НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці

"На правах рукопису"
УДК 378.091:004

«До захисту допущено»
Завідувач кафедри
_____ Наталія АУШЕВА
“___”_____2022р.

Магістерська дисертація

зі спеціальності - 122 Комп’ютерні науки
за освітньо-професійною програмою магістерської підготовки -
Комп’ютерний моніторинг та геометричне моделювання процесів і систем

на тему “Управління навчанням студентів за допомогою Google API”

Виконав: студент 2 курсу, групи ТР-13мп
Гуковський Владислав Геннадійович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доц., к.в.н., Онисько А. І. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент доц., к.т.н., Гагарін О. О. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ - 2022

Національний технічний університет України
“Київський політехнічний інститут ім. Ігоря Сікорського”

Навчально-науковий інститут атомної та теплової енергетики
Кафедри цифрових технологій в енергетиці
Рівень вищої освіти другий, магістерський
За освітньою програмою "Комп’ютерний моніторинг та геометричне моделювання процесів і систем"
Спеціальності 122 Комп’ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Наталія АУШЕВА
(підпис)
«____» _____ 2022р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Гуковському Владиславу Геннадійовичу

1. Тема дисертації Управління навчанням студентів за допомогою Google API
Науковий керівник Онисько Андрій Ілліч, доцент, к.в.н.
затверджені наказом по університету від “07”листопада 2022 року №4067-с
2. Строк подання студентом дисертації 1 грудня 2022року
3. Вихідні дані до роботи мова програмування TypeScript, фреймворк Nest.js, бібліотека React, schedule.kpi.ua API, Google API
4. Перелік питань, які потрібно розробити синхронізація викладачів, груп і факультетів із schedule.kpi.ua; менеджмент ролей системи; надсилання запрошення на пошту; автоматичне створення курсів Google Classroom.
5. Орієнтований перелік ілюстративного матеріалу Призначення системи, користувачі системи, схема взаємодії програмних модулів, синхронізація із schedule.kpi.ua, підключення до Google API, структура бази даних, автентифікація, архітектура системи, демонстрація функціоналу.
6. Орієнтований перелік публікацій Контроль доступу до систем на основі ролей
7. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

8. Дата видачі завдання «30» червня 2022р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Строки виконання етапів магістерської дисертації	Примітка
1	Затвердження теми роботи	15.09.2022-25.10.2022	
2	Вивчення та аналіз задачі	01.09.2022-10.09.2022	
3	Розробка архітектури та загальної структури системи	11.09.2022-15.09.2022	
4	Розробка структур окремих підсистем	16.09.2022-05.10.2022	
5	Програмна реалізація системи	16.09.2022-20.10.2022	
6	Оформлення пояснювальної записки	21.10.2022-15.11.2022	
8	Передзахист	21.11.2022-30.11.2022	
9	Захист	12.12.2022-18.12.2022	

Студент _____ Гуковський В. Г.
(підпис)

Науковий керівник _____ Онисько А. І.
(підпис)

РЕФЕРАТ

Магістерська дисертація за темою “Управління навчанням студентів за допомогою Google API” виконана студентом кафедри цифрових технологій в енергетиці ННІ ІАТЕ Гуковським Владиславом Геннадійовичем зі спеціальності 122 “Комп’ютерні науки” за освітньо-професійною програмою “Цифрові технології в енергетиці” та складається зі: вступу; 5 розділів (Задача системи управління студентами за допомогою Google API, Опис методів вирішення задачі, Опис програмної реалізації, Робота користувача з програмою, Розроблення стартап-проету), висновків до кожного з цих розділів; загальних висновків; списку використаних джерел, який налічує 14 джерела та додатків. Загальний обсяг роботи 85 сторінок.

Актуальність теми. На сьогоднішній день все більшої популярності набувають системи контролю навчання студентів, які дозволяють спростити будь-яку взаємодію між деканатом, викладачами та студентами. Найбільш активно подібні системи почали використовуватися в останні роки з початком пандемії коронавірусу, а згодом, з початком повномасштабної війни, через що навчання було переведено у режим онлайн. Всі ці події черговий раз підкреслюють важливість розробки і вдосконалення середовищ для віддаленого контролю за навчанням, де будь-яка потреба студента, викладача або працівника адміністрації може бути задоволена у декілька кліків мишкою.

Метою роботи розробка система навчання з елементами автоматизації, яка матиме можливість працювати з Google API та синхронізовуватися із сервісом schedule.kpi.ua.

Завдання дослідження:

1. Аналіз вітчизняних та зарубіжних джерел.
2. Дослідити варіанти синхронізації із schedule.kpi.ua.
3. Дослідити документацію Google Classroom API на предмет інтеграції в систему.
4. Реалізувати інтеграцію перелічених сервісіву у єдину систему керування навчанням.

Об’єкт дослідження. Автоматизація та синхронізація даних з Schedule KPI, інтеграція з Google Classroom.

Предмет дослідження. Документація Google API, Schedule KPI API.

Методи дослідження. Алгоритм хешування SHA256, протокол HTTP, специфікація REST.

Ключові слова. Синхронізація, schedule.kpi.ua, Google, Classroom, викладачі, студенти, адміністратор, реєстрація.

ABSTRACT

The master's thesis on the topic "Student learning management using Google API" was written by a student of the Department of Digital Technologies in Energy of the National Institute of Energy of IATE, Vladyslav Hennadiyovych, major 122 "Computer Science" in the educational and professional program "Digital Technologies in Energy" and consists of: introduction; 4 sections (Task of student management system using Google API, Description of problem-solving methods, Description of software implementation, User work with the program, Developing of startup), conclusions to each of these sections; general conclusions; of the list of used sources, which includes 14 sources and applications. The total volume of work is 85 pages.

The actuality of the theme. Today, student learning control systems are gaining more and more popularity, which makes it possible to simplify any interaction between the dean's office, teachers, and students. Such systems have been most actively used in recent years with the onset of the coronavirus pandemic, and later, with the onset of a full-scale war, due to which training was transferred to the online mode. These events again emphasize the importance of developing and improving environments for remote monitoring of learning, where any need of a student, teacher, or administrative employee can be satisfied with a few mouse clicks.

The goal of the work is to develop a learning system with automation elements that will be able to work with Google API and synchronize with the schedule.kpi.ua.

Objectives of the study:

1. Analysis of domestic and foreign sources.
2. Explore options for synchronization with schedule.kpi.ua.
3. Explore the Google Classroom API documentation for system integration.
4. Implement the integration of the listed services into a single system.

The object of study. Automation and data synchronization with Schedule KPI, integration with Google Classroom.

The subject of study. Google API Documentation, Schedule KPI API.

Research methods. SHA256 hashing algorithm, HTTP protocol, REST specification.

Keywords. Synchronization, schedule.kpi.ua, Google, Classroom, teachers, students, administrator, registration.

ЗМІСТ

ВСТУП

На сьогоднішній день все більшої популярності набувають системи контролю навчання студентів, які дозволяють спростити будь-яку взаємодію між деканатом, викладачами та студентами. Найбільш активно подібні системи почали використовуватися в останні роки з початком пандемії коронавірусу, а згодом, з початком повномасштабної війни, через що навчання було переведено у режим онлайн. Всі ці події черговий раз підкреслюють важливість розробки і вдосконалення середовищ для віддаленого контролю за навчанням, де будь-яка потреба студента, викладача або працівника адміністрації може бути задоволена у декілька кліків мишкою.

Протягом декількох років було достатньо часу виділити основні потреби користувачів систем управління навчанням. Крім цього була можливість порівняти велику кількість конкурентів, а також взяти до уваги найважливіші недоліки подібних систем. Серед них є як і недостатня автоматизація, так і нестача інтеграції з популярними сервісами типу Google Classroom, які активно використовуються і підтвердили свою користь останніми роками. На жаль на даний момент доволі часто викладачам і працівникам адміністрації доводиться виконувати достатньо великий обсяг роботи, яка потенційно могла б бути автоматизована, а багато готових рішень перевикористано.

Аналізуючи порушені вище проблеми, можна виділити найголовніші задачі, яка має вирішити новостворена система контролю навчання. Як вже було зазначено, найголовнішими функціями продукту має бути автоматизація, задля найбільш швидкого старту роботи із системою та інтеграція з популярними сервісами для роботи зі студентами, які встигли полюбитися багатьма користувачами за останні роки (наприклад Google Classroom). Автоматизацію можна було б забезпечити через синхронізацію даних із вже існуючими сервісами навчального закладу. Крім того, безумовно важливо, щоб взаємодія із системою була максимально простою та інтуїтивно зрозумілою. Такі питання могло б вирішити додання безлічі підказок до інтерфейсу і гарно продуманий дизайн, який не буде вивалювати на користувача безліч функцій одразу, а буде подавати

інформацію порціонно. Також необхідно забезпечити доступ користувачів з будь-якого девайсу. Це можна виконати, зробивши систему веб додатком.

Мінімальний набір ролей користувачів має виглядати приблизно наступним чином: викладачі та студенти, які будуть проводити основну взаємодію з додатком, та працівники адміністрації, які курують усіма користувачами та корегують списки груп.

Як зазначено вище, можна забезпечити автоматизацію через синхронізацію викладачів, факультетів та груп із відкритими ресурсами. Студентів треба буде додавати адміністраторам, а система в свою чергу має полегшити цей процес, відправляючи запрошення на пошту.

Система має дозволяти викладачам зручно створювати курси та керувати ними у пару кліків, маючи заздалегідь визначений список груп, замість того, щоб у ручному режимі знаходити пошти студентів, щоб надати необхідні доступи.

Подібна система могла б цілковито забезпечити потреби як одного університету, так і багатьох інших, забезпечивши необхідний набір інструментів для синхронізації даних з конкретними навчальними закладами. Система поєднала б у собі увесь найголовніший функціонал, необхідний для навчання, дозволивши б користувачам відмовитися від застарілих рішень та перестати використовувати безліч різних сервісів, щоб мати змогу повноцінно проводити навчання.

1 ЗАДАЧА СИСТЕМИ УПРАВЛІННЯ НАВЧАННЯМ СТУДЕНТІВ ЗА ДОПОМОГОЮ GOOGLE API

У роботі будуть розглянуті теоретичні та практичні аспекти розробки програмного продукту для системи управління навчанням.

Метою роботи є розробка системи, яка дозволяє здійснювати процес навчання шляхом синхронізації користувачів, груп та розкладу із сервісом schedule.kpi.ua та створенню навчальних курсів викладачами з використання інструментів Google Classroom.

1.1 Призначення системи

Головним призначенням системи управління навчання є полегшення взаємодії викладачів у режимі онлайн зі студентами шляхом створення курсів, які автоматично будуть створюватися у Google Classroom з усіма студентами із груп, вибраних викладачем. Також система має суттєво полегшити її адміністрування шляхом синхронізації даних із schedule.kpi.ua, що дозволяє завжди мати актуальну інформацію про нові групи та викладачів, тим самим зменшуючи навантаження на працівників адміністрації.

1.2 Користувачі системи

Користувачами системи є студенти, викладачі та працівники адміністрації. Як показано на рисунку 1.1, для кожної категорії користувачів існує роль у системі з унікальним інтерфейсом, можливостями та правами.

Рисунок 1.1 — Актори та їх можливості

У користувачів немає можливості самостійної реєстрації у системі. Усі дані для входу будуть надаватися адміністратором або безпосереднім способом (файл з логінами та паролями), або через автоматичну розсилку даних для входу на пошту. Після першого входу користувача до системи йому буде запропоновано заповнити основні контактні дані та змінити пароль. Також обов’язковою умовою під час реєстрації є під’єднання профілю до Google та надання відповідних прав (якщо такі необхідні).

1.3 Задачі, які вирішуються системою

Для досягнення мети був складений наступний перелік задач:

1. Створення адміністратора за замовчуванням при першому запуску додатку з чистою базою даних або при відсутності у системі адміністратора за замовчуванням.
2. Можливість додання нових адміністраторів. Необхідно ввести ПІБ адміністратора та пароль для нього.
3. Синхронізація факультетів, груп і викладачів зі schedule.kpi.ua. При кожній синхронізації до бази даних додатку мають додаватися лише нові дані.
4. Експорт логінів та одноразових паролів викладачів для реєстрації в системі. З оглядів безпеки, після генерації нових паролей, старі стають неактуальними.
5. Додання нових студентів та автоматичне висилання логіну та паролю на пошту кожному новоствореному студенту.
6. Можливість зміни паролю користувачами.
7. Налаштувати інфраструктуру Google Console для роботи додатку з Google API.
8. Реалізація надання прав доступу від викладачів на використання Google Classroom API додатком від їх імені.
9. Передбачити можливість реєстрації пошти користувачів за допомогою інструментів Google.
10. Реалізувати прив’язку груп і викладачів до розкладу schedule.kpi.ua з подальшою можливістю переглянути цей розклад з їх профілю.
11. Створення курсів викладачами з автоматичним створенням курсу на Google Classroom та доданням туди усіх студентів вибранних груп. При створенні курсу, викладач має обрати назву курсу, факультет та групу. Крім того викладач має можливість відмовитися від сворення курсу у Google Classroom.
12. Передбачити можливість видалення навчального курсу разом із курсом у Classroom.

13. Передбачити можливість заповнення користувачами профілю їх контактними даними.

14. Забезпечити роботу додатку на різних девайсах.

1.4 Засоби розробки

Для розробки програмного продукту було використане велике різноманіття сучасних інструментів. Більшість з них можна побачити на рисунку 1.2. Для забезпечення роботи користувачів на різних девайсах, було прийнято рішення розробити саме веб застосунок.

Рисунок 1.2 — Засоби розробки

Ці інструменти включають в себе як засоби для розробки серверної частини та управління базами даних, так і інструменти розробки користувацького

інтерфейсу у браузері. Для створення дизайну макетів було використано інструмент Figma. Робота з інструментами Google проводилася через Google Developers Console.

1.4.1 Мова програмування

Для розробки системи була обрана мова програмування TypeScript [5]. Ця мова користується великою популярністю станом на 2022 рік та має добре оформлену документацію. Мова TypeScript дозволяє писати серверну і клієнтську частину, що також вплинуло на її вибір.

TypeScript — це мова програмування, яка була представлена компанією Microsoft ще в 2012 році. Вона позиціонується як інструмент для розробки веб-додатків і значно розширює обмежені можливості JavaScript [6] як показано на рисунку 1.3.

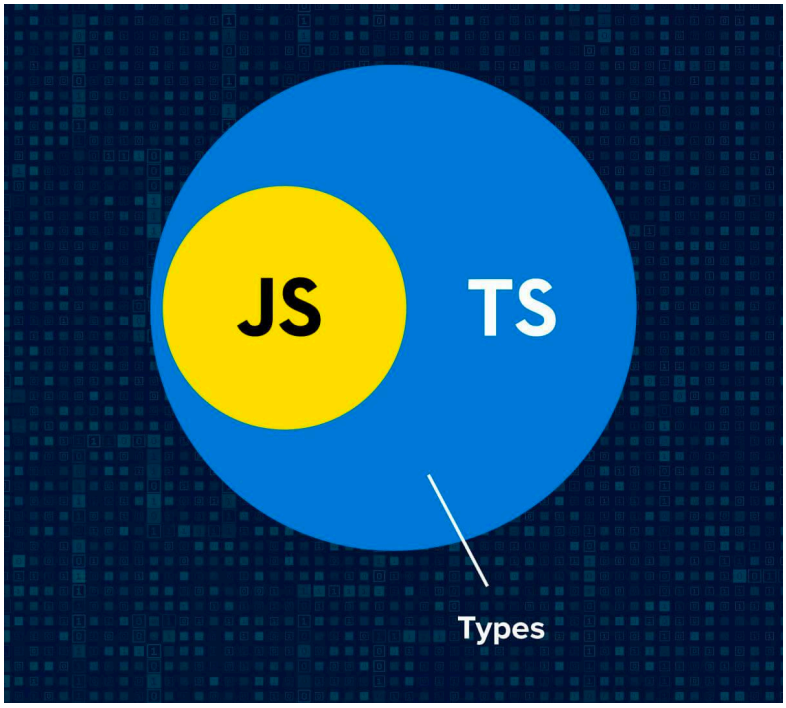


Рисунок 1.3 — Мова TypeScript

Специфікації мови відкриті і опубліковані в рамках угоди Open Web Foundation Specification Agreement (OWFa 1.0).

TypeScript є обернено сумісним з JavaScript. Через свою здатність компілюватися у JavaScript, він фактично забезпечує виконання у будь-якому сучасному браузері або використовуватися на сервері в оточенні Node.js.

1.4.2 Node.js

Хоча на сьогоднішній день існує безліч оточень для роботи з JavaScript, була обрана саме Node.js [7] через її величезну підтримку і кількість користувацьких пакетів. Усі бібліотеки та фреймворки, які будуть описані далі, працюють на базі саме цього оточення як показано на рисунку 1.4.

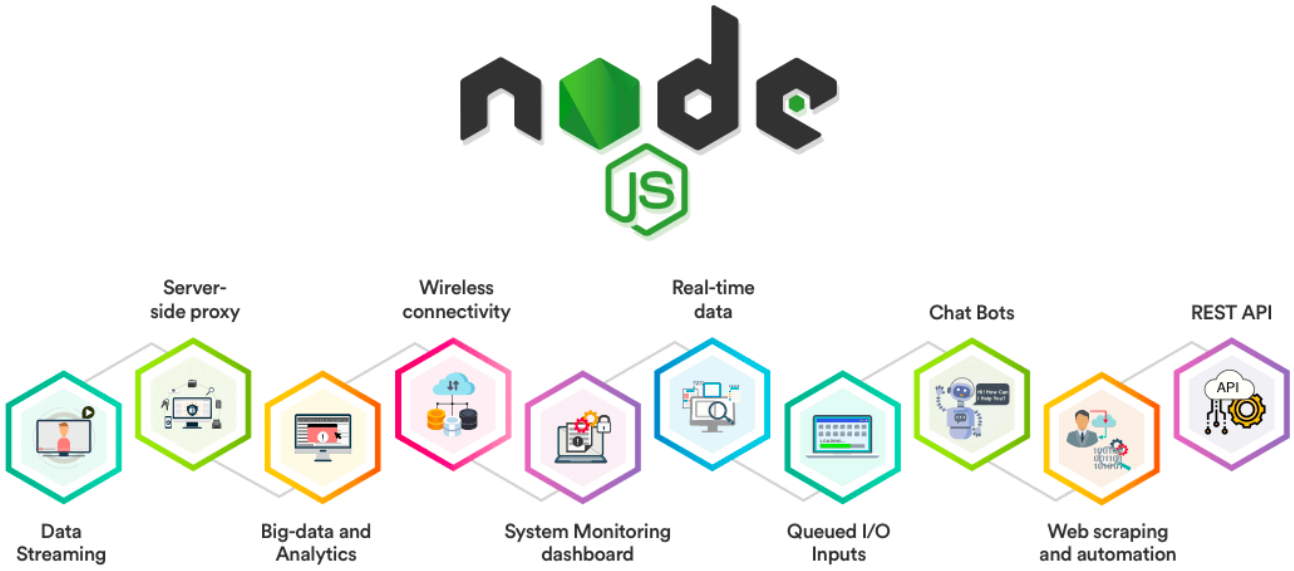


Рисунок 1.4 — Різноманіття пакетів Node.js

Node або Node.js — це програмна платформа, заснована на двигуні V8 (компілюючому JavaScript в машинний код), що перетворює JavaScript з вузькоспеціалізованої мови на мову загального призначення. Node.js додає можливість JavaScript взаємодіяти з пристроями вводу-виводу через свій API, написаний на C++, підключати інші зовнішні бібліотеки, написані різними мовами, забезпечуючи виклики до них із JavaScript-коду. Node.js застосовується переважно на сервері, виконуючи роль веб-сервера, але є можливість розробляти на Node.js та десктопні віконні програми (за допомогою NW.js, AppJS або Electron для Linux, Windows та macOS) і навіть програмувати мікроконтролери (наприклад,

tessel, low.js та espruino). В основі Node.js лежить подієво-орієнтоване та асинхронне (або реактивне) програмування з неблокуючим введенням/виводом.

1.4.3 JWT автентифікація

Для забезпечення прав доступу користувачів до системи була вибрана автентифікація через JSON Web Token (JWT) [4].

JSON Web Token — це відкритий стандарт (RFC 7519), який визначає компактний і самодостатній спосіб безпечної передачі інформації між сторонами як об’єкт JSON [3]. Цю інформацію можна перевірити та довіряти їй, оскільки вона має цифровий підпис. JWT можна підписати за допомогою секрету (з алгоритмом HMAC) або пари відкритих/приватних ключів за допомогою RSA або ECDSA.

Хоча JWT можна зашифрувати, щоб також забезпечити секретність між сторонами, ми зосередимося на підписаних токенах. Підписані токени можуть підтвердити цілісність претензій, що містяться в ньому, тоді як зашифровані токени приховують ці претензії від інших сторін. Коли токени підписуються за допомогою пар відкритих/приватних ключів, підпис також засвідчує, що лише сторона, яка володіє закритим ключем, є тією, яка підписала його як показано на рисунку 1.5.

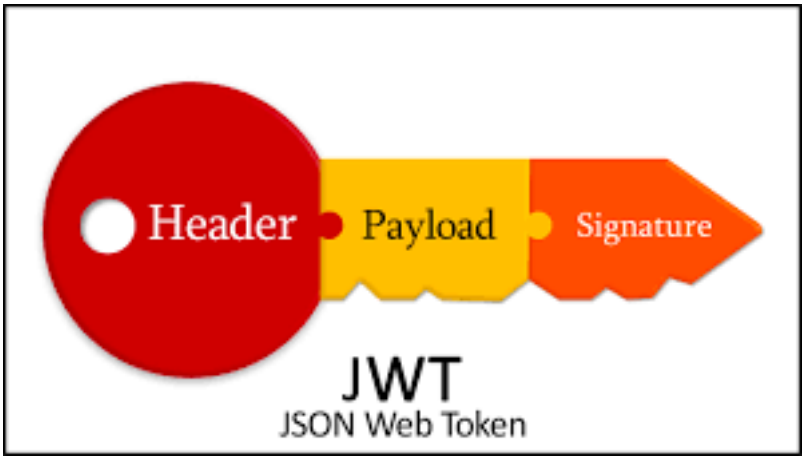


Рисунок 1.5 — Структура JSON Web Token

Ось кілька сценаріїв, коли JWT корисні:

1. Авторизація: це найпоширеніший сценарій використання JWT. Після входу користувача кожен наступний запит включатиме JWT, дозволяючи користувачеві отримувати доступ до маршрутів, послуг і ресурсів, які дозволені цим токеном. Єдиний вхід — це функція, яка сьогодні широко використовує JWT через невеликі накладні витрати та можливість легкого використання в різних доменах.

2. Обмін інформацією: веб-токени JSON є хорошим способом безпечної передачі інформації між сторонами. Оскільки JWT можна підписувати, наприклад, за допомогою пари відкритих/приватних ключів, ви можете бути впевнені, що відправники є тими, за кого себе видають. Крім того, оскільки підпис обчислюється за допомогою заголовка та корисного навантаження, ви також можете перевірити, чи вміст не було змінено.

1.4.4 Контейнеризація через Docker

Для комфортної роботи з базами даних впродовж розробки був використаний Docker [9] як інструмент контейнеризації, що дозволяє створювати ізольовані середовища (контейнери), як показано на рисунку 1.6, в яких можна легко розгорнути базу даних і так само легко її потім видалити разом із контейнером (рис. 1.6).

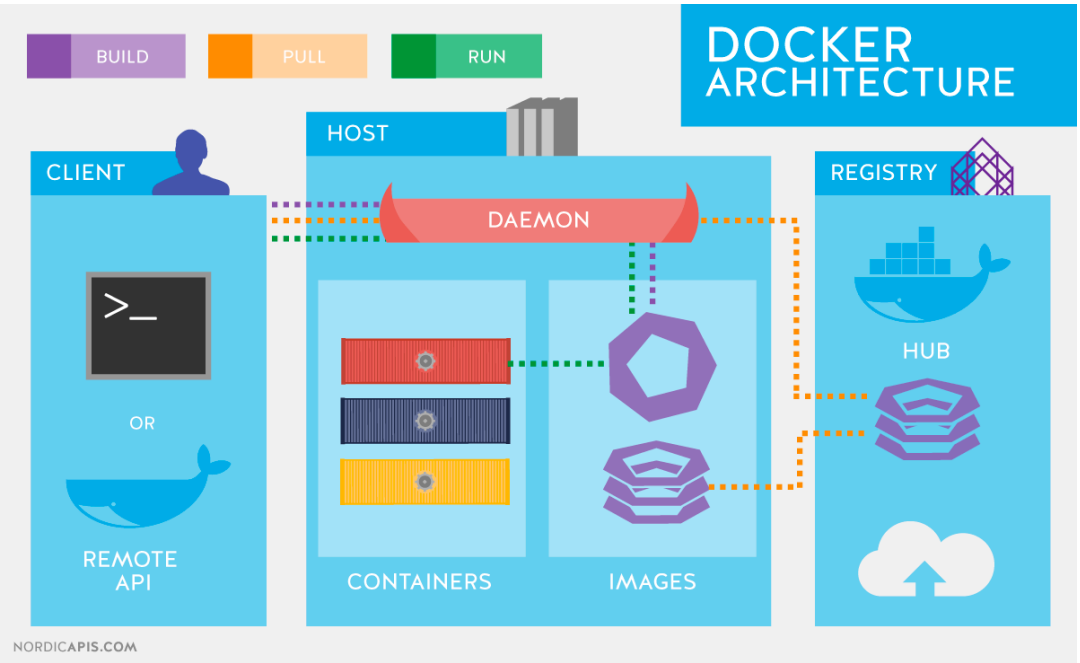


Рисунок 1.6 — Архітектура Docker

Крім роботи з базами даних, Docker надає можливість пакувати та запускати програму в слабко ізольованому середовищі. Ізоляція та безпека дозволяють запускати багато контейнерів одночасно на певному хості. Контейнери легкі та містять усе необхідне для запуску програми, тому нам не потрібно покладатися на те, що зараз встановлено на хості.

Docker спрощує життєвий цикл розробки, дозволяючи розробникам працювати в стандартизованих середовищах, використовуючи локальні контейнери, які надають ваші програми та служби. Контейнери чудово підходять для безперервної інтеграції та робочих процесів безперервної доставки (CI/CD).

Контейнерна платформа Docker дозволяє виконувати робочі навантаження з високою переносимістю. Контейнери Docker можуть працювати на локальному ноутбуці розробника, на фізичних або віртуальних машинах у центрі обробки даних, у хмарних провайдерах або в різних середовищах.

Портативність і легкість Docker також спрощує динамічне керування робочими навантаженнями, масштабуючи або видаляючи додатки та служби відповідно до потреб бізнесу майже в реальному часі.

Docker легкий і швидкий. Він забезпечує життєздатну, економічно ефективну альтернативу віртуальним машинам на основі гіпервізора, тож ви можете використовувати більше потужності свого сервера для досягнення своїх бізнес-цілей. Docker ідеально підходить для середовищ із високою щільністю, а також для малих і середніх розгортань, де потрібно робити більше з меншими ресурсами.

1.4.5 База даних PostgreSQL

PostgreSQL — це потужна об’єктно-реляційна база даних з відкритим вихідним кодом, активна розробка якої триває понад 35 років, що заслужило їй міцну репутацію надійності, надійності функцій і продуктивності як показано на рисунку 1.7.

Для розробки була обрана саме вона через свою надійність. Також одним із найважливіших критерієм вибору є відкритий вихідний код, що виділяє базу даних на тлі конкурентів.

PostgreSQL [12] заслужив міцну репутацію завдяки своїй перевіреній архітектурі, цілісності даних, надійному набору функцій, розширюваності та відданості спільноти з відкритим кодом, яка стоїть за програмним забезпеченням, щоб постійно надавати ефективні та інноваційні рішення. PostgreSQL працює на всіх основних операційних системах, сумісний з ACID з 2001 року та має потужні додаткові компоненти, такі як популярний розширювач геопросторових баз даних PostGIS.

Рисунок 1.7 — PostgreSQL

PostgreSQL має багато функцій, які допомагають розробникам створювати програми, адміністраторам — захищати цілісність даних і створювати відмовостійке середовище, а також допомагає керувати своїми даними, незалежно від того, наскільки великий чи малий набір даних.

PostgreSQL намагається відповідати стандарту SQL, якщо така відповідність не суперечить традиційним функціям або може призвести до невдалих архітектурних рішень. Багато функцій, необхідних стандарту SQL, підтримуються, хоча іноді з дещо відмінним синтаксисом або функціями. З часом можна очікувати подальших кроків у напрямку відповідності. Починаючи з випуску версії 15 у жовтні 2022 року, PostgreSQL відповідає принаймні 170 із 179 обов’язкових функцій для відповідності SQL:2016 Core.

PostgreSQL включає вбудовану двійкову реплікацію, засновану на асинхронній доставці змін (журналів попереднього запису (WAL)) до вузлів реплік із можливістю виконувати запити лише для читання до цих реплікованих вузлів. Це дозволяє ефективно розподіляти трафік читання між кількома вузлами. Попереднє програмне забезпечення реплікації, яке дозволяло подібне масштабування читання, зазвичай покладалося на додавання тригерів реплікації до основного, збільшуючи навантаження.

PostgreSQL включає вбудовану синхронну реплікацію, яка гарантує, що для кожної транзакції запису головний пристрій чекає, доки принаймні один вузол-реплікатор не запише дані до свого журналу транзакцій. На відміну від інших систем баз даних, довговічність транзакції (незалежно від того, чи є вона асинхронною чи синхронною) може бути визначена для бази даних, для кожного користувача, для кожного сеансу або навіть для окремої транзакції. Це може бути корисним для робочих навантажень, які не вимагають таких гарантій, і може бути необхідним для всіх даних, оскільки це сповільнює продуктивність через вимогу підтвердження транзакції, що досягає синхронного режиму очікування.

Резервні сервери можуть бути синхронними і асинхронними. Синхронні резервні сервери можна вказати в конфігурації, яка визначає, які сервери є кандидатами для синхронної реплікації. Перший у списку активний потоковий сервер буде використано як поточний синхронний сервер. Якщо це не вдається, система переходить до наступного в черзі.

