Вінницький національний технічний університет
Факультет інформаційних технологій та комп’ютерної інженерії
Кафедра комп’ютерних наук

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему «Інформаційна технологія для прогнозування комунальних платежів»

Виконав: студент 2 курсу,
групи 1КН-18 м
спеціальності 122 «Комп'ютерні науки»
Перегуда А. В.
Керівник: к.т.н, доц. Колодний В. В.
Рецензент: к.т.н, доц. Черноволик Г. О.

Вінниця - 2019 року

ЗАТВЕРДЖУЮ

Завідувач кафедри ____КН_____

д. т. н., проф., Яровий А. А.

_____

(підпис)

«_____»_____2019 року

ЗАВДАННЯ

на магістерську кваліфікаційну роботу на здобуття кваліфікації магістра зі спеціальності: 122 – «Комп'ютерні науки»_____

08-22.МКР.016.18.000.ПЗ

Магістранта групи 1КН-18м Перегуди Андрія Вікторовича

Тема магістерської кваліфікаційної роботи: «Інформаційна технологія для прогнозування комунальних платежів»

Вхідні дані: мова програмування об'єктно-орієнтована; мультиплатформність; мінімальна кількість років для вибірки - 3; врахування розрахункового періоду - 12.

Короткий зміст частин магістерської кваліфікаційної роботи:

1. Графічна: приклад роботи програмного додатку; загальна структурна схема; діаграма діяльності системи; структура інформаційної технології.
2. Текстова (пояснювальна записка): Вступ, аналіз методів прогнозування комунальних платежів, розробка моделі інформаційної технології для прогнозування комунальних платежів, проектування інформаційної технології для прогнозування комунальних платежів, економічна частина, висновки, перелік використаних джерел, додатки.

АНОТАЦІЯ

У даній магістерській кваліфікаційній роботі було спроектовано, реалізовано й протестовано інформаційну технологію для прогнозування комунальних платежів.

Було проведено аналіз сучасних програм-аналогів та наведено коротку порівняльну характеристику знайдених програм-аналогів. Було розроблено математичну модель прогнозування комунальних платежів, досліджено методи, які можуть бути використані.

Програмну було написано мовою JavaScript з використанням React.js, Node.js та інтегрованого середовища розробки WebStorm для пристроїв під управлінням будь-якої операційної системи в якій є можливість переглядання web-сторінок.

ABSTRACT

In this master's qualification work, information technology for forecasting utility bills was designed, implemented and tested.

An analysis of modern analog programs was performed and a brief comparative description of the found analog programs was given. A mathematical model of utility billing forecasting has been developed and methods that can be used are investigated.

The program was written in JavaScript using React.js, Node.js and an integrated development environment named WebStorm for devices running any operating system that has the ability to view web pages.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Регулярно люди стикаються із питанням оплати комунальних платежів. Часто, така задача віднімає надто багато часу, так як необхідно провести розрахунки, результати яких можуть варіюватися в залежності від об’єму споживаних послуг. Можна зекономити значну частину часу, якщо перекласти дану роботу на автоматизовану систему розрахунку. Опираючись на накопичені дані є змога провести аналіз, спрогнозувати якими можуть бути майбутні трати енергоносіїв в залежності від пори року та спланувати домашній бюджет опираючись на цю інформацію.

Такий підхід дозволяє знизити ймовірність похибки в розрахунках, через мінімізацію впливу людини на процес, та зробити роботу з комунальними платежами значно легшою і приємнішою.

Проблема доступноті програмного забезпечення для обліку комунальних платежів потребує вирішення шляхом створення безкоштовного, або дешевого продукту, який повинен бути простим у вивченні та використанні.

Надто складна організація програмного забезпечення відлякує потенційних користувачів та не викликає бажання користуватися даним рішенням, що ніяк не змінить ситуацію на ринку. Потрібно врахувати даний факт та, по можливості, спростити інтерфейс для кращого розуміння користувачем дій необхідних для роботи з програмним забезпеченням. Максимально лаконічний, інформативний та простий продукт западе до душі індивідуальним користувачам та зможе допомогти їм у вирішенні цієї не надто приємної задачі.

Якомога більше виключення “людинного фактору” забезпечить точність та актуальність розрахунків, не дивлячись на фактор плаваючого тарифу, котрий залежить від обсягу споживання енергоносіїв споживачем та, в деяких випадках, часу споживання ресурсів.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного

університету 22 К1 «Моделі, методи, технології та пристрої інтелектуальних інформаційних систем управління, економіки, навчання та комунікацій» та плану наукової та навчально-методичної роботи кафедри.

