НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»
на тему: «Веб-сервіс для аналізу транзакцій інтернет банку за допомогою методів Big data**»**

Виконав
студент IV курсу, групи КП-51
Страмоусов Євген Дмитрович _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент,
А.В. Гречко _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н.,
Онай М.В. _____

Рецензент:
доц. каф. ММСА ІПСА, доц., к.т.н.
Замятін С.В. _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2019 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра прикладної математики

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – 6.050103 «Програмна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ І.А. Дичка

«___»_____ 2018 р.

ЗАВДАННЯ

на дипломний проект студенту

Страмоусова Євгена Дмитровича

1. Тема роботи «Веб-сервіс для аналізу транзакцій інтернет банку за допомогою методів Big data», керівник роботи Гречко Анастасія Валеріївна, к.т.н., доцент, затверджені наказом по університету від «22» травня 2019 р. №1331-С

2. Термін подання студентом роботи «21» червня 2019 р.

3. Вихідні дані до роботи:
   - відкриті дані результатів транзакцій “Хоум кредит” за 2014 р.

4. Зміст роботи:
   - аналіз методів Big Data;
   - вибір методу побудови моделі;
   - побудова веб-додатка для обробки транзакцій;
   - перевірка моделі для обробки даних БД.

5. Перелік обов’язкового ілюстративного матеріалу:
   - алгоритм обробки;
   - статистичні характеристики динаміки зміни кількості транзакцій;

– 6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2018 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вивчення літератури за тематикою роботи та збір даних	16.11.2018	
2.	Проведення порівняльного аналізу методів математичної статистики	09.12.2018	
3.	Підготовка матеріалів першого розділу дипломної роботи	30.12.2018	
4.	Розробка математичної моделі для обробки даних	16.01.2019	
5.	Підготовка матеріалів другого розділу дипломної роботи	10.02.2018	
6.	Підготовка тез доповіді на конференцію	20.02.2019	
7.	Підготовка матеріалів третього розділу дипломної роботи	10.03.2019	
8.	Підготовка матеріалів четвертого розділу дипломної роботи	11.04.2019	
9.	Підготовка графічної частини дипломної роботи	19.05.2019	
10.	Оформлення дипломної роботи	26.05.2019	

Студент Страмоусов Є.Д.

Керівник роботи Гречко А.В.

АНОТАЦІЯ

Об’єктом розробки є повноцінний веб-сервіс для аналізу транзакцій інтернет банку, що дозволяє виконувати певні операції користувачам.

Метою роботи є розробка веб-сервісу аналізу транзакцій банку та програми управління цим сервісом.

Програма дозволяє автоматично перевірити базу даних банка у форматі .xlsx. В програмі передбачені механізми захисту від помилок. В процесі розробки була використана мова програмування C# з використанням ASP .NET та ClosedXML. База даних побудована на MySQL Server. Для стилізація використано CSS, фреймворк Bootstrap. В ході виконання дипломного проекту проведено аналіз методів побудови існуючих веб-сервісів для аналізування транзакцій та розроблена структура бази даних для зберігання інформації.

АННОТАЦИЯ

Объектом разработки является полноценный веб-сервис для анализа транзакций интернет банка, позволяет выполнять определенные операции пользователям.

Целью работы является разработка веб-сервиса анализа транзакций банка и программы управления этим сервисом.

Программа позволяет автоматически проверить базу данных банка в формате .xlsx. В программе предусмотрены механизмы защиты от ошибок. В процессе разработки была использована язык программирования C # с использованием ASP .NET и ClosedXML. База данных построена на MySQL Server. Для стилизация использовано CSS, фреймворк Bootstrap. В ходе выполнения дипломного проекта.

Проведен анализ методов построения существующих веб-сервисов для анализа транзакций и разработана структура базы данных для хранения информации.

ABSTRACT

The object of development is a full-fledged web service for analyzing online banking transactions, which allows users to perform certain operations.

The aim of the work is to develop a web service for the analysis of bank transactions and a program for managing this service.

The program allows you to automatically check the database of the bank in the .xlsx format. The program provides mechanisms for protection against errors. In the development process was used C # programming language using ASP .NET and ClosedXML. The database is built on MySQL Server. For styling used CSS framework Bootstrap. During the implementation of the graduation project.

The analysis of methods for constructing existing web services for analyzing transactions was conducted and a database structure for storing information was developed.

ДП.045440-01-90 Веб-застосунок для аналізу транзакцій Інтернет-банку на основі технологій Big Data. Відомість проекту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проекту		
ДП.045440-02-91	Веб-застосунок для аналізу	5	
	транзакцій інтернет банку.		
	Технічне завдання		
ДП.045440-03-81	Веб-застосунок для аналізу	70	
	транзакцій інтернет банку.		
	Пояснювальна записка		
ДП.045440-04-51	Веб-застосунок для аналізу	4	
	транзакцій інтернет банку.		
	Програма та методика		
	тестування		
ДП.045440-05-34	Веб-застосунок для аналізу	10	
	транзакцій інтернет банку.		
	Керівництво користувача		
ДП.045440-06-99	Веб-застосунок для аналізу	1	
	транзакцій інтернет банку.		
	Авторизація користувача.		
	Схема алгоритму		
ДП.045440-07-99	Веб-застосунок аналізу для	1	
	транзакцій інтернет банку.		
	Схема роботи програмної		
	системи		
ДП.045440-08-98	Веб-застосунок аналізу для	1	
	транзакцій інтерет банку.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

Веб-застосунок для аналізу транзакцій інтернет банку на основі технологій Big data

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проекту:

_____ А.В. Гречко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____Є.Д.Страмоусов

2018

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Web- сервіс для аналізу транзакцій інтернет банку.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання в якості інформаційного забезпечення роботи статистичного відділу банку.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Web-ресурс повинен забезпечувати такі основні функції:

1) можливість динамічного оновлення вмісту web-сторінок;
2) підтримка доступу до бази даних.
3) можливість розширення структури web-ресурсу;
4) підтримка стандарту SCORM;

Розробку виконати на платформі Microsoft .Net з використанням технології ASP.NET AJAX.

Додаткові вимоги:

1) наявність динамічного меню;

2) наявність анімованих кнопок;

3) наявність на web-сторінках місця для розміщення логотипів партнерських організацій;

4) дизайн сторінок з використанням в якості базових білого та синьо-зеленого кольорів.

5. ВИМОГИ ДО ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

1) пояснювальна записка;

2) програма та методика тестування;

3) керівництво користувача;

4) креслення:

- «Синхронізація даних. Схема роботи програмної системи».

6. ЕТАПИ ПРОЕКТУВАННЯ

Вивчення літератури за тематикою роботи……..14.11.2018

Розроблення та узгодження технічного завдання……....28.11.2018

Розроблення структури web-ресурсу…….....15.12.2018

Розроблення дизайну сторінок та графічних елементів……..03.02.2018

Програмна реалізація web-ресурсу……....17.03.2019

Тестування web-ресурсу…….....03.04.2019

Підготовка матеріалів текстової частини проекту……..….28.04.2019

Підготовка матеріалів графічної частини проекту……...…12.05.2019

Оформлення технічної документації проекту……..…25.05.2019

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ВЕБ-СЕРВІС ДЛЯ АНАЛІЗУ ТРАНЗАКЦІЙ ІНТЕРНЕТ БАНКУ НА ОСНОВІ ТЕХНОЛОГІЙ BIG DATA

Пояснювальна записка

ДП.045450-03-81

"ПОГОДЖЕНО"

Керівник проекту:

_____ Гречко А.В.

Нормоконтроль:

_____ Онай М.В.

Виконавець:

_____Страмоусов Є.Д.

2019

ЗМІСТ

ВСТУП

За словами провідного аналітика кредитного рейтингового агентства "Експерт РА» Олексія Сучкова, більшість роздрібних банків вже можуть відстежувати і запобігати виконанню нетипових транзакцій, відсіваючи їх у випадку спрацювання певних індикаторів, наприклад, невластивого клієнту IP або часу платежу. При цьому банк отримує і зберігає величезну кількість інформації про клієнта від його доходу і середнього чека в магазині до часу здійснення операцій по карті і їх локації. З усіх цих даних складається поведінкова картина клієнта, проаналізувавши яку банк може не тільки формувати індивідуальні пропозиції, але й знижувати свої ризики.

З інформації про карткові транзакції банки бачать, де клієнти витрачають свої гроші. За цими даними банки можуть розуміти, що відбувається в житті клієнта: від незначних речей на зразок улюблених місць для сніданку до важливих подій, наприклад, відпустки, ремонту, зміни машини, народження дітей тощо. Ці знання дозволяють дуже точково зорієнтувати комунікації з клієнтами.

Новий тренд в використанні великих даних – це їх аналіз з метою надання клієнту рекомендацій або порад. Така комунікація може бути не пов'язана з продажем банківського продукту, але за рахунок точкового таргетування приносить клієнту додаткову цінність (а банку дозволяє знизити витрати на продаж конкретного продукту). Наприклад, банк, знаючи, що клієнт збирається у відпустку і що у нього є абонемент у фітнес-клуб, може порадити йому заморозити абонемент на час поїздки, щоб не втрачати гроші. Поради від банку підвищують лояльність клієнта, його залученість у взаємини з банком. В результаті клієнти

набагато рідше змінюють банк, сприймаючи його як фінансового помічника, а банку стає простіше продати лояльному клієнту маржинальні продукти.

За даними компанії IBS, до 2003 року світ накопичив 5 ексабайт даних (1 ЕБ = 1 млрд гігабайтів). До 2008 року цей обсяг зріс до 0,18 зеттабайт (1 ЗБ = 1024 ексабайта), до 2011 року – до 1,76 зеттабайт, до 2013 року – до 4,4 зеттабайт. У травні 2015 року глобальна кількість даних перевищила 6,5 зеттабайт (докладніше).

До 2020 року, за прогнозами, людство сформує 40-44 зеттабайт інформації, а до 2025 року ця цифра виросте в 10 разів, говориться в доповіді The Data Age 2025, підготовленій аналітиками компанії IDC. У доповіді наголошується, що більшу частину даних генерувати будуть самі підприємства, а не звичайні споживачі.

Аналітики дослідження вважають, що дані стануть життєво-важливим активом, а безпека – критично важливим фундаментом в житті та в банківській діяльності зокрема. Також автори роботи впевнені, що технологія змінить економічний ландшафт, а звичайний користувач буде комунікувати з підключеними пристроями близько 4800 разів в день.