1.4.6 Фреймворк Nest.js

Nest.js — це потужний фреймворк для створення серверних додатків. Він використовує мову TypeScript на працює в оточенні Node.js. Цей фреймворк був обраний з рахунок великоъ кількості готових рішень в своїй основі, а також чудово оформленій документації, що дозволяє досить швидко. Nest.js [8] використовує багато корисних архітектурних патернів, що дозволяє зручно будувати архітектуру великих додатків та налаштувати взаємодію внутрішніх модулів як показано на рисунку 1.8. Це дозволило досить швидко інтегрувати у

систему різні рівні доступу для ролей користувачів, налаштувати валідацію і додало багато можливостей для тестування програми.

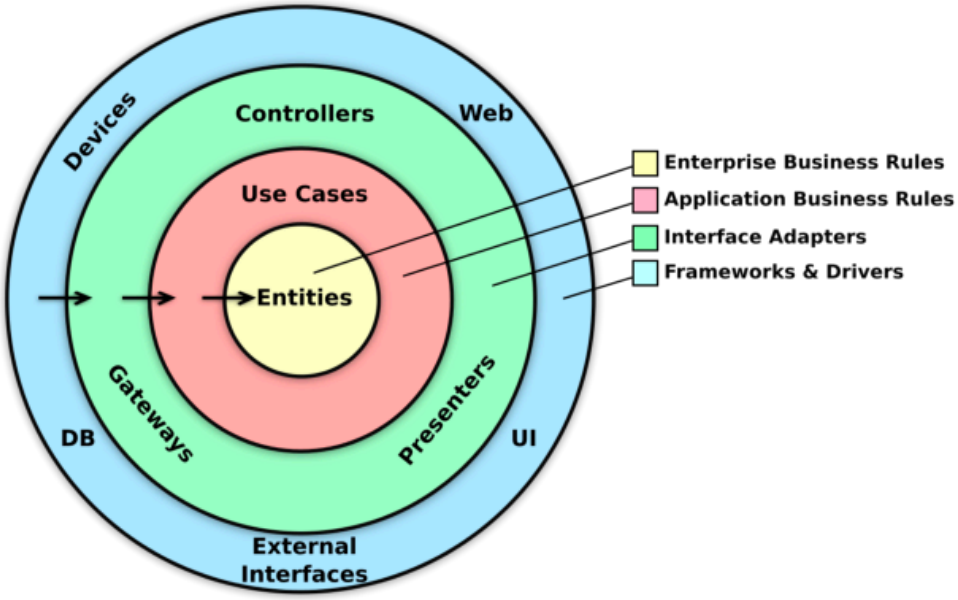


Рисунок 1.8 — Архітектура проекту на Nest.js

Крім того Nest.js має внутрішню інтеграцією з бібліотеками TypeORM [11], Passport.js та Express [10], що дозволило досить швидко налаштувати підключення додатку до бази даних та налаштувати автентифікацію. Express це найпопулярніший пакет для роботи Node.js з http [1], тому його інтеграція в Nest.js дозволяє використовувати вже готові рішення для цієї бібліотеки.

Passport — це проміжне програмне забезпечення автентифікації для Node.js. Надзвичайно гнучкий і модульний Passport можна непомітно вставити в будь-яку веб-програму на основі Express. Повний набір стратегій підтримує автентифікацію за допомогою імені користувача та пароля, Facebook, Twitter тощо.

1.4.7 Figma

Figma — це веб-додаток для розробки інтерфейсу з додатковими офлайн-функціями, доступними в настільних програмах для macOS і Windows. Цей інструмент був обраний для розробки макетів дизайну для системи управління навчанням, так як це один із найбільш популярних додатків розробки інтерфейсу

зз широким набором функціоналу. Набір функцій Figma зосереджений на дизайні інтерфейсу користувача та взаємодії з користувачем, з акцентом на співпрацю в реальному часі, використовуючи різноманітні редактори векторної графіки та інструменти для створення прототипів як показано на рисунку 1.9.

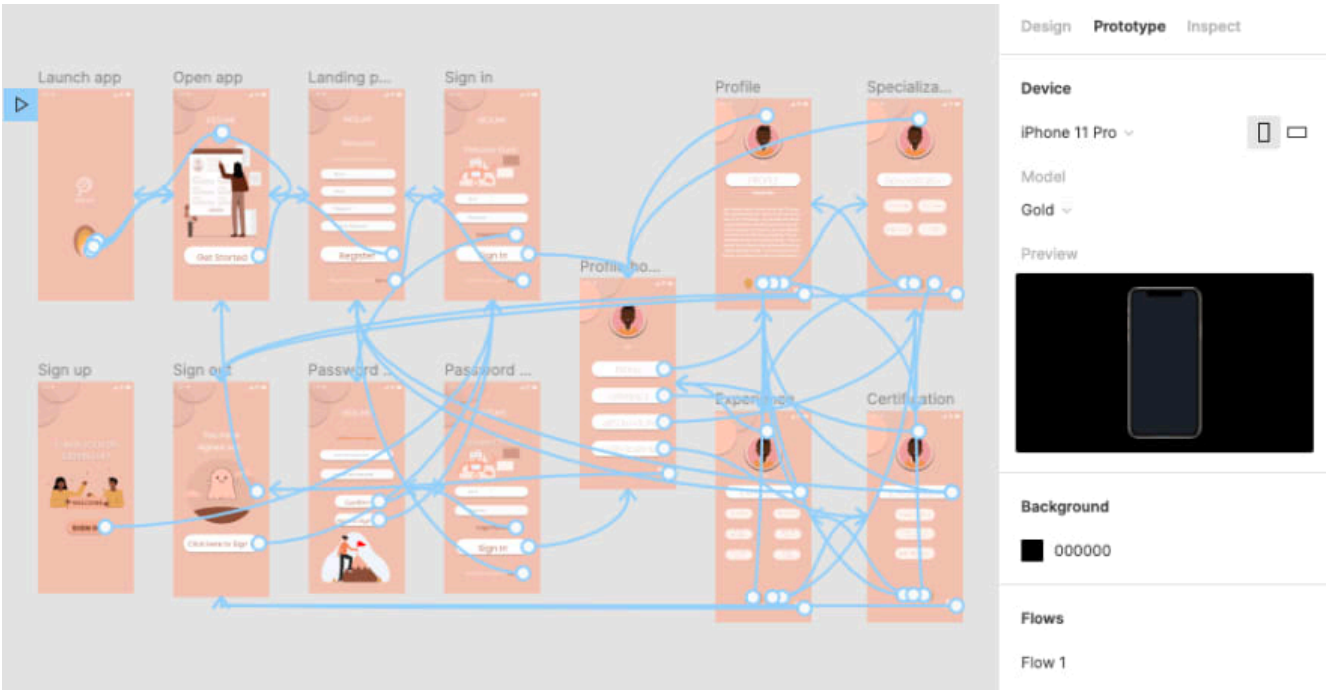


Рисунок 1.9 — Створення макету інтерфейсу за допомогою Figma

Сервіс є безкоштовним для індивідуальних користувачів і платним для фахових команд. Даний редактор підходить як для створення простих прототипів і дизайн-систем, так і складних проєктів (мобільні додатки, портали).

1.4.8 React

Бібліотекою для створення користувацького інтерфейсу (фронтенд) було обрано React [13], так як він вважається найпопулярнішою біблотекою серед конкурентів, має величезну підтримку серед розробників та має безліч готових користувацьких готових рішень.

React це бібліотека, яка використовує принципи реактивного програмування. Ключовими елементами React є JSX та віртуальний DOM.

JSX — це спеціальний синтаксис бібліотеки React, який поєднує у собі можливості HTML [14] та JavaScript як показано на рисунку 1.10.

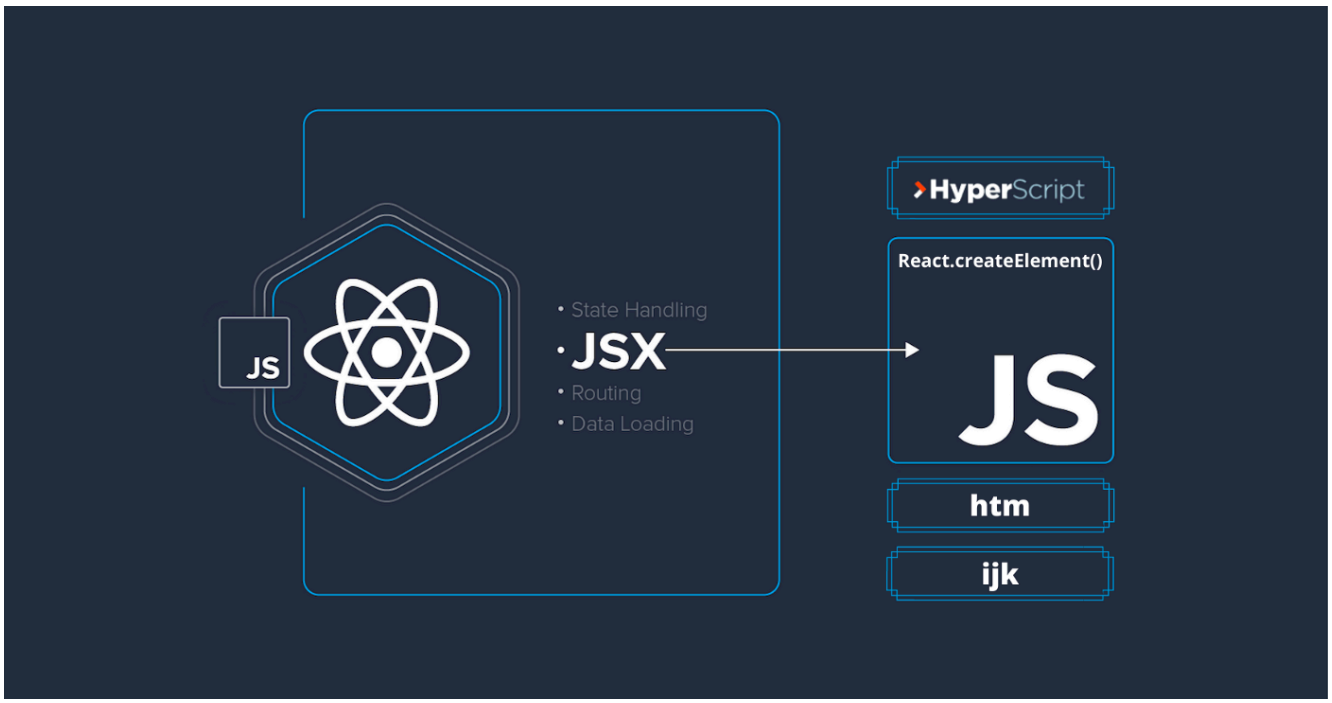


Рисунок 1.10 — JSX

Віртуальний DOM — це концепція програмування, в якій ідеальне чи «віртуальне» представлення інтерфейсу користувача зберігається в пам’яті і синхронізується зі «справжнім» DOM за допомогою бібліотеки, такої як ReactDOM. Цей процес називається узгодженням.

Оскільки «віртуальний DOM» — це швидше патерн, ніж конкретна технологія, цим терміном іноді позначають різні поняття. У світі React термін «віртуальний DOM» зазвичай асоціюється з React елементами, оскільки вони є об’єктами, що представляють інтерфейс користувача. Однак, React також використовує внутрішні об’єкти, так звані «волокна» (fibers) для зберігання додаткової інформації про дерево компонентів. Вони також можуть вважатися частиною реалізації «віртуального DOM» в React.

1.4.9 Material UI

Бібліотекою стилів було обрано Material UI (MUI). Це популярне рішення, яке містить велику кількість заготовлених елементів кнопок, форм, попереджень, текстів тощо. При роботі з Material UI не використовується CSS, на томість всі стилі прописуються безпосередньо за допомогою JavaScript.

Material UI – це бібліотека компонентів React з відкритим кодом, яка реалізує Material Design від Google. Вона містить повну колекцію готових компонентів, які готові до використання у виробництві одразу після вилучення. Інтерфейс Material UI красивий за дизайном і має набір параметрів налаштування, які спрощують реалізацію власної системи дизайну на основі готових компонентів. Деякі з них можна побачити на рисунку 1.11.

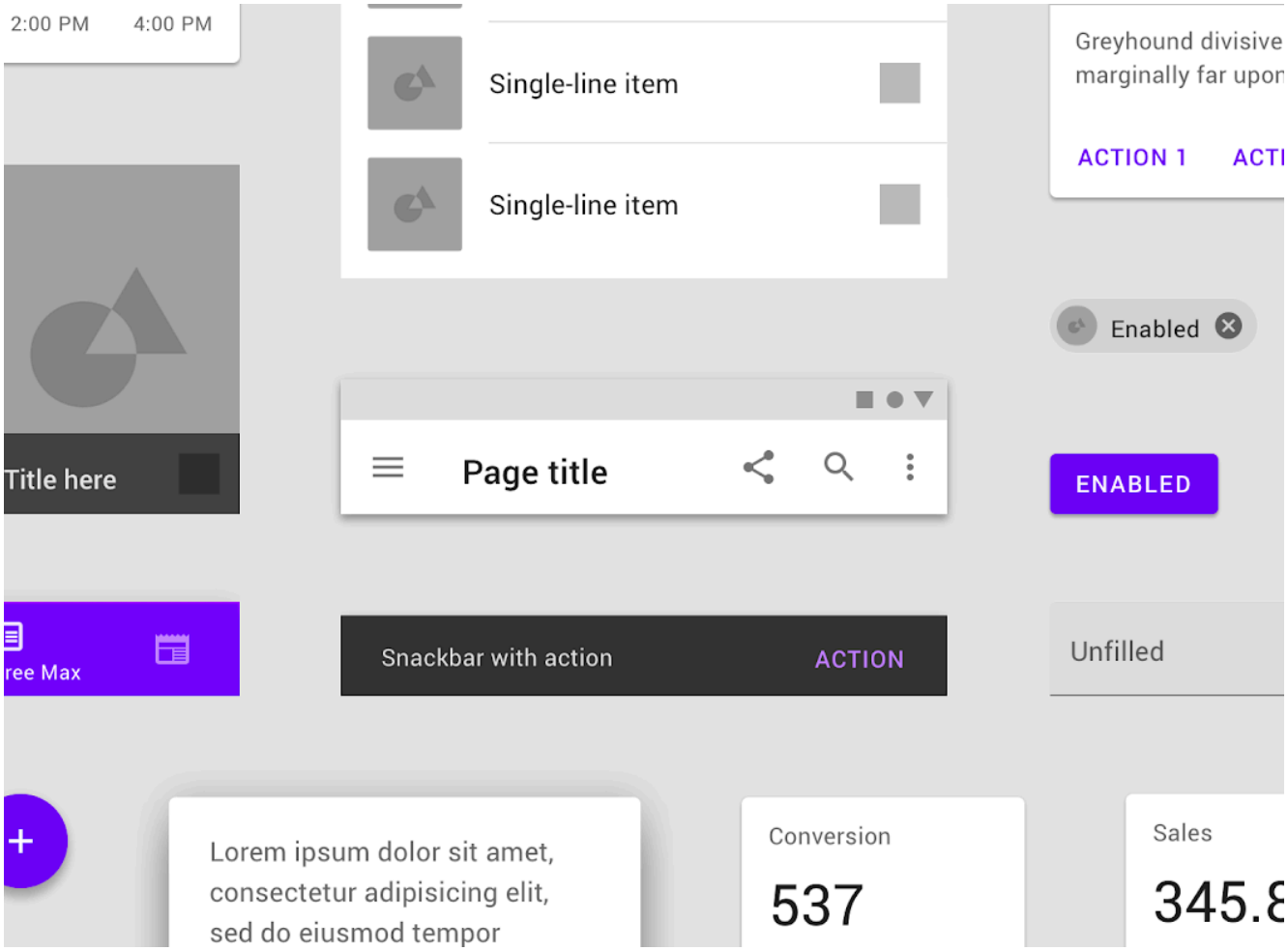


Рисунок 1.11 — Бібліотека компонентів MUI

Material Design – це мова дизайну, розроблена Google у 2014 році. Розширюючи «картки», які дебютували в Google Now, Material Design

використовує більше макетів на основі сітки, адаптивну анімацію та переходи, відступи та глибину. такі ефекти, як освітлення та тіні. Google анонсувала Material Design 25 червня 2014 року на конференції Google I/O 2014. Основною метою Material Design є створення нової візуальної мови, яка поєднує принципи хорошого дизайну з технічними та науковими інноваціями. Дизайнер Матіас Дуарте пояснив, що, «на відміну від справжнього паперу, наш цифровий матеріал може розумно розширюватися та змінюватися. Матеріал має фізичні поверхні та краї. Шви та тіні надають значення тому, до чого можна торкнутися». Google заявляє, що їхня нова мова дизайну заснована на папері та чорнилі, але реалізація відбувається в просунутій манері. У 2018 році Google детально описав оновлену мову, зосередившись на наданні дизайнерам більшої гнучкості для створення спеціальних «тем» із різною геометрією, кольорами та типографікою.

1.4.10 Google Developers Console

Google Developers — це хмарова інфраструктура Google, яка використовується веб додатком управління студентами для налаштування усіх необхідних доступів до Google API та тестових даних. Лише деяка частина сервісів показана на рисунку 1.12.

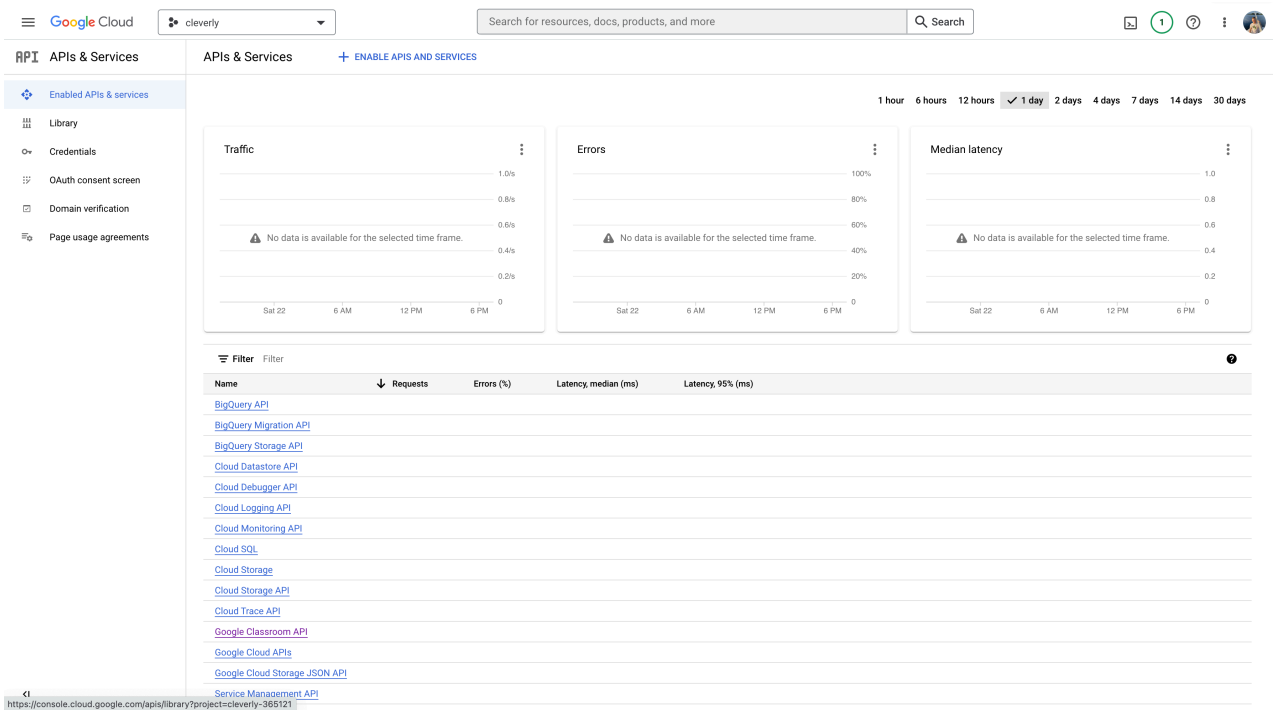


Рисунок 1.12 — Головна сторінка Google Developers

Google Developers – це сайт Google для інструментів і платформ розробки програмного забезпечення, інтерфейсів прикладного програмування (API) і технічних ресурсів. Сайт містить документацію про використання інструментів розробника Google і API, включаючи групи обговорень і блоги для розробників, які використовують продукти Google для розробників. Майже для всіх популярних споживчих продуктів Google, як-от Google Maps, YouTube, Google Apps та інших, пропонуються API. На сайті також є різноманітні продукти та інструменти для розробників, створені спеціально для розробників. Google App Engine – це служба розміщення веб-програм. Project Hosting надає користувачам керування версіями для відкритого коду.

1.4.11 Heroku

Heroku — це хмаровий сервіс, що дозволяє безкоштовно розгортати веб додатки. Основні функції з головної сторінки Heroku показані на рисунку 1.13.

Рисунок 1.13 — Платформа Heroku

Безкоштовність цього сервісуце головна перевага серед конкурентів і причина чому саме Heroku був вибраний для розгортки системи управління

навчанням. Сервіс також безкоштовно надає можливість підключати базу даних до розгорнутих у його середовищі додатків.

Heroku запускає програми всередині dynos — розумних контейнерів у надійному, повністю керованому середовищі виконання як показано на рисунку 1.14.

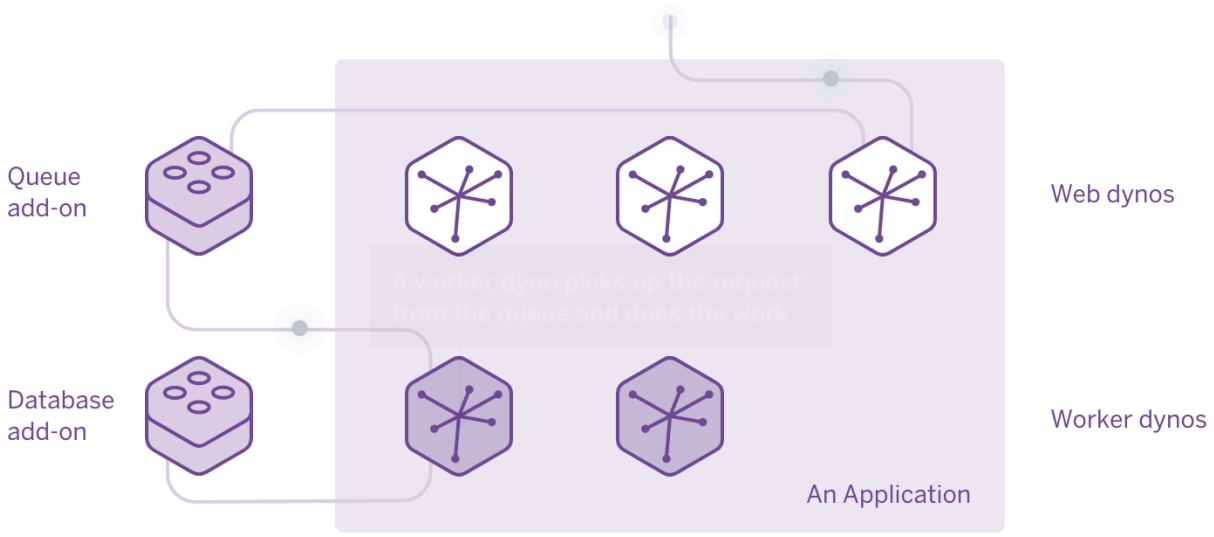


Рисунок 1.14 — Heroku Dynos

Розробники розгортають свій код, написаний на Node, Ruby, Java, PHP, Python, Go, Scala або Clojure, у систему збірки, яка створює програму, готову до виконання. Системні та мовні стеки контролюються, виправляються та оновлюються, тому вони завжди готові та оновлені. Середовище виконання забезпечує роботу програм без ручного втручання.

1.5 Схема взаємодії програмних модулів

Глобально система складається із двох програм — веб додаток з інтерфейсом користувача (частина фронтенд) та серверна частина, яка надає відповідає за взаємодію усіх внутрішніх модулів.

Як показано на рисунку 1.15, серверний додаток складається з наступних модулів:

• центральний модуль — містить основну імплементацію додатку. Він відповідальний в тому числі за автентифікацію, авторизацію, усю бізнес логіку, а також саме цей модуль взаємодіє із базою даних;

• модуль google — відповідає за взаємодію з Google API. Він приймає на вхід ключі доступу до користувача Google. Цей модуль не знає нічого про сутності центрального модулю і не взаємодіє з базою даних;

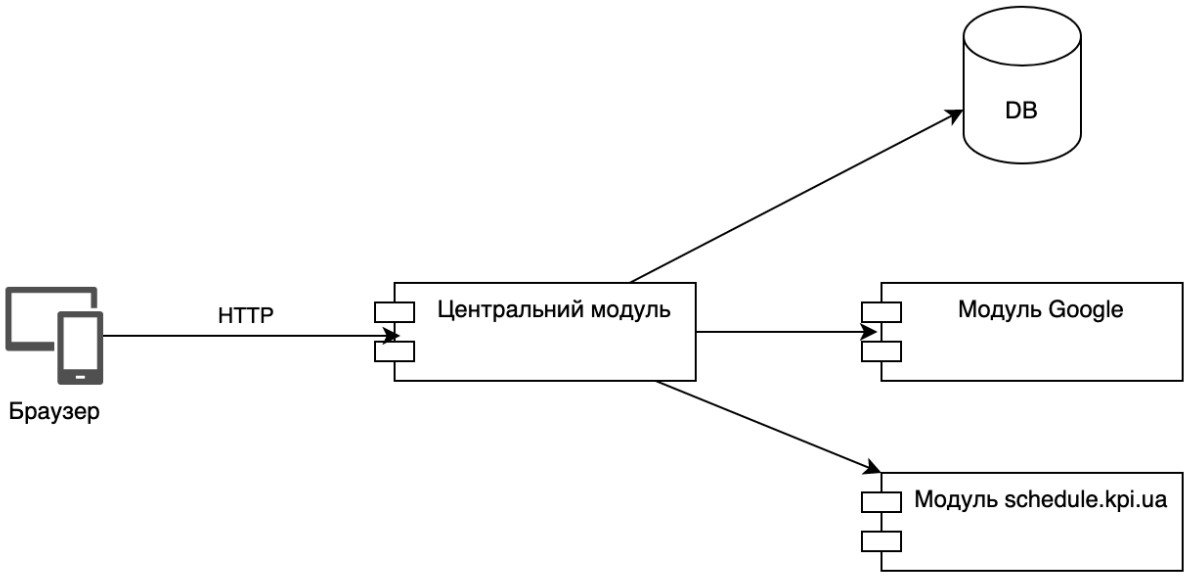


Рисунок 1.15 — Взаємодія модулів системи

• модуль schedule.kpi.ua — відповідає лише за отримання інформації про викладачів, групи, розклад безпосередньо із сервісу schedule.kpi.ua. Увесь парсинг даних та їх збереження до бази даних відбувається у центральному модулі.

Висновки до розділу 1

У цьому розділі було розглянуто основні засоби розробки програмного продукту. Були наведені їх основні переваги над конкурентами та переваги у застосуванні у даній нам предметній області. Оглянуті інструменти розраховані на розробку системи модульним підходом.

2 ОПИС МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ

В цьому розділі буде описано опис методів виконання поставленої задачі, які складають новизну системи та вирізняють її на тлі інших рішень. Будуть наведені дизайнерські рішення для побудови зручного інтерфейсу користувача, а також деталі імплементації сторонніх сервісів у додаток.

2.1 Вибір кольорової палітри

Дизайн будь-якого успішного додатка починається з вибору кольорової палітри. Зазвичай сучасні дизайни сайтів мають у своїй основі 2 кольори — головний та другорядний, включаючи їх відтинки для різних станів. В додаток до головного та другорядного кольору можуть ідти декілька побічних кольорів, відповідальних за фон, текст тощо. Така ж система лежить в основі Material Design. Приклад палітри показаний на рисунку 2.1.

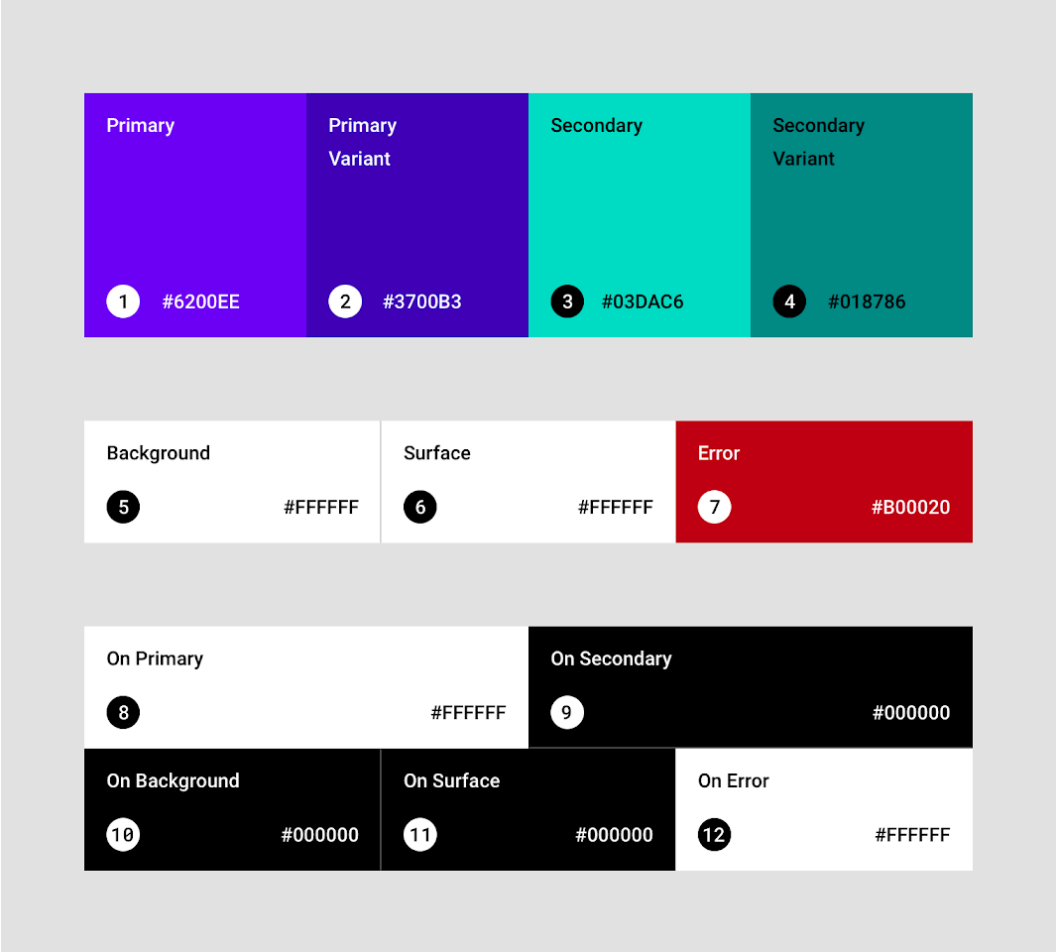


Рисунок 2.1 — Кольорова палітра Material Design

Було проведено дослідження над тим, який колір найбільш сприятливий для навчальних платформ. Для подібних платформ слід звернути увагу на безліч факторів. Серед них:

• доступність — кольори повинні легко комбінуватися з контрастним чорним та білим кольором для комфортного читання тексту;

• колір і навчання — основний колір навчальних платформ повинен фокусувати увагу користувача;

• послідовність кольорів — необхідно відповідати своїй палітрі. Наприклад, якщо це можливо, треба використовувати один колір для всіх виділень і один колір для всіх стрілок.

