ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

ПОЯСНЮВАЛЬНА ЗАПИСКА

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: **«РОЗРОБКА ВЕБ-СЕРВІСУ КОНТРОЛЮ НАВЧАЛЬНОГО ПРОЦЕСУ СТУДЕНТІВ МОВОЮ JAVA»**

Виконав: студент 4 курсу, групи ПД– 41
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
_____Кухаренко Ю.Д._____
(прізвище та ініціали)
Керівник ____Трінтіна Н.А_____
(прізвище та ініціали)
Рецензент_____
(прізвище та ініціали)

Київ – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти - «Бакалавр»

Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного забезпечення

_____Негоденко О.В.

“____” _____2022 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТА

Кухаренко Юрій Дмитрович

(прізвище, ім’я, по батькові)

1. Тема роботи: «Розробка веб-сервісу контролю навчального процесу студентів мовою Java»

Керівник роботи: Трінтіна Наталія Альбертівна, кандидат технічних наук, доцент

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «18» лютого 2022 року №.

2. Строк подання студентом роботи «3» червня 2022 року

3. Вхідні дані до роботи:

3.1 Науково-технічна література з питань, пов’язаних з програмним забезпеченням щодо розробки веб-додатків.

3.2 Офіційна документація мови Java.

3.3 Офіційна документація Intellij IDEA.

4. Зміст розрахунково-пояснювальної записки(перелік питань, які потрібно розробити).

4.1 Аналіз актуальності та проблематики розроблюваного веб-сервісу.

4.2 Аналіз та вибір інструментів для реалізації продукту.

4.3 Проектування веб-сервісу.

4.4 Висновки.

5. Перелік демонстраційного матеріалу (назва основних слайдів)

5.1 Титульний слайд.

5.2 Мета, об’єкт, предмет, ноукова новизна дослідження.

5.3 Аналіз аналогів.

5.4 Технічні завдання.

5.5 Програмні засоби та інструменти реалізації.

5.6 Архітектура системи

5.7 Use case та діаграма класів програми

5.8 Приклад застосування

5.9 Наукова новизна та практична значисмість

5.10 Висновки.

5.11 Кінцевий слайд.

6. Дата видачі завдання «12» квітня 2022р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	12.04. – 15.04	Виконано
2	Дослідження аналогів та актуальності додатку	16.04 – 22.04	Виконано
3	Аналіз та вибір інструментів для розробки додатку	22.04 – 27.04	Виконано
4	Проектування та реалізація	27.04 – 10.05	Виконано
5	Вступ, висновки, реферат	11.05 – 15.05	Виконано
6	Розробка обов‘язкових демонстраційних матеріалів	16.05	Виконано
7	Попередній захист роботи		
8	Здача роботи	3.06.2022	

Студент _____ Кухаренко Ю.Д.

(підпис) (прізвище та ініціали)

Керівник роботи _____ Трінтіна Н.А.

(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

В сучасному світі кожного дня кількість інформації, яку необхідно зберігати росте в геометричній прогресії , набуваючи величезних розмірів. В певний момент старі варіанти зберігання інформації, такі як паперові носії, або навіть усна форма, віджили своє і більше не могли задовольняти потреби сучасного соціуму.

Саме в цей момент і з’явилися системи автоматизації, які допомагають людям у структуруванні та зберіганні інформації різного роду, взаємодії між ролями користувачів, тощо.

На сьогоднішній день багато підприємств та установ мають свої внутрішні сервіси, які допомагають зберігати і структурувати інформцію про роботу даної установи, автоматизувати деякі процеси, тощо.

Виходячи з актуальності явища систем автоматизації, об’єктом дослідження є інформаційні системи і їх використання в різного роду структурах.

Предмет дослідження — система контролю навчального процесу студентів.

Мета роботи — розроблення системи контролю навчального процесу студентів.

Для досягнення поставленої мети слід виконати наступні завдання:

- Провести огляд поняття веб-додатку
- Провести аналіз методів та засобів розробки веб-сайтів
- Провести огляд поняття контролю навчання
- Провести огляд існуючих рішень
- Обрати мову програмування
- Обрати додаткові інструменти
- Обрати середовище розробки
- Провести аналіз варіантів використання
- Провести проектування внутрішньої будови
- Розробити графічний інтерфейс користувача
- Провести тестування системи

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Поняття веб-додатку

Веб-додаток — це прикладне програмне забезпечення, запущене на веб-сервері. Від настільних програм його відрізняє те, що вони запускаються локально в операційній системі. Аби отримати доступ до веб-додатку, користувачу необхідно мати мережеве підключення. Працюють такі додатки за структурою клієнт. Серед широко використовуваних веб-додатків можна виділити наступні: онлайн-банкінги, інтернет магазини, аукціони, торгові майданчики, відео-хостинги, ектронна пошта.

Насправді, веб-додатки мають багато функціональних подібностей до звичайного ПО для настільких комп’ютерів. З появою HTML5 розробники отримали можливість створювати програми, що будуть завантажені як веб-сторінки, але матимуть змогу зберігати дані локально на комп’ютері користувача та працювати автономно.

Історія

Раніше у архітектурі типу клієнт-сервер навантаження на програму було розподілене між кодом, що працює на віддаленому сервері та кодом, що локально встановлений на кожному клієнті. Простіше кажучи, клієнтська програма уже була скомпільована і виступала як інтерфейс користувача для взаємодії з кодом на сервері. Витрати на підтримку такого роду структур були значно високими, оскільки зміни та оновлення серверного коду тягло за собою зміни коду на стороні клієнта, а це в свою чергу негативно впливало на продуктивність. На додаток, такі програми часто писались під конкретну комп’ютерну архітектуру і ОС, тож ресурсоємною була їх міграція на інші платформи.

Веб-додатки ж, на відміну від наведеної вище архітектури, за основу мають веб-документи, написані за допомогою HTML, CSS та JavaScript, і можуть запускатись однаково у різних веб-браузерах. Клієнтське програмне забезпечення

завантажується за допомогою таких процедур як HTTP тоді, коли користувач переходить на веб-сторінку. При цьому сторінка може оновлюватись при кожному новому її відвідуванні. Проте, при зміні веб сторінки, необхідно було заново її завантажувати.

У 1995 році Netscape представив мову сценаріїв на стороні клієнта під назвою JavaScript, що дозволяє програмістам додавати деякі динамічні елементи до інтерфейсу користувача, який працює на стороні клієнта. Таким чином, замість надсилання даних на сервер, щоб створити всю веб-сторінку, вбудовані сценарії завантаженої сторінки можуть виконувати різні завдання, такі як перевірка введених даних або відображення/приховування частин сторінки.

У 1996 році Macromedia представила Flash, програвач векторної анімації, який можна було додати до браузерів як плагін для вбудовування анімації на веб-сторінки. Це дозволило використовувати мову сценаріїв для програмування взаємодії на стороні клієнта без необхідності спілкуватися з сервером.

У 1999 році концепція «веб-додаток» була представлена мовою Java у специфікації сервлетів версії 2.2. На той час і JavaScript, і XML вже були розроблені, але Ajax ще не був придуманий, а об’єкт XMLHttpRequest лише нещодавно був представлений в Internet Explorer 5 як об’єкт ActiveX.

У 2005 році був придуманий термін Ajax, і такі програми, як Gmail, почали робити свої клієнтські сторони все більш інтерактивними. Сценарій веб-сторінки може зв'язатися з сервером для зберігання/отримання даних, не завантажуючи всю веб-сторінку.

У 2007 році Стів Джобс оголосив, що веб-додатки, розроблені на HTML5 з використанням архітектури AJAX, будуть стандартним форматом у додатках для iPhone. Набір для розробки програмного забезпечення (SDK) не потрібен, і програми будуть повністю інтегровані в пристрій за допомогою браузера Safari. Пізніше ця модель була переведена на App Store, щоб запобігти джейлбрейкерам і заспокоїти розчарованих розробників.

У 2014 році було завершено розробку HTML5, який надає графічні та мультимедійні можливості без потреби клієнтських плагінів. HTML5 також

збагатив семантичний зміст документів. API та об’єктна модель документа (DOM) більше не є запізнілими, а є фундаментальними частинами специфікації HTML5. WebGL API проклав шлях до розширеної 3D-графіки на основі полотна HTML5 і мови JavaScript. Вони мають важливе значення для створення дійсно незалежних від платформи та браузера багатофункціональних веб-додатків.

У 2016 році під час щорічної конференції Google IO Ерік Бідельман (старший інженер з розробки програм) представив прогресивні веб-додатки (PWA) як новий стандарт у веб-розробці. Джефф Бертофт, головний програмний менеджер Microsoft, сказав, що «Google лідирував із прогресивними веб-додатками, і після тривалого процесу ми вирішили, що нам потрібно повністю підтримувати його». Таким чином, і Microsoft, і Google підтримували стандарт PWA.

Інтерфейс

За допомогою JavaScript, CSS та історично Java, Flash, Silverlight можливі спеціальні методи програми, такі як малювання на екрані, відтворення звуку та доступ до клавіатури та миші. Багато служб працювали, щоб об’єднати все це в більш знайомий інтерфейс, який нагадує зовнішній вигляд операційної системи. Веб-розробники часто використовують сценарії на стороні клієнта, щоб додати функціональність, особливо для створення інтерактивного досвіду, який не вимагає перезавантаження сторінки. Нещодавно були розроблені технології для координації сценаріїв на стороні клієнта з технологіями на стороні сервера, такими як ASP.NET, J2EE, Perl/Plack і PHP.

Ajax, техніка веб-розробки, що використовує комбінацію різних технологій, є прикладом технології, яка створює більш інтерактивний досвід.

Структура

Програми прийнято розбивати на логічні блоки, які називаються «рівні», де кожен рівень має свою роль. Настільні програми частіше за все мають один рівень, який встановлений на клієнтській машині, але веб-додатки за своєю природою проектуються використовуючи n-рівневий підхід. Традиційно виділяють мінімум 3 рівні: додаток, презентація, зберіганням. Перший рівень(презентація) – це веб-браузер. Веб-браузер використовує технології динамічного веб-контенту. Такими

технологіями є, наприклад, Node.js, Dart, PHP, Python, JSP/Java Ruby on Rails, ASP. Вони являють собою другий, середній рівень — це логіка додатку. Останнім рівнем є база даних, що виступає сховищем всієї інформації. Працює це таким чином: веб-браузер надсилає запити на середній рівень, який обслуговує їх, виконуючи запити та оновлення бази даних, і створює інтерфейс користувача, тобто, презентацію, що ми безпосередньом бачимо у вигляді веб-сторінок.

Не всі веб-додатки можливо спроектувати в рамках трьох рівнів і саме тоді використовують n-рівневий підхід, в якому бізнес-логіка розбивається на дрібні моделі. Також, вагомим плюсом такого підходу може бути відокремлення рівня сховища від інших за допомогою додавання рівня інтеграції. Це забезпечить зручний і незалежний інтерфейс доступу до даних

Іноді веб-додаток представляють як дворівнену систему. З одного боку, це коли клієнт виконує основну роботу і надсилає запити на «німий» сервер, або ж навпаки, коли основна робота виконується на веб-сервері, а клієнт на нього покладається. При такому розкладі клієнт відповідає за рівень презентації, сервер має доступ до бази даних, а рівень логіки розташований на одному з них або обох.

1.2 Основні принципи веб-розробки

Веб-сайт - це сукупність веб-сторінок та пов’язаного вмісту, які ідентифікуються загальним доменним ім’ям та публікуються принаймні на одному веб-сервері. Помітними прикладами є wikipedia.org, google.com та amazon.com.

Усі загальнодоступні веб-сайти в сукупності складають Всесвітню павутину. Існують також приватні веб-сайти, доступ до яких доступний лише в приватній мережі, наприклад, внутрішній веб-сайт компанії для її співробітників.

Веб-сайти, як правило, присвячені певній темі або меті, такі як новини, освіта, комерція, розваги чи соціальні мережі. Гіперпосилання між веб-сторінками спрямовує навігацію по сайту, яке часто починається з домашньої сторінки.

Користувачі можуть отримати доступ до веб-сайтів на різних пристроях, включаючи настільні, ноутбуки, планшети та смартфони. Програма, що використовується на цих пристроях, називається веб-браузером.

Всесвітня павутина (WWW) була створена в 1990 році британським фізиком ЦЕРН Тімом Бернерсом-Лі. 30 квітня 1993 р. ЦЕРН оголосив, що Всесвітня павутина буде безкоштовною для використання будь-ким. До впровадження протоколу передачі гіпертексту (HTTP) для отримання окремих файлів із сервера використовувались інші протоколи, такі як протокол передачі файлів та протокол gopher. Ці протоколи пропонують просту структуру каталогів, в якій користувач обирає файли для завантаження. Документи найчастіше подавалися у вигляді текстових файлів без форматування або кодувались у форматах текстового процесора.

Веб-сайти можуть використовуватися по-різному: персональний веб-сайт, корпоративний веб-сайт компанії, урядовий веб-сайт, веб-сайт організації тощо. Веб-сайти можуть бути роботою приватної особи, бізнесу чи іншої організації і, як правило, присвячені конкретна тема або мета. Будь-який веб-сайт може містити гіперпосилання на будь-який інший веб-сайт, тому різниця між окремими сайтами, як сприймається користувачем, може бути розмитою.

Деякі веб-сайти вимагають реєстрації користувачів або передплати для доступу до вмісту. Приклади веб-сайтів, на які здійснюється підписка, включають багато бізнес-сайтів, веб-сайти новин, веб-сайти академічних журналів, ігрові веб-сайти, веб-сайти спільного використання файлів, дошки оголошень, веб-адреси електронної пошти, веб-сайти соціальних мереж, веб-сайти, що надають дані про фондовий ринок у режимі реального часу та багато інших.

Хоча "веб-сайт" був оригінальним написанням (іноді з великої літери "Веб-сайт", оскільки "Веб" є власним іменником, коли йдеться про Всесвітню павутину), цей варіант став рідкісним, а "веб-сайт" став стандартним написанням . Усі основні керівництва стилем, такі як Чиказький посібник стилю та AP Stylebook , відображають цю зміну.