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

БД – база даних;

ОС – операційна система;

UML – Unified Modeling Language (уніфікована мова моделювання);

ORM – Object-Relation Mapping (об’єктно-реляційне відображення);

HTML – Hyper Text Markup Language (мова гіпертекстової розмітки).

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Пошук існуючих систем для аналізу транзакцій інтернет-банку дав такі результати:

Перемігши в тендері, оголошеному АКБ «ХФБ Україна» на розробку і впровадження системи автоматизації фінансового моніторингу, фахівці Finport Technologies менш ніж за рік реалізували цей проект. Суть розробки на замовлення полягала в створенні незалежної від ОДБ (операційний день банку) системи, призначеної для автоматизації процедур не тільки обов'язкового, а й внутрішнього фінансового моніторингу. Фахівці Finport Technologies створили унікальний програмний продукт, що задовольнив всі вимоги та побажання замовника: після закінчення дослідної експлуатації в квітні цього року, система почала функціонувати в повному циклі.

Розроблена система надала в розпорядження фахівців банку «ХФБ Україна» ефективний інструментарій для аналізу і дослідження накопичуваної бази даних банківських операцій. Відомості про клієнтів і інша інформація збирається в спеціальні досьє з метою виявлення операцій, що підлягають обов'язковому фінансовому моніторингу. Для реалізації цих функцій до складу системи входить спеціальний програмний модуль, який забезпечує створення і кодування файлів інформаційного обміну для передачі їх в Держкомітет з фінансового моніторингу. У той же час, система збирає інформацію про «підозрілих» операціях з точки зору правил внутрішнього фінансового моніторингу – для виявлення клієнтів з сумнівним поведінкою і для подальшого аналізу їх діяльності. Система «FinMonitor», надаючи механізми реєстрації та контролю, забезпечує управління процесом розслідувань

таких підозрілих ситуацій. В системі використовується скоринг-механізм, що використовується для обробки даних і виявлення серед загальної кількості клієнтів тих, хто веде підозрілу діяльність. Крім цього, завдяки наявності в системі інструментів для аналізу діяльності клієнтів, вона може бути використана іншими службами банку, наприклад, відділом кредитування, службою безпеки та ін.

Система автоматизації фінансового моніторингу «AML Insighter» спрямована на реалізацію не тільки положень обов'язкового, а й внутрішнього фінансового моніторингу, та дозволяє вирішити ряд наступних завдань:

- забезпечення дотримання регуляторних вимог;
- відбір і аналіз фінансових операцій
- обов'язковий і внутрішній моніторинг клієнтів і транзакцій
- реалізація політики «Знай свого клієнта» (KYC)
- квартальний аналіз діяльності клієнтів і моніторинг підозрілої активності (на основі правил і скоринг (ризик) механізму)
- призначення рівня ризику клієнта
- призначення рівня ризику відділенню банку
- призначення рівня ризику банку в цілому
- менеджмент інцидентів
- складання статистичних звітів

«Альфа-Банк» взявся за великі дані в 2013 році. Банк використовує технології для аналізу соцмереж і поведінки користувачів сайту, оцінки кредитоспроможності, прогнозування відтоку клієнтів, персоналізації контенту і вторинних продажів. Для цього він працює з платформами

зберігання і обробки Oracle Exadata, Oracle Big data Appliance і фреймворком Hadoop.

«Тінькофф-банк» за допомогою EMC Greenplum, SAS Visual Analytics і Hadoop управляє ризиками, аналізує потреби потенційних і існуючих клієнтіыв. Великі дані задіяні також в скорингу, маркетингу та продажу.

1.1. Постановка завдання

Розробити Веб-застосунок для аналізу інтернет банку.

Досягнення поставленої мети передбачає вирішення наступних завдань:

1) проведення збору даних по існуючим методам та , виявлення сильних та слабких сторін розглянутих рішень;

2) визначення методу BiData для подальшої реалізації в рамках розроблюваної системи ;

3) розробка застосування, що здійснює аналіз транзакцій.

1.2. Аналіз вимог до функціональності програмних засобів

В процесі аналізу вимог до системи було виділено такі функціональні вимоги:

1. Система може скачувати файл, над яким працює користувач.
2. Система дозволяє Адміністратору зміняти методу аналізу БД.
3. Система надає Адміністратору можливість проведення покрокової класифікації даних

2. АНАЛІЗ ТА ОБГРУНТУВАННЯ ВИБІРУ МОВИ ПРОГРАМУВАННЯ ТА ТЕХНОЛОГІЇ РОЗРОБЛЕННЯ ЗАСТОСУВАННЯ

2.1. Обгрунтування вибору веб-застосування в якості типу програмного забезпечення

Розроблювана система являється аналізатором транзакцій інтернет банку.

Клієнтська частина системи для взаємодії з користувачем може бути реалізована у вигляді веб-застосування або десктопної програми. Переваги та недоліки кожного варіанту наведено в табл. 1.1.

Вибір веб-застосування в якості типу програмного забезпечення аргументується його перевагами над десктопними програмами з врахуванням вимог, пред’явлених до розроблюваної системи. Доступ до системи стає можливим із будь-якого пристрою з веб-браузером та підключенням до мережі. Відпадає необхідність розробки клієнтів для різних операційних систем та пристроїв. Також можна відзначити гнучкість при зміні інтерфейсу користувача та додаванні нового функціоналу.

Вибір такого типу програмного забезпечення спрощує розроблення, встановлення, підтримку та вдосконалення системи.

Таблиця 1.1

Критерій	Веб-застосування	Десктопна програма
Кросплатформеність	Працює на всіх веб-браузерах з підключенням до мережі .	Необхідий перенос програми для кожної конкретної ОС
Швидкодія	Можлива погана оптимізація та швидкодія для певних девайсів	Оптимізована для кожного конкретного девайсу чи ОС
Вартість розробки	Незначніший ніж десктопні програми	Дорожча ніж веб-застосування
Інсталяція	Не потребує установки на стороні користувача	Потребує встановлення на стороні користувача
Інтерфейс користувача	Уніфікований інтерфейс	Зазвичай привичний для користувача стандартний інтерфейс відповідної ОС

2.2. Порівняння мов програмування C# та java

2.2.1. Мова C#

Аналіз платформи C# для розробки веб-застосувань

Є багато важливих особливостей мови C #, які роблять його більш корисним і унікальним порівняно з іншими мовами.

- Швидка швидкість.
- Простий.
- Об'єктно орієнтований.
- Сучасна мова програмування.
- Введіть безпечний тип.
- Оперативна сумісність.
- Масштабована і оновлювана.
- Структурована мова програмування.
- Багата бібліотека.
- Компонент орієнтований.
- Швидка швидкість.

Мова C # є дуже швидкою, її компіляція та час виконання занадто швидко.

C # – це проста мова. Це дає структурований підхід до розв'язання проблеми на частини. Він має багаті набори бібліотечних функцій і типів даних. Код мови C # не вимагає файлів заголовків. Його код написаний вбудовано.

Мова C # – це об'єктно-орієнтована мова програмування. ООП полегшує розробку та обслуговування, порівняно з мовою програмування, орієнтованої на процедури.

Керувати занадто складно, якщо код зростає з ростом розміру проекту. Програмування C # підтримує інкапсуляцію даних, успадкування, поліморфізм, інтерфейси.

Мова C # є однією з сучасних мов програмування, оскільки вона ґрунтується на поточній тенденції. Вона дуже проста, потужна для створення масштабованих, сумісних і надійних програм.

Мова C # є безпечним для типу кодом, який може отримати доступ лише до пам'яті і має дозвіл на виконання. Тому це покращує безпеку програми.

На мові C # не можна виконувати небезпечні листи, такі як перетворення double на булево. Її типи значень (примітивні типи) ініціалізуються до нулів, а типи посилання (об'єкти та класи) автоматично ініціалізуються компілятором.

Операційна сумісність – це процес, який дозволяє програмам C # робити майже все, що може виконувати рідне додаток C ++. Коротше кажучи, сумісність мови - це здатність коду взаємодіяти з кодом, який написаний з використанням іншої мови програмування. Це може допомогти в максимальному використанні коду і, отже, підвищити ефективність процесу розробки.

Мова C # надає підтримку для використання COM-об'єктів, незалежно від того, якою мовою їх було використано. Однак він також підтримує спеціальну функцію, яка дозволяє програмі викликати будь-який рідний API.

Мова C # – це автоматична масштабована і оновлювана мова програмування. Для оновлення програми потрібно видалити старі файли та оновити їх новим.

2.2.2. Мова Python

Мова Python є потужною інтерпретованою об’єктно-орієнтованою мовою програмування із динамічною семантикою, яка використовується в різних прикладних областях.

Ключові особливості мови [3, 4]:

1. Чіткий та читабельний синтаксис.
2. Переносимість програм.
3. Сильні можливості самоаналізу.
4. Модульність, підтримка ієрархічних пакетів.
5. Високорівневі динамічні типи даних.
6. Широкий набір стандартних бібліотек і сторонніх модулів практично для будь-яких задач.
7. Можливість використання Python в діалоговому режимі (що є зручно для експериментування та розв’язання простих задач).
8. Розширюваність та використання модулів, написаних на C, C++ (або Java для Jython або. NET мов для IronPython).
9. Висока швидкість.

Елегантний синтаксис, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.

Python може інтегруватися з COM, .NET і CORBA об'єктами.

Дизайн мови Python побудований навколо об'єктно-орієнтованої моделі програмування. Реалізація ООП в Python є елегантною,

потужною та добре продуманою, але разом з тим, достатньо специфічною в порівнянні з іншими об'єктно-орієнтованими мовами.

Можливості та особливості:

1. Класи є одночасно об'єктами з усіма нижче наведеними можливостями.
2. Успадкування, в тому числі множинне.
3. Поліморфізм (всі функції віртуальні).
4. Інкапсуляція (два рівні – загальнодоступні та приховані методи і поля). Особливість – приховані члени доступні для використання та помічені як приховані лише особливими іменами.
5. Спеціальні методи, керуючі життєвим циклом об'єкта: конструктори, деструктори, розподільники пам'яті.
6. Перевантаження операторів (усіх, крім is, '.', '=' і символьних логічних).
7. Властивості (імітація поля за допомогою функцій).
8. Управління доступу до полів (емуляція полів і методів, частковий доступ тощо).
9. Методи для управління найпоширенішими операціями (істинносне значення, len(), глибоке копіювання, серіалізація, ітерація по об'єкту, та ін).
10. Метапрограмування (управління створенням класів, тригери на створення класів, та ін).
11. Повна інтроспекція.
12. Класові та статичні методи, класові поля.
13. Класи, вкладені у функції та інші класи.