В ході дослідження та аналізу різних навчальних платформ, було виявлено, що одними з найсприятливіших кольорів для навчання є зелений та синій. Тож для нашої системи було обрано відповідні відтінки кольорів як показано на рисунку 2.2.

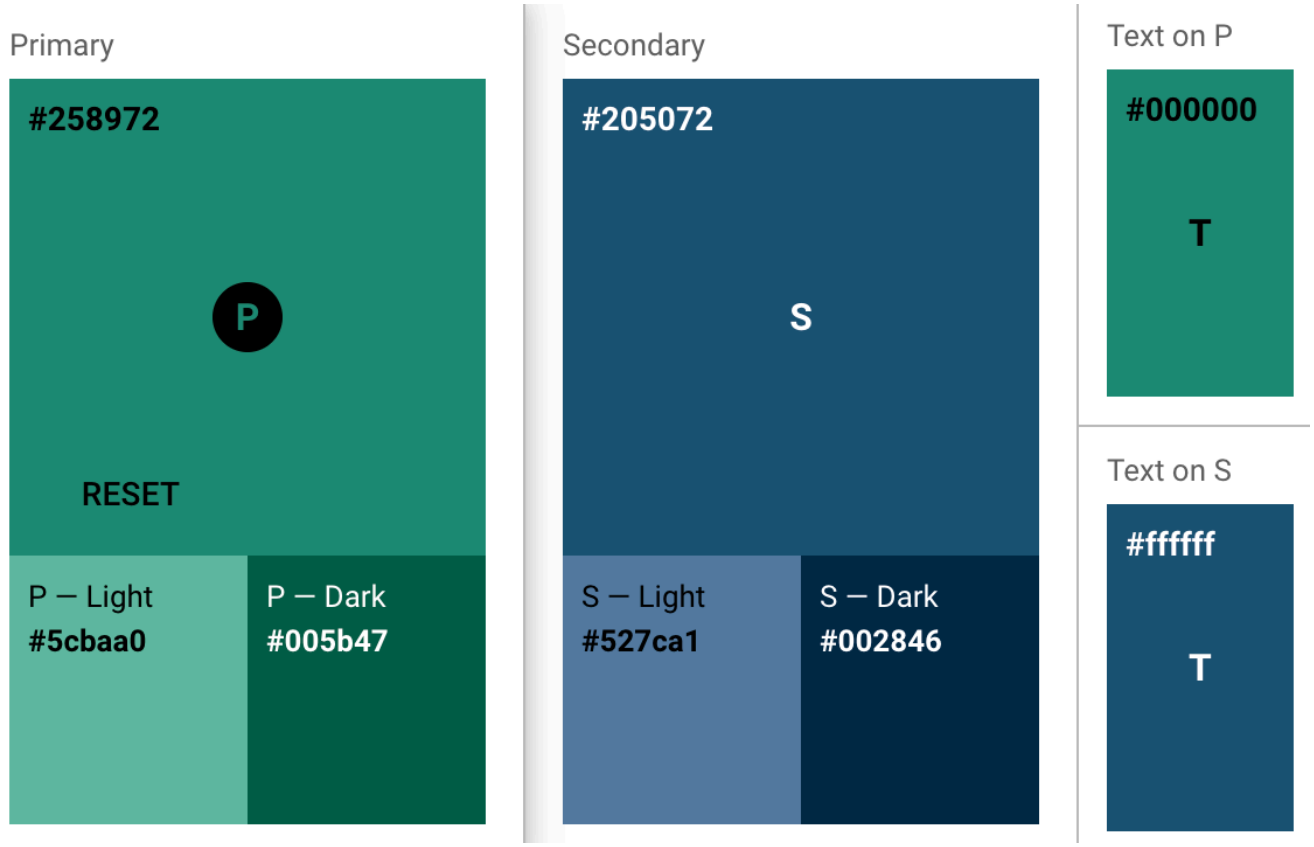


Рисунок 2.2 — Кольорова палітра додатку

Для тестування доступності палітри було використано сервіс m2.material.io як показано на рисунку 2.3.

Рисунок 2.3 — Доступність палітри на m2.material.io

Як бачимо з аналізу доступності, більшість текстів сприймається людським оком нормально на обраній палітрі кольорів.

2.2 Синхронізація даних зі schedule.kpi.ua

Сервіс schedule.kpi.ua надає вільний доступ до користування своїм API. Сервіс доступний за посиланням https://schedule.kpi.ua/api/ і має наступні кінцеві точки API:

• список груп: schedule/groups — містить інформацію назву групи, ідентифікатор та назву факультету як показано на рисунку 2.4;

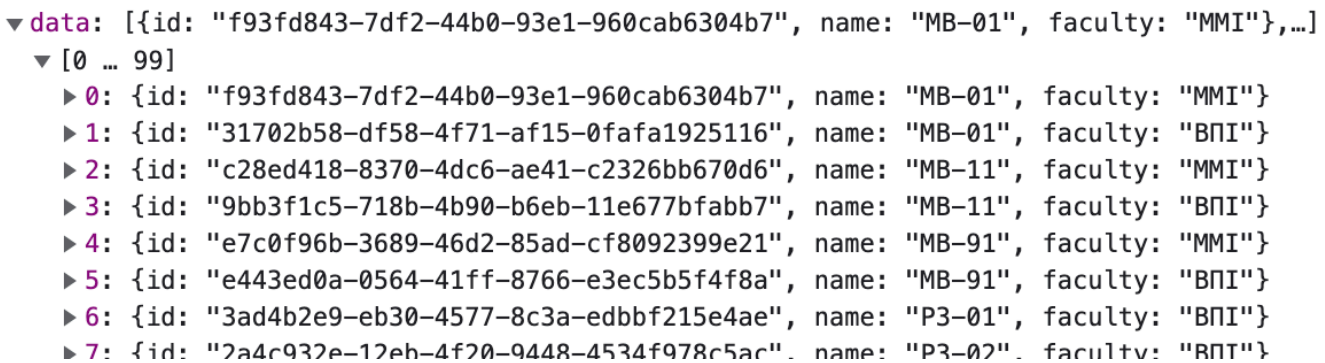


Рисунок 2.4 — Дані груп зі schedule.kpi.ua

• список викладачів: schedule/lecturer/list — містить ідентифікатор викладача та повне ім’я як показано на рисунку 2.5;

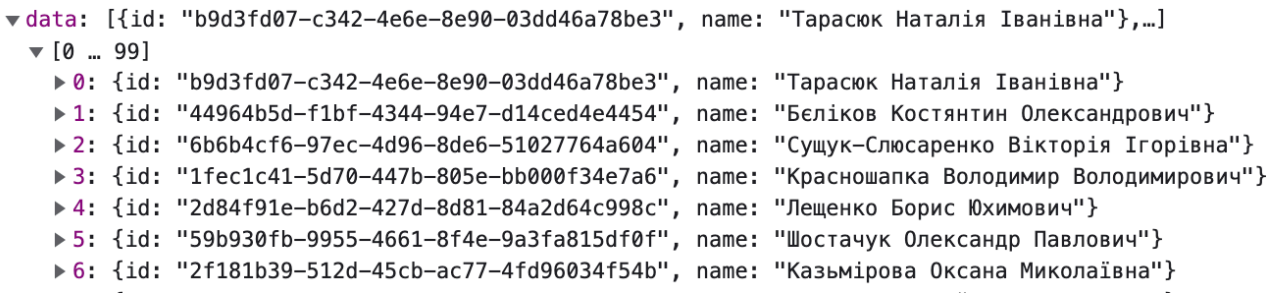


Рисунок 2.5 — Дані про викладачів зі schedule.kpi.ua

- розклад групи: schedule/lessons?groupId={group id};
- розклад екзаменів групи: exams/group?groupId={group id};
- розклад викладача: schedule/lecturer?lecturerId={id from list};
- сьогоднішній тиждень та день: time/current.

В ході дослідження було з’ясовано, що для нашого додатку достатньо спарсити лише перші 2 кінцеві точки, які надають інформацію про викладачів, групи та факультети і з їх допомогою можна отримати доступ до розкладу.

Профілі викладачів заносяться до таблиць користувачів і викладачів. Далі адміністратор вже може створити логіни на паролі для них, як було показано у минулих розділах. Щоб синхронізувати викладачів із сервісом розкладу, необхідно натиснути кнопку як показано на рисунку 2.6. При повторному виклику

синхронізації, будуть додані лише викладачі, ідентифікатори яких ще не занесені в базу даних додатку.

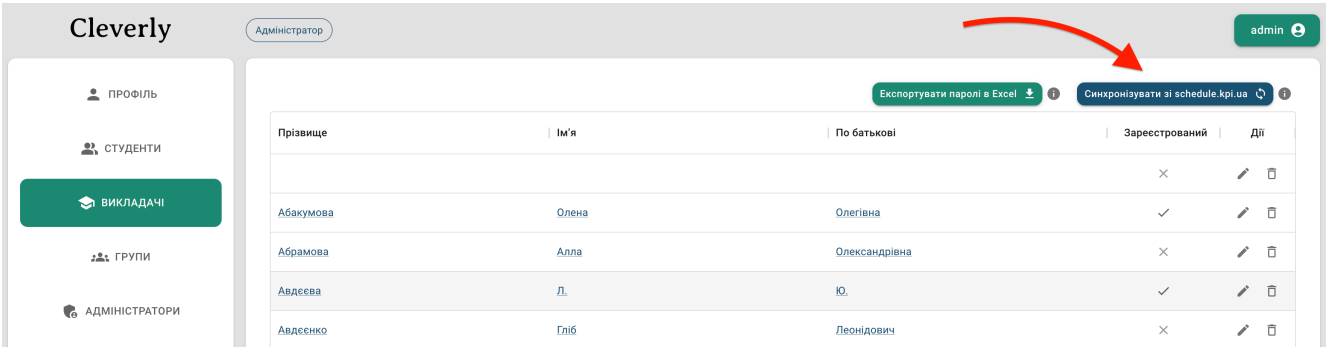


Рисунок 2.6 — Синхронізація викладачів у системі

Синхронізація груп відбувається приблизно тим самим способом, але за рахунок того, що кожна група у сервісі schedule.kpi.ua має назву факультету, у нас є можливість спарсити дані окремо в сутність групи і факультету. Синхронізація викликається адміністратором на сторінці груп як показано на рисунку 2.7.

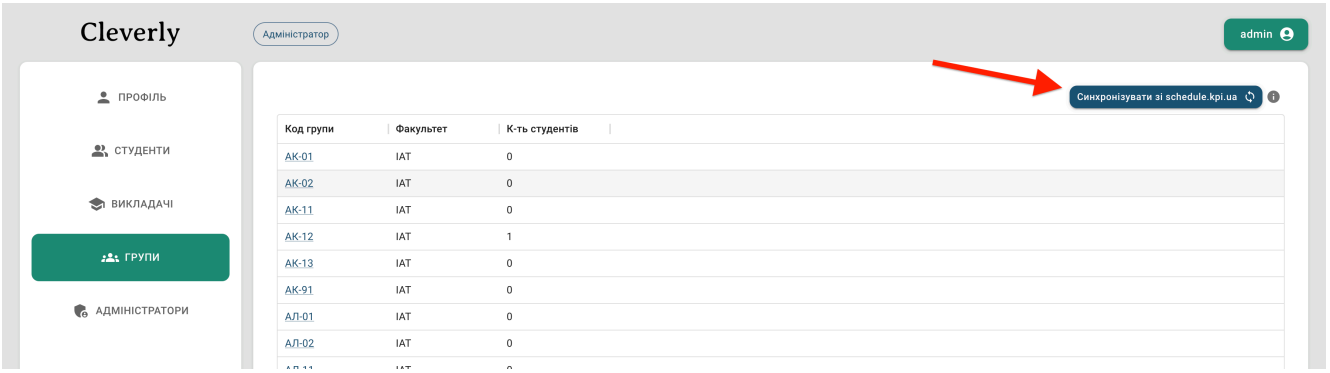


Рисунок 2.7 — Синхронізація груп

Після синхронізації до списку додадуться лише групи, яких нема в базі даних додатку.

2.3 Підключення до Google API

Перш ніж використовувати інструменти Google від імені користувачів, необхідно налаштувати доступ додатку до Google Sign-In у Google Developers Console.

Google Sign-In керує потоком OAuth 2.0 і життєвим циклом токену, спрощуючи вашу інтеграцію з Google API. Користувач завжди має можливість у будь-який момент скасувати доступ до програми.

2.3.1 Створення облікових даних авторизації

Будь-яка програма, яка використовує OAuth 2.0 для доступу до Google API, повинна мати облікові дані авторизації, які ідентифікують програму на сервері Google OAuth 2.0. Наступні кроки пояснюють, як створити облікові дані для проекту. Потім додаток зможе використовувати облікові дані для доступу до API, які ми ввімкнули для цього проекту.

1. Переходимо на сторінку облікових даних.
2. Натискаємо Створити облікові дані > Ідентифікатор клієнта OAuth як показано на рисунку 2.8.

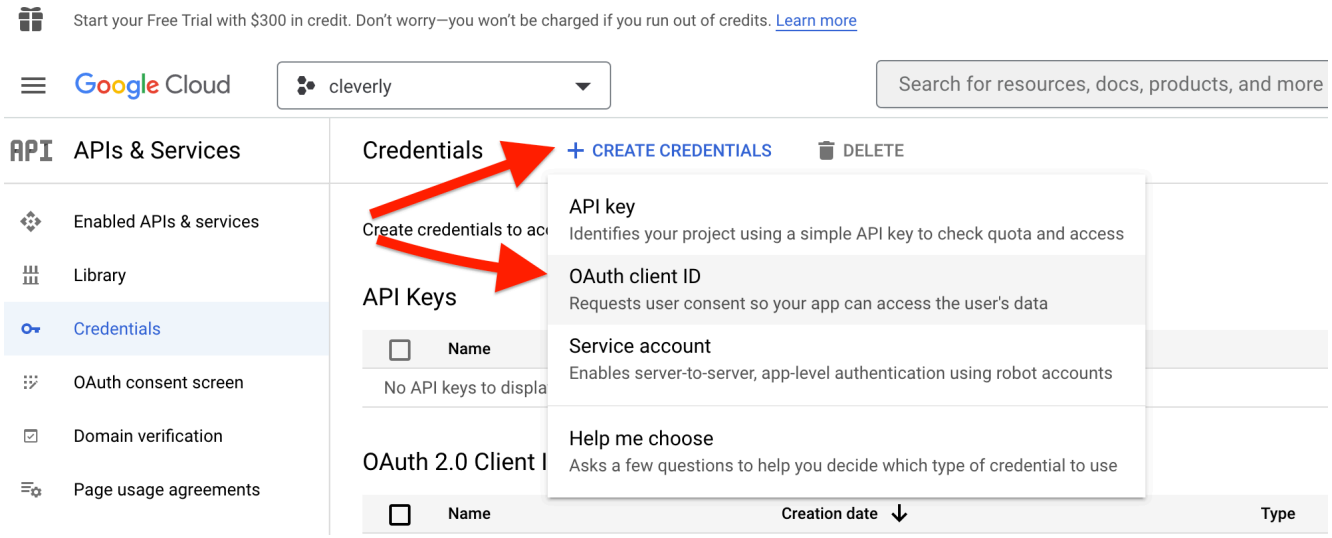


Рисунок 2.8 — Створення клієнту OAuth

3. Виберіть тип програми веб-програми.
4. Назвіть свій клієнт OAuth 2.0 і натисніть «Створити».
5. Додаємо посилання, на яких буде працювати клієнт як показано на рисунку 2.9.

Після завершення налаштування створюється ідентифікатор клієнта. Далі ідентифікатор нам знадобиться як для серверного додатку, так і для клієнтського

для роботи з Google. Також створюється секрет клієнта, але він потрібен лише для операцій на стороні сервера.

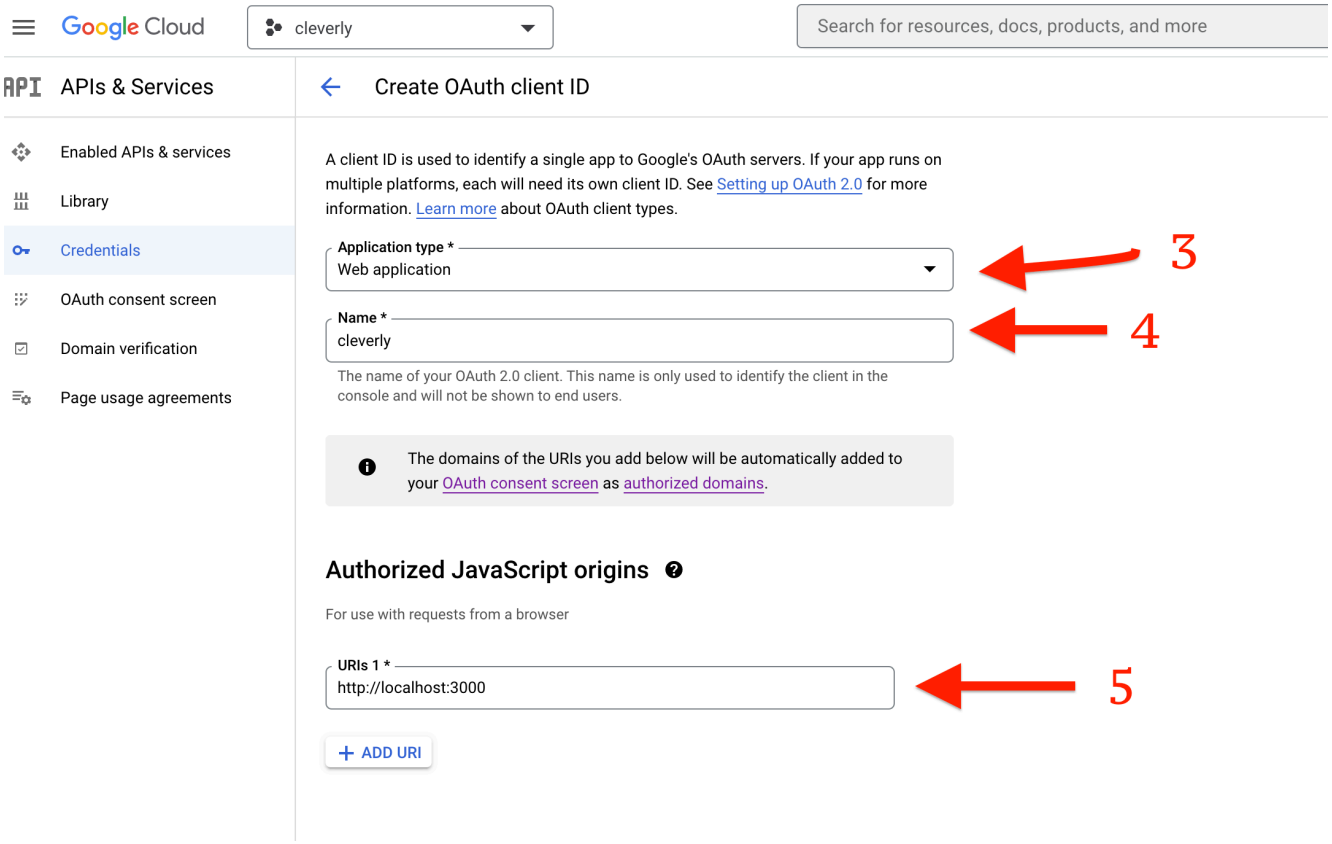


Рисунок 2.9 — Додання інформації про клієнт OAuth

Після створення облікових даних необхідно налаштувати тестових користувачів на екрані згоди OAuth.

2.3.2 Екран згоди OAuth

Тепер для користування додатком нам потрібні тестові користувачі. Їх потрібно додати на вкладці екрану згоди OAuth як показано на рисунку 2.10, всі інші налаштування можна опустити, але треба вказати тип додатку External. Проекти, налаштовані з типом користувача External, доступні для будь-якого користувача з обліковим записом Google. На можливість користувача авторизувати запитані області дії вашого додатка залежить від статусу публікації вашого проекту.

Рисунок 2.10 — Налаштування тестових користувачів на екрані згоди

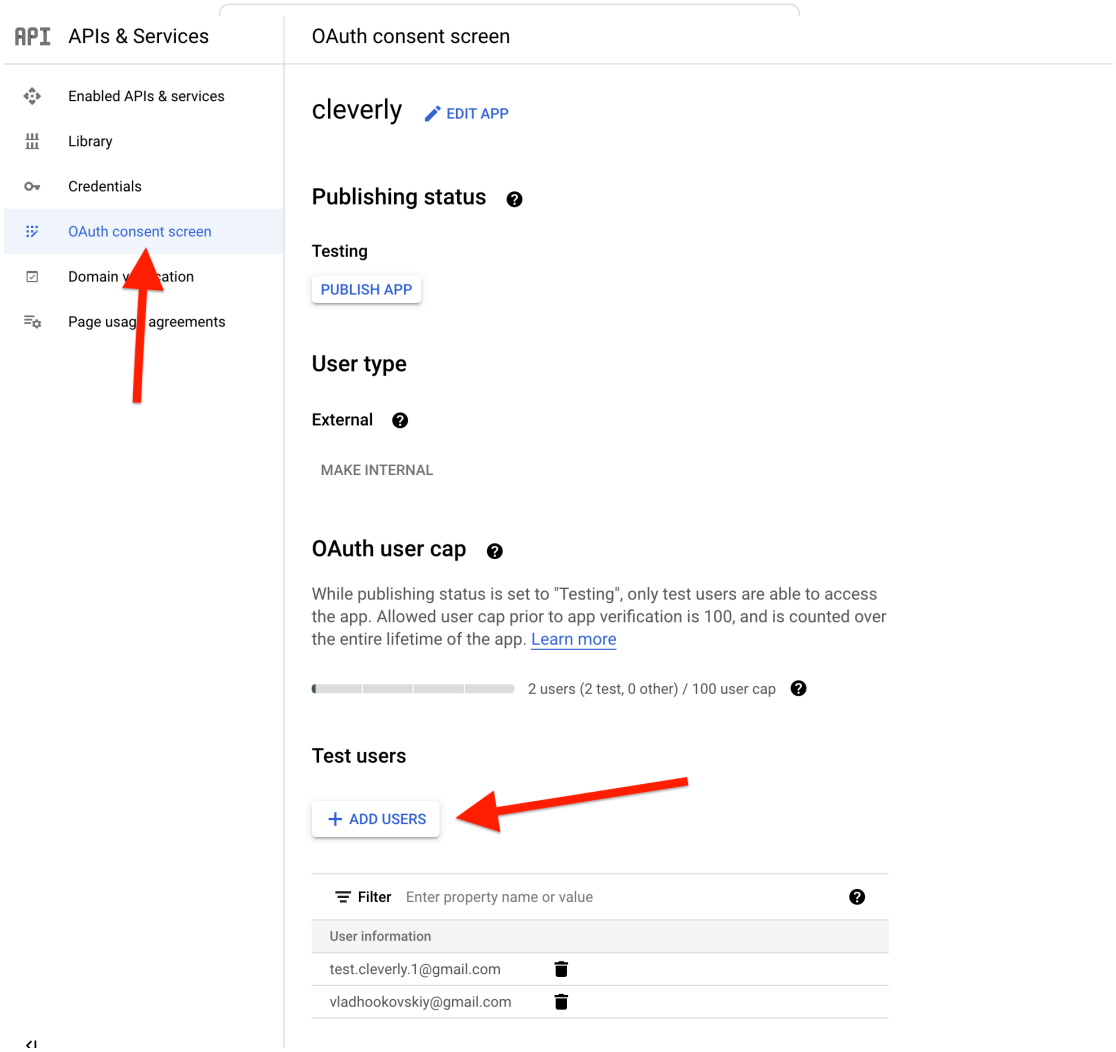
Без цих налаштувань, вікно авторизації не буде працювати при спробі підключити користувача за допомогою Google.

2.3.3 Надання користувачем прав на зміну в Google Classroom

Усі програми дотримуються основного шаблону під час доступу до Google API за допомогою OAuth 2.0. Для цього необхідно виконати п’ять кроків:

1. Отримати облікові дані OAuth 2.0 із Google API Console, що описао пунктом вище. Для цього треба відвідати Google API Console, щоб отримати облікові дані OAuth 2.0, як-от ідентифікатор клієнта та секрет клієнта, які відомі як Google, так і нашій програмі. Набір значень залежить від того, який тип програми ми створюємо. Наприклад, програма JavaScript не потребує секрету, але програма веб-сервера вимагає.
2. Отримати токен доступу на сервері авторизації Google. Перш ніж наша програма зможе отримати доступ до приватних даних за допомогою Google API, вона має отримати токен доступу, який надає доступ до цього API. Один токен доступу може надати різний ступінь доступу до кількох API. Змінний параметр під назвою scope контролює набір ресурсів і операцій, які дозволяє токен доступу. Під час запиту токену доступу наша програма надсилає одне або кілька значень у параметрі scope. Є кілька способів зробити цей запит, і вони відрізняються залежно від типу програми, яку ви створюєте. Наприклад, програма JavaScript може запитувати токен доступу за допомогою переспрямування браузера до Google, тоді як програма, встановлена на пристрої без браузера, використовує запити веб-служб. Деякі запити вимагають етапу автентифікації, на якому користувач входить у свій обліковий запис Google як показано на рисунку 2.11.

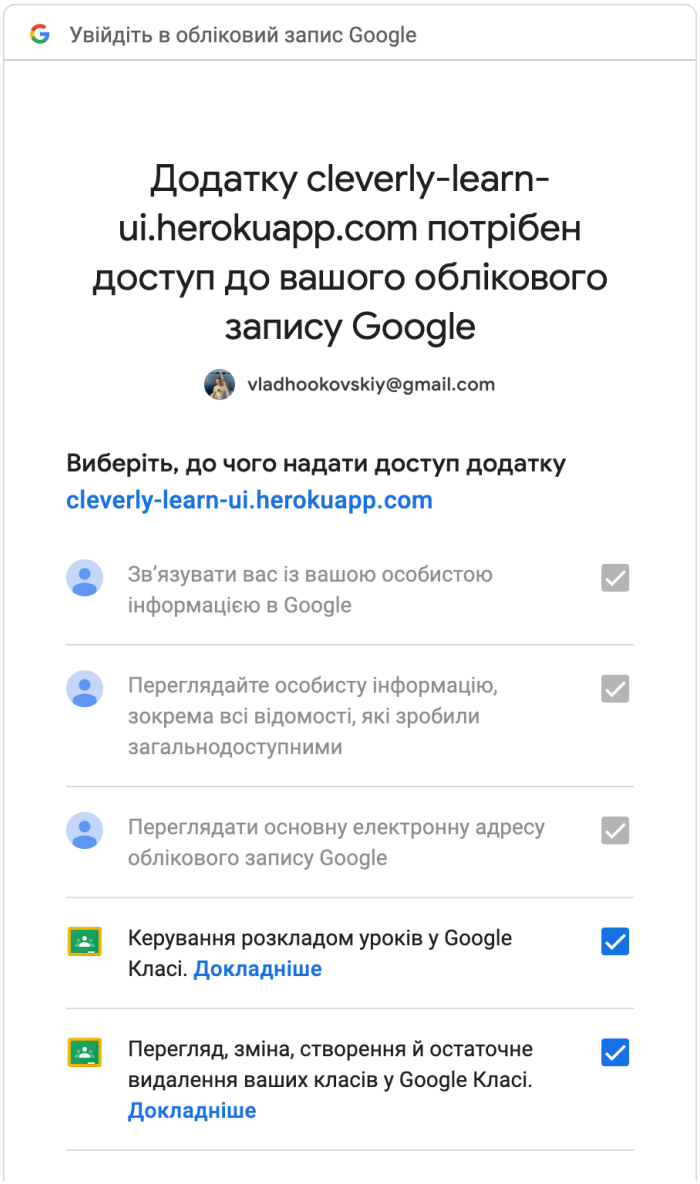
Рисунок 2.11 — Вікно запиту входу у свій обліковий запис від Google



Після входу в систему користувача запитують, чи бажають вони надати один або кілька дозволів, які запитує ваша програма як показано на рисунку 2.12.

Рисунок 2.12 — Вікно надання дозволу на використання сервісів

Цей процес називається згодою користувача. Якщо користувач надає принаймні один дозвіл, сервер авторизації Google надсилає вашій програмі токен доступу (або код авторизації, який ваша програма може використовувати для отримання токену доступу) і список областей доступу, наданих цим токеном. Якщо користувач не надає дозвіл, сервер повертає помилку. Загалом найкраще запитувати обсяги поступово, у той час, коли потрібен доступ, а не заздалегідь. Наприклад, програма, яка хоче підтримувати збереження події в календарі, не повинна запитувати доступ до Календаря Google, доки користувач не натисне кнопку «Додати до календаря». див. Додаткову авторизацію.



3. Вивчити області доступу, надані користувачем. Необхідно порівняти області, включені у відповідь токену доступу, з областями, необхідними для доступу до функцій і можливостей вашої програми залежно від доступу до пов'язаного API Google. Вимкніть будь-які функції вашої програми, які не можуть працювати без доступу до відповідного API. Область, включена у ваш запит, може не збігатися з областю, включеною у вашу відповідь, навіть якщо користувач надав усі запитані області. Зверніться до документації для кожного API Google, щоб дізнатися про обсяги, необхідні для доступу. API може відображати кілька значень рядка області дії в одну область доступу, повертаючи той самий рядок області для всіх значень, дозволених у запиті.

