МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра інженерії програмного забезпечення

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Олексій Горський
"____"_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
МАГІСТРА

Тема: «Веб-застосунок «Система управління школою»

Виконавець: Альохін Максим Юрійович

Керівник: к.т.н доцент Ходаков Данііл Вікторович

Нормоконтролер: ст.викл Трофимчук Вікторія Миколаївна

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра інженерії програмного забезпечення
Освітній ступінь магістр
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення»
Форма навчання заочна

ЗАТВЕРДЖУЮ
Завідувач кафедри
Олексій Горський
"___" _____ 2023 р

ЗАВДАННЯ
на виконання кваліфікаційної роботи студента
Альохіна Максима Юрійовича

1. Тема кваліфікаційної роботи: «Веб-застосунок «Система управління школою» затверджена наказом ректора від 29.09.2023 р. № 1994/ст.
2. Термін виконання проекту: з 02.10.2022 р. по 31.12.2023 р.
3. Вихідні дані до роботи : розробити веб-застосунок «Система управління школою» із застосуванням фреймворку ASP.NET Core, Entity Framework Core, реляційної бази даних SQL Server, мови програмування C#.
4. Зміст пояснювальної записки:
    1. Огляд існуючих рішень для управління середньою школою.
    2. Технології веб-застосунку.
    3. Проєктування таблиць бази даних.
    4. Модулі веб-застосунку.
5. Перелік обов'язкових слайдів презентації:
    1. Технології веб-застосунку.
    2. Модулі веб-застосунку.
    3. Безпека, валідація даних, захист від XSS атак.
    4. Адаптивний дизайн.
    5. Алгоритм генерації розкладу.
    6. Безперервна інтеграція та розгортання.

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1.	Вибрати тему дипломного проєкту. Поставити задачу. Ознайомитись з предметною областю. Розробити та затвердити графік роботи.	02.10.2023 р. – 08.10.2023 р.	5% Данііл ХОДАКОВ
2.	Розробити технічне завдання. Дослідити предметну область. Опрацювати відповідну літературу. Підготувати та написати 1 розділ.	09.10.2023 р. – 18.10.2023 р.	20% Данііл ХОДАКОВ
3.	Розробити узагальнені алгоритми розв’язування задач. Спроєктувати БД, запити та звіти. Створити основу програмного продукту. Підготувати та написати 2 розділ.	19.10.2023 р. – 29.10.2023 р.	40% Данііл ХОДАКОВ
4.	Спроєктувати програмні інтерфейси, компоненти, структури даних, алгоритми. Підготувати та написати 3 розділ.	30.10.2023 р. – 12.11.2023 р.	60% Данііл ХОДАКОВ
5.	Доопрацювати програмний продукт, додати функціонал. Підготувати та написати 4 розділ. Запрограмувати, протестувати та налагодити застосунок. Закінчити роботу над програмним продуктом, виправити помилки.	13.11.2023 р. – 03.12.2023 р.	80% Данііл ХОДАКОВ
6.	Завершити написання пояснювальної записки. Відредагувати та надрукувати пояснювальну записку, графічний матеріал.	04.12.2023 р. – 10.12.2023 р.	100% Данііл ХОДАКОВ
7.	Надіслати пояснювальну записку для перевірки на плагіат. Пройти нормо-контроль, переплести пояснювальну записку. Отримати відгук керівника. Підготувати презентацію та доповідь. Отримати рецензії.	11.12.2023 р. – 17.12.2023 р.	100% Данііл ХОДАКОВ

8.	Підготувати документи до захисту та здати їх секретарю ДЕК.	18.12.2023 р. – 24.12.2023 р.	100% Данііл ХОДАКОВ
9.	Захистити кваліфікаційну роботу.	25.12.2023 р.	100% Данііл ХОДАКОВ

Дата видачі завдання: 02.10.2023 р.

Керівник дипломної роботи: к.т.н доцент Данііл ХОДАКОВ

Завдання прийняв до виконання: Максим АЛЬОХІН

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Веб-застосунок «Система управління школою»: 107 сторінок, 62 рисунка, 2 таблиці, 34 використаних джерел, 3 додатки.

БІБЛІОТЕКА, ДИСТАНЦІЙНЕ НАВЧАННЯ, ІНВЕНТАРИЗАЦІЯ, КЛАСНИЙ ЖУРНАЛ, МЕНЕДЖМЕНТ, НАВЧАЛЬНИЙ ПЛАН, НАВЧАЛЬНІ ДОСЯГНЕННЯ, РОЗКЛАД УРОКІВ, СЕРЕДНЯ ЗАГАЛЬНООСВІТНЯ ШКОЛА, СИСТЕМА УПРАВЛІННЯ.

Об'єкт дослідження – система управління середньою загальноосвітньою школою.

Мета дипломної роботи – структурувати, автоматизувати та спростити організацію та управління навчальним процесом у середній школі.

Метод дослідження – аналіз робочого процесу та задач, що стоять перед працівниками середньої загальноосвітньої школи, розробка на цій основі модулів програмного засобу. Аналіз існуючих програмних засобів з метою виділення функціоналу, який дозволить краще організувати, оптимізувати та автоматизувати навчальні та адміністративні процеси у школі. Вивчення алгоритмів для автоматичної генерації розкладу та впровадження функціоналу генератора шкільного розкладу у одному з модулів веб-застосунку.

В ході роботи над програмним продуктом створені модулі для керування школою (вчителі, учні, класи, предмети, бібліотека, інвентаризація), впроваджено алгоритм генерації розкладу, розроблена база даних, запроваджена система статистичного аналізу роботи закладу.

Результати роботи та отриманий програмний продукт можна застосовувати для автоматизації управління середнього загальноосвітнього навчального закладу.

Розробка та дослідження проводилися під управлінням ОС Windows 10, у середовищі програмування Visual Studio 2022 на мові програмування C#.

ABSTRACT

Explanatory note for the diploma thesis “Web Application “School Management System”: 107 pages, 62 figures, 2 tables, 34 references, 3 appendices.

LIBRARY, DISTANCE LEARNING, INVENTORY, CLASS JOURNAL, MANAGEMENT, CURRICULUM, ACADEMIC ACHIEVEMENTS, LESSON SCHEDULE, SECONDARY SCHOOL, MANAGEMENT SYSTEM.

Research object – is the management system of a secondary school.

The aim of the diploma thesis – is to structure, automate, and simplify the organization and management of the educational process in a secondary school.

Research method – is analysis of the work process and tasks facing the staff of a secondary school, development of software modules based on it. Analysis of existing software tools to identify functionality that will better organize, optimize, and automate educational and administrative processes in the school. Study of algorithms for automatic schedule generation and implementation of the functionality of a school schedule generator in one of the modules of the web application.

During the work on the software product, modules for school management (teachers, students, classes, subjects, library, inventory) were created, an algorithm for generating the schedule was implemented, a database was developed, and a system for statistical analysis of the institution's work was introduced.

The results of the work and the obtained software product can be applied to automate the management of a secondary general education institution.

The development and research were conducted in Windows 10 OS, using Visual Studio 2022 integrated development environment and C# programming language.

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

API – Application Programming Interface

CI/CD – Continuous Integration / Continuous Delivery (Continuous Deployment)

CLI – Command-line Interface

CRUD – Create, Read, Update, Delete

CSS – Cascading Style Sheets

DB – Database

DDL – Data Definition Language

DML – Data Manipulation Language

EF – Entity Framework

HTML – HyperText Markup Language

HTTP – Hypertext Transfer Protocol

IIS – Internet Information Services

LINQ - Language-Integrated Query

MVC – Model–view–controller

MVP – Model–view–presenter

MVVM – Model–view–viewmodel

O/RM – Object–relational mapping

REST – Representational State Transfer

SQL – Structured Query Language

SSH – The Secure Shell Protocol

TDD – Test-driven development

UML – Unified Modeling Language

WYSIWYG – What You See Is What You Get

XSS – Cross-site scripting

YAML – Yet Another Markup Language, YAML Ain't Markup Language

ВСТУП

Ринок програмного забезпечення для середніх загальноосвітніх закладів порівняно молодий і мало насичений. Бракує дієвих інструментів, які б автоматизували і оптимізували роботу по організації навчально-виховного процесу, дозволили б суб'єктам навчального процесу сконцентруватись на творчих проблемах, звільнили б від рутини.

Об'єктом роботи є системи менеджменту освітніх закладів. Предмет роботи – модульна система з управління процесами у середньому загальноосвітньому закладі.

Метою даної роботи є проаналізувати існуючі програмні рішення з управління школою та створити веб-застосунок «Система управління школою», що буде мати модулі для організації роботи адміністрації, вчителів, бібліотекаря, завгоспа.

Веб-застосунок буде працювати на фреймворку ASP.NET Core і написаний на мові програмування C#. Дані застосунку будуть зберігатись у реляційній базі даних SQL Server. Доступ до даних буде здійснюватися за допомогою об'єктно-реляційного перетворювача Entity Framework Core.

Завданнями в ході виконання роботи будуть створення класів-сутностей, зв'язків між ними, проєктування таблиць бази даних, розробка функціоналу модулів застосунку. Для кожної з сутностей необхідно буде розробити набір сторінок, які виконуватимуть операції зі створення, редагування та видалення.

Особливу увагу буде приділено модулям автентифікації і авторизації для захисту даних учасників навчального процесу. Перевірка введених користувачем даних буде здійснюватися на клієнтській частині застосунку за допомогою jQuery та на сервері засобами ASP.NET Core.

Одним з викликів сучасності є дистанційне навчання. Однойменний модуль буде розроблено для організації навчального процесу онлайн.

Процес складання розкладу є важливим для кожного навчального закладу. Для вирішення цієї задачі буде створено модуль з генерації розкладу, який автоматизує та полегшить роботу заступника директора з навчально-виховної роботи у школі.

Статистичний модуль у наглядній графічній формі відображатиме дані і допоможе у створенні звітів.

Зменшувати кількість паперу та спрощувати роботу бібліотекаря буде модуль «Бібліотека».

Модуль «Інвентаризація» служитиме для постановки майна на облік та для його списання. Він спростить роботу завідуючому з господарської частини школи.

Вчителі будуть вести облік навчальних досягнень здобувачів освіти та відвідування ними уроків у модулі «Класний журнал».

Для батьків будуть створені модулі, де вони зможуть переглядати навчальні досягнення учнів, розклад уроків, завдання з дистанційного навчання.

Розроблений застосунок може бути використаний у середніх загальноосвітніх навчальних закладах України для стандартизаії, автоматизації та більш ефективної їх роботи.

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ДЛЯ УПРАВЛІННЯ СЕРЕДНЬОЮ ШКОЛОЮ

1.1. Система «Єдина школа»

Система «Єдина школа» - це інформаційно-комунікаційна автоматизована система, призначена для закладів освіти, учнів та їх батьків, а також для органів управління освіти [1].

Система «Єдина школа» є веб-застосунком і містить наступні модулі.

а) Журнал вчителя:

1) журнал оцінок;
2) зміст уроків;
3) журнал екскурсій.

б) Індивідуальна картка учня.

в) Тестування.

г) Друк звітності.

д) Довідка.

Система являє собою «електронний журнал». Адміністратор наповнює систему розкладом, педагогічним навантаженням та списками учнів, розподілом на групи тощо. Класні керівники мають змогу відмічати відсутніх. Вчителі ведуть свої сторінки, заповнюють календарний план, виставляють оцінки, результати підсумкового оцінювання. Календарне планування завантажується у вигляді Excel файлу.

У журналі вчителя можна переглядати та редагувати зміст уроків. На «Індивідуальній картці учня» виставляються навчальні досягнення у класах, де ще немає оцінювання по дванадцятибальній шкалі. Це такі оцінки: «має значні успіхи», «демонструє помітний прогрес», «досягає результату за допомогою вчителя», «ще потребує уваги і допомоги».

1.2. Проєкт «КУРС освіта»

Проєкт «КУРС освіта» є платним програмним комплексом, що містить ряд Windows desktop застосунків. До комплексу входять наступні компоненти:

- люди;
- класи;
- розклад;
- журнал;
- навантаження;
- навчальні плани;
- предмети;
- приміщення;
- звіти;
- свідоцтва;
- бібліотека.

Програма призначена для роботи з базою даних школи. Адміністратор закладу заносить у програму інформацію про учнів, адміністрацію школи, персонал, навчальне навантаження, розподіл на групи, предмети, приміщення, розклад уроків тощо. Програма дозволяє вести електронний журнал успішності та відвідування учнів. Модуль бібліотеки слугує для обліку та замовлення навчальних підручників.

У програмі можливе складання розкладу як в автоматичному, так і ручному режимі. Програма «КУРС школа» може працювати у двох режимах: з доступом та без доступу до мережі «Інтернет» [2].

Як додатковий оплачуваний модуль поставляється програма друку свідоцтв про повну загальну середню освіту.

Програма «КУРС школа» є однією з програм пакету «КУРС» (рис. 1.1). До неї входять програми «КУРС сайт» для завантаження інформації на централізований портал, «КУРС школа +» для місцевих органів освіти. Сайт підтримки: www.ekurs.org.

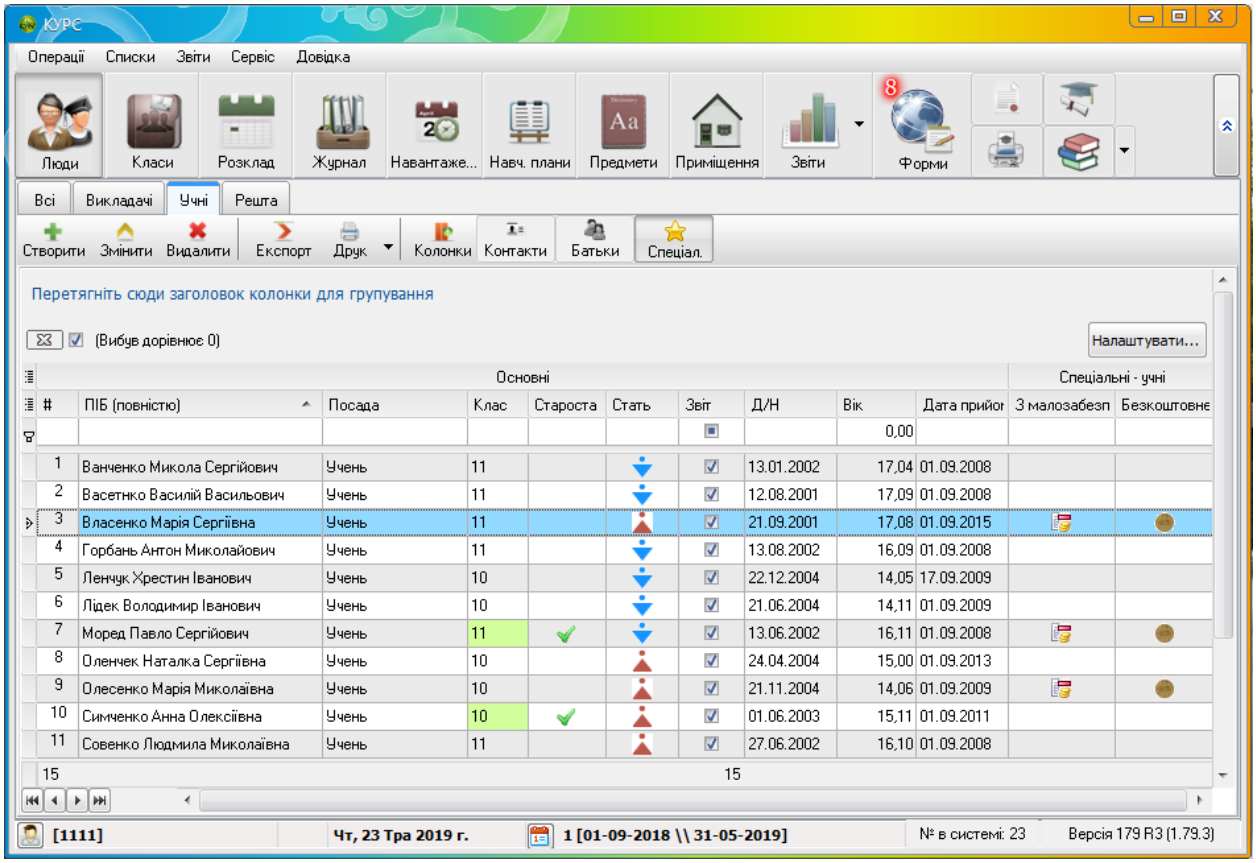

Рис. 1.1. Програма «КУРС школа»

Портал www.isuo.org є другим компонентом комплексу. Це інформаційна система керування освітою. Він отримує, аналізує та консолідує інформацію та звітність з програм «КУРС школа». Портал має механізм пошуку інформації, що спрощує складання користувацьких звітів. Він має надійні алгоритми захисту інформації від несанкціонованого використання [2].

«КУРС школа» інтегрований з освітнім порталом www.isuo.org і дозволяє завантажувати на сайт дані та звіти. За допомогою «КУРС школа» на основі введених даних створюються звітність для відділів освіти: форми 76-РВК, 83-РВК, 77-РВК, Д-4, Д-5, Д-6, Д-7-8, Д-9. Форми направляються в електронному вигляді на портал www.isuo.org та роздруковуються у паперовому вигляді.

Портал www.nz.ua – третій компонент комплексу (рис. 1.2). Дозволяє вчителям вести електронні журнали, виставляти оцінки, задавати домашні завдання. Батьки мають можливість слідкувати за навчальними досягненнями дітей, отримувати новини та інформацію про роботу школи.

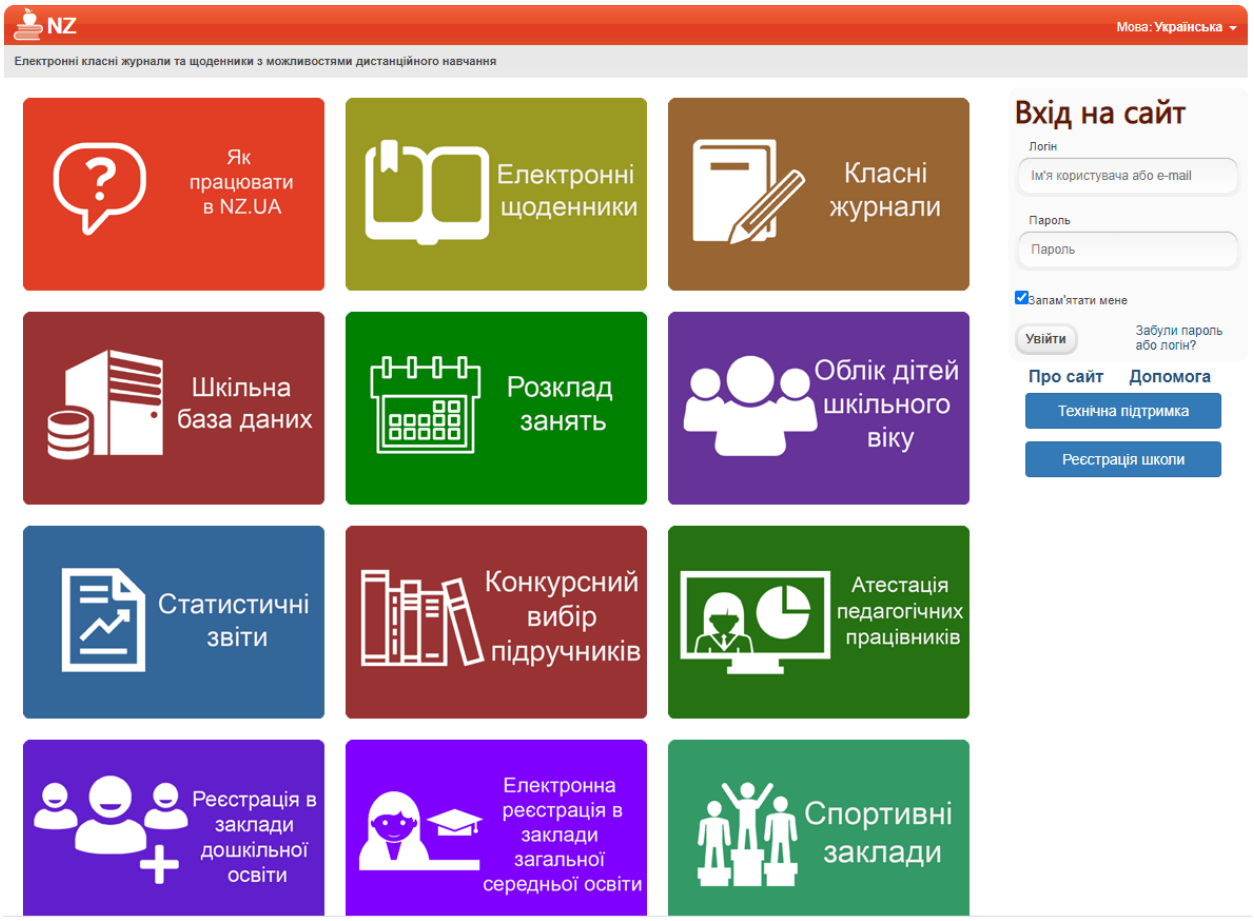

Рис. 1.2. Портал www.nz.ua

Четвертий компонент комплексу – програма «КУРС: дошкілля». Вона є аналогом «КУРС: школа», дозволяє вести базу даних дошкільного дитячого закладу. У програмі можливе складання обов'язкових звітів. Програма веде облік педагогічного складу, вихованців, їхніх батьків та опікунів. Сайт підтримки: www.dnz.ekurs.org.

Висновок

Ринок програмного забезпечення для середніх загальноосвітніх закладів порівняно молодий і мало насичений. Лише в останні роки заклади освіти почали частково переходити на електронний документообіг. Причин цьому декілька. По-перше, це недостатнє фінансування. По-друге, це нестача проєктів по підвищенню комп'ютерної грамотності працівників освіти. По-третє, це нерозвинена

інфраструктура. Не останнім фактором є інертність і небажання змін серед учасників освітнього процесу.

Системи управління школою досить поширені закордоном. Однак останніми роками все більше областей нашої країни почали вводити електронні системи, які об’єднують школи, дитячі садки, районні та обласні відділи освіти у єдину мережу. Зараз у багатьох середніх загальноосвітніх закладах ведеться база даних персоналу, учнів, складаються електронні звіти. «Нова школа» працює з електронними щоденниками та журналами.

Електронні системи менеджменту автоматизують багато процесів, що раніше виконувались вручну. Типовими сценаріями застосунку є ведення обліку персоналу, учнів, журналу уроків, екскурсій, звітності, складання розкладу, облік книжкового фонду, зв’язок з батьками.

Системи управління школою ще досить недосконалі, вони не враховують регіональні та місцеві специфіки, мають функціонал, який у реальних умовах використовувати незручно чи навіть контрпродуктивно. Деякі функції таких систем не приживаються, а результатом є спротив персоналу, якому нав’язують зайву роботу замість обіцяного спрощення і автоматизації.

Однак майбутнє галузі освіти – це єдина електронна система, яка надає адміністрації різних рівнів, персоналу, учням та батькам дієві інструменти управління, моніторингу, автоматизації та стандартизації.

РОЗДІЛ 2
ТЕХНОЛОГІЇ ВЕБ-ЗАСТОСУНКУ

2.1. ASP.NET Core як феймворк веб-застосунку

ASP.NET Core є безкоштовним веб фреймворком з відкритим кодом, що дозволяє створювати веб-застосунки на мові C#. Фреймворк розроблений компанією Microsoft і є крос-платформеним наступником ASP.NET. З версії 3 ASP.NET Core працює лише на платформі .NET Core. Переваги .NET Core включають крос-платформеність, відкритість коду, покращену швидкодію, нові API.

Фреймворк ASP.NET Core було повністю переписано і він об’єднав у собі окремі продукти ASP.NET MVC та ASP.NET Web Api. Пізніше, одним з необов’язкових компонентів фреймворку став Blazor, що дозволило створювати інтерактивні веб-застосунки за допомогою C# [3].

ASP.NET Core дозволяє:

- будувати веб-застосунки і сервіси, застосунки для інтернету речей, мобільні бекенди;
- вести розробку на Windows, macOS та Linux;
- розгортати застосунки на хмарних сервісах чи локально;
- працювати на платформі .NET Core.

Переваги ASP.NET Core:

- уніфікована система для побудови як інтерфейсів користувачів, так і веб API;
- легкість впровадження модульних тестів;
- Razor Pages полегшує створення веб-сторінок, підвищує продуктивність;
- Blazor дозволяє використовувати C# у браузері разом з JavaScript. Можливо створювати серверну і клієнтську частини однією мовою програмування;
- інтеграція з сучасними клієнтськими фреймворками (Blazor, Angular, React, Bootstrap);

- готове рішення для хмарних технологій;
- високопродуктивний, модульний конвеєр HTTP запитів;
- вбудоване впровадження залежностей (dependency injection);
- наявність інструментів, що спрощують сучасну веб-розробку [4].

Веб-застосунок «Система управління школою» працює локально на HTTP сервері IIS та розгорнутий на хмарний сервіс Microsoft Azure.

ASP.NET Core використовує шаблон дизайну «Модель-представлення-контролер» (англ. Model-View-Controller (MVC)). Razor синтаксис дає змогу використовувати мову програмування C# у HTML файлах. Завдяки Tag Helpers серверний код приймає участь у створенні та відображенні HTML елементів. Веб API підтримує багато форматів даних і може працювати на різних пристроях, в тому числі мобільних. Прив'язка до моделі автоматично прив'язує HTTP запити до методів контролера. Валідація моделі відбувається як на сервері, так і на клієнті.

Razor сторінки з'явились з виходом фреймворку .NET Core 2.0, який спростив створення веб-сторінок. Razor сторінки схожі на представлення ASP.NET MVC. Проте фреймворк .NET Core 2.0 має у своїй основі модель MVVM (Model-View-ViewModel). Одним з недоліків шаблону MVC були файли-контролери, які росли до великих розмірів по мірі розширення функціоналу застосунку. Razor сторінки містять лише те, що необхідно для їхньої роботи. Виконується принцип єдиної відповідальності [5].

