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 для перевірки та оцінювання знань мовою Java»

Виконав: студент 4 курсу, групи ПД–41

Спеціальності

121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

_____Митяй О.В._____
(прізвище та ініціали)

Керівник _Негоденко О.В._____
(прізвище та ініціали)

Рецензент_____
(прізвище та ініціали)

Нормоконтроль _____
(прізвище та ініціали)

Київ – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення_____

Ступінь вищої освіти - «Бакалавр»

Напрям підготовки -121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного
забезпечення
О.В. Негоденко
«____» _____2022 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Митяй Олег Володимирович
(прізвище, ім‘я, по батькові)

1.Тема роботи: «Розробка веб-сервісу для перевірки та оцінювання знань мовою Java»

Керівник роботи Негоденко О.В. к.т.н., доцент ,
(прізвище, ім‘я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «16» лютого 2022 року №22.

2. Строк подання студентом роботи 03.06.2022

3. Вхідні дані до роботи:

Науково-технічна література з питань, пов‘язаних з викладацькими тестами

Науково-технічна література з розробки системи тестування

Перелік аналогів

Науково-технічна література з розробки програмного забезпечення

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

4.1. Аналіз предметного середовища. Існуючі застосунки на ринку

4.2. Дослідження методів тестування

4.3 Проектування програмного забезпечення

4.4 Розробка програмного забезпечення та тестування

5. Перелік графічного матеріалу

5.1. Презентація в Power Point Слайд «Мета, Об’єкт та предмет дослідження»

5.2. Презентація в Power Point Слайд «Аналоги»

5.3. Презентація в Power Point Слайд «Технічне завдання»

5.4. Презентація в Power Point Слайд «Програмні засоби реалізації»

5.5. Презентація в Power Point Слайд «Архітектура програми»

5.6. Презентація в Power Point Слайд «Проектування та реалізація БД»

5.7. Презентація в Power Point Слайд «Апробація результатів дослідження»

5.8. Презентація в Power Point Слайд «Висновки»

5.9. Скріншоти робочої програми

6. Дата видачі завдання 11.04.2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	11.04-14.04	
2	Ознайомлення з викладацькими тестами	15.04-17.04	
3	Написання першого розділу	18.04-21.04	
4	Написання другого розділу	22.04-25.05	
5	Написання третього розділу	26.05-05.05	
6	Розробка власного продукту	05.05-07.05	
8	Вступ, висновки, реферат	07.05-10.05	
9	Розробка демонстраційних матеріалів	11.05-15.05	
10	Попередній захист роботи	16.05-01.06	
11	Подання роботи в деканат	03.06	

Студент _____ Митяй О.В.
(підпис) (прізвище та ініціали)
Керівник роботи _____ Негоденко О.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи 73с., 13 рис., 35 джерел.

Ключові слова: Web-додаток, JetBrains IntelliJ IDEA, Java, Spring, Hibernate, MySQL, HTML, CSS, Bootstrap, Tomcat.

Об’єкт дослідження – процес автоматизації створення та проведення тестування.

Предмет дослідження – технології для автоматизації та проведення тестування.

Мета дослідження – підвищення ефективності перевірки та оцінювання знань за допомогою веб-додатку.

Для реалізації поставленої мети потрібно вирішити наступні завдання:

1. Проаналізувати технічні засоби, які використовуються для розробки та обрати необхідні для створення веб-додатку;
2. Розробити вимоги до застосунку на основі аналізу переваг та недоліків аналогів;
3. Спроектувати та розробити веб-додаток на основі аналізу потреб користувачів;
4. Провести тестування веб-додатку.

Практичне значення отриманих результатів: Даний веб-додаток може бути використаний в університетах, школах, для власного використання та для бізнесу, де потрібна автоматизація в проведені перевірки та оцінювання знань.

Застосунок розроблений з використанням інструментів JetBrains IntelliJ IDEA, MySQL Workbench (графічний інтерфейс для MySQL), Tomcat та мовою програмування Java.

Галузь використання – перевірка та оцінювання знань в університетах, школах, підприємствах, для особистого використання тощо.

У роботі було репрезентовано квестію дієвості використання новітніх WEB-технологій при реалізації WEB-інтерфейсу споживача, та окрім цього проблему з тестуванням учнів викладачем. Висунуте рішення дає змогу вдало інтегрувати стек сьогоденних WEB-технологій та вжити його при розробці системи для ймовірного тестування учнів. Опрацьований WEB-додаток дає змогу в режимі реального часу пройти тестування викладача.

Було здійснено аналіз проблем сучасного WEB-розробки та розбір шляхів їх вирішення, аналіз ринку існуючих автоматизованих систем управління, було розкрито їх недоліки та переваги. Пропонується вивчити сучасні WEB-технології та розробити систему керуючись їх основами, яка полегшує процес взаємодії викладачів з учнями.

Зміст

ВСТУП

Обґрунтування вибору теми та її актуальність: Актуальність проведення тестування полягає в необхідності регулярних перевірок знань та навичок учнів, а отже в необхідності оцінювання таких знань. Проведення тестування являється невід'ємною частиною навчання та поглиблення знань, і як наслідок це допоможе користувачам краще розібратись у вивченому матеріалі, знайти свої слабкі сторони та пропрацювати їх. Тому контроль знань користувачів є важливою складовою частиною навчального процесу, а головна суть – дізнатись, наскільки вдало відбувається процес навчання та засвоєння нового матеріалу.

Існуючі платформи та програми, які використовуються в університетах, школах тощо мають як переваги, серед яких досить різноманітний функціонал та інформативність, так і недоліки: акцент здебільшого на окремі сфери зайнятості та собівартість.

Предмет дослідження – технології для розробки веб-додатку для перевірки та оцінювання знань.

Метою дослідження - підвищення ефективності перевірки та оцінювання знань за допомогою веб-додатку.

Для реалізації поставленої мети потрібно вирішити наступні завдання:

1. Провести аналіз технічних засобів, що використовуються для розробки веб-додатку для перевірки та оцінювання знань;
2. Розробити вимоги до застосунку на основі аналізу переваг та недоліків існуючих рішень;
3. Спроектувати та розробити веб-додаток на основі аналізу потреб користувачів;
4. Провести тестування веб-додатку.

Практичне значення отриманих результатів: Даний веб-додаток може бути використаний в університетах, школах, для власного використання та для бізнесу, де потрібна автоматизація в проведені перевірки та оцінювання знань.

Для розробки використовували JetBrains IntelliJ IDEA, MySQL Workbench (графічний інтерфейс для MySQL) та Tomcat.

Інтерфейс користувача реалізований за допомогою HTML, CSS, Bootstrap і Thymeleaf. Для серверної частини було використано технологію Spring Framework, яка написана мовою Java.

Були представлені діаграми для повноцінного та якісного розуміння усіх наявних аспектів додатку. В межах випускної кваліфікаційної роботи було розроблено програмний комплекс, який включає в собі WEB-інтерфейс користувача. Додаток був протестований за допомогою методики manual testing.

Додаток було визначено актуальним для використання викладачами різних вікових категорій, а також всебічних спеціальностей в особистих та загальних цілях, для автоматизації проходження численних (різноманітних) тестів у навчальних закладах та поза їх межами, для автоматизації збору та аналізу результатів.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Поняття веб-додатку

Веб-додаток — це прикладне програмне забезпечення, розроблене з архітектури «клієнт-сервер» яке працює на сторонньому веб-сервері, на відміну від комп’ютерних програм, які запускаються локально в операційній системі (ОС) пристрою. Користувач за допомогою браузера відправляє HTTP-запит за певною URL-адресою, в свою чергу сервер відправляє набір даних за цим запитом.

Найчастіше веб додатки не є самостійною системою, вони запускаються (виконуються) під управлінням зовнішнього середовища. В якості таких середовищ може виступати контейнер додатків.

Відмінність між веб-додатком Загальна відмінність між динамічною веб-сторінкою будь-якого виду та «веб-додатком» незрозуміла. Веб-сайти, які, швидше за все, називаються "веб-додатками", - це ті, які мають схожі функціональні можливості з програмним забезпеченням для настільних комп'ютерів або мобільним додатком. HTML5 представив явну мовну підтримку для створення програм, які завантажуються як веб-сторінки, але можуть зберігати дані локально та продовжувати працювати в автономному режимі.

Односторінкові програми більше схожі на програми, оскільки вони відкидають більш типову веб-парадигму переміщення між різними сторінками з різними URL-адресами. Це пов’язано з тим, що окремі компоненти можна замінити або оновити без необхідності оновлювати всю веб-сторінку. Односторінкові фреймворки можуть бути використані для прискорення розробки такого веб-додатка для мобільної платформи, оскільки він здатний заощадити пропускну здатність, а також припинити завантаження зовнішніх файлів.

Історія

У минулих системах обчислень, таких як клієнт-сервер, навантаження в цілому розподілялось між сервером та клієнтом. Проте таке рішення могло надто сильно навантажувати клієнтську систему. Також при оновленні коду на сервері, клієнтська частина також потребувала змін. Архітектура, зазвичай, в них також була однакова. Це означало що програми написані по такій моделі були в основному закриті для розширення, оскільки це ставало все складніше.

На відміну від цього, веб-додатки використовують веб-документи, написані в стандартних форматах, таких як HTML і JavaScript, які підтримуються різними веб-браузерами. Веб-додаток можна розглядати як специфічний варіант клієнт-серверного програмного забезпечення, де клієнтське програмне забезпечення завантажується на клієнтський комп'ютер за допомогою стандартних процедур (наприклад, HTTP) під час доступу до відповідної веб-сторінки. Клієнтські веб-програми можна оновлювати щоразу, коли здійснюється доступ до веб-сторінки. Під час сеансу веб-браузер інтерпретує та відображає сторінку та діє як загальний клієнт для будь-якої веб-програми.

У 1995 році Netscape представив мову сценаріїв на стороні клієнта під назвою JavaScript, що дозволяє програмістам додавати деякі динамічні елементи до інтерфейсу користувача, який працював 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.[1]

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.[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.[3]

У 2007 році Стів Джобс оголосив, що веб-додатки, розроблені на HTML5 з використанням архітектури AJAX, будуть стандартним форматом для додатків 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.

Ці технології також підтримують такі методи загального призначення, як перетягування. 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 зазвичай поділяються на логічні блоки, які називаються «рівнями», де кожному рівню призначається роль. [4] Традиційні програми складаються лише з рівня 1 і знаходяться на клієнтському комп'ютері, але веб-додатки за своєю природою підпорядковані дворівневому підходу. [4] Хоча можливо багато варіантів, найпоширенішою структурою є трирівнева програма. [4] У своїй найпоширенішій формі ці три рівні називаються представленням, додатком і зберіганням у такому порядку. Веб-браузер — це перший рівень (презентація), механізм, за допомогою якого деякі технології (наприклад, ASP, CGI, ColdFusion, Dart, JSP/Java, Node.js, PHP, Python або Ruby on Rails), які використовують динамічний веб-контент. Середній рівень (логіка програми), база даних третій рівень (сховище). [4] Веб-браузери надсилають запити на середній рівень, який обслуговує їх, запитуючи та оновлюючи базу даних, а також створюючи інтерфейс користувача.

Для більш складних додатків 3-рівневе рішення може виявитися недостатнім, і може бути корисно використовувати n-рівневий підхід, де найбільша перевага полягає в тому, щоб розбити бізнес-логіку, яка знаходиться на рівні програми, на більш детальну модель.[4] Іншою перевагою може бути додавання рівня інтеграції, який відокремлює рівень даних від решти рівнів, забезпечуючи простий у використанні інтерфейс для доступу до даних.[4] Наприклад, доступ до даних клієнта буде здійснюватися шляхом виклику

функції "list_clients()" замість того, щоб робити запит SQL безпосередньо до таблиці клієнта в базі даних. Це дозволяє замінити базову базу даних без будь-яких змін на інших рівнях.[4]

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.

Веб-розробка - це діяльність, яка пов’язана з розробкою веб-сайту для Інтернету (Всесвітня павутина) або інтрамережі (приватна мережа). [1] Веб-розробка може включати в себе як односторінкову розробку, тобто статичну, так і може включати розробку «клієнт-серверної» архітектури. Більш повний перелік завдань, до яких зазвичай відноситься веб-розробка, може включати 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.

Серед веб-спеціалістів "веб-розробка" зазвичай відноситься до основних недизайнерських аспектів побудови веб-сайтів: написання розмітки та кодування. [2] Веб-розробка може використовувати системи управління вмістом (CMS), щоб зробити зміст вмістом простішим та доступнішим з базовими технічними навичками.

Для великих організацій та бізнесу групи веб-розробників можуть складатися з сотень людей (веб-розробників) і дотримуватися стандартних методів, таких як Agile, під час розробки веб-сайтів. Менші організації можуть вимагати лише одного постійного або підрядного розробника, або вторинне призначення на відповідні посади, такі як графічний дизайнер або технік інформаційних систем. Веб-розробка може бути спільним зусиллям між

департаментами, а не областю призначеного департаменту. Існує три різновиди спеціалізації веб-розробників: розробник з інтерфейсу, розробник з інтерфейсу та розробник із повним стеком. Інтернетні розробники несуть відповідальність за поведінку та візуальні ефекти, які працюють у браузері користувача, тоді як інтерфейсні розробники мають справу з серверами.

З моменту комерціалізації Інтернету веб-розробка стала зростаючою галуззю. Зростання цієї галузі відбувається за рахунок підприємств, які бажають використовувати свій веб-сайт для реклами та продажу товарів та послуг споживачам. [3]

Існує багато інструментів з відкритим кодом для веб-розробки, таких як BerkeleyDB, GlassFish, стеки LAMP (Linux, Apache, MySQL, PHP) і Perl/Plack. Це мінімізує витрати на навчання веб-розробці. Іншим фактором, що стимулює зростання галузі, є зростання простого у використанні програмного забезпечення для веб-розробки WYSIWYG, такого як Adobe Dreamweaver, BlueGriffon та Microsoft Visual Studio. Використання такого програмного забезпечення все ще вимагає знання мови гіпертекстової розмітки (HTML) або мов програмування, але основи можна вивчити та впровадити швидко.

Постійно зростаючий набір інструментів і технологій допоміг розробникам створювати більш динамічні та інтерактивні веб-сайти. Крім того, веб-розробники зараз допомагають розробляти програми як веб-сервіси, які традиційно були доступні лише як програми для настільних комп’ютерів. Це дає багато можливостей для поширення децентралізованої інформації та засобів масової інформації. Приклади включають зростання хмарних сервісів, таких як Adobe Creative Cloud, Dropbox і Google Drive. Ці веб-сервіси дозволяють користувачам взаємодіяти з додатками з кількох місць, а не бути прив’язаними до певної робочої станції у своєму середовищі додатків.

