Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська кваліфікаційна робота на тему:

РОЗРОБКА WEB-РЕСУРСУ КОМЕРЦІЙНОГО БАНКУ

Виконав: студент (4) курсу, групи 1КН–21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Стукаленко Р.В.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф.КН
Денисюк В.О.
(прізвище та ініціали)

«04» червня 2025 р.

Рецензент: к.т.н, професор каф. АІІТ
Папінов В.М,
(прізвище та ініціали)

«05» червня 2025 р.

Допущено до захисту
Зав. кафедри Яровий А.А.
«06» червня 2025 р.

Вінниця ВНТУ - 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
«05» травня 2025 р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Стукаленко Роману Валерійовичу

1. Тема роботи: Розробка WEB-ресурсу комерційного банку.
Керівник роботи: Денисюк Валерій Олександрович, к.т.н., доцент
затверджені наказом вищого навчального закладу "20" 03 2025 року №97
2. Строк подання студентом роботи 30 травня 2025р.
3. Вихідні дані до роботи :
об'єктно-орієнтована мова програмування; середовище розробки Visual Studio; зручний інтерфейс; швидке завантаження сторінок; інформаційне наповнення; забезпечення статичного та динамічного наповнення сайту.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
аналіз можливостей HTML; аналіз можливостей каскадних таблиць стилів CSS; аналіз можливостей клієнтської мови програмування JavaScript; постановка завдання; розробка структури сайту; реалізація web-сайтів; розробка інтерфейсу сайту; вибір колірної гами; розробка динамічного контенту; тестування роботи сайту.
5. Перелік графічного матеріалу:
схема алгоритму роботи модуля; діаграми класів та компонентів програмного забезпечення; загальний вигляд інтерфейсу користувача; приклад роботи програми.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Денисюк В.О., к.т.н., доцент		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
	Аналіз предметної області web-ресурсу комерційного банку	05.05.25	07.05.25	
	Постановка задачі	08.05.25	10.05.25	
	Розробка програмного модуля web-ресурсу комерційного банку	11.05.25	13.05.25	
	Обгрунтування засобів розробки web-ресурсу комерційного банку	14.05.25	15.05.25	
	Програмна реалізація web-ресурсу комерційного банку	16.05.25	26.05.25	
	Тестування web-ресурсу комерційного банку	27.05.25	29.05.25	
	Розробка інструкції користувача	30.05.25	01.06.25	
	Оформлення матеріалів до захисту БКР	02.06.25	04.06.25	

Студент _____ Стукаленко Р.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Денисюк В.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 76 сторінок формату А4, на яких є 10 рисунків, список використаних джерел містить 32 найменування.

У роботі розглянуто актуальну задачу розробки програмного модуля, який забезпечує роботу комерційного банку. Даний веб-сайт відповідає основним правилам побудови та наповненню графічною та текстовою інформацією. Статичний контент складається статичними та динамічними зображеннями різного формату, які були розроблені у таких графічних редакторах як: Gimp, Photoshop, Gif Animator. При розробці проекту веб-сайту було пройдено розробку структури, розробку інформаційного забезпечення, розробку графічного забезпечення, розробка анімаційного забезпечення, тестування. Дані етапи торкаються усіх аспектів розробки і дають на виході повноцінний продукт, що відповідає усім нормам та правилам веб-дизайну. Веб-додаток написано мовою Java з використанням таких технологій як Spring MVC, Spring security, Hibernate, log4j. Побудова проекту виконується задопомогою додатку Maven.

Ключові слова: веб-додаток, контент, комерційний банк, Java, Photoshop.

ABSTRACT

The bachelor's qualification work consists of 76 pages of A4 format, on which there are 10 figures, the list of used sources contains 32 names.

The work considers the current task of developing a software module that ensures the operation of a commercial bank. This website complies with the basic rules of construction and filling with graphic and text information. Static content consists of static and dynamic images of various formats, which were developed in such graphic editors as: Gimp, Photoshop, Gif Animator. When developing the website project, the development of the structure, development of information support, development of graphic support, development of animation support, testing were completed. These stages touch on all aspects of development and give the output a full-fledged product that meets all the norms and rules of web design. The web application is written in Java using technologies such as Spring MVC, Spring security, Hibernate, log4j. The project is built using the Maven application.

Keywords: web application, content, commercial bank, Java, Photoshop.

ЗМІСТ

ВСТУП

Актуальність досліджень. Цифрові технології суттєво трансформували управління фінансами для приватних осіб, фрілансерів і компаній, відкривши нову епоху онлайн-банкінгу. Сьогодні інтернет-банкінг значно спростив доступ до фінансових послуг і зробив їх більш зручними та продуктивними. Тепер користувачам достатньо лише кількох дотиків до екрана смартфона, щоб дізнатися баланс рахунку, здійснити переказ, оплатити послугу або навіть вкласти кошти. За останні п’ять років більшість клієнтів почали активно користуватися фінансовими сервісами в цифровому форматі. Згідно з аналітичними прогнозами, глобальний ринок онлайн-банкінгу продовжить зростати. Така динаміка зумовлена передусім зміною очікувань користувачів і розширенням цифрових можливостей бізнесу. У результаті, популярність дистанційного банківського обслуговування постійно зростає, сприяючи модернізації банківських інфраструктур. Саме тому багато фінансових установ активно інвестують у новітні банківські рішення, прагнучи відповідати потребам клієнтів і забезпечити їм зручний доступ до фінансових продуктів. Проте впровадження цифрових сервісів потребує ретельного опрацювання як технічних, так і бізнесових аспектів. Онлайн-банкінг — це інструмент, який дозволяє фінансовим установам надавати послуги через веб-платформи чи мобільні додатки. Завдяки цьому клієнти можуть самостійно контролювати свої кошти, виконувати транзакції, оформлювати кредити або депозити, не виходячи з дому. Розробляючи онлайн-сервіси, банк може автоматизувати багато процесів, використовуючи сучасні IT-рішення для покращення взаємодії з користувачем і підвищення ефективності бізнес-процесів. З вищенаведеного можна зробити висновок, що розробка WEB-ресурсу комерційного банку є актуальною та потребує подальшого дослідження.

Метою дослідження є розширення функціональних можливостей WEB-ресурсу комерційного банку за рахунок покращення етапу реєстрації та створення дружнього інтерфейсу.

Об'єктом дослідження є процеси, які виконуються WEB-ресурсом комерційного банку.

Предметом дослідження є алгоритми та програмне забезпечення, що втілює WEB-ресурс комерційного банку.

Задачі дослідження:

- дослідження вже наявних аналогічних рішень;
- вибір та обґрунтування підходу до вирішення поставленого завдання;
- розробка структури та дизайну WEB-ресурсу для комерційного банку;
- програмне створення WEB-ресурсу банківської установи;
- перевірка функціонування ресурсу та аналіз результатів тестування;
- створення користувацької інструкції з експлуатації WEB-ресурсу.

Апробація роботи.

Робота пройшла апробацію на конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025), 2025р.», доповідь «Розробка сайту комерційного банку» [1].

Публікація: електронні тези конференції «Молодь у науці: дослідження, проблеми, перспективи (МН-2025), 2025р.» [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗРОБКИ WEB-РЕСУРСУ КОМЕРЦІЙНОГО БАНКУ

1.1 Аналіз галузі WEB-ресурсів комерційних банків

Перед розробкою WEB-ресурсу комерційного банку необхідно провести аналіз та дослідження вже наявних аналогічних рішень [2].

Керування своїми фінансами для клієнтів, фрілансерів та компаній докорінно змінено онлайн-банкінгом [3].

Інтернет-банкінг із часом значно полегшив користування банківськими послугами та підвищив їхню ефективність. Тепер достатньо кількох натискань на екрані мобільного пристрою, щоб переглянути стан рахунку, здійснити оплату, переказати кошти або навіть інвестувати.

Протягом останніх п'яти років взаємодія клієнтів з фінансовими послугами значно перейшла в онлайн-формат [4].

Згідно з прогнозами ринку, світовий ринок онлайн-банкінгу, за прогнозами, досягне. Ця тенденція також зумовлена двома основними факторами: розширенням бізнес-можливостей та зміною очікувань клієнтів. Отже, кількість користувачів онлайн-банкінгу продовжуватиме зростати, сприяючи цифровій трансформації банківських систем.

З цих причин багато банків та фінансових установ обирають шлях до майбутнього банківських технологій, пропонуючи ефективні рішення для задоволення потреб своїх клієнтів та розширення їхніх можливостей. Однак на шляху до досягнення цієї мети учасникам фінтех-індустрії слід врахувати багато технічних та комерційних аспектів, перш ніж впроваджувати цю ідею.

Онлайн-банкінг – це спосіб, за допомогою якого банки та фінансові установи можуть надавати віддалені послуги через веб-сайт або додаток. Таким чином, користувачі можуть керувати своїми фінансами, здійснювати платежі та перекази, виконувати різні фінансові операції, а також робити

депозити та оформлювати позики, не відвідуючи банк особисто.

Розробляючи веб-сайт банку, клієнт може автоматизувати фінансові процедури за допомогою провідних програмних технологій, щоб забезпечити простий та швидкий клієнтський досвід і підвищити загальну ефективність бізнесу.

Виходячи з огляду ринку, основними рушійними силами інвестицій у розробку веб-сайтів онлайн-банкінгу також є зростаючий ринок фінтех, бурхливий розвиток індустрії електронної комерції, державні стимули для швидкої цифровізації та споживчий попит на віддалені та багатофункціональні послуги.

Це означає, що сьогодні ви можете не лише автоматизувати свої фінансові та банківські практики для вирішення конкретних бізнес-завдань, але й розробляти власні функції для відкриття додаткових каналів доходу та просування банківських послуг.

Оскільки ринок онлайн-банкінгу величезний, завжди є місце для стартапів, гравців, що розвиваються, та існуючих гравців фінтех-сфери, які можуть розглянути можливість розробки банківських рішень на основі технологічних технологій для задоволення будь-яких потреб.

Розгянемо найпопулярніші тенденції в розробці веб-сайтів для онлайн-банкінгу, які сьогодні змінюють банківську галузь:

- безконтактні платежі,
- машинне навчання,
- біометрія, необанкінг,
- блокчейн.

1. Безконтактні платежі.

Завдяки кредитним карткам із мікрочіпом клієнти не мають клопоту з введенням пароля щоразу, коли він їм потрібен. Сьогодні безконтактні платежі – це спосіб зробити покупки безпечнішими та безготівковими.

Банківські картки, портативні пристрої, смартфони та інші пристрої

можуть використовувати технології радіочастотної ідентифікації (RFID - radio frequency identification) та зв'язку ближнього поля (NFC - near-field communication) для здійснення платіжних транзакцій. У місцях продажу продавці мають технологічно підготовлені платіжні термінали, які зчитують безконтактні дані картки та завершують транзакцію.

2. Машинне навчання.

Машинне навчання. (ML - machine learning) та розумні боти дозволяють банківським рішенням оптимізувати підтримку клієнтів, персоналізувати послуги та прогнозувати їхню поведінку.

Розумні боти можуть вирішувати будь-які запити клієнтів цілодобово, оптимізуючи при цьому операційні витрати.

Алгоритми машинного навчання аналізують дані та поведінку клієнтів, щоб надавати персоналізовані пропозиції та різні продукти в будь-який час.

Однак, застосування алгоритмів машинного навчання також може допомогти керівникам банків помічати зміни в поведінці клієнтів або вказувати на порушення безпеки.

3. Біометрія.

Одним із найважливіших аспектів під час побудови систем онлайн-банкінгу є безпека. Онлайн-банкінг та фінансовий досвід повинні бути надзвичайно безпечними, конфіденційними та послідовними. Для забезпечення найвищого рівня захисту від несанкціонованих транзакцій необхідно інтегрувати біометрію в банківські системи.

Біометричні технології включають датчики відбитків пальців, сканування голосових шаблонів, сканування обличчя та сканери сітківки ока. Один із цих варіантів може стати гідною відповіддю, щоб допомогти користувачам безпечно реєструватися та входити в систему для доступу до банківських послуг.

4. Необанкінг.

Необанкінг (Neobanks) – це чудова нова бізнес-модель, яка може

пропонувати широкий спектр послуг лише дистанційно. Представники Neobank більше зосереджені на місцевому ринку та включають функції, що задовольняють місцевих клієнтів.

Ці рішення можуть допомогти клієнтам керувати фінансовими операціями через месенджери та надавати вигідні фінтех-послуги. Такі банки також можна створювати з нуля без початкової клієнтської бази.

5. Блокчейн.

Учасники фінтех-сектору можуть використовувати технологію блокчейн для захисту та децентралізації фінансових транзакцій. Таким чином, банківська система може виключити багатьох посередників, залучених до кожної транзакції, одночасно пришвидшуючи її проведення. Крім того, прозорість та безпека технології блокчейн дозволяють банкам здійснювати аудит та бухгалтерський облік.

1.2 Аналіз основних тенденцій в розробці веб-сайтів банків

Серед найпопулярніших тенденцій в розробці веб-сайтів банків виділяють [4-6]:

- візуалізація даних;
- власна типографіка;
- послідовність (наступність);
- власна іконографія;
- зручний інтерфейс користувача;
- безпека веб-сайту

1. Візуалізація даних.

Обсяг інформації, з якою користувачі Інтернету стикаються щодня, зростає, і банки шукають методи структурування та організації даних у зручний для використання спосіб.

Візуалізація даних у вигляді графіків, діаграм та інфографіки широко

використовується в Інтернеті.

Найкращі банківські вебсайти також дотримуються ширшої тенденції використання візуалізації даних для навчання клієнтів з важливих банківських питань.

2. Власна типографіка.

Пануюча тенденція до мінімалізму в розробці банківського дизайну вигідна для сайтів з унікальною типографікою, оскільки вони відрізняються від інших.

Типографіка додає сайту креативного шарму. Типографіка також є важливим компонентом стратегії ідентифікації бренду. За умови правильного виконання вона може донести відповідне повідомлення до відвідувачів.

3. Послідовність.

Для ефективного дизайну веб-сайту банку важлива послідовність. Невідповідність у користувацькому інтерфейсі ніколи не буде визнана чи похвалена. Клієнти, яким доводиться з нею стикатися, ймовірно, будуть спантеличені та/або роздратовані.

Єдиний дизайн легко зрозуміти. Він ретельно проводить клієнтів через веб-сайт банку, усуваючи будь-яку невизначеність та створюючи відчуття контролю, комфорту та надійності.

4. Власна іконографія.

Піктограми символізують глобальну мову, яку розуміє кожен. Вони забезпечують візуальний контекст і покращують зручність використання веб-сайту.

Розумним дизайнерам слід уникати стандартних наборів іконок, оскільки вони можуть бути оманливими та ніколи не підкреслять стиль бренду. Власні іконки стають дедалі популярнішими завдяки простоті їх використання.

5. Зручний інтерфейс.

Зручний інтерфейс користувача (інтерфейс користувача без інтерфейсу

користувача). Мета такого дизайнерського підходу — позбутися непотрібних мотивів інтерфейсу користувача. Люди не хочуть витрачати час на прокручування веб-сайту в пошуках потрібної інформації. Вони очікують отримати її, щойно відкриють його.

Отже, застосовувати досвід роботи з клієнтами, щоб розробити ідеальний веб-сайт. Необхідно видалити будь-які зайві шаблони, щоб зробити інтерфейс користувача максимально простим та зрозумілим.

6. Створення безпечного веб-сайту онлайн-банкінгу.

Оскільки користувачі довіряють фінансові дані та особисту інформацію, коли працюють з банківським веб-сайтом, безпека є однією з головних функцій.

Будь-які прогалини в безпеці та потенційні ризики кібератак повинні бути ефективно усунені. В іншому випадку потенційна шкода та порушення безпеки можуть коштувати вашому бізнесу статку або призвести до повного банкрутства банку.

Щоб запобігти цьому сценарію, існують ефективні практики безпеки, які допоможуть банківському веб-сайту забезпечити найвищий рівень безпеки та надійну стійкість до збоїв і втрати даних.

1) Розміщувати банківський веб-сайт у виділеному середовищі, а не у спільному. Спільні середовища хостингу дешевші, але вам доведеться використовувати той самий серверний простір, що й іншим веб-сайтам. Це може погіршити стабільну роботу вашого веб-сайту через залежність від інших веб-сайтів, які можуть використовувати той самий сервер.
2) Забезпечити 256-бітове SSL-шифрування, щоб забезпечити безпеку та конфіденційність даних ваших клієнтів, а також захистити веб-сайт вашого банку від зовнішніх загроз.
3) Зосередитися на запобіганні розподіленим атакам типу «відмова в обслуговуванні» (DDoS), якщо хакери намагатимуться

перевантажити ваш сервер. Ви можете використовувати виділений сервер та інші методи, згадані тут, щоб запобігти цьому.

4) Виконати сканування вразливостей, щоб виявити потенційні місця порушення в певній мережі або групі мереж.
5) Регулярно оновлювати CMS та плагіни для постійної підтримки безпеки.
6) Впроваджувати біометричні дані та двофакторну автентифікацію (2FA) для перевірки входу та авторизації користувачів.
7) Регулярно перевіряти безпеку веб-сайту вашого банку.
8) Переконатися, що центр обробки даних відповідає галузевому стандарту для атестаційних завдань SSAE 16.
9) Регулярно створювати резервні копії веб-сайтів зашифрованим способом.
10) Мати наявний план відновлення після аварій.

Застосування цих методів до розробки веб-сайту банку гарантує першокласну безпеку банківського веб-сайту та може допомогти забезпечити надійне обслуговування клієнтів.

1.3 Архітектури розробки веб-сайтів банків

При створенні веб-сайту для банківської системи, інвестицій недостатньо, щоб допомогти зайняти особливе місце на фінансовому ринку. Ще однією важливою вимогою для кращого обслуговування клієнтів та безпеки найвищого рівня є дизайн цифрової архітектури банку. Існує два основні типи [4, 6]:

- багатошарова архітектура;
- архітектура мікросервісів.

1. Багатошарова архітектура.

