Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2022 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

за другим (магістерським) рівнем вищої освіти

на тему

“ Дослідження та програмна реалізація процесів міграції баз даних в корпоративних інформаційних системах ”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-21М
ОПП «Комп'ютерна інженерія»
спеціальності 122 «Комп'ютерні науки»
_____Чередніченко А.М.
«____» _____ 2022р.

Керівник проекту
кандидат технічних наук, доцент
_____ Босько В.В.
«____» _____ 2022 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *магістр*

Галузь знань *12 "Інформаційні технології"*

Спеціальність *122 "Комп'ютерні науки"*

Освітньо-професійна (освітньо-наукова) програма *"Комп'ютерні науки"*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
Олексій СМІРНОВ
"____" _____20___року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Черднічeнку Антону Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація процесів міграції баз даних в корпоративних інформаційних системах*

2. Керівник роботи *Босько Віктор Васильович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 18-13 від 17.08.22

3. Строк подання роботи до захисту *20.12.2022 р.*

4. Мета та завдання випускної кваліфікаційної роботи: *Метою розробки є програмне дослідження та програмна реалізація процесів міграцій баз даних в ІС*

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

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Наукова новизна
7. Економічна ефективність розробленої програми.
8. Заходи з охорони праці та техніки безпеки.
9. Висновки.

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Діаграма процесів	*1 аркуш*
Показники економічної ефективності	*1 аркуш*

6. Консультанти по роботі, із зазначенням розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В., к.т.н., доцент	09.11.2022 р.	17.11.2022 р.
Охорона праці	Оришака О.В., к.т.н., доцент	03.11.2022 р.	21.11.2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	12.10.2022 р.	
2.	Постановка задачі, оформлення ТЗ	18.10.2022 р.	
3.	Розробка моделі компонента	23.10.2022 р.	
4.	Розробка структур даних	25.10.2022 р.	
5.	Розробка алгоритмів зв’язку та відображення	32.10.2022 р.	
6.	Програмування алгоритмів	11.11.2022 р.	
7.	Розрахунок економічної ефективності	13.11.2022 р.	
8.	Розрахунки з охорони праці та техніки безпеки	16.11.2022 р.	
9.	Оформлення ПЗ	18.11.2022 р.	
10.	Попередній захист роботи	10.12.2022 р	

Дата видачі завдання
«__»_____2022р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
«__»_____2022р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Чередніченко А.М. Дослідження та програмна реалізація процесів міграції баз даних в корпоративних інформаційних системах. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький 2022.

В даній магістерській роботі розроблено програмне забезпечення, яке призначено для реалізації міграцій в базах даних .

Метою розробки є дослідження міграції баз даних та створення методу міграції даних.

Об'єктом дослідження є процес міграції баз даних у корпоративних системах.

Предметом дослідження є методи та алгоритми міграції, а саме між СУБД MS Server та PostgreSQL.

Методи дослідження базуються на методах теорії кодування, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація проекту міграції між СУБД MS Server та PostgreSQL.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися для міграції між СУБД MS Server та PostgreSQL. Програму розроблено в середовищі Java.

Ключові слова: комп'ютерна інженерія, СУБД MS Server, PostgreSQL, бази даних, ООП, ООБД, java.

ANNOTATION

Cherednichenko A.M. Research and software implementation of database migration processes in corporate information systems. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi 2022.

In this master's thesis, software was developed, which is intended for the implementation of migrations in databases.

The purpose of the development is the research of database migration and the creation of a data migration method.

The object of research is the process of database migration in corporate systems.

The subject of the study is the methods and algorithms of migration, namely between the DBMS MS Server and PostgreSQL.

Research methods are based on coding theory methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the migration project between the DBMS MS Server and PostgreSQL.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used for migration between DBMS MS Server and PostgreSQL. The program is developed in the Java environment.

Keywords: computer engineering, DBMS MS Server, PostgreSQL, databases, OOP, OOBD, java.

ЗМІСТ

					ВКРМ-122.22.0017.00.00.ПЗ	
Вим.	Арк.	№ докум.	Підпис	Дат		
Розроб.		Чередніченко А			*Дослідження та програмна реалізація процесів міграції баз даних в корпоративних інформаційних системах*	Літ. М / Арк. 1 / Аркушів
Перевір.		Босько В.В				
Н. Контр.		Гермак В.С				ЦНТУ КН-21М
Затверд.		Смірнов О.А.				

ПЕРЕЛІК СКОРОЧЕНЬ, СИМВОЛІВ ТА СПЕЦІАЛЬНИХ ТЕРМІНІВ

ІС – Інформаційна система

ООБД – Об’єктно-Орієнтована База Даних

СУБД – Система Управління Базою Даних

JSON - JavaScript Object Notation

MIME - Multipurpose Internet Mail Extensions

REST - Representational State Transfer

MVC - Model-View-Controller

CSS - Cascading Style Sheets

HTML - Hypertext Markup Language

ОС - операційна система

XML - Extensible Markup Language

CMS - Content Management System

ORM - Object-relational mapping

Sass - Syntactically Awesome Stylesheets

DOM - Document Object Model

EJS - Embedded JavaScript templating

ВСТУП

Актуальність теми. Зростання ролі інформаційних технологій у житті суспільства призводить до необхідності використання відповідно і зручних інструментів для управління даними чи інформацією. На сьогоднішній день спостерігається широке поширення технологій реляційних баз даних, що призвело до появи багатьох програмних продуктів з різними можливостями. Це ставить перед користувачем завдання вибрати відповідну СУБД, здатну вирішити всі поставлені перед користувачем завдання. Міграція - це метод переходу від однієї СУБД до іншої зі збереженням існуючих даних. Міграція бази даних - досить складний процес, який має свої проблеми та ризики. У разі міграції промислових баз даних накладається ще більше умов і обмежень. У базі даних зберігається величезна кількість інформації, яка необхідна компанії для продовження діяльності. Тому в результаті міграції нова база повинна містити абсолютно всі дані. Крім того, база даних є частиною великої бізнес-системи, вона повинна спілкуватися з іншими елементами, обмінюватися з ними даними. Після міграції всі інші програми повинні отримати правильні дані з нової бази даних. Крім того, час для міграції дуже обмежений, оскільки система має працювати практично 24 години на добу, 7 днів на тиждень. Перенесення можна здійснити лише під час короткої технічної перерви. У результаті необхідно негайно отримати робочу систему, яка містить усі дані, які є у вихідній базі даних, щоб після завершення процесу користувачі системи могли продовжувати роботу. Основна мета проекту міграції - перевести вихідну базу даних з однієї бази в абсолютно ідентичну, але від іншого постачальника.

До результату міграції застосовуються такі вимоги:

- необхідно перенести всі дані з вихідної бази даних;
- функціональність вихідної бази даних має бути збережена.

Існує також додаткова нефункціональна і, на перший погляд, неформалізована вимога щодо мінімізації змін у схемі даних і в сигнатурах

збережених процедур, щоб спростити подальшу адаптацію реальних клієнтських додатків до нової бази даних.

У цій роботі розглядатимуться такі СУБД, як Microsoft SQL Server і PostgreSQL. Для написання функцій в PostgreSQL використовується вбудована процедурна мова PL / SQL, можливості якого багато в чому схожі з Transact-SKL, що використовується в СУБД MS SQL Server. Але між цими мовами є численні відмінності. Postgres не підтримує низку конструкцій Transact-SQL, а деякі мають альтернативи. Метою цього випускного проекту буде реалізація програми, яка перетворювала б компоненти бази даних MS SQL Server (таблиці, представлення, процедури, функції) у код, який коректно працює на PostgreSQL, який має таку саму функціональність. Під час перетворення компонентів бази даних виникають такі проблеми:

- відмінність засобів СУБД MS SQL Server і PostgreSQL;
- відсутність збережених процедур у PostgreSQL;
- різні типи даних у MS SQL Server та PostgreSQL.

Серед основних завдань будуть розглянуті наступні завдання:

- аналіз проблем міграції компонентів бази даних;
- аналоговий аналіз;
- формулювання завдання перетворення;
- аналіз відмінностей між засобами СУБД MS SQL Server і PostgreSQL - розробка програмного алгоритму перетворення;
- розробка міграційного програмного забезпечення;
- тестування програми міграції.

Крім вирішення основних завдань, можуть виникнути додаткові, другорядні завдання, які дозволять остаточно спроектувати додаток для міграції компонентів бази даних MS SQL Server.

Мета й завдання дослідження. Метою роботи є дослідження міграції бази даних і створення методу міграції даних, який має бути більш ефективним, ніж існуючі методи, і виконувати роботу оптимально.

Для досягнення мети була визначена програма дослідження, яка складається з наступних завдань:

– огляд існуючих типів БД;

– повинні бути перенесені всі дані з вихідної бази;

– повинна бути збережена функціональність вихідної бази.

Об'єктом дослідження є *процес міграції баз даних у корпоративних інформаційних системах*.

Предметом дослідження є методи я є методи та алгоритми міграції, а саме між СУБД MS Server та PostgreSQL.

Методи дослідження базуються на методах теорії кодування, методах побудови архітектури програмного забезпечення, методах розробки програмного забезпечення та методах зберігання даних.

Наукова новизна отриманих результатів. У процесі вирішення завдань, визначених цілями дослідження, отримано наступні результати:

- досліджено новий метод міграції даних та проаналізовано область використання нового методу;

- розроблено та реалізовано алгоритм побудови послідовності обходу схеми бази даних;

- розроблено та реалізовано алгоритм порівняння схем баз даних.

Практична цінність отриманих результатів полягає в тому, що досягнуті наступні результати:

- розширений механізм перенесення даних для виконання паралельної міграції даних;
- створений прототип додатка для міграції даних.

Достовірність наукових результатів підтверджена теоретичними викладенням та отриманими даними під час тестування розроблених систем.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи міграції БД в сучасних ІС, є актуальною задачею, яка потребує вирішення у даній магістерській роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1. Призначення системи

Метою даної роботи буде реалізація програми, яка б перетворювала компоненти бази даних MS SQL Server (таблиці, представлення, процедури, функції) у код, який коректно працює на PostgreSQL, який має таку саму функціональність.

Під час перетворення компонентів бази даних виникають такі проблеми:

- відмінність засобів СУБД MS SQL Server і PostgreSQL;
- відсутність збережених процедур у PostgreSQL;
- різні типи даних у MS SQL Server та PostgreSQL.

Серед основних завдань будуть розглянуті наступні завдання:

- аналіз проблем міграції компонентів бази даних;
- аналоговий аналіз;
- формулювання завдання перетворення;
- аналіз відмінностей між MS SQL Server і PostgreSQL
- розробка програмного алгоритму перетворення;
- розробка міграційного програмного забезпечення;
- тестування програми міграції;

Крім вирішення основних завдань, можуть виникнути додаткові, другорядні завдання, які дозволять остаточно спроектувати додаток для міграції компонентів бази даних MS SQL Server.

Магістерська робота має на меті показати можливості використання різних методів міграції баз даних, їх переваги та недоліки.

1.2. Область застосування

Міграція даних — це процес перенесення даних із джерела до цільової бази даних, і їх схеми можуть відрізнятися. Може бути багато причин, чому організації починають проекти міграції даних, від оновлення програм і впровадження нових корпоративних систем до повної реструктуризації через злиття компаній.

Аналізуючи досвід міграції даних великої кількості інформаційних систем, можна виділити типову процедуру міграції даних, яка включає:

- аналіз формату даних вихідної та цільової структури бази даних, підготовку плану міграції та конвертацію даних;
- визначення зв'язків між таблицями (ієрархії об'єктів);
- визначення порядку передачі даних відповідно до ієрархії залежностей.

Механізм міграції має гарантувати, що всі записи з вихідної бази даних переглядаються та мігрують, цілісність даних гарантується, а записи переносяться з урахуванням усіх залежностей. Запис можна перенести, лише якщо він незалежний або записи, від яких він залежить, уже перенесено до цільової бази даних. Тому це основне правило, яке визначає порядок перегляду та передачі залежностей між записами. Залежно від запису вони визначаються зовнішніми ключами.

Механізм міграції даних реалізовано у вигляді процедури, яка спочатку викликається для незалежних записів, а потім рекурсивно для залежних від цього запису. Тому порядок руху між ними повинен починатися з таблиці, що містить незалежні записи. Тоді основний рядок, визначений у наборі таблиць схеми бази даних, є залежністю між таблицями.

Іноді ви можете ігнорувати зв’язок, а просто вимкнути всі зовнішні ключі перед міграцією та відновити їх після завершення всіх маніпуляцій даними. Винятком є, наприклад, коли нова версія бази даних вже працює. Виконання

сценарію зміни об’єктів у новій версії бази даних. пряма передача даних з необхідними перетвореннями - на льоту. Запуск сценарію для відновлення вимкнених індексів, додаткових перетворень тощо. після завершення процесу міграції даних.

Найпростіший варіант – створити проміжне ПЗ. Воно повине взаємодіяти з вихідною та цільовою базами даних і виконувати необхідні перетворення.

Рисунок 1.1 – Варіант міграції даних з проміжною програмою

Інший варіант міграції даних (цей варіант більш кращий при переході на нову систему управління базами даних (далі, СУБД)) – попереднє завантаження старих даних в тимчасові таблиці нової СУБД. Сучасні СУБД (наприклад, Oracle)зазвичай містять спеціальні утиліти, які дозволяють виробляти дуже швидке завантаження зовнішніх даних різних форматів. В такому випадку модуль міграції пишеться засобами процедурних мов, вбудованих в СУБД (наприклад, PL / SQL або java).

Рисунок 1.2 – Варіант міграції даних засобами СУБД

Міграція бази даних - у контексті бізнес-додатків - означає переміщення ваших даних з однієї платформи на іншу. Є багато причин, чому ви можете перейти на іншу платформу. Наприклад, компанія може вирішити заощадити гроші, перейшовши на хмарну базу даних. Або компанія може виявити, що певне програмне забезпечення для баз даних має функції, які є критично важливими для її бізнес-потреб. Або старі системи просто застаріли. Процес міграції бази даних може включати кілька етапів і ітерацій, включаючи оцінку поточних баз даних компанії та майбутніх потреб, міграцію схеми та нормалізацію та міграцію даних. Плюс тестування, тестування та інше.

Недоліками міграції бази даних є витрати. Однією з головних причин, чому компанії переносять бази даних, є економія грошей. Компанії часто переходять від локальної бази даних до хмарної бази даних. Це економить на інфраструктурі, робочій силі та досвіді, необхідних для її підтримки.

Оновлене програмне забезпечення. Іншою поширеною причиною міграції є перехід від застарілої системи або застарілих систем до системи, розробленої для сучасних потреб у даних. В епоху великих даних необхідні нові методи зберігання. Наприклад, компанія може вибрати міграцію зі старої бази даних SQL до озера даних або іншої гнучкої системи.

Іншою поширеною причиною міграції даних є переміщення всіх даних в одне місце, доступне для всіх відділів компанії. Іноді таке трапляється після покупки, коли потрібно об’єднати системи. Або це може статися, коли в компанії впроваджуються різні системи. Наприклад, IT-відділ може використовувати одну базу даних, тоді як маркетингова група використовує іншу базу даних, і ці системи не можуть «розмовляти» одна з одною. Якщо у вас є різні бази даних, які не сумісні, важко зрозуміти ваші дані.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1. Огляд існуючих систем

Проблеми з міграцією баз даних

Міграція бази даних може бути дуже складною, але за умови належного планування ці поширені проблеми можна пом’якшити.

Існують такі проблеми міграції:

- пошук бази даних;
- втрата або пошкодження даних;
- безпека.

Проблема №1. Пошук по базі даних. Якщо компанія існує більше 5 років, вона, ймовірно, має багато різних баз даних, які існують у різних частинах компанії. Вони можуть бути в різних департаментах і в різних країнах. Можливо, вони були включені через придбання. Частиною завдання міграції бази даних є пошук різних баз даних у компанії та планування того, як нормалізувати дані та трансформувати схеми.

Проблема №2. Втрата або пошкодження даних. Під час міграції баз даних важливо переконатися, що ваші дані переміщуються безпечно без втрат чи пошкоджень. Вам потрібно буде спланувати, як перевірити втрату або пошкодження даних, які можуть виникнути під час переміщення даних з однієї системи в іншу.

Проблема №3: Безпека. Переміщуючи дані з однієї платформи на іншу, важливо захистити дані. На жаль, є багато поганих акторів, які хотіли б отримати в свої руки особисту інформацію, яку ви зберігаєте. Ви можете зашифрувати свої дані або видалити особисту інформацію (PI) у рамках процесу міграції.

Міграція бази даних — це багатофазний процес, який включає деякі або всі наступні кроки:

- Оцінювання;

- Перетворення схеми бази даних;

- Міграція даних;

- Тестування та настройка.

Оцінка. Цей етап включатиме збір бізнес-вимог, оцінку витрат і вигод, а також виконання профілювання даних. Профілювання даних - це процес, за допомогою якого можна ознайомитися зі схемою існуючих даних і баз даних. Вам також потрібно буде спланувати, як ви будете переміщувати дані – чи будете ви використовувати інструмент ETL (Extract, Transform and Load), сценарій чи інший інструмент для переміщення даних. Перетворення схеми бази даних. Схема - це схема структури бази даних і змін на основі правил цієї бази даних. Коли ви переміщуєте дані з однієї системи в іншу, вам потрібно буде перетворити схеми, щоб структура даних працювала з новою базою даних. Міграція даних. Після того, як ви виконаєте всі попередні умови, вам потрібно буде перемістити свої дані. Це може включати створення сценаріїв за допомогою інструменту ETL або іншого засобу переміщення даних. Під час міграції ви, ймовірно, конвертуватимете дані, нормалізуватимете типи даних і перевірятимете наявність помилок.

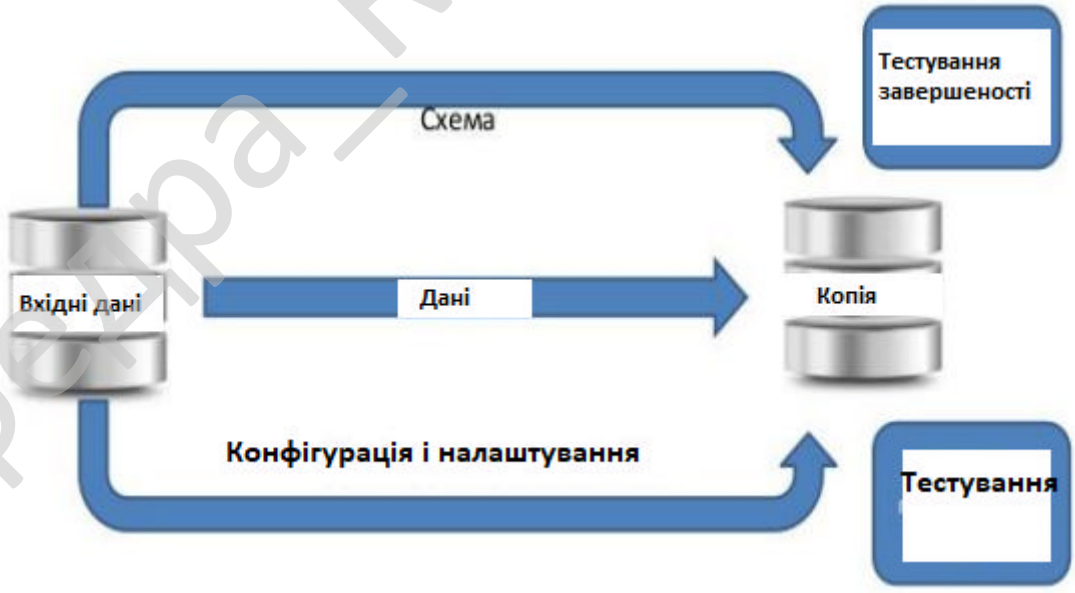

Рисунок 2.1 – Практична міграція даних

Тестування та налаштування. Після переміщення даних вам потрібно переконатися, що дані: були переміщені правильно, повні, не містять пропущених значень, не містять нульових значень і є дійсними. Схеми обох баз даних і самі дані повинні бути ідентичними.

Аналіз представлених на ринку продуктів міграції даних

Міграція дуже часто потрібна в корпоративних системах, якщо компанія здійснює перехід з однієї технології на іншу, або взагалі попередня база застаріла. Отже, міграція – це не нововведення, є кілька варіантів програм міграції, але вони дуже дорогі для малого бізнесу.

Нижче наведено список аналогів системи:

1) Виробник Migration Architect: Evoke Software. Ця система призначена для планування та реалізації наступних проектів: створення інформаційних сховищ та вітрин даних; міграція даних у готові програми; міграція даних і додатків, пов'язана з переходом на нову СУБД; міграція даних при зміні схеми бази даних.

Основні можливості:

- Аналіз не тільки метаданих, але й фактичних даних для більш точного визначення діапазонів значень атрибутів і перевірки наявності первинних ключів.

- Визначення зв'язків між атрибутами (функціональні залежності) і зв'язків між таблицями (виявлення надмірності даних).

- Підтримка простої моделі для зберігання та визначення відповідностей між атрибутами схеми вихідної та цільової бази даних.

2) TRUmigrate Виробник: Valiance partners, http://www.valiancepartners.com

Основні можливості:

- Призначення правил. Перегляди визначаються за допомогою графічного інтерфейсу користувача та зберігаються у файлі XML;

- експорт. Вилучення змісту та призначення інформації з одного або кількох сховищ даних і передача отриманих даних до сховища-посередника;

- приєднання та злиття. Додавання даних з допоміжних джерел і злиття з раніше отриманими даними;

- трансформація. Застосування правил перетворення отриманих даних;

- імпорт. Завантаження вмісту об’єктів у цільову програму та ініціалізація їх параметрами конфігурації.

Служба міграції бази даних AWS і Amazon DinamoDB. Багато організацій розглядають можливість переходу з комерційних реляційних баз даних, таких як Microsoft SQL Server або Oracle, на Amazon DinamoDB, повністю керовану, швидку, високомасштабовану та гнучку службу баз даних NoSQL. Amazon DynamoDB може збільшити або зменшити пропускну здатність на основі трафіку для задоволення потреб бізнесу. Це повністю керована хмарна база даних, яка підтримує моделі зберігання як документів, так і ключових значень. Його гнучка модель даних, надійна продуктивність і автоматичне масштабування пропускної здатності роблять його ідеальним для мобільних пристроїв, Інтернету, ігор, рекламних технологій, Інтернету речей і багатьох інших додатків..

Переваги AWS Database Migration Service:

- простота використання;

- мінімальні простої;

- підтримка поширених баз даних.

Служба міграції бази даних AWS проста у використанні. Вам не потрібно встановлювати драйвери чи програми, і в більшості випадків вам не потрібно вносити жодних змін у вихідну базу даних.

AWS Database Migration Service дозволяє перенести бази даних на платформу AWS практично без простоїв. Будь-які зміни, внесені до вихідної бази даних під час міграції, постійно копіюються в цільову базу даних, тоді як вихідна база даних залишається повністю функціональною. AWS Database Migration Service дозволяє виконувати міграцію даних, використовуючи

найбільш поширені бази даних, як комерційні, так і відкриті, як вихідні та цільові бази даних. Сервіс підтримує як однорідні міграції, наприклад з Oracle на Oracle, так і гетерогенні міграції між різними платформами баз даних, наприклад з Oracle на Amazon Aurora.

Рисунок 2.2 – AWS Database Migration Service

Види міграції даних. Використання міграції даних для досягнення різних цілей призвело до формування окремих видів міграції даних:

1) за кількістю виконаних скриптів в одиницю часу:

- послідовний;

- паралельні.

Наведено пояснення цих типів міграції на прикладі реляційних баз даних. При послідовній міграції за одиницю часу виконується один сценарій. Порядок, у якому виконуються сценарії, базується на залежностях між зв'язками. При паралельній міграції даних виконується більш детальний аналіз залежностей між зв'язками. У цьому випадку виконання скриптів відбувається в момент, коли вже передані дані всіх необхідних відносин.

2) за типом зберігання даних:

- між однотипними сховищами;

- між складами різних типів.

Сховища даних є основою для побудови систем підтримки прийняття рішень. Основна мета створення сховища - зробити всі дані, що стосуються

управління бізнесом, доступними в стандартизованій формі, придатній для аналізу та отримання необхідних звітів. Міграція однорідних баз даних. Під час міграції однорідних баз даних ядра вихідної та цільової баз даних однакові або сумісні одна з одною, наприклад Oracle і Amazon RDS для Oracle, MySQL і Amazon Aurora, MySQL і Amazon RDS для 20 MySQL або Microsoft SQL Server і Amazon RDS для SQL Server. Оскільки структури схем, типи даних і коди вихідної та цільової баз даних сумісні, така міграція виконується за один крок. Ви створюєте завдання міграції, яке визначає підключення до вихідної та цільової баз даних, і запускаєте міграцію одним натисканням кнопки. Все інше виконує служба міграції бази даних AVS. Вихідна база даних може знаходитися у вашій локальній мережі, за межами AWS, працювати на примірниках Amazon EC2 або бути базою даних Amazon RDS. Цільовою базою даних може бути база даних в Amazon EC2 або Amazon RDS.

