Національний університет «Полтавська політехніка імені Юрія Кондратюка»

(повне найменування вищого навчального закладу)

Навчально-науковий інститут інформаційних технологій та робототехніки

(повна назва факультету)

Кафедра комп’ютерних та інформаційних технологій і систем

(повна назва кафедри)

**Пояснювальна записка**

до дипломної роботи

бакалавра

(ступінь вищої освіти)

на тему:

«Автоматизація контролю успішності студента засобами web-технологій»

|  |  |  |  |
| --- | --- | --- | --- |
| Виконав: студент | 4 | курсу, групи | 402-ТК |
| напряму підготовки (спеціальності) |
| 123 Комп’ютерна інженерія |
| (шифр і назва напряму підготовки, спеціальності) |
| Олефіренко В.С. |
| (прізвище та ініціали) |
| Керівник | д.т.н., професор Ляхов О.Л. |
|  | (прізвище та ініціали) |
| Рецензент |  |
|  | (прізвище та ініціали) |

м. Полтава – 2021

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**«ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»**

**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ**

**КАФЕДРА КОМП’ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ**

**КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА**

**спеціальність 123 «Комп’ютерна інженерія»**

**на тему:**

**«Автоматизація контролю успішності студента засобами web-технологій»**

**Студента групи 402-ТК Олефіренка Владислава Сергійовича**

Керівник роботи:

доктор технічних наук,

професор Ляхов О.Л.

Завідувач кафедри:

кандидат технічних наук,

доцент Головко Г.В.

Полтава – 2021

**РЕФЕРАТ**

Загальний обсяг текстової частини дипломної роботи складає 93 сторінок формату А4. Вона складається з переліку скорочень, умовних позначень, вступу, трьох розділів, висновків, списку використаних джерел та одного додатку. Робота містить 21 рисунки, використано 23 науково-технічних джерел.

Дипломна робота бакалавра присвячена автоматизації контролю успішності студента із застосуванням сучасних веб-технологій.

У першому розділі дипломної роботи розглядаються теоретичні основи розробки електронного журналу засобами web-технологій.

У другому розділі розглядаються методи та засоби веб-технологій для реалізації програмної системи.

У третьому розділі наведена реалізація та опис розробленого веб-додатка. Приведені приклади функціонування веб-ресурсу, можливості взаємодії з користувачем.

**Об’єкт дослідження:** процес проектування веб-додатку.

**Предмет дослідження:** засоби розробки веб-додатку.

**Мета роботи:** розробка функціонального веб-додатку.

**Методи дослідження:** автоматизація контролю успішності студента із допомогою функціонального веб-додатку.

**Отримані результати:** реалізовано веб-додаток для автоматизації контролю упішності студента.

**Ключові слова:** веб-технології, веб-сторінка, розробка додатку, веб-додаток, електронний журнал, API, JavaScript, Node.js, архітектура, HTML, CSS, додаток.

**ABSTRACT**

The total volume of the text part of the thesis is 93 A4 pages. It consists of a list of abbreviations, symbols, introduction, three sections, conclusions, a list of sources used and one appendix. The work contains 21 drawings, 23 scientific and technical sources were used.

The thesis of the bachelor is devoted to the automation of student performance control with the use of modern web technologies.

In the first section of the thesis the theoretical bases of e-journal development by means of web-technologies are considered.

The second section discusses the methods and tools of web technologies for the implementation of the software system.

The third section provides the implementation and description of the developed web application. Examples of functioning of a web resource, possibilities of interaction with the user are resulted.

**Object of study:** web application design process.

**Subject of study:** development tools web application.

**The goal of the work:** development of a functional web application.

**Research methods:**automation of student performance monitoring with the help of a functional web application.

**The results obtained:**implemented a web application to automate student performance control.

**Keywords:** web technology, web page, application development, web application, ezine, API, JavaScript, Node.js, architecture, HTML, CSS, application.

**ЗМІСТ**

[ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 7](#_Toc75297919)

[ВСТУП 9](#_Toc75297920)

[РОЗДІЛ 1 РОЗРОБКА ЕЛЕКТРОННОГО ЖУРНАЛУ ЗАСОБАМИ WEB-ТЕХНОЛОГІЙ 11](#_Toc75297921)

[1.1. Загальна характеристика веб-технологій 11](#_Toc75297922)

[1.1.1. Структура і принципи Всесвітньої мережі 11](#_Toc75297923)

[1.2. Поняття веб-додатків 12](#_Toc75297924)

[1.2.1. Принципи роботи веб-додатків 14](#_Toc75297925)

[1.2.2. Типи додатків 15](#_Toc75297926)

[1.3. Базові технології побудови веб-додатків 16](#_Toc75297927)

[1.4. Вимоги до структури і функціонування веб-додатка для електронного журналу 18](#_Toc75297928)

[Висновок до розділу 1 20](#_Toc75297929)

[РОЗДІЛ 2 МЕТОДИ ТА ЗАСОБИ WEB-ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ 21](#_Toc75297930)

[2.1. Вибір архітектури програмного комплексу 21](#_Toc75297931)

[2.2. Опис інструментів розробки 22](#_Toc75297932)

[2.2.1. Мова програмування JavaScript 23](#_Toc75297933)

[2.2.2. Application Programming Interface 25](#_Toc75297934)

[2.2.3. Серверна платформа Node.js 26](#_Toc75297935)

[2.2.4. База даних NeDB 28](#_Toc75297936)

[2.2.5. HyperText Markup Language 29](#_Toc75297937)

[2.2.6. Cascading Style Sheets 32](#_Toc75297938)

[2.3. Обґрунтування вибору програмної реалізації 36](#_Toc75297939)

[Висновок до розділу 2 38](#_Toc75297940)

[РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА ОПИС РОЗРОБЛЕНОГО ВЕБ ДОДАТКА 39](#_Toc75297941)

[3.1. Внутрішня реалізація веб-додатка 39](#_Toc75297942)

[3.2. Аутентифікація і авторизація користувачів на сервері 41](#_Toc75297943)

[3.3. API сервера 42](#_Toc75297944)

[3.4. Загальні відомості про електронний журнал 45](#_Toc75297945)

[3.4.1. Авторизація в електронному журналі 46](#_Toc75297946)

[3.4.2. Ролі користувача 47](#_Toc75297947)

[3.4.3. Зміна пароля і вихід 47](#_Toc75297948)

[3.4.4. Початок навчального року 48](#_Toc75297949)

[3.5. Робота в ролі студента 48](#_Toc75297950)

[3.6. Робота в ролі батьків 50](#_Toc75297951)

[3.7. Робота в ролі викладача 50](#_Toc75297952)

[3.8. Робота в ролі куратора 60](#_Toc75297953)

[3.9. Робота в ролі адміністратора 65](#_Toc75297954)

[3.10. Головний адміністратор 71](#_Toc75297955)

[Висновок до розділу 3 75](#_Toc75297956)

[ВИСНОВКИ 76](#_Toc75297957)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 77](#_Toc75297958)

[ДОДАТКИ 79](#_Toc75297959)

**ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ**

CSS – каскадні таблиці стилів.

JS – Java Script.

URL – уніфікований локатор ресурсів.

HTML – мова розмітки гіпертексту.

XML – розширювана мова розмітки.

HTTP – протокол передачі гіпертексту.

API – інтерфейс прикладного програмування.

DOM – Об’єктна модель документа.

WWW – Всесвітня мережа.

AJAX – асинхронний JavaScript и XML.

HTML – мова розмітки гіпертексту.

CSS – Каскадні таблиці стилів.

JS – JavaScript.

JSON – запис об’єктів JavaScript.

URI – Уніфікований ідентифікатор ресурсів.

DNS – Система доменних імен.

IP – ідентифікатор (унікальний числовий номер) мережевого рівня.

SVG – масштабована векторна графіка.

CGI – загальний інтерфейс шлюзу.

jQuery – JavaScript-бібліотека з відкритим кодом.

CMS – Система керування вмістом.

SEO – Пошукова оптимізація сайту.

XML – Розширювана мова розмітки.

CSV – значення, розділені комою.

W3C – Консорціум Всесвітнього павутиння.

NeDB – вбудована база даних для проектів на Node.js.

SOAP – простий протокол доступу до об’єктів.

SOA – сервісно-орієнтована архітектура.

REST – передача репрезентативного стану.

ROA – Рентабельність активів.

RDF – технологія семантичної павутини.

DOM – Об’єктна модель документа.

NPM – менеджер пакунків.

NoSQL – не тільки мова структурованих запитів.

MongoDB – документо-орієнтована система керування базами даних.

АSСII – Американський стандартний код для інформаційного обміну.

БД – база даних.

СУБД – система управління базою даних.

СКБД – система контролю базою даних.

**ВСТУП**

**Актуальність.** Українська система освіти реорганізовується та показує досить хороші успіхи. Але, на сьогодні, є багато освітянських аспектів, які потребують покращення. В основному це стосується того, що учні/студенти не володіють достатньою інформацією про власну успішність та свій рейтинг, що знижує мотивацією до навчання, яка необхідна для кращого засвоєння навчального матеріалу.

Викладачі та вчителі в сучасній системі освіти витрачають багато часу на заповнення різноманітних паперових документів, зокрема журналів, замість того, щоб більше часу приділяти процесу навчання. Поточна оцінка в журналі є важливим фактором успішності учня/студента [9]. Підсумкова/ модульна оцінка в освітніх закладах розглядається як істотний фактор прогнозу успішності знань в рамках річної/семестрової оцінки по предмету/дисципліні [19]. Саме тому систематичне виставлення оцінок у журнали необхідно, але робота із паперовими носіями потребує набагато більше часу ніж ведення електронного журналу.

На сьогоднішній день в Україні є велика кількість закладів освіти, мета яких – якісно та в повній мірі надати учням/студентам освітній рівень, відповідно до рівня навчального закладу. Станом на 2021 р. в Україні нараховується більше 15 тисяч закладів загальної середньої освіти та більше 600 закладів вищої освіти [20].

Україна, як і багато інших розвинених держав використовує модульну систему освіти. Модуль – це логічно завершена система теоретичних знань та фактичних умінь з даної навчальної дисципліни, адаптованих до індивідуальних особливостей суб’єктів учіння з визначеним оптимальним часом на організацію її засвоєння [3]. Розбиття навчального матеріалу на модулі покращує сприйняття нового матеріалу студентами. Введення модульної системи також робить систему оцінювання знань більш прозорою. Але, через це викладачам необхідно проводити розрахунок успішності та відвідуваності кожного студента в кінці кожного модуля, що може займати багато часу. Впровадження системи державних стандартів освіти також додає навантаження на працівників системи освіти, що підштовхує освітян до пошуку нових методів оптимізації роботи у навчальних закладах.

Впровадження системи державних стандартів освіти також додає навантаження на працівників системи освіти, що підштовхує освітян до пошуку нових методів оптимізації роботи у навчальних закладах.

**Мета роботи.** Вирішення згаданих проблем системи освіти можна частково знівелювати розробкою та впровадженням на основі web-технологій електронного журналу обліку успішності та відвідуваності студентів. Використання такого журналу дозволить зекономити багато часу викладачів, адже їм необхідно буде витрачати менше часу на аналіз успішності студентів у кінці модуля, а також доступ до нього може здійснюватися з будь-якого комп’ютера, підключеного до мережі. Викладачі зможуть якісніше використовувати цей час для того, щоб краще підготувати навчальний матеріал, ретельніше перевіряти роботи студентів, шукати індивідуальний підхід до кожного студентів та багато іншого. А це, у свою чергу, значно поліпшить рівень наданих освітянських послуг та підвищить якість знань із дисципліни.

**РОЗДІЛ 1
РОЗРОБКА ЕЛЕКТРОННОГО ЖУРНАЛУ ЗАСОБАМИ WEB-ТЕХНОЛОГІЙ**

* 1. **Загальна характеристика веб-технологій**

Поняття веб-технології є похідним від синонімічного ряду понять Web, World Wide Web, Всесітня павутина, веб, що визначають глобальний інформаційний простір, заснований на фізичній інфраструктурі Інтернету і протоколі передачі даних HTTP. Веб-технологія – це сукупність методів та програмно-технічних засобів, інтегрованих з метою ефективного опрацювання веб-ресурсів, які знаходяться у веб-просторі. Вузьке трактування поняття веб-технологій пов’язано з методами і засобами створення веб-сторінок з підтримкою мультимедіа, що поєднують у собі різні види інформації: текст, графіку, звук, анімацію й відео. Таке трактування веб-технологій охоплює базові сервіси Інтернету і не втрачає свого змісту і сьогодні. Проте в сучасних умовах, коли Інтернет є не тільки мультимедійною картинкою з текстом у веб-просторі, але потужним засобом комунікації, інтеграції, пошуку веб-ресурсів, надання різноманітних сервісів, поняття веб-технологій трактується ширше – як комплекс технічних, комунікаційних, програмних методів розв’язання завдань організації спільної діяльності користувачів із застосування мережі Інтернет.

* + 1. **Структура і принципи Всесвітньої мережі**

Всесвітню мережу утворюють мільйони веб-серверів мережі Інтернет, розташованих по всьому світу. Веб-сервер є програмою, що запускається на підключеному до мережі комп’ютері і використовує протокол HTTP для передачі даних. У простому вигляді така програма отримує по мережі HTTP-запит на певний ресурс, знаходить відповідний файл на локальному жорсткому диску і відправляє його по мережі комп’ютеру, що запитав. Складніші веб-сервери здатні динамічно формувати ресурси у відповідь на HTTP-запит. Для ідентифікації ресурсів (часто файлів або їх частин) у Всесвітній павутині використовуються одноманітні ідентифікатори ресурсів URI (англ. Uniform Resource Identifier). Для визначення місцезнаходження ресурсів в мережі використовуються одноманітні локатори ресурсів URL (англ. Uniform Resource Locator). Такі URL-локатори поєднують в собі технологію ідентифікації URI і систему доменних імен DNS (англ. Domain Name System) – доменне ім’я (або безпосередньо IP-адрес в числовому записі) входить до складу URL для позначення комп’ютера (точніше – одного з його мережевих інтерфейсів), який виконує код потрібного веб-сервера.

Для перегляду інформації, отриманої від веб-сервера, на клієнтському комп’ютері застосовується спеціальна програма – веб-браузер. Основна функція веб-сервера браузера – відображення гипертекста.

Користувач формує запит, ввівши адресу ресурсу в рядок адреси браузера або натиснувши посилання на веб-сторінці. Браузер пересилає запит на сервер, де відбувається пошук заданого об’єкту. Якщо файл знайдено – він надсилається у відповідь, інакше надсилається повідомлення про помилку.

Якщо веб-сторінка містить додаткові об’єкти, такі як зображення, аркуші стилів чи програмні скрипти, відбувається дозавантаження цих елементів та веб-сторінка відображається у вікні браузера.

веб-сторінки, тематично пов’язаних між собою і розроблених як єдине ціле, називають веб-сайтом

Будь-який веб-сайт з елементами інтерактиву – це веб-додаток.

* 1. **Поняття веб-додатків**

Веб-додаток являє собою веб-сайт, на якому розміщені сторінки з частково або повністю несформованим вмістом. Остаточний вміст формується тільки після того, як відвідувач сайту запросить сторінку з веб-серверу. У зв’язку з тим що остаточний вміст сторінки залежить від запиту, створеного на основі дій користувача, така сторінка називається динамічною. Спектр використання веб-додатків досить широкий. Веб-дизайн та додатки передбачають стандарти для створення та надання веб-сторінок, включаючи HTML, CSS, SVG, API пристроїв та інші технології для веб-додатків.