Прикладами кардинальних перетворень у спілкуванні та комерції на чолі з веб-розробкою є електронна комерція. Інтернет-сайти аукціонів, такі як eBay, змінили спосіб споживачів знаходити та купувати товари та послуги. Інтернет-

магазини, такі як Amazon.com та eBay.com (серед багатьох інших), змінили досвід покупок та торгівлі для багатьох споживачів. Іншим прикладом трансформаційного спілкування, керованого веб-розробкою, є блог. Веб-програми, такі як WordPress та Movable Type, створили середовища для ведення блогів для окремих веб-сайтів. Поширене використання систем управління вмістом з відкритим кодом та систем управління корпоративним контентом розширило вплив веб-розробки на онлайн-взаємодію та спілкування.

Веб-розробка враховує багато міркувань безпеки, таких як перевірка помилок при введенні даних через форми, фільтрація вихідних даних та шифрування. Шкідливі практики, такі як введення SQL, можуть виконуватися користувачами з ненавмисними намірами, але лише з примітивними знаннями про веб-розробку в цілому. Сценарії можна використовувати для використання веб-сайтів, надаючи несанкціонований доступ зловмисним користувачам, які намагаються збирати таку інформацію, як адреси електронної пошти, паролі та захищений вміст, наприклад номери кредитних карток.

Якщо на веб-сайті є якась контактна форма, вона повинна включати в неї поле captcha, яке не дозволяє комп’ютерним програмам автоматично заповнювати форми, а також розсилати пошту.

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 безпеки та виявляються нові проблеми безпеки.

1.3. Поняття оцінювання знань

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. Змінено можливо, як зміст матеріалу, і форми, способи навчання. За необхідності вся система навчальної роботи то, можливо принципово перебудована. 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.

Можна сказати, що системою оцінювання є система оцінювання якості освоєння освітніх програм учням, найважливіший елемент освітнього процесу.

У світі є безліч шкал оцінювання знань. У деяких шкалах прийнято використовувати цифрові позначення розрядів, причому допускаються дробові оцінки, інші шкали (наприклад, США) за традицією мають справу з літерними позначеннями. Американська шкала також має чисельну інтерпретацію, при якій найвищим оцінкам A та A+ відповідають 4 та 5 балів відповідно.

Більшість країн має національну систему шкільних оцінок у своїх школах. Також існують і стандартні міжнародні системи оцінювання знань

1.4. Огляд існуючих рішень

Рішень, які повноцінно 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.

Можна сказати, що система тестування дає можливість швидше просканувати знання і отримати звіти щодо успішності. Існує декілька найбільш популярних та адекватних задачам рішень.

Взагалі, за допомогою програми перевірки та оцінювання знань можна зібрати тест чи опитування, призначити їх необхідним людям та швидко отримати результати.

Зазвичай процес роботи починається зі збору тесту. Програма надсилає посилання на завдання необхідним людям. Вони відповідають на запитання у зручний час із комп'ютера, планшета або телефону.

На виході можна отримати докладний звіт: хто пройшов тест, який бал набрав і яких помилок припустився. При цьому не важливо, скільки людей проходять таке тестування.

Системи тестування поділяються на два типи:

- коробкові
- хмарні.

Коробочні встановлюють на сервер компанії, а хмарні працюють через інтернет - достатньо створити обліковий запис.

Найбільш популярними існуючими рішеннями можна назвати наступні.

Таблиця 1 - Порівняльна таблиця існуючих систем перевірки та оцінювання знань

Система	Тип	Особливість	Види тестів	Ціна
1. SunRav Web Class	Коробкова версія	Майданчик для зберігання	можна «правильно - неправильно	від 13000 гривень за

		готових тестів. Щоб створювати завдання, знадобиться окремий конструктор tMaker	», «бал від 1 до 10»	ліцензію
2. iSpring	Хмарна версія	Платформа тестування та дистанційного навчання для бізнесу. Автоматизує атестацію навчання та допомагає перевести очні тренінги до онлайн-формату	можна збирати опитування, психологічні тести та тести на перевірку знань. 14 типів завдань: відповідність, вибір одного або декількох варіантів відповіді, вибір області, drag-and-drop, послідовність.	від 38 гривень на місяць за користувача
3. StartExam	Хмарна версія	Є метод 360 градусів	опитування та перевірочні тести з 9 типів завдань: єдиний та множинний вибір, сортування, відповідність, текстове	від 2790 гривень на місяць

			введення, есе, шкала Лікерта, відео інтерв'ю та оцінка 360 градусів.	
4. Indigo	Хмарна та коробкова версії	Недороге рішення. Можна створювати тести, анкети та відслідковувати результати	три: опитування, тест на перевірку знань та навчальний тест	від 460 гривень на місяць.

Також можна згадати такі популярні 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.

1.5. Постановка задачі

Задачею виконання роботи є створення веб-додатку для перевірки та оцінювання знань.

Основними задачами при розробці є організація таких функціональних можливостей:

- Можливість реєстрації студента та вчителя
- Створення тестування
- Формування результатів відповідей на тести студентами
- Проходження тестування.

1.6. Висновки до розділу 1

Веб-розробка – це робота з розробки веб-сайту для Інтернету (World Wide Web) або інтранету (приватна мережа). Веб-розробка може варіюватися від розробки простої статичної сторінки звичайного тексту до складних Інтернет-програм (інтернет-додатків), т. -послуги для бізнесу та соціальних мереж. Для великих організацій і підприємств команди веб-розробників можуть складатися з сотень людей (веб-розробників).Контроль у навчанні - це процес, суть якого полягає в забезпеченні так званого зворотного зв'язку між студентом і викладачем. Вона спрямована на отримання певної інформації, яка потім аналізується викладачем, щоб швидко внести необхідні корективи в ході навчання. Можлива зміна як змісту матеріалу, так і форм і методів навчання. Веб-сайт – це сукупність веб-сторінок і пов’язаного з ними вмісту, які ідентифікуються загальним доменним іменем і публікуються принаймні на одному веб-сервері.

Зростання цієї галузі відбувається завдяки компаніям, які хочуть використовувати свій веб-сайт для реклами та продажу товарів і послуг

споживачам. Скрипти можна використовувати для використання веб-сайтів для надання несанкціонованого доступу зловмисним користувачам. Багато розробників часто використовують різні форми шифрування під час передачі та зберігання конфіденційної інформації. Системи електронних журналів — це комерційні проекти, які коштують великих грошей і не є загальнодоступними, що унеможливлює їх перегляд.

РОЗДІЛ 2 АНАЛІЗ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Огляд мови програмування

Java — це високорівнева, заснована на класах, об'єктно-орієнтована мова програмування, яка розроблена так, щоб мати якомога менше залежностей реалізації. Це мова програмування загального призначення, призначена для того, щоб програмісти могли писати один раз, запускати в будь-якому місці (WORA),[17] що означає, що скомпільований код Java може працювати на всіх платформах, які підтримують Java, без необхідності перекомпіляції.[18] Програми Java зазвичай компілюються у байт-код, який може працювати на будь-якій віртуальній машині Java (JVM) незалежно від базової архітектури комп’ютера. Синтаксис Java подібний до C і C++, але має менше засобів низького рівня, ніж будь-який з них. Середовище виконання Java надає динамічні можливості (такі як відображення та модифікація коду під час виконання), які зазвичай недоступні в традиційних скомпільованих мовах. Станом на 2019 рік Java була однією з найпопулярніших мов програмування, що використовуються відповідно до GitHub, [19][20] особливо для веб-додатків клієнт-сервер, із 9 мільйонами розробників.[21]

Спочатку Java була розроблена Джеймсом Гослінгом у Sun Microsystems та випущена в травні 1995 року. Станом на травень 2007 року, відповідно до специфікацій Java Community Process, Sun переліцензувала більшість своїх технологій Java за ліцензією лише GPL-2.0. Oracle пропонує власну віртуальну машину HotSpot Java, однак офіційною довідковою реалізацією є OpenJDK JVM, яка є безкоштовним програмним забезпеченням з відкритим вихідним кодом і використовується більшістю розробників і є JVM за замовчуванням для майже всіх дистрибутивів Linux.

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 за нульовою вартістю 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 настійно рекомендують переходити на більш нові версії Java, для вирішень проблем пов'язаних з безпекою в застарілих версіях. [22] Вони радять користувачам перейти на версії, які отримали подовжену підтримку від Oracle: LTS (8, 11, 17).

Однією з цілей дизайну Java є переносимість, що означає, що програми, написані для платформи Java, повинні працювати подібним чином на будь-якій комбінації апаратного забезпечення та операційної системи з належною підтримкою часу виконання. Це досягається шляхом компіляції коду мови Java в проміжне представлення, яке називається байт-кодом Java, замість безпосереднього машинного коду, специфічного для архітектури. Інструкції байт-коду Java аналогічні машинному коду, але вони призначені для виконання віртуальною машиною (VM), написаною спеціально для апаратного забезпечення хоста. Кінцеві користувачі зазвичай використовують Java Runtime Environment (JRE), встановлену на своєму пристрої для автономних програм Java або веб-браузер для Java-аплетів.

Використання універсального байт-коду спрощує перенесення. Однак накладні витрати на інтерпретацію байт-коду в машинні інструкції змушували інтерпретовані програми майже завжди працювати повільніше, ніж рідні виконувані файли. Компілятори Just-in-time (JIT), які компілюють байт-код у машинний код під час виконання, були представлені на ранньому етапі. Компілятор Hotspot Java насправді є двома компіляторами в одному; і з GraalVM

(включений, наприклад, у Java 11, але вилучений з Java 16), що дозволяє багаторівневу компіляцію. Сама Java не залежить від платформи і адаптована до конкретної платформи, на якій вона має працювати, за допомогою віртуальної машини Java (JVM), яка перекладає байт-код Java на машинну мову платформи.

Програми, написані на Java, мають репутацію повільніших і вимагають більше пам’яті, ніж програми, написані на C++.Однак швидкість виконання програм Java значно покращилася із запровадженням компіляції «точно вчасно» в 1997/1998 роках для програм на Java. Java 1.1, додавання мовних функцій, які підтримують кращий аналіз коду (наприклад, внутрішні класи, клас StringBuilder, необов’язкові твердження 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 безблоковані реалізації ConcurrentMaps та інших багатоядерних колекцій, а також покращено з Java 1.6.

Java використовує автоматичний збірник сміття (Garbage Collector) для управління пам'яттю в життєвому циклі об'єкта. Програміст може явно визначити час створення об’єктів, а середовище виконання Java відповідає за відновлення пам’яті, коли об’єкти більше не використовуються. Очистка об’єктів, які вже не використовуються проходить в два етапи: Mark (коли система ідентифікує та помічає об’єкти, які потребують видалення з пам’яті) та Sweep (етап, коли Garbage Collector видаляє об’єкти з пам’яті, які були промарковані на етапі Mark).

Одна з ідей, що лежить в основі моделі автоматичного управління пам’яттю Java, полягає в тому, що програмісти можуть позбутися від тягаря виконання ручного керування пам’яттю. У деяких мовах пам'ять для створення об'єктів неявно виділяється в стеку або явно виділяється і звільняється з купи. В останньому випадку відповідальність за управління пам'яттю покладається на програміста. В Java цей процес повністю автоматизований, та програмісту не потрібно явно вказувати об’єкти, які вже не будуть використовуватись в пам’яті.

Збір сміття може статися в будь-який момент, та сам програміст не може знати впевнено в який саме.

Java не підтримує арифметику вказівників у стилі C/C++, де адресами об’єктів можна арифметично маніпулювати (наприклад, додаючи або віднімаючи зміщення). Це дозволяє збірнику сміття переміщувати об’єкти, на які посилаються, і гарантує безпеку та безпеку типу.

Java містить кілька типів збирачів сміття. Починаючи з Java 9, HotSpot використовує Garbage First Garbage Collector (G1GC) за замовчуванням. Однак є також кілька інших збирачів сміття, які можна використовувати для керування купою. Для більшості програм на Java достатньо G1GC. Раніше в Java 8 використовувався Parallel Garbage Collector.

Вирішення проблеми управління пам’яттю не звільняє програміста від тягаря належної обробки інших видів ресурсів, таких як мережеві з’єднання або підключення до бази даних, дескриптори файлів тощо, особливо за наявності винятків.

Jakarta Servlet (раніше Java Servlet) — це програмний компонент Java, який розширює можливості сервера. Хоч сервлети можуть відповідати на багато типів запитів, вони найчастіше реалізують веб-контейнери для розміщення веб-додатків на веб-серверах і, таким чином, кваліфікуються як веб-API сервлетів на стороні сервера.

Сервлет Jakarta обробляє або зберігає клас Java в Jakarta EE, який відповідає API сервлетів Jakarta [1], 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».[2] 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, а також
- 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.[2]

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.

Поточна версія Servlet 5.0.[3]

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[1], 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.[2]

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.[3]

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.[4]

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 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.[9]

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.[10]

2.2. Огляд середовища розробки

IntelliJ IDEA — це інтегроване середовище розробки (IDE), написане на Java для розробки комп’ютерного програмного забезпечення. Він розроблений компанією JetBrains (раніше відомий як IntelliJ) і доступний у вигляді ліцензованої спільноти Apache 2[2] та у приватній комерційній версії. Обидва можна використовувати для комерційного розвитку.[3][4]

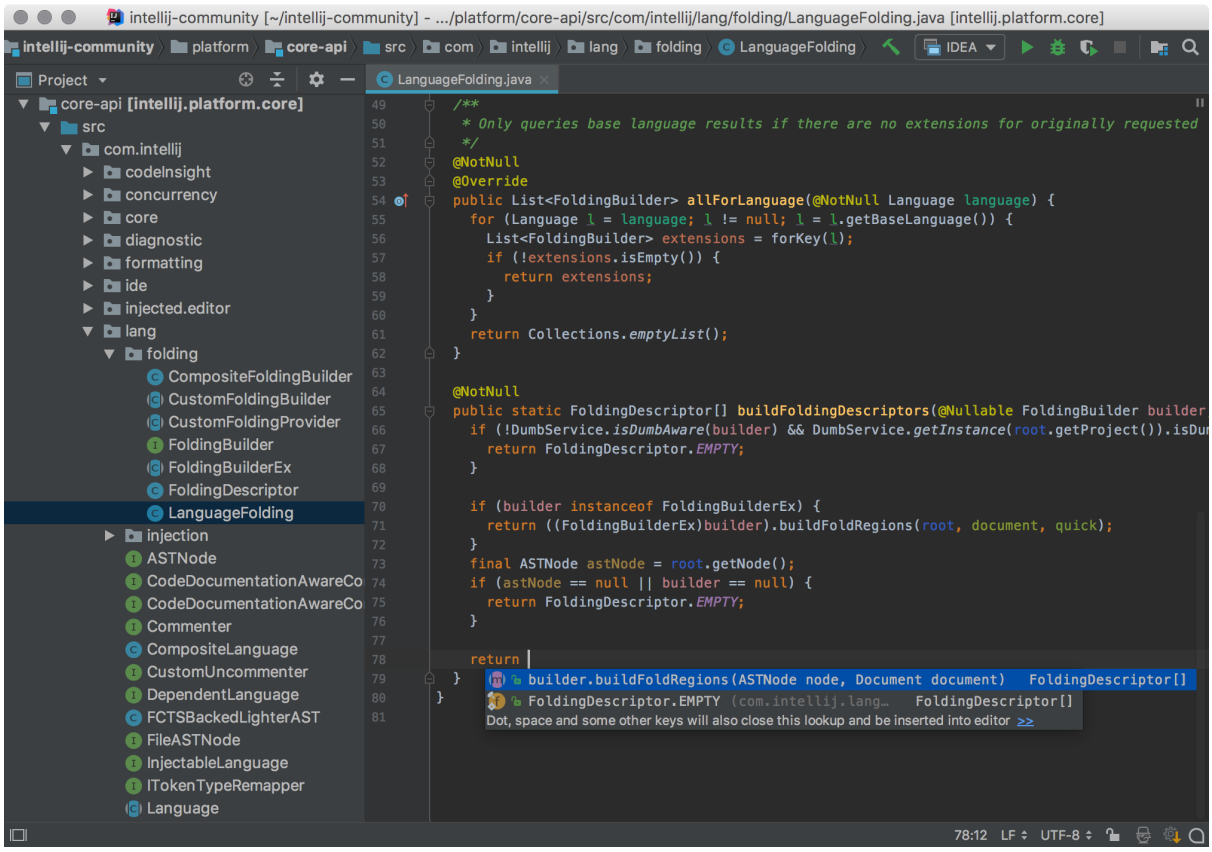


Рисунок 1 - Інтерфейс системи JetBrains Inteliji IDEA

Перша версія IntelliJ IDEA була випущена в січні 2001 року і була однією з перших доступних середовищ IDE Java з інтегрованими можливостями розширеної навігації та рефакторингу коду.[5][6].

