістинг CSS-стилів кнопки-посилання

Елемент реалізовано як тег *<a>* із стилізацією під кнопку, що є семантично коректним підходом для посилань: атрибут *target="_blank"* забезпечує відкриття форми у новій вкладці, не перериваючи перегляд основного сайту.

```
<div class="btn-wrapper">
  <a href="https://docs.google.com/forms/d/e/..."
     class="custom-button"
     target="_blank">
    Записатися на пробний урок →
  </a>
</div>
};
}
```

Рисунок 2.29 – Лістинг HTML-розмітки кнопки-посилання

Обгортковий контейнер *.btn-wrapper* із властивостями *display: inline-block* та width: 100% забезпечує коректне центрування кнопки в межах iframe-пісочниці незалежно від ширини блоку, у який вставлено компонент. Атрибут *rel="noopener"* рекомендується додавати до посилань із *target="_blank"* з міркувань безпеки, однак у межах iframe з обмеженнями CSP Google Sites цей ризик мінімізовано на рівні платформи (Рисунок 2.30).

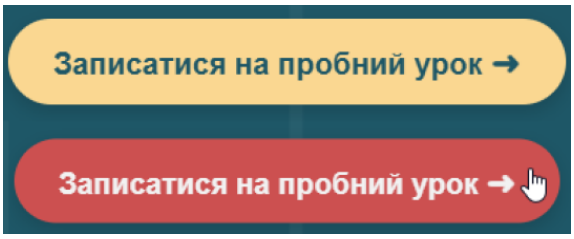

Рисунок 2.30 – Кнопка «Записатися на пробний урок» у звичайному стані та при наведенні

Реалізовані HTML/CSS/JS компоненти демонструють практичне застосування гібридного підходу: обмеження платформи Google Sites компенсуються цілеспрямованим написанням коду там, де стандартних засобів недостатньо, зберігаючи при цьому всі переваги serverless-хостингу та автоматичного масштабування.

2.5 Конфігурація мережевої інфраструктури та топологія розгортання

Завершивши розробку фронтенд- та бекенд-компонентів системи, необхідно забезпечити її публічну доступність в глобальній мережі. Цей підрозділ охоплює повний цикл мережевого розгортання: від реєстрації та делегування доменного імені до налаштування захищеного з'єднання та верифікації коректної роботи всіх складових. Конфігурація виконується у три послідовні етапи, кожен із яких є необхідною умовою для переходу до наступного.

2.5.1 Впровадження CDN-сервісу Cloudflare та конфігурація DNS-інфраструктури

Топологія розгортання веб-ресурсу Mother Tongue School будується на взаємодії трьох незалежних сервісів: реєстратора доменних імен TheHost, мережевого посередника Cloudflare та хостинг-платформи Google Sites. Розподіл відповідальності між ними відображено на діаграмі розгортання (Рисунок 2.31).

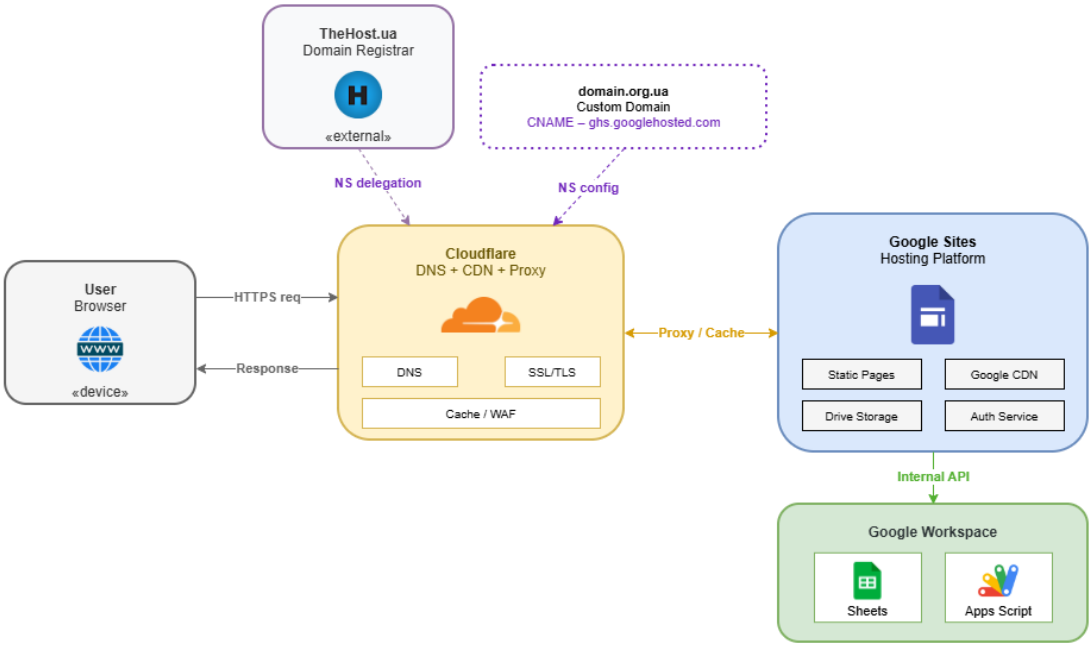

Рисунок 2.31 – Deployment Diagram веб-ресурсу Mother Tongue School

Відповідно до діаграми, запит користувача проходить три рівні обробки: DNS-резолюція через Cloudflare, SSL-термінація та CDN-кешування на edge-вузлах

Cloudflare, і нарешті — доставка контенту з платформи Google Sites через Google CDN. Така архітектура забезпечує наскрізний захист з'єднання та мінімізує затримку для кінцевого користувача.

Реєстрація домену та делегування NS

Доменне ім'я mtschool.org.ua зареєстровано через українського реєстратора TheHost. Після успішної реєстрації в панелі керування TheHost виконано делегування DNS-керування зоною на сервери імен Cloudflare. Для цього в налаштуваннях домену замінено стандартні NS-записи реєстратора на два записи Cloudflare: meiling.ns.cloudflare.com та pranab.ns.cloudflare.com (Рисунок 2.32).