Використання веб-додатка замість типової веб-сторінки приносить користь для користувача, а саме:

* Веб-додатки дозволяють відвідувачам швидко і легко знаходити необхідну інформацію на веб-сайтах з великим об’ємом інформації.
* Веб-додатки дозволяють збирати, зберігати і аналізувати дані, отримані від відвідувачів сайту.
* Веб-додаток може використовуватися для оновлення веб-сайтів з динамічно створюваним вмістом.

Перший вид додатків дозволяє здійснювати пошук у вмісті, упорядковувати вміст і переміщатися по ньому зручним для відвідувачів способом.

Другий тип додатків доволі довго використовував метод при якому дані, введені в HTML-форми, відсилалися для обробки CGI-додатків або спеціально призначеним працівникам у вигляді повідомлень електронної пошти. Веб-додаток дозволяє зберігати дані безпосередньо в базі даних, а також отримувати дані і формувати звіти на основі отриманих даних для аналізу. Наприклад можна привести інтерактивні сторінки банків, сторінки для контролю товарних запасів, соціологічні дослідження та опитування, а також форми для зворотного зв’язку з користувачами.

Результатом виконання дипломного проекту повинен стати третій тип веб-додатків. Даний тип характерний тим, що інформація яка виводиться на екран, створюється динамічно.

* + 1. **Принципи роботи веб-додатків**

Для початку необхідно визначити різницю між веб-сайтом та веб-додатком. Веб-сайт це в першу чергу щось інформаційне і статичне: візитка компанії, сайт рецептів, міський портал або Wiki. Набір підготовлених заздалегідь HTML-файлів, які лежать на віддаленому сервері і віддаються браузеру за запитом.

Веб-сайт показує статичні або динамічні дані, які переважно надсилаються з сервера лише користувачеві, тоді як веб-додаток обслуговує динамічні дані з повною двосторонньою взаємодією.

Веб-сайт показує по суті ті ж дані. Деякі з них можуть бути динамічними, але це, як правило, справа в одну сторону – ви лише споживач.

Веб-додаток – двосторонній. Ви бачите дані, які не тільки динамічні, але часто також специфічні для вас. Ви можете працювати з цими даними через веб-додаток, щоб публікувати новий вміст або надсилати змістовні запити назад на сервер або через сервер третім особам (включаючи інших користувачів).

Наприклад:

• додаток для продажу акцій з даними про ціни та рахунок у режимі реального часу, що дозволяє здійснювати операції в режимі реального часу;

• додаток для редагування фотографій;

• генератор списку бажань на весілля, який можна поділитися з вашими гостями;

• веб-гра зі стійким світом;

• інтерфейс публікації веб-сайту Wordpress (де ви пишете дописи в блозі та загалом керуєте сайтом) – це веб-додаток, але сам блог ні;

• Youtube після входу (щоб ви могли розміщувати коментарі чи відео);

Що таке веб-сайт, загально прийнято, але те, що також не відноситься до веб-додатків, є трохи сірою зоною; Наприклад Google Maps – це не веб-додаток (це динамічна веб-сторінка), тоді як пошта Google – справжня веб-програма.

* + 1. **Типи додатків**

Розробка веб-додатків не обмежується лише смартфонами або планшетами. Він призначений для роботи у будь-якому браузері, роботі на настільних комп’ютерах, ноутбуках або мобільних пристроях. Вирізняють такі типи веб-додатків:

**Статичний веб-додаток**. Цей тип веб-додатків відображає дуже мало вмісту і не є досить гнучким. Зазвичай вони розробляються в HTML та CSS. Однак анімовані об’єкти, такі як банери, GIF, відео тощо, також можуть бути включені і показані в них. Вони також можуть бути розроблені за допомогою jQuery та Ajax.

Крім того, змінити вміст статичних веб-додатків непросто. Для цього спочатку потрібно завантажити HTML-код, потім змінити його і, нарешті, знову завантажити його на сервер. Ці зміни можуть бути внесені лише вебмайстром.

**Динамічний веб-додаток.** Динамічні веб-додатки набагато складніші на технічному рівні. Вони використовують бази даних для завантаження даних, а їх вміст оновлюється кожного разу, коли користувач звертається до них. Зазвичай вони мають панель адміністрування (називається CMS), звідки адміністратори можуть виправляти або змінювати вміст програми, включаючи текст та зображення.

Для динамічної розробки веб-додатків можна використовувати безліч різних мов програмування. JavaScript – це найпоширеніша мова, яка використовуються для цієї мети.

У такому додатку оновлення вмісту дуже просте, і сервер навіть не потребує доступу до нього, змінюючи його. Крім того, це дозволяє реалізувати безліч функцій, таких як форуми або бази даних. Дизайн, окрім вмісту, може бути змінений відповідно до уподобань адміністратора.

**Інтернет-магазин або електронна комерція.** Якщо веб-додаток – це інтернет-магазин чи магазин, його розробка, швидше за все, буде схожа на розвиток електронної комерції або веб-сайт електронної комерції. Цей процес розробки додатків є складнішим, оскільки він повинен включати електронні платежі за допомогою кредитних карток, PayPal або інших способів оплати. Розробник також повинен створити панель управління для адміністратора. Він буде використовуватися для перерахування нових продуктів, їх оновлення або видалення та управління замовленнями та платежами.

**Веб-додаток порталу.** Під порталом маємо на увазі своєрідну програму, в якій ми отримуємо доступ до декількох її розділів або категорій через домашню сторінку. Ці програми можуть включати багато речей: форуми, чати, електронну пошту, браузери, області, до яких можна отримати реєстрацію, останній вміст тощо.

**Анімаційний веб-додаток.** Неминуче пов’язаний з технологією FLASH. Цей підхід програмування дозволяє представити контент з анімованими ефектами. Це також дозволяє більш інноваційні та сучасні конструкції та є однією з найбільш широко використовуваних технологій дизайнерів та креативних директорів. Недолік, властивий розробці анімованих веб-додатків, полягає в тому, що такий вид технології не підходить для цілей розміщення в Інтернеті та оптимізації SEO, оскільки пошукові системи не можуть правильно читати інформацію, яку вони містять.

Вміст повинен постійно оновлюватися, коли мова йде про розробку веб-додатків, тому встановлення системи управління вмістом (CMS) є серйозним варіантом. Адміністратор може використовувати цю CMS для впровадження змін та оновлень. Ці менеджери вмісту інтуїтивно зрозумілі та дуже прості в обробці.

* 1. **Базові технології побудови веб-додатків**

Веб-розробка має величезний набір правил та методик, про які повинен знати кожен розробник веб-сайту та веб-додатку. Якщо ви хочете, щоб веб-сайт виглядав і працював так, як ви і хотіли, вам потрібно ознайомитися з веб-технологіями, які допоможуть вам досягти поставленої мети. Розробка програми або веб-сайту зазвичай зводиться до знання трьох основних мов:

• JavaScript;

• CSS;

• HTML;

Оскільки комп’ютери не можуть спілкуватися між собою так, як це роблять люди, їм потрібні коди. Веб-технології – це мови розмітки та мультимедійні пакети, які комп’ютери використовують для спілкування.

При створенні веб-додатків, доволі часто використовуються API. API (інтерфейс програмування додатків) дозволяє іншим розробникам використовувати частину функцій програми без спільного використання коду. Кінцеві точки відкриваються розробниками, тоді як API може контролювати доступ за допомогою ключа API.

Дані зберігаються в структурі, що називається формат даних.

JSON – JavaScript Object Notation – це синтаксис для зберігання та обміну даними (як і XML). В даний час це найпопулярніший формат даних там.

XML – в основному використовується системами Microsoft, він раніше був найпопулярнішим форматом даних.

CSV – це формати даних комами; наприклад, дані Excel.

Кожен користувач програми називається клієнтом. Клієнтами можуть бути комп’ютери, мобільні пристрої, планшети тощо. Зазвичай кілька клієнтів взаємодіють з одним і тим же додатком, що зберігається на сервері.

Код програми зазвичай зберігається на сервері. Клієнти роблять запити до серверів. Потім сервери відповідають на ці запити після збору запитуваної інформації.

* 1. **Вимоги до структури і функціонування веб-додатка для електронного журналу**

До веб-додатку було поставлено перелік вимог:

• забезпечення надійності;

Основним критерієм збереження інформації в незмінному вигляді є її надійність зберігання. Використання розмежування доступу до об’єктів в цьому випадку буде невід’ємним атрибутом політики безпеки.

• простота інтерфейсу;

Зручність інтерфейсу необхідно для швидкої навігації і є важливою частиною при проектуванні системи.

• кросбраузерність;

Сторінки повинні однаково відображатися у різних браузерах (Mozilla Firefox, Google Chrome, Opera, Internet Explorer, Safari).

• гнучкість верстки;

Можливість легко додавати / видаляти інформацію на сторінці.

• валідність;

Відповідність вихідного коду сайту нормам і правилам, описаним Консорціумом Всесвітньої Павутини (W3C).

• адаптивність;

Забезпечення правильного відображення сайту на різних пристроях, що мають з’єднання з Інтернет, що динамічно підлаштовується під задані розміри вікна браузера.

• приємний дизайн;

При виборі оформлення ресурсу і продумуванні його дизайну варто пам’ятати про розумну міру.

Також, веб-додаток має виконувати свою пряму функцію як електронного журналу – облік відвідування та успішності студентів.

Для цього, викладач повинен зайти на інформаційний ресурс. Додаток має надати викладачу список груп на вибір, заняття яких записані в журнал. Список має складатися з груп, заняття в яких проводиться в поточний день (в групі є заняття в поточний день тижня та ще триває період навчання групи). При бажанні, список можна розширити, щоб він складав усі групи і навіть ті, навчання в яких на поточний день не заплановано.

Після вибору бажаної групи, має бути зображено пронумерований список групи. Якщо в програму вже було записано дані за сьогодні, то програма має відобразити ці дані (оцінки та відвідуваність). Після відзначення відвідуваності та виставлення оцінки за заняття, викладач вносить введені дані в програму.

Для того, щоб мати можливість вводити данні в групу, необхідно мати на додатку можливість створити групу. Було б добре й мати можливість її редагувати та видаляти з сайту.

Для створення групи, необхідно натиснути на відповідну кнопку, внести назву нової групи та натиснути на кнопку створення. В ході розробки було зроблено рішення, що оптимально буде створювати «порожню групу», яка не матиме днів тижня навчання, яка матиме нелогічний термін навчання (з дня її створення по той ж самий день) та без студентів і, відповідно, записів присутності та успішності.

Надалі потрібна можливість редагувати щойно створену групу (або вже існуючу), змінюючи назву, дні навчання групи та її період навчання. Також необхідно додавати/вилучати студентів з групи.

Необхідно також, щоб на сайті була функція перегляду записів журналу певної групи за певний вибраний період. Також необхідно реалізувати функцію редагування запису в групі за минулий день.

Не менш важливою є й можливість перегляду статистики студента. Дана статистика повинна включати в себе середній бал, кількість відсутностей/присутностей за певний вибраний період. Бажано ще й надати графічне представлення цих статистичних даних (графіки).

Для реалізації поставленої задачи треба обрати найбільш раціональний шлях для вирішення поставлених задач. У ході роботи було проведено аналіз, який допоміг обрати технології та методи програмної реалізації для кращого виконання роботи.

Аналізуючи поставлену задачу та методи її вирішення, було вирішено розроблювати програмний комплекс на основі веб-технологій. Головною перевагою веб-застосунку перед іншими варіантами є його універсальність і можливість використання на будь-яких пристроях без портування на цільову операційну систему (браузер і його віртуальна машина виступає як цільова універсальна операційна система і комп’ютер).

**Висновок до розділу 1**

Отже у висновку хотілося б зазначити наступне: розроблений веб-додаток електронного журналу повинен бути в першу чергу зручним для користувача, багатофункціональним, інтуїтивно зрозумілим, щоб користувач міг розібратися з її базовими функціями без попереднього навчання користування чи прочитання інструкції.

**РОЗДІЛ 2
МЕТОДИ ТА ЗАСОБИ WEB-ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ**

**2.1. Вибір архітектури програмного комплексу**

Архітектура програмного забезпечення – це структура програмного застосунку, що складається з певних програмних компонентів, видимих зовні властивостей цих компонентів та зв’язками між ними.

Архітектуру програмного забезпечення можна уявити у вигляді розробки стратегії – визначенням глобальних обмежень та вимог, що необхідно застосувати до системи на етапі її проектування. Детальне проектування – це не що інше як визначення певних локальних обмежень для проекту. Мова йде про архітектурні моделі, шаблони проектування та ідіоми програмування.

Для реалізації поставленої задачі було вирішено використовувати триланкову архітектуру, яка складається з таких компонентів: сервер, база даних і клієнт. Схема даної архітектури зображена на рисунку 2.1.



Рисунок 2.1 – Триланкова архітектура програмного комплексу

Головним центром програмного комплексу є сервер. Основною функцією серверної частини є ідентифікація користувача для надання доступу до головного функціоналу веб-застосунку. Єдиним зв’язком між користувачем та базою даних – є сервер. Це зроблено для того, щоб унеможливити пошкодження даних та їх використання не за призначенням. Для того, щоб користуватися основним функціоналом системи, потрібно бути авторизованим у програмі, тому дана логіка реалізуються на рівні серверу, тому що на рівні користувача можлива підміна прав доступу та інші методи неконтрольованого доступу до даних.

Під час користування програмою, користувач взаємодіє з клієнтським додатком, яким є веб-сайт в даному випадку. На рівні користувача відбувається налаштування програми та перегляд результатів роботи реалізованого інтерфейсу. Також на користувацькому рівні відбувається попередня обробка даних перед відправленням на сервер і також опрацювання результатів від сервера. Ще на цьому рівні відбувається перший етап аутентифікації користувача для обмеження неконтрольованого доступу до програмного застосунку.

**2.2. Опис інструментів розробки**

Програмний комплекс побудований за принципами триланкової архітектури побудови програм. Кожен рівень цієї архітектури реалізовано з використанням різних технологій і головною ціллю було створення мультиплатформного рішення з використанням відкритих технологій.

Для клієнтського рівня було використано такий набір технологій: мова програмування JavaScript, стандартна мова розмітки для створення веб-сторінок і веб-додатків HTML5, скриптова мова CSS.

Для серверного рівня було використано мова програмування JavaScript, прикладний програмний інтерфейс – API, серверна платформа Node js.

Для рівня бази даних було обрано NeDB.

**2.2.1. Мова програмування JavaScript**

JavaScript – це легка об’єктно-орієнтована мова програмування з функціями першого класу, найвідоміша скриптова мова для веб-сторінок, але також використовується у багатьох не браузерних середовищах. JavaScript являє собою прототипно-орієнтовану, мультипарадигмальну мову сценаріїв, яка підтримує динамічні, об’єктно-орієнтовані, імперативні і функціональні стилі програмування.

JavaScript запускається на стороні клієнта Інтернету, який може використовуватися для створення сценарії поведінки веб-сторінки при настанні будь-яких подій. JavaScript – потужна скриптова мова, що широко використовується для контролювання поведінки веб-сторінок.

JavaScript використовується практично на всіх сучасних веб-сайтах. У всіх сучасних браузерах і клієнтських пристроях – настільних комп’ютерах, ігрових консолях, планшетах і смартфонах – є інтерпретатори JavaScript, що робить його найбільш поширеною мовою в історії програмування. Він входить в базову тріаду технологій, які необхідно знати всім розробникам веб-додатків.

JavaScript підтримує кілька стилів програмування, зокрема об’єктно-орієнтований, функціональний та імперативний. Cеред його основних архітектурних рис можна виділити наступні:

• Динамічна типізація. Такий підхід означає, що змінна зв’язується з типом не в момент її об’явлення, а в момент присвоєння їй значення. Це має як свої недоліки, так і переваги. Зокрема, це суттєво полегшує написання скриптів, але водночас, такий прийом може часто призводити до помилок в коді, які непросто виявити. Також через втрати процесорного часу на динамічну перевірку типу змінної можливі втрати у продуктивності програми [15].