4. Надіслати токен доступу до API. Після того як програма отримує токен доступу, вона надсилає токен до API Google у заголовку запиту авторизації

HTTP. Можна надсилати токени як параметри рядка запиту URI, але ми не рекомендуємо цього, оскільки параметри URI можуть опинитися у файлах журналу, які не є повністю безпечними. Крім того, хороша практика REST [2] — уникати створення непотрібних імен параметрів URI. Токени доступу дійсні лише для набору операцій і ресурсів, описаних в області запиту токену. Наприклад, якщо токен доступу видається для Google Calendar API, він не надає доступу до Google Contacts API. Однак ви можете надіслати цей токен доступу до Google Calendar API кілька разів для подібних операцій.

5. Якщо необхідно, треба оновити токен доступу. Жетони доступу мають обмежений термін служби. Якщо вашій програмі потрібен доступ до Google API після закінчення терміну дії одного токен доступу, вона може отримати токен оновлення. Токен оновлення дозволяє вашій програмі отримувати нові токени доступу.

Для нашої програми крім стандартних прав на перегляд профілю користувача Google, нам також потрібні права на користування Google Classroom API, а саме:

• https://www.googleapis.com/auth/classroom.courses — перегляд курсів та управління ними в Google Classroom;

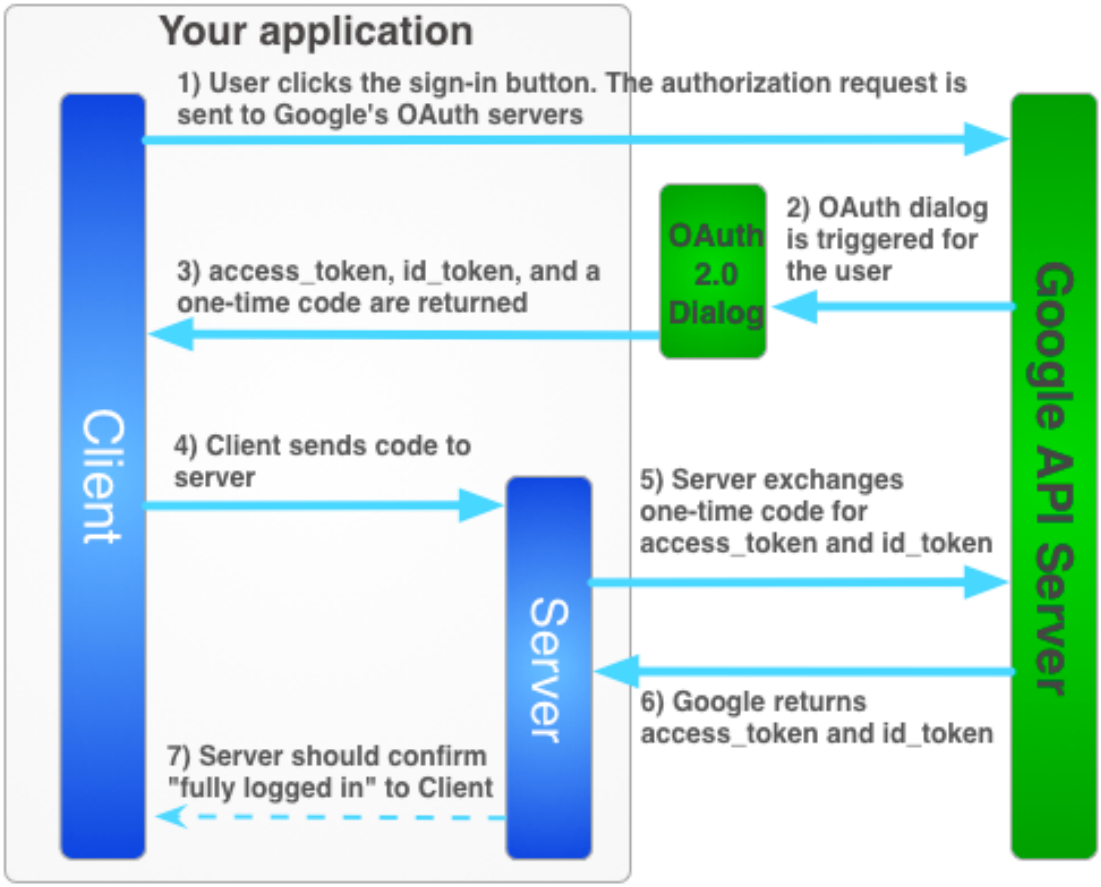
• https://www.googleapis.com/auth/classroom.rosters — керування списками своїх класів у Google Classroom.

Для актуальності токенів доступу, ми зберігаємо токен оновлення кожного користувача у базі даних.

Всі ці кроки нам необхідні для можливості спілкуватися із Google Classroom API від імені користувача на стороні сервера. Загалом це вигладає так, як показано на рисунку 2.13.

Рисунок 2.13 — Авторизація додатку у Google API

Після отримання прав доступу на використання Google Classroom API, ми можемо перейти до безпосередньої роботи із курсами.



2.3.4 Робота з Google Classroom API

Google Classroom API має безліч методів, але у нашому додатку ми будемо використовувати 3 з них:

- courses.create — створення курсу;
- courses.patch — оновити дані курсу. Використовуємо для архівування курсів при видаленні;
- invitations.create — створити запрошення для студента.

Тепер розглянемо використання кожного метода окремо.

1. Метод: courses.create — створює курс.

Користувач, указаний у ownerId, є власником створеного курсу та доданий як викладач. Користувач, який не є адміністратором і надсилає запит, може створити курс лише з власником. Адміністратори домену можуть створювати курси, які належать будь-якому користувачеві в їх домені. Цей метод повертає такі коди помилок:

• PERMISSION_DENIED — якщо користувачеві, який надіслав запит, не дозволено створювати курси або через помилки доступу;

• NOT_FOUND — якщо вчитель початкових класів не є дійсним користувачем.

• FAILED_PRECONDITION — якщо обліковий запис власника курсу вимкнено;

• ALREADY_EXISTS — якщо псевдонім було вказано в ідентифікаторі та вже існує.

HTTP POST запит: https://classroom.googleapis.com/v1/courses

Тіло запиту містить екземпляр курсу.

Екземпляр курсу складається з полів, які показані в таблиці 2.1:

Таблиця 2.1 — Екземпляр курсу

name	Назва курсу. Наприклад, «Біологія 10 клас». Необхідно вказати назву. Він має містити від 1 до 750 символів і дійсний рядок UTF-8.
section	Розділ курсу. Наприклад, «Період 2». Якщо встановлено, це поле має бути дійсним рядком UTF-8 і не містити більше 2800 символів.
descriptionHeading	Додатковий заголовок для опису. Наприклад, «Ласкаво просимо до біології в 10 класі». Якщо встановлено, це поле має бути дійсним рядком UTF-8 і не довше 3600 символів.
description	Додатковий опис. Наприклад, «Ми будемо вивчати будову живих істот із поєднання підручників, гостьових лекцій і лабораторних робіт. Якщо встановлено, це поле має бути дійсним рядком UTF-8 і не містити більше 30 000 символів.
room	Додаткове розташування кімнати. Наприклад, «301». Якщо встановлено, це поле має бути дійсним рядком UTF-8 і не містити більше 650 символів.

Таблиця 2.1 — Екземпляр курсу

ownerId	Ідентифікатор власника курсу. Якщо вказано як параметр create course request, це поле є обов’язковим. Ідентифікатор може бути одним із таких: • числовий ідентифікатор користувача; • адреса електронної пошти користувача; • рядковий літерал "me", що вказує користувача, який запитує. Це має бути встановлено в запиті на створення. Адміністратори також можуть указати це поле в patch course request, щоб передати право власності. В інших контекстах він доступний лише для читання.
creationTime	Час створення курсу. Вказівка цього поля в масці оновлення курсу призводить до помилки. Лише для читання.
	Позначка часу у форматі RFC3339 UTC з наносекундною роздільною здатністю та до дев’яти дробових цифр.
updateTime	Час останнього оновлення цього курсу. Вказівка цього поля в масці оновлення курсу призводить до помилки.
enrollmentCode	Реєстраційний код, який потрібно використовувати під час приєднання до цього курсу. Вказівка цього поля в масці оновлення курсу призводить до помилки.

Таблиця 2.1 — Екземпляр курсу

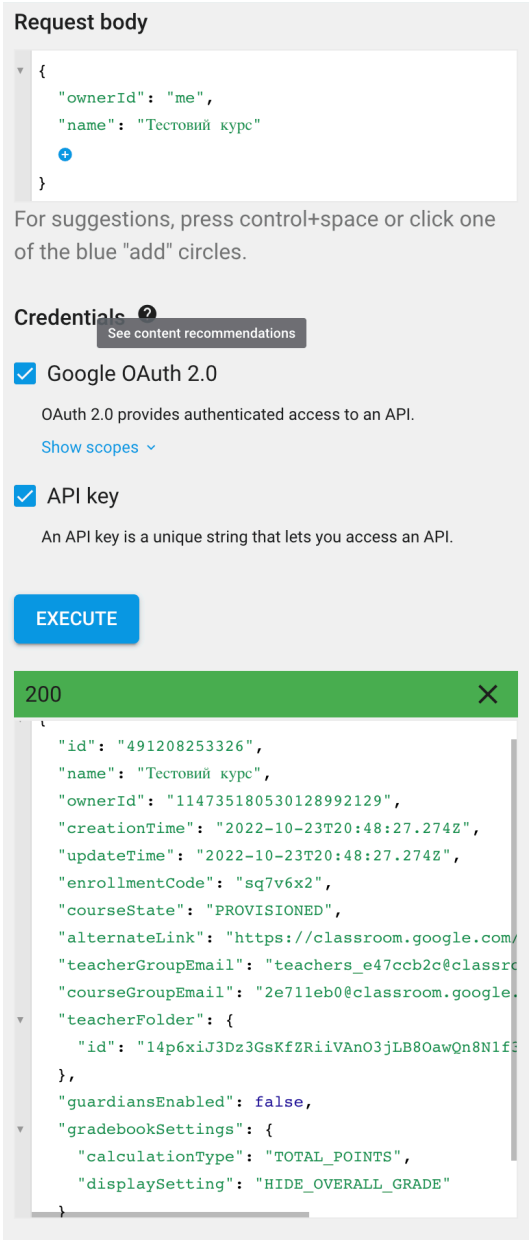
courseState	Стан курсу. Якщо не вказано, стандартним станом є PROVISIONED. Існує наступний перелік станів курсів: • COURSE_STATE_UNSPECIFIED — без стану курсу. Жодне повернуте повідомлення курсу не використовуватиме це значення; • ACTIVE — курс активний; • ARCHIVED — курс заархівовано. Ви не можете змінити його, окрім зміни стану; • PROVISIONED — курс створений, але ще не активований. Він доступний для основного вчителя та адміністраторів домену, які можуть змінювати його або змінювати на стани ACTIVE або DECLINED; • DECLINED — курс був створений, але відхилений. Він доступний для власника курсу та адміністраторів домену, але він не відображатиметься у веб-інтерфейсі користувача. Ви не можете змінити курс, окрім як змінити його на PROVISIONED стан;
	• SUSPENDED — курс призупинено. Ви не можете змінити курс, і лише користувач, ідентифікований за допомогою, ownerIdможе переглядати курс. Курс може бути переведено в цей стан, якщо він потенційно порушує Умови обслуговування
alternateLink	Повне посилання на цей курс у веб-інтерфейсі Classroom.
teacherGroupEmail	Електронна адреса групи Google, яка містить усіх викладачів курсу. Ця група не приймає електронну пошту та може використовуватися лише для дозволів.

Таблиця 2.1 — Екземпляр курсу

courseGroupEmail	Електронна адреса групи Google, яка містить усіх учасників курсу. Ця група не приймає електронну пошту та може використовуватися лише для дозволів.

Для створення курсу ми використовуємо лише його назву та ідентифікатор власника, але як можна помітити, курс можна створити з великим набором додаткових даних, які могли б розширити систему.

У разі успіху тіло відповіді містить щойно створений екземпляр курсу. На рисунку 2.14 представлене тіло відповіді на запит створення курсу до API Google.



Для створення курсу необхідна така область OAuth:

https://www.googleapis.com/auth/classroom.courses

2. courses.patch — оновлює одне або кілька полів у курсі.

Цей метод має такі ж коди помилок і таку ж структуру екземпляру курсу, як і метод створення курсу.

HTTP PATCH запит: https://classroom.googleapis.com/v1/courses/{id}

Параметри шляху:

• id — ідентифікатор курсу для оновлення. Цей ідентифікатор може бути ідентифікатором, призначеним Класом, або alias.

Рисунок 2.14 — Створення курсу за допомогою Google Classroom API

Параметри запиту:

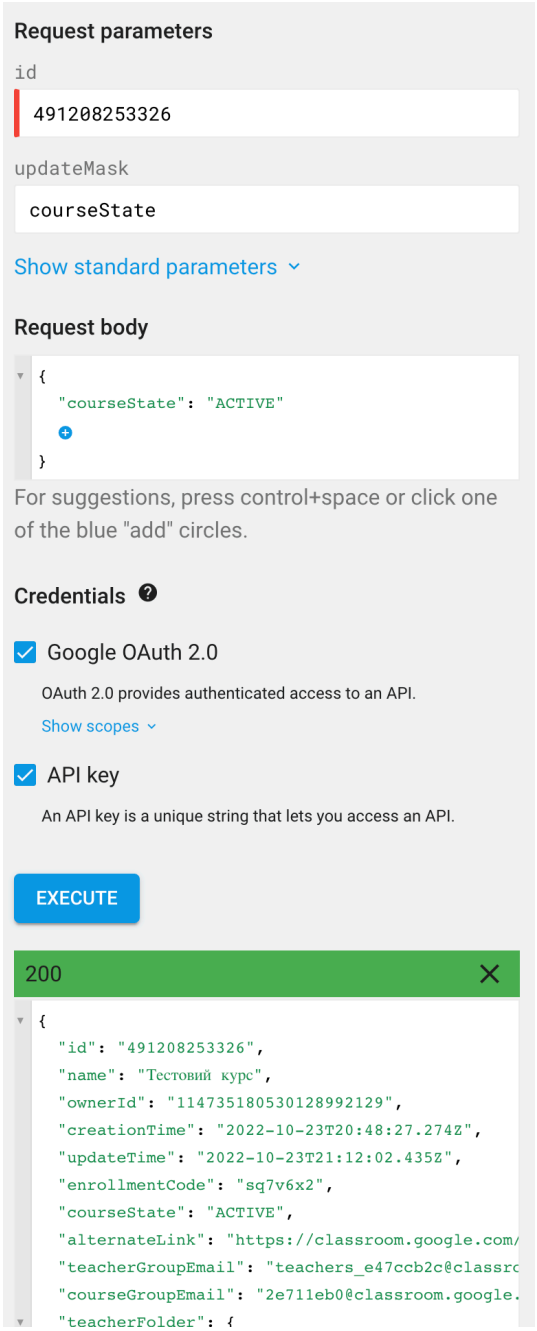
• updateMask — маска, що визначає, які поля курсу потрібно оновити. Це поле обов’язкове для оновлення. Оновлення не вдасться, якщо вказано недійсні поля. Наступні поля дійсні:

- name
- section
- descriptionHeading
- description
- room
- courseState
- ownerId

Примітка: виправлення для ownerId розглядаються як такі, що набувають чинності негайно, але на практиці може знадобитися деякий час для завершення передачі права власності на всі уражені ресурси.

Використання оновлення курсу на прикладі зміни статусу показано на рисунку 2.15:

Рисунок 2.15 — Приклад оновлення курсу



3. invitations.create — створює запрошення. Одночасно може існувати лише одне запрошення для користувача та курсу. Видаліть і повторно створіть запрошення, щоб внести зміни.

Цей метод повертає такі коди помилок:

• PERMISSION_DENIED — якщо користувачеві, який запитує, не дозволено створювати запрошення для цього курсу або через помилки доступу;

• NOT_FOUND — якщо курс або користувач не існує;

• FAILED_PRECONDITION — якщо обліковий запис запитуваного користувача вимкнено або якщо користувач уже має цю роль або роль із більшими дозволами;

• ALREADY_EXISTSякщо запрошення для зазначеного користувача та курсу вже існує.

HTTP POST запит: https://classroom.googleapis.com/v1/invitations

Тіло запиту містить екземпляр запрошення з полями на таблиці 2.2.

Таблиця 2.2 — Екземпляр запрошення

id	Ідентифікатор, призначений Classroom.
userId	Ідентифікатор запрошеного користувача. Якщо вказано як параметр запиту, цей ідентифікатор може мати одне з наступних значень: • числовий ідентифікатор користувача; • адреса електронної пошти користувача; • рядковий літерал "me", що вказує користувача, який запитує.
courseId	Ідентифікатор курсу, на який буде запрошено користувача.
role	Роль, яку потрібно запросити для користувача. Не повинно бути COURSE_ROLE_UNSPECIFIED. Роль може мати наступні значенні: • STUDENT — студент на курсі; • TEACHER — викладач курсу; • OWNER — власник курсу.

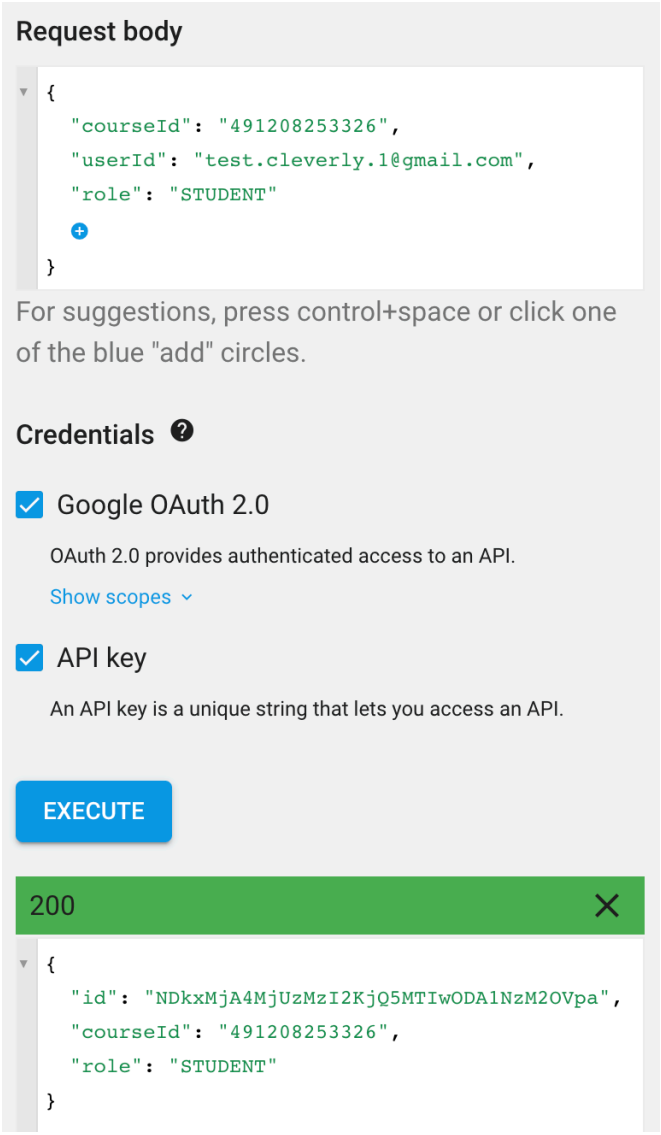
Тіло відповіді

У разі успіху тіло відповіді містить щойно створений екземпляр запрошення як показано на рисунку 2.16.

Рисунок 2.16 — Створення запрошення

Потрібна така область OAuth:

https://www.googleapis.com/auth/classroom.rosters



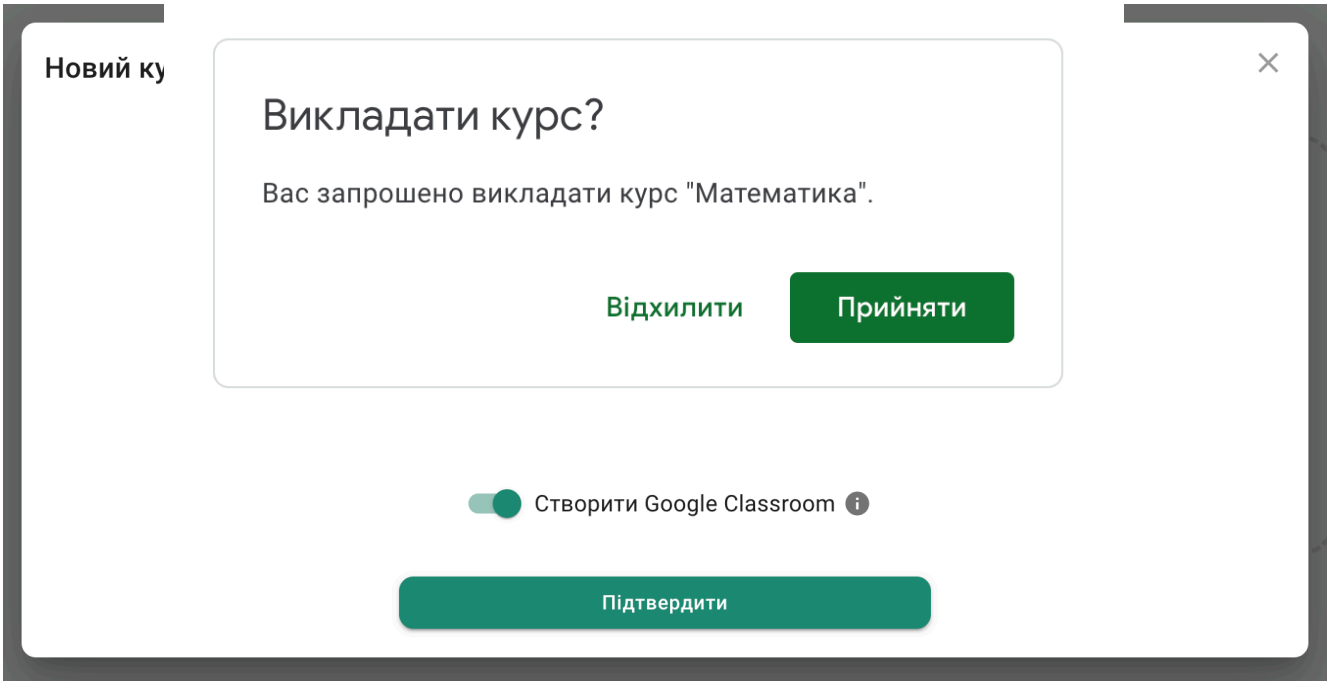
2.3.5 Приклад роботи з Google Classroom API у додатку

Після завершення реєстрації і початку роботи із системою, викладач може перейти у панель створення курсів. На початку там не створено жодного курсу, їх усі має створювати викладач, що і буде йому запропоновано. Як тільки викладача натисне кнопку створення курсу, йому буде запропоновано вибрати назву курсу, факультет та групи, студенти яких повинні бути запрошені до курсу у Classroom як показано на рисунку 2.17.

Рисунок 2.17 — Створення курсу

Після створення курсу, викладачу буде запропоновано підтвердити викладання у Google Classroom за посиланням як показано на рисунку 2.18.

Рис. 2.18 — Вікно підтвердження



Після підтвердження викладання, система автоматично запросить усіх зареєстрованих студентів вибраних груп до класрум. Ці студенти мають зайти до себе на пошту та підтвердити участь у курсі.

При видаленні викладачем курсу із системи, курс так само буде видалений із Google Classroom.

2.4 Використання серверу Google SMTP

Для надсилання листів із даними для реєстрації на електронну пошту, у додатку використовується SMTP (Simple Mail Transfer Protocol) сервер від Google.

SMTP — це протокол прикладного рівня. Клієнт, який хоче надіслати пошту, відкриває TCP-з'єднання з сервером SMTP, а потім надсилає пошту через з'єднання. Сервер SMTP є режимом постійного прослуховування. Щойно він прослуховує TCP-з'єднання від будь-якого клієнта, процес SMTP ініціює з'єднання через порт 25. Після успішного встановлення TCP-з'єднання клієнтський процес миттєво надсилає пошту.

Google SMTP сервер має наступні переваги над іншими сервісами:

• надійність — SMTP-сервер Google не використовує порт 25, щоб уникнути позначення спаму. Таким чином, це забезпечує кращу доставку електронної пошти;

• зручність — не потрібно налаштовувати власний сервер вихідної пошти (якщо ви використовуєте VPS);

• безпечний — для цього потрібна двофакторна автентифікація, а резервні копії електронних листів зберігатимуться на серверах Google. Крім того, Google вимагає від вас безпечного з'єднання для надсилання листів.

Для використання SMTP серверу від Google необхідний наступни мінімальний об'єм вхідних даних:

Transport: smtps://smtp.gmail.com

Username — пошта gmail користувача. Наприклад: example@gmail.com

Password — пароль від пошти. Якщо користувач gmail підключив подвійну аутентифікацію, то необхідно створити тимчасовий пароль для сервісу. Зробити це можна наступним чином як показано на рисунку 2.19:

1. Перейти у вкладку "Безпека" на головній сторінці акаунту.
2. Перейти у розділ "Паролі додатків" та створити пароль, за яким ваш додаток зможе авторизовуватися.

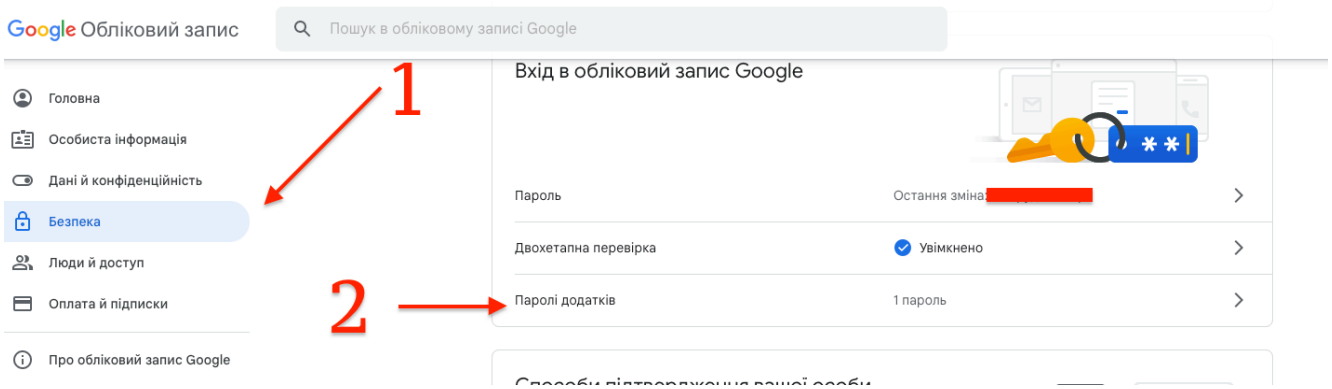


Рисунок 2.19 — Створення паролю додатку

Після цих дій додаток зможе самостійно надсилати листи іншим користувачам від імені акаунту, до якого було надано доступ.

Висновки до розділу 2

У цьому розділі було розглянуто методи вирішення поставленої задачі. Було досліджено та проаналізовано новизну методів та підхід до їх використання у системі навчання. Було продемонстровано використання інструментів Google API у тісній інтеграції із розроблюваною системою.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

У цьому розділі буде детально описано кожен крок проектування та розробки системи управління навчання. Буде описано архітектурні проблеми, з якими довелося стикнутися під час проектування та способи їх вирішення.

3.1 Дизайн макету додатку

Для додатку були розроблені наступні макети як показано на рисунку 3.1. Відповідно до дизайну, більшість свого часу користувач проводить на головній сторінці системи.

