МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

ЛЕВОНЮК Наталія Михайлівна

Алгоритми опрацювання даних витрат енергоносіїв "інтелектуального" міста / Data processing algorithms of cost energy " smart " city

спеціальність: 123 - Комп'ютерна інженерія
магістерська програма - Комп'ютерна інженерія

Магістерська робота

Виконала студентка групи КІм-21
Н. М. Левонюк (Бакалюк)
_____

Науковий керівник:
д.т.н., професор, В. М. Теслюк
_____

Магістерську роботу допущено
до захисту:

"8" 01 2018 р.

Завідувач кафедри
_____ О. М. Березький

ТЕРНОПІЛЬ - 2018

Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії
Освітній ступінь «магістр»
спеціальність: 123 - Комп'ютерна інженерія
магістерська програма - Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.М.Березький
"13" 11 2017р.

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Левонюк Наталії Михайлівні
(прізвище, ім'я, по батькові)

1. Тема магістерської роботи «Алгоритми опрацювання даних витрат енергоносіїв "інтелектуального" міста»/Data processing algorithms of cost energy "smart" city
керівник роботи д.т.н., проф., В.М. Теслюк
затверджені наказом по університету від 6 листопада 2017 р. № 1367.

2. Строк подання студентом роботи «22» січня 2018 року

3. Вихідні дані до магістерської роботи
Об'єкт дослідження – процес опрацювання даних витрат енергоносіїв.
Предмет дослідження – методи і алгоритми опрацювання даних витрат енергоносіїв «інтелектуального міста».

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- проаналізувати особливості алгоритмів опрацювання даних;
- розробити ефективну колірну ознаку;
- розробити структуру системи опрацювання даних витрат та відповідні алгоритми їх обчислення;
- розробити алгоритми опрацювання даних витрат енергоносіїв;
- здійснити програмну реалізацію розроблених алгоритмів;
- провести тестування розроблених алгоритмів.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):
- структурна схема програми опрацювання даних витрат енергоносіїв ;
- блок-схема алгоритму підключення до бази даних;
- блок-схема алгоритму опрацювання даних;
- структурна схема бази даних.

6. Консультанти розділів магістерської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Антиплагіат	Мельник Г.М., доцент		
Нормо-контроль	Гураль І. В., викладач		

7. Дата видачі завдання « 8 » листопада 2017 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз інтелектуального міста структура витрат та аналіз енергоносіїв, алгоритмів опрацювання даних	7.11.2017 – 20.11.2017	
2	Розробка алгоритмів опрацювання даних витрат енергоносіїв	21.11.2017 – 12.12.2017	
3	Програмна реалізація алгоритмів опрацювання даних витрат енергоносіїв	13.12.2017 – 15.01.2018	
4	Нормоконтроль, попередній захист	16.01.2018 – 3.02.2018	
5	Захист	4.02.2018	

Студент _____ Левонюк Н.М.
(підпис)

Керівник магістерської роботи _____ д.т.н., проф., В.М. Теслюк
(підпис)

РЕЗЮМЕ

Магістерська робота на тему “ Алгоритми опрацювання даних витрат енергоносіїв “інтелектуального” міста ” на здобуття освітнього ступня “Магістр” зі спеціальності 123 - Комп’ютерна інженерія, магістерська програма - Комп’ютерна інженерія містить __ сторінок пояснюючої записки, __ рисунків, __ таблицю, __ додатків, з яких __ – графічні схеми формату А3.

Метою магістерської роботи є розробка алгоритмів опрацювання даних витрат енергоносіїв «інтелектуального» міста.

Методи досліджень базуються на теорії алгоритмів (для аналізу розроблених методів та алгоритмів), теорії нечіткої логіки (для визначення оптимальних алгоритмів розробки та доступу до баз даних), технологій структурного та об’єктно-орієнтованого програмування (для розробки системи моніторингу реєстрації абітурієнтів).

(для розробки системи опрацювання даних витрат енергоносіїв «інтелектуального» міста).

В магістерській роботі на основі аналізу показників енергоносіїв обґрунтовується використання відповідного алгоритму опрацювання даних з метою отримання системи опрацювання даних яка демонструє використання енергоносіїв для моніторингу, розрахунку та економії. .

Проведено тестування розробленої системи на базі реальних показників енергоносіїв.

Розроблений програмний продукт є ефективним засобом з простим інтерфейсом, що дозволяє вирішувати проблему опрацювання даних витрат енергоносіївї.

Ключові слова: ІНТЕЛЕКТУАЛЬНЕ МІСТО, АЛГОРИТМИ ОПРАЦЮВАННЯ ДАНИХ, ЕНЕРГОНОСІЇ.

RESUME

Master's work a theme the “"Algorithms for processing energy data data" intellectual "city" for obtaining an educational foot "Master" in specialty 123 - Computer engineering, master's program - Computer engineering contains __ pages of explanatory note, __ figures, __ tables, __ applications, __ of which are A3 format graphic circuits.

The aim of the master's thesis is to develop algorithms for processing energy data data of the "intellectual" city.

Research methods are based on the theory of algorithms (for the analysis of the developed methods and algorithms) , theory of fuzzy logic (algorithms to determine the optimal design and database access) technologies structural and object- oriented programming (for the development of a system for processing data on energy costs of the "smart" city).

In master's work on the basis of the analysis of energy carriers, the use of the appropriate data processing algorithm is grounded in order to obtain a data processing system that demonstrates the use of energy carriers for monitoring, calculation and economy. .

The tested system was tested based on real energy indicators.

The developed software product is an effective tool with a simple interface, which allows solving the problem of processing energy consumption data.

Key words: INTELLECTUAL CITY, DIGITAL DEGREE ALGORITHMS, ENERGY.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

PNG	–	Portable Network Graphics
JPEG		Joint Photographic Experts Group
PDF	–	Portable Document Format
SVG	–	Scalable Vector Graphics

ВСТУП

«Інтелектуальне » місто – визначається також як «місто знань», «цифрове місто», «кібермісто», «екомісто» – в залежності від цілей міського планування. Вони ведуть постійний моніторинг найважливіших об’єктів інфраструктури – автомобільних доріг, мостів, тунелів, залізниць, метро, аеропортів, морських портів, систем зв’язку, водопостачання, енергопостачання, найважливіших будівель з метою оптимального розподілу ресурсів і забезпечення безпеки.[1].Метою дослідження є розробка системи опрацювання даних енергоносіїв «інтелектуального» міста, для їх систематизації та економії, розробка моделі системи та ефективних алгоритмів опрацювання даних та роботи програмного забезпечення, призначеного для обробки отриманих показників енергоносіїв з об’єктів “інтелектуального” міста.

Системи збору та обробки показників енергоносіїв є телеметричними системами, що призначені для отримання, перетворення, зберігання, передачі, обробки та відображення інформації про використання енергоносіїв із віддалених об`єктів, без присутності спостерігача [2]. Однією з проблем систем обробки показників енергоносіїв є:

- швидкодія – дані з лічильників можуть приходити з запізненням і цим самим буде заповільнюватись робота всієї системи;
- точність – при надходженні даних є велика ймовірність неточності показника;
- обчислення не всіх даних- можливе при несвоєчасному поданні даних і їх не береться до уваги, та нараховуються такі ж показники які були в аналогічний місяць попереднього року.

Алгоритм роботи системи визначається її структурою та необхідною функціональністю.

1 АНАЛІЗ ІНТЕЛЕКТУАЛЬНОГО МІСТА СТРУКТУРА ВИТРАТ ТА АНАЛІЗ ЕНЕРГОНОСІЇВ, АЛГОРИТМІВ ОПРАЦЮВАННЯ ДАНИХ

1.1 Аналіз поняття інтелектуального міста

Розумний місто можна визначити, як місто знань , цифрове місто, кібермісто або екомісто - Залежно від цілей міського планування. Розумні міста в економічному та соціальному аспектах спрямовані в майбутнє. Вони ведуть постійний моніторинг найважливіших об'єктів інфраструктури - автомобільних доріг, мостів, тунелів, залізниць, метро, аеропортів, морських портів, систем зв'язку, водопостачання, енергопостачання, навіть найважливіших будівель - з метою оптимального розподілу ресурсів і забезпечення безпеки. Вони постійно нарощують число надаваних населенню послуг, забезпечуючи стійке середовище, яка сприяє благополуччю і збереженню здоров'я городян. Основу цих послуг становить інфраструктура інформаційно-комунікаційних технологій (ІКТ).

У структурному аспекті розумний місто - це система взаємодіючих систем. Така взаємодія величезного числа систем вимагає відкритості та стандартизації, які є основними принципами створення розумних міст. Проект розумного міста, в якому відсутня відкритість і стандартизація, дуже скоро стане громіздким і дорогим. До складових розумний місто технологій відносяться високошвидкісні оптичні, сенсорні, провідні та безпровідні мережі, необхідні для реалізації таких переваг, як забезпечувані завдяки інтелектуальним транспортним системам, розумним електромереж і організації домашніх мереж.

Головна відмінність розумного міста від міста традиційного полягає в характері взаємин з городянами. У звичайному місті послуги на основі ІКТ не можуть так само гнучко реагувати на зміни економічних, культурних та соціальних умов, як послуги в розумному місті. Таким чином, розумний місто насамперед орієнтований на людину, базується на інфраструктурі ІКТ та

безперервному міському розвитку при постійному обліку вимог екологічної та економічної стійкості

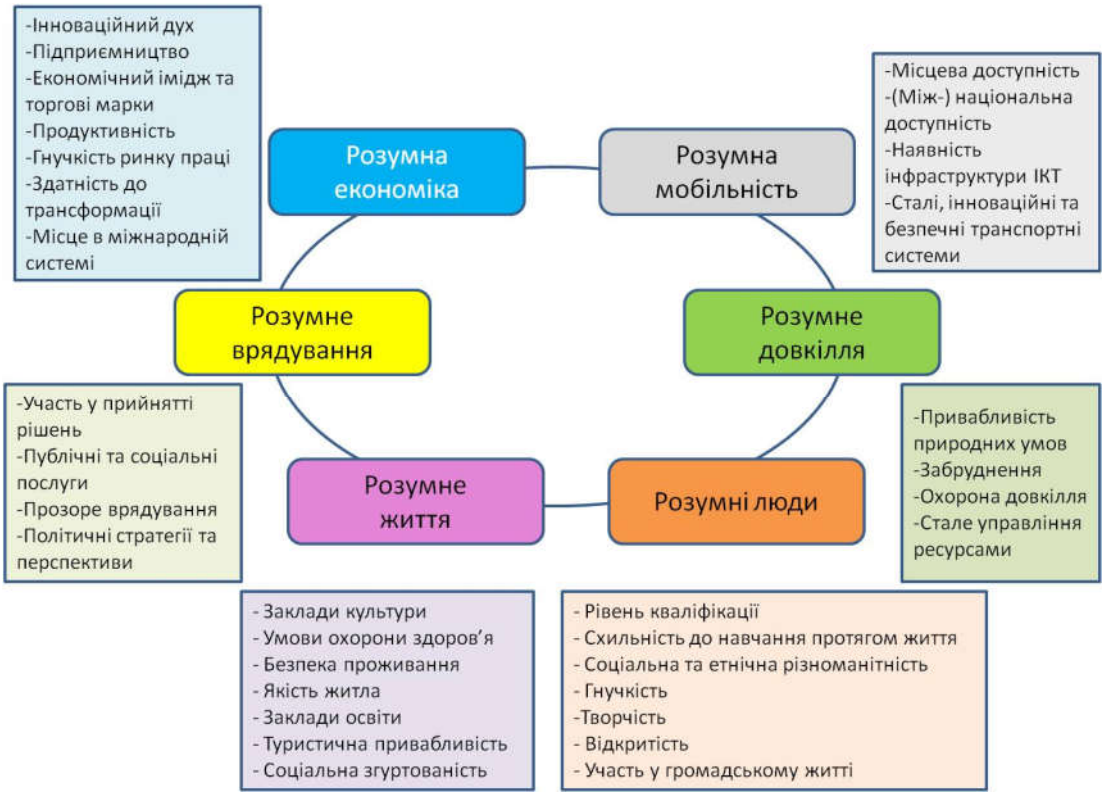

Рисунок 1. 1 - Модель «розумний» місто.

Сьогодні розвивати території, на яких ми живемо, колишніми методами вже неможливо. Старі моделі заводять у глухий кут. Це розуміють якщо не все, то вже дуже багато. Необхідно створювати і впроваджувати абсолютно нові способи розвитку.

У відповідь на цей виклик на заході виникла концепція «розумного міста», яка стала стратегічним механізмом розвитку, з'єднавши в собі зростаюче значення інформації, сучасних технологій і комунікацій, соціального та природного капіталу.

Соціальний і природний капітал разом створюють таке поняття, як сприятлива середа для проживання (безпечна, екологічна, чиста, красива, що дає можливість реалізуватися). Таке уявлення про місто йде врозріз з уявленнями індустріальної епохи, коли задимлене місто був лише придатком заводу.

Розумне місто - це нова модель. І вона стрімко завойовує світ.

Успішність міста сьогодні залежить не тільки від того, який в місті є актив, який раніше називали основними фондами, або фізичним капіталом (це будівлі, земля, виробничі потужності). З кожним днем місто стає все більш залежним від наявності та якості «м'якої інфраструктури», тобто економіки знань, особливої соціальної інфраструктури, всього того, що називають інтелектуальним і соціальним капіталом. Останні поняття - соціальний та інтелектуальний капітал - в російському поданні майже повністю відсутні, або це «іграшки» для інтелектуалів, про які говорять на конференціях і пишуть в дисертаціях.

Проте, світовий досвід показав, що саме вони визначають розвиток сучасного міста.

Дати визначення «розумному місту» не так-то просто. «Розумне місто» є перекладом «smart city». Англійське слово smart - живий, енергійний, тлумачний, розторопний, кмітливий, інтелектуальний, розвиненою, зухвалий і нахабний, охайний, елегантний - як бачите, має цілий комплекс значень. Російське слово «розумний» - породжений ясним розумом, що володіє розумом - уже за значенням.

Сьогодні існує досить багато тлумачень, які можуть разюче відрізнятися один від одного.

Підхід до визначення «розумного міста» спирається на глибоке розуміння ролі соціальних зв'язків і людського капіталу в міському розвитку.

У даному розумінні це місто, в якому місцеві спільноти постійно навчаються, адаптуються, створюють і використовують інновації. Завдання полягає в тому, щоб самі різні жителі міста були включені в соціальне життя і брали участь в управлінні містом і зміні його на краще.

І тут велика роль відводиться екологічної стійкості. Вона необхідна у світі, де ресурси постійно зменшуються, а міста все більше і більше у своєму розвитку роблять ставку на туризм і відпочинок. Використання ресурсів має бути поновлюваним і гарантувати збереження природної спадщини. Тільки баланс між економічним зростанням і захистом навколишнього середовища допоможе досягти по-справжньому сталого розвитку.

Отже, комплексний підхід до «розумного місту» будується на шести головних аспектах:

- розумна економіка,
- розумне (комфортне) навколишнє середовище,
- розумне пересування,
- розумні люди,
- сучасна соціальна система,
- розумна система управління,
- сучасні технології, нова енергетика.

Ключовими елементами тут є:

- місцева конкурентоспроможність;
- новий підхід до організації житла;
- упор на локальну економіку;
- передовий технологічний уклад;
- участь громадян в управлінні;
- зростання людського і соціального капіталу та якості життя.