Мета та завдання дослідження. Метою дослідження магістерської кваліфікаційної роботи є підвищення точності прогнозування комунальних платежів.

Для досягнення наведеної мети необхідно вирішити наступні завдання:

1) Провести аналіз існуючих рішень;
2) Вдосконалити інформаційну технологію прогнозування комунальних платежів;
3) Розробити структуру інформаційної системи для прогнозування комунальних платежів;
4) Сформувати вимоги до програмного забезпечення;
5) Розробити клієнт-серверний програмний додаток.

Об’єкт дослідження – процес прогнозування комунальних платежів.

Предмет дослідження – інформаційна технологія та програмні засоби для прогнозування комунальних платежів.

Методи дослідження. У роботі використані наступні методи наукових досліджень: системного аналізу для аналізу структури інформаційної системи, метод Ньютона для реалізації інформаційної технології до прогнозування комунальних платежів, об’єктно-орієнтованого програмування для автоматизації розрахунків.

Наукова новизна одержаних результатів полягає в наступному:

– вдосконалено інформаційну технологію для прогнозування комунальних платежів, яка відрізняється від існуючих аналогів використанням методу інтерполяції Ньютона, що забезпечило підвищення точності прогнозування комунальних платежів;

– вдосконалено модель прогнозування комунальних платежів, яка відрізняється від існуючих звуженням кожного періоду, що використовується для інтерполяції, що забезпечує підвищення точності прогнозування.

Практичне значення одержаних результатів полягає у наступному:

1. Розроблено алгоритм для прогнозування комунальних платежів.
2. Розроблено інформаційну технологію для прогнозування комунальних платеджів.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним застосуванням математичних методів під час доведення наукових положень, строгим виведенням аналітичних співвідношень, порівнянням результатів з відомими, та збіжністю результатів математичного моделювання з результатами, що отримані під час впровадження розроблених програмних засобів.

Особистий внесок магістранта. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно.

Апробація результатів роботи. Результати роботи були апробовані на міжнародній науково-практичній конференції «The results of scientific mind's development: 2019» (Сеул, 2019 р.) [1].

1 АНАЛІЗ МЕТОДІВ ПРОГНОЗУВАННЯ КОМУНАЛЬНИХ ПЛАТЕЖІВ

1.1 Аналіз предметної області

Прогнозування – це сукупність прийомів, способів, які дають змогу на підставі аналізу колишніх (ретроспективних) внутрішніх і зовнішніх зв'язків (даних) зробити висновки про можливий розвиток економіки у майбутньому. Нині кількість відомих методів і прийомів прогнозування перевищує 150. Вибір методів прогнозування здійснюється відповідно до характеру об'єкта та вимог, які висуваються до інформаційного забезпечення прогнозів [2].

Досвід, накопичений сучасною прогностикою, показує, що за всієї Прогностика (грец). – це наука, що розробляє теоретичні основи та методи прогнозування.

В залежності від ступеня формалізації), методи прогнозування можна об'єднати в три групи:

- фактографічні (формалізовані);
- експертні (інтуїтивні);
- комбіновані.

Фактографічні методи прогнозування ґрунтуються на достатньому інформаційному матеріалі про об'єкт прогнозування та його минулий розвиток. До них належить група методів прогнозної екстраполяції та моделювання.

Методи екстраполяції базуються на припущенні того, що закономірність (тенденція) розвитку об'єкта в минулому буде незмінною протягом певного часу і в майбутньому. Залежно від особливостей змін рівнів у динамічних рядах екстраполяції можуть бути простими і складними.

Методи простої екстраполяції базуються на припущенні відносної стійкості в майбутньому абсолютних значень рівнів, середнього рівня ряду, середнього абсолютного приросту, середнього темпу зростання.

Методи складної екстраполяції базуються на визначенні основної тенденції, тобто використанні статистичних формул, які описують тренд. Тренд

– це відображення певною функцією тенденції ряду динаміки .Його зображують у вигляді гладкої кривої (траєкторії). Тренд характеризує головну закономірність руху об’єкта в часі. Складні методи екстраполяції поділяються на два типи: адаптивні і аналітичні. До адаптивних методів прогнозування належать методи ковзної та експотенціальної середніх, метод гармонійних ваг, авторегресія. До аналітичних методів прогнозування (кривих зростання) відноситься метод найменших квадратів [3].

Методам багатофакторного моделювання належить особливе місце в сучасному прогнозуванні. Це методи логічного, інформаційного та статистичного моделювання.

До логічного моделювання належать методи прогнозування за історичною аналогією, побудови сценарію, дерева цілей і т.д.