У звіті InfoWorld за 2010 рік IntelliJ отримав найвищу оцінку центру тестування з чотирьох найкращих інструментів програмування Java: Eclipse, IntelliJ IDEA, NetBeans і JDeveloper.[7]

У грудні 2014 року Google оголосила про версію 1.0 Android Studio, IDE з відкритим кодом для програм Android, засновану на версії IntelliJ IDEA для спільноти з відкритим кодом.[8] Інші середовища розробки, засновані на фреймворку IntelliJ, включають AppCode, CLion, DataGrip, GoLand, PhpStorm, PyCharm, Rider, RubyMine, WebStorm і MPS.[9]

Основними можливостями системи є

- Ергономічне середовище

IntelliJ IDEA продумана у кожному аспекті та готова до використання одразу після встановлення. Середовище забезпечує швидкий доступ до всіх функцій та вбудованих інструментів, необхідних розробнику, а також широкі можливості індивідуального налаштування. Можна повністю налаштувати середовище відповідно до робочого процесу: задати поєднання клавіш, встановити плагіни, налаштувати інтерфейс на розсуд розробника і т.д.

- Глибокий аналіз коду

IntelliJ IDEA була створена в першу чергу для розробки на Java, але вона розуміє і багато інших мов програмування, у тому числі Groovy, Kotlin, Scala, JavaScript, TypeScript та SQL, і пропонує інтелектуальну допомогу в написанні коду на кожній з цих мов. Початкова індексація вихідного коду дозволяє IDE створити віртуальну карту проекту. Використовуючи інформацію віртуальної карти, вона миттєво виявляє помилки, пропонує варіанти автодоповнення коду з урахуванням контексту, виконує рефакторинг тощо.

- Миттєва навігація та пошук

У IntelliJ IDEA є безліч різноманітних функцій для прискорення та спрощення навігації та пошуку. Вони допомагають зосередитися на написанні коду та працювати швидше

- Просунуті можливості запуску, тестування та налагодження

В IntelliJ IDEA вбудований ефективний набір інструментів для налаштування параметрів запуску та складання програми, налагодження коду, а також застосування та розробки тестів JUnit прямо в IDE.

- Вбудовані інструменти та інтеграція

IntelliJ IDEA пропонує важливі вбудовані інструменти та можливості інтеграції, завдяки яким ви можете працювати у звичному середовищі, не перемикаючись між різними програмами.

- Інтеграція із системами контролю версій

IntelliJ IDEA за промовчанням підтримує найпопулярніші системи контролю версій, такі як Git, Subversion, Mercurial та Perforce. Проект із системи

контролю версій можна клонувати прямо на початковому екрані, проаналізувати різницю між двома версіями, керувати гілками, записувати та відправляти зміни, вирішувати конфлікти злиття, переглядати історію змін тощо.

- Фреймворки JVM

IntelliJ IDEA Ultimate забезпечує першокласну підтримку провідних фреймворків та технологій для розробки сучасних додатків та мікросервісів. В IDE вбудована підтримка Spring та Spring Boot, Jakarta EE, JPA, Reactor та інших фреймворків.

- Розробка на JavaScript та створення клієнтських додатків

Система пропонує всі необхідні можливості WebStorm – нашої IDE для JavaScript та пов'язаних технологій. Усі функції або входять у комплект поставки, або доступні як безкоштовних плагінів.

- Просунуті можливості розгортання

Щоб не відставати від сучасних тенденцій у розробці ПЗ, IntelliJ IDEA Ultimate пропонує інтеграцію з найпопулярнішими системами управління контейнерами: Kubernetes та Docker. Крім того, у нас є сторонні плагіни для розгортання коду в AWS, Google Cloud та Azure.

- Можливість сумісної праці

IntelliJ IDEA пропонує сервіс спільної розробки та парного програмування: Code With Me. З його допомогою ви можете працювати в реальному часі над проектом, відкритим у вашій IDE, разом з колегами. Крім того, Code With Me підтримує відео- та голосові виклики з IDE, дозволяючи організувати зустрічі як віч-на-віч, так і за участю десятків людей.

IntelliJ IDEA інтегрована з JetBrains Space - комплексним рішенням для команд розробників та проектів створення ПЗ. Підключивши IntelliJ IDEA до своєї організації у Space, ви зможете переглядати та клонувати репозиторії проекту, читати код своїх колег, а також писати скрипти автоматизації Space.

IntelliJ IDEA Ultimate підтримує процес віддаленої розробки у бета-режимі. Тепер можна підключатися до віддаленого сервера, на якому запущено бекенд

IntelliJ IDEA, з будь-якої точки світу. Вся обробка даних виконується на потужній віддаленій машині, а ви працюєте над проектом, ніби він розміщений локально. До цієї функції можна перейти з початкового екрана IntelliJ IDEA або з нової програми JetBrains Gateway, яка встановлюється через Toolbox App..

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. Браузер може отримувати дані, які передає сервер в виді HTML-документу, або ж JSON об’єкту.

Елементи HTML є будівельними блоками сторінок HTML. За допомогою структури HTML зображення та інші об’єкти (наприклад, інтерактивні форми) можуть бути вбудовані в скопійовані сторінки. HTML надає інструменти для створення структурованих документів, які представляють структурну семантику тексту, наприклад заголовки, абзаци, списки, посилання, посилання та інші елементи. Елементи HTML розділені тегами, записаними в кутових дужках. Такі теги, як <img /> і <input /> вносять вміст безпосередньо на сторінку. Інші теги, такі як <p>, оточують і надають інформацію про текст документа, а також можуть включати інші теги як дочірні елементи. Замість відображення тегів HTML браузери використовують їх для інтерпретації вмісту сторінки.

HTML може вбудовувати програми, написані мовою сценаріїв, наприклад JavaScript, що впливає на поведінку та вміст веб-сторінок. Включення CSS визначає вигляд і макет вмісту. Консорціум World Wide Web Consortium (W3C), колишній розпорядник HTML і нинішній розпорядник стандартів CSS, заохочує використання CSS замість явного презентаційного HTML з 1997 року.[2] Форма

HTML, відома як HTML5, використовується для відображення відео та аудіо, переважно за допомогою елемента <canvas> у співпраці з javascript.

Розмітка HTML складається з кількох ключових компонентів, включаючи ті, які називаються тегами (та їх атрибутами), типи даних на основі символів, посилання на символи та посилання на сутність. HTML-теги найчастіше зустрічаються парами, як-от <h1> і </h1>, хоча деякі представляють порожні елементи і тому є непарними, наприклад <img>. Першим тегом у такій парі є початковий тег, а другий — кінцевий (їх ще називають відкриваючими та закриваючими тегами).

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 на увазі структуру вкладених елементів HTML. Вони представлені в документі тегами HTML, укладеними в кутові дужки наступним чином: .

У простому загальному випадку область дії елемента визначається парою тегів: «відкриваючий тег» і «завершальний тег» . Текстовий вміст елемента, якщо такий є, розміщується між цими тегами.

Мітки також можуть містити додаткові теги міток між початком і кінцем, включаючи поєднання міток і тексту. Це вказує наступні (вкладені) елементи як дочірні елементи батьківського елемента.

Початковий тег також може включати атрибути елементів всередині тегу. Вони вказують додаткову інформацію, таку як ідентифікатор розділу в документі, ідентифікатор, який використовується для прив’язки інформації про стиль до представлення документа, а для деяких тегів, наприклад <img> для вбудовування зображень, посилання на ресурси зображень у таких документах. : <img src="example.com/example.jpg">

Деякі елементи, такі як розриви рядків або
, не дозволяють вставляти будь-який вміст, текст чи інші теги. Вони вимагають лише порожнього тега (як початкового тегу) і не використовують закриваючий тег.

Багато тегів, включаючи дуже поширений закриваючий тег елемента абзацу , є необов’язковими. Браузер HTML або інший проксі-сервер може визначити кінець елемента відповідно до контекстних і структурних правил, визначених стандартом HTML. Ці правила складні, і більшість програмістів HTML не розуміють.

Таким чином, загальна форма елемента HTML: <tag attribute1="value1" attribute2="value2">"content"</tag>. Деякі елементи HTML визначаються як

порожні елементи і мають форму <tag attribute1="value1" attribute2="value2">. Порожні елементи можуть не містити вмісту, наприклад, тег
 або вбудований тег <img>. Ім'я елемента HTML - це ім'я, яке використовується в тегах. Зверніть увагу, що назві кінцевого тегу передує символ косої риски /, і що в порожніх елементах кінцевий тег не є ані обов'язковим, ані дозволеним. Якщо атрибути не згадуються, у кожному випадку використовуються значення за замовчуванням.

Структурна розмітка вказує на призначення тексту

Наприклад, <h2>Golf</h2> встановлює "Гольф" як заголовок другого рівня. Структурна розмітка не позначає жодного конкретного візуалізації, але більшість веб-браузерів мають стилі за замовчуванням для форматування елементів. Вміст можна додатково стилізувати за допомогою каскадних таблиць стилів (CSS).

Презентаційна розмітка вказує на зовнішній вигляд тексту, незалежно від його призначення

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.[5] 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.[7]

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 (виглядом). HTML включав семантичну розмітку з самого початку,[7] 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.

Під час дискусії 2001 року про семантичну мережу Тім Бернерс-Лі та інші навели приклади того, як інтелектуальні програмні «агенти» можуть одного дня автоматично сканувати Інтернет і знаходити, фільтрувати та співвідносити раніше не пов’язані опубліковані факти на користь користувачів. Такі агенти навіть зараз не є звичайним явищем, але деякі ідеї Web 2.0, мішапи та веб-сайти порівняння цін можуть наблизитися. Основна відмінність між цими гібридами веб-додатків і семантичними агентами Бернерса-Лі полягає в тому, що поточну агрегацію та гібридизацію інформації зазвичай розробляють веб-розробники, які вже знають веб-розташування та семантику API конкретних даних, які вони бажають. м’яти, порівнювати та з’єднувати.

Веб-сканер або пошукова система є важливим типом веб-проксі, який автоматично сканує та читає веб-сторінки, не знаючи, що він може знайти. Ці програмні агенти покладаються на семантичну чіткість веб-сторінок, які вони знаходять, оскільки вони використовують різні методи та алгоритми для читання та індексації мільйонів веб-сторінок щодня, а також надають користувачам Інтернету інструменти пошуку, які значно знизять корисність у всьому світі. Інтернет.

Щоб павуки пошукових систем розуміли важливість фрагментів тексту, які вони знаходять у HTML-документах, а також для тих, хто створює діаграми сітки та інші гібриди, а також у більш автоматизованих проксі, які вони розробляють, існуюча семантична структура в HTML повинна використовуватися широко і рівномірно. Використовується для визначення значення опублікованого тексту. [17]

Теги презентаційної розмітки не підтримуються в поточних рекомендаціях HTML і XHTML. Більшість презентаційних функцій з попередніх версій HTML більше не дозволені, оскільки вони призводять до погіршення доступності, дорожчих витрат на обслуговування сайту та збільшення розмірів документів.[17]

Хороший семантичний HTML також покращує доступність веб-документів (див. також Інструкції щодо доступності веб-вмісту). Наприклад, коли програми зчитування з екрана або аудіобраузери можуть правильно структурувати документ, вони не витрачатимуть час на читання повторюваної або невідповідної інформації для користувачів із вадами зору, якщо вони правильно розмічені.

Каскадні таблиці стилів (CSS) — це мова таблиць стилів, яка використовується для опису подання документа, написаного мовою розмітки, як-от HTML.[1] CSS є наріжною технологією всесвітньої мережі, поряд з HTML і JavaScript.[2]

CSS призначений для розділення презентації та вмісту, включаючи макет, кольори та шрифти. [3] Це поділ покращує зручність використання вмісту; забезпечує більшу гнучкість та контроль у специфікації функцій презентації; дозволяє кільком веб-сторінкам спільно використовувати форматування, вказуючи відповідний CSS в окремих файлах .css, зменшуючи складність і повторюваність структурованого вмісту. А також увімкніть кешування файлів .css, щоб покращити швидкість завантаження сторінки та спільний доступ до файлів.

Розділення формату та вмісту також дозволяє відображати ту саму сторінку розмітки в різних стилях для різних методів візуалізації, наприклад, на екрані, у друкованому вигляді, мовленні (через мовний браузер або програму зчитування з екрана) і шрифт Брайля. Тактильні пристрої. CSS також має альтернативні правила форматування під час доступу до вмісту на мобільних пристроях. [4]

Каскадність назв походить від зазначеної схеми пріоритету, щоб визначити, яке правило стилю застосовується, якщо більше одного правила відповідає певному елементу. Ця каскадна схема пріоритетів є передбачуваною.