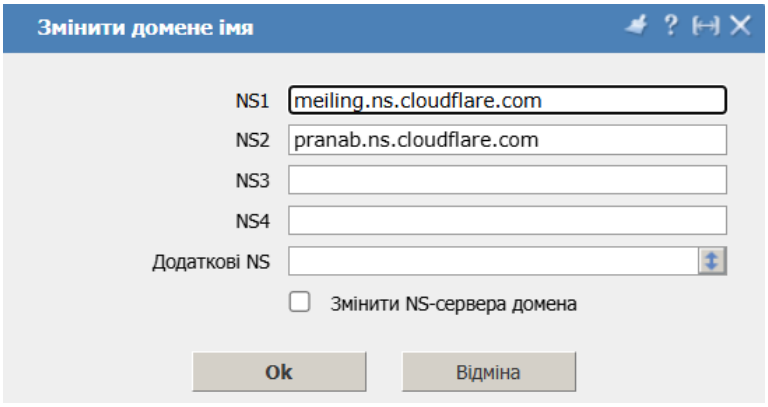

Рисунок 2.32 – Панель керування TheHost: діалог зміни NS-серверів на сервери Cloudflare

Делегування NS-серверів є критичним кроком: воно передає повне керування DNS-зоною домену від реєстратора до Cloudflare. Після цього всі DNS-записи домену управляються виключно через інтерфейс Cloudflare, незалежно від панелі TheHost. Час поширення змін NS-записів у глобальній DNS-системі становить від кількох хвилин до 48 годин, проте на практиці для домену mtschool.org.ua зміни набули чинності протягом приблизно однієї години.

Отримання NS-серверів у Cloudflare

Після додавання домену в панелі Cloudflare система автоматично призначає унікальну пару NS-серверів для зони. Для зони mtschool.org.ua призначено сервери meiling.ns.cloudflare.com та pranab.ns.cloudflare.com — саме ці значення вносяться в панель TheHost на попередньому кроці (Рисунок 2.33).

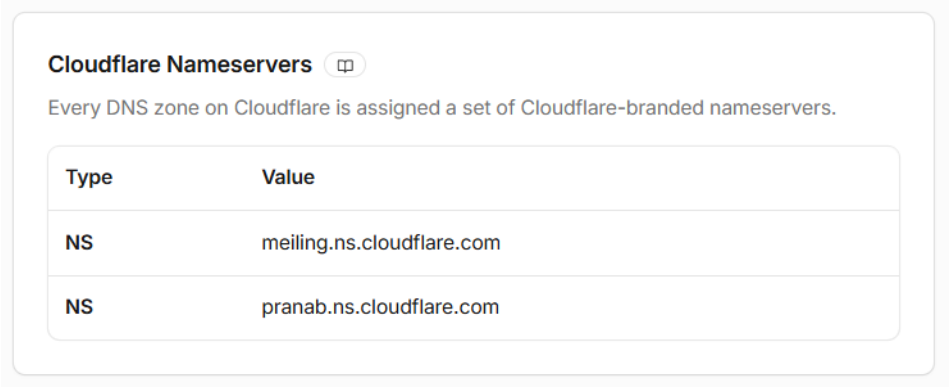

Рисунок 2.33 – Cloudflare: призначені NS-сервери для зони mtschool.org.ua

Конфігурація DNS-записів

Після активації зони в Cloudflare налаштовано три типи DNS-записів, кожен із яких виконує окрему функцію у загальній архітектурі (Таблиця 2.2 – DNS-записи зони mtschool.org.ua у CloudflareТаблиця 2.2).

Таблиця 2.2 – DNS-записи зони mtschool.org.ua у Cloudflare

Тип	Ім'я	Вміст	Proxy	Призначення
A	mtschool	91.234.35.6	Proxied	Кореневий домен → IP TheHost
CNAME	www	ghs.googlehosted.com	Proxied	Піддомен www → Google Sites
TXT	mtschool	google-site-verification=...	DNS only	Верифікація власника домену

Запис типу A для кореневого домену mtschool.org.ua вказує на IP-адресу 91.234.35.6, що належить інфраструктурі TheHost. Цей запис є технічно необхідним, оскільки Google Sites не підтримує пряму прив'язку кореневого домену — обмеження, описане в підрозділі 2.1.4. Реальна переадресація з кореневого домену на www вирішується через правило Cloudflare Page Rules, що детально описується в підрозділі 2.5.3.

Запис типу CNAME для піддомену www вказує на адресу ghs.googlehosted.com — стандартний хост Google Sites для кастомних доменів.

Саме цей запис забезпечує фактичне відображення сайту за адресою www.mtschool.org.ua. Статус «Proxied» (помаранчева хмаринка Cloudflare) означає, що трафік проходить через мережу Cloudflare, отримуючи переваги CDN-кешування та WAF-захисту.

Запис типу TXT із значенням google-site-verification=tJmOJYiwHzPccpgDi5Meuq… слугує токеном підтвердження права власності на домен для Google. Статус «DNS only» означає, що цей запис не проксується через Cloudflare — Google зчитує його безпосередньо, минаючи посередника (Рисунок 2.34).

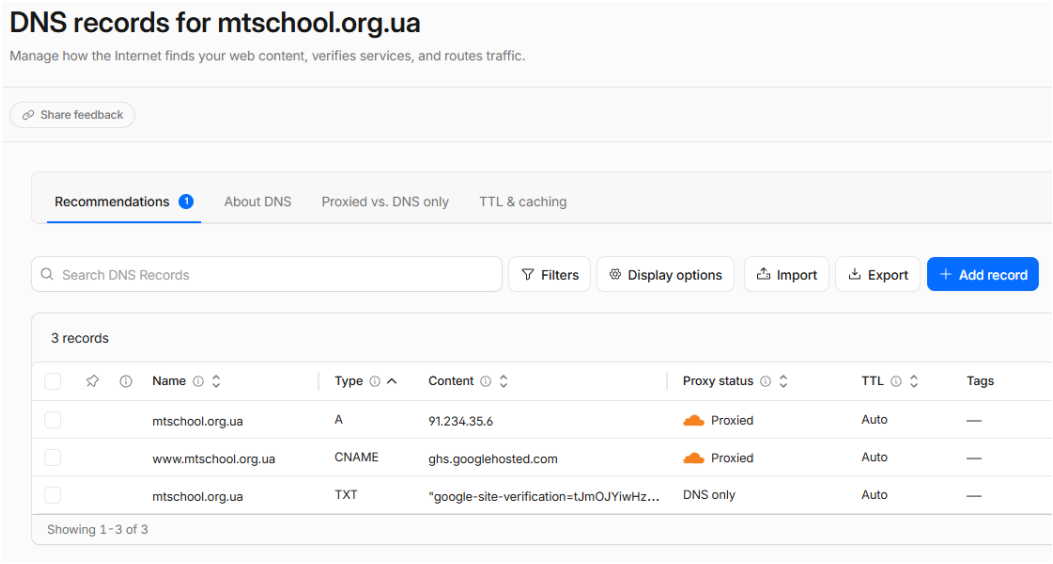

Рисунок 2.34 – Cloudflare: таблиця DNS-записів зони mtschool.org.ua

Верифікація домену в Google Sites

Після додавання TXT-запису в Cloudflare виконано верифікацію права власності на домен через інтерфейс Google Sites. Процедура складається з чотирьох кроків: вибір типу запису TXT, копіювання згенерованого токена верифікації, його додавання до DNS-зони та натискання кнопки «Перевірити». Google Sites самостійно опитує DNS-зону та підтверджує наявність запису (Рисунок 2.35).