У шаблоні MVVM (модель – представлення – модель представлення) модель є набором класів, що описують бізнес-логіку і дані. Модель визначає набір правил для роботи з даними. Представлення містить у собі HTML код, CSS, JavaScript, jQuery та інші компоненти. Представлення відповідає лише за показ інформації, яку надсилає модель представлення. Модель представлення містить інструменти, необхідні для роботи представлень, методи для зміни моделі, відповідає за обробку подій у представленні [6] (рис. 2.1).

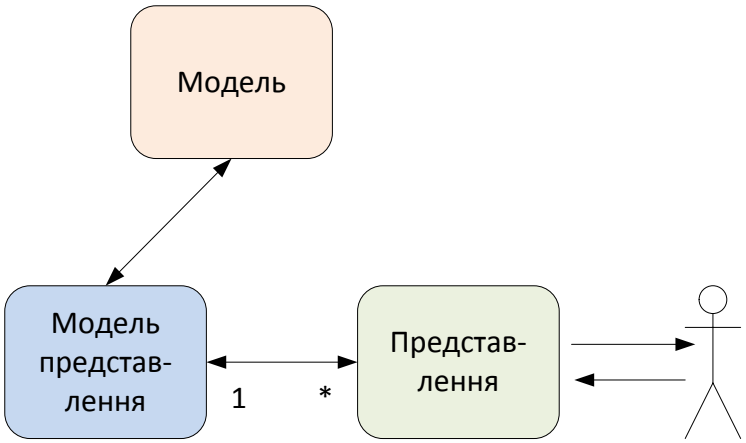

Рис. 2.1. Шаблон MVVM

У кожному веб-застосунку є клас «Startup». З цього класу починається робота застосунку. У «Startup» підключаються і налаштовуються сервіси, необхідні для роботи застосунку. Наприклад, у застосунку «Система управління школою», тут підключається провайдер бази даних SQL Server, сервіси автентифікації, авторизації, налаштовується маршрутизація тощо.

ASP.NET Core поставляється з фреймворком впровадження залежностей (dependency injection). За його допомогою сервіси, що підключені у класі Startup стають доступними у всьому застосунку. Наприклад, у «Системі управління школою» за допомогою впровадження залежностей у будь-який клас імпортуються контекст бази даних та модуль протоколювання. На рисунку видно, як це зроблено через конструктор класу (рис. 2.2).

```
private readonly sms.Data.ApplicationDbContext _context;
private readonly ILogger<IndexModel> _logger;
public Generator(sms.Data.ApplicationDbContext context,
                 ILogger<IndexModel> logger)
{
    _context = context;
    _logger = logger;
```

Рис. 2.2. Впровадження залежностей

ASP.NET Core застосунок обробляє HTTP контекст (HTTPContext) за допомогою конвеєра проміжних програмних компонентів. Назва кожного компонента починається зі слова «Use» (рис. 2.3).

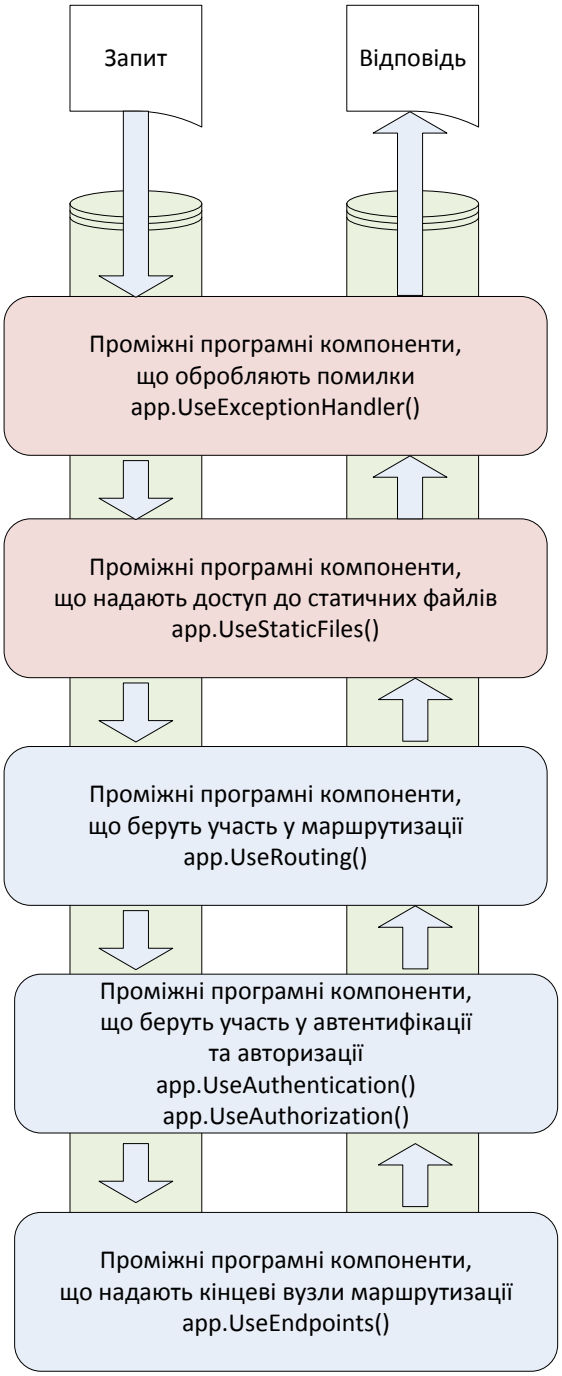

Рис. 2.3. Конвеєр проміжних компонентів ASP.NET Core [7]

Коли до веб-застосунку на базі ASP.NET Core надходить запит, маршрутизатор направляє його до обробника сторінки OnGet() та будує модель

зв'язування. Обробник сторінок викликає сервіси, які містяться у моделі застосунку, отримує інформацію з бази даних та будує модель представлення. Обробник сторінок надсилає у представлення модель та необхідні дані для відображення. Представлення використовує модель для генерації HTML відповіді, яка повертається користувачу (рис. 2.4) [7].

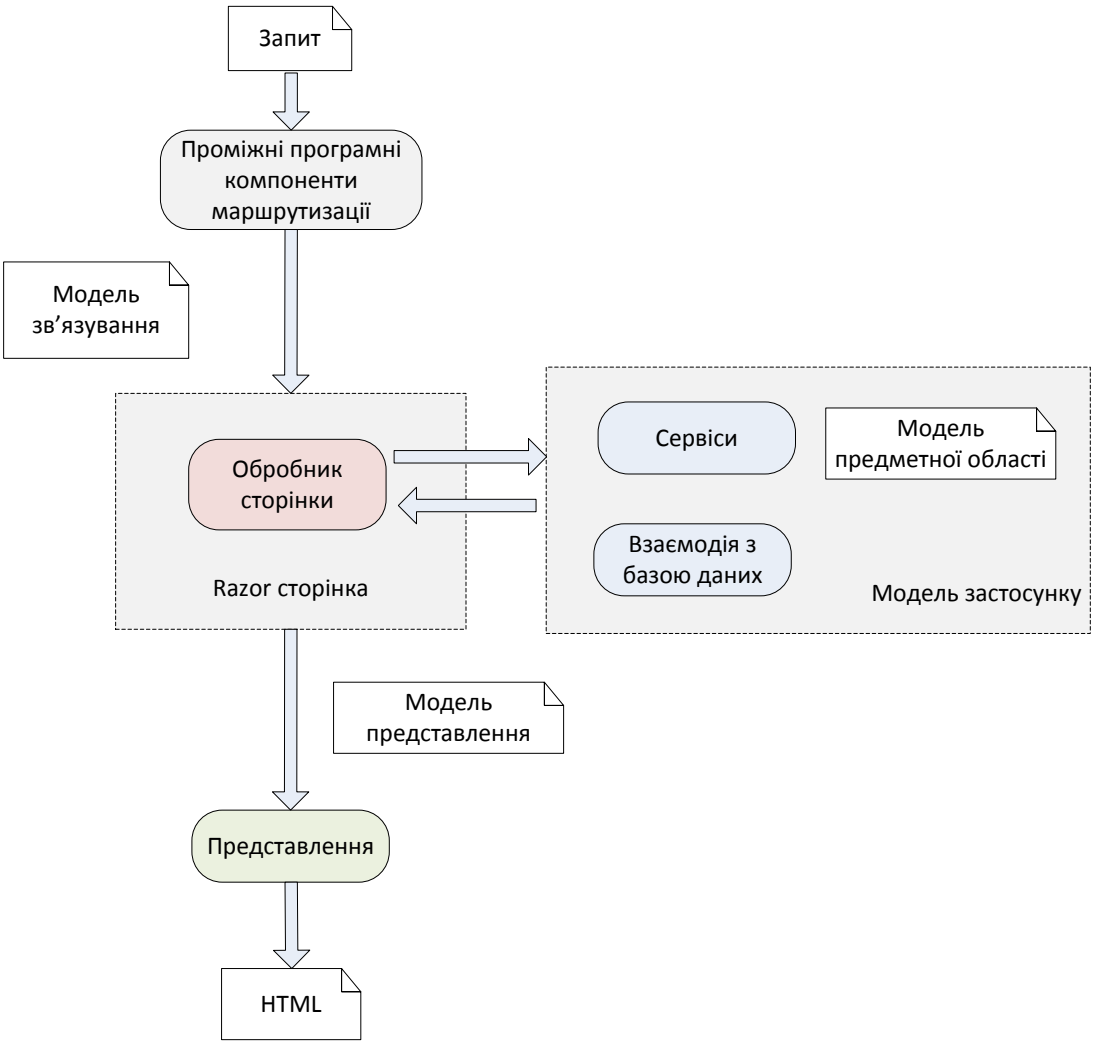

Рис. 2.4. Запит до Razor сторінки

На початку роботи, веб-застосунок ASP.NET Core будує так званий host. Хост поєднує у собі всі ресурси застосунку:

- HTTP сервер;
- проміжні програмні компоненти;
- функціонал протоколювання;

– сервіси впровадження залежностей;
– конфігурування [8].

Під час побудови хосту завантажується конфігурація з файлу «appsettings.json». У застосунку «Система управління школою» у цьому файлі зберігаються інформація про шлях до бази даних SQL Server, налаштування автентифікації Google, список дозволених хостів, а також тестовий пароль для ініціалізації бази даних. Налаштування також завантажуються зі змінних оточуючого середовища, що мають вищий пріоритет, ніж файл «appsettings.json». Ще одне джерело налаштувань – командний рядок.

ASP.NET Core застосунки можуть запускатись у таких режимах:

– режим розробника (development environment);
– передвиробничий, постановчий режим (staging environment);
– виробничий режим (production environment).

Застосунок ASP.NET Core зчитує інформацію про середовище зі змінної «ASPNETCORE_ENVIRONMENT». В залежності від значення цієї змінної, виконується різний код і користувач бачить різний зміст сторінок. Змінюються налаштування безпеки та налагоджування (debugging).

Development environment – це середовище, де версія продукту постійно змінюється розробником. Staging environment – це передвиробниче середовище, що якомога точно копіює виробниче середовище і служить для фінального стрес-тестування програмного продукту. Production environment – виробниче (або живе) середовище – це середовище, з яким кінцеві користувачі взаємодіють безпосередньо [9].

Інтегроване середовище розробки Visual Studio при створенні проєкту автоматично створює файл «launchSettings.json». У ньому зберігаються профілі для запуску застосунків у режимі розробника, постановочному чи виробничому режимі.

У застосунку «Система управління школою» у файлі «Startup.cs» містяться сценарії запуску у різних режимах. При запуску на локальному комп’ютері застосунок працює у режимі розробника. На сервері Microsoft Azure за

замовчуванням встановлено виробниче середовище, тому застосунок використовує інші налаштування при роботі у хмарі. Зокрема, для доступу до виробничої бази даних використовуються інші налаштування підключення, якщо застосунок запускається у виробничому середовищі.

Режим запуску застосунку можна обрати у властивостях проєкту Visual Studio (Project Settings – Debug).

2.2. Entity Framework Core – технологія доступу до даних веб-застосунку

Entity Framework (EF) Core – це кросплатформена версія популярної технології доступу до даних Entity Framework з відкритим кодом. EF Core може працювати як об’єктно-реляційний відображувач/перетворювач (object-relational mapper, OR/M), який:

- дозволяє .NET розробникам працювати з базами даних, використовуючи .NET об’єкти;
- спрощує написання коду для доступу до даних [10].

Об’єктно-реляційний відображувач (O/RM) дозволяє обмінюватись даними з несумісними системами, використовуючи об’єктно-орієнтовані мови програмування. Створюється віртуальна об’єктна база даних, запити до якої можна формувати засобами мови програмування [11].

Бази даних оперують переважно скалярними величинами (числа, рядки), які вони зберігають у таблицях, а мови програмування мають справу з об’єктами. Об’єктно-реляційний відображувачі перетворюють об’єкти об’єктно-орієнтованих мов програмування у скалярні величини для збереження у базах даних і, навпаки, отримують з баз даних скалярні величини і створюють об’єкти на їхній основі. Об’єкти, які таким чином збережені у базі даних, називають стійкими (persistent).

Об’єктно-реляційні перетворювачі значно зменшують кількість коду, необхідного для реалізації взаємодії з базами даних. Недоліком їх є високий рівень абстракції, що приховує процеси, які відбуваються у імплементації. O/RM полегшує

розробку, але масове використання цієї технології стало одним з головних факторів погіршення якості проєктування баз даних [10].

EF Core підтримує таких постачальників баз даних:

- SQL Server;
- SQLite;
- Azure Cosmos DB SQL API;
- PostgreSQL;
- MySQL;
- MariaDB;
- Oracle DB;
- Firebird 3.0;
- Microsoft Access;
- та інші.

2.2.1. Моделі та міграції

У EF Core доступ до даних здійснюється за допомогою моделі. Модель складається з класів-сутностей та об’єкту «контекст», який надає доступ до інформації у базі даних і дозволяє зберігати у ній дані.

Перенести структуру бази даних на модель можна наступним чином:

- на основі існуючої бази даних, згенерувати похідну модель;
- аналізуючи структуру бази даних, вручну написати код моделі.

І навпаки, маючи модель, за допомогою міграцій можна створити базу даних. Коли змінюється модель, міграція допомагає автоматично змінити базу даних так, щоб вона відповідала моделі. Так, наприклад, якщо додати нову властивість, то міграція створить відповідну колонку у таблиці бази даних. Якщо змінити тип властивості, міграція змінить тип колонки. Управляти міграцією можна з «Package Manager Console» у Visual Studio або через консоль «Windows Powershell».

Після кожної міграції створюється окремий клас з часовим штампом на початку імені файлу та назвою міграції. Міграції застосунку «Система управління школою» показані нижче (рис. 2.5).

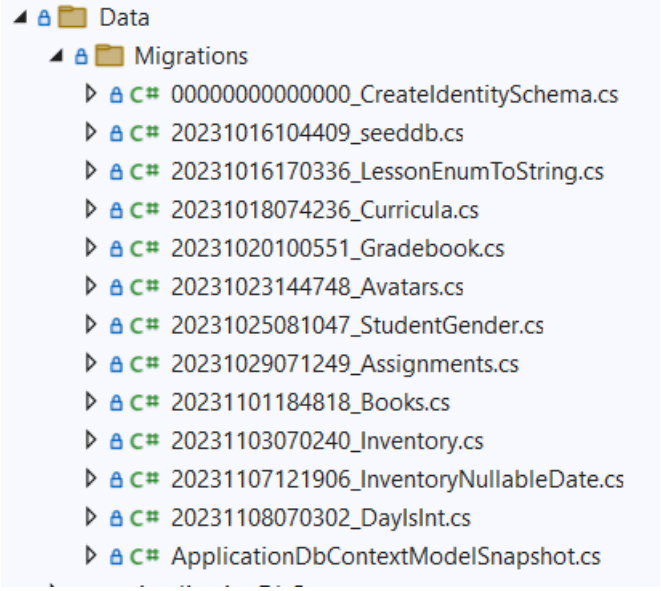

Рис. 2.5. Файли міграції веб-застосунку

Кожна міграція містить два методи: Up() – для внесення змін до бази даних та Down() – для повернення до попередньої версії бази даних.

Актуальна структура бази даних міститься у файлі «ApplicationDbContextModelSnapshot». Це знімок поточної бази даних, який необхідний для генерації файлу міграції. Саме завдяки цьому знімку EF Core визначає, що змінилось у базі даних.

Конфігурація моделі може здійснюватися декількома способами. Один з них – це анотації. За допомогою анотацій можна задати мінімальну і максимальну довжину полів, вказати тип та формат даних, альтернативну назву поля, колонку таблиці, до якої належатиме поле тощо (рис. 2.6).

Ще один спосіб конфігурації моделі – мова «Fluent API». Це потужний інструмент, який дозволяє налаштувати модель на найнижчому рівні. Багато налаштувань Fluent API не мають аналогів у анотаціях. Fluent API використовується у методі OnModelCreating() при описі об’єкту контексту.

```
[DataType(DataType.Date)]
[DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
[Display(Name = "Дата")]
16 references | gform, 18 days ago | 1 author, 1 change
public DateTime DateOfPost { get; set; } = DateTime.Now;
```

Рис. 2.6. Конфігурація моделі за допомогою анотації

Нижче можна побачити, як через Fluent API налаштовується колонка «ціна» таблиці «Inventories»: встановлюється точність та масштаб (precision and scale) (рис. 2.7).

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    base.OnModelCreating(modelBuilder); // do not comment, causes
    modelBuilder.Entity<Inventory>()
        .Property(o => o.Price).HasColumnType("decimal(18,2)");
}
```

Рис. 2.7. Fluent API для конфігурації моделі

На мові Fluent API описується структура бази даних у файлі-знімку, який потрібний для генерації міграцій.

У веб-застосунку «Система управління школою» об’єкт контексту створюється у файлі «ApplicationDbContext.cs».

Запити до бази даних здійснюються через Language Integrated Query (LINQ). Ця мова має два види синтаксису:

- синтаксис запитів, що схожий на мову SQL;
- синтаксис методів, який більш близький до синтаксису мов програмування.

У веб-застосунку «Система управління школою» використовується синтаксис методів.

Дані маніпулюються за допомогою екземплярів класів-сутностей: створюються, видаляються, змінюються, зберігаються.

Хоча EF Core значно спрощує «спілкування» з базою даних, для створення ефективних і швидких програм необхідно мати добрі знання архітектури бази даних, з якою відбувається взаємодія. Наприклад, це знання про первинні і зовнішні ключі, нормалізацію, індексацію, обмеження бази даних, мови DML і DDL для взаємодії з базою даних, типи даних, профілювання даних [10].

2.2.2. Механізм відслідковування змін (Change Tracking)

У Entity Framework Core є важливий механізм «відслідковування змін» (Change Tracking). Екземпляри сутностей попадають у список таких, що відслідковуються, у таких випадках:

- коли екземпляри сутностей отримані з бази даних шляхом запитів;
- коли екземпляри сутностей явно приєднують до контексту командами Add, Attach, Update;
- коли екземпляри сутностей пов’язані з сутностями, що вже відслідковуються [12].

Кожна сутність EF Core має один з таких станів (рис. 2.8):

- відокремлена (Detached) – не відслідковується;
- додана (Added) – нова сутність, що ще не збережена у базі даних командою «SaveChanges()». Дія у базі даних – INSERT;
- без змін (Unchanged) – саме у такому стані перебувають всі сутності, які отримані з бази даних через запити;
- змінена (Modified) – сутності змінились відтоді, як їх отримали з бази даних. Ці сутності оновлять у базі даних, коли виконуватиметься команда «SaveChanges()». Дія у базі даних – UPDATE;
- видалена (Deleted) – сутність ще існує у базі даних, але вона буде видалена по команді збереження змін. Дія у базі даних – DELETE [12].

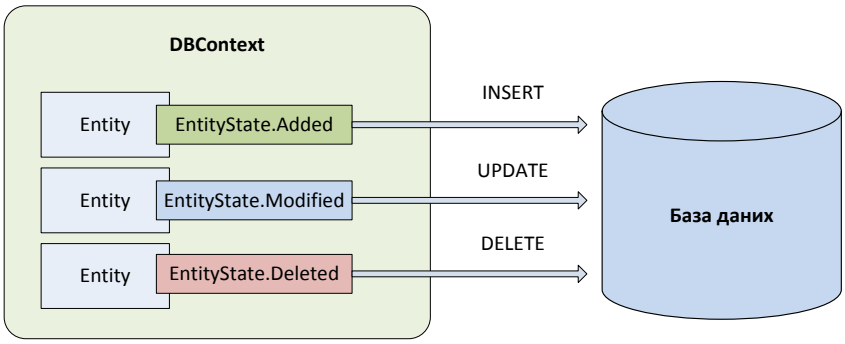

Рис. 2.8. Збереження даних у EF Core

Окрім автоматичного режиму відстеження змін, можна вручну змінювати стани сутностей.

На рисунках нижче можна побачити, як у веб-застосунку «Система управління школою» створюються, редагуються і видаляються сутності (рис. 2.9, рис. 2.10 та рис. 2.11).

```
//Save new item to DB
//Збереження створеної сутності у БД
_context.Grades.Add(Grade);
await _context.SaveChangesAsync();

return RedirectToPage("./Index");
```

Рис. 2.9. Додавання нової сутності до контексту

```
//Save changes to DB
//Збереження відредагованих змін у БД
_context.Attach(Curriculum).State = EntityState.Modified;

try
{
    await _context.SaveChangesAsync();
}
```

Рис. 2.10. Редагування сутності

```
//Delete student from DB
//Видалення учня з БД
if (Student != null)
{
    _context.Students.Remove(Student);
    await _context.SaveChangesAsync();
```

Рис. 2.11. Видалення сутності

2.2.3. Зв’язки між класами-сутностями

У веб-застосунку «Система управління школою» використовуються такі два види зв’язків між класами-сутностями:

- один-до-багатьох;
- багато-до-багатьох.

Для того, щоб реалізувати ці зв’язки у Entity Framework Core є ряд умовностей. Зв’язок буде створений тоді, коли у класі-сутності є навігаційна властивість, що вказує на інший клас-сутність. У Entity Framework Core зв’язки можна описувати з різним рівнем деталізації. Відсутність необхідності повністю описувати зв’язки спрощує програмування. У веб-застосунку «Система управління школою» усі зв’язки описані повністю. Це зроблено для зручності читання тексту програми та для уникнення неоднозначності при трактуванні зв’язків програмістом. Для EF Core повний опис зв’язку необов’язковий.

Повний опис зв’язку один-до-багатьох передбачає присутність навігаційних властивостей на обох боках зв’язку, тобто у обох класах-сутностях та присутність зовнішнього ключа у залежному класі [13]. Якщо зовнішнього ключа не вказати, автоматично створюється прихований тіньовий ключ. Можна також не вказувати навігаційну властивість у залежному класі.

У веб-застосунку «Система управління школою» зв’язок один-до-багатьох представлений у класах-сутностях «Клас» та «Учні» (рис. 2.12, рис. 2.13).

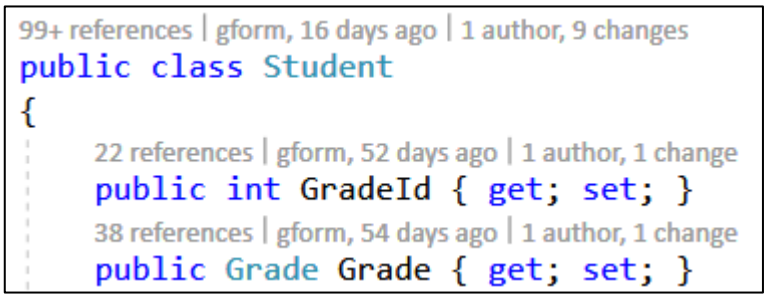

Рис. 2.12. Навігаційні властивості сутності «Учень»

Рис. 2.13. Навігаційні властивості сутності «Клас»

До версії Entity Framework Core 5 зв’язок багато-до-багатьох необхідно було створювати за допомогою додаткової зв’язуючої таблиці. Тепер це необхідно лише тоді, коли у зв’язуючу таблицю поміщають додаткову інформацію. Зв’язок багато-до-багатьох створюється автоматично, якщо на обох боках зв’язку присутня навігаційна властивість у вигляді колекції.

У веб-застосунку «Система управління школою» прикладами зв’язку багато-до-багатьох є наступні пари класів:

– «Вчитель» і «Предмет»;
– «Вчитель» і «Книга»;
– «Учень» і «Книга».

На рисунках нижче можна побачити приклад зв’язку багато-до-багатьох між вчителем і предметом (рис. 2.14, рис. 2.15).

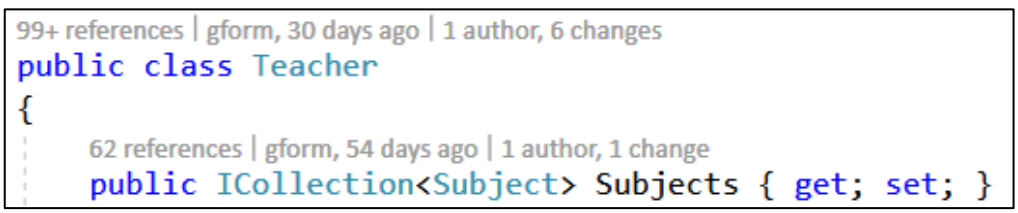

Рис. 2.14. Навігаційні властивості класу «Вчитель»

Рис. 2.15. Навігаційні властивості класу «Предмет»

2.3. Перевірка введених користувачем даних

Перевірка введених даних передбачає:

- перевірку типу даних;
- перевірку діапазону допустимих значень;
- перевірку наявності даних.

Якщо дані не перевіряти, це призведе до неправильної роботи програми, спотворення результатів роботи або навіть до зупинки програми з помилкою винятку.

Перевірці підлягають:

- форми, заповнені користувачами;
- значення, які містяться у адресному рядку;
- дані у файлах «cookie».

Оптимальною політикою безпеки є така: за замовчуванням не довіряти жодним даним, отриманим від користувачів. Перевірка даних, введених користувачем, може здійснюватися у двох місцях:

- на клієнтському комп’ютері (клієнтський скрипт чи вбудовані у браузер інструменти валідації);
- на сервері (рис. 2.16).

Локальна перевірка є ненадійною, її легко обійти. Навіть початкові знання HTML та JavaScript достатні для обходу локальної перевірки. Інструменти браузера змінюються від версії до версії і на цей інструмент перевірки теж не слід покладатись. Клієнтська перевірка введених даних слугує для зручності користувача, тому що неправильні дані миттєво підсвічуються, і це дає зрозуміти, де

саме сталася помилка. Локальна перевірка – лише перший етап перевірки. Завдяки локальній перевірці можливо зменшити навантаження на сервер шляхом відсікання завідомо неправильних даних.

Рис. 2.16. Валідація даних, введених користувачем

Фреймворк MVC, який є основою Razor сторінок, має надійну систему перевірки, яка включає у себе перевірку як на клієнті, так і на сервері. Перевірці підлягають введені властивості моделі.

У перевірці беруть участь:

- атрибути анотації даних (DataAnnotation Attributes);
- теги-помічники (Tag Helpers);
- непомітна валідація jQuery (jQuery Unobtrusive Validation);
- стан моделі (ModelState);
- обмеження маршрутизації (Route Constraints).

Атрибути анотації даних у веб-застосунку «Система управління школою» представлені атрибутами [Required] (обов'язкове поле), [StringLength] (довжина поля) та [RegularExpression] (регулярний вираз) (рис. 2.17).