Специфікації CSS підтримуються Консорціумом World Wide Web Consortium (W3C). Тип Інтернет-медіа (тип MIME) text/css зареєстровано для використання з CSS згідно RFC 2318 (березень 1998 р.). W3C надає безкоштовну службу перевірки CSS для документів CSS.[5]

На додаток до HTML, інші мови розмітки підтримують використання CSS, включаючи XHTML, звичайний XML, SVG і XUL.

Блок декларацій

Блок оголошення 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.[9]

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.[10][11]

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) %)).[12]

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 (піса); 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[13] 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.[14]

Використовуйте

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.[15]

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.

MySQL [5] — це система управління реляційними базами даних з відкритим вихідним кодом (RDBMS). [5] [6] Його назва є комбінацією «My» (дочка

співзасновника Майкла Віденіуса [7]) і «SQL» (скорочення від Structured Query Language). Реляційна база даних організовує дані в одну або кілька таблиць даних, у яких типи даних можуть бути пов’язані; ці відносини допомагають структурувати дані. SQL — це мова, яку програмісти використовують для створення, модифікації та вилучення даних з реляційних баз даних і для контролю доступу користувачів до баз даних. На додаток до реляційних баз даних і SQL, такі бази даних, як MySQL, працюють з операційними системами, щоб впровадити реляційні бази даних у систему зберігання комп’ютера, керувати користувачами, забезпечувати доступ до мережі та полегшувати цілісність бази даних і резервне копіювання.

MySQL є безкоштовним програмним забезпеченням з відкритим вихідним кодом відповідно до умов GNU General Public License і доступне за різними ліцензіями власності. MySQL належить і спонсорується шведською компанією MySQL AB, яку придбала Sun Microsystems (тепер Oracle Corporation). [8] У 2010 році, коли Oracle придбала Sun, Widenius розширив проект MySQL з відкритим кодом, щоб створити MariaDB. [9]

MySQL має окремий клієнт, який дозволяє користувачам безпосередньо взаємодіяти з базою даних MySQL за допомогою SQL, але частіше MySQL використовується з іншими програмами для реалізації програм, які потребують функціональності реляційної бази даних. MySQL є компонентом стеку веб-додатків LAMP (та інших), який є акронімом для Linux, Apache, MySQL, Perl/PHP/Python. MySQL використовується багатьма веб-додатками баз даних, включаючи Drupal, Joomla, phpBB і WordPress. MySQL також використовується багатьма популярними веб-сайтами, включаючи Facebook,[10][11] Flickr,[12] MediaWiki,[13] Twitter[14] і YouTube.[15]

Огляд

MySQL написаний на C і C++. Його синтаксичний аналізатор SQL написаний на yacc, але він використовує саморобний lexer. [16] MySQL працює на багатьох системних платформах, включаючи AIX, BSDi, FreeBSD, HP-UX,

ArcaOS, eComStation, IBM i, IRIX, Linux, macOS, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2 Warp , QNX, Oracle Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos і Tru64. OpenVMS також має порт MySQL. [17]

Серверне програмне забезпечення MySQL і клієнтські бібліотеки використовують дистрибутив з подвійною ліцензією. Вони надаються відповідно до GPL версії 2 або вашої власної ліцензії. [18]

Підтримку можна знайти в офіційному посібнику. [19] Крім того, різноманітні канали та форуми IRC надають безкоштовну підтримку. Oracle пропонує платну підтримку через свій продукт MySQL Enterprise. Вони відрізняються за обсягом і ціною. Крім того, існує багато сторонніх організацій, які надають підтримку та послуги.

MySQL отримав позитивні відгуки, і рецензенти помітили, що він «надзвичайно добре працює в середньому», і що «інтерфейси розробника є, а документація (не кажучи вже про зворотний зв'язок у реальному світі через веб-сайти тощо) дуже, дуже добре".[20] Він також був протестований як «швидкий, стабільний і справжній багатокористувацький, багатопоточний сервер баз даних SQL».[21]

Розгортання

MySQL можна зібрати та встановити вручну з вихідного коду, але частіше він встановлюється з бінарного пакета, якщо не потрібні спеціальні налаштування. У більшості дистрибутивів Linux система керування пакетами може завантажити та встановити MySQL з мінімальними зусиллями, хоча для налаштування параметрів безпеки та оптимізації часто потрібна додаткова конфігурація.

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.

Також в розробцы використовувався Tomcat, що являє собою контейнер сервлетів з відкритим вихідним кодом, що розробляється Apache Software Foundation. Реалізує специфікацію сервлетів, специфікацію JavaServer Pages (JSP) та JavaServer Faces (JSF). Написаний мовою Java.

Tomcat дозволяє запускати веб-програми та містить ряд програм для самоконфігурування.

Tomcat використовується як самостійний веб-сервер, як сервер контенту в поєднанні з веб-сервером Apache HTTP Server, а також як контейнер сервлетів в серверах додатків JBoss і GlassFish.

Розробка та підтримка Tomcat здійснюється фондом Apache Software Foundation та добровольцями. Користувачі мають вільний доступ до вихідних кодів та бінарних файлів Tomcat згідно з ліцензією Apache License 2.0. Номери версій Tomcat починаються з 3.0.x (попередні версії випущені Sun для внутрішнього користування).

2.4 Висновки до розділу 2

Java — це високорівнева, заснована на класах, об’єктно-орієнтована мова програмування, розроблена для того, щоб мати якомога менше залежностей реалізації. Це мова програмування загального призначення, розроблена, щоб дозволити програмістам писати один раз, запускати де завгодно (WORA)

Програми Java зазвичай компілюються у байт-код, який може працювати на будь-якій віртуальній машині Java (JVM), незалежно від базової архітектури комп’ютера. Синтаксис Java подібний до C і C++.

Середовище виконання Java надає динамічні можливості (наприклад, відображення та зміна коду під час виконання), які зазвичай не зустрічаються в традиційних компільованих мовах. Станом на 2019 рік Java є однією з найпопулярніших мов програмування, яку використовує GitHub.

IntelliJ IDEA — це інтегроване середовище розробки (IDE), написане на Java для розробки комп’ютерного програмного забезпечення. Він розроблений компанією JetBrains (раніше відомий як IntelliJ) і доступний у вигляді ліцензованої спільноти Apache 2 та у приватній комерційній версії. Обидва можна використовувати для комерційного розвитку.

Додатковим інструментарієм є мова розмітки гіпертексту або HTML, каскадні таблиці стилів (CSS) і мови сценаріїв, такі як JavaScript.

Також в розробці використовувався Tomcat, що являє собою контейнер сервлетів з відкритим вихідним кодом, що розробляється Apache Software

Foundation. Реалізує специфікацію сервлетів, специфікацію JavaServer Pages (JSP) та JavaServer Faces (JSF). Написаний мовою Java.

РОЗДІЛ 3 ПРОЕКТУВАННЯ І РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1 Аналіз варіантів використання

Діаграма прецедентів або діаграма варіантів використання в UML - діаграма, що відображає відносини між актерами та прецедентами і є складовою моделі прецедентів, що дозволяє описати систему на концептуальному рівні.

Прецедент - можливість системи (частина її функціональності), завдяки якій користувач може отримати конкретний, вимірний і потрібний йому результат. Прецедент відповідає окремому сервісу системи, визначає один із варіантів її використання та визначає типовий спосіб взаємодії користувача з системою. Варіанти використання зазвичай застосовуються у специфікації зовнішніх вимог до системи.

Основне призначення діаграми - опис функціональності та поведінки, що дозволяє замовнику, кінцевому користувачеві та розробнику спільно обговорювати проектовану або існуючу систему.

При моделюванні системи за допомогою діаграми прецедентів системний аналітик прагне:

- чітко відокремити систему від її оточення;
- визначити дійових осіб (акторів), їхню взаємодію із системою та очікувану функціональність системи;
- визначити у глосарії предметної області поняття, що належать до детального опису функціональності системи (тобто прецедентів).

Першим етапом проектування системи є аналіз варіантів використання системи.

Для цього було розроблено діаграму варіантів використання, яка зображена на рисунку 3.1.

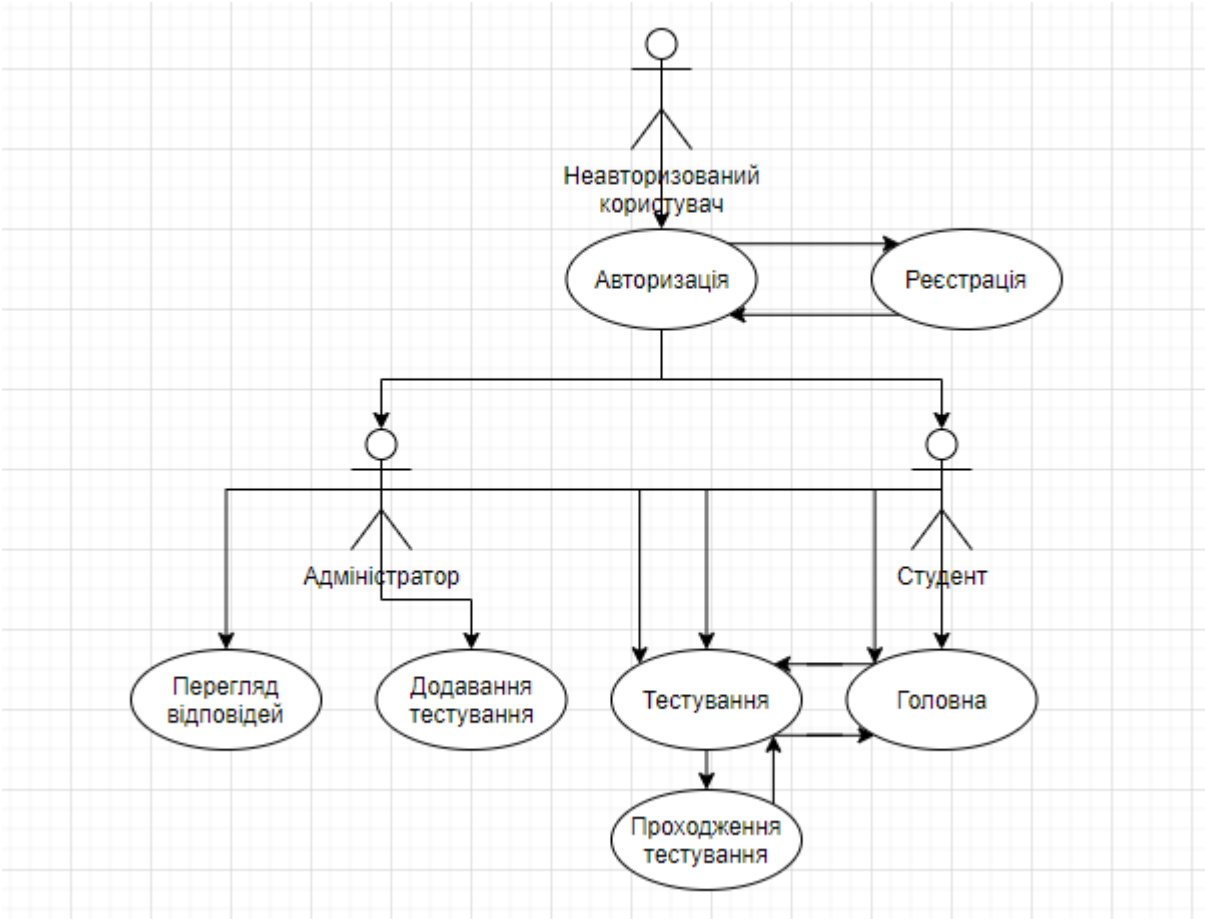


Рисунок 3.1 — Діаграма варіантів використання

Як видно з діаграми, користувач має можливість переглядати, додавати і видаляти записи з каталогу.

3.2. Проектування внутрішньої будови системи

Наступним кроком проектування є створення діаграми класів системи, яка, фактично, є декомпозицією діаграми пакетів на більш мілкі складові.

Діаграма класів повністю відображає внутрішню структуру системи.

Діаграма класів зображена на рисунку 3.3.

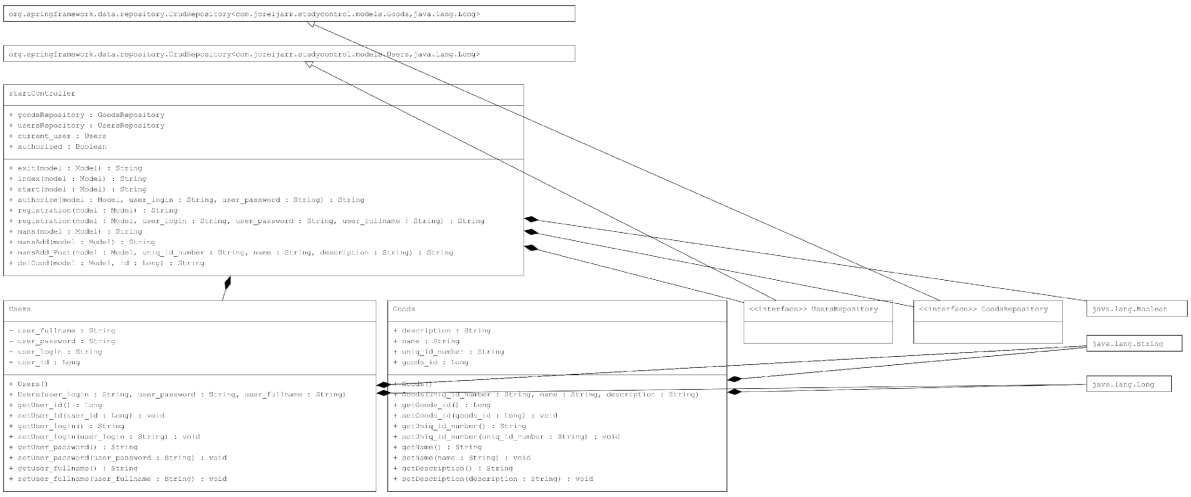


Рисунок 3.3 — Діаграма класів

3.3. Розробка графічного інтерфейсу системи

Графічним інтерфейсом користувача називається система засобів для взаємодії користувача з електронними пристроями, заснована на представленні всіх доступних користувачеві системних об'єктів і функцій у вигляді графічних компонентів екрану піктограм, меню, кнопок, списків тощо.

Найчастіше елементи інтерфейсу в GUI реалізовані на основі метафор та відображають їх призначення та властивості, що полегшує розуміння та використання електронних пристроїв непідготовленими користувачами.

Графічний інтерфейс користувача є частиною інтерфейсу користувача і визначає взаємодію з користувачем на рівні візуалізованої інформації.