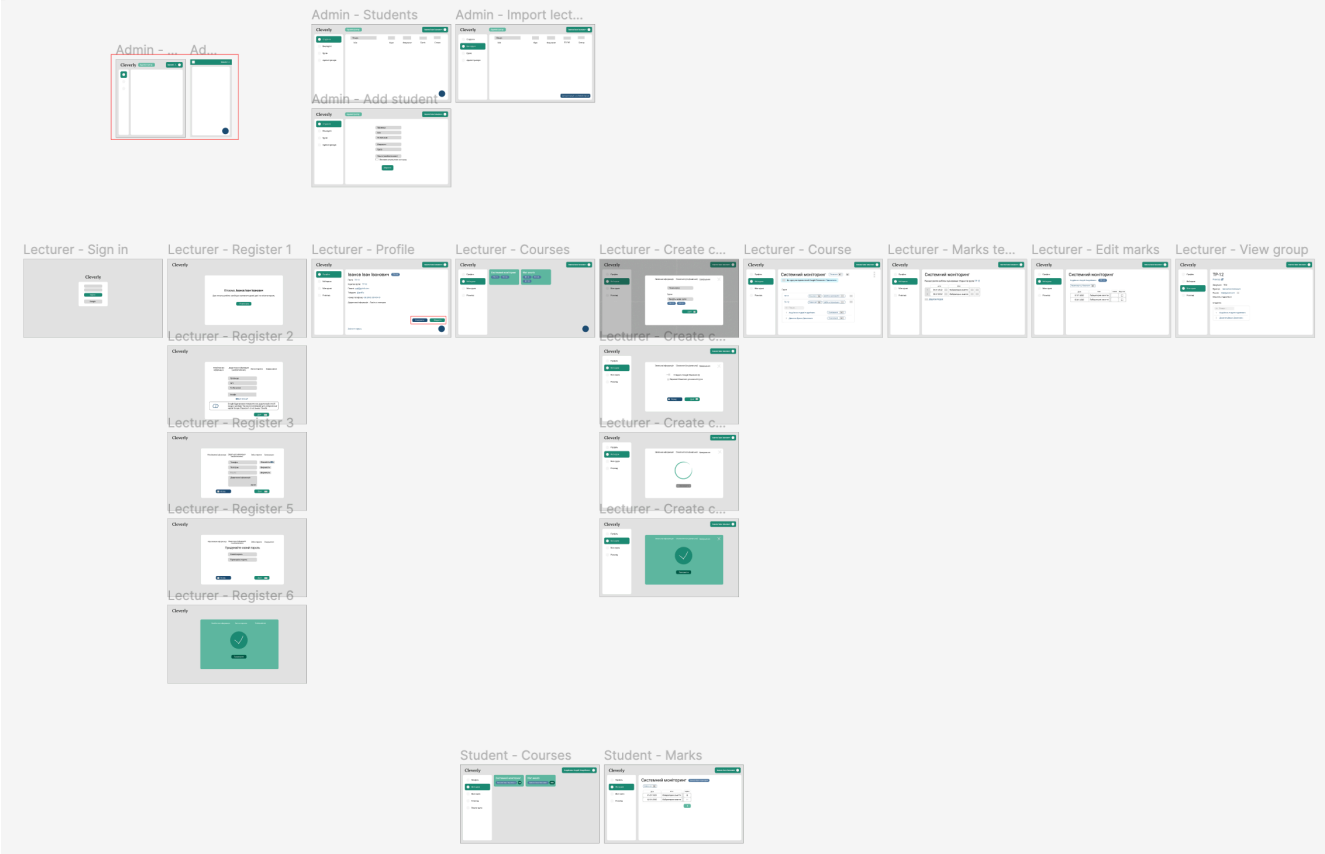


Рисунок 3.1 — Макети сторінок додатку у Figma

Розглянемо головну сторінку користувача. Для всіх ролей вона завжди складається з трьох елементів, показаних на рисунку 3.2:

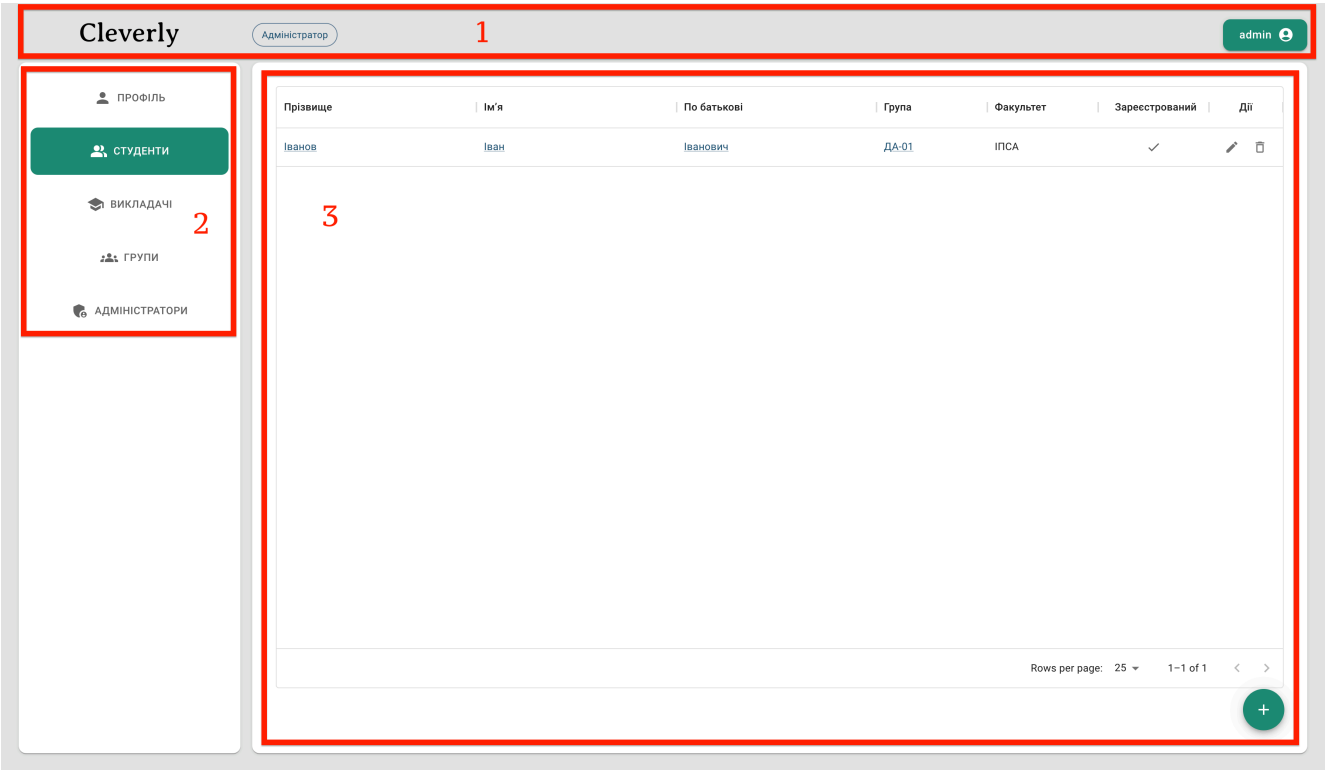


Рисунок 3.2 — Головна сторінка додатку

1. “Шапка” сторінки — тут відображається логотип додатку та повне ім’я користувача, при натисканні на який з’явиться спливаюче меню, через яке можна вийти з профілю.

2. Бічна панель — тут розташовується навігація користувача. При натисканні на кожен пункт, буде переключатися головний екран.

3. Головний екран — частина, на якій відображається запитувана користувачем інформація.

Як бачимо на прикладі вкладки зі студентами на рисунку 3.2, увага користувача акцентується на ключових елементах сторінки за допомогою головного кольору палітри, а саме на головному меню у шапці (1), навігації (2) та кнопці додання нового студента на гловному екрані (3).

До додаткових екранів додатку відносяться екрани входу до системи та екран реєстрації при першому вході у додаток з новоствореним логіном та паролем.

3.2 Структура бази даних

Додаток має одну централізовану SQL базу даних яка складається із 7 таблиць, як показано на рисунку 3.3. Усі користувачі мають одну загальну таблицю users, яка містить спільні колонки для всіх ролей. Для ролей студентів і викладачів передбачені окремі таблиці, які посилаються на загальну таблицю з користувачами. Таблиці груп і викладачів мають спеціальну колонку з посиланням на розклад зі schedule.kpi.ua. Так само таблиця користувачів та курсів мають колонки з даними для доступу до Google API.

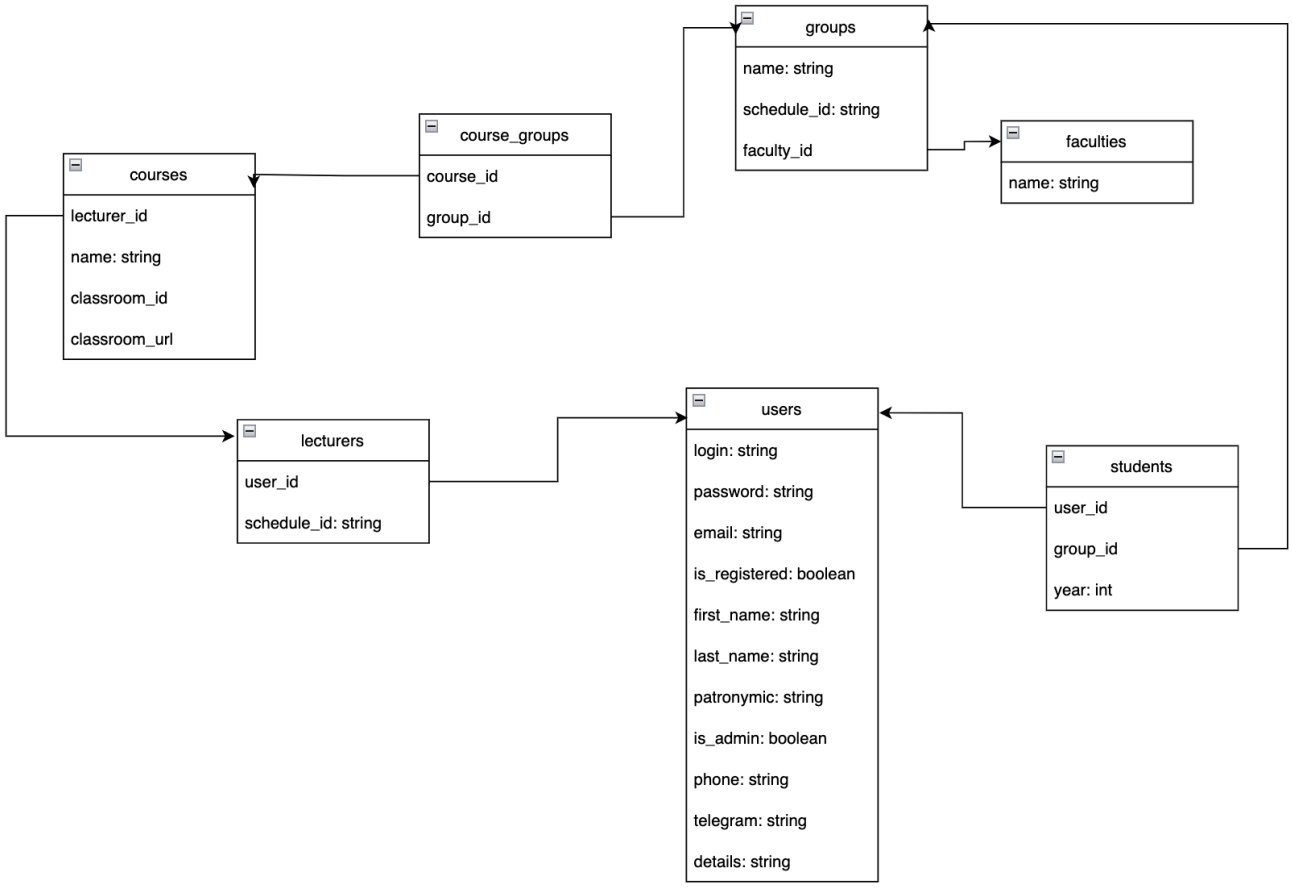


Рисунок 3.3 — Структура бази даних

Наведена схема бази даних спроектована таким чином, щоб у майбутньому можна було вільно додавати нові модулі системи не переробляючи минулу структуру. Наприклад, модуль виставлення оцінок міг би обійтися кількома новии таблицями з посиланнями на викладачів та студентів.

3.3 Автентифікація

Вхід у додаток проходить за допомогою логіну та пароля. Після цього сервер повертає пару “токен доступу”-“токен оновлення”, приклад якого показано на рисунку 3.4. Токен доступу має короткий час життя. Коли він стікає, необхідно оновити пару токенів надіславши на сервер токен оновлення.

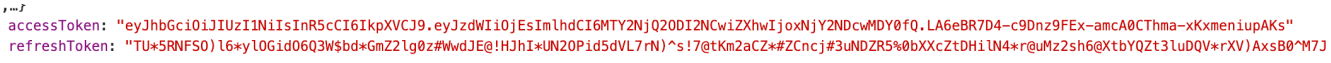


Рисунок 3.4 — Приклад пари токенів доступу

Токен доступу являє собою JWT token. Серед корисної інформації в ньому зберігається ідентифікатор користувача та час життя токену. Токен не зашифрований і може вільно зчитуватися на стороні клієнта.

Токен оновлення являє собою випадково згенерований набір симолів довжиною 500 і зберігається в базі даних до його використання або до того як в нього закінчиться час життся.

Система спроектована так, що користувачі не мають можливості в ній реєструватися. Логіни та паролі надаються студентам та викладачам адміністраторами одним із декількох способів. При першому запуску системи із чистою базою даних в системі одразу створюється адміністратор за замовчуванням зі стандартним логіном та паролем. Після цього адміністратор може змінити пароль на інший і додавати нових адміністраторів.

У разі якщо всі адміністратори втрачають пароль, то можна відновити адміністратора за замовчуванням видаливши його із бази даних вручну. Після перезапуску додатку буде додано адміністратора за замовчуванням із стандартним логіном та паролем.

Усі паролі хешуються перед занесенням у базу даних. Це означає, що їх неможливо зчитати, якщо база даних буде зчитана зловмисниками.

Згенерувати паролі можна наступним способом. Для викладачі передбачена функція експорту логінів та паролів у файл ексель як показано на рисунку 3.5.

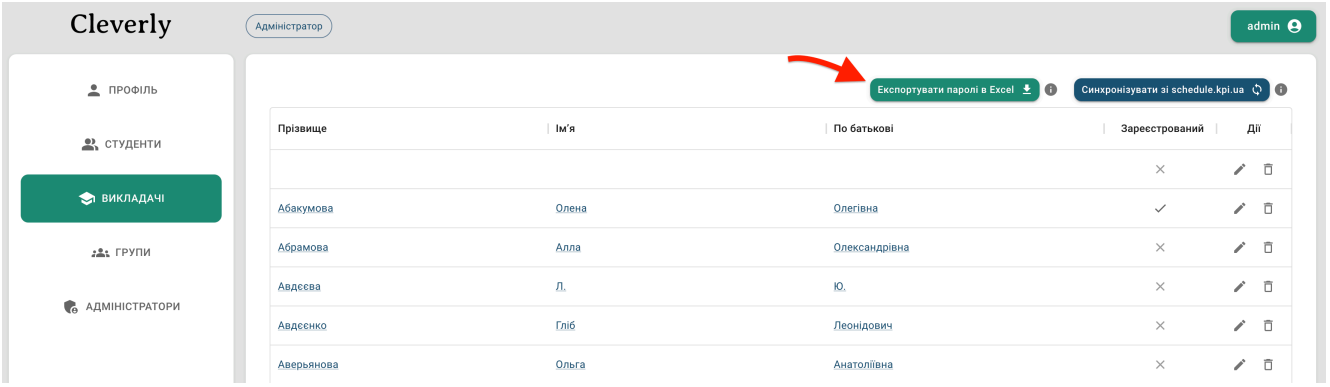


Рисунок 3.5 — Експорт паролів викладачів

Після цього адміністратор має передати це файл викладачам. Приклад файлу зображено на рисунку 3.6.

ФІО	Логін	Пароль
	hrVqZdofMQ	6OaNT&6m6&
Абакумова Олена Олегівна	aZyCFerOPL	T6a8b3A*g1
Абрамова Алла Олександрівна	xKatmUtrZf	8q#dtfwKlq
Авдєєва Л. Ю.	REtBZOehcE	Jn3R7E9%s2
Авдєєнко Гліб Леонідович	ymfLxxxBPG	kwI8bT!8yw
Аверьянова Ольга Анатоліївна	YRfROkIJGs	YKyWdSQ7yJ
Аврутов Вадим Вікторович	EHYZDqmjrq	I!UqvsBZpb
Адаменко Володимир Олексійович	yYOeYGHRny	K5Ltty&He#
Адаменко І. О.	jaImNRydRg	A*Wcrslnnn
Адаменко Юлія Федорівна	tWWKNdSHaU	vOghnY9UNa
Адаменко Юрій Іванович	CCoEQwbeBw	W^r3LPc6W3
Алексеїк Євгеній Сергійович	XsycUKSdhZ	0uD&&RSwew
Алексейчук Леся Борисівна	VuGdLAFPmW	S&28HcNPm%
Алексейчук Ольга Миколаївна	syjELAdgYC	PSz&7FvSQM
Алексенко Віталія Федорівна	mfqnhuqGQH	WJYPCBTpB(
Алещенко Олексій Вадимович	ZppAERvYME	&JykK98^3i
Алєксєєв М О	BkddKhSXZB	Cg5E1)2GEP
Алєксєєва Ірина Віталіївна	TmxEpSWZYP	zjKRwE3$dl
Алєнін О. І.	rxUkGTAMqd	342YMSAlS2
Алхімова Світлана Миколаївна	muGxMFoyly	58Xb)cuEf0

Рисунок 3.6 — Файл із логінами

Для студентів процедура виглядає іншим способом. При створенні студента, адміністратор має ввести його пошту (рис. 3.7), після чого логін та пароль будуть надіслані студенту на пошту, як показано на рисунках 3.7 і 3.8.

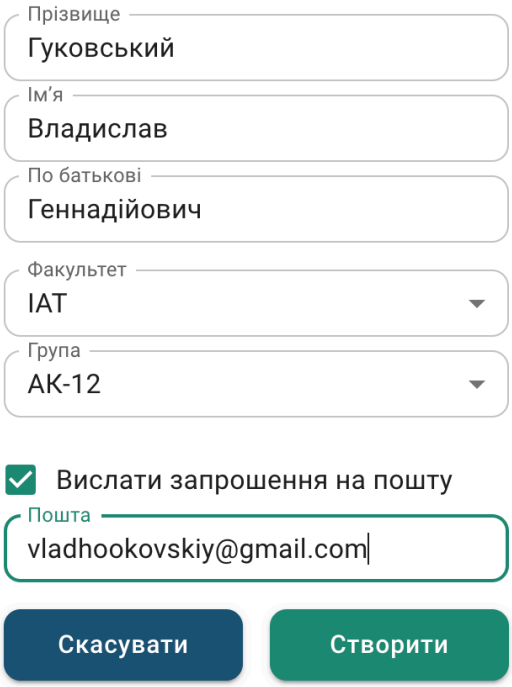


Рисунок 3.7 — Надсилання даних для реєстрації на пошту студента

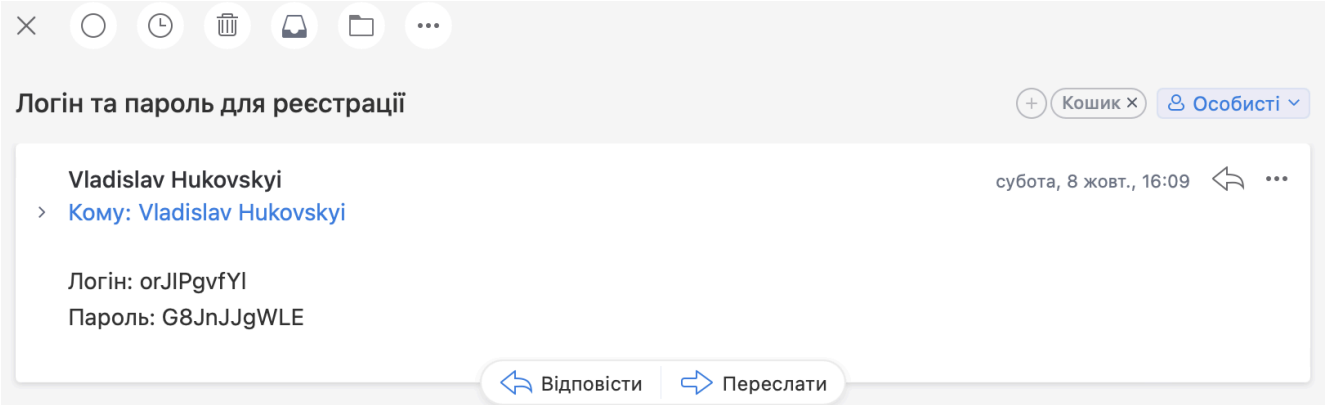


Рисунок 3.8 — Лист із даними для входу для студента

Після того як користувачі входять до системи за допомогою отриманого логіну та паролю, вони мають обов'язково змінити пароль для завершення реєстрації як показано на рисунку 3.9.

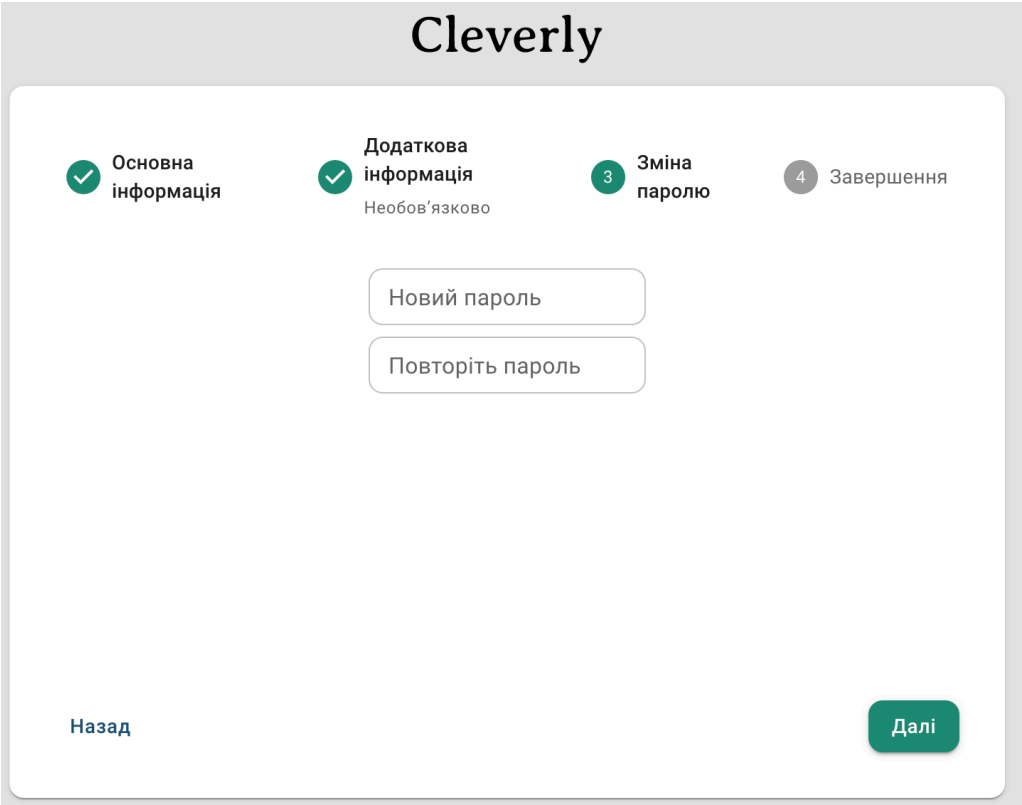


Рисунок 3.9 — зміна паролю при реєстрації

Це додаткова міра безпеки, яка унеможливлює вхід до системи шахраями, якщо файл із логінами та паролями був втрачений.

3.4 Авторизація та ролі користувачів

Система управління навчанням має 3 ролі користувачів: адміністратор, викладач та студент. Для всіх ролей передбачений єдиний інтерфейс користування додатком але різний рівень доступу. Єдиним відмінним елементом інтерфейсу є бічна панель, яка має різний набір навігаційних кнопок в залежності від ролі, як показано на рисунку 3.10. Для адміністратора це вкладки профілю, студентів, викладачів, груп та адміністраторів; для студента — профіль, курси, група та розклад; для викладача — профіль, курси та розклад.

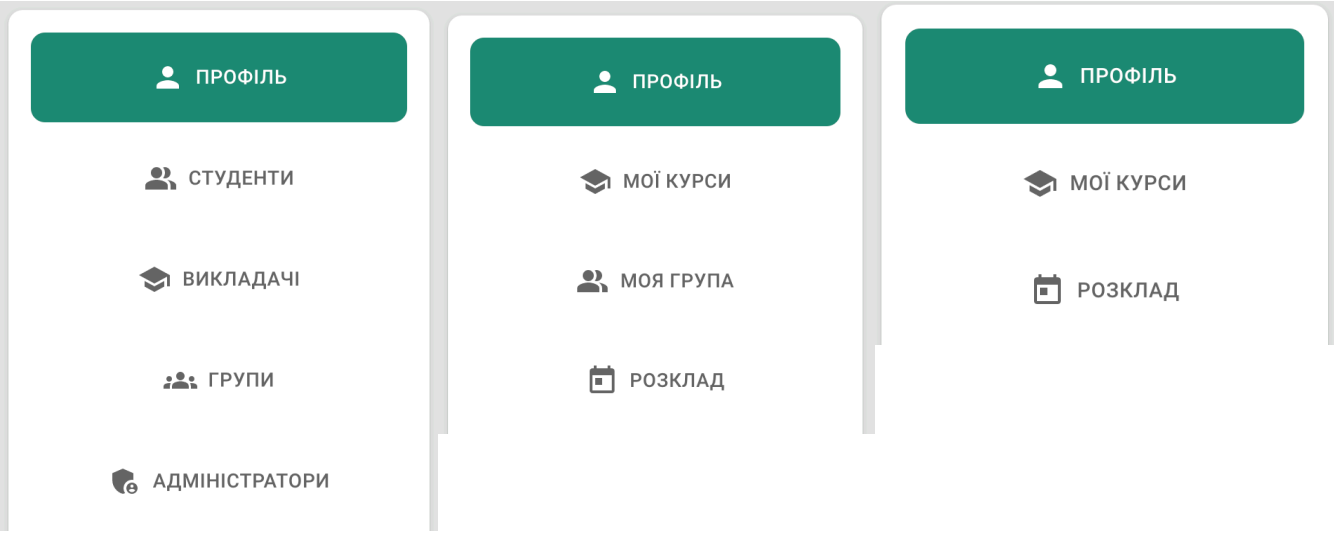


Рисунок 3.10 — Навігаційні панелі адміністратора, студента та викладача

Перевірка рівню доступу відбувається під час прийому сервером запиту. В першу чергу додаток перевіряє чи є користувач адміністратором, а вже після цього перевіряє чи є користувач у таблицях студентів або викладачів. Роль користувача відсилається назад на клієн, щоб він мав змогу правильно відобразити інтерфейс користувача.

3.5 Архітектура системи

Серверний додаток має модульну архітектуру, яку пропонує фреймворк Nest.js. Кожен модуль має всередині власні залежності та підключається до інших за допомогою вбудованій у фреймворк ін’єкції залежностей. Кожен модуль має власний набір сутностей, контролерів, сервісів та тестів до них як показано на рисунку 3.11.

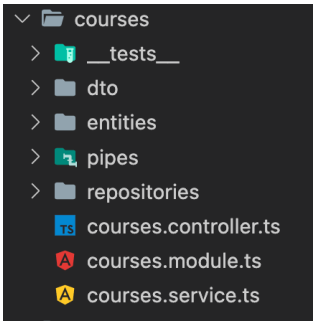


Рисунок 3.11 — Модуль курсів

Клієнтський веб додаток має компонентну структуру. Кожен компонент це незалежна атомарна одиниця, яка може використовувати власний набір елементів, утиліт, стилів та сервісів всередині, як показано на рисунку 3.12.

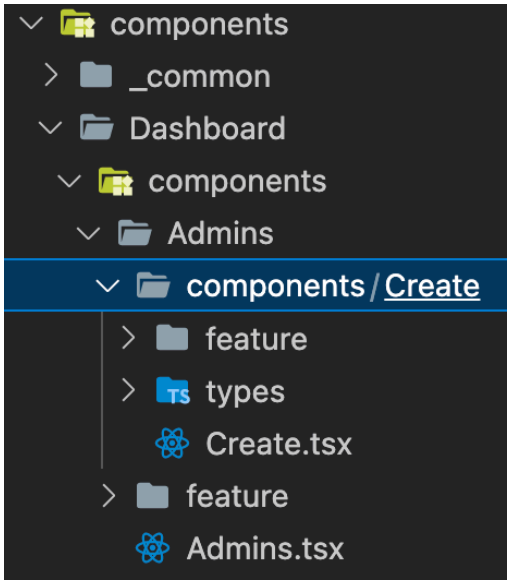


Рисунок 3.12 — Вкладена структура незалежних компонентів

Кожен компонент може бути перевикористаний, а видалення однорівневих компонентів не повинно впливати на стан інших компонентів.

3.6 Характеристика стану виконання ПЗ

Продукт виконаний з дотриманням високої рівня якості програмного коду. Було дотримано багато кращих практик по роботі з використаними бібліотеками та фреймворками.

Архітектура проекту побудована таким чином, що в систему буде дуже легко додавати нові модулі (наприклад виставлення оцінок, заліків, робота з кураторами, перегляд пошти групи). Готова інтеграція Google API відкриває широкі можливості по вдосконаленню проекту. Наприклад, у майбутньому у систему можна додати можливість автоматичного створення пошти групи, парсити листи і напряму відображати пошту у систему. Це саме стосується і schedule.kpi.ua, інтеграція з яким дозволить додати власний користувацький інтерфейс для розкладу. База даних побудована таким чином, щоб сутності чітко

відокремлювалися один від одного і передбачали безпечне додання нових таблиць у майбутньому.

У кодову базу додані строгі правила по форматуванню й аналізу коду, які недопустять внесення критичних помилок до системи.

Усі модулі серверного додатку ретельно покриті юніт тестами, що унеможливлює поламку вже створених модулів при доданні нового функціоналу.

Для проекту налаштована CI/CD, яка автоматично проходить всі тести системи і розгортує проект на платформі Heroku. Таким чином розробник має змогу не відволікатися від написання функціоналу, всі низькорівневі операції по розгортці будуть виконані автоматично при відправленні коду у центральний репозиторій GitHub.

Висновки до розділу 3

У цьому розділі було наведено архітектурні рішення у розробці та дизайні додатку. Було розглянуто метод автентифікації та розподілення ролей у системі. Також було продемонстровано структуру бази даних та охарактеризовано розроблений продукт. Було покроково оглянуто етапи розробки та реалізації програмного продукту.

4 РОБОТА КОРИСТУВАЧА З ПРОГРАМОЮ

У цьому розділі буде розглянуто взаємодію кінцевих користувачів із програмою, а саме: як отримати доступ, процес реєстрації і безпосередню роботу всередині системи. Також буде наведено вимоги системи до середовища, а також необхідні засоби для запуску системи в цілому.

4.1 Інсталяція

Програмний продукт являє собою веб додаток із двох модулів — серверна частина і клієнтська (браузер). Обидва проекти можна розгорнути на будь-якій хмарній платформі. Під час розробки була використана безкоштовна платформа Heroku, яка включає в себе як середовище для розгортки, так і базу даних як показано на рисунку 4.1.

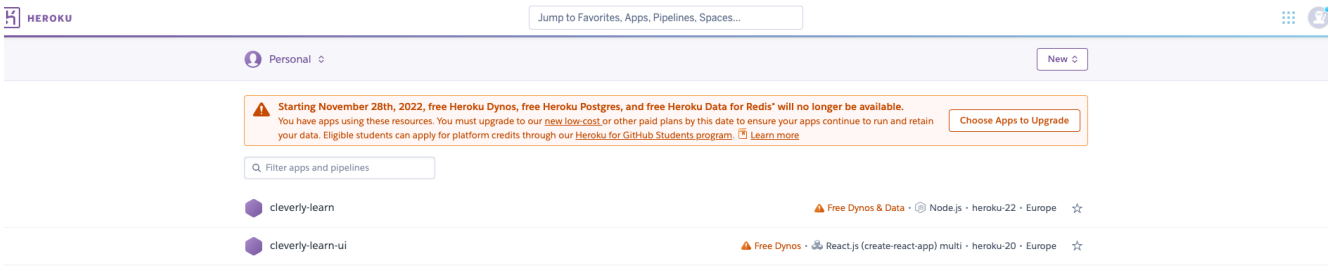


Рисунок 4.1 — Модулі веб-додатку на платформі Heroku

Також для роботи додатку з Google API необхідно налаштувати проект на Google Cloud платформі та налаштувати SMPT сервер. Цы налаштування були описані у минулих розділах.

Для локальної розгортки бази даних достатньо мати встановлений Docker, де можна створити контейнер з чистою базою даних.

4.2 Вимоги до обчислювальної техніки

Розробка проводилася на ноутбуці з характеристиками як показано на рисунку 4.2, але цілком допускається використання слабшої машини, так як програма не є ресурсозатратною.