Це формує зовсім нове розуміння, що таке зростання і розвиток міста.

При такому підході місто може бути визначений як «розумний», коли інвестиції в соціальний і людський капітал, сучасні інформаційно-комунікаційні інфраструктури і технології виробництва тягнуть за собою сталий економічний розвиток, підвищення якості життя та управління навколишнім середовищем (через спільне управління).

До недавнього часу поняття «розумний місто» застосовувалося переважно до великих міст, що наочно демонструє англійська мова, де великий і маленький місто позначаються різними словами (відповідно «city» and «town»).

Але світовий досвід показує, що розумними можуть бути й малі міста. Більш того, розвиток малого міста сьогодні виключено, якщо це місто не стає розумним.

Очевидно, що здатність переробляти інформацію, сучасна інфраструктура і креативний клас можуть бути пов'язані і з маленькими міста. Мале місто може

надати комфортне середовище проживання (у порівнянні з великими містами) - зелену, екологічну, і позбавлену стресу.

Принципи розумного міста.

Розумне місто - це таке місто, інфраструктура якого побудована на нових технологіях, що дозволяють раціонально використовувати джерела енергії і мінімізувати їх шкідливий вплив. Сюди відносяться нові рішення в електроенергетиці, водопостачанні, створенні транспортної системи та «розумних будинків».

Розумне місто - складне поняття, яке складається з розумної економіки, розумних технології, розумною середовища, розумних людей і, нарешті, розумного управління. Всі ці складові частини з'єднуються з традиційними регіональними теоріями міського росту і розвитку. Кожне з цих понять будується на теоріях регіональної конкурентоспроможності, поліпшення якості життя, участі громадян в управлінні містом.

Головна ідея системи «Розумне місто» - створення інформаційного простору, що містить дані про роботу контрольованих об'єктів (лічильників теплової та електричної енергії, ліфтів, електротехнічного обладнання, технічних засобів безпеки і т.д.). Управління об'єктами ведеться на будь-якій відстані в реальному режимі часу, незалежно від місця розташування об'єктів та центрального керуючого пункту в місті.

Аналіз зібраних даних дозволяє знайти слабкі місця в роботі організації, постачальників ресурсів, обладнання та персоналу. Введення в експлуатацію системи «Розумне місто» дозволяє не тільки контролювати роботу устаткування, але і приймати максимально вірні управлінські рішення.

Основою для єдиної системи «Розумне місто» є функціонально закінчені підсистеми:

- диспетчеризації та контролю ліфтів;

- автоматизованого комерційного контролю й обліку енергоресурсів та електроенергії;

- охоронно-пожежної сигналізації та відеоспостереження;

- контролю доступу в приміщення і до обладнання;

- управління обладнанням та інженерними спорудами;

- інші додаткові системи, такі як контроль затоплення підвалів, сигналізація загазованості горючими газами, екстреної голосового зв'язку

Принципи Розумного міста (Smart City):

- Мікрорайон як містобудівна одиниця

- Автономність міста

- Соціальна, ділове і культурне самодостатність

- Розробка за стандартами екологічного будівництва

- Використання новітніх інформаційних і комунікаційних технологій

- Впровадження інноваційних технологій енергетики, транспорту та будівництва

Основні механізми оптимізації споживання ресурсів в «розумному» місті

- розподіл навантаження на інфраструктурні мережі в часі, тобто зниження нерівномірності споживання в період піків і провалів (основна проблема всіх інфраструктур);

- розподіл в просторі, тобто створення мережевих, а не лінійних систем поставки ресурсу дозволять маневрувати потоками і «обходити» аварійні або пікові ділянки;

- створення динамічно керованих джерел потужності: малоінерційні генератори, накопичувачі, демпфери та ін .;

- створення розподіленої генерації різного масштабу;

- зниження втрат і ресурсоспоживання кінцевих користувачів («розумні» будинки, енергоефективне обладнання та ін.)

Розумні міста світу.

Розумними можуть бути нові міста, які відразу будуються як розумні , або міста, засновані для конкретних цілей (наприклад, промислові міста або технопарки), або, що частіше, звичайні міста, які крок за кроком стають розумними ;. Багато найбільші міста світу почали здійснення проектів створення розумного міста, в тому числі Сеул, Нью-Йорк, Токіо, Шанхай, Сінгапур, Амстердам, Каїр, Дубай, Коті і Малага. Враховуючи сучасні темпи інновацій, цілком імовірно, що вже в найближче десятиліття моделі розумних міст стануть широко поширеними реальними і популярними стратегіями міського розвитку.

Існуючі проекти розумного міста різняться. В Амстердамі основна увага приділяється посиленню екологічної стійкості на основі більш раціональної організації робіт, застосування новітніх технологій для скорочення шкідливих викидів в атмосферу, більш ефективного використання енергії. В інших містах вживаються заходи для перетворення широкого діапазону міських функцій в розумні, використовуючи повсюдно поширені розумні технології у всіх аспектах життя городян. Двома прикладами такої стратегії можуть служити проект Місто електронної інтеграції (u-місто) в Республіці Кореї (реалізація почалася в 2004 році) та проект Deutsche Telekom T-місто в Німеччині (реалізація почалася в 2006 році). Проект Розумний Сеул здійснюється з метою перетворення системи управління містом в більш розумну та підвищення якості життя городян.

У різних містах ставляться різні пріоритетні цілі і завдання, але все розумні міста мають три найважливіші риси. Перша - наявність інфраструктури ІКТ. Захищена інфраструктура ІКТ наступних поколінь має першорядне значення для успішного надання нових послуг в розумних містах і для забезпечення готовності до майбутнього попиту на нові послуги. Друга - в місті повинна бути створена чітко вибудувана і інтегрована система управління. Численні системи розумного міста будуть діяти злагоджено тільки на основі суворого дотримання єдиних стандартів. Третя - в розумному місті мають бути розумні користувачі. ІКТ - це кошти, обеспечівающіе функціонування розумного міста, але вони марні в відсутність компетентних користувачів, які вміють взаємодіяти з розумними послугами. Розумний місто має не тільки розширювати доступ до розумним пристроям для всіх категорій населення з різними рівнями доходів і різних вікових груп, а й забезпечувати доступ до навчання роботі з цими пристроями. Основу розумного міста становить відкрита для всіх мережу користувачів розумних пристроїв, а городяни вимагають або створюють послуги, які представляють для них найбільшу цінність.

Стандартизація для розумних міст.

Враховуючи велике значення стандартизації для створення розумних міст, в різних організаціях здійснюються в цій області різноманітні заходи.

Наприклад, Міжнародна організація по стандартизації (ICO) розглядає стандарти розумних міст в рамках групи, що займається темою системи показників розумною інфраструктури спільнот raquo ;. У Секторі стандартизації електрозв'язку МСЕ (МСЕ-Т) сформована Оперативна група по розумним стійким містам для оцінки потреб у стандартизації міст, які прагнуть посилити свою соціальну, економічну та екологічну стійкість через інтеграцію ІКТ в міську інфраструктуру і діяльність.

Рішення про створення цієї нової Оперативної групи прийняла 5 я Дослідницька комісія (Навколишнє середовище та зміна клімату) на своєму зібранні, яке відбулося 29 січня - 7 лютого 2013 в Женеві. Створення цієї Оперативної групи стало відповіддю на заклик до дій, який пролунав у ході другої Тижня зелених стандартів МСЕ raquo ;, що проводилася у вересні 2012 року в Парижі. Розумні стійкі міста - це також тема третього конкурсу МСЕ Додатки на базі екологічно чистих ІКТ .

Для того щоб формування розумних міст стало наступним етапом процесу урбанізації, будуть потрібні нові стандарти, інфраструктура і рішення ІКТ, тільки тоді ця концепція отримає реальне втілення. Оперативна група МСЕ-Т по розумним стійким містам буде служити відкритим майданчиком для зацікавлених сторін створення розумних міст - муніципалітетів, академічних та науково-дослідних установ, неурядових організацій та організацій у сфері ІКТ, а також галузевих форумів і консорціумів. Зацікавлені сторони зможуть обмінюватися знаннями з метою вироблення стандартизованих основ, необхідних для забезпечення інтеграції послуг ІКТ в розумних містах.

1.2 Аналіз енергоносіїв, структура витрат

Більшість технологічних процесів відбуваються з використанням енергоносіїв різного виду та призначення. Під енергоносіями в промисловості розуміють матеріальне тіло або матеріальну середовище, що володіє певним

потенціалом і передавальну енергію від одного матеріального тіла до інших. Промислові підприємства при організації своєї діяльності використовують енергоресурси різних параметрів, різних видів і різного призначення. Для великих підприємств говорять про потоках енергоносіїв. Напрямок цих потоків тісно пов'язані між собою і мають різні характеристики. На підприємстві вони об'єднуються під загальною назвою «енергоресурси підприємства». Найчастіше в якості енергоресурсів на підприємстві використовуються:

- електрична енергія (60-70% споживання);
- вода;
- тепло;
- повітря;
- ПРВ (продукти розділення повітря);
- розплави і солі.

Головним завданням енергоносіїв на підприємстві є забезпечення умов технологічного процесу. При виборі енергоносіїв та їх характеристик керуються в першу чергу умовою максимальної дешевизни в рамках заданих параметрів. При цьому в першу чергу звертається увага на наступні фактори:

- характеристики і умови протікання технологічного процесу;
- характеристики і параметри встановленого обладнання;
- параметри самого енергоносія;
- характер забезпечення енергоносіями підприємства (внутрішнє або зовнішнє) і т.д.

В якості основних характеристик енергоносіїв при їх виборі враховують:

потенціал або параметри (струм, напруга, температура, тиск і т.д.);

- вартість;
- якість;
- надійність постачання;
- Режими споживання.

Параметри енергоносія визначаються характеристиками споживає обладнання. Якщо на реальному підприємстві застосовуються енергоносії з явно завищеними параметрами, це призводить до збільшення експлуатаційних витрат і грошових витрат на допоміжне обладнання (діаметр жил кабелю,

збільшення металоємності для труб і т.д.). Тому остаточний вибір енергоносія, його якісних і кількісних характеристик проводиться шляхом порівняння декількох варіантів в ході техніко-економічних розрахунків.

1.3 Аналіз алгоритмів опрацювання даних

Генетичний алгоритм (англ.*genetic algorithm*) — це еволюційний алгоритм пошуку, що використовується для вирішення задач оптимізації і моделювання шляхом послідовного підбору, комбінування і варіації шуканих параметрів з використанням механізмів, що нагадують біологічну еволюцію.

Особливістю генетичного алгоритму є акцент на використання оператора "схрещення", який виконує операцію рекомбінацію рішень-кандидатів, роль якої аналогічна ролі схрещення в живій природі.

"Батьком-засновником" генетичних алгоритмів вважається Джон Голланд (англ. John Holland), книга якого "Адаптація в природних і штучних системах" (англ. Adaptation in Natural and Artificial Systems) є фундаментальною в цій сфері досліджень.

Генетичні алгоритми є новим напрямком у алгоритміці. Вони здатні не тільки вирішувати і скорочувати перебір у складних завданнях, але й легко адаптуватися до зміни проблеми.

Схема генетичного алгоритму:

- *Генерація випадкового початкового стану*

Перше покоління створюється з довільно вибраних рішень (хромосом). Це відрізняється від стандартних методів, коли початковий стан завжди одне й те саме.

- *Обчислення коефіцієнта пристасованності (fitness)*

Кожному рішенню (хромосомі) зіставляється якесь чисельне значення, яке залежить від його близькості до відповіді.

- *Відтворення*

Хромосоми, що мають високий рівень пристасованності (fitness), потрапляють до нащадків (які потім можуть мутувати) з більшою ймовірністю. Нащадок,

результат злиття 'батька' і 'матері', є комбінацією їх ген. Цей процес називається 'кроссіннговер' (crossing over).

- *Наступне покоління*

Якщо нове покоління містить рішення, досить близьке до відповіді, то задача вирішена. У протилежному випадку воно проходить через той же процес. Це продовжується до досягнення рішення.

Мурашині алгоритми

У 1992 році у своїй дисертації Марко Дориго запропонував запозичити описаний природний механізм для вирішення завдань оптимізації. Імітуючи поведінку колонії мурах в природі, мурашині алгоритми використовують багатоагентні системи, агенти яких функціонують по вкрай простим правилам. Мурашині алгоритми є ефективними при вирішенні складних комбінаторних завдань та задач оптимізації.

Ідея алгоритму

Мураха прямує від мурашника у випадковому напрямку. Якщо вона знаходить їжу ,то повертається до мурашника і залишає по собі слід з феромону.

Ці феромону приваблюють інших мурах, що знаходяться поруч, і скоріш за все вони рушать цим маршрутом. Повертаючись до мурашника, вони також залишають свій феромон і укріплюють доріжку.

Якщо існує два маршрути, то коротким за цей час встигне пройти більше мурах ніж по довгому і короткий маршрут стане більш вигідним. Довші доріжки повільно зникають внаслідок випаровування феромонів.

Базова ідея алгоритму мурахи полягає в оптимізації шляхом непрямого зв'язку між автономними агентами.

Мурашиний алгоритм моделює багатоагентну систему. Її агентів надалі будемо називати мурахами. Як і справжні мурахи, вони досить просто влаштовані: для виконання своїх обов'язків вони вимагають невелику кількість пам'яті, а на кожному кроці роботи виконують нескладні обчислення.

Кожна мураха зберігає в пам'яті список пройдених їм вузлів. Цей список називають списком заборон (tabu list) або просто пам'яттю мурашки. Вибираючи вузол для наступного кроку, мураха «пам'ятає» про вже пройдені

вузли і не розглядає їх як можливих для переходу. На кожному кроці список заборон поповнюється новим вузлом, а перед новою ітерацією алгоритму - тобто перед тим, як мурашка знову проходить шлях - він очищується.

Окрім списку заборон, при виборі вузла для переходу мураха керується «привабливістю» ребер, які вона може пройти. Привабливість залежить, по-перше, від відстані між вузлами (тобто від ваги ребра), а по-друге, від слідів феромонів, залишених на ребрі мурахами, що пройшли по ньому раніше. Природно, що на відміну від ваг ребер, які є константними, сліди феромонів оновлюються на кожній ітерації алгоритму: як і в природі, з часом сліди випаровуються, а мурахи, що проходять, навпаки, посилюють їх.

Узагальнено, базовий мурашиний алгоритм, незалежно від модифікацій, можна представити у вигляді схеми

Рисунок 1.2- Узагальнено, базовий мурашиний алгоритм, незалежно від модифікацій.

Покроковий опис загальної схеми

Припустимо, що навколишнє середовище для мурах представляє повнозв’язний неорієнтований граф. Кожне ребро має вагу, яка позначається як відстань між двома вершинами, що ним з’єднується. Граф є двохскерованим, тому мураха може подорожувати по грані в будь-якому напрямку.

Ймовірність включення ребра в маршрут окремої мурахи пропорційна до кількості феромонів на цьому ребрі, а кількість відкладеного феромону пропорційне до довжини маршруту. Чим коротший маршрут, тим більше феромону буде відкладено на його ребрах, отже, більша кількість мурах буде включати його в синтез власних маршрутів. Моделювання такого підходу, що використовує тільки додатній зворотний зв'язок, призводить до передчасної збіжності - більшість мурашок рухається по локально-оптимальному маршруту.