```
[Required(ErrorMessage = "Поле \"Прізвище\" обов'язкове")]
[StringLength(50, ErrorMessage = "Прізвище не може бути довше за 50 символів.")]
[RegularExpression(@"^[А-ЯІЇЄ']+[а-яА-Яіїє'-]*$")]
[Display(Name = "Прізвище")]
```

Рис. 2.17. Атрибути анотації у класі «Учень»

Для контролю введених даних використовуються маски з регулярних виразів (таблиця 2.1).

Таблиця 2.1

Вимоги до форми створення запису про учня

Елемент	**Тип елемента**	**Регулярний вираз**	**Вимоги**
Прізвище	text-box	^[А-ЯІЇЄ']+[а-яА-Яіїє'-]*$	Обов'язкове поле. Максимальна довжина 50. Дозволені лише букви кирилиці і тире. Перша буква – велика.
Ім'я	text-box	^[А-ЯІЇЄ']+[а-яА-Яіїє'-]*$	Обов'язкове поле. Максимальна довжина 50. Дозволені лише букви кирилиці і тире. Перша буква – велика.
По батькові	text-box	^[А-ЯІЇЄ']+[а-яА-Яіїє'-]*$	Обов'язкове поле. Максимальна довжина 50. Дозволені лише букви кирилиці і тире. Перша буква – велика.
Дата народження	text-box	Контролюється атрибутами [DisplayFormat]	Обов'язкове поле. Тип: дата. Формат уууу-ММ-dd.
Адреса	text-box	^[а-яА-Я0-9A-Za-zІЇЄіїє""\s-.,]*$	Максимальна довжина 50. Дозволені цифри і букви (кирилиця, латиниця), дефіс, пробіл, подвійні та одиничні лапки, крапка, кома.

Для непомітної валідації jQuery на кожну веб-сторінку додається файл «_ValidationScriptsPartial.cshtml».

Для безпосередньої перевірки на стороні клієнта використовуються теги-помічники asp-validation-for, asp-validation-summary. Теги-помічники генерують спеціальні HTML5 data-* атрибути.

Внаслідок роботи jQuery валідації під полем з помилкою з’являється текст написаний червоним кольором (рис. 2.18).

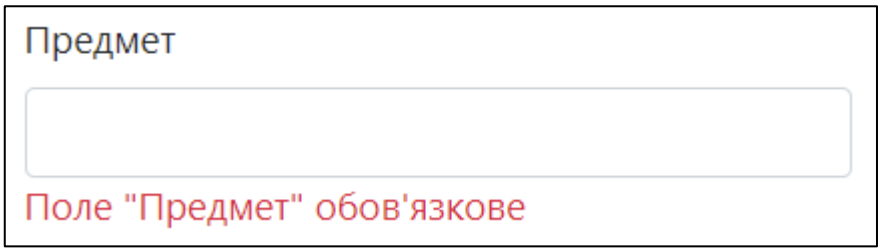

Рис. 2.18. jQuery перевірка обов’язковості полів

Другий етап перевірки введених даних – перевірка на сервері засобами фреймворку ASP.NET Core. Після аналізу властивостей моделі, усі помилки записуються у спеціальний словник «ModelStateDictionary». У веб-застосунку «Система управління школою» перевірка моделі на помилки відбувається після створення чи редагування сутностей. На рисунку нижче можна побачити, як новий запис заноситься у базу даних лише за умови відсутності помилок «ModelState» (рис. 2.19).

```
public async Task<IActionResult> OnPostAsync()
{
    if (!ModelState.IsValid)
    {
        return Page();
    }

    //Add new record to DB
    //Додати новий запис до БД
    _context.Inventories.Add(Inventory);
    await _context.SaveChangesAsync();

    return RedirectToPage("./Index");
}
```

Рис. 2.19. Перевірка моделі на рівні серверу

У веб-застосунку використовується бібліотека «FluentValidation» для перевірки правильності введення даних користувачами. Перевірка відбувається на рівні серверу. Бібліотека встановлюється:

– засобами NuGet package manager (Install-Package FluentValidation);
– командним інтерфейсом dotnet CLI (dotnet add package FluentValidation).

Бібліотека FluentValidation використовується у даному застосунку для перевірки правильності введення дат (рис. 2.20).

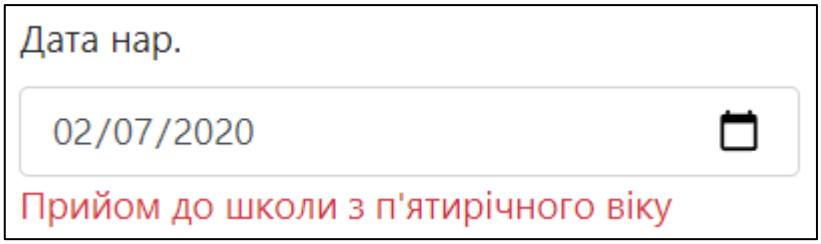

Рис. 2.20. Перевірка дати народження учня

Щоб створити набір правил для перевірки певного об’єкту, необхідно створити клас, який наслідує від «AbstractValidator<T>», де «<T>» - це тип класу, який необхідно валідувати. Щоб задати правило перевірки для певної властивості, викликається метод «RuleFor», в який передається лямбда вираз, в якому вказується властивість, яку необхідно валідувати [14] (рис. 2.21).

```
public StudentValidator()
{
    RuleFor(m => m.DateOfBirth)
        .LessThan(DateTime.Now.AddYears(-5))
        .WithMessage("Прийом до школи з п'ятирічного віку");

    RuleFor(m => m.DateOfBirth)
        .GreaterThan(DateTime.Now.AddYears(-19))
        .WithMessage("Прийом до школи до 18 років включно");
}
```

Рис. 2.21. Код FluentValidation перевірки віку учня

2.4. Тестування веб-застосунку

Тестування необхідне, щоб впевнитись, що код працює так, як це задумувалось. Тести необхідні для підтримки коду у належному стані, для знаходження помилок до того, як це зроблять користувачі. Для цього слугують модульні, інтеграційні та інші тести. Під час тестування функціонал програми розбивається на дискретні частини, поведінку яких можна перевірити окремо від інших за допомогою модульних тестів.

Модульні тести дозволяють швидко перевірити наявність логічних помилок у методах класів.

Середовище розробки Visual Studio надає гнучкі та ефективні інструменти виконання модульних тестів та перегляду їх результатів. Test Explorer показує тести за такими категоріями: провалені тести (Failed Tests), пройдені тести (Passed Tests), пропущені тести (Skipped Tests), не виконувані тести (Not Run Tests). Тести можна виконувати вибірково, статистика виконання/невиконання відображається вгорі тестового вікна (рис. 2.22).

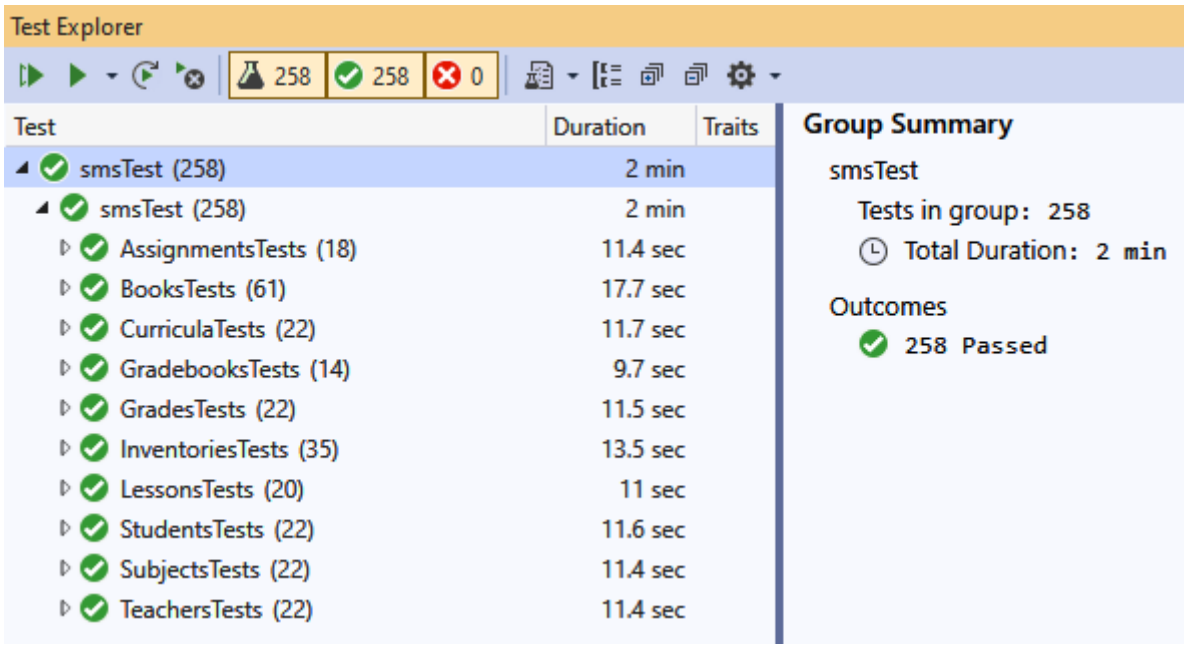

Рис. 2.22. Перегляд тестів у Visual Studio

Тести можна групувати у категорії, фільтрувати список, створювати, зберігати та запускати список тестів. Якщо тести не залежать один від одного, то можна увімкнути паралельний запуск, тоді тести виконуватимуться значно швидше. Для запуску тестів після кожної компіляції проєкту існує налаштування робити це автоматично. Режим відлагодження тестів дозволяє знаходити помилки у самих тестах.

Для створення модульних тестів використовують спеціальні фреймворки. Після кожної зміни у коді програми, тести запускають знову, щоб впевнитись, що код продовжує працювати правильно. Версія Visual Studio Enterprise для підприємств автоматично виконує ці тести, коли змінюється код. Live Unit Testing у реальному часі показує результати тестів та відсоток охопленості коду тестами. Такий зворотній зв'язок дозволяє побачити, як зміни у коді вплинули на існуючі тести, а також слугує нагадуванням створювати модульні тести по мірі виправлення помилок чи додавання нового функціоналу [15].

Visual Studio після встановлення необхідних розширень може запускати також сторонні тестові фреймворки. Найпопулярнішими тестовими бібліотеками для платформи .NET є MSTest, xUnit.net, NUnit. Тести можна створювати автоматично з існуючого коду за допомогою інструменту IntelliTest чи писати вручну.

Модульне тестування має великий вплив на якість коду, коли це невід'ємна частина процесу розробки. Тести варто писати одразу після створення функції чи бізнес логіки щоб перевірити, як код працює зі стандартними, межовими та нестандартними вхідними параметрами та припущеннями, які наявні у коді. Існує підхід розробки через тестування (test driven development – TDD), коли модульні тести створюються до написання коду. В такому випадку тести використовуються як проєктна документація і як функціональні специфікації.

Для того, щоб ефективно запобігати помилкам у коді, тести повинні покривати досить велику частину коду. Для перевірки охопленості коду тестами у Visual Studio Enterprise наявний інструмент «Code coverage tool».

Модульні тести зазвичай пишуть за моделлю AAA (Arrange, Act, Assert) – підготовка, виконання, оцінка:

- у секції «Arrange» ініціалізуються об’єкти та встановлюються значення змінних, які передаються методу, який тестують;
- у частині «Act» запускається на виконання метод, дані для якого були підготовані на етапі Arrange;
- на заключному етапі «Assert» оцінюється, чи поводить себе метод так, як очікується [16].

Для тестування сторінок даного веб-застосунку використовується безкоштовний інструмент xUnit.net. Він має відкритий код, підтримується спільнотою та незалежною організацією .NET Foundation. xUnit.net використовується для тестування мов сімейства .NET: C#, F#, VB.NET та інших. Він працює з такими розширеннями: ReSharper, CodeRush, TestDriven.NET та разом з Xamarin [17]. xUnit.net. встановлюється у середу розробки Visual Studio як NuGet пакет.

Інша бібліотека, яка використовується для тестування цього веб-застосунку – Moq. Це популярна і легка у використанні строго типізована бібліотека, що імітує .NET сутності, методи, інтерфейси, класи, вирази. Вона також працює з мовою запитів Linq, лямбда виразами. Moq створювалась з нуля для тих розробників, хто або не використовував імітацію у своїх тестах, або писав власні імплементації. Moq має низький поріг входження та повністю інтегрований у довідкову систему Visual Studio – IntelliSense [18].

За допомогою xUnit.net та Moq було створено більше 250 модульних тестів для веб-застосунку «Система управління школою». Тести перевіряють наступні аспекти програми:

- видалення сутності коли її знайдено у БД і коли не знайдено;
- показ сутності коли її знайдено у БД і коли не знайдено;
- редагування сутності з правильними та неправильними вхідними даними;

- створення сутності з правильними та неправильними вхідними даними;
- відображення списку сутностей у алфавітному і зворотному порядку;
- фільтрація списку сутностей правильними і неправильними вхідними даними;
- перехід по сторінкам списку сутностей з правильними і неправильними номерами сторінок.

Написані модульні тести використовують контекст (базу даних), що очищається до початкового стану після кожного виконаного тесту. Це можливо завдяки тому, що тестовий клас імплементує інтерфейс «IDisposable», містить у своєму конструкторі ініціалізацію даних, а у методі «Dispose» - їх очистку. Для кожного тесту xUnit.net створює новий екземпляр тестового класу, тому код у конструкторі виконується для кожного тесту. Якщо помістити у конструктор ініціалізацію даних, то вони будуть унікальними для кожного тесту. Таким чином створення і очистка контексту однакові для всіх тестів, а контекст – індивідуальний. Таку модель інколи називають «тестовий клас як контекст». У xUnit.net можливий також інший варіант – коли контекст спільний для всіх тестів певного класу чи навіть різних класів [19].

Під час написання модульних тестів код застосунку було уніфіковано, були виявлені та усунуті деякі помилки, додані додаткові перевірки вхідних даних на рівні бізнес логіки. Зокрема, була додана перевірка на null у деяких частинах коду. Виправлена помилка поділу сторінок, тепер при вказанні неправильної сторінки дані надаються з першої сторінки. Циклічний процес аналізу, розробки тестів та виправлення роблять результуючий код більш стійким до помилок та ефективним [20].

2.5. Протоколювання

Протоколювання (логування) не тільки слугує інструментом вирішення проблем та помилок. Воно дає картину про те, як клієнти використовують застосунок. Через протоколи ми дізнаємось про помилки користувачів та проблеми

безпеки. З одного боку важливо, щоб протоколи містили достатньо інформації про те, які дії виконувались до виникнення помилки. З іншого боку, слід пам’ятати, що протоколювання може знижувати продуктивність застосунку, а файли-логи можуть досягати значних розмірів, якщо протоколювати кожну дію. Фреймвморки для протоколювання мають налаштування рівнів інформації, яка підлягає логуванню, що спрощує процес налагодження.

При створенні ASP.NET Core веб-застосунку за замовчуванням додаються наступні провайдери протоколювання:

– консоль (console);
– вікно відлагодження (debug);
– джерело подій (EventSource);
– протокол подій для Windows (EventLog) [21].

Протоколювання конфігурується у JSON-файлі «appsettings.json» (рис. 2.23). У цьому файлі не вказується конкретний провайдер протоколювання. Ці налаштування діють на всі провайдери. Кожний фреймворк логування може мати додаткові налаштування.

```
"Logging": {
  "LogLevel": {
    "Default": "Information",
    "Microsoft": "Warning",
    "Microsoft.Hosting.Lifetime": "Information"
  }
},
```

Рис. 2.23. Налаштування протоколювання

Рівень протоколювання (LogLevel) визначає мінімальний рівень логування для вказаних категорій. LogLevel вказує на серйозність логу і варіюється в межах від 0 до 6:

– Trace = 0 (протоколювання кожного кроку);

– Debug = 1 (протоколювання інформації налагодження);
– Information = 2 (протоколювання інформаційних повідомлень);
– Warning = 3 (протоколювання застережень);
– Error = 4 (протоколювання помилок);
– Critical = 5 (протоколювання критичних збоїв);
– None = 6 (відключення протоколювання) [21].

Коли вказаний рівень протоколювання, логування увімкнене для повідомлень вказаного рівня і вище. За замовчуванням рівень протоколювання встановлений на інформаційний (Information). Рівень логування можна вказувати окремо для кожного провайдера протоколювання, для певної категорії чи для всіх провайдерів і категорій.

Інформація про роботу веб-застосунку «Система управління школою» протоколюється у файл засобами бібліотеки Serilog. У звітах відображуються інформаційні повідомлення, попередження та помилки.

Для інтеграції у веб-застосунок бібліотеки Serilog необхідно скористатись менеджером NuGet Packet Manager чи командним рядком dotnet CLI (dotnet add package Serilog). Для роботи необхідні наступні компоненти:

– Serilog – основний пакет протоколювання;
– Serilog.AspNetCore – модуль для інтеграції з ASP.NET Core та протоколювання повідомлень цього фреймворку;
– Serilog.Settings.Configuration – за допомогою цього модулю конфігурація протоколювання може відбуватись через файл «appsettings.json»;
– Serilog.Sinks.Console – модуль для виводу протоколу у консоль/на термінал;
– Serilog.Sinks.File – модуль для виводу протоколу у файл у вигляді тексту чи у форматі JSON;
– Serilog.Sinks.Debug – модуль для виводу протоколу у вікно налагодження.

Підключення бібліотеки Serilog відбувається під час ініціалізації застосунку у файлі «Program.cs». Статичний клас для протоколювання декларується і конфігурується у класі «Startup» при запускові веб-застосунку.

Створювач протоколів передається у клас засобами впровадження залежностей (dependency injection) (див. рис. 2.2) або статично.

Протоколювання викликається або через екземпляр інтерфейсу ILogger (рис. 2.24) або через статичну властивість класу Serilog (рис. 2.25).

```
_logger.LogInformation("Teacher {1}", teacher);
var grade = _context.Grades
                .Where(g => g.Id == curriculum.GradeId)
                .SingleOrDefault();
```

Рис. 2.24. Протоколювання через екземпляр ILogger

