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
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

за другим (магістерським) рівнем вищої освіти

на тему

“ Дослідження та програмна реалізація застосування об’єктно-орієнтованої бази даних”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-22М-1
ОПП «Комп'ютерні науки»
спеціальності 122 «Комп'ютерні науки»
_____ Тімченко М.М.
«____» _____ 2023р.

Керівник проекту
кандидат технічних наук, доцент
_____ Босько В.В.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *магістр*

Галузь знань *12 "Комп'ютерні науки"*

Спеціальність *122 "Комп'ютерні науки"*

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

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____Олексій СМІРНОВ
"____" _____2023року

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

Тімченку Миколі Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація застосування об'єктно-орієнтованої бази даних*

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

затверджені наказом вищого навчального закладу № 32-13 від 04.08.22

3. Строк подання роботи до захисту *08.01.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи: *Метою розробки є програмне дослідження та програмна реалізація застосування ООБД 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 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	09.11.2023 р.	17.11.2023 р.
Охорона праці	Оришака О.В., к.т.н., доцент	03.11.2023 р.	21.11.2023 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	12.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	18.10.2023 р.	
3.	Розробка моделі компонента	23.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	32.10.2023 р.	
6.	Програмування алгоритмів	11.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	16.11.2023 р.	
9.	Оформлення ПЗ	18.11.2023 р.	
10.	Попередній захист роботи	08.12.2023 р	

Дата видачі завдання
«__»_____20 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
«__»_____20 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Тімченко М.М. Дослідження та програмна реалізація застосування об'єктно-орієнтованої бази даних. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній магістерській роботі розроблено програмне забезпечення, яке призначено для реалізації застосувань ООБД баз даних в розробці ПЗ.

Метою розробки є дослідження та програмна реалізація системи аналізу застосування різних типів баз даних при розробці WEB застосунків.

Об'єктом дослідження є аналіз існуючих реалізацій об'єктно-орієнтованих БД з погляду на швидкості роботи.

Предметом дослідження є розробка ПЗ на основі ООБД.

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

Результат роботи – програмна реалізація ІС на основі ООБД.

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 у браузері Chrome, Firefox, Safari.

Програму розроблено в середовищі HTML 5, CSS 3, JavaScript + jQuery + AJAX, PHP + PDO, JSON+ООБД VelocityBD.

Ключові слова: комп'ютерна інженерія, веб-сайт, бази даних, ООП, ООБД.

ANNOTATION

Timchenko M.M. Research and software implementation of object-oriented database application. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this master's thesis, software was developed, which is intended for the implementation of OOBD database applications in software development.

The purpose of the development is research and software implementation of a system for analyzing the use of various types of databases in the development of WEB applications.

The object of the study is the analysis of existing implementations of object-oriented databases from the point of view of the speed of operation.

The subject of research is the development of software based on OOBD.

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 IS based on OOBD.

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 in Chrome, Firefox, and Safari browsers.

The program was developed in the HTML 5, CSS 3, JavaScript + jQuery + AJAX, PHP + PDO, JSON+OOBD VelocityBD environment.

Keywords: computer engineering, website, databases, OOP, OOBD.

ЗМІСТ

					ВКРМ-122.23.0022.00.00.ПЗ	
Вим.	Арк.	№ докум.	Підпис	Дат		
Розроб.		Тімченко М.М			*Дослідження та програмна реалізація застосування об'єктно-орієнтованої бази даних*	Літ. М / Арк. 1 / Аркушів
Перевір.		Босько В.В				
Н. Контр.		Коваленко А.С				ЦНТУ КН-22М-1
Затверд.		Смірнов О.А.				

ПЕРЕЛІК СКОРОЧЕНЬ, СИМВОЛІВ ТА СПЕЦІАЛЬНИХ ТЕРМІНІВ

ПЗ	–	Програмне забезпечення.
Хеш	–	Результат обробки даних хеш-функцією.
AJAX	–	Asynchronous Javascript and XML – підхід до побудови інтерактивних користувацьких інтерфейсів веб-додатків.
CSS3	–	Cascading Style Sheets 3 – каскадні таблиці стилів третього покоління.
HTML	–	HyperText Markup Language – мова розмітки гіпертексту.
		Hypertext Preprocessor – скриптова мова загального призначення, інтенсивно застосовується для розробки веб додатків.
БД	–	База даних.
PDO	–	PDO простий і універсальний інтерфейс для доступу до різних баз даних.
JSON	–	JavaScript Object Notation – текстовий формат обміну даними.
SQL		Structured query language – мова структурованих запитів.
ПК	–	Персональний комп’ютер
ООБД	–	Об’єктно-орієнтована база даних

ВСТУП

Актуальність теми. Сьогодні неможливо уявити відсутність інформаційних технологій у нашому житті. Сучасне виробництво, науку, культуру, економіку чи спорт важко уявити без участі комп'ютерів. Комп'ютеризоване програмне забезпечення має місце у всіх перерахованих вище сферах життєзабезпечення. Це допомагає людям у роботі, розвагах, освіті та наукових дослідженнях.

Жодна сучасна організація не може обійтися без бази даних. Це навчальні заклади, банки, магазини, заводи, будь-які компанії та державні установи. Вони використовують їх для перекладу даних в електронну форму та об'єднання даних, а також для оперативного доступу до них. Це дозволяє заощадити час і гроші на витратах. Звичайно, скорочення часу є лише побічним ефектом автоматизації. Найважливіше завдання розвитку інформаційних технологій полягає зовсім в іншому - в набутті тією чи іншою організацією виключно нових якостей, які надають їй значну конкурентоспроможність. А це дорого коштує. Крім того, зараз встановлення та керування базою даних не є таким складним процесом, як це було десятиліття тому. Коли проектування бази даних і керування не були автоматизовані. Система управління базами даних дозволяє створювати базу даних, оновлювати збережену в ній інформацію, надаючи швидкий доступ для перегляду та пошуку інформації. Актуальність теми полягає в тому, що в нових системах управління базами даних з'явилася функція не тільки зберігання даних в її структурах, а й можливе зберігання програмного коду, який використовується для взаємодії з користувачем, або програмно-апаратних засобів.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи аналізу застосування ООБД в сучасних ІС.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– огляд існуючих систем баз даних які використовуються в сучасних ІС;

– дослідження концепції об’єктно-орієнтованих баз даних;

– програмна реалізація ІС на основі СУООБД у якості сховища даних.

Об’єктом дослідження є аналіз існуючих реалізацій об’єктно-орієнтованих БД з погляду на продуктивність роботи.

Предметом дослідження є методи розробки сховищ даних на основі ООБД.

Методи дослідження базуються на методах зберігання даних, методах математичної статистики, методах розробки програмного забезпечення.

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

– удосконалено метод вибору ООСУБД для роботи з сучасними ІС;

– розроблено інформаційну систему на основі ООБД, яка має більш широкі можливості та швидкість доступу до даних на відміну від існуючих аналогів побудованих на класичних реляційних БД.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють більш успішно вирішувати задачі зберігання і обробки даних на основі ООБД ніж із застосуванням класичних реляційних СУБД.

Достовірність наукових результатів підтверджена теоретичними викладенням, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

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

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

Інформаційна система – це система, призначена для пошуку, зберігання та обробки інформації, та відповідні організаційні ресурси (технічні, фінансові, людські тощо), які забезпечують і поширюють інформацію [4].

Інформаційні системи були створені в 60-х роках минулого століття у військовій промисловості та бізнесі, де накопичувалися великі обсяги корисної інформації.

Спочатку інформаційні системи призначалися тільки для роботи з інформацією реального типу - текстовими або числовими характеристиками об'єктів. Потім, у міру розвитку технічного оснащення комп'ютерів, з'явилася можливість обробляти текстові дані природною мовою.

Інформаційна система служить для оперативного надання необхідних даних певній групі людей, тобто для задоволення всіх інформаційних потреб у межах обраної предметної області, а результатом роботи інформаційних систем є інформаційні продукти – текстові документи, рядки інформації, база даних.

Тому, інформаційні системи застосовують в нижченаведених сферах:

- створення сховища для інформації;
- концепція оцінки інформації;
- налагодженість затвердження постанови;
- мобільні та локальні бази даних;
- географічні бази даних;
- мультимедійні бази даних;
- розподілені інформаційні системи;
- бази даних для всесвітньої мережі.

1.1. Призначення системи

База даних є основною і складною частиною інформаційних систем, призначених для обробки та зберігання інформації. Спочатку ці системи були тільки в паперовому вигляді. Для їх зберігання використовувалися різні папки, кімнати та архіви.

Розвиток комп’ютерних технологій уможливив широке застосування автоматизованих інформаційних систем. На сьогоднішній день розробляються інформаційні системи, що обслуговують різні сфери діяльності, системи управління економічними і технічними об'єктами, модельні комплекси для наукових досліджень, системи автоматизації моделювання і виробництва, різноманітні навчальні системи.

Сучасні інформаційні системи базуються на ідеї сумісності даних, яка характеризується великими об’єктами збереженої інформації, складною організацією та необхідністю задовольнити різноманітні потреби багатьох користувачів.

Системи керування базами даних призначені для керування цими даними та забезпечення ефективного доступу до них.

У магістерській роботі реалізована інформаційна система кафедри, яка використовує як сховище даних об’єктно-орієнтовану базу даних VelocitiDb. Вибір бази даних базується на результатах порівняльного аналізу баз даних.

Інформаційна система в базі даних зберігає дані про працівників університету, а також методичні вказівки та посібники, видані працівниками. Співробітник може додавати роботу.

Також інформаційна система здатна повідомити користувача про те, що у колег сьогодні день народження та іншу важливу інформацію. Також є можливість переглянути книги та методичні вказівки.

1.2. Область застосування

Програмне забезпечення, що розробляється, може мати досить широке застосування. Це пов'язано з тим, що такі питання, проекти і завдання завжди з'являються в нашому житті.

Це програмне забезпечення може бути спрямоване на будь-яку організацію чи групу людей, оскільки кожне завдання може бути представлено як проект у будь-якій галузі.

Розроблене програмне забезпечення можна використовувати в навчальних закладах, на підприємствах великого та малого бізнесу, некомерційних організаціях тощо.

Наприклад, у технічних закладах це програмне забезпечення можна використовувати для відстеження процесу розробки студентських курсових проектів.

Таким чином, дане програмне забезпечення може бути використано під час навчального процесу учнів та студентів у різних навчальних закладах, але сфера його застосування може бути дещо ширшою у навчальних закладах розроблене програмне забезпечення можна використовувати при викладанні дисциплін, пов’язаних з веб-розробкою.

Оскільки кінцевим продуктом є система управління сайтом для управління командними проектами, студенти мають можливість ознайомитися зі структурою, методами та технологіями, які будуть використовуватися при розробці цього сайту. Оскільки програмне забезпечення буде відкритим, його можна використовувати як основу для більших проектів тощо.

Отже, виходячи з вищевикладеного, дослідження та програмна реалізація системи аналізу використання ООБД у сучасних ІС є актуальним завданням, яке потребує вирішення у даній магістерській роботі.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1. Огляд існуючих систем

Більшість інформаційних систем сьогодні зберігають дані в сховищах, які називаються базами даних. Найпопулярніші бази даних - Microsoft SQL Server, Oracle, PostgreSQL або MySQL - використовують реляційну модель представлення даних.

У цих базах даних сутності представлені у вигляді рядків таблиці, їх параметри - у вигляді стовпців, а зв'язок між сутностями різних типів здійснюється за допомогою відношення «один до одного» або «один до багатьох».

Типи даних, які можуть мати стовпці, обмежені основними типами (ціле, подвійне, varchar, текст).

Системи керування базами даних зазвичай не дозволяють розширити систему типів шляхом додавання власного типу даних.

Реляційні бази даних мають власний математичний апарат. Для реляційних баз даних свого часу Едгар Код заклав основи математичного апарату реляційної алгебри. Цей математичний інструмент пояснює, як виконувати основні операції зі зв'язками бази даних.

У 1986 році був прийнятий перший стандарт SQL-86, який визначив всю долю реляційних баз даних. Після прийняття стандарту всі розробники реляційних СУБД були зобов'язані йому слідувати. Так, реляційні бази даних зазвичай використовують SQL як мову запитів.

Реляційні бази даних дуже успішні на ринку, і було написано багато програм, які використовують ці бази даних як сховище даних. У багато з цих продуктів вкладено багато грошей, і клієнти продовжуватимуть інвестувати в їх розвиток.

Але традиційні бази даних не завжди є ідеальними для зберігання даних. Так, інколи важко використовувати реляційні бази даних в інформаційних системах зі складною об’єктно-орієнтованою архітектурою. Програма може мати, наприклад, типи даних для великих і неструктурованих мультимедійних об’єктів, географічних інформаційних систем або систем автоматизованого проектування/виробництва. Складність полягає в тому, що, по-перше, необхідно розробити не тільки архітектуру класів програми, але і схему бази даних. По-друге, також необхідно приділити час розробці функціональності, яка зможе відображати табличні дані у вигляді об’єктів класу і навпаки.

Дані, що зберігаються в базі даних, мають визначену логічну структуру - іншими словами, вони описуються деякою моделлю представлення даних (моделлю даних), що підтримується СУБД.

Класичні моделі даних включають наступний підхід:

- ієрархічний;
- мережевий;
- реляційний.

Крім того, в останні роки з'явилися і почали більш активно застосовуватися на практиці такі моделі даних:

- постреляційна БД;
- багатовимірна;
- об'єктно-орієнтовані.

На основі інших моделей даних також розробляються різні системи, які розширюють відомі моделі.

Вони включають об’єктно-реляційні, об’єктно-орієнтовані, семантичні, концептуальні та документно об’єктно-орієнтовані моделі. Деякі з цих моделей служать для інтеграції баз даних, баз знань і мов програмування. Деякі СУБД підтримують декілька моделей даних одночасно. Наприклад, в системі «Інтербаза» для додатків використовується мережева мова маніпулювання даними, а в інтерфейсі користувача реалізовані мови SQL і KBE.

Об'єктно-орієнтована модель

В об'єктно-орієнтованій моделі при представленні даних можлива ідентифікація окремих записів бази даних.

Зв'язки встановлюються між записами бази даних і функціями їх обробки за допомогою механізмів, подібних до відповідних засобів в мовах об'єктно-орієнтованого програмування.

Стандартизована об'єктно-орієнтована модель описана в рекомендаціях стандарту ODMG-93 (Object Database Management Group).

Повністю реалізувати рекомендації ODMG-93 поки не вдається. Щоб проілюструвати ключові ідеї, розглянемо трохи спрощену модель об'єктно-орієнтованої бази даних.

Структуру об'єктно-орієнтованої бази даних (ООБД) можна графічно представити у вигляді дерева, вузлами якого є об'єкти. Властивості об'єкта описуються деяким стандартним типом (наприклад, рядок) або створеним користувачем типом (визначеним як клас).

Значенням властивості string є рядок символів. Значенням властивості класу є об’єкт, який є екземпляром відповідного класу. Кожен об'єкт-примірник класу вважається нащадком об'єкта, в якому він визначений як властивість. Екземпляр об’єкта класу належить до власного класу та має одного батька. Загальні зв’язки в базі даних утворюють узгоджену ієрархію об’єктів.

Приклад логічної структури ООБД «Бібліотека» наведено на рисунку 2.1.

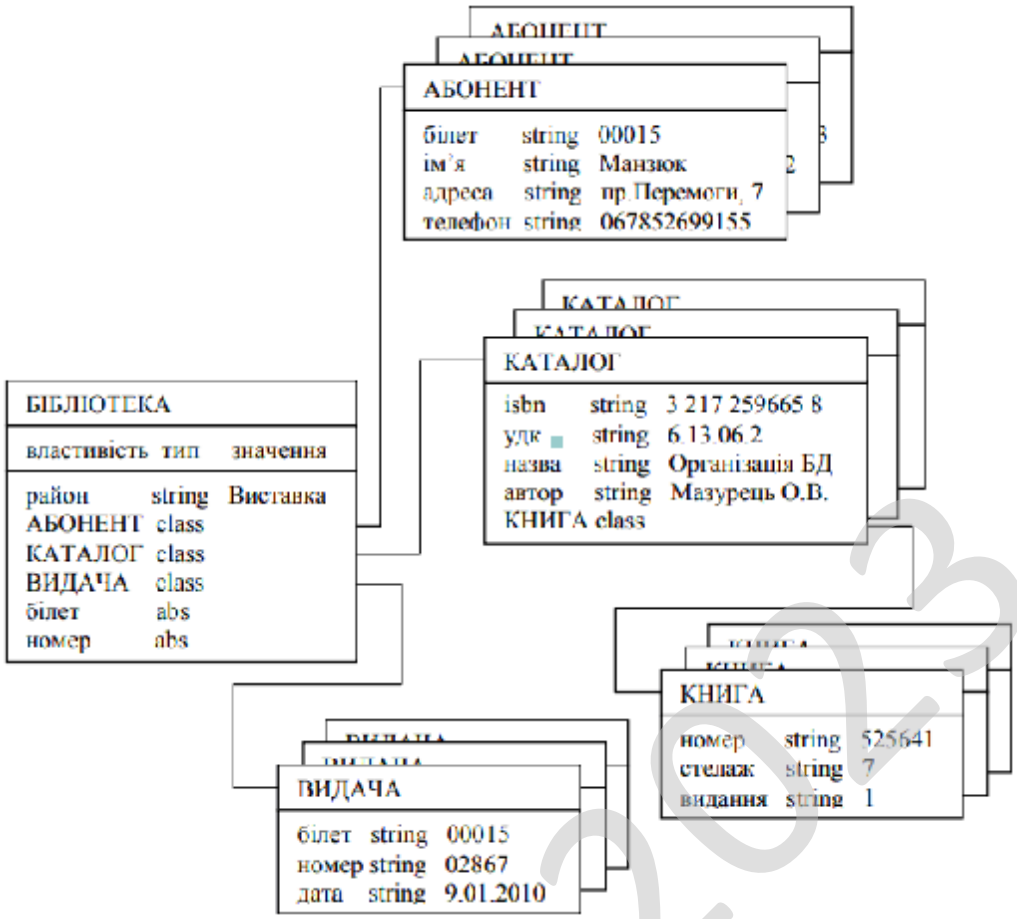

Рисунок 2.1 – Логічна структура ООБД «Бібліотека»

Об'єктно-орієнтовані механізми інкапсуляції, успадкування та поліморфізму. Операції, подібні до команд SQL (наприклад, створення бази даних), можна використовувати обмежено.

Створення та модифікація бази даних супроводжується автоматичним формуванням і подальшою корекцією індексів (індексних таблиць), що містять інформацію для швидкого пошуку даних. Давайте коротко розглянемо концепції інкапсуляції, успадкування та поліморфізму стосовно об’єктно-орієнтованої моделі бази даних.

Інкапсуляція обмежує область дії назви властивості межами об’єкта, в якому вона визначена. Отже, якщо ми додамо властивість, яка містить номер телефону автора книги, до об’єкта типу CATALOG і маємо назву phone, то ми отримаємо однойменні властивості в об’єктах SUBSCRIBER і CATALOG.

Значення такої властивості буде визначатися об'єктом, в якому воно інкапсульоване.

З іншого боку, спадкування розширює сферу власності на всіх нащадків об’єкта. Отже, всім об’єктам типу КНИГА, які є нащадками об’єктів типу КАТАЛОГ, можна присвоїти властивості батьківського об’єкта: isbn, УДК, назву та автора. Якщо необхідно поширити дію механізму успадкування на об’єкти, які не є безпосередніми родичами (наприклад, між двома нащадками одного батька), то в загальному предку визначається абстрактна властивість типу ahs. Таким чином, визначення карти абстрактних властивостей і числа в об'єкті БІБЛІОТЕКА призводить до успадкування цих властивостей усіма підлеглими об'єктами ПЕРЕДПЛАТНЕЦЬ, КНИГА і ВИДАННЯ. Не випадково показане на малюнку значення властивості квитка класів SUBSCRIBER і ISSUER буде однаковим – 00015.

Поліморфізм в об'єктно-орієнтованих мовах програмування означає здатність одного і того ж програмного коду працювати з різними типами даних. Іншими словами, це означає прийняття в об'єктах різних типів мати методи (процедури або функції) з однаковими іменами. Під час виконання об’єктної програми одні й ті ж методи діють на різних об’єктах залежно від типу аргументу. З точки зору нашого OODB, поліморфізм означає, що об’єкти класу BOOK, які мають відмінні від батьків класу CATALOG, можуть мати інший набір властивостей. Тому програми для роботи з об'єктами класу BOOK можуть містити поліморфний код.

Пошук в об'єктно-орієнтованій базі даних полягає в пошуку подібності між об'єктом, указаним користувачем, і об'єктами, що зберігаються в базі даних. Визначений користувачем об’єкт (цільовий об’єкт, властивість об’єкта має тип goat), загалом, може представляти підмножину всієї ієрархії об’єктів, що зберігаються в базі даних. Цільовий об'єкт, а також результат запиту можуть зберігатися в базі даних. Приклад запиту на зчитування номерів білетів та

прізвищ абонентів, які отримали хоча б одну книгу в бібліотеці, наведено на рисунку 2.2.

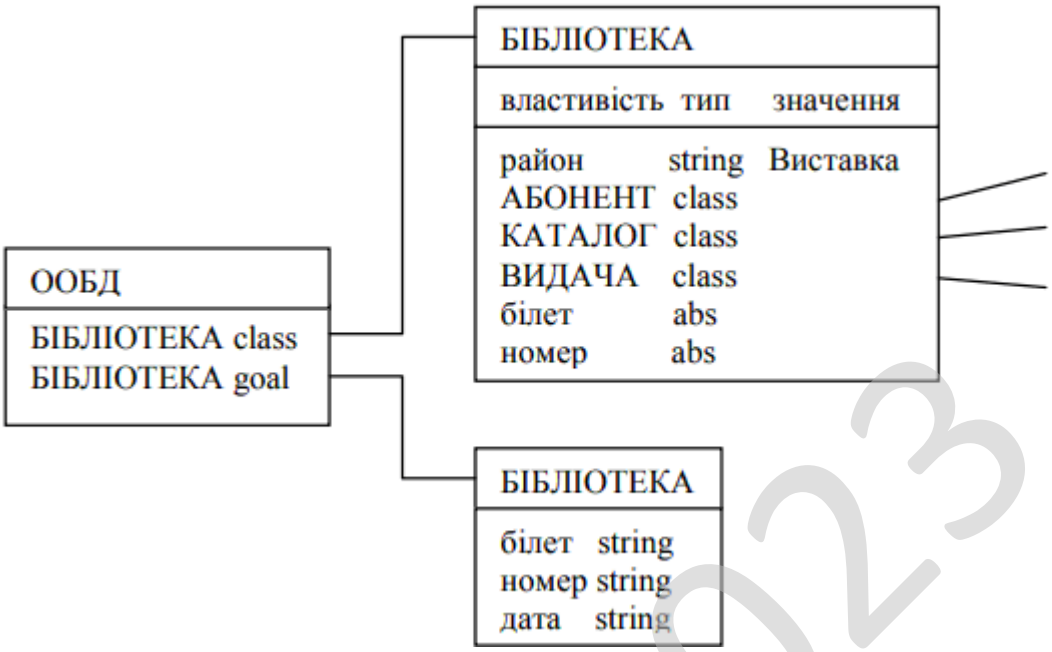

Рисунок 2.2 – Фрагмент ООБД з об’єктом – метою

Основна перевага об’єктно-орієнтованої моделі даних порівняно з реляційною – можливість відображення інформації про складні взаємозв’язки об’єктів. Об'єктно-орієнтована модель даних дозволяє ідентифікувати певний запис бази даних і визначити функції його обробки.

Недоліками об'єктно-орієнтованої моделі є висока концептуальна складність, незручність обробки даних і низька швидкість виконання запитів. У 1990-х роках існували лише експериментальні прототипи об'єктно-орієнтованих систем управління базами даних. На сьогоднішній день такі системи набули широкого поширення, до них зокрема відносяться такі бази даних: ROET (ROET Software), Jasmine (Computer Associates), Versant (Versant Technologies), 02 (Ardent Software), ODB-Jupiter (Науково-виробничий центр «Інтелтек»). Плюс »), а також Iris, Orion і Postgres.

2.2 Обґрунтування вибору методів розробки

Більшість інформаційних систем сьогодні зберігають дані в сховищах, які називаються базами даних. Найпопулярніші бази даних - Microsoft SQL Server, Oracle, PostgreSQL або MySQL - використовують реляційну модель представлення даних. У цих базах даних сутності представлені у вигляді рядків таблиці, їх параметри — у вигляді стовпців, а зв’язок між сутностями різних типів здійснюється за допомогою відношення «один до одного» або «один до багатьох».

Типи даних, які можуть мати стовпці, обмежені основними типами (ціле, подвійне, varchar, текст).

Системи керування базами даних зазвичай не дозволяють розширити систему типів шляхом додавання власного типу даних.