Уникнути цього можна моделюючи від’ємний зворотний зв'язок у вигляді випаровування феромону. Причому, якщо феромон випаровується швидко, то це призводить до втрати пам'яті колонії і забування хороших рішень, з іншого боку, збільшення часу випарів може призвести до отримання стійкого локального оптимального рішення.

Мураха - це програмний агент, який є членом великої колонії і використовується для вирішення певної проблеми. Мураха забезпечується набором простих правил, які дозволяють їй вибирати шлях у графі. Вона підтримує список вузлів, які вже відвідала і може пройти через кожен вузол лише один раз. Шлях між двома вузлами графа, за яким мураха відвідала кожен вузол, називається шляхом Гамільтона.

Вузли в списку "поточної подорожі" розташовуються в тому порядку, в якому їх відвідала мураха. Пізніше список використовується для визначення протяжності шляху між вузлами. Справжня мураха під час переміщення по шляху буде залишати за собою феромони. В мурашиному алгоритмі агент залишає феромони на ребрах графа після завершення подорожі.

Стартова точка, куди поміщається мураха, залежить від обмежень, накладених умовами завдання, оскільки для кожного завдання спосіб розміщення мурашок є визначальним. Або всі вони поміщаються в одну точку, або в різні з повтореннями, або без повторень.

На цьому ж етапі задається початковий рівень феромону. Він ініціалізується невеликим додатнім числом для того, щоб на початковому кроці ймовірності переходу в наступну вершину були нульовими.

Рух мурашки

Рух мурашки ґрунтується на простому ймовірнісному рівнянні. Якщо мураха ще не закінчила шлях, тобто не відвідала усі вузли мережі, для визначення наступного ребра шляху використовується рівняння

$$P = \frac{\tau(r,u)^{\alpha} \times \eta(r,u)^{\beta}}{\sum_k \tau(r,u)^{\alpha} \times \eta(r,u)^{\beta}}. \quad (1)$$

Тут τ (r,u) - інтенсивність ферменту на ребрі між вузлами r і u, η (r, u) - функція, яка представляє вимір зворотньої відстані для грані, α - вага ферменту, а β - коефіцієнт евристики. Параметри α і β визначають відносну значимість двох параметрів, а також їх вплив на рівняння. Оскільки мураха подорожує тільки по вузлах, які ще не були відвідані (як зазначено списком табу), ймовірність обчислюється лише для ребер, які ведуть до ще не відвіданих вузлів. Ці ребра представляє змінна k.

Подорож мурашки

Пройдений мурахою шлях відображається, коли мураха відвідає всі вузли графа. Цикли заборонено, оскільки в алгоритм включено список табу. Після завершення довжина шляху може бути підрахована - вона дорівнює сумі довжин всіх ребер, якими подорожувала мураха. Рівняння (2) показує кількість феромону, який був залишений на кожному ребрі шляху для мурашки k. Змінна Q є константою.

$$\Delta\tau_{ij}^{k}(t) = \frac{Q}{L^{k}(t)}. \quad (2)$$

Результат рівняння є засобом вимірювання шляху, - короткий шлях характеризується високою концентрацією феромонів, а більш довгий шлях - більш низькою. Далі, отриманий результат використовується в рівнянні (3), щоб збільшити кількість феромону вздовж кожного ребра пройденого мурахою шляху.

$$\tau_{ij}(t) = \Delta\tau_{ij}(t) + (\tau_{ij}^{k}(t) \times \rho). \quad (3)$$

Важливо, що дане рівняння застосовується до всього шляху, при цьому кожне ребро позначається феромоном пропорційне до довжини шляху. Тому слід дочекатися, поки мураха закінчить подорож і лише потім оновити рівні

феромону, в іншому випадку справжня довжина шляху залишиться невідомою. Константа p - значення між 0 і 1.

Випаровування феромонів

На початку шляху у кожного ребра є шанс бути обраним. Щоб поступово видалити ребра, які входять в гірші шляхи графа, до всіх ребер застосовується процедура випаровування феромону. Використовуючи константу p з рівняння (3), отримуємо рівняння (4):

$$\tau_{ij}(t) = \tau_{ij}(t) \times (1 - \rho). \quad (4)$$

Для випаровування феромону використовується зворотний коефіцієнт оновлення шляху.

Повторний запуск

Після того, як шлях мурашки завершено, ребра оновлено відповідно до довжини шляху і сталося випаровування феромону на всіх ребрах, алгоритм запускається повторно. Список табу очищується, і довжина шляху обнулюється. Мурахам дозволяється переміщатися по графу, засновуючи вибір ребра на рівнянні (1). Цей процес може виконуватися для постійної кількості шляхів або до моменту, коли протягом кількох запусків не було відзначено повторних змін. Потім визначається кращий шлях, який і є рішенням.

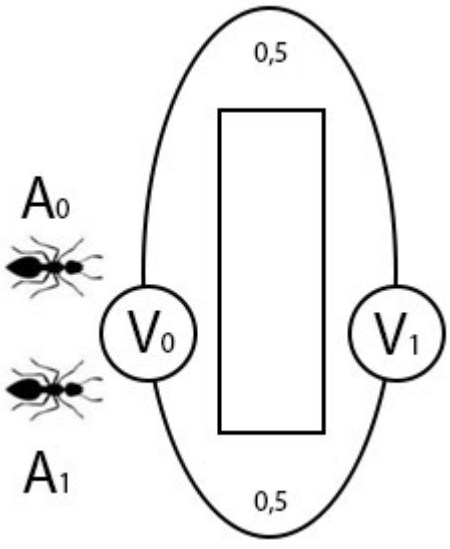

Рисунок 1.3- Демонстраційний приклад

Щоб побачити, як працюють рівняння, розберемо функціонування розглянутого вище алгоритму на простому прикладі. Візьмемо простий сценарій з двома мурашками з прикладу яке розглянуто вище.

На рисунку 1.3 показано приклад з двома ребрами між двома вузлами (V_0 і V_1). Кожне ребро ініціалізується і має однакові шанси на те, щоб бути обраним.

Два мурахи, що знаходяться у вузлі V_0 позначаються як A_0 і A_1. Оскільки ймовірність вибору будь-якого шляху однакова, в цьому циклі проігноруємо рівняння вибору шляху.

Області застосування

Алгоритм оптимізації мурашиної колонії може бути успішно застосований для вирішення складних комплексних завдань оптимізації. Мета вирішення складних комплексних завдань оптимізації - пошук і визначення найбільш відповідного рішення для оптимізації (знаходження мінімуму або максимуму) цільової функції (ціни, точності, часу, відстані тощо) з дискретної множини можливих рішень.

Типовими прикладами вирішення такого завдання є задача календарного планування, завдання маршрутизації транспорту, різних мереж (GPRS, телефонні, комп'ютерні тощо), розподіл ресурсів та робіт. Ці задачі виникають у бізнесі, інженерії, виробництві та багатьох інших областях. Дослідження показали, що метод мурашиних колоній може давати результати, навіть кращі ніж при використанні генетичних алгоритмів і нейронних мереж.

Модифікації класичного алгоритму

Результати перших експериментів із застосуванням мурашиного алгоритму для вирішення завдання комівояжера були багатообіцяючими, проте далеко не кращими порівняно з вже існуючими методами. Однак, простота класичного мурашиного алгоритму («мурашиної системи») залишала можливості для доробок - і саме алгоритмічні вдосконалення стали предметом подальших досліджень фахівців у галузі комбінаторної оптимізації. В основному, ці вдосконалення пов'язано з великим використанням історії пошуку та більш ретельним дослідженням областей навколо вже знайдених вдалих рішень.

Elitist Ant System

Введення в алгоритм так званих «елітних мурах». Досвід показує, що проходячи ребра, що входять в короткі шляхи, мурахи з більшою ймовірністю будуть знаходити коротші шляхи. Ефективною стратегією є штучне збільшення рівня феромонів на найвдаліших маршрутах. Для цього на кожній ітерації алгоритму кожна з елітних мурах проходить шлях, який є найкоротшим із знайдених на даний момент.

Експерименти показують, що, до певного рівня, збільшення числа елітних мурах є досить ефективним, дозволяючи значно скоротити число ітерацій алгоритму. Однак, якщо число елітних мурах занадто велике, то алгоритм досить швидко знаходить субоптимальне рішення і застряє в ньому. Як і інші змінні параметри, оптимальне число елітних мурах слід визначати дослідним шляхом.

Ant-Q

Мурашиний алгоритм, який отримав свою назву за аналогією з методом машинного навчання Q-learning. В основі алгоритму лежить ідея про те, що мурашину систему можна інтерпретувати як систему навчання з підкріпленням. Ant-Q підсилює цю аналогію, запозичуючи багато ідей з Q-навчання.

Алгоритм зберігає Q-таблицю, що співставляє кожному з ребер величину, яка визначає «корисність» переходу по цьому ребру. Q-таблиця змінюється в процесі роботи алгоритму - відбувається навчання системи. Значення корисності переходу по ребру обчислюється виходячи із значень корисності переходу за наступними ребрам в результаті попереднього визначення можливих наступних станів. Після кожної ітерації корисності оновлюються виходячи з довжин шляхів, до складу яких було включено відповідні ребра.

Ant Colony System

Для підвищення ефективності в порівнянні з класичним алгоритмом введено три основних зміни.

По-перше, рівень феромонів на ребрах оновлюється не лише в кінці чергової ітерації, але і при кожному переході мурах з вузла у вузол. По-друге, наприкінці ітерації рівень феромонів підвищується тільки на найкоротшому із знайдених шляхів. По-третє, алгоритм використовує змінене правило переходу:

або, з певною часткою ймовірності, мураха безумовно вибирає краще ребро у відповідності до довжини і рівня феромонів, або робить вибір так само, як і в класичному алгоритмі.

Max-min Ant System

Мурашиний алгоритм, в якому підвищення концентрації феромонів відбувається тільки на кращих шляхах з пройдених мурахами. Така велика увага до локальних оптимумів компенсується введенням обмежень на максимальну і мінімальну концентрацію феромонів на ребрах, які вкрай ефективно захищають алгоритм від передчасної збіжності до субоптимальних рішень.

На етапі ініціалізації, концентрація феромонів на всіх ребрах встановлюється рівною максимальній. Після кожної ітерації алгоритму тільки одна мураха залишає за собою слід - або найбільш успішна на даній ітерації, або, аналогічно до алгоритму з елітизмом, елітна. Цим досягається, з одного боку, більш ретельне дослідження області пошуку, з іншого - його прискорення.

ASrank

Модифікація класичного мурашиного алгоритму, в якому в кінці кожної ітерації мурахи ранжуються у відповідно до довжин пройдених ними шляхів. Кількість феромонів, що залишається мурахою на ребрах, таким чином, призначається пропорційно до її позиції. Для ретельного дослідження околу вже знайдених вдалих рішень, алгоритм використовує елітних мурах.

Висновок

Ефективність мурашиних алгоритмів порівнянна з ефективністю загальних мета евристичних методів, а в ряді випадків - і з проблемно-орієнтованими методами. Найкращі результати мурашині алгоритми показують для задач з великою розмірністю областей пошуку і можуть бути успішно застосовані для вирішення складних комбінаторних задач оптимізації. Мурашині алгоритми добре підходять для застосування разом з процедурами локального пошуку, дозволяючи швидко знаходити початкові точки для них.

Найбільш перспективними напрямками подальших досліджень у даному напрямку слід вважати аналіз способу вибору параметрів, що налаштовуються

в алгоритмах. В останні роки пропонуються різні способи адаптації параметрів алгоритмів «на льоту». Оскільки від вибору параметрів сильно залежить поведінка мурашиних алгоритмів, саме до цієї проблеми звернено найбільшу увагу дослідників на даний момент.

1.4 Постановка задач магістерської роботи та шляхи її вирішення

В даному розділі проаналізовано поняття «інтелектуальне» місто, проаналізована система витрат енергоносіїв, та проведений аналіз алгоритмів опрацювання даних.

Для досягнення поставленої мети необхідно розв’язати наступні задачі:

1) Провести опис систем опрацювання витрат енергоносіїв;
2) Провести огляд підходів до створення та роботи баз даних;
3) Проаналізувати алгоритм доступу до віддаленої бази даних;
4) Проаналізувати алгоритм опрацювання даних витрат енергоносіїв
5) Розробити алгоритм опрацювання даних витрат енергоносіїв;
6) Розробити структуру програмної системи опрацювання даних витрат енергоносіїв.

Проведено опис «інтелектуального» міста. Результати досліджень дозволили оцінити стан розвитку даних систем та спрогнозувати подальші шляхи розвитку.

Проведено аналіз алгоритмів опрацювання даних . Це дозволило окреслити особливості доступу та програмної реалізації інформативних технологій для доступу та редагування баз даних.

2 РОЗРОБКА АЛГОРИТМІВ ОПРАЦЮВАННЯ ДАНИХ ВИТРАТ ЕНЕРГОНОСІЇВ

2.1 Алгоритм доступу до віддалених баз даних

Оскільки база даних містить дані які потрібні багатьом користувачам, а також отримання одночасного доступу одного або декількох користувачів до спільної бази даних. Доступ до бази даних за певними алгоритмами (рисунок 2.1).

Доступ до даних здійснюється за допомогою трансакції або віддаленого запиту. Для початку роботи доступу до віддаленої бази даних відправляється запит на отримання доступу, після чого користувачу потрібно вести логін і пароль до бази даних, якщо відповідні логін і пароль присутні в базі акантів то користувачу надається доступ до бази даних з відповідним рівнем доступу, якщо ж відповідних логіну і паролю не знайдено то система повертається на початок для повторного вводу.

При успішному вході формуються запити на отримання інформації з бази даних.

У блоці визначення бази даних формуються запити до таблиць в яких зберігається потрібна інформація. Після визначення таблиці формуються SQL запити на отримання даних, та виконання цих запитів.

Наступним блоком йде перевірка наявності даних в таблиці якщо вони присутні то відбувається агрегація в протилежному випадку система повертається на визначення таблиць.

Агрегація даних являє собою взаємозв’язок одного класу даних з іншими. Після агрегації формується відповідь на запит у зручному для користувача вигляді.

Коли формування відповідей на запит завершується, то виводиться результат на отримання даних в якому знаходиться уся інформація по наших запитах. По закінченню отримання і перегляду результатів система пропонує

вихід з віддаленої бази даних, якщо «так» то ми виходимо з системи, якщо ж «ні» то наш алгоритм повертається до запитів на отримання даних.