Python підтримує парадигму функціонального програмування, зокрема:

1. Функція є об'єктом.
2. Функції вищих порядків.
3. Рекурсія.
4. Розвинена обробка списків (спискові вирази, операції над послідовностями, ітератори).
5. Аналог замикань.
6. Часткове застосування функції.
7. Можливість реалізації інших засобів на самій мові (наприклад, каррінг).

Програмне забезпечення (застосування чи бібліотека) на Python оформлюється у вигляді модулів, які у свою чергу можуть бути зібрані в пакети. Модулі можуть розташовуватися як у каталогах, так і в ZIP-архівах. Модулі можуть бути двох типів за своїм походженням: модулі, написані на «чистому» Python, і модулі розширення (extension modules), написані на інших мовах програмування. Наприклад, в стандартній бібліотеці є «чистий» модуль pickle і його аналог на C: cPickle. Модуль оформляється у вигляді окремого файлу, а пакет – у вигляді окремого каталогу. Підключення модуля до програми здійснюється оператором import. Після імпорту модуль представлений окремим об'єктом, що дає доступ до простору імен модуля. У ході виконання програми модуль можна перезавантажити функцією reload().

Python підтримує повну інтроспекцію часу виконання. Це означає, що для будь-якого об'єкта можна отримати всю інформацію про його внутрішню структуру.

Застосування інтроспекції (метапрограмування) є важливою частиною того, що називають «pythonic style», і широко застосовується в бібліотеках і фреймворках Python, таких як PyRO, PLY, Cherry, Django та інших, заощаджуючи час програміста, що ними користується.

Багата стандартна бібліотека є однією з привабливих сторін Python. Тут є засоби для роботи з багатьма мережевими протоколами та форматами Інтернету, наприклад, модулі для написання HTTP-серверів та клієнтів, для розбору та створення поштових повідомлень, для роботи з XML, тощо. Набір модулів для роботи з операційної системою дозволяє писати кросплатформні додатки. Існують модулі для роботи з регулярними виразами, текстовими кодуваннями, мультимедійними форматами, криптографічними протоколами, архівами, серіалізацією даних, підтримки юніт-тестування та ін.

Крім стандартної бібліотеки існує безліч бібліотек, що надають інтерфейс до всіх системних викликів на різних платформах; зокрема, на платформі Win32 підтримуються всі виклики Win32 API, а також COM в обсязі не меншому, ніж у Visual Basic або Delphi. Кількість прикладних бібліотек для Python в самих різних областях без перебільшення величезна (веб, бази даних, обробка зображень, обробка тексту, чисельні методи, програми операційної системи і т.д.).

Для Python прийнята специфікація програмного інтерфейсу до баз даних DB-API 2 та розроблено відповідні цієї специфікації пакети для доступу до різних СУБД: PostgreSQL, Oracle, Sybase, Firebird (Interbase), Informix, Microsoft SQL Server, MySQL та sqlite. На платформі Microsoft Windows доступ до БД можливий через ADO (ADOdb). Комерційний пакет mxODBC для доступу до СУБД через ODBC для платформ Windows і UNIX розроблений eGenix. Для Python

написано багато ORM: (SQLObject, SQLAlchemy, Dejavu, Django), виконані програмні каркаси для розробки веб-застосунків (Django, Pylons).

Бібліотека NumPy для роботи з багатовимірними масивами дозволяє досягти продуктивності наукових розрахунків, у порівнянні зі спеціалізованими пакетами. SciPy використовує NumPy і надає доступ до широкого спектру математичних алгоритмів (матрична алгебра – BLAS, БПФ).

WSGI – інтерфейс шлюзу з web-сервером (Python Web Server Gateway Interface).

Python надає простий і зручний програмний інтерфейс C API для написання власних модулів на мовах C та C++. Такий інструмент як SWIG дозволяє майже автоматично отримувати прив'язки для використання C/C++ бібліотек в коді на Python. Можливості цього та інших інструментів варіюються від автоматичної генерації (C/C++/Fortran)-Python інтерфейсів за спеціальними файлів (SWIG, pyste, SIP, pyfort), до надання зручніших API (boost::python, CXX та ін.). Інструмент стандартної бібліотеки ctypes дозволяє програмам Python безпосередньо звертатися до динамічних бібліотек/DLL, написаним на C. Існують модулі, що дозволяють вбудовувати код на C/C++ прямо у вихідні файли Python, створюючи розширення «на льоту» (pyinline, weave). Для підключення математичних функцій, особливо з застосуванням NumPy, наразі офіційно рекомендованим є Cython.

Інший підхід полягає у вбудовуванні інтерпретатора Python у застосунки. Python легко вбудовується в програми на Java, C/C++, Ocaml. Взаємодія Python-застосунків з іншими системами можлива також за допомогою CORBA, XML-RPC, SOAP, COM.

За допомогою Pyrex можлива компіляція Python-подібної мови (додана можливість типізації) у еквівалентний C-код і зв'язування із зовнішніми модулями.

Python та переважна більшість бібліотек до нього безкоштовні й поставляються у вихідних кодах. Більше того, на відміну від багатьох відкритих систем, ліцензія ніяк не обмежує використання Python в комерційних розробках та не накладає ніяких зобов'язань крім вказівки авторських прав.

2.2.3. Відмінності між Java і C #

Java – це об'єктно-орієнтована мова програмування загального призначення, заснований на класах. Розробники можуть використовувати принцип – «пиши один раз, біжи куди завгодно» з Java. Спочатку він був розроблений Джеймсом Гослінгом в Sun Microsystem.

C # є об'єктно-орієнтованим, функціональним, універсальним і компонентно-орієнтованою мовою програмування. Він був розроблений Microsoft з ініціативи .NET з командою розробників на чолі з Андерсом Хейлсберг. Остання версія C # – 7.2, випущена в 2017 році разом з Visual Studio 2017 версії 15.5.

Велика частина синтаксису Java є похідною від C ++, який також заснований на класах і об'єктно-орієнтованих. Java поставляється в формі JDK (Java Development Kit), який включає в себе різні компоненти, необхідні для запуску Java-програми, однак не всі компоненти JDK є обов'язковими для запуску Java. Остання версія – Java 10, випущена в березні 2018 року.

C # використовується для створення різних додатків, він особливо сильний при створенні додатків та ігор для Windows. Веб-розробка також може бути ефективно виконана за допомогою C #, і вона стає все

більш популярною для мобільних розробок. Таким чином, це відмінний вибір для будь-якого програміста, який хоче зайнятися веб-розробкою і розробкою ігор. Існують різні Кросплатформені інструменти, які дозволяють додаткам, написаним на C #, які можна використовувати для мобільних і настільних комп'ютерів.

Вихідна програма Java перетворюється в байт-код компілятором Java, і потім цей скомпільований байт-код може бути виконаний в будь-якій операційній системі, на якій встановлено сумісну JRE (Виконавча Java). Таким чином, вихідний код, колись написаний на Java, може бути запущений на будь-якій платформі, що є одним з найбільших переваг. Просто відповідна JRE повинна бути встановлена в необхідної операційній системі, яку можна завантажити з офіційного сайту Java.

C # поставляється з великою кількістю функцій, тому може бути легко вивчений. Багато складні завдання абстрагуються від мови, тому не потрібно турбуватися про такі проблеми, як управління пам'яттю і збірка сміття при розробці логіки для програми або гри. Це мова високого рівня, який легше читати.

3. ОПИС РОЗРОБЛЕНИХ ПРОГРАМНИХ ЗАСОБІВ

3.1. Вибір методу Big Data

Згідно з опублікованою компанією Oracle білій книзі `Інформаційна архітектура Oracle: керівництво архітектора по великим данним` (Oracle Information Architecture: An Architect's Guide to Big Data), при роботі з великими даними ми підходимо до інформації інакше, ніж при проведенні бізнес-аналізу.

3.1.1. Загальна структура

Програмні засоби реалізовані у вигляді веб-сервера та web-додатку. Структурна схема системи зображена на рис. 3.1

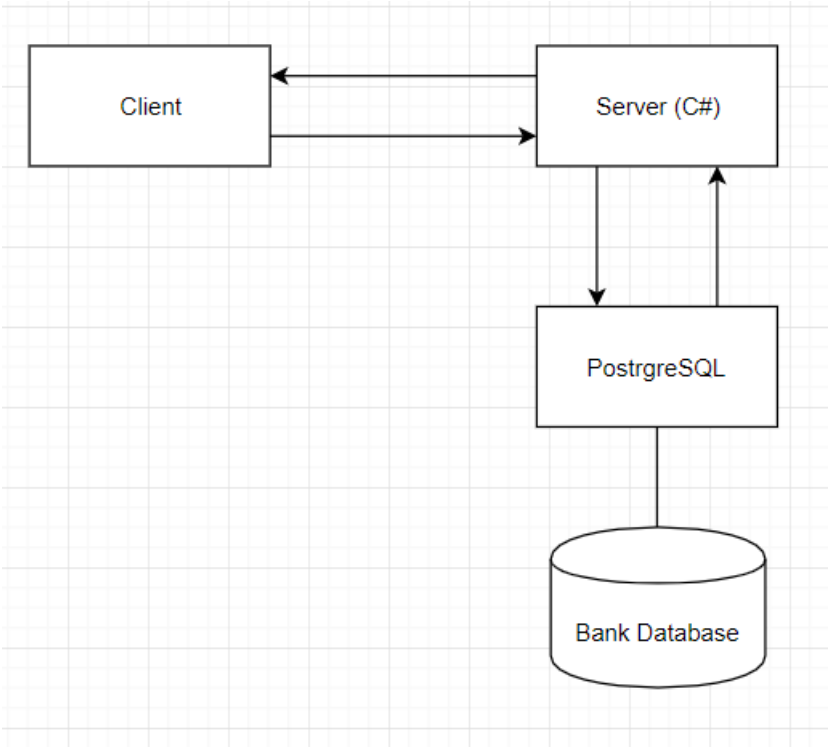


Рис. 3.1. Структурна схема системи

Сервер системи обробляє запити від користувачів та виконує більшість функцій:

– Підключення до бази даних транзакцій.
– Виведення обробленої інформації з БД.
– Формування web-сторінок.