Реляційні бази даних мають власний математичний апарат. Для реляційних баз даних свого часу Едгар Код заклав основи математичного апарату реляційної алгебри. Цей математичний інструмент пояснює, як виконувати основні операції зі зв’язками бази даних.

У 1986 році був прийнятий перший стандарт SQL-86, який визначив всю долю реляційних баз даних. Після прийняття стандарту всі розробники реляційних СУБД були зобов'язані йому слідувати. Так, реляційні бази даних зазвичай використовують SQL як мову запитів.

Реляційні бази даних дуже успішні на ринку, і було написано багато програм, які використовують ці бази даних як сховище даних. У багато з цих продуктів вкладено багато грошей, і клієнти продовжуватимуть інвестувати в їх розвиток.

Але традиційні бази даних не завжди є ідеальними для зберігання даних. Так, іноді важко використовувати реляційні бази даних, де програма може мати, наприклад, типи даних для великих і неструктурованих мультимедійних

об’єктів, географічних інформаційних систем або систем автоматизованого проектування/автоматизованого виробництва.

Складність полягає в тому, що, по-перше, необхідно розробити не тільки архітектуру класів програми, але і схему бази даних. По-друге, також необхідно приділити час розробці функціональності, яка зможе відображати табличні дані у вигляді об’єктів класу і навпаки.

Проаналізувавши результати, можна зробити висновок, що всі бази даних є життєздатними та дають хороші результати, особливо на невеликих обсягах даних.

Концепція ООБД

Об'єктно-орієнтовані бази даних - бази даних, в яких інформація представлена у вигляді об'єктів, як в об'єктно-орієнтованих мовах програмування.

Причиною появи об'єктно-орієнтованих систем баз даних була потреба в більш адекватному представленні та моделюванні об'єктів реального світу, оскільки ООБД забезпечують набагато більш розвинену модель даних, ніж традиційні реляційні бази даних.

Парадигма ООБД базується на низці основних концепцій, таких як об’єкт, ідентифікатор об’єкта, клас, успадкування, перевантаження та пізнє зв’язування. Кожен об'єкт при створенні отримує унікальний ідентифікатор, згенерований системою, який пов'язаний з об'єктом протягом його існування і не змінюється при зміні стану об'єкта. Кожен об'єкт має стан і поведінку:

1) Стан об'єкта - набір значень його атрибутів.

2) Значення атрибута об'єкта - це теж певний об'єкт або безліч об'єктів.

3) Поведінка об'єкта - набір методів (програмний код), що оперують над станом об'єкта.

Об'єктно-орієнтовані бази даних зазвичай рекомендуються для тих випадків, коли потрібна високопродуктивна обробка даних зі складною структурою.

Основні труднощі об'єктно-орієнтованого моделювання даних є наслідком того, що настільки розвиненого математичного апарату, на якому могла б базуватися загальна об'єктно-орієнтована модель даних, не існує.

Розгляд характеристик ООБД привів до визначення ООБД, яке було представлено на Першій міжнародній конференції з дедуктивних та об’єктно-орієнтованих баз даних у формі маніфесту в 1989 році. Він представляє можливості об'єктно-орієнтованих баз даних.

Можливості ООБД:

Підтримка складних об'єктів

Механізм складеного об’єкта дозволяє об’єкту мати атрибут, який також може бути об’єктом. Атрибутами можуть бути рядки, списки, хеш-таблиці тощо.

Ідентифікація об'єктів

Кожна сутність у базі даних має унікальний OID. Це властивість об’єкта, яка відрізняє його від інших об’єктів і існує протягом усього життєвого циклу об’єкта. Ідентифікатор об'єкта повинен бути незалежним від значень його атрибутів.

Інкапсуляція

Об'єктно-орієнтована модель представлення даних передбачає інкапсуляцію та приховування інформації. Це означає, що стан об'єкта можна приховати, щоб запобігти доступу до внутрішньої логіки роботи об'єкта.

Підтримка структур і класів

В ООП структура поєднує загальні характеристики набору сутностей. Наприклад, векторна структура поєднує координати X, I та Z.

Концепція класу схожа на структуру, але більше стосується розподілу пам’яті для об’єктів класу під час виконання програми.

Імітація та поліморфізм

Імітація є однією з основних концепцій об'єктно-орієнтованого програмування. Це пояснюється тим, що клас може мати дочірній клас, який має ті самі властивості, що й базовий клас, але також має нові властивості.

Метод базового класу можна замінити в класі-нащадку. Ця властивість називається поліморфізмом.

Обчислювальна повнота

SQL не має всіх функцій звичайних мов програмування. Такі мови, як Pascal, C, C# або Java, можна назвати обчислювально повними, оскільки вони використовують усі обчислювальні можливості комп'ютера. SKL можна назвати реляційно повним, оскільки в ньому реалізована вся логіка реляційної алгебри. Так, будь-який код, написаний на SQL, можна переписати на C++, але не весь код на C++ можна переписати на SQL.

Тому більшість програм, які використовують реляційні бази даних, передбачають використання вбудованих запитів SQL у звичайній мові програмування.

Це означає, що мова обробки даних має бути мовою програмування загального призначення.

Паралелізм

Відновлення

Спеціальна система запитів

Реляційні бази даних використовують SQL як мови запитів.

Засіб для відправки запитів на базу даних повинно бути і в об'єктно-орієнтованих баз даних. База даних повинна забезпечувати високорівневе, незалежне від додатка засіб для відправлення запитів. Це не обов'язково може бути мова запитів типу SQL.

Можливість додавання нових типів даних

Набір типів даних повинен бути розширюваним. Користувач повинен мати засоби створення нових типів даних на основі набору визначених системних типів. Більш того, між способами використання системних і призначених для користувача типів даних не повинно бути ніяких відмінностей.

Недоліки використання об'єктно-орієнтованих баз даних перед реляційними БД:

- відсутність можливості оптимізації запитів. Однією з найбільш значущих проблем в ООБД є оптимізація декларативних запитів. Оптимізація запитів за ООБД ускладнюється додатковою складністю самої об'єктно-орієнтованої моделі даних;

- відсутність стандартної алгебри запитів. Ця обставина також ускладнює оптимізацію вимог;

- питання безпеки. RDB підтримує авторизацію, тоді як більшість OOBD – ні. RDB дозволяють користувачам передавати та скасовувати права на читання або зміну. ООБД зможуть отримати більш широке поширення у сфері бізнесу лише за умови вдосконалення цієї функції в них;

- обмежені налаштування продуктивності. Більшість ООБД мають лише обмежені можливості налаштування продуктивності. У RBD інсталяторам надається можливість налаштовувати продуктивність системи шляхом встановлення великої кількості параметрів, встановлених системним адміністратором;

- недостатня підтримка складних об'єктів. Повна функціональність складних об'єктів ще не підтримується. Можна переміщатися по посиланнях і кодувати операції за допомогою цих посилань, але немає визначених загальних операцій, які використовують різні типи семантики посилань;

- обмежена інтеграція з системами об'єктно-орієнтованого програмування. Конфлікти імен; необхідність перегляду класової ієрархії; Схильність ООБД перевантажувати системні операції.

Як бачите, ООБД має дуже цікаву концепцію. Маніпулювання даними у вигляді об'єктів дозволяє зберігати дані зі складною структурою і не витрачати зусилля на вирішення такої проблеми, як неузгодженість імпедансів. Це дає можливість отримати досить великий виграш у продуктивності інформаційної

системи. Розглянуті приклади демонструють легкість виконання запитів порівняно з реляційними базами даних.

Однак варто зауважити, що ООБД ще не ідеальні, і на ринку є серйозні конкуренти, куди вкладаються великі гроші.

Для розробки програмного забезпечення системи управління проектами були обрані наступні засоби та мови програмування:

HTML — це мова гіпертекстової розмітки. Документ HTML обробляється в браузері і відображається на екрані у звичному для людини вигляді.

HTML у магістерській роботі буде використовуватися для створення фреймворків веб-сторінок, до яких будуть підключені різні типи баз даних і виконуватиметься тестування швидкості системи залежно від вибраних баз даних.

CSS і CSS3 анімації. CSS - каскадні стилі найчастіше використовуються для візуального представлення сторінок, написаних на HTML. Анімація CSS3 дозволяє анімувати більшість елементів HTML без використання технологій JavaScript або Flash.

CSS у цій статті буде використано для створення приємного інтерфейсу користувача з поєднанням анімації CSS3.

PHP - це мова програмування сценаріїв, призначена для створення HTML - сторінок на стороні веб-сервера. Веб-сервер інтерпретує його в код HTML, який передається на сторону клієнта. На відміну від скриптової мови JavaScript, користувач не бачить PHP-код, тому що браузер отримує готовий HTML-код. Це перевага з точки зору безпеки.

У цьому документі PHP буде використано для створення HTML-сторінок з даними, які будуть отримані з бази даних (БД) за допомогою класу PDO php.

Бази даних - об'єктно-орієнтовані бази даних будуть перевірені на використання як сховища даних. І для порівняння з класичними реляційними базами даних.

Об'єктно-орієнтовані бази даних - Cache, db4object, Prest, Volante, VelocitiDb.

Реляційні бази даних - Microsoft SQL Server, PostgreSQL.

2.3 Розгорнута постановка завдання

Відповідно до технічного завдання магістерської роботи буде реалізовано програмне забезпечення, що відображає роботу ІС з концепцією ООБД та SQL для порівняння продуктивності.

У процесі розробки магістерської роботи необхідно виконати наступний обсяг робіт:

а) проаналізувати існуючі аналогові системи для виявлення їх позитивних і негативних характеристик. Результати аналізу будуть враховані при подальшій розробці;

б) вибрати та обґрунтувати спосіб побудови системи контролю за роботою технічного обладнання виробництва в автоматизованому режимі. Розробляти системні функціональні та структурні системи;

в) розробити системне програмне забезпечення, що забезпечує реалізацію поставленого технічним завданням завдання. Побудувати блок-схеми програмних алгоритмів і підпрограм;

г) організувати інтерфейс користувача для створення та відображення повідомлень про некоректні дії користувача та нетипові ситуації;

д) розробити рекомендації щодо організаційно-методичних заходів щодо забезпечення впровадження;

д) виконати розрахунки для визначення економічної ефективності розробленої системи;

ж) розробити заходи з охорони праці під час впровадження та експлуатації системи та розробити заходи з охорони праці;

з) зробити висновки про обсяги виконаної роботи та досягнуті результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

Для тестування використання різних типів СУБД пропонується реалізувати додаток, який відповідає за аналіз бази користувачів соціальних мереж.

Для розробки програми для аналізу соціальної бази користувачів мережі доцільно використовувати ІТ. Кілька наукових методів, перерахованих нижче, будуть використані для вирішення процесу розробки програми.

Методи аналізу та синтезу були використані для створення моделі даних, яка міститиме такі дані: нові підписники, існуючі підписники користувачів, які скасували підписку. Порівняння даних з наявними користувачами в базі даних.

3.1 Опис функціонування системи

Виходячи з теми магістерської роботи необхідно розробити програмне забезпечення системи управління сайтом та підключення баз даних різного типу для аналізу її продуктивності.

Ідея розробленого програмного забезпечення полягає у створенні веб-сайту факультету, зручного у використанні для входу в посібник для вчителя, до якого для перевірки продуктивності будуть підключені та наповнені відповідними даними різні типи баз даних.

Аналіз і вибір бази даних

Ми порівнюємо продуктивність реляційних та об’єктно-орієнтованих баз даних на практиці. Для цього ми розглянемо швидкість і обсяг пам'яті, що використовується під час додавання, читання та повторного читання великих обсягів даних. Крім того, критерієм для порівняння буде розмір бази даних, наповненої даними.

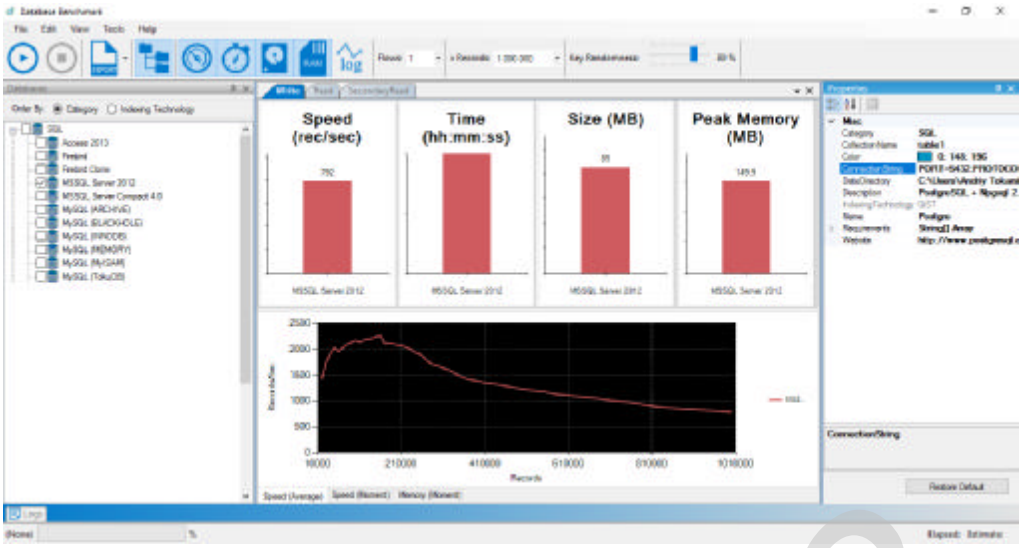

Рисунок 3.1 - Інтерфейс Database Benchmark

Для порівняння будемо використовувати програму Database Benchmark v3.0.0. Ця програма може вимірювати продуктивність дуже великої кількості баз даних, включаючи реляційні та об'єктні бази даних. Database Benchmark дозволяє контролювати процес тестування в реальному часі. Розроблено з використанням .NET Framework + VPF.

Порівняємо такі бази даних як Microsoft SQL Server 2016, Db4Objects, VelocitiDb, Volante, Prest. Усі ці бази даних можна протестувати за допомогою Database Benchmark.

Ми перевіримо швидкість і споживання пам'яті при додаванні та читанні мільйона записів, а також розмір відповідних баз даних.

Таблиця 3.1 - Швидкість операцій, виконуваних базами даних (у записах/сек)

	Додавання	Читання	Повторне читання
Db4Objects	9898	4839	4868
Perst	1894	1175	1194
VelocityDb	145603	30452	426333
Volante	15447	136968	119346
MS SQL Server 2016	792	412031	435002

Таблиця 3.2 - Максимальні затрати оперативної пам’яті (у Мб)

	Додавання	Читання	Повторне читання
Db4Objects	1134.1	1134.2	738.3
Perst	130	122.3	112.1
VelocityDb	256.2	272.5	272.6
Volante	2423.5	2505.7	2498
MS SQL Server 2016	149.9	130.5	130.5

Таблиця 3.3 - Розмір заповнених баз даних (у Мб)

Db4Objects	Perst	VelocityDb	Volante	MS SQL Server 2016
229.9	188.6	53.1	123	85

У таблиці 3.1-3.3 занесені результати вимірювання продуктивності баз даних.

Таблиця 3.1 показує швидкість виконання операцій, 3.2 показує затрати пам’яті на виконання цих операцій, а 3.3 – розміри баз даних.

Результати тестування реляційних баз даних

Для порівняння з об’єктно-орієнтованими базами даних я протестував реляційну базу даних Microsoft SQL Server. Database Benchmark створив таблицю та заповнив її такими даними: число BIGINT як первинний ключ таблиці, два стовпці VARCHAR(255), два стовпці INT, два стовпці REAL і стовпець DATETIME. Таблиця була заповнена мільйоном записів, як показано на рисунку 3.2.

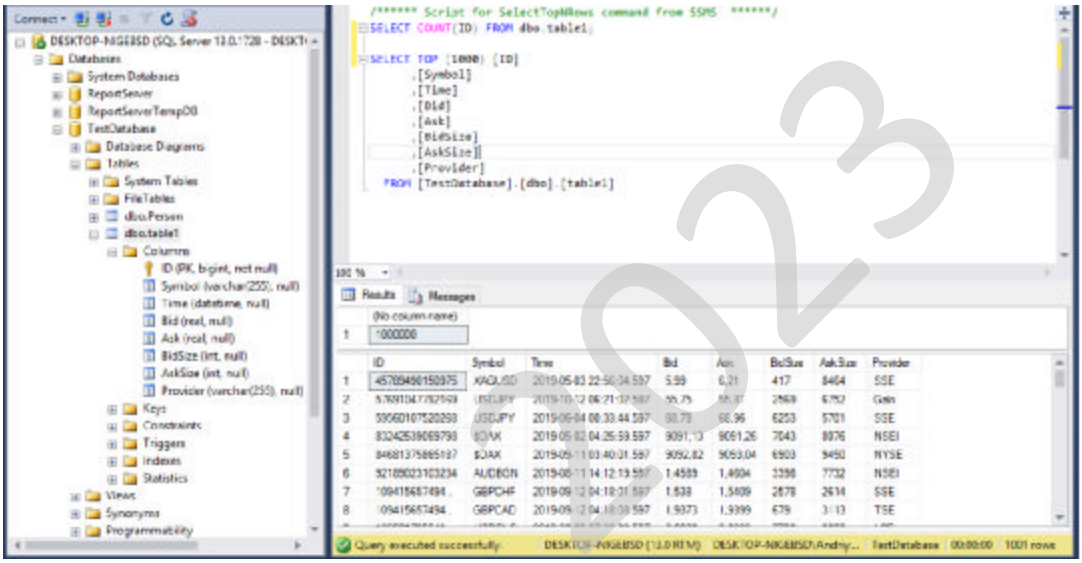

Рисунок 3.2 – Заповнення таблиці для тестування

Із результатів виконання програми Database Benchmark отримуємо, що Microsoft SQL Server є найповільнішою серед протестованих баз даних по швидкості додання нових записів.

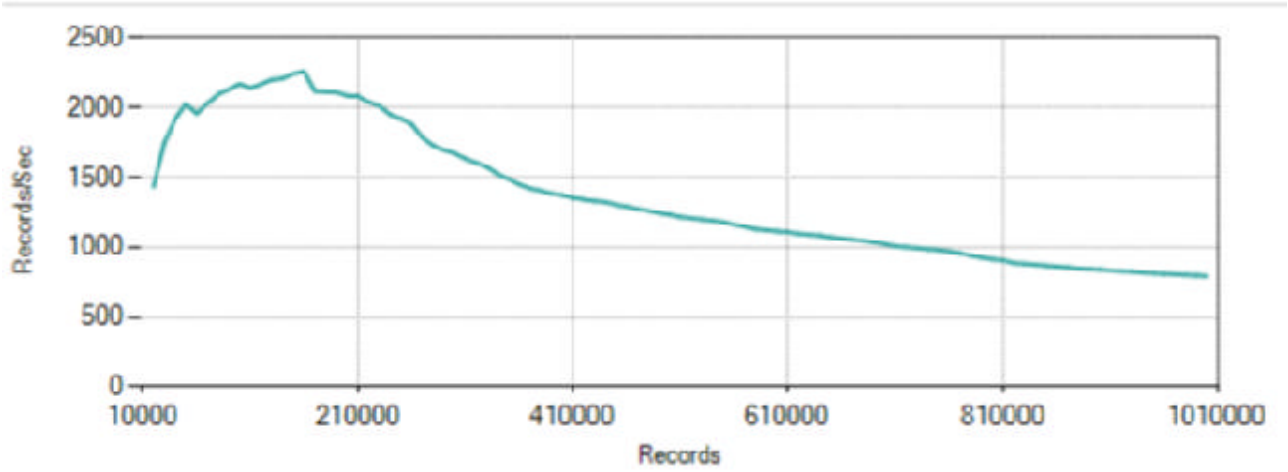

Рисунок 3.3 - Графік залежності середньої швидкості додання нових записів від кількості записів в SQL Server.

На рисунку 3.3 показано, що швидкість додавання нових записів починає швидко знижуватися після того, як кількість записів досягає приблизно 200 000, але навіть у найкращому випадку SQL Server не може конкурувати з більш швидкими базами даних VelocitiDb, db4objects і Volante у цьому відношенні.

Однак розмір бази даних становить лише 85 мегабайт, поступаючись лише VelocityDb.

Але жодна з протестованих об’єктних баз даних не перевершила SQL Server за швидкістю читання даних. Крім того, SQL Server не вимагав великого обсягу оперативної пам'яті. Тільки база даних Prest вимагає менше оперативної пам'яті.

MS SQL Server ідеально підходить для зберігання даних, якщо інформаційна система вимагає швидкого завантаження даних, а нові дані вводяться не часто. Також важливою перевагою SQL Server в порівнянні з іншими базами даних є великий обсяг добре структурованої документації, чого не можна сказати про інші бази даних.

Результат тестування об’єктно-орієнтованої бази даних показано на рисунку 3.4

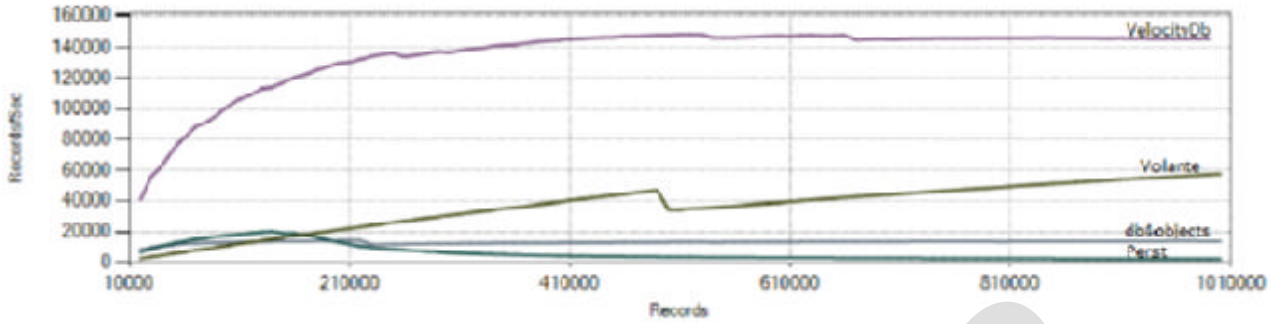

Рисунок 3.4 – Тестування ООБД VelocityDb

Серед об'єктних баз даних, перш за все, слід виділити базу даних, яка явно програла порівняно з іншими за продуктивністю. Ця база даних є Prest. За споживанням оперативної пам'яті він хоч і показує найкращі результати, але за швидкістю вилучення інформації явно програє всім, а за швидкістю додавання інформації виграє лише SQL Server.

Єдиною причиною використання Prest є мінімізація споживання оперативної пам’яті, але MS SQL Server не дуже програє Prest у цьому відношенні. Крім того, Prest не має багатьох очевидних переваг, ніж SQL Server.

База даних VelocitiDb виграє відразу за трьома характеристиками: розміром заповненої бази даних і швидкістю додавання нових записів. Крім того, швидкість перечитування даних близька до значення цієї швидкості в MS SQL Server.

Volante DB перевершує VelocitiDb за швидкістю першого читання запису, але всі операції додавання, читання та повторного читання записів дуже дорогі. Додавання та читання мільйона записів за допомогою Volante коштує приблизно 2,5 ГБ оперативної пам’яті.

Volante та VelocitiDb — це бази даних у пам’яті. Вони вирішують проблему дуже повільного доступу до пам'яті жорсткого диска. База даних у

пам’яті базується на зберіганні даних на пам’яті з довільним доступом. Це робить доступ до даних дуже швидким.

Серед протестованих об’єктних баз даних VelocityDb однозначно найкраща. Його можна використовувати, коли потрібно швидко обробити великий обсяг даних, а також для швидкого читання даних, особливо коли їх потрібно читати кілька разів. Якщо вам потрібно використовувати безкоштовну базу даних об’єктів, DbVolante — хороший варіант.

Опис інформаційної системи

У магістерській роботі реалізовано інформаційну систему, яка використовує як сховище даних об’єктно-орієнтовану базу даних VelocitiDb. Вибір бази даних базується на результатах порівняльного аналізу баз даних.

Інформаційна система в базі даних зберігає інформацію про співробітників і клієнтів, а також інструкції та історію їх перегляду або завантаження. Співробітник може додавати нові книги, а також відзначати, що певний клієнт (студент) користувався (завантажував) вказану літературу.

Також інформаційна система має можливість повідомити користувача про те, що у когось із колег сьогодні день народження, а також є можливість переглянути книги, які переглянули колеги.

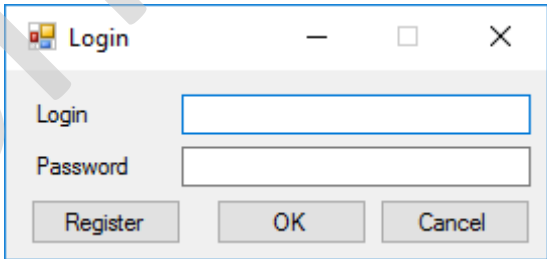

Рисунок 3.5 – Форма входу користувача

Для того, щоб почати користуватися програмою, користувачу необхідно авторизуватися (форма авторизації показана на рисунку 3.6). Якщо у співробітника є обліковий запис, ви можете ввести логін і пароль, в іншому випадку натисніть кнопку Зареєструватися.