Міграція різнорідних баз даних. При міграції різнорідних баз даних ядра вихідної і цільової баз даних відрізняються. Це може бути міграція з Oracle в Amazon Aurora, з Oracle в PostgreSQL або з Microsoft SQL Server в MySQL. В цьому випадку структури схем, типи даних і коди вихідної і цільової баз даних сильно відрізняються, і перед міграцією даних необхідно виконати перетворення схеми і коду бази даних. Тому міграція різнорідних баз даних виконується в два етапи. Спочатку використовується інструмент AWS Schema Conversion Tool для конвертації схеми і коду вихідної бази даних у відповідну схему і код цільової бази даних, а потім за допомогою сервісу AWS Database Migration Service виконується міграція даних з вихідної бази даних в цільову. Необхідне перетворення типів даних автоматично виконується сервісом AWS Database Migration Service під час міграції. Вихідна база даних може знаходитися у вашій локальній мережі, поза AWS, працювати в інстанси Amazon EC2 або бути базою даних Amazon RDS. Цільовий базою даних може бути база даних в Amazon EC2 або Amazon RDS.

Загальні принципи виконання міграції даних. Успіх міграції даних залежить від багатьох факторів. Вплив технологічних, людських і даних факторів. Прикладами технологічних факторів є обсяг переданих ними даних і частота їх передачі. Фактори, пов'язані з даними, включають якість даних, подібність вихідних і цільових структур даних, залежності між даними тощо. Як і будь-який IT-проект, проекти міграції даних піддаються ризикам. Постійно зростаючий попит на ефективні IT-рішення, стрімке зростання пропозиції нових технологій і високий рівень конкуренції в IT-секторі значно збільшили потребу в міграції даних з одного середовища в інше. З частою передачею даних кількість помилок також зросла. Брак досвіду та знань є основною причиною головного болю міграції даних. Найпоширеніші проблеми компанії:

- бюджет;
- простота;
- безпека процесу;
- екологічна сумісність;
- досвід співробітників;
- поведінка бізнес-процесів в нових умовах;
- час;
- тестування.

Занадто часто проекти міграції зазнають невдачі, оскільки непрофесійний або необережний план міграції призводить до додаткових витрат або навіть збитків для бізнесу під час простою. Міграцію даних слід розглядати як інвестицію, яка потрібна одночасно з впровадженням нової IT-системи. Не планувати витрати на міграцію під час впровадження нової IT-платформи - це все одно, що не планувати витрати на паливо при купівлі нового автомобіля. Самі дані відіграють значну роль у міграції. Від того, як дані інтегруються в бізнес, як вони змінюються з часом, певні набори даних стають взаємозалежними, що ускладнює їх перенесення в нове середовище. Для покращення керування даними та мобільності додатків важливо досліджувати

дані, додатки та системні компоненти, що використовуються, на предмет складності та взаємозалежності. Оскільки майже всі дані інтегровані в програми, рівень їх інтеграції є важливою складовою процесу міграції.

Успішна міграція даних залежить від точних деталей, визначених у плані міграції. Це не лише план, який описує, що робити, але також коли це робити та що робити у разі помилки чи проблеми. Добре продуманий і точний план потребує додаткового часу, досвіду та знань. План визначає, який метод використовуватиметься для імпорту та експорту даних, відновлення мережі та підготовки ресурсів. Один із способів зменшити ризики для проектів міграції даних - розділити їх на етапи або етапи. Так, існує процес міграції даних, який складається з семи фаз: стратегія, аналіз, дизайн, компіляція, тестування/впровадження, перевірка, обслуговування, а іноді лише з трьох фаз – аналіз, тестування та виконання.

На прикладі міграції з СУБД Oracle на СУБД Microsoft SQL Server при використанні процесу міграції бази даних, що складається з семи кроків, [12] описано можливості використання програмних продуктів DBBest на кожному кроці та показано відсоток завдань, що виконуються в процесі міграції. даних при використанні описаних програмних продуктів. Такі розбивки дозволяють покращити управління проектами, а також вибрати відповідні програмні продукти для виконання окремих завдань або процесу міграції даних в цілому. У літературі вказуються функції, які повинні мати програми передачі даних: журналювання, звітування про помилки.

Важливою частиною процесу міграції даних є тестування. Це дозволяє гарантувати відсутність дефектів у цільовому сховищі даних, сховище містить усі необхідні дані, дані перенесені до необхідних структур тощо. Незважаючи на важливість цього аспекту, тестування залишається недостатньо висвітленим при описі методів і підходів до міграції даних. Тестування міграції даних життєвого циклу, огляд методів тестування та контроль якості описані в [27]. Основою механізму контролю якості процесу перевантаження даних є аналіз кодів

повернення та журналів усіх комунальних служб, задіяних у процесі перевантаження, моніторинг цілісності за допомогою засобів БД та додатковий контроль результатів перевантаження – верифікація. Перевірка підтверджує, що всі 23 дані були успішно завантажені в нову базу даних із задокументованими необхідними змінами. Важливість перевірки під час міграції бази даних такого масштабу важко недооцінити. По-перше, схема містить понад 2000 таблиць і майже 10 000 стовпців, причому деякі таблиці містять десятки мільйонів записів. По-друге, зовнішні ключі, як засіб контролю цілісності, практично не використовуються у вихідній базі даних. Процес перевірки має впливати не лише на вміст об'єктів, але й на всю схему даних, перевіряючи наявність і стан об'єктів. Повна перевірка для визначення ідентичності двох баз такого розміру вимагає багато ресурсів, особливо часу. Отже, процес потребує підходу, який потребує значно менше ресурсів і водночас має високий ступінь достовірності результатів.

Підзадачі міграції даних. Відомі підходи та методи міграції даних явно або неявно виконують аналіз сховищ даних і міграцію даних. У рамках аналізу сховищ даних виконується порівняння або порівняння структур вихідного та цільового сховищ даних. Зіставлення схем відноситься до завдання визначення семантичних відповідностей між елементами схеми [2]. Зіставлення схем традиційно вважається завданням, яке виконує ШІ, і це накладає обмеження на варіанти його вирішення. Класифікації підходів схем узгодження можна знайти в оглядах [5,12]. У літературі як синонім порівняння схем використовується термін «порівняння схем» 2 . Ідеї запропонованих методів і підходів до схематичного відображення базуються на використанні відображення графів [2], теорії ймовірностей [3], машинного навчання 24 [3] та інших галузей [21]. Найважливішими характеристиками методів і підходів до порівняння схем є точність результатів і складність [23]. Оскільки AI-повнота відображення схеми вимагає корекції результатів відображення людиною, а великі розміри схем збільшують час виконання відображення, надто багато уваги приділяється роботі

запропонованих алгоритмів. Наприклад, при використанні відображення графів слід враховувати, що в загальному випадку задача відображення графів не відноситься ні до класу P, ні до класу NP-повних задач, а проблема ізоморфізму підграфів (одна з типи відображення графів [7]) є NP -повним [4]. Тому при необхідності порівняння схем вибір способу його виконання слід здійснювати, керуючись допустимими співвідношеннями точності та трудомісткості. При міграції даних між сховищами даних різних типів подання структур ускладнюється використанням різних моделей даних. Наприклад, розглядається міграція реляційних баз даних до об'єктно-орієнтованих або XML баз даних. Складність полягає в тому, що часто не існує готових або однозначних правил порівняння елементів різних моделей даних. Можливі кілька підходів до відображення елементів різних моделей даних. Одним із таких підходів є визначення правила порівняння для кожної можливої пари моделей даних. Другий підхід заснований на використанні проміжного концептуального подання, в якому будуть представлені вихідна і цільова структури даних. Таким чином, графи [2, 3], реляційні моделі [25], семантичні моделі [5] та XML-файли [8] пропонуються як проміжні представлення. Недоліками такого підходу є можлива втрата семантики при переході на проміжне представлення, а також збільшення складності. Очевидно, що міграція даних між сховищами даних одного типу не стикається з проблемами, описаними вище, але в той же час її масштаби обмежені. Однак, навіть коли дані переносяться між сховищами даних одного типу, виникають труднощі під час порівняння структур даних. Вони стосуються відмінностей у конкретних реалізаціях сховищ даних [12, 2]. Наприклад, в одному сховищі даних довжина типу даних «рядок» 25 обмежена, а в іншому ні; в реляційних СУБД можуть використовуватися різні стандарти мови SQL.

Різні уявлення об'єктів моделюється світу і зв'язків між ними також ускладнюють порівняння структур даних вихідних і цільових сховищ даних. Говорячи термінами ER–моделі Чена, одне й те саме явище, що моделюється

світу може бути представлено як атрибут, як безліч сутностей і як безліч зв'язків. Таким чином, з'являється проблема зіставлення об'єктів і їх зв'язків в силу можливості використання різних їх уявлень. До цієї проблеми також стосується використання еквівалентних конструкцій моделей даних і різних точок зору при поданні об'єктів моделюється світу, наприклад, використання різних імен (синонімів, гіперонімів), обмежень цілісності, типів даних.

Класифікації можливих відмінностей між уявленнями об'єктів і їх зв'язків наведені в [5, 21]. Здійснення перенесення даних пов'язано з рішенням деяких питань. Одним з них є визначення порядку перенесення даних. Розв'язання даного питання залежить від наявності зв'язків між об'єктами модельованого світу, представленими в сховище даних. У разі, коли в сховище даних представлений ще один тип об'єктів моделюється світу без зв'язків (наприклад, для реляційної моделі це означає, що є тільки одне відношення, яке не має зовнішніх ключів), складнощів у визначенні порядку перенесення даних немає. Випадок, коли в сховище даних представлено кілька типів об'єктів без зв'язків, також не приносить особливих труднощів: перенесення виконується для кожного типу об'єктів в будь-якому порядку.

2.2 Обґрунтування вибору методів розробки

Запропонований процес орієнтований на виконання наступних сценаріїв міграції даних, що зберігаються в реляційних базах даних: перенесення даних до бази даних зі схемою, зміненою в результаті модифікації, замовленої користувачем, або зміни версії програми, яка використовує база даних;

- перехід на іншу версію СУБД;
- перехід на іншу реляційну СУБД;
- різні комбінації вищевказаних пунктів.

Рисунок 2.3– Процес міграції у простому вигляді

Етапи такого процесу міграції даних:

- вилучення схем вихідної та цільової баз даних;
- побудова послідовностей для обходу схем БД;
- порівняння схем бази даних;
- призначення правил передачі даних;
- передача даних.

Реалізація першого етапу передбачає представлення схеми бази даних в оперативній пам’яті у вигляді, що містить інформацію, необхідну для передачі даних, і приховує деталі роботи з конкретною СУБД від інших етапів процесу міграції даних. Ця інформація потрібна протягом усього процесу, тому якщо її не отримати, процес буде перервано. Другий етап вирішує завдання правильного перенесення даних з вихідної бази даних, а також формує інформацію, необхідну для наступних етапів. Під порядком обходу схеми бази даних розуміється ряд відносин, сформованих за принципом топологічного сортування орієнтованого графа з деякими ознаками. Такий масив будується для забезпечення збереження посилальної цілісності під час міграції даних, а також використовується при порівнянні схем бази даних. Порівняння схем у запропонованому процесі міграції даних є допоміжним етапом, результатом якого буде підказка для людини при призначенні правил передачі – алгоритмічних дій над кортежами з відношень, які визначають, як, куди і які дані потрібно передавати**.**

Рисунок 2.4 – Схема вихідної бази даних

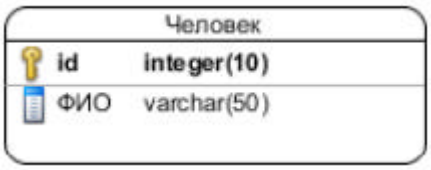

Рисунок 2.5– Схема цільової бази даних

На формування правил передачі даних перш за все впливають відмінності між схемами вихідної та цільової баз даних. Рисунки 2.4 і 2.5 ілюструють такий випадок. Однак варто зазначити, що існують правила міграції даних, які застосовуються до перетворень самих даних, незалежно від відмінностей між схемами бази даних. Прикладом такого правила є формування одного кортежу в цільовій базі даних з кількох кортежів із вихідної бази даних із точним відповідністю схеми. Передбачається, що правила перенесення будуть написані у вигляді скриптів, а шаблони скриптів будуть надані для призначення правил перетворення в умовних випадках, наприклад, при перейменуванні атрибутів.

На етапі налаштування правила також налаштовується виконання передачі даних шляхом вибору однієї з можливих стратегій обробки помилок:

- зупинка при першій помилці – процес міграції завершується при першій помилці;

- виправлення помилок користувачем - при виникненні помилки користувачеві пропонується виправити дані, після чого продовжується міграція даних, починаючи зі спроби передачі даних, на яких сталася помилка;

- пропускати кортежі з помилками – при виникненні помилки кортежі з некоректними даними та всі залежні від них пропускаються; триває міграція даних; заповнення значеннями за замовчуванням - в цьому випадку кортежі з

неправильними даними виправляються за допомогою значень за замовчуванням; Виконується міграція даних, яка намагається перенести власні дані.

Прикладом помилки передачі даних є перетворення нечислового рядка на число.

На останньому етапі дані передаються з вихідної бази даних у цільову базу даних, ідея якої полягає в концепції кортежної залежності. Незалежний кортеж - це кортеж із зв'язками, який не має зовнішніх ключів, або кортеж із зв'язками, який має зовнішні ключі, значення яких для цього кортежу дорівнюють 30 невизначеним (NULL). В іншому випадку кортеж називається залежним. Дані передаються за таким принципом: спочатку передаються незалежні кортежі, потім залежні. Варто зазначити, що залежний кортеж можна перенести лише тоді, коли всі набори, від яких він залежить, уже перенесено. Механізм передачі даних використовує два інших типи сценаріїв на додаток до сценаріїв перетворення, які визначають правила передачі. Перший тип скриптів служить для відключення деяких обмежень цілісності в цільовій базі даних, за наявності яких не може бути виконана передача даних, а другий - для їх увімкнення після передачі даних..

2.3 Розгорнута постановка завдання

Відповідно до технічного завдання на магістерську роботу передбачено впровадження програмного забезпечення, яке демонструватиме роботу ООБД.

У процесі написання магістерської роботи необхідно виконати наступний обсяг робіт:

- Дослідження процесів і механізмів міграції даних і впровадження програмного продукту, який мігрує дані з бази даних MS Server до PostgreSQL.

Серед завдань можна виділити:

1. Отримання метаінформації про схеми баз даних

2. Порівняння схем

3. Побудова послідовності обходу вихідної схеми БД

4. Передача даних з перетвореннями - виконується.

В рамках даної роботи вирішуються останні три завдання, а саме:

- аналіз різних топології схем бази даних і встановлення правила перевизначення схем БД для конкретної топології;

- на основі отриманих правил розробити алгоритм побудови серії обходів схеми вихідної бази даних, яка надалі визначатиме порядок, в якому дані будуть передаватися з вихідної в цільову базу даних;

- Розробити механізм передачі даних.

У рамках даної роботи необхідно дослідити механізми міграції даних та реалізувати програмний продукт, який виробляє міграцію даних. На даний момент не існує єдиного підходу до вирішення проблем міграції даних. Компанія-розробник програмного забезпечення не завжди підтримує можливість збереження накопичених даних при переході на нову версію програмного продукту. За бажанням клієнта ця проблема вирішується індивідуально, а послуга зазвичай коштує чималих грошей. Розробники повинні аналізувати всі зміни і кожного разу створювати нові процедури передачі даних для конкретного випадку зміни схеми бази даних. При цьому кожна компанія дотримується власної політики вирішення цього завдання, яка часто не розголошується. Тому постає питання дослідження проблеми міграції даних, що зберігаються в реляційній СУБД, і написання відповідного програмного продукту.

Передбачається, що цей програмний продукт буде виконувати автоматичний аналіз змін схеми бази даних і генерувати правила передачі, надаючи користувачеві можливість вибору альтернатив у разі виникнення спірних ситуацій і помилок..

Основні етапи роботи

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи

технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран повідомлень про некоректні дії користувача та нестандартні ситуації;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи з охорони праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Виходячи з теми магістерської роботи, необхідно розробити програмне забезпечення для міграції баз даних з однієї СУБД в іншу.

Успіх міграції даних залежить від багатьох факторів. Вплив технологічних, людських і даних факторів. Прикладами технологічних факторів є обсяг переданих ними даних і частота їх передачі.

Фактори, пов'язані з даними, включають якість даних, подібність вихідних і цільових структур даних, залежності між даними тощо. Як і будь-який IT-проект, проекти міграції даних піддаються ризикам. Зростаючий попит на ефективні IT-рішення, стрімке зростання пропозиції нових технологій і високий рівень конкуренції в IT-секторі значно збільшили потребу в міграції даних з одного середовища в інше. З частою передачею даних кількість помилок також зросла. Брак досвіду та знань є основною причиною головного болю міграції даних. Найпоширеніші проблеми компанії:

- бюджет;
- простота;
- безпека процесу;
- комбінувати середовища;
- досвід співробітників;
- поведінка бізнес-процесів в нових умовах;
- час;
- тестування.

Занадто часто проекти міграції зазнають невдачі, оскільки непрофесійний або необережний план міграції призводить до додаткових витрат або навіть збитків для бізнесу під час простою. Міграцію даних слід розглядати як інвестицію, яка потрібна одночасно з впровадженням нової IT-системи.

Не планувати витрати на міграцію під час впровадження нової IT-платформи - це все одно, що не планувати витрати на паливо при купівлі нового автомобіля. Самі дані відіграють значну роль у міграції. Від того, як дані інтегруються в бізнес, як вони змінюються з часом, певні набори даних стають взаємозалежними, що ускладнює їх перенесення в нове середовище.

Для покращення керування даними та мобільності додатків важливо досліджувати дані, додатки та системні компоненти, що використовуються, на предмет складності та взаємозалежності. Оскільки майже всі дані інтегровані в програми, рівень їх інтеграції є важливою складовою процесу міграції. Успішна міграція даних залежить від точних деталей, визначених у плані міграції. Це не лише план, який описує, що робити, але також коли це робити та що робити у разі помилки чи проблеми.

Добре продуманий і точний план потребує додаткового часу, досвіду та знань. План визначає, який метод використовуватиметься для імпорту та експорту даних, відновлення мережі та підготовки ресурсів.

Один із способів зменшити ризики для проектів міграції даних — розділити їх на етапи або етапи. Таким чином, існує процес міграції даних, який складається з 7 фаз: стратегія, аналіз, проектування, збірка, тестування [17] / впровадження, верифікація, підтримка, а іноді лише з трьох фаз – аналіз, тестування та виконання [8].

На прикладі міграції з СУБД Oracle на СУБД Microsoft SQL Server при використанні процесу міграції бази даних, що складається з 7 кроків, [12] описано можливості використання програмних продуктів DB Best на кожному кроці та показано відсоток завдань, що виконуються в процесі міграції. даних при використанні описаних програмних продуктів. Такі розбивки дозволяють покращити управління проектами, а також вибрати відповідні програмні продукти для виконання окремих завдань або процесу міграції даних в цілому. У літературі вказуються функції, які повинні мати програми передачі даних: журналювання, звітування про помилки. Важливою частиною процесу міграції

даних є тестування. Це дозволяє гарантувати відсутність дефектів у цільовому сховищі даних, що сховище містить усі необхідні дані, що дані передаються до необхідних структур тощо.

Незважаючи на важливість цього аспекту, тестування залишається недостатньо висвітленим при описі методів і підходів міграції даних [8, 12, 17]. Життєвий цикл тестування міграції даних описано в [4], огляд методів тестування та контролю якості наведено в [27]. Основою механізму контролю якості процесу перевантаження даних є аналіз кодів повернення та журналів усіх комунальних служб, задіяних у процесі перевантаження, моніторинг цілісності за допомогою засобів БД та додатковий контроль результатів перевантаження – верифікація. Перевірка гарантує, що всі дані було успішно завантажено до нової бази даних із задокументованими необхідними змінами. Важливість перевірки під час міграції бази даних такого масштабу важко недооцінити. По-перше, схема містить понад 2000 таблиць і майже 10 000 стовпців, причому деякі таблиці містять десятки мільйонів записів. По-друге, ключі, як засіб контролю цілісності, практично не використовуються у вихідній базі даних. Процес перевірки має впливати не лише на вміст об’єктів, але й на всю схему даних, перевіряючи наявність і стан об’єктів. Повна перевірка для визначення ідентичності двох баз такого розміру вимагає багато ресурсів, особливо часу. Отже, процес потребує підходу, який потребує значно менше ресурсів і водночас має високий ступінь достовірності результатів.

3.2 Розробка структурної схеми.

Структурна діаграма надає інформацію про взаємодію майбутніх частин програми, за яку функціональність вони будуть відповідати і як вони будуть спілкуватися між собою. Не можна сказати, що вони інформативні, тому що не можна простежити, як дані на них передаються і змінюються. Тому залежно від технічного завдання створюють структурні схеми окремих частин програми.

Метод покрокової деталізації зазвичай використовується для розробки структурної діаграми, щоб визначити всі компоненти, які складають діаграму, а також набори підпрограм і пов’язаних бібліотек. Структурними компонентами програми можна назвати підсистеми, бази даних, бібліотеки тощо. А за допомогою посилань можна продемонструвати, як вони спілкуються між собою, обмінюються інформацією, підключають нові бібліотеки. При розробці структурної схеми програмного забезпечення були розглянуті основні модулі програми та їх взаємодія. Дуже важливо під час розробки, особливо якщо проект великий і зі складною архітектурою, розділити програму на структури, щоб відобразити логіку програми, показати, де підключені бібліотеки і за що вони відповідають. У заданій схемі взаємодія модулів відображається в ієрархічній схемі. Однак там, де розроблені модулі підключені до основного модуля, схема не відображає функціонування системи. Схема доповнюється розшифровкою функцій, що з'єднують модулі.

Структурна схема (рисунок 3.1) показує зовнішні специфікації програми, які дають розуміння функціональної частини програми, за що вони відповідають, як взаємодіють із вхідними та вихідними даними. Особливо важливо розуміти тип структури даних.

Рисунок 3.1 - Структурна схема

Поділ програми на окремі модулі здійснювався за принципом від

загального до конкретного, при цьому окремі модулі виступали як сервіси, які виконували окремі завдання. Хоча існують інші варіанти створення ієрархій, циклічного перегляду кількох параметрів і підтримки реалізації шаблону MVC, цей дистрибутив забезпечує найзручніший спосіб налаштування частин, які буде легко підтримувати та тестувати. Структурна схема відображає головний принцип програми, її основні та другорядні блоки, їх призначення та взаємну взаємодію. Це допомагає зрозуміти, як працює програма, що полегшує її подальшу підтримку.

3.3 Розробка функціональної схеми

Функціональна схема, або також звана схемою даних, відображає взаємодію компонентів програми, вказуючи склад даних, а також їх призначення. При створенні такої схеми використовуються позначення, прийняті стандартом. Розроблена функціональна схема допомагає відстежувати виконання існуючих вимог і функцій, відстежувати, які компоненти і модулі використовуються, як вони взаємодіють один з одним, за допомогою вхідних і вихідних даних. Така схема відображає роль кожного функціонального елемента в додатку і з якими компонентами він взаємодіє. Для розробки даної схеми за основу була взята конструктивна схема. Переглядаючи функціональну схему, ви можете прослідкувати принцип роботи всієї програми, як взаємодіють між собою клієнтська та серверна частини, а також які бібліотеки та технології використовуються для зберігання та передачі даних.

Цей тип діаграм зазвичай використовується для наочної демонстрації роботи алгоритму в спрощеній формі, щоб ви могли стежити за функціями та діями, які виконує алгоритм, які дані він використовує та який результат отримує.

Це допомагає оцінити складність алгоритму, зрозуміти принципи його роботи та контролювати можливі зміни та коригування. Подивіться, як функціональні схеми пов’язані одна з одною. Для опису компонентів та їх

взаємодії використовувався національний стандарт. Там, де прямокутник показує окремі функціональні частини або як вони об'єднані у функціональні групи, причому кожна група має власну позначку та назву.