• Слабка типізація. Дана риса означає, що змінні достатньо просто конвертуються з одного типу в інший. Як і у випадку з динамічною типізацією, це має як позитивні, так і негативні сторони. Такий підхід суттєво покращує гнучкість та віртуозність коду, оскільки, наприклад, при виклику певної функції може бути виконане автоматичне приведення типів її параметрів. З іншої сторони, це також може призвести до появи помилок та незапланованої поведінки програми, які достатньо складно відслідкувати [16].

• Автоматичне керування пам’яттю. Дана характеристика передбачає існування спеціального процесу, так званого «garbage collector», який періодично звільняє пам’ять, видаляючи з неї об’єкти, які вже не будуть використовуватись додатком. При ручному керуванні пам’яттю програміст мав би власноруч слідкувати за видаленням відпрацьованих об’єктів. Це могло б призвести до суттєвих порушень у роботі програми, утворення «висячих» посилань та втрат пам’яті, що може призвести до збоїв у роботі, а також до значної втрати продуктивності.

• Прототипне програмування – це підстиль об’єктно-орієнтованого програмування, в якому відсутнє поняття класу як такого, а наслідування відбувається шляхом клонування прототипу вже існуючого екземпляра об’єкта. Такий підхід дозволяє уникнути надлишкової концентрації на таксономії класів та зв’язків між ними і зосередити увагу на поведінці невеликої кількості екземплярів, які розглядаються як базові об’єкти і далі використовуються для створення інших об’єктів [17].

• Підтримка об’єктів першого класу. Об’єкти першого класу – це елементи, які можуть бути повернені або передані функції як параметр, присвоєнні змінній та ін. Ця характеристика випливає з підходу прототипного програмування.

**2.2.2. Application Programming Interface**

Для реалізації серверу було використано API, який є інтерфейсом прикладного програмування.

Прикладний програмний інтерфейс API – це набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення. Спрощено – це набір чітко визначених методів для взаємодії різних компонентів. Програмний інтерфейс API надає розробнику засоби для швидкої розробки програмного забезпечення. Програмний інтерфейс API може бути створеним для веб-базованих систем, операційних систем, баз даних, апаратного забезпечення, програмних бібліотек.

Одним з найпоширеніших призначень API є надання набору широко використовуваних функцій, наприклад, для малювання вікна чи іконок на екрані. Програмісти використовують переваги API у функціональності, таким чином їм не доводиться розробляти все з нуля. Програмний інтерфейс API є абстрактним поняттям – програмне забезпечення, що пропонує деякий API, часто називають реалізацією даного API. У багатьох випадках API є частиною набору розробки програмного забезпечення, водночас, набір розробки може включати як API, так і інші інструменти/апаратне забезпечення, отже ці два терміни не є взаємозамінювані.

При використанні прикладного програмного інтерфейсу в контексті веб-розробки, як правило, API визначається набором повідомлень запиту HTTP, також визначається структура повідомлень-відповідей, зазвичай у розширенні мови розмітки XML або в форматі об’єктного запису JavaScript (JSON) [7]. У той час як прикладний програмний інтерфейс у Web історично був практично синонімом для веб-служби, останнім часом тенденція змінилась (так званий Web 2.0) на відхід від Simple Object Access Protocol (SOAP) на основі веб-сервісів і сервіс-орієнтованої архітектури (SOA) на більш прямі передачі репрезентативного стану (REST) стилів веб-ресурсів та ресурсів орієнтованої архітектури (ROA). Частина цієї тенденції пов’язана з рухом семантичного веб-ресурсу до опису платформ (RDF), концепції розвитку веб-технологій інженерних онтологій. Прикладні програмні інтерфейси у веб, що дозволяють комбінуватися декількома прикладними програмними інтерфейсами в нові додатки називають гібридними.

**2.2.3. Серверна платформа Node.js**

Node.js – платформа з відкритим кодом, що використовується для створення клієнтських та серверних застосунків [16].

Один з найважливіших плюсів платформи – це асинхронність в поєднанні з подієвим підходом. На відміну від потокового, подієво-орієнтоване програмування засноване на зовнішніх діях. Тобто виконання програми залежить від дій користувача. Таке рішення спрощує програмування інтерактивних додатків, тобто заснованих на роботі з використанням введення-виведення. Це можуть бути онлайн-чати або ігрові програми, барвисті інтерактивні веб-сервіси або всілякі калькулятори, рейтинги, голосування і т.д.

Крім того, до переваг платформи відносять:

* простий і широко відомий JavaScript. Звичайно, платформа передбачає власні інструменти та особливості, наприклад, тут немає браузерних API, cookie або DOM, однак присутні власні бібліотеки. В основному використовуються можливості і синтаксис всім звичного JavaScript;
* велика стандартна бібліотека. Платформа від самого початку мала широкий набор можливостей, а в нових версіях бібліотека поповнюється і поліпшується;
* велика кількість зовнішніх бібліотек і готових модулів. Використання пакетного менеджера NPM дозволяє постійно розвивати екосистему Node. Сьогодні число пакетів в ньому перевалило за цифру 500 тисяч і постійно зростає.

Ще однією особливістю Node.js є те, що він орієнтований на роботу з NoSQL базами даних, зокрема NeDB. Такий підхід дозволяє ще покращити масштабованість продукту та швидкість його роботи.

З самої назви платформи «node» – вузол, можна побачити, що вона акцентує увагу на можливість побудови систем із великої кількості невеликих розподілених обчислювальних вузлів, які мають можливість обмінюватися даними між собою.

Важливо розуміти, що на відміну від звичайного браузерного JavaScript, із застосуванням Node додаток може звертатися до широкого переліку глобальних об’єктів, в тому числі document або window. В результаті програма може звертатися до жорсткого диску і файлових систем користувача, а також до бібліотек і програмних рішеннь, що знаходяться на комп’ютері, що помітно розширює можливості взаємодії.

Найчастіше цю платформу використовують для створення веб-сервісів, що вимагають інтенсивного обміну інформацією з користувачами, в тому числі, для реалізації чатів, систем спільної роботи, соціальних мереж і т.д. Сервер на Node.js ніколи не виявиться потужнішим, ніж потрібно. Вся суть архітектури Node – у її мінімалізмі. При цьому серверну частину додатку можна масштабувати в залежності від потреб проекту. Багато програм, створені на Node.js, складаються з серверної і клієнтської частин.

Для розуміння причин такого використання варто трохи докладніше зупинитися на асинхронності і можливостях подієво-орієнтованої парадигми. Суть програмування при цьому зводиться до того, що ви вказуєте програмі, які дії вона повинна виконати, а також яку функцію вона повинна викликати після виконання цих дій. Поки Node.js чекає виконання коду і виклику функції, вона може виконувати інші завдання. При цьому потік в очікуванні запиту блокуватися не буде.

Таким чином, програма може звернутися із запитом до бази даних сервера, а поки очікується відповідь, обробити інші запити. В результаті одночасно обробляються тисячі з’єднань, кожне з яких витрачає мало часу і ресурсів. Це помітно прискорює обмін інформацією з сервером і, як підсумок, роботу браузерного додатку.

**2.2.4. База даних NeDB**

NeDB – це легкий вбудований СУБД для документів, написаний на JavaScript. Він підтримує середовища Node.js, nw.js, Electron та веб-браузер. Він розроблений, щоб бути частково сумісним з API запитів на основі JSON від MongoDB. NeDB корисний для зберігання невеликих обсягів даних у пам’яті. Коли обсяг даних перевищує межі того, що NeDB може ефективно вмістити, перехід на MongoDB має бути простим, оскільки він використовує той самий API.

У травні 2013 року Луї Чатріот розпочав розробку полегшеної вбудованої системи даних в базі даних, яка використовувала підмножину API MongoDB. Він був завершений у 2017 році, і зараз він знаходиться на GitHub, де приватні особи можуть внести свій вклад у проект, виправивши кілька помилок, але додаткові функції наразі не реалізовані. Однак кілька окремих команд роздрібнили копії оригінальної бази даних NeDB і почали реалізовувати власні ідеї та додаткові функції, які спочатку не призначені початковим творцем.

NeDB повністю написаний на Javascript, і оскільки Javascript за своєю суттю є однопотоковою, в системі немає паралелізму. Крім того, кожен процес отримує власну копію бази даних, що означає, що він не може синхронізувати подання даних, які мають два різні процеси.

NeDB підтримує індексацію та допомагає забезпечити достатній приріст швидкості. Це дозволяє індексувати будь-яке поле, включаючи поля у вкладених документах. Індекс можна створити у будь-який момент, навіть після вставки даних. Однак це найефективніше, якщо це робиться під час запуску програми, оскільки індекс створюється синхронно. Час створення індексу досить швидкий, тому це не надто впливає на час запуску. Крім того, індекси дозволяють базі даних виконувати швидке читання та запис.

NeDB не підтримує ведення журналу, оскільки всі дані зберігаються в пам’яті. Однак було створено реєстратор NeDB, за допомогою якого можна вставляти документи до читаної NeDB бази даних.

NeDB зберігає всі бази даних у пам’яті, щоб швидко виконувати операції пошуку та запиту. Зберігання всіх даних у пам’яті призводить до того, що NeDB працює добре лише на менших наборах даних.

NeDB використовує формат лише додавання, що означає, що всі оновлення та видалення будуть додані до кінця файлу даних. Це робиться з міркувань продуктивності. База даних автоматично ущільнюється щоразу, коли вона завантажується у програму. Можна виконати ручне ущільнення, але для цього потрібно, щоб все інше було призупинено, і це може зайняти деякий час. Тому більшості додатків цього робити не потрібно.

NeDB використовує вбудований системний підхід до бази даних. Система баз даних знаходиться у програмі під час її запуску, оскільки вона імпортується до програми як бібліотека. Крім того, система використовується через виклики API, які є однаковими з MongoDB.

**2.2.5. HyperText Markup Language**

HTML (Hypertext Markup Language – мова гіпертекстової розмітки) – це мова опису структури сторінок документів, яка дозволяє звичайний текст форматувати в абзаци, заголовки, списки та інші структури, створювати посилання на інші сторінки. Це текстова мова, в якій інструкції з форматування, що називаються тегами, вбудовані в розділи документа, які містять конкретну інформацію. Теги повідомляють браузерам, як форматувати і представляти інформацію на екрані.

Мова гіпертекстової розмітки HTML була запропонована Тімом Бернерсом-Лі у 1989 як один з компонентів технології розробки розподіленої гіпертекстової системи World Wide Web. Ідея гіпертекстової інформаційної системи полягає у тому, що користувач має можливість переглядати документи (сторінки тексту) у найбільш зручному для себе порядку, а не послідовно, як це прийнято при читанні книг. Досягається це шляхом створення спеціального механізму пов`язування різних сторінок тексту за допомогою гіпертекстових посилань.

Мова НТМL дозволяє визначити структуру електронного документа з поліграфічним рівнем оформлення. Результуючий документ може містити різноманітні елементи: ілюстрації, аудіо і відеофрагменти. Мова НТМL включає розвинені засоби для визначення кількох рівнів заголовків, шрифтових виділень, різних груп об’єктів та багато інших можливостей.

Важливим чинником, який вплинув на розвиток мови HTML, став її вибір за основу для гіпертекстової бази даних звичайного текстового файла, який можна створювати у будь-якому текстовому редакторі на будь-якій апаратній платформі у середовищі будь-якої операційної системи.

Таким чином, гіпертекстова база даних у концепції WWW – це набір текстових файлів, розмічених мовою HTML, яка визначає форму представлення інформації (розмітка) і структуру зв’язків цих файлів (гіпертекстові посилання).

За основу моделі розмітки документів у HTML прийнята тегова модель. Тегова модель описує документ як сукупність контейнерів, кожен з яких починається і закінчується тегами. Тобто документ НТМL є не чим іншим, як звичайним АSСII-файлом з доданими до нього керуючими НТМL-кодами (тегами).

Теги НТМL-документів в основному є простими і зрозумілими для використання, оскільки вони створені за допомогою загальновживаних слів англійської мови, зрозумілих скорочень і позначень.

НТМL-тег складається з імені, за яким може слідувати необов’язковий список атрибутів тегу. Текст тегу вміщується у кутові дужки (<І>). Найпростіший варіант тегу – ім’я, вміщене у кутові дужки, наприклад, <HEAD>. Для більш складних тегів характерна наявність різних атрибутів, які можуть мати конкретні значення, визначені для видозмінення функцій тегу.

Атрибути тегу слідують за ім’ям і відділяються один від одного одним або кількома пропусками. Порядок запису атрибутів у тегу не має значення. Значення атрибута слідує за знаком рівняння, який стоїть після імені атрибута. Якщо значення атрибута – одне слово або число, його можна вказати безпосередньо після знаку рівняння, не виділяючи додатково. Решту значень необхідно вміщувати у одинарні або подвійні лапки, особливо якщо вони містять декілька розділених пропусками слів.

Найчастіше НТМL-теги складаються з початкового і кінцевого компонентів, між якими розміщуються текст та інші елементи документа. Ім’я кінцевого тега ідентичне імені початкового тегу, але перед ім’ям ставиться коса риска (/) (наприклад, для тегу заголовка <ТIТLЕ> закриваючою парою буде </ТIТLЕ>). Кінцеві теги не містять атрибутів.

При використанні вкладених тегів їх слід закривати, починаючи з останнього і рухаючись до першого.

Деякі НТМL-теги не мають кінцевого компонента, оскільки є автономними елементами. Наприклад, тег зображення <IMG>, призначений для вставки зображення у документ, не має кінцевого компонента.

Для створення HTML-документа можна застосувати редактор ASCII (зокрема, Блокнот системи Windows). Такі редактори дозволяють вводити HTML-теги, не додаючи до створеного нічого додатково. Створення документа у такому редакторі дозволяє паралельно переглядати результат у програмі – браузері. Інший тип редакторів – візуальні HTML-редактори, наприклад, Microsoft FrontPage. Їх інтерфейс побудований за тим же принципом, що і інтерфейс текстового процесора, такого, як, наприклад, Word. Для роботи з візуальним редактором можна взагалі не володіти мовою HTML. Недоліком візуальних редакторів є те, що розмір створюваного ними HTML-документа у декілька разів більший, ніж документа, створеного звичайним Блокнотом системи Windows. В умовах низької пропускної здатності вітчизняних мереж цей недолік, який стосується швидкості завантаження сторінки (і, відповідно, вартості часу, який на це витрачається), є досить суттєвим недоліком (файл .htm, створений у WORD, в 4-9 разів більший, ніж файл аналогічного змісту, створений програмою «Блокнот»).

**2.2.6. Cascading Style Sheets**

Веб-сайти складаються з декількох окремих фрагментів, включаючи зображення, текст та різні документи. Ці документи включають не лише ті, до яких можуть бути пов’язані різні сторінки, наприклад файли PDF, але й документи, які використовуються для побудови самих сторінок, як HTML-документи для визначення структури сторінки та документи CSS (Cascading Style Sheet). щоб продиктувати вигляд сторінки.

CSS вперше був розроблений в 1997 році як спосіб веб-розробникам визначити візуальний зовнішній вигляд веб-сторінок, які вони створювали. Він мав на меті дозволити веб-професіоналам відокремити зміст та структуру коду веб-сайту від візуального дизайну, що було неможливим до цього часу.

Розмежування структури та стилю дозволяє HTML виконувати більшу функцію, на якій вона була заснована спочатку – розмітку вмісту, не турбуючись про дизайн та компонування самої сторінки, щось загальновідоме як «зовнішній вигляд» сторінки.