Для реєстрації користувача необхідно вказати своє прізвище та ім'я, а також логін і пароль (рисунок. 3.6).

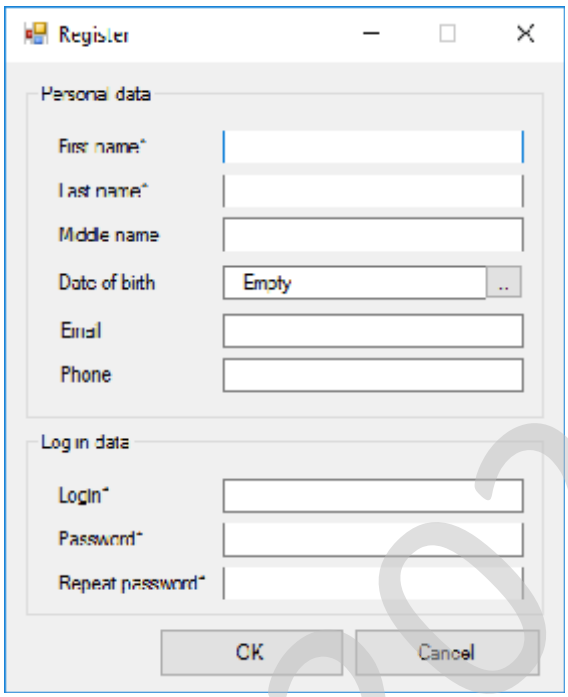

Рисунок 3.6 – форма реєстрації

Після авторизації програма перенаправляє користувача на вікно із головним меню програми, зображеним на рисунку 3.7.

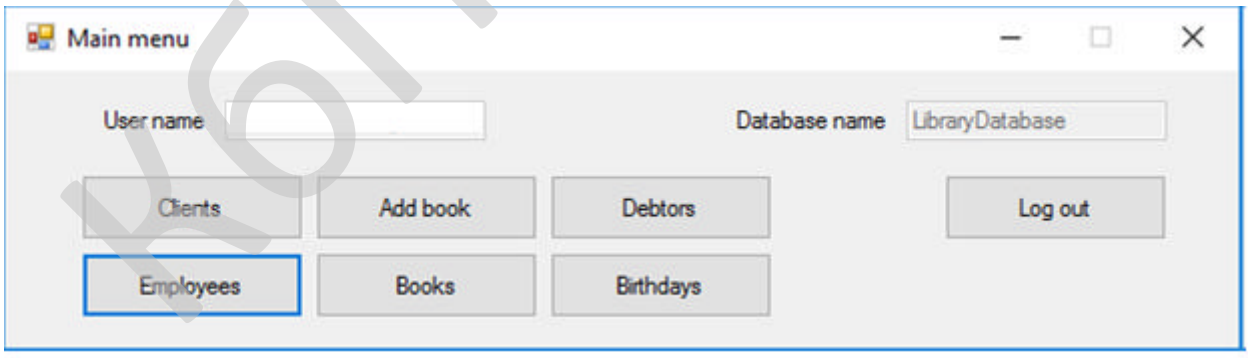

Рисунок 3.7 - Головне меню програми

З меню можна перейти до вікон з клієнтами та співробітниками. Ви також можете перейти до вікна, щоб додати книгу, список книг і список людей, у яких сьогодні день народження.

Щоб переглянути список книг, натисніть кнопку Книги. Відкриється вікно, зображене на рисунку 3.8. Подвійне клацання на рядку з книгою відкриває детальну інформацію про книгу. Тоді відкриється вікно, зображене на рисунку 3.8.

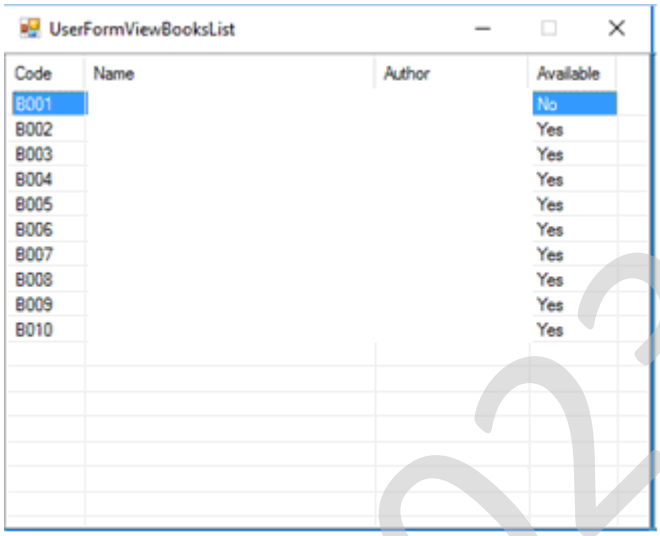

Рисунок 3.8 – Вікно із списком посібників

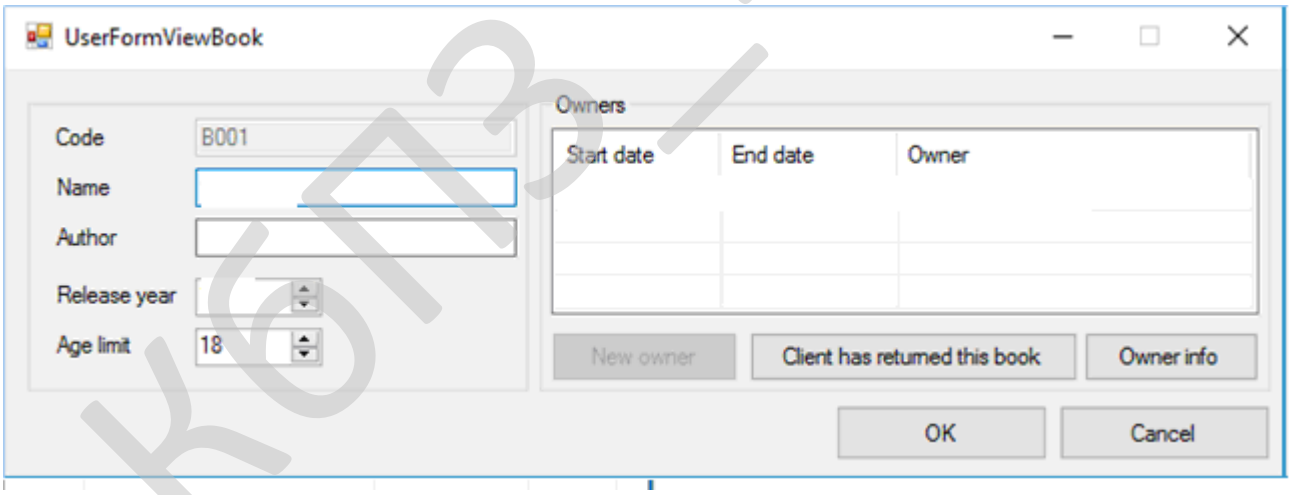

Рисунок 3.9 – Вікно із детальною інформацією про посібники

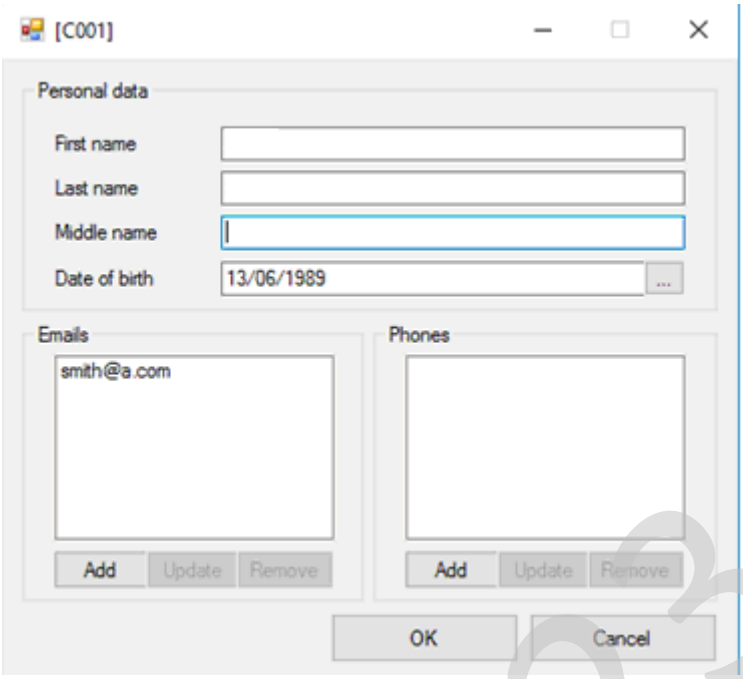

Рисунок 3.10 – Вікно із інформацією про людину

На рисунку 3.10 показано це вікно. Як бачите, особиста інформація людини включає її повне ім’я, дату народження, яка може не вказуватися, а також списки номерів телефонів і адрес електронної пошти. Натисканням кнопки ОК ви підтверджуєте зміни інформації.

Верхні вікна показують структуру даних, що зберігаються в розробленій інформаційній системі.

Серед інформації, що зберігається в книзі, можна виділити її код, назву, ім'я автора, рік видання, вікові обмеження та історію використання. З інформації у вікні, зображеному на малюнку 3.10, можна сказати, що користувач із кодом C001 та іменем завантажив книгу «Відьмак» 13 червня 2019 року. Натискання кнопки Клієнт повернув цю книгу підтверджує, що клієнт повернув книгу. За допомогою кнопки «Новий власник» ви можете вибрати нового користувача книги, і дата початку для нього буде негайно встановлена на сьогоднішній день. В іншому випадку для книг із віковими обмеженнями ви не можете вказати клієнта з невизначеним віком або віком, молодшим за вікове обмеження, як користувача.

На рисунку 3.11 показано UML-діаграму класів, які використовує ця інформаційна система. Клас OptimizedPersistable — це клас, який міститься у вбудованій бібліотеці баз даних VelocitiDb, тобто він створений не мною. Він має дуже велику кількість методів і властивостей, діаграма сну показує лише ті, які мені найбільше потрібні – властивість Id і метод Unpersist.

Id використовується для зберігання ідентифікатора об'єкта, який зберігається в базі даних.

Метод Unpersist призначений для видалення об'єкта з бази даних. Варто зазначити, що цей метод є віртуальним і може бути перевантажений, якщо потрібно, і ви можете написати власну реалізацію очищення, якщо потрібно. Наприклад, у класі Person цей метод перевантажений так, що він спочатку видаляє списки телефонів і адрес електронної пошти, а потім сам об’єкт.

OptimizedPersistable у цій інформаційній системі є базовим класом для класів Person і Book. У свою чергу, класи Emploiee і Client походять від Person.

Клас бази даних є статичним, містить лише один статичний метод, реалізований для спрощення отримання об’єктів із бази даних.

Клас ThisApplication — це клас, який зберігає параметри програми, тобто назву бази даних і авторизованого працівника.

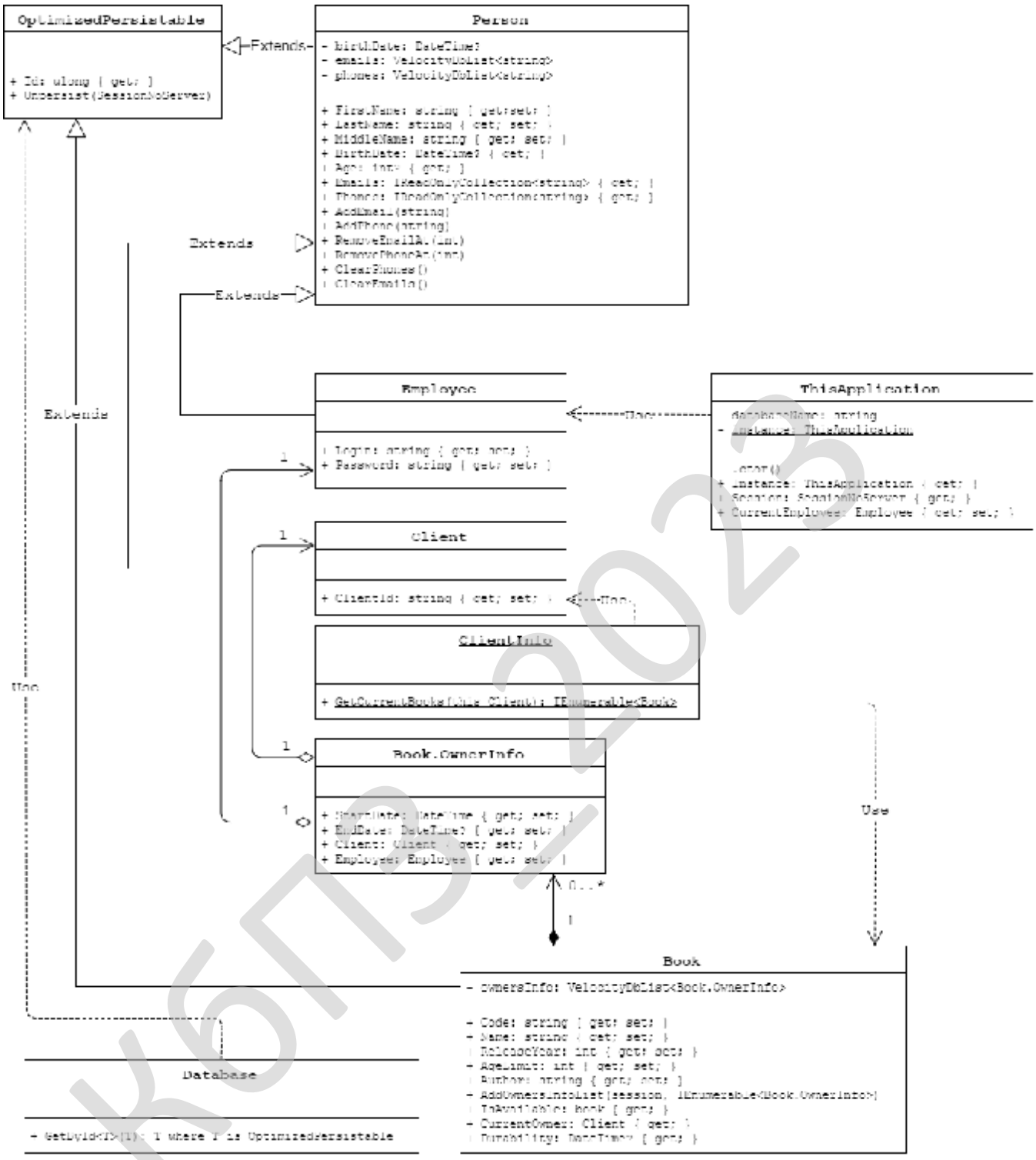

Рисунок 3.11 – UML-діаграма класів інформаційної системи

3.2 Розробка структурної схеми

Структурна схема — це сукупність об’єктів і частин та зв’язків між ними. Призначення діаграми структури — наочно показати складові частини сайту, його основні блоки, вузли та взаємозв’язок між ними.

Структурна схема розробленої системи наведена на рисунку 3.12.

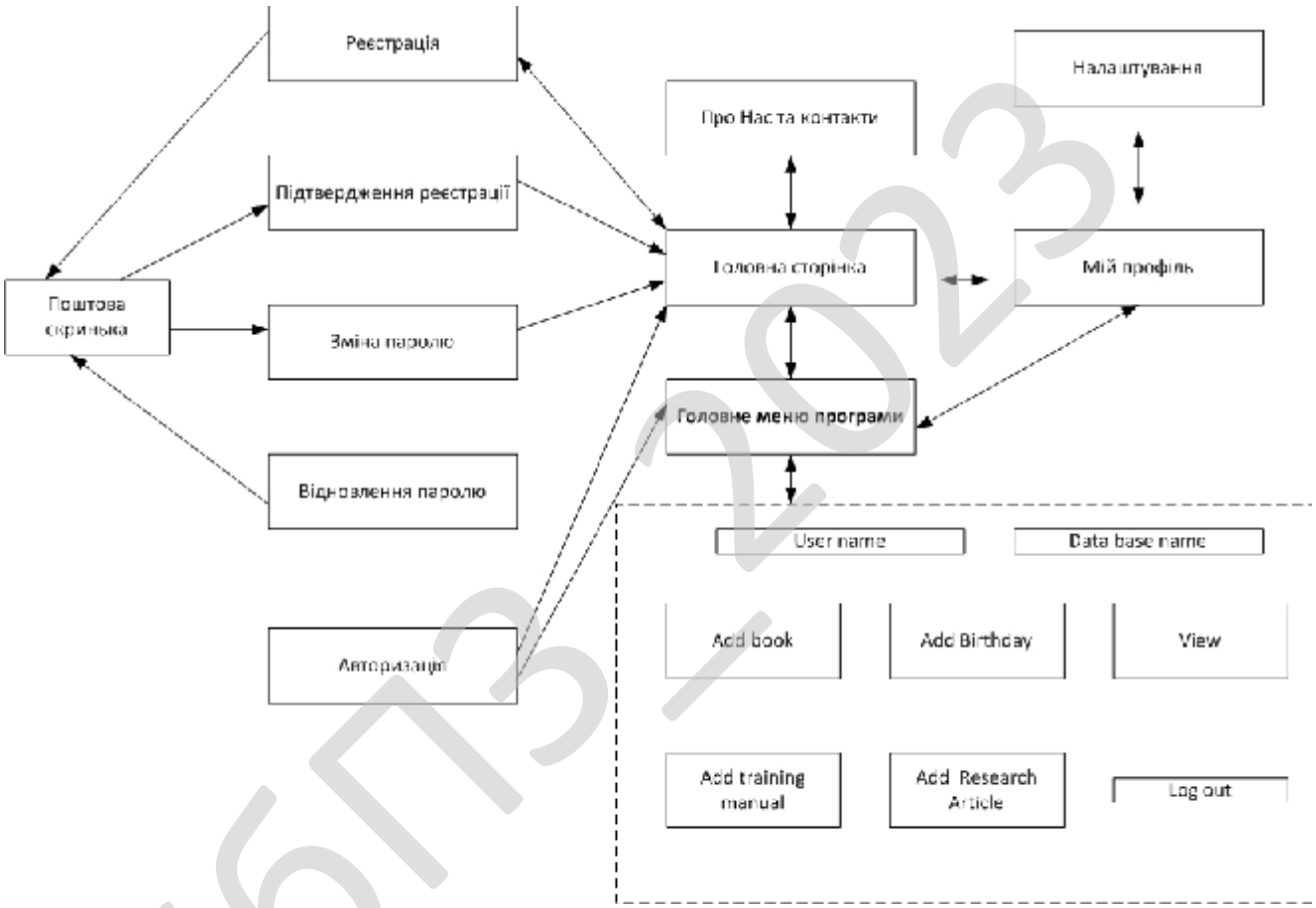

Рисунок 3.12 – Структурна схема

На схемі видно, що сайт має головну сторінку, з якої користувач може перейти на сторінку реєстрації, сторінку авторизації та сторінку панелі адміністрування сайту. Зі сторінки реєстрації користувач може перейти на сторінку авторизації.

На сторінці авторизації користувач має можливість перейти на сторінку

відновлення пароля, якщо він забув свій пароль для входу. Ви також можете перейти на сторінку всіх проектів користувачів. Звідси користувач може перейти на сторінку окремого проекту або на сторінку свого профілю. Користувач також може вийти зі свого профілю на цій сторінці.

Інформаційна система в базі даних зберігає дані про співробітників і клієнтів, а також посібники, методичні вказівки та опубліковані наукові статті. Співробітник може додавати нові книги. Кожна з книг зберігає історію їх перегляду співробітниками.

Також інформаційна система здатна повідомити користувача про те, що у когось із колег сьогодні день народження, а також є можливість перегляду інформації, доданої співробітниками.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.13.

З рисунку видно, що розроблена система складається з наступних частин:

– блок головної сторінки;

– блок сторінки профілю користувача;

– блок бази даних сайту;

– блок адміністративної панелі;

– блок головного меню програми.

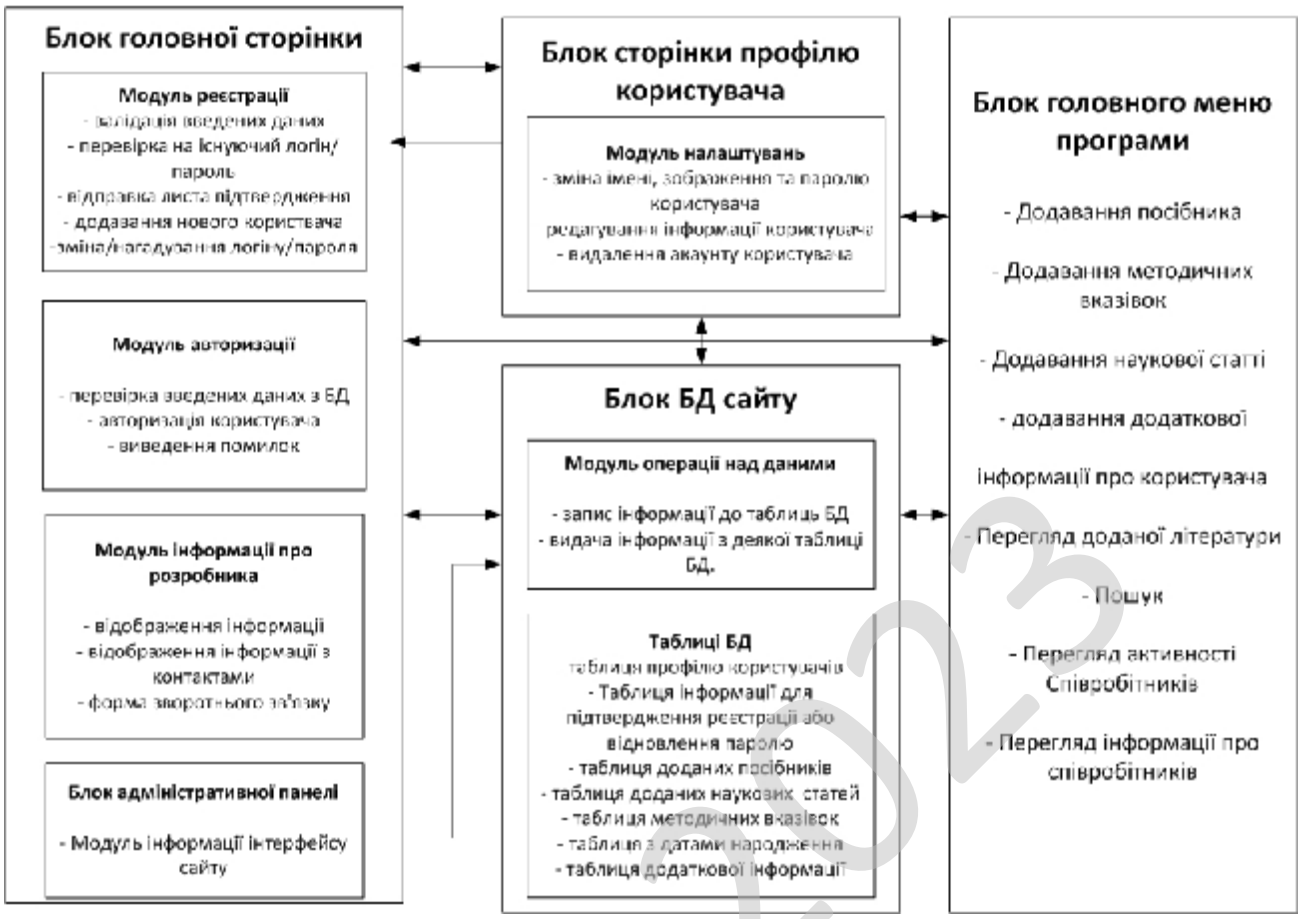

Рисунок 3.13 – функціональна схема

Розглянемо кожен з блоків окремо.

Блок головної сторінки

Цей блок призначений ознайомленню користувача з функціоналом сайту і містить чотири модулі.

Перший модуль - це модуль для реєстрації нових користувачів. Цей модуль забезпечує перевірку введених користувачем даних, тобто звіряє введені дані з дійсною електронною поштою та правилами, визначеними під час реєстрації.

Далі цей модуль перевіряє наявність введених даних у таблиці бази даних користувача.

Якщо користувач із таким іменем або електронною поштою вже зареєстрований, відображається повідомлення про помилку, інакше система надсилає лист на адресу, надану користувачем, для підтвердження реєстрації.

При натисканні на посилання в листі система автоматично додає нового користувача в базу.

Цей модуль також дозволяє змінити або нагадати пароль користувача на вимогу.

Другий модуль — модуль авторизації користувача, призначений для перевірки введених користувачем даних, тобто логіна та пароля, з подальшою перевіркою їх у таблиці користувача бази даних. Якщо логін і пароль збігаються з будь-яким рядком таблиці, користувач авторизується і перенаправляється на сторінку проектів користувача. Якщо відповідності не знайдено, користувачеві відображається відповідна помилка.