Є кілька можливостей представлення функціональної схеми, використовуючи принципову схему або змішану структуру, але в будь-якому випадку функціональна схема повинна відображати призначення пристрою.

На схемі (рисунок 3.2) показано:

- Умовна позначка в прямокутнику, якщо це функціональна частина

- Умовне графічне позначення, що відображає положення і дію функціональної частини.

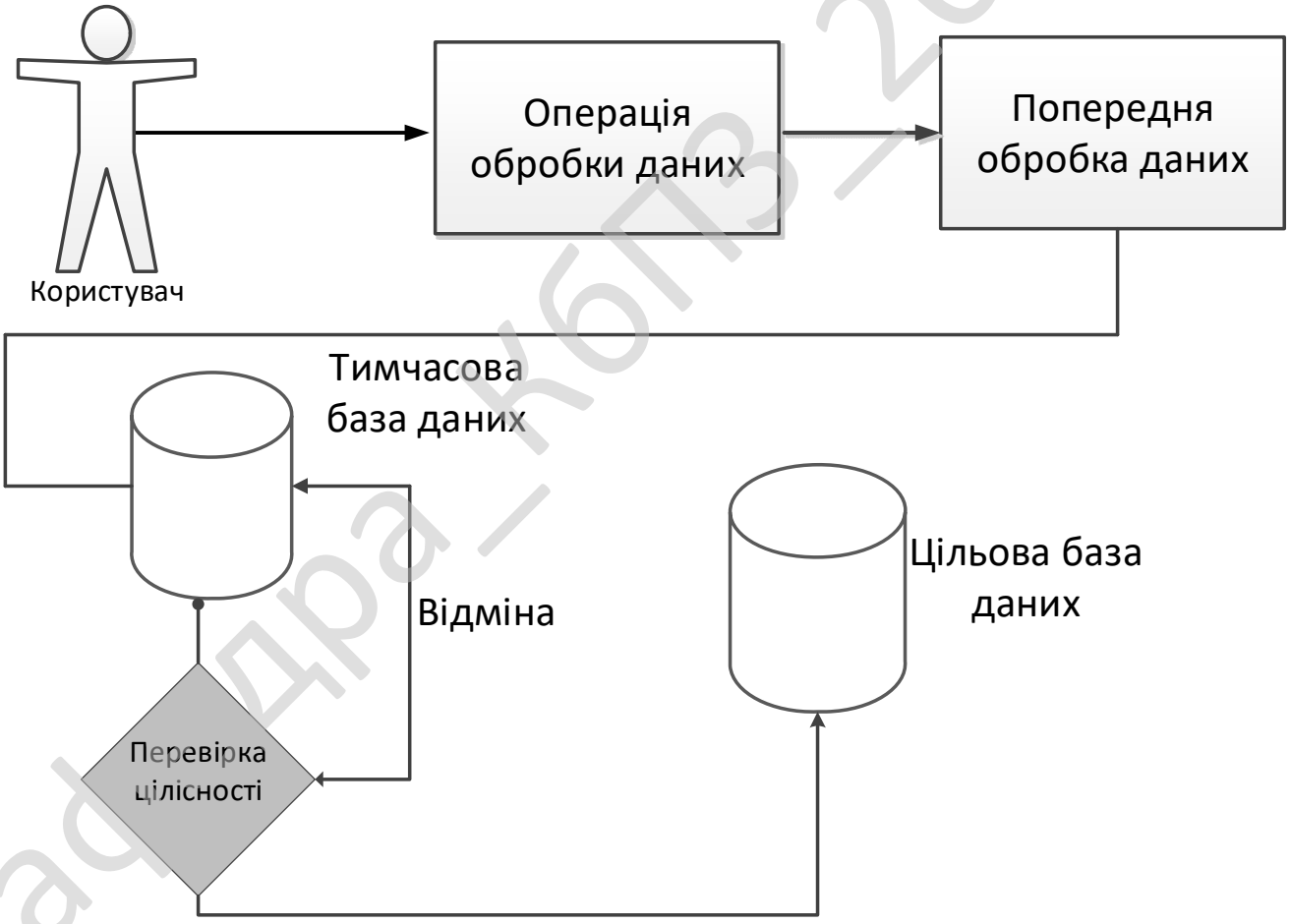

Рисунок 3.2 - Функціональна схема системи

У випадках, коли використовується принципова схема, функціональні частини разом із позначеннями розташування компонентів мають бути подібними до схеми. У таких випадках список елементів не використовується через використання даних принципової схеми. Коли функціональна схема розробляється без використання схеми, для представлення функціональних частин використовуються загальні правила.

3.4 Розробка діаграми процесів

При моделюванні поведінки розробленої системи можна відслідковувати, як змінювалася робота програми, її стан, вхідні та вихідні дані, і як все це вплинуло на розробку програми. Як працюють алгоритми в додатку і які результати вони дають в залежності від вхідних даних. На діаграмі процесу за допомогою графічних об’єктів можна побачити в зручному для сприйняття форматі, які операції виконуються та як вони взаємопов’язані за допомогою стрілок. На діаграмі (рисунок 3.3) можна побачити всі фази програми, від самого початку програми, коли користувач робить запит, як він потрапляє на сервер, потрапляє на контролер, обробляється, отримує дані з бази даних. і повертає його як відповідь користувачеві. Об’єкти на схемі позначають процеси, які виконуються під час роботи програми, стрілки показують, як переходить управління від одного об’єкта до іншого. Існують також структурні діаграми, вони схожі на діаграми процесів, але вони відображають статичність даних, такий опис більше описує зв'язки об'єктів, тоді як діаграми процесів відображають динаміку програми.

Головна особливість у діаграмі процесу — динаміка, на відміну від інших, де превалює сценарій. На таких діаграмах сценарій не повинен конфліктувати з порядком виконання програми і бути безперервним. Проблеми виникають, коли об'єкт повинен знаходитися в кількох місцях одночасно. Це відбувається, коли в послідовності дій у програмі дотримується нечітка логіка.

Метою створення діаграми є розуміння послідовності дій процесу в програмі для отримання кінцевого результату. Спостерігаючи за тим, як працюють процеси, які дані вони отримують, як вони їх обробляють і передають користувачеві, ви можете передбачити, як має працювати програма.

Існують різні способи побудови діаграм, які допомагають слідкувати за логікою та послідовністю дій у процесах, але найкраще зарекомендував себе покроковий алгоритмічний метод побудови діаграм.

З усіх згаданих методів і технічних рішень у проектуванні та розробці програмного забезпечення, дивлячись на результати, можна зробити висновок, що я вибрав оптимальні дизайнерські рішення, які відповідають основним потребам, які були поставлені переді мною на початку проекту. Як видно з результатів, програма виконує всі поставлені завдання.

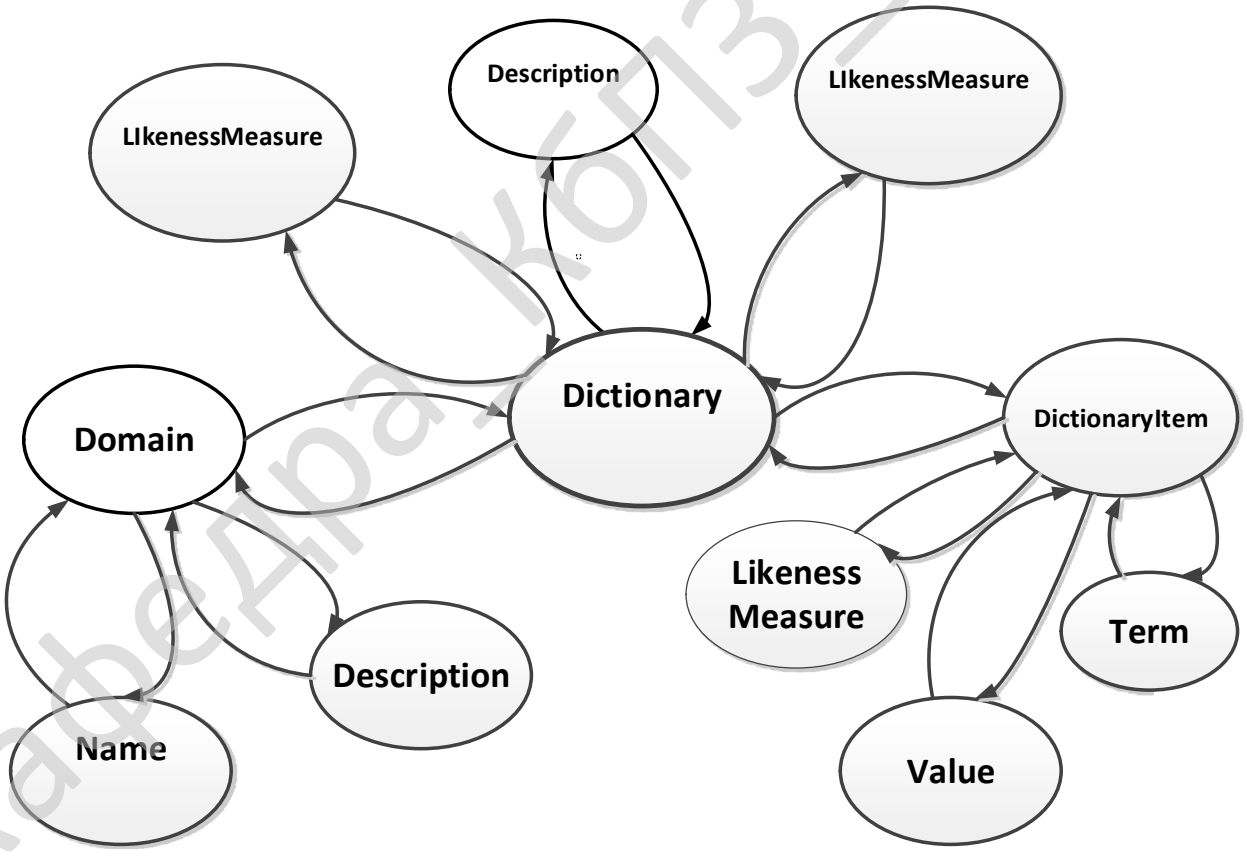

Рисунок 3.3 – Діаграма процесів системи

На розробленій схемі показано роботу програми, як клієнтської, так і серверної частини. Вона демонструє, як окремі частини програми взаємодіють між собою і який результат отримує користувач при роботі з програмою.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ ТА ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ І ПРОГРАМНИХ РІШЕНЬ

Технології для реалізації програми. Для реалізації програми міграції даних були обрані такі технології:

- технологія Java;
- база даних;
- брокер Hibernate;
- платформа Eclipse RCP.

Технологія Java – це і мова програмування, і платформа. Мова програмування Java - потужна й універсальна мова високого рівня, заснована на парадигмі об’єктно-орієнтованого програмування. Платформа Java складається з двох компонентів: віртуальної машини Java (JVM) та інтерфейсу прикладного програмування Java (API). Вихідний код програми Java перетворюється на байт-код, який виконується на JVM. JVM - це програма, реалізована для різного апаратного забезпечення та багатьох операційних систем, що забезпечує крос-платформну сумісність програм Java. JVM поставляється з набором стандартних бібліотек, які разом складають Java Runtime Environment (JRE). Відсутність такого способу виконання програм часто називають зниженням продуктивності через інтерпретацію байт-коду віртуальною машиною. Проте ряд удосконалень, реалізованих у пізніших версіях Java, дозволяють досягти продуктивності, близької до продуктивності «рідного» коду, без негативного впливу на такі функції, як переносимість.

Сьогодні Java включає багато різних сервісів, технологій, компонентів, наприклад, JDBC, JMF, RMI, JINI, JNI, що також полегшує написання програм. База даних HSKLDB (HyperSKL DataBase), реляційна база даних, написана на Java з відкритим кодом, була обрана для зберігання словників, які

використовуються для порівняння схем бази даних. Він поширюється за власною ліцензією, близькою до ліцензії BSD, і сумісний з усіма основними ліцензіями на програмне забезпечення з відкритим кодом. HSKLDB майже повністю підтримує стандарт ANSI–92 SQL і Основні положення стандарту SQL: 2008. Він невеликий за розміром і забезпечує швидкий процесор бази даних, який підтримує багаторазові та транзакційні БД.

HSQLDB може використовуватися і як окремий сервер з підтримкою мережевих з'єднань по JDBC, і як вбудована СУБД. Для забезпечення роботи зі словниками обраний Hibernate – високо продуктивний брокер, призначений для вирішення завдань об'єктно–реляційного відображення, для Java. Hibernate є вільним програмним забезпеченням з відкритим вихідним кодом, поширюваним на умовах GNU LGPL, і надає легкий у використанні фреймворк, найбільш корисний при використанні об'єктно– орієнтованих моделей предметних областей і бізнес логіки в проміжному шарі, написаному на Java.

Eclipse RCP (Eclipse Rich Client Platform) – платформа, що забезпечує можливість створення додатків з використанням потужної універсальної оболонки при мінімальній кількості необхідних плагінів. Ідея цієї технології виникла при розробці чергової версії Eclipse IDE і з'явилася у версії 3. Eclipse IDE – інтегроване середовище розробки, що підтримує широкий спектр інструментаріїв для мов і систем і володіє такими характеристиками, як кросплатформеність, модульність і іншими. До версії 3 в Eclipse IDE підтримувалася розробка плагінів тільки для розширення безпосередньо середовища розробки Eclipse, а з версії 3.0 Eclipse перестав бути монолітною IDE, що підтримує розширення, а сам став набором розширень. Використання фреймворків OSGi і SWT / JFace, що лежать в основі, дозволило створити платформу для розробки повноцінних клієнтських додатків RCP. Така платформа являє собою мінімальний набір плагінів, необхідних для створення RCP додатки і використовуються разом. Поверх цієї платформи була створена

платформа Eclipse, що представляє собою набір розширень RCP – перспективи, редактори, навігатори, модуль Java Development Tools (JDT) та інші.

Отримана платформа має низку переваг, які визначають її придатність для розробки повноцінних клієнтських додатків: модульність, кросплатформна сумісність, багатомовна підтримка, безкоштовність, велика кількість існуючих полігонів, бібліотек і фреймворків. Публічна ліцензія Eclipse дозволяє використовувати створені програми в комерційних цілях. Отримана платформа має ряд переваг, які визначають її придатність для розробки повноцінних клієнтських додатків: модульність, багатомовність, безкоштовність, велика кількість існуючих полігонів, бібліотек і фреймворків. Публічна ліцензія Eclipse дозволяє використовувати створені програми в комерційних цілях. Отримана платформа має ряд переваг, які визначають її придатність для розробки повноцінних клієнтських додатків: модульність, багатомовність, безкоштовність, велика кількість існуючих полігонів, бібліотек і фреймворків. Публічна ліцензія Eclipse дозволяє використовувати створені програми в комерційних цілях.

Особливості реалізації порівняння схем бази даних На реалізацію запропонованого методу порівняння схем бази даних у програмі, що розробляється, вплинули обрані технології. Мова Java визначила використання JDBC [20] у реалізації програми. JDBC дозволяє використовувати вбудоване відображення типів даних різних СУБД на типи даних JDBC, а також отримувати додаткову інформацію про зв’язки, атрибути, типи даних: автоінкрементне значення (autoincrement), можливість використовувати атрибути з певним тип даних у реченні SQL WHERE (з можливістю пошуку), прийняття невизначених значень (з можливістю null), кількість маленьких символів (десяткових цифр), якщо для 46 можна застосувати тип даних, чутливість до регістру (caseSensitive), загальна кількість значущих цифр (точність), можливість використання типу даних для грошових значень (fikedPreScale) .

Для обчислення ступеня подібності імен зв’язків і атрибутів були застосовані такі методи: порівняння на основі максимального підрядка, відстані

Левенштейна, використання словника порівняння без урахування регістру для рівності. Діаграма класів для порівняння імен відношень і атрибутів показана на рисунку 4.1.

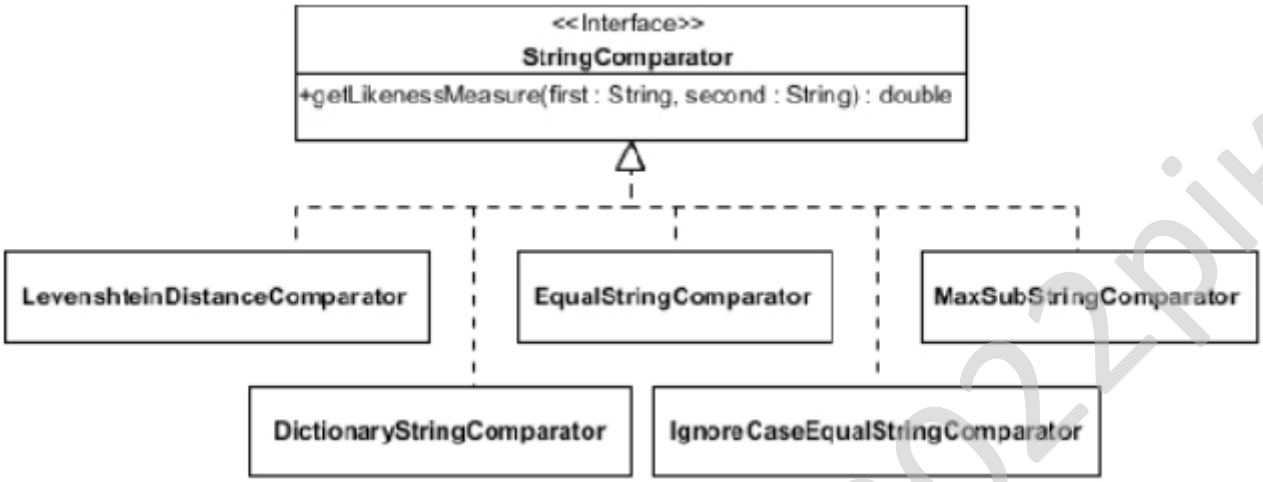

Рисунок 4.1 – Діаграма класів для порівняння імен відношень і атрибутів

Для обробки додаткової інформації про імена відношень і атрибутів, що дозволяє поліпшити результат порівняння, наприклад, про додавання префікса, можуть використовуватися конвертори, які виконують попередню обробку порівнюваних рядків. Реалізовані на даний момент перетворювачі показані на діаграмі класів, показаній на рисунку 4.2.

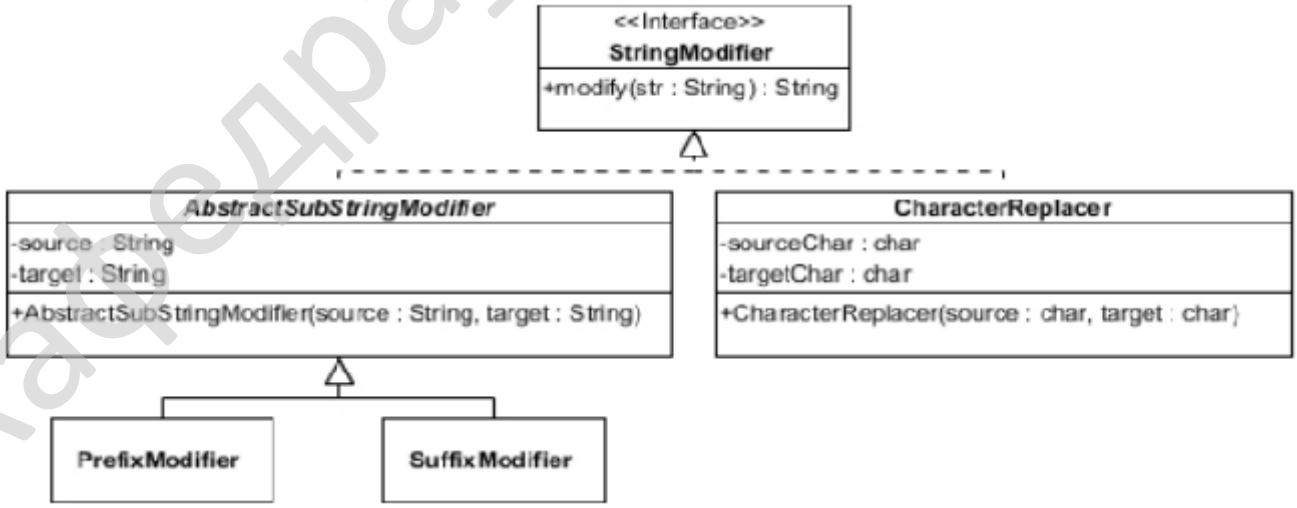

Рисунок 4.2 – Діаграма класів для попередньої обробки імен відношень та атрибутів

Для представлення схем баз даних в ОЗП у вигляді, що містить необхідну інформацію і приховує деталі роботи з конкретною СУБД, використовується модель, діаграма класів якої зображена на рисунку 4.3. На цій діаграмі операції класу опущено для зручності відображення. Клас Schema на діаграмі відповідає концепції "схеми", яка використовується в деяких СУБД (наприклад, PostgreSQL)

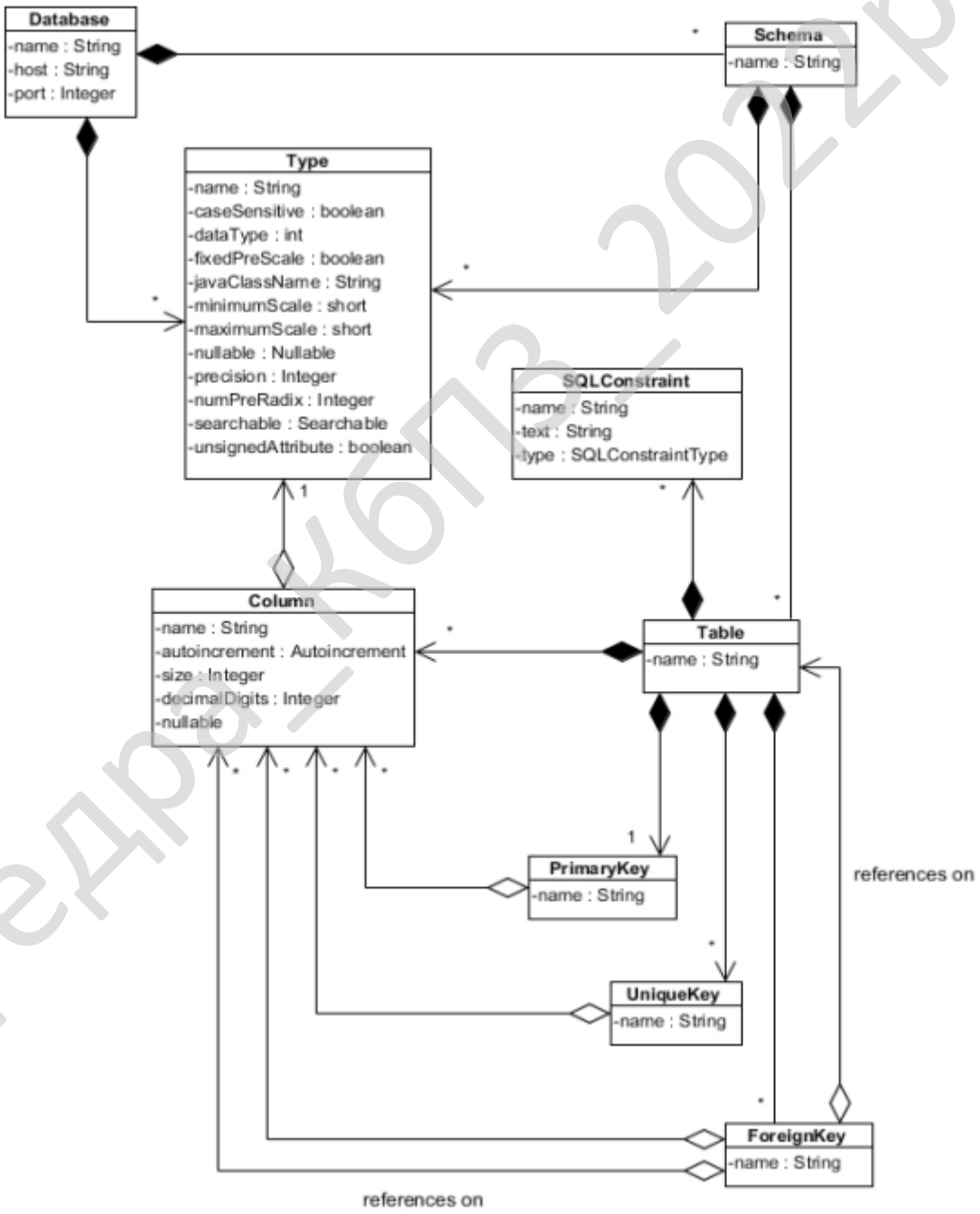

Рисунок 4.3 – Діаграма класів

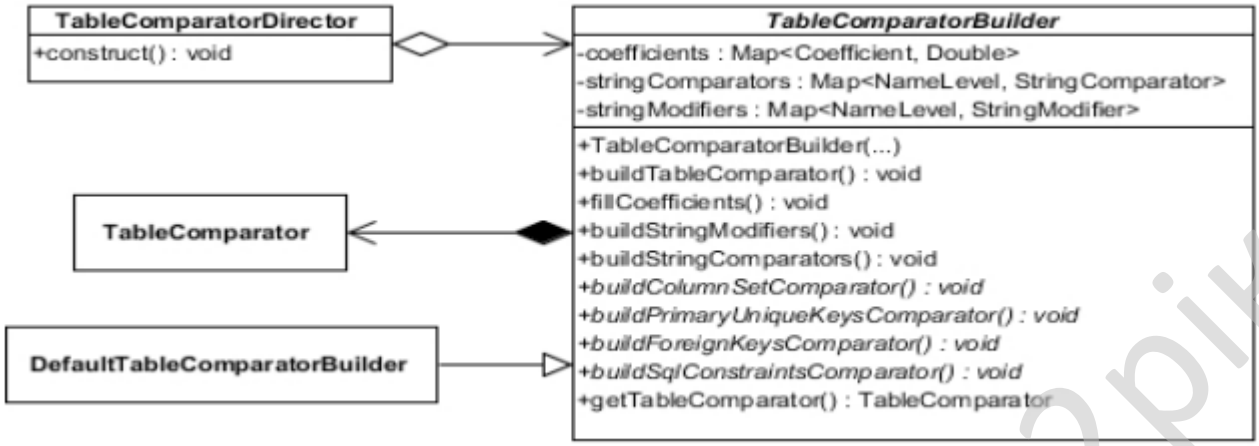

Рисунок 4.4 – Діаграма класів для обчислення заходів подібності

4.1 Розробка блок-схем та опис алгоритмів функціонування системи

Блок-схема дає можливість представити роботу програми в зручній для розуміння формі за допомогою графічних блоків і елементів. За допомогою графічних блоків можна представити потоки, операції, дані, як вони обробляються та змінюються. Дані, які використовуються для введення або виведення інформації, прийнято позначати паралелограмом, дані, які обробляються та змінюються, — прямокутником, графічний блок для прийняття рішення за заданою умовою — ромбом, а початок і кінець алгоритм з еліпсом. Усі ці графічні символи використовуються відповідно до стандарту ДСТУ ISO 5807:2016. За допомогою схеми зручно стежити за тим, як виконується програма, з якими даними їй потрібно працювати, як вона послідовно змінює свій стан. Це допомагає зрозуміти основне призначення програми та її функціонал.