У типовій багатошаровій архітектурі є три рівні: бекенд, проміжне

програмне забезпечення та фронтенд.

Серверний рівень включає певні компоненти:

- базова банківська система. базова банківська система банку подібна до двигуна, який керує основними операціями банку, інформація про клієнтів, платіжні операції, видача кредитів та бухгалтерські системи є частинами основної іт-інфраструктури;
- цифровий банкінг, завдяки цьому компоненту всі продукти та послуги банку доступні на рівні фронтенду;
- дані та аналітика, цей компонент включає посилання та транзакційні дані, необхідні для аналізу даних клієнтів з метою прогнозування майбутніх дій;
- приватні та публічні API (application programming interface - прикладни́й програ́мний інтерфе́йс), приватні API включають сервіси для здійснення платежів або отримання балансів рахунків, тоді як публічні API забезпечують безперешкодну інтеграцію зі сторонніми сервісами.

Рівень проміжного програмного забезпечення – це програмне забезпечення, яке з'єднує фронтенд, бекенд та інші бізнес-додатки. Він отримує повідомлення від фронтенд-рівня, інтерпретує їх та передає до відповідної системи у відповідному форматі. Рівень проміжного програмного забезпечення також включає функціональність для автентифікації клієнтів.

Фронтенд-рівень забезпечує взаємодію клієнта з цифровою банківською системою. Технологічні рішення для фронтенду включають мобільні та веб-банківські додатки, чат-боти та програмне забезпечення для портативних пристроїв.

2. Архітектура мікросервісів.

Ще одним популярним типом архітектури цифрового банкінгу є мікросервіси. Це технічне рішення забезпечує більшу гнучкість, безліч інтеграцій зі сторонніми сервісами та можливості масштабованості. Це буде

правильним варіантом, якщо ви плануєте масштабувати своє рішення з точки зору функціональності та клієнтської бази. Архітектура мікросервісів складається з API-шлюзу, мережі сервісів та ядра на основі мікросервісів.

Шлюз API. Через шлюз API мікросервіси доступні зовнішнім каналам, що дозволяє їм взаємодіяти з даними, продуктами та послугами банків.

Сервісна сітка. Сервісна сітка може масштабувати сервіси при пікових навантаженнях та виконувати функції управління навантаженням.

Ядро на основі мікросервісів. Це ядро складається з слабо пов'язаних мікросервісів та невеликих незалежних процесів, які взаємодіють один з одним через API та протоколи.

Отже, вибір архітектури цифрового банкінгу залежить від поточних бізнес-потреб, функціональних вимог та вимог безпеки.

1.4 Особливості проектування веб-сайтів банків

Банківський веб-сайт – це платформа для постійного спілкування між банківським бізнесом та його клієнтами. Прибутковість фінансової структури залежить від надійності та ефективності її роботи.

Розглянемо основні особливості проектування веб-сайту банку, а саме: питання навігації, інтуїтивна зрозумілість та простота у використанні, використання прототипування на основі каркасної конструкції, відповідність інтересам та очікуванням цільової аудиторії [4].

1. Навігація зумовлується завданнями сайту.

Все залежить від головного меню, структури розділів та організації взаємозв'язків. Завдання просування та продажу банківських продуктів повинні підпорядковувати дизайн сайту. Неможливо ставити дизайн на перше місце, якщо від цього постраждає простота та ефективність роботи інтернет-користувачів з веб-ресурсом.

2. Інтуїтивна зрозумілість та простота у використанні.

Вебсайт банку має бути зрозумілим з першого погляду. У сайта є від 10 до 30 секунд, щоб або завоювати потенційного клієнта, або він піде. Тому релевантний налаштовуваний пошук по сайту, структура розділів та інтуїтивно зрозуміле меню навігації на головній сторінці – це основні моменти, з яких слід почати розробку вашого онлайн-банкінгу.

3. Використання прототипування на основі каркасної конструкції.

Прототипи допомагають зробити навігацію інтуїтивно зрозумілою, видалити зайві елементи та відстежити основні шляхи, якими клієнти йдуть на ваш сайт і з нього. Не забувайте про заклики до дії, банери та кнопки, організацію елементів та взаємозв'язок різних розділів сайту один з одним.

4. Відповідність інтересам та очікуванням цільової аудиторії.

Дизайн брендбуку та вебсайту має збігатися – це перше. Навіть якщо у вашому онлайн-банкінгу немає єдиного переліку вимог до дизайну (крім логотипу та кольорів), зробіть його впізнаваним. А найголовніше, майбутній позичальник чи інвестор повинен знайти на сайті саме те, що він очікує там побачити. Він повинен отримати відповіді на ті питання, які у нього (неї) виникають.

Важливим критерієм є те, що сайт має працювати на благо загального іміджу банку, а не бути просто «візитною карткою», щоб додавати її до щоквартальних звітів маркетологів про виконану роботу.

1.5 Аналіз відомих програм аналогів

Аналіз відомих програмних аналогів WEB-ресурсів комерційних банків підтверджує унікальність дизайну кожної розробки, та слідування традиційним вищеназваним вимогам та архітектурам. Орієнтовна вартість розробки становить диапазон від $800 до $40 000 [4, 7-10]. Загальні риси послуг розробників WEB-ресурсу комерційного банку задовільняють

вимогам:

- розробка сайту на зручній та зрозумілій платформі;
- вбудована інтеграція сервісів онлайн-кредитування;
- іміджевий вебресурс для банківської установи;
- опція замовлення брендбуку;
- оптимізація для покращення видимості в пошукових системах;
- всеоб’ємний технічний супровід розробки.

1.6 Постановка задачі

У зв’язку з розвитком Інтернет технологій дедалі частіше державні та недержавні установи звертаються до різноманітних фірм, що займаються веб розробкою із замовленням виготовити для них власний веб-сайт. Сучасні технології дозволяють вести бізнес, взнавати новини або просто переглядати різні сторінки заради задоволення, не виходячи із власного будинку, саме завдяки існуванню інтернета та веб-технологій. Тому всі без виключення бажають стати учасниками інтернет спільноти, щоб завжди бути в курсі усіх подій. Не виключенням є і комерційні банки. Інтернет банкінг є дуже актуальним сьогодні. З розвитком всесвітньої мережі можна проводити банківські операції на виходячи з дому чи перебуваючи на відпочинку далеко за межами країни.

Для того щоб розробка була корисною необхідно визначитися з тим які саме функції повинні бути реалізовані. Оскільки, хоча дана розробка є дуже актуальною, для початку необхідно створити захищену сторінку яка матиме зв’язок з базою даних, яка містить дані про вкладників, кількість та грошей на рахунках.

Відповідно до поставленого завдання необхідно розробити веб додаток для обслуговування. Даний сайт повинен забезпечувати наступні функції:

- створення рахунку для клієнта

- видалення рахунку
- внесення коштів на рахунок
- переведення коштів з одного рахунку на інший
- бізнес логіка, яка б забезпечувала умови контрактів щодо депозитів та кредитів фізичним та юридичним особам.

1.7 Висновок до розділу 1

Створення власного WEB-ресурсу комерційного банку є достатньо актуальною задачею, розв’язання якої надає ряд переваг, а саме: індивідуальний дизайн; наявність тільки необхідних функцій; контроль трафіку; заощадження коштів. У якості завдання обираємо розробку WEB-ресурсу комерційного банку.

2 РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ WEB-РЕСУРСУ КОМЕРЦІЙНОГО БАНКУ

2.1 Етапи проектування WEB-ресурсу

Проектування – це ключовий етап створення ресурсу і відповідає на наступні питання:

- яка мета розробки – навіщо ми робимо сайт;
- як реалізуємо поставлені цілі;
- як сайт буде виглядати і працювати.

Проектування сайту забезпечує дуже багато переваг, тобто суттєво підвищує гарантію досягнення результату. Тільки чітко сформулювавши завдання, визначивши цільову аудиторію сайту та потреби, змоделювавши взаємодію сайту і користувачів, ми можемо бути впевнені у досягненні високих результатів.

Виправити помилку на етапі проектування досить просто: міняємо кілька блоків тексту і схем. Зробити це на етапі розробки дизайну або верстки буде вже дорожче.

Якщо помилка виявляється на етапі програмування, її виправлення може спричинити суттєві фінансові витрати та затрати часу це додаткові місяці, а то й роки роботи.

Проектне завдання – це самодостатній документ. Отримавши його, клієнт може зробити сайт своїми силами або найняти іншу команду, яка, на його думку, краще впорається з безпосередньо розробкою.

Нехтувати проектуванням не бажано навіть при розробці маленьких сайтів; навіть для елементарної візитки це буде виключно корисно.

Інколи етап проектування навмисно пропускають, зазвичай це відбувається через брак коштів.

Проектування можна умовно розбити на чотири основні частини:

- визначення мети;
- дослідження контексту;
- створення концепції;
- моделювання.

Перш за все необхідно визначитися, яка необхідність розробки сайту і яких саме результатів необхідно досягти. Даний етап є орієнтиром для всієї подальшої роботи: що б ми не робили – моделювання, створення інтерфейсів, додавання нового функціоналу або зміна існуючого – все це має відповідати визначеній меті.

У майбутньому вони допомагають оцінити успішність проекту.

Перший крок в цілепокладання зроблено баченням проекту. На етапі ж проектування цілі формулюються більш точно і детально, а також визначаються завдання сайту, виконання яких буде працювати на досягнення кожної мети.

Дослідження необхідно для отримання інформації, яку ми будемо називати контекстом сайту. Під контекстом ми розуміємо різні обставини, що оточують сайт і здатні вплинути на його роботу. До таких обставин належать:

- цільова аудиторія і її потреби;
- характеристика і тенденції області;
- конкуренти і їх діяльність;
- досвід інших проектів;
- законодавчі чи інші обмеження;
- інші фактори впливу, залежно від тематики проекту.

Дослідження контексту.

Контекст проекту допомагає нам зрозуміти цільову аудиторію і те, яким потрібно зробити сайт: як його позиціонувати, який контент на ньому повинна бути і якою мовою він повинен відображатися (це називають

комунікативною стратегією), як він буде відрізнятися від конкурентів (природно, у вигідну сторону).

Крім того, дослідження занурює команду проекту в тему, дозволяє десь навіть на підсвідомому рівні бачити весь процес та приймати вірні рішення.

Найкраще, звичайно, дослідження допомагає зрозуміти аудиторію.

По-перше, як спілкуватися: які слова використовувати в «розмові», як називати свій продукт або послугу (можливо, потрібно використовувати професійний сленг, або навпаки, нарочито спрощену термінологію), чи звертатися на «ви» чи на « ти».

По-друге, які потреби: що пріоритетно, що необов'язково, але бажане, а що не можна робити ні в якому разі.

Джерела інформації. В ідеалі, дані про контекст ми повинні отримати від клієнта, але якщо таких даних у клієнта немає, то дослідження контексту належить провести самим.

Для дослідження необхідно використати два методи, які як найкраще зарекомендували себе на більшості веп проектів:

- дослідження доступних джерел – література, інтернет-ресурси тощо;
- інтерв'ю з ключовими дійовими особами – користувачами та експертами.

Інколи при розробці виникає думка, що самостійне дослідження не має досить високого рівня у порівнянні з професійними маркетинговими дослідженнями – так навіщо витрачати час і ресурси? На те є кілька причин:

- без даних про контекст шанси зробити добре користувачеві зводиться до нуля;
- самостійне дослідження дає корисну інформацію, нехай і в меншому обсязі, ніж професійне;
- навіть якщо ви настільки невміло проводите дослідження, що не можете отримати відповіді на поставлені питання, ви все одно занурюєтесь в контекст проекту і хоча б підсвідомо вловлюєте щось

корисне.

Після визначення основних цілей, отримано дані про контекст. Необхідно трансформувати всю інформацію в концепцію. Під концепцією ми розуміємо основні ідеї та можливості, закладені в проект:

- що і для кого виконується замовлення – загальна ідея і цільова аудиторія;
- як сайт буде працювати і яку інформацію утримувати;
- як сайт буде заробляти (якщо це комерційний проект);
- які будуть відмітні особливості сайту порівняно з конкурентами;
- як він буде позиціонуватися;
- як сайт розвиватиметься після запуску.

Концепція задає напрямок проектування і допомагає, аналогічно баченню, ще раз зістикувати точки зору на проект.

Моделювання – це створення моделі сайту, яка описує функціональні можливості та інформаційну структуру.

У функціональній частині моделі ми описуємо можливості, які сайт надає своїм користувачам: наприклад, викладати, групувати та коментувати фотографії (соціальна мережа) або замовляти і оплачувати товар (інтернет-магазин).

Важливо розуміти, що можливості є інструментами вирішення завдань. Якщо придумана можливість не вирішує жодну з завдань – це може означати, що вона зайва.

У проектному завданні описуємо можливості на досить високому рівні абстракції – не так детально, як ми зробимо це в технічному завданні на програмування, оскільки це просто не потрібно [1, 11].

Інформаційна структура – це схема, що показує, з яких розділів складається сайт, які завдання вони вирішують, і як користувач буде рухатися по сайту (схема навігації).

Затим необхідно опрацювати схему розділу – це більш глибоко і детально пророблена схема (порівняно з інформаційною структурою сайту), що показує навігацію по розділу, зв’язки і переходи між підрозділами. Схема розділу включає такі елементи:

- завдання які з раніше поставлених завдань вирішує розділ; наприклад, розділ «Фотографії» у соціальній мережі вирішує завдання обміну інформацією між друзями і подальшого спілкування;
- повідомлення – це в буквальному сенсі повідомлення, які розділ або його частина передає відвідувачу; наприклад, повідомлення вітання «Привіт, ти на порталі! Ми раді бачити тебе тут» або «Агов, це наш найкращий товар – спробуй, замов зараз і переконайся в цьому сам!»;
- повідомлення бувають різних типів, найчастіше зустрічаються: рекламні, заклики до дії, повідомлення та іміджеві повідомлення.
- функціональні елементи – елементи інтерфейсу, що дають можливість відвідувачеві виконати деяку операцію: наприклад, функціональним елементом є форма для введення повідомлення, що дозволяє відправити повідомлення, або кнопка в інтерфейсі, що зберігає зроблені зміни.
- варіанти поведінки відвідувача – припущення про те, що відвідувач сайту може або повинен зробити після вивчення інтерфейсу або окремих його частин.

2.2 Реалізація WEB-сайтів

Структура сайту – це організація даних, ієрархія матеріалів, необхідна для подальшого подання їх цільової аудиторії. Саме тому, структура сайту повинна бути логічною, практичною і зручною.

Який би не був вражаючий дизайн, як би дорого не виглядав сайт, відвідувачі будуть з нього йти, якщо його структура буде для них не зручною, не зрозумілою і не комфортною.

Процес створення структури виконується в два етапи:

- структуризація інформації , що міститься на сайті;
- візуальне подання самої структури.

Перший етап полягає в аналізі і згрупуванню інформації, яку планується розміщувати на сайті, об’єднання її в різні категорії і рубрики , а також визначення назв для цих категорій. Відразу ж необхідно визначитися, що назва категорій повинні бути в першу чергу зрозумілі потенційним користувачам.

Наступним дуже важливим кроком є розробка ієрархії сторінок сайту. Необхідно виділити категорії першого і другого рівня, при необхідності третій. Кінцева ієрархія розділів представлятиме собою карту сайту.

При розробці структури дуже важливо думати як потенційний відвідувач сайту , виходити з логіки цільової аудиторії. Помилки, допущені на цьому етапі роботи, можуть спричинити за собою серйозні витрати часу і грошових коштів на наступних етапах, так як доведеться переробляти багато роботи, яка була виконана з помилками у зв’язку з самого початку неправильно закладених рішень.

Візуальне представлення структури сайту дозволяє розташувати елементи структури відносно один одного таким чином, що користувач зможе з першого погляду «просканувати» сторінку, зорієнтуватися на ній, інтуїтивно зрозуміти структуру і відповісти на головне питання: куди потрібно клацнути мишкою , щоб знайти те, що потрібно.

Структура сайту – це його скелет. Якщо структура ретельно продумана, логічна і відповідає стандартам юзабіліті, то сайт буде життєздатним. Користувачі будуть приходити і повертатися знову і знову. У подяку за те, що творці сайту зберегли їх час і нерви.

Основні елементи структури сайту – візитки :

- про компанію;
- послуги або товари;
- виконані роботи;
- контакти.

Основні елементи структури інтернет-магазину:

- головна сторінка (з формою реєстрації та авторизації користувачів , з формою розширеного пошуку);
- каталог товарів ділиться на категорії , а категорії – на конкретні товари;
- кошик (для роботи з вибраними товарами і подальшого оформлення замовлення);
- про компанію;
- контактна інформація;
- умови доставки або правила роботи;

Елементи структури комерційного банку:

- головна сторінка з меню;
- сторінка реєстрації;
- персональна сторінка клієнта;
- інформаційна сторінка про заклад.

Варіацій може бути безліч. Головне при розробці структури завжди пам’ятати , що вона повинна бути зручною і логічною для користувача.

Структура сайту включає в себе схему ієрархії сторінок, систему навігації і єдину структуру сторінок.

Кінцевим результатом розробки структури сайту, є так званий прототип сайту – діюча модель, яка не має ні дизайну, ні контенту, однак дає повне уявлення про те, яка буде ієрархія сторінок і як буде виглядати система навігації.

Система навігації – найважливіший елемент сайту, якому при розробці структури приділяється особлива увага. У цю систему включаються три основні компоненти.

1. Базова навігація, що визначає основні шляхи переміщення користувача по сайту. Сюди входить загальне меню, що присутнє на всіх сторінках сайту і посилання на головні сторінки розділів, а також внутрішня навігація кожного розділу і підрозділу.

2. Функціональні елементи – форми реєстрації та авторизації, система контекстного пошуку, форми підписки на розсилки та повідомлення.

3. Додаткові навігаційні елементи для переміщень в межах сторінки або між сусідніми сторінками.