Сьогодні всі сучасні браузери підтримують усі рівні CSS 1, більшість рівнів CSS 2 і навіть більшість аспектів рівня 3 CSS. Коли CSS продовжує розвиватися та впроваджувати нові стилі, веб-браузери почали впроваджувати модулі, які пропонують нову підтримку CSS у цих веб-переглядачах та надають веб-дизайнерам нові потужні інструменти для роботи.

У минулих роках існували вибрані веб-дизайнери, які відмовилися використовувати CSS для розробки та розробки веб-сайтів, але ця практика майже не вийшла сьогодні з галузі. CSS зараз є широко використовуваним стандартом у веб-дизайні, і вам би важко було знайти того, хто сьогодні працює в галузі, хто не мав хоча б базового розуміння цієї мови.

Як уже згадувалося, термін CSS означає «Каскадний аркуш стилів». Давайте трохи розіб’ємо цю фразу, щоб більш повно пояснити, що роблять ці документи.

Слово «таблиця стилів» стосується самого документа (наприклад, HTML, CSS-файли – це лише текстові документи, які можна редагувати за допомогою різних програм). Аркуші стилів використовуються для оформлення документів вже багато років. Вони є технічними характеристиками для макета, будь то друковані чи онлайн. Дизайнери друку давно використовують аркуші стилів, щоб гарантувати, що їхні малюнки надруковані саме за їхніми характеристиками. Таблиця стилів для веб-сторінки виконує ту саму мету, але з додатковою функціональністю також розповідає веб-браузеру, як зробити документ, який переглядається. Сьогодні аркуші стилів CSS також можуть використовувати медіа-запити для зміни способу пошуку сторінки для різних пристроїв та розмірів екрана. Це надзвичайно важливо, оскільки дозволяє один документ HTML виводити по-різному відповідно до екрана, який використовується для доступу до нього.

Каскад – це справді особлива частина терміна «каскадний аркуш стилів». Аркуш веб-стилів призначений для каскаду через низку стилів на цьому аркуші, як річка над водоспадом. Вода в річці вражає всі скелі водоспаду, але лише ті, що знаходяться на дні, впливають саме там, де вода буде текти. Те саме стосується каскаду в таблицях стилів веб-сайту.

На кожну веб-сторінку впливає принаймні один аркуш стилів, навіть якщо веб-дизайнер не застосовує жодних стилів. Цей аркуш стилів – аркуш стилів користувальницьких агентів – також відомий як стилі за замовчуванням, які веб-браузер використовуватиме для відображення сторінки, якщо інші інструкції не надані. Наприклад, гіперпосилання за замовчуванням викладені в синій колір і вони підкреслені. Ці стилі надходять із таблиці стилів веб-браузера за замовчуванням. Якщо веб-дизайнер надає інші інструкції браузеру потрібно знати, які інструкції мають перевагу. Усі браузери мають свої власні стилі за замовчуванням, але багато з цих за замовчуванням (наприклад, сині підкреслені текстові посилання) поділяються на:

• зовнішні таблиці стилів (англ. stylesheet), найчастіше окремий файл або файли .css;

• внутрішні таблиці стилів, включені як частина документу або блоку;

• стилі для окремого елементу;

• локальний css-файл, вказаний користувачем для використання на сторінках і вказаний в налаштуваннях браузера (наприклад Opera);

• стандартний стиль переглядача, наприклад стандартні стилі для елементів, визначені браузером, використовуються коли немає інформації про стиль елемента або вона неповна.

Один і той самий HTML або XML документ може бути відображений по-різному залежно від використаного CSS. Стилі для відображення сторінки можуть бути:

• стилі автора (інформація надана автором сторінки):

o зовнішні таблиці стилів (англ. stylesheet), найчастіше окремий файл або файли .css;

o внутрішні таблиці стилів, включені як частина документу або блоку;

o стилі для окремого елементу;

• стилі користувача:

o локальний .css-файл, вказаний користувачем для використання на сторінках і вказаний в налаштуваннях браузера (наприклад Opera);

• стилі переглядача (браузера):

o стандартний стиль переглядача, наприклад стандартні стилі для елементів, визначені браузером, використовуються коли немає інформації про стиль елемента або вона неповна.

Стандарт CSS визначає порядок та діапазон застосування стилів, тобто, в якій послідовності і для яких елементів застосовуються стилі. Таким чином, використовується принцип «каскадності», коли для елементів вказується лише та інформація про стилі, що змінилася або не визначена загальнішими стилями.

Переваги CSS:

• інформація про стиль для усього сайту або його частин може міститися в одному .css-файлі, що дозволяє швидко робити зміни в дизайні та презентації сторінок;

• різна інформація про стилі для різних типів користувачів: наприклад великий розмір шрифту для користувачів з послабленим зором, стилі для виводу сторінки на принтер, стиль для мобільних пристроїв;

• сторінки зменшуються в об’ємі та стають більш структурованими, оскільки інформація про стилі відділена від тексту та має певні правила застосування і сторінка побудована з урахуванням їх;

• прискорення завантаження сторінок і зменшення обсягів інформації, що передається, навантаження на сервер та канал передачі. Досягається за рахунок того, що сучасні браузери здатні кешувати (запам’ятовувати) інформацію про стилі і використовувати для всіх сторінок, а не завантажувати для кожної.

CSS також може бути використаний для визначення того, як повинні виглядати веб-сторінки, коли вони переглядаються в інших медіа, ніж веб-браузері. Наприклад, ви можете створити аркуш стилю друку, який визначатиме спосіб друку веб-сторінки. Оскільки елементи веб-сторінки, такі як кнопки навігації або веб-форми, не матимуть ніякої мети на друкованій сторінці, лист стилю друку може використовуватися для «вимкнення» тих областей, коли сторінка друкується. Хоча насправді це не звичайна практика на багатьох сайтах, можливість створення аркушів стилів друку є потужним і привабливим.

CSS – це один з найпотужніших інструментів, який веб-дизайнер може вивчити, оскільки за допомогою нього ви можете вплинути на весь візуальний вигляд веб-сайту. Добре написані таблиці стилів можна швидко оновлювати і дозволяти сайтам змінювати пріоритет візуально на екрані, що, в свою чергу, показує цінність та зосередженість відвідувачів, без змін у необхідній розмітці HTML.

Головною проблемою CSS є те, що потрібно навчитися небагато – і коли браузери змінюються щодня, те, що працює добре сьогодні, може не мати сенсу завтра, коли нові стилі стають підтримкою, а інші відмовляються або випадають з користі з тієї чи іншої причини.

Оскільки CSS може каскадувати та комбінувати та враховуючи, як різні браузери можуть по-різному інтерпретувати та реалізовувати директиви, CSS може бути складнішим, ніж звичайний HTML. CSS також змінює браузери таким чином, що HTML насправді не робить.

**2.3. Обґрунтування вибору програмної реалізації**

В процесі проектування системи було ретельного вивчено та проаналізовано предметну область. Після детального аналізу було вирішено розроблювати програмний продукт, який заснований на веб-технологіях для використання за допомогою веб-браузера.

Технології для серверної частини застосунку були обрані на основі таких характеристик, як зручність у використанні, відкритість вихідних кодів, актуальність в наш час та можливість виконання на будь-якій операційній системі. Платформа Node.js надає змогу створювати програми на будь-якій операційній системі. Node.js надає величезні можливості для створення веб-серверів будь-якої складності, забезпечуючи при цьому велику швидкодію та надійність.

Платформа Node.js було вибрано для розробки веб застосунку через декілька причин:

* Node.js – це середовище з відкритим вихідним кодом;
* Node.js – безкоштовний фреймворк;
* Node.js працює на різних операційних системах (Windows, Linux, Mac OS X тощо);
* Node.js використовує JavaScript на сервері.

Мова JavaScript була обрана через те, що вона гнучка, підтримується більшістю браузерів, має високу швидкість роботи та зрозумілий синтаксис.

Одне з ключових переваг даного програмного продукту – це підтримка практично всіма відомими і найпопулярнішими браузерами.

Іншими перевагами JavaScript є:

* пряме підключення скриптів до HTML коду;
* можливість запуску програм в браузері і на сервері;
* широкий вибір додаткових бібліотек та фреймворків.

Базою даних було обрано NeDB через те, що це проект з відкритим вихідним кодом, призначений для використання в проектах з невеликою кількістю даних для зберігання. NeDB – це система зберігання даних, заснована на принципі зберіганні документів в BSON (Binary JSON) форматі. Ця база даних не потребує опису схеми таблиць.

Основні переваги бази даних:

* має розподілений доступ до даних, розташованих на декількох серверах;
* можливо паралельне вилучення даних MapReduce;
* більш швидке вилучення простих структур даних;
* може зберігати не структуровану інформацію.

Дані технології в сукупності дають змогу створити якісний та надійний продукт, що буде захищений від патентних позовів з боку розробників, оскільки всі використані технології покриті ліцензіями, що виключають таку можливість і надають доступ до вихідних кодів даних проектів.

**Висновок до розділу 2**

У даному розділі були розглянуті основні засоби розробки програмного забезпечення. Також було розглянуто основні переваги обраних інструментів розробки програмного забезпечення.

**РОЗДІЛ 3
РЕАЛІЗАЦІЯ ТА ОПИС РОЗРОБЛЕНОГО ВЕБ ДОДАТКА**

**3.1. Внутрішня реалізація веб-додатка**

Структура і вміст основних файлів створеного додатку:

* Скрипти, що реалізують api сервера:
* index.js

Головний скрипт api; до нього підключені інші

* auth.js

Аутентифікація і авторизація користувача

* captchaGen.js

генерування капчі

* ім’я.js

Скріпти, що реалізують різні функції API

* Файли (колекції) бази даних NeDB:
* authlog.db

Лог авторизації користувачів

* curric.db

Групи, куратори, дисципліни,

підгрупи груп, міжгрупові групи

* distrib.db

Розподіл педагогічного навантаження

* grades.db

відмітки успішності

* notes.db

Нотатки викладачів для батьків і студентів

* pupils.db

Студенти

* spravki.db

Документи про причини пропусків занять

* staff.db

Працівники (викладачі та адміністратори)

* topics.db

Теми занять і ваги відміток

* Серверні логи (щодоби новий файл)
* Файли клієнтської частини електронного журнала:
* cкрипти, що генерують контент клієнтської частини:
* header.js

Хідер і футер сторінок журналу

* info.js

Видача повідомлення замість alert ()

* ini.js

Ініціалізація констант і функцій

* login.js

Авторизація користувача і завантаження скриптів

* блок.js

Скріпти, які генерують різні блоки фронтенда

* static/

css-файли, службові зображення, шрифти:

* browsers.html

Інформація про підтримувані версії браузерів

* index.html

Точка входу клієнтської частини електронного журнала

* favicon.ico
* robots.txt
* hashgen.html

Генератор хеша пароля головного адміністратора

* pincodegen.html

Генератор PIN-кодів для редагування позначок

* server.js

Точка входу серверної частини

* serverLocal.js

Точка входу для локальної роботи з електронним журналом

* config.json

Налаштування серверної частини електронного журнала

* README.md

Короткий опис програмного продукту

**3.2. Аутентифікація і авторизація користувачів на сервері**

Аутентифікація і авторизація користувачів здійснюються за допомогою fetch-запитів до відповідного API сервера.

При первинній аутентифікації користувача на сервер передаються логін і пароль; після їх перевірки сервер повертає токен і повноваження користувача (зокрема, доступні йому ролі) або none при відмові в аутентифікації. Токен генерується з дати (номер дня від початку юнікс-ери), IP користувача і його логіна з використанням серверної солі (генерування хеш паролів користувачів). При наступних зверненнях до API сервера передаються логін і токен, отриманий при первинній аутентифікації; в цьому випадку сервер повертає тільки повноваження користувача. При первинній аутентифікації використовується капча.

**3.3. API сервера**

При запиті /cpt.a (будь-яким методом) повертається капча (в форматі «png») і її ID (в заголовку відповіді сервера «X-Cpt»). Всі інші запити до API сервера виробляються методом POST; post-дані повинні представляти із себе серіалізовані в рядок (utf-8) JSON-об’єкт; в цьому ж форматі повертається відповідь сервера (з заголовком Content-Type: text / plain, а не application / json!) Що стосується неправильного запиту, відмови в аутентифікації, недостатність повноважень у клієнта тощо. повертається відповідь none.

Повноваження клієнта (перелік доступних йому функцій API) в залежності від його ролі (адміністратор, учитель та інші) Визначено у файлі api / index.js. У всіх запитах до API присутній ключ «f» (запитувана функція API) зі значенням типу «ogin» «userEdit» і т.п.

**Первинний запит авторизації.**

Приклад запиту:

{ «T»: «staff», «l»: «ivanov», «p»: «myPass», «f»: «login», «ci»: «123456», «c»: «223344»}

Приклад відповіді:

{ «Token»: «abcd123», «roles»: [ «teacher», «tutor»]}

У запиті t – це категорія користувача (staff – співробітник, pupil – студент, par – батьки); ci – це ID капчи (отриманий від сервера при її запиті), c – власне капча.

**Інші звернення до API сервера.**

Приклад запиту:

{ «T»: «staff», «l»: «admin», «p»: «abcd123», «f»: «classDel», «z»: «10Б»}

В даному запиті z – це дані (крім категорії, логіна, токена і запитуваної функції), що передаються сервером (рядок або серіалізовані в рядок об’єкт). Відповіді API розрізняються залежно від його функцій. Повна інформація про функції, форматі запитів і відповідей API міститься у вихідних кодах модулів API; коротка інформація наведена в таблиці нижче.

Таблиця 3.1 **–** Вихідні коди модулів API

|  |  |
| --- | --- |
| **Ім’я функції (f)** | **Опис** |
| absentGet | Отримання даних про відвідуваність |
| adminsList | Отримання списка адміністраторів |
| attendGet | Отримання даних про відсутніх за один день |
| auth | Аутентифікація і авторизація користувача |
| captchaGen | Генерування капчи (картинки) з числа |
| classAdd | Додавання номера групи (з літерою) в БД |
| classDel | Видалення номера групи з БД |
| classesGroups | Отримання списку груп і їх підгруп |
| classesList | Отримання списку груп (з літерами) |
| distrEdit | Редагування педагогічного навантаження |
| distrGet | Отримання розподілу педагогічного навантаження |
| electGet | Отримання даних про позаурочної діяльності одного студента |
| export | Експорт журналу одного групи або підгрупи в html-файл |
| gradeAdd | Додавання позначки в БД |
| gradesGet | Отримання списку студентів і відміток для однієї сторінки |
| interGroupDel | Видалення міжгрупових груп з БД |
| interGroupEdit | Додавання або редагування міжгрупових груп в БД |
| interGroupGet | Отримання списку студентів міжгрупових груп |
| interGroupList | Отримання списку міжгрупових груп |
| interGroupPup | Робота зі списком студентів міжгрупових груп |
| jrnGet | Отримання даних для інтерфейсу студента і рідних |
| logGet | Отримання виписки з логу авторизації користувачів |
| notesAdd | Додавання нотатків для студентів і батьків |
| notesCheck | Перевірка наявності нових заміток для студента (одного з батьків) |
| notesDel | Видалення замітки |
| notesGet | Отримання всіх заміток для одного студента або всі нотатки, доданих одним співробітником |
| parCodes | Отримання кодів авторизації для батьків |
| pupilsList | Отримання списку студентів однієї групи |
| sprAdd | Додавання документа про причини пропуску занять |
| sprDel | Видалення документа про причини пропуску занять |
| sprGet | Отримання документів про причини пропуску занять |
| sprResp | Отримання даних про поважні причини пропусків занять |
| statGet | Отримання статистичних даних |
| subgrEdit | Додавання або видалення підгрупи групи |
| subgrPups | Редагування облікового складу підгрупи групи |
| subjAdd | Додавання додаткової дисципліни в БД |
| subjDel | Видалення додаткової дисципліни з БД |
| subjEdit | Редагування назви додаткової дисципліни |
| subjList | Отримання списку додаткових дисциплін і груп позаурочної діяльності |
| tabelGet | Отримання підсумкових оцінок одного студента |
| teachList | Отримання списку всіх студентів |
| topicEdit | Редагування тем занять, д / з та ваг відміток |
| topicsGet | Отримання тем занять однієї сторінки |
| tutorSet | Призначення групи куратору |
| tutorsList | Отримання списку кураторів |
| usAddEdit | Додавання або редагування користувача |
| usBlock | Блокування та розблокування користувачів |
| usChPwd | Зміна пароля користувачем самостійно |
| usFind | Пошук користувача |
| usFindLogin | Перевірка зайнятості логіна користувача |
| usImport | Імпорт користувачів з файлу |
| usSetAdmin | Призначення або розжалування користувача з адміністраторів |

**3.4. Загальні відомості про електронний журнал**

Електронний журнал використовує технології сучасного стандарту ECMAScript, який, на жаль, підтримується не всіма браузерами. У разі, якщо браузер клієнта не підтримується, виводиться сторінка з інформацією про підтримувані версіях браузерів.