Статичний веб-сайт - це веб-сторінки, що зберігаються на сервері у форматі, який надсилається клієнтському веб-браузеру. В більшості випадків він написаний мовою розмітки гіпертексту HTML. CSS – це каскадні таблиці стилів, використовуються для контролю зовнішнього вигляду за базовим HTML. Зображення зазвичай використовуються для отримання бажаного вигляду та як частина основного змісту. Аудіо чи відео також можна вважати "статичним" вмістом, якщо він відтворюється автоматично або, як правило, є неінтерактивним. Цей тип веб-сайтів зазвичай відображає однакову інформацію для всіх відвідувачів. Подібно до роздачі друкованої брошури клієнтам, статичний веб-сайт, як правило, надає послідовну стандартну інформацію протягом тривалого періоду часу. Незважаючи на те, що власник веб-сайту може періодично робити оновлення, редагування тексту, фотографій та іншого вмісту здійснюється вручну, і можуть знадобитися базові навички дизайну веб-сайту та програмне забезпечення. Маркетингові приклади веб-сайтів, такі як класичний веб-сайт, веб-сайт із декількома сторінками, часто є статичними, оскільки вони представляють користувачеві заздалегідь визначену, статичну інформацію. Це може включати інформацію про компанію та її продукти та послуги через текст, фотографії, анімацію, аудіо / відео та навігаційні меню.

Статичні веб-сайти все ще можуть використовувати серверні компоненти (SSI) як зручність редагування, наприклад, спільний доступ до загального рядка меню на багатьох сторінках. Оскільки поведінка сайту до читача все ще залишається статичною, це не вважається динамічним сайтом.

Динамічний веб-сайт - це веб-сайт, який часто і автоматично змінюється або налаштовується. Динамічні сторінки на сервері генеруються "на льоту" за допомогою комп'ютерного коду. Існує широкий спектр програмних систем, таких як CGI, Java-сервлети та Java Server Pages (JSP), Active Server Pages і ColdFusion (CFML), які доступні для створення динамічних веб-систем та динамічних веб-сайтів. Різні фреймворки веб-програм та системи веб-шаблонів доступні для загальновживаних мов програмування, таких як Perl, PHP, Python та Ruby, щоб полегшити та спростити створення складних динамічних веб-сайтів.

Сайт може відображати поточний стан діалогу між користувачами, відстежувати мінливу ситуацію або надавати інформацію певним чином персоналізовану відповідно до вимог окремого користувача. Наприклад, коли запитується головна сторінка сайту новин, код, що працює на веб-сервері, може поєднувати збережені фрагменти HTML із новинами, отриманими з бази даних або іншого веб-сайту за допомогою RSS, щоб створити сторінку, що включає найсвіжішу інформацію. Динамічні сайти можуть бути інтерактивними, використовуючи HTML-форми, зберігаючи та зчитуючи файли cookie браузера, або створюючи ряд сторінок, що відображають попередню історію кліків. Інший приклад динамічного вмісту - це коли роздрібний веб-сайт з базою даних медіа-продуктів дозволяє користувачеві вводити запит на пошук, наприклад за ключовим словом "Бітлз". У відповідь вміст веб-сторінки спонтанно змінить те, як він виглядав раніше, а потім відобразить список продуктів "Бітлз", таких як компакт-диски, DVD-диски та книги. Динамічний HTML використовує код JavaScript, щоб вказувати веб-браузеру, як інтерактивно змінювати вміст сторінки. Одним із способів моделювання певного типу динамічного веб-сайту, уникаючи втрати продуктивності ініціювання динамічного механізму для кожного користувача або підключення, є періодична автоматична регенерація великої серії статичних сторінок.

Ранні веб-сайти мали лише текст, а незабаром і зображення. Потім плагіни веб-браузера використовувались для додавання аудіо, відео та інтерактивності (наприклад, для багатофункціональної програми Інтернету, яка відображає складність настільної програми, як текстовий процесор). Прикладами таких плагінів є Microsoft Silverlight, Adobe Flash, Adobe Shockwave та аплети, написані на Java. HTML 5 містить положення щодо аудіо та відео без плагінів. JavaScript також вбудований у більшість сучасних веб-браузерів і дозволяє творцям веб-сайтів надсилати код веб-браузеру, який вказує йому, як інтерактивно змінювати вміст сторінки та спілкуватися з веб-сервером, якщо це необхідно. Внутрішнє представлення вмісту браузера відомо як об’єктна модель документа (DOM), а техніка - динамічний HTML.

WebGL (Web Graphics Library) - це сучасний API JavaScript для візуалізації інтерактивної 3D-графіки без використання плагінів. Це дозволяє представити інтерактивний контент, такий як 3D-анімація, візуалізація користувачам найбільш інтуїтивно зрозумілим способом.

Тенденція 2010 року на веб-сайтах під назвою "адаптивний дизайн" забезпечила найкращий досвід перегляду, оскільки надає користувачам макет на основі пристрою. Ці веб-сайти змінюють свій макет відповідно до пристрою або мобільної платформи, що забезпечує багатий досвід користувачів.

Веб-сайти можна розділити на дві великі категорії - статичні та інтерактивні. Інтерактивні сайти є частиною спільноти веб-сайтів Web 2.0 і дозволяють взаємодіяти між власником сайту та відвідувачами або користувачами сайту. Статичні сайти обслуговують або збирають інформацію, але не дозволяють взаємодіяти з аудиторією або користувачами безпосередньо. Деякі веб-сайти є інформаційними або виготовляються ентузіастами для особистого користування чи розваги. Багато веб-сайтів мають на меті заробляти гроші, використовуючи одну або кілька бізнес-моделей, зокрема:

- Розміщення цікавого контенту та продаж контекстної реклами або шляхом прямих продажів, або через рекламну мережу.
- Електронна комерція: товари чи послуги купуються безпосередньо через веб-сайт.
- Реклама товарів або послуг.
- Freemium: базовий вміст доступний безкоштовно, але преміум-вміст вимагає оплати (наприклад, веб-сайт WordPress, це платформа з відкритим кодом для створення блогу чи веб-сайту).

Деякі веб-сайти можуть бути включені в одну або кілька з цих категорій. Наприклад, веб-сайт бізнесу може рекламувати продукцію бізнесу, але також може розміщувати інформативні документи, такі як технічні документи. Існують також численні підкатегорії до перерахованих вище. Наприклад, порносайт - це певний тип веб-сайту електронної комерції або бізнес-сайту (тобто

він намагається продати членство для доступу до свого сайту) або має можливості соціальних мереж. Фан-сайт може бути посвятою власника певній знаменитості. Веб-сайти обмежені архітектурними обмеженнями (наприклад, обчислювальна потужність). Дуже великі веб-сайти, такі як Facebook, Yahoo!, Microsoft та Google, використовують багато серверів та обладнання для балансування навантажень, наприклад, комутатори служб вмісту Cisco, для розподілу навантажень відвідувачів на декілька комп'ютерів у різних місцях. На початок 2011 року Facebook використовував 9 центрів обробки даних із приблизно 63000 серверами.

У лютому 2009 року компанія Netcraft, компанія з моніторингу Інтернету, яка відслідковувала зростання Інтернету з 1995 р., повідомила, що в 2009 р. існувало 215 675 903 веб-сайтів з доменними іменами та вмістом на них, порівняно з лише 19 732 веб-сайтами в серпні 1995 р. Після досягнення 1 мільярда веб-сайтів у вересні 2014 року - етап, підтверджений NetCraft в огляді веб-серверів за жовтень 2014 року, і те, що Інтернет-статистика стала першим, хто повідомив про це, про що свідчить цей твіт від самого винахідника Всесвітньої мережі, Тіма Бернерса-Лі - кількість веб-сайтів у світі згодом зменшилась, повернувшись до рівня нижче 1 мільярда. Це пов'язано з щомісячними коливаннями кількості неактивних веб-сайтів.

Веб-розробка - це ремесло, пов'язане з розробкою веб-сайтів, додатків. Починаючи з розробки однієї статичної сторінки тексту до складних веб-додатків, веб-банків, соціальних мереж. Беручи глобальніше, веб-розробка включає веб-інженерію, веб-дизайн, розробку веб-вмісту, зв'язок з клієнтом, сценарії на стороні клієнта / сервера, налаштування безпеки веб-сервера та мережі та розвиток електронної комерції.

У великих компаніях та бізнесу команди веб-розробників можуть складатися з сотень розробників і дотримуватися традиційних методологій, таких як Agile, під час веб-розробки. Невеликі організації можуть вимагати лише одного постійного або підрядного розробника, або вторинне призначення на відповідні посади, такі як графічний дизайнер або технік інформаційних систем.

З моменту комерціалізації Інтернету веб-розробка стала зростаючою галуззю. Багато підприємств бажають використовувати свій веб-сайт для реклами та продажу товарів або послуг споживачам, що сприяє зростанню цієї галзуі останні роки.

Набір інструментів і технологій веб-розробки постійно зростає, що дає можливість розробникам створювати більш інтерактивні веб-сайти. На додаток, розробники отримали змогу створювати програми у вигляді веб-служб, які початково були доступні лише у вигляді програм на настільному комп'ютері. Це надало небачені раніше можливості дценетралізації розповсюдження засобів масової інформації. Як приклад, це набираючі популярності хмарні сервіси, як от Dropbox, Google Drive, які надають можливість працювати з програмами з будь-якого місця, відмовляючись від прив'язки до певного робочого середовища.

Прикладами кардинальних перетворень у комерції та спілкуванні, чому сприяла саме веб-розробка, є електронна комерція. Сайти аукціонів, як eBay, змінили спосіб споживачів знаходження та купівлі користувачами послуг та товарів. Тим часом інтернет-магазини, на кшталт Amazon.com кардинально змінив досвід користувачів у сфері торгівлі. Прикладом трансформації, керованої веб-розробкою, є блог. Сервісти як WordPress та Movable Type, надають можливість простого ведення блогів в мережі Інтернет. Поширене використання таких систем управління вмістом з відкритим кодом розширило вплив веб-розробки на онлайн-взаємодію та спілкування.

Розробка веб-сайтів має великий вплив на особисті маркетинг та особисті відносини. Веб-сайти зараз – це не просто інструмент для роботи чи розваг, а часто майданчик, що служить для спілкування людей. Такі соціальні мережі, як Facebook та Twitter, надають користувачам платформу де можна обмінюватись думками і підтримувати зв'язок з людьми. Для бізнесу переваги очевидні – це більш особистий та інтерактивний спосіб залучення клієнтів.

У веб-розробці високо стоїть питання безпеки. Такі процедури, як валідація введення даних у форми, фільтрація даних та шифрування є надзвичайно важливими в наш час. Перед випуском проводиться жорстке тестування веб-

додатку, аби запобігти проблемам з безпекою. Наприклад, за наявності на веб-сайті контактної форми, вона повинна мати поле captcha, що не дозволить стороннім програмам автоматично заповнювати форми, а потім розсилати користувачам пошту.

Веб-сервера часто захищають від вторгнень зміцненням порту сервера. Багато технологій застосовуються для захисту інформації в Інтернеті, коли вона передається з одного місця в інше. Аби запобігти шахрайству в мережі органами видаються сертифікати TLS (SSL). Також, при передачі та зберіганні конфіденційної інформації, використовуються різні форми шифрування. Від веб-розробників вимагаються базові знання в сфері безпеки інформаційних технологій.

Оскільки нові веб-програми виявляють нові діри в безпеці навіть після тестування та запуску, оновлення виправлень безпеки часто трапляються для широко використовуваних програм. Часто робота веб-розробників - постійно оновлювати програми, коли випускаються виправлення багів та виявляються нові проблеми безпеки.

1.3 Поняття контролю навчального процесу

Контроль у навчанні – це процес, суть якого полягає у забезпеченні так званого зворотного взаємозв'язку між учнем та вчителем. Він має на меті отримати певну інформацію, яка потім аналізується педагогом для оперативного внесення необхідних коректив у перебіг навчання. Наприклад, зміненню змісту матеріалу, форми, способів навчання. Процеси контролю у навчальному та виховному процесі вже повноцінно відпрацьовані з теоретичної та методичної точки зору. Будучи умовно самостійним кроком, контроль виконує розвиваючу, виховну та навчальну роль. Перевірка знань має важливе значення, що реалізується не тільки в отриманні учнями нових знань при самостійній і груповій роботі, але і в активній участі у вибіркових опитуваннях. Формулювання питань та відповідей сприяє повторенню матеріалу; слухаючи відповіді однокласників, учень повторює вивчене і готується

до того, що він сам може бути запитаний будь-якої миті. У разі неправильної та неповної відповіді він також отримує додаткове пояснення вчителя.

1.4 Огляд існуючих рішень

Рішень, які повноцінну перекривають поняття «системи контролю навчального процесу» як таких не існує у відкритому доступі. Зазвичай, під даним заголовком публікуються рішення, які за своєю суттю є або електронними журналами, або системами створення тестувань з різних тем.

Системи електронних журналів — комерційні проекти, що коштують великих грошей і відсутні у відкритому доступі, що унеможливлює їх огляд.

Серед системи створення тестувань найпопулярнішім (хоча і не призначеним конкретно для таких цілей) можна назвати рішення Google Forms.

Google Forms представляє собою програмне забезпечення, що надає функції для адміністрування опитувань та є одним з сервісів Google. Веб-додаток дає змогу користувачу створювати та редагувати опитування прямо у веб-браузері. Зібрана інформація може бути автоматично введена в електронну таблицю.

Особливості

Протягом багатьох років служба Google Forms оновлювалась неоднократно, разом з чим розширювався її функціонал. Функціонал включає у себе перемішування запитань у опитуванні, обмеження відповідей на одну особу, кастомні теми, генерування відповідей у момент ствrення форм, завантаження файлів у тих випадках, коли відповідь вимагає від користувача роботи з файлами, що розташовані локально на комп’ютері або на Google Диск.

У 2014 році Google Forms зазнали важливих оновлень, суть яких полягала в наданні стороннім розробникам можливості видозмінювати опитування на свій розсуд, додаючи нових функцій. Така функція, як «Інтелектуальна перевірка» виявляє введення невірного тексту в полях форми і повідомляє користувача про те, що необхідно зробити виправлення, для отримання коректного результату. Спільний доступ до файлів на Google Drive дає змогу користувачам завантажувати

файли. В налаштуваннях є можливість задавати шаблони для всіх майбутніх форм, наприклад, можна вказати завжди запитувати номер телефону у того, хто проходить опитування.

Google Forms наділена усіма функціями спільної роботи що наявні в інших сервісах Google, таких як, Таблиці, Документи, Сайти, Презентації.

Окрім рішення від Google популярні такі рішення:

- SurveyMonkey - один із найпопулярніших онлайн-сервісів для проведення опитувань.
- Survio - сервіс для створення опитувань, голосувань та анкет.
- Typeform – сервіс для збору даних для створення форм-опитувальників для блогу або сайту.
- Simpoll – платформа для створення опитувань будь-якої складності.
- Анкетолог – це онлайн-сервіс, призначений для з'ясування громадської думки та проведення досліджень.
- Online Test Pad - сервіс для створення тестів, опитувань, логічних ігор та інших продуктів для дистанційного навчання.
- SurveyGizmo – це сервіс для маркетологів, дослідників та викладачів.
- Formstack – зручний конструктор форм із набором додаткових інструментів.
- Mentimeter – зручний інструмент для створення опитувань.