3.1.2. Структура бази даних

На рис. 3.2 представлена діаграма структури бази даних системи.

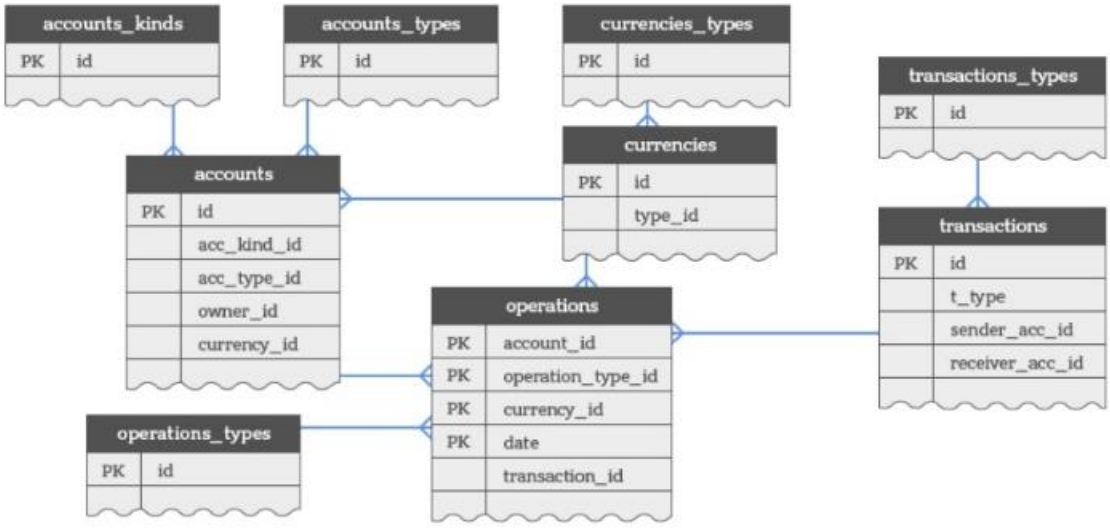


Рис. 3.2. Структура бази даних

Робота з великими даними не схожа на звичайний процес бізнес-аналітики, де просте додавання відомих значень приносить результат: наприклад, результат складання даних про сплачені рахунки стає обсягом продажів за рік. При роботі з великими даними результат виходить в процесі їх очищення шляхом послідовного моделювання: спочатку висувається гіпотеза, будується статистична, візуальна або семантична модель, на її підставі перевіряється вірність висунутої гіпотези і потім висувається наступна. Цей процес вимагає від

дослідника або інтерпретації візуальних значень або складання інтерактивних запитів на основі знань, або розробки адаптивних алгоритмів `машинного обученія`, здатних отримати шуканий результат. Причому час життя такого алгоритму може бути досить коротким.Існує безліч різноманітних методик аналізу масивів даних, в основі яких лежить інструментарій, запозичений з статистики та інформатики (наприклад, машинне навчання). Список не претендує на повноту, проте в ньому відображені найбільш затребувані в різних галузях підходи. При цьому слід розуміти, що дослідники продовжують працювати над створенням нових методик і вдосконаленням існуючих. Крім того, деякі з перерахованих них методик зовсім не обов'язково можуть бути застосовані виключно до великих даними і можуть з успіхом використовуватися для менших за обсягом масивів (наприклад, A / B-тестування, регресійний аналіз). Безумовно, чим більше об'ємний і диверсифіціруемий масив піддається аналізу, тим точніші і релевантні дані вдається отримати на виході.

- A / B testing. Методика, в якій контрольна вибірка по черзі порівнюється з іншими. Таким чином удасться виявити оптимальну комбінацію показників для досягнення, наприклад, найкращою відповідної реакції споживачів на маркетингову пропозицію. Великі дані дозволяють провести величезну кількість ітерацій і таким чином отримати статистично достовірний результат.
- Association rule learning. Набір методик для виявлення взаємозв'язків, тобто асоціативних правил, між змінними величинами в великих масивах даних. Використовується в data mining.

- Classification. Набір методик, які дозволяє передбачити поведінку споживачів в певному сегменті ринку (прийняття рішень про покупку, відтік, обсяг споживання і ін.). Використовується в data mining.
- Cluster analysis. Статистичний метод класифікації об'єктів по групах за рахунок виявлення наперед невідомих загальних ознак. Використовується в data mining.
- Crowdsourcing. Методика збору даних з великої кількості джерел. Data fusion and data integration. Набір методик, який дозволяє аналізувати коментарі користувачів соціальних мереж і зіставляти з результатами продажів в режимі реального часу.
- Data mining. Набір методик, який дозволяє визначити найбільш сприйнятливі для продукту, що просувається або послуги категорії споживачів, виявити особливості найбільш успішних працівників, передбачити поведінкову модель споживачів.
- Ensemble learning. У цьому методі задіюється безліч предикативних моделей за рахунок чого підвищується якість зроблених прогнозів.
- Genetic algorithms. У цій методиці можливі рішення представляють у вигляді `хромосом`, які можуть комбінуватися і мутувати. Як і в процесі природної еволюції, виживає найбільш пристосована особина.
- Machine learning. Напрямок в інформатиці (історично за ним закріпилася назва `штучний інтеллект`), яке має на меті створення алгоритмів самонавчання на основі аналізу емпіричних даних.

- Natural language processing (NLP). Набір запозичених з інформатики та лінгвістики методик розпізнавання природної мови людини.
- Network analysis. Набір методик аналізу зв'язків між вузлами в мережах. Стосовно до соціальних мереж дозволяє аналізувати взаємозв'язку між окремими користувачами, компаніями, спільнотами і т.п.
- Optimization. Набір чисельних методів для редизайну складних систем і процесів для поліпшення одного або декількох показників. Допомагає в прийнятті стратегічних рішень, наприклад, складу виведеної на ринок продуктової лінійки, проведенні інвестиційного аналізу та ін.
- Pattern recognition. Набір методик з елементами самонавчання для передбачення поведінкової моделі споживачів.
- Predictive modeling. Набір методик, які дозволяють створити математичну модель наперед заданого ймовірного сценарію розвитку подій. Наприклад, аналіз бази даних CRM-системи на предмет можливих умов, які підштовхнуть абоненти змінити провайдера.
- Regression. Набір статистичних методів для виявлення закономірності між зміною залежною змінною і однією або декількома незалежними. Часто застосовується для прогнозування і пророкувань. Використовується в data mining.
- Sentiment analysis. В основі методик оцінки настроїв споживачів лежать технології розпізнавання природної мови людини. Вони дозволяють вичленувати із загального інформаційного потоку

повідомлення, пов'язані з цікавлять предметом (наприклад, споживчим продуктом). Далі оцінити полярність судження (позитивне чи негативне), ступінь емоційності та ін.

- Signal processing. Запозичений з радіотехніки набір методик, який має на меті розпізнавання сигналу на тлі шуму і його подальшого аналізу.
- Spatial analysis. Набір частково запозичених з статистики методик аналізу просторових даних – топології місцевості, географічних координат, геометрії об'єктів. Джерелом великих даних в цьому випадку часто виступають геоінформаційні системи (ГІС).
- Statistics. Наука про збір, організації та інтерпретації даних, включаючи розробку опитувальників і проведення експериментів. Статистичні методи часто застосовуються для оціночних суджень про взаємозв'язки між тими чи іншими подіями.
- Supervised learning. Набір заснованих на технологіях машинного навчання методик, які дозволяють виявити функціональні взаємозв'язки в аналізованих масивах даних.
- Simulation. Моделювання поведінки складних систем часто використовується для прогнозування, передбачення і опрацювання різних сценаріїв при плануванні.
- Time series analysis. Набір запозичених з статистики та цифрової обробки сигналів методів аналізу повторюваних з плином часу послідовностей даних. Одні з очевидних застосувань – відстеження ринку цінних паперів або захворюваності пацієнтів.
- Unsupervised learning. Набір заснованих на технологіях машинного навчання методик, які дозволяють виявити приховані

функціональні взаємозв'язки в аналізованих масивах даних. Має спільні риси з Cluster Analysis.

- Візуалізація. Методи графічного представлення результатів аналізу великих даних у вигляді діаграм або анімації для спрощення інтерпретації полегшення розуміння отриманих результатів.

3.3. Переваги великих даних в банківській справі

Нижче наведені основні переваги великих даних в банківській сфері:

1. Ефективне управління ризиками для запобігання помилок і шахрайства

Інструменти бізнес-аналітики (BI) здатні виявляти потенційні ризики, пов'язані з процесами кредитування в банках. За допомогою аналітики великих даних банки можуть аналізувати тенденції ринку і приймати рішення про зниження або підвищення процентних ставок для різних людей в різних регіонах.

Помилки введення даних з ручних форм можуть бути зведені до мінімуму, оскільки великі дані також вказують на аномалії в даних клієнтів.

За допомогою алгоритмів виявлення шахрайства можна ідентифікувати клієнтів з низьким кредитним рейтингом, щоб банки не давали їм грошей в кредит. Ще одне важливе застосування в банківській сфері - обмеження випадків шахрайських або сумнівних операцій, які можуть сприяти антигромадської діяльності або тероризму.

2. Надає персоналізовані банківські рішення для клієнтів

Аналітика великих даних може допомогти банкам в розумінні поведінки клієнтів на основі даних, отриманих від їх моделей інвестування, тенденцій покупок, мотивації до інвестування і особистих або фінансових даних. Ці дані відіграють вирішальну роль в завоюванні лояльності клієнтів, розробляючи для них індивідуальні банківські рішення. Це призводить до симбіотичних відносин між банками і клієнтами. Індивідуальні банківські рішення також можуть значно збільшити кількість потенційних клієнтів.

3. Спрощена подача нормативних вимог

Більшість банківських службовців стверджують, що забезпечення банківських послуг відповідає всім нормативним критеріям відповідності, встановленим Урядом. 68% банківських службовців вважають, що їх найбільше занепокоєння щодо банківських послуг

Інструменти BI можуть допомогти проаналізувати і відстежувати всі нормативні вимоги, переглядаючи кожне окреме додаток від клієнтів для точної перевірки.

4. Підвищує загальну продуктивність

За допомогою аналізу продуктивності можна оцінити ефективність роботи співробітників, незалежно від того, чи досягли вони щомісячних / квартальних / річних цілей. На підставі даних, отриманих за поточними продажу співробітників, аналітика великих даних може визначити способи, які допоможуть їм краще масштабироваться.