Існує декілька видів структур сайтів: лінійна, лінійна з відхиленнями, деревовидна та решіткова структури [11, 12]. Даний сайт відповідає ґратчастій структурі сайтів з деякими втраченими зв’язками. Зображення класичної ґратчастої структури зображено на рисунку 2.1.

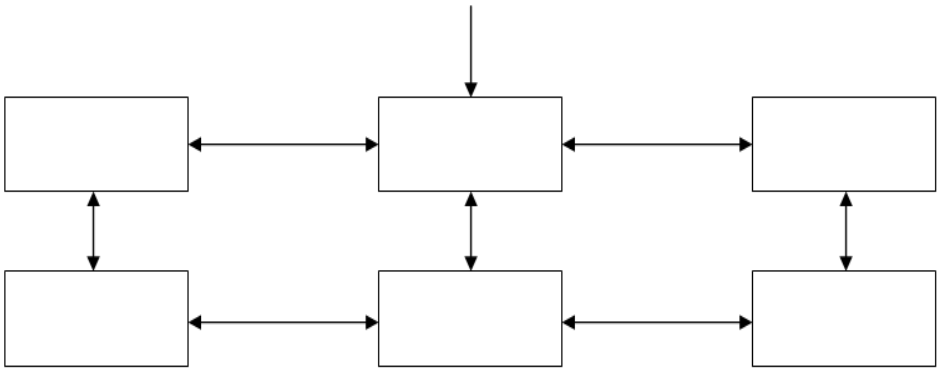

Рисунок 2.1 – Гратчаста структура веб-сайту

2.3 Розробка інтерфейсу сайту

Важливим етапом розробки є візуальний дизайн інтерфейсу. Спершу на необхідно відпрацьовано креативну концепцію. Загальна стилістика схвалена,

На цьому етапі продукт знаходить зовнішній вигляд – до цього ми займалися його суттю і принципами роботи.

Відповідно до теми «Комерційний банк» структура сайту повинна бути такою, щоб користувач міг переглянути його персональну інформацію, його фото, стан рахунків, журнал транзакцій.

Також необхідно розробити структуру так, щоб користувач завжди міг повернутися із будь-якого місця до власної сторінки. Початкова сторінка сайту повинна бути найбільш інформаційною, принципова схема структури зображено на рисунку 2.2.

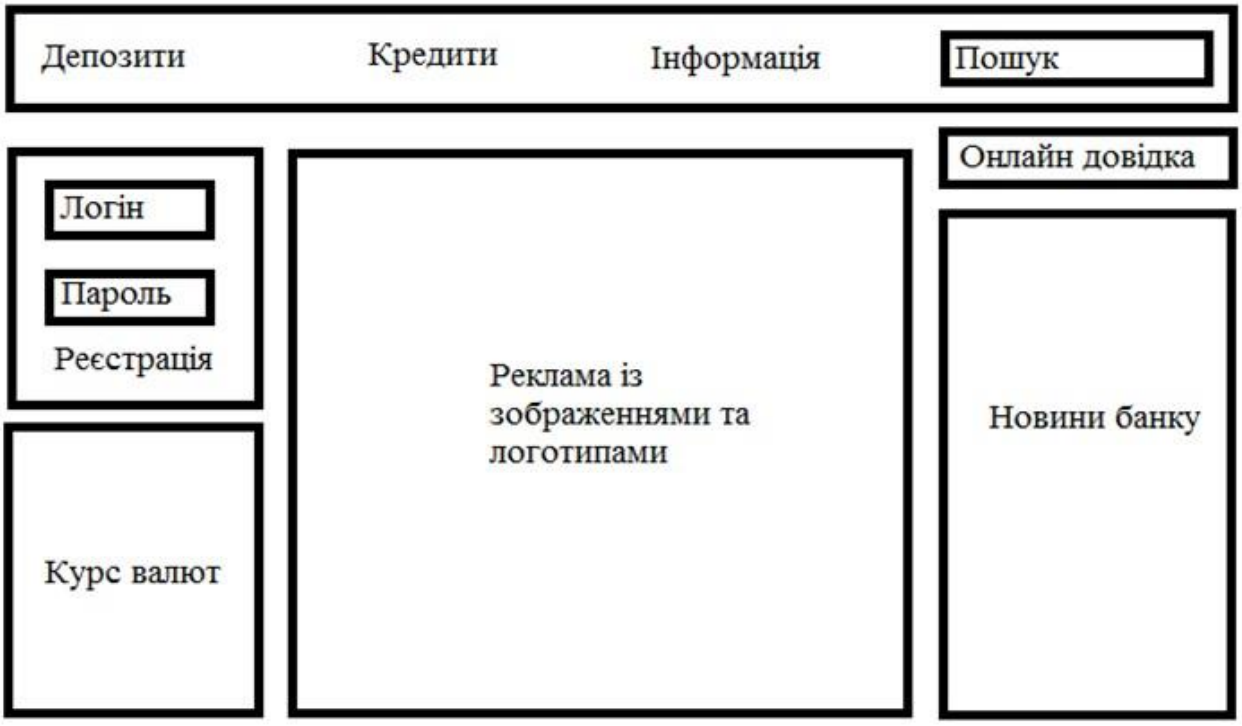

Рисунок 2.2 – Структура головної сторінки сайту

2.4 Вибір колірної гами

Вибір колірної гами на сайті є дуже важливим елементом його дизайну. Кольори, що нас оточують в житті зазвичай мають невелику насиченість і складаються з декількох відтінків, зате вони приємніші для очей, ніж

відтворені чисті кольори. При використанні природних кольорів користувачі мають можливість краще зосередитися на самому сайті і не відволікатися через надто яскраві відтінки.

Відомо, що зосереджуючись на інформації, наші очі повинні бачити тільки цю картину, без сторонніх подразників [13, 14]. Коли на сайті присутні яскраві відтінки кольорів, повністю зосередитись на чомусь одному неможливо.

Однак, існують такі відтінки, наприклад: зелений та помаранчевий, які поєднуються між собою гармонійно. Така комбінація кольорів застосовується для розфарбування панелей навігації та контенту, між якими користувач буде постійно балансувати.

Для підбору кольорів використовуємо коло Йоганнеса Іттена (рис.2.3) [11, 12].

Вибір кольорів відбувається за вершинами вписаних у коло трикутника або шестикутника. Обрані таким чином відтінки кольорів будуть пасувати одне одному.

Необхідно підібрати різні кольори для різних частин сторінки. Найпомітнішим кольором потрібно робити блоки з контентом і меню.

У веб розробках це особливо важливо, адже користувачі довгий час, просиджують за монітором.

Колірна палітра повинна не відволікати користувача під час роботи, а навпаки зобов’язана допомогти зосередитися.

Провідні дизайнери закликають до того, щоб користувач міг переміщати увагу з однієї точки сторінки на іншу легко і вільно. Бажано, щоб при цьому у користувача не виникало неприємних відчуттів.

Кольори повинні бути природними. Але природні кольори – це не панацея, крім цього є ще емоційна частина, яка також буває різною. Смаки користувача змінюються разом з настроєм, тому підібрати кольори, що будуть подобатись абсолютно всім неможливо.

Для розробки даного сайту будуть використовуватись пастельні кольори, що нагадують собою художнє полотно. Здебільшого оливкові кольори.

Для розробки логотипу буде використано відтінки жовтого та блакитного. Блок контенту необхідно зробити білого кольору, для підкреслення його як головної частини сайту.

Рисунок 2.3 – Коло Йоганнеса Іттена

Відповідно до кола Іттена, використовуючи вписаний трикутник та споріднені кольори можна застосувати також червоний але опустимо цей колір, тому що цей колір здебільшого використовується для передачі чогось різкого та агресивного.

2.5 Висновок до розділу 2

Обрано ґратчасту структуру сайта з деякими втраченими зв’язками. Обрано структуру головної сторінки сайту. Обрано кольори для розробки сайту банку.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ WEB-РЕСУРСУ КОМЕРЦІЙНОГО БАНКУ

3.1 Аналіз можливостей HTML

HTML – мова гіпертекстової розмітки. Стандартна мова розмітки документів у мережі. Більшість веб-сторінок містять опис розмітки на мові HTML (або XHTML), що інтерпретується браузерами і відображається у вигляді документа в зручній для сприйняття людиною формі [15-16].

Мова HTML є додатком окремим випадком SGML (стандартної узагальненої мови розмітки) і відповідає міжнародному стандарту ISO 8879.

Мова XHTML є деяким варіантом HTML, у зв'язку з обмеженням XML і, фактично, XHTML можна сприймати як додаток мови XML у області розмітки гіпертексту.

У мережі HTML сторінки, як правило, передаються на бік клієнта від сервера за протоколами HTTP або HTTPS, у вигляді простого тексту або з використанням шифрування.

Дана мова була розроблена британським вченим Тімом Бернерс-Лі у 1986-1991 роках у Європейському Центрі ядерних досліджень в Женеві (Швейцарія). І створювалася як мова для обміну науковою і технічною документацією,яка придатна для використання людьми, що не є фахівцями в галузі верстки. Даний продукт успішно справлявся з проблемою складності SGML шляхом визначення невеликого набору структурних і семантичних елементів – дескрипторів. Дескриптори також часто називають «тегами». За допомогою HTML можна легко створити відносно простий, але красиво оформлений документ. Крім спрощення структури документа, в HTML внесена підтримка гіпертексту. Трохи пізніше було додано мультимедійні можливості [16].

Спочатку мову було спроектовано та створено як засіб структурування та форматування документів без їх прив’язки до засобів відтворення (відображення). Текст з розміткою HTML повинен був без стилістичних та структурних спотворень відтворюватися на обладнанні з різною технічною оснащеністю (кольоровий екран, монохромний екран, обмежений за розмірами екран або пристрій та програма голосового відтворення текстів). Сучасне застосування HTML дуже далеко від початкової задачі. Наприклад, тег <TABLE> призначений для створення в документах таблиць, але часто використовується і для оформлення розміщення елементів на сторінці. З плином часу основну ідею платформо незалежності мови було використано на потреби у мультимедійному і графічному оформленні.

На сьогодні Консорціум всесвітньої павутини розробив HTML версії 5. Варіант специфікації мови з’явився у мережі 20 листопада 2007 року.

Співтовариством Web Hypertext Application Technology Working Group, з початку 2004 року, розробляється специфікація Web Applications 1.0, часто неофіційно звана HTML 5, яка розширює HTML (втім, маючи і сумісний з XHTML 1.0 XML синтаксис) для кращого подання семантики різних типових сторінок, наприклад форумів, сайтів аукціонів, пошукових систем, онлайн-магазинів, які не дуже вдало вписуються в модель XHTML 2.

3.2 Аналіз можливостей каскадних таблиць стилів CSS

Cascading Style Sheets – каскадні таблиці стилів, формальна мова опису зовнішнього вигляду документа, написаного з використанням мови розмітки [15, 16].

Зазвичай використовується як засіб опису, оформлення зовнішнього вигляду веб-сторінок, що було написано мовами розмітки HTML або XHTML, але може також застосовуватися до будь-яких XML – документів таких як SVG або XUL.

CSS використовується розробниками веб-сторінок для завдання кольорів, шрифтів, розташування окремих блоків та інших аспектів представлення зовнішнього вигляду цих ресурсів. Основною метою розробки каскадних таблиць було розділення опису логічної структури сторінки (яке проводиться за допомогою HTML або інших мов розмітки) від опису зовнішнього вигляду цієї веб-сторінки (яке тепер проводиться за допомогою формальної мови CSS). Такий поділ може збільшити доступність документа, надати велику гнучкість і можливість управління його поданням, а також зменшити складність і повторюваність в структурному вмісті. Також, CSS дозволяє представляти один і той же документ в різних стилях або методах виведення, таких як екранне уявлення, друковане уявлення, читання голосом (спеціальним голосовим браузером або програмою читання з екрану), або при виведенні пристроями, що використовують шрифт Бройля.

HTML документи будуються на підставі ієрархії елементів, яка може бути наочно представлена в деревовидної формі. Елементи HTML по відношенню до інших можуть бути батьківськими або дочірніми, елементами – предками, елементами – нащадками, сестринськими.

Елемент є батьком іншого елемента, якщо в ієрархічній структурі документа він знаходиться відразу, безпосередньо над цим елементом. Елемент є предком іншого елемента, якщо в ієрархічній структурі документа він знаходиться десь вище цього елемента. Наприклад, в документі присутні два абзаци <p>, що включають в себе шрифт з напівжирним зображенням <b>. Тоді елементи <b> будуть дочірніми елементами своїх батьківських елементів <p>, і нащадками своїх предків <body>. У свою чергу, для елементів p елемент <body> буде тільки батьком. Також, ці два елементи <p> будуть сестринськими елементами, як мають одного і того ж батька – <body>. У CSS за допомогою селекторів можуть задаватися не лише поодинокі елементи, але й елементи, які є нащадками, дочірніми або сестринськими елементами інших елементів.

3.3 Аналіз можливостей мови програмування JavaScript

JavaScript – це прототипно-орієнтована сценарна мова програмування. Є діалектом мови ECMAScript [18]. Зазвичай використовується як вбудовувана мова для програмного доступу до об’єктів додатків. Найбільш широке застосування знаходить в браузерах як мова сценаріїв для додання інтерактивності на веб ресурсах.

Основні архітектурні риси: динамічна типізація, слабка типізація, автоматичне керування пам’яттю, прототипно-орієнтоване програмування, функції як об’єкти першого класу.

На дану мову програмування вплинули багато мов, при розробці була мета зробити мову схожим на Java, але при цьому легким для використання непрограмістів. Мовою JavaScript не володіє будь-яка компанія або організація, що відрізняє його від ряду мов програмування, використовуваних у веб-розробці. Назва «JavaScript» є зареєстрованим товарним знаком компанії Oracle Corporation.

Дана мова програмування є об’єктно-орієнтованою мовою, але у мові використовується прототипування, що обумовлює відмінності в роботі з об’єктами в порівнянні з традиційними клас-орієнтованими мовами. Крім того, JavaScript має ряд властивостей, властивих функціональним мовам, – функції як об’єкти першого класу, об’єкти як списки, каррінг, анонімні функції, замикання, що додає мові додаткову гнучкість [18].

Незважаючи на схожий з C синтаксис, JavaScript порівняно з мовою C має суттєві відмінності:

- об’єкти, з можливістю інтроспекції;
- функції як об’єкти першого класу;
- автоматичне приведення типів;
- автоматичне збирання сміття;
- анонімні функції.

Мова має ряд недоліків:

- модульна система JavaScript не надає можливості управляти залежностями та ізоляцією областей видимості;
- відсутній інтерфейс програмування додатків по роботі з файловою системою, управлінню потоками введення-виведення, базових типів для бінарних даних;
- стандартні інтерфейси до веб-серверів і баз даних;
- система управління пакетами, яка б відстежувала залежності і автоматично встановлювала їх.

Структурно JavaScript можна представити у вигляді об’єднання трьох чітко помітних один від одного частин:

- ядро (ECMAScript);
- об’єктна модель браузера (Browser Object Model або BOM);
- об’єктна модель документа (Document Object Model або DOM).

Якщо розглядати JavaScript не у контексты браузера, то об’єктна модель браузера і об’єктна модель документа можуть не підтримуватися.

Об’єктну модель документа іноді розглядають як окрему від JavaScript сутність, що узгоджується з визначенням DOM як незалежної від мови інтерфейсу документа. На противагу цьому ряд авторів знаходять BOM і DOM тісно взаємопов’язаними сутностями.

ECMAScript не є браузерною мовою і у ній не визначаються методи введення і виведення інформації. Це, основа для побудови скриптових мов. Специфікація ECMAScript описує типи даних, інструкції, ключові і зарезервовані слова, оператори, об’єкти, регулярні вирази, не обмежуючи авторів похідних мов у розширенні їх новими складовими.

Об’єктна модель браузера – специфічна частина мови, що є прошарком між ядром і об’єктною моделлю документа. Основне призначення об’єктної моделі браузера – управління вікнами браузера та забезпечення їх взаємодії. Кожне з вікон браузера представляється об’єктом window, центральним

об’єктом DOM. Об’єктна модель браузера на даний момент не стандартизована, однак специфікація знаходиться в розробці WHATWG і W3C.

Крім управління вікнами, в рамках об’єктної моделі браузера, браузерами зазвичай забезпечується підтримка наступних сутностей: управління фреймами; підтримка затримки у виконанні коду і зациклення із затримкою; системні діалоги; управління адресою відкритої сторінки; управління інформацією про браузер; управління інформацією про параметри монітора; обмежене керування історією перегляду сторінок; підтримка роботи з HTTP cookie.

Об’єктна модель документа – інтерфейс програмування додатків для HTML і XML – документів. Згідно DOM, документ може бути представлений у вигляді дерева об’єктів, що володіють рядом властивостей, які дозволяють виконувати з ним різні маніпуляції: генерація і додавання вузлів; отримання вузлів; зміна вузлів; зміна зв’язків між вузлами; видалення вузлів [19].

3.4 Вибір мови розробки WEB-ресурсу

Для побудови сайту банку необхідно використати багатошарову архітектуру. Дана архітектура є базовою на сьогодні і надає можливість логічно розбити розробку кожного з шарів окремо. Що значно полегшить сприйняття загальної структури проекту, розробку та підтримку проекту. Наступним етапом у розробці є вибір мови програмування для створення веб ресурсу. Розвиток комп’ютерних технологій з кожним днем все більше і більше полегшують роботу розробників веб-сайтів. Існує безліч програмних пакетів для реалізації розробки та відлагоджування різноманітних програмних продуктів. Майже кожен день створюються нові, більш потужні, мови програмування та технології. Різноманітні фреймворки дозволяють за допомогою декількох сотень рядків коду створити складний сайт, у той час,

коли ще декілька років тому для такої ж реалізації необхідно було б написати неодну тисячу рядків коду.