Журнал відображається в двох варіантах: для робочого столу (стаціонарного комп’ютера) – при ширині екрану більше 1200 px; для мобільних пристроїв (телефонів, планшетів та ін.) – при ширині екрану 1200 px і менш.

Було вирішено дати назву електронному журналу «Studenote». Назва журналу утворилась від поєднання двох слів – це student і notebook.

**3.4.1. Авторизація в електронному журналі**

При вході на сайт електронного журналу користувач потрапляє на сторінку авторизації:



Рисунок 3.1 – Сторінка авторизації користувача

Всі користувачі діляться на 4 категорії: головний адміністратор, співробітник, студент, батьки. У верхній частині форми авторизації за замовчуванням показується категорія «Студент»; при натисканні мишею на цьому написі категорія циклічно перемикається на «Співробітник», а потім на «Батьки». Для головного адміністратора електронного журналу категорія, яка показується у верхній частині, не має значення; можна залишити категорію «Студент» за замовчуванням.

Після вибору категорії необхідно ввести свої логін і пароль, а також капчу (цифри, що показуються на картинці), після чого мишею натиснути кнопку «Вхід» або просто натиснути клавішу «Enter» на клавіатурі. Слід бути уважним при введенні капчі: жирна цифра 7 схожа на цифру 1. На введення логіна, пароля і капчи користувачеві дається 3 хвилини.

**3.4.2. Ролі користувача**

Після авторизації співробітнику надається можливість додатково вибрати роль користувача (у головного адміністратора, студента і батьків такої можливості немає). У правому верхньому кутку екрану (в мобільній версії – в центрі верхньої частини екрану) відображається логін користувача і його роль:

Рисунок 3.2 – Логін користувача і його роль

Для співробітників, крім ролі «Викладач» (вона є за замовчуванням у будь-якого співробітника), можуть бути доступні також ролі «Адміністратор» (директор освітньої організації, його заступники і т. п.) і «Куратор». Для вибору ролі слід натиснути мишею на відображається ролі користувача.

**3.4.3. Зміна пароля і вихід**

Самостійно змінити пароль можуть лише співробітники; студентам і батькам слід для цієї мети звернутися до головного адміністратора електронного журналу. При цьому пароль батьків «прив’язаний» до паролю студента; змінити їх можна тільки одночасно. Співробітникові для самостійної зміни пароля слід натиснути мишею на значку з перехрещеними молотками в правому верхньому куті екрану.

Для виходу (log out) з електронного журналу необхідно натиснути мишею на значку «Вихід» (жовтий квадратик у правому верхньому куті). Слід мати на увазі, що log out і повернення до форми авторизації відбудеться також при спробі оновити сторінку в браузері. Не оновлюйте сторінку при роботі з електронним журналом.

**3.4.4. Початок навчального року**

На початку навчального року для «запуску» електронного журналу необхідно зробити наступні дії:

* **головному адміністраторові** ввести перелік груп, додаткові предмети (список предметів за замовчуванням вже міститься в електронному журналі), імпортувати користувачів, призначити адміністраторів електронного журналу (директора освітньої організації, його заступників і ін.);
* **одному з адміністраторів** призначити кураторів і ввести розподіл навчального навантаження (крім навчального навантаження в нестворених поки підгрупах груп);
* **кураторам** створити відсутні (крім наявних за замовчуванням) підгрупи груп; розподілити студентів по підгрупах; роздрукувати і видати батькам коди доступу в електронний журнал;
* **одному з адміністраторів** ввести остаточний розподіл навчального навантаження, включивши в неї навантаження викладачів у новостворених підгрупах.

**3.5. Робота в ролі студента**

При роботі в ролі студента або батьків користувачеві доступні 6 пунктів меню (в десктопній версії – вкладки; в мобільній версії – розкриваються пункти при натисканні на значок меню в лівому верхньому кутку екрану): «Журнал», «Внеуроч», «Залікові», «Відвідуваність», «Довідки», «Нотатки».

**Журнал.** Журнал студента відображається у вигляді таблиці з 5 колонками: дата, вага позначки, тема заняття, домашнє завдання, відмітка. Вага позначки (може бути від 0 до 4) характеризує її значимість. Відмітки з вагою 0 в підрахунку сум балів і у визначенні середньозважених балів не беруть участь, тобто не впливають на виставлення підсумкової позначки за навчальний період. За замовчуванням вага позначки дорівнює «1»; в цьому випадку він не показується.

У верхній частині сторінки журнала можна вибрати відображену навчальну дисципліну. Слід мати на увазі, що якщо одну й ту саму дисципліну ведуть кілька викладачів (наприклад, один у всієї групи, а інший тільки в однієї із підгруп), будуть показуватися різні сторінки журналу. Якщо по якій-небудь дисципліні поки немає записаних вкладачем занять, в списку доступних занять він не показується.

Заняття, що проводяться в групах з студентами кількох груп (факультативи, гуртки, творчі об’єднання та ін.), показуються в кінці списку занять.

Підсумкові оцінки (показуються на більш темному тлі) відображаються лише після їх появи в електронному журналі. На більш темному тлі показуються і екзаменаційні оцінки (у разі, якщо в темі заняття є слово «Іспит» з великої літери).

У нижній частині сторінки щоденника відображається зведена таблиця, яка містить суми балів (з урахуванням ваг, тобто S = Σ bi wi), середньозважені бали m = S / Σ wi і кількість пропущених занять.

**Залікова.** У заліковій показуються всі позначки проміжної атестації (за чверті, півріччя і навчальний рік), виставлені до моменту генерування залікової. Якщо за будь-яку дисципліну не виставлено жодної позначки проміжної атестації, відповідна дисципліна не показується.

Після таблиці з відмітками показується посилання «Версія для друку».

**Відвідуваність.** На цій сторінці показуються зведені дані відвідуваності (кількість пропущених занять з навчальних періодів: як загальне, так і з поважної причини), а також, окремо з кожного навчальної дисципліни, дати пропуску занять без поважної причини (після кожної дати в дужках вказується кількість занять з даної дисципліни, пропущених в даний день). «Поважність» причини пропуску занять визначається за наявністю введених в електронний журнал куратором документів про причини пропусків.

Нижче цих відомостей є посилання «Версія для друку», по якій в новому вікні відкривається документ для друку його на папері (слід в меню браузера вибрати пункт «Друк»).

**Довідки.** В даному розділі показуються документи про поважні причини пропусків занять. Документи вносяться в електронний журнал куратором. Потрібно вчасно надавати куратору паперові документи про причини пропусків занять для внесення їх в електронний журнал.

**Нотатки.** В даному розділі показуються всі замітки, опубліковані викладачами або іншими співробітниками освітньої організації для студента, його групи або підгруп (як підгруп групи, так і міжгрупових груп), в яких він складається.

При наявності нових заміток (доданих пізніше, ніж остання авторизація в електронному журналі) в правому верхньому кутку показується інформаційна іконка.

**3.6. Робота в ролі батьків**

Робота користувача електронного журналу в ролі батьків повністю аналогічна робота в ролі студента. Різниця полягає лише у використовуваному паролі. Паролі батькам видаються кураторами.

**3.7. Робота в ролі викладача**

Викладачеві доступні п’ять вкладок (в мобільній версії – п’ять пунктів меню): «Журнал», «Явка», «Групи», «Внеуроч» і «Нотатки».

**Журнал.** На початку роботи з журналом необхідно вибрати групу і дисципліну за допомогою випадаючих списків, які перебувають у верхній частині сторінки. Відображаються лише доступні викладачу групи і дисципліни відповідно до розподілу педагогічного навантаження, введеним адміністратором електронного журналу:



Рисунок 3.3 – Головне меню користувача в ролі викладача

Після вибору групи і дисципліни викладач бачить перед собою журнальну сторінку, що складається з трьох блоків:



Рисунок 3.4 – Сторінка «Журнал» користувача в ролі викладача

* Список студентів і позначки успішності;
* Введення нового заняття або редагування існуючого;
* Відомості про проведені заняття.

Розглянемо роботу з кожним блоком окремо.

**Список студентів і позначки успішності.** Студенти в списку показуються в алфавітному порядку (див. рис. 3.5). Заблоковані студенти (відраховані з освітньої організації) також показуються в алфавітному порядку, але в кінці списку; всі раніше виставлені їм позначки зберігаються.



Рисунок 3.5 – список студентів на сторінці «Журнал»

У верхньому рядку показуються дати введених в електронний журнал занять, під ними – ваги відміток (вага позначки характеризує її значимість). Вага за замовчуванням дорівнює 1; діапазон можливих ваг – від 0 до 4 з кроком 0.5.

У разі, якщо в темі заняття міститься слово «Іспит» (з великої літери), замість дати показується слово «Прим»; колонка відміток, крім того, має більш темний фон, аналогічно колонкам підсумкових оцінок. При наведенні мишею на слово «Прим» показується підказка – дата проведення іспиту.

В основній частині таблиці показуються позначки успішності і відомості про відвідуваність. Для прокрутки таблиці вліво і вправо можна використовувати смугу прокрутки в нижній частині таблиці, а можна (що зручніше), утримуючи клавішу Shift на клавіатурі і тримаючи курсор миші над таблицею, обертати коліщатко миші.

Для виставлення або редагування позначки необхідно натиснути мишею на відповідну клітинку таблиці; фон клітинки при цьому стане білим, клітинка перетвориться в поле для введення. При введенні і редагуванні відміток необхідно мати на увазі наступне:

* в одній клітинкі може бути не більше 5 символів;
* допускаються лише цифри, пробіл і українська мала літера «н»;
* позначки виставляються в будь-якій системі (не обов’язково п’ятибальною); програмне забезпечення електронного журналу це ніяк не аналізує;
* при необхідності виставити в одну клітку кілька позначок вони розділяються пропуском; таким чином, «4 3» буде інтерпретовано як дві позначки, «4» і «3», в той час як «43» буде витлумачено як одна позначка «43»;
* відсутність студента на занятті (незалежно від його причини) відзначається українською малою буквою «н». Якщо студент не був на здвоєному занятті, слід виставити «нн» (разом). У разі, якщо студент на першому занятті отримав позначку, а з другого пішов, або якщо він не був, а потім здав тему і отримав позначку тощо. Допускаються варіанти типу «4н», «нн4», «нн3 4». Букву «н» слід ставити впритул до цифр, пробіл використовується тільки для відділення відміток один від одного;
* для запису введеної позначки в електронний журнал слід або натиснути клавішу Enter на клавіатурі, або натиснути клавішу ↓ (стрілка вниз), або натиснути мишею поза полем, де тільки що була введена відмітка. При цьому фон клітинки стане рожевим; після успішного запису в електронний журнал фон клітини повернеться до звичайного стану;
* при спробі ввести або відредагувати позначку заблокованого студена (що знаходиться в нижній частині списку) виводиться відповідне попередження, оцінка не змінюється;
* підсумковіпозначки виставляються аналогічно поточним (природно, буква «н» при цьому не використовується, в клітку виставляється тільки одна підсумкова відмітка, що складається не більше ніж з трьох цифр). Якщо студент не атестований, виставляється підсумкова оцінка «0» (в журналі студента і в заліковій підсумкових оцінок вона буде автоматично замінено на «н / а»);
* якщо в якості поточної або підсумкової позначки виставлено «999», в журналі студента і в заліковій ця позначка буде автоматично замінено на «зар.» (Зараховано); в якості позначки «не зараховано» слід виставляти «0» (студенту і в заліковій буде показаний як «н / а»);
* за замовчуванням четверта чверть і друге півріччя закінчуються 31 травня, а навчальний рік – 25 червня (в освітній організації може бути, наприклад, літня сесія). Тому допустимо записувати заняття (наприклад, з темою «Іспит за рік») з датою в червні. Відповідна колонка відміток буде показуватися після колонки підсумкових оцінок за друге півріччя, але до колонки підсумкових оцінок за рік;
* редагування існуючих і введення нових відміток, що відносяться до закінченого навчального періоду (чверті) за замовчуванням заборонено (ця заборона не поширюється на створення, редагування і видалення тем занять – колонок відміток). Для введення або редагування позначки, що відноситься до закінченого навчального періоду (чверті), необхідно ввести запитуваний в цьому випадку PIN-код. PIN-коди видаються одним з адміністраторів електронного журналу; кожен PIN-код дійсний тільки для одного конкретного викладача, групи, дати і діє тільки протягом дня отримання PIN-коду;
* поява повідомлення «Неправильний PIN-код» до введення позначки в поле введення свідчить про те, що формат введеного коду був невірним (наприклад, замість чотирьох цифр були введені тільки три); поява такого повідомлення після введення позначки і відправки її на сервер свідчить про неправильність самого введеного PIN-коду. В останньому випадку слід оновити сторінку в браузері і заново авторизуватися в електронному журналі.

При натисканні мишею на прізвища студента показується вікно зі статистичними даними:



Рисунок 3.6 – Статистичні дані студента

Колонки таблиці означають наступне:

* Σ – сума всіх отриманих за відповідний навчальний період відміток з урахуванням їх ваг, тобто S = Σ bi wi (може використовуватися, наприклад, при бально-рейтинговою системою оцінки);
* m – середньозважена відмітка за відповідний навчальний період (сума балів з урахуванням ваг, поділена на суму ваг, тобто m = S / Σ wi);
* Н – кількість занять, пропущених за навчальний період.

З урахуванням сказаного вище про можливість виставлення оцінок в червні сума балів за рік може не збігатися із загальною сумою за перше і друге півріччя.

**Введення нового заняття або редагування існуючого.** Перед введенням позначок необхідно створити колонкув таблиці відміток; це робиться шляхом введення нового заняття за допомогою форми, що знаходиться в правому верхньому кутку інтерфейсу викладача:



Рисунок 3.7 – Колонка для введення нового заняття або редагування існуючого

Дата вводиться шляхом вибору її з календаря (дата може бути тільки в межах навчального року, встановлених в налаштуваннях електронного журналу, за замовчуванням – з 1 вересня по 25 червня). При виборі дати вже існуючого заняття нова колонка не створюється, редагуються тема заняття, домашнє завдання та вага відміток існуючої колонки (самі позначки при цьому зберігаються). Таким чином, може бути створена тільки одна колонка з даної датою.

Вказівка теми заняття обов’язково; якщо при редагуванні вже існуючого заняття видалити наявну тему (або вказати «порожню» тему), колонка з відповідною датою видаляється разом з усіма відмітками. Таким способом (видаленням теми заняття) можна видаляти помилково записані заняття.

Домашнє завдання вказувати не обов’язково; якщо воно не зазначено, в відомостях про проведені заняття і в журналі студента домашнє завдання не відображається.