На блок-схемі (рисунок 4.5) можна прослідкувати послідовність дій розробленого клієнтського додатку, як воно сукупно завантажується в браузер, як приймає дії користувача, обробляє їх і відправляє на сервер.

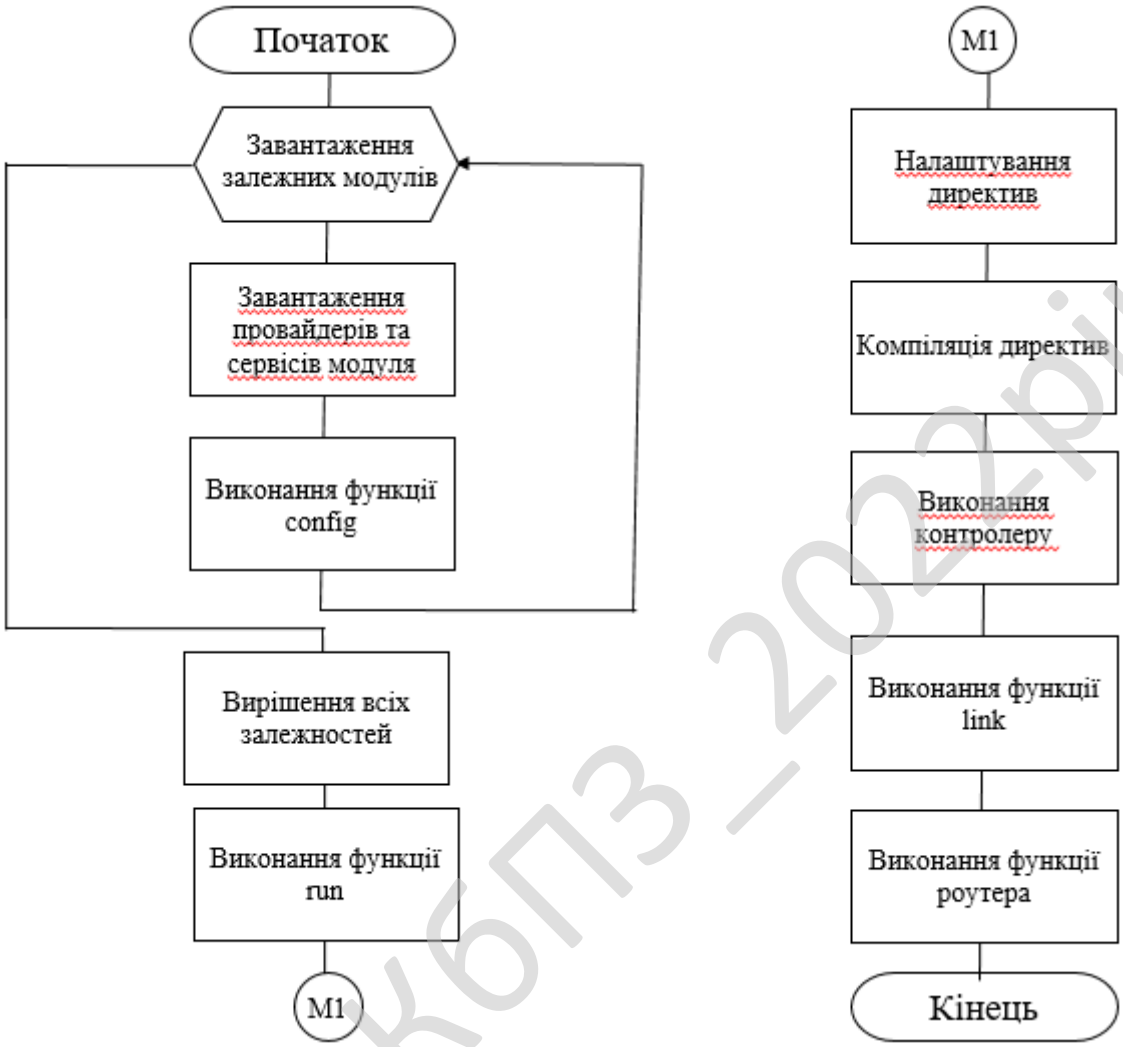

Рисунок 4.5 - Блок-схема головного модулю

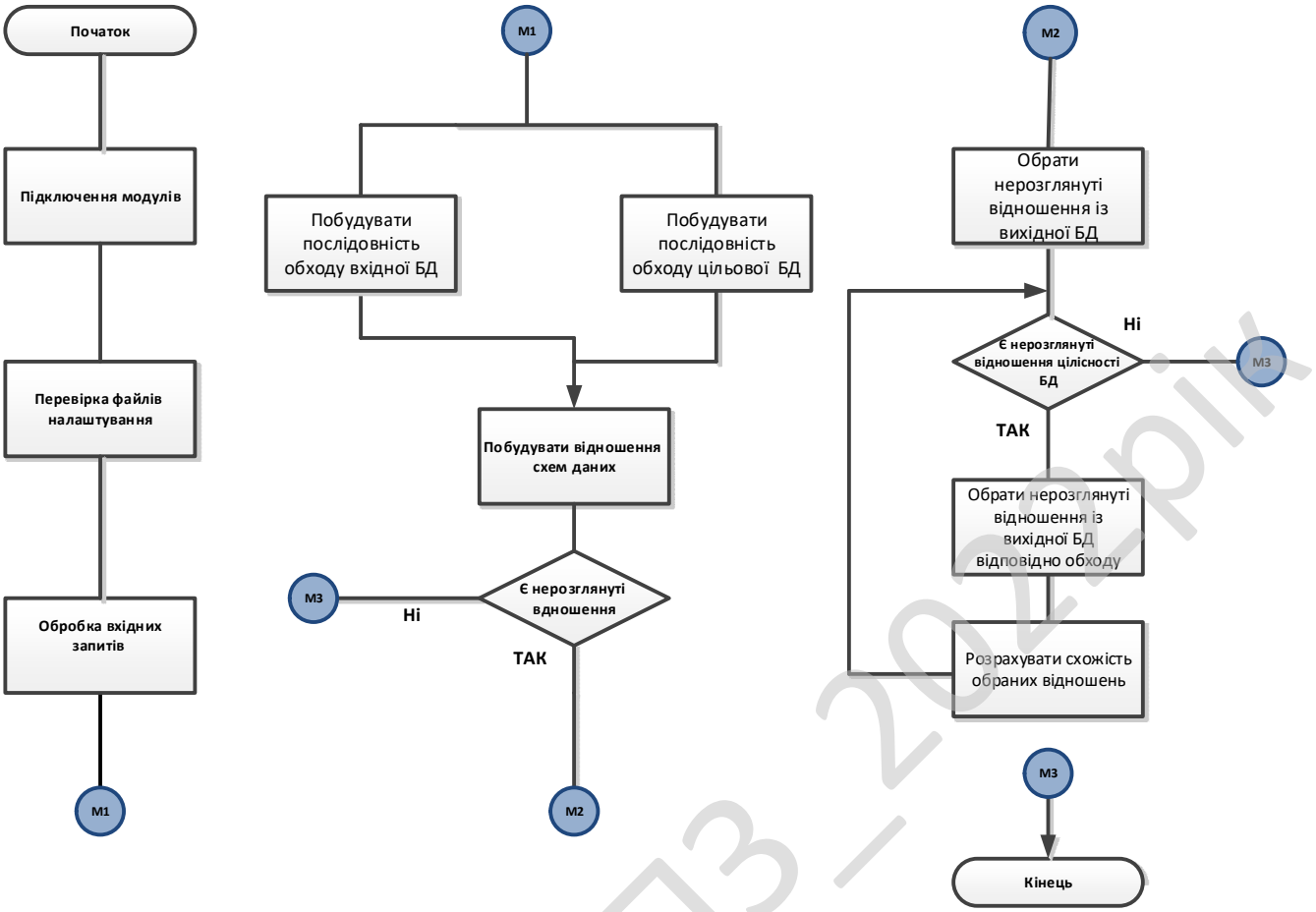

Рисунок 4.6 - Блок-схеми роботи системи

Опис інформаційної системи

Програму міграції даних можна використовувати без доступу до Інтернету, просто завантаживши JVM на комп’ютер і відкривши bd.file. Інтерфейс програми зроблено приємним і зрозумілим для користувача програми.

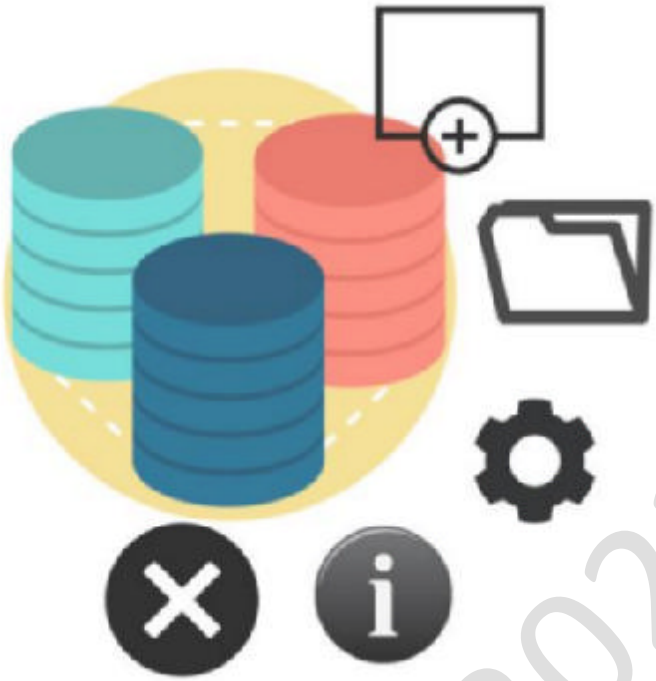

Рисунок 4.7 – Головна сторінка програми

Користувач може завантажити свій файл xml або відкрити існуючий проект для виконання міграції..

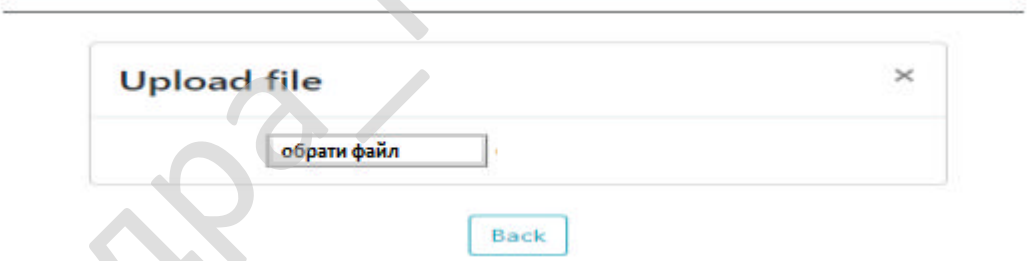

Рисунок 4.8 – Вкладка “Завантажити файл”

Рисунок 4.9 – Вкладка “Деталі БД”

Дерево бази даних будується в результаті завантаження схеми бази даних, і коли ви клацаєте на існуючому об’єкті, таке саме дерево будується у вихідній базі даних. Перетворений скрипт завантажується в центральне вікно. Бази даних представлені у вигляді деревоподібної структури даних, причому кожна база даних відображає схему даних. Коли ви клацаєте об’єкт бази даних, у центральному вікні відображаються сценарії бази даних PostgreSQL і MS Server.

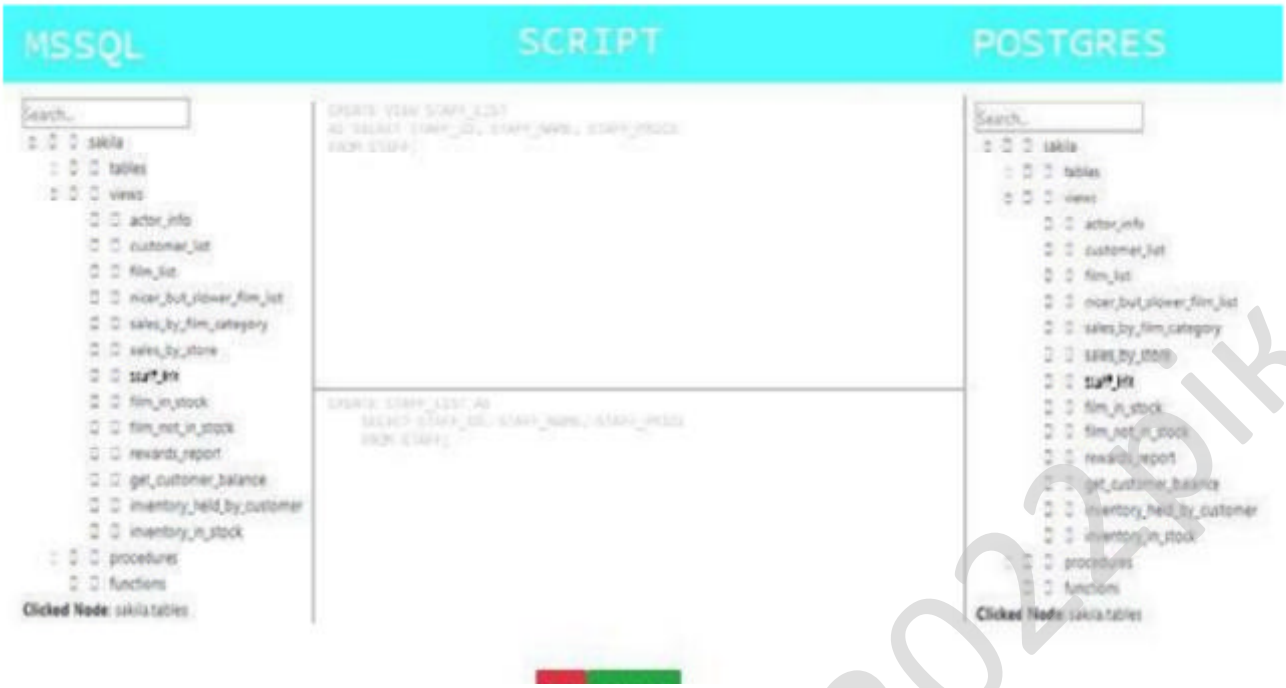

Рисунок 4.10 – Результат роботи програми

У результаті виконання програми дані з бази MS Server були повністю мігровані до бази PostgresSQL без втрати даних.

4.2 Захист розробленого програмного забезпечення

В першу чергу варто перевірити розроблений додаток автоматизованими засобами, вони зможуть перевірити додаток на найрозповсюдженіші помилки. Вони перевіряють не тільки код написаний розробником, а й бібліотеки та модулі які використовує розробник, а вони складають найбільшу частку зломів. Згідно з останніми перевірками, понад 50% npm модулів, не оновлюються більше декількох років.

Є декілька способів перевірити модулі перед їх використанням, в першу чергу варто переглянути код, та перевірити чи повністю модуль покриває потреби розробника. Він повинен перевірити які ще бібліотеки використовує цей модуль і коли вони останній раз оновлювалися, чим менше він має залежностей, тим краще.

Великі компанії мають спеціальних працівників, які проводять ревізії пакетів перед їх використанням, та складають спеціальні білі списки дозволених модулів.

Сама npm занепокоєна тим, як часто у бібліотеки почали впроваджувати майнерів та віруси для збору приватних даних. Тому вони випустили спеціальний модуль npm audit. Він сканує встановленні модулі в проект і порівнює їх з чорним списком модулів, які містять уразливості. Сьогодні навіть команда npm install підкаже, чи мають встановленні модулі вразливості. А нова команда npm audit fix, пропонує замінити вразливі пакети, або оновити до більш захищених версій, якщо такі існують. Але так можна знайти лише ті модулі, про які вже повідомили користувачі та розробники.

Також для перевірки безпеки додатку, важливим є написання тестів. Зазвичай розробники на AngularJS використовують концепцію Jasmine's Behavior-Driven Development (BDD), при написанні тестів. Jasmine − це вільний фреймворк для тестування коду написаного за допомогою JavaScript, його можна запустити на будь-якій платформі. Його перевагою є зручний інтерфейс, та зрозуміле налаштування. Для тестування одиничних тестів часто використовують бібліотеку Karma, головна мета якого наблизити тестування додатку до його налаштувань в продакшен.

Також важливо використовувати протокол HTTPS, він є набагато надійнішим HTTP. HyperText Transfer Protocol Secure (HTTPS) — забезпечує шифрування даних і дозволяє надійно захищати дані користувачів при передачі в Інтернеті. Сьогодні більшість сайтів використовують саме цей протокол, оскільки він є обов'язковим при передачі приватних даних на сервер, особливо коли передаються паролі або дані кредитних карт. Також варто бути обережним с cookie файлами, які передаються під час авторизації. Зловмисник може перехопити їх та підробити запит на сервер, щоб цього уникнути потрібно використовувати HTTPS протокол.

Також слід брати до уваги поширені CORS та CSRF атаки. CSRF- атака це коли на сторінки, робиться непомітна форма, щоб відправляє запит з приватними даним користувача. Якщо на сайті авторизація відбувається тільки за допомогою кукі, то така атака може бути успішна. Щоб цього уникнути потрібно використовувати спеціальні токени. Для підпису XMLHttpRequest токен також зберігається в кукі. Тоді JavaScript може прочитати його з домену і додати в заголовок, а сервер - перевірити, що в заголовку міститься коректний токен.

При крос-домених запитів, браузер додає заголовок Origin, який зберігає домен, з якого відбуваються запити. Сервер у цьому випадку повинен перевірити домен і відповісти спеціальним заголовком. Якщо сервер дозволяє доступ, він повинен відправити заголовок Access-Control-Allow-Origin, що зберігає домен запиту. Якщо Access-Control-Allow-Origin відсутній, то сервер завершує запит з помилкою. При таких запитах не передаються куки і заголовки HTTP-авторизації.

5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Нижче наведено приклад застосування користування розробленою системою в різних випадка міграцій.

Створення міграцій

Для створення нової міграції (наприклад, яка створює таблицю для новин), ми повинні ввести в консолі:

yiic migrate create <name>

Обов'язковий параметр name повинен містити дуже короткий опис міграції (таке, як, наприклад, create_news_table). Як буде показано далі, цей параметр використовується як частина імені класу міграції, тому використовувати можна тільки букви, цифри та знаки підкреслення.

yiic migrate create create_news_table

Наведена команда створить в директорії protected/migrations файл m101129_185401_create_news_table.php, який містить наступне:

```
class m101129_185401_create_news_table extends CDbMigration
{
    public function up()
    {
    }
    public function down()
    {
        echo "m101129_185401_create_news_table does not support migration down.\n";
        return false;
    }
```

```
        /*
        // якщо потрібно виконати зміни в транзакції, використовуйте safeUp/safeDown
        // замість up/down
        public function safeUp()
        {
        }
        public function safeDown()
        {
        }
        */
    }
```

Варто відзначити, що ім’я класу збігається з іменем файлу і будується як m<timestamp>_<name>, де <timestamp> — це час створення міграції в UTC (у форматі yymmdd_hhmmss), а <name> — те, що передано в параметрі name команди.

Метод up() повинен містити код, який виконує міграцію, а метод down() може містити код, який скасовує зроблене в up().

Іноді реалізувати down() не виходить. Приміром, якщо в up() із таблиці видаляються дані, в down() повернути їх не вийде. У цьому випадку міграція називається незворотною, що означає неможливість повернення до попереднього стану бази даних. У наведеному вище коді метод down() повертає false. Це означає неможливість відкату міграції.

Інформація: якщо метод up() або метод down() повертають false, всі наступні міграції не будуть застосовані, для цього було необхідно викинути виняток.

Розглянемо як приклад міграцію, яка створює таблицю з новинами.

```
class m101129_185401_create_news_table extends CDbMigration
{
```

```
    public function up()
    {
        $this->createTable('tbl_news', array(
            'id' => 'pk',
            'title' => 'string NOT NULL',
            'content' => 'text',
        ));
    }
    public function down()
    {
        $this->dropTable('tbl_news');
    }
}
```

Базовий клас CDbMigration надає набір методів для роботи з даними і структурою бази даних. Приміром, за допомогою CDbMigration::createTable можна створити нову таблицю, а CDbMigration::insert додасть рядок з даними. Всі ці методи використовують підключення до бази даних, що повертає CDbMigration::getDbConnection(), що за замовчуванням еквівалентно Yii::app()->db.

Інформація: Як ви могли помітити, методи CDbMigration дуже схожі на методи CDbCommand. І це насправді так. Єдина відмінність у тому, що методи CDbMigration підраховують витрачений час на їх виконання і виводять повідомлення про параметри методів.

Транзакційні міграції

При застосуванні складних міграцій для дотримання цілісності та зв'язності потрібно або виконати всі запити, або, якщо запит не виконався, скасувати попередні запити. Для досягнення цієї мети можна використовувати транзакції.

Можна почати транзакцію явно і заключити в неї весь код, який змінює базу даних:

```
class m101129_185401_create_news_table extends CDbMigration
{
   public function up()
   {
      $transaction=$this->getDbConnection()->beginTransaction();
      try
      {
         $this->createTable('tbl_news', array(
            'id' => 'pk',
            'title' => 'string NOT NULL',
            'content' => 'text',
         ));
         $transaction->commit();
      }
      catch(Exception $e)
      {
         echo "Exception: ".$e->getMessage()."\n";
         $transaction->rollback();
         return false;
      }
   }

   // …схожий код для down()
}
```

А можна зробити це простіше, реалізувавши метод safeUp() замість up() і safeDown() замість down():

```
class m101129_185401_create_news_table extends CDbMigration
{
    public function safeUp()
    {
        $this->createTable('tbl_news', array(
            'id' => 'pk',
            'title' => 'string NOT NULL',
            'content' => 'text',
        ));
    }
    public function safeDown()
    {
        $this->dropTable('tbl_news');
    }
}
```

При застосуванні міграції почне транзакцію і, потім, виконає код в safeUp() або safeDown(). Якщо при цьому виникне яка-небудь помилка, відбудеться відкат транзакції, тобто база повернеться у початковий стан.

Примітка: Не всі СУБД повністю підтримують транзакції і не для всіх виразів. У тому випадку, якщо підтримки транзакцій немає, реалізовувати треба up() і down(). У разі MySQL та MariaDB, деякі вирази SQL можуть викликати неявне застосування транзакції (детальніше про це у документації до MySQL та MariaDB).