На сьогодні найбільш популярними для створення веб-сайтів є такі технології та мови програмування: Java EE, ASP.NET C#, PHP та інші. Усі з цих технологій є досить потужними і дозволяють створити веб-сайт практично будь-якої складності. Це все, що стосується «Back-end» частини і веб-програмування. Що ж стосується клієнтської частини, то тут робота полягає власне у створенні користувацького інтерфейсу, інформаційного забезпечення, створення динамічних та статичних зображень, розробка навігації веб-сайту. Все це відноситься до категорії веб-дизайну [19].

Порівняння популярних мов програмування для створення веб-сайтів представлено у таблиці 3.1 [20-22].

Аналіз дозволяє зробити такі загальні висновки.

1. Java — чудово підходить для великих корпоративних сайтів та систем із високими вимогами до стабільності. Якщо потрібна висока масштабованість, безпека та підтримка великих навантажень.
2. Python — кращий для швидкого прототипування та стартапів, але вимагає уважності при масштабуванні. Якщо важлива швидкість створення і мінімальний бюджет.
3. JavaScript (Node.js) — найкращий варіант для реактивних, реального часу додатків і повної розробки (frontend+backend). Якщо проєкт вимагає реактивності і роботи в реальному часі.
4. PHP — дешевий і швидкий для створення простих сайтів або CMS (WordPress, Joomla). Якщо важлива швидкість створення і мінімальний бюджет. Якщо основна задача — створити сайт-каталог або блог.
5. C# (.NET) — ідеальний для корпоративних проектів у середовищах Microsoft (особливо інтеграції з Windows-сервісами). Якщо потрібна висока масштабованість, безпека та підтримка великих навантажень.

Оскільки дана розробка орієнтована на обробку великих об’ємів даних, складну бізнес логіку, динамічний розвиток та модифікацію проекта в цілому необхідно використовувати такі технології як ASP.NET або Java EE.

Таблиця 3.1 - Порівняння популярних мов програмування для створення веб-сайтів.

Критерій	Java	Python (Django/Flask)	JavaScript (Node.js/Next.js)	PHP (Laravel/Symfony)	C# (.NET)
Швидкість роботи	Висока при належній оптимізації	Помірна	Висока для API, середня для великих обчислень	Середня	Висока
Простота розробки	Складніша через сувору типізацію	Дуже проста і швидка розробка	Помірна (особливо з сучасними фреймворками)	Досить проста	Помірна
Масштабо-ваність	Відмінна	Хороша, але потребує більше оптимізації	Хороша	Підходить для середніх проектів	Відмінна
Безпека	Висока (Spring Security)	Висока (Django має багато вбудованого захисту)	Залежить від фреймворка (Next.js краще)	Середня	Висока
Навчання	Потребує більше часу	Просте для новачків	Просте для новачків (але асинхронність складна)	Дуже легке	Помірне
Вартість розробки	Вища (дорогі спеціа-лісти)	Невисока	Помірна	Низька	Вища (особливо для enterprise)
Приклади сайтів	LinkedIn, Twitter (раніше)	Instagram, Pinterest	Netflix, Uber, LinkedIn (частково)	Wikipedia, Facebook (на початку)	StackOverflow, Microsoft сайти

Звісно, існує багато інших новітніх розробок мов програмування, що дозволяють розробляти веб орієнтовані додатки, однак вказані вище мови програмування є найпопулярнішими на сьогодні. Як перша так і друга мови програмування є дуже потужними та такими, що динамічно розвиваються

відмінністю є те, що ASP.NET розроблено і підтримується компанією Microsoft, а Java EE компанією Oracle. Продукти компанії Microsoft дещо простіші для сприйняття, більш автоматизовані та деякою мірою надійніші однак вимагають значних фінансових затрат. З іншого боку Java EE від Oracle може працювати майже на всіх існуючих операційних системах, не поступається надійністю та є безкоштовною. Технології на базі Java мають велику кількість реалізацій від багатьох компаній, не усі з цих реалізацій на 100% відповідають специфікації від Oracle, що інколи призводить до певних труднощів при розробці кінцевих продуктів. Також слід відмітити, що фінальна побудова проекта на Java повністю лежить на плечах розробника і здебільшого вимагає використання таких інструментів як Ant, Maven, Gradle(groovy), sbt(scala) та багато інших. Також для роботи з базами даних слід використовувати не plain SQL а якусь із реалізацій ORM(object oriented mapping), однією з найпопулярніших фреймворків для ORM під Java є Hibernate від JBoss.

Для виконання дизайнерської роботи необхідно визначитись які інструменти використовувати для написання HTML коду та для створення статичних та динамічних зображень. Для правки коду сторінок найкраще використовувати звичайний текстовий редактор, що має функцію підсвічування HTML тегів. Наприклад, редактор Notepad++ під MS Windows або один з найпопулярніших крос платформних редакторів серед розробників Front End-а Sublime. Не слід використовувати засоби для візуальної розробки сайтів, тому що такі інструменти зазвичай генерують надлишковий код, що робить сторінку «тяжкою» для відображення. Для створення якісних статичних зображень слід використовувати багатофункціональний графічний редактор. Наприклад, Photoshop від Adobe Systems або безкоштовний графічний редактор GIMP [23]. Існує багато програмних пакетів для створення анімації серед яких: Active GIF Creator, Barbarosa GIF Animator, CoffeeCap GIF Animator та багато інших. У якості динамічних зображень

буде використано GIF-анімацію [24, 25]. Для створення нескладної анімації у форматі GIF буде використано редактор Easy GIF Animator. Тому що дана програма відрізняється від інших простотою у використанні та зручним користувацьким інтерфейсом [25].

3.5 Створення WEB-ресурсу комерційного банку

Розділ контенту головної сторінки користувача містить список. Кожен рядок якого містить код, вид транзакції. Структуру сайту зображено на рисунку 3.1.

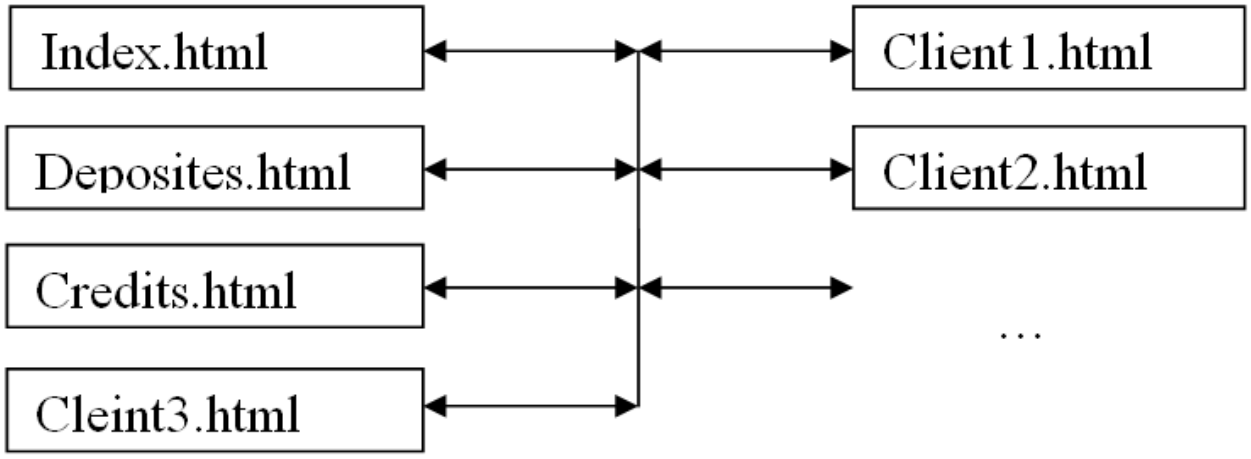

Рисунок 3.1 – Структура веб-сайту «Комерційний банк»

Головна сторінка міститься у файлі Index.html.

Кожен окремий користувач міститься у файлі client.html.

У файлі Deposites.html міститься список депозитів, що представлені на сайті.

У файлі Credits.html містяться список кредитів.

Кожний файл сторінки розроблено таким чином, що з будь-якої точки сайту можна повернутися на головну сторінку [11, 12, 19].

Усі без виключення веб-сайти фізично розміщуються на серверах. Доступ до сервера здійснюється по його IP адресі.

Для полегшення запам’ятовування адреси було розроблено систему DNS – Domain name system. Це ієрархічна система, яка що включає в себе всі IP адреси серверів і прив’язані до цих адресів оригінальні назви веб-сайтів у певному форматі. Кожен клієнт(браузер) «знає» адреси DNS серверів і при запиті на завантаження сайту, спочатку визначає IP адресу сервера, а за тим відкриває з ним TCP з’єднання яке забезпечує з’єднання з веб-сервером по протоколу HTTP або HTTPS і завантаження власне сторінки.

Дані сайту зазвичай зберігаються у базах даних. Зазвичай доступ до бази даних здійснюється по мережі по протоколу TCP. Тому дані можуть міститися як на машині на якій встановлено веб сервер, або на віддаленій машині.

Скомпільований веб-сайт мовою Java зазвичай міститься у файлі з розширенням war або ear залежно від того який еб сервер буде використовуватися для реалізації сайту. War файл дослівно означає web archive який містить усі необхідні файли для роботи сайту. Такий архів необхідно завантажити у веб контейнер. Наприклад Tomcat, Jetty, JBoss або GlassFish.

Зазвичай усі веб-додатки побудовані за багатошаровою архітектурою. Приклад архітектури зображено на рисунку 3.2.

Найвище у цій ієрархії містяться веб компоненти, такі як jsp, js, css та інші файли. Нижче розміщено шар контролерів, основна задача цього шару це мапінг запитів від клієнта по URL.

За ним викликається шар служб, який містить бізнес логіку та має доступ до рівня DAO Data Access Object. У свою чергу DAO викликає ORM, яка у сутностях DAO поставляє у рівень служб дані із бази.

Після чого служба передає оброблені дані у контролер а контролер переносить дані у веб компонент і відправляє відповідь клієнту на браузер.

Дана багатошарова архітектура включає в себе багато таких визначених прийомів рішень задач, які мають назву шаблони.

За допомогою шаблонів можна значно спростити розробку програми в цілому і полегшити розробникам підтримку проекта. Наприклад, загальна архітектура базується на шаблоні MVC, який у свою чергу включає у себе ще ряд інших шаблонів.

На рисунку 3.3 зображено UML діаграму гілки роботи з користувачами.

3.6 Розробка динамічного контенту

Динамічний або адаптивний, або смарт-контент – це термін, який відноситься до роботи сайту або поштової розсилки і означає, що вміст змінюється в залежності від інтересів чи поведінки користувача [26]. Складається таке враження, що цей контент був спеціально створений для цього конкретного користувача саме в даний момент перегляду. Найбільш відомий приклад динамічного контенту – це рушій рекомендацій Amazon. Інші форми смарт-контенту можуть включати вставку графіки і пропозицій на сторінку, яка показується в залежності від того, хто переглядає її в даний момент, персоналізацію полів в електронних листах.

Ефективність смарт-контенту в його релевантності. Численні дослідження підтвердили, що таргінг маркетинг з більш високою релевантністю до кінцевого одержувача дає набагато кращі результати. Наприклад, за даними Jupiter Research релевантні електронні листи приносять у 18 разів більше доходів, ніж розсилка загального характеру [27].

Ліди (потенційні клієнти, які виявили інтерес до продукту чи послуги компанії), отримані в результаті споживання цільового контенту , зумовлюють зростання продажів до 20 % (за даними Annuitas Group) [28].

Лід – багатозначний термін, що відноситься до рекламної діяльності в інтернеті. Лідом називається: особлива схема оплати реклами в інтернеті, окрема категорія інтернет-користувачів, дію цих інтернет-користувачів.

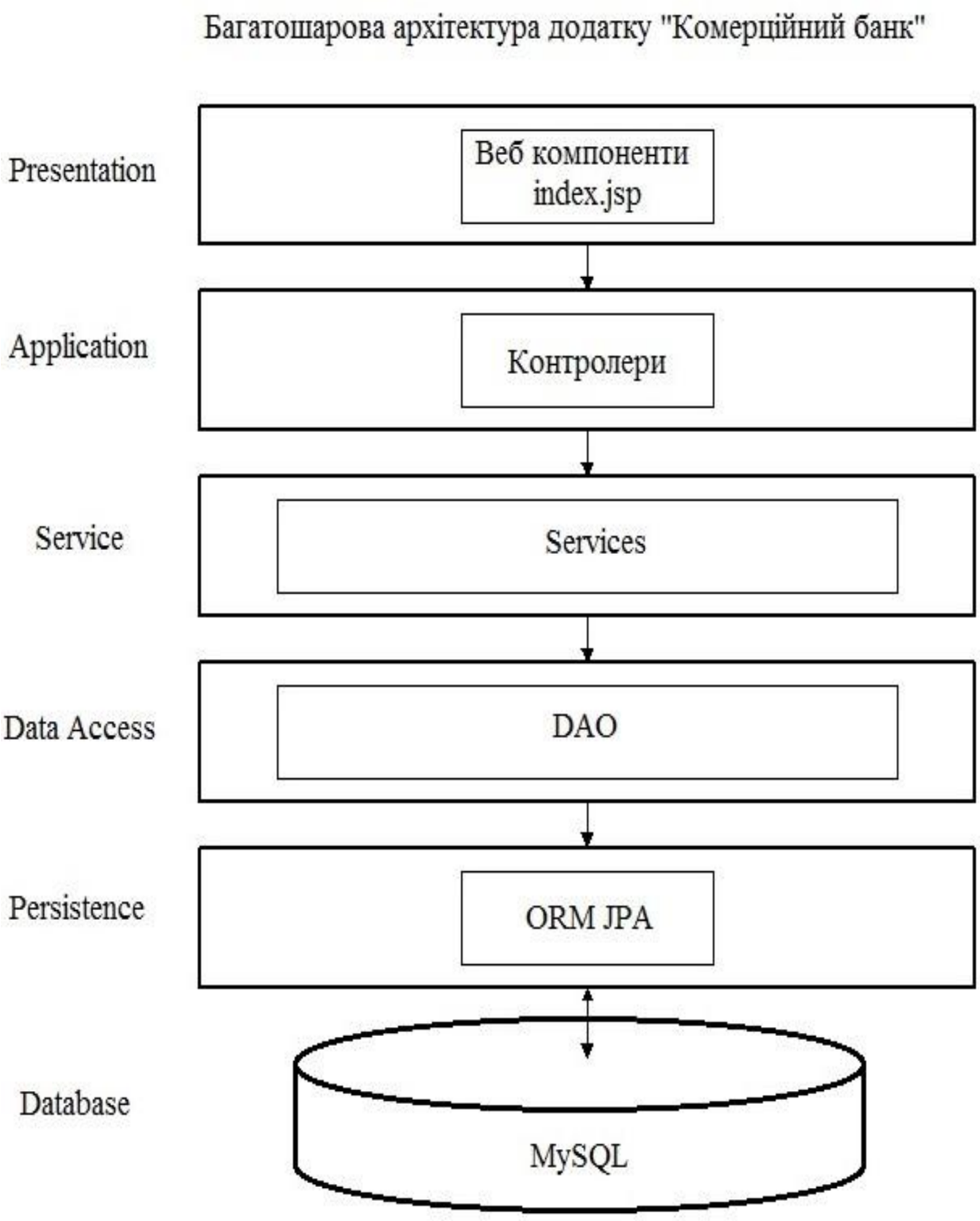

Рисунок 3.2 – Ієрархічне зображення веб-додатку

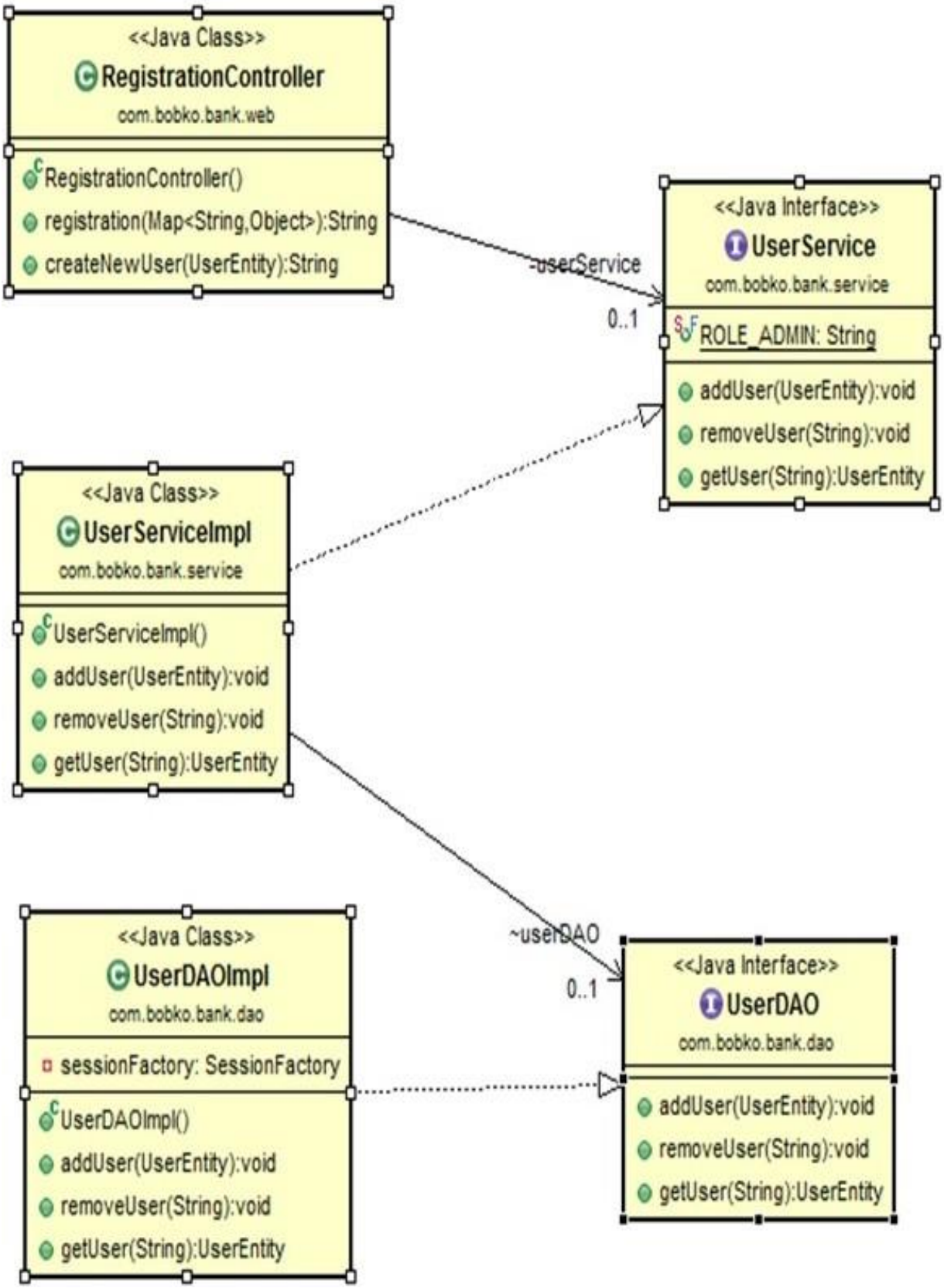

Рисунок 3.3 – UML діаграма гілки для роботи з користувачем

У рамках ліда рекламодавець оплачує не розміщення рекламних матеріалів на майданчику, а значущі для нього користувача дії. До таких дій належать, наприклад, реєстрація на сайті рекламодавця, заповнення клієнтської анкети, власне замовлення послуг або придбання товару.