У домашнє завдання можна включати посилання на зовнішні ресурси. Для створення посилання виділіть фрагмент тексту домашнього завдання (це буде текст посилання), після чого натисніть мишею на кнопку із зображенням якоря. У вікні введіть URL (інтернет-адреса) посилання і натисніть кнопку «Створити посилання». Якщо URL вказано без вказівки на протокол http: //, https: // або ftp: //, автоматично підставить протокол http: // (тобто посилання ya.ua, Наприклад, буде перетворена в http://ya.ua).

У разі, якщо в один день в групі проводиться кілька занять по по одній і тій же дисципліні, слід вибрати зі списку кількість проведених годин (за замовчуванням використовується значення «1 ч»).

Вага відміток вибирається зі списку. Вага, що показується за замовчуванням, дорівнює 1.

Заповнивши всі поля форми введення нового заняття або редагування існуючого, натисніть кнопку введення >>.

**Відомості про проведені заняття.** Нижче форми введення нової лекції або лабораторної роботи (або редагування існуючого заняття) розташоване вікно, в якому показуються відомості про проведені заняття:



Рисунок 3.8 – Колонка для відомостей про проведені заняття

Показуються дата, теми лекцій та лабораторних робіт і (при наявності) домашнє завдання сірим кольором в квадратних дужках. Якщо в даний день проведено кілька занять з даної дисципліни, після найменування теми в дужках показується кількість проведених годин.

Для навігації по датах можна використовувати два способи:

* прокрутивши коліщатком миші вікно з темами (можна також використовувати смугу прокрутки), натиснути мишею на потрібної дати, при цьому: 1) тема заняття, домашнє завдання та вага відміток з’являються у вікні редагування зверху і стають доступні для редагування; 2) таблиця з відмітками успішності «прокручується» так, що колонка з відповідною датою виявляється найпершої (відразу після списку студентів);
* прокрутивши таблицю з відмітками успішності (за допомогою смуги прокрутки в нижній її частині, або коліщатком миші при утримуваної клавіші Shift), натиснути мишею на дату заняття зверху, при цьому відповідна дата, тема, домашнє завдання та вага відміток з’являються у вікні редагування і стають доступні для редагування.

**Явка.** Сервіс «Явка» показує відомості про студентів (група, прізвище, ім’я, назви дисциплін), що не були хоча б на одному занятті в даний день (незалежно від причини відсутності). За замовчуванням показуються дані для поточного дня; при бажанні можна відобразити дані за будь-який інший день. Явка на заняття, що проводяться в міжгрупових групах (групах позаурочної діяльності) в цьому сервісі не враховується.

Якщо користувач є адміністратором (незалежно від того, в якій ролі він працює в даний момент), йому показуються дані всієї освітньої організації (вибірка проводиться по всім студентам). В іншому випадку показуються дані про явку тільки тих груп, де у користувача є педагогічне навантаження або кураторське керівництво.

**Групи позаурочної діяльності.** Розділ «Групи позаурочної діяльності» (див. рис. 3.9) дозволяє редагувати обліковий склад, в яких викладач веде заняття (факультативи, гуртки, секції, творчі об’єднання і т. п.). Попередньо самі групи повинні бути створені адміністратором електронного журналу та «призначені» даному викладачеві.



Рисунок 3.9 – Сторінка груп позаурочної діяльності

Спочатку необхідно вибрати групу зі списку груп позаурочних занять, закріплених за даним викладачем адміністратором електронного журналу. Після вибору групи ви побачите її обліковий склад (якщо він не порожній). Для видалення студента з групи слід кликнути мишею на іконці видалення (хрестику) зліва від прізвища; для додавання в групу нового студента слід ввести фрагмент його прізвища в полі введення, після чого вибрати учня з випавшого списку.

Завантаження списку починається після введення трьох букв прізвища. Якщо прізвище того, хто навчається складається з двох букв (наприклад, «Лі»), після другої літери необхідно ввести пробіл.

Слід мати на увазі, що в залежності від швидкості інтернет-з’єднання подгрузка списку може зайняти деякий час, тому після введення фрагмента прізвища може знадобитися деякий очікування.

**Позаурочна діяльність.** Робота з цим розділом повністю аналогічна роботі з розділом «Журнал» з трьома відмінностями:

* замість випадаючих списків груп і дисциплін показується, що випадають групи позаурочної діяльності, в яких у викладача є педагогічне навантаження;
* праворуч від списку груп показується посилання «Експорт», при натисканні по якій проводиться експорт журналу обліку занять в обраній групі позаурочної діяльності в html-файл;
* в списку студентів зліва від кожного прізвища показується група, в якому навчається цей студент.

**Нотатки.** Даний розділ дозволяє публікувати замітки для студентів і їх батьків: персонально для одного студента, для групи в цілому, для підгрупи студентів (якщо викладач має педагогічне навантаження в цій групі або в цій підгрупі), для міжгрупових груп. Опубліковані замітки можна видаляти, натиснувши на відповідну іконку (хрестик) у списку всіх заміток, опублікованих даними викладачем.

**3.8. Робота в ролі куратора**

Куратору доступні вкладки (пункти меню) «Журнал», «Відвідуваність», «Явка», «Довідки», «Нотатки», «Підгрупи», «Залікові», «Внеуроч», «Батьки», «Експорт» і «Лог ».

**Журнал.** Робота куратора з вкладкою (пунктом меню) «Журнал» аналогічний роботі з цією вкладкою з викладачем з наступними відмінностями:

* в списку груп і дисциплін показуються всі групи, де користувач є куратором; для кожної групи показуються всі дисципліни (відповідно до розподілу педагогічного навантаження, введеним в електронний журнал адміністратором);
* відсутня можливість введення і редагування позначок;
* відсутня можливість введення нових занять і редагування існуючих (відповідна форма введення не відображається).

**Відвідуваність.** У верхній частині відповідної вкладки показуються випадають списки груп (де користувач є куратором) і студентів. Можна вибрати як конкретного студента зі списку, так і пункт «Вся група».

При виборі всієї групи показується таблиця зі зведеними даними відвідуваності; при виборі конкретного студента, крім того, показуються дати пропуску ними занять без поважної причини (після кожної дати в дужках вказано кількість уроків з даного предмету, пропущених в даний день):



Рисунок 3.10 – Дані про відвідуваніть

«Поважність» причини пропуску занять визначається за наявністю введених в електронний журнал куратором документів про причини пропусків.

Слід мати на увазі, що дані в зведеній таблиці по групі можуть відрізнятися від даних окремих студентів: для кожного студента показуються всі пропуски занять (включаючи заняття, що проводяться в міжгрупових групах позаурочної діяльності), в той час як в зведеній таблиці враховуються тільки пропуски занять, що проводяться в даній групі.

Нижче цих відомостей є посилання «Версія для друку», по якій в новому вікні відкривається документ для друку його на папері (слід в меню браузера вибрати пункт «Друк»).

**Явка.** Робота куратора з вкладкою (пунктом меню) «Явка» аналогічна роботі з цією вкладкою викладача.

**Довідки.** Дана вкладка (пункт меню) призначена для введення і відображення документів про причини пропуску студентами занять. Слід вибрати групу, студента, вид документа про причини пропуску занять, дати початку і закінчення (включно) дії документа; при бажанні можна ввести [коротку] примітку (воно буде показуватися студенту і його батькам). Після введення цих даних слід натиснути кнопку введення **>>**.

Нижче форми введення нового документа відображається таблиця з даними вже введених документів. Видалити документ можна натисканням на хрестик в лівій колонці таблиці.

Всі пропуски занять з датами, що входять в діапазони дат введених документів, вважаються поважними.

**Нотатки.** Даний розділ дозволяє публікувати замітки для студентів і їх батьків: персонально для одного студента або для групи в цілому. Опубліковані замітки можна видаляти, натиснувши на відповідну іконку (хрестик) у списку всіх заміток, опублікованих даними куратором.

**Підгрупи.** Вкладка (пункт меню) для роботи з підгрупами дозволяє як створювати в групі нові підгрупи і видаляти існуючі, так і редагувати складу кожної підгрупи.

За замовчуванням в кожній групі створені підгрупи «KI», «KT», «TД», «ТК», «ТН» і «ТТ»:



Рисунок 3.11 – Підгрупи групи

При необхідності куратор може як видалити якісь з цих підгруп (студенти при цьому зі списку групи не видаляються), так і додати нову підгрупу.

Видаляти підгрупу, якщо для її студентів вже виставлені будь-які позначки (викладачем, викладаючий заняття в цій підгрупі), неприпустимо, це призведе до спотворення даних в електронному журналі. Те ж саме стосується і ситуації, коли відміток немає, але викладачем вже введені теми занять.

Щоб додати новий підгрупи введіть її назву (можна використовувати тільки рядкові українські букви і цифри; назва підгрупи не повинно бути довше 10 символів) і натисніть кнопку «Додати». Знову додана підгрупа буде порожній, так як поки в неї не включено жодного студента.

Для редагування складу підгрупи виберіть її назву та у списку студентів групи нижче відзначте тих, хто входить в дану підгрупу; не забудьте після цього натиснути кнопку «Зберегти зміни» (на скріншоті показаний тестову групу, що складається з чотирьох студентів, три з яких входять до підгрупи «TT»):



Рисунок 3.12 – Склад підгрупи

Відзначати (або прибирати позначку) навпроти студента можна натисканням миші в комірці таблиці праворуч від прізвища.

Видаляти студента зі складу підгрупи, якщо у нього вже виставлені позначки (викладачем, провідним в цій підгрупі заняття) неприпустимо, це призведе до спотворення даних в електронному журналі. Тому для перекладу студента в іншу підгрупу (якщо виникне така необхідність) слід додати його в нову підгрупу, не видаляючи зі старої.

**Залікова.** Для генерування залікової відміток проміжної атестації куратор повинен вибрати групу і студента з випадаючих списків.

У заліковій показуються всі позначки проміжної атестації (за чверті, півріччя і навчальний рік), виставлені до моменту генерування залікової. Якщо за будь-яку дисципліну не виставлено жодної позначки проміжної атестації, відповідна дисципліна не показується.

Передбачається, що адміністрація освітньої організації стежить, щоб з одной дисципліни одному студенту позначки проміжної атестації виставлялися тільки одним викладачем; програмне забезпечення електронного журналу це ніяк не відслідковує. У разі, якщо одну і ту ж дисципліну у даного студента ведуть два викладача (наприклад, один лекційні, а другий практичні заняття) і обидва виставили позначки проміжної атестації, в залікову потрапить тільки одна з двох оцінок (випадковим чином). Тому виставлення відміток проміжної атестації одному студенту по одній і тій же дисципліні декількома викладами слід уникати.

Після таблиці з відмітками показується посилання «Версія для друку». Варіант для друку має верхнє поле 5 см для того, щоб залікову можна було надрукувати на бланку, у верхній частині якого зазначено найменування освітньої організації.

**Позаурочна діяльність.** Після вибору групи і студента куратору показується список відвідуваних даними студентами курсів позаурочної діяльності (курсів за вибором, факультативів, занять творчих об’єднань та ін., Що проводяться в міжгрупових групах, тобто в групах, що складаються з студентів різних груп). По кожному курсу позаурочної діяльності відображаються результати обліку відвідування та успішності.

Результати обліку відвідування та успішності на заняттях, що проводяться в підгрупах однієї групи, відображаються у вкладці «Журнал», навіть якщо ці заняття за своїм змістом є позаурочної діяльністю.

**Батьки.** На цій сторінці відображаються коди доступу до електронного журналу (паролі) для батьків студентів (логін для входу батьків збігається з логіном студента). Нижче відповідної таблички є посилання «Версія для друку». Заблоковані (відраховані з освітньої організації) студенти в табличці не відображаються.

**Експорт.** Дана вкладка призначена для експорту електронного журналу однієї групи в файл (використовується формат **html**). Отриманий файл потім можна або переглянути на екрані монітора, або роздрукувати на папері.

Порядок дій для експорту журналу в файл докладно описаний в інструкції, яка відображається у вкладці «Експорт».

**Лог.** У цій вкладці можна переглядати логи авторизації студентів і їх батьків в електронному журналі. Слід вибрати групу, студента, місяць і натиснути кнопку **>>**.

**3.9. Робота в ролі адміністратора**

Адміністратору доступні наступні вкладки (пункти меню): «Журнал», «Відвідуваність», «Явка», «Нотатки», «Навантаження», «Групи», «Внеуроч», «Залікові», «Експорт», «Статистика», «Лог».

**Журнал.** Робота адміністратора зі вкладкою (пунктом меню) «Журнал» аналогічна робот з цією вкладкою викладача з наступними відмінностями:

* в списку груп і дисциплін показуються всі групи освітньої організації; для кожного класу показуються всі дисципліни (відповідно до розподілу педагогічного навантаження, введеним в електронний журнал адміністратором);
* відсутня можливість введення і редагування позначок;
* відсутня можливість введення нових занять і редагування існуючих (відповідна форма введення не відображається).

**Відвідуваність.** Дана сторінка повністю аналогічна відповідній сторінці інтерфейсу куратора з тією відмінністю, що в випадаючому списку груп адміністратору доступні всі групи освітньої організації.

**Явка.** Робота адміністратора зі вкладкою (пунктом меню) «Явка» аналогічна роботі з цією вкладкою викладача. Незалежно від того, в якій ролі (адміністратора, викладача або куратора) працює в даний момент користувач, який має права адміністратора, йому показуються дані про всю освітньої організації (вибірка проводиться по всім студентам).

**Нотатки.** Даний розділ дозволяє публікувати замітки для студентів і їх батьків: персонально для одного студента або для групи в цілому. Опубліковані замітки можна видаляти, натиснувши на відповідну іконку (хрестик) у списку всіх заміток, опублікованих адміністратором. Публікація адміністратором заміток для міжгрупових груп не передбачена.

**Навантаження.** У цій вкладці адміністратор може виконувати дві операції: розподіляти педагогічне навантаження між викладачами та призначати кураторів. Природно, попередньо головний адміністратор електронного журналу повинен створити облікові записи співробітників, додати додаткові навчальні предмети (якщо вони є) і ввести перелік наявних в освітній організації груп.

Для розподілу педагогічного навантаження виберіть співробітника зі списку, потім виберіть навчальну дисципліну, групу та підгрупу групи (або пункт меню «вся»). У таблиці з розподілом навантаження обраного співробітника з’явиться новий рядок:

Рисунок 3.13 – Розподіл педагогічного навантаження

Відредагувати педагогічне навантаження (видалити рядок з таблиці) можна за допомогою хрестика в правій колонці таблиці.

Для призначення (і подальшого редагування списку) кураторів просто натисніть мишею на прізвищі куратора (або на словах «Не призначений», як показано на скріншоті) і виберіть прізвище якого призначаємого куратором співробітника зі списку (див. рис. 3.14).

Рисунок 3.14 – Призначення групі куратора

Після завершення введення педагогічного навантаження всіх педагогів рекомендується за допомогою технічного фахівця перезавантажити сервер електронного журналу, щоб оптимізувати базу даних.

**Групи.** У цьому пункті меню адміністратор може створювати і редагувати (тільки назви груп і закріплених викладачів, але не списковий склад студентів) міжгрупові групи для організації позаурочної діяльності (факультативів, гуртків, секцій, творчих об’єднань і т. п.). Обліковий склад міжгрупових груп редагується викладачами, викладаючими заняття в цих групах (див. рис. 3.15).



Рисунок 3.15 – Міжгрупові групи.

У верхній частині відповідної сторінки відображається список створених міжгрупових груп; в лівій частині кожного рядка є іконка (хрестик) для видалення групи. Нижче списку груп розташована форма для додавання нової групи або редагування існуючої.