Нажаль, жодне з цих рішень не перекриває повноцінно всіх потреб користувача у такій системі.

РОЗДІЛ 2
АНАЛІЗ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Огляд мови програмування

Java —об'єктно-орієнтована мова програмування вищого рівня. Є мовою програмування загального призначення, перевагою якої є те, що програмісти можуть писати код один раз, і запускати його в будь-якому місці (WORA), адже скомпільована Java програма коректно працює на всіх платформах, які підтримують Java, і не потребує перекомпіляції. Програми Java компілюються у байт-код, який може працювати на будь-якій віртуальній машині Java (JVM) незалежно від архітектури комп’ютера. З точки зору синтаксису, Java сходжа до C і C++, проте має менше засобів низького рівня. Середовище виконання Java надає вкрай корисні динамічні функції, наприклад, відображення та внесення правок до коду під час його виконання, які є недоступними в класичних компільованих мовах. Відповідно до статистики з GitHub, станом на 2019 рік Java була однією з найпопулярніших мов, що використовуються для розробки веб-додатків.

Першим розробником Java є програміст Sun Microsystems Джеймс Гослінг, а випущена перша версія мови була в травні 1995 року, в якості основного елементу Java-платформи Sun Microsystems. Напочатку всі оригінальні та довідкові компілятори, а також бібліотеки з класами та віртуальні машини були випущені під власною ліцензією Sun. Проте, вже у 2007 році, відповідно до Java Community Process специфікації, компанія переліцензувала більшу частину своїх технологій за ліцензією GPL 2.0. Офіційною реалізацією віртуальної машини Java є OpenJDK JVM, яка безкоштовно розповсюджується і має відкритий вихідний код та є стандартом використання серед розробників.

Станом на березень 2022 року Java 18 є останньою версією, тоді як Java 17, 11 і 8 є поточними версіями довгострокової підтримки (LTS). Oracle випустила останнє загальнодоступне оновлення для застарілої версії Java 8 LTS у січні 2019 року для комерційного використання, хоча в іншому випадку вона все ще

підтримуватиме Java 8 з загальнодоступними оновленнями для особистого використання на невизначений термін. Інші виробники почали пропонувати безкоштовні збірки OpenJDK 8 і 11, які все ще отримують безпеку та інші оновлення.

Oracle (та інші) настійно рекомендують видаляти застарілі та непідтримувані версії Java через невирішені проблеми безпеки в старих версіях. Oracle радить своїм користувачам негайно перейти на підтримувану версію, наприклад одну з версій LTS (8, 11, 17).

Java JVM і байт-код

Чи не найголовнішою ціллю мови Java є переносимість, а саме доктрина, яка наголошує, що програма, написана для платформи Java, повинна працювати на будь-якій платформі, не привязуючись до її апаратного забезпечення. Ціль досягається за допомогою компіляції коду Java в проміжний код, або так званий байт-код Java, а не напряму в машинний код, як це властиво для інших компільованих мов. Інструкції, що прописані в байт-коді призначені для виконання JVM, яка в свою чергу написана спеціально для апаратного забезпечення платформи, на якій запускається. Користувачі, як правило, використовують Java Runtime Environment(JRE), яку необхідно встановлювати на робочому пристрої для програма Java, для Java-аплетів використовують веб-браузер.

Java має надзвичайно обширну стандартну бібліотеку, завдяки якій можна гнучко використовувати функції класів, як у настільних програмах, так і у веб-додатках.

Використання універсального байт-коду спрощує перенесення. Проте, інтерпретація байт-коду в машинний потребує накладних витрат, через що інтерпретовані програми поступаються швидкістю виконання звучайним. Компілятори Just-in-time (JIT), які компілюють байт-код у машинний код під час виконання, були представлені на ранньому етапі. Сама ж мова Java не є залежною від платформи і адаптована до конкретної платформи, на якій воназапускається, за допомогою JVM, яка інтерпретує байт-код Java на машинну мову платформи.

Продуктивність

Програми, написані на Java, мають репутацію повільніших і вимагають більше пам’яті, ніж програми, написані на C++. Але швидкість виконання програм Java суттєво зменшилась, коли запровадили компіляцію «на льоту» в 1997/1998 роках для програм на Java версії 1.1, додали мовні функції, що підтримують кращий аналіз коду (наприклад, внутрішні класи, клас StringBuilder, необов’язкові параметри тощо), а також оптимізації у віртуальній машині Java, наприклад HotSpot, ставши JVM Sun за замовчуванням у 2000 році. З Java 1.5 продуктивність була покращена за допомогою додавання пакета java.util.concurrent, включаючи реалізації ConcurrentMaps та інших багатоядерних колекцій, а також покращено з версією Java 1.6.

Автоматичне управління пам'яттю

Java має вбудований автоматичний збірник сміття для управління пам'яттю в процесі життєвого цикла об'єкта. Програміст в свою чергу визначає час створення об’єктів, а вже середовище виконання Java несе відповідальність за звільнення пам’яті, після того, як об’єкти більше не використовуються програмою. Як тільки посилання на об’єкт зникає, зайнята ним пам’ять стає пригодною для звільнення автоматичним збирачем сміття. Витоки пам’яті все ще можуть траплятися, якщо код містить посилання на об’єкт, який більше не потрібен, наприклад, коли об’єкти, які більше не використовуються, зберігаються в контейнерах, які все ще необхідні програмі. При спробі викликати методи неіснуючого об’єкта, програма видає виняток нульового вказівника.

Однією з ідей, що лежить в базисі моделі автоматичного управління пам’яттю Java, є можливість програмістів позбутися від необхідності ручного керування пам’яттю. У деяких мовах програмування пам'ять, необхідна для створення об'єктів неявно виділяється в стеку або явно виділяється і звільняється з купи. У випадку з купою відповідальність за управління пам'яттю лягає на плечі програміста. Витік пам’яті відбувається тоді, коли програма не звільняє пам’ять, зайняту під об’єкт. У разі коли програма спробує отримати доступ або звільнити пам’ять, яка вже була вивільнена, результат може бути непередбачуваним, і програма, скоріш всього, стане нестабільною або аварійно завершить роботу.

Частково ця проблема вирішується за допомогою розумних покажчиків, але вони сприяють ускладненню кода. Слід зауважити, що збір сміття не запобігає витоку логічної пам’яті, тієї, на яку програма все ще посилається, але ніколи не використовує.

Збір сміття відбувається в будь-який момент, частіше, це тоді, коли програма неактивна. Гарантією того, що він спрацює є нестача в купі вільної пам’яті для виділення нового об’єкта, що може призвести до миттєвої аварійної зупинки програми. В мові Java неможливе явне керування пам’яттю.

Арифметика вказівників, коли адресами об’єктів можна арифметично маніпулювати, у стилі C/C++ в Java не підтримується. Це дозволяє збірнику сміття переміщувати об’єкти, на які посилаються, і гарантує безпеку типу.

Подібно до C++ та багатьох інших об'єктно-орієнтованих мов, змінні примітивних типів даних Java можуть зхберігатись або безпосередньо в полях (для об'єктів), або в стеку (для методів), але не в купі, як в ситуації з непримітивними типами даних. Це рішення дизайнерів Java було прийняте з міркувань продуктивності.

Java налічує кілька типів збирачів сміття. Починаючи з Java 9, HotSpot використовує Garbage First Garbage Collector (G1GC) за замовчуванням. Проте є також кілька інших збирачів сміття, які використовуються для керування пам’ятю в купі. Більшість програм на Java задовольняє G1GC. У версії Java 8 використовувався збірник сміття, відомий як Parallel Garbage Collector.

Вирішивши проблеми управління пам’яттю, програміст все ще зобов’язаний вміти правильно працювати з обробкою інших видів ресурсів, таких як мережеві з’єднання або підключення до бази даних, дескриптори файлів тощо, особливо за наявності винятків.

Jakarta Servlet — це програмний компонент Java, призначений для розширення можливостей сервера. Хоча сервлети можуть обробляти багато типів запитів, найчастіше вони є реалізацією веб-контейнерів для розміщення веб-додатків на веб-серверах. Веб-сервлети є аналогом Java для інших технологій динамічного веб-контенту, таких як PHP і ASP.NET.

Сервлет Jakarta обробляє або зберігає клас Java в Jakarta EE, який відповідає API сервлетів Jakarta, стандарту для реалізації класів Java, які відповідають на запити. Сервлети в принципі можуть спілкуватися за будь-яким протоколом клієнт-сервер, але найчастіше вони використовуються з HTTP. Таким чином, «сервлет» часто використовується як скорочення «сервлет HTTP». Таким чином, розробник програмного забезпечення може використовувати сервлет для додавання динамічного вмісту на веб-сервер за допомогою платформи Java. Згенерований вміст зазвичай являє собою HTML, але можуть бути й інші дані, такі як XML і частіше JSON. Сервлети можуть підтримувати стан змінних сеансу в багатьох транзакціях сервера за допомогою файлів cookie HTTP або зіставлення URL-адрес.

API Jakarta Servlet певною мірою було замінено двома стандартними технологіями Java для веб-сервісів:

- веб-сервіси Jakarta RESTful Web Services (JAX-RS 2.0), корисні для служб AJAX, JSON і REST,
- Веб-служби Джакарти XML (JAX-WS), корисні для веб-служб SOAP.

Для розгортання та запуску сервлета необхідно використовувати веб-контейнер. Веб-контейнер (також відомий як контейнер сервлетів) по суті є компонентом веб-сервера, який взаємодіє із сервлетами. Веб-контейнер відповідає за керування життєвим циклом сервлетів, зіставлення URL-адреси з певним сервлетом і забезпечення того, що запитувач URL-адрес має правильні права доступу.

API сервлетів, що міститься в ієрархії пакетів Java javax.servlet, визначає очікувані взаємодії веб-контейнера та сервлета.

Сервлет - це об'єкт, який отримує запит і генерує відповідь на основі цього запиту. Базовий пакет сервлетів визначає об’єкти Java для представлення запитів і відповідей сервлетів, а також об’єкти, що відображають параметри конфігурації сервлета та середовище виконання. Пакет javax.servlet.http визначає HTTP-специфічні підкласи загальних елементів сервлетів, включаючи об'єкти керування

сеансом, які відстежують численні запити та відповіді між веб-сервером і клієнтом. Сервлети можуть бути запаковані у файл WAR як веб-додаток.

Сервлети можуть бути автоматично створені з Jakarta Server Pages (JSP) компілятором Jakarta Server Pages. Різниця між сервлетами і JSP полягає в тому, що сервлети зазвичай вбудовують HTML в код Java, тоді як JSP вбудовують код Java в HTML. Хоча пряме використання сервлетів для генерації HTML (як показано в прикладі нижче) стало рідкістю, веб-фреймворк MVC вищого рівня в Jakarta EE (JSF) все ще явно використовує технологію сервлетів для низькорівневої обробки запитів/відповідей через FacesServlet. . Дещо давнішим є використання сервлетів у поєднанні з JSP у шаблоні, який називається «Модель 2», який є різновидом контролера model–view–controller.

Jakarta Server Pages (JSP; раніше JavaServer Pages) — це набір технологій, які допомагають розробникам програмного забезпечення створювати динамічно створювані веб-сторінки на основі HTML, XML, SOAP або інших типів документів. Випущений у 1999 році компанією Sun Microsystems, JSP подібний до PHP і ASP, але використовує мову програмування Java.

Для розгортання та запуску Jakarta Server Pages потрібен сумісний веб-сервер із контейнером сервлетів, наприклад Apache Tomcat або Jetty.

Архітектурно JSP можна розглядати як високорівневу абстракцію сервлетів Java. JSP транслюються в сервлети під час виконання, тому JSP є сервлетом; кожен сервлет JSP кешується і повторно використовується, доки оригінальний JSP не буде змінено.

Серверні сторінки Jakarta можна використовувати окремо або як компонент перегляду серверної моделі-вигляду-контролера, як правило, з JavaBeans як моделлю і сервлетами Java (або структурою, як-от Apache Struts) як контролером. Це тип архітектури Model 2.

JSP дозволяє перемежовувати код Java і певні заздалегідь визначені дії зі статичним вмістом веб-розмітки, таким як HTML. Отримана сторінка компілюється і виконується на сервері для доставки документа. Скомпільовані сторінки, а також будь-які залежні бібліотеки Java містять байт-код Java, а не

машинний код. Як і будь-яка інша програма .jar або Java, код має виконуватися на віртуальній машині Java (JVM), яка взаємодіє з операційною системою хоста сервера, щоб забезпечити абстрактне, нейтральне для платформи середовище.

JSP зазвичай використовуються для доставки документів HTML і XML, але за допомогою OutputStream вони також можуть доставляти інші типи даних.

Веб-контейнер створює неявні об'єкти JSP, такі як запит, відповідь, сесія, програма, конфігурація, сторінка, pageContext, вихід і виняток. JSP Engine створює ці об'єкти на етапі перекладу.

Компілятор

Компілятор сторінок JavaServer — це програма, яка аналізує JSP і перетворює їх у виконувані сервлети Java. Програма такого типу зазвичай вбудовується в сервер додатків і запускається автоматично під час першого доступу до JSP, але сторінки також можуть бути попередньо скомпільовані для кращої продуктивності або скомпільовані як частина процесу збірки для перевірки на наявність помилок.

Деякі контейнери JSP підтримують налаштування того, як часто контейнер перевіряє часові позначки файлу JSP, щоб побачити, чи змінилася сторінка. Зазвичай ця позначка часу встановлюється на короткий інтервал (можливо, секунди) під час розробки програмного забезпечення та довший інтервал (можливо, хвилини або навіть ніколи) для розгорнутого веб-додатка.

2.2 Огляд середовища розробки

В процесі розробки веб-додатку було використано середовище розробки IntelliJ IDEA. Intellij IDEA являє собою інтегроване середовище розробки (IDE), призначене для розробки як настільних програм, так і веб та андроїд додатків. Середовище розроблене компанією JetBrains і доступне у вигляді ліцензованої спільноти Apache 2 та у приватній комерційній версії.

Історія

У січні 2001 року була випущена перша версія IntelliJ IDEA, що стала однією з перших середовищ IDE Java з інтегрованими можливостями рефакторингу коду та розширеної навігації.