Термін походить від англійського слова «lead» – вести за собою. Рекламодавцем оплачується дія користувача, якого «привела» на його сайт реклама.

Можливість релевантності обумовлена зібраними даними (релевантність – загальному сенсі «адекватність», тобто не тільки оцінка ступеня відповідності, але і ступеня практичної застосовності результату, а також ступеня соціальної застосовності варіанта рішення задачі). І це не тільки контактні дані і демографічний профіль, але також збережені інсайти про матеріали, які допомогли привести лід або клієнта до побудови відносин з компанією.

Ця інформація дозволяє надавати контент тим, кому він потрібний, і в необхідний час. Технологія надання смарт-контенту включає в себе централізовану маркетингову базу даних;

- генератор смарт-контенту;
- гнучкі в налаштуванні веб –сторінки;
- інтегровану систему електронної пошти.

Для початку роботи з подібним видом контенту необхідні наполегливість і впевненість у його доцільності. Динамічний контент повинен поліпшити враження відвідувачів вашого сайту від роботи з ним і принести більше лідів.

Перш ніж інтегрувати будь-який варіант динамічного контенту в вашу маркетингову стратегію, дайте відповідь на питання «Як використання смарт-контенту поліпшить показники перебування відвідувача на сайті або прочитання ним листа?».

Існує декілька практичних рекомендацій, з яких варто почати, якщо ви зрозуміли не до кінця, як впровадити смарт-контент у маркетингові заходи на веб-сайті та в електронні розсилки.

Якщо відвідувач сайту раніше вже скачував конкретну пропозицію або купив певну одиницю товару, необхідно застосовувати «розумні» політики і правила роботи з контентом, щоб ця пропозиція більше не відображалося на сторінці, коли її буде переглядати даний користувач.

Життєвий цикл ліда показує, наскільки далеко відвідувач сайту заходить, перш ніж щось придбає. Чи це перше його відвідування сайту? Чи готовий користувач до замовлення або ще оцінює опції і різні альтернативи? Інформація про те, який користувальницький досвід спричинив певний лід, дозволить уникнути зайвого тиску на тих, хто, можливо, прицінюється і не визначився з остаточним вибором [29].Замість того, щоб знову змушувати клієнта заповнювати форму, ви можете використовувати динамічний контент, який автоматично буде розпізнавати повернувся користувача і давати йому можливість скористатися закликом до дії, який дозволить уникнути безлічі зайвих кроків.

На рисунку 3.4 надано UML-діаграма послідовності для процесу реєстрації користувача. Ця діаграма показує логіку створення нового користувача, обробку можливих помилок і взаємодію з базою даних через сервісний рівень.

Учасники (Actors & Lifelines):

- User (користувач) — ініціює запит на реєстрацію;
- RegistrationController — обробляє запити від користувача;
- UserService — сервіс для роботи з даними користувачів;
- Database (база даних) — зберігає дані користувача.

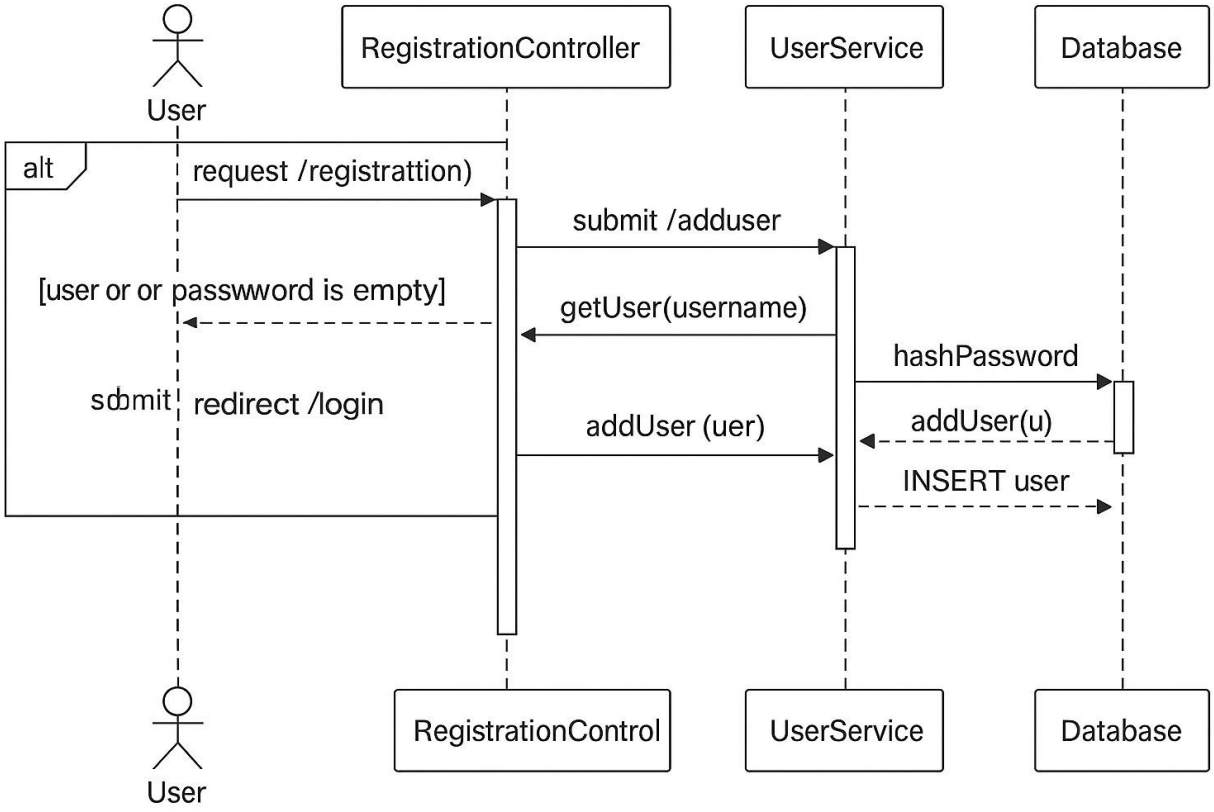

Рисунок 3.4 - UML-діаграма послідовності для процесу реєстрації користувача

Послідовність дій така.

1. Користувач → RegistrationController:
   GET /registration – користувач переходить на сторінку реєстрації, контролер повертає шаблон сторінки registrationPage з об'єктом UserEntity у моделі.
2. Користувач → RegistrationController:
   POST /adduser – відправка форми реєстрації з логіном і паролем.
3. RegistrationController (локально):
   - перевіряє заповнення обов’язкових полів;
   - встановлює роль ROLE_ADMIN і активність користувача.
4. RegistrationController (локально). Генерує хеш паролю за алгоритмом MD5.
5. RegistrationController → UserService:

getUser(user.usrName) – перевіряє, чи існує вже користувач з таким ім’ям.

6. UserService → Database:
   запит до бази на пошук користувача.
7. UserService → RegistrationController:
   повертає null або об'єкт UserEntity.
8. RegistrationController (умова):
   якщо користувач існує → редірект з помилкою.
9. RegistrationController → UserService:
   addUser(user) – додає нового користувача до бази.
10. UserService → Database:
    створює запис нового користувача.
11. UserService → RegistrationController:
    підтвердження успішного збереження.
12. RegistrationController → Користувач:
    redirect:/login – редірект на сторінку входу.

У додатку В надано програмний код реалізації WEB-ресурсу комерційного банку.

На рисунку 3.5 надано приклад головної сторінки комерційного банку.

На рисунку 3.6 надано зовнішній вигляд сторінки логінізації.

На рисунку 3.7 надано зовнішній вигляд сторінки реєстрації.

3.7 Тестування роботи сайту

Завершальним етапом розробки сайту є етап тестування. Тестування є найважливішим етапом на стадії завершення розробки. Що дозволяє виявити помилки, які неможливо виявити підчас розробки.

Під час роботи над сайтом розробник має вважати на багато чинників, які впливають на вид документа.

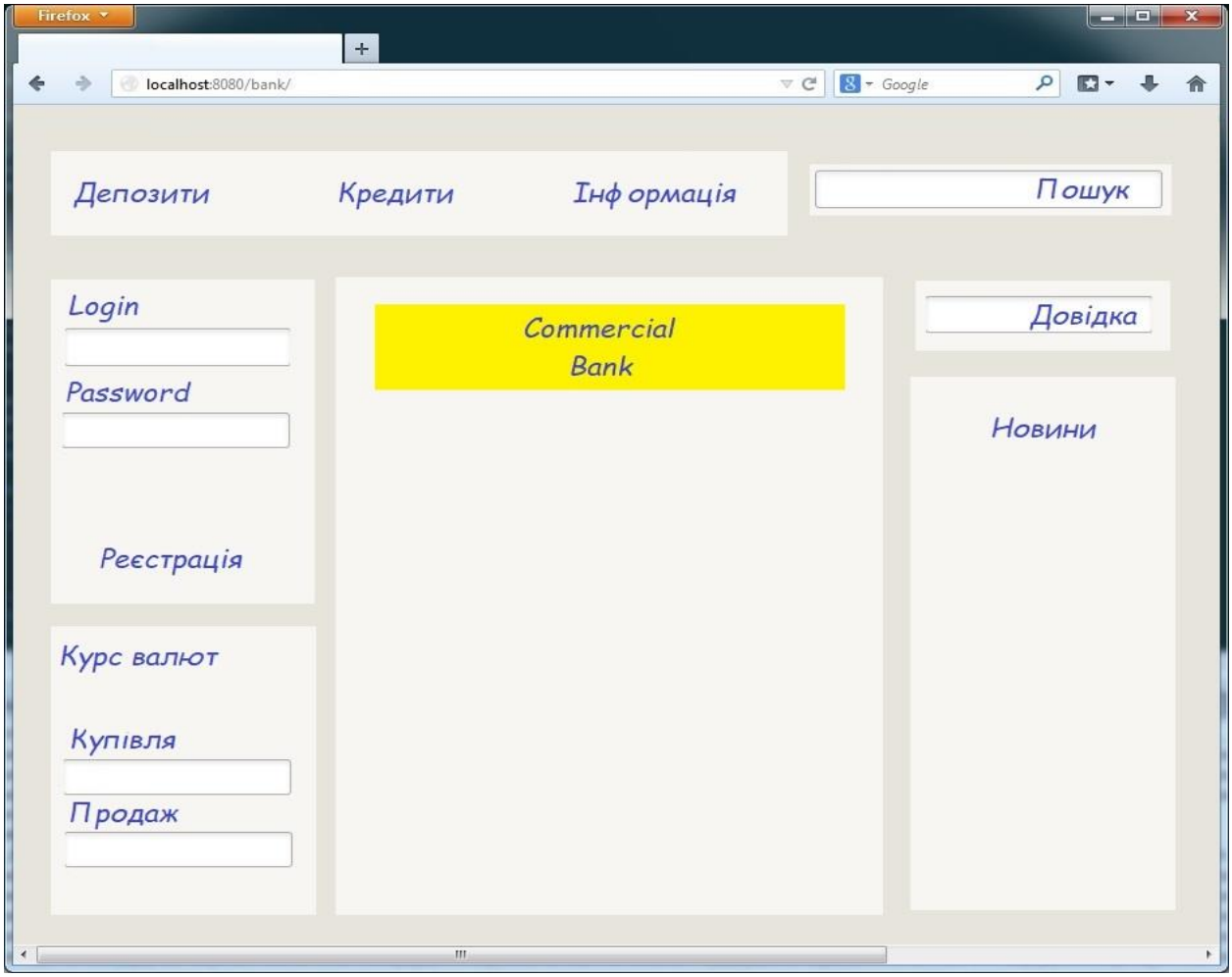

Рисунок 3.5 - Приклад головної сторінки комерційного банку.

Відвідувачі сайту мають не лише різні операційні системи та браузери, але і такі параметри, як кількість кольорів на моніторі, його роздільність, доступність JavaScript тощо.

Після закінчення верстання слід провести ряд перевірок і у разі виявлення явних помилок, внести в код відповідні зміни. Зрозуміло, це зручніше робити за допомогою спеціалізованих програм.

Відлагодження - це процес знаходження помилок в коді і виправлення небажаного поведінки елементів в браузері. Як правило, відхилення макета від первісного дизайну відстежується в процесі верстання, але бувають ситуації, коли помилки необхідно виправити вже на робочому сайті [19].

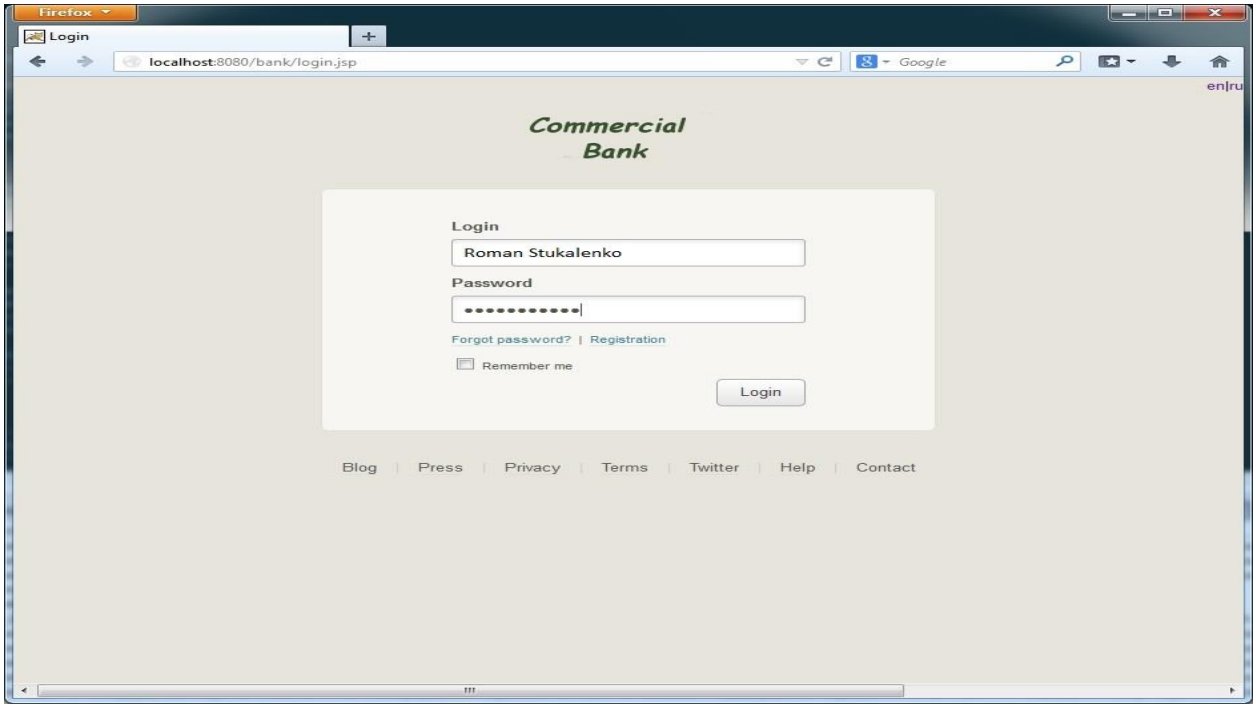

Рисунок 3.6 - Зовнішній вигляд сторінки логінізації

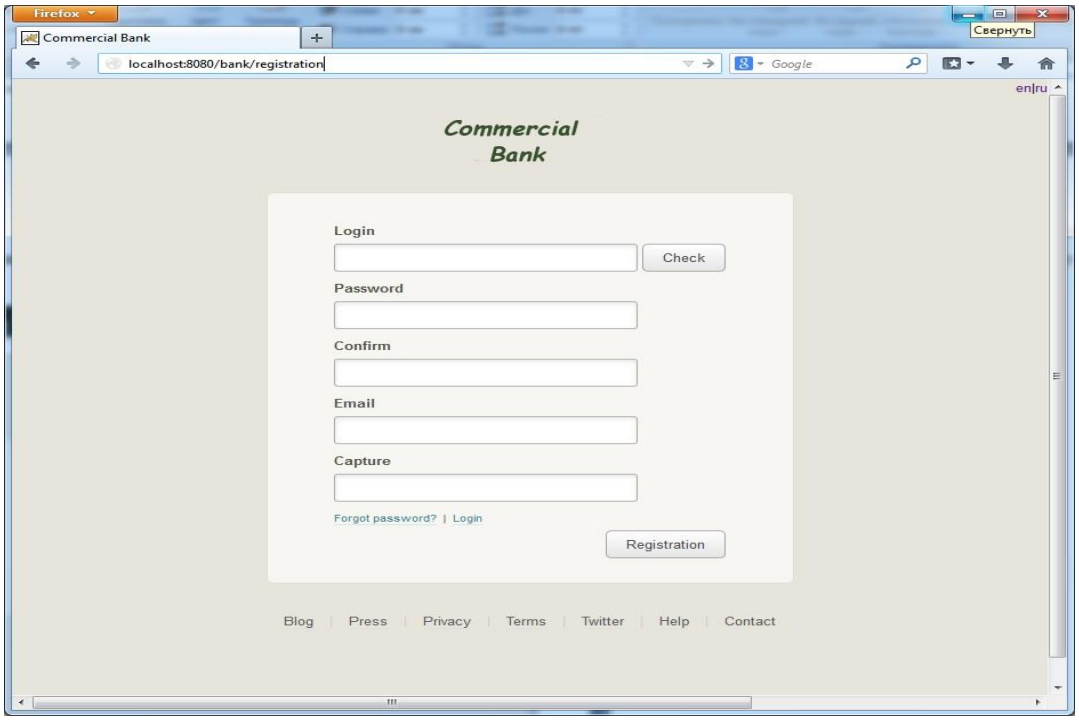

Рисунок 3.7 - Зовнішній вигляд сторінки реєстрації

Наприклад, помилка може бути виявлена після додавання нового блоку контенту, тестування сайту в різних версіях браузерів, при різних роздільностях монітора і інших умов.