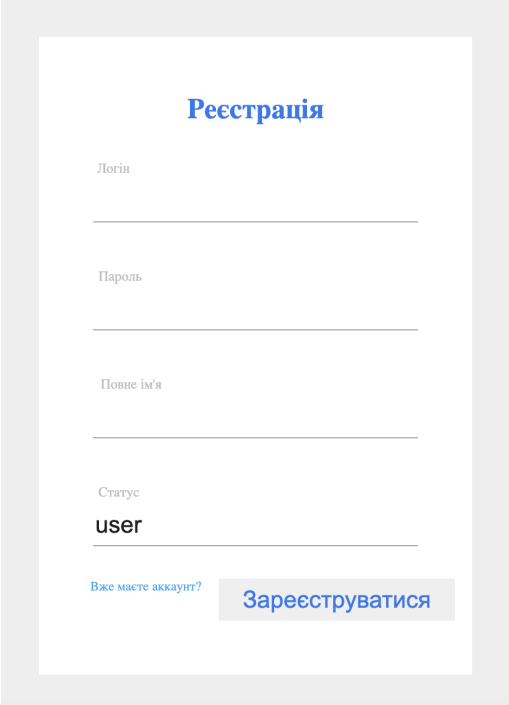


Рисунок 2 - Реєстрація нового студента

Після реєстрації студента його головна сторінка виглядає наступним чином.

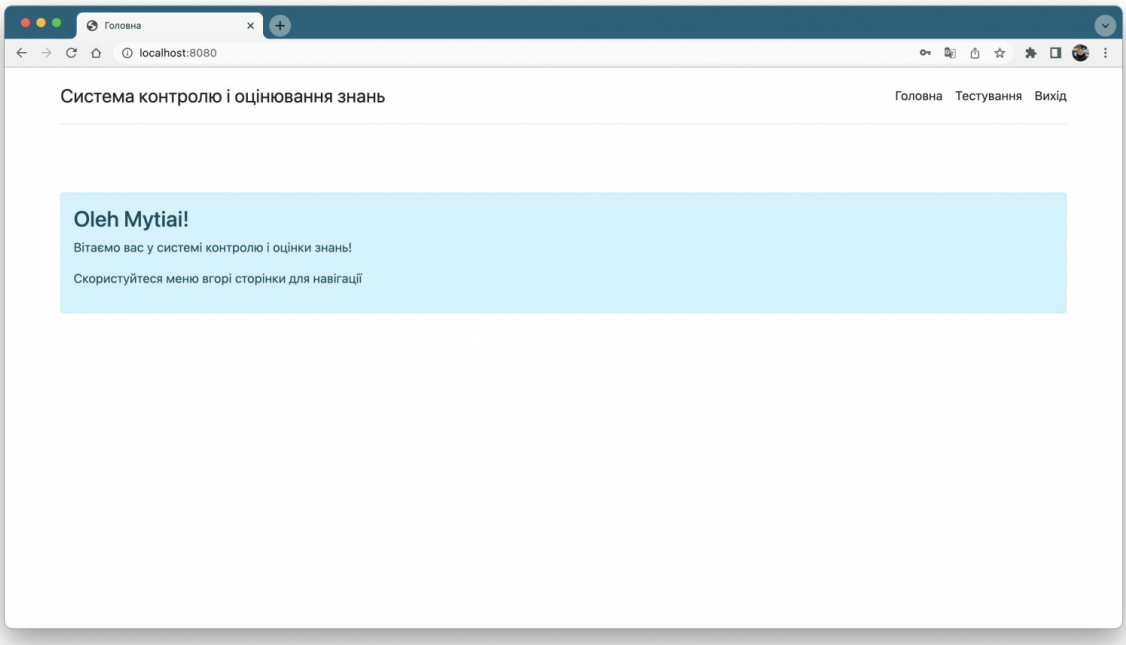


Рисунок 3 - Головна сторінка студента

Меню студента виглядає наступним чином.

Рисунок 4 - Меню студента

Рисунок 5 - Меню вчителя

При цьому головна сторінка вчителя виглядає наступним чином.

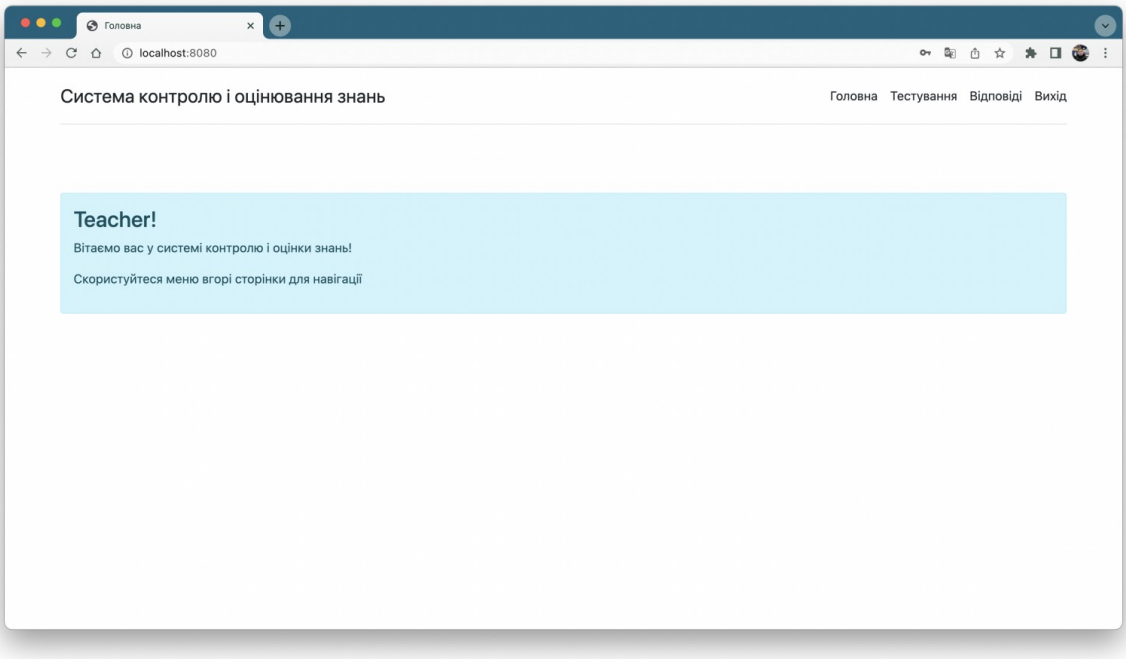


Рисунок 6 - Головна сторінка вчителя

Вчитель може створити тестування, інтерфейс чого представлений нижче.

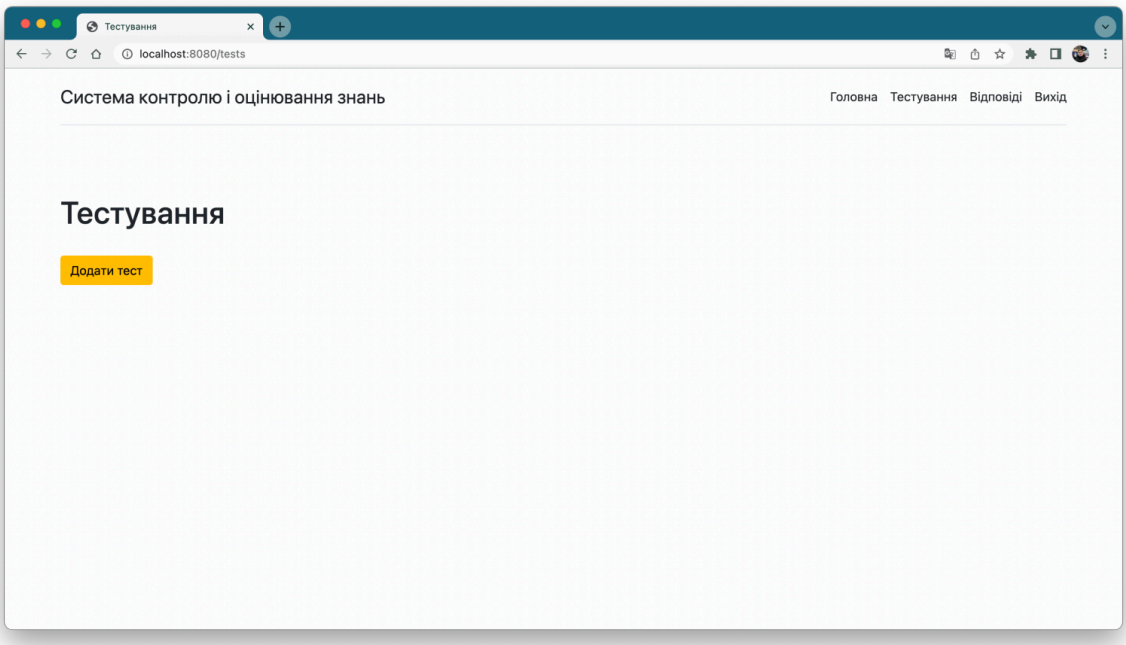


Рисунок 7 - Сторінка вчителя з можливістю створення тестування

Інтерфейс створення тесту виглядає наступним чином.

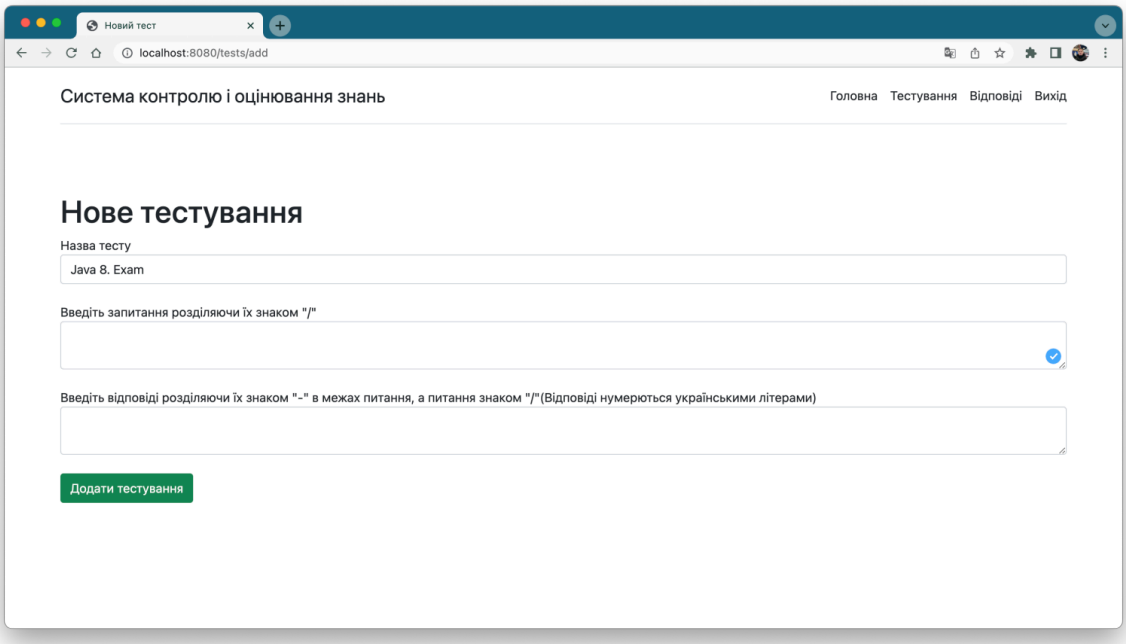


Рисунок 8 - Інтерфейс створення тесту

Інтерфейс відповіді на тести студентами виглядає так.

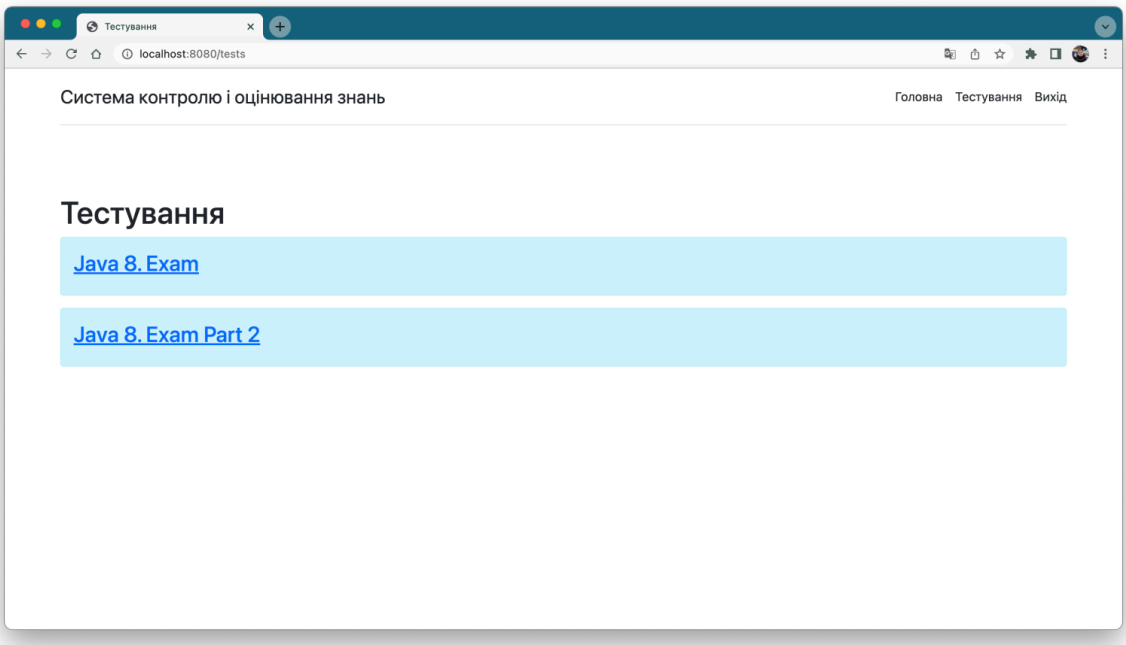


Рисунок 9 - Інтерфейс відповіді на тести

Сторінка проходження тесту виглядає так.

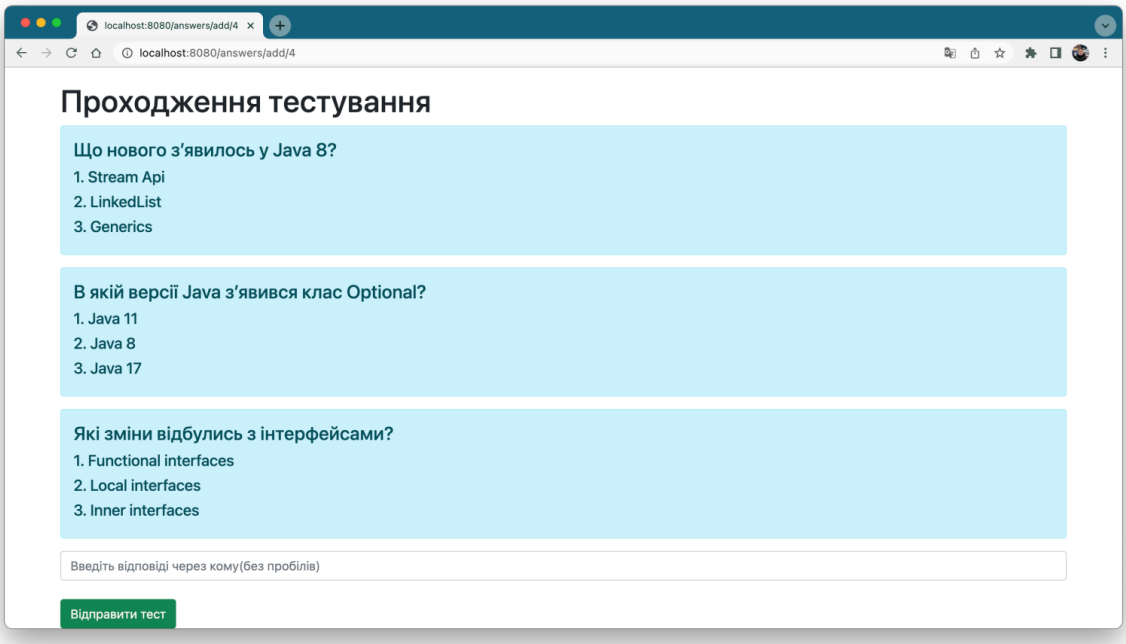


Рисунок 10 - Сторінка проходження тестів

Сторінка вчителя з відповідями студентів виглядає так.