У грудні 2014 року компанією Google була оголошена версія Android Studio 1.0, IDE з відкритим кодом для програм Android, засновану на версії IntelliJ IDEA для спільноти з відкритим кодом. Інші середовища розробки, засновані на фреймворку IntelliJ, включають AppCode, CLion, GoLand, PyCharm, PhpStorm, Rider, RubyMine, DataGrip, WebStorm і MPS.

Особливості

Допомога в кодуванні

IDE надає певні функції, як-от завершення коду шляхом аналізу контексту, навігацію по коду, яка дозволяє безпосередньо переходити до класу або оголошення в коді, рефакторинг коду, налагодження коду, лінтинг і параметри для виправлення невідповідностей за допомогою пропозицій.

Вбудовані інструменти та інтеграція

IDE забезпечує інтеграцію з інструментами збірки/пакування, такими як grunt, bower, gradle і SBT. Він підтримує системи контролю версій, такі як Git, Mercurial, Perforce і SVN. До таких баз даних, як Microsoft SQL Server, Oracle, PostgreSQL, SQLite і MySQL, можна отримати доступ безпосередньо з IDE у версії Ultimate через вбудовану версію DataGrip.

Екосистема плагінів

IntelliJ підтримує плагіни, за допомогою яких можна додати додаткові функції до IDE. Плагіни можна завантажити та встановити або з веб-сайту сховища плагінів IntelliJ, або через вбудовану функцію пошуку та встановлення плагінів IDE. Станом на 2019 рік у виданнях Community та Ultimate є понад 3000 плагінів.

2.3 Огляд додаткового інструментарію

Мова розмітки гіпертексту або HTML є стандартною мовою розмітки для документів, призначених для відображення у веб-браузері. Цьому можуть

допомогти такі технології, як каскадні таблиці стилів (CSS) і мови сценаріїв, такі як JavaScript.

Веб-браузер відправляє запит на сервер і у відповідь отримує HTML-документ, який користувач бачить як веб-сторінку. HTML слугує інструментом для опису структури веб-сторінки семантично.

Сторінки HTML будуються за допомогою HTML елементів, ще відомих як теги. Теги записуються в кутових дужках і слугують розмежувачами для HTML-елементів. Є теги, які вносять вміст на сторінку, наприклад, <img /> і <input />. Інші теги, такі як <p>, слугують для оточення тексту документа, а також можуть бути батьківськими для інших тегів. Безпосередньо теги не відображаються на сторінці, а слугують для інтерпретації вмісту сторінки.

HTML дає можливість розробнику вбудовувати програми, написані мовою сценаріїв, наприклад, Javascript, завдяки якій можна змінювати поведінку сайту і маніпулювати його вмістом. Включення ж CSS задає вигляд вмісту сайту.

Розмітка HTML має декілька ключових елементів, включно з тими, що звучться тегами, типи даних на основі символів, посилання на символи та посилання на сутність. Частіше HTML-теги використовуються парами, як-от <h1> і </h1>, проте є й такі, що представляють порожні елементи і є непарними, наприклад <input>. У парних тегах першим є відкриваючий тег, а другим — закриваючий.

Іншим важливим компонентом є оголошення типу документа HTML, яке запускає відтворення стандартного режиму.

Нижче наведено приклад класичного "Hello, World!" програма:

```
<!DOCTYPE html>
<html>
  <голова>
    <title>Це назва</title>
  </head>
  <тіло>
    <div>
```

```
        <p>Привіт, світ!</p>
    </div>
  </body>
</html>
```

Текст між <html> і </html> описує веб-сторінку, а текст між <body> і </body> є видимим вмістом сторінки. Текст розмітки <title>Це заголовок</title> визначає заголовок сторінки браузера, що відображається на вкладках і заголовках вікон браузера, а тег визначає поділ сторінки, який використовується для легкого оформлення. Між <head> і </head> для визначення метаданих веб-сторінки можна використовувати елемент <meta>.

Декларація типу документа <!DOCTYPE html> призначена для HTML5. Якщо оголошення не включено, різні браузери повертатимуться до «режиму химерності» для відтворення.

Елементи

Документи HTML мають вкладену структуру. Початковий тег нерідко включає в себе атрибути елемента всередині тегу. Атрибути вказують додаткову інформацію.

Такі елементи, як розрив рядка
 або
, забороняють вбудовувати в себе вміст, текст або другі теги. В такому випадку потрібен лише один порожній початковий тег і кінцевий тег необов’язковий.

Є багато необов’язкових тегів, наприклад, розповсюджений тег <p> не потребує закриваючого тега в пару. Браузер сам із структури зрозуміє коли необхідно закрити тег, що спрощує роботу програмісту.

В загальному елемент HTML має наступний вигляд: <tag attribute1="value1" attribute2="value2">"content"</tag>. Структурна розмітка вказує на призначення тексту

Наприклад, <h2>Golf</h2> встановлює "Golf" як заголовок другого рівня. Вміст HTML додатково стилізується за допомогою каскадних таблиць стилів - CSS.

Незалежно від призначення вмісту, CSS вказує браузеру як він повинен виглядати. Наприклад, жирний текст вказує на те, що пристрої візуального

виводу мають відображати жирний текст, але дає мало вказівок, які пристрої, які не в змозі це зробити (наприклад, звукові пристрої, які читають текст вголос), повинні робити. У випадку як жирного тексту, так і курсивного тексту, існують інші елементи, які можуть мати еквівалентне візуальне відображення, але більш семантичний характер, наприклад сильний текст і підкреслений текст відповідно. Легше зрозуміти, як слуховий користувальницький агент повинен інтерпретувати останні два елементи. Однак вони не є еквівалентними своїм презентаційним аналогам: було б небажано, щоб програма читання з екрана, наприклад, підкреслювала назву книги, але на екрані таку назву було б виділено курсивом. Більшість елементів розмітки презентації застаріли відповідно до специфікації HTML 4.0 на користь використання CSS для стилізації.

Гіпертекстова розмітка перетворює частини документа на посилання на інші документи

Елемент прив’язки створює гіперпосилання в документі, а його атрибут href встановлює цільову URL-адресу посилання. Наприклад, HTML-розмітка Вікіпедія відображатиме слово "Вікіпедія" як гіперпосилання. Щоб відобразити зображення як гіперпосилання, елемент img вставляється як вміст в елемент a. Як і br, img є порожнім елементом з атрибутами, але без вмісту чи закриваючого тега. <img src="image.gif" alt="descriptive text" width="50" height="50" border="0">.

Атрибути

Більшість атрибутів елемента – це пари ім’я-значення, розділені знаком = і записані в початковий тег елемента після імені елемента. Значення можна брати в одинарні або подвійні лапки, хоча значення, що складаються з певних символів, можна залишити без лапок у HTML (але не в XHTML). Залишати значення атрибутів без лапок вважається небезпечним. На відміну від атрибутів пари ім'я-значення, є деякі атрибути, які впливають на елемент просто своєю присутністю в початковому тегу елемента, як-от атрибут ismap для елемента img.

Існує кілька загальних атрибутів, які можуть з’являтися в багатьох елементах:

- Атрибут id забезпечує унікальний ідентифікатор елемента для всього документа. Це використовується для ідентифікації елемента, щоб таблиці стилів могли змінювати його презентаційні властивості, а сценарії могли змінювати, анімувати або видаляти його вміст або презентацію. Доданий до URL-адреси сторінки, він надає глобальний унікальний ідентифікатор для елемента, як правило, підрозділу сторінки. Наприклад, ідентифікатор «Атрибути» в https://en.wikipedia.org/wiki/HTML#Attributes.
- Атрибут class забезпечує спосіб класифікації подібних елементів. Це можна використовувати для семантичних або презентаційних цілей. Наприклад, документ HTML може семантично використовувати позначення <class="notation">, щоб вказати, що всі елементи з цим значенням класу підпорядковані основному тексту документа. У презентації такі елементи можуть бути зібрані разом і представлені у вигляді виносок на сторінці замість того, щоб з’являтися там, де вони зустрічаються у джерелі HTML. Атрибути класу використовуються семантично в мікроформатах. Можна вказати кілька значень класів; наприклад, <class="notation important"> поміщає елемент як до нотації, так і до важливих класів.
- Автор може використовувати атрибут style для призначення властивостей презентації певному елементу. Вважається кращою практикою використовувати ідентифікатор елемента або атрибути класу для вибору елемента з таблиці стилів, хоча іноді це може бути занадто громіздким для простого, конкретного або спеціального стилю.
- Атрибут title використовується для додавання підтекстового пояснення до елемента. У більшості браузерів цей атрибут відображається як підказка.
- Атрибут lang визначає природну мову вмісту елемента, яка може відрізнятися від мови решти документа. Наприклад, в англомовному документі:
  Ну що ж, c'est la vie, як кажуть у Франції.

Семантичний HTML – це спосіб написання HTML, який підкреслює значення закодованої інформації над її представленням. HTML включав семантичну розмітку з самого початку, але також включав презентаційну розмітку, таку як теги <font>, <i> і . Існують також семантично нейтральні теги span і div. З кінця 1990-х років, коли каскадні таблиці стилів почали працювати в більшості браузерів, веб-авторів заохочували уникати використання презентаційної HTML-розмітки з метою розділення презентації та вмісту.

Важливим типом веб-агента, який сканує та читає веб-сторінки автоматично, не знаючи, що він може знайти, є веб-сканер або пошукова система. Ці програмні агенти залежать від семантичної чіткості веб-сторінок, які вони знаходять, оскільки вони використовують різні методи та алгоритми для читання та індексації мільйонів веб-сторінок на день і надають користувачам Інтернету засоби пошуку, без яких корисність Всесвітньої мережі була б значно зменшена.

Грамотний семантичний HTML впливає на доступність для користувачів веб-документів. Наприклад, якщо програма зчитування з екрана або аудіобраузер правильно визначить структуру документа, то час користувача з вадами зору не буде витрачатись на зчитування беззістовної інформації на сайті, у тому випадку, якщо вона була грамотно позначена.

CSS — це каскадні таблиці стилів, або ж мова, яка використовується для надання змісту документа презентабельного вигляду. CSS є такою ж невід’ємною технологією веб-розробки, як HTML і JavaScript.

CSS призначений для розділення вигляду та вмісту. Наприклад, за допомогою CSS можна дозволити кільком веб-сторінкам спільно використовувати стилі, написані програмістом, створивши один файл .css.

CSS має каскадну схему пріоритетів, щоб визначити, яке правило стилю використовувати для певного елементу, якщо декілька правил відповідають одному HTML елементу.

CSS специфікації підтримуються World Wide Web Consortium (W3C).

Використання CSS підтримує не тільки HTML, але й інші мови розмітки, такі як XHTML, XML, SVG, XUL.

Блок декларацій

Блок оголошення правила в CSS складається зі списку декларацій у дужках. Кожне оголошення складається з властивості, двокрапки (:) і значення. Якщо в блоці є кілька декларацій, для розділення кожної декларації потрібно вставити крапку з комою (;). Додаткова крапка з комою після останнього (або окремого) оголошення може використовуватися.

Властивості вказані в стандарті CSS. Кожна властивість має набір можливих значень. Деякі властивості можуть впливати на будь-який тип елемента, а інші застосовуються лише до окремих груп елементів.

Значеннями можуть бути ключові слова, наприклад «центр» або «успадкувати», або числові значення, наприклад 200 пікселів (200 пікселів), 50vw (50 відсотків ширини вікна перегляду) або 80% (80 відсотків ширини батьківського елемента). Значення кольору можна вказати за допомогою ключових слів (наприклад, "червоний"), шістнадцяткових значень (наприклад, #FF0000, також скорочено як #F00), значень RGB у шкалі від 0 до 255 (наприклад, rgb(255, 0, 0)), значень RGBA які вказують як колір, так і альфа-прозорість (наприклад, rgba(255, 0, 0, 0,8)), або значення HSL або HSLA (наприклад, hsl(000, 100%, 50%), hsla(000, 100%, 50%, 80) %)).

Одиниці довжини

Ненульові числові значення, що представляють лінійні міри, повинні включати одиницю довжини, яка є або алфавітним кодом, або абревіатурою, як у 200px або 50vw; або знак відсотка, як у 80%. Деякі одиниці – см (сантиметр); в (дюйм); мм (міліметр); ПК (pica); і pt (точка) – абсолютні, що означає, що відтворений розмір не залежить від структури сторінки; інші – em (em); ex (ex) і px (піксель) [потрібне уточнення] – є відносними, що означає, що такі фактори, як розмір шрифту батьківського елемента, можуть вплинути на відтворене вимірювання. Ці вісім одиниць були властивістю CSS 1 і зберігалися в усіх наступних версіях. Запропонований модуль 3-го рівня значень та одиниць CSS, якщо його буде прийнято як рекомендацію W3C, надасть сім додаткових одиниць довжини: ch; Q; rem; vh; vmax; vmin; і vw.

До CSS майже всі атрибути презентації HTML-документів містилися в розмітці HTML. Усі кольори шрифту, стилі фону, вирівнювання елементів, межі та розміри повинні бути чітко описані, часто неодноразово, у HTML. CSS дозволяє авторам переміщувати більшу частину цієї інформації в інший файл, таблицю стилів, що призводить до значно простішого HTML.

Наприклад, заголовки (елементи h1), підзаголовки (h2), підзаголовки (h3) тощо визначаються структурно за допомогою HTML. У друку та на екрані вибір шрифту, розміру, кольору та акцентування цих елементів є презентаційним.

До CSS автори документів, які хотіли призначити такі типографічні характеристики, скажімо, всім заголовкам h2, повинні були повторювати презентаційну розмітку HTML для кожного входження цього типу заголовка. Це зробило документи складнішими, більшими, більш схильними до помилок і ускладнювало обслуговування. CSS дозволяє відокремити презентацію від структури. CSS може визначати колір, шрифт, вирівнювання тексту, розмір, межі, інтервали, макет та багато інших типографічних характеристик, і може робити це незалежно для екранного та друкованого вигляду. CSS також визначає невізуальні стилі, такі як швидкість читання та наголос для читачів на слух. Тепер W3C відмовився від використання всієї презентаційної розмітки HTML.

Наприклад, у попередньому CSS HTML елемент заголовка, визначений червоним текстом, буде записаний так:

<h1>Розділ 1.</h1>

Використовуючи CSS, той самий елемент можна закодувати, використовуючи властивості стилю замість атрибутів HTML презентації:

<h1 style="color: red;">Розділ 1.</h1>

Переваги цього можуть бути не відразу зрозумілі, але потужність CSS стає більш очевидною, коли властивості стилю розміщені у внутрішньому елементі стилю або, ще краще, у зовнішньому файлі CSS. Наприклад, припустимо, що документ містить елемент стилю:

<стиль>

```
    h1 {
        колір: червоний;
    }
</style>
```

Усі елементи h1 в документі автоматично стануть червоними, не вимагаючи жодного явного коду. Якщо пізніше автор захотів зробити елементи h1 синіми, це можна зробити, змінивши елемент стилю на:

```
<стиль>
    h1 {
        колір: синій;
    }
</style>
```

замість того, щоб копітко переглядати документ і змінювати колір для кожного окремого елемента h1.