Також розробник повинен вміти швидко розуміти чужий код, відстежити причину появи помилки і усунути її. Розуміння логіки чужого коду потрібне при роботі в команді, або при поверненні до власної роботи через якийсь час, коли вона вже сприймається як чужа.

На об'ємних сайтах з десятками тисяч рядків вихідного коду HTML вичленувати проблемне місце досить складно, тому потрібно мати інструмент, який дозволяє показати код HTML і CSS вибраного фрагмента і провести над ним експерименти.

Більшість існуючих в Інтернеті сторінок, на жаль, не є валідними. Багато розробників вважають, що абсолютно суворе дотримання стандартів HTML - зовсім не обов'язкова умова того, щоб сайт вийшов якісним і ефективним.

Дійсно, існує багато успішних сайтів, код яких не проходить прискіпливої перевірки валідатором. Немає сумнівів, що можна зробити хороший сайт, не домагаючись повної відповідності коду до формальних правил мови. Однак, розробка валідного коду має ряд переваг як для самого розробника, так і для кінцевого користувача. Тому, варто привчити себе писати грамотний код і перевіряти його валідатором.

1. Перевірка документів валідатором дозволяє уникнути дрібних прикрих помилок - неправильно вкладених тегів, пропущених дужок і лапок тощо.
2. Сучасні браузери підтримують стандарти W3C значно краще, ніж їх попередні версії. Ця тенденція зберігається, тому відповідність DTD набуває все більшої важливості для правильного відображення сторінок в браузерах.
3. Оскільки валідний код відповідає певним формальним правилам, його

легше інтерпретувати й обробляти. Він швидше аналізується і відображається в браузері, з ним легше працювати пошуковикам і системам індексації.

4. Валідність коду гарантує сумісність сторінок не лише з існуючими, але і з майбутніми версіями браузерів. Тому, не доведеться переписувати ваші сторінки після виходу нової версії Internet Explorer або Opera.

Валідатор Консорціуму W3C - кращий з існуючих. З його допомогою можна перевірити любий документ, що розташований в Інтернеті або на локальному комп'ютері [1, 30].

Не рекомендується користуватися іншими валідаторами - принаймні до тих пір, поки розробник не навчиться відрізняти справжні валідатори від програм, які просто так називаються.

Піклуючись про грамотність сторінок, про їх відповідність стандартам W3C, не треба впадати в крайнощі і думати, що якщо сторінка успішно перевірена валідатором, то вона автоматично є якісною і ефективною.

Грамотний код - це важливий, але далеко не єдиний показник якості сторінки. Дійсно, можна написати текст, який буде цілком задовольняти граматичним правилам, але при цьому він виявиться невдалим або зовсім безглуздим.

Для перевірки веб-сторінок на наявність помилок і зауважень існує багато підходів. Умовно вони поділяються на онлайнові і локальні. Онлайнові призначені для перевірки сторінок за допомогою браузера через Інтернет, а локальні використовуються для перевірки документів на комп'ютері.

Якщо не приділяти уваги до кросбраузерності , можна втратити багато відвідувачів. Адже перша думка про сайті, складається при перегляді зовнішнього вигляду сайту, його фону та впорядкованого розташування елементів. Некоректне відображення певних елементів дизайну, розбіжність

стиків, може викликати негативну реакцію користувачів, і вони схильні покинути сайт. Пізніше, вони будуть просто ігнорувати цей сайт.

Кросбраузерність - це властивість певного сайту практично однаково відображатися і працювати в браузерах [31, 32]. Сайт має відображати матеріал з однаковим рівнем читабельності, динамічні елементи виконувати всі ядії, що закладені розробником. Кросбраузерність є обов'язковою для кожного сайту.

На сьогоднішній день більше 95% користувачів користується браузерами Opera, Mozilla Firefox, Google Chrome і Internet Explorer. Не можна з впевненістю сказати, що кожна програма відтворює HTML код або компілює CSS по-своєму, бо вони мають дотримуються певних правил. Але іноді, розробник щось упускає і браузер не відображає властивості елементів або, ж відображає їх некоректно.

Браузери можуть мати три різні версії.

1. Альфа-версія. Це сира версія браузера - пілотний проект. Вона випускається виключно для того щоб тестувати нові функції та виправляти різні недоробки.
2. Бета-версія. Це протестована і допрацьована версія браузера. Вона може повноцінно використовуватися користувачами. Однак, в дану версію постійно вносяться доповнення та поновлення. Отже, вона ця версія не є остаточною і стабільною.
3. Стабільна версія. Це означає, що версія є остаточною, враховано і виправлено всі слабкі місця, закладені функції працюють без збоїв. Для веб-серфінгу варто використовувати ці версії.

Тестувати власні сторінки і сайти слід на стабільних версіях браузера. Таким чином, можна уникнути помилок, які пізніше можуть з'явитися і сприяти збою або неправильному результату. Зазвичай, помилки виникають через недоробки, які були допущені розробниками в програмі в її кодах, або в дизайні. Також, збої можуть виникнути із-за некоректної роботи

компілятора, який, відповідно, виробляє некоректний або неправильний код. Отже, щоб уникнути таких прикрощів, потрібно ретельно перевіряти кожну версію браузера.

Стара школа тестування передбачала завантаження коду розробленого сайту на кілька комп'ютерів з різними комбінаціями браузерів і операційних систем. Такий спосіб чудово працює в умовах наявності великої кількості різних комп'ютерів і часу, який доведеться витратити на перевірку. Більш ефективним підходом до перевірки кросбраузерності сайту є безкоштовні і комерційні веб-сервіси чи програми [31, 32].

Порядок тестування сайту.

1. Інсталювати на комп’ютер браузери Opera, Mozilla Firefox, Google Chrome і Internet Explorer.
2. Ознайомитися з засобами відлагодження в різних браузерах.
3. Додати до браузера Mozilla Firefox плагін Firebug. Дослідити його роботу на прикладі власного сайту. Розглянути код чужої сторінки, витягнути цікаві стилі чи HTML фрагменти, застосувати у власній розробці.
4. Протестувати сторінки у валідаторі, виправити виявлені помилки.
5. Перевірити сайт на кросбраузерність, зробити скріншоти сторінок в обраних браузерах. У разі виявлення недоречностей - виправити.

Під час тестування сайт працював без помилок на всіх браузерах. Шрифти відображались нормально. Зображення були розміщенні в коректних місцях.

3.8 Висновок до розділу 3

У результаті досліджень та розробки виконано вибір мовних засобів розробки WEB-ресурсу комерційного банку, створено WEB-ресурс, проведено успішне тестування роботи сайту.

ВИСНОВКИ

Поставлені перед бакалаврською кваліфікаційною роботою задачі виконано в повному обсязі [2], а саме:

- досліджено вже наявні аналогічні рішення;
- обрано та обґрунтувао підхід до вирішення поставленого завдання;
- розроблено структури та дизайн WEB-ресурсу для комерційного банку;
- створено програму WEB-ресурсу банківської установи;
- протестовано та перевірено функціонування ресурсу;
- створено користувацьку інструкцію з експлуатації WEB-ресурсу.

В ході виконання бакалаврської роботи були проаналізовані найпопулярніші системи для розборки WEB-ресурсу банківської установи. Відповідно до завдання було проведено аналіз проблеми та обґрунтування необхідності даної розробки. Розроблено структуру та систему навігації сайту. Наступним етапом проектування було створення інформаційного забезпечення, одного із важливих етапів, який відповідає за наявність на сайті якісного контенту, що призводить до популярності сторінки серед користувачів. Відповідно до всіх вимог було проведено розробку графічного забезпечення, що включало в себе розробку графіки та анімації а саме: текстур, логотипів, «шапки» сайту, різноманітних розділювачів та інших елементів дизайну. Відповідно з завданням до бакалаврської роботи був розроблений WEB-ресурс банківської установи. Також було проведено тестування, яке підтвердило роботоспроможність WEB-ресурсу.

Мета роботи – розширення функціональних можливостей WEB-ресурсу комерційного банку за рахунок покращення етапу реєстрації та створення дружнього інтерфейсу досягнута.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Р.В. Стукаленко, В.О. Денисюк. Розробка сайту комерційного банку. Молодь в науці: дослідження, проблеми, перспективи (МН-2025). Вінницький національний технічний університет. URL: https:// conferences.vntu.edu.ua/index.php/mn/mn2025/paper/viewFile/24615/ 20386
2. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 122 «Комп'ютерні науки» [Електронний ресурс] / уклад.: А. А. Яровий, О. В. Сілагін, І. В. Богач. Вінниця: ВНТУ, 2023. (PDF, 54 с.). URL: https://iq.vntu.edu.ua/method/getfile.php?fname= 124285.pdf&x= 1&card_id=46522&id=124285
3. Онлайн-банкінг: що це таке, як він працює та найкращі варіанти саме для вас. URL: https://www.payoneer.com/uk/resources/business/what-is-online-banking-how-to-use/
4. How to Create an Online Banking Website. URL: https://rewisoft.com/blog/guide-on-banking-website-development/
5. Технічна підтримка сайту для банківських установ.URL: https://seo-evolution.com.ua/blog/razrabotka/tehnichna-pidtrimka-saytu-bankivskih-ustanov
6. .Bank – багатосторінковий шаблон веб-сайту Bootstrap 5 з фінансів та банківської справи. URL: https://www.templatemonster.com/ua/html-templates/68812.html?srsltid=AfmBOorKpTmlPwgmGQxMQqJNHhjFbN742lyJ2gYI0DmOtSqQEmu6CrRF
7. Розробка сайтів та застосунків для банку. URL: https://redchameleon.com.ua/ua/industries/razrabotka-saytov-i-prilozheniy-dlya-banka/
8. Створення сайту для банку. URL: https://wezom.com.ua/ua/sozdanie-sajta-dlja-banka

9. Технічна підтримка сайту для банківських установ. URL: https://seo-evolution.com.ua/blog/razrabotka/tehnichna-pidtrimka-saytu-bankivskih-ustanov
10. 7 Найкращих Open Banking Платформ. URL: https://stfalcon.com/uk/blog/post/top-7-open-banking-platforms
11. Пасічник О.В., Пасічник В.В. Веб-дизайн: Підручник. 2-ге вид., стер. Львів: ПП “Магнолія 2006”, 2024. 520 с.
12. Двірничук К.В., Вацек Д.О. Веб-програмування та веб-дизайн : навч. посіб. Чернівці: Чернівец. нац. ун-т ім. Ю. Федьковича, 2022. 472 с.
13. Базові правила роботи з кольором для дизайну сайтів. URL: https://skvot.io/uk/blog/glavnoe-o-tsvetovykh-skhemakh
14. Психологія кольору у веб-дизайні. URL: https://ux.pub/designer_kaliaieva/psikhologhiia-koloru-u-vieb-dizaini-stvoriennia-divovizhnogho-dosvidu-koristuvacha-chieriez-kolirni-skhiemi-7e4
15. Jon Duckett. HTML & CSS: Design and Build Websites. URL: https://www.htmlandcssbook.com
16. Philippe Hong. Practical Web Design: Learn the fundamentals of web design with HTML5, CSS3, Bootstrap, jQuery, and Vue.js. Packt Publishing. 2018. 368.
17. Eric Freeman, Elisabeth Robson. Head First HTML5 Programming. URL: https://ebooks.allfree-stuff.com/eBooks_down/HTML5%20and%20CSS3/Head%20First%20HTML5%20Programming.rar
18. Сучасний підручник з JavaScript. URL: https://uk.javascript.info/
19. Розробка веб сайтів для початківців HTML, CSS, JAVASCRIPT. URL: https://nvkarta.com/project/library/uploads/engineering/programming/(uk)_rozrobka_veb-saitiv_dlia_pochatkivtsiv_html_css_javascript.pdf
20. Рейтинг мов програмування: ТОП 5 мов програмування у 2024 році. URL: https://beetroot.academy/blog/top-5-mov-programuvannya-u-2024-roci

21. Топ-10 мов програмування, які будуть актуальні в 2025. URL: https://career.softserveinc.com/uk-ua/stories/top-10-programming-languages-to-learn
22. Найпопулярніші мови програмування у 2025 році. URL: https://dan-it.com.ua/uk/blog/samye-populjarnye-jazyki-programmirovanija-v-2025-godu/
23. Створення 3D-об’єктів і анімації. URL: https://helpx.adobe.com/ua/photoshop/using/creating-3d-objects-animations-photoshop.html
24. Convert your videos to GIFs for free. URL: https://www.adobe.com/express/feature/video/convert/video-to-gif
25. Number one GIF animator for Windows. URL: https://www.easygifanimator.net/
26. Динамічна веб-сторінка. URL: http://uk.wikipedia.org/wiki/Динамічна_веб-сторінка .
27. Jupiter Research Inc. URL: https://www.jupiterfinancialresearch.com/
28. ANNUITAS Builds Perpetual Demand Engines URL: https://www.annuitas.com/
29. The lead lifecycle. URL: https://learn.microsoft.com/uk-ua/dynamics365/customer-insights/journeys/lead-lifecycle
30. Markup Validation Service. URL: https://validator.w3.org/
31. Інструменти для тестування кросбраузерності верстки. URL: https://training.qatestlab.com/blog/technical-articles/tools-for-testing-cross-browser-layout/
32. Як перевірити кросбраузерність сайту. URL: https://seo-akademiya.com/ua/baza-znan/servisi-dlya-seo/top-12-servisiv-perevirki-krossbrauzernosti-sajtu/

ДОДАТОК А (ОБОВ'ЯЗКОВИЙ)
ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка WEB-ресурсу комерційного банку

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 28,55 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Яровий А.А., зав. каф. КН _____ _____
(прізвище, ініціали, посада) (підпис)

Колесницький О.К., проф. каф КН _____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Озеранський В.С.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Денисюк В.О.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Стукаленко Р.В.
(підпис) (прізвище, ініціали)

ДОДАТОК Б (ОБОВ’ЯЗКОВИЙ)
ЛІСТИНГ ПРОГРАМИ

bank.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Комерційний Банк</title>
  <style>
    :root {
      --primary-color: #003366;
      --accent-color: #00bcd4;
      --bg-color: #f4f6f8;
      --white: #fff;
      --radius: 12px;
      --shadow: 0 4px 12px rgba(0,0,0,0.1);
    }

    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    body {
      font-family: 'Segoe UI', sans-serif;
      background: var(--bg-color);
      color: #333;
    }

    header {
      background: var(--primary-color);
      color: var(--white);
      text-align: center;
      padding: 2rem;
      box-shadow: var(--shadow);
    }

    header h1 {
      font-size: 2.2rem;
```

```
  margin-bottom: 0.5rem;
}

nav {
  background: var(--accent-color);
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  padding: 0.5rem 0;
}

nav a {
  padding: 1rem;
  color: var(--white);
  text-decoration: none;
  transition: background 0.3s;
}

nav a:hover {
  background: rgba(255,255,255,0.15);
  border-radius: var(--radius);
}

main {
  max-width: 700px;
  margin: 2rem auto;
  padding: 1rem;
}

.section {
  background: var(--white);
  padding: 2rem;
  margin-bottom: 1rem;
  border-radius: var(--radius);
  box-shadow: var(--shadow);
  display: none;
  animation: fadeIn 0.5s ease forwards;
}

.section.active {
  display: block;
}

form input, form button {
```

```
      width: 100%;
      padding: 0.75rem;
      margin-top: 1rem;
      border: 1px solid #ccc;
      border-radius: var(--radius);
    }

    form button {
      background: var(--accent-color);
      color: white;
      font-weight: bold;
      border: none;
      transition: background 0.3s;
      cursor: pointer;
    }

    form button:hover {
      background: #0097a7;
    }

    .error {
      color: red;
      margin-top: 0.5rem;
    }

    @keyframes fadeIn {
      from { opacity: 0; transform: translateY(10px); }
      to { opacity: 1; transform: translateY(0); }
    }
  </style>
</head>
<body>

  <header>
    <h1>Комерційний Банк</h1>
    <p>Надійність. Безпека. Ваш комфорт.</p>
  </header>

  <nav>
    <a href="#" onclick="showSection('home')">Головна</a>
    <a href="#" onclick="showSection('about')">Про банк</a>
    <a href="#" onclick="showSection('services')">Послуги</a>
    <a href="#" onclick="showSection('login')">Увійти</a>
    <a href="#" onclick="showSection('register')">Реєстрація</a>
```

```
  </nav>

  <main>
    <section id="home" class="section active">
      <h2>Ласкаво просимо!</h2>
      <p>Наш банк — це сучасна фінансова установа з найкращими умовами обслуговування клієнтів. Працюємо чесно — обслуговуємо якісно.</p>
    </section>

    <section id="about" class="section">
      <h2>Про нас</h2>
      <p>Ми — провідний український банк з 2001 року. Маємо понад 300 відділень по всій країні, мільйони клієнтів і сотні партнерів.</p>
    </section>