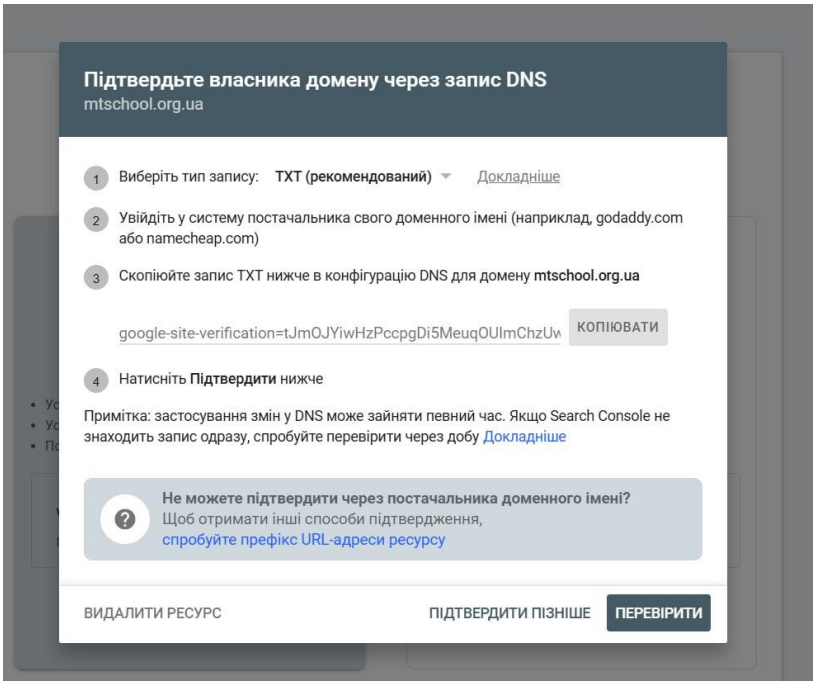

Рисунок 2.35 – Google Sites: діалог верифікації права власності на домен через TXT-запис

Успішна верифікація є обов'язковою умовою для подальшої прив'язки домену до опублікованого сайту. Без підтвердження Google не дозволяє встановити відповідність між кастомним доменом та ресурсом Google Sites.

2.5.2 Налаштування SSL/TLS-шифрування та засобів захисту від мережевих загроз

Підтвердивши право власності на домен та сконфігурувавши DNS-записи, необхідно встановити режим шифрування трафіку між усіма вузлами системи. Cloudflare пропонує чотири режими SSL/TLS, вибір між якими суттєво впливає на рівень безпеки з'єднання.

Порівняння режимів SSL/TLS та обґрунтування вибору

Режим Off повністю вимикає шифрування — трафік між браузером і сервером передається у відкритому вигляді. Цей варіант неприйнятний для будь-якого сайту, що збирає персональні дані.

Режим Flexible шифрує лише з'єднання між браузером користувача та edge-сервером Cloudflare. З'єднання між Cloudflare та origin-сервером (Google Sites)

залишається незахищеним HTTP. Попри наявність замка в адресному рядку браузера, цей режим створює вразливість на ділянці Cloudflare → Google Sites.

Режим Full забезпечує наскрізне шифрування: і браузер↔Cloudflare, і Cloudflare↔Google Sites з'єднання захищені HTTPS. Вимагає, щоб origin-сервер підтримував SSL, але не перевіряє валідність сертифіката.

Режим Full (Strict) додає до Full обов'язкову валідацію сертифіката origin-сервера через довірений центр сертифікації. Є найбезпечнішим варіантом, проте потребує встановлення Origin CA-сертифіката.

Для проєкту Mother Tongue School обрано режим Full, оскільки Google Sites як origin-сервер підтримує HTTPS та надає власний сертифікат, проте не через Cloudflare Origin CA. Режим Full забезпечує наскрізне шифрування без необхідності встановлення додаткових сертифікатів, що оптимально для serverless-платформи без прямого доступу до серверної конфігурації (Рисунок 2.36).

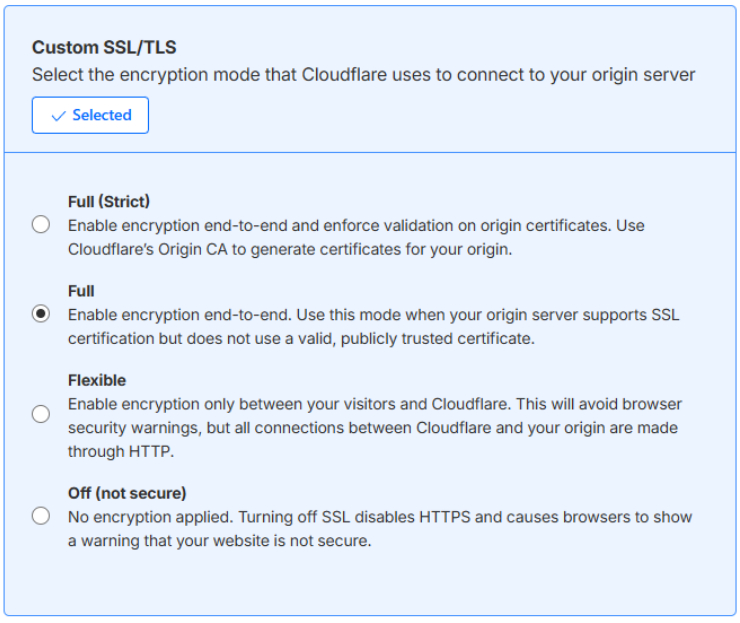

Рисунок 2.36 – Cloudflare: вибір режиму SSL/TLS

Статистика трафіку за протоколами TLS

Панель SSL/TLS у Cloudflare відображає розподіл з'єднань за версіями протоколу за останні 24 години. Отримані дані для зони mtschool.org.ua наведено нижче.