Навіть банківські послуги в цілому можна перевірити, щоб знати, що працює, а що ні.

5. Ефективний аналіз відгуків клієнтів

Центри підтримки клієнтів Банку будуть отримувати безліч запитів і відгуків на регулярній основі. Навіть соціальні медіа-платформи сьогодні служать основою для взаємодії з клієнтами. Інструменти великих даних можуть допомогти відсівати великі обсяги даних і швидко і адекватно реагувати на кожен з них. Клієнти, які вважають, що їх банки відразу ж оцінять їхні відгуки, залишаться лояльними до бренду.

В кінцевому рахунку, банки, які не розвиваються і не використовують велику хвилю даних, не тільки залишаться позаду, але і втратять актуальності. Використання аналітики великих даних і інших високотехнологічних інструментів для перетворення існуючого банківського сектора відіграє важливу роль у визначенні довговічності банків в цифрову епоху. Ви можете покататися на хвилі великих даних майбутнього, записавшись на курс по сертифікації великих даних за допомогою Spark & Hadoop Training від Acadgild.

Все ж я вирішив використовувати Hadoop. Це проект з відкритим вихідним кодом, що знаходиться під управлінням Apache Software Foundation. Hadoop використовується для надійних, масштабованих і розподілених обчислень, але може також застосовуватися і як сховище файлів загального призначення, здатне вмістити петабайт даних.

Проект Apache ™ Hadoop® розробляє програмне забезпечення з відкритим кодом для надійного, масштабного, розподіленого обчислень.

Бібліотека програмного забезпечення Apache Hadoop – це структура, яка дозволяє розподілити великі обсяги даних за кластерами комп'ютерів, використовуючи прості моделі програми. Він розроблений для збільшення кількості окремих серверів до тисяч машин, які пропонують місце обчислення та зберігання. Для того, щоб покластися на апаратне забезпечення для забезпечення високої доступності, та сама бібліотека, призначена для виявлення та відтворення на прикладному рівні, таким чином.

На рис. 3.3 представлена діаграма структури Hadoop.

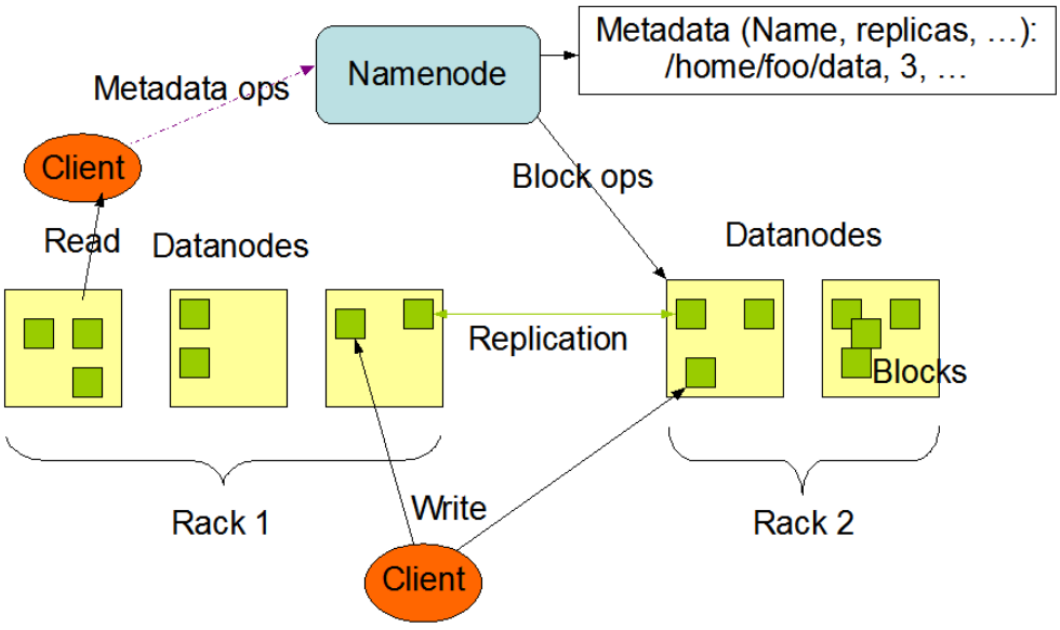


Рис. 3.3. Структура роботи Hadoop.

4.ОПИС РЕАЛІЗАЦІЇ СИСТЕМИ

4.1. Особливості реалізації

4.1.1. Серверна частина

На основі шаблону ASP .Net були розроблені відповідні модулі та доповнені додатковими статичними класами для спрощення роботи із БД. При отриманні заданих параметрів користувачем в залежності від запиту відбувається процедура доступу до данних, та вибірка потрібних даних.

4.1.2. Клієнтська частина

Логіка клієнтської частини реалізована методами C#, що описані у шаблонах сторінок і завантажуються з сервера.

4.2. Дизайн та вміст сторінок

Згідно з вимогами, дизайн сторінок додатку виконано у мінімалістичному стилі. Для відображення сторінок використовувались технології HTML5, CSS3 та Asp.net , а також платформа front-end розробки Twitter Bootstrap. Сторінки мають фіксовану ширину 980 пікселів і центральне вирівнювання відносно вікна браузера.

Основний контент візуально розділено на дві колонки. Контент в колонках подається строго, за допомогою кнопок та підписів.

Взаємодія з графіками викликає спливаючі підказки. Неможливість виконання певних дій чи відсутність результатів пошуку відображаються за допомогою стандартних оповіщень Bootstrap. Також користувачу показує, скільки помилок при було зроблено при переносі інформації з .xmls файлу до PostgreSQL.

Елементи управління розміщені так, щоби мінімізувати час доступу до них навіть при переході із сторінки на сторінку.

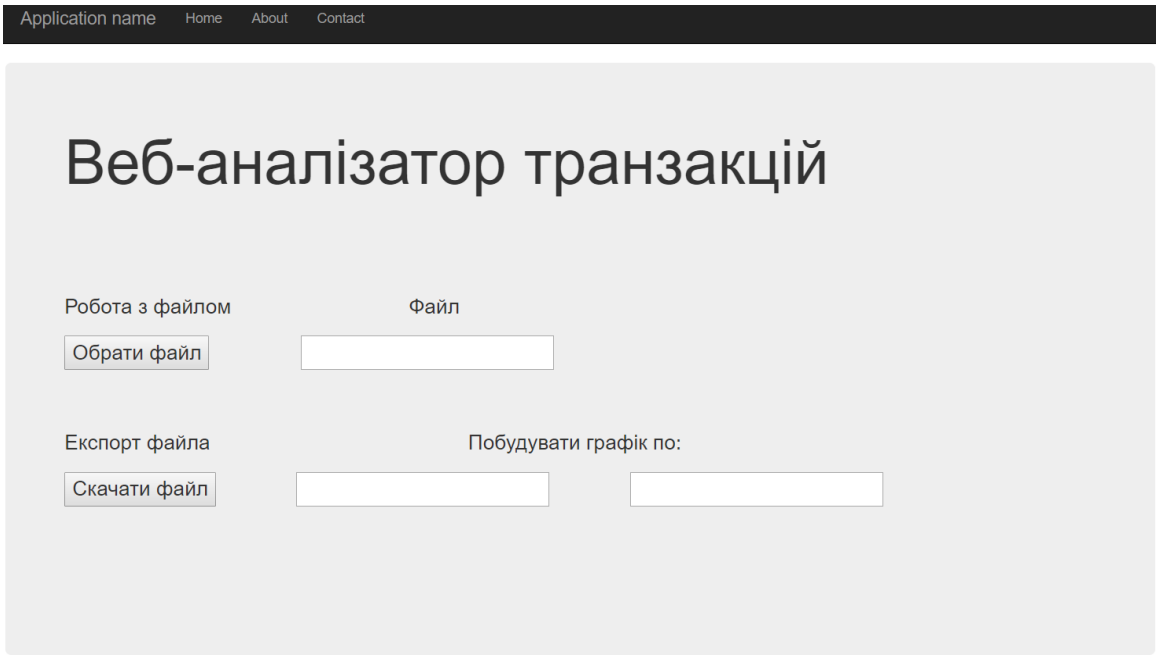


Рис. 4.1. Перша сторінка проекту

Можемо бачити кнопку для обирання файла, та текстове поле для візуалізації його імені, нижче можемо бачити кнопку для єкспорту файла, якщо його оригінала у користувача немає.

Також є два текстових поля, завдяки яким користувач може побудувати графік по файлу, який завантажен у БД.

Таблицы с результатом

Ошибок при импорте: 0

	Змінено	Сессия	Транзакція
2314	07.09.2014 17:04:45	18900	8900
2531	05.09.2014 17:06:57	15900	6300
7912	05.09.2014 16:36:20	9300	5500
1235	27.08.2014 18:17:32	8500	4500
9851	14.08.2014 15:59:36	5000	3000
6712	04.08.2014 13:47:30	-	7850
4512	26.07.2014 11:05:20	9700	7850
2312	18.07.2014 17:49:45	5700	4700
8763	14.07.2014 17:34:30	8500	5900
2415	09.07.2014 12:20:10	16300	12850

Рис. 4.2. Виведення частини бази даних.

Результат обробки файла у вигляді списка з інформацією по ID транзакції, коли вона була викнонана, яка сессія обробки та величина транзакції. Також буде доступна інформація по реєстрацію на сайті, ім’я, почту та телефон.