При підключені до бази даних використовуються стандартні функції підключення. Нижче наведений приклад для з'єднання з БД. Опис функцій присутній в таблиці 2.1.

```
public void Form1_Load(object sender, EventArgse)
{
string connect_string = "Data Source=NAME_PC;Initial
Catalog=DB_NAME;Integrated Security=false;User ID = Login; Password =
Pass";
SqlConnection sql_connect = new SqlConnection(connect_string);

sql_connect.Open();
}
```

Таблиця 2.1- Функції для з'єднання з глобальною БД

Параметри	Опис функції
EventArgse	Являє собою базовий клас для класів, які містять дані про події та надає значення використовуватиметься для подій, які не включають дані події.
параметр sender	визначає об'єкт
string connect_string	задає рядок, що використовується для підключення до бази даних
Integrated Security	Параметр, який визначає, чи є з'єднання захищеним. True, False і ССПІ - можливі значення (ССПІ (Інтегрована безоеквівалент True)
sql_connect.Open()	Зазначає що підключення до бази даних відкрите і база готова для використання

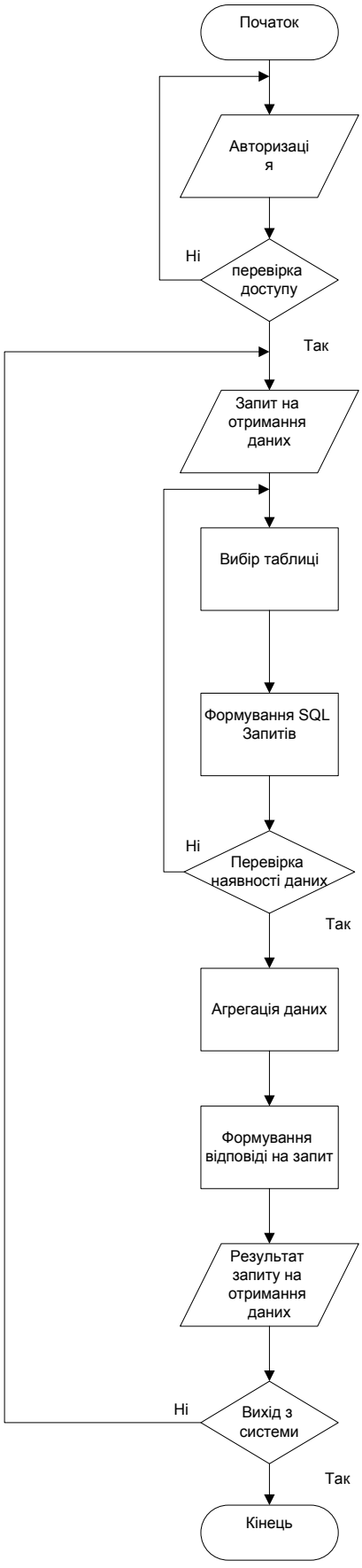

Рисунок 2.1- Блок-схема роботи алгоритму доступу до бази даних

При підключенні та роботою з віддаленою базою даних можуть виникнути ряд помилок. Детальніше про помилки та їх вирішення наведені в таблиці 2.2.

Таблиця 2.2- Помилки при роботі з базою даних

Помилка	Вирішення помилки
Неможливо підключитися до бази даних	Якщо при підключенні до бази даних вказується пароль , термін дії якого скоро закінчується , то може виникнути помилка . Виправлення : Для усунення такої неполадки вручну скиньте пароль користувача бази даних за допомогою застосовуваної системи управління базами даних. В результаті попереджувала повідомлення не буде відправлятися , і WebSphere Portal Express зможе успішно підключитися до бази даних.
Помилка SQL SQL0969N в протоколі SystemErr	У протокол SystemErr.log може бути занесено таке повідомлення про помилку: SQL0969N У файлі повідомлень на цій робочій станції відсутній текст повідомлення для помилки SQL "-873". Помилка повернута модулем "DSNX0END" з вихідними маркерами. SQLSTATE = 53090 Виправлення: Ця помилка виникає в

Продовження таблиці 2.2

	тому випадку, якщо бази даних необхідні для персоналізації, були створені з одним значенням, а таблиці ресурсів знаходяться в базі даних, що використовує інше значення.
помилка у файлі trace.log	Дана помилка виникає при відсутності драйвера Microsoft Type 4 JDBC з версією 2000 SP3a або пізнішої версією.
EJPSJ0004E:	Системі не вдалося знайти дозвіл. Дозвіл не існує. Дії: Додайте дозвіл в сховище
EJPSJ0010E: Системі не вдалося знайти ресурс.	Ресурс не знайдений в сховищі. Дії: Перевірте правильність імені ресурсу.
EJPSJ0022E: Ім'я ресурсу не може містити символ .	Неприпустимий символ в імені ресурсу. Дії: Замініть неприпустимий символ.
EJPSJ0041E: Тип паролю не підтримується.	Пояснення: Даний тип пароля не підтримується. Дії : Виберіть підтримуваний тип пароля.

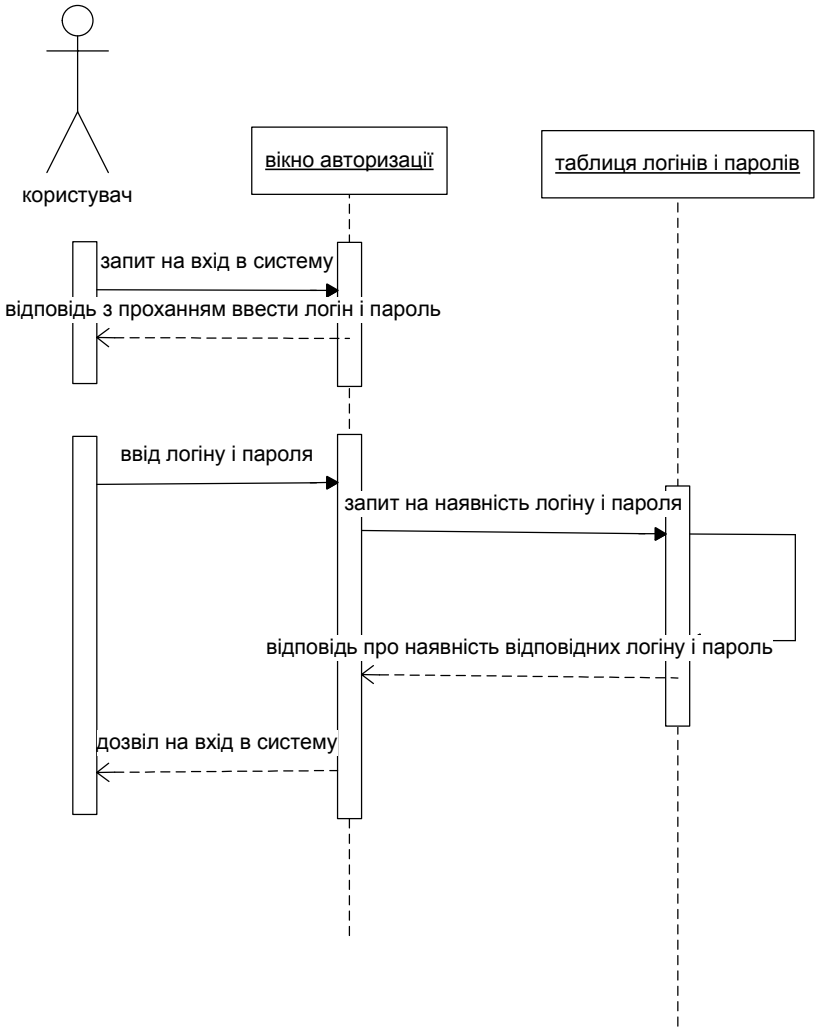

Рисунок 2.2- Діаграма послідовності для авторизації

На рисунку 2.2 зображено процес авторизації користувача за допомогою діаграми послідовності. Діаграма послідовності це- UML діаграма яка відображає взаємодії об’єктів впорядкованих за часом. Авторизація для доступу в базу даних один із головних аспектів користування і адміністрування. Це дозволяє надавати різні рівні доступу, а також здійснює захист від несанкціонованого доступу. Як видно з рисунку користувач посилає запит ініціалізації до вікна авторизації і створює запит до записів даних авторизації, користувач вводить логін і пароль для перевірки після чого система відправляє запит на встановлення даних, база даних робить перевірку на присутність відповідних даних і видає відповідь про наявність чи не наявність даних даних після того якщо є присутній відповідний запит то користувачу надається доступ до віддаленої бази даних.

Даний алгоритм віддаленого доступу до бази даних має низку переваг.

Зниження завантаження локальної мережі завдяки тому що по мережі прямують на SQL-запити, а виклики процедур. Для веб- доступу не потрібно ніяких додаткових настройок і встановлення програмного забезпечення на комп'ютер користувача. Алгоритм є уніфікованим і надає користувачу всі можливості передбачені розробником БД і пришвидшує виконання типових операцій. Можна також зазначити що алгоритм має можливості для розширення.

До недоліків можна віднести те що алгоритм не до кінця розроблений з можливістю подальшого розширення і розробкиі тому є можливість виникнення помилок які наведені в таблиці 2.2.

2.2 Алгоритм опрацювання даних витрат енергоносіїв

Алгоритм опрацювання даних витрат енергоносіїв базується на зчитувані даних з бази даних порівняння даних з попередніми значеннями, обчислення оплати, врахування пільг якщо вони є, а також перевірку на наявність заборгованостей, виведення загальної інформації у вигляді графіків на яких зображено загальну інформацію про енергоносії введений показник , та використання на кожне число. Друк або зберігання у зручному для користувача форматі PNG, JPEG, PDF, SVG. Загальний алгоритм опрацювання даних зображено на рисунку 2.3

- PNG (Portable Network Graphics) — растровий формат збереження графічної інформації, що використовує стиснення без втрат. PNG був створений для заміни формату GIF графічним форматом, який не потребує ліцензії для використання.
- JPEG (Joint Photographic Experts Group) — растровий формат збереження графічної інформації, що використовує стиснення з втратами.
- PDF (Portable Document Format) — відкритий формат файлу, створений і підтримуваний компанією Adobe Systems, для представлення двовимірних документів у незалежному від пристрою виведення та роздільної

здатності вигляді. Кожен PDF-файл може містити повну інформацію про 2D-документ, таку як: тексти, зображення, векторні зображення, відео, інтерактивні форми та ін.

- SVG(Scalable Vector Graphics)(скорочено (з англ. *масштабована векторна графіка*) — специфікація мови розмітки що базується на XML, та формат файлів для двовимірної векторної графіки, як статичної, так і анімованої та інтерактивної. SVG може бути виключно декларативним, або містити описи сценаріїв

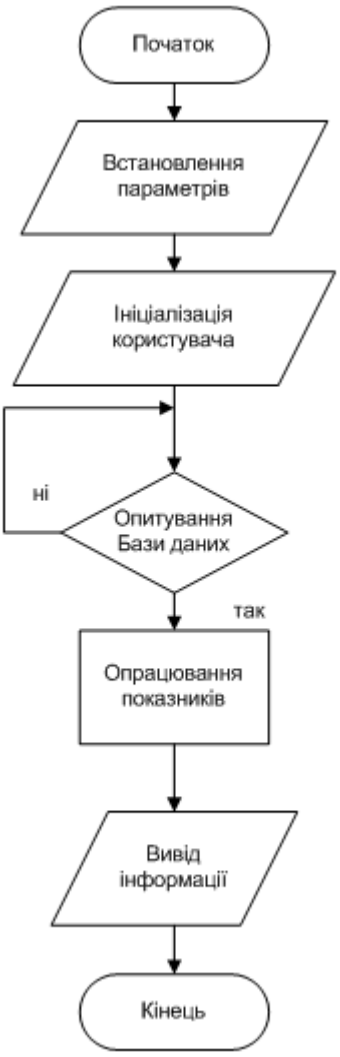

Рисунок 2.3- Блок-схема опрацювання витрат

Опрацювання витрат електроенергії відбувається за наступним алгоритмом зображеним на рисунку 2.4. при опрацюванні електроенергії система зчитує дані показників з бази даних.

Рисунок 2.4- Блок-схема опрацювання витрат електроенергії

Після цього визначає тариф(звичайний , нічний - розраховуватись за спожиту електричну енергію у час пікового навантаження (з 8:00 до 11:00 год. та з 20:00 до 22:00 год.) плата по тарифу, який на 50% більший від діючого. У час напівпікового навантаження (з 7:00 до 8:00 год., з 11:00 до 20:00 год., з 22:00 до 23:00 год.) — по діючому тарифу, в нічний час (з 23:00 до 07:00 год.) — по тарифу, який на 60% менший від діючого, зелений - економічний

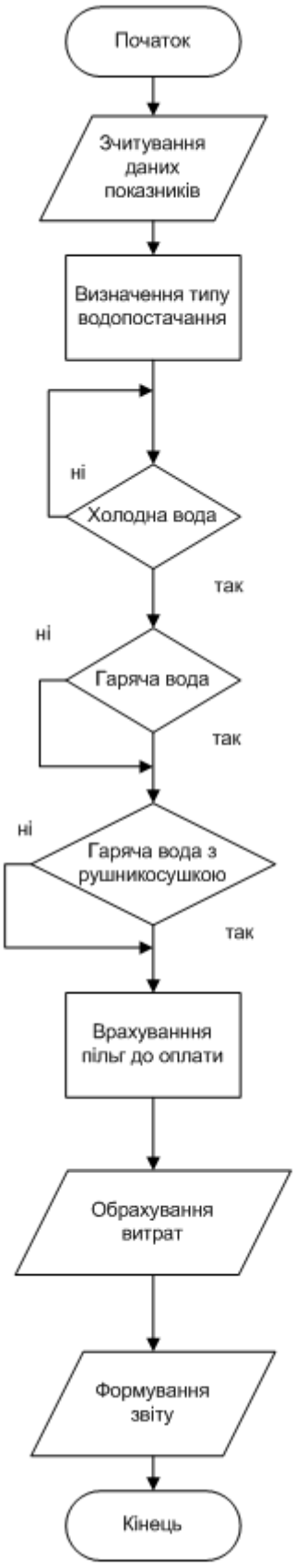

Рисунок 2.5- Блок-схема опрацювання витрат водопостачання

механізм, спрямований на заохочення генерації електроенергії відновлюваною енергетикою.:) по якою буде здійснене обрахування, при

обрахуванні також враховується чи присутні в користувача комбіновані тарифи(звичайний –нічний, звичайний – зелений). Від цих даних залежить тариф та кінцева сума оплати при потребі якщо в користувача є пільги вони також враховуються при опрацюванні і після цього відбувається обрахування витрат та формування звіту.

Рисунок 2.6- Блок-схема опрацювання витрат газу

Опрацювання витрат водопостачання відбувається за наступним алгоритмом зображеним на рисунку 2.5. При опрацюванні даних витрат водопостачання враховується який тип води подається користувачеві(

холодна,гаряча, гаряча з рушникосушкою) від цього залежить тариф ,коли ці дані враховані враховується наявність пільг та формується звіт

Опрацювання даних газопостачання відбувається за наступним алгоритмом зображеному на рисунку 2.6. Відбувається зчитування з лічильника перевіряється наявність пільг та обраховується сума до сплати та формуються звіт.

2.3 Порівняння систем для збору та опрацювання даних витрат енергоносіїв

Для визначення якості програмного модуля опрацювання даних витрат енергоносіїв інтелектуального міста, необхідно порівняти його з аналогічними відомим системами. Для порівняння оберемо наступні аналогічні системи : oplata.te.ua, easypay.ua.

При дослідженні форми входу на сайт системи мають різне спрамування. Oplata.te.ua системи входу як такої немає, тільки при здійснені оплати сайт переправляє на сайт Приватбанку і там вже користувач авторизується для оплати. На easypay.ua присутня система авторизації , але можна обійтись і без неї знаючи розрахунковий рахунок.

Для порівняння якісних характеристик систем необхідно дослідити ряд параметрів:

- зовнішній вигляд;
- зручність;
- швидкість входу;
- підтримка кросбраузерності;
- надійність;
- захищеність.