```
Log.Information("Користувач створив новий урок - день: {0}, урок№: {1}",
    newLesson.Day, newLesson.Slot);
```

Рис. 2.25. Протоколювання через статичну властивість

Протоколювання зберігається у файл на сервері.

Для доступу до протоколів на сервері Azure необхідно скористатись Web SSH консоллю (рис. 2.26).

Рис. 2.26. Web SSH консоль на порталі Azure

2.6. Адаптивний дизайн

Адаптивний дизайн – це підхід до розробки веб-сторінок, що дає змогу однаково добре відображати сторінки на різних пристроях з різними розмірами вікон та екранів починаючи з мінімального і до максимального розмірів. Сайт, розроблений з виконанням вимог адаптивного дизайну, підлаштовує розташування елементів сторінки використовуючи гнучку пропорційну сітку, адаптивні зображення та медіазапити CSS3 [22].

Елементи адаптивного сайту вимірюються у відносних величинах та відсотках. Малюнки підлаштовують свій розмір відповідно до оточуючого контейнеру. Завдяки медіа запитам можливе використання різних CSS стилів для пристроїв з різними характеристиками.

Адаптивний дизайн характеризується такими ознаками:

- сторінка, створена із дотриманням вимог адаптивного дизайну, виглядає однаково добре на будь-якому пристрої;
- адаптивний дизайн використовує лише мову HTML та CSS;
- адаптивний дизайн – це не програма чи JavaScript.

На присторях з маленьким екраном не слід видаляти чи приховувати частину змісту сторінки. Необхідно пристосовувати зміст сторінки до різних форматів екрану [23] (рис. 2.27). Для зміни розміру, приховування, зменшення, збільшення, зміни положення елементів веб-сторінки використовуються HTML та CSS.

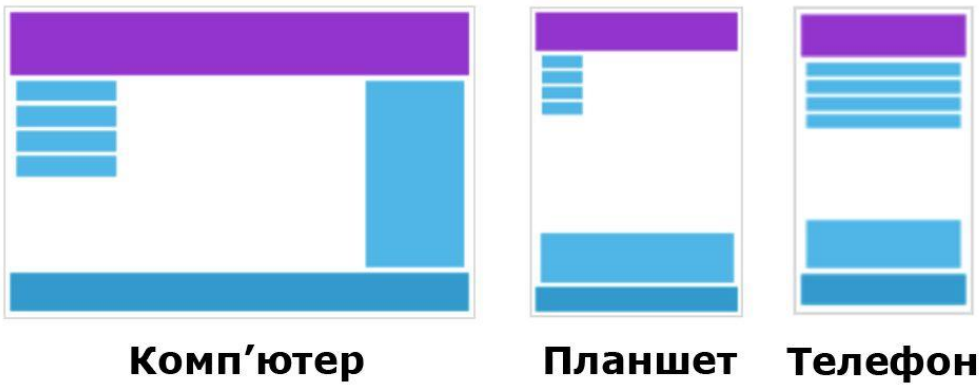

Рис. 2.27. Адаптивний дизайн на різних присторях

У застосунку «Система управління школою» інтегрований Bootstrap 5, який дозволяє використовувати готові CSS шаблони для створення адаптивних сторінок для пристроїв з різними розмірами екрану – маленький, середній, великий, дуже великий (sm, md, lg, xl):

- адаптивна сітка grid для всіх елементів сторінки: .row, .col{-sm|-md|-lg|-xl}-*, .row-cols-*;
- адаптивні таблиці: .table-responsive{-sm|-md|-lg|-xl};
- адаптивні малюнки: .img-fluid (адаптивне зображення), що еквівалентно .h-auto (автоматична висота) та .mw-100 (ширина на весь контейнер);
- адаптивне меню: .navbar-toggler, .navbar-collapse, .navbar-expand{-sm|-md|-lg|-xl};
- адаптивна ширина сторінки: .container{-sm|-md|-lg|-xl}.

У застосунку «Система управління школою» елементи сторінки змінюють своє розташування в залежності від розміру екрану пристрою. На великих дисплеях навігаційне меню відображається повністю, наповнення сторінки розташовується по всій доступній ширині екрану. Навпаки, на вузьких екранах мобільних пристроїв, навігаційне меню сховане за кнопкою, а елементи сторінки розташовуються зверху вниз, один елемент під іншим (рис. 2.28).

При використанні адаптивних таблиць з’являється полоса прокрутки на маленьких дисплеях. Можливе налаштування, при яких розмірах екрану з’являтиметься полоса прокрутки: лише на маленьких екранах (.table-responsive-sm), середніх (-md), великих (-lg) чи дуже великих (-xl).

Адаптивні зображення займають всю ширину наданого їм контейнеру, але не заходять за межі країв екрану по вертикалі, масштабують свій розмір в залежності від розмірів екрану пристрою. Прикладом адаптивного зображення у веб-застосунку «Система управління школою» є слайд-шоу із зображень на головній сторінці додатку: воно змінює свою ширину в залежності від ширини екрану.

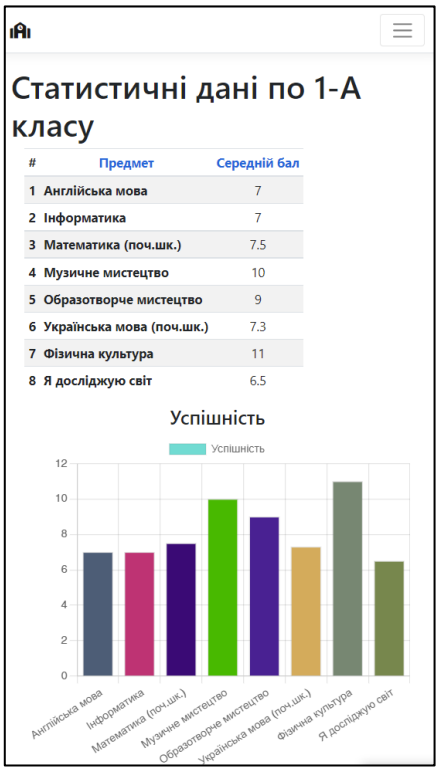

Рис. 2.28. Сторінка статистики на мобільному пристрої

2.7. Безперервна інтеграція та безперервне розгортання (CI/CD)

CI/CD – це спосіб часто доставляти клієнтам програмні продукти. Це досягається автоматизацією усіх етапів розробки програмного забезпечення. Головними поняттями CI/CD є безперервна інтеграція, безперервна доставка та безперервне розгортання, які спрямовані на інтеграцію нового коду.

CI/CD запроваджує автоматизацію та моніторинг на всіх стадіях життєвого циклу програми, від інтеграції і фаз тестування до доставки та розгортання. Разом ці етапи утворюють «конвеєр CI/CD».

Безперервна інтеграція (CI) є процесом автоматизації для розробників. Успішна CI означає, що зміни у коді регулярно компілюються, тестуються і інтегруються у спільному репозиторії. CI покликана вирішити проблему великої кількості гілок у програмі, які одночасно розробляються і можуть конфліктувати одна з одною.

CD може мати значення «безперервна доставка» (continuous delivery) чи «безперервне розгортання» (continuous deployment). Це пов’язані поняття, які

взаємозамінні при позначенні автоматизації пізніх стадій розробки програмного забезпечення. Вони інколи вживаються окремо для позначення міри автоматизації.

Безперервна доставка означає, що зміни, які вносить розробник у програмне забезпечення, автоматично перевіряються на помилки і завантажуються у сховище (репозиторій), наприклад, на GitHub. З репозиторію оперативні команди розгорають програму у виробниче середовище. Це відповідь на проблему поганої прозорості та зворотного зв'язку між командами розробників та бізнес підрозділами. Безперервна доставка має на меті мінімізувати зусилля для розгортання нового коду.

Безперервне розгортання може означати автоматичне впровадження змін, внесених розробником у програмне забезпечення, з репозиторію у виробництво, де програмою можуть користуватись клієнти. Це полегшує роботу оперативних команд, прискорює їх роботу, автоматизуючи повільні ручні процеси. Безперервне розгортання – це наступна після доставки ланка у конвеєрі створення програмного забезпечення [24] (рис. 2.29).

Одне з рішень, яке забезпечує модель CI/CD і яке використовується у веб-застосунку «Система управління школою» - це GitHub Actions.

GitHub Actions – це платформа безперервної інтеграції та безперервної доставки, яка дозволяє автоматизувати процес компіляції, тестування і розгортання. Користувач може створювати робочі процеси, які компілюють і тестують кожен запит на зміну коду (pull request) або розгортають затверджені зміни (merged pull requests) у виробництво.

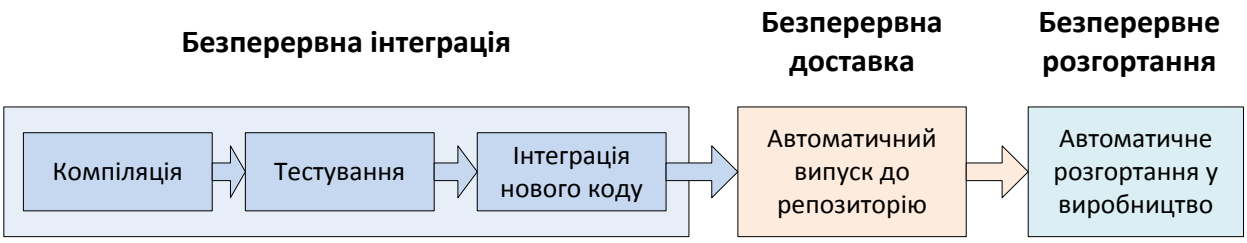

Рис. 2.29. Конвеєр безперервної розробки

Робочий процес GitHub Actions це автоматичний процес, який запускає одне або декілька завдань. Він конфігурується сценарієм, написаним на мові YAML. Файл зі сценарієм повинен бути поміщений у репозиторій разом з проєктом у теці «.github/workflows». Виконання YAML-файлу здійснюється внаслідок події у репозиторії, вручну чи за розкладом. Кожен репозиторій може мати декілька робочих процесів, кожен з яких виконує свій набір кроків. Один процес може компілювати і тестувати запит на зміну коду, інший розгортати застосунок після нових релізів, а третій додавати теги кожного разу, як хтось піднімає питання по коду. Робочі процеси можуть посилатись один на одного.

Подія – це певна активність у репозиторії, яка служить спусковим гачком (тригером) для виконання робочого процесу. Джерелом подій можуть бути портал GitHub, коли хтось створює пропозицію на зміну коду, відкриває питання або вносить зміни у репозиторій. Робочі процеси можна запускати за графіком, за допомогою REST API чи вручну.

Завдання – це набір кроків, за які відповідає один виконавець. Кожен крок це або сценарій командної оболонки або певна дія. Кроки виконуються строго послідовно і кожен наступний залежить від попереднього. Так як кроки запускаються одним виконавцем, вони мають доступ до спільних даних. Наприклад, один крок може компілювати програму, другий тестувати скомпільований код (рис. 2.30).

Можна вказувати залежності одних завдань від інших. За замовчуванням кожне завдання незалежне і виконується паралельно до інших. Якщо завдання залежить від іншого, то воно буде запущене тільки після його виконання. Наприклад, можна мати декілька завдань на компіляцію для різних архітектур, які не будуть залежними одне від одного, та інше залежне від них завдання, яке повинно упакувати результати компіляції. Завдання на компіляцію будуть виконуватись паралельно і після успішного їх завершення, стартує завдання з упаковки.

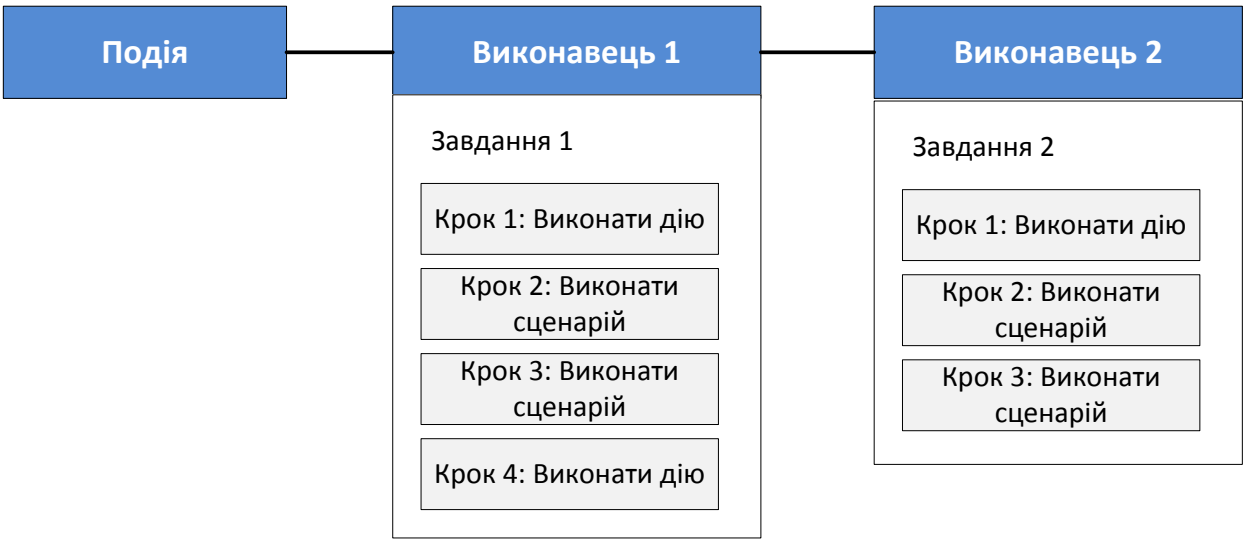

Рис. 2.30. Робочі процеси GitHub Actions

Дія – це виготовлений для платформи GitHub Action застосунок, який виконує складне, але часто повторюване завдання. Дії використовують щоб зменшити кількість повторюваного коду для опису робочих процесів. Дія може завантажити ваш репозиторій з GitHub, налаштувати потрібні інструменти для середовища компіляції, налаштувати автентифікацію до хмарного провайдера. Можна створювати свої дії, а можна завантажувати готові з GitHub маркету.

Виконавець – це сервер, який виконує користувацькі робочі процеси, коли вони активуються. Кожен виконавець може виконувати лише одне завдання. GitHub надає виконавців, що працюють у середовищі Ubuntu Linux, Microsoft Windows та macOS. Кожен робочий процес виконується на чистій, щойно створеній віртуальній машині. Якщо користувачу потрібна інша операційна система чи специфічне обладнання, можна розмістити виконавців на власних обчислювальних потужностях [25].

За допомогою GitHub Actions веб-застосунок «Система управління школою» компілюється, тестується та розгортається на хмарний сервіс Microsoft Azure.

Azure використовує потужні віртуальні машини у датацентрах Microsoft і пропонує сотні сервісів, серед яких:

– віртуальні машини Linux, Windows;

– платформа для розміщення веб-застосунків;
– сервери бази даних (Cosmos DB, NoSQL, Azure SQL).

Для автоматичного розгортування на сервери Microsoft Azure спочатку необхідно створити обліковий запис Microsoft. Новий користувач може безкоштовно використовувати усі сервіси Azure впродовж 30 днів.