Застосування міграцій

Для того, щоб застосувати всі нові міграції (тобто привести локальну БД в актуальний стан), слід запустити наступну команду:

```
yiic migrate
```

Команда відобразить перелік всіх нових міграцій і, у випадку позитивної відповіді, по черзі запустить метод up() у кожному класі міграції в порядку його створення.

Після застосування міграції у таблицю tbl_migration буде внесений відповідний запис. Це дозволяє дізнатися, які міграції вже застосовані, а які ні. Якщо таблиця tbl_migration не існує, вона буде створена автоматично в базі даних, зазначеної у компоненті db додатка.

Іноді потрібно застосувати лише одну або декілька нових міграцій. Для цього можна використовувати наступну команду:

yiic migrate up 3

При цьому застосовується три нові міграції. Замість трьох можна вказати будь-яку кількість застосовуваних міграцій.

Також можна привести стан бази даних до певної версії:

yiic migrate to 101129_185401

У якості параметра, що вказує версію, до якої потрібно привести базу даних, використовується частина імені файлу, що відповідає часу створення міграції. Якщо між останньою застосованою і вказаною міграціями кілька міграцій, то всі вони будуть застосовані. Якщо зазначена міграція вже застосовувалася, то буде проведений відкат всіх міграцій, застосованих після неї (описано в наступному розділі).

Відкат міграцій

Для відкату однієї або декількох останніх застосованих міграцій можна скористатися наступною командою:

yiic migrate down [step]

де необов'язковий параметр step задає кількість міграцій, які треба відкотити. За замовчуванням відкочується одна остання застосована міграція.

Як було описано раніше, не всі міграції можна відкотити. При спробі відкату таких міграцій буде викинуто виключення і процес відкату буде перерваний.

Повторне застосування міграцій

Повторне застосування міграції проводиться шляхом послідовного відкату і застосування. Здійснити це можна наступною командою:

yiic migrate redo [step]

де необов'язковий параметр step вказує кількість міграцій, які необхідно застосувати ще раз. За замовчуванням повторюється одна остання міграція.

Перегляд інформації про міграції

Крім застосування і відкату міграцій, інструмент міграцій може відображати історію міграцій і нові, ще не застосовані міграції.

yiic migrate history [limit]

yiic migrate new [limit]

Тут параметр limit вказує кількість міграцій, що відображаються. Якщо limit не вказаний, відображаються всі міграції.

Перша команда показує вже застосовані міграції, друга - міграції, які ще не були застосовані.

Зміна історії міграцій

Іноді потрібно змінити історію міграцій так, щоб поточна версія була замінена на вказану без застосування або відкату міграцій. Часто це потрібно при створенні нової міграції. Для цього можна використовувати наступну команду:

yiic migrate mark 101129_185401

Ця команда дуже схожа на yiic migrate to, але вона лише змінює таблицю історії міграцій до зазначеної версії без застосування або відкату самих міграцій.

Налаштування команди міграцій

Є кілька способів налаштувати команду міграцій.

Використовуючи параметри командного рядка

Команда міграцій може бути налаштована чотирма опціями:

interactive: чи використовувати інтерактивний режим. За замовчуванням true, тобто при застосуванні міграції буде виводитися підтвердження. Якщо параметр виставлений у false, то міграції можна застосувати у фоновому режимі;

migrationPath: вказує директорію, в якій зберігаються всі файли міграцій. Шлях повинен вказуватися у форматі псевдоніма і відповідна йому директорія повинна існувати. Якщо параметр не вказаний, буде використана піддиректорії migrations, що знаходиться всередині директорії з додатком;

migrationTable: вказує ім'я таблиці в базі даних, яка зберігає історію міграцій. За замовчуванням воно дорівнює tbl_migration. Структура таблиці наступна: version varchar(255) primary key, apply_time integer;

connectionID: вказує ідентифікатор компонента бази даних. За умовчанням це 'db';

templateFile: вказує шлях до файлу, який використовується як шаблон для генерації класів міграцій. Шлях повинен вказуватися як псевдонім (тобто як application.migrations.template). Якщо шлях не заданий, буде використовуватися внутрішній шаблон. У шаблоні токен {ClassName} буде замінений ім'ям класу міграції.

Для зазначення опцій використовується наступний формат:

yiic migrate up --option1=value1 --option2=value2 ...

Наприклад, якщо необхідно мігрувати модуль forum, файли міграцій якого розташовані у директорії модуля migrations, можна скористатися наступною командою:

yiic migrate up --migrationPath=ext.forum.migrations

Варто відзначити, що при передачі через командний рядок прапорів, таких як interactive, необхідно використовувати значення 1 або 0:

yiic migrate --interactive=0

Глобальна конфігурація команди

У той час, як опції командного рядка дозволяють нам на льоту конфігурувати команду міграцій, іноді потрібно застосувати налаштування раз і

назавжди. Приміром, нам може знадобитися використовувати іншу таблицю для зберігання історії міграцій або використовувати свій шаблон міграції. Це можна зробити, змінивши налаштування консольного додатку наступним чином:

```
return array(
    ......
    'commandMap'=>array(
        'migrate'=>array(
            'class'=>'system.cli.commands.MigrateCommand',
            'migrationPath'=>'application.migrations',
            'migrationTable'=>'tbl_migration',
            'connectionID'=>'db',
            'templateFile'=>'application.migrations.template',
        ),
        ......
    ),
    ......
);
```

Тепер, при запуску команди migrate, зазначені вище налаштування будуть застосовані без введення яких-небудь додаткових параметрів

6 НАУКОВА НОВИЗНА

У магістерській роботі розроблено програмне забезпечення для системи міграції в базах даних.

Метою розробки є програмна реалізація та питання, пов'язані з міграцією даних, що зберігаються в реляційних базах даних, і процес її виконання та дослідження міграції баз даних і створення методу міграції даних, який має бути більш ефективним, ані ж методи, що існують та виконувати роботу оптимально.

Об'єктом дослідження є дослідження міграції між СУБД MS Server та PostgreSQL.

Предметом дослідження є методи та алгоритми міграції, а саме між СУБД MS Server та PostgreSQL.

Методи дослідження базуються на методах теорії кодування, паттернах проектування, методологіях розробки програмного забезпечення та технологіях взаємодії клієнт-серверних додатках.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

- розширений механізм перенесення даних для виконання паралельної міграції даних;
- розроблений і реалізований алгоритм побудови послідовності обходу схеми бази даних;
- розроблений і реалізований алгоритм порівняння схем баз даних;
- створений виконуваний еволюційний прототип додатка для міграції даних.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми магістерської роботи

Після ознайомлення з засобами розробки та необхідними технологіями які використовувались для впровадження та тестування системи, був розроблений план розробки програмного забезпечення. Був визначений необхідний час для розробки та впровадження програми, який склав 22 дні.

В магістерській роботі було проведене дослідження та розроблено два клієнтських додатка та один серверний з використанням бази даних, для виявлення переваг та недоліків кожної з платформ. Розроблені додатки мають функціонал, які використовують більшість веб-сайтів в Інтернеті, що дає можливість чітко порівняти їх можливості. Серед основного функціоналу наявні:

- Можливість спілкуватися за допомогою чата;
- Редагувати профіль користувача, змінювати медіа файли;
- Надійна реєстрація та авторизація;
- Можливість надсилати e-mail сповіщення;
- Динамічне оновлення інформації на стороні клієнта.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	3
2. Кількість екземплярів програм, шт.	Ne	240
3. Запланований термін розробки, днів	Fpq	22
4. Група задачі підсистеми управління (1-6)	–	3
5. Ступінь новизни задачі (А, Б, В, Г)	–	Г
6. Складність алгоритму (1, 2, 3)	–	3
7. Кількість макетів вхідної інформації	–	8
8. Кількість форм вихідної інформації.	–	6
9. Мова програмування (1-6)	–	6
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	4
12. Детальність проекту ПП (1-6)	–	4
13. Рівень спрацьованості колективу (1-6)	–	3
14. Ступінь вимірності процесів (1-6)	–	4
15. Необхідна надійність програмного забезпечення (1-6)	–	4
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	2
17. Складність кінцевого програмного продукту (1-6)	–	3
18. Необхідний рівень забезпечення повторного використання (1-6)	–	3
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	3

Продовження таблиці 7.1

20. Вимоги до швидкодії ПП (1-6)	–	4
21. Обмеження на розміри основного сховища даних (1-6)	–	
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	4
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	2
25. Постійність складу команди розробників (1-6)	–	3
26. Досвід розробки додатків (1-6)	–	3
27. Досвід роботи з обчислювальною платформою (1-6)	–	4
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	3
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт	–	3
32. Вартість ПЗ у розробника (НМА), грн.	–	36000
33. Норматив додаткової зарплати, % :	$Н_д$	10%
34. Норматив відрахувань у соціальні фонди, %	Нс	22%
35. Норматив загальногосподарських витрат, %	Нг	15%
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15%
37. Рівень рентабельності програмної продукції, %	Ре	30%
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Трудомісткість розробки програмного забезпечення визначається в людино-днях. Такі норми часу охоплюють всі стадії розробки документації до технічного завдання, технічного проекту та впровадження у виробництво. Стадія технічного завдання (ТЗ) включає в себе збирання необхідної інформаії, розрахування можливих методів вирішення задачи, вибір мови програмування, визначення часу на розробку документації, та затвердження технічгог завдання. Стадія технічного проекту (ТП) включає розробку блок-схеми, визначення технологій, розробку документації, визначення вхідних та вихідних даних а також структуру програми. Стадія робочого проекту (РП) включає етап розробки програми та її налаштування. Стадія впровадження (ВП) включає перевірку і вихід програмного забезпечення в промислову експлуатацію. Проце створення програмного забезпечення, в першу чергу починається з планування. Правильна оцінка трудовитрат є основою для планування всього подальшого проекту.

Існує багато методик для визначення трудовитрат, але найбільшу отримала методика COCOMO (Constructive Cost Model) в основі якої лежить визначення об'єму робіт по кількості строчок коду. Використаємо її для визнаення трудоміскості розробки ПЗ для тадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A\ \text{Size}^{B}, \tag{7.1}$$

де: A – коефіцієнт Боема, $A = 2{,}45$;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001 \sum Wi\ , \tag{7.2}$$

де: *Wi* – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

Таблиця 7.2 – Масштабуючі показники:

Попередній досвід	3	2,43
Гнучкість проекту ПП	4	2,43
Детальність проекту ПП	4	1,69
Рівень спрацьованості колективу	3	2,97
Ступінь вимірності процесів	4	2,97

$$\sum Wi = 2,43 + 2,43 + 1,69 + 2,97 + 2,97 = 12,49,$$

$$B = 1,01+0,001*12,49= 1,02249,$$

$$T_{ном} = 2,45 \cdot 2,5^{1,02249} = 6,2525 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПV_j, \qquad (7.3)$$

де: $ПV_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в додатку 3.

Таблиця 7.3 – Коефіцієнти трудовитрат:

Необхідна надійність програмного забезпечення	4	1,15
Розмір бази даних (порівняно з розміром програми)	2	0,93
Складність кінцевого програмного продукту	3	1
Необхідний рівень забезпечення повторного використання	3	1
Документованість відповідно до планового життєвого циклу	3	1

Продовження таблиці 7.3

Вимоги до швидкодії ПП	4	1,11
Обмеження на розміри основного сховища даних	1	1
Різноманітність використовуваних обчислювальних платформ	4	1,15
Професійний рівень аналітиків	2	1,22
Професійний рівень програмістів	2	1,16
Постійний склад команди розробників	3	1
Досвід розробки додатків	3	1
Досвід роботи з обчислювальною платформою	4	0,88
Досвід роботи з мовою і інструментами середовища розробки	3	1
Досвід роботи з програмними інструментами розробки	3	1
Розробка ПО для декількох серверів одночасно	2	1,10
Вимоги до дотримання встановленого графіка робіт	3	1

$$\Pi v_j = 1{,}15*0{,}93*1*1*1*1{,}11*1*1{,}15*1{,}22*1{,}16*1*1*0{,}88*1*1*1{,}10*1 = 1{,}8702$$

$$T_{уточн} = 6{,}2525*1{,}8702 = 11{,}6934 \text{ люд-міс.}$$

Визначаємо підсумкові трудовитрати, люд-дні:

$$T_{рп} = 0{,}3CT^{0{,}33+0{,}2(B-1{,}01)}S \quad (7.4)$$

де: C – визначений емпірично коефіцієнт;

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0,3 \cdot 2,66 \cdot 11,6934^{0,33+0,2(1,02249-1,01)} \cdot 100 = 180,75 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.4.

Таблиця 7.4 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Додаток №5
Ескізний проект	10	Додаток №6
Технічний проект	6	Додаток №7
Робочий проект	180,75	Ф 7.1-7.4
Впровадження	18	Додаток №11
Всього	223,75	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Кількість інженерів-програмістів розраховується за формолою:

$$Ч = (T_{пз} * N) / (F_{рq} - H_{ев})$$

(7.5)

$T_{пз}$ - трудомісткість розробки програми

N - кількість розроблених програм за рік

$F_{рq}$ - запланований термін розробки

$H_{ев}$ - невиходи за поважних причин, днів

Ч = (223,75 * 1) / (22- 3) = 11,7763

Кількість працівників для проведення технічного обслуговування визначається від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Таблиця 7.5 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	385	3	1155	19
Монітор	160	3	480	8
Клавіатура	140	3	420	7
Маніпулятор «мишка»	30	4	120	2
Принтер матричний	185	1	185	3
Принтер лазерний	355	1	355	6
Принтер струминний	300	1	300	5
Сканер	155	1	155	2.5
Концентратор-маршрутизатор	155	2	310	5
Кабельні господарства ЛВС на 1 м п.	2,5	5	12,5	0.2
Копіювальний апарат	285	1	285	5
Усього за рік:			$З_ч$	62,7

Норми часу на профілактику ЕОМ дорівнюють 10%.

$$Ф^{c}_{dp} = (З_{ч} * n_{mic})/1,2 \quad (7.6)$$

$$Ф^{c}_{dp} = (62,7 * 1)/1,2 = 52,25 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = Ф^{c}_{dp} / (F_{pq} * T_{зм}) \quad (7.7)$$

$$Ч_{ел} = 52,25 /(22 \cdot 8) = 0,296875 \text{ ставки.}$$

Чисельність персоналу інженерів-системотехніків, адміністраторів мереж, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.6 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 20012 R2, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN, PPPoE, Frame Relay, Wi-Fi	1,5	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1,5	
Всього		4	
Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	2	0,5
	Підтримка постійних клієнтів	1	
	Оформлення договорів, ведення тендерів	0,5	
	Контроль взаєморозрахунків з постачальниками	0,5	
Всього		4	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	2	0,5
	Створення графічних і стилістичних елементів сайту	1	

Продовження таблиці 7.6

	Оформлення банерів і промо-сторінок	0,5	
	Розміщення графіки і контенту на Інтернет сторінках	0,5	
Всього		4	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців.

Таблиця 7.7 - Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (IT-менеджер)	1	10000	10000
Продакт-менеджер	1	8000	8000
Інженер-програміст	3	8000	24000
Інженер-електронщик	1	8000	8000
Інженер-системотехнік	0,5	8000	4000
Адміністратор мережі	0,5	8000	4000
Системний програміст	1	8000	8000
Дизайнер WEB	0,5	8000	4000
Інженер-верстальник	0,5	8000	4000
Бухгалтер-економіст	1	8000	8000
Всього за період розробки	$R_{сп} = 10$	-	$Ф_{роб} = 82000$

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{cd} = Ф_{роб} / (R_{cn} * F_{pq}) \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{cd} = 82000/ (10*22) = 372 \text{ грн}$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість складається з ціни, транспортно-заготівельних витрат, вартості будівельно-монтажних та пуско-налагоджувальних робіт, а також витрат на випробовування у виробничих умовах. Норма амортизації визначається в залежності від нормативного строку служби.

Балансова вартість будівель (В буд) визначається з урахуванням кількості робочих місць виконавців (R ñї 1), питомої площі на одне робоче місце (S у = 8...12 м 2) та вартості одного квадратного метра виробничої площі (Ц пл = 1500 грн. і вище).

$$В_{буд} = R_{cn} * S_у * Ц_{пл} \quad (7.9)$$

де: R_{cn} – кількість робочих місць виконавців, шт. Приймаємо 13 робочих місць;

$S_у$ – питома площа на одне робоче місце, м^2;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

$$В_{буд} = 10 * 8 * 1500 = 75000$$

Вартість передавальних пристроїв (електромережі, водопровід, тепломережі тощо) можна орієнтовно прийняти до 12% вартості будівель і у даному випадку вона складе: 9000 грн.

$$В_{перед} = 0,01В_{буд}П_{перед} \quad (7.10)$$

$$В_{перед} = 0,01*75000*5 =3750$$

Балансову вартість господарського інвентарю (меблі, стелажі, крісла тощо) доцільно розрахувати за орієнтовною нормою від 800 грн. до 6000 грн. на одне робоче місце:

$$І_{нв} = R_{сп} * Ц_{м} \quad (7.11)$$

де: $Ц_{м}$ – ціна меблів для одного робочого місця, грн.

$$І_{нв} =10·4000 =40000 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.8.

Дані по оптовій ціні на обладнання та комплектуючі вибирались фірми Rozetka за 11.11.22– джерело http://rozetka.com.ua/

Вартість вимірювальних пристроїв можна прийняти до 5% від вартості персональних комп'ютерів:

$$В_{вим.пристр} = 0,01В_{пк}П_{вим.пристр}, \text{ грн} \quad (7.12)$$

$$В_{вим.пристр} = 0,01 * 51150 * 5 = 2557,5$$

де $В_{ПК}$ – вартість персонального комп’ютера;

$П_{вим.пристр}$ – відсоток, який встановлює залежність між вартістю вимірювальних пристроїв та вартістю персонального комп'ютера.

Таблиця 7.8 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп'ютер		15500
Системний блок		
Процесор	Intel Core i3-10105F 3.7GHz/6MB (BX8070110105F) s1200 BOX	2 599
Системна плата	Asus Prime H410M-K (s1200, Intel H410, PCI-Ex16)	2 118
Відеокарта	Gigabyte PCI-Ex GeForce GT 710 2048MB DDR3 (64bit) (954/1800) (HDMI, DVI, VGA) (GV-N710D3-2GL)	1 949
Жорсткий диск	Toshiba P300 1TB 7200rpm 64MB HDWD110UZSVA 3.5 SATA III	1 049
Оперативна пам'ять	AMD 8 GB DDR4 2400 MHz (R748G2400U2S-U)	792

Продовження таблиці 7.8

DVD-привод	Asus DVD±R/RW USB 2.0 SDRW-08D2S-U LITE Black External	861
Корпус	GameMax MT508-NP-2U3	751
Кулер	–	–
Кардрідер внутрішній	Gembird SD/MMC/RS-MMC/MicroSD картами памяти и 2.5" HDD/SSD (FDI2-ALLIN1-03)	468
Інше (Клавіатура, мишка)	A4Tech KM-72620D USB	399
Монітор	23.8" Acer SA240YBbmipux (UM.QS0EE.B01) USB Type-C 15W	4 499
Принтер лазерний	HP LaserJet 107a (4ZB77A)	4 421
Найменування комплектуючої або обладнання	Тип	Оптова ціна
Принтер струминний	Canon PIXMA Ink Efficiency E414 (1366C009)	1 709
Сканер	IRISCan Express 4 (458510)	4 019
Копіювальний апарат	RICOH MP1600	8 344

Продовження таблиці 7.8

Пристрій безперебійного живлення	LogicPower LPM U1400VA-P (LP10394)	3 014

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.9.

Таблиця 7.9 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Середня ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп'ютери	3	15500	4650	51150
Принтер лаз.	1	4 421	442,1	4863,1
Принтер струм.	1	1 709	170,9	1879,9
Сканери	1	4 019	401,9	4420,9
Копіюв. апарат	1	8 344	834,4	9178,4
Всього	–	–	–	71492,3

Таблиця 7.10 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	*75000*	-	-
2. Передавальні пристрої	*9000*	-	-
Всього по групі	84000	5	4200
Група 4			
3. Обчислювальна техніка	71493	50	35746,5
4. Інше обладненння	2557,5	50	1278.75
Всього по групі	74050,5	50	37025,25
Група 5			
5. Транспортні засоби	714,92	20	142,944
Група 6			
6. Вимірювальні пристрої	2557,5	-	-
7. Господарський інвентар	40000	-	-
Всього по групі	42557,5	25	10639,375
8. Нематеріальні активи	36000	50	18000
Разом	$К_р$ = 237322,92		$А_р$ = 70007,569

Примітка: вартість не враховуємо. Тому вартість транспортних засобів приймаємо рівним нулю.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = (З_{сд} * Т_{пз})/N_е \quad (7.13)$$

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 372 * 223,75/ 240 = 347$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о * Н_д * 0,01, \quad (7.14)$$

де: H_q – норматив додаткової зарплати, %.

$$З_д = 347 * 10 * 0,01 = 34,7$$

Відрахування на соціальні потреби за нормативом $H_с$ = 22% від суми основної та додаткової зарплати:

$$С_{оц} = 0,01 * Н_с * (З_о + З_д), \quad (7.15)$$

де: $H_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0,01 \cdot 22(346 + 34,7) = 83,754$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $H_г$ = 15% від основної зарплати:

$$Г_{осп} = З_о * Н_с * 0,01, \quad (7.16)$$

де: $Н_г$– загальногосподарські витрати, %.

$$Г_{осп} = 347*15*0,01 = 52,05$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.17)$$

де: $З_{М1}$– вартість паперу, грн.;

$З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.;

$З_{М3}$ – вартість фарби, картріджей, тонеру, грн.;

$N_е$ – кількість екземплярів програм, шт.

Згідно виданих викладачем норм приймаємо одну пачку паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$ = 100 грн., визначаємо вартість паперу за період розробки $N_м$ = 1 міс:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.18)$$

$$З_{М1} = 100 \cdot 1 = 100 \text{ грн.}$$

Згідно виданих викладачем норм до вартості запам'ятовуючих пристроїв входить вартість CD дисків в кількості, що дорівнює кількості екземплярів програм та одного DVD диска для збереження резервної копії програми:

$$З_{М2} = \sum Ц_д , \quad (7.19)$$

де: $Ц_д$ – вартість дисків

Наприклад: CDR TDK 700Mb, 80Min, 52x Cake box – 5 грн/шт., DVD-R LG 4,7Gb, 16x speed Cake box - 7 грн/шт.

$$З_{М2} = 240 \cdot 12 = 2880 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_{з}., \quad (7.20)$$

де: $Ц_{з}$ – вартість розхідних матеріалів друкуючих пристроїв: картридж для CANON LBP-3010 Black Canon 712 – 574 грн.; картридж для EPSON STYLUS PHOTO R390 – 558 грн.; картридж для CANON IR-1022A – LJ Q2612A Cart. HP LJ 1010/1012/1015/3015/3020/3030 (2500 стр.) – 570 грн.

$$З_{М3} = 574+558+570 = 1702 \text{ грн},$$

$$З_{М} = (100 + 2880 + 1702)/240 = 19,5 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_{п}$=15%) від основної зарплати виконавців:

$$О_{п} = З_{о} * Н_{п} * 0,01 \quad (7.21)$$

де: $Н_{п}$ – норматив витрат на освоєння нових мов програмування, %.

$$О_{п} = 347 * 15 * 0,01 = 52,05 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм (N_e = 240 прим.):

$$А_м = (А_р * N_{міс})/(N_е * 12) \quad (7.22)$$

де: A_p – загальна річна сума амортизаційних відрахувань, грн.

$$А_м = 70007,569 * 1/(240*12) = 24,30 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$С_п = З_о + З_д + С_{оц} + Г_{осп} + З_{М +} О_{п +} А_м \quad (7.23)$$

$$С_п = 347+34,7 + 83,754 +52,05 + 19,5 +52,05 + 24,30 =613,354 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності ($P_п$) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 30%.

$$П_р = 0,01 * Р_п * С_п \quad (7.24)$$

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0,01 * 30 * 613,354 = 184,0062 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Оптова ціна підприємства складається з повної собівартості і планового прибутку. Визначаємо оптову ціну підприємства:

$$Ц_п = С_п + П_р \quad (7.25)$$

$$Ц_п = 613,354 + 184,0062 = 797,3602$$

Величина податку на додану вартість:

$$ПДВ = 0,01 \square 20 * 797,3602 = 159,47204$$

де: Нпдв – ставка податку на додану вартість.