    <section id="services" class="section">
      <h2>Послуги</h2>
      <ul>
        <li>Кредити для бізнесу та фізичних осіб</li>
        <li>Онлайн-банкінг 24/7</li>
        <li>Депозити з вигідними ставками</li>
        <li>Міжнародні перекази</li>
      </ul>
    </section>

    <section id="login" class="section">
      <h2>Увійти</h2>
      <form method="post" action="/j_spring_security_check">
        <input type="text" name="j_username" placeholder="Логін" required />
        <input type="password" name="j_password" placeholder="Пароль" required />
        <label><input type="checkbox" name="remember" /> Запам'ятати мене</label>
        <button type="submit">Увійти</button>
      </form>
    </section>

    <section id="register" class="section">
      <h2>Реєстрація</h2>
      <form id="registerForm">
        <input type="text" name="usrName" placeholder="Ваш логін" required />
        <input type="password" name="pw" placeholder="Ваш пароль" required />
        <button type="submit">Зареєструватися</button>
        <div class="error" id="errorMessage"></div>
```

```
      </form>
    </section>
  </main>

  <script>
    function showSection(id) {
      document.querySelectorAll('.section').forEach(sec => {
        sec.classList.remove('active');
      });
      document.getElementById(id).classList.add('active');
    }

    document.getElementById('registerForm').addEventListener('submit', async (e)
=> {
      e.preventDefault();
      const form = e.target;
      const formData = new FormData(form);

      try {
        const response = await fetch("/adduser", {
          method: "POST",
          body: formData
        });
        if (response.redirected) {
          window.location.href = response.url;
        } else {
          document.getElementById("errorMessage").textContent = "Не вдалося
створити акаунт.";
        }
      } catch (err) {
        document.getElementById("errorMessage").textContent = "Сервер
недоступний.";
      }
    });
  </script>

</body>
</html>
```

Вихідний код контролера, який реалізує створення користувачів

```
package com.bank.web;
/**
 * Controller class that provides registration of new users
 * @author roman stukalenko roman
 * @data 12.05.2025
 */

import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Map;

import org.hibernate.exception.ConstraintViolationException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com. stukalenko.bank.domain.UserEntity;
import com. stukalenko.bank.service.UserService;

@Controller
public class RegistrationController {

    @Autowired
    private UserService userService;
    /**
     * redirect to registration page
     * and put to Map UserEntity object
     * */
    @RequestMapping("/registration")
    public String registration(Map<String, Object> map) {
        map.put("user", new UserEntity());
        return "registrationPage";
    }

    /**
     * perform adding new user to db
     * */
    @RequestMapping(value = "/adduser", method = RequestMethod.POST)
    public String createNewUser  (
            @ModelAttribute("user") UserEntity user) {

            //check empty fields
            if ((user == null) || user.getPw() == null ||
user.getPw().isEmpty() || user.getUsrName() == null ||
user.getUsrName().isEmpty()) {
```

```
                return "redirect:/registration?error=true";
            }

            //set default user role
            user.setRole(UserService.ROLE_ADMIN);
            user.setActive(true);

            MessageDigest messageDigest = null;
            String hashedPass = "";

            //encode pw to md5 hash
            try {
                messageDigest = MessageDigest.getInstance("MD5");
            messageDigest.update(user.getPw().getBytes(),0,
user.getPw().length());
            hashedPass = new
BigInteger(1,messageDigest.digest()).toString(16);
            if (hashedPass.length() < 32) {
               hashedPass = "0" + hashedPass;
            }

            } catch (NoSuchAlgorithmException e) {
                return "redirect:/registration?error=true";
            }

            user.setPw(hashedPass);

            UserEntity checkUser = null;
            checkUser = userService.getUser(user.getUsrName());

            //check if new user already exists in base
            if((checkUser != null) &&
(checkUser.getUsrName().equalsIgnoreCase(user.getUsrName()))) {
                return "redirect:/registration?error=true";
            }
            try {
            userService.addUser(user);
            } catch (ConstraintViolationException ex) {
                return "redirect:/registration?error=true";
            }
            //redirect to login page on success
            return "redirect:/login";
    }
}
```

Вихідний код сторінки логінізації

```
<%@ page language="java" contentType="text/html; charset=utf8"
         pageEncoding="utf8"%>
<%@taglib uri="http://www.springframework.org/tags" prefix="spring"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 5.0 Transitional//EN"
"http://www.w3.org/TR/html5/loose.dtd">

<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
<meta name="robots" content="noodp,noydir">
<meta name="description" content="Survs is a collaborative tool that
enables you to create online surveys with simplicity and elegance.">
<meta name="author" content="Skell">

<title>Login</title>

        <link rel="stylesheet" type="text/css"
href="resources/style/home2.css">

<style type="text/css">
.ll-content-notification *{
letter-spacing: normal !important;
margin: 0 !important;
    padding: 0 !important;
background: none !important;
border: 0 !important;
float: none !important;
text-align: left !important;
text-decoration: none !important;
 font: normal 15px 'Lucida Grande', 'Lucida Sans Unicode', Lucida,
Arial, Helvetica, sans-serif !important;
}

.ll-content-notification-header-pic img{
     border: 0 !important;
     padding: 0 !important;
     margin: 0 !important;
     line-height: 1px !important;
}

</style>

</head>
```

```
    <body class="small login">

    <span style="float: right"><a href="?lang=en">en</a>|<a
href="?lang=ru">ru</a></span>

    <div id="header">
          <div class="wrapper">
               <h1><a href="https://www.survs.com/">Survs –
Collaborative Online Survey Tool, Web Survey Software</a></h1>
          </div>
     </div>

    <div id="container">
     <div id="content">
          <div class="wrapper">
               <c:if test="${not empty param.error}">
                   <font color="red"> <spring:message
code="label.loginerror" />
                    :
${sessionScope["SPRING_SECURITY_LAST_EXCEPTION"].message}
                   </font>
               </c:if>
                    <form name="login" method="post" action="<c:url
value="/j_spring_security_check" />">
                    <input value="false" name="returning"
id="returning" type="hidden">
                       <table class="login" style="margin-left:
auto; margin-right: auto;">
                            <tbody>
                      <tr>
                          <td>
                              <p class="mtop0 mbottom025">
                              <strong>
                                  <label for="email">
                                      <spring:message
code="label.login"/>
                                  </label>
                              </strong>
                          </p>
                              <input id="email" tabindex="1"
class="inputtext" name="j_username" type="text">
                          </td>
                      </tr>
                              <tr>
                                   <td>
                              <p class="mtop05 mbottom025">
                                  <strong>
                                      <label
for="password">Password</label>
                                  </strong>
```

```
                                    </p>
                                    <input tabindex="2" class="inputtext"
name="j_password" id="password" type="password">
                                </td>
                                    </tr>

                                    <tr>
                                          <td>
                                          <p class="mtop025 mbottom0
smalltxt">
                                    <a
href="https://www.survs.com/app/wa/Password/forgot">Forgot
password?</a>
                                                <span style="padding: 0
3px;">|</span>
                                    <a href="<c:url value="/registration"
/>"><spring:message code="label.registration" /></a>
                                          </p>
                                </td>
                                    </tr>
                            <tr>
                                <td>
                                    <p class="mtop025 mbottom0 smalltxt">
                                        <input name="remember"
id="rememberMe" tabindex="3" type="checkbox"> <label
                                            for="rememberMe">Remember
me</label>
                                    </p>
                                </td>
                            </tr>

                            <tr>
                                <td colspan="2" align="right">
                                <input class="public-button" tabindex="4"
type="submit" value="<spring:message code="label.login" />" />
                                </td>
                                    </tr>
                                  </tbody>
                        </table>
                          </form>
                </div> <!-- wrapper -->
                </div> <!-- content -->
          </div> <!-- container -->

        <div id="footer">
          <div class="wrapper">
                <p>
                      <a href="http://blog.survs.com/"><span>Blog</span></a>
<span class="bar">|</span>
```