Блок бази даних

У магістерській роботі реалізовано інформаційну систему, яка використовує як сховище даних об’єктно-орієнтовану базу даних VelocitiDb. Вибір бази даних обґрунтовується результатами порівняльного аналізу баз даних, результати якого представлені в попередньому розділі.

VelocitiDb - об'єктно-орієнтована база даних для роботи з платформою .NET.

Функціонал надсилання запитів на читання або обробку інформації реалізований у вигляді методів абстрактного класу SessionBase, спадкоємці якого визначають спосіб підключення до бази даних.

Класи Persistable мають бути нащадками класу OptimizedPersistable, який містить ідентифікатор об’єкта та методи для додавання об’єкта до бази даних і для його видалення. Класи, які не є нащадками OptimizedPersistable, зберігаються в базі даних лише до тих пір, поки вони мають принаймні одне посилання з об’єктів OptimizedPersistable.

Бібліотека також реалізує класи VelocitiDbList<T> і VelocitiDbDictionary<T>, які є аналогами List<T> і Dictionary<T>, але в той же час є нащадками OptimizedPersistable.

Для надсилання запитів зручно використовувати LINK, як показано на рисувнку 3.14. LINK — це SQL-подібний синтаксис мови запитів, вбудований у мови програмування .NET Framework (C#, VB.NET, F# тощо).

```
1.  private void SendQuery()
2.  {
3.      using (var conn = new SessionNoServer("FolderName"))
4.      {
5.          conn.BeginRead();
6.          var query =
7.              from person in conn.AllObjects<Person>()
8.              where person.BirthDate != null
9.              let day = person.BirthDate.Value.Day
10.             let month = person.BirthDate.Value.Month
11.             where day == DateTime.Today.Day
12.                && month == DateTime.Today.Month
13.             orderby person.FirstName, person.LastName
14.             group person by person.GetType();
15.
16.         foreach (var personGroup in query)
17.         {
18.             Console.WriteLine(personGroup.Key == typeof(Employee)
19.                 ? "Employees" : "Clients");
20.             foreach (var person in personGroup)
21.             {
22.                 Console.WriteLine("\t{0}", person);
23.             }
24.         }
25.
26.         conn.Commit();
27.     }
28. }
```

Рисунок 3.14– Приклад запиту до бази даних VelocityDb із використанням LINQ

Розглянемо приклад запиту. Зробіть людей одночасно співробітниками та клієнтами в базі даних компанії. Є база даних людей. Необхідно відобразити всіх людей, у яких сьогодні день народження, але згрупувати клієнтів і співробітників окремо.

На рисунку 3.15 показано рішення цієї проблеми.

Функції BeginRead() і Commit() визначають межі транзакцій. Функція AllObjects<T>() отримує об’єкти типу T. СУБД VelocitiDb має два типи транзакцій — Read і Update, які оголошуються викликом методів BeginRead() і BeginUpdate() відповідно, і завершуються викликом методу Commit().

Транзакції зчитування дозволяють лише зчитувати інформацію та не дозволяють змінювати дані. Це зроблено для того, щоб кілька транзакцій Read

могли безболісно використовуватися одночасно, що значно прискорює вилучення даних, коли одночасно працюють кілька клієнтів.

Блок головного меню програми

Щоб почати користуватися програмою, користувачу необхідно буде авторизуватися. Якщо у співробітника є обліковий запис, ви можете ввести логін і пароль, в іншому випадку натисніть кнопку Зареєструватися.

Для реєстрації користувача необхідно ввести ім'я та прізвище, а також логін і пароль.

Висновок

Впроваджено інформаційну систему, яка використовує базу даних VelocitiDb як сховище даних. Розроблено з використанням .NET Framework + WinForms. База даних наповнюється тестовими даними про книги, клієнтів і співробітників за допомогою інформаційної системи. Заявки обробляються досить швидко.

Слід зазначити, що архітектура OOBD хоч і повністю повторює архітектуру класів, але для використання OOBD необхідні зміни в архітектурі класів. Так, наприклад, необхідно зробити збережені класи даних нащадками класу OptimizedPersistable.

Крім того, об'єктні бази даних незручно використовувати для передачі даних в іншу об'єктну базу даних через необхідність знову змінювати архітектуру.

3.4 Розробка діаграми процесів

Діаграму процесів зображено на рисунку 3.15.

Перейшовши на головну сторінку, користувач може увійти або зареєструватися, якщо у нього ще немає облікового запису. Він також може скинути свій пароль, перейшовши на сторінку скидання пароля з форми авторизації.

Після авторизації користувач може перейти на свою сторінку. На цій сторінці користувач може переглянути важливі проекти, нещодавно відкриті проекти, шукати проекти за назвою або створити новий проект. Також користувач має можливість переглянути свої повідомлення, зайти в свій профіль або перейти на сторінку спеціального проекту.

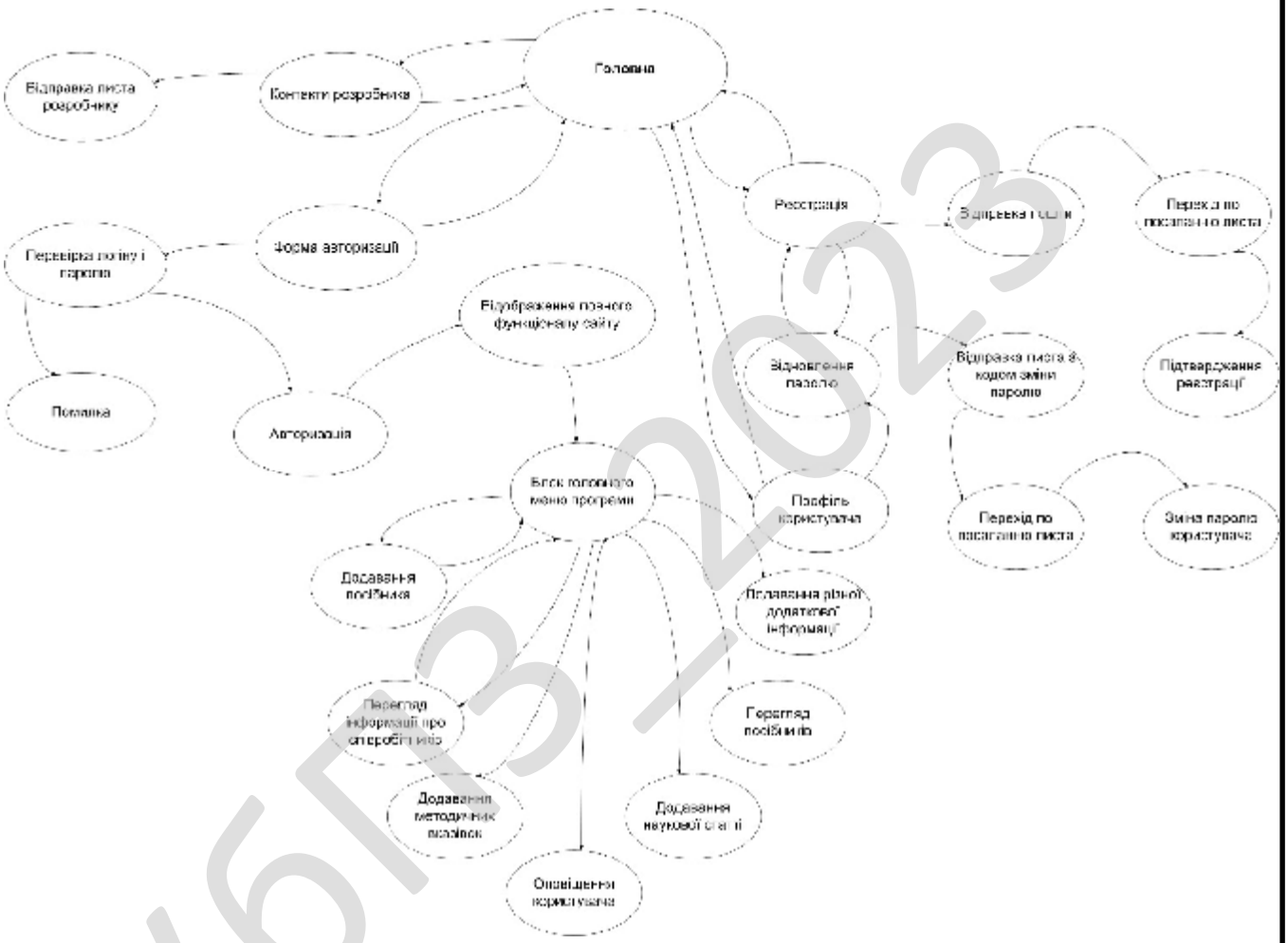

Рисунок 3.15 – Діаграма процесів

Отже, після розгляду опису системи, структурно-функціональної схеми системи та діаграми взаємодії процесів, переходимо до опису блок-схем основної програми та підпрограм, які використовуються для реалізації системи.

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

4.1 Розробка блок-схем та опис алгоритмів функціонування системи

Розглянемо алгоритм основної програми. Його структурна схема показана на рисунку 4.1.

Як тільки користувач потрапляє на головну сторінку сайту, він повинен пройти авторизацію або реєстрацію, якщо у нього немає облікового запису. На рисунку 4.1 показана блок-схема алгоритму сторінки реєстрації користувача.

На сторінці реєстрації користувач має можливість зареєструвати новий обліковий запис для доступу до закритої частини сайту. Для успішної реєстрації користувач повинен вказати свою фотографію, ім'я, поштову адресу та бажаний пароль. Після введення всіх даних система перевіряє, чи всі поля заповнені та чи присутній завантажене зображення, і або виводить помилку, якщо щось пропущено, або створює два записи в таблицях бази даних. Перший запис містить дані, введені користувачем про його обліковий запис, а другий запис містить дані, необхідні для активації цього облікового запису, які зберігаються в таблиці з інформацією про облікові записи, які очікують на активацію. Потім система надсилає на введену користувачем адресу електронної пошти лист із посиланням, що містить код активації облікового запису, і повідомляє користувача, що лист із кодом активації надіслано на вказану ним адресу.

Після переходу користувача за отриманим посиланням система перевіряє інформацію за цим посиланням, тобто код активації облікового запису, і якщо цей код збігається з кодом у рядку таблиці з неактивними обліковими записами, система видаляє інформацію про неактивні облікові записи користувачів. обліковий запис і переконайтеся, що цей обліковий запис активовано

Потім система перенаправляє користувача на сторінку авторизації, де він може увійти в свій обліковий запис і почати роботу.

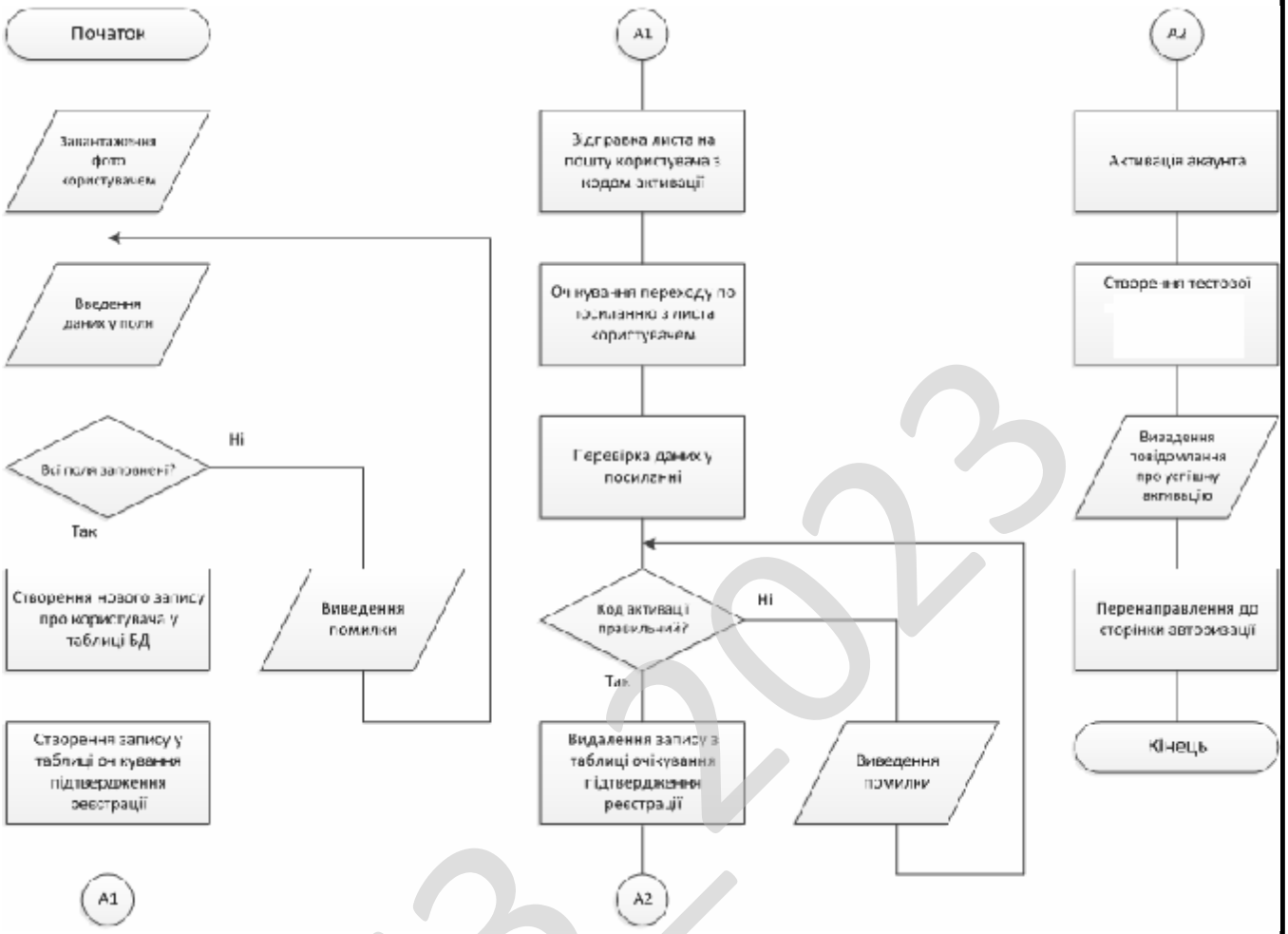

Рисунок 4.1 – Блок-схема алгоритму роботи сторінки реєстрації

Наступний блок коду відповідає за реалізацію вищенаведеного алгоритму:

```
// Обробник кнопки завантаження зображення
$('.btn-file :file').on('fileselect', function(event, numFiles, label) {
    var _validFileExtensions = [".jpg", ".jpeg", ".png"];
    var sFileName = $("#photo_field").val();
    // Перевірка розширення зображення
    if (sFileName.length > 0) {
        var blnValid = false;
        for (var j = 0; j < _validFileExtensions.length; j++) {
            var sCurExtension = _validFileExtensions[j];
            if (sFileName.substr(sFileName.length - sCurExtension.length,
sCurExtension.length).toLowerCase() == sCurExtension.toLowerCase()) {
```

```
                        blnValid = true;
                        break;
                    }
                }
                $("#filename").val(label);
                // Виведення помилок, якщо файл не є зображенням
                if (!blnValid) {
                   $("#photo-label").css({"display": "inline-block"});
                   $(".photo_message").html("Файл не є зображенням!");
                   $('#logo-image').attr('src', "assets/img/logo_short.png");
                   can_register.photo = false;
                } else {
                   $("#photo-label").css({"display": "block"});
                   $(".photo_message").html("");
                   $(".error_notification").css({"height": "0px", "background-
color": "#fff", "border-left": "0px solid #fff"});
                        $(".notification_message").html("");
                   readURL(this);
                }
            }
        });
        // Допоміжний обробник кнопки завантаження зображення
        $(document).on('change', '.btn-file :file', function() {
             fileinput = this,
             input = $(this),
             numFiles = input.get(0).files ? input.get(0).files.length : 1,
             label = input.val().replace(/\\/g, '/').replace(/.*\//, '');
             input.trigger('fileselect', [numFiles, label]);
        });
        // Обробник натиснення клавіші у полі логіну
        $("#login_field").keyup(function(event){
             // Перевірка введеного значення
             var login = $("#login_field").val();
             if(login.length < 4){
                      $(".login_message").html("Занадто коротке ім'я!");
                 can_register.login = false;
                 return false;
             }
             if(login.length > 15){
                 $(".login_message").html("Занадто довге ім'я!");
                 can_register.login = false;
                 return false;
```

```
        }
        if(!/^[a-zA-Z0-9_-]{3,15}$/.test(login)){
            $(".login_message").html("Некоректні символи в імені!");
                  can_register.login = false;
            return false;
        }
        // Перевірка на зайнятий логін
        check('login');
        if(event.keyCode == '13'){ $("#register").click(); }
    });
    // Обробник натиснення клавіші у полі e-mail'у
    $("#email_field").keyup(function(event){
        var email = $("#email_field").val();
        // Перевірка на валідний e-mail
         if(!/^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)*)|(\".+\"))
@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-
9]+\.)+[a-zA-Z]{2,}))$/.test(email)){
            $(".email_message").html("Не дійсний E-mail!");
            can_register.email = false;
            return false;
        }
        // Перевірка на зайнятий e-mail
        check('email');
        if(event.keyCode == '13'){ $("#register").click(); }
    });
    // Обробник натиснення клавіші у полі паролю
    $("#password_field").keyup(function(event){
        var pass = $("#password_field").val();

        if(pass.length < 6){
            $(".password_message").html("Занадто короткий пароль!");
            can_register.pass = false;
        } else {
            $(".password_message").html('');
            can_register.pass = true;
        }
         if(event.keyCode == '13'){ $("#register").click(); }
    });
    // Функція перевірки на зайняті значення у таблиці БД
    function check(field){
        var value = $("#" + field + "_field").val(),
            errors = {login: "Логін вже зайнятий!", email: "E-mail вже
```

```
зайнятий!"};
            $.ajax({
               type: "POST",
               url: "ajax/register.php",
               data: {
                  check : true,
                  field : field,
                  value : value
               }
            }).done(function(msg){
               if(msg == 'false'){
                  $("." + field + "_message").html(errors[field]);
                  can_register[field] = false;
               } else {
                  $("." + field + "_message").html('');
                  can_register[field] = true;
               }
            });
      }
      // Функція валідації всіх полів при відправці форми
      function validate() {
           var login = $("#login_field").val(),
               email = $("#email_field").val(),
               pass = $("#password_field").val(),
                      photo = $("#photo_field").val();
           if(login == '' || email == '' || pass == '' || photo == '') {
               showError("Заповніть порожні поля!", false);
               return false;
           }
           if(!(can_register.login && can_register.email && can_register.pass &&
can_register.photo)) {
               showError("Перевірте заповнені поля на правильність!", false);
               return false;
           }
           $(".error_notification").css({"height": "0px"});
           return true;
      }
      // Файл для AJAX-запитів для перевірки на зайнятий логін або e-mail
      $config = include "../config/db.php";
      if(isset($_POST['check'])){
           $field = $_POST['field'];
           $value = $_POST['value'];
```

```
            $col = '';
            if($field == 'login'){ $col = 'username'; } else { $col = 'email'; }
            try{
                  $dbh = new PDO("mysql:host = $config[host]; dbname =
$config[db_name]", $config['username'], $config['password']);
                  $sth = $dbh->query("SELECT $col FROM users");
                  $sth->setFetchMode(PDO::FETCH_ASSOC);
                  while($row = $sth->fetch()){
                      if($row[$col] === $value){
                          echo 'false';
                          exit();
                      }
                  }
                  echo 'true';
            } catch (PDOException $e){
                  echo $e->getMessage();
            }
            $dbh = null;
      }
      // Обробник відправки форми
      if (isset($_POST['signup'])) {
          // Отримуємо введені значення
          $login = $_POST['user_login'];
          $password = password_hash($_POST['user_password'], PASSWORD_DEFAULT);
          $email = $_POST['user_email'];
          // Перевірка зображення наданого користувачем
          $file_exts = array("jpg", "jpeg", "png");
          $temp = explode(".", $_FILES["file"]["name"]);
          $upload_exts = end($temp);
          $file_name = md5(date('m/d/Y h:i:s a', time()) . $_FILES["file"]
["name"]) ."." . $upload_exts;
          if ((($_FILES["file"]["type"] == "image/jpeg")
              || ($_FILES["file"]["type"] == "image/png")
              || ($_FILES["file"]["type"] == "image/pjpeg"))
              && ($_FILES["file"]["size"] < 8000000)
              && in_array($upload_exts, $file_exts)) {
              if ($_FILES["file"]["error"] > 0) {
                  echo "<script> showError(\"Помилка! Код: " .
$_FILES["file"]["error"] . "\", false);</script>";
              } else {
                  if (file_exists("/images/user_avatars/" . $file_name)) {
                      echo "<script> showError(\"Файл з назвою " .
```

```
$_FILES["file"]["name"] . " вже існує.\", false);</script>";
                    } else {
                        move_uploaded_file($_FILES["file"]["tmp_name"],
"images/user_avatars/" . $file_name);
                        try {
                            $dbh = new PDO("mysql: host = $config[host]; dbname =
$config[db_name]", $config['username'], $config['password']);
                            $sth = $dbh->prepare("INSERT INTO users (username,
email, password, active, photo) values ('$login', '$email', '$password', 0,
'$file_name')");
                            $sth->execute();
                            $userid = $dbh->lastInsertId();
                            $key = $login . $email . date('Y-m-d H:i:s');
                            $key = md5($key);
                            $sth = $dbh->prepare("INSERT INTO confirm VALUES
(NULL, '$userid', '$key', '$email', 0, 1)");
                            $sth->execute();
                            require("assets/class.phpmailer.php");
                            $mail = new PHPMailer();
                            $mail->CharSet = 'UTF-8';
                            $mail->From     = "noreply@pandya.esy.es";
                            $mail->FromName = "Pandya";
                            $mail->AddAddress($email);
                            $mail->Subject  = $words->email_subject;
                            $mail->Body     = $words->email_body .
"\n\nhttp://pandya.esy.es/confirm.php?username=" . $login . "&email=" . $email .
"&secure_key=" . $key;
                            echo "<script> showError(\"Протягом хвилини на " .
$email . " буде надіслано листа з кодом підтвердження реєстрації!\", true);
                                $(\"#login_field\").val('');
                                $(\"#email_field\").val('');
                                $(\"#password_field\").val('');
                            </script>";
                        }
                    } catch (PDOException $e){
                        echo $e->getMessage();
                    }
                    $dbh = null;
                }
            }
        } else {
            echo "<script> showError(\"Недопустимий файл.\", false);</script>";
```

На сторінці авторизації користувач має можливість авторизуватися на сайті, ввівши свій логін і пароль, вказані при реєстрації. Блок-схема алгоритму сторінки авторизації показана на рисунку 4.2.

Після введення даних у відповідні поля система перевіряє записи в таблиці бази даних на відповідність логіна та пароля даним, введеним користувачем. Якщо відповідності не знайдено, користувачеві відображається повідомлення про помилку, інакше система перевіряє активацію облікового запису користувача. Якщо обліковий запис не активовано, користувачеві відображається повідомлення про помилку, інакше система зберігає інформацію про користувача в сеансі та створює відповідний запис у «cookie» браузера, який триватиме не більше доби. Це потрібно для того, щоб користувачеві входити не кожен раз, а тільки один раз на день, що дуже зручно.

Користувач зможе вийти зі свого облікового запису в будь-який момент. Також на сторінці авторизації є можливість відновити пароль, якщо він випадково загублений. У цьому випадку система надсилає на пошту користувача лист із посиланням на сторінку на сайті, перейшовши на яку користувач матиме можливість змінити свій старий пароль на новий. Це робиться для того, щоб переконатися, що справжній власник облікового запису запитує зміну пароля, а не якийсь зловмисник.