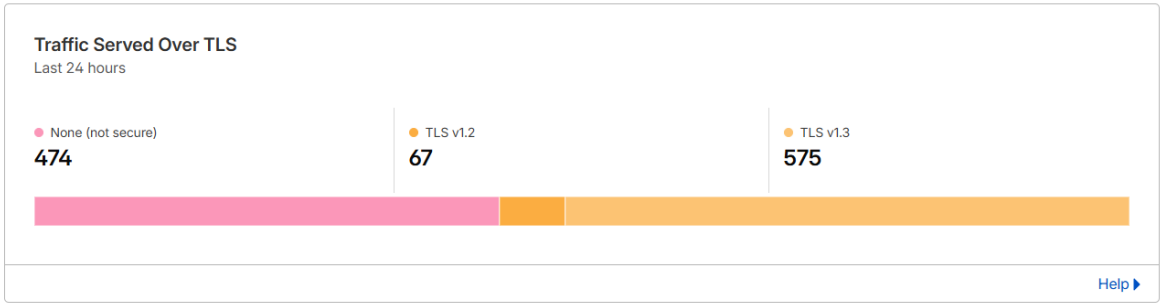

Рисунок 2.37 – Розподіл трафіку за протоколами TLS (останні 24 год)

Незахищені з'єднання (474 запитів) відповідають запитам до кореневого домену http://mtschool.org.ua до їх переадресації на https://www.mtschool.org.ua через Page Rule. Після впровадження редиректу частка незахищеного трафіку зменшується до нуля на рівні кінцевого користувача, оскільки Cloudflare виконує переадресацію ще на своїх edge-вузлах. Переважання TLS v1.3 (54%) є позитивним показником: ця версія протоколу є актуальною і забезпечує вищу швидкість встановлення з'єднання (1-RTT handshake) порівняно з TLS v1.2.

Захист на рівні мережевої інфраструктури

Статус «Proxied» для DNS-записів A та CNAME, встановлений у підпункті 2.5.1, автоматично активує весь захисний стек Cloudflare. По-перше, реальна IP-адреса origin-сервера прихована від зовнішніх спостерігачів: DNS-запит на www.mtschool.org.ua повертає IP Cloudflare edge-вузла, а не адресу Google Sites. Це унеможливлює прямі атаки на origin, минаючи Cloudflare. По-друге, активується вбудований Web Application Firewall (WAF), що фільтрує шкідливий трафік за сигнатурами відомих атак (SQL injection, XSS, bot-трафік) ще до його досягнення origin-сервера. По-третє, автоматичний DDoS-захист Layer 3/4 поглинає об'ємні атаки на рівні мережевого транспорту без участі розробника.

Для малого освітнього бізнесу особливо важливим є те, що описані засоби захисту надаються у рамках безкоштовного тарифного плану Cloudflare Free та не потребують окремого налаштування після активації проксування.

2.5.3 Процес публікації, прив'язка домену та верифікація працездатності системи

Маючи налаштовані DNS-записи та активоване SSL-шифрування, можна переходити до фінального етапу розгортання: публікації сайту в Google Sites з прив'язкою кастомного домену, налаштування переадресації кореневого домену та комплексної верифікації працездатності всієї системи.

Публікація сайту та прив'язка домену в Google Sites

Публікація сайту виконується через кнопку «Опублікувати» в редакторі Google Sites. У діалозі публікації вказано кастомний домен www.mtschool.org.ua. Google Sites автоматично перевіряє наявність відповідного CNAME-запису в DNS та підтверджує прив'язку. Після успішної публікації сайт стає доступним за адресою www.mtschool.org.ua, а всі подальші зміни контенту публікуються натисканням тієї ж кнопки без необхідності повторного налаштування домену.

Розгортання Apps Script Web App, здійснене в підпункті 2.4.2, є незалежним від публікації Google Sites і залишається активним з моменту першого розгортання. URL ендпоінту не змінюється при оновленні контенту сайту.

Налаштування переадресації кореневого домену

Як зазначено в підпункті 2.1.4, Google Sites підтримує прив'язку лише піддомену www, тому запит до кореневого домену mtschool.org.ua без переадресації завершується помилкою.

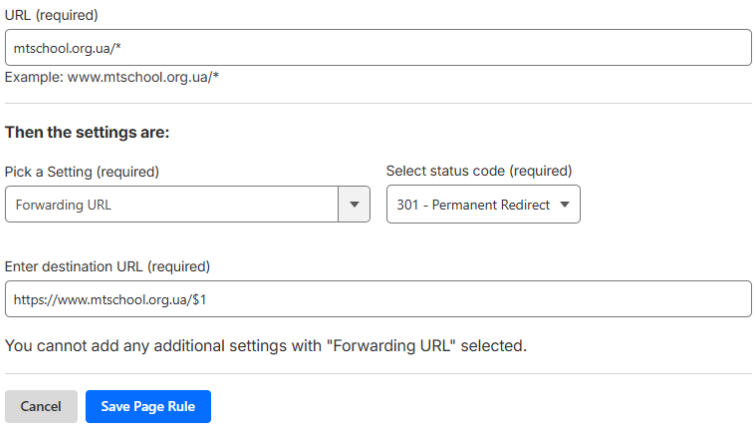

Рисунок 2.38 – Cloudflare Page Rules: налаштування правила 301-переадресації

Для вирішення цього обмеження в Cloudflare створено правило Page Rule із такими параметрами: шаблон URL — mtschool.org.ua/*; дія — «Forwarding URL»; тип переадресації — «301 — Permanent Redirect»; цільова адреса — https://www.mtschool.org.ua/$1

Символ $1 у цільовій URL є груповим захоплювачем шляху: він забезпечує збереження будь-якого суфікса після кореневого домену при переадресації. Наприклад, запит mtschool.org.ua/courses автоматично переадресовується на https://www.mtschool.org.ua/courses. Код 301 (Permanent Redirect) повідомляє пошуковим системам про постійний характер переадресації, що дозволяє передати SEO-вагу з кореневого домену на www-версію (Рисунок 2.39).

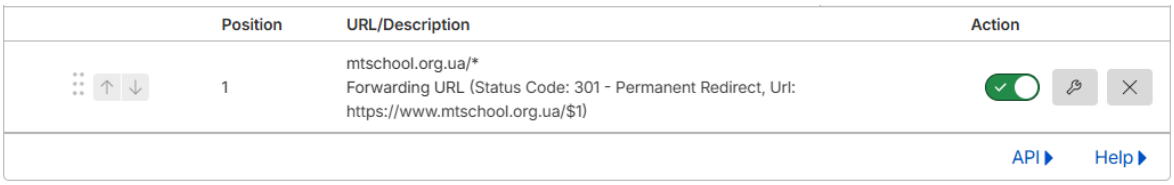

Рисунок 2.39 – Cloudflare Page Rules: активне правило переадресації

Верифікація поширення DNS та поведінки редиректів

Для підтвердження коректного поширення DNS-записів у глобальній мережі використано сервіс dnschecker.org. (Рисунок 2.40).