```
                <a
href="https://www.survs.com/press/"><span>Press</span></a> <span
class="bar">|</span>
                <a
href="https://www.survs.com/privacy/"><span>Privacy</span></a> <span
class="bar">|</span>
                <a
href="https://www.survs.com/tos/"><span>Terms</span></a> <span
class="bar">|</span>
                <a
href="http://www.twitter.com/survs"><span>Twitter</span></a> <span
class="bar">|</span>
                <a
href="https://www.survs.com/help/"><span>Help</span></a> <span
class="bar">|</span>
                <a href="https://www.survs.com/contact/">Contact</a>
           </p>
           <p id="copyright">
                Copyright © 2014 <a
href="http://www.enoughpepper.com/">Enough Pepper Lda</a>. All rights
reserved.
           </p>
      </div>
</div>
</body>
</html>
```

ДОДАТОК В (ОБОВ'ЯЗКОВИЙ)

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА WEB-РЕСУРСУ КОМЕРЦІЙНОГО БАНКУ

Виконав: студент 4-го курсу,
групи 1КН-22м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
_____ Стукаленко Р.В.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
_____ Денисюк В.О.
(прізвище та ініціали)

«03» червня 2025 р.

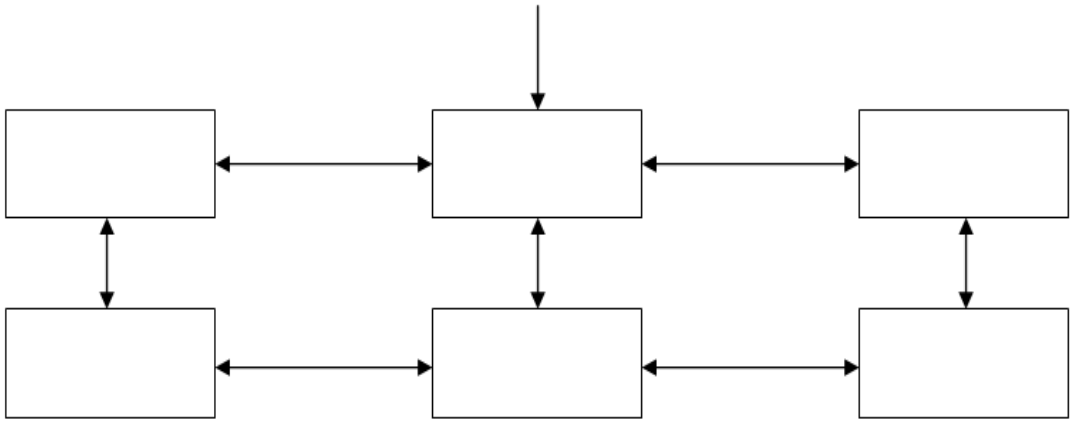

Рисунок В.1 – Гратчаста структура веб-сайту

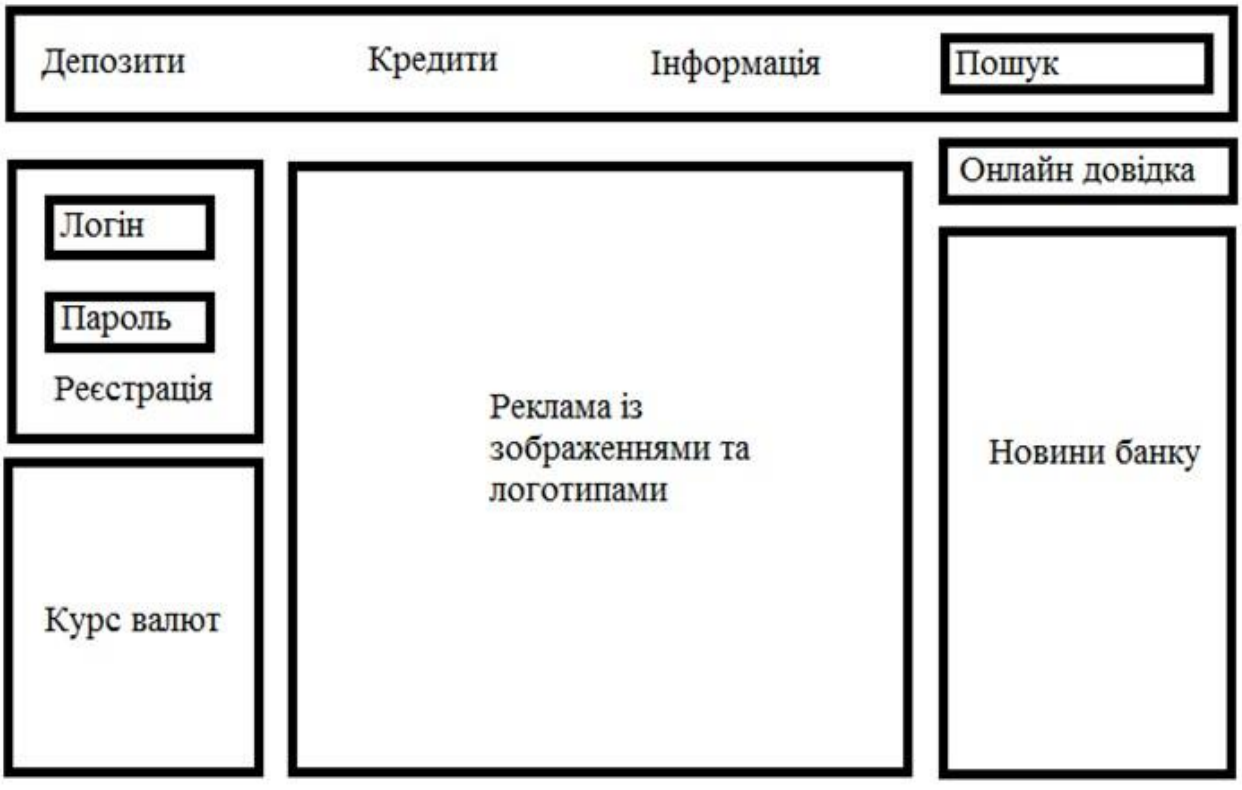

Рисунок В.2 – Структура головної сторінки сайту

Рисунок В.3 – Коло Йоганнеса Іттена

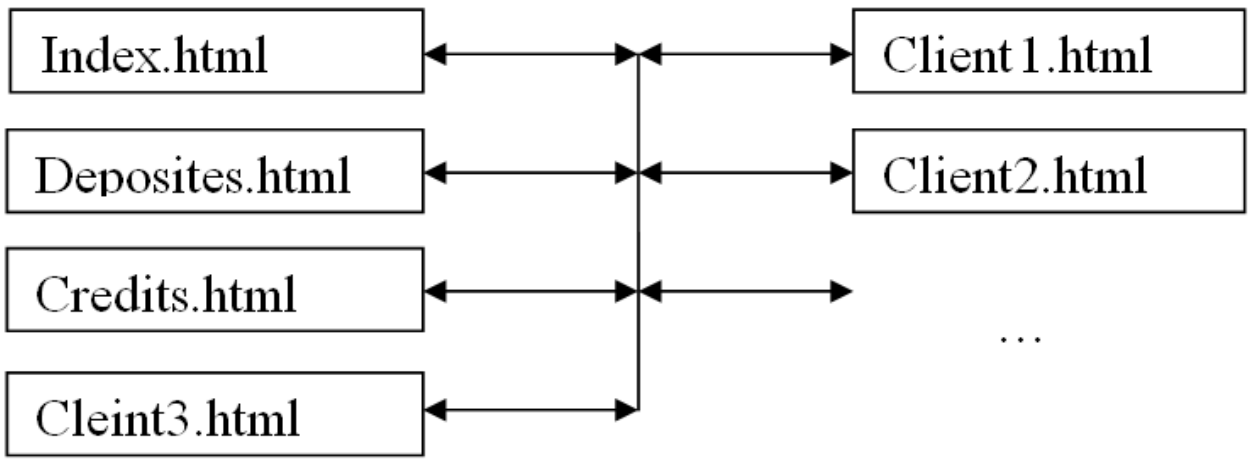

Рисунок В.4 – Структура веб-сайту «Комерційний банк»

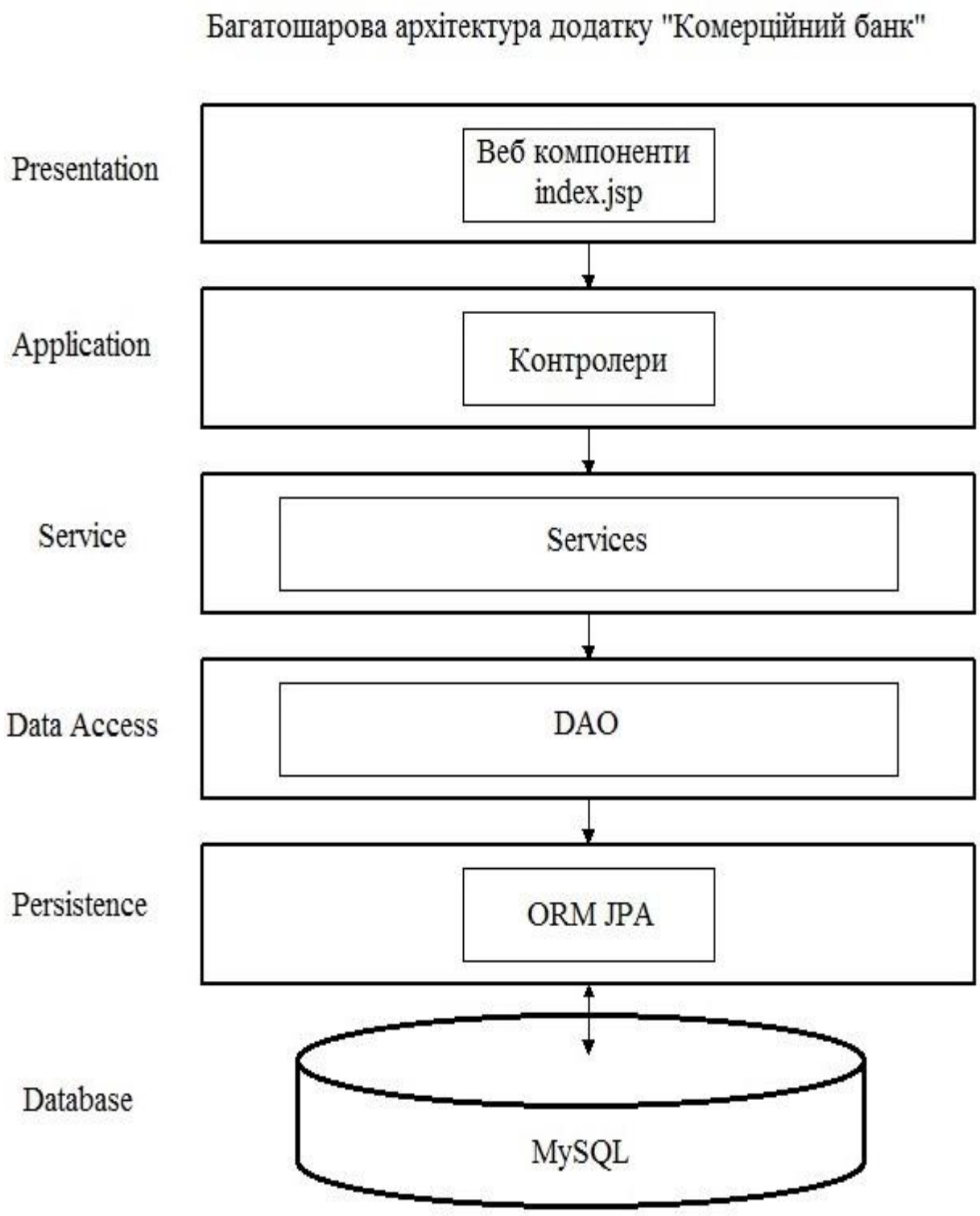

Рисунок В.5 – Ієрархічне зображення веб-додатку

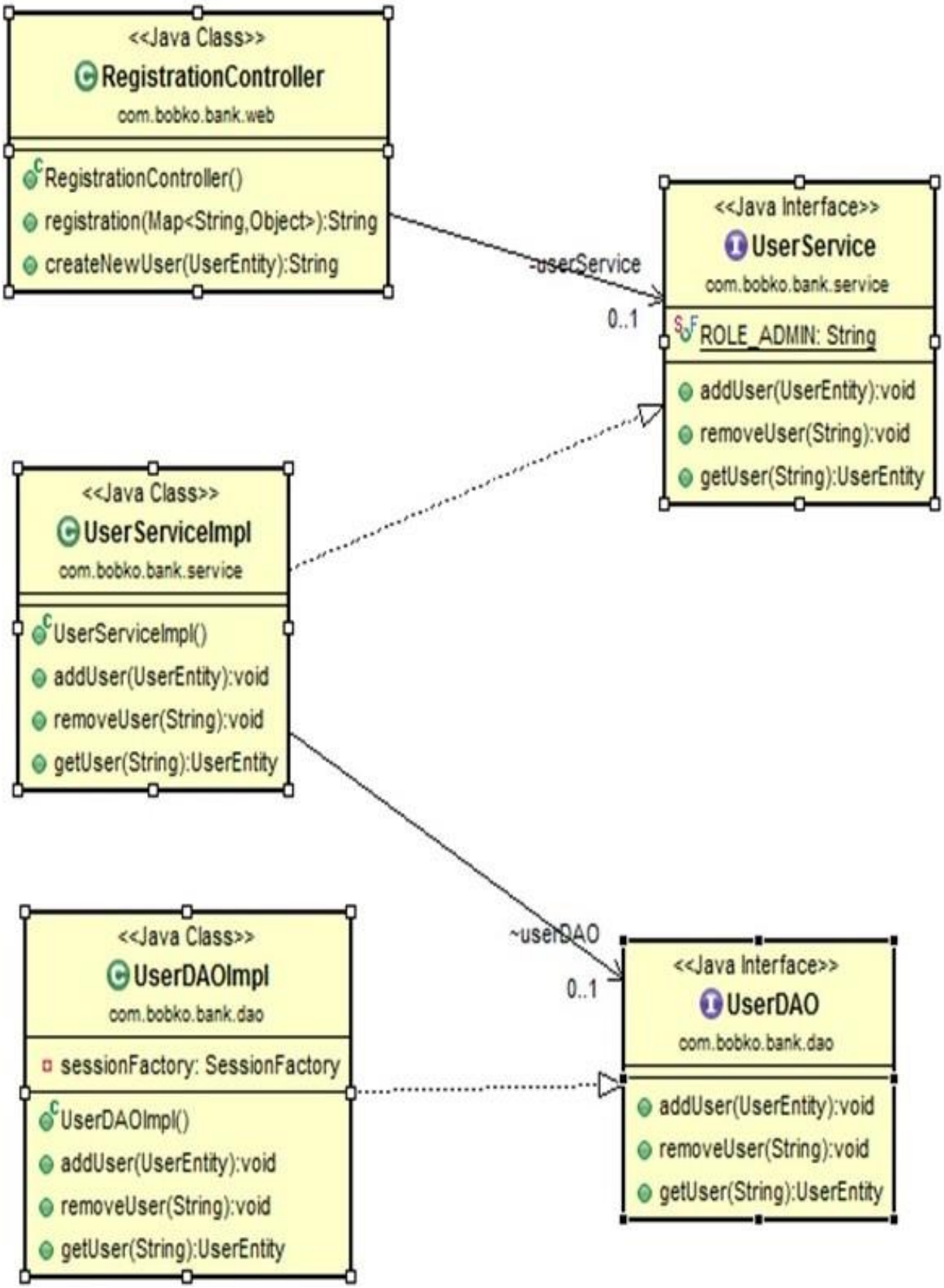

Рисунок В.6 – UML діаграма гілки для роботи з користувачем

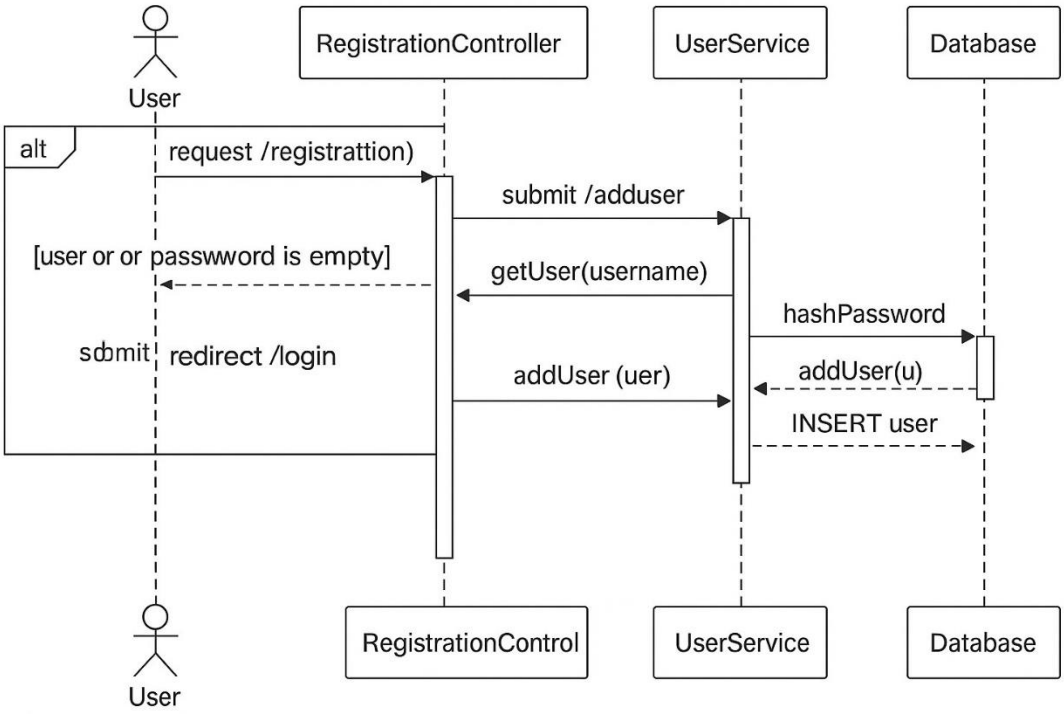

Рисунок В.7 - UML-діаграма послідовності для процесу реєстрації користувача

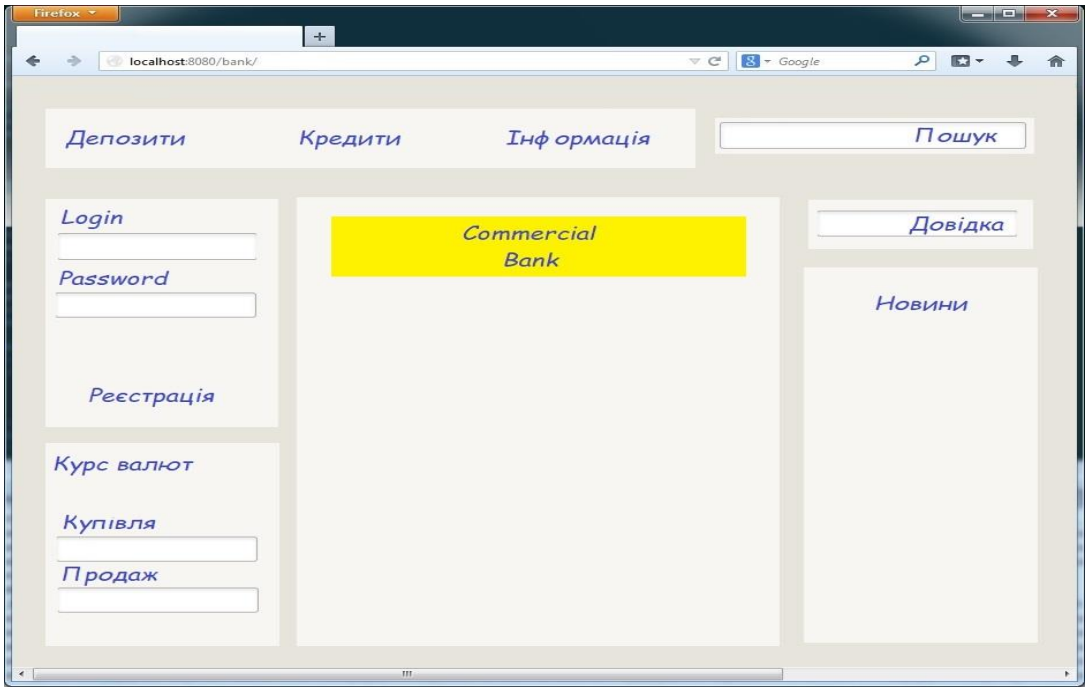

Рисунок В.8 - Приклад головної сторінки комерційного банку.

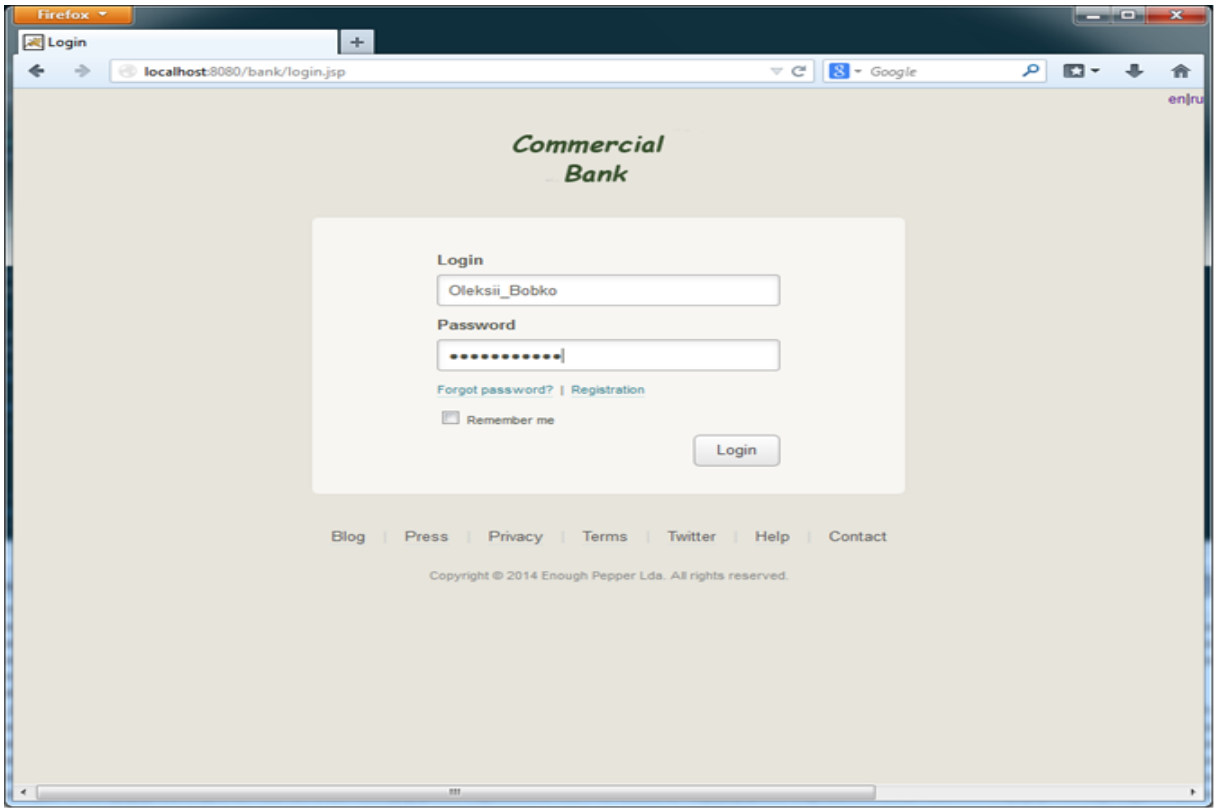

Рисунок В. 9 - Зовнішній вигляд сторінки логінізації

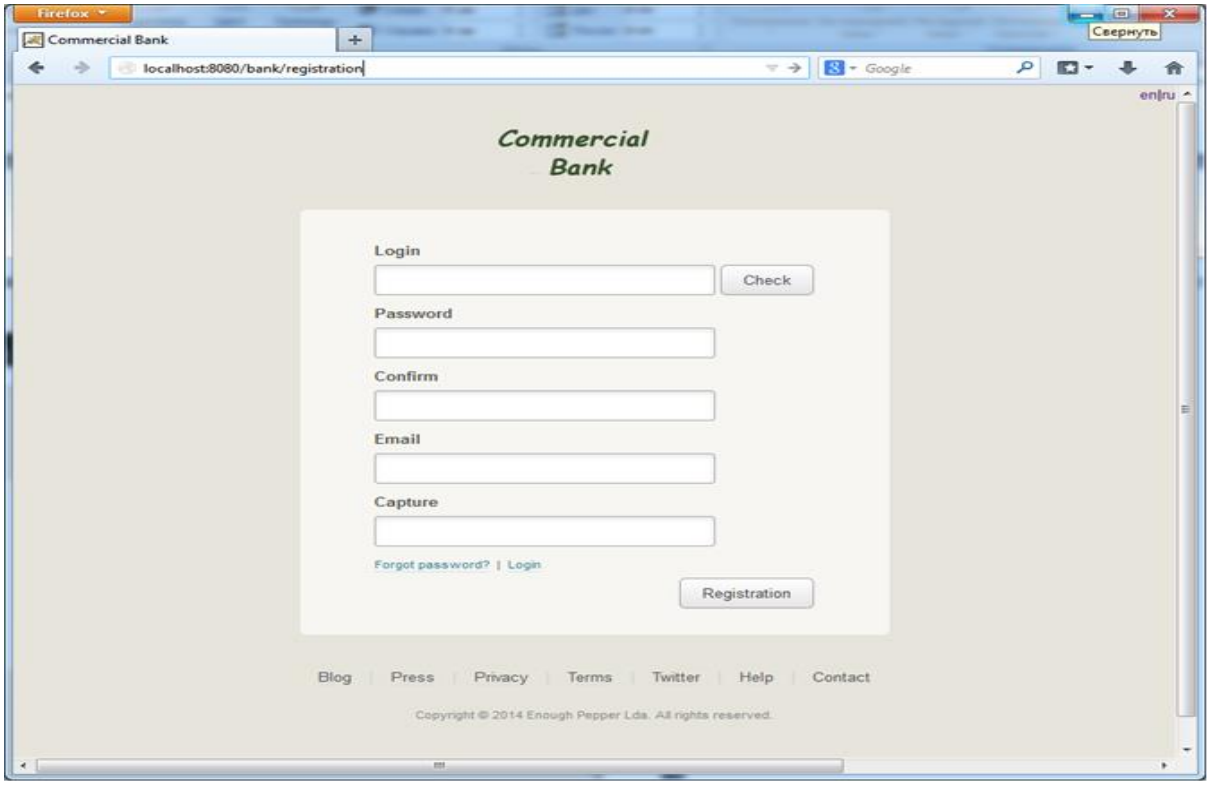

Рисунок В. 10 - Зовнішній вигляд сторінки реєстрації

ДОДАТОК Г (ДОВІДНИКОВИЙ)
ІНСТРУКЦІЯ КОРИСТУВАЧА

Для розвертання web-ресурсу на власному сервері або локальному комп’ютері потрібно виконати таку послідовність дій:

1) Скопіювати файли проекту в директорію.

2) Виконати команду composer install.

3) Виконати команду npm i.

4) Скопіювати файл .env.example та змінити його назву на .env

5) Запустити команду php artisan key:generate

6) Запустити команду php artisan storage:link

7) Створити базу даних

8) Заповнити в файлі .env параметри APP_URL, DB_DATABASE, DB_USERNAME та DB_PASSWORD

9) Виконати команду php artisan migrate

10) Перейти за посиланням «APP_URL/panel»

Для реєстрації в панелі керування потрібно перейти за відповідним посиланням /panel/register або натиснути кнопку «не маю акаунту» на сторінці авторизації.

Зовнішні вигляд сторінки реєстрації наведено у додатку Г.

Для реєстрації потрібно буде ввести такі дані:

− Пошта,

− Ім’я,

− Пароль.

Для авторизації варто використовувати форму на сторінці /panel/login (додаток Г).

Поля для авторизації:

- Пошта

- Пароль