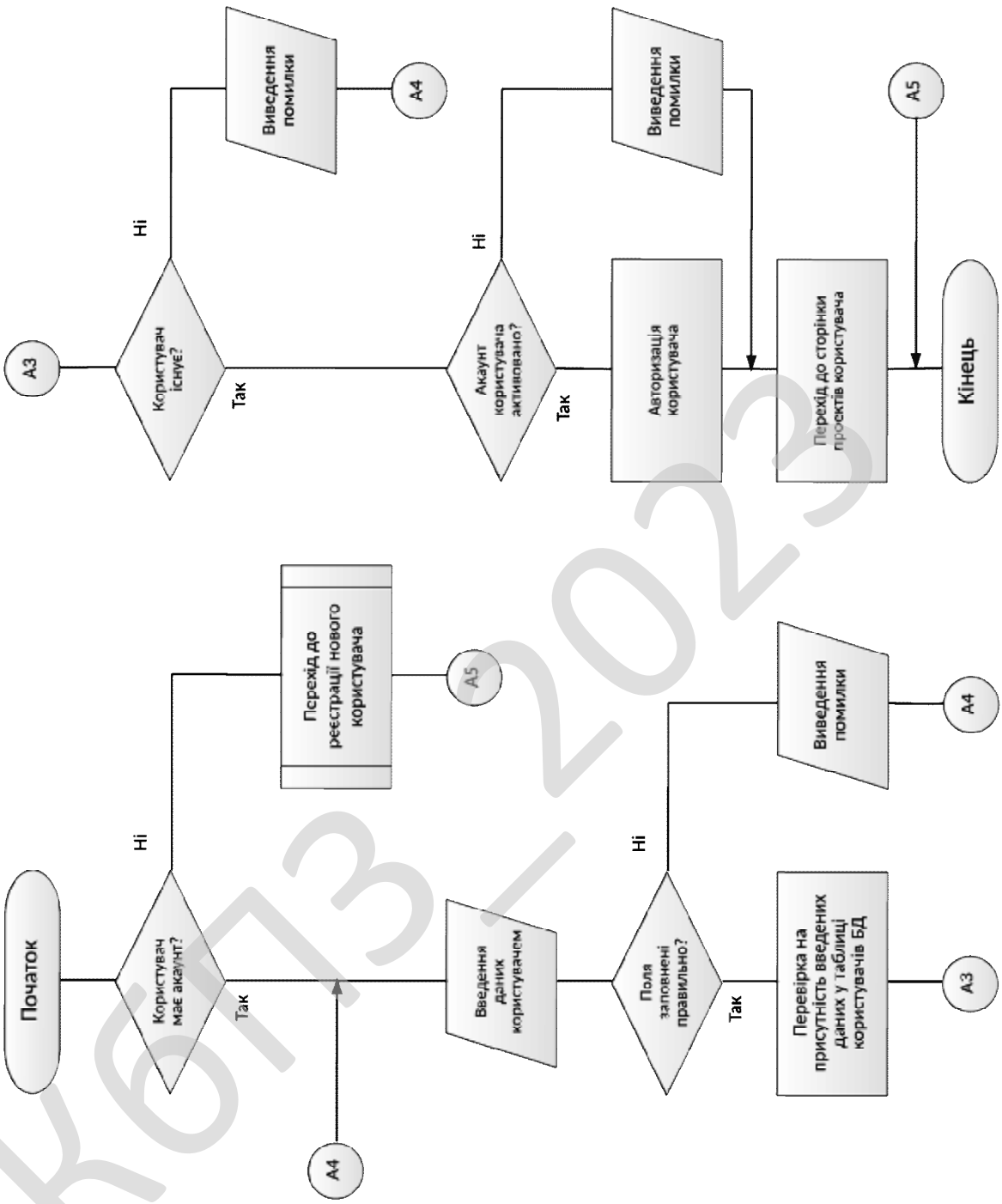

Рисунок 4.2 –Блок-схема алгоритму роботи сторінки авторизації

Наступний блок коду відповідає за реалізацію вищенаведеного алгоритму:

```
        // Обробник кнопки авторизації
        $("#login_signup").click(function(){
            // Отримуємо значення полів логіну та паролю
                 var username = $("#login_username").val(),
                password = $("#login_password").val();
            // Якщо вони пусті виводимо помилку
            if(username == '' || password == '') {
                $('.login_error_confirm').css({'height': '35px', 'background-color':
'#F9D8D3', 'border-left': '6px solid #F9756B'});
                $('.login_confirm_message').html("Заповніть порожні поля!");
                return false;
            } else {
                $('.login_error_confirm').css({'height': '0px', 'background-color':
'#fff', 'border-left': '0px', 'margin': '0 auto'});
                $('.login_confirm_message').html('');
            }
            // Інакше виконуємо AJAX-запит до файлу check_user.php з даними
            $.ajax({
                type: "POST",
                url: "ajax/check_user.php",
                data: {
                    check_user: true,
                    username: username,
                    password: password
                }
            // Функція виконується при завершенні AJAX-запиту
            }).success(function(msg){
                // Якщо прийшла відповідь, що такого користувача не існує, виводимо
помилку
                if(msg == 'false') {
                    $('.login_error_confirm').css({'height': '50px', 'background-
color': '#F9D8D3', 'border-left': '6px solid #F9756B'});
                    $('.login_confirm_message').html("Неправильний логін або
пароль!");
                } else {
                        // Інакше переходимо до сторінки з проектами
                    window.location.href = "/me";
                }
            });
        });
        // Обробник AJAX-запитів check_user.php
        session_start();
```

```
        $config = include "../config/db.php";
        if(isset($_POST['check_user'])){
           $user = $_POST['username'];
           $pass = $_POST['password'];
           try {
              $dbh = new PDO("mysql:host=$config[host];dbname=$config[db_name]",
$config['username'], $config['password']);
              $sth = $dbh->prepare("SELECT * FROM users WHERE username =
'$user'");
              $sth->execute();
              $sth->setFetchMode(PDO::FETCH_ASSOC);
              $row = $sth->fetch();
              if($row['id'] != ''){
                 if (password_verify($pass, $row['password']) && $row['active'] ==
1) {
                    echo 'true';
                    $_SESSION['is_logined'] = true;
                    $_SESSION['login'] = $user;
                    setcookie("login", $user, strtotime( '+30 days' ), "/", "",
"", TRUE);
                    setcookie("id", $row['id'], strtotime( '+30 days' ), "/", "",
             "", TRUE);
                 }
              }
              echo 'false';
           } catch (PDOException $e){
              echo $e->getMessage();
        }
        $dbh = null;
            }
```

4.2 Захист розробленого програмного забезпечення

Захист облікових записів

Для захисту даних користувачів на сайті використовується крипто алгоритм bcrypt.

bcrypt - адаптивна криптографічний хеш функція формування ключа, що використовується для захищеного зберігання паролів. Розробник: Нільс Провос. Функція заснована на шифрі Blowfish, вперше представлена на USENIX в 1999 році. Для захисту від атак за допомогою райдужних таблиць bcrypt використовує salt; крім того, функція є адаптивною, час її роботи легко налаштовується і її можна уповільнити, щоб ускладнити атаку перебором.

Шифр Blowfish відрізняється від багатьох алгоритмів обчислювально складною фазою підготовки ключів шифрування.

Провос скористався цією особливістю, але змінили алгоритм підготовки ключів, отримавши шифр «Eksblowfish» (expensive key schedule Blowfish). Кількість раундів в підготовці ключів повинно бути ступенем двійки; конкретна ступінь може здаватися при використанні bcrypt.

Спочатку реалізовано в функції crypt в OpenBSD. Існують реалізації для Java, Python, Nim, C #, Ruby, Perl, PHP 5.3, Node.js і деяких інших

Алгоритм bcrypt використовує алгоритм настройки ключів з «Eksblowfish»:

```
EksBlowfishSetup(cost, salt, key)
    state
InitState()
    state
ExpandKey(state, salt, key)
    repeat (2cost)

        state
ExpandKey(state, 0, key)
        state
ExpandKey(state, 0, salt)
    return state
ExpandKey(state, salt, key)
    for(n = 1..18)
        Pn
```

Функція InitState відповідає оригінальній функції з шифру Blowfish; для заповнення масиву P і S-box використовується дрібна частина числа {\ displaystyle \ pi} \ pi.

Функція ExpandKey:

```
key[32(n-1)..32n-1]

\oplus  Pn //treat the key as cyclic

    ctext

Encrypt(salt[0..63])

    P1

ctext[0..31]

    P2

ctext[32..63]

    for(n = 2..9)

        ctext

Encrypt(ctext

salt[64(n-1)..64n-1]) //encrypt using the current key schedule and treat the salt as cyclic

        P2n-1)

  ctext[0..31]

        P2n

  ctext[32..63]

    for(i = 1..4)

        for(n = 0..127)

            ctext

  Encrypt(ctext


salt[64(n-1)..64n-1]) //as above

            Si[2n]

  ctext[0..31]

            Si[2n+1]

  ctext[32..63]

    return state
```

Для обчислення хешу bcrypt обробляє вхідні дані еквівалентно шифрування 'eksblowfish (усіленний_ключ, input)':

```
bcrypt(cost, salt, key, input)
    state
EksBlowfishSetup(cost, salt, key)
    ctext
input
    repeat (64)
        ctext
EncryptECB(state, ctext)
    return Concatenate(cost, salt, ctext)
```

У різних ОС (linux, OpenBSD), що використовують алгоритм bcrypt в стандартній функції crypt (3), як input подається константа «OrpheanBeholderScryDoubt» bcrypt був розроблений в 1999 році і був захищений від ефективного перебору на апаратних засобах того часу. В даний час одержали широке поширення ПЛІС, в яких bcrypt реалізується ефективніше. У 2009 був створений алгоритм scrypt, що вимагає для своєї роботи значний обсяг пам'яті (, об'єм пам'яті налаштовується.

SQL-ін'єкції

Багато веб-розробників навіть не усвідомлюють, що SQL-запити можуть бути фальшивими, і вважають, що SQL-запити завжди дійсні. Насправді шахрайські запити можуть обійти обмеження доступу, стандартні перевірки авторизації, а деякі типи запитів можуть дозволити виконання команд операційної системи.

Безпосереднє впровадження зловмисних інструкцій у запити SQL — це техніка, за якої зловмисник створює або змінює поточні запити SQL, щоб відобразити приховані дані, змінити їх або навіть виконати зловмисні команди операційної системи на сервері бази даних. Атака заснована на програмі, яка створює SQL-запити з введених користувачем і статичних параметрів. Наступні приклади, на жаль, базуються на реальних фактах.

Через відсутність перевірки введення користувача та підключення до бази даних під обліковим записом суперкористувача (або будь-якого іншого користувача з відповідними привілеями), зловмисник може створити іншого користувача бази даних з правами суперкористувача.

Створення суперкористувача в VelocityDb

```
<?php
$offset = $argv[0]; // внимание, нет проверки вводимых данных!
$query  = "SELECT id, name FROM products ORDER BY name LIMIT 20 OFFSET
$offset;";
$result = pg_query($conn, $query);
?>
```

Зазвичай користувачі переходять по посиланнях 'вперед' і 'назад', внаслідок чого значення змінної $ offset заноситься в URL . Скрипт очікує, що $ offset - десяткове число. Однак, зловмисник може спробувати зламати систему, приєднавши до URL додатковий рядок, оброблену функцією urlencode () :

```
0;
insert into pg_shadow(usename,usesysid,usesuper,usecatupd,passwd)
    select 'crack', usesysid, 't','t','crack'
```

```
    from pg_shadow where usename='postgres';
--
```

Якщо це станеться, сценарій надасть зловмиснику суперкористувача доступ до бази даних. Зверніть увагу, що значення дорівнює 0; використовується для встановлення правильного зсуву для першого запиту та його правильного завершення.

Ще один можливий спосіб отримати паролі облікових записів бази даних — атакувати сайти, які забезпечують пошук у базі даних. Зловмиснику потрібно лише перевірити, чи запит використовує змінну, яка передається на сервер і не обробляється належним чином. Це може бути один із фільтрів, встановлених на попередній сторінці, як-от WHERE, ORDER BY, LIMIT і OFFSET, які використовуються під час виконання запиту SELECT. Якщо база даних, яку ви використовуєте, підтримує конструкцію UNION, зловмисник може додати іншу конструкцію до початкового запиту, щоб отримати паролі користувачів. Ми наполегливо рекомендуємо використовувати лише зашифровані паролі.

Лістинг паролів для будь-якої бази даних

```
<?php
$query  = "SELECT id, name, inserted, size FROM products
           WHERE size = '$size'";
$result = odbc_exec($conn, $query);
?>
```

Статична частина запиту може комбінуватися з іншим запитом SELECT, який виведе всі паролі:

```
    union select '1', concat(uname||'-'||passwd) as name, '1971-01-01', '0'
from usertable;
--
```

Якщо цей запит приєднати до значення однієї із змінних, використовуваних для формування $ query , то запит помітно зміниться.

Команди UPDATE також можна використовувати для атаки. І знову загрожує розбити інструкцію на кілька частин і додати додаткову вимогу.

Зловмисник також може змінити вираз SET. В цьому випадку потенційний зловмисник повинен мати щось більше інформація про структуру бази даних для успішної обробки запитів. Цю інформацію можна отримати, проаналізувавши імена змінних, які використовуються у формі, або просто перебравши всі найпоширеніші варіанти імен відповідних полів (а їх не так багато).Отримання додаткових привілеїв (для будь-якої бази даних):

```
<?php

$query = "UPDATE usertable SET pwd='$pwd' WHERE uid='$uid';";

?>
```

Але зловмисник може ввести значення 'or uid like'% admin% ' для змінної $ uid для зміни пароля адміністратора або просто привласнити змінної $ pwd значення hehehe', trusted = 100, admin = 'yes для отримання додаткових привілеїв. При виконанні запити переплітаються.

Приклад того, як на сервері баз даних можуть виконуватися команди операційної системи

```
<?php

// $uid: ' or uid like '%admin%

$query = "UPDATE usertable SET pwd='...' WHERE uid='' or uid like
'%admin%';";

// $pwd: hehehe', trusted=100, admin='yes

$query = "UPDATE usertable SET pwd='hehehe', trusted=100, admin='yes'
WHERE

...;";

?>
```

Виконання команд операційної системи на сервері

```
<?php

$query  = "SELECT * FROM products WHERE id LIKE '%$prod%'";

$result = mssql_query($query);?>
```

Якщо зловмисник введе значення a% 'exec master..xp_cmdshell' net user test testpass / ADD '- для змінної $ prod , тоді запит $ query буде виглядати так:

```
<?php

$query  = "SELECT * FROM products
           WHERE id LIKE '%a%'
           exec master..xp_cmdshell 'net user test testpass /ADD' --%'";
$result = mssql_query($query);
?>
```

SQL Server виконує команди SQL у пакетному режимі, включаючи операції для встановлення облікових записів локальної бази даних. Якщо програма запускається з правами адміністратора, а служба SQL запускається з необхідними правами, то, виконуючи описані вище дії, зловмисник отримає команду на доступ до сервера.

Методи захисту

Хоча все ще очевидно, що зловмисник повинен мати принаймні деякі знання про структуру бази даних, щоб розпочати успішну атаку, отримати цю інформацію часто дуже просто. Наприклад, якщо база даних є частиною пакета програмного забезпечення з відкритим вихідним кодом або іншого загальнодоступного програмного забезпечення з інсталяцією за замовчуванням, ця інформація є повністю відкритою та доступною. Ці дані також можна отримати з закритого проекту, навіть якщо він закодований, складний або скомпільований, і навіть з вашого власного коду через появу повідомлення про помилку. Інші методи включають використання загальних імен таблиць і стовпців. Наприклад, форма входу, яка використовує таблицю «користувачі» з назвами стовпців «id», «username»

Більшість успішних атак базується на коді, написаному без належних міркувань безпеки. Не довіряйте будь-якому введенню, особливо якщо воно надходить від клієнта, навіть якщо це списки у формі, приховані поля чи файли cookie.

Заходи безпеки

Ніколи не підключайтеся до бази даних за допомогою облікового запису власника бази даних або суперкористувача. Завжди намагайтеся використовувати спеціально створених користувачів з максимально обмеженими правами.

Використовуйте підготовлені вирази зі зв'язаними змінними. Таку можливість надають розширення PDO, MySQL та інші бібліотеки.

Завжди перевіряйте, чи введені дані відповідають очікуваному типу. PHP має багато функцій для перевірки даних, від найпростіших функцій для роботи зі змінними та функцій для визначення типу символу (таких як is_numeric () і ctipe_digit () відповідно) до регулярних виразів, сумісних з Perl.

Якщо програма очікує введення, використовуйте функцію ctipe_digit (), щоб перевірити введення, або примусово введіть тип за допомогою settype (), або просто використовуйте числове представлення за допомогою функції sprintf ().

Безпечніша реалізація посторінкової навігації

```
<?php

settype($offset, 'integer');

$query = "SELECT id, name FROM products ORDER BY name LIMIT 20 OFFSET
$offset;";

$query = sprintf("SELECT id, name FROM products ORDER BY name LIMIT 20
OFFSET %d;",

                 $offset);

?>
```

Якщо змінні прив'язки не підтримуються на рівні бази даних, завжди екрануйте нечислові дані, які використовуються в запитах до бази даних, за допомогою спеціальних функцій екранування, специфічних для використовуваної бази даних (наприклад, miskl_real_escape_string() , sklite_escape_string() тощо).). Загальні функції, такі як addslashes(), корисні лише в певних випадках, тому краще уникати їх використання.

Ні в якому разі не виводьте будь-яку інформацію про базу даних, особливо про її структуру.

Можливе використання процедур абстрактної роботи з даними, які надають користувачам прямий доступ до даних, але це рішення має свою специфіку.

Очевидно, що відстеження не може запобігти пошкодженню, але воно може допомогти відстежити скомпрометовану програму. Файл журналу корисний не сам по собі, а через інформацію, яку він містить.

5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Для того, щоб ВЕБ-сайт став доступним для відвідувачів, його необхідно розмістити на Інтернет-сервері, підключеному до Інтернету, і отримати доменне ім’я.

Дуже важливо, щоб сайт розміщувався на потужному сервері 24/7 з високою пропускною здатністю. Вибір доменного імені також є дуже важливим моментом у створенні та просуванні WEB сайту. Його відвідуваність і успіх значною мірою залежать від назви, обраної для веб-сайту. Щоб використовувати доменне ім’я (стати адміністратором домену), його необхідно зареєструвати.

Кожен комп’ютер в Інтернеті має свою унікальну IP-адресу. Отже, якщо сторінка знаходиться на комп'ютері з IP-адресою, наприклад, 194.34.45.8 в каталозі / info, то будь-який користувач може переглянути її, ввівши в адресному рядку браузера таку адресу: http:// 194.34.45.8/інфо . Однак запам'ятовувати і вводити адреси в цифровому вигляді не дуже зручно. Для полегшення завдання існує система так званих доменних імен (DNS - Domain Name System).

Домен — це область ієрархічного простору імен Інтернету, яка обслуговується та централізовано адмініструється набором серверів доменних імен (DNS). Домен ідентифікується доменним іменем. Більшості IP-адрес призначається ім'я - ім'я домену. Наприклад, веб-сайт Xerox Corporation розташований за адресою http://www.kerok.com. запам'ятати та ввести такий запис набагато легше, ніж безглузду комбінацію цифр. Доменні імена видаються за суворими правилами. В кінці назви, після останньої крапки, знаходяться так звані доменні імена верхнього рівня. У більшості випадків це двобуквений код країни, якій належить цей ресурс. Наприклад, закінчення. Ru означає Росія. Ua - Україна,. De - Німеччина,.Fr – Франція тощо.

Крім того, домен верхнього рівня також може бути трибуквеним кодом, який означає організаційну приналежність ресурсу. Приклад,. Com - комерційна організація. Правительство – уряд. Військові тощо Доменні імена читаються справа наліво. Та частина доменного імені, яка знаходиться безпосередньо перед останньою крапкою, називається доменним іменем другого рівня. Домени другого рівня видаються (зазвичай за гроші) власниками доменів першого (верхнього) рівня. У будь-якому випадку мати власний домен другого рівня вважається досить престижним. Власники доменних імен другого рівня, у свою чергу, можуть розпоряджатися доменними іменами третього рівня тощо, але подальше розміщення імен у більшості випадків має лише декоративне значення. Крайню ліву позицію в записі доменного імені займає мережеве ім’я комп’ютера (в локальній мережі), на якому запущена серверна програма. Багато дають цьому комп'ютеру ім'я VVV, щоб підкреслити приналежність його ресурсів сервісу Всесвітньої павутини - тоді доменне ім'я сайту починається з поєднання VVV. Реєстрація домену - отримання та внесення необхідної інформації до бази даних адміністратора зони, перевірка правильності функціонування відповідних серверів доменних імен (DNS) в рамках міжнародної системи Інтернет. Реєстрація забезпечує збереження інформації в базі даних адміністратора зони протягом облікового періоду. Він також передає права адміністрування домену адміністратору домену. Адміністратор домену - юридична або фізична особа, на яку зареєстровано домен. Щоб забезпечити досягнення цілей, заради яких створювався веб-сайт, він мав бути побудований на високому рівні. Треба, щоб про це дізналися ті, для кого вона створена.

Існує досить багатий арсенал інструментів для забезпечення відвідуваності сайту:

– приміщення оголошень в Internet конференціях;
– обмін посиланнями з сайтами, що мають схожу аудиторію;
– участь у мережах обміну банерами;
– платне розміщення банерів на «розкручених» порталах;

– реєстрація сервера на пошукових серверах і в каталогах;

– реклама на серверах розсилок новин. Такі сервери розсилають по електронній пошті інформацію на різні теми своїм передплатникам. У розсилку може бути включена реклама WEB-сайту;

– розширення змісту WEB-сайту з метою залучення допоміжних груп аудиторії;

– розсилка повідомлень про відкриття сайту потенційним споживачам продукції компанії, як по електронній пошті, так і традиційним чином;

– розміщення адреси сайту на всіх вихідних документах компанії, рекламних та інформаційних матеріалах, упаковці товарів;

– реклама сайту в традиційних засобах масової інформації;

Який із цих інструментів використовувати, залежить від багатьох факторів: цілей проекту, характеру аудиторії сервера, бюджету тощо. Основні характеристики рекламної компанії для просування WEB сайту компанії повинні бути визначені на етапі розробки концепції. Обов'язковими елементами, на мій погляд, є реєстрація сайту на пошукових серверах і в каталогах, а також розміщення адреси сайту на всіх вихідних документах компанії, рекламних та інформаційних матеріалах. У будь-якому випадку непогано розіслати клієнтам і потенційним споживачам продукції компанії спеціальні повідомлення про відкриття сервера.

Реєстрація домену

Реєстрація домену - це процес внесення нового доменного імені до зонального реєстру першого рівня. Процедура реєстрації домену проста, достатньо зареєструвати обліковий запис у реєстратора доменних імен, заповнити обліковий запис, перевірити доменне ім’я на працевлаштування та створити заявку, якщо доменне ім’я є. Після реєстрації домену (внесення до реєстру запису з даними про адміністратора, реєстратора, дату реєстрації та закінчення терміну дії, статус делегування) доменне ім’я готове до використання, як правило, через 5-10 хвилин.

Для використання домену необхідно в інтерфейсі реєстратора вказати (делегувати) для нього dns-сервер (хостинг).

Реєстр доменних імен - це організація, уповноважена створювати (реєструвати) нові доменні імена та продовжувати термін дії існуючих доменних імен в домені, для якого встановлено обов'язкову реєстрацію. Такими доменами є:

– всі домени верхнього (першого) рівня;

– деякі домени другого рівня (наприклад, com.ru або co.uk);

– домен нульового рівня (кореневий домен).

У всіх інших доменах для створення субдоменів не потрібні спеціальні.

Функції реєстратора базових доменів (таких як домени .com, .ru, .de, .org та ін.) виконує організація ICANN. Часто реєстратор домену верхнього рівня є власником або адміністратором кореневого домену. Власником базового домену може бути держава, недержавне (в тому числі міжнародне) співтовариство або навіть окрема особа. Але часто державний власник або міжнародна організація-власник конкретного кореневого домену в особі свого спеціалізованого органу виконує функції координатора та ліцензіара реєстрації домену, тоді як безпосередню реєстрацію домену в цьому базовому домені здійснюють реєстратори. уповноважений (ліцензований, акредитований) цим координуючим органом.

Звичайно, в таких кореневих доменах реєстратор не єдиний. У разі наявності декількох реєстраторів усі вони повинні використовувати одну (централізовану або розподілену) базу даних для усунення конфліктів і забезпечення унікальності доменного імені. Наприклад, у домені .ua всі реєстратори доменів використовують одну доменну базу даних у кореневому домені .ua. Щоб стати реєстратором доменів у зонах .com .net .org .biz .info .name .mobi .asia .aero .tel .travel .jobs, вам потрібно отримати акредитацію ICANN.

Хостинг

Хостинг (англ. hosting) — послуга, що включає надання дискового простору, мережевого підключення та інших ресурсів для розміщення фізичної інформації на сервері, який постійно знаходиться в мережі (наприклад, Інтернет).

Поняття хостинг включає в себе широкий спектр послуг з використанням різного апаратного та програмного забезпечення. Зазвичай поняття послуг хостингу має на увазі, як мінімум, послугу розміщення файлів сайту на сервері, на якому працює програмне забезпечення, необхідне для обробки запитів на ці файли (веб-сервер). Як правило, послуги хостингу вже включені надання місця для поштової кореспонденції, баз даних, зберігання DNS-файлів тощо, а також забезпечення функціонування відповідних сервісів, але вони можуть надаватися і окремо. Зокрема, послуга може бути обмежена розміщенням пошти та відповідного програмного забезпечення (поштовий хостинг), клієнтських файлів (файловий хостинг), виключно відеофайлів (відеохостинг) або інших файлів певного типу та за певних умов.