Рисунок 4.2 — Опис системи, на якій велася розробка

Розробка проводилася у середовищі Node.js версії 16.10.0. У якості бази даних використовувалася СУБД PostgreSQL.

4.3 Демонстрація функціоналу

При першому заході на сторінку додатку, користувач побачить перед собою сторінку входу як показано на рисунку 4.3.

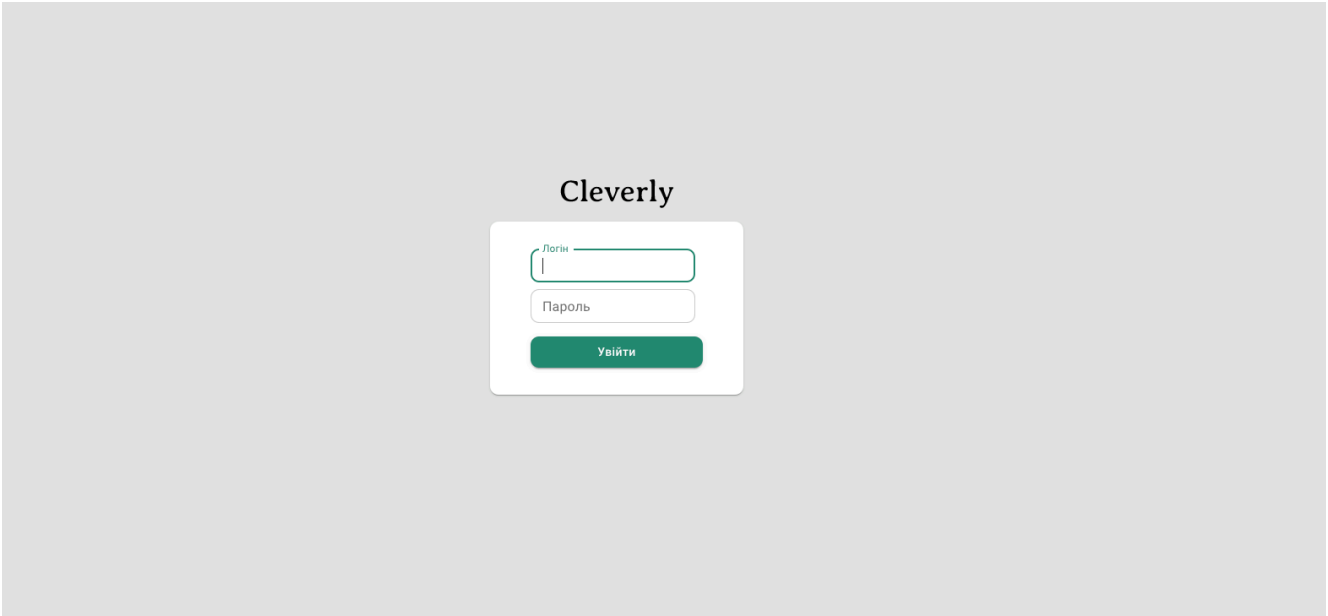


Рисунок 4.3 — Сторінка входу

Розглянемо роботу користувача від імені кожної із ролей системи.

4.3.1 Інтерфейс адміністратора

Будь-який користувач при вході у систему побачить свій профіль, який є однаковий для всіх ролей як показано на рисунку 4.4.

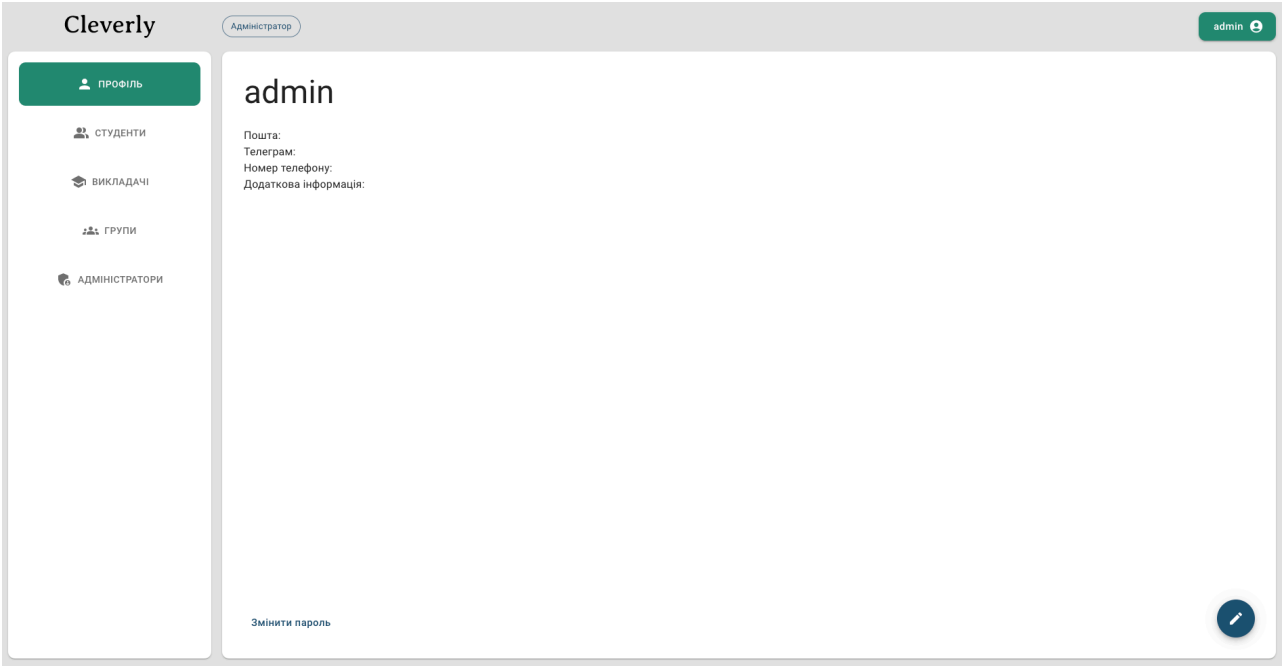


Рисунок 4.4 — Профіль користувача

На цій сторінці користувач може переглянути основну інформацію про себе та змінити її як показано на рисунку 4.5. Таку ж є сторінку можна побачити при перегляді сторінки іншого користувача (студента або викладача).

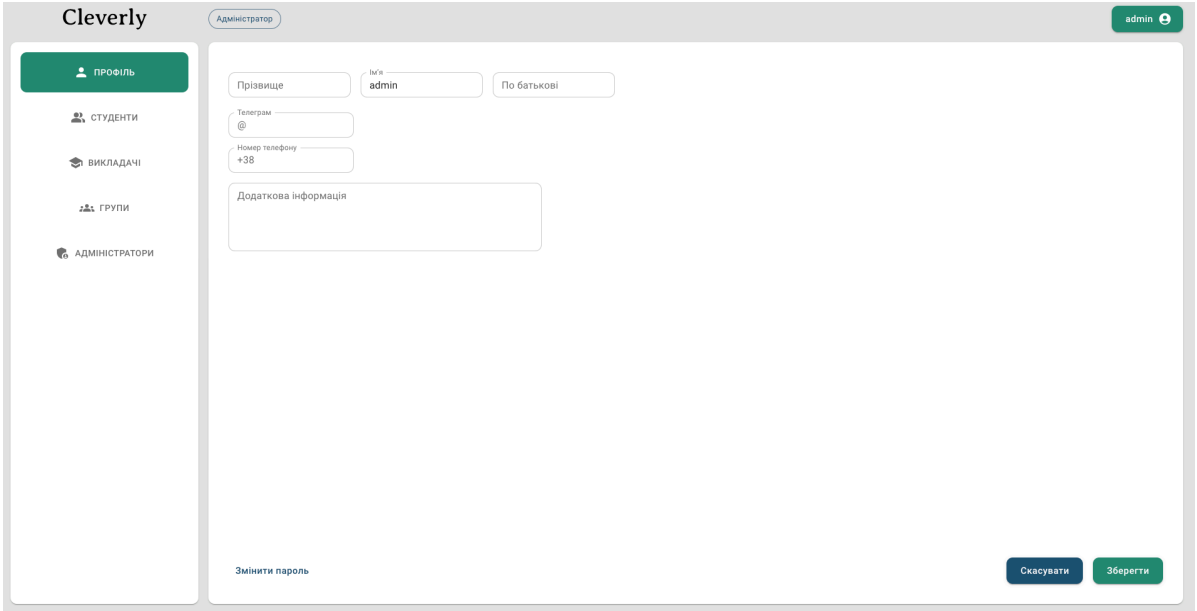


Рисунок 4.5 — Редагування профілю

Також на цій сторінці можна змінити свій пароль клікнувши на кнопку у нижньому лівому куті як показано на рисунку 4.6.

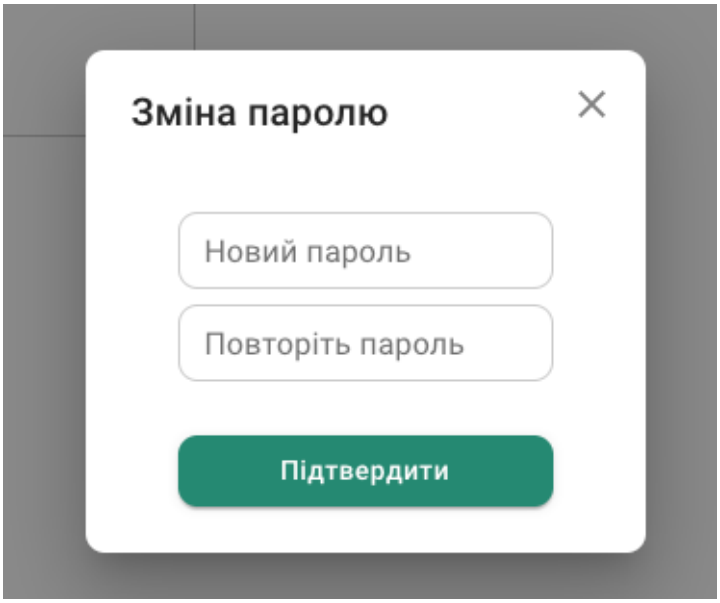
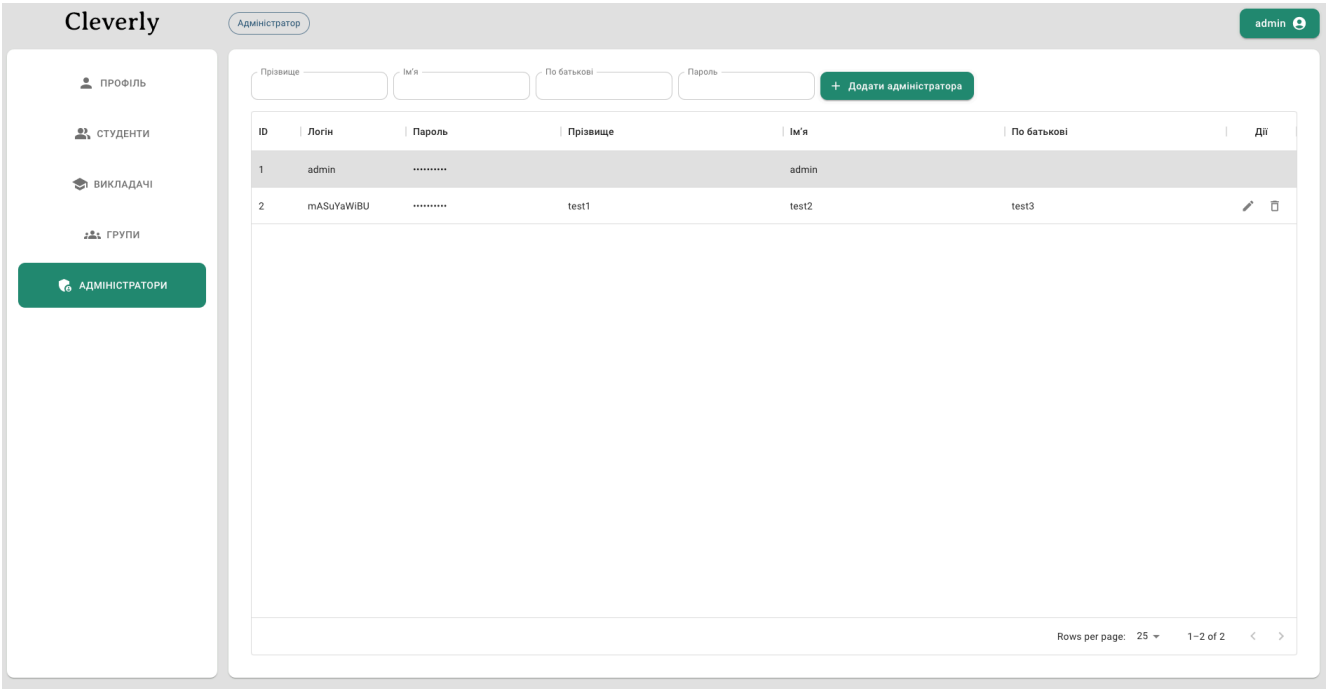


Рисунок 4.6 — Зміна паролю

На сторінці адміністраторів можна додавати нових адміністраторів, змінювати існуючих або видаляти їх як показано на рисунку 4.7.



Риснок 4.7 — Сторінка адміністраторів

На сторінці викладачів як показано на рисунку 4.8 можна побачити перелік усіх викладачів, які були синхронізовані зі schedule.kpi.ua. На цій же сторінці можна помітити кнопку синхронізації та експотру паролів.

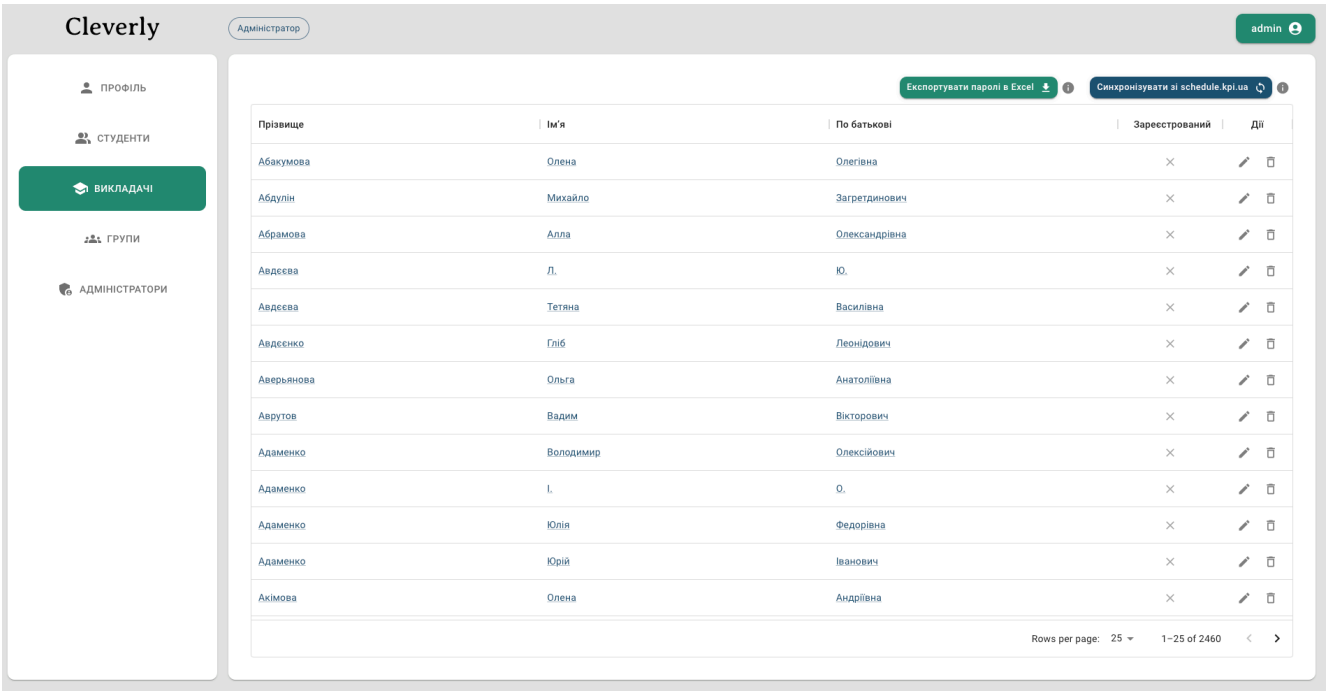


Рисунок 4.8 — Сторінка викладачів

На сторінці студентів, як показано на рисунку 4.9, адміністратор може побачити перелік доступних студентів. На даний момент студентів у системі не зареєстровано.

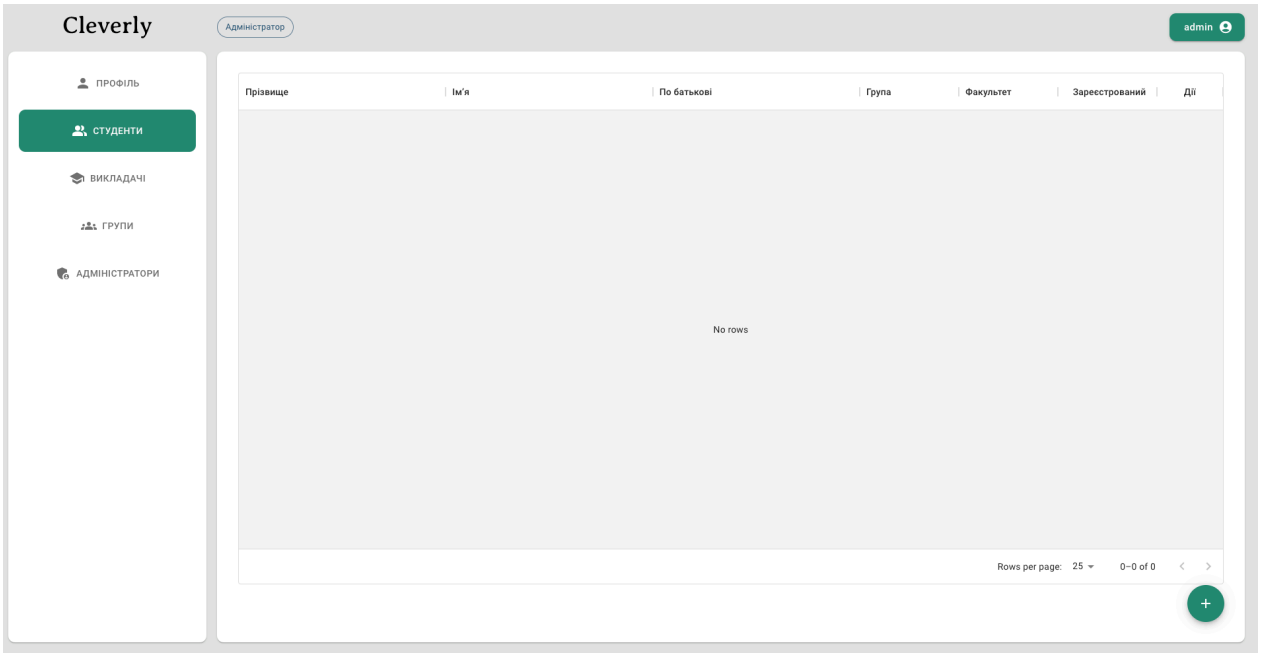


Рисунок 4.9 — Список студентів

При настисканні кнопки “Плюс” у правому нижньому кутку, відкриється форма реєстрації нового студента у системі як показано на рисунку 4.10.

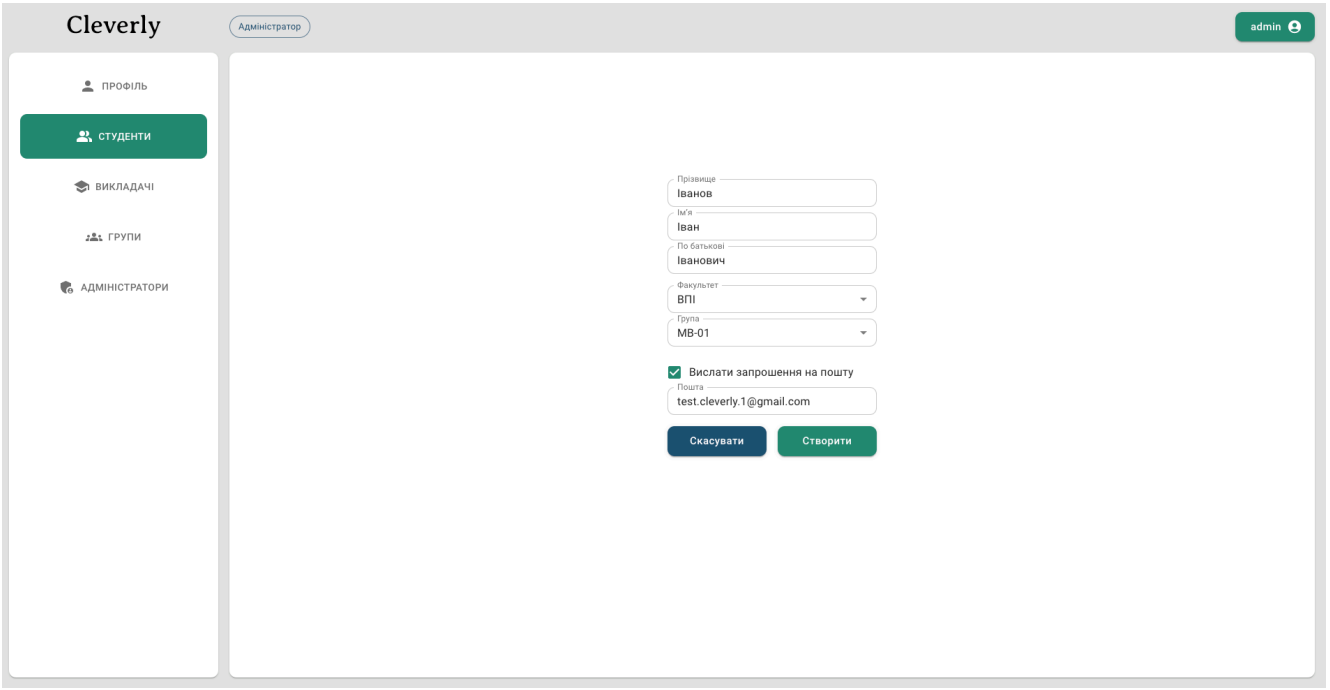


Рисунок 4.10 — Створення студента

Після створення студента, йому буде надіслано запрошення на пошту, а в списку студентів з’явиться новий запис як показано на рисунку 4.11. Цей запис можна редагувати або взагалі видалити.

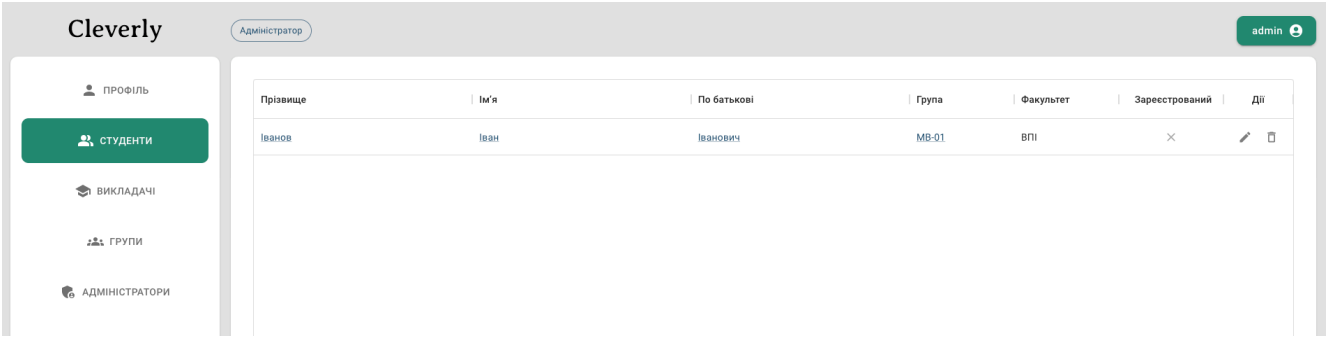


Рисунок 4.11 — Список із доданоим студентом

На сторінці з групами можна побачити перелік усіх груп, їх факультет і кількість студентві як показано на рисунку 4.12. Групи будуть відображатися лише після їх синхронізації із schedule.kpi.ua.

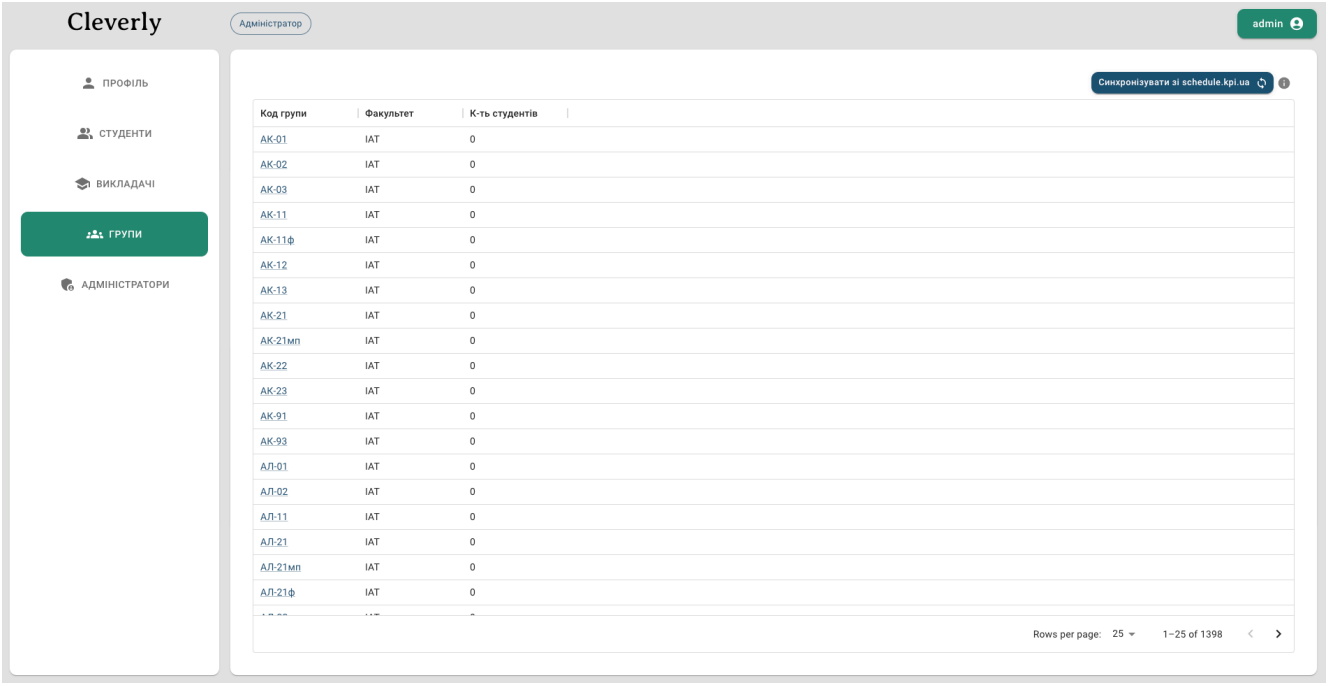


Рисунок 4.12 — Список груп

При переході на одну з груп, відобразиться інформація про неї, в тому числі і список студентів як показано на рисунку 4.13.

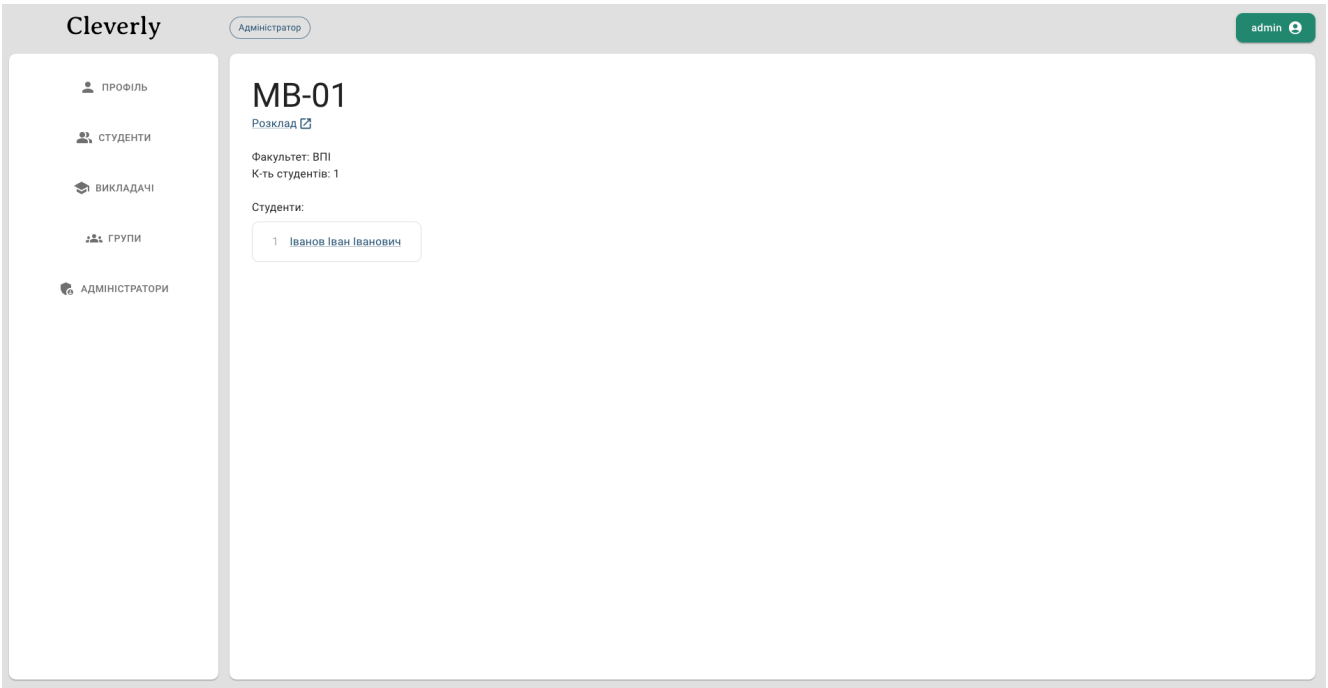


Рисунок 4.13 — Сторінка групи

Також на сторінці групи можна перейти за посиланням на розклад групи, як показано на рисунку 4.14, або перейти на сторінку окремого студента.