Стилі також можна помістити у зовнішній файл CSS, як описано нижче, і завантажити за допомогою синтаксису, схожого на:

```
<link href="path/to/file.css" rel="stylesheet" type="text/css">
```

Це ще більше відокремлює стиль від HTML-документа і дає можливість змінити стиль кількох документів, просто редагуючи спільний зовнішній файл CSS.

У якості бази даних обрано було MySQL – СУБД з відкритим вихідним кодом. MySQL є реляційною базою даних, а саме організовує дані в одну або кілька таблиць даних, у яких типи даних можуть бути пов’язані один з одним. Такий тип відносин допомагає в структуруванні даних. SQL — це мова, розроблена для створення, зміни та вилучення даних з реляційної бази даних, а також для керування доступом користувачів до бази даних.

MySQL є безкоштовним програмним забезпеченням з відкритим вихідним кодом згідно з умовами Загальної публічної ліцензії GNU, а також доступне на підставі різноманітних власних ліцензій. MySQL належав і спонсорувався

шведською компанією MySQL AB, яку купила Sun Microsystems (тепер Oracle Corporation). У 2010 році, коли Oracle придбала Sun, Widenius розширив проект MySQL з відкритим кодом для створення MariaDB.

MySQL використовується в багатьох веб-додатках, що керують базами даних, серед них: Drupal, Joomla, phpBB, WordPress. Зважаючи на свою популярність, MySQL використовується в якості бази даних в таких відомих веб-сервісах, як Twitter, YouTube, Facebook.

Огляд

MySQL написаний на C і C++. Приклади платформ, на яких працює MySQL: OpenSolaris, OS/2 Warp, QNX, SCO OpenServer, SCO UnixWare, Sanos і Tru64, AIX, BSDi, FreeBSD, HP-UX, ArcaOS, eComStation, IBM, IRIX, Linux, macOS, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, Oracle Solaris, Symbian, SunOS.

Саме серверне програмне забезпечення MySQL і клієнтські бібліотеки використовують дистрибутив з подвійним ліцензуванням. Вони пропонуються під GPL версії 2 або за власною ліцензією. MySQL має офіційний посібник, який допомагає вирішити будь-яку проблему, з якою може зіткнутись програміст в роботі з цією БД.

Незважаючи на те, що MySQL починався як альтернатива більш потужним власним базам даних, він поступово розвивався для підтримки потреб більшого масштабу. Він як і раніше найчастіше використовується в малих і середніх розгортаннях з одним сервером, або як компонент у веб-додатку на основі LAMP, або як окремий сервер бази даних. Значна частина привабливості MySQL походить від її відносної простоти та зручності використання, що забезпечується екосистемою інструментів з відкритим кодом, наприклад phpMyAdmin. У середньому діапазоні MySQL можна масштабувати, розгортаючи його на більш потужному обладнанні, такому як багатопроцесорний сервер з гігабайтами пам’яті.

Однак існують обмеження щодо того, наскільки продуктивність може розширюватися на одному сервері («збільшення масштабу»), тому для більших масштабів для підвищення продуктивності та надійності потрібні багатосерверні розгортання MySQL («масштабування»). Типова конфігурація високого класу

може включати потужну головну базу даних, яка обробляє операції запису даних і реплікується на кілька ведених, які обробляють усі операції читання. Головний сервер постійно передає події binlog на підключені підпорядковані пристрої, тому в разі збою підпорядкованого можна підвищити до нового головного, що мінімізує час простою. Подальшого покращення продуктивності можна досягти шляхом кешування результатів запитів до бази даних у пам’яті за допомогою memcached або розбиття бази даних на менші фрагменти, які називаються сегментами, які можуть бути розподілені між кількома розподіленими серверними кластерами.

РОЗДІЛ 3
ПРОЕКТУВАННЯ І РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1 Аналіз варіантів використання

Діаграма використання - це графічне представлення взаємодії користувача із системою, яка показує взаємозв'язок між користувачем та різними випадками використання, в яких він бере участь. Діаграма випадків використання може ідентифікувати різні типи користувачів системи та різні випадки використання, і часто вона супроводжується також іншими типами діаграм. Варіанти використання представлені кругами або еліпсами.

Не беручи в увагу те, що сам use case може детально вивчити кожну можливість, діаграма прикладів використання допомагає забезпечити огляд системи на більш глобальному, вищому рівні.

Через їх простоту, use case схеми можуть бути хорошим інструментом комунікації для зацікавлених сторін. Креслення намагаються імітувати реальний світ і дають зацікавленій стороні уявлення про те, як буде розроблена система.

Метою діаграми використання є відображення динамічного аспекту системи. Додаткові схеми та документація можуть бути використані для забезпечення повного функціонального та технічного уявлення про систему. Вони забезпечують спрощене та графічне представлення того, що система насправді повинна робити.

Дані схеми складаються з наступних елементів:

- рамки системи (англ. system border) - прямокутник із назвою у верхніх частинах та еліпсами (прецедентами) всередині,
- актор (англ. actor) - стилізований людський персонаж, обозначаючий набір ролей користувача (розуміється в широкому змісті: людина, зовнішня сутність, клас, інша система), взаємодіючого з деякою сутністю (системною, підсистемою, класом). Актори не можуть бути пов'язані між собою з іншим (за вимкнення відносин щодо обробки / дослідження),

- прецедент - еліпс із надписом, що обозначає виконувану систематичну дію (може включати можливі варіанти), що призводить до спостерігаємих акторами результатів. Надпис може бути ім'ям або описом (з точки зору актора) того, "що" робить система (а не "як"). Ім’я прецедента зв’язано з неперервним (атомарним) сценарієм - конкретною послідовністю дій, ілюстріруючою повіддю. Під час сценарію актори обмінюються із систематичними повідомленнями. Сценарій може бути приведений на діаграмі прецедентів у відео UML-коментарі. З одним прецедентом може бути пов'язано кілька різних сценаріїв.

На рисунку 3.1 зображено діаграму варіантів використання, яка описує можливі дії користувача в системі.

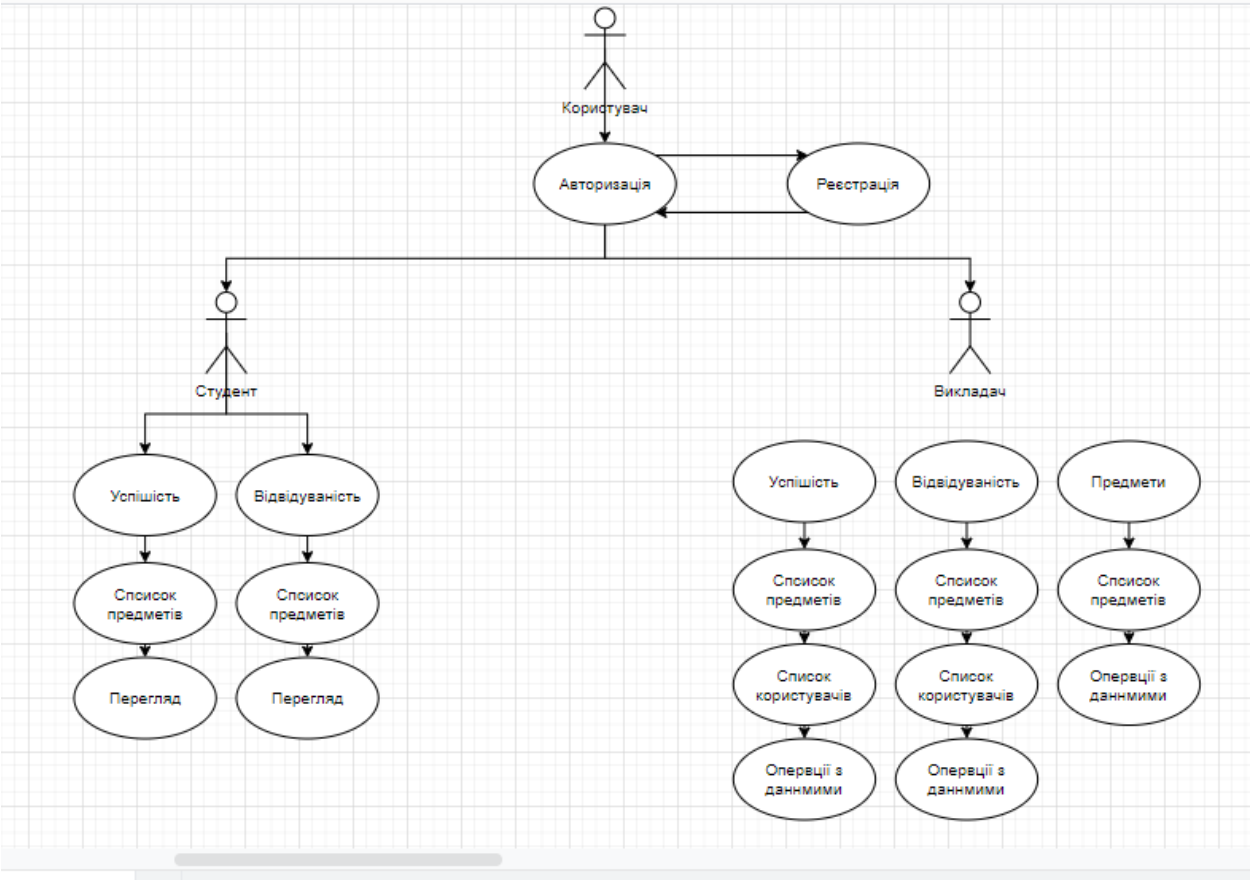


Рис. 3.1 — Діаграма варіантів використання

3.2 Проектування внутрішньої будови

У програмній інженерії діаграма класів являє собою тип статичної структурної діаграми, що описує структуру системи, показуючи класи системи, їх атрибути, операції (або методи) та взаємозв'язки між об'єктами.

Діаграма класів є основним будівельним елементом об’єктно-орієнтованого моделювання. Використовують її для загального концептуального моделювання структури програми та для детального моделювання переведення моделей у програмний код. Діаграми класів також використовуються для моделювання даних.

На схемі класи представлені вікнами, які містять три відділення:

- Назва класу міститься у верхньому відділенні. Текст жирний і відцентрований, перша літера велика.
- Середній відсік- це атрибути класу. Вирівнюються за лівим краєм, з малої літери.
- Нижнє відділення містить функції, які наявні в класі. Вирівняні за лівим краєм, з малої літери.

При проектуванні системи ряд класів ідентифікується та згруповується у схему класів, яка допомагає визначити статичні відносини між ними. При детальному моделюванні класи концептуального проекту часто поділяються на ряд підкласів.

Залежність - це семантичний зв’язок між залежними та незалежними елементами моделі. Він існує між двома елементами, якщо зміни у визначенні одного елемента (сервера або цілі) можуть спричинити зміни для іншого (клієнта або джерела). Ця асоціація є односпрямованою. Залежність відображається у вигляді штрихової лінії з відкритою стрілкою, яка вказує від клієнта до постачальника.

Асоціація представляє родину посилань. Двійкова асоціація (з двома кінцями) зазвичай представляється у вигляді рядка. Асоціація може пов’язувати будь-яку кількість класів. Асоціація з трьома ланками називається потрійною

асоціацією. Асоціацію можна назвати, а кінці асоціації можна прикрасити іменами ролей, показниками власності, кратністю, видимістю та іншими властивостями.

Існує чотири різні типи асоціацій: двонаправлена, односпрямована, агрегаційна (включає агрегацію композиції) та рефлексивна. Двонаправлені та односпрямовані асоціації є найбільш поширеними.

Наприклад, клас польоту асоціюється з класом літака двонаправлено. Асоціація представляє статичне відношення, яке ділиться між об'єктами двох класів.

Агрегація є варіантом взаємозв'язку "має"; агрегація є більш конкретною, ніж асоціація. Це асоціація, яка представляє частково цілі або часткові стосунки. Як показано на зображенні, професор "має" клас для викладання. Як тип асоціації, агрегація може бути названа та мати ті самі прикраси, що і асоціація. Однак агрегація не може включати більше двох класів; це має бути бінарна асоціація. Крім того, навряд чи існує різниця між агрегаціями та асоціаціями під час реалізації, і діаграма може взагалі пропустити відносини агрегування.

Агрегація може відбуватися, коли клас є колекцією або контейнером інших класів, але вміщені класи не мають сильної залежності життєвого циклу від контейнера. Вміст контейнера все ще існує, коли контейнер знищений.

В UML він графічно представлений у вигляді порожнистої форми ромба на вміщуючому класі одним рядком, що зв’язує його із вміщеним класом. Сукупність - це семантично розширений об’єкт, який у багатьох операціях трактується як одиниця, хоча фізично він складається з декількох менших об’єктів.

Приклад: Бібліотека та студенти. Тут студент може існувати без бібліотеки, зв'язок між студентом і бібліотекою є агрегацією.

Це вказує на те, що один із двох пов’язаних класів (підклас) вважається спеціалізованою формою іншого (супер тип), а суперклас - узагальненням підкласу. На практиці це означає, що будь-який екземпляр підтипу є також екземпляром суперкласу.

Графічне представлення UML узагальнення - це форма порожнистого трикутника на кінці суперкласу рядка (або дерева рядків), що зв’язує його з одним або кількома підтипами.

Відносини узагальнення також відомі як спадщина або відносини "є".

Суперклас у відносинах узагальнення також відомий як "батьківський", базовий клас або базовий тип.

Підтип у відносинах спеціалізації також відомий як "дочірній", підклас, похідний клас, похідний тип, клас успадкування або тип успадкування.

Наприклад, "дуб - це тип дерева", "автомобіль - це тип транспортного засобу"

Узагальнення може бути показано лише на діаграмах класів та на діаграмах використання.

При моделюванні UML взаємозв'язок реалізації - це взаємозв'язок між двома елементами моделі, в яких один елемент моделі (клієнт) реалізує поведінку, яку вказує інший елемент моделі (постачальник).