Послуги хостингу можуть надаватися в комплексі з іншими інформаційними послугами, такими як реєстрація доменного імені, створення сайту, надання додаткового програмного забезпечення тощо.

Хостинг-провайдерами можуть бути як компанії, що спеціалізуються на цих послугах («хостери»), так і великі постачальники інформаційних послуг, що спеціалізуються на інших сервісах (таких як Google, Microsoft, Yahoo та інші).

Є безкоштовний і платний хостинг. Безкоштовні «хостери» заробляють, розміщуючи рекламу на своїх сайтах або надаючи інші платні послуги (в комплекті з безкоштовними або як опцію).

Завантаження вихідного коду

Після придбання доменного імені та послуг по хостингу необхідно завантажити на сервер вихідний код розробленого ПЗ за допомогою програм на кшталт FileZilla та перевірити правильність роботи сайту.

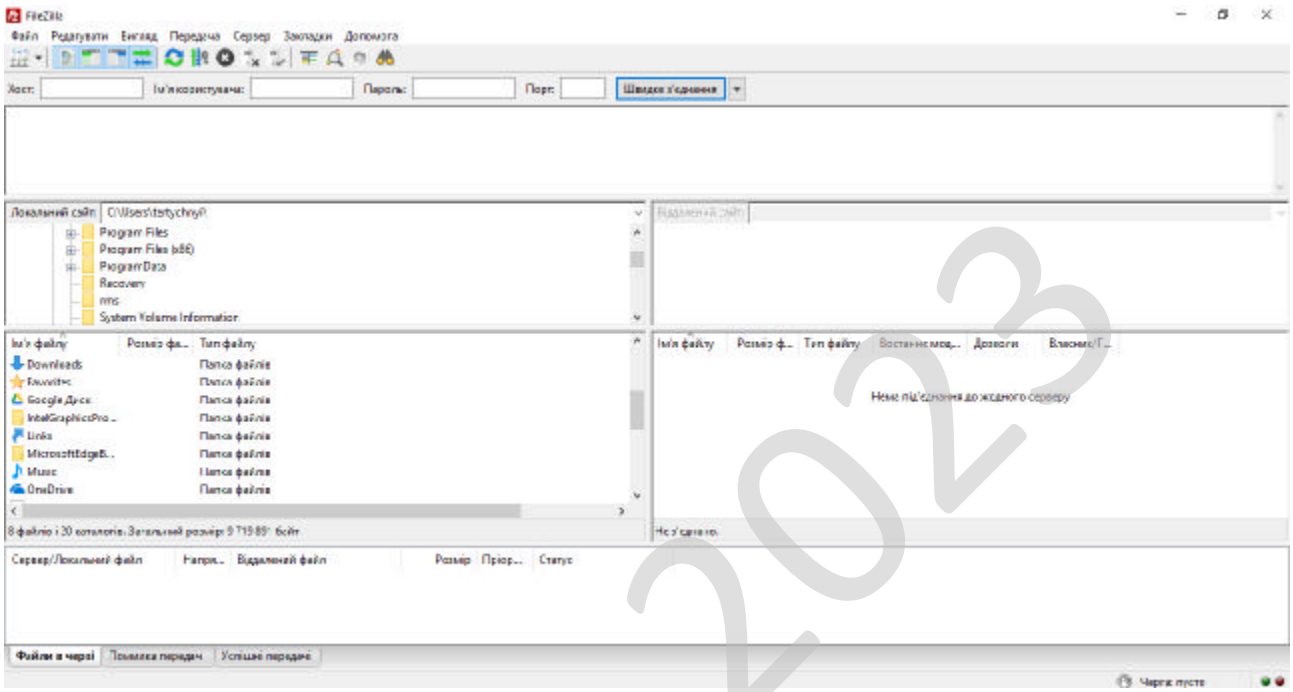

Рисунок 5.2 – Головне вікно програми для завантаження файлів на сервер

Після завантаження на сервер БД сайту можна керувати через спеціальні додатки.

Веб-додаток phpMyAdmin з відкритим кодом на мові PHP із графічним веб-інтерфейсом для адміністрування бази даних MySQL або VelocityDB.

phpMyAdmin дозволяє через браузер здійснювати адміністрування сервера MySQL, запускати запити SQL, переглядати та редагувати вміст таблиць баз даних. Ця програма користується великою популярністю у веб-розробників, оскільки дозволяє керувати базу даних MySQL без вводу SQL команд через дружній інтерфейс і з будь-якого комп'ютера під'єднаного до інтернету без необхідності встановлення додаткового програмного забезпечення.

На сьогоднішній день phpMyAdmin широко застосовується на практиці. Останнє пов'язано з тим, що розробники інтенсивно розвивають свій продукт, з огляду на всі нововведення СКБД. Переважна більшість українських провайдерів використовують цей застосунок як панель керування для того, щоб надати своїм клієнтам можливість адміністрування виділених їм баз даних.

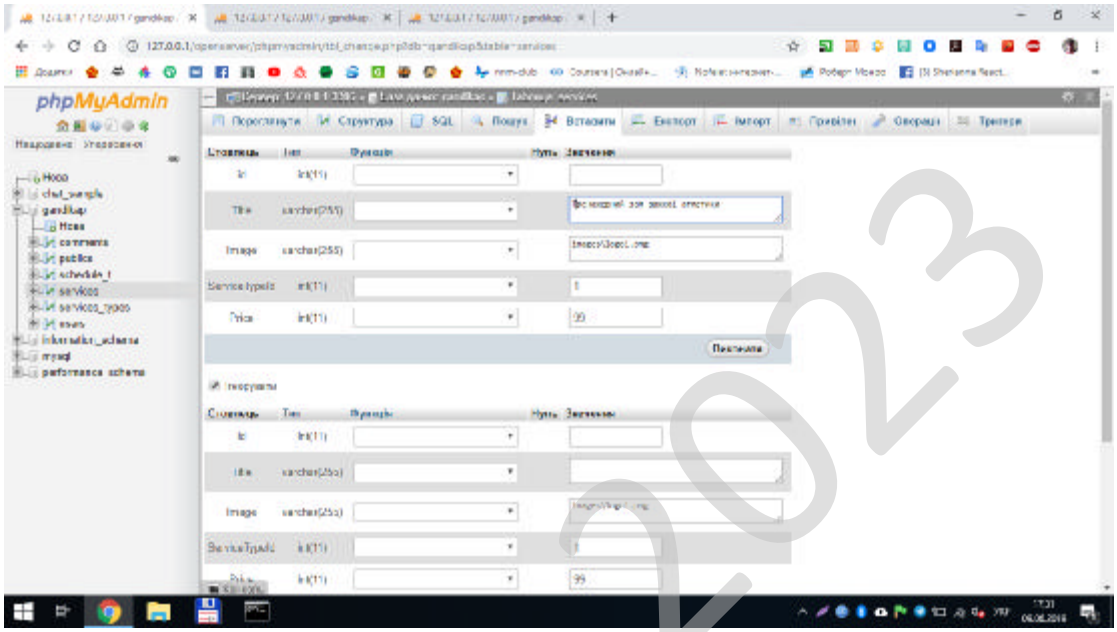

Рисунок 5.3 – Додаток phpMyAdmin

Самим сайтом та хостингом можна керувати зі спеціальних панелей керування.

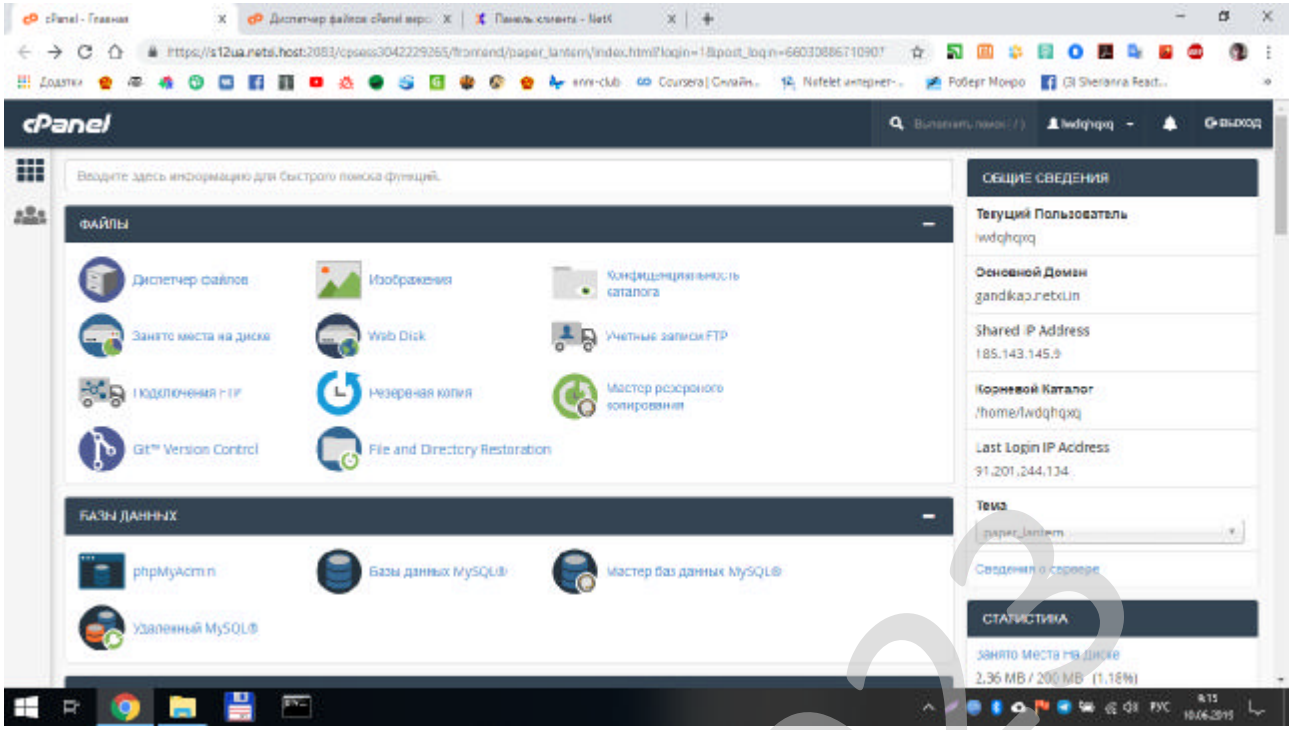

Рисунок 5.5 – cPanel хостингу Nexti

6 НАУКОВА НОВИЗНА

У магістерській роботі розроблено програмне забезпечення, яке призначено для системи аналізу застосування різних типів баз даних в сучасних ІС.

Метою розробки є дослідження та програмна реалізація системи аналізу застосування ООБД при розробці ПЗ.

Об'єктом дослідження є процес застосування об'єктно-орієнтованих баз даних при розробці WEB застосунків.

Предметом дослідження є методи реалізації побудови ІС з застосуванням об'єктно-орієнтованих баз даних.

Методи дослідження базуються на методах розробки баз даних, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів.

У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– удосконалено метод застосування БД при розробці ІС;

– розроблено вітчизняний продукт (web сайт)з застосування ООБД, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми дипломного проекту

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці).

В магістерській роботі було проведено дослідження та виконана програмна реалізація застосування об’єктно-орієнтованої бази даних.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп’ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	24
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	3
8. Кількість форм вихідної інформації	–	4
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	2
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	2
20. Вимоги до швидкодії ПП (1-6)	–	2
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	2
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	2
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	2
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
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 (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	24000
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	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B, \qquad (7.1)$$

де: A – коефіцієнт Боема, A = 2,45;

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

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i\,, \qquad (7.2)$$

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

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45\cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

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

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

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

$$T_{уточн} = 6{,}78\cdot(0{,}88\cdot 0{,}93\cdot 0{,}88\cdot 0{,}91\cdot 0{,}95\cdot 1\cdot 1\cdot 0{,}87\cdot 1{,}22\cdot 1{,}16\cdot 1{,}1\cdot 1{,}1\cdot$$
$$\cdot 1{,}12\cdot 1{,}1\cdot 1\cdot 1{,}1\cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3CT_{уточн}^{0{,}33+0{,}2(B-1{,}01)}S\,, \qquad (7.4)$$

де: *C* – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4);

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

$$T_{РП} = 0{,}3\cdot 2{,}66\cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot 100 = 168 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	168	Ф 7.1-7.4
Впровадження	13	Д13
Всього	209	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \quad (7.5)$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{209 \cdot 1}{60 - 5} = 3,8 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – 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	90	7	630	10,5
Монітор	60	7	420	7
Клавіатура	30	7	210	3,5
Маніпулятор «мишка»	30	7	210	3,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор–маршрутизатор	30	5	150	2,5
Кабельні господарства ЛВС на 1 м. п.	2,5	300	750	12,5
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	45,16

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{міс}}{1,2}, \quad (7.6)$$

$$Ф_{др}^{с} = \frac{45 \cdot 3}{1,2} = 112,5 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф^{с}_{др}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$Ч_{ел}$ = 112,5/(60·8) = 0,25 ставки.

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, 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 7.4 – 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, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN, PPPoE, Frame Relay, Wi-Fi	2	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	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
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 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	12450	37350
Продакт-менеджер	0,25	12000	9000
Інженер-програміст	3,8	12000	136800
Інженер-електронщик	0,25	11500	8625
Інженер-системотехнік	0,25	11500	8625
Адміністратор мережі	0,5	11500	17250
Системний програміст	0,25	11500	8625
Дизайнер WEB	0,25	12000	9000
Інженер-верстальник	0,25	11700	8775
Бухгалтер-економіст	0,5	12500	18750
Всього за період розробки	$R_{сп}$ = 7,3	-	$Ф_{роб}$ = 262800

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рq}}, \quad (7.8)$$

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

$$З_{сд} = \frac{262800}{7,3 \cdot 60} = 600 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

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

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

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторалъ» (м. Кіровоград, вул. Глинки 16) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 400…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 37 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно $8 м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн.}$$

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались за прайсом фірми Brain за 24.10.23 – джерело http: //brain.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		11771
Системний блок		7771
Процесор	AMD Ryzen 5 4500 (100-000000644) AM4, 6 ядер, 12 потоків, 3.6 GHz, 4.1 GHz, (TDP) - 65 Вт, 7nm	-
Системна плата	GIGABYTE A520M H AM4, DDR4, LAN - 1 Гбіт/с, DVI, HDMI, 1 x M.2 2280, 4 x Sata 6.0 Gb/s, Micro-ATX	-
Жорсткий диск	SSD M.2 2280 240GB Apacer 240 GB, TLC, M.2, SATA III (6Gb/s)	-
Оперативна пам’ять	DDR4 8GB 2400 MHz Patriot	-
Система охолодження	Deepcool BETA 10	-
Корпус	ProLogix E105 Minitower, Micro - ATX, Mini - ITX,	-
Блок живлення	Vinga 450W OEM	-
інше	Клавіатура, мишка	-
Монітор	Монітор BenQ GL2450HM Black	2600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Сканер	Epson Perfection V37	2800
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійного живлення	Powercom BNT-600AP USB	1400

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

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 - Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп'ютери	8	11771	9416,8	103584,8
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2800	280	3080
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	125216,3

Таблиця 7.8 – 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	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400

Продовження таблиці 7.8

1	2	3	4
Група 4			
3. Обчислювальна техніка	125216	-	-
Всього по групі	125216	50	62608
Група 5,6			
4. Вимірювальні пристрої	5190	20	-
5. Транспортні засоби	143000	25	
6. Господарський інвентар	28000	20	-
Всього по групам 5, 6	176190	-	35238
7. Нематеріальні активи	24000	10	2400
Разом	$К_р$= 1733406		$А_р$= 170646

Примітка: вартість автомобіля взята по даним з автосалону «Кіровоград-Авто», джерело http://kirovograd-avto.ukravto.ua/catalog/tm-9/model-80/description, складає 143000 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

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

$$З_о = 600 \cdot 209/24 = 5225 \text{ грн.}$$

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

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 5225 \cdot 10 \cdot 0{,}01 = 523 \text{ грн.}$$

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

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

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

$$С_{оц} = 0{,}01 \cdot 22(5225+523) = 1265 \text{ грн.}$$

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

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 5225 \cdot 15 \cdot 0{,}01 = 784 \text{ грн.}$$

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

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.;

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

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

$N_е$ – кількість екземплярів програм, шт.

Згідно виданих норм приймаємо 1/2 пачки паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п = 210$ грн., визначаємо вартість паперу за період розробки $N_м = 3$ міс:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.16)$$

$$З_{М1} = 210 \cdot 1/2 \cdot 3 = 315 \text{ грн.}$$

Згідно виданих норм до вартості запам’ятовуючих пристроїв входить вартість CD/DVD дисків в кількості 10 примірників:

$$З_{М2} = \sum Ц_д, \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 30 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 30 грн./шт.

$$З_{М2} = 30 \cdot 10 = 300 \text{ грн.}$$

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

$$З_{МЗ} = \sum Ц_{з}., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{МЗ} = 574+558+570 = 1702 \text{ грн.}$$

$$З_{М} = (315 + 300 + 1702)/24 = 97 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п = 15\%$) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 5225 \cdot 15 \cdot 0{,}01 = 784 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е = 24$ прим.):

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $А_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$А_м = 170646 \cdot 3/(24 \cdot 12) = 1777 \text{ грн.}$$

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

$$С_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$С_п = 5225+523+1265+784+97+784+1777 = 10455 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 50%.

$$П_р = 0{,}01 \cdot Р_п \cdot С_п, \quad (7.22)$$

де: $Р_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 10455 = 5228 \text{ грн.}$$

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

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1	2	3
1. Основна зарплата виконавців	$З_о$	5225
2. Додаткова зарплата виконавців	$З_д$	523
3. Відрахування на соціальні потреби	$С_{оц}$	1265
4. Загальногосподарські витрати	$Г_{осп}$	784
5. Витрати на матеріали	$З_М$	97
6. Освоєння нових операційних систем, мов програмування	$О_п$	784
7. Амортизація основних фондів	$А_м$	1777
8. Повна собівартість програмного забезпечення	$С_п$	10455
9. Плановий прибуток	$П_р$	5228
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	15683
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	3136,6
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	18819,6

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 7.9.

Таблиця 7.10 – Розрахунок об'єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	18820
Всього капітальних витрат	–	18820

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

$$З_о = Ч_е О_е М_е (1 + 0{,}01 Н_q)(1 + 0{,}01 Н_c), \quad (7.23)$$

де $Ч_е$ – чисельність працівників, чол.;

$О_е$ – заробітна плата, грн./год;

$М_е$ – час, що витрачається на нарахування ЗП.

Після купівлі нового програмного забезпечення час на нарахування зменшився з 530 годин на рік до 393 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{о\,баз} = 1 \cdot 100 \cdot 530 \cdot 1{,}1 \cdot 1{,}22 = 71126\ грн,$$

до:

$$З_{о\,нов} = 1 \cdot 100 \cdot 393 \cdot 1{,}1 \cdot 1{,}22 = 52741\ грн.$$

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Заробітна плата(основна, додаткова, відрахування в соціальні фонди)	$З_р$	71126	52741
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	0	4705
Всього витрат за рік	$І$	71126	57446

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

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \qquad (7.24)$$

Після впровадження ПЗ витрати на електроенергію не зміняться, тому: $З_{ел\ баз} = З_{ел\ нов}$

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – 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	–	18820	–	4705
Всього відрахувань	-	–	18820	–	4705

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:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (15683 - 10455) \cdot 24 - (0,05 \cdot 1408000 + 0,5 \cdot 125216 +$$
$$+0,25 \cdot 33190 + 0,2 \cdot 143000 + 0,1 \cdot 24000) \cdot 3/12 = 82396 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: $K_р^*$ – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{1733406}{(15683 - 10455) \cdot 24 \cdot 12 / 3} = 3,45 \text{ років}.$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \quad (7.27)$$

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

$K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (71126\text{-}57446)\text{-}0,25 \cdot 18820 = 8975 \text{ грн.}$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	24
2. Повна собівартість розробленої програми	Грн.	10455
3. Ціна розробленої програми	Грн.	15683
4. Плановий прибуток від реалізації розробленої програми	Грн.	5228
5. Рентабельність програмної продукції	%	50
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1733406
7. Загальний прибуток від реалізації програмної продукції	Грн.	125472
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	82396
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	3,45
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	18820
11.Величина економічного ефекту у користувача програмної продукції	Грн.	8975
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Рік	1,4

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

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \qquad (7.28)$$

$$T_{сп} = \frac{18820}{71126 - 57446} = 1,4 \text{ років}.$$

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 чинників, 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 чинників окремо, проаналізуємо які повинні бути стандарти кожного з чинників, 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 розповсюдженіший 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 освітленість робочого приміщення повинна бути Е =300-500 Лк, оскільки робота програміста відноситься до робіт середньої точності з присвоєнням розряду зорових робіт IV[49].

Рівень світла повинен бути достатнім, щоб працівник міг працювати без навантаження на зір. Це залежить від системи освітлення, кількості світильнику, 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 комп'ютерне обладнання, повинні бути встановлені певні норми, оскільки офісна техніка є джерелом тепловиділень, що може спричинити підвищення температури. Серед

оптимальних параметрів для роботи встановлена температура 23 – 25^0С 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 обладнання, наприклад кондиціонери, вентиляцію. Як правило в холодний період часу використовуються додаткове опалення для підтримання комфортної температури, а в літку встановлюються кондиціонери.

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 цю може спричини головні болі, біль у вухах, підвищення стомлюваності, зниження концентрації та уваги. Такі симптоми можуть викликати стресові ситуації у людини. Все це шкодить продуктивності працівника та його стану здоров’я.

Щоб встановити необхідний рівень шуму, використовують додаткову звукоізоляцію. Для цього найчастіше 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 здороває працівників, їх рівень 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, повинні бути заземлені всі прилади і пристрої, час від часу повинна 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 нормам 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 комп'ютером не повинна безперервно тривати більше 4 годин.

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

В приміщеннях також повинні бути протипожежне обладнання, та інструкція у разі надзвичайних ситуаціях. Повинна бути особа, яка відповідає за пожежну безпеку, перевіряє обладнання, та системи протипожежного захисту а також щорічне проведення інструктажів серед працівників.

Автоматична пожежна сигналізація повинна 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 повинно 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:

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

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

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:

$$Iпус = 5 * Iном \quad (8.2)$$

$$Іпус = 5 * 8,9 = 44,5 А$$

Визначаємо номінальну силу струму апарата захисту:

$$Ін = Іпус/b \quad (8.3)$$

$$Ін = 44,5 /2,5 = 17,8 А$$

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:

$$Іктіп = Ін * К \quad (8.4)$$

$$Іктіп = 50 * 3 = 150 А$$

Ін−номінальний струм апарата захисту;

К − коефіцієнт надійності;

Знаходимо переріз провода або кабеля розгалуження з умови допустимого нагрівання:

$$Ідоп = Івст/а \quad (8.5)$$

$$Ідоп = 50 / 3 = 16,6 А$$

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

$$Iроб = Ко(Кз*(Р*1000)/(\sqrt{3}*Uл*cosф)*m+Кз*(Ро*1000)/(\sqrt{3}*Uл*cosф)) \quad (8.6)$$

Iроб = 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:

$$Iпер = Ко(Кз*(Р*1000)/(\sqrt{3}*Uл*cosф)*n+Кз*(Ро*1000)/(\sqrt{3}*Uл*cosф))+Iпус \quad (8.7)$$

Iпер = 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:

$$Iспр >= 1,25 * Iпер \quad (8.7)$$

Iспр >=1,25 * 130,0643 = 162,5803 А

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

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

$$Iдоп >= Iспр/4,5 \quad (8.8)$$

$$Iдоп >= 162/4,5 = 36 А$$

Проводимо узгодження з номінальним струмом автомата:

$$Iдоп = Iспр/3 \quad (8.9)$$

$$Iдоп = 162/3 = 54 А$$

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:

$$Sn_1 >= 0,5 * Sф \quad (8.11)$$

$$Sn_1 >= 0,5 * Sф = 0,5*70=35 мм^2$$

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 максимальноі напруги на корпусі обладнання відносно землі при замиканні фази на корпус.

$$Uктах = Iкр * Zн \quad (8.15)$$

$$Uкmax = 418,18 * 0,1586 = 66,32 В$$

Z_H − повний опір нульового провідника.

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

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

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

$$Uкmax = 600,21 * 0,0896 = 53,77 В.$$

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 способи їх уникнення та ліквідації.

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

9 ОСНОВНІ ВИСНОВКИ

Створене за результатами магістерської роботи програмне забезпечення призначене для системи аналізу використання різних типів баз даних при розробці WEB-додатків.

В Україні вітчизняні розробки в цьому напрямку з використанням ООБД представлені недостатньо.

Магістерська робота являє собою теоретичне узагальнення та вирішення наукового завдання дослідження методів використання ООБД.