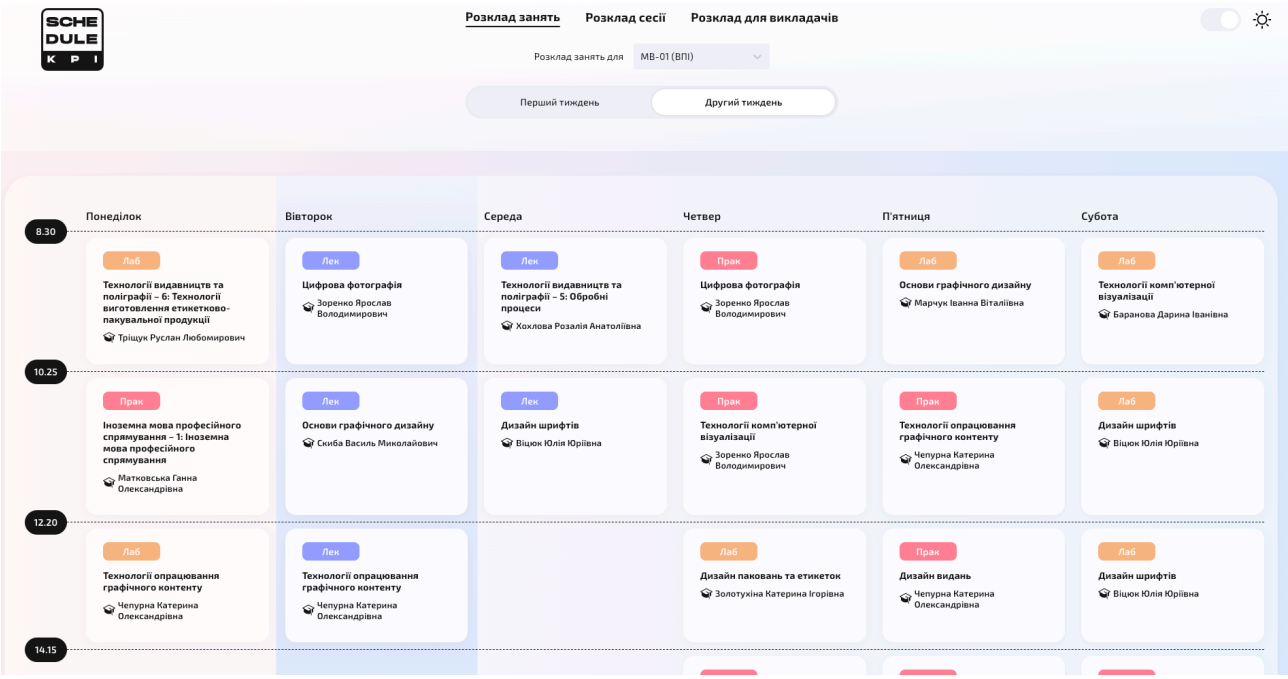


Рисунок 4.14 — Сторінка групи на schedule.kpi.ua

При переході на сторінку сервісу schedule.kpi.ua, група буде вибрана автоматично.

4.3.2 Інтерфейс викладача

При першому вході у систему як викладач або студент, буде запропоновано пройти обов’язкову процедуру завершення реєстрації як показано на рисунку 4.15.

Рисунок 4.15 — Екран завершення реєстрації

Після кліку на кнопку “Розпочати”, почнеться процедура самої реєстрації як показано на рисунку 4.16.

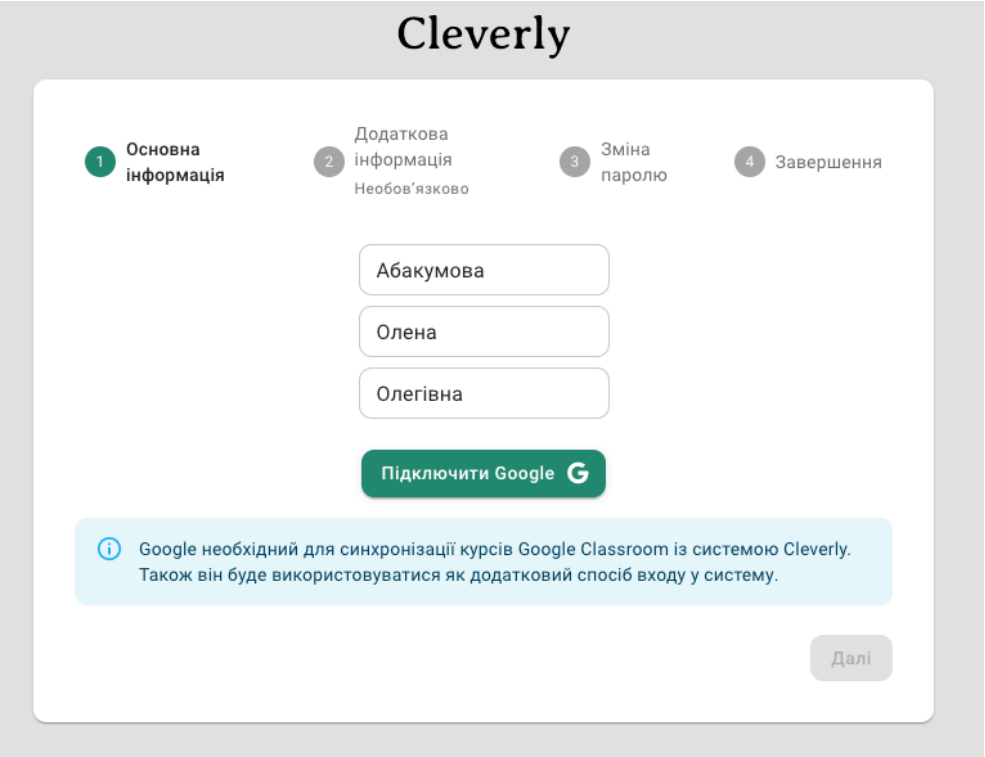


Рисунок 4.16 — Перший крок реєстрації

На цьому екрані можна перевірити правильність автоматично заповнених ПІБ, а також є обов’язковим надання доступу до сервісів Google через електронну пошту як показано на рисунку 4.17.

Тепер можна перейти на наступний крок, де буде запропоновано ввести необов’язкову інформацію про користувача як показано на рисунку 4.18, а саме — посиланя на телеграм, номер телефону та додаткову інформацію. У будь який момент часу є можливість повернутися на попередній крок, щоб дозаповнити дані, клікнувши на кнопку “Назад” у лівому нижньому кутку або перейти далі, пропустивши цей крок, натиснувши кнопку “Далі”.

На останньому кроці необхідно обов’язково змінити пароль на новий і тільки після цього можна переходити до завершення як показано на рисунку 4.19.

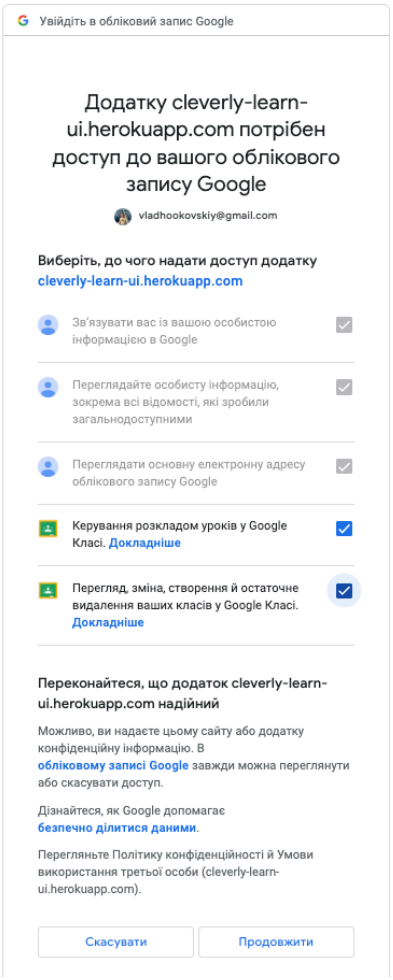


Рисунок 4.17 — Надання прав доступу до сервісів Google

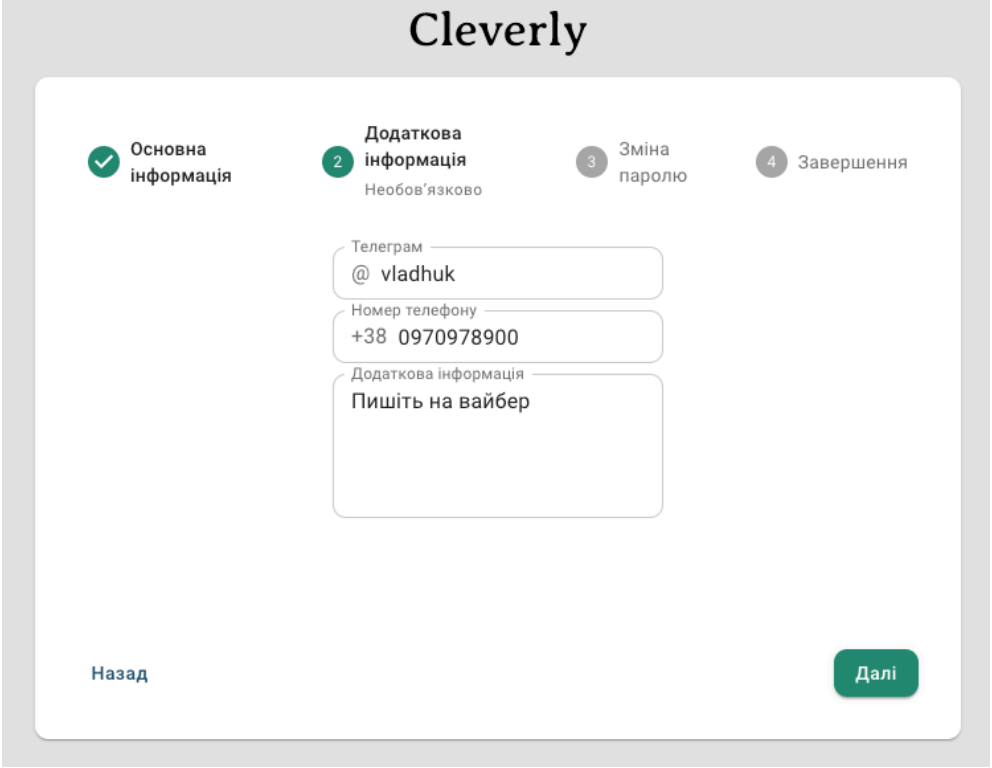


Рисунок 4.18 — Заповнення необов'язкових даних

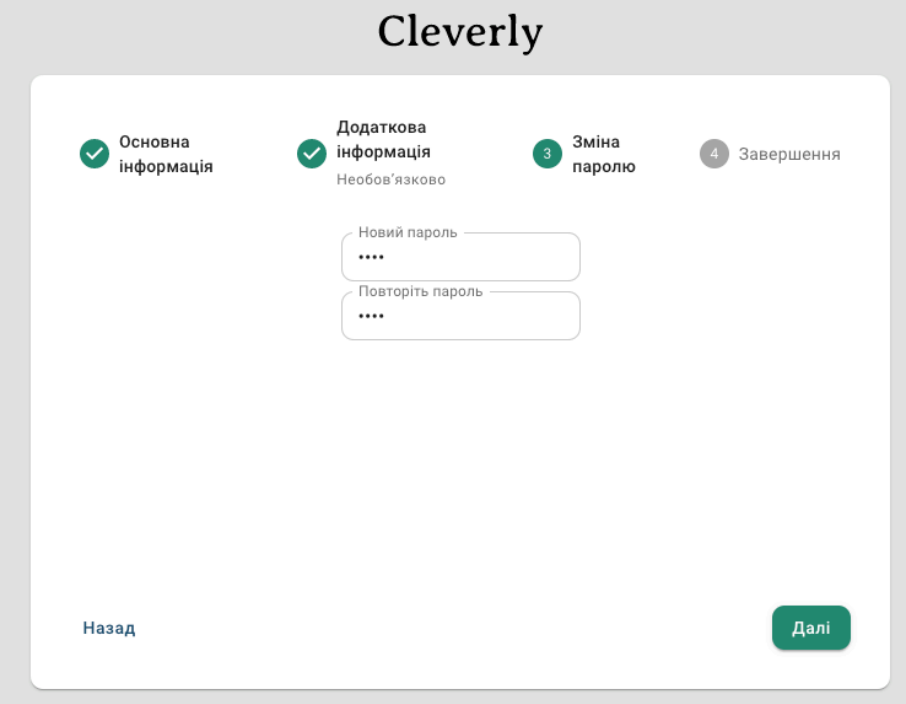


Рисунок 4.19 — Введення нового паролю

Після збереження введених даних, система оповістить користувача про завершення і запропонує перейти на головну сторінку як показано на рисунку 4.20.

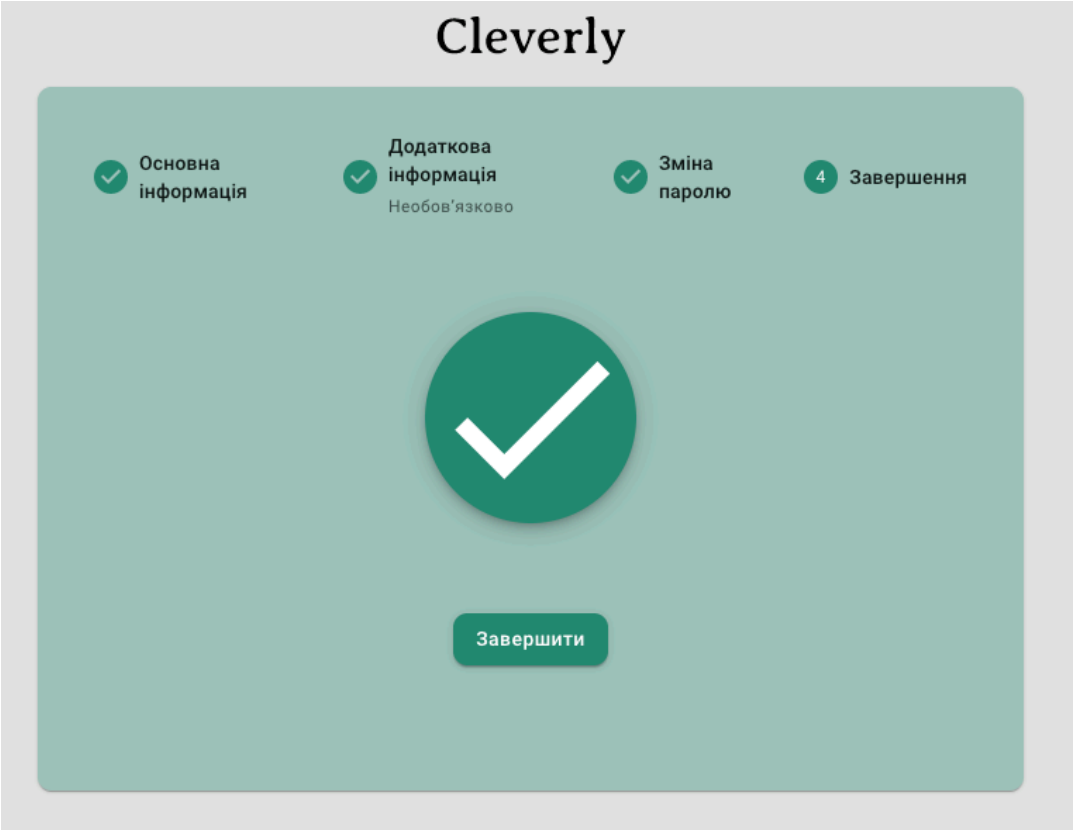


Рисунок 4.20 — Завершення реєстрації

На головній сторінці можна побачити профіль користувача із вже заповненими даними як показано на рисунку 4.21. При кліку на електронну пошту, у користувача відкриється його поштовий клієнт за замовчуванням, при кліку на посилання в телеграм, користувача буде направлено у відповідний профіль у телеграм, а при кліку по номеру телефону буде відкрито додаток для дзвінків за замовчуванням.

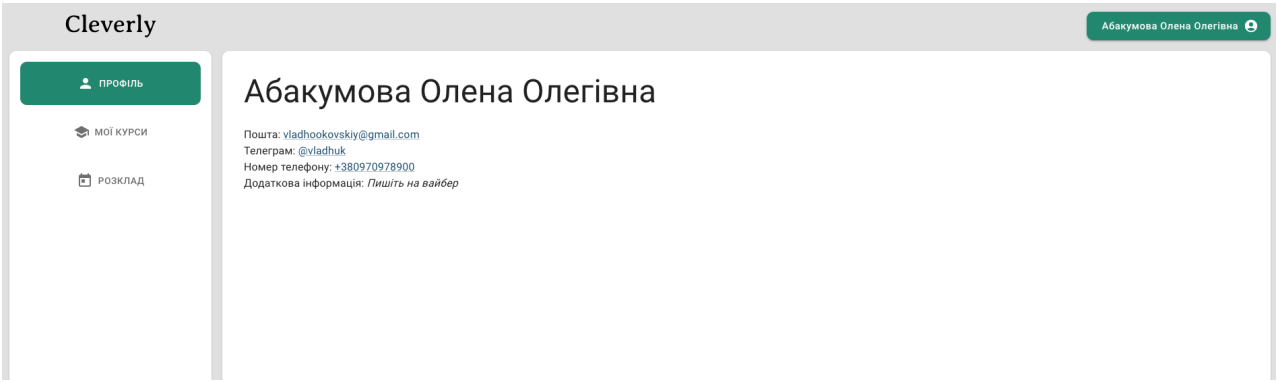


Рисунок 4.21 — Профіль викладача

Також через бічну панель можна перейти на свій розклад викладача на сервісі schedule.kpi.ua.

При переході на сторінку курсів, користувач побачить пусту сторінку, де йому буде запропоновано створити новий курс як показано на рисунку 4.22.

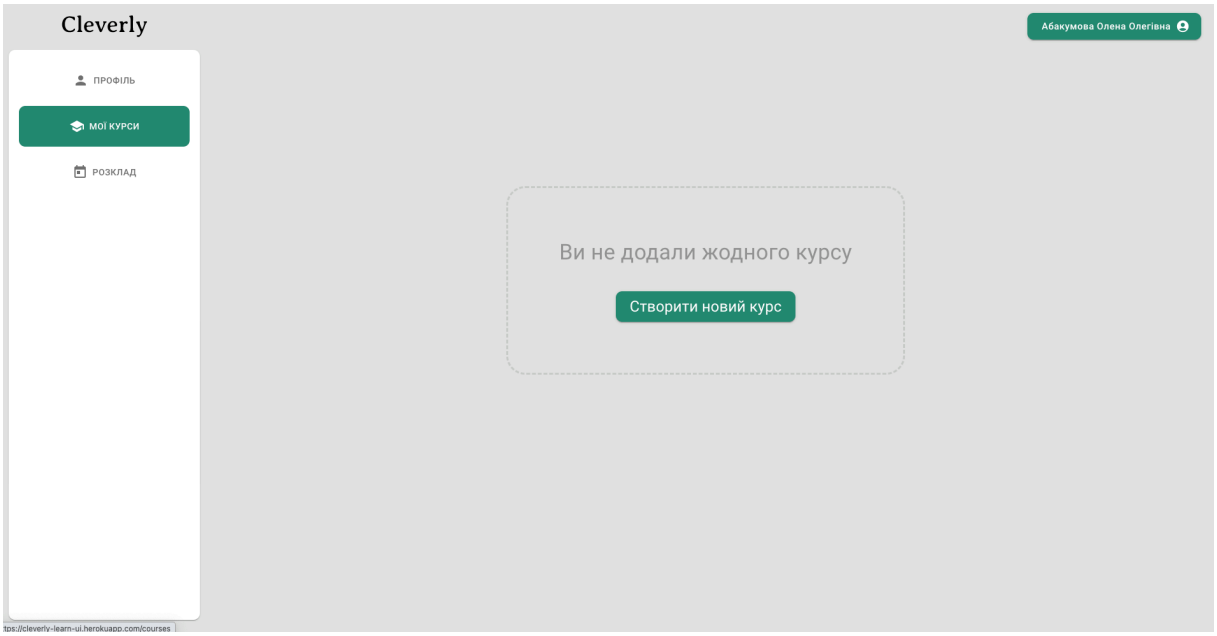


Рисунок 4.22 — Сторінка курсів

При створенні курсу можна увести його назву, а також додати до курсу кілька груп вибраного факультету як показано на рисунку 4.23. При увімкненні перемикача “Створити Google Classroom”, курс буде автоматично створено на Google Classroom, а також туди буде автоматично запрошено усіх студентів вибраних груп.

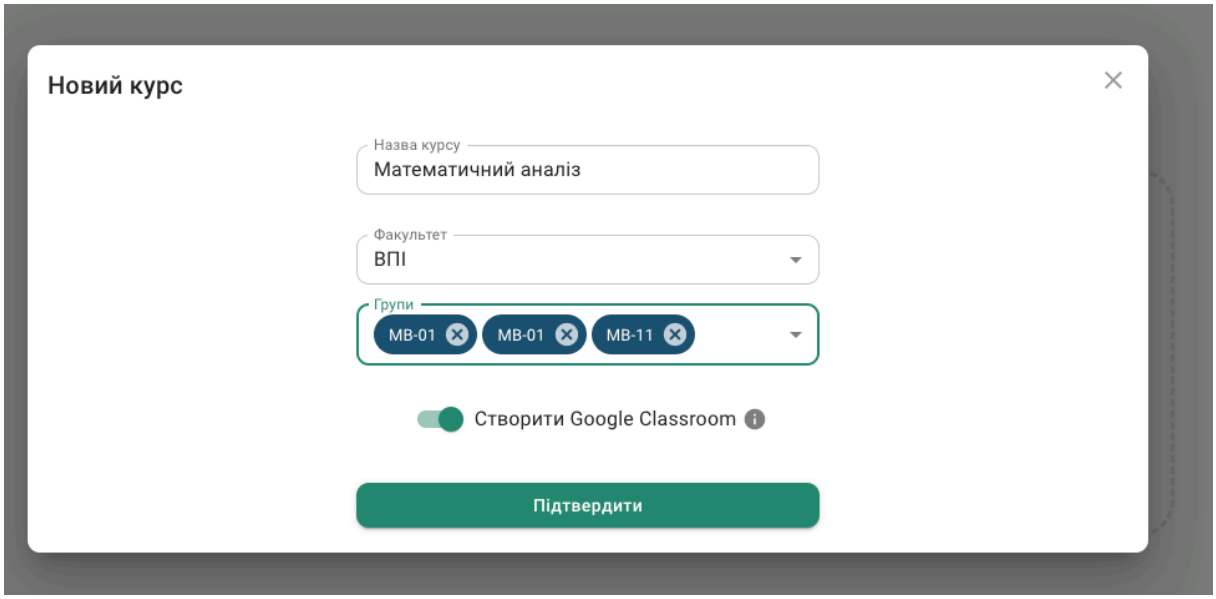


Рисунок 4.23 — Створення нового курсу

Після створення курсу, система попрохає користувача підтвердити створення курсу за посиланням на Classroom як показано на рисунку 4.24.

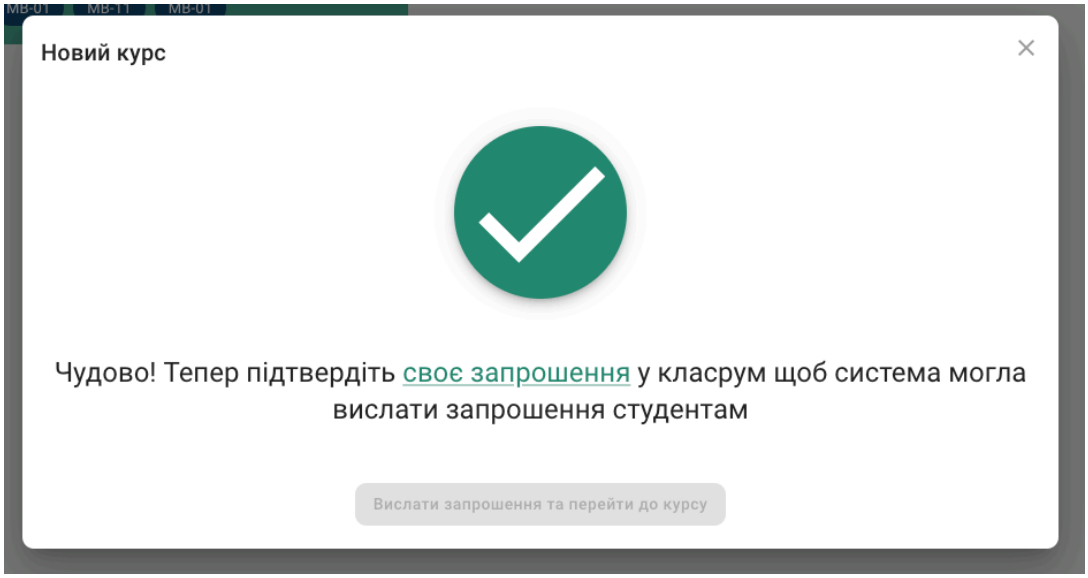


Рисунок 4.24 — Екран завершення створення курсу

Підтвердити запрошення і розіслати запрошення студентам можна буде пізніше на сторінці курсу.

Після завершення, викладача буде переправлено на головну сторінку курсу у системі, де можна побачити список груп із студентами та перейти до них, як показано на рисунку 4.25. Також з цієї сторінки можна перейти до сторінки на Google Classroom, видалити курс або надіслати студентам запрошення повторно через меню у верхньому правому кутку. При видаленні курсу, він буде також видалений і з Classroom.

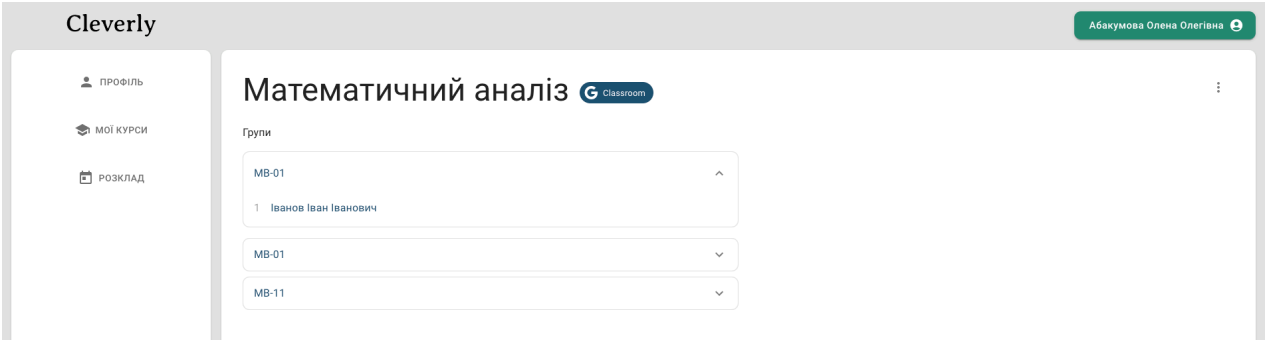


Рисунок 4.25 — Сторінка курсу

При поверненні на сторінку списку курсів, буде відображено усі курси у формі карток як показано на рисунку 4.26. Для цього створимо ще декілька тестових курсів без підключення до Google Classroom.

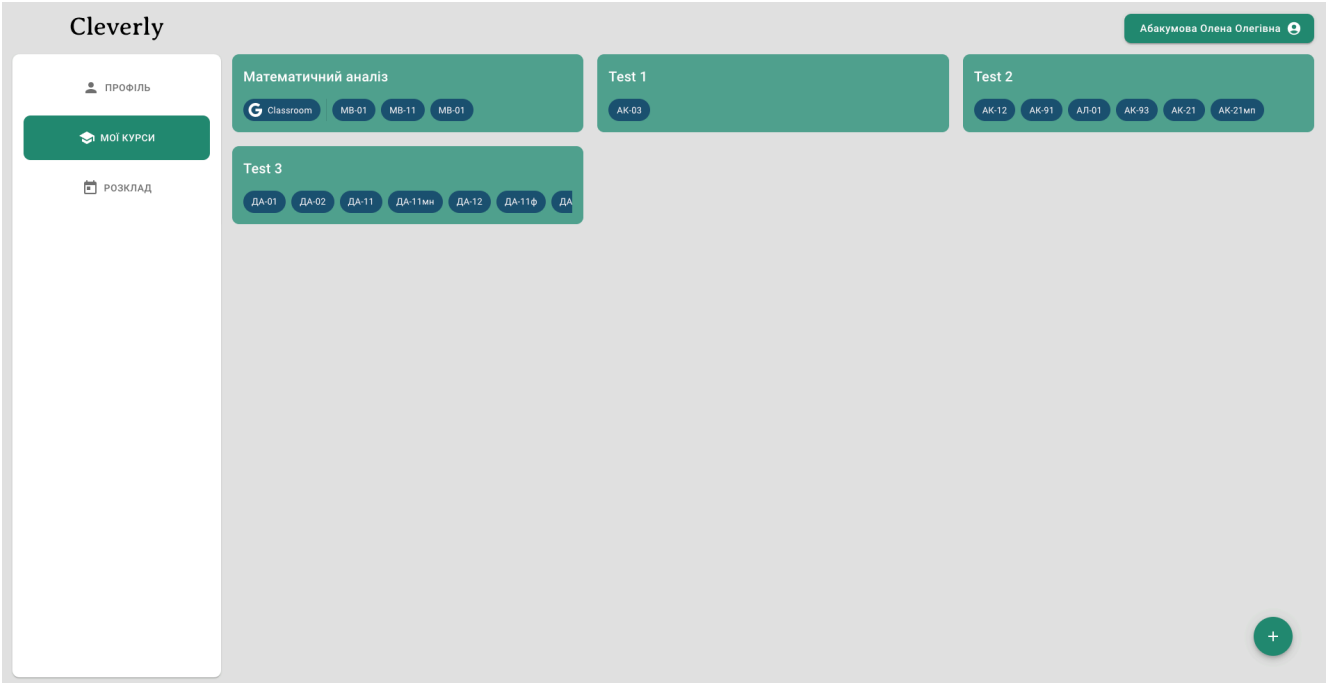


Рисунок 4.26 — Сторінка курсів викладача

На картці в скороченому вигляді відображена найголовніша інформація прокурс, а саме — назва курсу, посилання на Classroom (якщо підключено) і список груп. Якщо список груп завеликий, його можна прогортати горизонтальним скролом.

4.3.3 Інтерфейс студента

Інтерфейс студента дуже схожий на інтерфейс викладача, а також містить багато схожих типів сторінок із попередніх інтерфейсів.

Після реєстрації студента буде також показано головний екран профілю з мінорними змінами спеціально для студента (додано посилання на факультет і групу) як показано на рисунку 4.27.