Після створення облікового запису необхідно створити ресурсні групи для веб-застосунку та бази даних. Потім створюється SQL Server та база даних. Окремо створюється веб-застосунок, куди GitHub Actions буде розгортати код. Після створення веб-застосунку на його сторінці необхідно отримати профіль для розгортання. Налаштування підключення (connection string) до SQL Server на Azure необхідно скопіювати і додати у власний веб-застосунок у файл «appsettings.json». У скопійовані налаштування необхідно вписати пароль до бази даних, який вказувався під час її створення.

Для того, щоб під час розробки продовжувати використовувати не хмарну, а локальну базу даних SQL, необхідно у методі «ConfigureServices» файлу «Startup.cs» прописати умову використання різних серверів у різних середовищах розробки. Azure має за замовчуванням середу розробки «production», а локальний комп'ютер розробника – середу «development».

Наступним кроком у налаштуванні безперервного розгортання є створення на потралі GitHub у розділі «налаштування» репозиторію так званого «секрету». У «секрет» необхідно скопіювати інформацію з профілю розгортання, який завантажується з сайту Azure.

Після кожного оновлення застосунку на порталі GitHub автоматично виконуються дії, записані у файлі-скрипті (рис. 2.31). В цьому файлі прописані відновлення залежностей, побудова застосунку, його тестування, публікація та розгортання на Azure. Слід зауважити, що якщо у тестах присутня кирилиця, то файли тестів слід зберігати у кодуванні UTF-8, інакше тести не завершаться успішно, і як наслідок, розгортання не відбудеться.

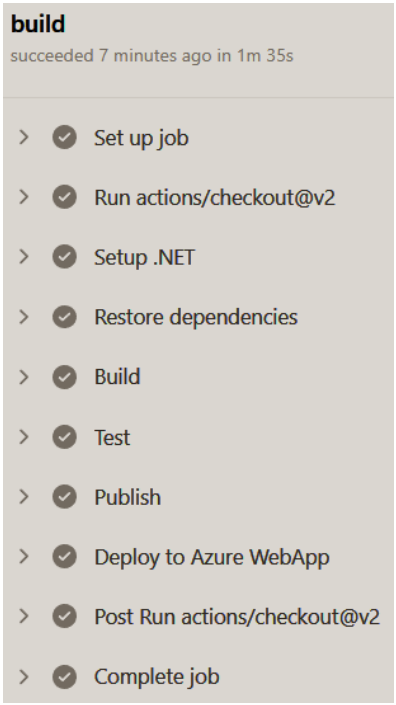

Рис. 2.31. Виконання завдань GitHub Actions

Після успішного завершення всіх дій, у розділі «Actions» на порталі GitHub можна бачити статус виконання дій. Також на цій сторінці можна скопіювати HTML код статусу проєкту для розміщення, наприклад, у README.md файлі, який відображається на головній сторінці GitHub проєкту (рис. 2.32).

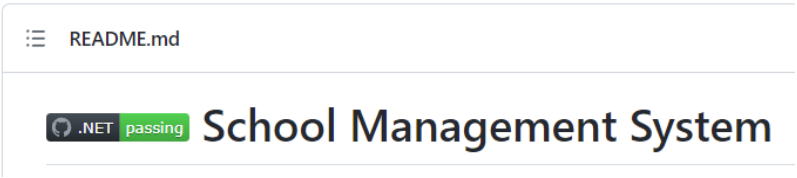

Рис. 2.32. Індикатор здоров’я проєкту на GitHub

Висновок

Даний веб-застосунок побудований на технологіях та екосистемі компанії Microsoft. Застосунок працює на фреймворку ASP.NET Core, перевагами якого є крос-платформеність, відкритість коду, швидкодія, уніфікована система, легкість впровадження модульних тестів, інтеграція з клієнтськими фреймворками Angular, React, Bootstrap.

Фреймворк ASP.NET Core було повністю переписано і він об’єднав у собі продукти, які раніше розроблялись окремо. ASP.NET Core дозволяє будувати веб-застосунки і сервіси, застосунки для інтернету речей, мобільні бекенди, вести розробку на Windows, macOS та Linux.

«Система управління школою» використовує шаблон Model-View-ViewModel та Razor сторінки, які прийшли на зміну моделі MVC і є рекомендованими Microsoft для нових проєктів. Razor сторінки містять лише те, що необхідно для їхньої роботи. Виконується принцип єдиної відповідальності.

Веб-застосунок здійснює свою роботу на основі даних, що містяться у базі даних SQL Server та Azure SQL. Зв'язок між базою даних та застосунком здійснюється через об’єктно-реляційний перетворювач Entity Framework (EF) Core, кросплатформену версію популярної технології доступу до даних з відкритим кодом.

У EF Core доступ до даних відбувається за допомогою моделі. Модель складається з класів-сутностей та об’єкту «контекст», який надає доступ до інформації у базі даних і дозволяє зберігати у ній дані.

Механізм міграцій допомагає підтримувати базу даних у актуальному стані. Маючи модель, за допомогою міграцій можна створити базу даних. Коли змінюється модель, міграція допомагає автоматично змінити базу даних так, щоб вона відповідала моделі. Запити до бази даних здійснюються за допомогою мови запитів LINQ (Language Integrated Query).

Перевірка введених користувачем даних здійснюється як у клієнті, так і на сервері. Клієнтська перевірка – перший етап перевірки введених даних. У веб-застосунку для клієнтської перевірки використовуються атрибути анотації, непомітна валідація jQuery, теги-помічники. Другий етап перевірки введених даних – перевірка на сервері засобами фреймворку ASP.NET Core та бібліотеки FluentValidation. Перевірка моделі на помилки відбувається після створення чи редагування сутностей.

Інформація про роботу застосунку протоколюється у файл на сервері засобами бібліотеки Serilog. У звітах відображуються інформаційні повідомлення, попередження та помилки.

Веб-застосунок розроблено з урахуванням вимог адаптивного дизайну. Адаптивний дизайн – це підхід до розробки веб-сторінок, що дає змогу однаково добре відображати сторінки на різних пристроях з різними розмірами вікон та екранів. Сайт, розроблений з виконанням вимог адаптивного дизайну, підлаштовує розташування елементів сторінки використовуючи гнучку пропорційну сітку. Сторінка, створена із дотриманням вимог адаптивного дизайну, виглядає однаково добре на будь-якому пристрої. У застосунку «Система управління школою» інтегрований Bootstrap 5, який дозволяє використовувати готові CSS шаблони для створення адаптивних сторінок для пристроїв з різними розмірами екрану – маленький, середній, великий, дуже великий. У застосунку використовуються адаптивна сітка, таблиці, меню, малюнки.

У веб-застосунку застосовується шаблон неперервної інтеграції та неперервного розгортання CI/CD. Застосунок автоматично тестується та розгортається на хмарні сервери Azure після кожного оновлення коду у репозиторії засобами GitHub Actions.

РОЗДІЛ 3
ПРОЄКТУВАННЯ ТАБЛИЦЬ БАЗИ ДАНИХ

Для зв’язку з базою даних SQL Server у веб-застосунку «Система управління школою» використовується Entity Framework Core, що створює на основі класів-сутностей таблиці бази даних. Кожній таблиці відповідає певний клас. На рисунку нижче можна побачити діаграму основних класів (рис. 3.1).

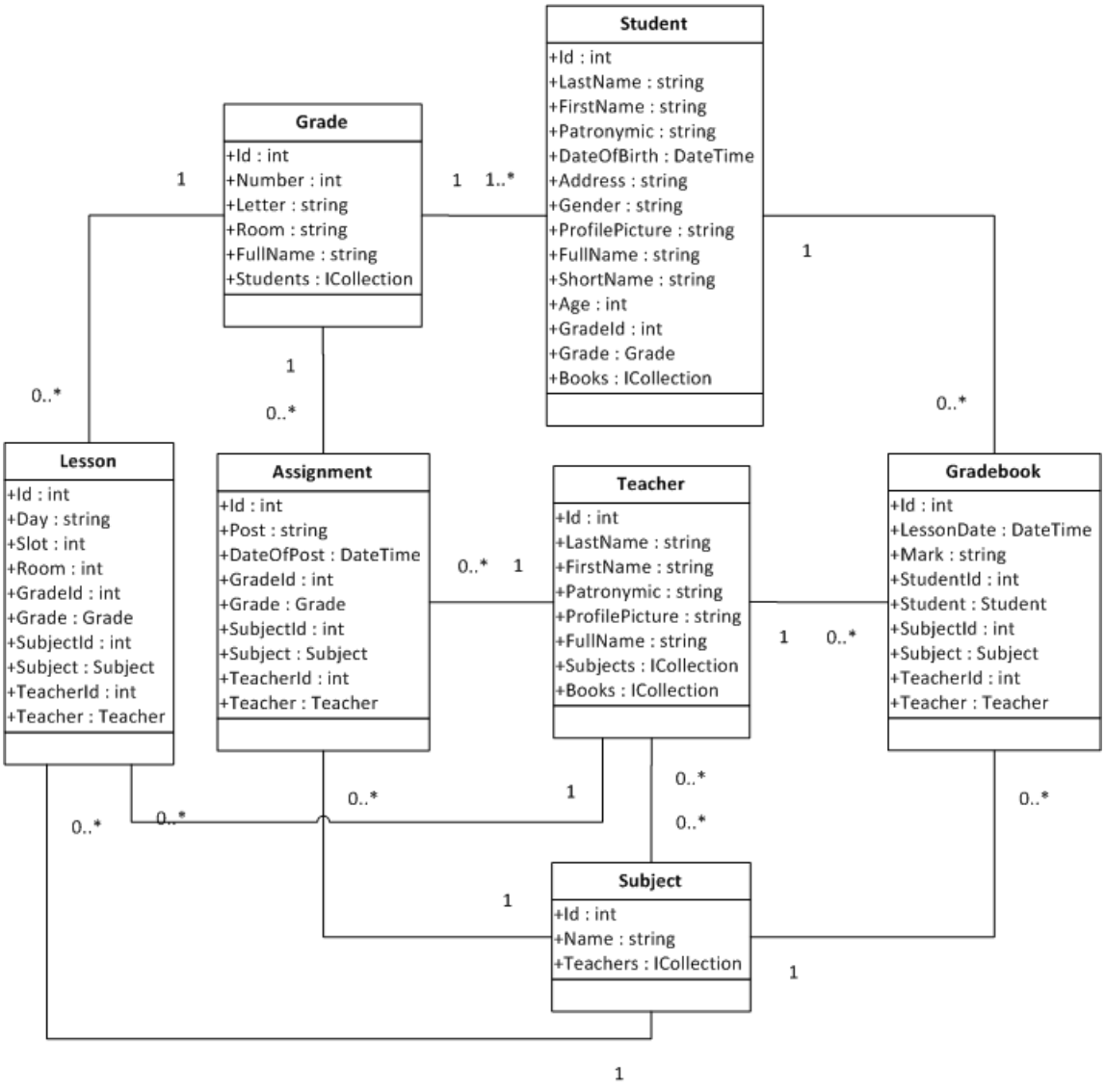

Рис. 3.1. UML діаграма класів-сутностей

3.1. Таблиця «Користувачі»

При створенні проєкту ASP.NET Core із автентифікацією створюються 7 таблиць у базі даних, що слугують для керування процесом реєстрації, входу

користувачів, зміни паролю, автентифікації через інші сервіси, авторизації, керування політиками (рис. 3.2).

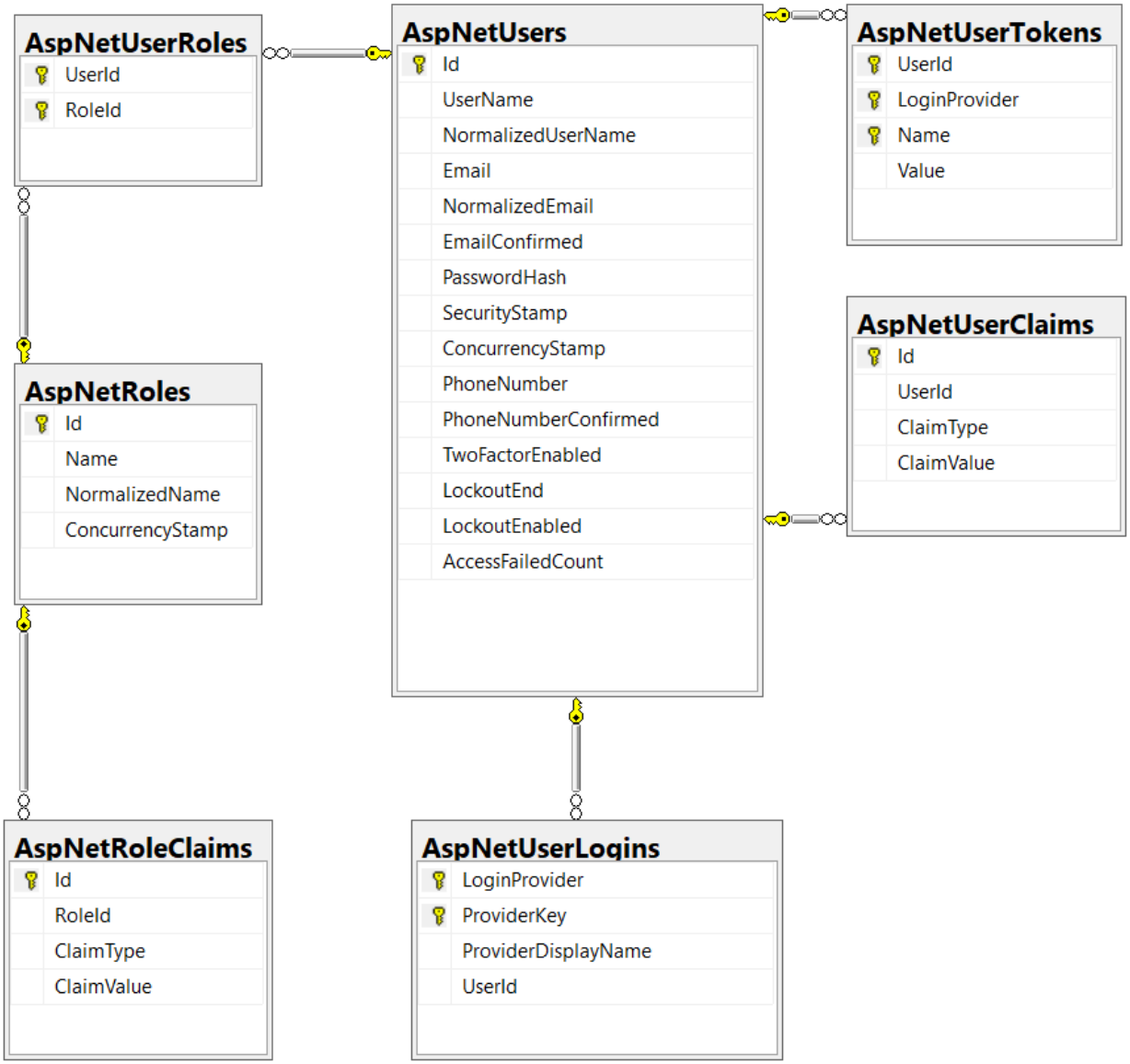

Рис. 3.2. Діаграма таблиць автентифікації

3.2. Таблиці «Класи» та «Учні»

Таблиці «Класи» та «Учні» представляють собою відношення один-до-багатьох та описують приналежність учнів до певного класу (рис. 3.3).

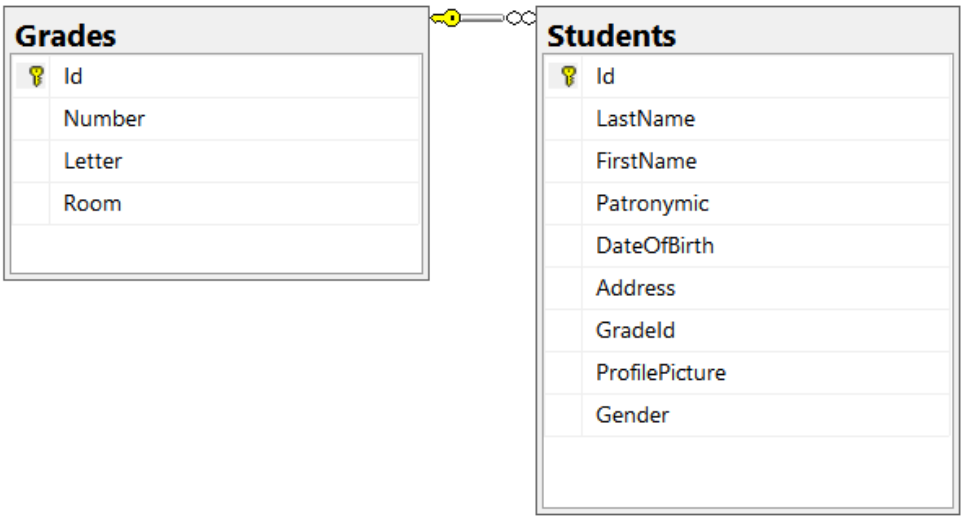

Рис. 3.3. Діаграма таблиць «Класи» та «Учні»

3.3. Таблиці «Книги» та читачі

Відношення класу «Книга» до читачів («Учні» та «Вчителі») є відношенням багато-до-багатьох. Для створення цього відношення використовуються зв’язуючі таблиці «BookTeacher» та «BookStudent» (рис. 3.4). Зв’язуючі таблиці створюються автоматично об’єктно-реляційним перетворювачем Entity Framework Core, їм не відповідають існуючі сутності.

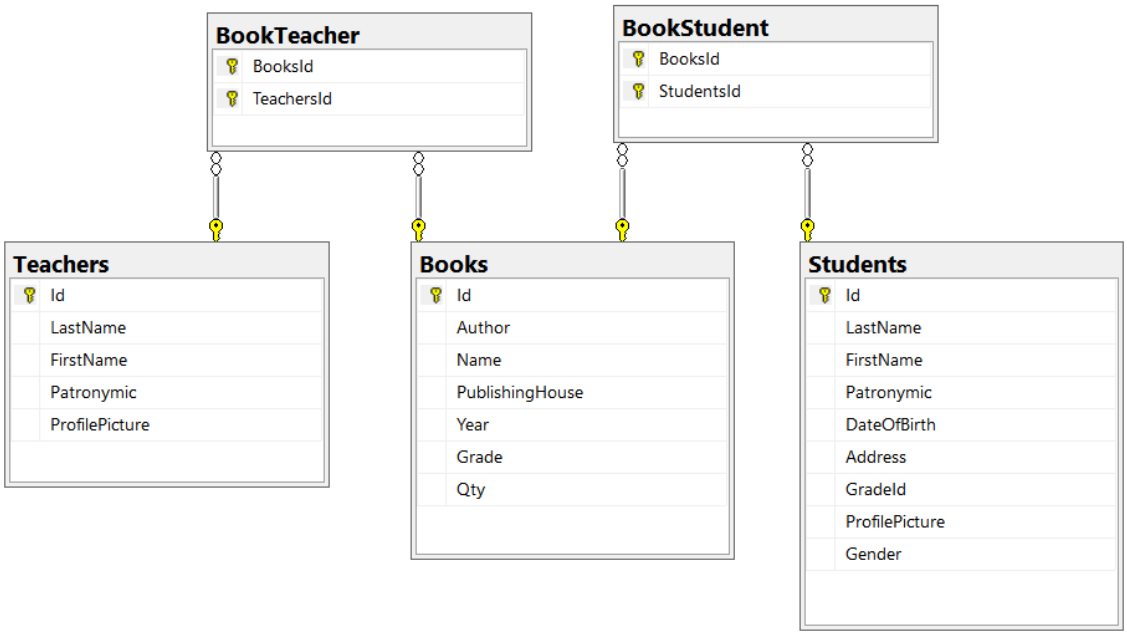

Рис. 3.4. Діаграма таблиць «Книги» та читачів

3.4. Таблиці «Вчителі» та «Предмети»

Відношення сутностей вчителі та предмети є відношенням багато-до-багатьох. Для з’єднання використовується зв’язуюча таблиця «SubjectTeacher», яка автоматично створюється об’єктно-реляційним перетворювачем Entity Framework Core (рис. 3.5).

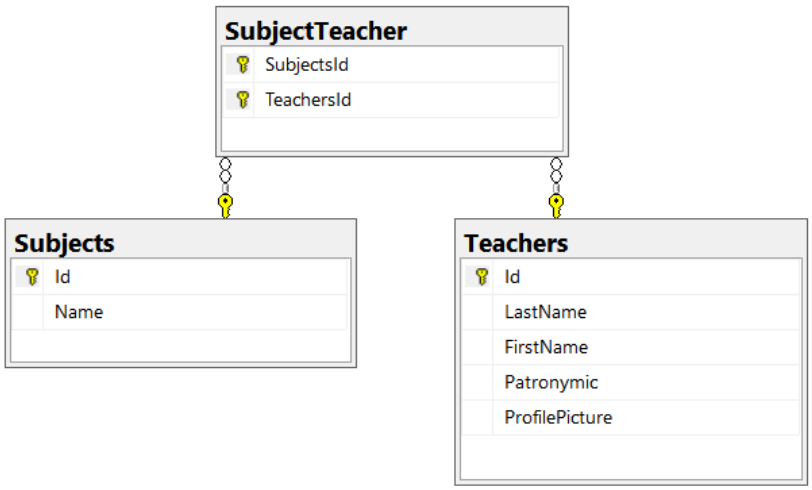

Рис. 3.5. Діаграма таблиць «Вчителі» та «Предмети»

3.5. Таблиця «Навчальний план»

Навчальний план представлений таблицею «Curricula», що містить кількість годин у кожному класі по кожному предмету з вказанням вчителя (рис. 3.6).

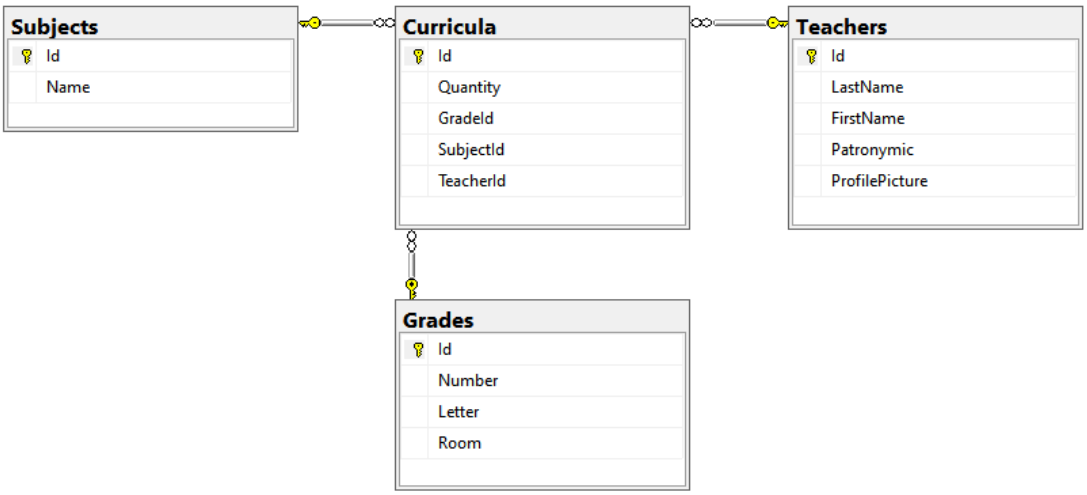

Рис. 3.6. Діаграма зв’язків таблиці «Навчальний план»

3.6. Таблиця «Урок»

Таблиця «Lessons» являє собою представлення сутності «Урок» (рис. 3.7). Урок веде певний вчитель у певному класі з певного предмету.

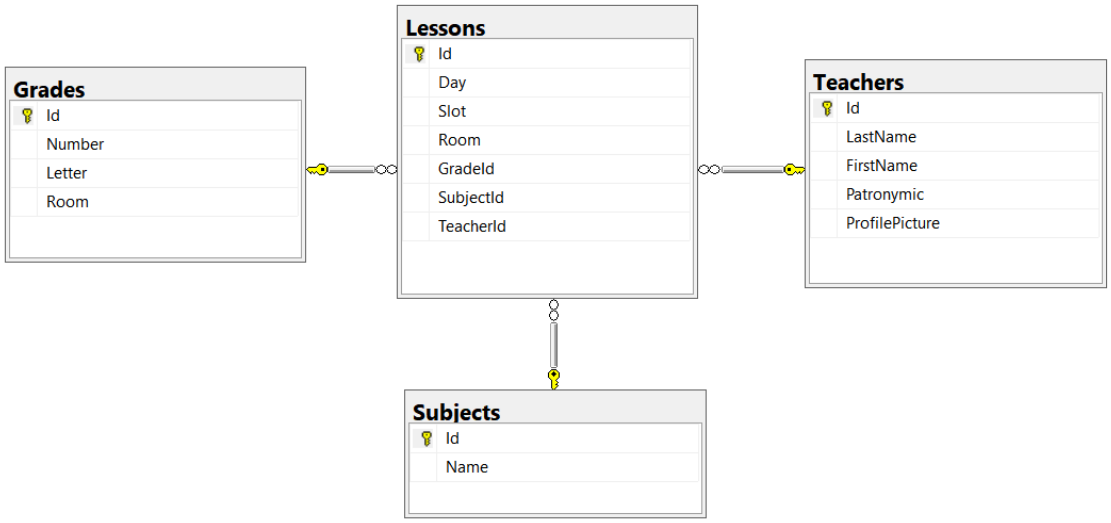

Рис. 3.7. Діаграма зв’язків таблиці «Урок»

3.7. Таблиця «Дистанційне навчання»

Таблиця «Assignments» містить інформацію про завдання з дистанційного навчання (рис. 3.8). Вона містить дані про вчителя, предмет, клас, дату.

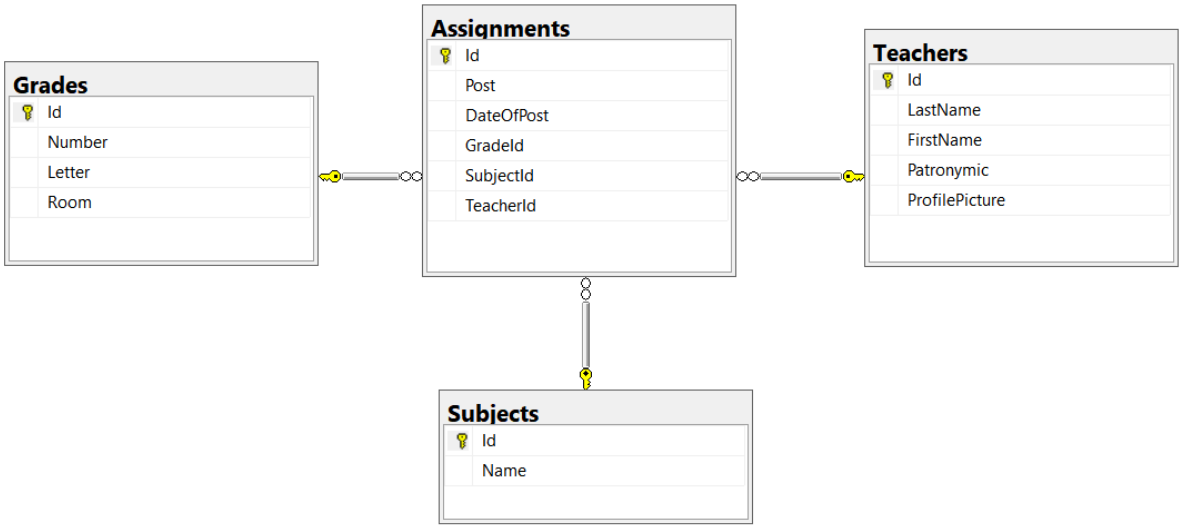

Рис. 3.8. Діаграма зв’язків таблиці «Дистанційне навчання»

3.8. Таблиця «Класний журнал»

Таблиця «Gradebooks» містить дані про навчальні досягнення та пропущені учнями уроки (рис. 3.9).

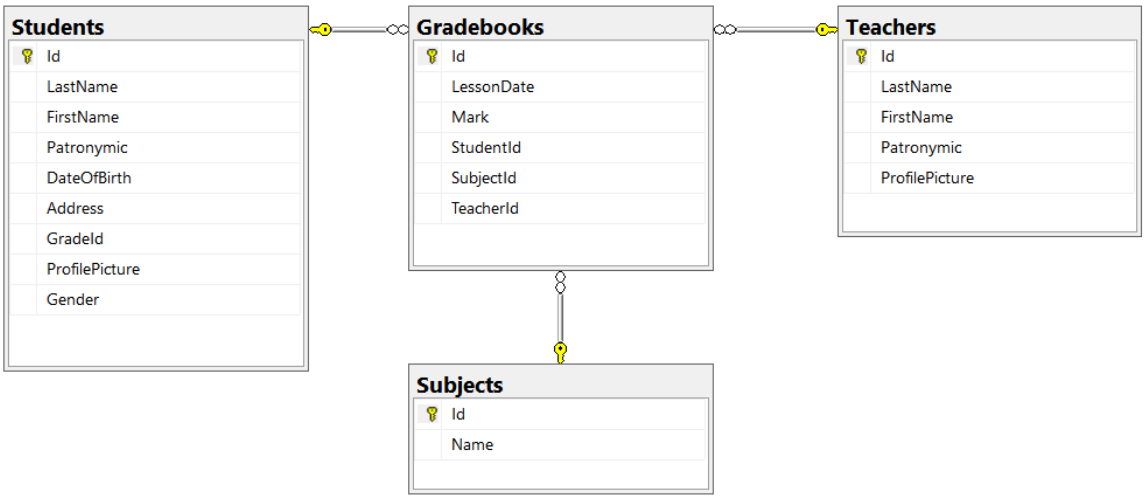

Рис. 3.9. Діаграма зв’язків таблиці «Класний журнал»

3.9. Таблиця «Шкільне майно»

У таблиці «Inventories» міститься інформація про шкільне майно (рис. 3.10). Тут розміщується інформація про інвентарний номер, назву майна, його кількість, вартість, дату взяття на облік, дату списання.

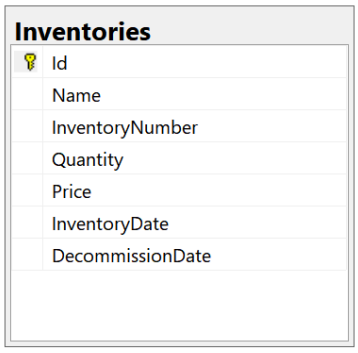

Рис. 3.10. Діаграма таблиці «Шкільне майно»

3.10. Каскадне видалення записів

Коли зв’язок між класами-сутностями Entity Framework Core є обов’язковим (первинний ключ не може бути null), то зовнішні ключі бази даних SQL Server

конфігуруються по правилу каскадного видалення: при видаленні головного запису видаляються усі залежні записи.

Так, при видаленні запису «Клас» видаляються:

- учні класу;
- уроки, що проходять у цьому класі;
- записи навчального плану;
- завдання з дистанційного навчання для цього класу.

При видаленні запису «Предмет» видаляються:

- уроки з цього предмету;
- записи навчального плану;
- оцінки, що виставлені з цього предмету;
- завдання з дистанційного навчання по цьому предмету.

Коли видаляється запис «Вчитель», також видаляються:

- записи навчального плану, в яких фігурує цей вчитель;
- оцінки, виставлені даним вчителем;
- завдання дистанційного навчання, створені цим вчителем;
- дані про відвідання вчителем бібліотеки.

Якщо видалити запис «Учень», то видаляються:

- записи про оцінки учня у журналі;
- дані про відвідання учнем бібліотеки.

При видаленні книги одночасно видаляються:

- інформація про цю книгу у учнів, які мають книгу на руках;
- інформація про книгу у персоналу, який має книгу на руках.

Дані про зовнішні ключі, які мають каскадне видалення, можна отримати з таблиці «REFERENTIAL_CONSTRAINTS» у SQL Server.

Дану інформацію можна отримати за допомогою наступних запитів до SQL Server (рис. 3.11):