Відпускна ціна програмної продукції:

$$Ц = Цп + ПДВ \quad (7.26)$$

$$Ц = 797,3602 + 159,47204 = 956,83224$$

Таблиця 7.11 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1	2	3
Основна зарплата виконавців	$З_о$	347
Додаткова зарплата виконавців	$З_д$	34,7
Відрахування на соціальні потреби	$С_{оц}$	83,754
Загальногосподарські витрати	$Г_{осп}$	52,05
Витрати на матеріали	$З_М$	19,5

Продовження таблиці 7.11

Освоєння нових операційних систем, мов програмування	$О_п$	52,05
Амортизація основних фондів	$А_м$	24,30
Повна собівартість програмного забезпечення	$С_п$	613,354
Плановий прибуток	$П_р$	184,0062
Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	797,3602
Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	159,47204
Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	956,83224

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.12 – Розрахунок об’єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	956,83224
Всього капітальних витрат	–	956,83224

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року.

Таблиця 7.13 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	15030	9662
2. Витрати на електроенергію	$З_{ел}$	1188	743
3. Витрати на амортизацію	$З_{ам}$	0	1580
Всього витрат за рік	$І$	16218	11985

Витрати на профілактичні роботи:

$$З_р = Т_р * З_г * (1 + 0,01 * Н_д) * (1 + 0,01 * Н_е) \quad (7.27)$$

де: $Т_р$ – кількість годин обслуговування кожного комп'ютера за рік, год;

$З_г$ – заробітна плата обслуговуючого персоналу, грн/год.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 280 годин на рік до 180 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{рбаз} = 280 * 40 * 1,1 * 1,22 = 15030,$$

$$З_{рнов} = 180 * 40 * 1,1 * 1,2 - 9662.$$

Витрати на електроенергію визначаються з урахуванням споживаємої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($T_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot T_р \cdot Ц_{ел,} \quad (7.28)$$

$$З_{ел\ баз} = 0{,}45 \cdot 1200 \cdot 2{,}2 = 1188 \text{ грн.},$$

$$З_{ел\ нов} = 0{,}45 \cdot 750 \cdot 2{,}2 = 743 \text{ грн.}$$

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.14.

Таблиця 7.14 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн. за варіантами		Сума відрахувань, грн., за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	956,83224	–	239.20806
Всього відрахувань	-	–	956,83224	–	239.20806

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$Е_в = (Ц_в — С_п) * N_е - Е_р * К_р \quad (7.29)$$

де: $К_р$ – балансова вартість основних фондів розробника, грн.

$$Е_в = (797,3602 — 613,354) * 240 — 0,15 * 237322,92 * 1/12 = 41194,9515$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$Т_в = К_р / ((Ц_в — С_п) * N_е), \quad (7.30)$$

$$Т_в = 237322,92/((797,3602 — 613,354) * 240 * 12/1) = 0,447831604$$

Таблиця 7.15 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	240
2. Повна собівартість розробленої програми	Грн.	613,354
3. Ціна розробленої програми	Грн.	797,3602
4. Плановий прибуток від реалізації розробленої програми	Грн.	184,0062
5. Рентабельність програмної продукції	%	30
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	237322,92
7. Загальний прибуток від реалізації програмної продукції	Грн.	44161
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	41194
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Рік	0,4

Продовження таблиці 7.15

10. Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	956,83224
11. Величина економічного ефекту у користувача програмної продукції	Грн.	3993,79
12. Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,22

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$Е_{сп} = (I_б - I_н) — Е_н (К_н - К_б), \quad (7.31)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$К_б$, $К_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$Е_{сп} = (16218\text{-}11985) — 0{,}25 * 956{,}83224 = 3993{,}79194 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$Т_{сп} = (К_н - К_б,)/(I_б - К_н), \quad (7.32)$$

$$Т_{сп} = 956{,}83224 /(16218\text{-}11985) = 0{,}226041162 \text{ роки}$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.15.

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Аналіз умов праці програміста

Інтернет відіграє важливу роль у житті сучасної людини. Кожного дня мільйони людей використовують Інтернет для пошуку необхідної інформації, спілкуванні у соціальних мережах, перегляду новин. Багато людей користуються Інтернетом у професійних цілях, оскільки завдяки Інтернету з'явилося багато нових професій. Тому для веб-розробника так важливо розробити зручний інтерфейс для зручного сприйняття інформації, та необхідний функціонал, який буде відповідати необхідним вимогам та навантаженням. Все це вимагає багато часу та великого навантаження з боку розробників.

Тому так важливо слідкувати за умовами праці, в яких відбувається робочий процес. Оскільки захворювання можуть бути спричинені надмірним фізичним або розумовим навантаженням, через велику нервово-емоційну напругу, або через виробниче середовище. В даному розділі магістерської роботи проведемо аналіз основних чинників при роботі програміста.

При роботі за комп'ютером, розробник має велике зорове навантаження, тому йому необхідне належне освітлення приміщення. Якщо в приміщенні недостатньо природного освітлення, потрібно використовувати спеціальні світильники. Також оскільки розробник значний час працює з електричними приборами є можливість, бути ураженим електричним струмом, тому потрібно дотримуватись всіх необхідних норм. Серед основних чиників, які впливають на розробників під час трудової діяльності можна виділити[46]:

1. Рівень освітлення в приміщенні.
2. Температура, вологість в приміщенні.
3. Рівень шуму на робочому місці.

4 .Напруга в електричному ланцюзі, електричні показники.

Розберемо кожний з чиників окремо, проаналізуємо які повинні бути стандарти кожного з чиників, відповідно до правил з охорони праці.

Рівень штучного освітлення

Головним документом для встановлення норм необхідних показників освітлення є ДБН В.2.5-28:2018 «Природне та штучне освітлення» [47].

Сьогодні найбільш розповсюдженішими є світлодіодні ламп. В середньому світловіддача від таких ламп знаходиться на рівні 80-120 Лм/Вт [48]. Джерелом живлення прийнято вважати електричну мережу у 220В. А освітленіть робочого приміщення повина бути Е =300-500 Лк, оскільки робота програміста відноситься до робіт середньої точності з присвоєнням розряду зорових робіт IV[49].

Рівень сітла повинен бути достатнім, щоб працівник міг працювати без навантаження на зір. Це залежить від системи освітлення, кількості світильниув, їх типу та розміщення у приміщенні. Допустиме значення освітленості робочої поверхні приймається Е = 400 лк [50].

Для покращення освітлення комп'ютерній лабораторії будуть використовуватися світлодіодні лампи, а саме FL-LED T8-900 світловий потік яких F=1500лм.

Мікроклімат робочої зони: температура, відносна вологості, швидкість руху повітря

Головним документом для встановлення норм мікроклімату робочої зони є ДСН 3.3. 6.042 -99 «Державні санітарні норми мікроклімату виробничих приміщень» [51].

Праця програміста за важкістю відноситься до легкої фізичної роботи категорії Іа [49]. Де вказано, що в приміщенні, я кому знаходиться компютерне обладнення, повнині бути встановлені певні норми, оскільки офісна техніка є джерелом тепловиділень, що може спричинити підвищення температури. Серед

потимальних параметрів для роботи встановлена температура 23 – 25^{0}С і вологість на рівні 40 – 60% в залежності ві періоду року.

Для підтримки комфортної температури можна використовувати як організаційні методи, наприклад розпорядок дня, так і технічне обладнення, наприклад кондиціонери, вентіляцію. Як правило в холодний період часу використовуються додаткове опалення для підтримання комфортної температури, а в літку встановлюються кондеціонери.

Рівень шуму на робочому місці

Як правило при використанні великої кількості компютерів в одному приміщенні, через гудіння, рівень шуму має значення більше норми. Допустима норма становить менше 50 дБ [52].

Гучний шум негативно впливає на умови праці та організм людини. Якщо шум триває тривалий час цу може спричинии головні болі, біль у вухах, підвищення стомлюваності, зниження концентрації та уваги. Такі симптоми можуть викликати стресові ситуації у людини. Все це шкодить продуктивності працініка та його стану здоровья.

Щоб встановити необхідний рівень шуму, використовують додаткову звукоізоляцію. Для цього найчатіше використовуються мати та плити із скляного та мінерального волокна, м'які плити з деревних стружок, картон, гуму, утеплений лінолеум, а також заміна вікон на звукоізолюючі.

8.2 Заходи профілактики при роботі з комп’ютерною технікою

Санітарно-гігієнічні норми є важливим критерієм при роботі в приміщенні. Від них залежить здоровє працівників, їх рівнень працездатності, втомлюваність. Щоб всього цього уникнути потрібно стежити за нормами на робочому місці.

Якщо говорити про електробезпеку в приміщенні, то в приміщенні необхідно устаткування розподільних щитів спеціальними розетками з

заземлюючими контактами, повині бути заземлені всі прилади і пристрої, час від часу повина проводитися перевірка всіх приладів, щорічна здача іспитів з охорони праці.

Для оптимальних показників мікроклімату та освітленості потрібні використовувати дефлектор, для організації вентиляції, та повітрообміну. Та перевірку освітленості в приміщенні згідно відділом охорони праці, щоб відповідати номам для зорової роботи .

Ще однією проблемою з якою часто зустрічаються програмісти є мала рухливість та повний сидячий робочий день. Тому рекомендується час від часу робити невеликі перерви, під час обіднього перериву вживати їжу не на робочому місці. У окремих випадках, коли при дотриманні всіх санітарних норм, працівник все одно себе погано почуває, дозволяється індивідуальний підхід для обмеження роботи з обчислювальними пристроями. Тривалість роботи за компютером не повина безперервно триввати більше 4 годин.

Для зменшення зорового та нервово-емоційного навантаженняє, та поліпшення мозкової діяльності рекомендується робити перерви для психологічного та фізичного розвантаження.

В приміщеннях також поині бути протипожежне обладнення, та інструкція у разі надзвичайних ситуаціх. Повина бути особа, яка відповідає за пожежну безпеку, перевіряє обладнення, та системи протипожежного захисту а також щорічне проведення інструктажів серед працивників.

Автоматична пожежна сигналізація повина відповідати вимогам ДБН ДБН В.2.5-56:2014, яке вимагає використання вогнестійких кабелів та автоматичну роботу системи оповіщення та евакуації людей у випадку надзвичайної ситуації [53].

При перевірці, приміщення повино відповідати всім нормам пожежної безпеки. Це виконується за допомогою перевірки пожежної охорони та техніки, проведенню інструктажу і своєчасне інформування пожежної охорони про несправність пожежної техніки, впровадження систем протипожежного

захисту. Організаційні та технічні заходи, спрямовані на попередження виникнення пожежі, обмеження поширення вогню та успішної евакуації людей.

8.3 Розрахнок занулення глухозаземленої нейтралі

Занулення, як основний засіб захисту, застосовується в електроустановках до 1 кВ з глухозаземленою нейтраллю трансформатора або генератора [54]. Початкові дані для розрахунку занулення глухозаземленої нейтралі трансформатора виробничого приміщення:

Загальна потужність: P = 5 кВт.

Кількість електродвигунів: m = 10

Потужність освітлювальних приладів: Po = 3 кВт.

Довжина магістрального кабеля: LM =70 м.

Довжина розгалудження: l=22м.

Лінійна напруга U=380 В.

Фазна напруга Uф=220 В.

Визначаємо силу номінального струму електроустановки:

$$Iном = Imax = (P * 1000) / (\sqrt{3} * Uл * cosф) \quad (8.1)$$

$$Iном = Imax = (5 * 1000)/(\sqrt{3} * 380 * 0,85) = 8,9 \text{ А}$$

де: P − номінальна сумарна потужність електроприладів, кВт;

Uл − лінійна напруга, В;

cosф − коефіцієнт потужності, приймається в залежності від типу електрообладнання в межах 0,8..0,87.

Визначаємо силу пускового струму електродвигуна:

$$I_{пус} = 5 * I_{ном}$$

(8.2)

$$I_{пус} = 5 * 8,9 = 44,5 \text{ А}$$

Визначаємо номінальну силу струму апарата захисту:

$$I_н = I_{пус}/b$$

(8.3)

$$I_н = 44,5 /2,5 = 17,8 \text{ А}$$

b − коефіцієнт пуску електродвигуна − для легких умов пуску − 2,5..3.

Вибираємо запобіжник ПН 2-100 з плавкою вставкою Іном = 50 А. Визначаємо найменше допустиме по умовам спрацьовування захисту значення сили струмукороткого замикання:

$$I_{ктіп} = I_н * K$$

(8.4)

$$I_{ктіп} = 50 * 3 = 150 \text{ А}$$

Ін−номінальний струм апарата захисту;

К − коефіцієнт надійності;

Знаходимо переріз провода або кабеля розгалуження з умови допустимого нагрівання:

$$I_{доп} = I_{вст}/a$$

(8.5)

Ідоп = 50 / 3 = 16,6 А

Вибираємо площу перерізу 10 мм^2 (Sф) при числі проводів і = 4 розташований у повітрі. Визначаємо максимальний робочий струм:

Іроб = Ко(Кз*(Р*1000)/(√3*Uл*cosф)*m+Кз*(Ро*1000)/(√3*Uл*cosф)) (8.6)

Іроб = 0,75*(0,85*((5*1000)/(1,73*380*0,85)*10+(3*1000)/(1,73*380*0,85))=60,4 А

Ко − коефіцієнт одночасності роботи групи електроприймачів;

Ко = 0.7...0.8; Кз = 0.8...0.9;

Кз – коефіцієнт завантажених електродвигунів;

Ро =3кВт − потужність освітлювальної мережі;

Визначається струм короткочасного перевантаження магістрального кабеля:

Іпер = Ко(Кз*(Р*1000)/(√3*Uл*cosф)*n+Кз*(Ро*1000)/(√3*Uл*cosф))+Іпус (8.7)

Іпер = 0,75*(0,85 *(5*1000)/(1,73*380*0,85)*9+0,85*(30*1000)/(1,73*380 * 0,85)) + 44,5 = 130,0643 А

Струм спрацювання електромагнітного розчеплювача додатково перевіряємо по максимальному струму перевантаження лінії:

Іспр >= 1,25 * Іпер

(8.7)

Іспр >=1,25 * 130,0643 = 162,5803 А

Приймаємо Іспр = 162,5803 А. Вимикач : А3714Б.

Вибираємо площу перерізу Sф магістрального кабеля (провідника) по Ідоп. Sф = 70mm^2 , − кабель АВРГ прокладений в землі, і=3 (число проводів). Вибрану площу перерізу перевіряємо для автоматів з електромагнітним розчеплювачем:

$$Ідоп >= Іспр/4,5 \quad (8.8)$$

$$Ідоп >= 162/4,5 = 36 А$$

Проводимо узгодження з номінальним струмом автомата:

$$Ідоп = Іспр/3 \quad (8.9)$$

$$Ідоп = 162/3 = 54 А$$

Значення 36 і 54 А. менше ніж Imax= 60,4 А., значить площа перерізу кабеля вибрана вірно. Визначаємо потужність трансформатора:

$$Nтр = ((Кп * Рном)/cosф) \quad (8.10)$$

$$Nтр = (0,7 * 53)/0,8 = 46,375 кВ*А$$

Рном − сумарна потужність електроприймачів, кВт;

Cosф − середній коефіцієнт потужності електроприймачів (0,8);

Кп − коефіцієнт попиту (0,7);

Одержане значення потужності трансформатора округляємо до ближчого стандартного значення. Визначаємо опір трансформатора Z_T. Вибираємо

трансформатор на 40 кВА (Z_T =0,562 Ом). визначаємо орієнтовно площу перерізу провідника. Для магістрального кабеля:

$$Sn_1 >= 0,5 * Sф \quad (8.11)$$

$$Sn_1 >= 0,5 * Sф = 0,5*70=35 \text{ мм}^2$$

Визначаємо для розгалуження:

$$Sn_2 >= 0,5*10 = 5 \text{ мм}^2$$

Округляємо ці значення до найближчих більших 35 мм^2 (Sn1), і 6 мм^2 (Sn2). Визначаємо активний і індуктивний опір фазного і нульового захисного провідників на дільницях 1 і 2:

$$Rф = p * (Lм/Sф_1) + p * (L/Sф_2) \quad (8.12)$$

$$Rф = 0,028 * (70/70) + 0,028 * (22/10) = 0,0896 \text{ Ом}$$

$$Rn = p * (Lm/Sn_1) + p * (L/Sn_2) \quad (8.13)$$

$$Rn = 0,028 * (70/35) + 0,028 * (22/6) = 0,1586 \text{ Ом}$$

Для окремо проложених нульових провідників иого приимають рівним 0,6 Ом/км. При прокладці кабелем, або в сталєвих трубах індуктивним опором нехтують.

Знаходимо дійсне значення (модуль) струма однофазного короткого замикання:

$$Iкр = Uф/((Zт/3) + \sqrt{((Rф + Rn)^2 + (Xф + Xn + X'n^2))} \quad (8.14)$$

$$Iкр = 220/((0,562/3) + \sqrt{(0,0896 + 0,1586)^2}) = 418,18 \text{ А}$$

Визначення максимальної напруги на корпусі обладнання відносно землі при замиканні фази на корпус.

$$Uктах = Iкр * Zн \quad (8.15)$$

$$Uкmax = 418,18 * 0,1586 = 66,32 \text{ В}$$

$Z_Н$ − повний опір нульового провідника.

Умова не виконується. Необхіно збільшити перерізи Sn 1 та Sn 2 до Sф1 та Sф2 і зробити перерахунок:

$$Rn = 0,028 * (70/70) + 0,028 * (22/10) = 0,0896 \text{ Ом},$$

$$Iкр = 220/((0,562/3) + \sqrt{(0,0896 + 0,0896)^2}) = 600,21 \text{ А},$$

$$Uкmax = 600,21 * 0,0896 = 53,77 \text{ В}.$$

Умова не виконується, необхідно або замінити запобіжник з плавкою вставкою на автоматичний вимикач із струмовим реле, що дає можливість зменшити час замикання на копус і підвищити допустиму напругу на корпусі або застосувати повторне заземлення нульового захисного провідника. Повторне заземлення нульового захисного провідника:

$$Rn=(Uдоп * Ro)/((Iкр * Zн)-Uдоп) \quad (8.15)$$

$$Rn= 36 * 4/((600,21 * 0,0896) - 36) = 8,09 Ом.$$

8.4 Висновки

Існує багато факторів, які можуть вплинути на роботу розробника, через некомфортні або навіть небезпечні умови праці, які знаходяться в приміщенні. Серед основних причин виділяється: недостатній рівень світла, гучний шум, високий рівень навантаження, умови мікроклімату. Під час дослідження теми, були переглянуті можливі шкідливі та небезпечні ситуації, які можуть виникнути на робочому місці та спооби їх уникнення та ліквідації.

В результаті можна зробити висновки, які умови повині бути для продуктивної роботи працівника, як повино бути організоване приміщення і його робоче місце. За допомогою проведених обчислень, можна становити необхідні для комфортної роботи умови.

9 ОСНОВНІ ВИСНОВКИ

В роботі розглянуті питання, пов'язані з міграцією даних, що зберігаються в реляційних базах даних, і процес її виконання. В рамках даної роботи досягнуті наступні результати:

- розширений механізм перенесення даних для виконання паралельної міграції даних;

- розроблений і реалізований алгоритм побудови послідовності обходу схеми бази даних;

- розроблений і реалізований алгоритм порівняння схем баз даних;

- створений виконуваний еволюційний прототип додатка для міграції даних.

У результаті виконання атестаційної роботи було проведено дослідження міграції даних з MS Server до PostgreSQL. У рамках атестаційної магістерської роботи була доведена актуальність дослідження, проаналізовані існуючі аналоги системи, обґрунтована мета розробки додатку для міграції, який розглядається в цієї роботі та сформовані критерії ефективності, поставлена задача дослідження. На основі цих досліджень був розроблений алгоритм міграції даних, виконаний процес міграції з MS Server до PostgeSQL.

Задля демонстрації міграції та його практичного використання було розроблене демонстраційне автоматизоване програмне забезпечення.

У майбутньому планується продовження досліджень на тему міграції даних, а саме розширення мов баз даних.

За результатами дослідження можна зробити висновок що поставлена задача була успішно виконана, міграція даних була успішно виконана та проведені досліди

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі *2793* грн. Враховуючи вартість розробки та необхідне впровадження, строк окупності становить 0,15 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бахрушин В. Є. Методи аналізу даних: Навч. посібник / В. Є. Бахрушин. – Запоріжжя: КПУ, 2011. – 268 с
2. Шумейко А. А. Интеллектуальный анализ данных (Введение в Data Mining) / А. А. Шумейко, С. Л. Сотник. – Днепропетровск: Белая Е. А., 2015. – 212 с.
3. http://www.nbuv.gov.ua/eb/ep.html - Національна бібліотека України імені В.І.Вернадського

4. Node.js v14.0.0 Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/api/

5. М. Кантелон , М. Хартер, Т. Головайчук, Н. Райлих // “Node.js в действии”.

6. Янг А., Мек Б., Кантелон М. // “Node.js в действии. 2-е издание”.

7. John Resig, Bear Bibeault, Josip Maras // “Secrets of theJavaScript Ninja”.

8. Express [Електронний ресурс] – Режим доступу до ресурсу: http://expressjs.com/

9. Фреймворк AngularJS [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/AngularJS

10. AngularJS [Електронний ресурс] – Режим доступу до ресурсу: https://angularjs.org/

11. Bootstrap [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.com/

12. React.js [Електронний ресурс] – Режим доступу до ресурсу: https://https://uk.reactjs.org/

13. AngularJS MVC [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/angularjs/angularjs_mvc_architecture.htm

14. Vue.js [Електронний ресурс] – Режим доступу до ресурсу: https://vuejs.org/

15. Angular 2 [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/

16. Односторінковий застосунок [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Односторінковий_застосунок

18. Build Node.js Apps [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/docs/nodejs/nodejs-tutorial

19. REST [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/REST

20. What is REST [Електронний ресурс] – Режим доступу до ресурсу: https://restfulapi.net/

22.Веб приложение [Електронний ресурс] – Режим доступу до ресурсу: https://webcase.com.ua/blog/cho-takoe-web-prilozhenie-vse-vidy/

23.Мова розмітки гіпертексту [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/HTML

24. HTML [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/uk/docs/Web/HTML

25. Каскадні таблиці стилів [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/CSS

26. Стек MEAN [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/MEAN_(веброзробка)

27. MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/MongoDB

28. Веб-технології для розробників [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/uk/docs/Web

29. CORS, XSS [Електронний ресурс] – Режим доступу до ресурсу: https://dev.to/maleta/cors-xss-and-csrf-with-examples-in-10-minutes-35k3

30. AJAX [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/AJAX

31. Fetch API [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/API/Fetch_API

32. AngularJS Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/angular/default.asp

33. jQuery [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/JQuery

34. Основи Web-технологій [Електронний ресурс] – Режим доступу до ресурсу: https://pidruchniki.com/1243020547796/informatika/web-tehnologiyi_pidpriyemstvah

35. npm [Електронний ресурс] – Режим доступу до ресурсу: https://www.npmjs.com/

36. Install MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://docs.mongodb.com/manual/administration/install-on-linux/

37. Как установить Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://www.digitalocean.com/community/tutorials/node-js-ubuntu-18-04-ru

38. Linux [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Linux

39. npm-audit [Електронний ресурс] – Режим доступу до ресурсу: https://docs.npmjs.com/cli/audit

40. TypeScript [Електронний ресурс] – Режим доступу до ресурсу: https://www.typescriptlang.org/

41. SQL [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/SQL

42. MongoDB Compass [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/products/compass

43. Postman [Електронний ресурс] – Режим доступу до ресурсу: https://www.postman.com/

44. SQL [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/SQL

45. SPA (Single-page application) [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Glossary/SPA

46. Охорона праці [Електронний ресурс]: реферат – Режим доступу до ресурсу: https://revolution.allbest.ru/life/00468031_0.html

47. Природне і штучне освітлення ДБН В.2.5-28:2018: державні будівельні норми України [Електронний ресурс] / Ю. Громадський, С. Облакевич, М.Громадський, Г. Фаренюк, Є. Фаренюк, О. Підгорний, О. Сергейчук, Є.Рейцен, В. Єгорченков, Л. Коваль, Д. Радомцев, В. Злоба, Н. Кучеренко, Г.Кожушко, О. Гончар, О. Козенко, Б. Шабашкевіч, Ю. Добровольський, В.Акіменко, С. Гозак, А. Яригін, В. Назаренко, В. Мартиросова, В. Сорокін, Є.Пугачов - Київ 2018 - Режим доступу до ресурсу: https://ledeffect.com.ua/images/__branding/dbn2018.pdf

48. Розрахунок світлодіодного освітлення кімнати [Електронний ресурс] – Режим доступу до ресурсу: https://luxled.biz.ua/rozrahynok-svitlodoidnogo-osvitlennja-kimnatu-v-kvarturi-abo-bydunky

49. Охорона праці, охорона праці та безпека в надзвичайних ситуаціях : метод. вказ. до викон. розділів у дипломних роботах / [укл. В.М. Челябієва, О.Л. Гуменюк] - Чернігів ЧДТУ 2013 - [Електронний ресурс] – Режим доступу до ресурсу: http://ir.stu.cn.ua/bitstream/handle/123456789/12461/Охорона праці та безпека. в надзв. ситуац;метод.вказ..pdf?sequence=1&isAllowed=y

50. Освітленість робочих місць [Електронний ресурс] – Режим доступу до ресурсу:https://ua-referat.com/Освітленість_робочих_місць_сучасні_підходи_до_вимірів_і_оцінки

51. Температурний режим праці [Електронний ресурс] – Режим доступу до ресурсу: http://poltava.medprof.org.ua/poltava/zakhist-trudovikh-ta-socialno-ekonomichnikh-prav-pracivnikiv-galuzi/pravova-dopomoga/temperaturnii-rezhim-praci-jakim-vin-maje-buti/

52. Шум. Методи захисту від його дії : метод. вказ. до лабораторної роботи / [укл. В. І. Шмирко, С. М. Журавель] — Запоріжжя: ЗНТУ, 2014. - 14 с.