Для додавання нової групи необхідно вказати її умовний номер (типу 23Б, числа від 20 до 99), найменування предмета позаурочної діяльності (факультативу, гуртка, секції тощо, наприклад, в’язання гачком або олімпіадних хімія), для якого створена група, а також викладача, провідного заняття в цій групі. Якщо ви вкажете умовний номер вже існуючої групи, її дані будуть перезаписані.

Найрозумніше заздалегідь розподілити номери між педагогами, провідними заняття в міжгрупових групах (наприклад, 20 – Іванов, 21 – Петров та інші), а потім створювати конкретні групи відповідно до цього класифікатора.

**Позаурочна діяльність.** Робота з даними розділом повністю аналогічна роботі з цим розділом в ролі викладача (З єдиною різницею, що складається у відсутності прав на редагування позначок і тим занять). Адміністратору при цьому доступні для перегляду дані всіх груп позаурочної діяльності.

**Заліки.** Робота з цим розділом повністю аналогічна роботі з заліками куратора. Адміністратору додатково надається можливість вибору групи.

**Експорт.** Дана вкладка призначена для експорту електронного журналу однієї групи в файл (використовується формат **html**). Отриманий файл потім можна або переглянути на екрані монітора, або роздрукувати на папері.

Порядок дій для експорту журналу в файл докладно описаний в інструкції, яка відображається у вкладці «Експорт».

Журнали обліку позаурочної діяльності в міжгрупових групах експортуються педагогами, які проводять заняття в цих групах, або адміністратором при перегляді журнальних сторінок цих груп (значок «Експорт» праворуч від назви групи).

**Статистика.** На вкладці «Статистика» адміністратору доступні 4 види статистичних даних:

* дані про своєчасність заповнення електронного журналу викладачами;
* статистика по паралелі груп;
* статистика по одному викладачу;
* статистика по одній дисципліні.

Дані про своєчасністьзаповнення електронного журналу викладачами публікуються у вигляді списку викладачів, які не заповнювали журнал протягом останніх 14 днів (окремо по кожній дисципліні і групі/підгрупі). При цьому існують три винятки:

* при запиті статистики в період з 26 грудня по 15 січня буде відображений список викладачів, які не заповнювали журнал з 10 грудня;
* при запиті статистики в період з 26 травня по 31 серпня буде відображений список викладачів, які не заповнювали журнал з 10 травня;
* при запиті статистики в період з 1 по 15 вересня буде відображений список викладачів, які не заповнювали журнал з 1 вересня.

У статистиці по паралелі груп показується список студентів даної паралелі, невстигаючих за підсумками кожного навчального періоду (із зазначенням дисциплін неуспішності і відміток).

У статистиці по одному викладачу показується кількість проведених годин занять в кожному навчальному періоді, а також список невстигаючих з дисциплін, які веде даний викладач.

У статистиці по одній дисципліні показується список невстигаючих за підсумками кожного навчального періоду (по викладачам і групам).

**Лог.** У цій вкладці можна переглядати логи авторизації користувачів в електронному журналі. Можливий перегляд трьох типів:

* лог авторизації одного студента або його батьків за один календарний місяць;
* лог авторизації одного співробітника за один календарний місяць;
* лог авторизації всіх користувачів електронного журналу за одну годину.

Для отримання і перегляду логів слід вибрати в одному з типів перегляду необхідні фільтри і натиснути кнопку **>>**.

**PIN-коди редагування позначок.** За замовчуванням введення нових або редагування викладачами існуючих позначок, що відносяться до закінченого навчального періоду (чверті) заборонені. При спробі виконання цих операцій запитується спеціальний PIN-код.

Видачу викладачам PIN-кодів для редагування позначок, що відносяться до завершиних навчальних періодів, доцільно доручити одному з адміністраторів електронного журналу або іншому працівникові освітньої організації. Генерування PIN-кодів за допомогою інтерфейсу адміністратора електронного журналу не передбачено; для цієї мети в комплект поставки електронного журналу (дистрибутив) входить спеціальний файл **pincodegen.html**. При установці електронного журналу на сервер освітньої організації цей файл слід скопіювати на робочу ЕОМ працівника освітньої організації, який буде видавати PIN-коди.

Перед початком роботи в файлі **pincodegen.html** слід відредагувати рядок 3, замінивши значення **«Abcd»** на значення ключа **SALTPIN** з файлу **config.json** (Це значення слід дізнатися у технічного фахівця, яка провадила установку журналу на сервер освітньої організації). Файл **pincodegen.html** слід зберігати тільки на робочій ЕОМ працівника освітньої організації, що видає PIN-коди, і не допускати попадання цього файлу в чужі руки.

Для генерування PIN-коду слід запустити файл **pincodegen.html** браузером, ввести логін викладача, групу, дату редагованої позначки, після чого натиснути кнопку «Отримати ПІН-код» (з’єднання з Інтернетом при цьому не потрібно). PIN-код є чотиризначним числом і дійсний тільки для зазначеного викладача, групи і дати протягом того дня, коли він виданий.

Датою відміток проміжної атестації за навчальний період (чверть, півріччя, рік) є дата останнього дня навчального періоду; ці дати можна уточнити в файлі **/www/js/ini.js** (змінна **STPER**).

**3.10. Головний адміністратор**

Головному адміністратору електронного журналу доступні вкладки (пункти меню) «Користувачі», «Адміністратори», «Групи» і «Дисципліни».

**Користувачі.** При роботі зі списком користувачів (студентів і співробітників) головний адміністратор може:

* додавати користувачів;
* редагувати їх дані (в тому числі змінювати пароль);
* імпортувати користувачів з заздалегідь підготовленого файлу;
* призначати користувачів адміністраторами електронного журналу;
* проводити пошук користувачів по найменуванню групи (для студентів), а також за фрагментом імені, прізвища або по батькові;
* блокувати користувачів (заблокованого користувачеві вхід в електронний журнал стає недоступний).

Перед додаванням першого користувача (навіть і співробітника) необхідно додати хоча б один групув список найменувань груп, інакше буде видано повідомлення про помилку.

Батьківстудентів додавати до числа користувачів не потрібно; вони працюють з електронним журналом за логіном своєї дитини і батьківського паролю, який генерується автоматично з пароля студента. Таким чином, при зміні пароля студента автоматично зміниться і батьківський пароль.

**Переведення студентів.** Іноді протягом навчального року виникає необхідність перевести студента в іншу групу (змінити літери групи), або перевести його в іншу підгрупу всередині однієї групи. Ці операції в електронному журналі «Studenote» неприпустимі і приведуть до спотворення всіх даних, що відносяться до цього студента. Перехід студента в іншу групу або в іншу підгрупу всередині групи (ця операція може бути здійснена куратором) неприпустимий!

Для коректного переходу студента необхідно:

* головному адміністраторові для переходу в іншу групу заблокувати студента і створити новий обліковий запис для цього студента з новим логіном і новою групою;
* куратору для переходу в іншу підгрупу всередині групи додати студента в нову підгрупу, але залишити його стару.

Після цих дій студент залишиться в списках старої групи (підгрупи) з усіма «старими» відмітками. Фактично в списках студентів будуть двоє: «старий» і «новий».

**Адміністратори.** При роботі зі списком адміністраторів електронного журналу головний адміністратор може:

* переглядати список адміністраторів;
* «Розжалувати» користувачів з адміністраторів.

При редагуванні даних користувача на вкладці «Користувачі» він автоматично видаляється зі списку адміністраторів; після редагування слід знову призначити його адміністратором (призначення користувача адміністратором проводиться на вкладці «Користувачі»).

**Групи.** На цій вкладці (пункті меню) адміністратор може додавати номер і літери (букви) групи в список груп, а також видаляти групи зі списку (при видаленні групи студенти цієї групи з бази не видаляються, їх дані ніяк не редагуються). Призначення групі куратора проводиться адміністратором електронного журналу.

**Дисципліни.** За замовчуванням в електронному журналі вже є список навчальних дисциплін. Для додавання додаткових навчальних дисциплін. Слід ввести умовний номер і назву, після чого натиснути кнопку «Додати». Для видалення або редагування додаткової дисципліни натисніть мишею на хрестик або значок з перехрещеними молотками зліва від назви.

Дисципліни за замовчуванням видаляти або редагувати веб-інтерфейсом можна. Якщо все ж дуже потрібно це зробити, можна відредагувати список предметів за замовчуванням в файлі **www / js / ini.js** при установці електронного журналу на сервер.

Умовний номер дисципліни визначає порядок відображення її в списку. Умовні номери за замовчуванням починаються з цифри, що визначає предметну область (1 – фізика; 2 – іноземні мови і т.д.) Номери присвоєні з кроком 10 для того, щоб можна було вставити додатковий предмет «на своє місце».

Найменування дисципліни може містити від 2 до 30 букв українського і латинського алфавітів, цифр, дужок, крапок і пробілів. Інші символи в найменуванні предмета не допускаються.

**Резервне копіювання.** Для резервного копіювання бази даних електронного журналу головному адміністраторові слід звернутися до технічного фахівця, який виготовляв установку журналу на сервер освітньої організації. Необхідно просто скопіювати папку **db** в файлової системі сервера в будь-яке зручне місце.

Для відновлення бази даних з копії необхідно скопіювати файли копій в папку **db** файлової системи сервера, після чого перезапустити сервер. Рекомендується періодично (наприклад, раз на тиждень) перезапускати сервер і поза зв’язком з відновленням бази даних (при перезапуску сервера файли бази даних оновлюються, що прискорює роботу електронного журналу).

**Новий навчальний рік.** Архівування даних журналу за минулий навчальний рік (ЕЖ «Studenote» розрахований тільки на один навчальний рік) проводиться збереженням файлів папки **db** у будь-яке зручне вам місце (безпосередньо перед цим рекомендується перезапустити сервер електронного журналу, щоб оновити зазначені файли). У цих файлах зберігається наступна інформація:

* **authlog.db** – лог авторизації користувачів;
* **curric.db** – групи, куратори, підгрупи, дисципліни, міжгрупові групи;
* **distrib.db** – розподіл педагогічного навантаження;
* **grades.db** – позначки успішності;
* **notes.db** – замітки викладачів для батьків і студентів;
* **pupils.db** – дані про студентів;
* **spravki.db** – документи про причини пропусків занять;
* **staff.db** – дані про співробітників;
* **topics.db** – теми занять і ваги відміток.

Для повного очищення електронного журналу просто видаліть зазначені файли (без видалення саму папку **db**). При введенні нових даних нові файли створяться автоматично.

Перед початком нового навчального року має сенс зберегти колишні файл **staff.db** (Щоб не вводити дані співробітників заново), а інші файли видалити.

**Перегляд архіву і журналу.** Архівні файли журналу за минулі роки (папка **db**) зберігаються локально головним адміністратором журналу.

Перегляд журналу здійснюється в браузері за URL **http: // localhost: 8080**

**Висновок до розділу 3**

В даному розділі було розглянуто основні модулі та процеси, які необхідні для реалізації веб-додатку електронного журналу. Визначено головні файлі додатку та описано їх взаємодія. Був описаний інтерфейс користувача, а також описано основні сценарії роботи користувача з програмним продуктом. Показано приклад роботи системи. При реалізації додатку відзначено його особливості.

**ВИСНОВКИ**

В результаті дипломної роботи було розроблено веб-додаток, повністю готовий до використання. Ця програма повністю реалізована з усіма необхідними функціональними можливостями, відповідно до поставленого завдання.

Для розробки проекту було обрано набір (стек) технологій для розробки програмного забезпечення, що легко поєднуються між собою та забезпечують швидкість роботи та можливості для легкого розширення функціоналу розробленого застосунку. Проведено огляд методів і засобів розробки програмної системи. Обґрунтовано вибір створення програмної системи, заснованої на веб-технологіях, а також побудованої за триланковою архітектурою. Це дає змогу підвищити гнучкість та зручність системи, як у розробці та супроводі, так і у використані.

Важливим критерієм вибору технологій була відкритість коду та ліцензій на дані технології. Абсолютно усі використані технології є вільними та безпечними, бо популярне відкрите програмне забезпечення постійно перевіряється великою спільнотою розробників, що його використовують. Крім того, важливим аспектом даного вибору є те, що використання даного програмного забезпечення безкоштовно і безпечно із юридичної точки зору. Безкоштовність використаного програмного забезпечення дозволило розробити великий веб-застосунок майже без фінансових затрат.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Azat Mardan Master Express.js: The Node.js Framework For Your Web Development. Apress, 2014. 372 с.
2. Э. Фримен. Изучаем HTML, XHTML и CSS = Head First HTML with CSS & XHTML. – П.: «Питер», 2010. – 656 с.
3. Сікорський П. І. Дидактичні поняття кредиту і модуля в контексті Болонського процесу // Шлях освіти. – 2004. – №2. – с. 19.
4. Стивен Шафер. HTML, XHTML и CSS. Библия пользователя, 5-е издание = HTML, XHTML, and CSS Bible, 5th Edition. – М.: «Диалектика», 2010. – 656 с.
5. Методичні вказівки до виконання магістерської роботи освітнього рівня – магістр‖ студентами усіх форм навчання для напряму підготовки 121 – Інженерія програмного забезпечення // [М. Р. Петрик, Д. М. Михалик, Я. І. Кінах та ін.]. – Тернопіль: ТНТУ імені Івана Пулюя, 2016. – 26 с.
6. API в веб-приложениях : [Електронний ресурс] – режим доступу до ресурсу : [https://mkdev.me/posts/chto-takoe-api-v-veb-prilozheniyah-i-zachem-on-](https://mkdev.me/posts/chto-takoe-api-v-veb-prilozheniyah-i-zachem-on-nuzhen)
[nuzhen](https://mkdev.me/posts/chto-takoe-api-v-veb-prilozheniyah-i-zachem-on-nuzhen)
7. CSS [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/CSS
8. developer.mozilla.org [Електронний ресурс]: режим доступу: https://developer.mozilla.org/ru/docs/Learn/Getting\_started\_with\_the\_web/
How\_the\_Web\_works
9. Hellas A. et. al. Predicting academic performance: a systematic literature review. In Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2018 Companion). ACM, New York, NY, USA, 2018. Pp. 175–199. doi: https://doi.org/10.1145/3293881.3295783
10. Martin Fowler – GUI Architectures. Часть 1 [Електронний ресурс]. – 2009. – Режим доступу: https://bit.ly/2CvCk1e
11. Martin Fowler – GUI Architectures. Часть 2 [Електронний ресурс]. – 2009 – Режим доступу: <https://habr.com/post/53536/>
12. Mathieu D. Node.js и JavaScript для серверной разработки [Електронний ресурс] / Dionne Mathieu – Режим доступу до ресурсу: <https://habr.com/ru/company/ruvds/blog/345164/>
13. NeDB [Електронний ресурс] – Режим доступу до ресурсу: <https://dbdb.io/db/nedb>
14. NeDB: аналог SQLite для NodeJS [Електронний ресурс]. – Режим доступу: <https://habr.com/ru/post/301916/>
15. NodeJs About Node.js. [Електронний ресурс]: Режим доступу: <https://nodejs.org/en/about/>
16. Node.js documentation [Електронний ресурс]: режим доступу: <https://nodejs.org/uk/docs/>
17. What is REST? [Електронний ресурс] – Режим доступу: http://www.restapitutorial.com/lessons/whatisrest.html
18. Working with Web API [Електронний ресурс] – режим доступу до ресурсу : <https://launchschool.com/books/working_with_apis>
19. World Economic Forum Global Competitiveness Report. 2011. URL: http://www3.weforum.org/docs/WEF\_GCR\_Report\_2011-12.pdf
20. Державна служба статистики України. [Електронний ресурс] – Режим доступу: http://www.ukrstat.gov.ua/
21. От Web-сайтов к Web-приложениям: Часть 1. Web-сайт или Webприложение? [Електронний ресурс]. – режим доступу до ресурсу: <http://www.ibm.com/developerworks/ru/library/wa-websiteapp/>
22. Специфікація html [Електронний ресурс] – Режим доступу: https://html.spec.whatwg.org
23. Що таке CSS і де він використовується? [Електронний ресурс] – Режим доступу: <https://ukr.4meahc.com/what-is-css-where-is-it-used-41800>