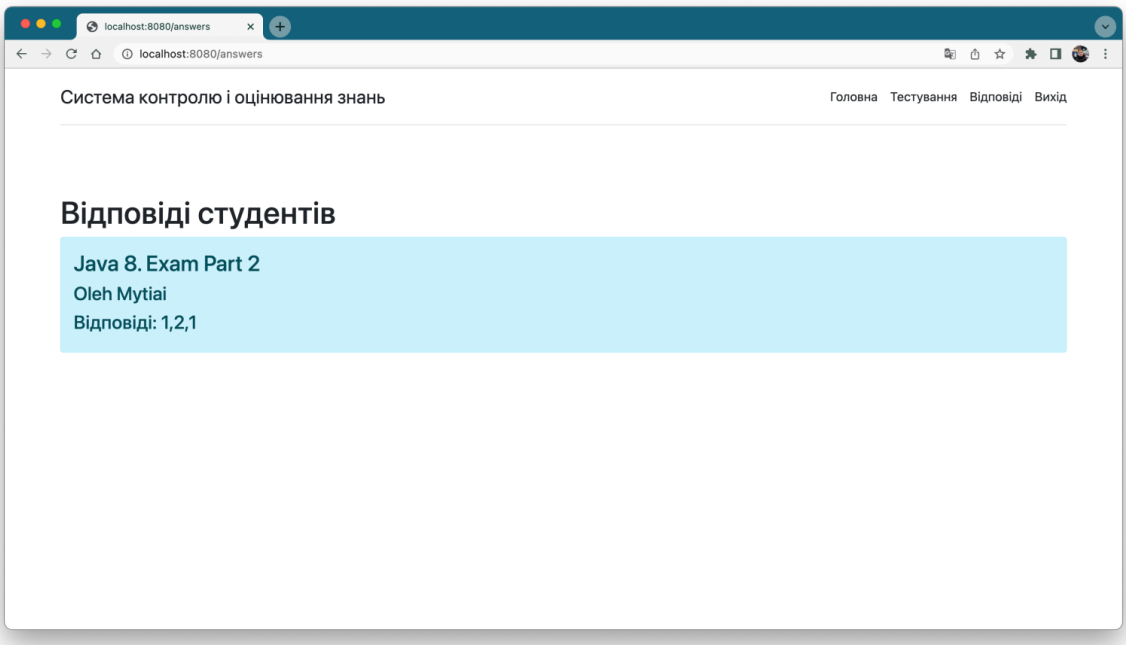


Рисунок 11 - Сторінка вчителя з відповідями студентів

3.4 Тестування і огляд результатів

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.

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	Введені невірні данні при авторизації	Система повідомляє користувача про те, що такого користувача не існує, або дані не валідні	При введенні невірних даних система повідомляє користувача про те, що такого користувача не існує, або дані введені невірно
2	Пусті поля при авторизації	Система повідомляє користувача про те, що поля не мають бути пустими	При спробі авторизації з пустими полями система повідомляє користувача про те, що ці поля необхідно заповнити.
3	Неспівпадаючі паролі при реєстрації	Система повідомляє користувача про те, що такого користувача не існує, або дані не валідні	Система повідомила користувача про те, що такого користувача не існує, або дані не валідні

4	Пусті поля при реєстрації	Система повідомляє користувача про те, що поля не мають бути пустими	Система повідомлила користувача про те, що поля не мають бути пустими
5	Створення тестування з пустими полями	Система повідомляє користувача про те, що поля не мають бути пустими та їх потрібно заповнити	Система повідомлила користувача про те, що поля не мають бути пустими та їх потрібно 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.

3.5. Висновки до розділу 3

Першим етапом проектування системи є аналіз варіантів використання системи.

Наступним кроком проектування є створення діаграми класів системи, яка, фактично, є декомпозицією діаграми пакетів на більш мілкі складові.

Діаграма класів повністю відображає внутрішню структуру системи.

Графічним інтерфейсом користувача називається система засобів для взаємодії користувача з електронними пристроями, заснована на представленні всіх доступних користувачеві системних об'єктів і функцій у вигляді графічних компонентів екрану піктограм, меню, кнопок, списків тощо.

Найчастіше елементи інтерфейсу в GUI реалізовані на основі метафор та відображають їх призначення та властивості, що полегшує розуміння та використання електронних пристроїв непідготовленими користувачами.

Графічний інтерфейс користувача є частиною інтерфейсу користувача і визначає взаємодію з користувачем на рівні візуалізованої інформації.

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.

ВИСНОВКИ

У результаті виконання дипломної роботи було розроблено та запроектовано програмне забезпечення для проведення всебічного тестування користувачів, у форматі веб-додатку для зручного користування.

Проведення тестування наразі являється невід'ємною частиною навчання та поглиблення знань, і як наслідок це допоможе користувачам краще розібратись у вивченому матеріалі, знайти свої слабкі сторони та пропрацювати їх.

Для виконання поставленої мети було виконано наступні завдання:

1. Під час роботи над дипломним проектом, були досліджені технічні засоби, необхідні для створення якісного веб-додатку для перевірки та оцінювання знань користувачів. Було вирішено обрати мову програмування Java разом з Spring Framework як основні інструменти.
2. Було проведено аналіз інструментів та програмних засобів реалізації, які найліпше впораються зі створенням серверної та клієнтської частини додатку. У ході даного аналізу було обрано мову програмування Java як основну, JetBrains IntelliJ IDEA, як середовище розробки та MySQL Workbench для управління базою даних серверу.
3. Був спроектований та розроблений веб-додаток, досліджені конфлікти та шляхи їх вирішення.
4. Проведено ручне тестування функціоналу веб-додатку.

При проектуванні архітектури додатку, розроблено моделі, які описують аспекти роботи застосунку з різних сторін:

- архітектура системи;
- логічна схема бази даних;
- модель компонентів сервера;
- модель предметної галузі;
- моделювання процесу тестування;

Завдяки чіткому виконанню завдань, поставлених на початку роботи, в результаті виконання роботи було отримано повноцінну систему, що здатна виконувати закладений в неї функціонал та готова до використання в реальних умовах.

Створений додаток має значні перспективи подальших досліджень. Можна й далі розвити веб-версію, спроектувати десктопний додаток, впровадити можливість створювати й інші типи тестів тощо.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Liam Tung (2020-06-15). "JavaScript creator Eich: My take on 20 years of the world's top programming language". ZDNet.
2. Jay Hoffmann (2019-03-04). "What Does AJAX Even Stand For?". Retrieved 2021-10-18.
3. Multiple (wiki). "Web application framework". Docforge. Archived from the original on 2020-06-20. Retrieved 2010-03-06.
4. Multiple (wiki). "Framework". Docforge. Archived from the original on 2018-10-07. Retrieved 2010-03-06.
5. "What is Web Development? - Definition from Techopedia". Techopedia.com. Retrieved 2018-12-07.
6. "Tim Berners-Lee". www.w3.org. Retrieved 17 November 2021.
7. "The website of the world's first-ever web server". Retrieved 30 August 2008.
8. "Internet, Web, and Other Post-Watergate Concerns". University of Chicago. Retrieved 18 September 2010.
9. "OpenGL ES for the Web". khronos.org. 19 July 2011. Retrieved 1 April 2019.
10. Pete LePage. "Responsive Web Design Basics | Web". Google Developers. Retrieved 13 March 2017.
11. "Web Server Survey". Netcraft.
12. A total number of Websites | Internet live stats. internetlivestats.com. Retrieved on 14 April 2015.
13. "Internet 2009 in numbers". Pingdom.
14. "Number of internet users worldwide". Statista.
15. ^ Object-oriented Programming with Java: Essentials and Applications. Tata McGraw-Hill Education. p. 34.
16. ^ "Why Java™ Was – Not – Standardized Twice" (PDF). Archived (PDF) from the original on January 13, 2014. Retrieved June 3, 2018.

17.^ "Java Community Process website". Jcp.org. May 24, 2010. Archived from the original on August 8, 2006. Retrieved June 9, 2010.

18.^ "Sun's Evolving Role as Java Evangelist". O'Reilly Media. Archived from the original on September 15, 2010. Retrieved August 2, 2009.

19.^ "Learn About Java Technology". Oracle. Archived from the original on November 24, 2011. Retrieved November 21, 2011.

20.^ Chander, Sharat. "Introducing Java SE 11". oracle.com. Archived from the original on September 26, 2018. Retrieved September 26, 2018.

21.^ "Java Card Overview". Oracle Technology Network. Oracle. Archived from the original on January 7, 2015. Retrieved December 18, 2014.

22.^ "Java SE". Oracle Technology Network. Oracle. Archived from the original on December 24, 2014. Retrieved December 18, 2014.

23.^ "Java Platform, Enterprise Edition (Java EE)". Oracle Technology Network. Oracle. Archived from the original on December 17, 2014.

24.^ "Deep Dive Into the New Java JIT Compiler - Graal | Baeldung". www.baeldung.com.

25.^ "NullPointerException". Oracle. Archived from the original on May 6, 2014.

26.^ "Exceptions in Java". Artima.com. Archived from the original on January 21, 2009.

27."HTML5 specification finalized, squabbling over specs continues". Ars Technica. 2014-10-29. Retrieved 2014-10-29.

28.^ "XHTM 2.0". World Wide Web Consortium. July 26, 2006. Retrieved November 16, 2008.

29.^ "XHTML 2 Working Group Expected to Stop Work End of 2009, W3C to Increase Resources on HTML5". World Wide Web Consortium. July 17, 2009. Retrieved November 16, 2008.

30."HTML Elements". w3schools.

31. "CSS Introduction". W3schools.

32.^ "XHTML 1.0 – Differences with HTML 4". World Wide Web Consortium. Retrieved November 16, 2008.

33.^ "The Unicode Standard: A Technical Introduction".

34.^ "HTML: The Markup Language (an HTML language reference)"

35.^ Nigel Shadbolt, Wendy Hall and Tim Berners-Lee (2006). "The Semantic Web Revisited" (PDF). IEEE Intelligent Systems. Retrieved October 2, 2009.

ДОДАТОК А

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА ВЕБ-СЕРВІСУ ДЛЯ ПЕРЕВІРКИ І ОЦІНЮВАННЯ ЗНАНЬ МОВОЮ JAVA

Виконав студент(ка) 4 курсу
групи ПД-41
Митяй Олег Володимирович

Керівник роботи
К.т.н., доцент Негоденко Олена Василівна

Київ – 2022

Порівняльна таблиця існуючих систем перевірки та оцінювання знань

Система	Тип	Особливість	Види тестів	Ціна
1. SunRav Web Class	Коробкова версія	Майданчик для зберігання готових тестів. Щоб створювати завдання, знадобиться окремий конструктор tMaker	Варіанти «правильно - неправильно», «бал від 1 до 10»	від 13000 гривень за ліцензію
2. iSpring	Хмарна версія	Платформа тестування та дистанційного навчання для бізнесу. Автоматизує атестацію навчання та допомагає перевести очні тренінги до онлайн-формату	Можна збирати опитування, психологічні тести та тести на перевірку знань. 14 типів завдань: відповідність, вибір одного або декількох варіантів відповіді, вибір області, drag-and-drop, послідовність.	від 38 гривень на місяць за користувача
3. StartExam	Хмарна версія	Є метод 360 градусів	Опитування та перевірочні тести з 9 типів завдань: єдиний та множинний вибір, сортування, відповідність, текстове введення, есе, шкала Лікерта, відеоінтерв'ю та оцінка 360 градусів.	від 2790 гривень на місяць

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Метою дослідження підвищення ефективності перевірки та оцінювання знань за допомогою веб-додатку.

Предметом дослідження є технології для розробки веб-додатку для перевірки та оцінювання знань .

Об'єктом дослідження є процес автоматизації створення та проведення тестування .