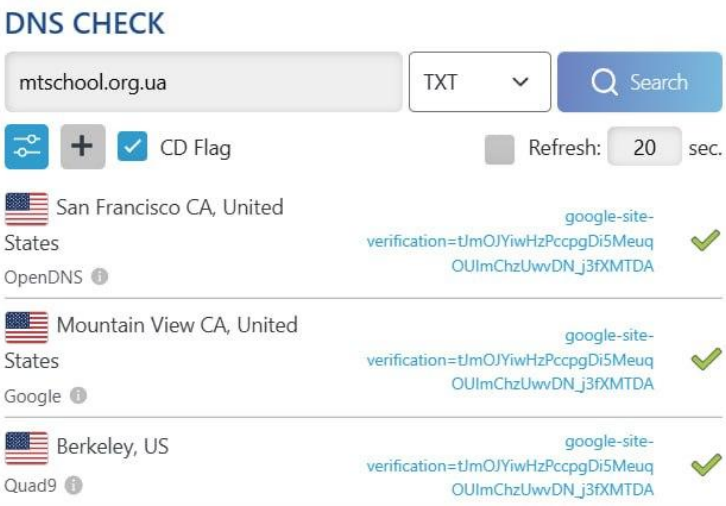

Рисунок 2.40 – dnschecker.org: глобальне поширення TXT-запису

Перевірка TXT-запису mtschool.org.ua підтвердила наявність токена google-site-verification на всіх перевірених DNS-резолверах: OpenDNS (San Francisco), Google (Mountain View), Quad9 (Berkeley, San Francisco), CenturyLink (United States) та Daniel Cid (Columbia) — усі вузли повернули коректне значення з позначкою успішної перевірки.

Перевірка NS-записів підтвердила коректне делегування зони: усі перевірені резолвери повертають пару meiling.ns.cloudflare.com та pranab.ns.cloudflare.com, що свідчить про повне поширення змін NS у глобальній DNS-системі (Рисунок 2.41).

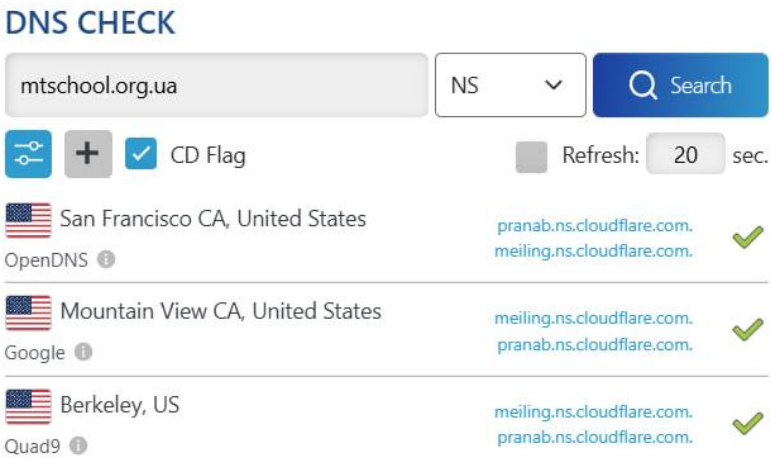

Рисунок 2.41 – dnschecker.org: глобальне поширення NS-записів mtschool

Фінальним кроком верифікації є перевірка поведінки переадресацій для всіх можливих варіантів вхідної URL. До впровадження Page Rule запити http://mtschool.org.ua та http://www.mtschool.org.ua повертали 301→200, https://www.mtschool.org.ua повертав 200 (коректно), а https://mtschool.org.ua завершувався помилкою. Після збереження правила переадресації всі чотири варіанти URL обробляються коректно: HTTP-варіанти виконують ланцюжок 301→200, а HTTPS-кореневий домен перенаправляється на www із кодом 301, після чого основна адреса повертає 200 (Рисунок 2.42).

Request URL	Status codes	Redirects
http://mtschool.org.ua	301 200	1
https://mtschool.org.ua	301 200	1
http://www.mtschool.org.ua	301 200	1
https://www.mtschool.org.ua	200	0

Рисунок 2.42 – Поведінка URL після впровадження Page Rule

Успішне проходження всіх перевірок підтверджує повну функціональність розгорнутої системи: доменне ім'я коректно резолюється в усіх географічних зонах, SSL-шифрування активне на всіх ділянках маршруту, кореневий домен перенаправляється на канонічну www-адресу, а сайт доступний за єдиною стабільною URL: https://www.mtschool.org.ua.

2.6 Верифікація результатів та оцінка технічної якості розробленого рішення

Завершальним етапом розробки є об'єктивна оцінка технічної якості створеного веб-ресурсу. Для цієї мети обрано інструмент Google Lighthouse — вбудований аналізатор браузера Chrome, що є галузевим стандартом верифікації веб-застосунків. Lighthouse виконує комплексний аудит сторінки за чотирма незалежними категоріями: ефективність завантаження (Performance), доступність (Accessibility), відповідність оптимальним методам розробки (Best Practices) та оптимізація для пошукових систем (SEO). Кожна категорія оцінюється за шкалою від 0 до 100 балів.

Аудит проведено у режимі «Навігація (за умовчанням)» безпосередньо через вкладку Lighthouse у панелі інструментів розробника браузера Chrome (DevTools). Тестування виконувалось на виробничій версії сайту за адресою https://www.mtschool.org.ua/ з імітацією мережевих умов середнього мобільного з'єднання, що є стандартною методологією для оцінки реального досвіду користувача (Рисунок 2.43).

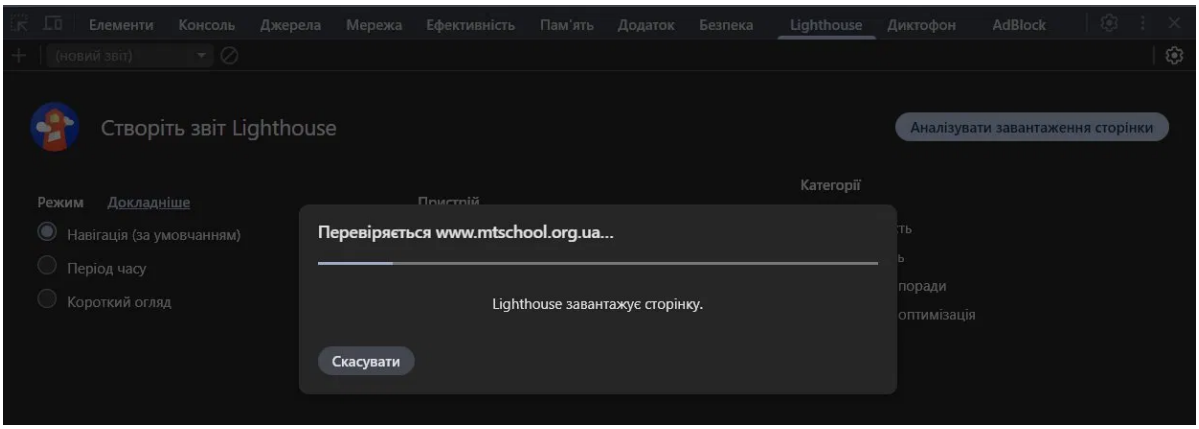

Рисунок 2.43 – Google Lighthouse: процес аналізу сторінки

Загальні результати аудиту

За результатами аналізу веб-ресурс отримав такі оцінки: Ефективність — 87 балів, Доступність — 100 балів, Оптимальні методи — 96 балів, Оптимізація пошукових систем — 92 бали (Рисунок 2.44).