ДОДАТКИ

**ДОДАТОК А**

**server.js**

/\*\*

 \* СЕРВЕР ЭЛЕКТРОННОГО ЖУРНАЛА «Studenote»

 \*/

"use strict";

/\* ПОДКЛЮЧЕНИЕ МОДУЛЕЙ И ОПРЕДЕЛЕНИЕ ГЛОБАЛЬНЫХ КОНСТАНТ

 \* ----------------------------------------------------------------------- \*/

const DOCROOT = \_\_dirname + "/www",

 https = require("https"),

 fs = require("fs"),

 nedb = require("nedb"),

 {

 PORT, SERVER, ERR404, MIME, PWD, SALT, SALTPIN,

 ADMIN, KEYPATH, CERTPATH, CAPATH

 } = require("./config"),

 api = require("./api"),

 captGen = require("./api/captchaGen"),

 httpsOpt = {

 key: fs.readFileSync(\_\_dirname + "/ssl/" + KEYPATH),

 cert: fs.readFileSync(\_\_dirname + "/ssl/" + CERTPATH)

 };

 if (CAPATH) httpsOpt.ca = CAPATH;

global.salt = SALT;

global.saltpin = SALTPIN;

global.admPwd = PWD;

/\* ИНИЦИАЛИЗАЦИЯ КОЛЛЕКЦИЙ БАЗЫ ДАННЫХ

 \* ----------------------------------------------------------------------- \*/

const dbTables = [

 "staff", "pupils", "curric", "distrib", "grades", "spravki", "topics",

 "authlog", "notes"

];

global.db = {};

for (let dbN of dbTables) db[dbN] =

 new nedb({filename: `${\_\_dirname}/db/${dbN}.db`, autoload: true});

/\* ОПРЕДЕЛЕНИЯ ФУНКЦИЙ

 \* ----------------------------------------------------------------------- \*/

// Запись серверного лога

const putlog = (ip, reqMeth, pathname, kodOtv, lengthOtv) => {

 let

 now = new Date(),

 y = now.getFullYear(),

 m = (now.getMonth() + 1).toString().padStart(2, '0'),

 d = now.getDate().toString().padStart(2, '0'),

 h = now.getHours().toString().padStart(2, '0'),

 i = now.getMinutes().toString().padStart(2, '0'),

 s = now. getSeconds().toString().padStart(2, '0'),

 dt = `${y}-${m}-${d}`,

 tm = `${h}:${i}:${s}`;

 // Пишем данные в серверный лог

 fs.appendFile(

 \_\_dirname + `/logs/${dt}.log`,

 `${ip} [${tm}] ${reqMeth} ${pathname} ${kodOtv} ${lengthOtv}\n`,

 e => {}

 )

 // Пишем успешный запрос авторизации в коллекцию authlog

 if (pathname.includes(" login]") && lengthOtv > 5) {

 let loginArr = pathname.replace(/[\[\], ]/g, '').replace("login", '')

 . split('\_'),

 login = loginArr[0],

 categ = loginArr[1] || "root";

 db.authlog.insert({d: `${dt} ${tm}`, l: login, c: categ, ip: ip});

 }

};

// Генерирование числового значения капчи по её Id

// (используется также для генерирования родительских паролей из детских)

global.captNumGen = str => {

 let captNum = '', s, h = 0;

 for (let j = 0; j < 6; j++) {

 s = global.salt + j + str;

 for (let i=0; i<s.length; i++) h = ((h << 5) - h) + s.charCodeAt(i);

 captNum += Math.abs(h) % 10;

 }

 return captNum;

}

// Промисификатор метода find() работы с базой db

// Пример вызова: let res = await dbFind("curric", {type: "class"})

global.dbFind = (collectionName, objFind) => {

 return new Promise((resolve, reject) => {

 db[collectionName].find(objFind, (err, docs) => {

 if (err) reject(err);

 else resolve(docs);

 })

 })

};

// Изготавление хэша длины 24 из строки str с солью slt

global.hash = (str, slt) => {

 let

 alph = "0123456789AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz",

 char,

 strNew,

 h = 0,

 pass = '';

 for (let j = 0; j < 24; j++) {

 strNew = slt + j + str;

 for (let i = 0; i < strNew.length; i++) {

 char = strNew.charCodeAt(i);

 h = ((h << 5) - h) + char;

 }

 pass += alph[Math.abs(h) % alph.length];

 }

 return pass;

}

// Отправка ответа (kod - код состояния, contType - mime-тип, content - тело)

const sendOtvet = (otvet, kod, contType, content) => {

 otvet.writeHead(kod, {

 "Content-Type": contType, "Server": SERVER,

 "Strict-Transport-Security": "max-age=32000000"

 });

 otvet.end(content);

}

/\* ОБЪЕКТЫ, ОБСЛУЖИВАЮЩИЕ РАБОТУ С КАПЧЕЙ

 \* ----------------------------------------------------------------------- \*/

// Параметры отдаваемой капчи

const captOpt = {

 bkR: 246, bkG: 243, bkB: 240, // фоновый цвет

 fnR: 214, fnG: 191, fnB: 168, // цвет шрифта

}

// Массив выданных сервером клиенту ID капчи и время жизни капчи в секундах

// (те, что вернулись от клиента, а также старые удаляются)

global.captchaIdArr = [];

const CAPTDEATH = 180;

/\* СОБСТВЕННО ЦИКЛ ОБРАБОТКИ ЗАПРОСА

 \* ----------------------------------------------------------------------- \*/

https.createServer(httpsOpt, (zapros, otvet) => {

 // Получаем параметры запроса

 let url = new URL("http://host" + zapros.url),

 pathname = url.pathname;

 if (!pathname.includes(".")) pathname += "/index.html";

 pathname = pathname.replace("//", '/').replace(/\.\./g, '');

 let ADDR = (zapros.socket.remoteAddress || "unknown")

 . replace("::1", "127.0.0.1").replace(/\:.\*\:/, '');

 // Если пришел запрос контактов администратора

 if (pathname == "/a.a") sendOtvet(otvet, 200, "text/plain", ADMIN);

 // Если пришел запрос капчи, отдаем ее вместе с ее Id (в заголовке X-Cpt)

 else if (pathname == "/cpt.a") {

 let tm = Date.now();

 // Удаляем все устаревшие Id капчи и кладем новый Id

 captchaIdArr = captchaIdArr.filter(

 x => Number(x) > Number(tm - CAPTDEATH \* 1000));

 captchaIdArr.push(tm);

 otvet.writeHead(200,

 {"Content-Type": "image/png", "Server": SERVER, "X-Cpt": tm});

 otvet.end(captGen(captNumGen(tm), captOpt));

 }

 // Если метод GET, просто отдаем запрошенный статический файл

 else if (zapros.method == "GET")

 fs.readFile(DOCROOT + pathname, function(err, cont) {

 let mtip = MIME[pathname.split(".")[1]];

 if (!mtip || err) {

 sendOtvet(otvet, 404, "text/html", ERR404);

 putlog(ADDR, "GET", pathname, 404, ERR404.length);

 }

 else {

 sendOtvet(otvet, 200, mtip, cont);

 // Из успешных GET-запросов логируем только запрос главной

 if (pathname == "/index.html")

 putlog(ADDR, "GET", '/', 200, cont.length);

 }

 });

 // Если метод POST - это запрос к API

 else {

 let postData = '';

 zapros.on("data", dann => postData += dann.toString());

 zapros.on("end", async () => {

 let cont = await api(postData, ADDR);

 sendOtvet(otvet, 200, "text/plain", cont);

 // Определяем логин и запрашиваемую функцию API;

 // пишем логин и запрашиваемую функцию в серверный лог

 // (если функция содержится в списке логируемых функций),

 // а успешный запрос авторизации - еще и в базу (authlog)

 // с помощью функции putlog (определена выше)

 let logCont = '';

 let logFuncs = [

 "login", "classAdd", "classDel", "subjAdd", "subjEdit", "subjDel",

 "usAddEdit", "usImport", "usSetAdmin", "usBlock", "usChPwd",

 "tutorSet", "distrEdit", "topicEdit", "gradeAdd", "subgrEdit",

 "subgrPups", "sprAdd", "sprDel", "notesAdd", "notesDel",

 "interGroupEdit", "interGroupDel", "interGroupPup"

 ];

 try {

 let postDataObj = JSON.parse(postData);

 let logLogin = postDataObj.l || "none";

 let logFunc = postDataObj.f || "none";

 let logRole = `\_${postDataObj.t}` || "\_none";

 if (logLogin == "admin") logRole = '';

 if (logFuncs.includes(logFunc))

 logCont = `[${logLogin}${logRole} ${logFunc}]`;

 }

 catch(e) {;}

 let codeOtv = (cont == "none") ? 403 : 200;

 if (logCont) putlog(ADDR, "POST", logCont, codeOtv, cont.length);

 });

 }

}).listen(PORT);

// Перенаправление с http на https

const http = require("http");

http.createServer((zapros, otvet) => {

 try {

 otvet.writeHead(

 301, {"Location": "https://" + zapros.headers["host"] + zapros.url}

 ); otvet.end();

 }

 catch (e) {

 otvet.writeHead(404, {"Content-Type": "text/html", "Server": SERVER});

 otvet.end(ERR404);

 }

}).listen(80);

let now = (new Date()).toString().replace(/ \(.\*\)/, '');

console.info(`${now} ScoleServer стартував на порту ${PORT}`);

**ДОДАТОК B**

**index.js**

/\*\*

 \* API ЭЛЕКТРОННОГО ЖУРНАЛА «Studenote»

 \*/

"use strict";

// Подключение необходимых модулей API (модуль авторизации подключен отдельно)

// Имя модуля - ключ, а значения - это пара вида [0,1],

// где первое значение - требуется ли await при вызове модуля,

// второе значение - передается ли модулю объект аргументов postDt.z

// Не забыть добавить функцию еще и в список логируемых функций (если нужно)

// в скрипте server.js, массив logFuncs

const modReq = {

 "classAdd": [1,1], "classesList": [1,0], "classDel": [0,1],

 "interGroupEdit": [1,1], "interGroupList": [1,0], "interGroupDel": [0,1],

 "subjList": [1,0], "subjAdd": [1,1], "subjEdit": [1,1],

 "subjDel": [0,1], "usAddEdit": [1,1], "usFind": [1,1],

 "usFindLogin": [1,1], "usImport": [1,1], "usSetAdmin": [1,1],

 "adminsList": [1,0], "usBlock": [1,1], "usChPwd": [1,1],

 "teachList": [1,0], "tutorSet": [0,1], "tutorsList": [1,0],

 "distrGet": [1,0], "distrEdit": [1,1], "classesGroups": [1,0],

 "topicEdit": [1,1], "topicsGet": [1,1], "gradesGet": [1,1],

 "gradeAdd": [1,1], "subgrEdit": [1,1], "subgrPups": [1,1],

 "parCodes": [1,1], "jrnGet": [1,1], "absentGet": [1,1],

 "pupilsList": [1,1], "sprAdd": [1,1], "sprGet": [1,1],

 "sprDel": [1,1], "sprResp": [1,1], "tabelGet": [1,1],

 "export": [1,1], "notesAdd": [1,1], "notesGet": [1,1],

 "notesDel": [1,1], "logGet": [1,1], "notesCheck": [1,1],

 "statGet": [1,1], "interGroupPup": [1,1], "interGroupGet": [1,1],

 "attendGet": [1,1], "electGet": [1,1]

};

let mod = {};

mod.auth = require("./auth");

for (let modName in modReq) mod[modName] = require("./" + modName);

// Полномочия (доступные модули) пользователей в зависимости от их роли

const RIGHTS = {

 "root": [

 "classAdd", "classesList", "classDel", "subjList", "subjAdd", "subjEdit",

 "subjDel", "usAddEdit", "usFind", "usFindLogin", "usImport", "usSetAdmin",

 "adminsList", "usBlock"

 ],

 "admin": [

 "classesList", "subjList", "teachList", "tutorSet", "tutorsList",

 "distrGet", "distrEdit", "classesGroups", "topicsGet", "gradesGet",

 "absentGet", "pupilsList", "sprResp", "export", "notesAdd", "notesGet",

 "notesDel", "logGet", "statGet", "interGroupEdit", "interGroupList",

 "interGroupDel", "attendGet"

 ],

 "teacher": [

 "usChPwd", "subjList", "topicEdit", "topicsGet", "gradesGet", "gradeAdd",

 "notesAdd", "notesGet", "notesDel", "interGroupList", "usFind",

 "interGroupPup", "interGroupGet", "attendGet"

 ],

 "tutor": [

 "subjList", "distrGet", "teachList", "classesGroups", "topicsGet",

 "gradesGet", "subgrEdit", "subgrPups", "parCodes", "absentGet",

 "pupilsList", "sprAdd", "sprGet", "sprDel", "sprResp", "tabelGet",

 "export", "notesAdd", "notesGet", "notesDel", "logGet", "attendGet",

 "interGroupList", "electGet"

 ],

 "pupil": [

 "subjList", "teachList", "jrnGet", "absentGet", "sprGet", "sprResp",

 "tabelGet", "notesGet", "notesCheck"

 ],

 "parent": [

 "subjList", "teachList", "jrnGet", "absentGet", "sprGet", "sprResp",

 "tabelGet", "notesGet", "notesCheck"

 ]

};

for (let item in RIGHTS) RIGHTS[item].push("login");

module.exports = async (post, addr) => {

 // Разбираем переданные в аргументе POST-данные

 let postDt = {};

 try {postDt = JSON.parse(post);} catch (e) {return "none";}

 if (!postDt.t) postDt.t = "noType";

 if (!postDt.f) postDt.f = "noFunc";

 if (!postDt.l) postDt.l = "noLogin";

 if (!postDt.p) postDt.p = "noPassw";

 if (!postDt.ci) postDt.ci = "noCptId";

 if (!postDt.c) postDt.c = "noCapt";

 // Проверяем результаты аутентификации юзера

 let authResult = await mod.auth(

 postDt.t, postDt.l, postDt.p, postDt.ci, postDt.c, addr);

 if (!authResult) return "none";

 // Проверяем полномочия юзера на запрашиваемую функцию

 let rolesArr = JSON.parse(authResult)["roles"];

 if (!rolesArr.some(r => RIGHTS[r].includes(postDt.f))) return "none";

 // Подписываем логин юзера в массив аргументов, передающийся модулю API

 // (для некоторых функций API, требующих валидного логина юзера)

 let fNames = [

 "topicEdit", "gradeAdd", "subgrEdit", "subgrPups", "parCodes", "jrnGet",

 "absentGet", "sprAdd", "sprGet", "sprDel", "sprResp", "tabelGet",

 "export", "notesAdd", "notesGet", "notesDel", "logGet", "statGet",

 "interGroupPup", "attendGet", "electGet"

 ];

 if (fNames.includes(postDt.f) && postDt.z) postDt.z.push(postDt.l);

 // Реализуем соответствующую функцию api в зависимости от переменной f

 // и необходимости использования await и передачи модулю аргументов

 if (postDt.f == "login") return authResult;

 if (!Object.keys(modReq).includes(postDt.f)) return "none";

 let argsObj = {};

 if (modReq[postDt.f][1]) {

 if (!postDt.z) return "none"; // аргументы модулю нужны, но они не пришли

 argsObj = postDt.z;

 }

 if (modReq[postDt.f][0]) return (await mod[postDt.f](argsObj));

 else return mod[postDt.f](argsObj);

}