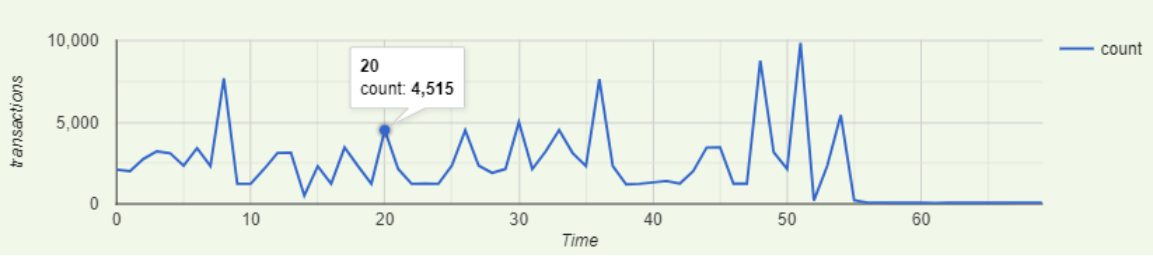
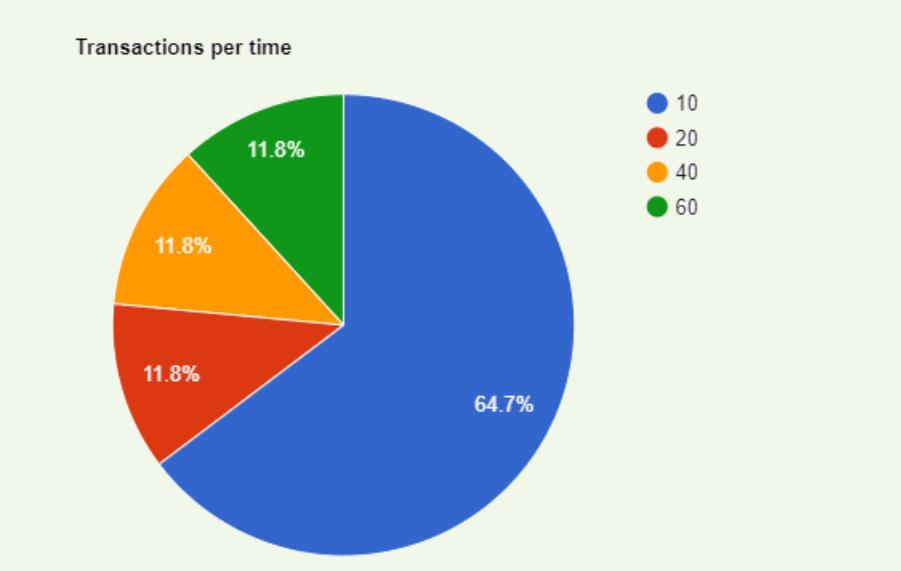


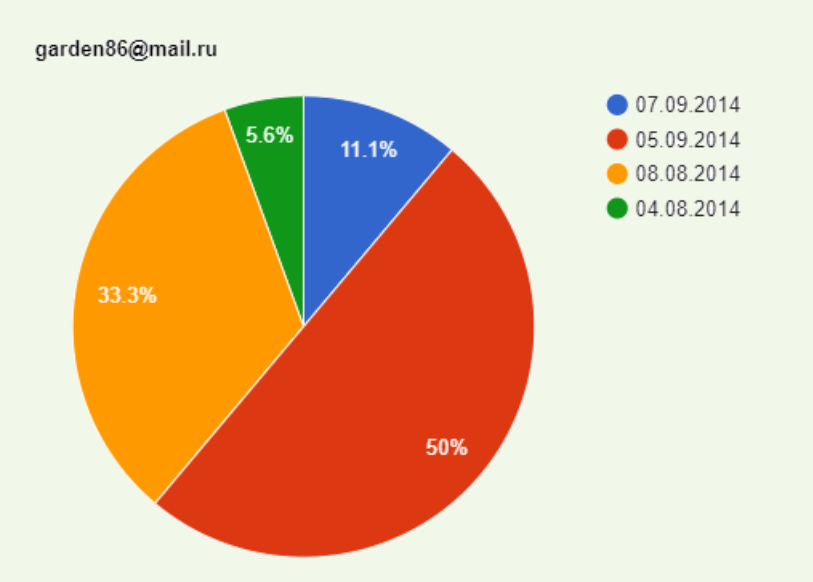
Рис. 4.3. Один з можливих графиків величина транзакції к часу.

Результат обробки запиту користувача у вигляді графіку з двома показниками величина та час.



4.4. Один з можливих графиків (транзакій в день).

Такий самий графік як у рис.4.5, але з другою реалізацією (Pie chart).



4.5. Графік- скільки транзакцій і коли.

Графік, який показує, скільки транзакцій, та у який день було здійснено користувачем з почтою garden86@mail.ru.

4.3. Тестування web-додатку

Для перевірки правильності функціонування програмних засобів було розроблено набір тест-кейсів для проведення димового тестування.

Сценарій димового тестування:

1. Встановити систему.
2. Обрати файл excel .
3. Єкспортувати файл у БД.
4. Вивести БД на сайті.
5. Відправити дані для виводу кількості транзакцій на день
6. Відправити дані для виводу кількості транзакцій по телефону.

Таблиця 4.1

Тест-кейси димового тестування

№ п/п	IDвимоги	Опис	Очікувані результати
1	–	1. Встановити систему.	1. Система встановлена без помилок і доступна із web-браузерів користувачів.
2	1	1. Зайти на сайт 2. Натиснути кнопку “обрати файл”	1. Система відображає сторінку вибору.

			2. Користувач робить вибір та надсилає запит . 3.Система відображає результат у текстовому блоку.

Продовження табл. 4.1

3	7	1. Зайти на сайт 2. Натиснути кнопку “Єкспортувати файл”	1. Система завантажує файл у необхідну папку.
4	5	1. Зайти на сайт 2. Натиснути “обрати файл” 3. Завантажуємо його у бд 4. Завантажуємо	1.Сайт відображається 2.Text box відображає ім’я файлу 3.Відкривається сторінка просмотру БД
5	5	1. Зайти на сайт 2. Обрати параметри графіка 3. Завантажуємо	1.Сайт відображається 2.Text box відображає параметр 3.Відкривається сторінка просмотру графіка
6	5	1. Зайти на сайт 2. Обрати параметри графіка 3. Завантажуємо	1.Сайт відображається 2.Text box відображає параметр 3.Відкривається сторінка просмотру графіка

ВИСНОВКИ

Метою даного дипломного проекту було розроблення web-додатку обробки транзакцій інтернет банку.

Аналіз засобів розроблення web-додатку, попередньо виконаний в дипломному проекті, показав доцільність створення системи у вигляді web-додатку.

Розроблена система:

- забезпечує доступ до веб- додатку;
- дозволяє отримувати доступ до одної БД з різних комп'ютерів, та завантажувати її у вигляді .xlns файлу;
- дає можливість отримувати статистичні та аналітичні дані по заданим параметрам;
- має зрозумілий та зручний інтерфейс користувача.

Особливу увагу під час розроблення даного програмного продукту було приділено інтерфейсу користувача і зручності роботи із системою.

Розробка виконана у повному обсязі, всі вимоги враховані, тестування продукту виконано у відповідності до затвердження програми та методики тестування.

Використання розробленої системи дасть змогу проаналізувати оброблені дані транзакцій банку

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. JavaScript: докладний посібник / Девід Фленаган / 2012
2. Node.js в дії / Натан Райлі / 2014
3. Чистий код: створення, аналіз і рефакторинг. Бібліотека програміста / Роберт Мартін / 2015
4. Маленька книга про MongoDB / Карл Сегуін
5. HTMLTutorial [Електроннийресурс]. —Режимдоступу: http://www.w3schools.com/html/
6. Документація CSS3 [Електронний ресурс]. — Режим доступу :http://www.w3schools.com/css3/
7. Node Package Manager [Електронний ресурс]. - Режим доступу: https://npmjs.com - Загл. з екрану. - яз. англ.
8. REST [Електронний ресурс]. - Режим доступу: https://ru.wikipedia.org/wiki/REST - Загл. з екрану. - яз. рус
9. ExpressJS® [Електронний ресурс] - Назва з екрану. Режим доступу: https://expressjs.com/
10. ChartJS® [Електронний ресурс] - Назва з екрану https://www.chartjs.org/docs/latest/
11. Data Visualization with chart,js [Електронний ресурс] - https://tobiasahlin.com/blog/introduction-to-chartjs/
12. The definitive guide to Express, the Node.js Web Application Framework https://hackernoon.com/the-definitive-guide-to-express-the-node-js-web-application-framework-649352e2ae87
13. Node.js [Електронний ресурс] - Назва з екрану https://nodejs.org/en/docs/

14. MongoDB [Електронний ресурс] - Назва з екрану https: //www.mongodb.org
15. Український центр оцінювання якості освіти [Електронний ресурс] - Назва з екрану https://zno.testportal.com.ua/
16. Аналіз результатів ЗНО [Електронний ресурс] - Назва з екрану https://zno.ua/news/analiz-testu-z-ukrajinskoji-movi-ta-literaturi-zno-2016.html

ДОДАТОК

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ВЕБ-СЕРВІС ДЛЯ АНАЛІЗУ ТРАНЗАКЦІЙ ІНТЕРНЕТ БАНКУ ЗА ДОПОМОГОЮ МЕТОДІВ BIG DATA

Программа та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проекту:

_____ А.В. Гречко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____Є.Д. Страмоусов

2019

ЗМІСТ

1. ОБ'ЄКТ ВИПРОБУВАНЬ

Web-ресурс анализу транзакцій інтернет банку, який являє собою web-сайт, створений на платформі Microsoft .Net з використанням технології ASP.NET AJAX.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1) функціональна працездатність елементів сторінок web-ресурсу;
2) наявність доступу до бази дистанційних sql;
3) відповідність форматів та протоколів передачі даних з системою банка;
4) забезпечення належного рівня безпеки даних;
5) зручність роботи з web-сайтом;
6) відповідність дизайну вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

1) функціональне тестування, зокрема на рівні Critical path test (базове тестування);
2) тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності) та Load testing (навантажувальне тестування);
3) тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується засобами інструментарію SpecFlow.

Працездатність web-ресурсу перевіряється шляхом:

1) динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;
2) динамічного ручного тестування на відповідність функціональним вимогам;
3) статичного тестування коду;
4) тестування web-ресурсу в різних web-браузерах;
5) тестування при максимальному навантаженні;
6) тестування стабільності роботи при різних умовах;
7) тестування зручності використання;
8) тестування інтерфейсу.