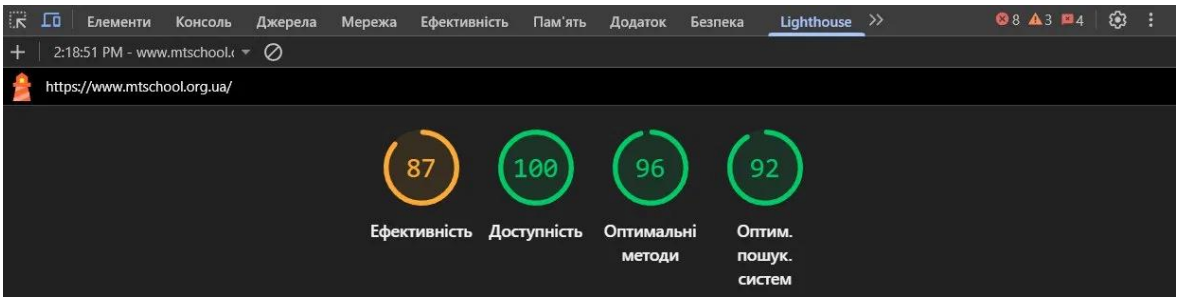

Рисунок 2.44 – Google Lighthouse: зведені результати аудиту

Максимальна оцінка 100 балів у категорії «Доступність» є особливо значущим результатом. Вона свідчить про повну відповідність ресурсу стандартам WCAG: усі зображення мають текстові альтернативи (атрибут alt), контрастність кольорів відповідає нормативним вимогам, інтерактивні елементи коректно опрацьовуються допоміжними технологіями (screen readers), а структура заголовків відповідає семантичній ієрархії. Це є прямим наслідком використання платформи Google Sites, яка генерує семантично коректну HTML-розмітку без необхідності ручного втручання розробника.

Оцінка 96 балів за категорією «Оптимальні методи» підтверджує відповідність реалізації сучасним стандартам веб-розробки: HTTPS активний на всіх ресурсах сторінки, відсутні виклики застарілих API браузера, JavaScript-код не містить відомих вразливостей, а сторінка коректно обробляється у різних браузерних середовищах. Зниження від максимуму на 4 бали пов'язане з характеристиками iframe-компонентів, вбудованих через механізм Embed, — незначними відхиленнями, що не впливають на функціональність системи.

Показник SEO на рівні 92 балів свідчить про те, що сайт коректно індексується пошуковими системами: мета-теги title присутні та описові, структура URL зрозуміла, сторінки доступні для сканування роботами. Незначне відхилення від максимуму зумовлено обмеженнями платформи Google Sites щодо гнучкого керування метаданими на рівні окремих сторінок, що було зазначено в підрозділі 2.1.4.

Аналіз показника ефективності

Оцінка ефективності в 87 балів є найбільш інформативним результатом аудиту і потребує детального розгляду. Lighthouse розраховує цей показник на основі п'яти зважених метрик веб-продуктивності: First Contentful Paint (FCP), Speed Index (SI), Largest Contentful Paint (LCP), Total Blocking Time (TBT) та Cumulative Layout Shift (CLS) (Рисунок 2.45).

Рисунок 2.45 – Google Lighthouse: деталізація показника ефективності

Серед локальних метрик особливу увагу привертають три показники. Метрика Largest Contentful Paint (LCP) зафіксована на рівні 1,62 секунди, що Lighthouse класифікує як «високе значення для локального показника» і що є прийнятним результатом, однак залишає простір для оптимізації. LCP відображає час до завантаження найбільшого видимого елемента сторінки (у даному випадку — div.IFu0kc, що відповідає hero-секції з великими PNG-зображеннями учнів). Cumulative Layout Shift (CLS) становить 0,01 — чудовий результат, що означає практичну відсутність небажаних зміщень елементів сторінки під час завантаження. Interaction to Next Paint (INP) на рівні 56 мс підтверджує високу реактивність інтерфейсу у відповідь на дії користувача (Рисунок 2.46).

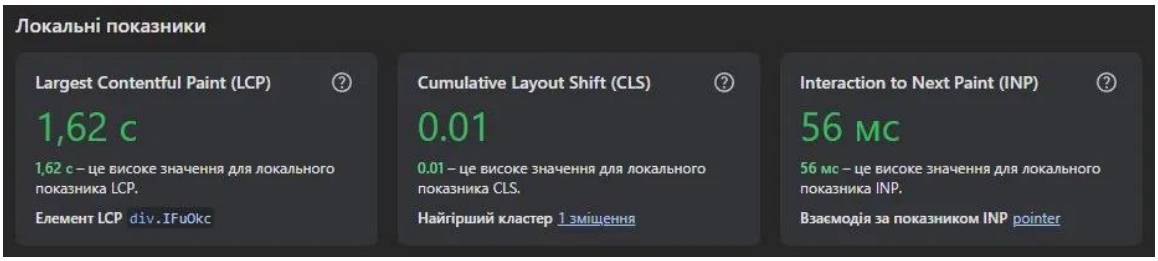

Рисунок 2.46 – Google Lighthouse: локальні показники LCP, CLS та INP

Діагностика та напрями подальшої оптимізації

Розділ статистики Lighthouse виявив кілька конкретних можливостей для покращення продуктивності. Найбільший потенціал заощадження — 14 239 КіБ — досягається шляхом оптимізації зображень: конвертація PNG-файлів hero-секції у сучасні формати WebP або AVIF та налаштування адаптивного завантаження через атрибут srcset суттєво скоротить обсяг переданих даних. Другий значущий резерв — підвищення ефективності TTL кешування (очікуване заощадження 9 645 КіБ): збільшення терміну зберігання статичних ресурсів у кеші Cloudflare зменшить кількість повторних запитів до origin-сервера. Додатково виявлено запити, що блокують відображення сторінки з очікуваним заощадженням 350 мс — переважно це стосується завантаження шрифту Montserrat через Google Fonts без атрибута font-display: swap (Рисунок 2.47).