3

ТЕХНІЧНЕ ЗАВДАННЯ

1. Проаналізувати технічні засоби, які використовуються для розробки та обрати необхідні для створення веб-додатку
2. Розробити вимоги до застосунку на основі аналізу переваг та недоліків існуючих рішень .
3. Спроектувати та розробити веб-додаток на основі аналізу потреб користувачів
4. Провести тестування веб-додатку

4

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

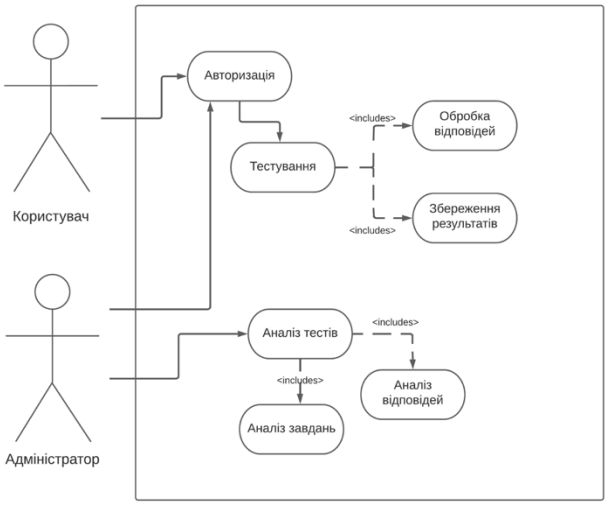
Java

Spring Framework

MySQL Database

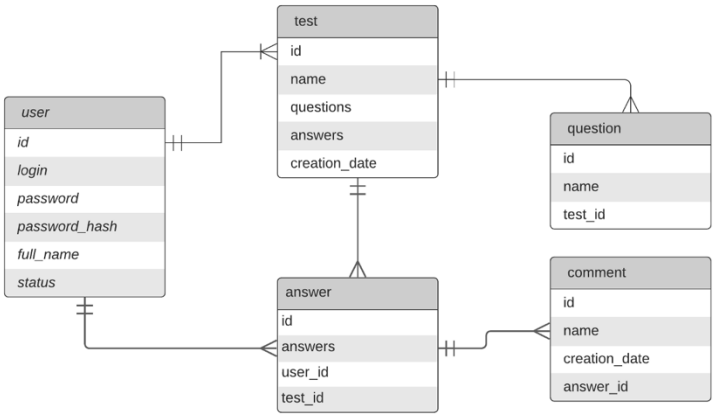
5

Модель прецедентів



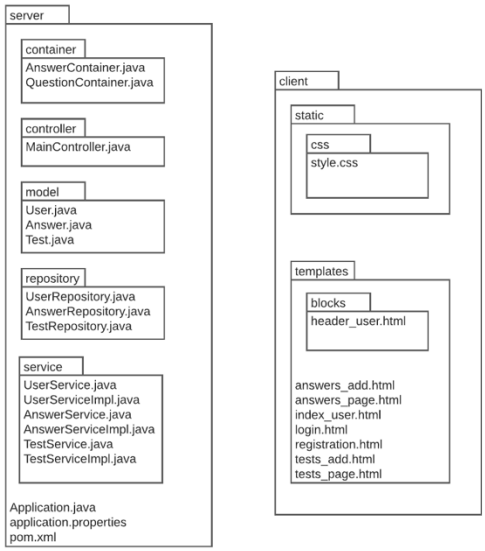
6

Логічна схема бази даних



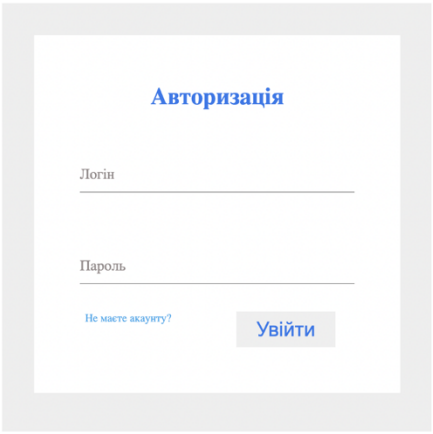
7

Архітектура системи

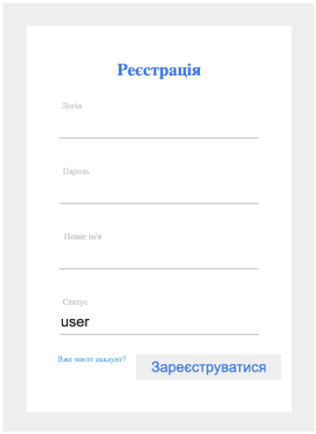


8

ЕКРАННІ ФОРМИ



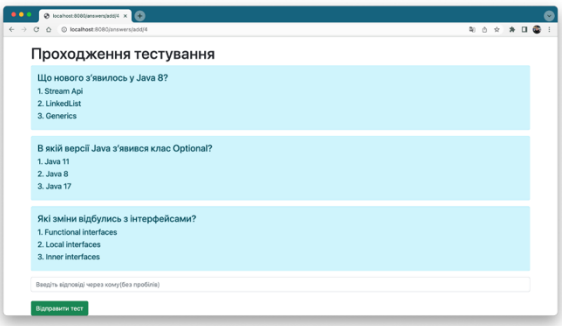
Сторінка авторизації



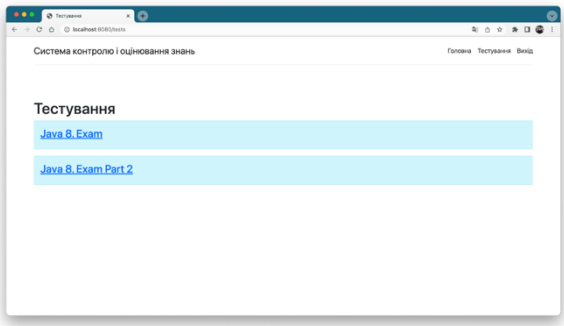
Сторінка реєстрації

9

ЕКРАННІ ФОРМИ



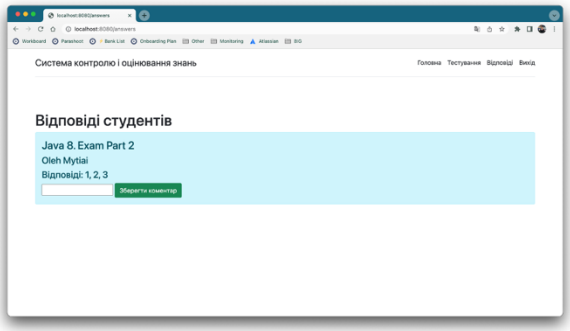
Проходження тестування



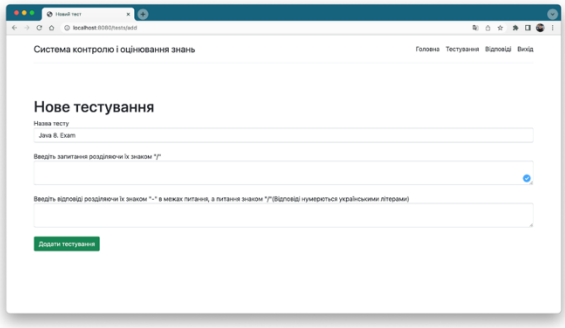
Сторінка з тестами

10

ЕКРАННІ ФОРМИ



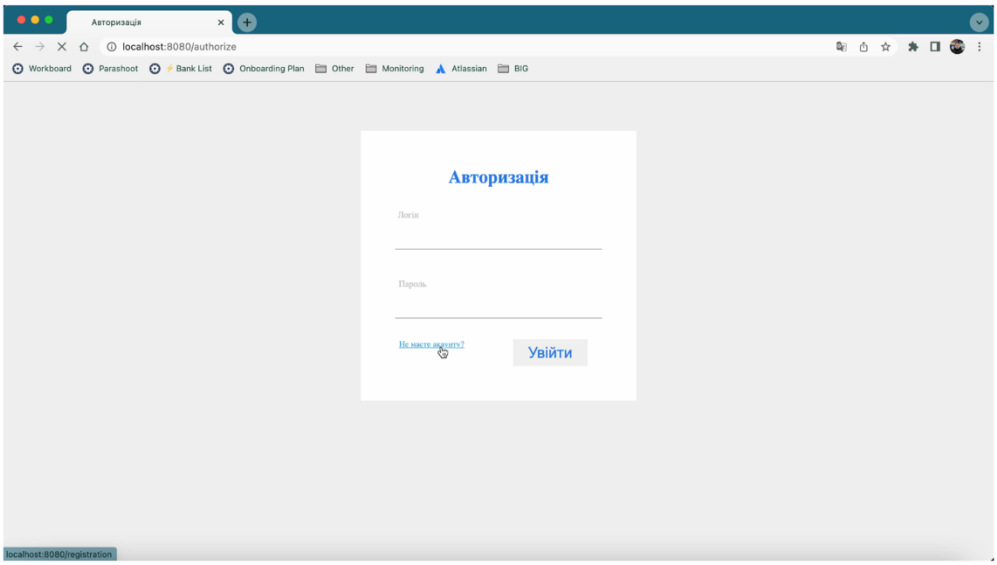
Відповіді студентів



Створення нового тесту

11

Відео



12

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОЛІДЖЕННЯ

1. Митяй О.В. Розробка веб-додатку для перевірки та оцінювання знань / Наука сьогодення: від досліджень до стратегічних рішень : матеріали наук.-тех. конф., м. Івано-Франківськ, 17 червня 2022 р. прийнято до друку / Державний університет телекомунікацій, кафедра інженерії програмного забезпечення, Київ, 2022

2. Митяй О.В. Роль ІКТ у викладанні та вивченні англійської мови / Наука сьогодення: від досліджень до стратегічних рішень : матеріали наук.-тех. конф., м. Івано-Франківськ, 17 червня 2022 р. прийнято до друку / Державний університет телекомунікацій, кафедра інженерії програмного забезпечення, Київ, 2022

Висновки

Для виконання поставленої мети було виконано наступні завдання:

1. Зроблено огляд на тему перевірки та оцінювання знань та програмних засобів для реалізації веб-додатку
2. Запроектовано базу даних MySQL та змодельовано наступні діаграми: Діаграма варіантів використання, Логічна схема бази даних, Архітектура системи
3. Розроблено серверну частину веб-додатку з використанням Spring Framework, клієнтську частину з використанням HTML, CSS та Bootstrap та базою даних MySQL
4. Проведене ручне тестування системи (manual testing)

Дякую за увагу

ДОДАТОК Б

model/User.java

```
package com.example.topeople.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "users")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String login;
    private String password;
    private String fullname;
    private String status;

    public User() {
    }

    public User(String login, String password, String fullname, String status) {
        this.login = login;
        this.password = password;
        this.fullname = fullname;
        this.status = status;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long user_id) {
        this.id = user_id;
    }

    public String getLogin() {
        return login;
    }

    public void setLogin(String user_login) {
        this.login = user_login;
    }

    public String getPassword() {
        return password;
    }
```

```
    public void setPassword(String user_password) {
        this.password = user_password;
    }

    public String getFullname() {
        return fullname;
    }

    public void setFullname(String user_fullname) {
        this.fullname = user_fullname;
    }

}
```

model/Test.java

```
package com.example.topeople.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Test {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    public Long id;
    public String name;
    @Column(length=10485760)
    public String questions;
    @Column(length=10485760)
    public String answers;

    public Test() {
    }

    public Test(String name, String questions, String answers) {
        this.name = name;
        this.questions = questions;
        this.answers = answers;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long test_id) {
        this.id = test_id;
    }

    public String getName() {
        return name;
    }

    public void setName(String test_name) {
        this.name = test_name;
```

```
    }

    public String getQuestions() {
        return questions;
    }

    public void setQuestions(String questions) {
        this.questions = questions;
    }

    public String getAnswers() {
        return answers;
    }

    public void setAnswers(String answers) {
        this.answers = answers;
    }
}
```

model/Answer.java

```
package com.example.topeople.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Answer {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private int user_id;
    private int test_id;
    @Column(length=10485760)
    private String answers;
    @Column(length=10485760)
    private String teacher_comment;


    public Answer() {
    }

    public Answer(int user_id, int test_id, String answers, String
teacher_comment) {
        this.user_id = user_id;
        this.answers = answers;
        this.test_id = test_id;
        this.teacher_comment = teacher_comment;
    }

    public int getTest_id() {
        return test_id;
    }

    public void setTest_id(int test_id) {
        this.test_id = test_id;
```

```
    }

    public Long getId() {
        return id;
    }

    public void setId(Long answer_id) {
        this.id = answer_id;
    }

    public int getUser_id() {
        return user_id;
    }

    public void setUser_id(int user_id) {
        this.user_id = user_id;
    }

    public String getAnswers() {
        return answers;
    }

    public void setAnswers(String answers) {
        this.answers = answers;
    }

    public String getTeacher_comment() {
        return teacher_comment;
    }

    public void setTeacher_comment(String teacher_comment) {
        this.teacher_comment = teacher_comment;
    }
}
```

service/AnswerService.java

```
package com.example.topeople.service;

import com.example.topeople.model.Answer;

public interface AnswerService {
    Answer save (Answer answer);

    Answer findById(Long id);

    Iterable<Answer> findAll();
}
```

service/AnswerServiceImpl.java

```
package com.example.topeople.service;

import com.example.topeople.model.Answer;
import com.example.topeople.repository.AnswerRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;

@Service
```

```
@RequiredArgsConstructor
public class AnswerServiceImpl implements AnswerService {
    private final AnswerRepository answerRepository;
    @Override
    public Answer save(Answer answer) {
        return answerRepository.save(answer);
    }

    @Override
    public Answer findById(Long id) {
        return answerRepository.findById(id).get();
    }

    @Override
    public Iterable<Answer> findAll() {
        return answerRepository.findAll();
    }
}
```

service/TestService.java

```
package com.example.topeople.service;

import com.example.topeople.model.Test;
import java.util.Optional;

public interface TestService {
    Test save (Test test);

    Optional<Test> findById(Long id);

    Iterable<Test> findAll();
}
```

service/TestServiceImpl.java

```
package com.example.topeople.service;

import com.example.topeople.model.Test;
import com.example.topeople.repository.TestRepository;
import com.example.topeople.repository.UserRepository;
import java.util.Optional;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;

@Service
@RequiredArgsConstructor
public class TestServiceImpl implements TestService {
    private final TestRepository testRepository;

    @Override
    public Test save(Test user) {
        return testRepository.save(user);
    }

    @Override
    public Optional<Test> findById(Long id) {
        return testRepository.findById(id);
    }
```

```
    @Override
    public Iterable<Test> findAll() {
        return testRepository.findAll();
    }
}
```

service/UserService.java

```
package com.example.topeople.service;

import com.example.topeople.model.User;
import java.util.Optional;

public interface UserService {
    User save (User user);

    Optional<User> findById(Long id);

    Iterable<User> findAll();
}
```

service/UserServiceImpl.java

```
package com.example.topeople.service;

import com.example.topeople.model.User;
import com.example.topeople.repository.UserRepository;
import java.util.Optional;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;

@Service
@RequiredArgsConstructor
public class UserServiceImpl implements UserService {
    private final UserRepository userRepository;

    @Override
    public User save(User user) {
        return userRepository.save(user);
    }

    @Override
    public Optional<User> findById(Long id) {
        return userRepository.findById(id);
    }

    @Override
    public Iterable<User> findAll() {
        return userRepository.findAll();
    }
}
```

container/AnswerContainer.java

```
package com.example.topeople.container;

public class AnswerContainer {
    public Long id;
    public String user_full_name;
    public String test_name;
    public String answers;
    public String teacher_comment = "";
}
```

container/QuestionContainer.java

```
package com.example.topeople.container;

import java.util.ArrayList;
import java.util.List;

public class QuestionContainer {
    public String question = "";
    public List<String> answers = new ArrayList<>();
}
```