Рішення даного завдання полягало у вирішенні наступних задач:

- проведено огляд існуючих систем, побудованих з використанням різних типів баз даних;

- досліджено систему побудови ВЕБ застосунка щодо використання ООБД;

- на основі отриманих результатів дослідження створено програмну реалізацію прикладної системи ООБД у розробці програмного забезпечення;

Алгоритми, розроблені під час створення магістерської роботи, дозволяють успішно застосовувати ООБД у розробці програмного забезпечення.

Проведено аналіз предметної області, під час якого визначено об’єкти, взаємодія яких має істотне значення для функціональної діяльності предметної області, та їх основні характеристики; будується алгоритм і вибирається середовище розробки.

Розроблене програмне забезпечення має простий, зрозумілий та зручний інтерфейс користувача, що забезпечує легкість освоєння роботи програмного продукту, зручність використання та не потребує спеціальних знань.

При створенні програмного забезпечення використовувався об’єктно-орієнтований підхід з використанням таких технологій та мов програмування:

HTML 5, CSS 3, JavaScript + jQuery, PHP + PDO, JSON, які відповідають сучасним тенденціям у сфері веб-розробки.

Програма реалізована на мові PHP високого рівня з відкритим кодом. Ця мова програмування забезпечує найефективнішу обробку даних і спеціально розроблена для веб-розробки та вбудована безпосередньо в HTML. Це дозволило мінімізувати термін розробки програмного забезпечення і, відповідно, знизити витрати на його розробку. Пропоноване програмне забезпечення поділяється на загальне програмне забезпечення, що постачається разом з комп'ютерною технікою, і спеціальне програмне забезпечення, яке спеціально розроблене для даної конкретної системи та містить програми, що реалізують її функції.

Програма розроблена для роботи під керуванням будь-якої операційної системи для багатозадачності, оскільки для її використання потрібен лише сучасний браузер, такий як «Google Chrome», «Mozilla Firefox», «Opera» та «Safari».

Надано необхідні рекомендації щодо встановлення розробленого програмного забезпечення.

Для підвищення рівня безпеки пропонується використовувати стандартні функції php: password_hash, password_verifi для збереження даних користувача, а також використовуються методи для придушення XSS-атак і SQL-ін'єкцій.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень і повністю відповідає вимогам технічного завдання.

Створене програмне забезпечення має потенціал для подальшого вдосконалення та застосування в різних сферах.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 8975 грн. Враховуючи вартість розробки та необхідне обладнання, строк окуплення становить 1,4 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. Інтелектуальний аналіз даних / А. А. Шумейко, С. Л. Сотник. – Дніпропетровськ: Бела 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 застосунок [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 дослідження та програмну реалізацію ПЗ з застосування ООБД в сучасних ІС.

2 Підстава для розробки

Підставою для розробки служить завдання на магістерську роботу, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 32-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою магістерської роботи є дослідження та програмна реалізація системи аналізу застосування ООБД в ІС.

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 забезпеченням.

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 HTML 5, CSS 3, JavaScript + jQuery + AJAX, PHP + PDO, JSON. Бази даних реляційні та ООБД.

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 2023 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 захисного заземлення.

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 – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 108 аркушів.

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 роботи на попередній захист 10.12.2023 р.

1.2 Подання магістерської роботи на захист 10.01.2024 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 застосування об’єктно-орієнтованої бази даних

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 35

Літера: РП

Кропивницький – 2023 року

```
// -------------------------------Database.cs-----------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using VelocityDb;
namespace LibraryHelper
{
public static class Database
{
public static T GetById<T>(ulong id) where T : OptimizedPersistable
{
using (var conn = ThisApplication.Instance.Session)
{
conn.BeginRead();
var ret = conn
.AllObjects<T>()
.FirstOrDefault(x => x.Id == id);
conn.Commit();
return ret;
}
}
}
}
// -------------------------------Person.cs-----------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using VelocityDb;
using VelocityDb.Collection;
using VelocityDb.Session;
namespace LibraryHelper
{
public class Person : OptimizedPersistable
{
private DateTime? birthDate;
private VelocityDbList<string> emails;
private VelocityDbList<string> phones;
public Person() : base()
{
emails = new VelocityDbList<string>();
phones = new VelocityDbList<string>();
}
/// <summary>
/// Имя
/// </summary>

public string FirstName { get; set; }
/// <summary>
/// Фамилия
/// </summary>
public string LastName { get; set; }
/// <summary>
/// Отчество
/// </summary>
public string MiddleName { get; set; }
/// <summary>
/// Дата рождения человека
/// </summary>
public DateTime? BirthDate
```

```
{
get { return birthDate; }
set
{
if (value.HasValue)
{
birthDate = value.Value.Date;
}
else
{
birthDate = null;
}
}
}
/// <summary>
/// Возраст человека. Явно установить нельзя, зависит от дня
/// рождения
/// и от текущей даты
/// </summary>
public int? Age
{
get
{
if (birthDate == null)
{
return null;
}
var today = DateTime.Today;
int age = today.Year - birthDate.Value.Year;
if (birthDate > today.AddYears(-age))
{
age--;
}
return age;
}
}
/// <summary>
/// Список электронных адресов человека
/// </summary>
public IReadOnlyCollection<string> Emails
{
get
{
return emails.ToList().AsReadOnly();

}
}
/// <summary>
/// Список телефонных номеров человека
/// </summary>
public IReadOnlyCollection<string> Phones
{
get
{
return phones.ToList().AsReadOnly();
}
}
public void AddEmail(string email)
{
emails.Add(email);
}
public void AddPhone(string phone)
{
```

```
phones.Add(phone);
}
public void RemoveEmail(string email)
{
emails.Remove(email);
}
public void RemovePhone(string phone)
{
phones.Remove(phone);
}
public void RemoveEmailAt(int index)
{
emails.RemoveAt(index);
}
public void RemovePhoneAt(int index)
{
phones.RemoveAt(index);
}
public void ClearEmails()
{
emails.Clear();
}
public void ClearPhones()
{
phones.Clear();
}
public override string ToString()
{
return $"{FirstName} {LastName}";
}
public override void Unpersist(SessionBase session)
{
var emails = this.emails;
this.emails = null;

emails.Unpersist(session);
var phones = this.phones;
this.phones = null;
phones.Unpersist(session);
base.Unpersist(session);
}
}
}
// ------------------------Employee.cs--------------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace LibraryHelper
{
public class Employee : Person
{
public string Login { get; set; }
public string Password { get; set; }
}
}
// ------------------------Client.cs----------------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
using VelocityDb.Collection;
namespace LibraryHelper
{
public class Client : Person
{
public string Code { get; set; }
public override string ToString()
{
return $"[{Code}] {base.ToString()}";
}
}
public static class ClientInformation
{
public static IEnumerable<Book> GetCurrentBooks(this Client client)
{
IEnumerable<Book> result;
using (var conn = ThisApplication.Instance.Session)
{
conn.BeginRead();
result =
from x in conn.AllObjects<Book>()
where x.CurrentOwner.Id == client.Id
select x;
conn.Commit();
}
return result.ToList();

}
}
}
// ---------------------------Book.cs---------------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using VelocityDb;
using VelocityDb.Collection;
using VelocityDb.Session;
namespace LibraryHelper
{
public class Book : OptimizedPersistable
{
public class OwnerInfo
{
public DateTime StartDate { get; set; }
public DateTime? EndDate { get; set; }
public Client Client { get; set; }
public Employee Employee { get; set; }
public TimeSpan Durability
{
get
{
return !EndDate.HasValue ?
DateTime.Today - StartDate :
EndDate.Value - StartDate;
}
}
public bool IsReturned
{
get
{
return EndDate != null;
```

```
}
}
}
public string Code { get; set; }
public string Name { get; set; }
public int ReleaseYear { get; set; }
public string Author { get; set; }
public int AgeLimit { get; set; }
private VelocityDbList<OwnerInfo> ownersInfo;
public Book()
{
ownersInfo = new VelocityDbList<OwnerInfo>();
}
public IReadOnlyList<OwnerInfo> Owners
{
get
{
return ownersInfo.ToList().AsReadOnly();
}

}
public void AddOwnerInfo(OwnerInfo info)
{
ownersInfo.Add(info);
}
public void RemoveAt(int index)
{
ownersInfo.RemoveAt(index);
}
public override void Unpersist(SessionBase session)
{
this.ownersInfo = null;
ownersInfo.Unpersist(session);
base.Unpersist(session);
}
public bool IsAvailable
{
get
{
return ownersInfo.Count != 0 ?
ownersInfo.Last().IsReturned :
true;
}
}
public Client CurrentOwner
{
get
{
OwnerInfo last = ownersInfo.LastOrDefault();
if (last == null)
{
return null;
}
if (last.EndDate.HasValue)
{
return null;
}
return last.Client;
}
}
public void AddOwnerInfoList(SessionNoServer session,
IEnumerable<OwnerInfo> info)
{
```

```
if (session == null || info == null)
{
throw new ArgumentNullException();
}
var oinfo = ownersInfo;
ownersInfo = null;
oinfo.Unpersist(session);
ownersInfo = new VelocityDbList<OwnerInfo>();
foreach (var elem in info)


{
ownersInfo.Add(elem);
session.UpdateObject(elem);
}
}
public TimeSpan? Durability
{
get
{
var last = ownersInfo.LastOrDefault();
if (last == null)
{
return null;
}
return last.EndDate.HasValue ? null : new TimeSpan?
(DateTime.Today - last.StartDate);
}
}
public class BookNotReturnedException : Exception { }
}
}
// ----------------------ThisApplication.cs----------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using VelocityDb.Session;
namespace LibraryHelper
{
public class ThisApplication
{
private static ThisApplication instance;
private string databaseName;
private ThisApplication()
{
databaseName = "TestDatabase7";
using (var conn = new SessionNoServer(databaseName))
{
conn.BeginUpdate();
conn.Commit();
}
}
/// <summary>
/// Instance of program settings
/// </summary>
public static ThisApplication Instance
{
get
{
return instance ?? (instance = new ThisApplication());
}
}
```

```
public Employee CurrentEmployee { get; set; }
public string DatabaseName => databaseName ?? String.Empty;

public SessionNoServer Session => new SessionNoServer(databaseName);
}
}
// -------------------------UserFormBirthdayReminder.cs----------------------
-
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace LibraryHelper
{
public partial class UserFormBirthdayReminder : Form
{
private class BirthdayInfo
{
public ulong PersonId { get; set; }
public string Name { get; set; }
public int Age { get; set; }
}
private Dictionary<Type, List<BirthdayInfo>> birthdaysInfo;
public UserFormBirthdayReminder()
{
InitializeComponent();
SendQuery();
FillBoxes();
}
private void SendQuery()
{
using (var conn = ThisApplication.Instance.Session)
{
conn.BeginRead();
var query =
from person in conn.AllObjects<Person>()
where person.BirthDate != null
let day = person.BirthDate.Value.Day
let month = person.BirthDate.Value.Month
where day == DateTime.Today.Day &&
month == DateTime.Today.Month
orderby person.FirstName, person.LastName
group new BirthdayInfo
{
PersonId = person.Id,
Name = person.FirstName,
Age = person.Age.Value
}
by person.GetType();
birthdaysInfo = new Dictionary<Type, List<BirthdayInfo>>();
foreach (var group in query)
{
birthdaysInfo.Add(group.Key, group.ToList());
}
conn.Commit();

}
```

```
}
private void FillBoxes()
{
if (birthdaysInfo.ContainsKey(typeof(Employee)))
{
foreach (var employee in birthdaysInfo[typeof(Employee)])
{
EmployeesBox.Items.Add(
$"{employee.Name}: {employee.Age} years old");
}
}
}
public void ShowIfBirthdaysExists(bool errorBox = false)
{
if (birthdaysInfo.Count != 0)
{
ShowDialog();
}
else if (errorBox)
{
MessageBox.Show("Did not find people who have birthday today. ",
"",
MessageBoxButtons.OK,
MessageBoxIcon.Information);
return;
}
}
}
}
//------------------------UserFormAddBook.cs-------------------------------
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace LibraryHelper
{
public partial class UserFormAddBook : Form
{
private Book currentBook;
public UserFormAddBook()
{
InitializeComponent();
ageLimitBox.Maximum = DateTime.Today.Year;
}
private void OKButton_Click(object sender, EventArgs e)
{
using (var conn = ThisApplication.Instance.Session)
{
conn.BeginUpdate();
currentBook = new Book
{
Code = codeBox.Text,
Name = nameBox.Text,
Author = authorBox.Text,
AgeLimit = (int)yearBox.Value,
ReleaseYear = (int)ageLimitBox.Value
};
conn.Persist(currentBook);
```

```
conn.Commit();
}
Close();
}
public Book ShowAndReturnBook()
{
ShowDialog();
return currentBook;
}
private void CancelButton_Click(object sender, EventArgs e)
{
currentBook = null;
Close();
}
}
}
// ---------------------UserFormViewPerson.cs--------------------------
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Microsoft.VisualBasic;
using System.Globalization;
namespace LibraryHelper
{
public partial class UserFormViewPerson : Form
{
private Person currentPerson;
private DateTime? birthDate;
public UserFormViewPerson(Person person)
{
currentPerson = person;
InitializeComponent();
InitializeUserFormData();
Text = person.ToString(); //Change the caption of the window
UpdateEmailButton.Enabled = false;
RemoveEmailButton.Enabled = false;
UpdatePhoneButton.Enabled = false;
RemovePhoneButton.Enabled = false;
}
private void InitializeUserFormData()
{
firstNameBox.Text = currentPerson.FirstName;
lastNameBox.Text = currentPerson.LastName;
middleNameBox.Text = currentPerson.MiddleName ?? String.Empty;
birthDate = currentPerson.BirthDate;
birthDateBox.Text = birthDate.HasValue ?
birthDate.Value.ToString("dd/MM/yyyy",
CultureInfo.InvariantCulture) :
"---Empty---";
foreach (var email in currentPerson.Emails)
{
emailsBox.Items.Add(email);
}
foreach (var phone in currentPerson.Phones)
{
phonesBox.Items.Add(phone);
}
```

```
}
private bool CheckEmpty()
{
return firstNameBox.Text == String.Empty ||
lastNameBox.Text == String.Empty;
}
private void OKButton_Click(object sender, EventArgs e)
{
if (CheckEmpty())
{
MessageBox.Show("First name and last name must be filled. ",
"Error",
MessageBoxButtons.OK,
MessageBoxIcon.Error);
return;
}
using (var session = ThisApplication.Instance.Session)
{
session.BeginUpdate();
currentPerson = session
.AllObjects<Person>()
.FirstOrDefault(x => x.Id == currentPerson.Id);
currentPerson.FirstName = firstNameBox.Text;
currentPerson.LastName = lastNameBox.Text;
currentPerson.MiddleName =
middleNameBox.Text != String.Empty ?
middleNameBox.Text : null;
currentPerson.BirthDate = birthDate;
currentPerson.ClearPhones();
foreach (string phone in phonesBox.Items)
{
currentPerson.AddPhone(phone);
}
currentPerson.ClearEmails();
foreach (string email in emailsBox.Items)
{
currentPerson.AddEmail(email);
}
currentPerson.Update();
session.Commit();
}
Close();
}
private void AddEmailButton_Click(object sender, EventArgs e)
{
var email = Interaction.InputBox("Enter an email");
if (email == String.Empty)
{
return;
}
emailsBox.Items.Add(email);
}
private void UpdateEmailButton_Click(object sender, EventArgs e)
{
int index = emailsBox.SelectedIndex;
if (index < 0)
{
return;
}
var email = Interaction.InputBox(
Prompt: "Enter an email",
DefaultResponse: (string)emailsBox.Items[index]);
emailsBox.Items[index] = email;
```

```
}
private void RemoveEmailButton_Click(object sender, EventArgs e)
{
int index = emailsBox.SelectedIndex;
if (index < 0)
{
return;
}
emailsBox.Items.RemoveAt(index);
}
private void CancelButton_Click(object sender, EventArgs e)
{
Close();
}
private void ChangeBirthDateButton_Click(object sender, EventArgs e)
{
var window = new UserFormDatePicker(currentPerson.BirthDate);
birthDate = window.ShowAndChooseDate();
birthDateBox.Text = birthDate.HasValue ?
birthDate.Value.ToString("dd/MM/yyyy",
CultureInfo.InvariantCulture) :
"---Empty---";
}
private void AddPhoneButton_Click(object sender, EventArgs e)
{
var phone = Interaction.InputBox("Enter a phone");
if (phone == String.Empty)
{
return;
}
phonesBox.Items.Add(phone);
}
private void UpdatePhoneButton_Click(object sender, EventArgs e)
{
int index = phonesBox.SelectedIndex;
if (index < 0)
{
return;
}
var phone = Interaction.InputBox(
Prompt: "Enter a phone ",
DefaultResponse: (string)phonesBox.Items[index]);
phonesBox.Items[index] = phone;
}
private void RemovePhoneButton_Click(object sender, EventArgs e)
{
int index = phonesBox.SelectedIndex;
if (index < 0)
{
return;
}
phonesBox.Items.RemoveAt(index);
}
private void emailsBox_SelectedIndexChanged(object sender, EventArgs e)
{
UpdateEmailButton.Enabled = emailsBox.SelectedIndex >= 0;
RemoveEmailButton.Enabled = emailsBox.SelectedIndex >= 0;
}
private void phonesBox_SelectedIndexChanged(object sender, EventArgs e)
{
UpdatePhoneButton.Enabled = phonesBox.SelectedIndex >= 0;
RemovePhoneButton.Enabled = phonesBox.SelectedIndex >= 0;
}
```

```
}
}
// -------------------UserFormViewPersonList.cs-----------------------------
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace LibraryHelper
{
public partial class UserFormViewPersonsList : Form
{
public enum ViewMode
{
All, Employees, Clients
}
private List<Tuple<string, ulong>> personsData;
private ViewMode mode;
private bool Suits(Person person)
{
switch (mode)
{
case ViewMode.Employees:
return person is Employee;
case ViewMode.Clients:
return person is Client;
default:
return true;
}
}
public UserFormViewPersonsList(ViewMode mode = ViewMode.All)
{
this.mode = mode;
InitializeComponent();
ReinitializePersonsList();
}
private void ReinitializePersonsList()
{
personsBox.Items.Clear();
using (var conn = ThisApplication.Instance.Session)
{
conn.BeginRead();
var query =
from person in conn.AllObjects<Person>()
where Suits(person)
orderby person.FirstName, person.LastName
select Tuple.Create(person.ToString(), person.Id);
personsData = query.ToList();
foreach (var personData in personsData)
{
personsBox.Items.Add(personData.Item1);
}
conn.Commit();
}
}
private void ShowEmployeeButton_Click(object sender, EventArgs e)
{
var index = personsBox.SelectedIndex;
if (index < 0)
```

```
{
return;
}
var person = Database.GetById<Person>(personsData[index].Item2);
if (person != null)
{
var window = new UserFormViewPerson(person);
window.ShowDialog();
}
ReinitializePersonsList();
}
private void DeleteEmployeeButton_Click(object sender, EventArgs e)
{
int index = personsBox.SelectedIndex;
if (index < 0)
{
return;
}
using (var conn = ThisApplication.Instance.Session)
{
conn.BeginUpdate();
conn.AllObjects<Person>()
.FirstOrDefault(x => x.Id == personsData[index].Item2)?
.Unpersist(conn);
conn.Commit();
}
ReinitializePersonsList();
}
private void personsBox_DoubleClick(object sender, EventArgs e)
{
var index = personsBox.SelectedIndex;
if (index < 0)
{
return;
}
var person = Database.GetById<Person>(personsData[index].Item2);
if (person != null)
{
var window = new UserFormViewPerson(person);
window.ShowDialog();
}
ReinitializePersonsList();
}
}
}
// --------------------UserFormRegisterEmployee.cs-------------------------
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Globalization;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using VelocityDb.Session;
namespace LibraryHelper
{
public partial class UserFormRegisterEmployee : Form
{
private DateTime? birthDate;
public UserFormRegisterEmployee()
```

```
{
InitializeComponent();
birthDateBox.Text = "---Empty---";
birthDate = null;
}
private void OKButton_Click(object sender, EventArgs e)
{
if (CheckEmpty())
{
MessageBox.Show("Error * ",
"Error",
MessageBoxButtons.OK,
MessageBoxIcon.Error);
return;
}
if (!PasswordMatch())
{
MessageBox.Show("Passwords must match. ",
"Error",
MessageBoxButtons.OK,
MessageBoxIcon.Error);
return;
}
if (CheckDuplicateLogin())
{
MessageBox.Show("This login already exists. ",
"Error",
MessageBoxButtons.OK,
MessageBoxIcon.Error);
return;
}
StoreEmployeeObject();
Close();
}
private Employee MakeEmployeeObject()
{
var employee = new Employee
{
FirstName = firstNameBox.Text,
LastName = lastNameBox.Text,
MiddleName = middleNameBox.Text != String.Empty ?
middleNameBox.Text : null,
BirthDate = birthDate,
Login = loginBox.Text,
Password = passwordBox.Text
};
if (emailBox.Text != String.Empty)
{
employee.AddEmail(emailBox.Text);
}
if (phoneBox.Text != String.Empty)
{
employee.AddPhone(phoneBox.Text);
}
return employee;
}
private void StoreEmployeeObject()
{
using (var conn = ThisApplication.Instance.Session)
{
conn.BeginUpdate();
var employee = MakeEmployeeObject();
conn.Persist(employee);
```

```
ThisApplication.Instance.CurrentEmployee = employee;
conn.Commit();
}
}
private bool CheckDuplicateLogin()
{
bool result;
using (var conn = ThisApplication.Instance.Session)
{
conn.BeginRead();
result = conn
.AllObjects<Employee>()
.Any(x => x.Login == loginBox.Text);
conn.Commit();
}
return result;
}
private bool CheckEmpty()
{
return firstNameBox.Text == String.Empty ||
lastNameBox.Text == String.Empty ||
loginBox.Text == String.Empty ||
passwordBox.Text == String.Empty;
}
private bool PasswordMatch()
{
return passwordBox.Text == password1Box.Text;
}
private void CancelButton_Click(object sender, EventArgs e)
{
Close();
}
private void ChangeBirthDateButton_Click(object sender, EventArgs e)
{
var window = new UserFormDatePicker();
birthDate = window.ShowAndChooseDate();
birthDateBox.Text = birthDate.HasValue ?
birthDate.Value.ToString("dd/MM/yyyy",
CultureInfo.InvariantCulture) :
"---Empty---";
}
}
}
// ------------------------UserFormLogin.cs------------------------------
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace LibraryHelper
{
public partial class UserFormLogin : Form
{
private Employee logined;
public UserFormLogin()
{
InitializeComponent();
}
private void OKButton_Click(object sender, EventArgs e)
```

```
{
using (var conn = ThisApplication.Instance.Session)
{
conn.BeginRead();
logined = conn.AllObjects<Employee>().FirstOrDefault(x =>
{
return x.Login == loginBox.Text &&
x.Password == passwordBox.Text;
});
conn.Commit();
}
if (logined == null)
{
MessageBox.Show("Access denied. ",
"Error",
MessageBoxButtons.OK,
MessageBoxIcon.Error);
return;
}
Close();
}
public Employee ShowAndLogin()
{
ShowDialog();
var screenSize = Screen.PrimaryScreen.WorkingArea.Size;
Location = new Point
{
X = screenSize.Width / 2 - Width / 2,
Y = screenSize.Height / 2 - Height / 2
};
return logined;
}
private void CancelButton_Click(object sender, EventArgs e)
{
logined = ThisApplication.Instance.CurrentEmployee;
Close();
}
private void RegisterButton_Click(object sender, EventArgs e)
{
Hide();
var window = new UserFormRegisterEmployee();
window.ShowDialog();
if (ThisApplication.Instance.CurrentEmployee == null)
{
Show();
return;
}
logined = ThisApplication.Instance.CurrentEmployee;
Close();
}
}
}
// ----------------------UserFormViewDebtors.cs-------------------------
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace LibraryHelper
```

```
{
public partial class UserFormViewDebtors : Form
{
private List<DebtorInfo> info;
private class DebtorInfo
{
public ulong DebtorId { get; set; }
public ulong BookId { get; set; }
public DateTime StartDate { get; set; }
public string BookName { get; set; }
public string DebtorName { get; set; }
public string EmployeeName { get; set; }
};
public UserFormViewDebtors()
{
InitializeComponent();
using (var conn = ThisApplication.Instance.Session)
{
conn.BeginRead();
IEnumerable<DebtorInfo> query =
from book in conn.AllObjects<Book>()
where book.CurrentOwner != null
//where book.Durability.Value > new TimeSpan(90, 0, 0, 0, 0)
select new DebtorInfo
{
DebtorId = book.CurrentOwner.Id,
BookId = book.Id,
StartDate = book.Owners.Last().StartDate,
BookName = book.Name,
DebtorName = book.CurrentOwner.ToString(),
EmployeeName = book.Owners.Last().Employee.ToString() ??
"---Empty---"
};
query = query.ToList();
info = (List<DebtorInfo>)query;
conn.Commit();
foreach (var elem in info)
{
infoBox.Items.Add(new ListViewItem(new string[]
{
elem.BookName,
elem.DebtorName,
elem.StartDate.ToString("dd/MM/yyyy"),
elem.EmployeeName
}));
}
}
}
private void ViewClientButton_Click(object sender, EventArgs e)
{
try
{
var index = infoBox.SelectedIndices[0];
if (index < 0)
{
return;
}
var person = Database.GetById<Person>(info[index].DebtorId);
var window = new UserFormViewPerson(person);
window.ShowDialog();
}
catch(ArgumentOutOfRangeException)
{
```

```
return;
}
}
private void BookInfoButton_Click(object sender, EventArgs e)
{
try
{
var index = infoBox.SelectedIndices[0];
if (index < 0)
{
return;
}
var book = Database.GetById<Book>(info[index].BookId);
var window = new UserFormViewBook(book);
window.ShowDialog();
}
catch(ArgumentOutOfRangeException)
{
return;
}
}
}
}
```