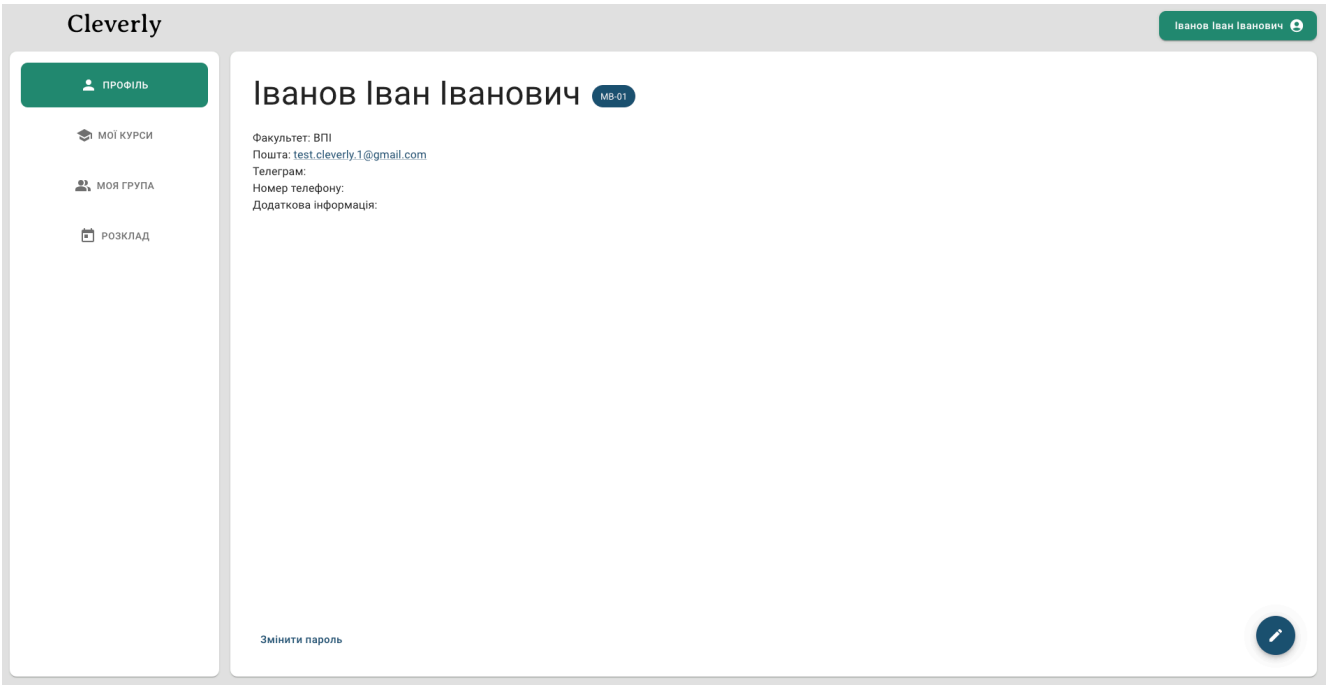


Рисунок 4.27 — Профіль студента

При переході на вкладку групи або розкладу, буде показана група студента або розклад на schedule.kpi.ua відповідно. Як показано на рисунку 4.28, при переході на сторінку курсів (рис. 4.28), буде показано дещо видозмінений інтерфейс курсів, відносно викладача — на картці курсу студента додано посилання на профіль викладача.

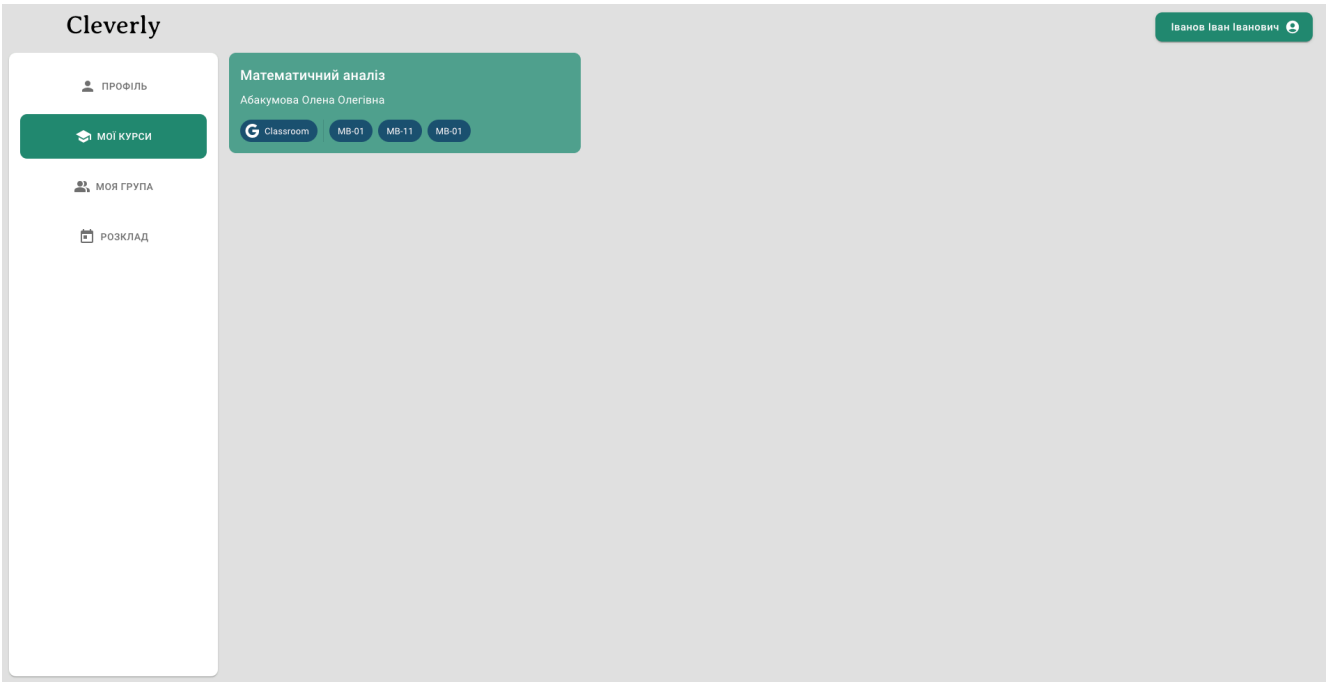


Рисунок 4.28 — Сторінка курсів студента

На картці курсу можна побачити назву курсу, ПІБ викладача, посилання на Classroom, а також список груп цього курсу.

Висновки до розділу 4

У цьому розділі було розглянуто варіанти інсталяцію програми у середовище та необхідні налаштування інструментів для цього. Було показано характеристики системи, використані під час розробки. Крім того, було дуже детально розглянуто роботу користувачів для всіх трьох ролей із інтерфейсом користувача.

5 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

Розділ має на меті проведення маркетингового аналізу стартап проек- ту задля визначення принципової можливості його ринкового впрова- дження та можливих напрямів реалізації цього впровадження.

5.1 Опис ідеї проекту

В межах підрозділу буде послідовно проаналізовано та подано у вигляді таблиць:

- зміст ідеї (що пропонується);
- можливі напрямки застосування;
- основні вигоди, що може отримати користувач товару (за кожним напрямком застосування);
- чим відрізняється від існуючих аналогів та замінників.

На таблиці 5.1 можна побачити опис ідеї стартапу. Виходячі з неї можна постановити, що проект буде застосовуватися переважно у навчальній сфері.

Таблиця 5.1 — Опис ідеї стартап-проекту

Зміст ідеї	*Напрямки застосування*	*Вигоди для користувача*
Синхронізація даних зі schedule.kpi.ua і автоматизація створення курсів на Google Classroom	Навчальні заклади та інші системи, де користувачі працюють із Google Classroom	Автоматизація великої кількості ручної роботи, такої як додання нових викладачів або створення курсів з готовими списками користувачів

Система вирізняється на фоні конкурентів за рахунок підтримки і роботи з Google Classroom за допомогою Google API, а також має змогу синхронізувати та актуалізувати дані із schedule.kpi.ua, через що усі групи та викладачів можна передзавантажити у систему в один “клік” на початку роботи з нею.

Таблиця 5.2 — Визначення сильних, слабких та нейтральних характеристик ідеї проекту

п/п	характеристики ідеї	Мій проект	КПІ Кампус	Google Classroom	W	N	S
1	Створення курсів по списках груп	+	—	—			+
2	Виставлення оцінок	—	+	+	+		
3	Синхронізація із schedule.kpi.ua	+	+ —	—			+
4	Додання нових користувачів	+	+	+		+	
5	Підключення до Classroom	+	—	+			+
6	Перегляд розкладу	+	—	—			+
7	Перегляд інформації про користувача	+	—	—			+
8	Перегляд користувачів курсу	+	—	+			+
9	Реєстрація по пошті	+	—	+		+	
10	Зручний інтерфейс	+	—	+		+	
11	Oauth2 вхід	—	—	+		+	
12	Безкоштовне користування	+	+	+ —		+	
13	Додання д/з	—	—	+	+		

Виходячи з наведеної таблиці 5.2, можна зробити висновок, що проект має багато переваг над конкурентами, але, не дивлячись на це, є функції, які можна було б додати або вдосконалити, щоб уникнути переваги конкурентів в цих сферах. Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності.

5.2 Технологічний аудит проекту

В межах даного підрозділу буде проведено аудит технології, за допомогою якої можна реалізувати ідею стартапу (технології створення товару). У таблиці 5.3 ми розглянемо технологічну здійсненість ідеї проекту.

Таблиця 5.3 — Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології реалізації	Наявність технологій	Доступність технологій
1	Підключення Google API	Google SDK	+	Доступна безкоштовно
2	Синхронізація із schedule.kpi.ua	JS	+	Доступна безкоштовно
3	Надсилання запрошень на пошту	SMPT сервер	+	Доступна безкоштовно
4	Веб інтерфейс	JS	+	Доступна безкоштовно
5	Авторизація та аутентифікація	JWT	+	Доступна безкоштовно

Підсумувавши усі ідеї проекту та технології їх реалізації, можна зробити висновок, що проект реалізувати можливо. Усі технології знаходяться у відкритому доступі та мають документацію з описанням їх інтеграції та реалізації.

5.3 Аналіз ринкових можливостей запуску проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Спочатку проведемо аналіз попиту на потенційного ринку стартап-проекту по таблиці 5.4.

Таблиця 5.4 — Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	~3
2	Загальний обсяг продаж, грн/ум.од	~$1000

3	Динаміка ринку (якісна оцінка)	Стагнує
4	Наявність обмежень для входу (вказати характер обмежень)	Більшість систем є локальними для кожного навчального закладу
5	Специфічні вимоги до стандартизації та сертифікації	У кожного навчального закладу свої ресурси із розкладом, викладачами, розкладом
6	Середня норма рентабельності в галузі (або по ринку), %	Невідомо

З наведеної таблиці можна зробити висновок, що продукт буде привабливим на ринку, оскільки прямих конкурентів не так і багато, а більшість систем, як і розроблюваний стартап, є локальними для конкретних навчальних закладів та орієнтуються на специфіку кожної окремої інфраструктури.

Тепер на таблиці 6 розглянемо потенційні групи клієнтів стартап проекту.

Таблиця 5.5 — Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Дистанційне навчання	Студенти та викладачі	Студентам необхідний зручний інтерфейс, а викладачам багатство функціоналу	Зручність та функціональність
2	Синхронізація різних сервісів в одному	Студенти та викладачі	Усі групи потребують надійність перш за все	Надійність
3	Підключення Google Classroom	Викладачі	Групи потребують автоматизацію процесу	Простота

Визначивши потенційні групи клієнтів, можемо провести аналіз ринкового середовища. Для початку складемо таблицю 5.6 факторів загроз та таблицю 5.7 можливостей, що сприяють ринковому впровадженню проекту.

Таблиця 5.6 — Фактори загроз

1	Слабке фінансування	Навчальні заклади не звикли оплачувати софт	Пошук інших інвесторів
2	Відмова у співпраці	Навчальним закладам може бути не цікавий наданий продукт	Запустити рекламну кампанію та розширити коло споживачів доданням нового функціоналу
3	Обмеженість функцій	Інструмент обмежений наявними функціями і не має деяких функцій, які мають конкуренти	Додавання нових функцій за потреби
4	Часті помилки в системи	Нова система потенційно може мати багато помилок на початку	Додання високоякісного логування у систему, швидке реагування на проблеми

Таблиця 5.7 — Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Відсутність повноцінних альтернатив	Існуючі альтернативи не надають повний набір можливостей	Розширення набору функцій
2	Популярність систем дистанційного навчання	Індустрія постійно набирає обертів	Вихід на глобальний ринок
3	Розширення клієнтської бази	Додаток може зацікавити багато навчальних закладів	Додання функцій, загальних для всіх навчальних закладів, проведення рекламної кампанії

Тепер на таблиці 5.8 проведемо аналіз пропозиції і визначимо загальні риси конкуренції на ринку.

Таблиця 5.8 — Ступеневий аналіз конкуренції на ринку

Вказати тип конкуренції — чиста	Велика кількість конкурентів на ринку	Додання нового функціоналу та проведення рекламної кампанії
За рівнем конкурентної боротьби — міжнародний	Гравці на ринку із різних країн світу	Локалізація продукту
За галузевою ознакою — внутрішньогалузева	Продукт використовується лише у навчальній сфері	Інтеграція потреб з інших галузей
Конкуренція за видами товарів: товарно-видова	Конкуренція між функціоналом програмних продуктів	Винайдення новітніх функцій
За характером конкурентних переваг — нецінова	Ціна не є основним фактором при виборі продукта споживачем	Оптимізація продукту, що зробить його менш затратним у використанні
За інтенсивністю — марочна	Постачальники виступають під певним брендом	Розвинення бренду, що робити продукт більш впізнавальним на ринку

Тепер проведемо більш детальний аналіз конкуренції на таблиці 5.9.

Таблиця 5.9 — Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	КПІ Кампус, Google Classroom, Moodle	Відсутні	Google	Навчальні заклади	Достатня кількість
Висновки	Інтенсивна конкуренція	Потенційні конкуренти не спостерігаються	Ціни від постачальників стабільні, різке підвіщення не прогнозується	Через високу конкуренцію, клієнти можуть висувати свох вимоги стосовно функціоналу	Локальні товари замінники відстутні

Виходячи з таблиці, можна зробити висновок, що продукт зможе комфортно себе почувати на ринку серед конкурентів. Розроблюваний стартап проект буде виділятися за рахунок новітніх функцій, які не були реалізовані у конкурентів до цього часу. Тим не менш важливо підтримувати актуальність нових функцій, тому що будь-який функціонал через який час може бути імплементований у проектах конкурентів.

На основі аналізу конкуренції, а також із урахуванням характеристик ідеї проекту, вимог споживачів до товару та факторів маркетингового середовища визначимо перелік факторів конкурентоспроможності та проведемо їх аналіз на таблиці 5.10.

Таблиця 5.10 — Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування
1	Економія часу при роботі	За рахунок автоматизованих функцій системи, користувачі будуть віддавати перевагу нашому продукту через менші витрати часу на заповнення бази даних
2	Зручний інтерфейс	Досить невелика кількість конкурентів має зручний користувацький інтерфейс. переважна кількість продуктів має важкий для сприйняття та старий дизайн.
3	Підключення сторонніх сервісів	За рахунок щільної інтеграції Google API, додаток матиме змогу підключати, крім вже інтегрованого Google Classroom, ще й інші сервіси від Google (наприклад пошту).

Визначивши фактори конкурентоспроможності, проведемо аналіз сильних та слабких сторін стартап-проекту на таблиці 5.11.

Таблиця 5.11 — Порівняльний аналіз сильних та слабких сторін «Управління навчанням студентів за допомогою Google API»

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів						
			-3	-2	-1	0	+1	+2	+3

Таблиця 5.11 — Порівняльний аналіз сильних та слабких сторін «Управління навчанням студентів за допомогою Google API»

1	Економія часу при роботі	14				+			
2	Зручний інтерфейс	20		+					
3	Підключення сторонніх сервісів	18		+					

Тепер проведемо SWOT-аналіз (матриця аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін) на таблиці 5.12.

Таблиця 5.12 — SWOT-аналіз стартап-проекту

Сильні сторони: інтеграція із Google Classroom, синхронізація із schedule.kpi.ua	Слабкі сторони: невелика кількість функціоналу на старті
Можливості: інтеграція інших сервісів від Google	Загрози: витіснення з ринку іншими гігантами індустрії, незацікавленість майбутніх споживачів у продукті

Взявши до уваги усі метрики із SWOT-аналізу, розробимо альтернативи ринкової поведінки для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок, як показано на таблиці 5.13.

Таблиця 5.13 — Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Пробний період	60%	6 міс.
2	Залучення студентів до розробки	80%	1 рік

Найбільш доцільною альтернативою впровадження є залучення студентів навчального закладу до розробки, так як більшість функціоналу додатку їм вже відома. Це дозволить скоротити витрати на розробку.

5.4 Розроблення ринкової стратегії проекту

При розробці ринкової стратегії проекту, перш за все необхідно визначити стратегію охоплення ринку, а саме описати цільові групи потенційних споживачів, як показано на таблиці 5.14.

Таблиця 5.14 — Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Навчальні заклади	Готові, так як в більшості випадків навчальні заклади не готові до розробки власної системи управління	Високий	Середня, так як навчальні заклади не готові вкладати великі кошти у подібні системи	Просто, навчальний заклад буде шукати найдешевше програмне рішення
2	Навчальні курси	Частково готові, так як більшість навчальних курсів вже мають власну систему організації студентів	Середній	Висока, так як навчальні курси біль платоспроможні	Складно, необхідна хороша рекламна кампанія

Провівши аналіз, можна визначити, що при виході продукту на ринок, найбільш доцільно буде орієнтуватися на ринок навчальних закладів, так як там нема високої конкуренції на подібні проекти, хоча і платоспроможність таких споживачів є невисокою.

Для роботи в обраному сегменті ринку необхідно сформувати базову стратегію розвитку як показано на таблиці 5.15.

Таблиця 5.15 — Визначення базової стратегії розвитку

1	Залучення інвестицій	Стратегія концентрованого маркетингу	Додання унікальних можливостей із залученням сторонніх сервісів	Стратегія спеціалізації

Наступним кроком є вибір стратегії конкурентної поведінки як показано на таблиці 5.16.

Таблиця 5.16 — Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Ні	Перш за все забирати існуючих у конкурентів, але по можливості і залучення нових споживачів	Так. Необхідно розробити модуль виставлення оцінок на основі готових рішень конкурентів	Стратегія заняття конкурентної ніші

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту, а також в залежності від обраної базової стратегії розвитку та стратегії конкурентної поведінки розробимо стратегію позиціонування, як показано на таблиці 5.17, що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 5.17 — Визначення стратегії позиціонування

1	Надійність	Стратегія спеціалізації	Надійність за рахунок використання сторонніх сервісів	Відчуття цілісності системи
2	Простота	Стратегія спеціалізації	Простота інтерфейсу за рахунок продуманого дизайну	Відчуття сучасності системи

Розробивши ринкову стратегію проекту, можна зробити висновок, що проект буде орієнтований на навчальні заклади, де головною перевагою серед конкурентів буде використання сторонніх сервісів для роботи, що має забрати споживачів у конкурентів. Проект буде позиціонуватися як проста, сучасна та надійна система управління навчання.

5.5 Розроблення маркетингової програми проекту

Першим кроком у розробленні маркетингової програми є формування маркетингової концепції товару, який отримає споживач. Для цього у таблиці 5.18 підсумуємо результати попереднього аналізу конкурентоспроможності товару.

Таблиця 5.18 — Визначення стратегії позиціонування

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	Автоматизація і синхронізація з існуючими системами	Синхронізація із schedule.kpi.ua	Автоімпорт існуючих даних у систему
2	Робота з Google Classroom через додаток	Інтеграція Google API	Інтеграція із сервісами Google

Тепер розробимо трирівневу маркетингову модель товару. Уточнимо ідею продукту та послуги, його фізичні складові, особливості процесу його надання як показано на таблиці 5.19.

Таблиця 5.19 — Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Система контролю навчання студентів		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Вартість 2. Собівартість 3. Відповідність сучасним технологіям 4. Дизайн 5. Надійність	М М Нм М Нм	Вр Тх Тх Ор Тл
	Якість: тестування за допомогою unit testing, manual testing		
	Пакування: розгортка збірки у хмарі		
	Марка: Cleverly		
III. Товар із підкріпленням	До продажу: безкоштовний пробний період		
	Після продажу: консультації та вирішення проблем		
За рахунок чого потенційний товар буде захищено від копіювання: комерційна таємниця, ліцензія, захист інтелектуальної власності			

Після опису рівнів моделі товару, визначимо цінові межі, якими необхідно керуватись при встановленні ціни на потенційний товар, як показано на таблиці 20.

Таблиця 5.20 — Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/ послугу
1	$1000/рік	$500/рік	$50000/міс	$500-2000/рік

Тепер визначимо оптимальну систему збуту на таблиці 5.21, в межах якої приймається рішення.

Таблиця 5.21 — Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Надання пробного періоду і платної підписки	Технічна підтримка та контакт із споживачами	Одноріневий	Проводити збут через посередників і маркетплейси

Спираючись на попередньо обрану основу для позиціонування, розробимо концепцю маркетингових комунікацій як показано на таблиці 5.22.

Таблиця 5.22 — Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Більшість закупок буде здійснюватися влітку перед початком вступної кампанії	Електронна пошта, телефонний зв’язок	Комплексний підхід	Зацікавити клієнта в унікальних функціях продукту	Демонстрація роботи унікальних функція

В результаті проведених аналізів ми отримали маркетингову програму, що включає в себе концепції товару, збуту, просування та попередній аналіз можливостей ціноутворення, спирається на цінності та потреби потенційних клієнтів, конкурентні переваги ідеї, стан та динаміку ринкового середовища, в

межах якого буде впроваджено проект, та відповідну обрану альтернативу ринкової поведінки.

Висновки до розділу 5

В результаті проведених аналізів у даному розділі, можна визначити, що в результаті розробки є можливість ринкової комерціалізації проекту. На продукт буде попит на ринку за рахунок унікального функціоналу, якого немає у конкурентів. Провівши аналіз конкуренції можна зробити висновок, що проект має перспективи, а також має усі шанси стати великим гравцем на ринку, якщо в майбутньому команда розробки продовжить впроваджувати новий функціонал. Головною ціллю такої стратегії буде витіснення конкурентів та впровадження продукту новим споживачам. Подальша імплементація проекту є доцільною, так як він має широку варіативність на додання функціоналу. Проект передбачає тісну інтеграцію із сервісами Google, що може стати головною перевагою продукту, коли перед споживачами постане вибір системи управління навчанням.

ВИСНОВОК

В результаті виконання роботи, було досягнути хороших результатів та отримано ряд корисних навичок. Було створено продукт, який готовий до використання реальними користувачами. Було глибоко досліджено використання Google API для інтеграції в додаток. Це дозволило побудувати застосунок, за допомогою якого користувачі мають змогу взаємодіяти із Classroom безпосередньо із внутрішньої системи навчального закладу. Система в свою чергу вийшла автоматизованою за рахунок інтеграції зі schedule.kpi.ua, що дозволяє вже на початку роботи із додатком мати тисячі синхронізованих записів. Це дозволить зекономити величезний пласт часу працівникам адміністрації та почати роботу із системою без підготовчого етапу.

В процесі розробки було значно покращено навички роботи з графічним редактором Figma, так як було витрачено багато часу на розробку макетів і дослідження кольорової палітри для дизайну. Були отримані навички використання Google API та його інтеграції у сторонні додатки, що значно полегшить розробку нових додатків побудованих на основі інструментів Google. Було також вдосконалено навички роботи з популярними бібліотеками та фреймворками, такими як React, Nest.js та графічною бібліотекою Material Design, а також покращені навички побудови архітектури великих додатків, які використовують реляційні бази даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hypertext Transfer Protocol — HTTP/1.1 [Електронний ресурс] / [R. Fielding, U. Irvine, J. Gettys та ін.]. – 1999. – Режим доступу до ресурсу: https://datatracker.ietf.org/doc/html/rfc2616.
2. Masse M. REST API Design Rulebook / Mark Masse., 2011. – 116 с. – (O'Reilly Media, Inc.).
3. iCode Academy. Json for Beginners: Your Guide to Easily Learn Json In 7 Days / iCode Academy., 2017. – 129 с.
4. JSON Web Token (JWT) [Електронний ресурс] / [M. Jones, Microsoft, J. Bradley та ін.]. – 2015. – Режим доступу до ресурсу: https://datatracker.ietf.org/doc/html/rfc7519.
5. Vanderkam D. Effective TypeScript: 62 Specific Ways to Improve Your TypeScript / Dan Vanderkam., 2019. – 266 с. – (O'Reilly Media).
6. Flanagan D. JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language / David Flanagan., 2020. – 706 с.
7. Hunter II T. Distributed Systems with Node.js: Building Enterprise-Ready Backend Services / Thomas Hunter II., 2020. – 377 с.
8. Bell J. Nest.js: A Progressive Node.js Framework / Jay Bell., 2018.
9. Poulton N. Docker Deep Dive: Zero to Docker in a single book / Nigel Poulton., 2016. – 251 с.
10. Hahn E. Express in Action: Writing, building, and testing Node.js applications / Evan Hahn., 2016. – 256 с.
11. Herron D. Quick Start to using Typescript and TypeORM in Node.js web applications / David Herron., 2019. – 171 с.
12. Regina O. Obe. PostgreSQL: Up and Running: A Practical Guide to the Advanced Open Source Database / Regina O. Obe., 2017. – 314 с.
13. Choi D. Full-Stack React, TypeScript, and Node: Build cloud-ready web applications using React 17 with Hooks and GraphQL / David Choi., 2020. – 648 с.

14. Frain B. Responsive Web Design with HTML5 and CSS: Develop future-proof responsive websites using the latest HTML5 and CSS techniques, 3rd Edition / Ben Frain., 2020. – 408 с.

ДОДАТОК А

Управління навчанням студентів за допомогою Google API

Апробація

IX Міжнародна науково-практична конференція “MODERN RESEARCH IN WORLD SCIENCE”

Аркушів 6


SCI-CONF.COM.UA
MODERN RESEARCH
IN WORLD SCIENCE
PROCEEDINGS OF IX INTERNATIONAL
SCIENTIFIC AND PRACTICAL CONFERENCE
NOVEMBER 28-30, 2022
LVIV
2022

Київ — 2022

КОНТРОЛЬ ДОСТУПУ ДО СИСТЕМ НА ОСНОВІ РОЛЕЙ

Гуковський Владислав Геннадійович
магістрант
Онисько Андрій Ілліч
к.в.н., доцент
Київський політехнічний інститут
імені Ігоря Сікорського
м. Київ, Україна
vladhookovskiy@gmail.com

Контроль доступу на основі ролей (“Role-Based Access Control” або RBAC) відноситься до ідеї призначення дозволів користувачам на основі їх ролі в організації. Він пропонує простий підхід до керування доступом, який менш схильний до помилок, ніж призначення дозволів користувачам окремо.

Використовуючи RBAC для керування ролями, ви аналізуєте потреби своїх користувачів і групуєте їх у ролі на основі спільних обов’язків. Потім ви призначаєте одну або кілька ролей кожному користувачеві та один або кілька дозволів для кожної ролі. Зв’язки «користувач-роль» і «роль-дозволи» спрощують виконання призначень користувачам, оскільки користувачам більше не потрібно керувати індивідуально, натомість вони мають привілеї, які відповідають дозволам, призначеним для їхніх ролей.

Наприклад, якщо ви використовуєте RBAC для керування доступом до програми відділу кадрів, ви можете надати менеджерам відділу кадрів роль, яка дозволить їм оновлювати відомості про співробітників, тоді як інші працівники зможуть переглядати лише свої дані.

По суті, роль — це набір дозволів, які ви можете застосувати до користувачів. За допомогою ролей легше додавати, видаляти та налаштовувати дозволи, ніж призначати дозволи користувачам окремо. У міру збільшення масштабів і складності вашої бази користувачів ролі стають особливо корисними. [1]

Переваги RBAC

З RBAC керувати доступом легше, якщо ви суворо дотримуєтеся вимог до ролі. RBAC допоможе вам:

- створити систематичне, повторюване призначення дозволів;
- легко перевіряти привілеї користувачів і виправляти виявлені проблеми;
- швидко додавати та змінювати ролі, а також застосовувати їх у різних API;
- зменшити ймовірність помилок під час призначення дозволів користувача;
- інтегрувати сторонніх користувачів, надавши їм попередньо визначені ролі;
- ефективніше дотримуватись нормативних та законодавчих вимог щодо конфіденційності та конфіденційності.

Альтернативи RBAC

Список контролю доступу (ACL)

Список контролю доступом (ACL) — це таблиця зі списком дозволів, доданих до обчислювальних ресурсів. Вона повідомляє операційній системі, які користувачі можуть отримати доступ до об'єкта та які дії вони можуть виконувати. Для кожного користувача є запис, який пов'язаний з атрибутами безпеки кожного об'єкта. Для більшості бізнес-додатків RBAC перевершує ACL з точки зору безпеки та адміністративних витрат. ACL краще підходить для впровадження безпеки на рівні окремого користувача та для низькорівневих даних, тоді як RBAC краще обслуговує систему безпеки всієї компанії з наглядовим адміністратором.

Контроль доступу на основі атрибутів (ABAC)

ABAC оцінює набір правил і політик для керування правами доступу відповідно до конкретних атрибутів, таких як інформація про середовище, систему, об'єкт або користувача. Він застосовує булеву логіку для надання або заборони доступу користувачам на основі комплексної оцінки атомарних атрибутів або атрибутів із заданими значеннями та зв'язку між ними.

Хоча RBAC покладається на попередньо визначені ролі, ABAC є більш динамічним і використовує контроль доступу на основі зв'язків. Ви можете використовувати RBAC для визначення елементів керування доступом широкими штрихами, тоді як ABAC пропонує більшу деталізацію. Наприклад, система

RBAC надає доступ усім менеджерам, але політика ABAC надає доступ лише менеджерам, які працюють у фінансовому відділі. ABAC виконує більш складний пошук, який потребує більше процесорної потужності та часу, тому ви повинні вдаватися до ABAC лише тоді, коли RBAC недостатньо. [2]

Перелік посилань:

1. https://auth0.com/docs/manage-users/access-control/rbac#overlapping-role-assignments

2. https://www.imperva.com/learn/data-security/role-based-access-control-rbac