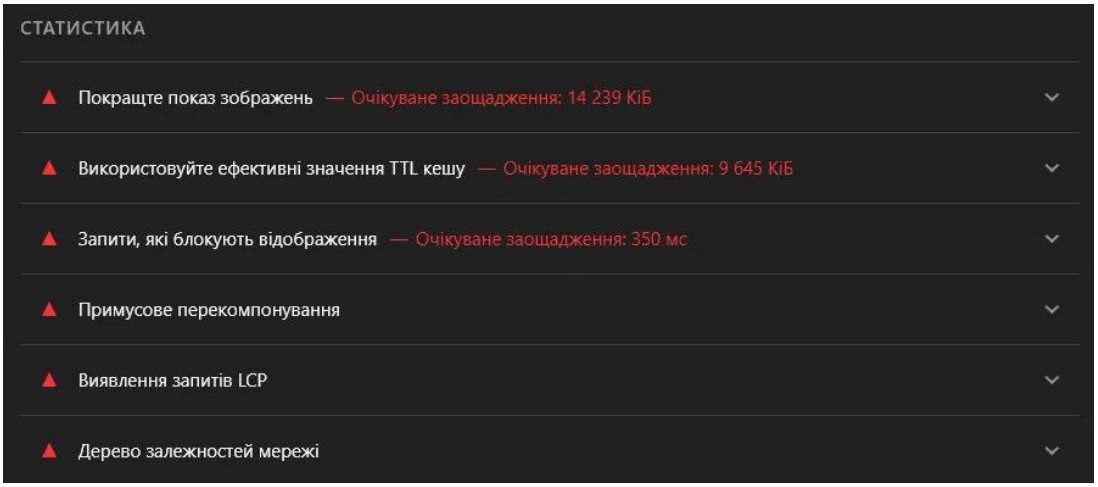

Рисунок 2.47 – Google Lighthouse: розділ статистики з рекомендаціями щодо оптимізації

Описані рекомендації є типовими для сайтів на платформі Google Sites, що використовують зовнішні медіаресурси та кастомні шрифти. Їх реалізація потребує або налаштувань на рівні Cloudflare (кешування, стиснення ресурсів), або доопрацювання HTML-коду кастомних компонентів (оптимізація шрифтів). Це визначає перспективний напрям подальшого розвитку системи в частині SEO та Core Web Vitals — метрик, що безпосередньо впливають на позиції сайту в результатах пошуку Google.

Висновок за результатами верифікації

Комплексний аудит Google Lighthouse підтверджує, що розроблений веб-ресурс відповідає вимогам, визначеним у постановці задачі. Бездоганна оцінка доступності свідчить про інклюзивність інтерфейсу, висока оцінка оптимальних методів підтверджує технічну якість реалізації, а показник SEO гарантує видимість ресурсу в пошукових системах. Показник ефективності 87/100, перебуваючи в жовтій зоні, не є критичним для інформаційного сайту мовної школи, проте окреслює конкретний технічний план подальшого доопрацювання — насамперед у напрямі оптимізації медіаконтенту та налаштування стратегії кешування.

Сукупність отриманих результатів підтверджує економічну доцільність обраного гібридного стеку: платформа Google Sites забезпечує максимальні показники доступності та безпеки без написання єдиного рядка інфраструктурного

коду, залишаючи простір для цільового покращення продуктивності засобами CDN-конфігурації Cloudflare.

Висновки до розділу :

У даному розділі було виконано повний цикл проєктування та практичної реалізації веб-орієнтованої інформаційної системи мовної школи Mother Tongue School. Розглянуто архітектурні особливості, можливості та обмеження платформи Google Sites, побудовано UML-діаграми варіантів використання, компонентну, архітектурну та діаграму послідовності. Спроєктовано структуру сайту та прототип інтерфейсу, реалізовано фронтенд-частину із застосуванням кастомної теми оформлення, серверну логіку на базі Google Apps Script з інтеграцією до хмарної бази даних Google Sheets, а також кастомні HTML/CSS/JavaScript-компоненти. Виконано мережеве розгортання з конфігурацією DNS-інфраструктури засобами Cloudflare, налаштуванням SSL/TLS-шифрування та переадресацією домену. Верифікацію якості розробленого рішення проведено інструментом Google Lighthouse, що підтвердило відповідність системи галузевим стандартам та визначило напрями подальшої оптимізації.

ВИСНОВКИ

У ході дослідження було проведено комплексний аналіз та практичну реалізацію хмарної веб-орієнтованої інформаційної системи для школи іноземних мов, що базується на інтеграції інструментів екосистеми Google Cloud із засобами мережевої безпеки Cloudflare.

Проведений аналіз сучасних тенденцій цифровізації у сфері EdTech підтвердив актуальність використання хмарних технологій для суб'єктів малого освітнього бізнесу. Дослідження існуючих веб-рішень мовних шкіл виявило суттєві обмеження традиційних підходів: High-code розробка характеризується високою вартістю створення та підтримки, тоді як більшість комерційних конструкторів сайтів не забезпечують необхідного рівня автоматизації серверної логіки та інтеграції бізнес-процесів. На основі проведеного аналізу обґрунтовано доцільність використання гібридного Low-code підходу на базі Google Cloud як оптимального за співвідношенням функціональності, гнучкості, масштабованості та економічної ефективності.

У межах роботи реалізовано повний цикл проєктування та розробки інформаційної системи. Формалізація архітектури засобами UML-моделювання забезпечила структурованість процесу розробки та відповідність програмного продукту визначеним функціональним вимогам. Створений веб-ресурс поєднує адаптивний користувацький інтерфейс на платформі Google Sites із серверною логікою на базі Google Apps Script, що забезпечує автоматизований прийом, обробку та збереження клієнтських заявок у хмарному середовищі Google Sheets. Використання кастомних компонентів дозволило подолати функціональні обмеження платформи та реалізувати рівень інтерактивності, характерний для повноцінних веб-застосунків.

Особливу увагу приділено питанням мережевої безпеки та стабільності функціонування системи. Інтеграція сервісів Cloudflare забезпечила наскрізне SSL/TLS-шифрування, захист веб-ресурсу засобами WAF, а також стабільну роботу

системи під власним доменним ім’ям. Реалізована архітектура відповідає сучасним вимогам до безпечного розгортання веб-рішень у хмарному середовищі.