// index.php – привітальна сторінка сайту

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Pandya | Зручне керування проектами</title>
        <link href="assets/css/libraries/bootstrap.min.css" rel="stylesheet">
        <link href="assets/css/index.css" rel="stylesheet">
        <link rel="shortcut icon" type="image/png" href="img/favicon.png"/>
    </head>
    <body>
        <div class="container-fluid">
            <div class="row">
                <div class="col-md-12">
                    <div class="logo">
                        <img src="assets/img/logo.png"/>
                    </div>
                </div>
            </div>
            <div class="row">
                <div class="col-md-12">
                    <div class="heading">
                        <h1>
                            <span style="display: block; width: 100%; text-
align: center;"><span style="font-weight: 500">Pandya</span> вітає
Вас!</span>
                            <span style="font-weight: 500">Pandya</span> –
Ваш цифровий помічник. Просте та зручне управління бізнес-проектами. Швидкий
та зручний доступ до Ваших даних: <span style="font-style: italic">заміток,
файлів, документів</span>. А найголовніше – БЕЗКОШТОВНО.<br>
                            <br><span class="text_center">Приєднуйтесь до
нас!</span>
                        </h1>
                    </div>
                </div>
            </div>
            <div class="row">
```

```
                <div class="col-md-12">
                    <div class="login_buttons">
                        <ul class="login_list">
                            <li><a href="register"
id="register_btn">Реєстрація</a></li>
                            <li>або</li>
                            <li><a href="login"
id="login_btn">Увійти</a></li>
                        </ul>
                    </div>
                </div>
            </div>
        </div>
        <script src="assets/js/libraries/jquery-2.1.4.js"></script>
        <script src="assets/js/libraries/bootstrap.min.js"></script>
    </body>
</html>
```

//index.css – файл опису стилів привітальної сторінки сайту

```
@import url("header.css");

.logo {
    margin-top: 100px;
}

.logo img {
    display: block;
    margin: 0 auto;
}

.heading {
    margin-top: 20px;
}

.heading h1 {
    margin: 0 auto;
    max-width: 900px;
    line-height: 1.2em;
    font-size: 2em;
    font-weight: 300;
    font-family: 'open sans', arial , sans-serif;
    text-align: justify;
}

.text_center {
    display: block;
    width: 100%;
    text-align: center;
}

.login_list {
    margin-top: 30px;
    padding: 0px;
    font-size: 1.5em;
    font-weight: 300;
    font-family: 'open sans', arial , sans-serif;
    text-align: center;
    list-style-type: none;
}

.login_list li {
```

```
    margin: 20px 0;
}

#register_btn, #login_btn {
    display: inline-table;
    width: 7.5em;
    height: 2.7em;
    line-height: 2.8em;
    margin: auto;
    color: #FFFFFF;
    border-bottom: 4px solid #008800;
    border-radius: 5px;
    background: #19AA00;
    text-decoration: none;
}

#register_btn:active, #login_btn:active {
    margin-top: 4px;
    border-bottom: 0px solid;
}
```

//header.css – файл опису стилів для заднього фону сторінок

```
@import url("reset.css");
@import url("fonts/google-fonts.css");

html {
  background: url(../img/leafs.png) no-repeat center center fixed;
  -webkit-background-size: cover;
  -moz-background-size: cover;
  -o-background-size: cover;
  background-size: cover;
}

body {
    background-color: transparent;
}
```

//.htaccsess – файл додаткової конфігурації веб-сервера Apache

```
AddDefaultCharset utf-8

ErrorDocument 404 /error.php

RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME}.php -f
RewriteRule ^(.*)$ $1.php [NC,L]
```

//register.php – сторінка реєстрації

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Pandya | Зручне керування проектами</title>
        <link href="assets/css/libraries/bootstrap.min.css" rel="stylesheet">
        <link href="assets/css/register.css" rel="stylesheet">
        <link rel="shortcut icon" type="image/png" href="img/favicon.png"/>
    </head>
    <body>
        <div class="container-fluid">
            <div class="row">
                <div class="col-md-12">
                    <div class="logo">
                        <a href="/"><img id="logo-image"
src="assets/img/logo_short.png"/></a>
                    </div>
                </div>
            </div>
            <div class="row">
                <div class="registration_form">
                    <h2 class="heading">Реєстрація</h2>
                    <div class="error_notification">
                        <span class="notification_message"></span>
                    </div>
                    <div class="form">
                        <form method="post" action="/register" enctype=
"multipart/form-data" id="file_upload" name="file_upload" onsubmit="return
validate();">
                            <div class="user_photo_area">
                                <label id="photo-label" for="photo_field">
                                    Ваше фото
                                </label>
                                <span class="photo_message"></span>
                                <span class="btn btn-file">
                                    Обрати.. <input id="photo_field"
name="file" type="file">
                                </span>
                                <input type="text" id="filename"
name="filename" />
                            </div>
                            <div class="login_area">
                                <label for="login_field">
                                    Ваше ім'я
                                </label>
                                <span class="login_message"></span>
                                <div class="field">
                                    <input type="text" name="user_login"
id="login_field" autocomplete="off"/>
                                </div>
                            </div>
                            <div class="email_area">
                                <label for="email_field">
                                    Ваш E-mail
                                </label>
                                <span class="email_message"></span>
                                <div class="field">
                                    <input type="email" name="user_email"
id="email_field" autocomplete="off"/>
                                </div>
```

```
                            </div>
                            <div class="password_area">
                                <label for="password_field">
                                    Пароль
                                </label>
                                <span class="password_message"></span>
                                <div class="field">
                                    <input type="password"
name="user_password" id="password_field" autocomplete="off"/>
                                </div>
                            </div>
                            <div class="register_btn">
                                <input type="submit" id="register"
name="signup" value="Зареєструватись" />
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        <script src="assets/js/libraries/jquery-2.1.4.js"></script>
        <script src="assets/js/libraries/bootstrap.min.js"></script>
        <script src="assets/js/register.js"></script>
    </body>
</html>
<?php

session_start();

$content = json_decode(file_get_contents("config/language.json"));
$lang = $_SESSION['lang'];
if($lang == '' || $lang == 'ua'){
    $words = $content->language[0]->ua[0];
} else if($lang == 'uk'){
    $words = $content->language[1]->uk[0];
} else {
    $words = $content->language[2]->ru[0];
}

$config = include "config/db.php";

if (isset($_POST['signup'])) {
    $login = $_POST['user_login'];
    $password = password_hash($_POST['user_password'], PASSWORD_DEFAULT);
    $email = $_POST['user_email'];

    $file_exts = array("jpg", "jpeg", "png");
    $temp = explode(".", $_FILES["file"]["name"]);
    $upload_exts = end($temp);
    $file_name = md5(date('m/d/Y h:i:s a', time()) . $_FILES["file"]["name"])
. "." . $upload_exts;
    if ((($_FILES["file"]["type"] == "image/jpeg")
        || ($_FILES["file"]["type"] == "image/png")
        || ($_FILES["file"]["type"] == "image/pjpeg"))
        && ($_FILES["file"]["size"] < 8000000)
        && in_array($upload_exts, $file_exts)) {

        if ($_FILES["file"]["error"] > 0) {
            echo "<script> showError(\"Помилка! Код: " .
$_FILES["file"]["error"] . "\", false);</script>";
        } else {
            if (file_exists("/images/user_avatars/" . $file_name)) {
                echo "<script> showError(\"Файл з назвою " .
$_FILES["file"]["name"] . " вже існує.\", false);</script>";
```

```
            } else {
                move_uploaded_file($_FILES["file"]["tmp_name"],
"images/user_avatars/" . $file_name);
                try {
                    $dbh = new
PDO("mysql:host=$config[host];dbname=$config[db_name]", $config['username'],
$config['password']);
                    $sth = $dbh->prepare("INSERT INTO users (username, email,
password, active, photo) values ('$login', '$email', '$password', 0,
'$file_name')");
                    $sth->execute();

                    $userid = $dbh->lastInsertId();

                    $key = $login . $email . date('Y-m-d H:i:s');
                    $key = md5($key);

                    $sth = $dbh->prepare("INSERT INTO confirm VALUES (NULL,
'$userid', '$key', '$email', 0, 1)");
                    $sth->execute();

                    require("assets/class.phpmailer.php");

                    $mail = new PHPMailer();
                    $mail->CharSet = 'UTF-8';
                    $mail->From     = "noreply@pandya.esy.es";
                    $mail->FromName = "Pandya";
                    $mail->AddAddress($email);
                    $mail->Subject  = $words->email_subject;
                    $mail->Body     = $words->email_body .
"\n\nhttp://pandya.esy.es/confirm.php?username=" . $login . "&email=" .
$email . "&secure_key=" . $key;

                    if(!$mail->Send()) {
                        echo 'Mailer error: ' . $mail->ErrorInfo;
                    } else {
                        echo "<script>
                            showError(\"Протягом хвилини на " . $email . "
буде надіслано листа з кодом підтвердження реєстрації!\", true);
                            $(\"#login_field\").val('');
                            $(\"#email_field\").val('');
                            $(\"#password_field\").val('');
                        </script>";
                    }
                } catch (PDOException $e){
                    echo $e->getMessage();
                }
                $dbh = null;
            }
        }
    } else {
        echo "<script> showError(\"Недопустимий файл.\", false);</script>";
    }
}

?>
```

//register.js – скрипт сторінки реєстрації

```
var can_register, fileinput;

$(function() {

      can_register = { login: true, email: true, pass: true, photo: true };

      $(document).on('change', '.btn-file :file', function() {
        fileinput = this,
      input = $(this),
        numFiles = input.get(0).files ? input.get(0).files.length : 1,
        label = input.val().replace(/\\/g, '/').replace(/.*\//, '');
          input.trigger('fileselect', [numFiles, label]);
      });

      function readURL(input) {
          if (input.files && input.files[0]) {
              var reader = new FileReader();
              reader.onload = function (e) {
                  $('#logo-image').attr('src', e.target.result);
              }
              reader.readAsDataURL(input.files[0]);
          }
      }

      $('.btn-file :file').on('fileselect', function(event, numFiles, label) {
            var _validFileExtensions = [".jpg", ".jpeg", ".png"];
            var sFileName = $("#photo_field").val();
            if (sFileName.length > 0) {
            var blnValid = false;
            for (var j = 0; j < _validFileExtensions.length; j++) {
                var sCurExtension = _validFileExtensions[j];
                if (sFileName.substr(sFileName.length - sCurExtension.length,
sCurExtension.length).toLowerCase() == sCurExtension.toLowerCase()) {
                    blnValid = true;
                    break;
                }
            }
            $("#filename").val(label);
            if (!blnValid) {
                $("#photo-label").css({"display": "inline-block"});
                      $(".photo_message").html("Файл не є зображенням!");
                      $('#logo-image').attr('src',
"assets/img/logo_short.png");
                      can_register.photo = false;
            } else {
                  $("#photo-label").css({"display": "block"});
                  $(".photo_message").html("");
                  $(".error_notification").css({"height": "0px", "background-
color": "#fff", "border-left": "0px solid #fff"});
                        $(".notification_message").html("");
                  readURL(this);
            }
        }
    });

      $("#login_field").attr("maxlength", 40);
      $("#email_field").attr("maxlength", 40);
      $("#password_field").attr("maxlength", 40);

      $("#login_field").keyup(function(event){
            var login = $("#login_field").val();
```

```
            if(login.length < 4){
                  $(".login_message").html("Занадто коротке ім'я!");
                  can_register.login = false;
                  return false;
            }
            if(login.length > 15){
                  $(".login_message").html("Занадто довге ім'я!");
                  can_register.login = false;
                  return false;
            }
            if(!/^[a-zA-Z0-9_-]{3,15}$/.test(login)){
                  $(".login_message").html("Некоректні символи в імені!");
                  can_register.login = false;
                  return false;
            }

            check('login');
            if(event.keyCode == '13'){
                  $("#register").click();
            }
      });

      $("#email_field").keyup(function(event){
            var email = $("#email_field").val();


      if(!/^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)*)|(\".+\"))@((\
[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-
Z]{2,}))$/.test(email)){
                  $(".email_message").html("Не дійсний E-mail!");
                  can_register.email = false;
                  return false;
            }

            check('email');
            if(event.keyCode == '13'){
                  $("#register").click();
            }
      });

      $("#password_field").keyup(function(event){
            var pass = $("#password_field").val();

            if(pass.length < 6){
                  $(".password_message").html("Занадто короткий пароль!");
                  can_register.pass = false;
            } else {
                  $(".password_message").html('');
                  can_register.pass = true;
            }

            if(event.keyCode == '13'){
                  $("#register").click();
            }
      });

      function check(field){
            var value = $("#" + field + "_field").val(),
                  errors = {login: "Логін вже зайнятий!", email: "E-mail вже
зайнятий!"};

            $.ajax({
```

```
                type: "POST",
                url: "ajax/register.php",
                data: {
                      check : true,
                      field : field,
                      value : value
                }
          })
          .done(function(msg){
                if(msg == 'false'){
                      $("." + field + "_message").html(errors[field]);
                      can_register[field] = false;
                } else {
                      $("." + field + "_message").html('');
                      can_register[field] = true;
                }
          });
    }

});

function showError(text, flag) {
    if (flag) {
        $(".error_notification").css({"height": "70px", "background-
color": "#D8FFD5", "border-left": "7px solid #78FF6E"});
    } else {
        $(".error_notification").css({"height": "50px", "background-
color": "#F9D8D3", "border-left": "7px solid #F9756B"});
    }
    $(".notification_message").html(text);
}

function validate() {
    var login = $("#login_field").val(),
        email = $("#email_field").val(),
        pass = $("#password_field").val(),
        photo = $("#photo_field").val();

    if(login == '' || email == '' || pass == '' || photo == '') {
        showError("Заповніть порожні поля!", false);
        return false;
    }
    if(!(can_register.login && can_register.email && can_register.pass &&
can_register.photo)) {
        showError("Перевірте заповнені поля на правильність!", false);
        return false;
    }
    $(".error_notification").css({"height": "0px"});
    return true;
}
```

//register.css – стилі сторінки реєстрації

```
@import url("header.css");

.logo {
      margin-top: 2em;
}

.logo img {
    display: block;
    margin: 0 auto;
}

#logo-image {
    width: 200px;
    height: 200px;
    max-height: 200px;
    border-radius: 100px;
    -webkit-border-radius: 100px;
    -moz-border-radius: 100px;
    object-fit: cover;
}

.registration_form{
      width: 1000px;
      margin: 0 auto;
}

.heading{
      padding: 0px 0 20px 0;
      font-size: 55px;
      font-weight: 300;
      font-family: 'open sans', arial , sans-serif;
      text-align: center;
      color: #000;
      vertical-align: bottom;
}

form > div{
      width: 40%;
      margin: auto;
      padding-top: 25px;
}

#login_field, #email_field, #password_field{
      width: 100%;
      padding: 0;
      height: 40px;
      text-indent: 10px;
      border-radius: 5px;
      font-size: 22px;
      font-weight: 300;
      font-family: 'open sans', arial , sans-serif;
      color: #000;
      border: 1px solid #000;
}

#password_field{
      font-weight: 600;
}

#login_field:focus, #email_field:focus, #password_field:focus{
```

```
      border: 1px solid #19AA00;
      outline: 0;
}

.form label{
      display: inline-block;
      cursor: pointer;
      font-size: 28px;
      font-weight: 300;
      font-family: 'open sans', arial , sans-serif;
      color: #000;
      text-indent: 10px;
}

.field{
      margin-top: 10px;
}

.photo_message {
    position: relative;
    left: 5%;
    font-size: 19px;
    font-weight: 300;
    font-family: 'open sans', arial , sans-serif;
    color: #F9756B;
}

.login_message, .email_message, .password_message{
      position: relative;
      float: right;
      top: 5px;
      right: 2%;
      font-size: 19px;
      font-weight: 300;
      font-family: 'open sans', arial , sans-serif;
      color: #F9756B;
      z-index: -1;
}

#register{
      width: 100%;
      height: 60px;
      font-size: 30px;
      font-weight: 300;
      font-family: 'open sans', arial , sans-serif;
      color: #fff;
      padding-bottom: 5px;
      text-indent: 0px;
      border: 0px;
      border-radius: 5px;
      background-color: #19AA00;
      -webkit-box-shadow: 0px 4px 0px #008800;
      -moz-box-shadow:    0px 4px 0px #008800;
      box-shadow:         0px 4px 0px #008800;
      outline: 0;
}

#register:active{
      outline: 0;
    margin-top: 4px;
    box-shadow: none;
}
```

```
.error_notification{
      width: 40%;
      margin: auto;
      display: table;
      border-radius: 5px;
      font-size: 17px;
      font-weight: 300;
      font-family: 'open sans', arial , sans-serif;
      box-sizing: border-box;
      -moz-box-sizing: border-box;
      -webkit-box-sizing: border-box;
      -webkit-transition: background-color .5s, height .5s;
      transition: background-color .5s, height .5s;
}

.notification_message{
      display: table-cell;
      vertical-align: middle;
      padding-left: 20px;
}

.btn-file {
    position: relative;
    top: -0.1em;
    display: inline;
    width: 35%;
    overflow: hidden;
    border-radius: 5px;
    background-color: #19AA00;
    border-bottom: 4px solid #008800;
    font-size: 20px;
    font-weight: 300;
    font-family: 'open sans', arial , sans-serif;
    color: #fff;
}

.btn-file:hover {
    color: #fff;
}

.btn-file:active {
    top: 0.1em;
    border-bottom: 0px solid;
}

.btn-file input[type=file] {
    position: absolute;
    top: 0;
    right: 0;
    min-width: 100%;
    min-height: 100%;
    font-size: 30px;
    text-align: right;
    filter: alpha(opacity=0);
    opacity: 0;
    outline: none;
    background: white;
    cursor: inherit;
    display: block;
}

#filename {
    position: relative;
```

```
    top: 0.2em;
    width: 70%;
    margin-left: 2%;
    height: 40px;
    text-indent: 10px;
    border-radius: 5px;
    font-size: 17px;
    font-weight: 300;
    font-family: 'open sans', arial , sans-serif;
    color: #000;
    border: 1px solid #000;
}

#photo-label {
    display: block;
    cursor: pointer;
    margin-bottom: 0.5em;
    font-size: 28px;
    font-weight: 300;
    font-family: 'open sans', arial , sans-serif;
    color: #000;
    text-indent: 10px;
}
```

//db.php – конфігурація БД

```
<?php

return array("host" => "mysql.hostinger.com.ua",
             "username" => "u362054579_root",
             "password" => "123456",
             "db_name" => "u362054579_pand");

?>
```

// recover.js – скрипт сторінки відновлення паролю

```
$(document).ready(function(){
	var can_register = {email: true, pass: true, pass_length: true};

	$("#email_field").attr("maxlength", 40);
	$("#new_password_field").attr("maxlength", 40);
	$("#new_password2_field").attr("maxlength", 40);

	$("#recover").click(function(){
		var email = $("#email_field").val();

		if(email == '')
		{
			$(".error_notification").css({"height": "50px", "background-
color": "#F9D8D3", "border-left": "7px solid #F9756B"});
			$(".notification_message").html(register_empty_field);
			return false;
		}

		if(!(can_register.email && can_register.pass)){
			$(".error_notification").css({"height": "50px", "background-
color": "#F9D8D3", "border-left": "7px solid #F9756B"});
			$(".notification_message").html(register_check_fields);
			return false;
		}

		$(".error_notification").css({"height": "0px", "background-color":
"#fff", "border-left": "0px solid #fff"});
		$(".notification_message").html("");

		$.ajax({
			type: "POST",
			url: "ajax/check_email.php",
			data: {
				recover : true,
				email: email
			}
		})
		.done(function(msg){
			if(msg == 'success'){
				$(".error_notification").css({"height": "70px",
"background-color": "#D8FFD5", "border-left": "7px solid #78FF6E"});
				$(".notification_message").html(register_minute_first
+ email + recover_minute_second);
				$("#email_field").val('');
			}
		});
	});

	$("#email_field").keyup(function(event){
		var email = $("#email_field").val();


	if(!/^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)*)|(\".+\"))@((\
[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-
Z]{2,}))$/.test(email)){
			$(".error_notification").css({"height": "50px", "background-
color": "#F9D8D3", "border-left": "7px solid #F9756B"});
			$(".notification_message").html(register_not_valid_email);
			can_register.email = false;
			return false;
```

```
            }

            $(".error_notification").css({"height": "0px", "background-color":
"#fff", "border-left": "0px solid #fff"});
            $(".notification_message").html("");
            can_register.email = true;

            check('email');
            if(event.keyCode == '13'){
                  $("#recover").click();
            }
      });

      function check(field){
            var value = $("#" + field + "_field").val(),
                  errors = {email: recover_email_not_exists};

            $.ajax({
                  type: "POST",
                  url: "ajax/check_email.php",
                  data: {
                        check_registered_email : true,
                        field : field,
                        value : value
                  }
            }).done(function(msg){
                  if(msg == 'false'){
                        $(".error_notification").css({"height": "50px",
"background-color": "#F9D8D3", "border-left": "7px solid #F9756B"});
                        $(".notification_message").html(errors[field]);
                        can_register[field] = false;
                  } else {
                        $(".error_notification").css({"height": "0px",
"background-color": "#fff", "border-left": "0px solid #fff"});
                        $(".notification_message").html('');
                        can_register[field] = true;
                  }
            });
      }

      $("#new_password_field").keyup(function(){
            var new_pass = $("#new_password_field").val();

            check_passwords();

            if(new_pass.length < 6){
                  $(".new_password_message").html(register_short_password);
                  can_register.pass_length = false;
            } else {
                  $(".new_password_message").html('');
                  can_register.pass_length = true;
            }

            if(event.keyCode == '13'){
                  $("#change_pass").click();
            }
      });

      $("#new_password2_field").keyup(function(){
            var new_pass2 = $("#new_password2_field").val();

            check_passwords();
```

```
            if(new_pass2.length < 6){
                  $(".new_password2_message").html(register_short_password);
                  can_register.pass_length = false;
            } else {
                  $(".new_password2_message").html('');
                  can_register.pass_length = true;
            }

            if(event.keyCode == '13'){
                  $("#change_pass").click();
            }
      });

      $("#change_pass").click(function(){
            var new_pass = $("#new_password_field"),
                  new_pass2 = $("#new_password2_field");

            if(new_pass.val() == '' || new_pass2.val() == ''){
                  $(".error_notification").css({"height": "50px", "background-
color": "#F9D8D3", "border-left": "7px solid #F9756B"});
                  $(".notification_message").html(register_empty_field);
                  return false;
            }

            if(!(can_register.pass && can_register.pass_length)){
                  $(".error_notification").css({"height": "50px", "background-
color": "#F9D8D3", "border-left": "7px solid #F9756B"});
                  $(".notification_message").html(register_check_fields);
                  return false;
            }

            $(".error_notification").css({"height": "0px", "background-color":
"#fff", "border-left": "0px solid #fff"});
            $(".notification_message").html("");
            $.ajax({
                  type: "POST",
                  url: "ajax/check_email.php",
                  data: {
                        change_pass : true,
                        email: email,
                        password: new_pass.val()
                  }
            })
            .done(function(msg){
                  if(msg == 'success'){
                        $(".error_notification").css({"height": "70px",
"background-color": "#D8FFD5", "border-left": "7px solid #78FF6E"});

      $(".notification_message").html(recover_password_changed);
                        $("#email_field").val('');
                  }
            });
      });

      function check_passwords(){
            var new_pass = $("#new_password_field"),
                  new_pass2 = $("#new_password2_field");

            if(new_pass.val() != new_pass2.val()){
                  $(".error_notification").css({"height": "50px", "background-
color": "#F9D8D3", "border-left": "7px solid #F9756B"});

      $(".notification_message").html(recover_password_dont_match);
```

```
                can_register.pass = false;
            } else {
                $(".error_notification").css({"height": "70px", "background-
color": "#D8FFD5", "border-left": "7px solid #78FF6E"});
                $(".notification_message").html(recover_password_match);
                can_register.pass = true;
            }
        }

});
```