```
1  select * from INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS
2  where DELETE_RULE ='CASCADE'
```

Рис. 3.11. Отримання інформації про каскадне видалення записів БД

Висновок

Дані веб-застосунку «Система управління школою» зберігаються у реляційній базі даних SQL Server та Azure SQL. Для зв'язку з базою даних використовується Entity Framework Core, що створює таблиці бази даних на основі класів-сутностей.

Таблиці, які відносяться до автентифікації, створюються автоматично при додаванні відповідного модулю до проєкту. Це сім таблиць, які зберігають дані про імена користувачів, паролі, ролі, зовнішні маркери автентифікації тощо.

Решта таблиць створюються за допомогою міграцій з класів-сутностей.

Таблиці «Класи» та «Учні» представляють собою відношення один-до-багатьох і містять дані про всіх учнів у класах.

Таблиці «Книги» та читачі («Учні» та «Вчителі») пов'язані відношенням багато-до-багатьох. Для створення цього відношення використовуються зв'язуючі таблиці, які створюються автоматично об'єктно-реляційним перетворювачем Entity Framework Core.

Таблиці «Вчителі» та «Предмети» являють відношення багато-до-багатьох і показують, який предмет веде кожен вчитель.

Таблиця «Навчальний план» містить інформацію про те, які предмети, у яких класах і якими вчителями повинні читатись.

Таблиця «Урок» містить інформацію про розклад уроків: день, номер уроку, кабінет, клас, вчителя.

Таблиця «Дистанційне навчання» містить інформацію про завдання з дистанційного навчання.

Таблиця «Класний журнал» містить дані про навчальні досягнення та пропущені учнями уроки.

У таблиці «Шкільне майно» міститься інформація про майно на обліку чи списане майно.

Зовнішні ключі таблиць сконфігуровані по правилу каскадного видалення: при видаленні головного запису видаляються усі залежні записи. Така конфігурація має місце, коли зв’язок між класами-сутностями є обов’язковим.

Веб-застосунок «Система управління школою» налічує 10 таблиць створених вручну, 10 таблиць створених автоматично (автентифікація, зв’язуючі таблиці) та одну службову таблицю, яка містить інформацію про міграції даних.

РОЗДІЛ 4
МОДУЛІ ВЕБ-ЗАСТОСУНКУ

4.1. Огляд веб-застосунку «Система управління школою»

Веб-застосунок «Система управління школою» призначений для керування роботою середнього загальноосвітнього навчального закладу. Він дозволяє автоматизувати процеси менеджменту та обліку, зменшити паперове навантаження та уникнути помилок, пов’язаних з людським фактором.

Користувачі застосунку працюють з певними ролями чи повноваженнями. Різним користувачам доступний різний функціонал.

Роль адміністратора дозволяє здійснювати редагування та налаштування всієї системи: вносити зміни у довідники, формувати навчальний план та генерувати розклад.

Роль вчителя передбачає роботу з класним журналом та модулем дистанційного навчання. Вчитель веде облік відвідування та оцінювання навчальних досягнень учнів у класному журналі. У модулі дистанційного навчання вчитель складає завдання для самостійного опрацювання учнями вдома.

Батьки можуть переглядати розклад уроків та завдання з дистанційного навчання. Для них доступний модуль статистики, де відображаються навчальні досягнення здобувачів освіти.

Користувачі з роллю бібліотекаря працюють з книжковим фондом. Книги можна створювати, редагувати, видаляти. Книги видаються учням та персоналу. Модуль надає список всіх читачів книги або всі книги певного читача. Завдяки бібліотечному модулю можна побачити наявну кількість примірників.

Якщо у веб-застосунок увійти із роллю завгоспа, буде доступний список майна, що стоїть на обліку школи або майна, яке було списане. Майно можна додавати і видаляти, списувати чи ставити на облік. Детальніше ознайомитись з ролями користувачів можна у діаграмі варіантів використання у додатку А.

4.2. Модуль автентифікації

При створенні проєкту ASP.NET Core з типом автентифікації «Individual Accounts» у програму додається модуль «Identity», що відповідає за реєстрацію і вхід користувачів. Це модуль з відкритим кодом.

Модуль «Identity» містить наступні компоненти:

- зміна особистих даних;
- встановлення, зміна, скидання, відновлення забутого паролю (рис. 4.1);
- завантаження, видалення персональних даних;
- підключення, відключення двофакторної автентифікації;
- підключення, скидання, відключення автентифікатора;
- зміна електронної адреси;
- надсилання електронного листа підтвердження;
- управління зовнішніми входами;
- генерація кодів відновлення;
- заборона доступу до ресурсів;
- внесення користувача у тимчасовий список заборонених при численних невдалих спробах автентифікації.

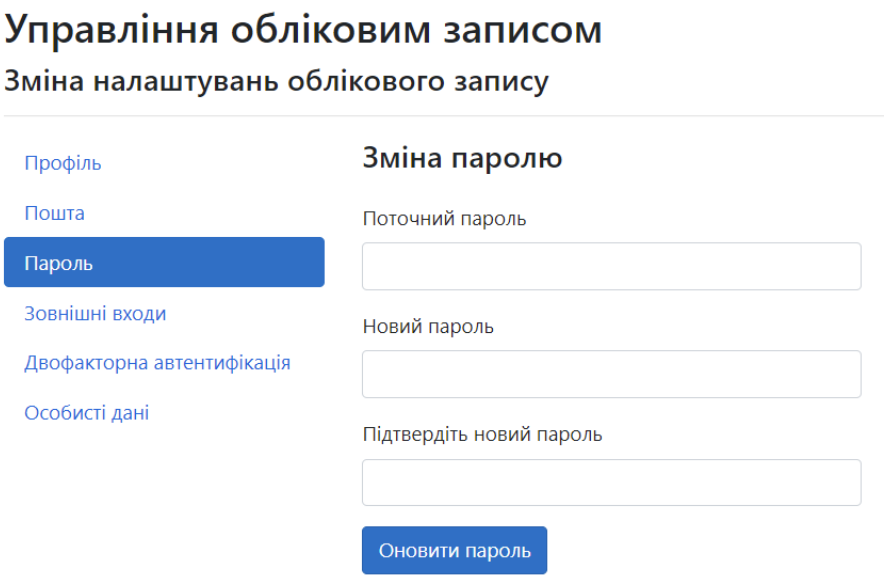

Рис. 4.1. Управління паролями

Модуль «Identity» керує користувачами, паролями, даними користувачів, ролями, маркерами доступу, підтвердженням електронних адрес тощо. Користувачі можуть створити обліковий запис із збереженням паролю у модулі «Identity», або скористатись зовнішнім надавачем послуг. Підтримується вхід через Facebook, Google, Microsoft Account, Twitter [26]. Вхід у застосунок «Система управління школою» здійснюється за допомогою логіну та паролю або облікового запису Google.

Модуль «Identity» зберігає дані про користувачів у базі даних SQL Server. Можливе також використання Azure Table Storage.

4.3. Модуль авторизації

Авторизація відповідає на питання «Що автентифікований користувач може робити?». У ASP.NET Core існує проста авторизація на основі ролей та більш складна, що базується на політиках. Авторизація виражається у вимогах, вона оцінює заяви користувачів (user claims) та порівнює їх з вимогами. Перевірки можуть аналізувати як права користувача, так і ресурс, доступ до якого користувач намагається отримати [27].

Модуль «Identity» містить у собі інструменти авторизації. У даному веб-застосунку використовується авторизація на основі ролей. Доступ до кожної сторінки обмежений списком ролей користувачів. Після реєстрації користувача, адміністратор має надати йому роль, після чого користувачу надаються відповідні повноваження (рис. 4.2).

Веб-застосунок «Система управління школою» має наступні ролі користувачів:

- адміністратор;
- вчитель;
- завгосп;
- бібліотекар;
- батьки.

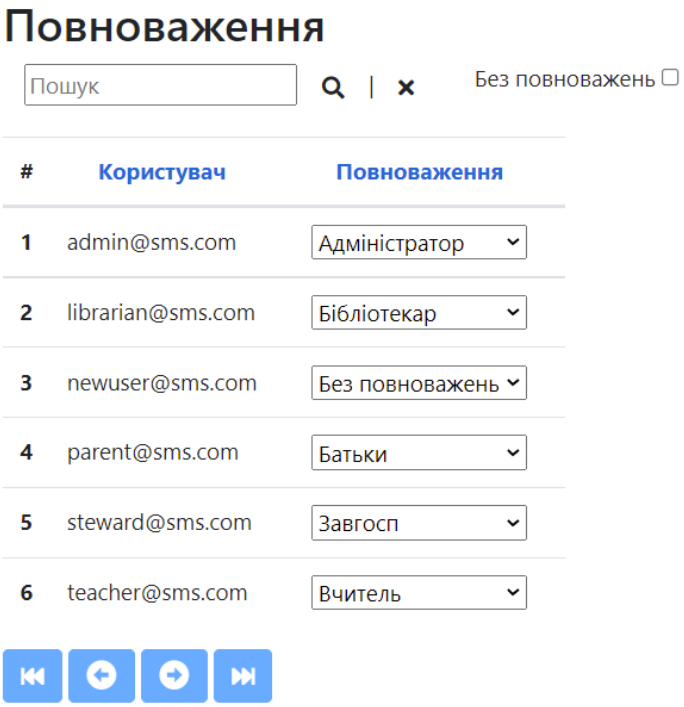

Рис. 4.2. Вікно редагування повноважень

Адміністратор має доступ до будь-якої сторінки та ексклюзивний доступ до наступних ресурсів:

- редагування повноважень;
- складання навчального плану;
- редагування і додавання вчителів;
- редагування і додавання учнів;
- редагування і додавання предметів;
- редагування і додавання класів;
- редагування і генерація розкладу;
- перегляд статистики по класам;
- перегляд статистики по предметам.

Вчитель має права змінювати:

- класний журнал;
- завдання для дистанційного навчання.

Завгосп має такі повноваження:

– редагувати базу даних майна на балансі школи;
– списувати та ставити на облік майно.

Бібліотекар має права:

– створювати, редагувати та видаляти книги;
– видавати та приймати книги у читачів.

Батьки можуть переглядати наступні сторінки:

– навчальні досягнення здобувача освіти;
– розклад уроків для учнів;
– завдання з дистанційного навчання.

Зазвичай, користувач не бачить посилань на сторінки, для перегляду яких у нього недостатньо повноважень. Але навіть якщо у користувача буде посилання на заборонену сторінку, при спробі доступу до неї, він отримає повідомлення про неможливість виконати дію (рис. 4.3).

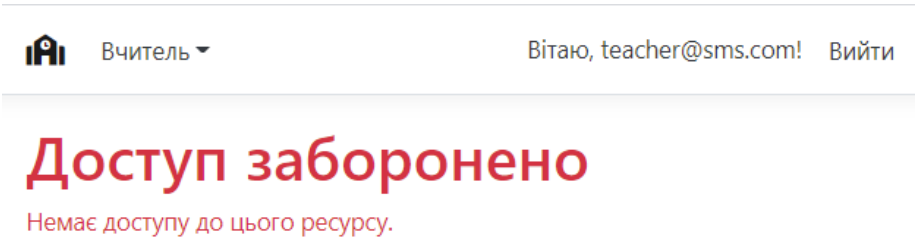

Рис. 4.3. Неавторизований доступ до сторінки

Різні користувачі бачать різне меню. Ті сторінки, які недоступні, не відображаються. Так, якщо користувач зайде з роллю «вчитель», то він не зможе побачити бібліотеку чи сторінки завгоспа. Меню адміністратора найбільш повне.

Крім того, зміст окремих сторінок різний для користувачів з різними ролями. Наприклад, у розкладі уроків для адміністратора є кнопка «Згенерувати розклад» та є кнопки для редагування і створення уроків. Для користувача з роллю «Вчитель» зазначені кнопки відсутні (рис. 4.4).

Адміністратор

Вчитель

Рис. 4.4. Варіанти сторінки «Розклад» для користувачів з різними ролями

Іншим прикладом різного дизайну сторінок є модуль «Дистанційне навчання». Для вчителя у ньому присутній функціонал створення, редагування та видалення завдань. Батьки можуть лише переглядати завдання.

4.4. Модуль «Вчителі»

Сторінка «Вчителі» знаходиться за посиланням «Довідник: вчителі» у меню адміністратора.

На сторінці «Список вчителів» представлені дані по кожному вчителю. Доступне сортування по прізвищу. Для цього необхідно натиснути на назву стовпця з прізвищем. Існує функція пошуку по прізвищу, імені, по батькові (рис. 4.5).

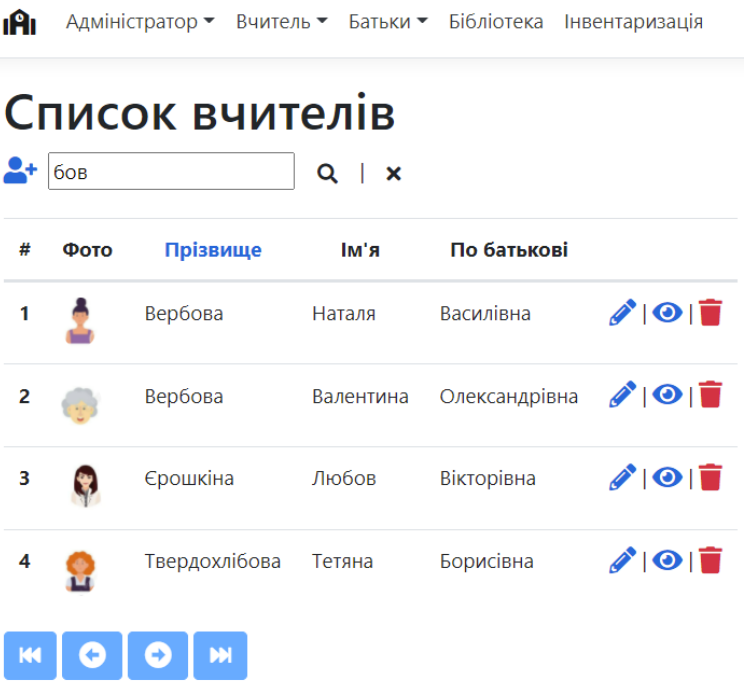

Рис. 4.5. Пошук по списку вчителів

Створення нової картки вчителя відбувається після натискання на кнопку зі знаком «плюс». Поля «Прізвище», «Ім’я», «По батькові» приймають лише букви кирилиці і тире. Перша буква – велика. Максимальна довжина – 50 символів.

Для створення нового вчителя достатньо вказати прізвище, ім’я, по батькові, обрати зі списку множинного вибору предмети, які веде вчитель. Можливо також додати фото.

Для перегляду детальної інформації по вчителю, необхідно у списку вчителів навпроти потрібного запису натиснути на кнопку із зображенням ока. Відкриється вікно «Деталі», де можна переглянути інформацію по обраному вчителю. Внизу вікна є кнопки для повернення назад та для редагування запису.

Редагування даних про вчителя доступне зі сторінок «Список вчителів» та «Деталі». Для відкриття сторінки «Редагувати» необхідно натиснути на кнопку із зображенням олівця у списку навпроти потрібного вчителя.

Для редагування доступні прізвище, ім’я, по батькові. Можна змінити чи додати предмети, які буде викладати вчитель та обрати нове фото.

Щоб видалити запис учителя, необхідно на сторінці зі списком учителів натиснути на червону кнопку із зображенням сміттєвого кошика. Після цього на сторінці підтвердження необхідно натиснути на кнопку «Видалити».

4.5. Модуль «Учні»

Модуль доступний з меню адміністратора за посиланням «Довідник: учні». Список учнів можна сортувати по прізвищу, даті народженню, класу, натиснувши на відповідний заголовок стовпця таблиці. Можливий пошук по прізвищу, імені, по батькові.

Для створення нової картки учня, необхідно натиснути на кнопку зі знаком «плюс» у на сторінці зі списком учнів. Поля «Прізвище», «Ім’я», «По батькові» приймають лише букви кирилиці і тире. Перша буква – велика. Максимальна довжина – 50 символів. Поле «Адреса» має такі обмеження: Максимальна довжина – 50 символів. Дозволені цифри і букви (кирилиця, латиниця), дефіс, пробіл, подвійні та одиничні лапки, крапка, кома.

Зі сторінки «Список учнів» доступний перегляд докладної інформації про учня. Для відкриття сторінки «Деталі» необхідно натиснути на кнопку із зображенням ока навпроти потрібного запису.

Зі сторінки «Деталі» можна повернутись назад до списку учнів натиснувши «Назад», або перейти до редагування запису за посиланням «Редагувати».

Редагування доступне зі сторінок «Список учнів» та «Деталі». Редагувати можна прізвище, ім’я, по батькові, дату народження, адресу, клас, стать, обрати нове фото.

Щоб видалити запис, необхідно у списку учнів натиснути на червону кнопку із зображенням корзини для сміття. Перед остаточним видаленням необхідно підтвердити свій вибір.

4.6. Модуль «Класи»

Сторінка зі списком класів доступна з меню «Адміністратор» за посиланням «Довідник: класи».

Для створення нового класу слугує кнопка зі знаком «плюс». Під час створення класу необхідно вказати цифру і букву нового класу та натиснути на кнопку «Створити». Поле «Кабінет» є необов'язковим для заповнення.

При натисканні на кнопку із зображенням ока навпроти певного запису можливий перегляд детальної інформації по класу. На сторінці «Деталі» відображається назва класу, список учнів (прізвище, ім'я, по батькові, вік). Також на цій сторінці відображається статистика по статі та віку у вигляді кругової діаграми та гістограми (рис. 4.6).

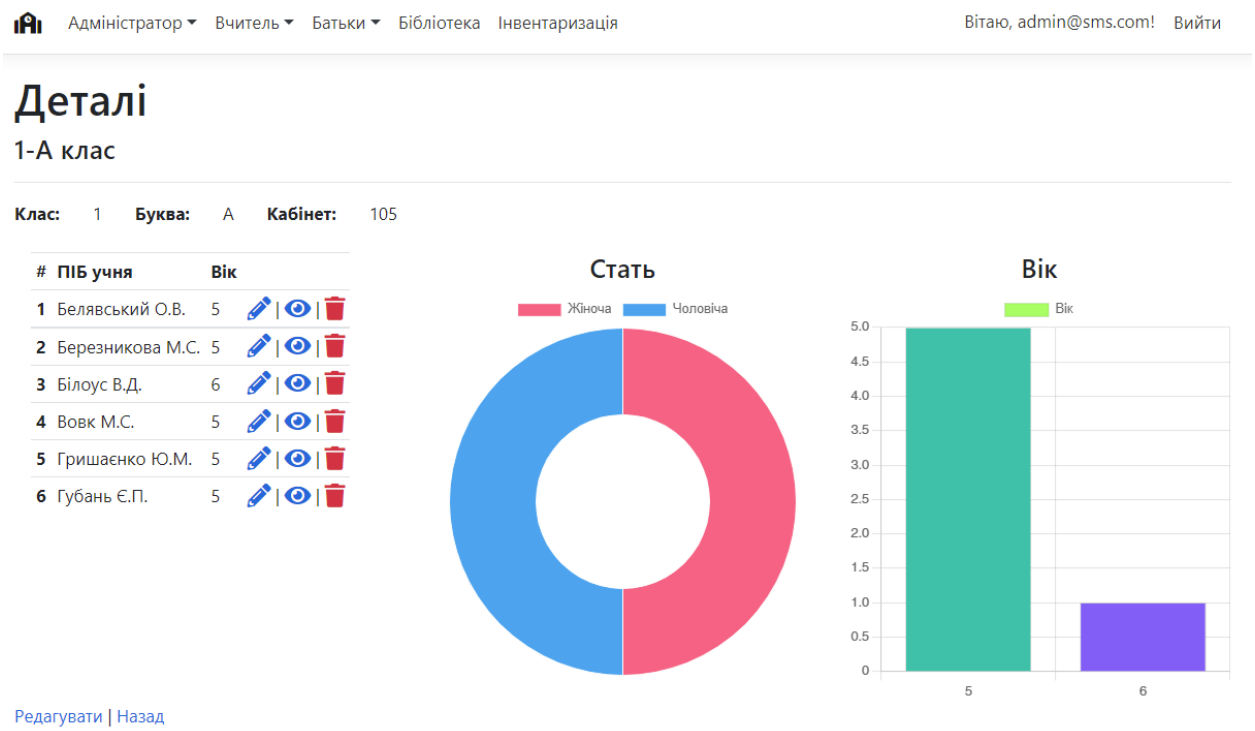

Рис. 4.6. Детальна інформація по класу

Для редагування назви класу необхідно натиснути на кнопку із зображенням олівця навпроти потрібного запису на сторінці «Список класів».

Щоб видалити клас, необхідно натиснути на червону кнопку із зображенням сміттєвої корзини. Перед видаленням необхідно підтвердити свій вибір.

4.7. Модуль «Предмети»

Список предметів можна переглянути з меню адміністратора за посиланням «Довідник: предмети». Можна сортувати та фільтрувати список.

Після натискання на кнопку із знаком «плюс» відкриється вікно створення предмету. Щоб створити предмет, необхідно вказати його назву та одного чи декількох вчителів, які будуть вести предмет.

На сторінку «Деталі» можна потрапити зі сторінки списку предметів, натиснувши на кнопку із зображенням ока. На сторінці можна переглянути вчителів, які ведуть даний предмет.

Редагування предмету доступне зі сторінки «Деталі» та «Список предметів». Кнопка редагування має вигляд олівця. Типовий сценарій редагування включає додавання та видалення вчителів зі списку тих, що викладають предмет, або зміну назви предмету, наприклад, зі «світова література» на «зарубіжна література».

На сторінці «Список предметів» при натисканні на червону кнопку у вигляді сміттєвої корзини навпроти предмету, користувач потрапляє на сторінку підтвердження видалення предмету. На сторінці «Видалити предмет» можна підтвердити видалення або повернутись назад до списку предметів.

4.8. Модуль «Навчальний план»

Модуль містить інформацію про клас, предмет, вчителя, кількість уроків на тиждень.

Для перегляду навчального плану певного класу необхідно обрати з випадаючого списку клас. Знайти потрібний предмет можна за допомогою кнопок «вперед» і «назад» та за допомогою пошукового рядку вверху сторінки.

Створити нове навантаження можна натиснувши кнопку зі знаком «плюс» вверху сторінки. На сторінці навантаження буде вказаний той клас, на сторінці якого користувач перебував. Якщо необхідно, клас можна змінити у випадаючому меню.

Слід зауважити, що не можна обрати предмет, не обравши вчителя, оскільки саме вчитель визначає, з якого списку предметів можливий вибір. За кожним вчителем закріплений певний перелік предметів. За замовчуванням, кількість уроків на тиждень дорівнює одному. Це поле можна редагувати і вказати правильну кількість відповідно до діючих навчальних програм. Кнопка «Назад» повертає користувача до перегляду навчального плану класу, який він обрав перед натисканням кнопки «Нове навантаження».

Для редагування запису, необхідно навпроти нього натиснути на кнопку із зображенням олівця. На сторінці редагування можна зберегти зміни чи повернутись назад.

Кнопка у вигляді ока навпроти певного запису слугує для перегляду деталей. З цієї сторінки можливий перехід до сторінки редагування або повернення назад до списку навантажень класу.

4.9. Модуль «Класний журнал»

Можливість заповнювати класний журнал з’являється після складання розкладу вручну чи генерації автоматично. Класний журнал дає змогу вести оцінювання навчальних досягнень здобувачів освіти, а також відмічати відсутність учнів. Після вибору класу, предмету, місяця та року, відображається класний журнал (рис. 4.7). Дати уроків беруться з розкладу. Якщо у певну дату уроку немає, то дата не відображається. Дані, що введені у журналі, можуть переглядати батьки. Дані про успішність учнів використовуються у модулі «Статистика» для показу успішності по класу і по предмету.

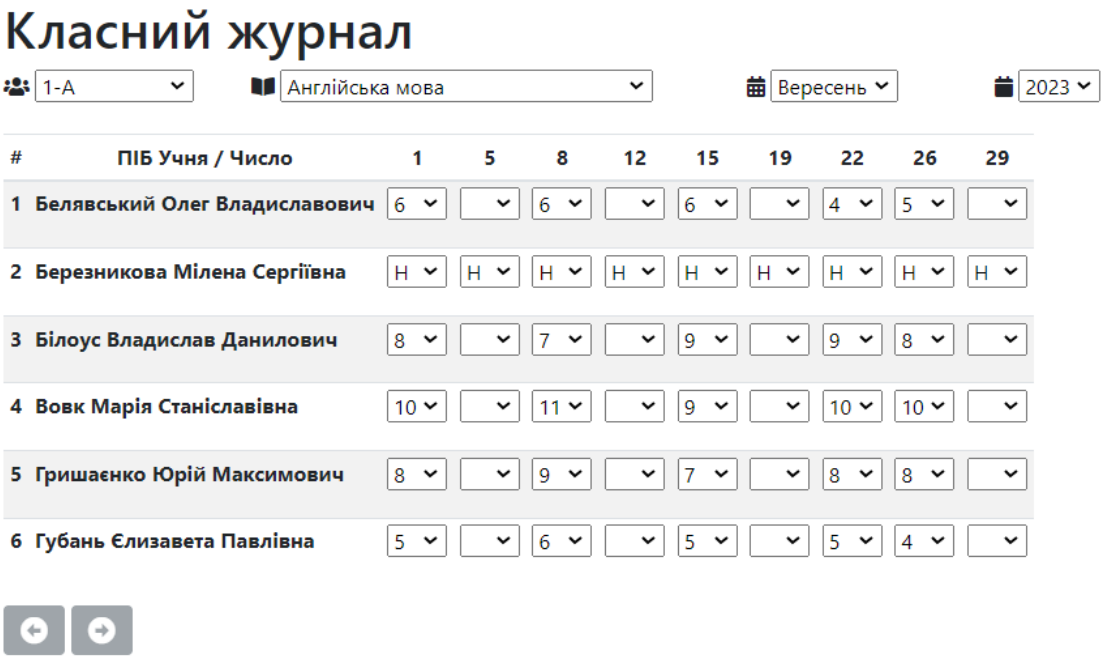

Рис. 4.7. Класний журнал

4.10. Модуль «Навчальні досягнення»

Модуль доступний для перегляду батьками. Відображає оцінки та пропуски уроків, введені у «Класному журналі» вчителями. Модуль також підраховує середній бал і відображає дані у вигляді стовпчикової діаграми. Таблицю досягнень можливо сортувати за предметом чи середнім балом. Для відображення даних слід обрати клас, учня, місяць та рік (рис. 4.8).

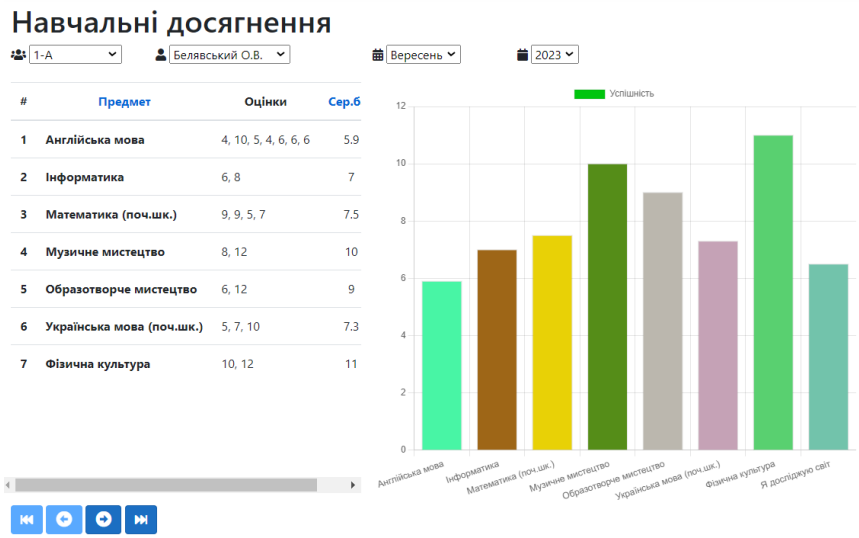

Рис. 4.8. Навчальні досягнення учня

4.11. Модуль «Статистика»

У веб-застосунку «Система управління школою» використана JavaScript бібліотека Chart.js для відображення статистичної інформації у вигляді гістограм та кругових діаграм. Chart.js є безкоштовною бібліотекою з відкритим кодом. Станом на жовтень 2023 року, на сайті www.github.com у бібліотеки налічується 473 розробника, що внесли свій вклад у покращення продукту [28].

Окрім широкого функціоналу, Chart.js має чудову документацію з детальними прикладами, що дозволяє швидко знайти потрібну інформацію та інтегрувати діаграми у свій код.

Chart.js виділяється серед своїх аналогів наступним функціоналом:

- яскраві анімації усіх типів графіків;
- швидкодія при великих обсягах даних;
- змішані типи графіків на одному листі;
- адаптивні гістограми, що підлаштовують розміри під сторінку;
- використання HTML5 Canvas, що підтримується усіма сучасними браузерами;
- відкритий код.

У Chart.js є наступні типи діаграм: лінійна діаграма (line chart), гістограма (bar chart), радарна карта (radar chart), кільцева діаграма (doughnut chart), кругова діаграма (pie chart), полярна карта (polar area chart), бульбашкова діаграма (bubble chart), діаграма розсіювання (scatter chart), діаграма з областями (area chart), змішані типи діаграм (mixed chart). За допомогою розширень/модулів можливо побудувати інші типи діаграм.

Перегляд статистики у веб-застосунку «Система управління школою» доступний користувачам з роллю «Адміністратор». Модуль «Статистика» відображає наступні дані:

- успішність по класам;
- успішність у обраному класі;
- успішність по предметам;

– успішність по обраному предмету (рис. 4.9).

Рис. 4.9. Статистичні дані по обраному предмету

Також статистичні дані містяться у розділі «Довідник: класи» при перегляді інформації про клас (див. рис. 4.6).

4.12. Модуль «Дистанційне навчання»

Модуль слугує для створення, редагування та видалення завдань для учнів, що перебувають на дистанційному навчанні.

Для початку роботи з модулем «Дистанційне навчання» необхідно обрати клас та предмет, інакше інформація не відображається.

Після вибору класу можна обрати предмет, який викладається у цьому класі. У списку є лише ті предмети, які присутні у навчальній програмі для даного класу.

Після вибору предмету з’являється список завдань. Можливо сортувати завдання по даті натиснувши заголовок колонки «Дата». По замовчуванню першими відображаються найновіші завдання.