Дані параметри потрібно занести у таблицю і оцінити. Результати занесені у таблицю 2.1

Таблиця 2.1 – Порівняння систем опрацювання витрат енергоносіїв

Параметри	oplata.te.ua	easypay.ua
зручність	В системі присутній інтелектуальний інтерфейс. Все легко заповнюється	В системі присутній інтелектуальний інтерфейс. Все легко заповнюється
швидкість входу	Для отримання даних не потрібено авторизувуватись це пришвидшує роботу з системою	Для роботи потрібно зареєструватись ввівши мобільний номер телефону і пароль.
підтримка кросбраузерності	Присутня	Присутня
надійність	Система недостатньо надійна оскільки дані певного користувача можна переглянути ввівши його адрес проживання	Середня надійність системи для користування потрібна авторизація або розрахунковий рахунок
захищеність	Пропонує при оплаті перейти на сайт приват банку для авторизації і оплати. Це забезпечує надійне зберігання персональних даних	При оплаті потрібно ввести номер карти та CCV код . через це краще не роботи оплату через ненадійні мережі

Недоліки oplata.te.ua:

Система посуті є більше для оплати енергоносіїв а ніж для моніторингу, дані потрібно вводити користувачеві і тільки у певний період часу коли проводиться оплата комунальних послуг. Якщо ввести пізніше зазначеного терміну система автоматично введе значення аналогічного місяця попереднього року, це є незручністю для користувача .

Недоліки easypay.ua:

Для вводу даних потрібно знати номер особового рахунку чи у порівнянні з попередньою системою є недоліком бо в корисувача незавжди є можливість її вводити , а також аналогічні з попередньою системою недоліки по вводу даних.

2.4 Розробка структури бази даних

Розробка структури БД- найважливіша задача, яка вирішується при проектувані БД.

Створюючи базу даних ми прагнемо впорядкувати інформацію по різним ознакам, щоб потім витягати з неї необхідні нам дані в будь-якому поєднані. Це можливо якщо дані структуровані. Структурування – це набір угод про способи представлення даних.

Як показано на рисунку 2.6 в залежності від структури є п’ять основних модель баз даних:

– ієрархічну;

– мережеву;

– реляційну;

– об’єктно-орієнтовану;

– гібридну.

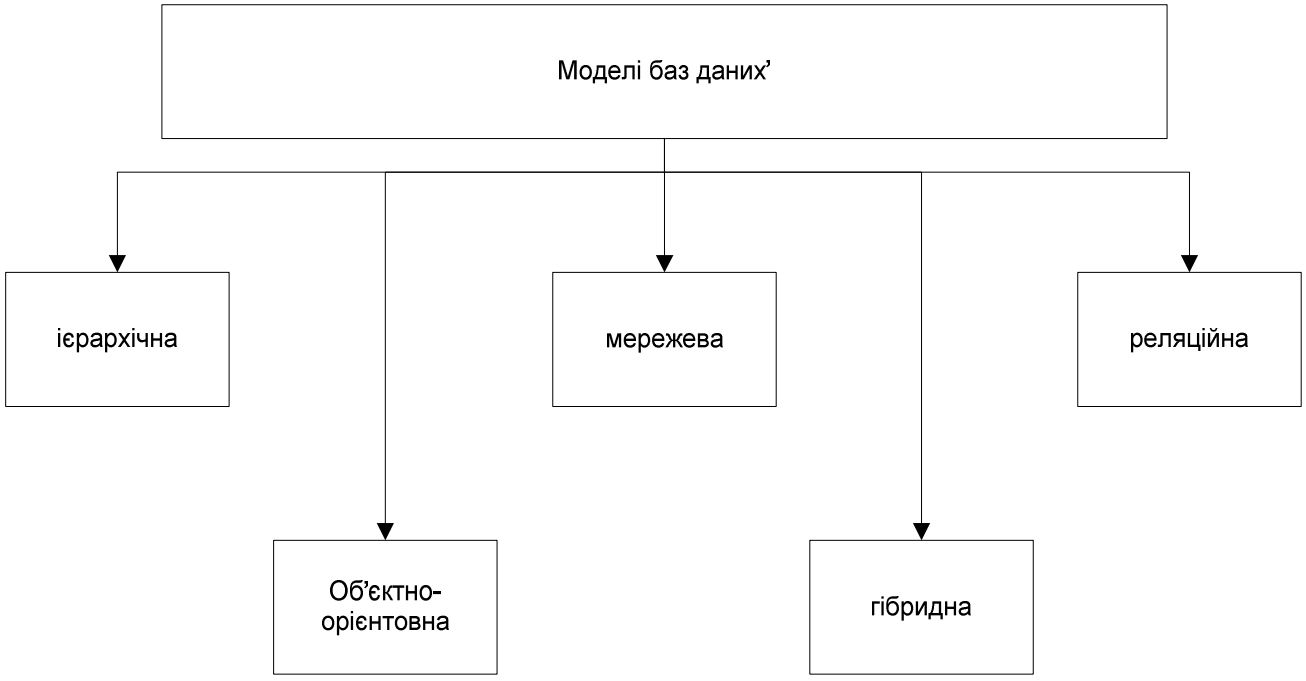

Рисунок 2.6- Моделі баз даних

Ієрархічна модель – дані будуються за принципом ієрархії об’єктів у якому один тип є головним, усі нижче лежачі підлеглими. У цій моделі встановлюється зв’язок «один до багатьох», тобто для деякого головного типу існує кілька підлеглих типів об'єктів. Інакше, головний тип іменується

вихідним типом, а підлеглі - породженими. У підлеглих типів можуть бути у свою чергу підлеглі типи. Найвищий в ієрархії вузол (сукупність атрибутів) називають кореневим.

Мережна модель даних- будується за принципом "головний і підлеглий тип одночасно", тобто будь-який тип даних одночасно може одночасно породжувати кілька підлеглих типів (бути власником набору) і бути підлеглим для декількох головних (бути членом набору).

Об’єктно-орієнтована модель- особливістю цієї моделі є те, що вся сукупність даних, що буде оброблятись і зберігатись в базі даних, подана не у вигляді наборів окремих картографічних шарів і таблиці, а у вигляді об’єктів певного класу.

Найпопулярнішою на даний час є реляційна структура .

Реляційна структура бази даних – всі дані приставлені в вигляді простих таблиць, які в свою чергу розбиті на стрічки і стовпці, на перетині яких розміщені дані. Цей тип структури даних дозволяє зберігати інформацію в електронних таблицях і здійснює пошук в одній таблиці на підстав ключових полів іншої таблиці.

Основними принципами нормалізації бази є такі:

- в кожній таблиці БД не повинно бути повторюваних полів ;
- кожна таблиці повинна мати первинний ключ;
- кожному первинному ключу повинна відповідати достатня інформація про тип суті або об’єкт таблиці(наприклад інформація про реєстрацію певного абітурієнта);
- зміна значень в полях таблиці не повинна впливати на інформацію в інших полях(крім змін у полях ключа).

Ключ – це стовбець в деяких випадках декілька стовпців які дозволяють встановити зв'язок таблиці із записами в іншій.

Існує два типи ключі:

- первинні;
- вторинні(зовнішні).

Первинний ключ (primary key) являє собою один з прикладів унікальних індексів і застосовується для унікальної ідентифікації записів таблиці . Ніякі з двох записів таблиці не можуть мати однакових значень первинного ключа. Первинний ключ зазвичай скорочено позначають як PK (primary key) .

В реляційних базах даних практично завжди різні таблиці логічно пов'язані один з одним. Первинні ключі якраз використовуються для однозначної організації такого зв'язку.

За способом завдання первинних ключів розрізняють логічні (природні) ключі і сурогатні (штучні) .

Для логічного завдання первинного ключа потрібно вибрати в базі даних те, що природним чином визначає запис . Прикладом такого ключа є номер паспорта в базі даних про паспортні дані жителів.

Якщо підходящих прикладів для природного завдання первинного ключа не знаходиться , користуються сурогатним ключем. Сурогатний ключ являє собою додаткове поле в базі даних , призначене для забезпечення записів первинним ключем. Навіть якщо в базі даних міститься природний первинний ключ , краще використовувати сурогатні ключі , оскільки їх застосування дозволяє абстрагувати первинний ключ від реальних даних. .Первинному ключу можна привласнити атрибут auto_increment , що дозволяє автоматично генерувати унікальний ключ , якщо його тип є цілочисловим . При вставці запису в базу даних значення ключа виставляється рівним нулю, MySQL автоматично обчислює максимальний номер первинного ключа , збільшує його на одиницю і присвоює це значення первинному ключу нового запису .

Схемою БД називають структуру зв’язків між полями і таблицями.

Нормалізацією схеми БД називають процедуру яка проводиться над базою даних з цілю видалення в ній надлишковості.

Нормалізація несе з собою чимало переваг. Очевидно, що в нормалізованої базі даних зменшується ймовірність виникнення помилок, вона займає менше місця на жорсткому диску і т.д.

Для того, щоб краще усвідомити наведене визначення нормалізації, розглянемо наступний приклад. Нижче показана таблиця, в якій вказані прізвища абітурієнтів на факультет вступу:

Таблиця 2.2- Приклад надлишковості в таблиці БД.

Вулиця	Абонент
Просвіти	Петров І.В.
Просвіти	Іванов С.О.
Лучаківського	Степанов Ю.М.
Лучаківського	ЯрмоленкоР.Б.
Лучаківського	Остапович О.М.

Ця таблиця надлишкова - для кожного із абонентів повторюються однакові назви вулиць. Тобто схема такої бази даних не нормалізована. Для невеликої бази даних це не критично, але у великих за обсягом базах даних це позначиться на розмірі бази і, в кінцевому рахунку, на швидкості доступу. Для нормалізації необхідно розбити цю таблицю на дві - для вулиць (таблиця 2.3) і для абонентів (див. таблиця2.4).

Таблиця 2.4-Таблиця вулиць

Вулиці	Первиний ключ
Просвіти	1
Лучаківського	2

Таблиця 2.5- Таблиця абонентів.

Вулиця(зовнішній ключ)	Абонент
1	Петров І.В.
1	Іванов С.О.
2	Степанов Ю.М.
2	ЯрмоленкоР.Б.

Тепер кожена вулиця позначена унікальним числом, і рядок в базі даних є тільки в єдиному екземплярі: в таблиці вулиці. Розмір поля "вулиці" зменшився, так як рядок займає більше пам'яті, ніж число.

Список найбільш часто зустрічаються типів наведено у таблицях 2.6– 2.9 Для багатьох типів даних задається максимальна ширина відображення, що вказується в дужках, яку ми далі будемо позначати символом max. Наприклад, запис INT (2) означає, що значення даного поля не може перевищувати 100.

До числовим типів відносяться цілі числа і числа з плаваючою крапкою. Для чисел з плаваючою точкою, крім максимальної ширини відображення можна також вказувати число значущих цифр після коми, далі позначається символом P.

Типи дати і часу наведені в таблиці 2.7.

Основні стрічкові типи наведені в таблиці 2.8.

Таблиця 2.6-Числові типи

Тип	Опис
INT[(max)]	Звичайні цілі числа.
FLOAT[(max,P)]	Числа з плаваючою точкою одинарної точності.
DOUBLE[(max,P)]	Числа з плаваючою точкою подвійної точності.

Таблиця 2.7- Типи дати і часу

Тип	Опис
DATETIME	Дата і Час в форматі РРРР-ММ-ДД ГГ-ХХ-СС.
TIMESTAMP	Мітка часу для звітів по трансакціям в форматі РРРР-ММ-ДД ГГ-ХХ-СС.

Таблиця 2.9- Стрічкові типи

Тип	Опис
CHAR	Синонім CHAR(1).
TEXT	Рядки з максимальною довжиною символів рівний 65535. Дані цього типу чутливі до регістру.

Переваги реляційної структури бази даних значно перевищують недоліки. Серед переваг можна зазначити такі переваги як:

– Ця модель даних відображає інформацію в найбільш простий для користувача формі ;

– Заснована на розвиненому математизації апараті, який дозволяє досить лаконічно описати основні операції над даними;

– Дозволяє створювати мови маніпулювання даними не процедурного типу;

– Маніпулювання даними на рівні вихідний БД і можливість зміни.

До недостатків можна віднести те, що в реляційної структури бази даних самий повільніший доступ до даних порівняно ї іншими структурами, а також доволі трудомісткий у плані розробки.

В даному розділі магістерської роботи було розроблено алгоритми роботи системи опрацювання даних, проведено порівняльний аналіз програмних систем для опрацювання даних витрат енергоносіїв дані порівнянь дозволили оцінити системи та спрогнозувати подальший шлях розвитку. А також було проведено розробку структури бази даних визначення її переваг та недоліків.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМІВ ОПРАЦЮВАННЯ ДАНИХ ВИТРАТ ЕНЕРГОНОСІЇВ

3.1 Опис структури програмної системи опрацювання даних витрат енергоносіїв

Для написання програмного засобу за допомогою якого буде здійснюванись опрацювання даних витрат енергоносіїв використовувалось програмне середовище PHP Storn.. JetBrains PhpStorm — комерційне крос-платформове інтегроване середовище розробки для PHP [1], яке розробляється компанією JetBrains на основі платформи IntelliJ IDEA.

PhpStorm являє собою інтелектуальний [2] редактор для PHP, HTML і JavaScript з можливостями аналізу коду на льоту, запобігання помилок у сирцевому коді і автоматизованими засобами рефакторинга для PHP і JavaScript. Автодоповнення коду в PhpStorm підтримує специфікацію PHP 5.3/5.4/5.5/5.6/7.0/7.1 (сучасні і традиційні проекти), включаючи генератори, співпрограми, простори імен, замикання, типажі і синтаксис коротких масивів. Присутній повноцінний SQL-редактор з можливістю редагування отриманих результатів запитів [3] [4] .

PhpStorm розроблений на основі платформи IntelliJ IDEA, написаної на Java. Користувачі можуть розширити функціональність середовища розробки за рахунок установки плагінів, розроблених для платформи IntelliJ, або написавши власні плагіни.

Вся функціональність WebStorm включена в PhpStorm.

PhpStorm надає багатий і інтелектуальний редактор коду для PHP з підсвічуванням коду , розширеною конфігурацією форматування коду, перевіркою на наявність помилок на льоту і розумним автодоповненням. [5]

- Підтримка PHP 5.3, 5.4 та 5.5, включаючи генератори, співпрограми, простори імен, замикання, типажі, синтаксис коротких масивів, доступ до члена класу при інстанціюванні, розіменування масиву при виклику

функції, бінарні літерали, вираження в статичних виклики тощо. PhpStorm може використовуватися як для сучасних, так і для традиційних проектів на PHP.

- Автодоповнення коду фіналізують класи, методи, імена змінних, ключові слова PHP, а також широко використовувані імена полів і змінних залежно від їхнього типу.
- Підтримка стандартів оформлення коду (PSR1/PSR2, Drupal, Symfony2, Zend).
- Підтримка PHPDoc. PhpStorm надає відповідне автодоповнення коду, засноване на анотаціях @property, @method і @var.
- Детектор дубльованого коду.
- PHP Code Sniffer (phpcs), котрий перевіряє код на льоту
- Рефакторинги (перейменування, введення змінної/константи/поля, вбудовування змінної).
- Підтримка редагування шаблонів Smarty (підсвічування синтаксичних помилок, автодоповнення функцій і атрибутів Smarty, автоматична вставка парних дужок, лапок і закриваючих тегів тощо)
- MVC подання для фреймворків Symfony2 і Yii
- Розпізнавання коду, запакованого в PHAR-архіви.

Для роботи програмного продукту необхідна робоча станція на основі операційної Windows, оскільки велика кількість користувачів мають навики користування з нею, а також вона інтуїтивна та зручна у роботі.

Згідно до стандарту IEEE, архітектура - це організаційна структура системи. За іншим визначенням, під архітектурою розуміється концепція взаємозв'язку елементів складної структури. Стосовно програмних систем (ПС) в архітектуру включаються компоненти логічної, фізичної і програмної структур.