Практична цінність отриманих результатів полягає у створенні готового до використання програмного продукту, що автоматизує процес первинної обробки клієнтських заявок та усуває фрагментацію даних між різними каналами комунікації. Важливою перевагою реалізованого рішення є відсутність постійних операційних витрат, оскільки вся інфраструктура функціонує в межах безкоштовних тарифних лімітів Google Cloud та Cloudflare. Крім того, система орієнтована на можливість самостійного адміністрування нетехнічним персоналом завдяки використанню інтуїтивних інструментів керування контентом і знайомого середовища Google Sheets.

Результати тестування та верифікації якості за допомогою інструменту Google Lighthouse підтвердили відповідність системи сучасним галузевим стандартам у сферах продуктивності, доступності, SEO та технічної якості веб-розробки. Отримані результати демонструють, що використання гібридного хмарного підходу дозволяє створювати технічно повноцінні та економічно ефективні веб-рішення без необхідності застосування складної та дорогої інфраструктури.

Таким чином, усі поставлені завдання кваліфікаційної роботи виконано у повному обсязі. Розроблена інформаційна система є готовим до практичного використання продуктом, що функціонує та забезпечує потреби діючої школи іноземних мов. Запропонований підхід може бути використаний як універсальна методологічна основа для створення хмарних веб-рішень у сфері малого освітнього бізнесу.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Тардаскіна Т., Рекліцька А. Сучасний цифровий інструмент для закладів вищої освіти в умовах розвитку EdTech. Економіка та суспільство. 2023. № 55. DOI: https://doi.org/10.32782/2524-0072/2023-55-51

2. Зацерківна М. Цифровізація освіти та маркетинг освітніх послуг в умовах збройної агресії російської федерації проти України. Цифрова платформа: інформаційні технології в соціокультурній сфері. 2023. Т. 6, № 1. С. 43–52. DOI: https://doi.org/10.31866/2617-796X.6.1.2023.283941

3. Сафонов Ю. М., Коротун О. П. Цифровізація освіти в Україні: технології та методики навчання. Трансформаційна економіка. 2024. № 7(15). С. 89–94. DOI: https://doi.org/10.32782/2786-8141/2024-7-15

4. ResearchAndMarkets. Education Technology (EdTech) Market Report 2025. URL: https://www.researchandmarkets.com/reports/5701122/edtech-market-size-share-trends-and-forecast (дата звернення: 03.02.2026).

5. Kyivstar Business Hub. Як технології впливали на український бізнес у 2024 році. URL: https://hub.kyivstar.ua/articles/yak-tehnologiyi-vplivali-na-ukrayinskij-biznes-u-2024-roczi-rezultati-doslidzhennya-kyivstar-business-hub (дата звернення: 11.02.2026).

6. Grand View Research. Low-Code Application Development Platform Market. URL: https://www.grandviewresearch.com/industry-analysis/low-code-application-development-platform-market (дата звернення: 17.02.2026).

7. Straits Research. Low-Code Development Platform Market Size, Share & Trends 2025–2033. URL: https://straitsresearch.com/report/low-code-development-platform-market (дата звернення: 24.02.2026).

8. Веб-орієнтовані системи автоматизації бізнес-процесів сучасних закладів. IT Synergy. 2023. Вип. 2 (5). URL: https://its.istu.edu.ua/index.php/ITS/article/download/36/32/107 (дата звернення: 03.03.2026).

9. Gartner. Forecasts Worldwide IT Spending to Grow 9.8 Percent in 2025. URL: https://www.gartner.com/en/newsroom/press-releases/2025-01-21-gartner-forecasts-worldwide-it-spending-to-grow-9-point8-percent-in-2025 (дата звернення: 10.03.2026).

10. Speak Up. Офіційний сайт мережі мовних шкіл. URL: https://speak-up.com.ua/ (дата звернення: 14.03.2026).

11. Wow English. Офіційний сайт мережі шкіл. URL: https://wow-english.ua/ (дата звернення: 15.03.2026).

12. LinguaDream. Офіційний сайт школи. URL: https://linguadreamvn.wixsite.com/linguadream (дата звернення: 16.03.2026).

13. Kissflow. Waterfall vs RAD vs Agile: which method is the best? URL: https://kissflow.com/application-development/rad/rad-waterfall-agile-which-one-is-best-for-you/ (дата звернення: 25.03.2026).

14. Харламенко В. Ю., Лут О. Ю. Дослідження low-code інструментів для розробки інформаційної системи мовної школи. Наука, промисловість, суспільство : матеріали міжнар. наук. конф., 27–29 травня 2026 р. Кривий Ріг : КНУ, 2026. С. 361.

15. UML-Diagrams.org. UML Use Case Extend Relationship. URL: https://www.uml-diagrams.org/use-case-extend.html (дата звернення: 02.04.2026).

16. Google Cloud. iFrame Sandbox Permissions Tutorial. URL: https://cloud.google.com/blog/products/data-analytics/iframe-sandbox-tutorial (дата звернення: 08.04.2026).

17. Google Developers. Web Apps | Apps Script. URL: https://developers.google.com/apps-script/guides/web (дата звернення: 14.04.2026).

18. Google Developers. HTML Service: Restrictions | Apps Script. URL: https://developers.google.com/apps-script/guides/html/restrictions (дата звернення: 14.04.2026).

19. MDN Web Docs. Content-Security-Policy: sandbox directive. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/Content-Security-Policy/sandbox (дата звернення: 21.04.2026).

20. Google Search Central. Understanding Core Web Vitals and Google Search Results. URL: https://developers.google.com/search/docs/appearance/core-web-vitals (дата звернення: 28.04.2026).

21. Gcore. TLS 1.3 vs TLS 1.2: performance, security, and protocol differences. URL: https://gcore.com/learning/tls-13-vs-12 (дата звернення: 30.04.2026).

22. Cloudflare Blog. Network Performance Update: Security Week 2024. URL: https://blog.cloudflare.com/network-performance-update-security-week-2024 (дата звернення: 05.05.2026).

23. Моркун Н. В., Завсегдашня І. В. Методичні вказівки до виконання кваліфікаційної роботи бакалавру для студентів спеціальності 122 «Комп'ютерні науки». Кривий Ріг : Видавничий центр КНУ, 2021. 50 с.

24. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. Київ : ДП «УкрННЦ», 2015. 26 с.

25. ДСТУ 8302:2015. Бібліографічне посилання. Загальні вимоги та правила складання. Київ : ДП «УкрННЦ», 2016. 16 с.

26. ДСТУ 3582:2013. Бібліографічний опис. Скорочення слів і словосполучень в українській мові. Загальні вимоги та правила. Київ : ДП «УкрННЦ», 2013. 23 с.