Графічне представлення UML реалізації - це порожниста форма трикутника на кінці інтерфейсу штрихової лінії (або дерева рядків), яка з'єднує її з одним або кількома реалізаторами. Проста головка стрілки використовується на кінці інтерфейсу штрихової лінії, що з'єднує її з користувачами. У діаграмах компонентів використовується графічна умова «м'яч і сокет» (реалізатори виставляють кульку або льодяник, тоді як користувачі показують сокет). Реалізації можна показати лише на діаграмах класів або компонентів. Реалізація - це взаємозв'язок між класами, інтерфейсами, компонентами та пакетами, що з'єднує елемент клієнта з елементом постачальника. Зв'язок реалізації між класами / компонентами та інтерфейсами показує, що клас / компонент реалізує операції, пропоновані інтерфейсом.

Залежність - це слабша форма зв'язку, яка вказує на те, що один клас залежить від іншого, оскільки він використовує його в певний момент часу. Один клас залежить від іншого, якщо незалежний клас є змінною параметра або локальною змінною методу залежного класу. Це відрізняється від асоціації, де атрибут залежного класу є екземпляром незалежного класу. Іноді відносини між двома

класами дуже слабкі. Вони взагалі не реалізовані зі змінними-членами. Швидше вони можуть бути реалізовані як аргументи функції-члена.

Ці відносини зазвичай описуються як «А має В».

Представлення UML асоціації - це лінія, що з'єднує два пов'язані класи. На кожному кінці рядка є додаткові позначення. Наприклад, ми можемо вказати, використовуючи наконечник стрілки, що загострений кінець видно з хвоста стрілки. Ми можемо вказати власність шляхом розміщення кульки, ролі, яку відіграють елементи цього кінця, вказавши ім'я ролі та множинність екземплярів цієї сутності (діапазон кількості об'єктів, які беруть участь в асоціації з точки зору іншого кінця).

Класи сутності моделюють довгоживучу інформацію, яку обробляє система, а іноді і поведінку, пов'язану з цією інформацією. Їх не слід ідентифікувати як таблиці баз даних чи інших сховищ даних.

Вони намальовані як кола з короткою лінією, прикріпленою до нижньої частини кола. Як варіант, їх можна намалювати як звичайні класи із позначенням стереотипу «сутність» над назвою класу.

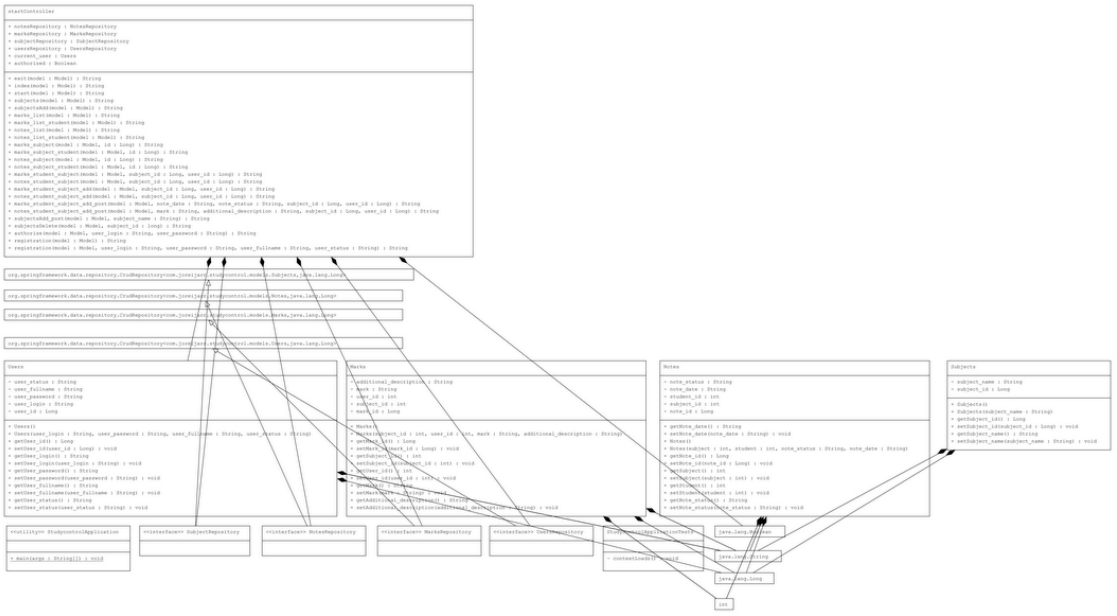


Рис. 3.2 — Діаграма класів

3.2 Розробка графічного інтерфейсу системи

Графічний інтерфейс системи складається з таких основних сторінок:

- сторінка реєстрації (рис. 3.3);
- сторінка авторизації (рис. 3.4);
- головна сторінка викладача (рис. 3.5);
- сторінка предметів (рис. 3.6);
- сторінка успішності для викладача (рис. 3.7);
- сторінка відвідуваності для викладача (рис. 3.8);
- сторінка редагування даних для викладача (рис. 3.9);
- головна сторінка для студента (рис. 3.10);
- сторінка успішності для студента (рис. 3.11);
- сторінка відвідуваності для студента (рис. 3.12).