Розробка архітектури - важливий етап в життєвому циклі ПС, основна задача концептуального проектування. Етап аналізу вимог до системи завершується розробкою функціональної архітектури (зовнішній опис системи, дані про основні компоненти і їх взаємозв'язки, про основні алгоритми, основні структури даних), етап проектування - розробкою системної архітектури

(модульно-ієрархічна структура системи, що включає рішення по організації даних і функціональні специфікації окремих модулів, деталізація до рівня, який робить можливою реалізацію).

На схемі представлено структурну схему програми (рисунок 3.1):

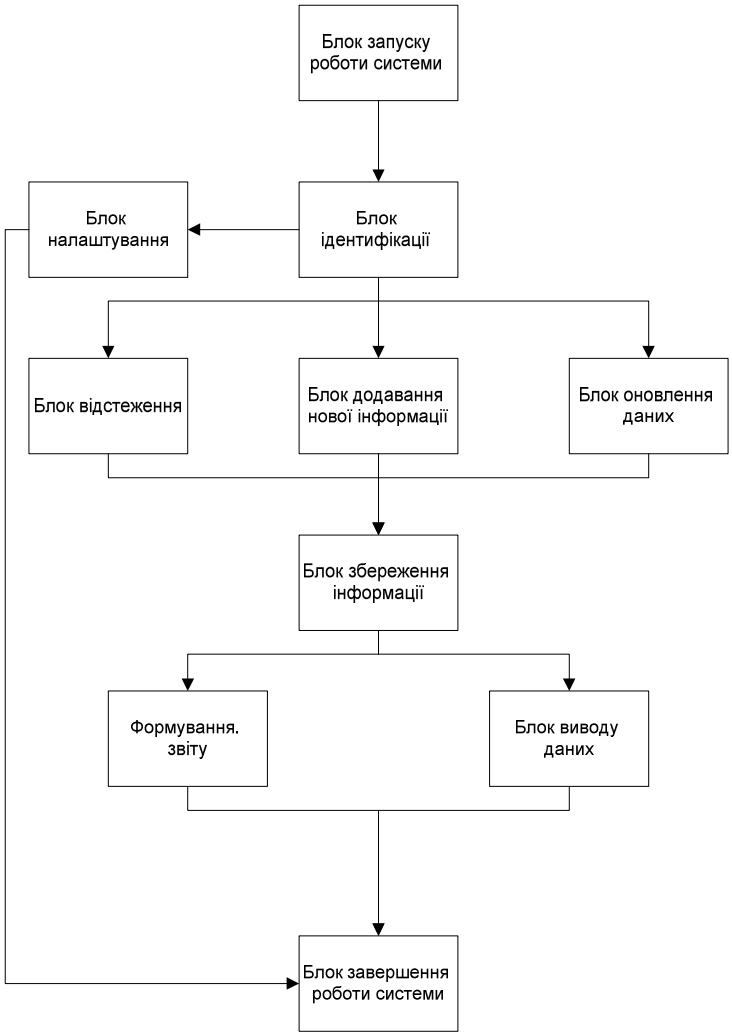

Рисунок 3.1 – Структура системи опрацювання витрат енергоносіїв

«Блок запуску роботи системи». Даний блок формують функціональні модулі ідентифікації користувача в системі, встановлення параметрів роботи системи (по замовчуванню), перевірка наявності необхідних програмно-апаратних складових роботи системи (програмні бібліотеки, пристрої вводу-виводу тощо).

«Блок ідентифікації». основною функцією якого є забезпечення надійного доступу в систему.

«Блок відстеження». Цей блок призначений для цілодобового відстеження показників енергоносіїв він забезпечує моніторинг та виведення даних.

«Блок оновлення даних». Для того щоб оновити інформацію про відстеження показників потрібно вибрати потрібний період а також вибрати тип енергоносія.

«Блок додавання нової інформації.» Додає нову інформацію про користувачав цьому блоці можна вибрати пільгу яка появилась у користувача а також внести нову інформацію про самого користувача.

«Блок збереження інформації». Даний блок використовується для збереження інформації в системі з можливість після цього або вивести її на екран, або ж використати її для формування звіту.

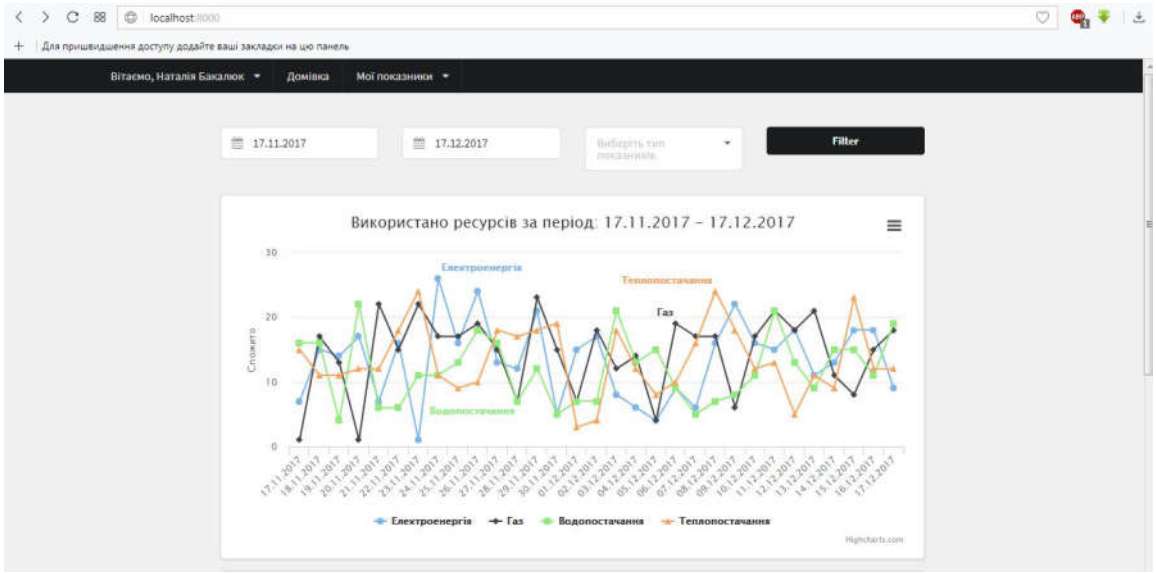

Рисунок 3.2- Головне вікно системи опрацювання даних витрат енергоносіїв

«Формування звітів» - дозволяє виводити отримані дані у зручному для користувача форматі. Даний вузол може бути відсутнім, проте це може вплинути на зручність процесі подальшої обробки даних на візуальному рівні.

«Вивід даних» – дозволяє виводити отримані дані у зручному для користувача форматі. Даний вузол повинен бути в системі обов’язково.

Для швидкого опанування практичними навичками роботи з системою був розроблений простий та інтуїтивно зрозумілий інтерфейс (рисунок 3.2)

На рисунку 3.3 зображено основні функціональні вузли системи.

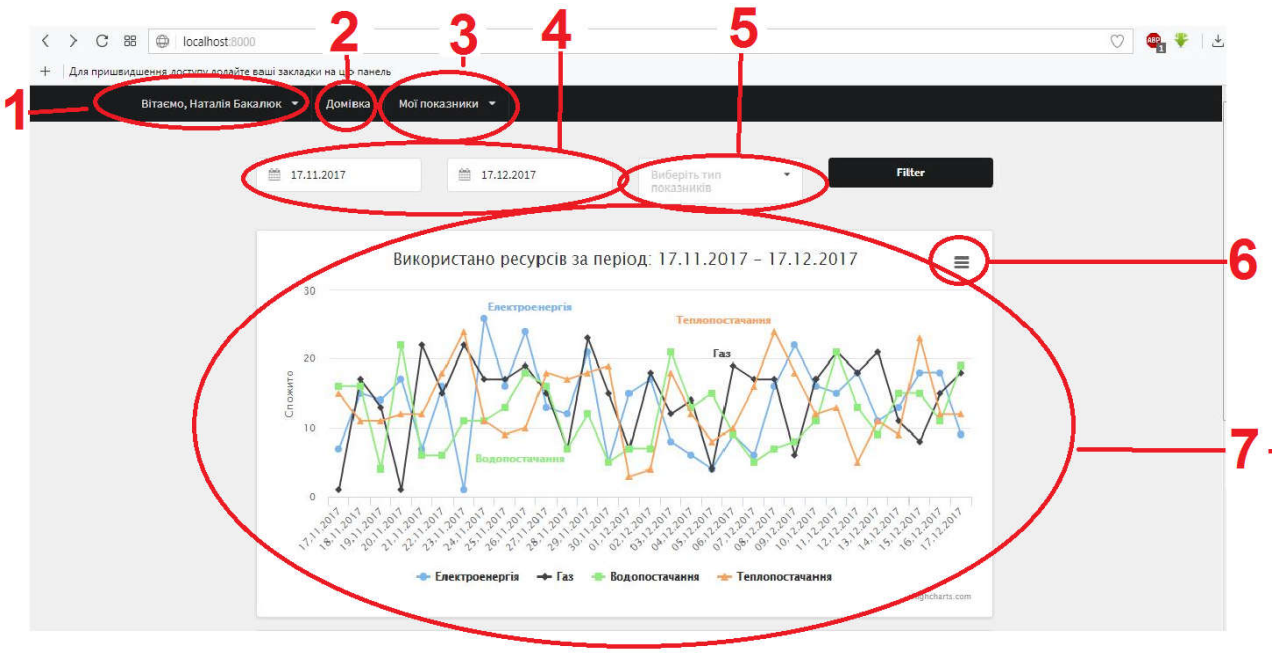

Рисунок 3.3- Основні функціональні вузли системи

Таблиця 3.1- Функціональні вузли системи.

№ п.п	Функція вузла
1	Вікно користувача
2	Домівка
3	Мої показники
4	Вікно вводу дати
5	Вибір типу показників
6	Вікно вибору зберігання документу
7	Графічне відображення показників

«Вікно користувача»- в цьому пункті відображається ім’я користувача що зайшов у програму, а також при наведенні мишки на даний пунк меню то також можна вийти з профілю або зайти в персональні дані .

«Домівка»- дозволяє повернутись на головне вікно програми.

«Мої показники»- у цьому вікні можна вибрати показники енергоносіїв для вводу при, а також в даному вікні присутня функція «Пільги» - яка дозволяє промоніторити чи надані пільги , які види пільг, а також подати дані для отримання пільг

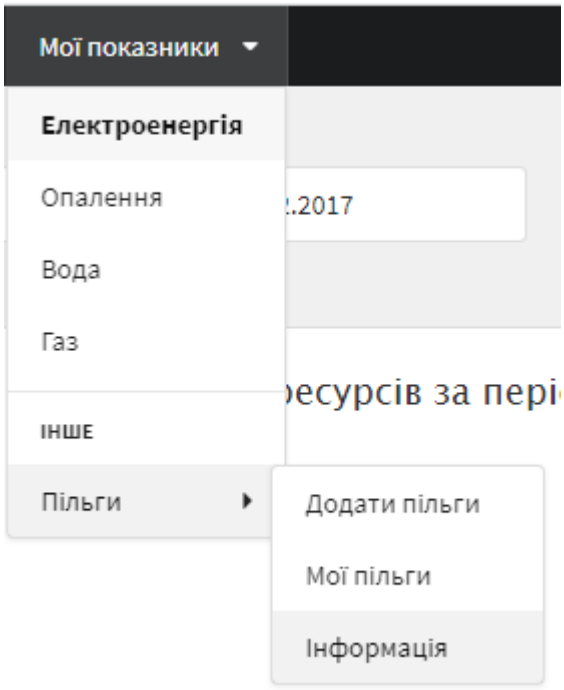

Рисунок 3.4- Основні функціональні вікна «мої показники»

«Вікно вводу дати»- дозволяє вибрати період за який потрібно подивитись показники енергоносіїв.

«Вибір типу показників»- цей пункт використовується для вибору енергоносія (електроенергія, водопостачання, газ , теплопостачання) для перегляду інформації за заданий період , якщо вибору не буде здійснено то графік показників буде відображатись по всіх показниках.

«Вікно вибору зберігання документу»- дозволяє вибрати спосіб збереження графіку у зручному для користувача форматі. Дане вікно дозволяє зберігати інформацію у наступних форматах:

- PNG image;
- JPEG image;
- PDF document;
- SVG vector image.

«Графічне відображення показників»- показує за допомогою графіків використання енергоносіїв да заданий період.

3.2 Опис програмної системи опрацювання даних витрат енергоносіїв

3.2.1 Модуль обробки бази даних

Обробка бази даних означає виконання над нею певних дій для досягнення певного результату.

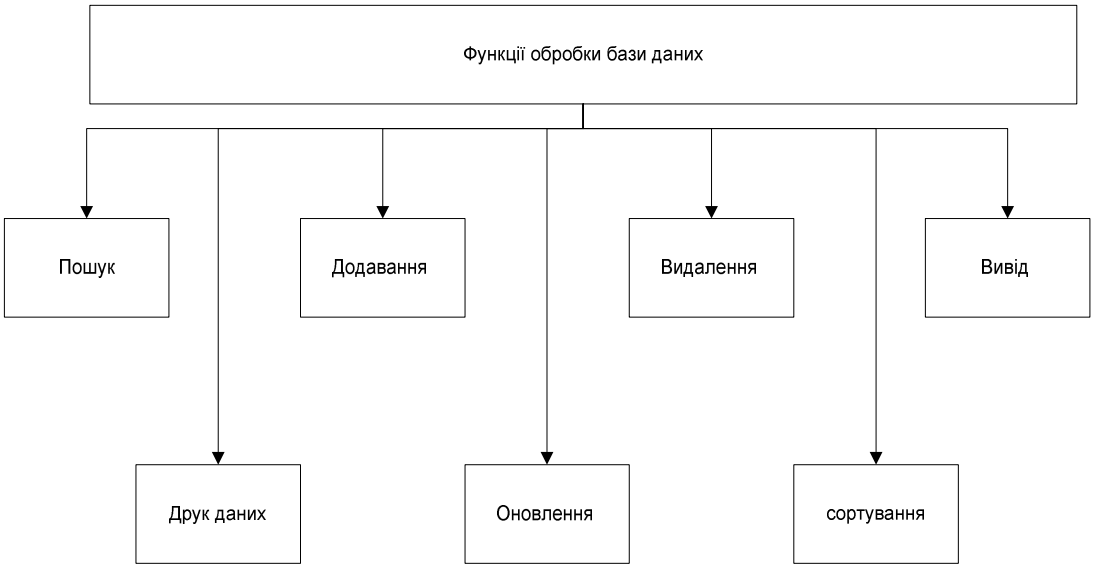

Рисунок 3.4- Функції обробки бази даних

«Пошук»- забезпечує пошук інформації в базі даних по певних заданих критеріях вибору енергоносіїв, типу пільг, дати подання та інших..

«Додавання даних»- додає нові дані в базу даних для подальшого опрацювання.

«Видалення»- функція використовується для видалення непотрібної інформації з бази даних.

«Вивід»- використовується для виводу інформації у зручному для користувача вигляді(у вигляді графіку на екран, або збереження цього графіку .у зручному форматі)

«Сортування»- дозволяє посортувати дані по певним критеріям які задає користувач.

«Оновлення»- дана функція дозволяє оновити базу після введених змін(додавання, видалення, редагування, тощо).