controller/MainController.java

```
package com.example.topeople.controller;

import com.example.topeople.container.AnswerContainer;
import com.example.topeople.container.QuestionContainer;
import com.example.topeople.model.Answer;
import com.example.topeople.model.Test;
import com.example.topeople.model.User;
import com.example.topeople.service.AnswerService;
import com.example.topeople.service.TestService;
import com.example.topeople.service.UserService;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Objects;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
@RequiredArgsConstructor
public class MainController {

    private Boolean authorized = false;
    private User current_user;
    private final UserService userService;
    private final TestService testService;
    private final AnswerService answerService;
```

```
    @GetMapping("/exit")
    public String exit(Model model) {
        current_user = null;
        authorized = false;
        return "redirect:/";
    }

    @GetMapping("/")
    public String index(Model model) {
        model.addAttribute("title", "Головна");
        if (authorized) {
            model.addAttribute("fullname", current_user.getFullname());
            model.addAttribute("currUser", current_user);
            return "index_user";
        } else {
            return "redirect:/login";
        }
    }

    @GetMapping("/login")
    public String start(Model model) {

        model.addAttribute("title", "Авторизація");
        if (!authorized) {
            return "login";
        } else {
            return "redirect:/";
        }
    }

    @PostMapping("/login")
    public String authorize(Model model, @RequestParam String login,
@RequestParam String password) {
        Iterable<User> users_it = userService.findAll();
        List<User> users = new ArrayList<>();
        users_it.forEach(users::add);
        for (int i = 0; i < users.size(); i++) {
            if (users.get(i).getLogin().equals(login) &&
users.get(i).getPassword()
                    .equals(password)) {
                authorized = true;
                current_user = users.get(i);
                break;
            }
        }
        return "redirect:/";
    }

    @GetMapping("/registration")
    public String registration(Model model) {
        model.addAttribute("title", "Реєстрація");
        if (!authorized) {
            return "registration";
        } else {
            return "redirect:/";
        }
    }

    @PostMapping("/registration")
    public String registration(Model model, @RequestParam String login,
@RequestParam String password,
```

```
                                @RequestParam String fullname, @RequestParam
String status) {
        User newUser = new User(login, password, fullname, status);
        userService.save(newUser);
        return "redirect:/login";
    }

    @GetMapping("/tests")
    public String tests(Model model) {

        model.addAttribute("title", "Тестування");

        Iterable<Test> tests_it = testService.findAll();
        List<Test> tests = new ArrayList<>();
        tests_it.forEach(tests::add);
        model.addAttribute("tests", tests);
        model.addAttribute("currUser", current_user);
        return "tests_page";
    }

    @GetMapping("/tests/add")
    public String testsAdd(Model model) {

        model.addAttribute("title", "Новий тест");
        model.addAttribute("currUser", current_user);
        return "tests_add";
    }

    @PostMapping("/tests/add")
    public String testsAdd_Post(Model model, @RequestParam String test_name,
@RequestParam String questions,
                                @RequestParam String answers) {
        Test man = new Test(test_name, questions, answers);
        testService.save(man);
        return "redirect:/tests";
    }

    @GetMapping("/answers")
    public String pairs(Model model) {
        Iterable<Answer> answers_it = answerService.findAll();
        List<Answer> answers = new ArrayList<>();
        answers_it.forEach(answers::add);
        List<AnswerContainer> cnt = new ArrayList<>();
        for (int i = 0; i < answers.size(); i++) {
            if (current_user.getStatus().equals("admin")) {
                AnswerContainer tmp = new AnswerContainer();
                tmp.answers = answers.get(i).getAnswers();
                tmp.teacher_comment = answers.get(i).getTeacher_comment();
                tmp.test_name = Objects.requireNonNull(testService.findById(
                    (long) answers.get(i).getTest_id()).orElse(null)).getName();
                tmp.user_full_name =
Objects.requireNonNull(userService.findById(
                    (long)
answers.get(i).getUser_id()).orElse(null)).getFullname();
                cnt.add(tmp);
            } else {
                if (answers.get(i).getUser_id() == current_user.getId()) {
                    AnswerContainer tmp = new AnswerContainer();
                    tmp.answers = answers.get(i).getAnswers();
                    tmp.id = answers.get(i).getId();
                    tmp.teacher_comment = answers.get(i).getTeacher_comment();
```

```
                    tmp.test_name = Objects.requireNonNull(
                            testService.findById((long)
answers.get(i).getTest_id()).orElse(null))
                        .getName();
                    tmp.user_full_name =
Objects.requireNonNull(userService.findById((long) answers.get(i).getUser_id())
                            .orElse(null))
                        .getFullname();
                    cnt.add(tmp);
                }
            }
        }
        model.addAttribute("answers", cnt);
        model.addAttribute("currUser", current_user);
        return "answers_page";
    }

    @GetMapping("/answers/add/{id}")
    public String answersAdd(Model model, @PathVariable Long id) {
        Test test = testService.findById(id).orElse(null);
        List<String> questions_test = Arrays.asList(test.questions.split("/"));
        List<String> answers_test = Arrays.asList(test.answers.split("/"));
        List<QuestionContainer> cnt = new ArrayList<>();
        for (int i = 0; i < questions_test.size(); i++) {
            QuestionContainer tmp = new QuestionContainer();
            tmp.question = questions_test.get(i);
            tmp.answers = Arrays.asList(answers_test.get(i).split("-"));
            cnt.add(tmp);
        }
        model.addAttribute("questions", cnt);
        model.addAttribute("currUser", current_user);
        return "answers_add";
    }

    @PostMapping("/answers/add/{id}")
    public String answersAdd_Post(Model model, @PathVariable Long id,
@RequestParam String answers) {

        Answer answer1 = new Answer(Math.toIntExact(current_user.getId()),
Math.toIntExact(id), answers, "");
        answerService.save(answer1);
        return "redirect:/tests";
    }

    @GetMapping("answers/{id}")
    public String setComment(@PathVariable Long id,
                             @RequestParam String comment) {
        Answer ans = answerService.findById(id);
        ans.setTeacher_comment(comment);
        answerService.save(ans);
        return "redirect:/answers";
    }
}
```

templates/answers_add.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title th:text="${title}"/>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">
</head>
<body>
<div class="container py-3">
    <header th:insert="blocks/header_user :: header_user"></header>
</div>
<div class="container mt-5">
    <h1>Проходження тестування</h1>
    <div th:each="el: ${questions}"  class="alert alert-info mt2">
        <h4 th:text="${el.question}"/>
        <div th:each="ans: ${el.answers}">
            <h5 th:text="${ans}"/>
        </div>
    </div>
    <form method="post">
        <input type="text" name = "answers" placeholder="Введіть відповіді через
кому(без пробілів)" class="form-control">
        <br>
        <button class="btn btn-success" type="submit">Відправити тест</button>
    </form>
</div>
</body>
</html>
```

templates/answers_page.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title th:text="${title}"/>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">
</head>
<body>
<div class="container py-3">
    <header th:insert="blocks/header_user :: header_user"></header>
</div>
<div class="container mt-5">
    <h1>Відповіді студентів</h1>
    <div th:each="el: ${answers}" class="alert alert-info mt2">
        <h3 th:text = "${el.test_name}"/>
        <h4 th:text = "${el.user_full_name}"/>
        <h4 th:text = "${'Відповіді: '+el.answers}"/>
        <div th:if="${currUser.status == 'admin'}">
            <form th:action="${'answers/' + el.id}">
                <input type="text" name = "comment"
th:value="${el.teacher_comment}">
                <button type="submit" class="btn btn-success">Зберегти
коментар</button>
            </form>
        </div>
        <div th:if="${currUser.status != 'admin'}">
```

```
            <div th:if="${el.teacher_comment != null}">
                <h4 th:text = "${'Коментар: '+el.teacher_comment}"/>
            </div>
        </div>
    </div>
    <br>
</div>
</body>
</html>
```

templates/index_user.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title th:text="${title}"/>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">
</head>
<body>
<div class="container py-3">
    <header th:insert="blocks/header_user :: header_user"></header>
</div>
<div class="container mt-5">
    <div  class="alert alert-info mt2">
        <h3 th:text = "${fullname} + '!'"/>
        <p>Вітаємо вас у системі контролю і оцінки знань!</p>
        <p>Скористуйтеся меню вгорі сторінки для навігації</p>
    </div>
    <br>
</div>
</body>
</html>
```

templates/login.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title th:text="${title}"/>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <link rel="stylesheet" href="/css/start.css">
</head>
<body>
<div id="container">
    <div class="log">
        <h2>Авторизація</h2>
        <form method="post">
            <div class="input-cont">
                <input type="text" name="login">
                <label>Логін</label>
                <div class="border1"></div>
            </div>
            <div class="input-cont">
                <input type="password" name="password">
                <label>Пароль</label>
                <div class="border2"></div>
```

```
            </div>

            <a href="/registration">Не маєте акаунту?</a>
            <input type="submit" value="Увійти">
        </form>
    </div>
</div>
</body>
</html>
```

templates/registration.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title th:text="${title}"/>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <link rel="stylesheet" href="/css/start.css">
</head>
<body>
<div id="container">
    <div class="log">
        <h2>Реєстрація</h2>
        <form method="post">
            <div class="input-cont">
                <input type="text" name="login">
                <label>Логін</label>
                <div class="border1"></div>
            </div>
            <div class="input-cont">
                <input type="password" name="password">
                <label>Пароль</label>
                <div class="border2"></div>
            </div>
            <div class="input-cont">
                <input type="text" name="fullname">
                <label>Повне ім'я</label>
                <div class="border1"></div>
            </div>
            <div class="input-cont">
                <input type="text" name="status" value="user">
                <label>Статус</label>
                <div class="border1"></div>
            </div>
            <a href="/authorize">Вже маєте аккаунт?</a>
            <input type="submit" value="Зареєструватися">
        </form>
    </div>
</div>
</body>
</html>
```

templates/tests_add.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title th:text="${title}"/>
```

```
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">
</head>
<body>
<div class="container py-3">
    <header th:insert="blocks/header_user :: header_user"></header>
</div>
<div class="container mt-5">
    <h1>Нове тестування</h1>
    <form method="post">
        <label>Назва тесту</label>
        <input type="text" name = "test_name" class="form-control">
        <br>
        <label>Введіть запитання розділяючи їх знаком "/"</label>
        <textarea name = "questions" class="form-control"></textarea>
        <br>
        <label>Введіть відповіді розділяючи їх знаком "-" в межах питання, а
питання знаком "/"(Відповіді нумеруються українськими літерами)</label>
        <textarea name = "answers" class="form-control"></textarea>
        <br>
        <button class="btn btn-success" type="submit">Додати тестування</button>
    </form>
</div>
</body>
</html>
```

templates/tests_page.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title th:text="${title}"/>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">
</head>
<body>
<div class="container py-3">
    <header th:insert="blocks/header_user :: header_user"></header>
</div>
<div class="container mt-5">
    <h1>Тестування</h1>
    <div th:each="el: ${tests}" class="alert alert-info mt2">
        <a th:href="${'answers/add/' + el.id}"><h3 th:text = "${el.name}"/></a>
    </div>
    <br>
    <a th:if="${currUser.status == 'admin'}" href="/tests/add" class="btn btn-
warning">Додати тест</a>
</div>
</body>
</html>
```

templates/block/header_user.html

```
<div th:fragment="header_user">
    <div class="d-flex flex-column flex-md-row align-items-center pb-3 mb-4
border-bottom">
        <a href="/" class="d-flex align-items-center text-dark text-decoration-
none">
            <span class="fs-4">Система контролю і оцінювання знань</span>
        </a>
        <nav class="d-inline-flex mt-2 mt-md-0 ms-md-auto">
            <a class="me-3 py-2 text-dark text-decoration-none"
href="/">Головна</a>
            <a class="me-3 py-2 text-dark text-decoration-none"
href="/tests">Тестування</a>
            <a class="me-3 py-2 text-dark text-decoration-none"
href="/answers">Відповіді</a>
            <a class="py-2 text-dark text-decoration-none"
href="/exit">Вихід</a>
        </nav>
    </div>
</div>
```

static/css/start.css

```
* {
  box-sizing: border-box;
}
body {
  background-color: #EEE;
}
.clear {
  clear: both;
}
.log {
  width: 400px;
  margin: 5% auto;
  background-color: #ffffff;
  padding: 30px 0;
}
.log h2 {
  text-align: center;
  color: #257aed;
  font-weight: bold;
  font-size: 26px;
  margin-bottom: 50px;
}
.log .input-cont {
  position: relative;
  margin: 0 50px 60px;
}
.log .input-cont:last-of-type {
  margin-bottom: 30px;
}
.log .input-cont input {
  position: relative;
  z-index: 1;
  width: 100%;
  height: 40px;
  outline: none;
```

```
  color: #212121;
  font-size: 22px;
  font-weight: 400;
  background: none;
  border: none;
}
.log .input-cont input:focus {
  outline: none;
}
.log .input-cont label {
  position: absolute;
  color: #948c8c;
  top: 0;
  left: 0;
  line-height: 40px;
  -webkit-transition: .3s;
  -moz-transition: .3s;
    -o-transition: .3s;
       transition: .3s;
}
.log .input-cont input:empty + label {
  margin-top: -30px;
  -webkit-transform: scale(.8);
  -moz-transform: scale(.8);
    -o-transform: scale(.8);
    transform: scale(.8);
  color: #bdbdbd;
}
.log .border1,
.log .border2 {
  position: absolute;
  height: 1px;
  background-color: #9E9E9E;
  left: 0;
  bottom: 0;
  width: 100%;
}
.log .border1::after,
.log .border1::before,
.log .border2::after,
.log .border2::before {
  content: "";
  position: absolute;
  bottom: 0;
  width: 0;
  height: 2px;
  -webkit-transition: .5s;
  -moz-transition: .5s;
    -o-transition: .5s;
       transition: .5s;
}
.log .border1::after,
.log .border2::after {

  right: 50%;
  background-color: #25a6ed;
}
.log .border1::before,
.log .border2::before {
  left: 50%;
  background-color: #25a6ed;
```

```
}
.log .input-cont input:focus ~ .border1::after,
.log .input-cont input:focus ~ .border1::before,
.log .input-cont input:focus ~ .border2::after,
.log .input-cont input:focus ~ .border2::before {
  width: 50%;
}
.log .check,
.log a {
  float: left;
  width: calc(50% - 50px);
  display: block;
  font-size: 12px;
  margin-bottom: 30px;
}
.log .check {
  margin-left: 50px;
}
.log a {
  text-align: right;
  text-decoration: none;
  color: #25a6ed;
}
.log a:hover {
  text-decoration: underline;
  color: #25a6ed;
}
.log form input[type="submit"] {
  display: block;
  margin: 0 auto 20px;
  border: 2px solid transparent;
  padding: 5px 20px;
  font-size: 22px;
  cursor: pointer;
  color: #257aed;
  -webkit-transition: .5s;
  -moz-transition: .5s;
    -o-transition: .5s;
       transition: .5s;
}
.log form input[type="submit"]:focus {
  outline: none;
}
.log form input[type="submit"]:hover {
  border: 2px solid #25a6ed;
}
```

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
```

```
    <version>2.7.0</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.example</groupId>
  <artifactId>to-people</artifactId>
  <version>0.0.1-SNAPSHOT</version>
<!--  <packaging>war</packaging>-->
  <name>to-people</name>
  <description>to-people</description>
  <properties>
    <java.version>17</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
      <optional>true</optional>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-tomcat</artifactId>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>com.h2database</groupId>
      <artifactId>h2</artifactId>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <optional>true</optional>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
        <configuration>
          <excludes>
            <exclude>
```

```
                <groupId>org.projectlombok</groupId>
                <artifactId>lombok</artifactId>
            </exclude>
          </excludes>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```