Додаток 1

Копії графічних матеріалів

Додаток 2

Лістінг програми

Додаток 3

Копія презентації

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using ClosedXML.Excel;
using WebApplication2.Models;

namespace WebApplication2.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        [HttpPost]
        [ValidateInput(false)]
        public ActionResult Import(HttpPostedFileBase fileExcel)
        {
            if (ModelState.IsValid)
            {
                PriceViewModel viewModel = new PriceViewModel();
                using (XLWorkbook workBook = new XLWorkbook(fileExcel.InputStream,
XLEventTracking.Disabled))
                {
                    foreach (IXLWorksheet worksheet in workBook.Worksheets)
                    {
                        PhoneBrand phoneBrand = new PhoneBrand();
                        phoneBrand.Title = worksheet.Name;

                        foreach (IXLColumn column in worksheet.ColumnsUsed().Skip(1))
                        {
                            PhoneModel phoneModel = new PhoneModel();
                            phoneModel.Title = column.Cell(1).Value.ToString();

                            foreach (IXLRow row in worksheet.RowsUsed().Skip(1))
                            {
                                try
                                {
                                    PricePosition pricePosition = new PricePosition();
                                    pricePosition.Problem = row.Cell(1).Value.ToString();
                                    pricePosition.Price =
row.Cell(column.ColumnNumber()).Value.ToString();
                                    phoneModel.PricePositions.Add(pricePosition);

                                }
                                catch (Exception e)
                                {
                                    //logging
                                    viewModel.ErrorsTotal++;
                                }
                            }
```

```
                        phoneBrand.PhoneModels.Add(phoneModel);
                    }
                    viewModel.PhoneBrands.Add(phoneBrand);
                }
            }
            //например, здесь сохраняем все позиции из прайса в БД

            return View(viewModel);
        }
        return RedirectToAction("Index");
    }

    public ActionResult Export()
    {
        List<PhoneBrand> phoneBrands = new List<PhoneBrand>();
        phoneBrands.Add(new PhoneBrand()
        {
            Title = "Apple",
            PhoneModels = new List<PhoneModel>()
        {
            new PhoneModel() { Title = "iPhone 7"},
            new PhoneModel() { Title = "iPhone 7 Plus"}
        }
        });
        phoneBrands.Add(new PhoneBrand()
        {
            Title = "Samsung",
            PhoneModels = new List<PhoneModel>()
        {
            new PhoneModel() { Title = "A3"},
            new PhoneModel() { Title = "A3 2016"},
            new PhoneModel() { Title = "A3 2017"}
        }
        });

        using (XLWorkbook workbook = new XLWorkbook(XLEventTracking.Disabled))
        {
            var worksheet = workbook.Worksheets.Add("Brands");

            worksheet.Cell("A1").Value = "Бренд";
            worksheet.Cell("B1").Value = "Модели";
            worksheet.Row(1).Style.Font.Bold = true;

            //нумерация строк/столбцов начинается с индекса 1 (не 0)
            for (int i = 0; i < phoneBrands.Count; i++)
            {
                worksheet.Cell(i + 2, 1).Value = phoneBrands[i].Title;
                worksheet.Cell(i + 2, 2).Value = string.Join(", ",
phoneBrands[i].PhoneModels.Select(x => x.Title));
            }

            using (var stream = new MemoryStream())
            {
                workbook.SaveAs(stream);
                stream.Flush();

                return new FileContentResult(stream.ToArray(),
```

```
                        "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet")
                    {
                        FileDownloadName =
$"brands_{DateTime.UtcNow.ToShortDateString()}.xlsx"
                    };
                }
            }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace WebApplication2.Models
{
    public class PricePosition
    {
        //Неисправность
        public string Problem { get; set; }

        //Стоимость ремонта
        public string Price { get; set; }
    }

    public class PhoneModel
    {
        public PhoneModel()
        {
            PricePositions = new List<PricePosition>();
        }

        //Название модели телефона
        public string Title { get; set; }

        public List<PricePosition> PricePositions { get; set; }
    }

    public class PhoneBrand
    {
        public PhoneBrand()
        {
            PhoneModels = new List<PhoneModel>();
        }

        //Название бренда
        public string Title { get; set; }

        public List<PhoneModel> PhoneModels { get; set; }
    }
                                            }
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace WebApplication2
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
            );
        }
    }
}
```

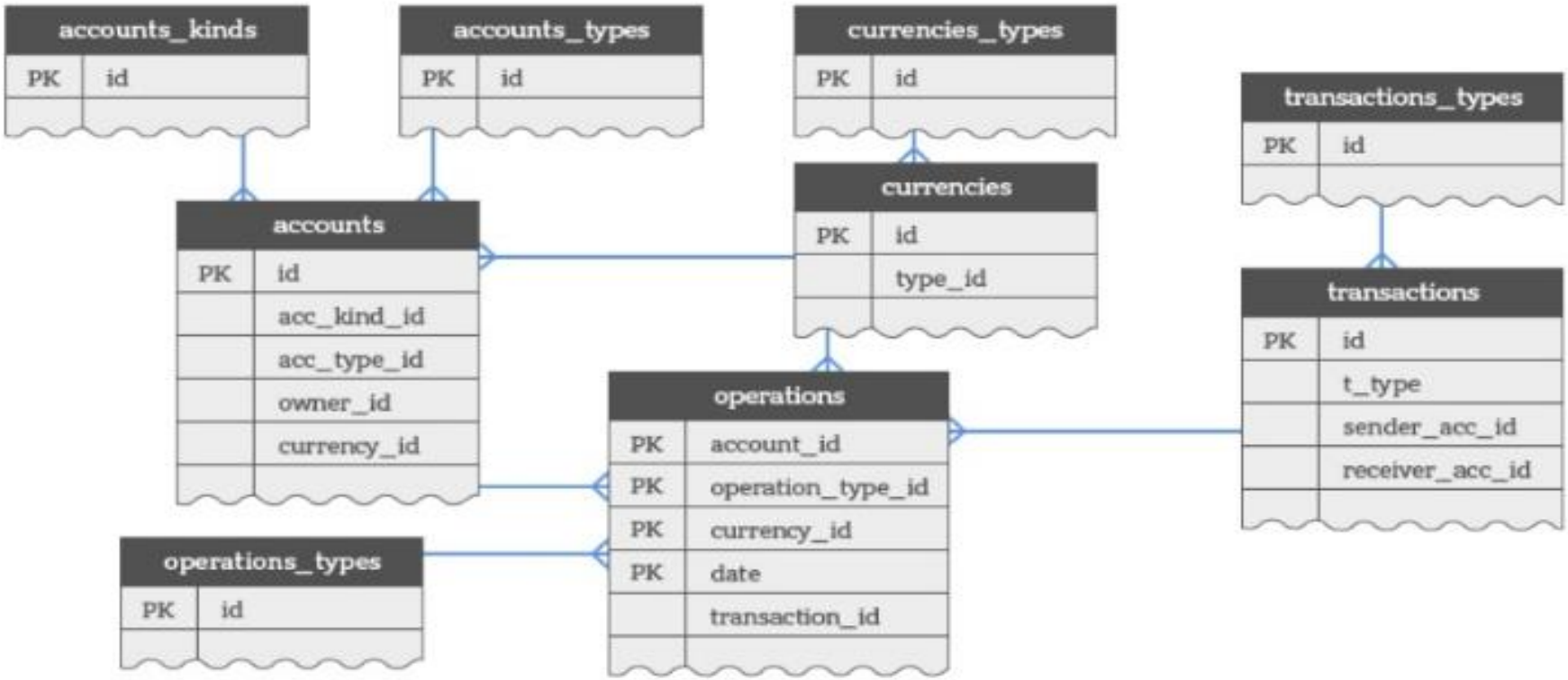
```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using ClosedXML.Excel;
using WebApplication3.Models;

namespace WebApplication3.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        [HttpPost]
        [ValidateInput(false)]
        public ActionResult Import(HttpPostedFileBase fileExcel)
        {
            if (ModelState.IsValid)
            {
                PriceViewModel viewModel = new PriceViewModel();
                using (XLWorkbook workBook = new XLWorkbook(fileExcel.InputStream,
XLEventTracking.Disabled))
                {
                    foreach (IXLWorksheet worksheet in workBook.Worksheets)
                    {
                        PhoneBrand phoneBrand = new PhoneBrand();
                        phoneBrand.Title = worksheet.Name;

                        foreach (IXLColumn column in worksheet.ColumnsUsed().Skip(1))
                        {
                            PhoneModel phoneModel = new PhoneModel();
                            phoneModel.Title = column.Cell(1).Value.ToString();

                            foreach (IXLRow row in worksheet.RowsUsed().Skip(1))
                            {
                                try
                                {
                                    PricePosition pricePosition = new PricePosition();
                                    pricePosition.Problem = row.Cell(1).Value.ToString();
                                    pricePosition.Price =
row.Cell(column.ColumnNumber()).Value.ToString();
                                    phoneModel.PricePositions.Add(pricePosition);

                                }
                                catch (Exception e)
                                {
                                    //logging
                                    viewModel.ErrorsTotal++;
                                }
                            }
```

```
                        phoneBrand.PhoneModels.Add(phoneModel);
                    }
                    viewModel.PhoneBrands.Add(phoneBrand);
                }
            }
            //например, здесь сохраняем все позиции из прайса в БД

            return View(viewModel);
        }
        return RedirectToAction("Index");
    }

    public ActionResult Export()
    {
        List<PhoneBrand> phoneBrands = new List<PhoneBrand>();
        phoneBrands.Add(new PhoneBrand()
        {
            Title = "Apple",
            PhoneModels = new List<PhoneModel>()
        {
            new PhoneModel() { Title = "iPhone 7"},
            new PhoneModel() { Title = "iPhone 7 Plus"}
        }
        });
        phoneBrands.Add(new PhoneBrand()
        {
            Title = "Samsung",
            PhoneModels = new List<PhoneModel>()
        {
            new PhoneModel() { Title = "A3"},
            new PhoneModel() { Title = "A3 2016"},
            new PhoneModel() { Title = "A3 2017"}
        }
        });

        using (XLWorkbook workbook = new XLWorkbook(XLEventTracking.Disabled))
        {
            var worksheet = workbook.Worksheets.Add("Brands");

            worksheet.Cell("A1").Value = "Бренд";
            worksheet.Cell("B1").Value = "Модели";
            worksheet.Row(1).Style.Font.Bold = true;

            //нумерация строк/столбцов начинается с индекса 1 (не 0)
            for (int i = 0; i < phoneBrands.Count; i++)
            {
                worksheet.Cell(i + 2, 1).Value = phoneBrands[i].Title;
                worksheet.Cell(i + 2, 2).Value = string.Join(", ",
phoneBrands[i].PhoneModels.Select(x => x.Title));
            }

            using (var stream = new MemoryStream())
            {
                workbook.SaveAs(stream);
                stream.Flush();

                return new FileContentResult(stream.ToArray(),
```

```
                        "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet")
                    {
                        FileDownloadName =
$"brands_{DateTime.UtcNow.ToShortDateString()}.xlsx"
                    };
                }
            }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace WebApplication3.Models
{
    public class PricePosition
    {
        //Неисправность
        public string Problem { get; set; }