Для розробки і опрацювання бази даних було вибрано об’єктно-орієнтоване програмування воно дозволяє розбити складні програмки на невеликі частинки. Об’єктно-орієнтовані системи більш відкриті і легше піддаються внесенню змін, оскільки їх конструкція базується на стійких формах і це дає можливість системі розвиватись поступово і не приводить до повної її переробки навіть у випадку істотних змін вихідних вимог. З’єднання з базою даних можна подивитись на наступному коді програми:

```
APP_NAME="Облік лічильників"
APP_ENV=local
APP_KEY=base64:na3T3WGKei0WaLvEmx1js3AyzSv+Z588V6srFFsDwD8=
APP_DEBUG=true
APP_LOG_LEVEL=debug
APP_URL=http://localhost
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=tata4
DB_USERNAME=root
DB_PASSWORD=
BROADCAST_DRIVER=log
CACHE_DRIVER=file
SESSION_DRIVER=file
SESSION_LIFETIME=120
QUEUE_DRIVER=sync
REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379
```

APP_KEY=base64:na3T3WGKei0WaLvEmx1js3AyzSv+Z588V6srFFsDwD8=
Ключ програми використовується для всіх зашифрованих даних.

Основні змінні та функції роботи із базою даних описанні в класі, та наведенні у таблиці 3.2

Таблиця 3.1 – Змінні та функції класу для роботи із базою даних

№	Назва	Тип	Опис
1	$_maxid	змінна	Останній ідентифікатор у таблиці
2	$_count	змінна	Лічильник кількості записів у таблиці
3	__construct()	функція	Підключення до бази даних при створенні екземпляру класу DB
4	select()	функція	Вибірка даних із бази даних
5	insert()	функція	Вставлення у базу даних нового запису
6	delete()	функція	Видалення записів із бази даних
7	update()	функція	Оновлення записів бази даних
8	lastInsertId()	функція	Отримання останнього ідентифікатора обраної таблиці
9	getCount()	функція	Отримання кількості вибраних записів після виконання останнього запиту до бази даних

Змінні $_maxid та $_count на початку роботи скрипта дорівнюють нулю. Після виклику функції lastInsertId() першій змінній присвоюється значення ідентифікатора останнього запису в таблиці. Врезультаті виконання функції select() змінній $_count присвоюється значення кількості вибраних записів із бази даних. Дані змінні є приватними, тож доступ до них здійснюється за допомогою функцій lastInsertId() та getCount() відповідно.

Функція select() призначена для здійснення вибірки записів із бази даних. Її опис:

```
select( $table, $sql, $where = null )
```

Дана функція приймає три параметри: два обов’язкових та третій необов’язковий. Параметр $table – це назва таблиці з якої потрібно здійснити вибірку. $sql – рядок із полями таблиці, які потрібно вибрати; вказуються через кому. Якщо необхідно вибрати всі поля таблиці, передається «*». $where –

умова запиту, встановлення ліміту чи порядку сортування. Приклад виклику функції select() наведений нижче.

```
DB->select (‘users’, ‘login, password’, ‘id=1’).
```

Даний запит виконує вибір інформації про користувача з полів login та password таблиці users, ідентифікатор якого дорівнює одиниці.

Функція insert() призначена для вставки нової інформації у базу даних. Її синтаксис:

```
insert( $table, $data ).
```

Параметр $table, як і в попередній функції, означає назву таблиці, з якою необхідно працювати. $data – це асоціативний масив, який складається із ключа (назва поля, в яке потрібно вставити дані) та значення (дані для вставлення). Приклад виклику даної функції:

```
$user = array(‘login’ => ‘Natali’,‘password’ => ‘2jbzcx7r’),
DB->insert (‘users’, $user).
```

В даному прикладі оголошується масив $user, в якому міститься інформація про нового користувача. При цьому, ключ масиву «login» означає, що стрічка «Natali» асоціюється із полем таблиці «login». Аналогічно з полем «password».

Функція delete() призначена для видалення записів із бази даних. Її синтаксис:

```
delete( $table, $where, $limit = 1 ).
```

Параметр $table має аналогічне призначення, як і в попередніх функціях. $where – обов’язкова умова для видалення запису. $limt – максимальна

кількість записів для видалення. Якщо не передавати даний параметр, то максимальна кількість дорівнюватиме одиниці. Приклад виклику функції:

DB->delete (‘users’, “login=‘Natali’”).

В результаті виконання даної функції користувача із логіном «Natali» буде видалено. Якщо такого користувача не існує, функція поверне текст помилки.

Функція update() оновлює інформацію вибраного запису. Її синтаксис:

update($table, $data, $where).

Параметр $table та $data мають таке ж призначення, як у функції insert(). Основною відмінністю є наявність обов’язкового параметру $where, що задає умову, при якій потрібно здійснити заміну. Приклад виклику функції:

$user = array(‘login’ => ‘Alonoslav’,‘password’ => ‘t4io8c2’),
DB->update (‘users’, $user, “login=‘Natali’”).

В результаті, користувачеві з логіном «Natali» буде змінено логін та пароль, на вказані в масиві.

3.2.2 Модуль опрацювання даних витрат енергоносіїв

Даний модуль є основним в програмній системі опрацювання даних витрат енергоносіїв, оскільки від результатів роботи даного модуля в основному залежить достовірність отриманих даних.

Панель інструментів програми зображається наступним кодом:

public function index(Request $request)

{ $from = $request->has('from') ? strtotime($request->get('from')) :

mktime(0,0,0, date('n') - 1, date('d'), date('Y'));

$to = $request->has('to') ? strtotime($request->get('to')) : mktime(0,0,0, date('n'), date('d'), date('Y'));

$userStatistic = new Metric();

$userStatistic = $userStatistic->selectRaw('max(amount) as amount, dn_timestamp as timestamp, type')

->where('user_id','=', $this->user->id)

->where('dn_timestamp','>=',$from - 86400)

->where('dn_timestamp','<=',$to)

->groupBy('type', 'dn_timestamp')

->orderBy('dn_timestamp', 'DESC');

if ($request->has('type') && $request->get('type') !== null)

$userStatistic->where('type', $request->get('type'));

$userStatistic = $userStatistic->get();

$tmp = $userStatistic->toArray();

В даній частині коду прописано основні з функцій сортування, та виводу запитів у заданих часових проміжках при заданому користувачеві та заданому типу енергоносіїв.

Основні функції та змінні які використанні при написанні програми, та наведенні у таблиці 3.3

Таблиця 3.3 – Змінні та функції використанні при написанні програми

№	Назва	Опис
1	$request	Змінні HTTP-запиту Змінні в масиві $ _REQUEST передаються в скрипт за допомогою методів GET, POST або COOKIE, тому їм не можна довіряти, тому що вони могли бути змінені віддаленим користувачем. Їх наявність і порядок додавання даних до відповідних масиви визначається директивою конфігурації
2	$ ARGC	Кількість аргументів переданих скрипту
3	$prev	Пересуває внутрішній покажчик масиву на одну позицію назад
4	key	Вибирає ключ з масиву
5	count	Підраховує кількість елементів масиву або щось в

		об'єкті
6	array_merge_recursive	Функція зливає елементи двох або більше масивів таким чином, що значення одного масиву приєднуються в кінець іншого. Повертає результуючий масив. Якщо вхідні масиви мають однакові рядкові ключі, то значення цих ключів зливаються в масив, і це робиться рекурсивно, так що якщо одне зі значень є масивом, то функція зливає його з відповідним значенням в іншому масиві. Однак, якщо масиви мають однакові числові ключі, кожне наступне значення не замінить початкове значення, а буде додано в кінець масиву.
7	Timestamp	послідовність символів або закодованої інформації, яка б показала, коли відбулося певне подія. Зазвичай показує дату і час
8	$ Php_errormsg	є змінною, що містить текст останньої помилки, згенерованої PHP. Ця змінна буде доступна тільки в блоці коду, в якому сталася помилка, і тільки якщо включена конфігураційна опція track_errors (за замовчуванням відключена).
9	array_fill_keys	Створює масив і заповнює його значеннями, з певними ключами
10	Imagick::statisticImage	Замініть кожен піксель відповідною статистикою з околиці вказаної ширини та висоти
11	floatval	Повертає значення змінної var у вигляді числа з плаваючою точкою (float).
12	ksort	Сортує масив по ключам, зберігаючи відносини між ключами і значеннями. Ця функція корисна, в основному, для роботи з асоціативними масивами.
13	compact	Створює масив, в якому знаходяться і їх значення. Для кожного з переданого параметрів, функція compact () шукає змінну з вказаним ім'ям в поточній таблиці символів і додає їх в виведений масив так, що ім'я змінної стає ключем, а вміст змінної стає значенням цього ключа
14	isset	Визначає, чи була встановлена змінна значенням відмінним від NULL Якщо змінна була видалена за допомогою unset (), то вона більше не вважається встановленої. isset () поверне FALSE, якщо перевіряється змінна має значення NULL. Слід пам'ятати, що null-байт ("\ 0") не є еквівалентом константі PHP NULL. Якщо були передані декілька параметрів, то isset () поверне TRUE тільки в тому випадку, якщо всі

		параметри визначені. Перевірка відбувається зліва направо і закінчується, як тільки буде зустрінута невизначена змінна.
15	reset	встановлює внутрішній покажчик масиву на його перший елемент
16	tmpfile	Створює тимчасовий файл з унікальним ім'ям, відкриваючи його в режимі читання і запису (w +), і повертає файловий покажчик. Цей файл автоматично видаляється після закриття (наприклад, шляхом виклику функції fclose (), або якщо не залишилося жодного посилання на покажчик файлу, що повертається tmpfile ()), або при завершенні роботи скрипта.
17	uasort	Ця функція сортує масив таким чином, що його індекси зберігають відносини з елементами, з якими раніше були асоційовані, за допомогою callback-функції, наданої користувачем. Це зазвичай використовується при сортуванні асоціативних масивів, в яких важливий актуальний порядок елементів.
18	natcasesort	Сортує масив, використовуючи алгоритм "natural order" без обліку регістра символів

3.3 Тестування системи опрацювання даних витрат енергоносіїв

Програмна система опрацювання витрат енергоносіїв призначена для моніторингу та економічного використання енергоносіїв. Основними можливостями системи є:

- можливість користувачу оцінити витрати енергоносіїв за певний проміжок часу;
- зберегти дані у зручному для користувача форматі.

Головною сторінкою програмної системи опрацювання даних витрат енергоносіїв є форма для входу в систему. Дана сторінка зображена на рисунку 3.5.

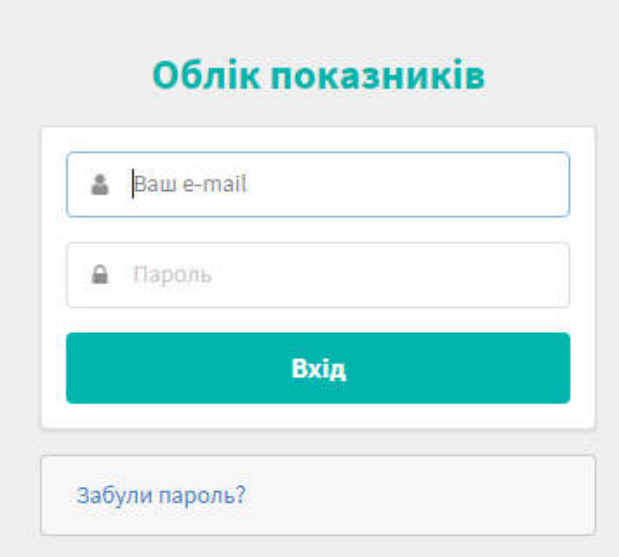

Рисунок 3.5-Форма входу систему

Якщо користувача ще не має в системі, йому потрібно перейти по посиланю «Зареєструватись».На даний сторінці розміщена форма реєстрації(Рисунок 3.6).

Для реєстрації потрібно ввести дані нового користувача та підтвердити надсилання форми. Якщо при реєстрації виникли помилки, їхній текст буде відображатись біля відповідного текстового поля, а саме поле матиме червоний колір.

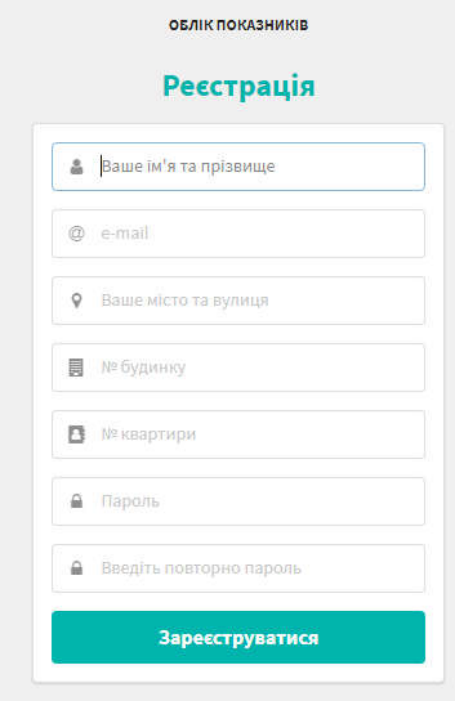

Рисунок 3.6-Форма реєстрації

Наприклад на рисунку 3.7 зображено вивід помилки про те що користувач з заданим поштовим адресом вже зареєстровано, а також помилку про те, що занадто короткий пароль і, що він повинен бути неменше 8 символів.

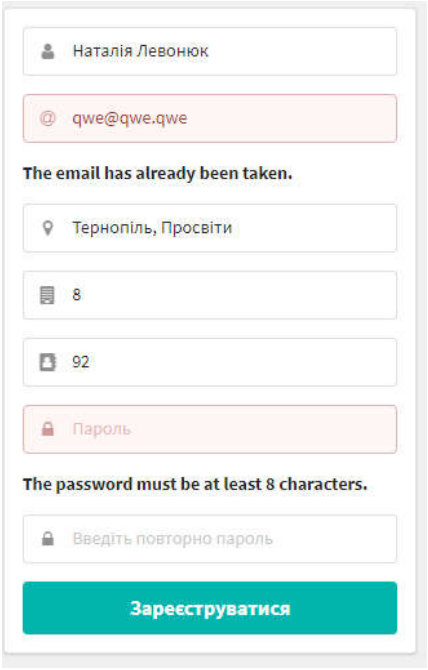

Рисунок 3.7 - Відображення помилок при реєстрації

Після реєстрації користувач може здійснити вхід в систему. На головній сторінці приставлено інтелектуально зрозуміле меню.

Доступні функції перегляду і моніторингу енергоносіїв даного користувача, для цього користувачу потрібно вибрати часовий інтервал та тип енергоносія для перегляду. Графік використання електроенергії за період 1.11.2017-31.11.2017 приставлено на рисунку 3.8.