[Електронний ресурс] – Режим доступу до ресурсу: https://zp.edu.ua/sites/default/files/konf/oop_shum-2014.pdf

53. Системи протипожежного захисту ДБН В.2.5-56:2014 / Б. Платкевич, В. Носач, В. Федюк, В. Мусійчук, В. Євстіфєєв, Г. Дубінський, В. Сокол, А.Бушиленко, В. Дунюшкін, Р. Уханський, С. Пономарьов, В. Приймаченко, А.Приймаченко, С. Пітайчук, Н. Морозова, І. Колосов, О. Лагода, П. Мізін, В.Савченко, М. Федорович, П. Шаповалов, Л. Фесенко — Київ 2015 — [Електронний ресурс] – Режим доступу до ресурсу: http://kbu.org.ua/assets/app/documents/dbn2/98.1. ДБН В.2.5-56~2014. Системи протипожежного захисту.pdf

54. Охорона праці. Ч. 2. Занулення : метод. вказ. до викон. розрахунків з викор. персон. ЕОМ IBM−сумісного типу / [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака, А. Е. Солових, С. Е. Катеринич] ; Центральноукраїн. нац. техн. ун-т. - 2−ге вид., перероб. та доп. - Кропивницький : ЦНТУ, 2019. - 27 с.[Електронний ресурс] – Режим доступу: http://dspace.kntu.kr.ua/jspui/handle/123456789/8769

55. The Top JavaScript Frameworks [Електронний ресурс] – Режим доступу до ресурсу: https://www.freecodecamp.org/news/complete-guide-for-front-end-developers-javascript-frameworks-2019/

56. JavaScript Frameworks 2020 [Електронний ресурс] – Режим доступу до ресурсу: https://www.npmjs.com/package/migrate-mongo

57. Web Technology [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/web-technology

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку програмного забезпечення системи міграції баз даних в корпоративних інформаційних системах.

2 Підстава для розробки

Підставою для розробки служить завдання на магістерську роботу, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 18-13 від 17.08.2022 року).

3 Мета та призначення розробки

Метою магістерської роботи є розробка програмного забезпечення для реалізації міграцій в базах даних, а саме між СУБД MS Server та PostgreSQL.

4 Джерела розробки

Джерелом цієї магістерської роботи є відносна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– Розробку додатку ;

– систему підключення та тестування баз даних ;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows XP/Vista/7/8/10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows XP/Vista/7/8/10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище PHP/Java

Бази даних MS Server та PostgreSQL

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2022 року.

8 Вимоги щодо охорони праці

В частині охорони праці та техніки безпеки в магістерській роботі повинен бути розглянутий аналіз умов праці програміста та розрахунок штучного освітлення.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 1 аркуш.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 103 аркуша.

10 Етапи розробки

10.1 Збір і обробка інформації по темі бакалаврської дипломної роботи. Постановка задачі на виконання бакалаврської дипломної роботи (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень бакалаврської дипломної роботи.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

11.1 Подання бакалаврської дипломної роботи на попередній захист 10.12.2022 р.

1.2 Подання магістерської роботи на захист 18.12.2022 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ
Керівник випускної кваліфікаційної роботи
за другим (магістерським) рівнем вищої освіти
_____ Босько В.В.

Дослідження та програмна реалізація процесів міграції баз даних в корпоративних інформаційних системах

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 38

Літера: РП

Кропивницький – 2022 року

Файл основного файлу програми для завантаження

```
/: c15:jdbc:VLookup.java
// GUI version of Lookup.java.
// <applet code=VLookup
// width=500 height=200></applet>
// {Broken}
import javax.swing.*;

import java.awt.*;

import java.awt.event.*;

import javax.swing.event.*;

import java.sql.*;

import com.bruceeckel.swing.*;

public class VLookup extends JApplet {
   String dbUrl = "jdbc:odbc:people";
   String user = "";
   String password = "";
   Statement s;
   JTextField searchFor = new JTextField(20);
   JLabel completion = new JLabel("                          ");
   JTextArea results = new JTextArea(40, 20);

   public void init() {
      searchFor.getDocument().addDocumentListener(new SearchL());
      JPanel p = new JPanel();
      p.add(new Label("Last name to search for:"));
      p.add(searchFor);
      p.add(completion);
      Container cp = getContentPane();
      cp.add(p, BorderLayout.NORTH);
      cp.add(results, BorderLayout.CENTER);
```

```
   try {
      // завантаження драйвера
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      Connection c = DriverManager.getConnection(dbUrl, user, password);
      s = c.createStatement();
   }
   catch (Exception e) {
      results.setText(e.toString());
   }
}

class SearchL implements DocumentListener {
   public void changedUpdate(DocumentEvent e) {
   }

   public void insertUpdate(DocumentEvent e) {
      textValueChanged();
   }

   public void removeUpdate(DocumentEvent e) {
      textValueChanged();
   }
}

public void textValueChanged() {
   ResultSet r;
   if (searchFor.getText().length() == 0) {
      completion.setText("");
      results.setText("");
      return;
   }
   try {
      // Автозавершение:
      r = s.executeQuery("SELECT LAST FROM people.csv people "
            + "WHERE (LAST Like '" + searchFor.getText()
            + "%') ORDER BY LAST");
      if (r.next())
```

```
            completion.setText(r.getString("last"));
            r = s.executeQuery("SELECT FIRST, LAST, EMAIL "
                  + "FROM people.csv people " + "WHERE (LAST='"
                  + completion.getText() + "') AND (EMAIL Is Not Null) "
                  + "ORDER BY FIRST");
         }
         catch (Exception e) {
            results.setText(searchFor.getText() + "n");
            results.append(e.toString());
            return;
         }
         results.setText("");
         try {
            while (r.next()) {
               results.append(r.getString("Last") + ", "
                     + r.getString("fIRST") + ": " + r.getString("EMAIL")
                     + "n");
            }
         }
         catch (Exception e) {
            results.setText(e.toString());
         }
      }

      public static void main(String[] args) {
         Console.run(new VLookup(), 500, 200);
      }
/: c15:jdbc:CIDSQL.java
// SQL створення строчок
public class CIDSQL {
   public static String[] sql = {
         // стоврення таблиці MEMBERS:
         "drop table MEMBERS",
         "create table MEMBERS " + "(MEM_ID INTEGER primary key, "
               + "MEM_UNAME VARCHAR(12) not null unique, "
               + "MEM_LNAME VARCHAR(40), " + "MEM_FNAME VARCHAR(20), "
               + "ADDRESS VARCHAR(40), " + "CITY VARCHAR(20), "
```

```
            + "STATE CHAR(4), " + "ZIP CHAR(5), " + "PHONE CHAR(12), "
            + "EMAIL VARCHAR(30))",
        "create unique index " + "LNAME_IDX on MEMBERS(MEM_LNAME)",
        //
        "drop table EVENTS",
        "create table EVENTS " + "(EVT_ID INTEGER primary key,"
            + "EVT_TITLE VARCHAR(30) not null,"
            + "EVT_TYPE VARCHAR(20), " + "LOC_ID INTEGER, "
            + "PRICE DECIMAL, " + "DATETIME TIMESTAMP)",
        "create unique index " + "TITLE_IDX on EVENTS(EVT_TITLE)",
        //
        "drop table EVTMEMS",
        "create table EVTMEMS " + "(MEM_ID INTEGER not null,"
            + "EVT_ID INTEGER not null, " + "MEM_ORD INTEGER)",
        "create unique index " + "EVTMEM_IDX on EVTMEMS(MEM_ID, EVT_ID)",
        // LOCATIONS
        "drop table LOCATIONS",
        "create table LOCATIONS " + "(LOC_ID INTEGER primary key,"
            + "LOC_NAME VARCHAR(30) not null,"
            + "CONTACT VARCHAR(50), " + "ADDRESS VARCHAR(40), "
            + "CITY VARCHAR(20), " + "STATE VARCHAR(4), "
            + "ZIP VARCHAR(5), " + "PHONE CHAR(12), "
            + "DIRECTIONS VARCHAR(4096))",
        "create unique index " + "NAME_IDX on LOCATIONS(LOC_NAME)", };
} // /:~
//: c15:jdbc:LoadDB.java
// Загрузка і тестування баз даних.
// {Broken}
import java.sql.*;

class TestSet {
   Object[][] data = {
        { "MEMBERS", new Integer(1), "dbartlett", "Bartlett", "David",
            "123 Mockingbird Lane", "Gettysburg", "PA", "19312",
            "123.456.7890", "bart@you.net" },
        { "MEMBERS", new Integer(2), "beckel", "Eckel", "Bruce",
            "123 Over Rainbow Lane", "Crested Butte", "CO", "81224",
```

```
            "123.456.7890", "beckel@you.net" },
        { "MEMBERS", new Integer(3), "rcastaneda", "Castaneda", "Robert",
            "123 Downunder Lane", "Sydney", "NSW", "12345",
            "123.456.7890", "rcastaneda@you.net" },
        { "LOCATIONS", new Integer(1), "Center for Arts", "Betty Wright",
            "123 Elk Ave.", "Crested Butte", "CO", "81224",
            "123.456.7890", "Go this way then that." },
        { "LOCATIONS", new Integer(2), "Witts End Conference Center",
            "John Wittig", "123 Music Drive", "Zoneville", "PA",
            "19123", "123.456.7890", "Go that way then this." },
        { "EVENTS", new Integer(1), "Project Management Myths",
            "Software Development", new Integer(1), new Float(2.50),
            "2000-07-17 19:30:00" },
        { "EVENTS", new Integer(2), "Life of the Crested Dog",
            "Archeology", new Integer(2), new Float(0.00),
            "2000-07-19 19:00:00" },
        // Match some people with events
        { "EVTMEMS", new Integer(1), // Dave is going to
            new Integer(1), // the Software event.
            new Integer(0) }, { "EVTMEMS", new Integer(2), // Bruce is
                                            // going to
            new Integer(2), // the Archeology event.
            new Integer(0) }, { "EVTMEMS", new Integer(3), // Robert is
                                            // going to
            new Integer(1), // the Software event.
            new Integer(1) }, { "EVTMEMS", new Integer(3), // ... and
            new Integer(2), // the Archeology event.
            new Integer(1) }, };

  public TestSet() {
  }
    public TestSet(Object[][] dat) {
     data = dat;
  }
}

public class LoadDB {
```

```
Statement statement;
Connection connection;
TestSet tset;

public LoadDB(TestSet t) throws SQLException {
   tset = t;
   try {
      // Загрузка драйвера (регистрирует себя)
      Class.forName(CIDConnect.dbDriver);
   }
   catch (java.lang.ClassNotFoundException e) {
      e.printStackTrace(System.err);
   }
   connection = DriverManager.getConnection(CIDConnect.dbURL,
         CIDConnect.user, CIDConnect.password);
   statement = connection.createStatement();
}

public void dispose() throws SQLException {
   statement.close();
   connection.close();
}

public void executeInsert(Object[] data) {
   String sql = "insert into " + data[0] + " values(";
   for (int i = 1; i < data.length; i++) {
      if (data[i] instanceof String)
         sql += "'" + data[i] + "'";
      else
         sql += data[i];
      if (i < data.length - 1)
         sql += ", ";
   }
   sql += ')';
   System.out.println(sql);
   try {
      statement.executeUpdate(sql);
```

```
      }
      catch (SQLException sqlEx) {
         System.err.println("Insert failed.");
         while (sqlEx != null) {
            System.err.println(sqlEx.toString());
            sqlEx = sqlEx.getNextException();
         }
      }
   }

   public void load() {
      for (int i = 0; i < tset.data.length; i++)
         executeInsert(tset.data[i]);
   }
     public static void main(String[] args) throws SQLException {
      LoadDB db = new LoadDB(new TestSet());
      db.load();
      try {
         // Отримуэмо ResultSet
         ResultSet rs = db.statement.executeQuery("select "
               + "e.EVT_TITLE, m.MEM_LNAME, m.MEM_FNAME "
               + "from EVENTS e, MEMBERS m, EVTMEMS em "
               + "where em.EVT_ID = 2 " + "and e.EVT_ID = em.EVT_ID "
               + "and m.MEM_ID = em.MEM_ID");
         while (rs.next())
            System.out.println(rs.getString(1) + " " + rs.getString(2)
               + ", " + rs.getString(3));
      }
      finally {
         db.dispose();
      }
   }
} // /:~

package com.javamaster.generics;

public class CollectionExample {
```

```
	private Object[] collectionElements;

	public CollectionExample() {
		collectionElements = new Object[10];
	}

	public void add(Object object, int position) {
		collectionElements[position] = object;
	}

	public Object get(int index) {
		return collectionElements[index];
	}

	public void printElements() {
		for(int i=0; i<collectionElements.length; i++) {
			if(collectionElements[i]!= null)
			System.out.println(collectionElements[i]);
		}
	}

	public static void main(String[] args) {
		CollectionExample c = new CollectionExample();
		c.add("Hi there", 0);
		c.add(1, 1);
		c.add(c, 2);
		c.printElements();
		System.out.println(c.get(0));
		int element = (int) c.get(0);
	}

}
```

```
package com.javamaster.generics;

public class CollectionExample<T> {
```

```
	private T [] collectionElements;

	public CollectionExample() {
		collectionElements = (T[]) new Object[10];
	}

	public void add(T object, int position) {
		collectionElements[position] = object;
	}

	public T get(int index) {
		return collectionElements[index];
	}

	public void printElements() {
		for(int i=0; i<collectionElements.length; i++) {
			if(collectionElements[i]!= null)
			System.out.println(collectionElements[i]);
		}
	}

	public static void main(String[] args) {
		CollectionExample<String> c = new CollectionExample();
		c.add("Hi there", 0);
	//	c.add(1, 1);
	//	c.add(c, 2);
		c.printElements();
		System.out.println(c.get(0));
		//int element = (int) c.get(0);
	}

//Приклади кода для міграцій
class m101129_185401_create_news_table extends CDbMigration
{
    public function up()
    {
```

```
        $this->createTable('tbl_news', array(
            'id' => 'pk',
            'title' => 'string NOT NULL',
            'content' => 'text',
        ));
    }
    public function down()
    {
        $this->dropTable('tbl_news');
    }
}
class m101129_185401_create_news_table extends CDbMigration
{
    public function up()
    {
        $transaction=$this->getDbConnection()->beginTransaction();
        try
        {
            $this->createTable('tbl_news', array(
                'id' => 'pk',
                'title' => 'string NOT NULL',
                'content' => 'text',
            ));
            $transaction->commit();
        }
        catch(Exception $e)
        {
            echo "Exception: ".$e->getMessage()."\n";
            $transaction->rollback();
            return false;
        }
    }

    // …схожий код для down()
}
class m101129_185401_create_news_table extends CDbMigration
{
```

```
    public function safeUp()
    {
        $this->createTable('tbl_news', array(
            'id' => 'pk',
            'title' => 'string NOT NULL',
            'content' => 'text',
        ));
    }
    public function safeDown()
    {
        $this->dropTable('tbl_news');
    }
}

return array(
    ......
    'commandMap'=>array(
        'migrate'=>array(
            'class'=>'system.cli.commands.MigrateCommand',
            'migrationPath'=>'application.migrations',
            'migrationTable'=>'tbl_migration',
            'connectionID'=>'db',
            'templateFile'=>'application.migrations.template',
        ),
        ......
    ),
    ......
```