Список завдань дистанційного навчання також доступний для перегляду батьками. Однак з облікового запису з правами «Батьки» неможливо створювати, редагувати чи видаляти записи, там відсутні відповідні кнопки. Навіть якщо користувач буде знати посилання на сторінку видалення, його прав буде недостатньо, щоб зробити це.

Кнопка у вигляді ока дозволяє переглянути деталі завдання у окремому вікні для зручності. На сторінці «Деталі» присутня кнопка редагування та повернення назад.

Червона кнопка у вигляді корзини дозволяє видалити виконане чи помилково введене завдання.

Кнопка у вигляді олівця у списку завдань дистанційного навчання дозволяє редагувати записи. Для створення і редагування дописів модулю «Дистанційне навчання», використовується WYSIWYG редактор тексту «TinyMCE» (рис. 4.10). TinyMCE – це редактор форматованого тексту, перекладений багатьма мовами. Редактор написаний на JavaScript та має відкритий код.

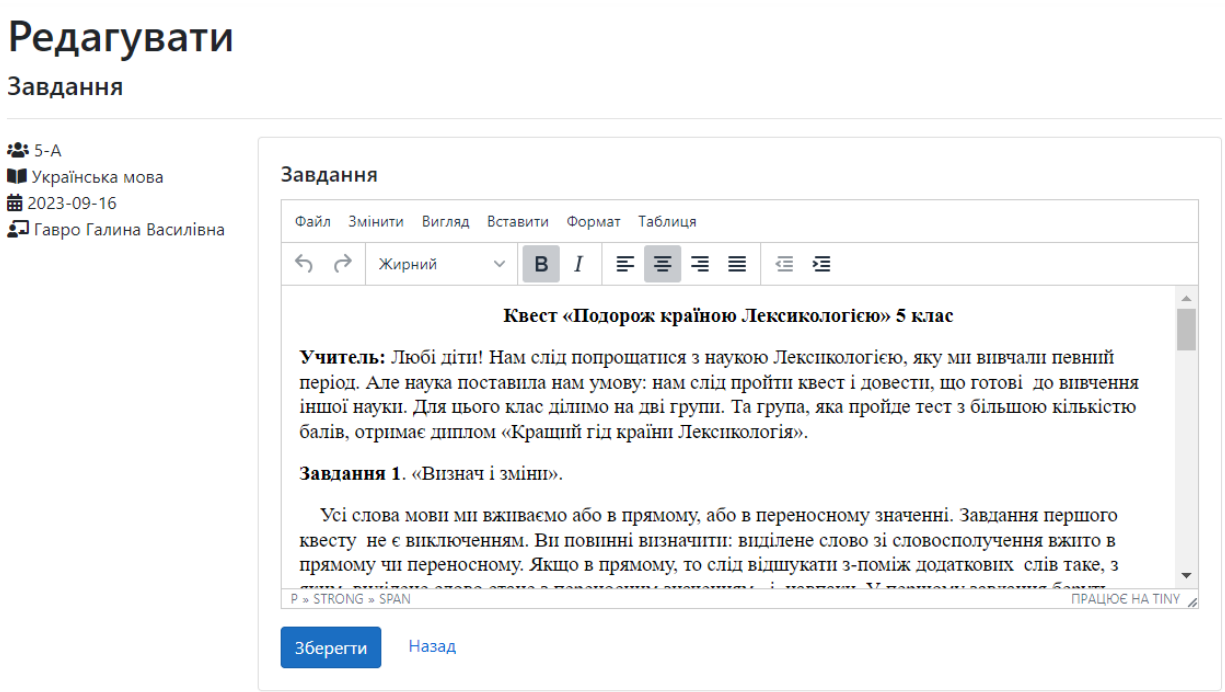

Рис. 4.10. Текстовий редактор «TinyMCE»

Редактор у базовій редакції є безкоштовним. Він легко підлаштовується під потреби шляхом кастомізації. У платній редакції можливе збереження файлів, створених у редакторі, у хмарному сервісі, надається комерційна ліцензія та присутня низка додаткового функціоналу:

- покращена вставка відео, зображень і тексту;
- форматування за зразком;
- покращений редактор таблиць;
- маркер;
- зміна регістру букв;
- перевірка орфографії [29].

Для підключення редактору, необхідно вказати шлях до нього у тезі «script» на веб-сторінці:

Редактор активується, якщо на сторінці є тег <textarea>. Можна вказати також інший селектор, здійснити налаштування мови інтерфейсу та плагінів у тезі <script> директивою «tinymce.init».

TinyMCE інтегрується з такими фреймворками: AngularJS/Angular 5+, Bootstrap, Django, jQuery, Node.js + Express, Rails, React, Swing, Vue, Web Components, WordPress.

Редактор дозволяє редагувати текст наступним чином.

Редагування типу шрифту: напівжирний, курсив, підкреслення, перекреслення, надрядковий символ, підрядковий символ, код.

Редагування форматування: створення заголовків різних рівнів, створення блоків цитування, блоків розділів, керування вирівнюванням (по лівому, правому краю, по центру, рівномірно), обирати висоту проміжків між рядками, збільшувати чи зменшувати відступ абзацу, очищати форматування.

Редагування шрифту: обирати шрифт зі встановлених у системі, обирати розмір шрифту, обирати колір шрифту, обирати колір фону.

Редактор має можливість встановлення плагінів. Існують платні і вільні плагіни. За допомогою плагінів можна вставляти у текст: малюнки, аудіо, відео, посилання, таблиці.

Так як користувач при редагуванні завдання з дистанційного навчання може вставити у вікно редагування будь-який HTML код, для безпечної роботи сайту необхідно захиститись від від міжсайтового скриптингу. Міжсайтовий скриптинг (cross site scripting, XSS) – це тип вразливості безпеки, що дозволяє зловмисникам запроваджувати власні скрипти на веб-сторінки. Для захисту від цієї вразливості у веб-застосунку «Система управління школою» використовується .NET бібліотека «HtmlSanitizer». Вона має відкритий код і використовується для очищення HTML від конструкцій, які можуть призвести до XSS атак. Ця бібліотека також має позитивну побічну дію: вона може захистити від так званого «отруєння тегів»: коли помилка у одному тезі може зіпсувати увесь документ та порушити макет сторінки [30].

4.13. Модуль «Бібліотека»

Модуль «Бібліотека» доступний для перегляду адміністратору та бібліотекареві. Модуль допомагає вести облік наявних книжок, їхню загальну кількість та наявні примірники. Модуль доступний за посиланням «Бібліотека» у навігаційному меню.

Список книг бібліотеки поділений на сторінки для зручності перегляду. У списку відображаються назва книги, для якого класу книга, хто автор книги, видавництво, рік видання. У таблиці можна побачити загальну кількість примірників та наявну кількість у вигляді двох цифр через дріб. З цієї сторінки можливо редагувати, видаляти книги, видавати читачам. Також можна створювати нові книги відповідною кнопкою. Список книг можна сортувати по назві, класу, року видання, кількості. Сортування як низхідне, так і висхідне. Можна фільтрувати список за ключовим словом. Також можна відображати лише книги для певного класу за допомогою випадаючого списку (рис. 4.11).

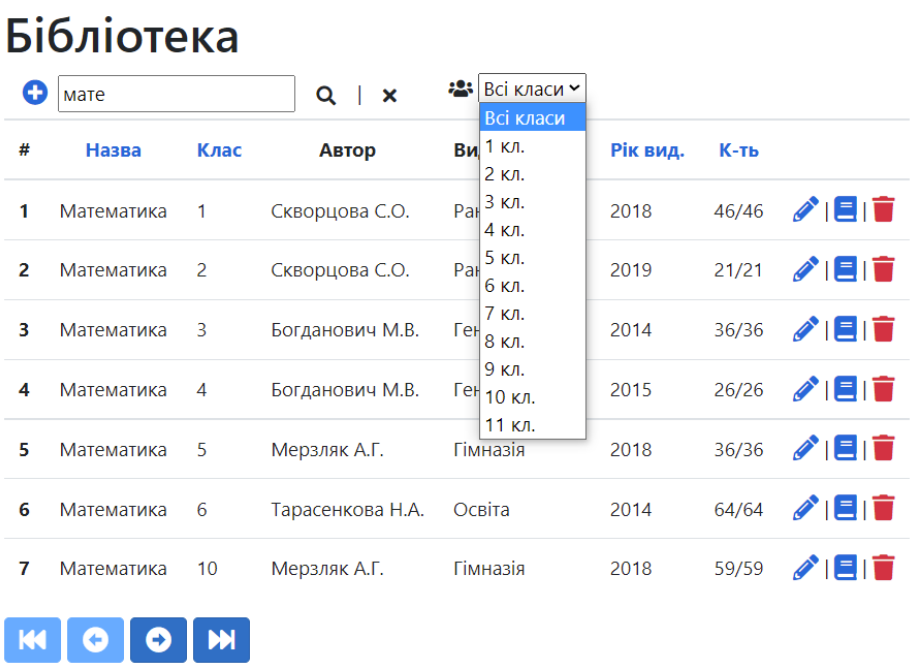

Рис. 4.11. Фільтрація книг по ключовому слову та по класу

Для додавання нової книги до книжкового фонду, необхідно натиснути клавішу зі знаком «плюс» над списком книг бібліотеки. Для занесення нової книги необхідно вказати автора, назву, видавництво, рік видання, клас, кількість примірників.

Редагування книги доступно зі сторінок «Бібліотека», «Видача і повернення книг», «Читачі з книгою». Для редагування є кнопка із зображенням олівця або посилання «Редагувати».

Для редагування доступні наступні поля: автор, назва, видавництво, рік видання, клас, кількість примірників.

Щоб видалити книгу, існує червона кнопка у вигляді кошика для сміття. Перед видаленням користувач повинен підтвердити свій вибір, натиснувши на кнопку «Видалити».

Після видалення користувач попадає на сторінку зі списком книг.

Видача книг здійснюється зі сторінки зі списком книг. Для цього необхідно натиснути на кнопку із зображенням книги. Користувач попадає на сторінку «Видача і повернення книг».

За замовчуванням, це сторінка для видачі книг учням. Для видачі книг вчителям, необхідно натиснути на кнопку «Персонал» у верхній частині екрану. На сторінці представлений список учнів. Клас залежить від обраної книги. Так, для книги для першого класу буде представлений список учнів одного з перших класів.

Біля прізвищ учнів є два види кнопок: «Видати» і «Повернути». Якщо в учня наявна книга на руках, то навпроти його прізвища у списку буде стояти кнопка «Повернути». І навпаки, якщо в учня немає книги, кнопка буде мати вигляд «Видати». На сторінці видачі підручників наявний функціонал пошуку по ключовому слову. Також за допомогою випадаючого списку можливо обрати лише учнів певного класу. Після видачі чи повернення книги змінюється кількість наявних підручників у відповідному рядку.

Зі сторінки «Видача і повернення книг» за відповідним посиланням можна перейти на сторінку «Читачі з книгою».

Сторінка «Читачі з книгою» показує список усіх читачів, що мають на руках певну книгу. Список містить прізвище, ім’я, по батькові читача, а також чи є читач учнем чи працівником школи. Список можна сортувати по прізвищу чи по посаді. Увівши ключові слова у рядок пошуку, можна відфільтрувати список.

4.14. Модуль «Інвентаризація»

Модуль допомагає завідуючому господарською частиною школи вести облік майна на балансі закладу. Модуль розділений на дві частини, перехід між якими здійснюється кнопкою у верхній частині екрану: «Майно на обліку» та «Списане майно». Сторінка модуля має список майна з вказанням найменування, інвентарного номеру, кількості, ціни, дати оприбуткування або списання. На сторінці є можливість пошуку по ключовому слову, сортування по всім колонкам. Для сортування необхідно натиснути на заголовок стовпчика. При повторному натисканні змінюється напрямок сортування: з низхідного на висхідне чи навпаки (рис. 4.12).

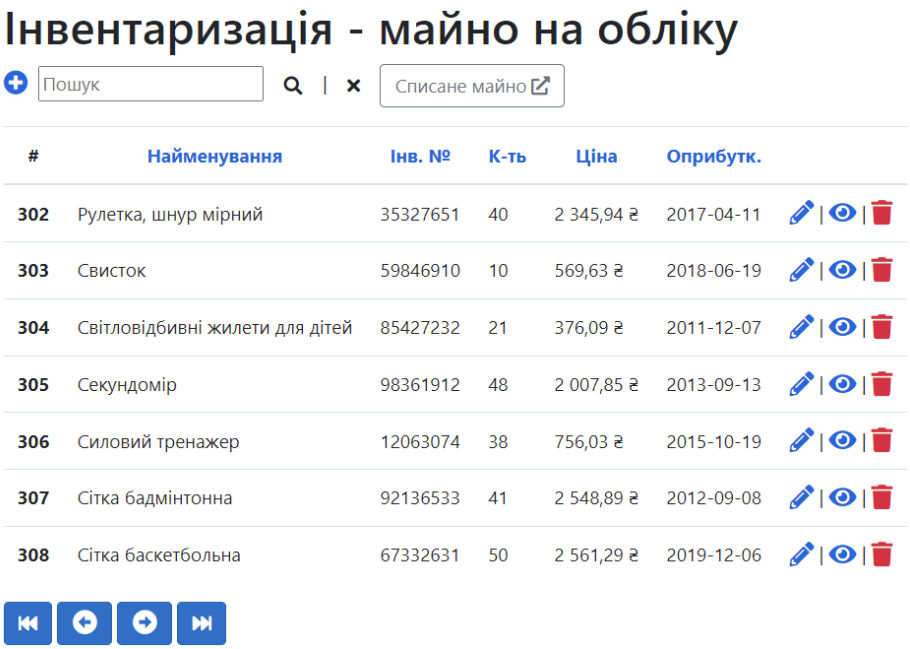

Рис. 4.12. Майно на обліку у модулі «Інвентаризація»

Для створення запису про нове майно необхідно натиснути на кнопку зі знаком «плюс». На сторінці створення необхідно вказати найменування, інвентарний номер, кількість, ціну, дату оприбуткування майна та натиснути на кнопку «Створити».

Для редагування необхідно натиснути на кнопку із зображенням олівця навпроти потрібного запису. Якщо майну, що стоїть на обліку, ввести дату списання, то воно потрапить у список списаного майна.

При натисканні на кнопку у вигляді ока навпроти відповідного запису, відкривається вікно детальної інформації про майно. Якщо майно списане помилково, його можна знову поставити на баланс школи. Якщо майно потрібно списати, на сторінці «Деталі» присутня необхідна кнопка (рис. 4.13).

Для помилково введеного запису можливе його видалення. Кнопка видалення має вигляд червоного кошика для сміття навпроти запису.

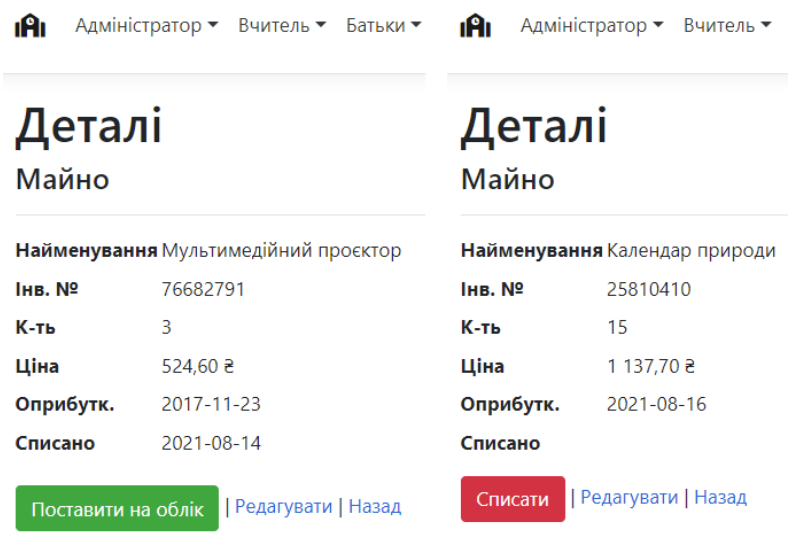

Рис. 4.13. Списання та постановка на облік майна

4.15. Модуль «Розклад уроків»

Цей модуль служить для формування учнівського розкладу. Передбачена автоматична генерація та ручне складання розкладу.

Для початку роботи у випадаючому списку необхідно обирати день тижня. Розклад представлено у вигляді таблиці. При натисканні на відповідну клітинку, можна створити заняття на певному уроці з певною комбінацією вчитель/клас. Кнопки для автоматичної генерації розкладу створюють його генетичним методом або методом перестановок на вибір користувача (рис. 4.14).

Рис. 4.14. Редактор та генератор розкладу

Для редагування уроку, необхідно натиснути на чарунці, що знаходиться на перетині рядка вчителя та стовпця з номером уроку і натиснути на кнопці з назвою класу.

У випадаючому списку «Клас» доступні лише ті класи, які вільні на даному уроці. У випадаючому списку «Предмет» перераховані лише предмети, які може викладати обраний вчитель. На цій сторінці можливе видалення уроку або повернення назад до розкладу уроків.

Для створення уроку, необхідно натиснути кнопку зі знаком «плюс», відкриється сторінка створення нового уроку.

У випадаючому списку «Клас» присутні лише ті класи, які вільні на цьому уроці. Випадаючий список «Предмет» містить лише ті предмети, які може викладати відповідний вчитель. Редагування полів «День», «№ уроку», «ПІБ вчителя» недоступно, тому що вони обираються автоматично на попередній сторінці, коли натискається кнопка з позначкою «плюс» у відповідній чарунці розкладу. Створені таким чином уроки відображаються у модулі «Класний журнал», що доступний для редагування адміністратору та для перегляду вчителю.

На основі «Розкладу уроків: вчителі» формується розклад уроків для учнів. Цей розклад доступний для перегляду батьками у розділі «Батьки – Розклад уроків». Він має інший формат, ніж розклад для вчителів. Для відображення інформації необхідно обрати клас у випадаючому списку (рис. 4.15).

Рис. 4.15. Розклад уроків для учнів та батьків

4.15.1. Алгоритми генерації розкладу у веб-застосунку

Проблема генерації розкладу постає у різних галузях: освіта, медицина, спорт, транспорт. Ефективне планування дозволяє економити час, кошти, підвищити ефективність навчального процесу. Проблема є комбінаторно вибуховою: існує безліч сполучень можливих варіантів розкладу. Не можна точно сказати, яка з комбінацій є найбільш оптимальною. Через це доцільно використовувати методи перебору та евристичні методи. При цьому слід розуміти, що евристичні методи дають лише наближений до ідеального результат.

Задача складання розкладу у великих навчальних закладах покладена на висококваліфікованих спеціалістів чи їх групи, які витрачають багато годин для досягнення прийнятного результату із задоволенням усіх вихідних умов [31].

Генерація розкладу передбачає розташування уроків у певні проміжки часу таким чином, щоб жоден клас, учитель чи кабінет не були задіяні в уроці більше ніж один раз, а також, щоб були задоволені інші другорядні умови.

У генераторі веб-застосунку «Система управління школою» присутні наступні обов’язкові обмеження:

- вчитель може проводити урок лише в одному класі у певний проміжок часу;
- клас не може мати вікон у розкладі;

М’які обмеження генератора:

- рівномірний розподіл предметів по дням;
- рівномірний розподіл кількості уроків впродовж тижня;
- максимальна кількість уроків на день;
- зменшення кількості вікон вчителів.

Обмеження генератора, які залишились не реалізованими:

- парні/непарні тижні;
- розподіл класів на групи для вивчення окремих предметів;
- вивчення окремих предметів у спеціалізованих кабінетах.

Генератор розкладу доступний зі сторінки «Розклад уроків: вчителі».

Генератор має два алгоритми для складання розкладу на вибір користувача: генетичний та метод перестановок.

4.15.2. Генетичний алгоритм

Генетичний алгоритм підтримується сутностями, описаними нижче.

Gene (ген) – одиниця розкладу уроків, що містить у собі розклад одного класу у вигляді послідовності цілих чисел. Довжина гену дорівнює кількості уроків на тиждень певного класу. При створенні гену числа розподіляються випадковим чином, що покликане урізноманітнити «населення». Завдяки випадковості, можливе виникнення позитивних послідовностей генів та сполучень генів у хромосомі, які будуть частиною рішення задачі.

Chromosome (хромосома) – містить у собі набір генів (розкладів одного класу). Саме вона складає одиницю «населення». Природний добір, схрещування і мутації відбуваються з хромосомою.

Важливою характеристикою хромосоми є її здоров’я (fitness). Завдяки цьому показнику визначається як здоров’я самої хромосоми, так і загальне здоров’я покоління. Результат роботи функції здоров’я показує, чи досяг алгоритм складання розкладу своєї цілі. Здоров’я підраховується в ході роботи функції оцінки – це найважливіша функція, від правильності роботи якої залежить успіх роботи всього алгоритму. Функція оцінки визначає, хто з населення матиме шанс продовжити рід, вижити. При неправильній роботі функції алгоритм не зможе досягти мети.

Slot (комірка розкладу) – найменший елемент розкладу, містить інформацію про клас, предмет, учителя.

Table (таблиця розкладу) – таблиця-вказівник, що містить у собі комірки (slots). Таблиця містить усі уроки навчального закладу. Уроки розташовані підряд. На комірки цієї таблиці посилаються гени.

Scheduler (планувальник) – відповідає за виконання генетичного алгоритму. У планувальнику ініціалізуються початкові дані, завантажуються налаштування,

запускається процес генерації поколінь, схрещування, мутацій та відбору найкращих представників.

Після ініціалізації даних, які завантажуються з бази даних та файлу налаштувань, починається робота алгоритму.

Випадковим чином генерується перше населення хромосом, що містять гени. Під час генерації відбувається оцінка здоров'я кожної хромосоми та підраховується загальне здоров'я покоління. Після генерації першого покоління, населення рангується відповідно до показника здоров'я – найздоровіші стають першими, невдалі комбінації – останніми.

Основна частина алгоритму полягає у генерації нових поколінь населення хромосом. На початку цього етапу обираються найкращі 10% попереднього населення і вони стають першими громадянами нового покоління. Цей відбір називається елітаризмом (elitism). Він дозволяє найкращим представникам потрапити у наступне покоління. Таким чином найбільш сприятливі комбінації ген захищаються від мутацій.

Подальше формування населення відбувається завдяки схрещуванню. Схрещування являє собою процес поєднання характеристик батьків для створення дитини, яка містить характеристики як батька, так і матері.

Для відбору батьків використовується принцип колеса рулетки (roulette wheel or fitness proportionate selection). Кожна хромосома має шанс бути обраною пропорційно до її здоров'я. Чим вище здоров'я, тим більший шанс стати батьком чи матір'ю. Цей принцип, на відміну від відбору відсіканням найслабших індивідуумів, дозволяє навіть менш здоровим хромосомам вижити у процесі відбору. І хоча шанс вижити у слабших представників низький, він не є нульовим. Це є перевагою, так як є шанс, що слабкі хромосоми містять якості чи характеристики, які будуть сприятливими після процесу рекомбінації [32].

Альтернативно у програмі присутній алгоритм відбору «змагання» (tournament selection). В цьому алгоритмі випадковим чином обираються декілька представників населення, після чого між ними організовується «змагання», де перемагає

найсильніший, який зможе продовжити рід. Кількість учасників впливає на шанс слабших індивідуумів бути обраними, так як при збільшенні кількості гравців збільшується шанс того, у серед них виявиться сильніший. Алгоритм змагання має переваги, так як його легко реалізувати програмно, він піддається паралелізації, у ньому легко регулювати тиск відбору [33].

Під час схрещування існує шанс, що дитина може прийняти від батька усі характеристики. Але в переважній більшості випадків, дитина отримує частину якостей батька і частину якостей матері. У даному веб-застосунку реалізоване одноточкове схрещування. Точка схрещування обирається випадковим чином у певному інтервалі. «Дитина» отримує генетичний матеріал матері до точки схрещування і матеріал батька – після.

Після схрещування відбувається однорідна мутація (uniform mutation). Однорідність мутації полягає у тому, що створюється новий життєздатний ген, який цілком замінює випадковим чином обраний ген у хромосомі. Мутація покликана підтримувати генетичне різноманіття. Вона допомагає уникнути локальних мінімумів перешкоджаючи популяції хромосом ставати занадто схожими одна на одну, водночас уповільнюючи чи навіть зупиняючи наближення до глобального оптимуму. Уникнення гомогенізації є причиною того, що генетичні алгоритми не обмежуються лише найкращими представниками для утворення наступного покоління. Батьків обирають випадковим чином, надаючи перевагу здоровішим особам [34].

Після схрещування і мутацій нова хромосома оцінюється і додається до популяції. Цей процес продовжується доки не буде знайдена хромосома, що задовольняє усі обмеження. Блок-схема алгоритму розміщена у додатку Б.

У алгоритмі даного веб-застосунку наявні декілька додаткових етапів, які дозволяють досягти потрібного результату швидше.

Перший додатковий етап – це механізм перестановок уроків для зменшення кількості вікон вчителів. Даний етап відбувається після схрещування і мутацій.

Ще один етап, який допомагає досягти потрібного результату швидше – це етап точкової мутації конфліктних ділянок генів. Конфліктні ділянки зі списку обмінюється з певною ймовірністю або з іншою такою ж конфліктною ділянкою, або з випадково обраною ділянкою цього ж гену. Без даних маніпуляцій алгоритм генерації розкладу лише підходить до правильного рішення, де найкращий представник населення має показник здоров'я, що коливається у межах 96-98%. Це можна пояснити щільним розкладом, неможливістю появи вікон у розкладі, обмеженою кількістю викладачів.

В роботі алгоритму були виявлені наступні закономірності.