        //Стоимость ремонта
        public string Price { get; set; }
    }

    public class PhoneModel
    {
        public PhoneModel()
        {
            PricePositions = new List<PricePosition>();
        }

        //Название модели телефона
        public string Title { get; set; }

        public List<PricePosition> PricePositions { get; set; }
    }

    public class PhoneBrand
    {
        public PhoneBrand()
        {
            PhoneModels = new List<PhoneModel>();
        }

        //Название бренда
        public string Title { get; set; }

        public List<PhoneModel> PhoneModels { get; set; }
    }
                                            }
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Routing;
using System.Web.UI;
using System.Web.UI.WebControls;
using Microsoft.AspNet.FriendlyUrls.Resolvers;

namespace WebApplication3
{
    public partial class ViewSwitcher : System.Web.UI.UserControl
    {
        protected string CurrentView { get; private set; }

        protected string AlternateView { get; private set; }

        protected string SwitchUrl { get; private set; }

        protected void Page_Load(object sender, EventArgs e)
        {
            // Determine current view
            var isMobile = WebFormsFriendlyUrlResolver.IsMobileView(new
HttpContextWrapper(Context));
            CurrentView = isMobile ? "Mobile" : "Desktop";

            // Determine alternate view
            AlternateView = isMobile ? "Desktop" : "Mobile";

            // Create switch URL from the route, e.g.
~/__FriendlyUrls_SwitchView/Mobile?ReturnUrl=/Page
            var switchViewRouteName = "AspNet.FriendlyUrls.SwitchView";
            var switchViewRoute = RouteTable.Routes[switchViewRouteName];
            if (switchViewRoute == null)
            {
                // Friendly URLs is not enabled or the name of the switch view route is
out of sync
                this.Visible = false;
                return;
            }
            var url = GetRouteUrl(switchViewRouteName, new { view = AlternateView,
__FriendlyUrls_SwitchViews = true });
            url += "?ReturnUrl=" + HttpUtility.UrlEncode(Request.RawUrl);
            SwitchUrl = url;
        }
    }
                                            }
```

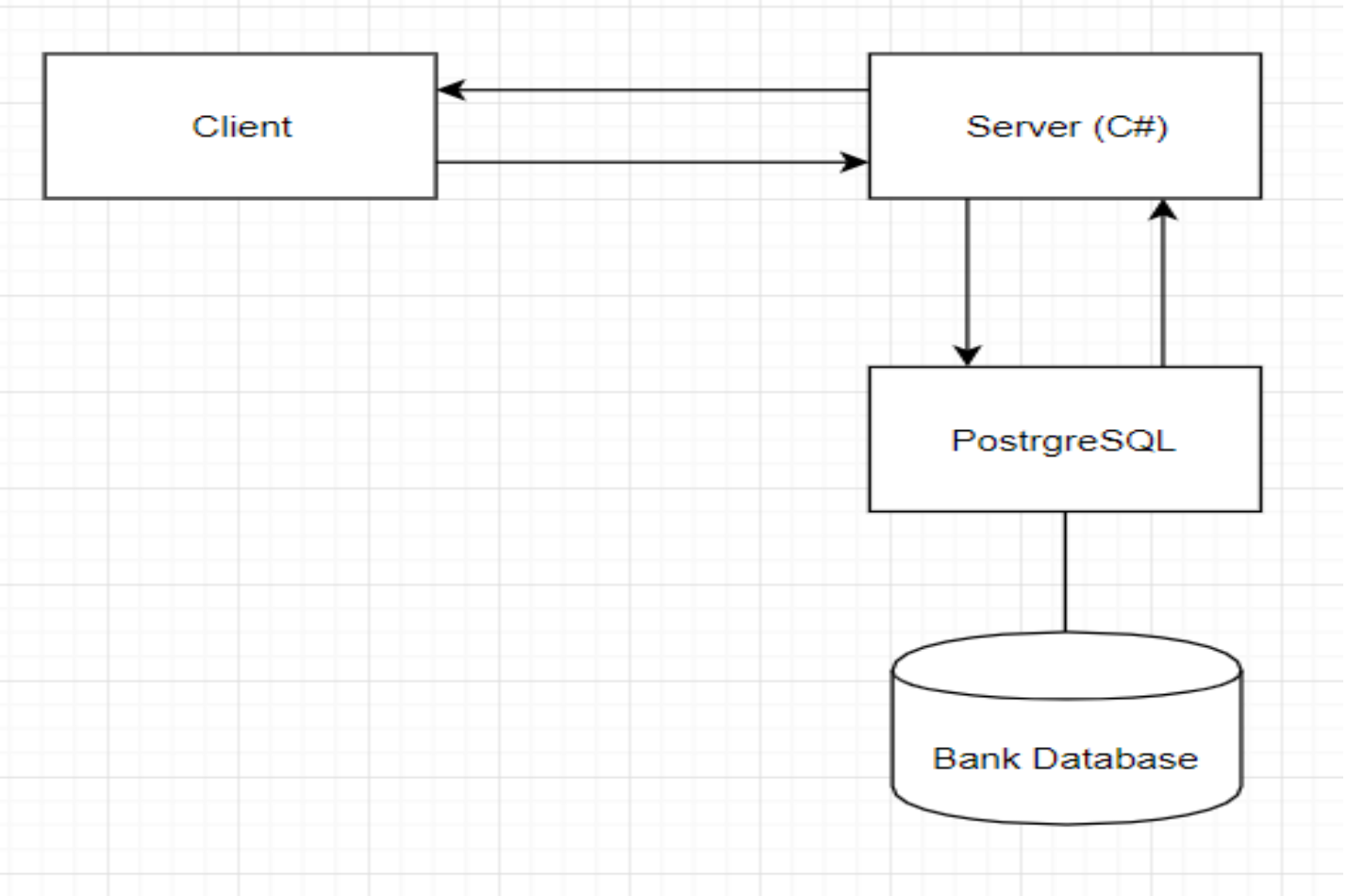


ДП.045440-06-99

Схема бази даних. Веб-застосунок для аналізу транзакцій Інтернет-банку на основі технологій Big Data.

ER діаграма

Структурна схема системи



Страмоусов Є.Д.

КП-51

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ВЕБ-СЕРВІС ДЛЯ АНАЛІЗУ ТРАНЗАКЦІЙ ІНТЕРНЕТ БАНКУ ЗА ДОПОМОГОЮ МЕТОДІВ BIG DATA

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проекту:

_____ А.В. Гречко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ Є.Д. Страмоусов

2019

ЗМІСТ

1. Опис структури web-ресурсу

Web-ресурс аналізу транзакцій інтернет банку складається із статичних web-сторінок та web-сторінок, вміст яких формується динамічно.

До статичних належать наступні web-сторінки:

- «About Center»;
- «About»;
- «Contact».

Динамічна частина web-ресурсу включає наступні web-сторінки:

- Сторінка перегляду бази даних ;
- Сторінка перегляду графіка;

Кожна web-сторінка містить посилання на головну сторінку ресурсу, web-сайти спонсорів та партнерів Центру.

2. Опис вмісту статичних web-сторінок

Головна сторінка містить базову інформацію та кнопки для забеспечення функціоналу сайта, мету діяльності та основні події, які відбулись в рамках діяльності сайту.

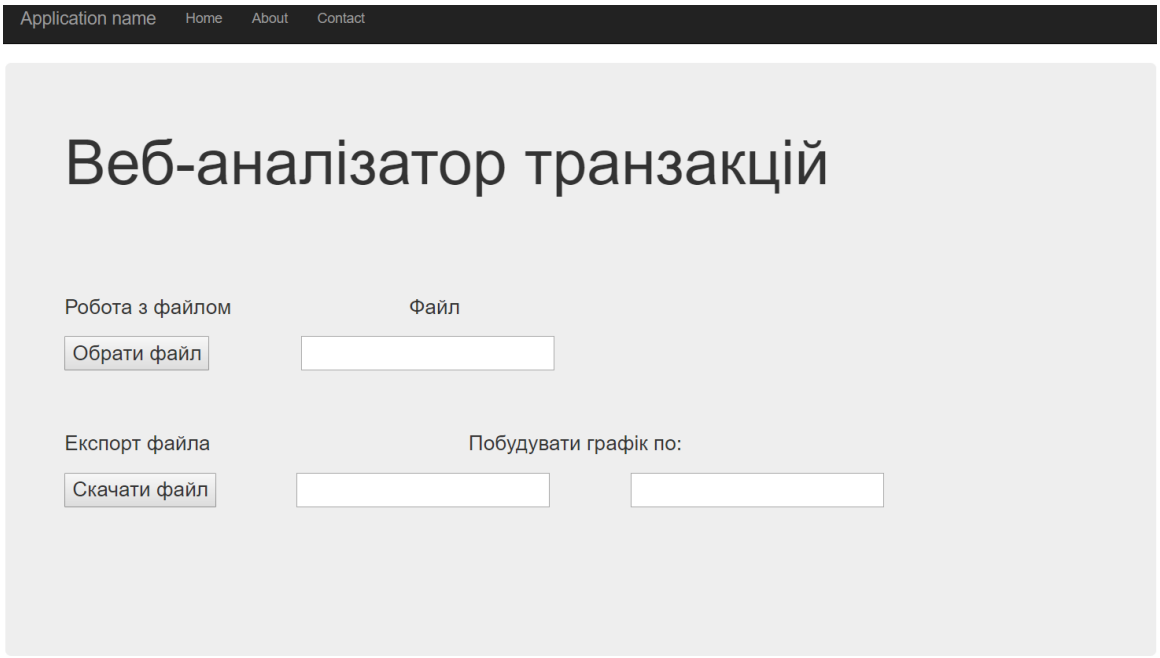


Рис. 1. Головна сторінка сторінка

Таблицы с результатом

Ошибок при импорте: 0

	Змінено	Сессия	Транзакція
2314	07.09.2014 17:04:45	18900	8900
2531	05.09.2014 17:06:57	15900	6300
7912	05.09.2014 16:36:20	9300	5500
1235	27.08.2014 18:17:32	8500	4500
9851	14.08.2014 15:59:36	5000	3000
6712	04.08.2014 13:47:30	-	7850
4512	26.07.2014 11:05:20	9700	7850
2312	18.07.2014 17:49:45	5700	4700
8763	14.07.2014 17:34:30	8500	5900
2415	09.07.2014 12:20:10	16300	12850

Рис. 2. Сторінка перегляду БД