Mig.rar Оболонка ПЗ

```
/** Make sure that the WordPress bootstrap has run before continuing. */
require( dirname( __FILE__ ) . '/wp-load.php' );

// Redirect to HTTPS login if forced to use SSL.
if ( force_ssl_admin() && ! is_ssl() ) {
      if ( 0 === strpos( $_SERVER['REQUEST_URI'], 'http' ) ) {
            wp_safe_redirect( set_url_scheme( $_SERVER['REQUEST_URI'], 'https' )
);
            exit();
```

```
	} else {
		wp_safe_redirect( 'https://' . $_SERVER['HTTP_HOST'] . $_SERVER['REQUEST_URI'] );
		exit();
	}
}

/**
 * Output the login page header.
 *
 * @since 2.1.0
 *
 * @param string   $title    Optional. WordPress login Page title to display in the `<title>` element.
 *                           Default 'Log In'.
 * @param string   $message  Optional. Message to display in header. Default empty.
 * @param WP_Error $wp_error Optional. The error to pass. Default is a WP_Error instance.
 */
function login_header( $title = 'Log In', $message = '', $wp_error = null ) {
	global $error, $interim_login, $action;

	// Don't index any of these forms
	add_action( 'login_head', 'wp_sensitive_page_meta' );

	add_action( 'login_head', 'wp_login_viewport_meta' );

	if ( ! is_wp_error( $wp_error ) ) {
		$wp_error = new WP_Error();
	}

	// Shake it!
	$shake_error_codes = array( 'empty_password', 'empty_email', 'invalid_email', 'invalidcombo', 'empty_username', 'invalid_username', 'incorrect_password' );
	/**
	 * Filters the error codes array for shaking the login form.
	 *
	 * @since 3.0.0
```

```
	 *
	 * @param array $shake_error_codes Error codes that shake the login form.
	 */
	$shake_error_codes = apply_filters( 'shake_error_codes', $shake_error_codes );

	if ( $shake_error_codes && $wp_error->has_errors() && in_array( $wp_error->get_error_code(), $shake_error_codes ) ) {
		add_action( 'login_head', 'wp_shake_js', 12 );
	}

	$login_title = get_bloginfo( 'name', 'display' );

	/* translators: Login screen title. 1: Login screen name, 2: Network or site name */
	$login_title = sprintf( __( '%1$s &lsaquo; %2$s &#8212; WordPress' ), $title, $login_title );

	/**
	 * Filters the title tag content for login page.
	 *
	 * @since 4.9.0
	 *
	 * @param string $login_title The page title, with extra context added.
	 * @param string $title       The original page title.
	 */
	$login_title = apply_filters( 'login_title', $login_title, $title );

	?><!DOCTYPE html>
	<!--[if IE 8]>
		<html xmlns="http://www.w3.org/1999/xhtml" class="ie8" <?php language_attributes(); ?>>
	<![endif]-->
	<!--[if !(IE 8) ]><!-->
		<html xmlns="http://www.w3.org/1999/xhtml" <?php language_attributes(); ?>>
	<!--<![endif]-->
	<head>
	<meta http-equiv="Content-Type" content="<?php bloginfo( 'html_type' ); ?>; charset=<?php bloginfo( 'charset' ); ?>" />
```

```
	<title><?php echo $login_title; ?></title>
	<?php

	wp_enqueue_style( 'login' );

	/*
	 * Remove all stored post data on logging out.
	 * This could be added by add_action('login_head'...) like wp_shake_js(),
	 * but maybe better if it's not removable by plugins.
	 */
	if ( 'loggedout' == $wp_error->get_error_code() ) {
		?>
		<script>if("sessionStorage" in window){try{for(var key in
sessionStorage){if(key.indexOf("wp-autosave-")!=-
1){sessionStorage.removeItem(key)}}}catch(e){}};</script>
		<?php
	}

	/**
	 * Enqueue scripts and styles for the login page.
	 *
	 * @since 3.1.0
	 */
	do_action( 'login_enqueue_scripts' );

	/**
	 * Fires in the login page header after scripts are enqueued.
	 *
	 * @since 2.1.0
	 */
	do_action( 'login_head' );

	if ( is_multisite() ) {
		$login_header_url   = network_home_url();
		$login_header_title = get_network()->site_name;
	} else {
		$login_header_url   = __( 'https://wordpress.org/' );
		$login_header_title = __( 'Powered by WordPress' );
```

```
	}

	/**
	 * Filters link URL of the header logo above login form.
	 *
	 * @since 2.1.0
	 *
	 * @param string $login_header_url Login header logo URL.
	 */
	$login_header_url = apply_filters( 'login_headerurl', $login_header_url );

	/**
	 * Filters the title attribute of the header logo above login form.
	 *
	 * @since 2.1.0
	 *
	 * @param string $login_header_title Login header logo title attribute.
	 */
	$login_header_title = apply_filters( 'login_headertitle',
$login_header_title );

	/*
	 * To match the URL/title set above, Multisite sites have the blog name,
	 * while single sites get the header title.
	 */
	if ( is_multisite() ) {
		$login_header_text = get_bloginfo( 'name', 'display' );
	} else {
		$login_header_text = $login_header_title;
	}

	$classes = array( 'login-action-' . $action, 'wp-core-ui' );
	if ( is_rtl() ) {
		$classes[] = 'rtl';
	}
	if ( $interim_login ) {
		$classes[] = 'interim-login';
		?>
```

```
            <style type="text/css">html{background-color: transparent;}</style>
            <?php

            if ( 'success' === $interim_login ) {
                  $classes[] = 'interim-login-success';
            }
      }
      $classes[] = ' locale-' . sanitize_html_class( strtolower( str_replace(
'_', '-', get_locale() ) ) );

      /**
       * Filters the login page body classes.
       *
       * @since 3.5.0
       *
       * @param array  $classes An array of body classes.
       * @param string $action  The action that brought the visitor to the login
page.
       */
      $classes = apply_filters( 'login_body_class', $classes, $action );

      ?>
      </head>
      <body class="login <?php echo esc_attr( implode( ' ', $classes ) ); ?>">
      <?php
      /**
       * Fires in the login page header after the body tag is opened.
       *
       * @since 4.6.0
       */
      do_action( 'login_header' );
      ?>
      <div id="login">
            <h1><a href="<?php echo esc_url( $login_header_url ); ?>"
title="<?php echo esc_attr( $login_header_title ); ?>"><?php echo
$login_header_text; ?></a></h1>
      <?php

      unset( $login_header_url, $login_header_title );
```

```
	/**
	 * Filters the message to display above the login form.
	 *
	 * @since 2.1.0
	 *
	 * @param string $message Login message text.
	 */
	$message = apply_filters( 'login_message', $message );
	if ( ! empty( $message ) ) {
		echo $message . "\n";
	}

	// In case a plugin uses $error rather than the $wp_errors object.
	if ( ! empty( $error ) ) {
		$wp_error->add( 'error', $error );
		unset( $error );
	}

	if ( $wp_error->has_errors() ) {
		$errors   = '';
		$messages = '';
		foreach ( $wp_error->get_error_codes() as $code ) {
			$severity = $wp_error->get_error_data( $code );
			foreach ( $wp_error->get_error_messages( $code ) as
$error_message ) {
				if ( 'message' == $severity ) {
					$messages .= '    ' . $error_message . "<br />\n";
				} else {
					$errors .= '      ' . $error_message . "<br />\n";
				}
			}
		}
		if ( ! empty( $errors ) ) {
			/**
			 * Filters the error messages displayed above the login form.
			 *
			 * @since 2.1.0
```

```
                 *
                 * @param string $errors Login error message.
                 */
                echo '<div id="login_error">' . apply_filters( 'login_errors',
$errors ) . "</div>\n";
            }
            if ( ! empty( $messages ) ) {
                /**
                 * Filters instructional messages displayed above the login
form.
                 *
                 * @since 2.5.0
                 *
                 * @param string $messages Login messages.
                 */
                echo '<p class="message">' . apply_filters( 'login_messages',
$messages ) . "</p>\n";
            }
      }
} // End of login_header()

/**
 * Outputs the footer for the login page.
 *
 * @since 3.1.0
 *
 * @param string $input_id Which input to auto-focus.
 */
function login_footer( $input_id = '' ) {
      global $interim_login;

      // Don't allow interim logins to navigate away from the page.
      if ( ! $interim_login ) :
            ?>
      <p id="backtoblog"><a href="<?php echo esc_url( home_url( '/' ) ); ?>">
            <?php
            /* translators: %s: site title */
            printf( _x( '&larr; Back to %s', 'site' ), get_bloginfo( 'title',
'display' ) );
```

```
            ?>
      </a></p>
            <?php the_privacy_policy_link( '<div class="privacy-policy-page-
link">', '</div>' ); ?>
      <?php endif; ?>


      </div>


      <?php if ( ! empty( $input_id ) ) : ?>
      <script type="text/javascript">
      try{document.getElementById('<?php echo $input_id;
?>').focus();}catch(e){}
      if(typeof wpOnload=='function')wpOnload();
      </script>
      <?php endif; ?>


      <?php
      /**
       * Fires in the login page footer.
       *
       * @since 3.1.0
       */
      do_action( 'login_footer' );
      ?>
      <div class="clear"></div>
      </body>
      </html>
      <?php
}


/**
 * Outputs the Javascript to handle the form shaking.
 *
 * @since 3.0.0
 */
function wp_shake_js() {
      ?>
<script type="text/javascript">
```

```
addLoadEvent = function(func){if(typeof
jQuery!="undefined")jQuery(document).ready(func);else if(typeof
wpOnload!='function'){wpOnload=func;}else{var
oldonload=wpOnload;wpOnload=function(){oldonload();func();}}};

function s(id,pos){g(id).left=pos+'px';}

function g(id){return document.getElementById(id).style;}

function
shake(id,a,d){c=a.shift();s(id,c);if(a.length>0){setTimeout(function(){shake(id,
a,d);},d);}else{try{g(id).position='static';wp_attempt_focus();}catch(e){}}}

addLoadEvent(function(){ var p=new Array(15,30,15,0,-15,-30,-
15,0);p=p.concat(p.concat(p));var
i=document.forms[0].id;g(i).position='relative';shake(i,p,20);});

</script>

      <?php

}


/**

 * Outputs the viewport meta tag.

 *

 * @since 3.7.0

 */

function wp_login_viewport_meta() {

      ?>

      <meta name="viewport" content="width=device-width" />

      <?php

}


/**

 * Handles sending password retrieval email to user.

 *

 * @since 2.5.0

 *

 * @return bool|WP_Error True: when finish. WP_Error on error

 */

function retrieve_password() {

      $errors = new WP_Error();


      if ( empty( $_POST['user_login'] ) || ! is_string( $_POST['user_login'] )
) {

            $errors->add( 'empty_username', __( '<strong>ERROR</strong>: Enter a
username or email address.' ) );
```

```
	} elseif ( strpos( $_POST['user_login'], '@' ) ) {
		$user_data = get_user_by( 'email', trim( wp_unslash( $_POST['user_login'] ) ) );
		if ( empty( $user_data ) ) {
			$errors->add( 'invalid_email', __( '<strong>ERROR</strong>: There is no account with that username or email address.' ) );
		}
	} else {
		$login     = trim( $_POST['user_login'] );
		$user_data = get_user_by( 'login', $login );
	}

	/**
	 * Fires before errors are returned from a password reset request.
	 *
	 * @since 2.1.0
	 * @since 4.4.0 Added the `$errors` parameter.
	 *
	 * @param WP_Error $errors A WP_Error object containing any errors generated
	 *                         by using invalid credentials.
	 */
	do_action( 'lostpassword_post', $errors );

	if ( $errors->has_errors() ) {
		return $errors;
	}

	if ( ! $user_data ) {
		$errors->add( 'invalidcombo', __( '<strong>ERROR</strong>: There is no account with that username or email address.' ) );
		return $errors;
	}

	// Redefining user_login ensures we return the right case in the email.
	$user_login = $user_data->user_login;
	$user_email = $user_data->user_email;
	$key        = get_password_reset_key( $user_data );
```

```
	if ( is_wp_error( $key ) ) {
		return $key;
	}

	if ( is_multisite() ) {
		$site_name = get_network()->site_name;
	} else {
		/*
		 * The blogname option is escaped with esc_html on the way into the database
		 * in sanitize_option we want to reverse this for the plain text arena of emails.
		 */
		$site_name = wp_specialchars_decode( get_option( 'blogname' ), ENT_QUOTES );
	}

	$message = __( 'Someone has requested a password reset for the following account:' ) . "\r\n\r\n";
	/* translators: %s: site name */
	$message .= sprintf( __( 'Site Name: %s' ), $site_name ) . "\r\n\r\n";
	/* translators: %s: user login */
	$message .= sprintf( __( 'Username: %s' ), $user_login ) . "\r\n\r\n";
	$message .= __( 'If this was a mistake, just ignore this email and nothing will happen.' ) . "\r\n\r\n";
	$message .= __( 'To reset your password, visit the following address:' ) . "\r\n\r\n";
	$message .= '<' . network_site_url( "wp-login.php?action=rp&key=$key&login=" . rawurlencode( $user_login ), 'login' ) . ">\r\n";

	/* translators: Password reset email subject. %s: Site name */
	$title = sprintf( __( '[%s] Password Reset' ), $site_name );

	/**
	 * Filters the subject of the password reset email.
	 *
	 * @since 2.8.0
	 * @since 4.4.0 Added the `$user_login` and `$user_data` parameters.
	 *
	 * @param string  $title      Default email title.
```

```
       * @param string  $user_login The username for the user.
       * @param WP_User $user_data  WP_User object.
       */
      $title = apply_filters( 'retrieve_password_title', $title, $user_login,
$user_data );

      /**
       * Filters the message body of the password reset mail.
       *
       * If the filtered message is empty, the password reset email will not be
sent.
       *
       * @since 2.8.0
       * @since 4.1.0 Added `$user_login` and `$user_data` parameters.
       *
       * @param string  $message    Default mail message.
       * @param string  $key        The activation key.
       * @param string  $user_login The username for the user.
       * @param WP_User $user_data  WP_User object.
       */
      $message = apply_filters( 'retrieve_password_message', $message, $key,
$user_login, $user_data );

      if ( $message && ! wp_mail( $user_email, wp_specialchars_decode( $title ),
$message ) ) {
            wp_die( __( 'The email could not be sent.' ) . "<br />\n" . __(
'Possible reason: your host may have disabled the mail() function.' ) );
      }

      return true;
}

//
// Main.
//

$action = isset( $_REQUEST['action'] ) ? $_REQUEST['action'] : 'login';
$errors = new WP_Error();
```

```
if ( isset( $_GET['key'] ) ) {
	$action = 'resetpass';
}

// Validate action so as to default to the login screen.
if ( ! in_array( $action, array( 'postpass', 'logout', 'lostpassword',
'retrievepassword', 'resetpass', 'rp', 'register', 'login', 'confirmaction' ),
true ) && false === has_filter( 'login_form_' . $action ) ) {
	$action = 'login';
}

nocache_headers();

header( 'Content-Type: ' . get_bloginfo( 'html_type' ) . '; charset=' .
get_bloginfo( 'charset' ) );

if ( defined( 'RELOCATE' ) && RELOCATE ) { // Move flag is set
	if ( isset( $_SERVER['PATH_INFO'] ) && ( $_SERVER['PATH_INFO'] !=
$_SERVER['PHP_SELF'] ) ) {
		$_SERVER['PHP_SELF'] = str_replace( $_SERVER['PATH_INFO'], '',
$_SERVER['PHP_SELF'] );
	}

	$url = dirname( set_url_scheme( 'http://' . $_SERVER['HTTP_HOST'] .
$_SERVER['PHP_SELF'] ) );
	if ( $url != get_option( 'siteurl' ) ) {
		update_option( 'siteurl', $url );
	}
}

//Set a cookie now to see if they are supported by the browser.
$secure = ( 'https' === parse_url( wp_login_url(), PHP_URL_SCHEME ) );
setcookie( TEST_COOKIE, 'WP Cookie check', 0, COOKIEPATH, COOKIE_DOMAIN, $secure
);
if ( SITECOOKIEPATH != COOKIEPATH ) {
	setcookie( TEST_COOKIE, 'WP Cookie check', 0, SITECOOKIEPATH,
COOKIE_DOMAIN, $secure );
}

/**
 * Fires when the login form is initialized.
```

```
 *
 * @since 3.2.0
 */
do_action( 'login_init' );

/**
 * Fires before a specified login form action.
 *
 * The dynamic portion of the hook name, `$action`, refers to the action
 * that brought the visitor to the login form. Actions include 'postpass',
 * 'logout', 'lostpassword', etc.
 *
 * @since 2.8.0
 */
do_action( "login_form_{$action}" );

$http_post     = ( 'POST' == $_SERVER['REQUEST_METHOD'] );
$interim_login = isset( $_REQUEST['interim-login'] );

/**
 * Filters the separator used between login form navigation links.
 *
 * @since 4.9.0
 *
 * @param string $login_link_separator The separator used between login form
navigation links.
 */
$login_link_separator = apply_filters( 'login_link_separator', ' | ' );

switch ( $action ) {

    case 'postpass':
        if ( ! array_key_exists( 'post_password', $_POST ) ) {
            wp_safe_redirect( wp_get_referer() );
            exit();
        }

        require_once ABSPATH . WPINC . '/class-phpass.php';
```

```
            $hasher = new PasswordHash( 8, true );

            /**
             * Filters the life span of the post password cookie.
             *
             * By default, the cookie expires 10 days from creation. To turn
this
             * into a session cookie, return 0.
             *
             * @since 3.7.0
             *
             * @param int $expires The expiry time, as passed to setcookie().
             */
            $expire  = apply_filters( 'post_password_expires', time() + 10 *
DAY_IN_SECONDS );
            $referer = wp_get_referer();
            if ( $referer ) {
                  $secure = ( 'https' === parse_url( $referer, PHP_URL_SCHEME )
);
            } else {
                  $secure = false;
            }
            setcookie( 'wp-postpass_' . COOKIEHASH, $hasher->HashPassword(
wp_unslash( $_POST['post_password'] ) ), $expire, COOKIEPATH, COOKIE_DOMAIN,
$secure );

            wp_safe_redirect( wp_get_referer() );
            exit();

      case 'logout':
            check_admin_referer( 'log-out' );

            $user = wp_get_current_user();

            wp_logout();

            if ( ! empty( $_REQUEST['redirect_to'] ) ) {
                  $redirect_to = $requested_redirect_to =
$_REQUEST['redirect_to'];
            } else {
```

```
                $redirect_to           = 'wp-login.php?loggedout=true';
                $requested_redirect_to = '';
            }

            /**
             * Filters the log out redirect URL.
             *
             * @since 4.2.0
             *
             * @param string  $redirect_to           The redirect destination
URL.
             * @param string  $requested_redirect_to The requested redirect
destination URL passed as a parameter.
             * @param WP_User $user                  The WP_User object for the
user that's logging out.
             */
            $redirect_to = apply_filters( 'logout_redirect', $redirect_to,
$requested_redirect_to, $user );
            wp_safe_redirect( $redirect_to );
            exit();

      case 'lostpassword':
      case 'retrievepassword':
            if ( $http_post ) {
                $errors = retrieve_password();
                if ( ! is_wp_error( $errors ) ) {
                    $redirect_to = ! empty( $_REQUEST['redirect_to'] ) ?
$_REQUEST['redirect_to'] : 'wp-login.php?checkemail=confirm';
                    wp_safe_redirect( $redirect_to );
                    exit();
                }
            }

            if ( isset( $_GET['error'] ) ) {
                if ( 'invalidkey' == $_GET['error'] ) {
                    $errors->add( 'invalidkey', __( 'Your password reset
link appears to be invalid. Please request a new link below.' ) );
                } elseif ( 'expiredkey' == $_GET['error'] ) {
                    $errors->add( 'expiredkey', __( 'Your password reset
link has expired. Please request a new link below.' ) );
```

```
                }
            }

            $lostpassword_redirect = ! empty( $_REQUEST['redirect_to'] ) ?
$_REQUEST['redirect_to'] : '';
            /**
             * Filters the URL redirected to after submitting the
lostpassword/retrievepassword form.
             *
             * @since 3.0.0
             *
             * @param string $lostpassword_redirect The redirect destination
URL.
             */
            $redirect_to = apply_filters( 'lostpassword_redirect',
$lostpassword_redirect );

            /**
             * Fires before the lost password form.
             *
             * @since 1.5.1
             * @since 5.1.0 Added the `$errors` parameter.
             *
             * @param WP_Error $errors A `WP_Error` object containing any errors
generated by using invalid
             *                         credentials. Note that the error object
may not contain any errors.
             */
            do_action( 'lost_password', $errors );

            login_header( __( 'Lost Password' ), '<p class="message">' . __(
'Please enter your username or email address. You will receive a link to create
a new password via email.' ) . '</p>', $errors );

            $user_login = '';

            if ( isset( $_POST['user_login'] ) && is_string(
$_POST['user_login'] ) ) {
                $user_login = wp_unslash( $_POST['user_login'] );
            }
```

```
            ?>


      <form name="lostpasswordform" id="lostpasswordform" action="<?php echo
esc_url( network_site_url( 'wp-login.php?action=lostpassword', 'login_post' ) );
?>" method="post">

      <p>

            <label for="user_login" ><?php _e( 'Username or Email Address' );
?><br />

            <input type="text" name="user_login" id="user_login" class="input"
value="<?php echo esc_attr( $user_login ); ?>" size="20" autocapitalize="off"
/></label>

      </p>

            <?php

            /**

             * Fires inside the lostpassword form tags, before the hidden
fields.

             *

             * @since 2.1.0

             */

            do_action( 'lostpassword_form' );

            ?>

            <input type="hidden" name="redirect_to" value="<?php echo esc_attr(
$redirect_to ); ?>" />

            <p class="submit"><input type="submit" name="wp-submit" id="wp-
submit" class="button button-primary button-large" value="<?php esc_attr_e( 'Get
New Password' ); ?>" /></p>

      </form>


      <p id="nav">

      <a href="<?php echo esc_url( wp_login_url() ); ?>"><?php _e( 'Log in' );
?></a>

            <?php

            if ( get_option( 'users_can_register' ) ) :

                  $registration_url = sprintf( '<a href="%s">%s</a>', esc_url(
wp_registration_url() ), __( 'Register' ) );


                  echo esc_html( $login_link_separator );


                  /** This filter is documented in wp-includes/general-
template.php */

                  echo apply_filters( 'register', $registration_url );

      endif;
```

```
            ?>
      </p>

            <?php
            login_footer( 'user_login' );

            break;

      case 'resetpass':
      case 'rp':
            list( $rp_path ) = explode( '?', wp_unslash( $_SERVER['REQUEST_URI']
) );
            $rp_cookie       = 'wp-resetpass-' . COOKIEHASH;
            if ( isset( $_GET['key'] ) ) {
                  $value = sprintf( '%s:%s', wp_unslash( $_GET['login'] ),
wp_unslash( $_GET['key'] ) );
                  setcookie( $rp_cookie, $value, 0, $rp_path, COOKIE_DOMAIN,
is_ssl(), true );
                  wp_safe_redirect( remove_query_arg( array( 'key', 'login' ) )
);
                  exit;
            }

            if ( isset( $_COOKIE[ $rp_cookie ] ) && 0 < strpos( $_COOKIE[
$rp_cookie ], ':' ) ) {
                  list( $rp_login, $rp_key ) = explode( ':', wp_unslash(
$_COOKIE[ $rp_cookie ] ), 2 );
                  $user                      = check_password_reset_key(
$rp_key, $rp_login );
                  if ( isset( $_POST['pass1'] ) && ! hash_equals( $rp_key,
$_POST['rp_key'] ) ) {
                        $user = false;
                  }
            } else {
                  $user = false;
            }

            if ( ! $user || is_wp_error( $user ) ) {
                  setcookie( $rp_cookie, ' ', time() - YEAR_IN_SECONDS,
$rp_path, COOKIE_DOMAIN, is_ssl(), true );
                  if ( $user && $user->get_error_code() === 'expired_key' ) {
```

```
                        wp_redirect( site_url( 'wp-
login.php?action=lostpassword&error=expiredkey' ) );

                } else {

                        wp_redirect( site_url( 'wp-
login.php?action=lostpassword&error=invalidkey' ) );

                }

                exit;

        }


        $errors = new WP_Error();


        if ( isset( $_POST['pass1'] ) && $_POST['pass1'] != $_POST['pass2']
) {

                $errors->add( 'password_reset_mismatch', __( 'The passwords do
not match.' ) );

        }


        /**
         * Fires before the password reset procedure is validated.
         *
         * @since 3.5.0
         *
         * @param object           $errors WP Error object.
         * @param WP_User|WP_Error $user   WP_User object if the login and
reset key match. WP_Error object otherwise.
         */
        do_action( 'validate_password_reset', $errors, $user );


        if ( ( ! $errors->has_errors() ) && isset( $_POST['pass1'] ) && !
empty( $_POST['pass1'] ) ) {

                reset_password( $user, $_POST['pass1'] );

                setcookie( $rp_cookie, ' ', time() - YEAR_IN_SECONDS,
$rp_path, COOKIE_DOMAIN, is_ssl(), true );

                login_header( __( 'Password Reset' ), '<p class="message
reset-pass">' . __( 'Your password has been reset.' ) . ' <a href="' . esc_url(
wp_login_url() ) . '">' . __( 'Log in' ) . '</a></p>' );

                login_footer();

                exit;

        }


        wp_enqueue_script( 'utils' );
```

```
            wp_enqueue_script( 'user-profile' );


            login_header( __( 'Reset Password' ), '<p class="message reset-
pass">' . __( 'Enter your new password below.' ) . '</p>', $errors );


            ?>

      <form name="resetpassform" id="resetpassform" action="<?php echo esc_url(
network_site_url( 'wp-login.php?action=resetpass', 'login_post' ) ); ?>"
method="post" autocomplete="off">

      <input type="hidden" id="user_login" value="<?php echo esc_attr( $rp_login
); ?>" autocomplete="off" />


      <div class="user-pass1-wrap">

            <p>

                  <label for="pass1"><?php _e( 'New password' ); ?></label>

            </p>


            <div class="wp-pwd">

                  <div class="password-input-wrapper">

                        <input type="password" data-reveal="1" data-pw="<?php
echo esc_attr( wp_generate_password( 16 ) ); ?>" name="pass1" id="pass1"
class="input password-input" size="24" value="" autocomplete="off" aria-
describedby="pass-strength-result" />

                        <span class="button button-secondary wp-hide-pw hide-if-
no-js">

                              <span class="dashicons dashicons-hidden"></span>

                        </span>

                  </div>

                  <div id="pass-strength-result" class="hide-if-no-js" aria-
live="polite"><?php _e( 'Strength indicator' ); ?></div>

            </div>

            <div class="pw-weak">

                  <label>

                        <input type="checkbox" name="pw_weak" class="pw-
checkbox" />

                        <?php _e( 'Confirm use of weak password' ); ?>

                  </label>

            </div>

      </div>


      <p class="user-pass2-wrap">
```

```
            <label for="pass2"><?php _e( 'Confirm new password' ); ?></label><br
/>
            <input type="password" name="pass2" id="pass2" class="input"
size="20" value="" autocomplete="off" />
      </p>


      <p class="description indicator-hint"><?php echo wp_get_password_hint();
?></p>
      <br class="clear" />


            <?php
            /**
             * Fires following the 'Strength indicator' meter in the user
password reset form.
             *
             * @since 3.9.0
             *
             * @param WP_User $user User object of the user whose password is
being reset.
             */
            do_action( 'resetpass_form', $user );
            ?>
      <input type="hidden" name="rp_key" value="<?php echo esc_attr( $rp_key );
?>" />
      <p class="submit"><input type="submit" name="wp-submit" id="wp-submit"
class="button button-primary button-large" value="<?php esc_attr_e( 'Reset
Password' ); ?>" /></p>
      </form>


      <p id="nav">
      <a href="<?php echo esc_url( wp_login_url() ); ?>"><?php _e( 'Log in' );
?></a>
            <?php
            if ( get_option( 'users_can_register' ) ) :
                  $registration_url = sprintf( '<a href="%s">%s</a>', esc_url(
wp_registration_url() ), __( 'Register' ) );


                  echo esc_html( $login_link_separator );


                  /** This filter is documented in wp-includes/general-
template.php */
                  echo apply_filters( 'register', $registration_url );
```

```
	endif;
		?>
	</p>

		<?php
		login_footer( 'user_pass' );

		break;

	case 'register':
		if ( is_multisite() ) {
			/**
			 * Filters the Multisite sign up URL.
			 *
			 * @since 3.0.0
			 *
			 * @param string $sign_up_url The sign up URL.
			 */
			wp_redirect( apply_filters( 'wp_signup_location',
network_site_url( 'wp-signup.php' ) ) );
			exit;
		}

		if ( ! get_option( 'users_can_register' ) ) {
			wp_redirect( site_url( 'wp-login.php?registration=disabled' )
);
			exit();
		}

		$user_login = '';
		$user_email = '';

		if ( $http_post ) {
			if ( isset( $_POST['user_login'] ) && is_string(
$_POST['user_login'] ) ) {
				$user_login = $_POST['user_login'];
			}
```

```
                if ( isset( $_POST['user_email'] ) && is_string(
$_POST['user_email'] ) ) {
                      $user_email = wp_unslash( $_POST['user_email'] );
                }

                $errors = register_new_user( $user_login, $user_email );
                if ( ! is_wp_error( $errors ) ) {
                      $redirect_to = ! empty( $_POST['redirect_to'] ) ?
$_POST['redirect_to'] : 'wp-login.php?checkemail=registered';
                      wp_safe_redirect( $redirect_to );
                      exit();
                }
          }

          $registration_redirect = ! empty( $_REQUEST['redirect_to'] ) ?
$_REQUEST['redirect_to'] : '';
          /**
           * Filters the registration redirect URL.
           *
           * @since 3.0.0
           *
           * @param string $registration_redirect The redirect destination
URL.
           */
          $redirect_to = apply_filters( 'registration_redirect',
$registration_redirect );
          login_header( __( 'Registration Form' ), '<p class="message
register">' . __( 'Register For This Site' ) . '</p>', $errors );
          ?>
     <form name="registerform" id="registerform" action="<?php echo esc_url(
site_url( 'wp-login.php?action=register', 'login_post' ) ); ?>" method="post"
novalidate="novalidate">
     <p>
          <label for="user_login"><?php _e( 'Username' ); ?><br />
          <input type="text" name="user_login" id="user_login" class="input"
value="<?php echo esc_attr( wp_unslash( $user_login ) ); ?>" size="20"
autocapitalize="off" /></label>
     </p>
     <p>
          <label for="user_email"><?php _e( 'Email' ); ?><br />
```

```
            <input type="email" name="user_email" id="user_email" class="input"
value="<?php echo esc_attr( wp_unslash( $user_email ) ); ?>" size="25"
/></label>

      </p>

            <?php

            /**

             * Fires following the 'Email' field in the user registration form.

             *

             * @since 2.1.0

             */

            do_action( 'register_form' );

            ?>

      <p id="reg_passmail"><?php _e( 'Registration confirmation will be emailed
to you.' ); ?></p>

      <br class="clear" />

      <input type="hidden" name="redirect_to" value="<?php echo esc_attr(
$redirect_to ); ?>" />

      <p class="submit"><input type="submit" name="wp-submit" id="wp-submit"
class="button button-primary button-large" value="<?php esc_attr_e( 'Register'
); ?>" /></p>

      </form>


      <p id="nav">

      <a href="<?php echo esc_url( wp_login_url() ); ?>"><?php _e( 'Log in' );
?></a>

            <?php echo esc_html( $login_link_separator ); ?>

      <a href="<?php echo esc_url( wp_lostpassword_url() ); ?>"><?php _e( 'Lost
your password?' ); ?></a>

      </p>


            <?php

            login_footer( 'user_login' );


            break;


      case 'confirmaction':

            if ( ! isset( $_GET['request_id'] ) ) {

                  wp_die( __( 'Invalid request.' ) );

            }


            $request_id = (int) $_GET['request_id'];
```

```
            if ( isset( $_GET['confirm_key'] ) ) {
                  $key    = sanitize_text_field( wp_unslash(
$_GET['confirm_key'] ) );
                  $result = wp_validate_user_request_key( $request_id, $key );
            } else {
                  $result = new WP_Error( 'invalid_key', __( 'Invalid key' ) );
            }

            if ( is_wp_error( $result ) ) {
                  wp_die( $result );
            }

            /**
```