Мутація необхідна для отримання результату. Нульова мутація означає, що результат цілковито залижить від першого випадково згенерованого покоління. Якщо у першому поколінні були наявні потрібні гени, то алгоритм може знайти відповідь. Занадто маленька мутація у 1% часто не дозволяє досягти рішення задачі. Мутація 5-40% дозволяє досягти результату за найменший час. Зі збільшенням відсотку мутації збільшується кількість випадків, коли алгоритм не знаходить рішення, а також росте час знаходження відповіді (таблиця 4.1).

Таблиця 4.1

Вплив мутації на швидкість досягнення результату

Мутація, %	**Не знайшов рішення за 500 поколінь, %**	**Кількість поколінь до розв'язку, од**
0	Залежить від початкових випадково згенерованих даних	
1	20	80
5	5	100
10	5	100
20	5	100
30	5	130
40	5	145
50	20	180
70	25	250-350
90	55	400+

Зі збільшенням популяції лінійно росте час, необхідний на створення кожного покоління (рис. 4.16).

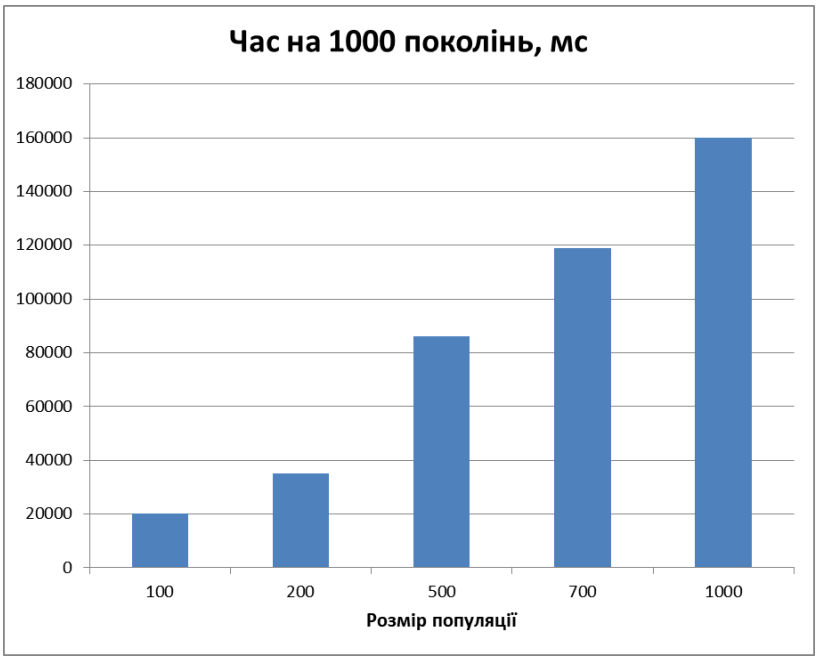

Рис. 4.16. Вплив розміру популяції на час досягнення результату

Чисельність населення не впливає на кількість поколінь, необхідних для знаходження розв’язку. Тому необхідно обирати це значення мінімальним (рис. 4.17).

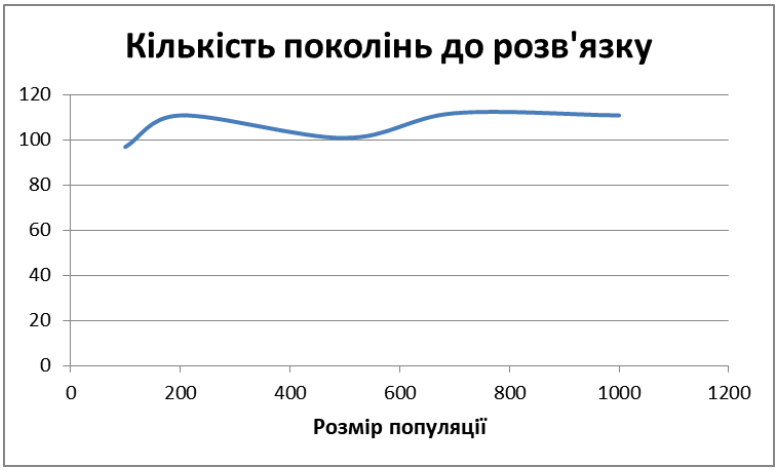

Рис. 4.17. Вплив розміру популяції на кількість поколінь, потрібних для розв’язку

Отже, алгоритм справляється зі своєю задачею, але 100% результату він досягає не завжди.

4.15.3. Генерація методом перестановок

Алгоритм генерації методом перестановок складається з двох частин. Перша частина – це безпосередньо генератор. Він складає перший варіант розкладу, який далі покращується другою частиною алгоритму, яка видаляє «вікна» у розкладі. Блок-схеми зазначених алгоритмів представлені у додатку В.

Алгоритм генерації розкладу по черзі обирає вчителя, номер уроку, день. Виставляються всі уроки вчителя на першому уроці у всі дні, потім на другому уроці і так далі, поки в учителя більше не залишиться уроків, передбачених навантаженням. Такий порядок заповнення журналу забезпечує рівномірне розташування уроків по дням (немає декількох однакових уроків в один день) та рівномірний розподіл уроків впродовж тижня (тобто початок тижня і кінець містять приблизно однакову кількість уроків). По мірі заповнення розкладу, все важче розташовувати вчителів без конфліктів. Конфлікт означає, що в даний день і на даному уроці у даного класу або у даного вчителя вже є інший урок.

Алгоритм передбачає вибір предмету і класу для вчителя випадковим чином. Це дозволяє перегенерувати розклад з іншим результатом. Хоча можливе редагування розкладу вручну, такий спосіб регенерації може допомогти швидше підібрати вихідний варіант розкладу для ручної корекції.

Після початкової генерації розкладу у ньому можливе утворення «вікон», тобто часових проміжків, коли в учнів між уроками виникає незайнятий урок. Це неприпустимо у шкільному розкладі. Тому після первинного складання розкладу вступає у дію частина алгоритму, яка відповідає за усунення «вікон» у розкладі.

У алгоритмі «RemoveGaps()» спочатку обирається клас, потім день і, нарешті, номер уроку. Перевіряється, чи є перед уроком «вікно». Після виявлення вікна алгоритм намагається з’ясувати, чи можливо просто підняти урок вище. У більшості випадків просте підняття неможливе, тому далі алгоритм визначає конфліктний урок, який не дає підняти урок з «вікном» вище.

Конфліктний урок – це урок, який не дає змогу наступному уроку з «вікном» зайняти часовий проміжок вище, тобто «піднятись» вище у розкладі на місце

«вікна». Підняття неможливе через те, що вчитель уроку з «вікном» зайнятий на попередньому уроці у іншому класі. Тому завдання алгоритму – знайти можливість поміняти місцями конфліктний урок, вивільнивши таким чином місце для підняття уроку з «вікном». Відповідно, конфліктний клас – це клас, у якому проходить конфліктний урок.

Алгоритм визначає кількість уроків щодня у класі з «вікном». Починаючи з дня, де найменша кількість уроків, визначається, чи можливо перенести урок з вікном на інший день і поставити останнім уроком. Якщо це неможливо, урок додається у список конфліктних.

Потім метод «RemoveGaps()» намагається поміняти кожен конфліктний урок місцями з іншим уроком у цьому ж класі. Це досягається наступними діями:

- знаходяться всі вільні уроки вчителя конфліктного уроку у цей день;
- виходячи з визначених на попередньому кроці вільних уроків алгоритм знаходить, хто з учителів має в цей час уроки у конфліктному класі;
- перевіряється, чи є у цих вчителів урок на місці конфліктного;
- якщо уроку немає, то уроки міняються місцями;
- піднімається урок з «вікном» вище.

Блок-схема алгоритму наведена у додатку В.

Слід зазначити, що не існує алгоритму генерації розкладу, який би повністю відповідав усім вимогам певного навчального закладу:

- вчитель працює у двох школах і не може починати роботу раніше певного часу чи у певні дні;
- у школі є кабінети праці, якими користується інший заклад, тому не можна проводити уроки у певні дні/години;
- деякі вчителі мають методичний день у певний день тижня, тому їм не можна ставити уроки в цей день;
- не можна розривати парні уроки;
- не бажано мати уроки фізкультури чи мистецтва на початку дня, а складні уроки – в кінці;

- не бажано мати на початку тижня переважно математику, а в кінці – переважно мову і літературу;
- хімію, інформатику та ряд інших предметів можна проводити лише коли вільний відповідний кабінет;
- деякі уроки повинні бути спареними, наприклад, для написання підсумкових робіт.

Існує безліч моментів, які не можна попередньо врахувати у алгоритмі. Без ручної корекції неможливо скласти розклад. Автоматична генерація – лише відправна точка складання розкладу уроків у школі.

Ручна корекція розкладу частково автоматизована. Якщо клас вже має урок у певний часовий проміжок, то програма не відображає цей клас у списку можливих при створенні уроку у розкладі. У розкладі чітко видно, коли вчитель зайнятий, а де у нього вікно, тому неможливо відправити вчителя навчати два різних класи в один і той же проміжок часу. Така автоматизація спрощує ручну корекцію і складання розкладу, допомагає уникнути помилок.

Висновок

Веб-застосунок «Система управління школою» є системою менеджменту середнього загальноосвітнього закладу.

Застосунок побудований за модульним принципом. Кожен модуль виконує певний функціонал. Така система побудови дозволяє вести розробку поступово.

Для доступу до застосунку необхідно пройти автентифікацію та авторизацію. Автентифікація полягає у введенні логіну чи паролю або використанні сервісу Google для входу. Авторизація дозволяє контролювати, які ресурси доступні користувачеві.

Застосунок працює з базою даних персоналу і учнів, тому тут наявні модулі «Вчителі», «Учні», які дозволяють створювати, видаляти, редагувати, переглядати відповідну інформацію.

Модуль «Класи» містить список учнів певного класі та графічно відображає статистичну інформацію щодо віку та статі учнів, що стане в пригоді класним керівникам.

Модуль «Предмети» дозволяє проводити операції додавання, редагування, видалення предметів та встановлювати відповідність між вчителем та предметами, які він веде.

Модуль «Розклад уроків» слугує для автоматичної генерації та ручного редагування розкладу. Розклад має два представлення: для вчителів та учнів. З цим модулем тісно пов’язаний модуль «Навчальний план». Він містить інформацію про клас, предмет, вчителя, кількість уроків на тиждень. Саме на основі введених у ньому даних генерується розклад уроків.

Модуль «Класний журнал» дає змогу вести оцінювання навчальних досягнень здобувачів освіти, а також відмічати відсутність учнів. Дані класного журналу відображаються у режимі перегляду у модулі «Навчальні досягнення», доступ до якого мають батьки.

Модуль «Статистика» відображає у вигляді стовпчастих та кругових діаграм інформацію по успішності, статі, віку. Графічне представлення здійснюється за допомогою JavaScript бібліотеки Chart.js.

Модуль «Дистанційне навчання» відповідає на виклики сучасності. Він дозволяє вчителю записувати форматований текст, вставляти посилання, зображення для дистанційних уроків. Для створення і редагування дописів модулю «Дистанційне навчання», використовується JavaScript редактор тексту «TinyMCE».

Так як у цьому модулі існує можливість вставки Html коду у вікні редагування, то для захисту від міжсайтового скриптингу використовується .NET бібліотека «HtmlSanitizer». Вона має відкритий код і використовується для очищення HTML від конструкцій, які можуть призвести до XSS атак.

Модуль «Бібліотека» покликаний автоматизувати роботу бібліотекаря з обліку книжкового фонду та видачі підручників і книг.

У модулі «Інвентаризація» здійснюється облік майна школи: постановка на облік та списання.

З функціоналом модулів можна ознайомитись у діаграмі варіантів використання у додатку А.

ВИСНОВКИ

У даній роботі здійснено аналіз ринку програмного забезпечення для середніх загальноосвітніх навчальних закладів. Розроблений веб-застосунок «Система управління школою», який включає модулі, що покликані вирішувати конкретні проблеми, які кожного дня постають перед адміністрацією та персоналом середньої загальноосвітньої школи.

Завдання, які виконують працівники школи включають: ведення обліку вчителів, учнів, підручників, матеріальної бази, заповнення електронного журналу, дистанційне навчання, ведення статистики навчальних досягнень, складання навчального плану та розкладу уроків. Для кожної із задач був розроблений окремий програмний модуль.

Для підтримки роботи застосунку спроєктована база даних, продумані зв’язки між сутностями, створені відповідні класи. Між таблицями впроваджені зв’язки один-до-багатьох, багато-до-багатьох.

Застосунок базується на фреймворку ASP.NET Core та написаний мовою програмування C#. У якості об’єктно-реляційного перетворювача використаний Entity Framework Core. Дані зберігаються у реляційній базі даних SQL Server та Azure SQL.

Модулі автентифікації та авторизації застосунку здійснюють захист даних від несанкціонованого доступу, обмежуючи доступ до сторінок відповідно до ролей користувачів. Перевірка введених користувачем даних здійснюється на клієнті (за допомогою jQuery, тегів-помічників) та на сервері засобами фреймворку ASP.NET Core та бібліотеки «FluentValidation».

На основі даних, закладених у таблиці бази даних, організована робота модулів веб-застосунку. Основними типами сторінок є загальний список, сторінка детальної інформації, сторінки для редагування, створення та видалення сутності. Статистичні модулі здійснюють аналіз та графічне представлення інформації.

Модуль генерації розкладу методом перестановок та генетичним методом розташовує пари вчитель-клас так, щоб уникнути конфліктів.

Для адміністрації закладу створені модулі «Навчальний план», «Розклад уроків», довідники персоналу та учнів. Статистичний модуль дозволяє керівництву школи проводити моніторинг навчальних досягнень здобувачів освіти.

Модуль «Бібліотека» зменшує кількість паперової роботи у бібліотеці. У ньому ведеться облік підручників та видача книг читачам. Модуль показує картину наявності примірників, список читачів з книгою, дозволяє видавати та приймати підручники.

Модуль «Інвентаризація» служить для постановки майна на облік та для його списання. Він покликаний підвищити ефективність роботи завідуючого з господарської частини школи.

Для батьків створені модулі «Навчальні досягнення», «Дистанційне навчання» та «Розклад уроків». Батьки можуть переглядати успіхи дітей у навчанні, бачити розклад уроків, дізнаватись завдання з дистанційного навчання.

Вчителі ведуть облік навчальних досягнень учнів та відвідування уроків у модулі «Класний журнал».

Відповіддю на виклики сучасного світу є модуль з дистанційного навчання. Він дозволяє вчителю проводити офлайн уроки, складати текстові завдання, вставляти малюнки, відео, схеми, таблиці, посилання, які бачать учні та батьки.

Даний застосунок може бути використаний у середніх загальноосвітніх закладах України. Для впровадження у конкретний навчальний заклад необхідно провести роботу з наповнення бази даних релевантною інформацією про працівників, учнів, книжковий фонд, матеріальну базу тощо. За потреби, можлива розробка додаткових модулів чи модифікація існуючих для урахування місцевої специфіки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Система «Єдина школа». *Eschool-ua.com*: веб-сайт. URL: https://eschool-ua.com/ (дата звернення: 10.10.2023).
2. Проєкт «КУРС: Освіта». *Ekyrs.org*: веб-сайт. URL: http://ekyrs.org/ua/project/ (дата звернення: 15.10.2023).
3. ASP.NET Core. *Wikipedia: The Free encyclopedia*: веб-сайт. URL: https://en.wikipedia.org/wiki/ASP.NET_Core (дата звернення: 19.10.2023).
4. Introduction to ASP.NET Core. *Microsoft.com*: веб-сайт. URL: https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core (дата звернення: 19.10.2023).
5. Watson M. ASP.NET Razor Pages vs MVC: How Do Razor Pages Fit in Your Toolbox? *Stackify.com*: веб-сайт. URL: https://stackify.com/asp-net-razor-pages-vs-mvc/ (дата звернення: 19.10.2023).
6. Chauhan S. Understanding MVC, MVP and MVVM Design Patterns *Dotnettricks.com*: веб-сайт. URL: https://www.dotnettricks.com/learn/designpatterns/understanding-mvc-mvp-and-mvvm-design-patterns (дата звернення: 20.10.2023).
7. Lock A. ASP.NET Core in Action: Вид. 2-ге: Manning, 2021. 832 с.
8. ASP.NET Core fundamentals. *Microsoft.com*: веб-сайт. URL: https://docs.microsoft.com/en-us/aspnet/core/fundamentals (дата звернення: 20.10.2023).
9. Deployment Environment. *Wikipedia: The Free encyclopedia*: веб-сайт. URL: https://en.wikipedia.org/wiki/Deployment_environment (дата звернення: 21.10.2023).
10. Entity Framework Core. *Microsoft.com*: веб-сайт. URL: https://docs.microsoft.com/en-us/ef/core/ (дата звернення: 21.10.2023)

11. Object–relational. *Wikipedia: The Free encyclopedia*: веб-сайт. URL: https://en.wikipedia.org/wiki/Object%E2%80%93relational_mapping (дата звернення: 22.10.2023).
12. Change Tracking in EF Core. *Microsoft.com*: веб-сайт. URL: https://docs.microsoft.com/en-us/ef/core/change-tracking/ (дата звернення: 22.10.2023).
13. Relationships. *Microsoft.com*: веб-сайт. URL: https://docs.microsoft.com/en-us/ef/core/modeling/relationships/ (дата звернення: 22.10.2023).
14. Creating your first validator. *Fluentvalidation.net*: веб-сайт. URL: https://docs.fluentvalidation.net/en/latest/start.html (дата звернення: 23.10.2023).
15. How to configure and use Live Unit Testing. *Microsoft.com*: веб-сайт. URL: https://docs.microsoft.com/en-us/visualstudio/test/live-unit-testing (дата звернення: 23.10.2023).
16. Unit test basics. *Microsoft.com*: веб-сайт. URL: https://docs.microsoft.com/en-us/visualstudio/test/unit-test-basics?view=vs-2019 (дата звернення: 24.10.2023).
17. About xUnit.net. *xUnit.net*: веб-сайт. URL: https://xunit.net/ (дата звернення: 24.10.2023).
18. Moq. *Github.com*: веб-сайт. URL: https://github.com/moq/moq4 (дата звернення: 25.10.2023).
19. Shared Context Between Tests. *xUnit.net*: веб-сайт. URL: https://xunit.net/docs/shared-context.html (дата звернення: 25.10.2023).
20. Walkthrough: Create and run unit tests for managed code. *Microsoft.com*: веб-сайт. URL: https://docs.microsoft.com/en-us/visualstudio/test/walkthrough-creating-and-running-unit-tests-for-managed-code (дата звернення: 25.10.2023).

21. Logging in .NET Core and ASP.NET Core. *Microsoft.com*: веб-сайт. URL: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/logging/ (дата звернення: 26.10.2023).
22. Responsive web design. *Wikipedia: The Free encyclopedia*: веб-сайт. URL: https://en.wikipedia.org/wiki/Responsive_web_design (дата звернення: 27.10.2023).
23. What is Responsive Web Design? *W3schools.com*: веб-сайт. URL: https://www.w3schools.com/css/css_rwd_intro.asp (дата звернення: 27.10.2023).
24. What is CI/CD? *Redhat.com*: веб-сайт. URL: https://www.redhat.com/en/topics/devops/what-is-ci-cd (дата звернення: 28.10.2023).
25. Understanding GitHub Actions. *GitHub.com*: веб-сайт URL: https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions (дата звернення: 29.10.2023).
26. Introduction to Identity on ASP.NET Core. *Microsoft.com*: веб-сайт. URL: https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-5.0&tabs=visual-studio (дата звернення: 13.11.2023).
27. Introduction to authorization in ASP.NET Core. *Microsoft.com*: веб-сайт. URL: https://docs.microsoft.com/en-us/aspnet/core/security/authorization/introduction?view=aspnetcore-5.0 (дата звернення: 14.11.2023).
28. Simple HTML5 Charts using the <canvas> tag. *Github.com*: веб-сайт. URL: https://github.com/chartjs/Chart.js (дата звернення: 15.11.2023).
29. TinyMCE Features. *TinyMCE*: веб-сайт. URL: https://www.tiny.cloud/tinymce/features/ (дата звернення: 16.11.2023).
30. HtmlSanitizer. *Github.com*: веб-сайт. URL: https://github.com/mganss/HtmlSanitizer (дата звернення: 19.11.2023).

31. Nanda A., Manisha P. Pai, Abhijeet G. An Algorithm to Automatically Generate Schedule for School Lectures Using a Heuristic Approach. *International Journal of Machine Learning and Computing*. 2012. Vol. 2. No. 4. P. 492 – 495. URL: https://www.researchgate.net/publication/271297445_An_Algorithm_to_Automatically_Generate_Schedule_for_School_Lectures_Using_a_Heuristic_Approach (дата звернення: 20.11.2023).
32. Fitness proportionate selection. *Wikipedia: The Free encyclopedia*: веб-сайт. URL: https://en.wikipedia.org/wiki/Fitness_proportionate_selection (дата звернення: 24.11.2023).
33. Tournament selection. *Wikipedia: The Free encyclopedia*: веб-сайт. URL: https://en.wikipedia.org/wiki/Tournament_selection (дата звернення: 26.11.2023).
34. Mutation (genetic algorithm). *Wikipedia: The Free encyclopedia*: веб-сайт. URL: https://en.wikipedia.org/wiki/Mutation_(genetic_algorithm) (дата звернення: 27.11.2023).

Діаграма варіантів використання веб-застосунку

Рис. А.1. Діаграма варіантів використання: Адміністратор

Рис. А.2. Діаграма варіантів використання: Вчитель

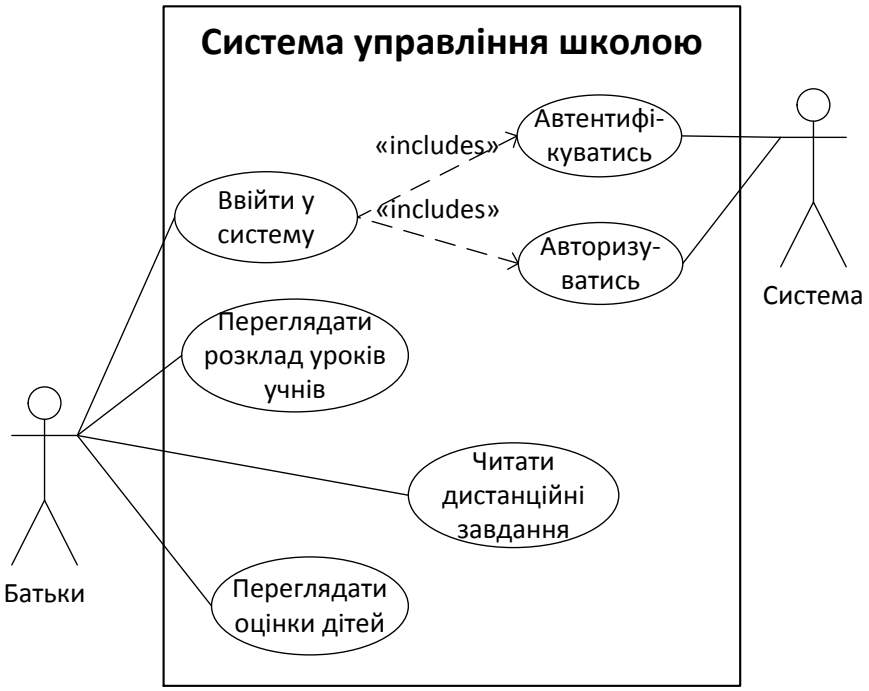

Рис. А.3. Діаграма варіантів використання: Батьки

Рис. А.4. Діаграма варіантів використання: Бібліотекар

Рис. А.5. Діаграма варіантів використання: Завгосп

ДОДАТОК Б

Блок-схема алгоритму генерації розкладу генетичним методом

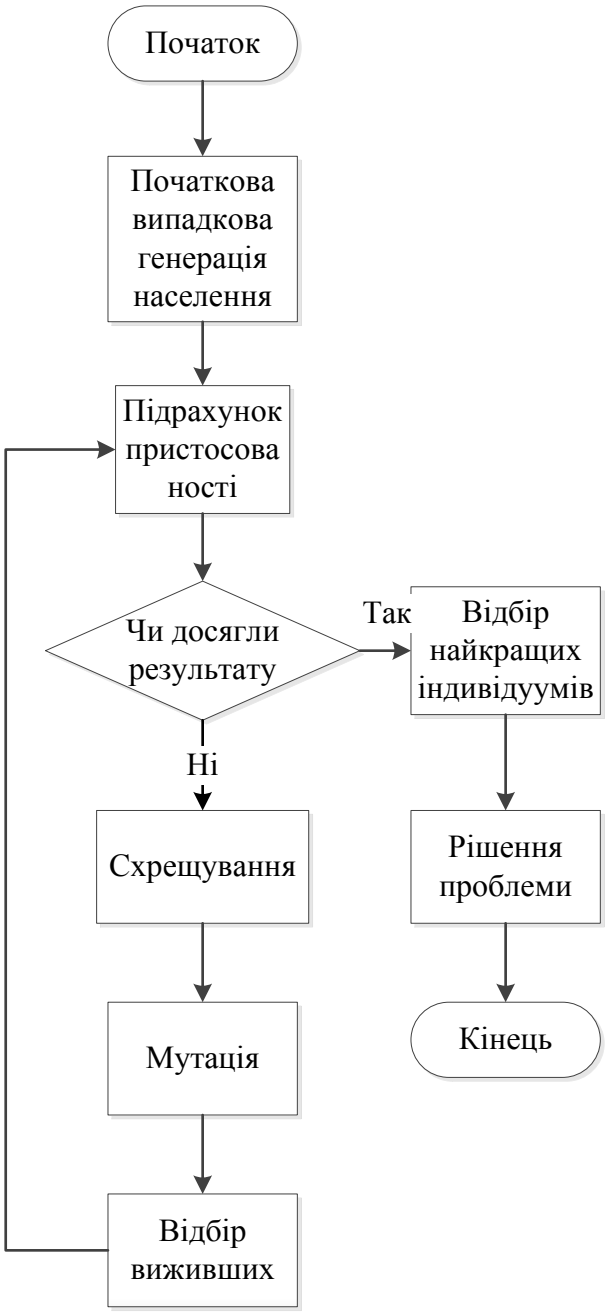

Рис. Б.1. Блок-схема генетичного алгоритму

ДОДАТОК В

Блок-схема алгоритму генерації розкладу методом перестановок

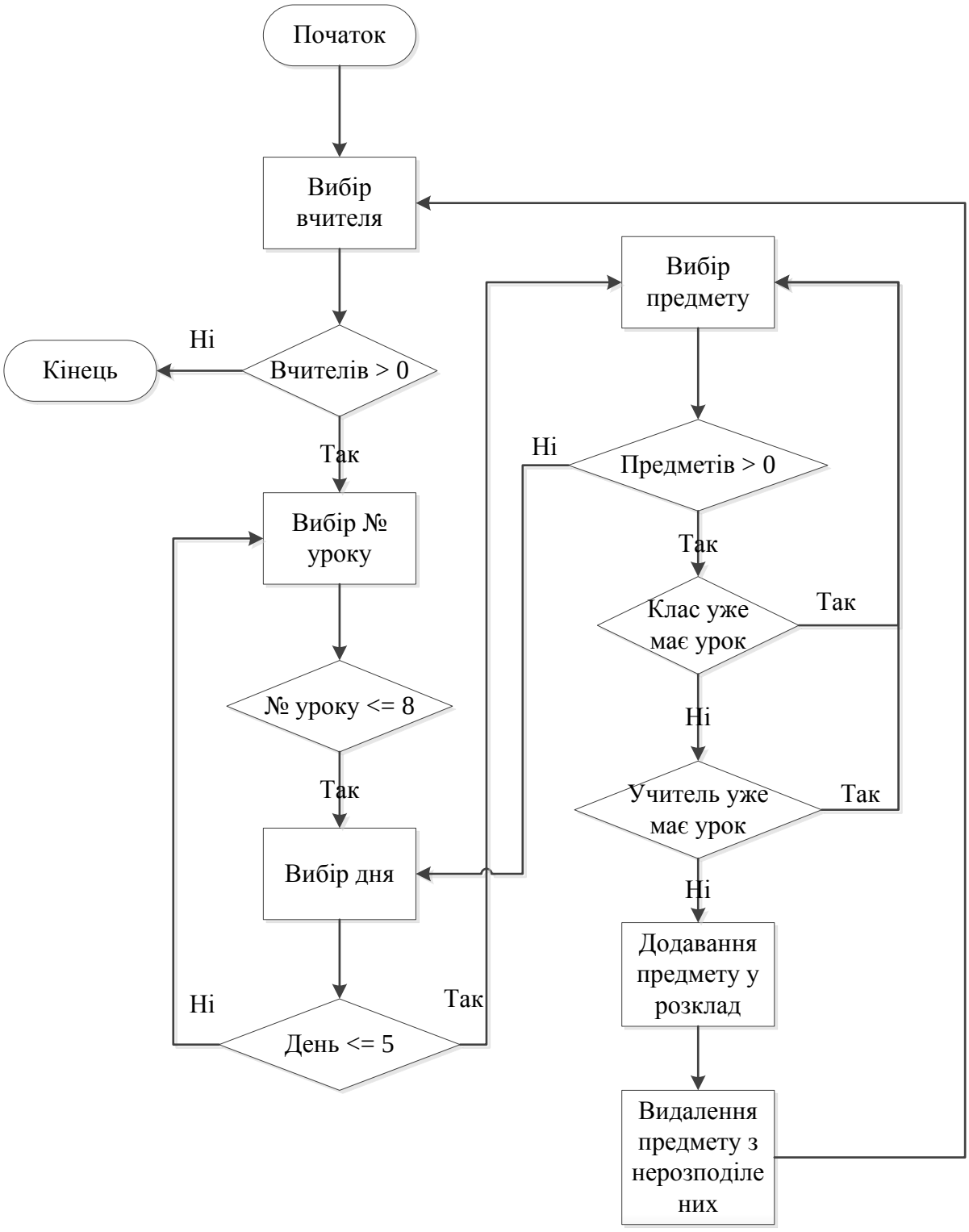

Рис. В.1. Алгоритм генерації розкладу. Метод «Generate()»

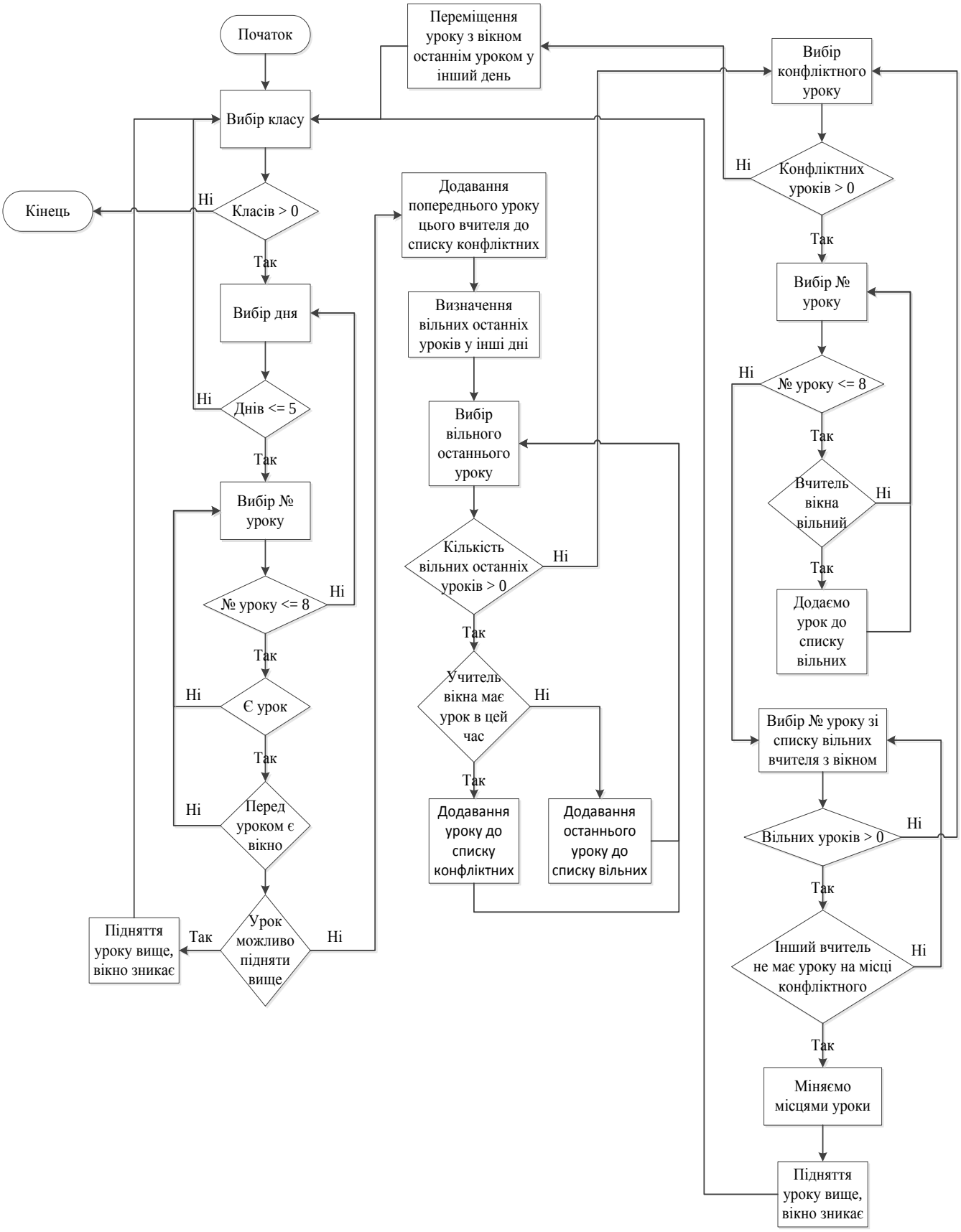

Рис. В.2. Алгоритм генерації розкладу. Метод «RemoveGaps()»