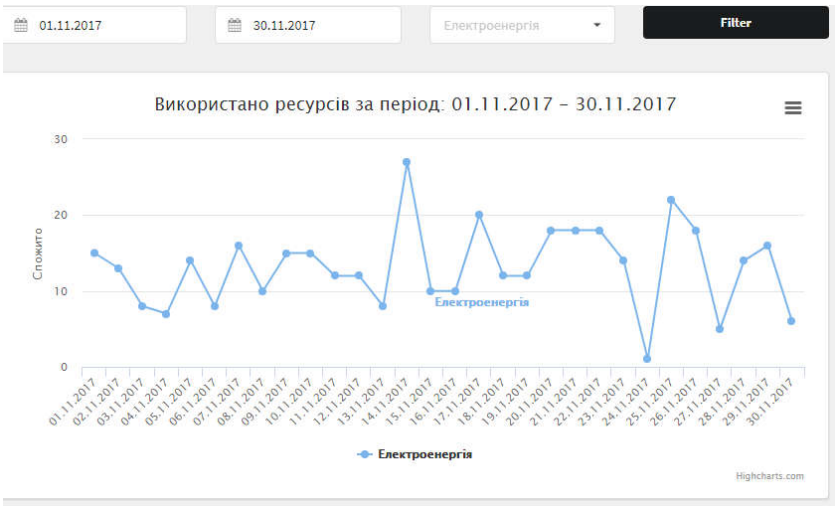

Рисунок 3.8-Графік використання електроенергії

На сторінці перегляду графіка енергоносіїв відображається дата та споживання електроенергії за день.

На сторінці мої показники можна вручну ввести дані, якщо з тихнічних причин система не зчитала дані, а також переглянути тарифікацію та оплату за спожиту енергію.

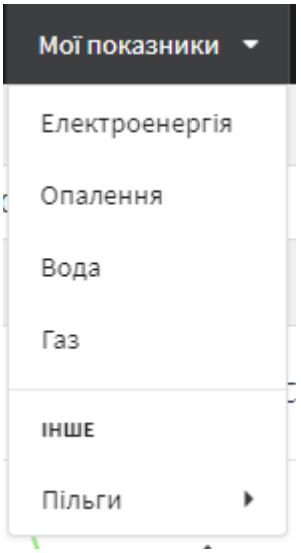

Рисунок 3.9-Сторінка перегляду «Мої показники»

На цій сторінці присутня функція вводу та перегляду наявних пільг.

В функції пільги можна переглянути які пільги доступні користувачеві , та який відсоток економії вони надають. Також можливо подати заявку на пільги, які будуть розглядати відповідні служби.

Збереження даних у зручному форматі приставлено на рисунку3.10.

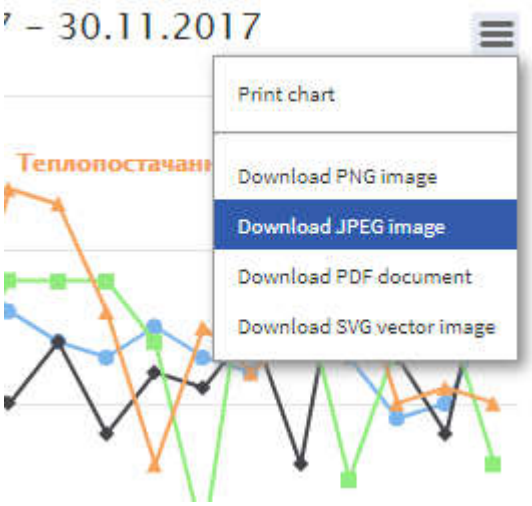

Рисунок 3.10-Збереження даних

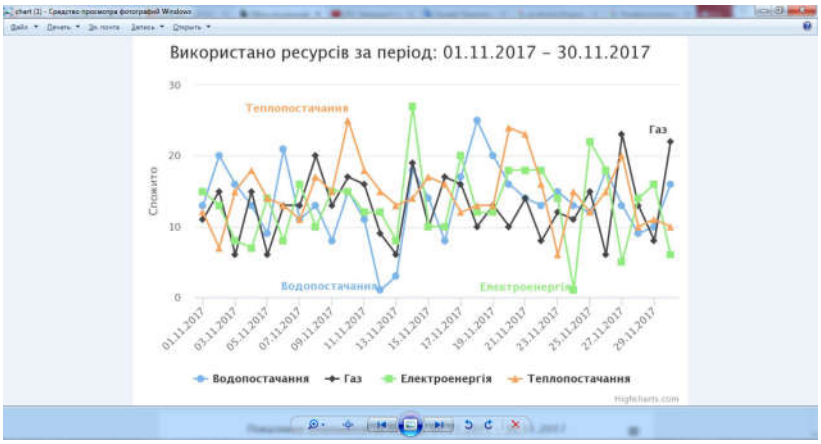

Рисунок 3.11-Збереження даних у форматі JPEG

Вміст збереженого файла приставлено на рисунку 3.11

В даному розділі було проведено розробку структури системи опрацювання даних витрат енергоносіїв на основі модульного підходу який забезпечує максимальну ефективність та простоту подальшого розвитку системи.

ВИСНОВКИ

На основі аналізу сучасних алгоритмів опрацювання даних, програмних засобів доступу до баз даних як в глобальних так і в локальних мережах систем опису інформаційних систем можна зробити наступні висновки:

1) На сьогоднішній день не існує єдиного універсального підходу опрацювання даних витрат енергоносіїв. Оптимальні результати можна отримати шляхом поєднання декількох алгоритмів;

2) Сучасні алгоритми кодування в достатній мірі забезпечують опис та доступ до баз даних.

3) Модульний підхід під час проектування програмної системи забезпечує можливість нарощування системи вертикально (додавання нових модулів) та горизонтально (додавання нових функцій у вже існуючі модулі);

4) Модульний підхід під час проектування програмної системи забезпечує можливість нарощування системи вертикально (додавання нових модулів) та горизонтально (додавання нових функцій у вже існуючі модулі);

5) Алгоритми доступу до баз даних мають високу швидкодію та низку переваг, доступ до віддаленої бази дозволить знизити завантаження локальної мережі.

6) Розроблена структура програмної системи на основі модульного підходу забезпечує максимальну ефективність та простоту подальшого розвитку системи як в напрямку збільшення функціональних модулів, так і збільшення кількості алгоритмів, що реалізовані в них.

7) Розроблена програмна системи опрацювання даних витрат енергоносіїв забезпечує можливість проведення опрацювання показників та виводу інформації у зручному вигляду а також збереження даних у декількох форматах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кібербезпека «розумного» міста – Режим доступу : http://dridu.dp.ua/konf/konf_dridu/itis%20seminar%202016/pdf/13.pdf

2. Теслюк В.М. Автоматизація проектування мікроелектромеханічних систем на компонентному рівні: Монографія /В.М. Теслюк, П.Ю. Денисюк // Львів: Видавництво Львівської політехніки – 2011. С. 192

3. Керівницкво по PHP – Режим доступу : http://phpnet.in.ua/nylrlttgiczshlgqkhrllolgkmjpcwnmnxfmpkuqjtipenfouvzeijksushvot.html

4. PhpStorm 10.0.1 Опис – Режим доступу : http://softarhiv.net/1315-phpstorm.html

5. В. В. Грицик, І. Г. Цмоць, О. В. Скорохода Методи паралельно-вертикального опрацювання даних у нейромережах // Доповіді Національної академії наук України – 2014 - № 10- С. 40-43

6. Цмоць І. Г., Скорохода О. В., Балич Б. І. Модель та НВІС-структури формального нейрона паралель- но-вертикального типу з використанням мультиплексування шин // Зб. наук. праць “Моделювання та інформаційні технології” Ін-ту проблем моделювання в енергетиці. – 2013. – Вип. 67. – С. 160–166

7. М.О. Медиковський, І.Г. Цмоць, О.В. Скорохода, Ю.В. Цимбал / ІНТЕЛЕКТУАЛЬНІ КОМПОНЕНТИ ЕНЕРГЕТИЧНИХ СИСТЕМ НА ОСНОВІ КОНЦЕПЦІЇ SMART GRID // Науковий журнал «Енергетика: економіка, технології, екологія» - 2015 – №4 - С. 25-29

8. Ромашев В. CMS Drupal: Система управления содержимым сайта. / В. Ромашев СПб.: Питер, 2010. – 255 с.;

9. Рассохин Д. World Wide Web - всемирная информационная паутина в сети Internet. / Д. Рассохин, А. Лебедев – М.: МГУ, 1997. – 288 с.;

10. Abrams M. World Wide Web: Beyond the Basics. / M. Abrams – Edinburgh: Prentice Hall, 1998. – 312 p.;

11. Прохоренок Н. А. HTML, javascript, PHP и MySQL. Джентльменский набор Web-мастера. / Н. А. Прохоренок – СПб.: БХВ-Петербург, 2011. – 912 с.;

12. Кеннеди Б. HTML и XHTML. Подробное руководство. / Б. Кеннеди, Ч Муссиано / Пер. с англ. – М.: Символ-Плюс, 2012. – 752 с.;

13. Фримен Эл. Изучаем HTML, XHTML и CSS. / Эл. Фримен, Эр. Фримен / Пер. с англ. – СПб.: Питер, 2010. – 656 с.;

14. Блам Э. Сеть. Как устроен и как работает Интернет. / Э. Блам / Пер. с англ. – М.: АСТ, 2014. – 320 с.;

15. Сергиенко Ю. Компьютерные сети. Принципы, технологии, протоколы. / Ю. Сергиенко – СПб.: Питер, 2014. – 944 с.;

16. David L. Stevens. Internetworking with TCP/IP: Client-Server Programming and Applications for the BSD Socket Version v. 3. / David L. Stevens, Douglas E. Comer – Edinburgh: Prentice Hall, 1993. – 389 p.;

17. Ли Дж. Использование Linux, Apache, MySQL и PHP для разработки Web-приложений. / Дж. Ли, У. Брент / Пер. с англ. – СПб.: Вильямс, 2004. – 432 с.;

18. Айвалиотис Д. Администрирование сервера NGINX. / Д. Айвалиотис – М.: ДМК Пресс, 2013. – 288 с.;

19. Арнольд М. Администрирование Apache. / М. Арнольд, К. Миллер / Пер. с англ. – М.: Лори, 2012. – 418 с.;

20. Unix и Linux: руководство системного администратора, 4-е издание / Немет Э., Снайдер Г., Хейн Т., Уэйли Б. / Пер. с англ. – М.: Вильямс, 2011. – 1312с.;

21. Операционная система UNIX / Робачевский А. М., Немнюгин С. А., Стесик О. Л. / Пер. с англ. – СПб.: БХВ-Петербург, 2010. – 656с.;

22. Шимонски Р. Освой самостоятельно Unix. 10 минут на урок. / Р. Шимонски / Пер. с англ. – М.: Вильямс, 2006. – 272 с.;

23. Эрик С. Реймонд. Искусство программирования для Unix. / Эрик С. Реймонд / Пер. с англ. – М.: Вильямс, 2005. – 544 с.;

24. Роббинс А. Unix. Справочник. Пер. с англ. 4-е издание. / А. Роббинс / Пер. с англ. – М.: КУДИЦ-ПРЕСС, 2007. – 864 с.;

25. The Apache Software Foundation [Електронний ресурс] .– Режим доступу: http://www.apache.org;

26. Apache Tomcat.– Режим доступу: http://tomcat.apache.org;

27. CERN httpd.– Режим доступу: http://www.w3.org/Daemon;

28. Cherokee HTTP Server .– Режим доступу: http://cherokee-project.com;

29. IIS [Електронний ресурс] .– Режим доступу: http://www.iis.net;

30. Lighttpd [Електронний ресурс] .– Режим доступу: http://www.lighttpd.net;

31. Nginx [Електронний ресурс] .– Режим доступу: http nginx.org;

32. Netcraft [Електронний ресурс] .– Режим доступу: http://news.netcraft.com/archives/category/web-server-survey;

33. Форта Б. Освой самостоятельно SQL. 10 минут на урок. / Б. Форта / Пер. с англ. – М.: Вильямс, 2005. – 287 с.;

34. Кузнецов М. MySQL 5. / М. Кузнецов, И. Симдянов – СПб.: БХВ-Петербург, 2010. – 1007 с.;

35. Гольцман В. MySQL 5.0. / В. Гольцман – СПб.: Питер, 2010. – 253 с.;

36. MySQL. Оптимизация производительности / Б. Шварц, П. Зайцев, В. Ткаченко и др. / Пер. с англ. – М.: Символ, 2010. – 823 с.;

37. Талманн Л. Обеспечение высокой доступности систем на основе MySQL / Талманн Л., Киндал М., Белл Ч. / Пер. с англ. – СПб.: БХВ-Петербург, 2012. – 624 с.;

38. Ульман Л. Основы программирования на PHP. / Л. Ульман / Пер. с англ. – М.: ДМК Пресс, 200. – 288 с.;

39. Уолл Л. Изучаем Perl. / Л. Уолл, Т. Кристиансен / Пер. с англ. – СПб.: БХВ-Петербург, 2002. – 288 с.;

40. Сиерра К. Изучаем Java / К. Сиерра, Б. Бейтс / Пер. с англ. – М.: O'Reilly, 2012. – 708 с.;

41. Кузнецов М. В. PHP. Пракгика создания Web-сайтов. / М. В. Кузнецов, И. В. Симдянов – СПб.: БХВ-Петербург, 2009. – 1264 с.;

42. Гаевский А. Ю. Самоучитель по созданию Web-страниц: HTML, JavaScript, Dynamic HTML / А. Ю. Гаевский, В. А.Романовский – К.: А.С.К., 2002.– 472 с.

43. Гутманс Э. PHP5. Профессиональное программирование / Э. Гутманс, С.Баккен, Д.Ретанс / За ред. Э. Гутманс. – СПб.: Символ Плюс, 2004. – 432 с.;

44. Дронов В.І. PHP, MySQL и Dreamweaver MX 2004 Разработка интерактивных Web-сайтов / В.І. Дронов – СПб.: БХВ-Петербург, 2005. – 448 с.;

45. Вікіпедія, вільна енциклопедія.– Режим доступу: http://uk.wikipedia.org/wiki/XAMPP;

46. htmlbook.ru .– Режим доступу: http://htmlbook.ru/webserver/xampp;

47. Денвер: Локальний сервер [Електронний ресурс] Режим доступу: http://www.

48. Microsoft .– Режим доступу: http://windows.microsoft.com/uk-ua/windows/what-is-encryption#1TC=windows-7;

49. Фергюсон Н. Практическая криптография. / Н. Фергюсон, Б. Шнейер / Пер. с англ. – М.: Вильямс, 2005. – 424 с.;

50. Мэйволд Э. Безопасность сетей. / Э. Мэйволд / Пер. с англ. – М.: ЭКОМ, 2010. – 342 с.;

51. Шнайдер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. / Б. Шнайдер / Пер. с англ. – М.: Триумф, 2002. – 816 с.;

52. Венбо М. Современная криптография. Теория и практика. / М. Венбо / Пер. с англ. – М.: Вильямс, 2005. – 768 с.;

53. Алферов А. П. / Основы криптографии / А. П. Алферов, А. Ю. Зубов, А. С. Кузьмин, А. В. Черемушкин – М.: Гелиос, 2002. – 480 с.;

54. Javascript.ru [Електронний ресурс] .– Режим доступу: http://javascript.ru/unsorted/id;

55. Panopticlick.– Режим доступу: http://panopticlick.eff.org;

56. Мейер М. Теория реляционных баз данных / М. Мейер – М.: Мир, 1987. – 608 с.;

57. Harlick R.M. "Image Segmentation Techniques". R.M. Harlick, L.G. Shapiro. – Computer Vision, Graphics and Image Processing, - 1985, - pp.100-132.

58. Марков А. С. Лисовский К.Ю. «Базы данных. Введение в теорию и методологию. - Финансы и статистика»-2006,-pp 24-35.

59. Hlavac V. Image Processing, Analysis, and Machine Vision, International Student Edition // V. Hlavac, M. Sonka, R. Boyle – Thomson Learning, part of the Thomson Corporation – 2008 – 829p.