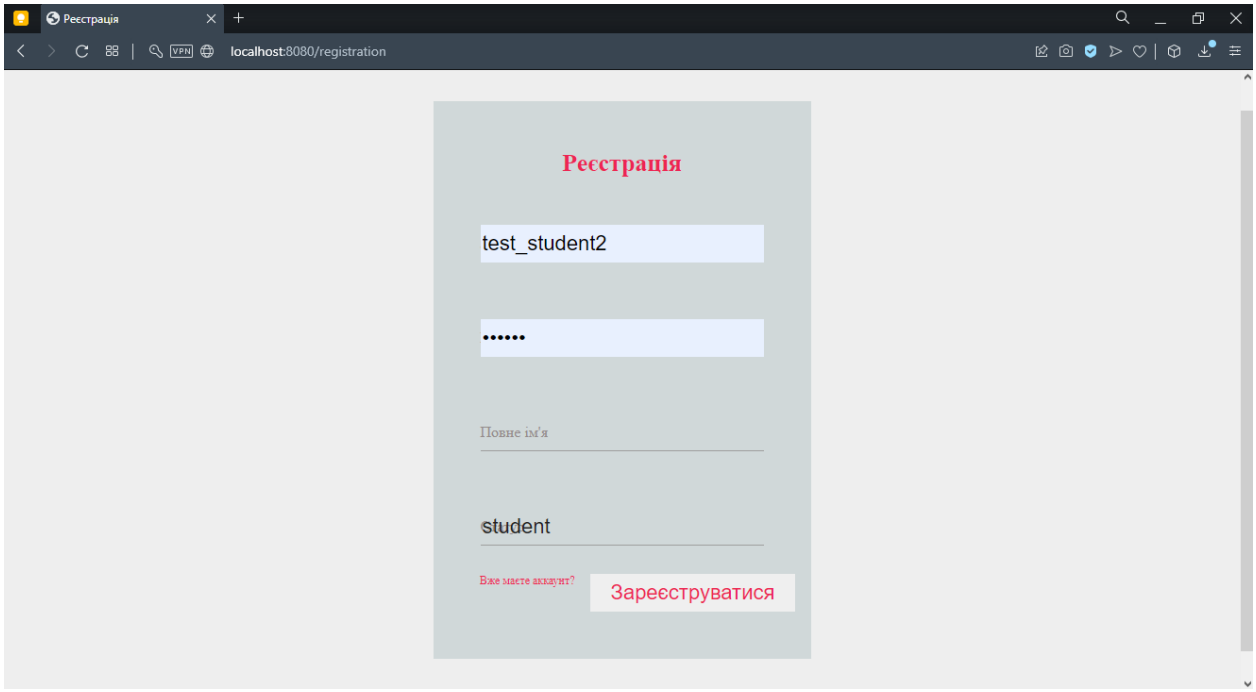


Рисунок 3.3 — Сторінка реєстрації

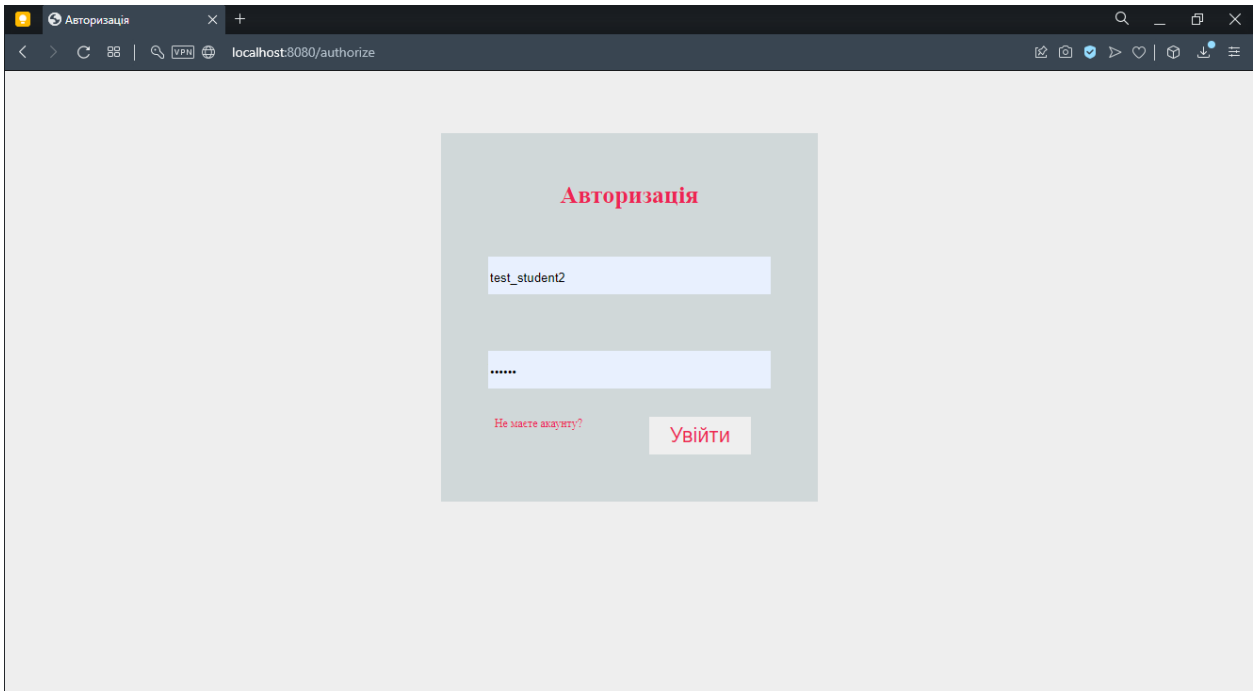


Рисунок 3.4 — Сторінка авторизації

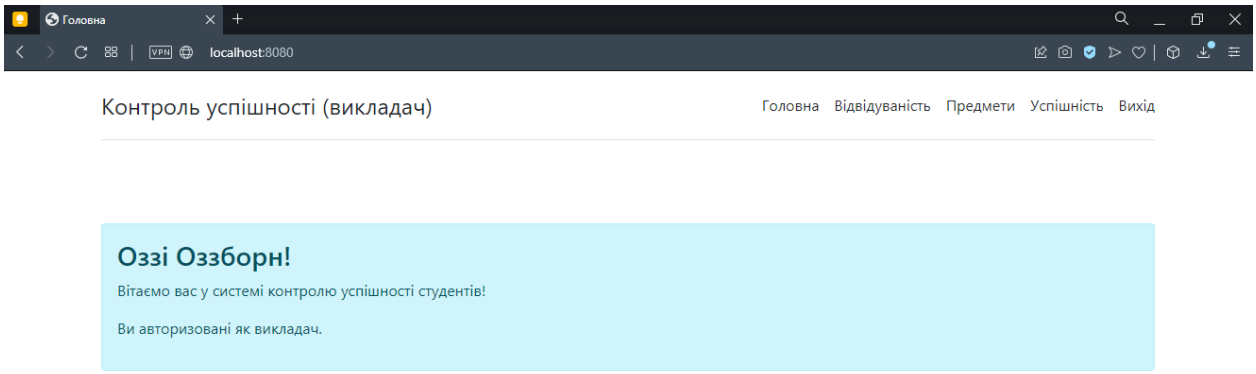


Рисунок 3.5 — Головна сторінка викладача

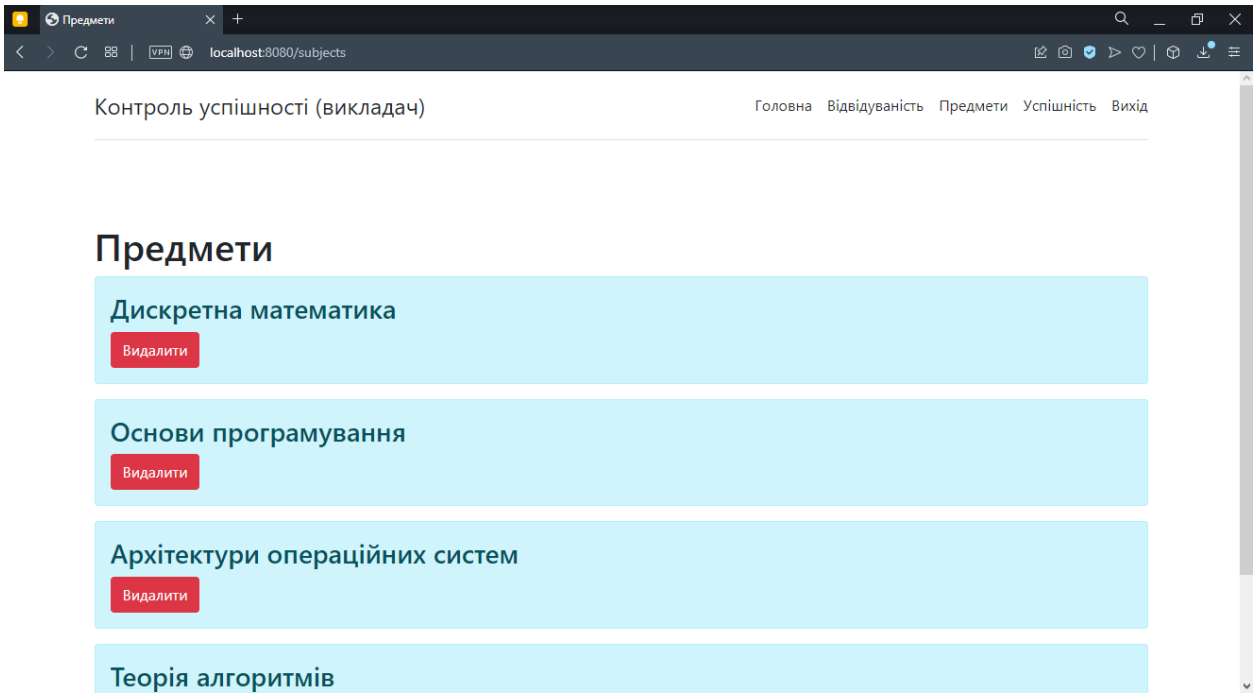


Рисунок 3.6 — Сторінка предметів

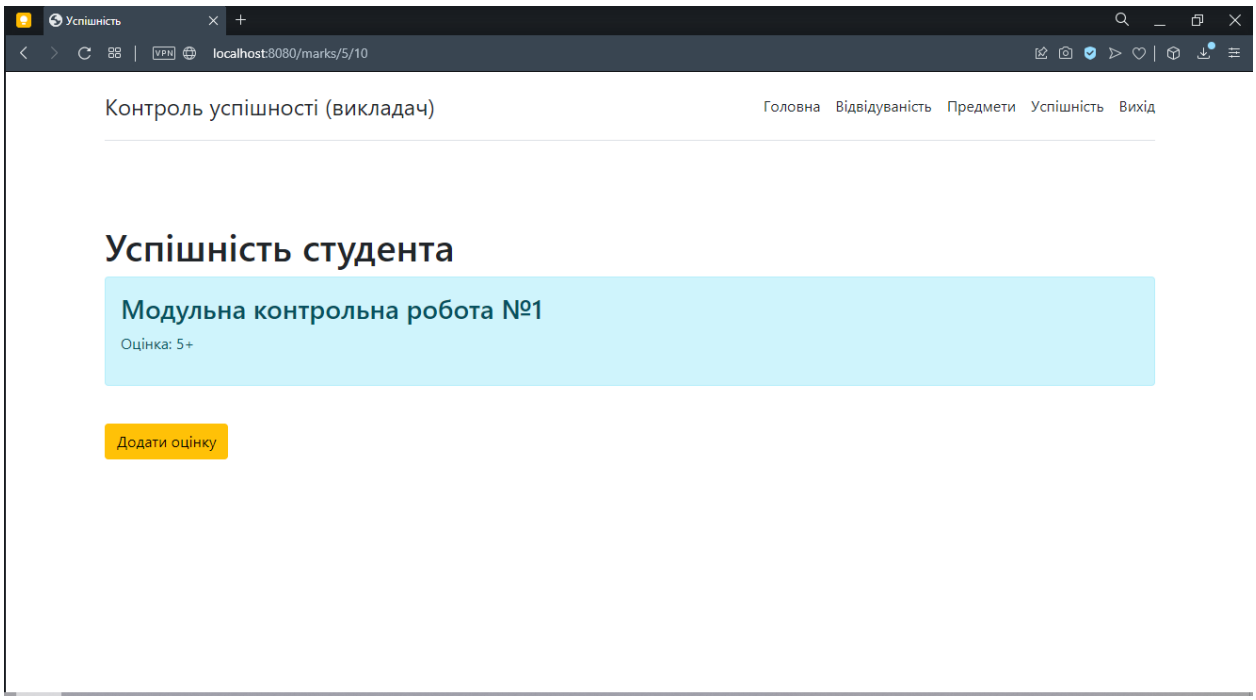


Рисунок 3.7 — Сторінка успішності студента для викладача

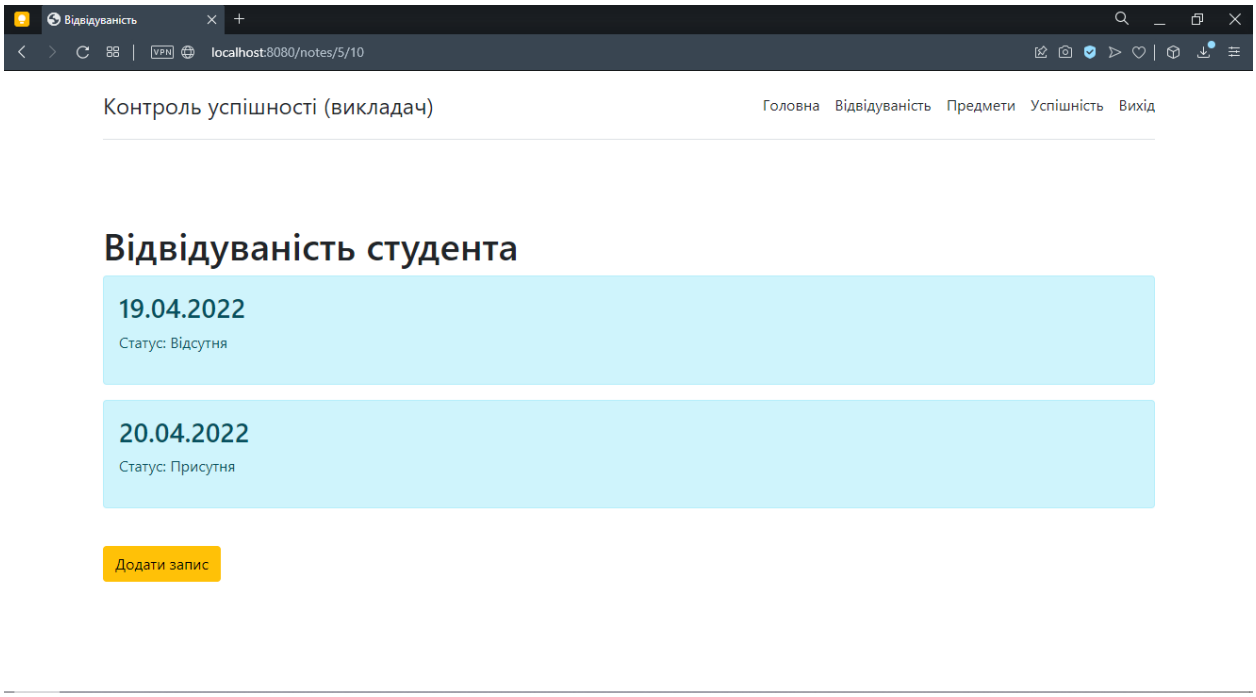


Рисунок 3.8 — Сторінка відвідуваності для викладача

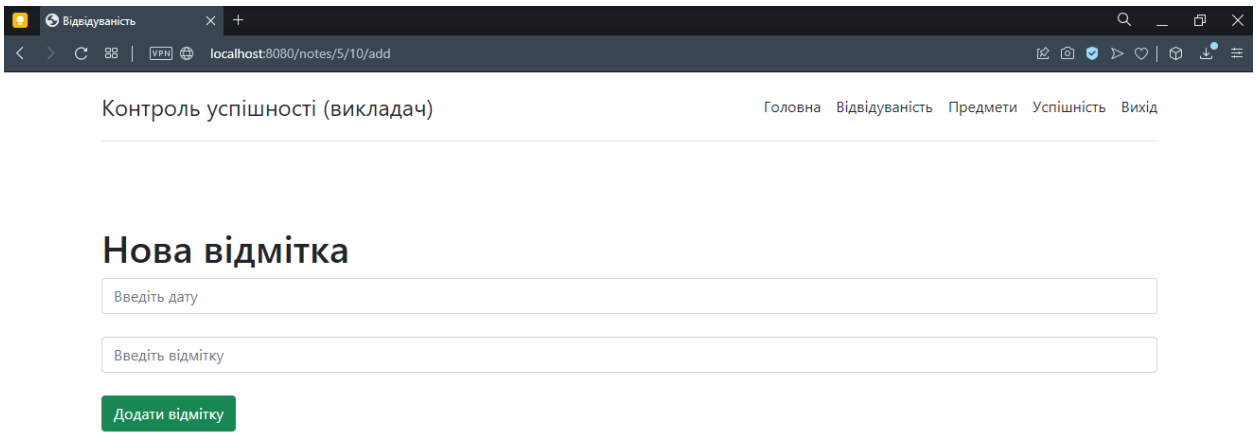


Рисунок 3.9 — Сторінка редагування даних для викладача

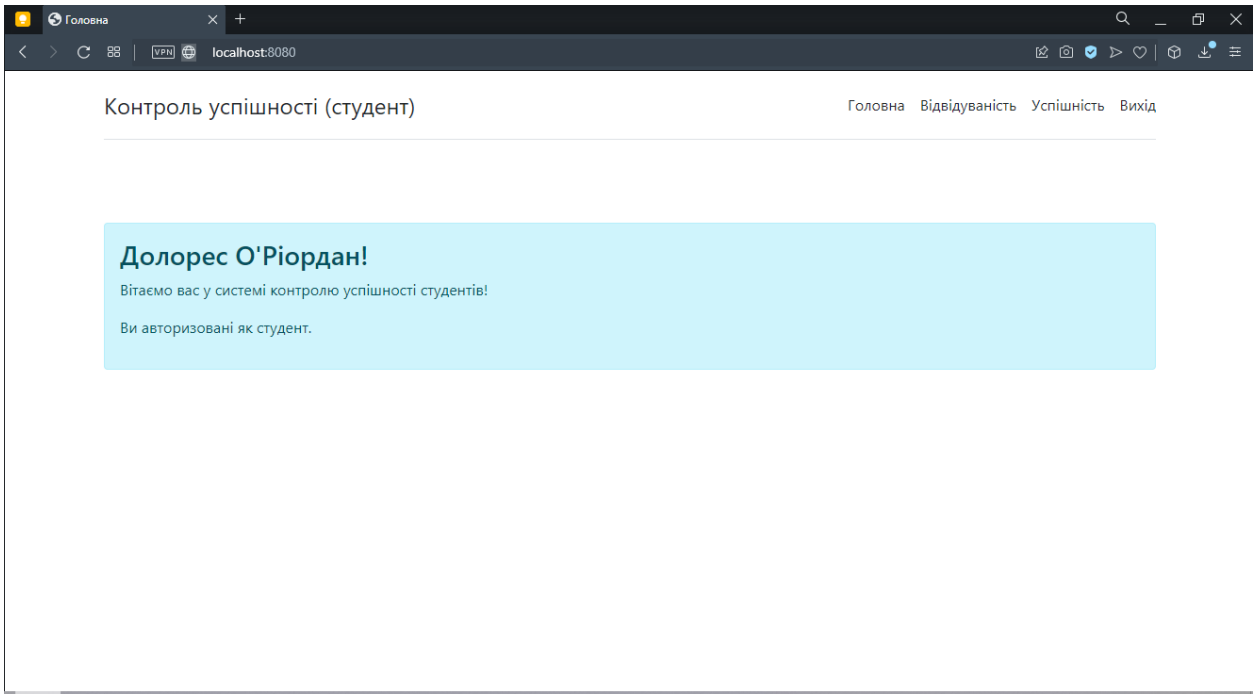


Рисунок 3.10 — Головна сторінка студента

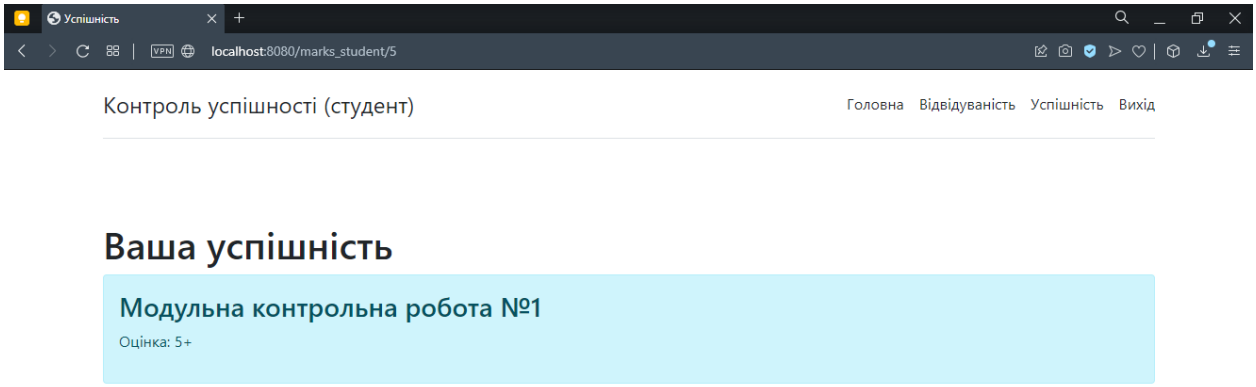


Рисунок 3.11 — Сторінка успішності студента

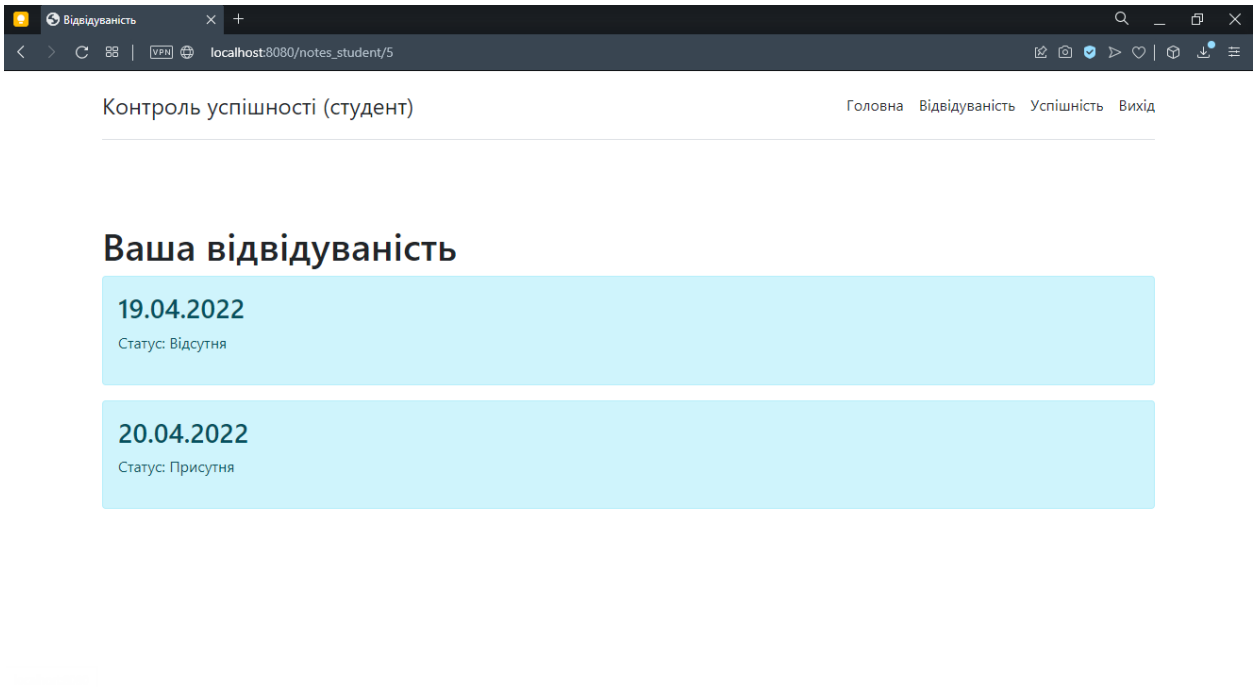


Рисунок 3.12 — Сторінка відвідуваності студента

3.4 Тестування

Тестування програмного забезпечення – це акт перевірки артефактів і поведінки програмного забезпечення, що тестується, шляхом перевірки та верифікації. Тестування програмного забезпечення також може забезпечити об’єктивне, незалежне уявлення про програмне забезпечення, щоб дозволити бізнесу оцінити та зрозуміти ризики впровадження програмного забезпечення. Методи тестування включають, але не обов’язково обмежуються:

- аналіз вимог до продукту щодо повноти та правильності в різних контекстах, таких як галузева перспектива, бізнес-перспектива, доцільність та життєздатність впровадження, зручність використання, продуктивність, безпека, міркування інфраструктури тощо.
- перегляд архітектури продукту та загального дизайну продукту
- робота з розробниками продуктів над удосконаленням техніки кодування, шаблонів проектування, тестів, які можна написати як частину коду на основі різних методів, таких як граничні умови тощо.
- виконання програми або програми з метою перевірки поведінки

- перевірка інфраструктури розгортання та пов’язаних скриптів та автоматизації
- брати участь у виробничій діяльності, використовуючи методи моніторингу та спостереження

Тестування програмного забезпечення може надати користувачам або спонсорам об’єктивну, незалежну інформацію про якість програмного забезпечення та ризик його збою.

Несправності та збої

Помилки програмного забезпечення виникають через наступний процес: Програміст робить помилку (помилку), що призводить до помилки (дефекту, помилки) у вихідному коді програмного забезпечення. Якщо ця помилка виконана, у певних ситуаціях система дасть неправильні результати, що призведе до збою.

Не всі несправності обов'язково призведуть до збоїв. Наприклад, помилки в мертвому коді ніколи не призведуть до збоїв. Несправність, яка не виявила збоїв, може призвести до збою при зміні середовища. Приклади цих змін у середовищі включають програмне забезпечення, яке запускається на новій апаратній платформі комп’ютера, зміни вихідних даних або взаємодію з іншим програмним забезпеченням. Одна несправність може призвести до широкого спектру симптомів відмови.

Не всі помилки програмного забезпечення викликані помилками кодування. Одним із поширених джерел дорогих дефектів є прогалини у вимогах, тобто нерозпізнані вимоги, які призводять до помилок, пропущених розробником програми. Прогалини вимог часто можуть бути нефункціональними вимогами, такими як тестованість, масштабованість, ремонтопридатність, продуктивність та інші вимоги. безпеки.

Статичне, динамічне та пасивне тестування

У тестуванні програмного забезпечення існує багато підходів. Огляди, покрокові інструкції або перевірки називаються статичним тестуванням, тоді як

виконання запрограмованого коду з заданим набором тестових випадків називається динамічним тестуванням.

Статичне тестування часто є неявним, як-от коректура, а також коли інструменти програмування/текстові редактори перевіряють структуру вихідного коду, а компілятори (попередні компілятори) перевіряють синтаксис і потік даних як статичний аналіз програми. Динамічне тестування відбувається під час запуску самої програми. Динамічне тестування може розпочатися до того, як програма буде на 100% завершена, щоб перевірити окремі розділи коду та застосувати до дискретних функцій або модулів. Типовими методами для них є або використання заглушок/драйверів, або виконання з середовища налагодження.

Статичне тестування передбачає перевірку, тоді як динамічне також передбачає перевірку.

Пасивне тестування означає перевірку поведінки системи без будь-якої взаємодії з програмним продуктом. На відміну від активного тестування, тестувальники не надають жодних тестових даних, а переглядають системні журнали та трасування. Вони шукають моделі та специфічну поведінку, щоб приймати якісь рішення. Це пов’язано з перевіркою часу виконання в автономному режимі та аналізом журналу.

Дослідницький підхід

Дослідницьке тестування — це підхід до тестування програмного забезпечення, який коротко описується як одночасне навчання, розробка тесту та виконання тесту. Джем Канер, який ввів цей термін у 1984 році,:2 визначає дослідницьке тестування як «стиль тестування програмного забезпечення, який підкреслює особисту свободу та відповідальність окремого тестувальника за постійну оптимізацію якості своєї роботи, розглядаючи тест- пов’язане навчання, дизайн тесту, виконання тесту та інтерпретація результатів тесту як взаємодопоміжні дії, які виконуються паралельно протягом усього проекту».

«Коробковий» підхід

Методи тестування програмного забезпечення традиційно поділяються на тестування білого та чорного ящиків. Ці два підходи використовуються для опису

точки зору, яку дотримується тестувальник під час розробки тестових випадків. До методології тестування програмного забезпечення також можна застосувати гібридний підхід, який називається тестуванням сірого ящика. Оскільки концепція тестування сірого ящика, яка розробляє тести на основі конкретних елементів дизайну, стає все більш популярною, це «довільне розходження» між тестуванням чорного та білого ящиків дещо зникло.

Таблиця 4.2 — Тестування додатку

№	Тест-кейс	Очікуваний результат	Отриманий результат
1	Невірні данні при авторизації	При введенні невірних даних система повідомляє користувача про те, що такого користувача не існує, або дані введені невірно	При введенні невірних даних система повідомляє користувача про те, що такого користувача не існує, або дані введені невірно
2	Пусті поля при авторизації	При спробі авторизації з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.	При спробі авторизації з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.
3	Неспівпадаючі паролі при реєстрації	При введенні неспівпадаючих паролів система повідомляє користувача про те,	При введенні неспівпадаючих паролів система повідомляє користувача про те,

		що паролі не співпадають.	що паролі не співпадають.
4	Пусті поля при реєстрації	При спробі реєстрації з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.	При спробі реєстрації з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.
5	Створення відмітки з пустими полями	При спробі створення відмітки з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.	При спробі створення відмітки з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.
6	Спроба перейти в панель адміністратора без наявності прав адміністратора	При спробі користувача без прав адміністратора перейти в панель адміністрування система повідомляє користувача, шо він не є адміністратором і не може перейти на дану сторінку.	При спробі користувача без прав адміністратора перейти в панель адміністрування система повідомляє користувача, шо він не є адміністратором і не може перейти на дану сторінку.

Результати тестування показують, що система повністю функціональна і готова до використання в реальних умовах.

ВИСНОВКИ

Мета роботи полягала в розробленні системи контролю успішності студентів.

Для досягнення поставленої мети було виконано наступні завдання:

- Провести огляд поняття веб-додатку
- Провести аналіз методів та засобів розробки веб-сайтів
- Провести огляд поняття контролю навчання
- Провести огляд існуючих рішень
- Обрати мову програмування
- Обрати додаткові інструменти
- Обрати середовище розробки
- Провести аналіз варіантів використання
- Провести проектування внутрішньої будови
- Розробити графічний інтерфейс користувача
- Провести тестування системи

Завдяки чіткому виконанню завдань, поставлених на початку роботи, в результаті отримано повноцінну систему контролю успішності студентів, яка готова до використання в реальних умовах.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Davidson, James Duncan; Coward, Danny. Java Servlet Specification Version: 2.2 Final Release. Sun Microsystems./- 1999 – с. 43–46.
2. Jay Hoffmann. What Does AJAX Even Stand For?/- 2011 – с. 11-17.
3. Petersen, Jeremy. Benefits of using the n-tiered approach for web applications./- 2008.
4. What is Web Development? - Definition from Techopedia [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://www.techopedia.com/
5. Tim Berners-Lee [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://www.w3.org/People/Berners-Lee/
6. Cailliau, Robert. A Little History of the World Wide Web [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://www.w3.org/History.html
7. Internet, Web, and Other Post-Watergate Concerns. University of Chicago [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://www.chicagomanualofstyle.org/search.epl?q=watergate&site=bblive&search_edition=16
8. Perrin, Andrew; Anderson, Monica. Social media usage in the U.S. in 2019 | Pew Research Center [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://www.pewresearch.org/fact-tank/2019/04/10/share-of-u-s-adults-using-social-media-including-facebook-is-mostly-unchanged-since-2018
9. A total number of Websites. Internet live stats [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: internetlivestats.com.
10. Internet 2009 in numbers [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://www.pingdom.com/blog/internet-2009-in-numbers
11. Deep Dive Into the New Java JIT Compiler - Graal | Baeldung [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://www.baeldung.com/graal-java-jit-compiler
12. Jelovic, Dejan. Why Java will always be slower than C++./- 2008.

13. HTML5. W3C [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://dev.w3.org/html5/spec-LC
14. HTML Elements [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://www.w3schools.com/html/default.asp
15. Tim Berners-Lee, James Hendler and Ora Lassila. The Semantic Web. Scientific American./- 2001 – с. 51 – 59.
16. Oracle MySQL University: MySQL Galera Multi-Master Replication. Oracle Corporation [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://blogs.oracle.com/mysql/post/mysql-university-mysql-galera-multi-master-replication
17. MySQL 8.0 Reference Manual [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://dev.mysql.com/doc/refman/8.0/en
18. Intellij IDEA Overview [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://www.jetbrains.com/help/idea/discover-intellij-idea.html

Додаток А

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра Інженерії програмного забезпечення

РОЗРОБКА ВЕБ-СЕРВІСУ КОНТРОЛЮ НАВЧАЛЬНОГО ПРОЦЕСУ СТУДЕНТІВ МОВОЮ JAVA

Виконавець: студент 4 курсу, групи ПД–41
Кухаренко Юрій Дмитрович

Керівник роботи: Трінтіна Наталія Альбертівна, кандидат технічних наук, доцент

Київ 2022

Мета, об’єкт та предмет роботи

- ***Мета роботи*** – розробка системи контролю навчального процесу студентів.
- ***Об'єкт дослідження*** – інформаційні системи та їх використання у різного роду структурах.
- ***Предмет дослідження*** – система контролю навчального процесу студентів.

Аналіз аналогів

Переваги

- Різноманітність опцій при створюванні опитувань.
- Зручне ведення статистики, збір даних у таблиці.

Недоліки

- Вузька спеціалізація.
- Не задовольняє всіх потреб користувача, має невеликий функціонал.
- Легко піддається «нечесним» діям користувачів.

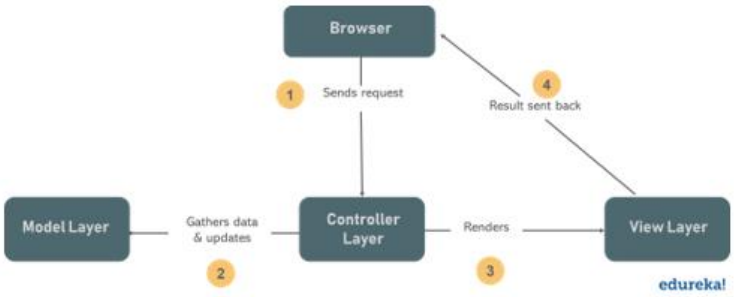
Технічне завдання

- реєстрація та авторизації користувачів та надання різного функціоналу в залежності від ролі користувача в системі;
- внесення в систему та редагування даних про відвідуваність кожного студента;
- внесення в систему та редагування даних про успішність кожного студента з обраного предмету;
- можливість моніторингу успішності студента з кожного окремого предмету;
- можливість моніторингу відвідуваності занять кожного окремого студента відповідно до дат;
- зрозумілий, сучасний та інтуїтивно зрозумілий інтерфейс;

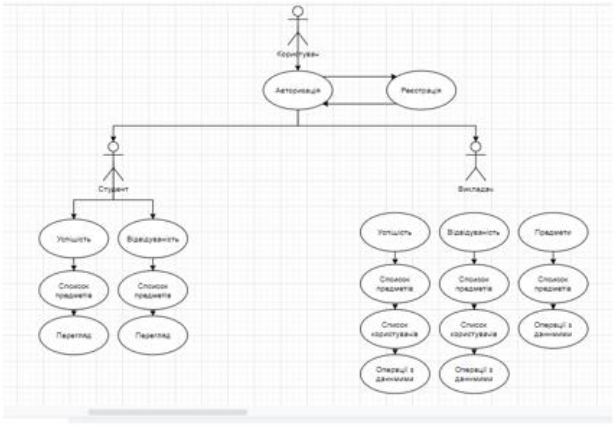
Спеціалізовані технології

- Візуальне середовище програмування(IDE) - Intellij IDEA.
- Мова програмування – Java.
- Framework Spring Boot.
- База даних – MySQL.
- Відображення веб-сторінок – HTML, CSS, шаблонізатор Thymeleaf
- Фреймворк для збірки проекту – Maven.

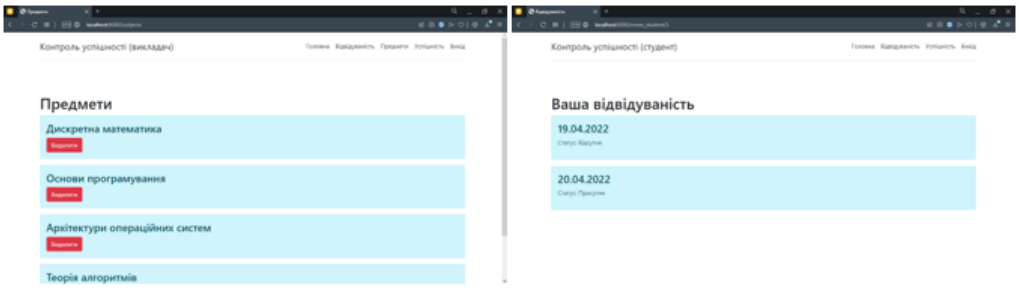
Архітектура Model-View-Controller



Use case діаграма



Практичне застосування



Наукова новизна та практична значимість

Наукова новизна дослідження полягає в розширенні поняття контролю навчального процесу.

Практична значимість дослідження полягає в створенні безкоштовної багатофункціональної системи для контролю навчального процесу. Система розроблюється у вигляді веб-сервісу, без необхідності додаткових завантажень.

Висновки

Створено безкоштовну та ефективну систему контролю навчального процесу студентів, що підтверджує досягнення мети дипломного проекту.

Проект має наступні переваги:

1. Доступ з браузеру без додаткових завантажувань.
2. Функції для контролю відвідуваності та успішності студентів.
3. Оптимізований код, що робить сервіс стійким до навантажень.
4. Сучасний дизайн і простота у використанні.
5. Продукт є повністю безкоштовним.

Даний проект пройшов апробацію на **«XIV НАУКОВО-ТЕХНІЧНІЙ КОНФЕРЕНЦІЇ «СУЧАСНІ ІНФОКОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ»**

Дякую за увагу!