Методи інформаційного моделювання є специфічною галуззю прогнозування і будуються на підставі вивчення джерел масової інформації, які містять певні необхідні, логічно впорядковані гіпотези.

Методи статистичного моделювання є найбільш поширеними в прогнозуванні. Вони поділяються на дві групи:

- методи прогнозування на підставі одиничних рівнянь регресії;

- методи прогнозування на підставі системи рівнянь взаємозв’язаних рядів динаміки.

Інтуїтивні (експертні) методи базуються на використанні експертної інформації. Ними користуються тоді, коли прогнозуються процеси, які не мають історичних аналогів, коли іншими методами прогнозування неможливо формалізувати оцінку впливу на розвиток об’єкта багатьох чинників. Розрізняють індивідуальні та колективні експертні оцінки.

Сутність методів індивідуальних експертних оцінок полягає в тому, що кожен експерт дає оцінку незалежно від інших, а потім, за допомогою певних прийомів ці оцінки об’єднуються і узагальнюються. Найпоширенішими серед методів індивідуальної експертної оцінки є:

❿ метод “інтерв’ю” – експерт безпосередньо опитує спеціалістів;

- метод “аналітичних записок” – передбачає всебічний аналіз прогнозованого економічного явища або процесу з підготовкою відповідної доповідної записки;
- метод “написання сценарію” – ґрунтується на визначенні логіки розвитку прогнозованого об’єкта за різних умов.

Колективні (групові) експертні оцінки як методи прогнозування ґрунтуються на спільній праці експертів і передбачають визначення колективом спеціалістів сумарної оцінки стану об’єкта в майбутньому. Найбільш поширеними методами колективної експертної оцінки є метод мозкової атаки, метод комісій, метод “Дельфі”, матричний метод та інші способи колективної генерації ідей. Серед колективних методів експертних оцінок доцільно виділити методи :

1. Метод колективної експертної комісії – група укладачів прогнозу уточнює головні напрями розвитку об’єкта, будує матрицю, у якій відображена генеральна мета, підцілі та засоби їх досягнення, і розробляє перелік питань для експертів.

2. Метод колективної генерації – активізує творчий потенціал експертів у разі шукання виходу з проблемної ситуації. Використання методу передбачає зіткнення протилежних напрямів думок і рекомендацій щодо вирішення конкретної проблеми: генерацію ідей і наступне руйнування (через критику).

3. Метод Дельфі – використовують для визначення й оцінки ймовірності настання тих або інших подій. Він дає змогу узагальнити думки окремих експертів в узгоджену групову думку. Особливість методу Дельфі полягає в тому, що він передбачає анонімність експертів, використання результатів попереднього туру опитування , статистичну характеристику групової відповіді.

Комбіновані методи прогнозування об’єднують експертні і фактографічні методи. Прикладом може бути метод ПАТТЕРН, при якому експерти формулюють колективні судження на основі використання принципу “дерева цілей”. Англ. Слово Pattern означає “шаблон”, “модель”, “схема” і

перекладається за першими літерами як”допомога плануванню за допомогою відносних показників технічної оцінки”.

Моделі математичного прогнозування дають змогу прогнозувати розміщення виробництва, темпи розвитку галузі, асортимент і обсяг випуску продукції, попит споживачів, використання ресурсів в галузі і потребу в матеріально-технічних ресурсах [4].

1.2 Аналіз програм-аналогів

Системи контролю та прогнозування комунальних платежів є не дуже великим сектором програмного забезпечення. Рішення, що наявні на ринку є недосконалими та, здебільшого, представляють собою візуалізацію бази клієнтів компаній, що надають послуги, тобто, про використання даних продуктів фізичними особами навіть не йде розмова. Крім того, дані програмні засоби мають обмеження стосовно платформи використання. Основним недоліком існуючих рішень є висока ціна ліцензування. Серед сучасних рішень для ведення обліку комунальних платежів існують наступні: “Коммунальные платежи 1.0”, “1С: Учёт в управляющих компаниях”. Результати їх порівняння за основними характеристиками наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння систем оцінювання знань за основними характеристиками

	Доступність для індивідуального	Лаконічність інтерфейсу	Доступність для різних платформ	Ведення обліку	Прогнозування

	користування				
«Коммунальные платежи 1.0»	-	-	-	+	+
«1С: Учёт в управляющих компаниях»	-	-	-	+	-
“Универсальная Система Учета”	+	-	-	+	+

“Комунальні платежі 1.0” – невелика програма для розрахунку комунальних платежів. Для розрахунку враховуються тарифи на комунальні послуги, показники лічильників гарячої, холодної води та електроенергії [5]. Приклад роботи програми представлено на рисунку 1.1.

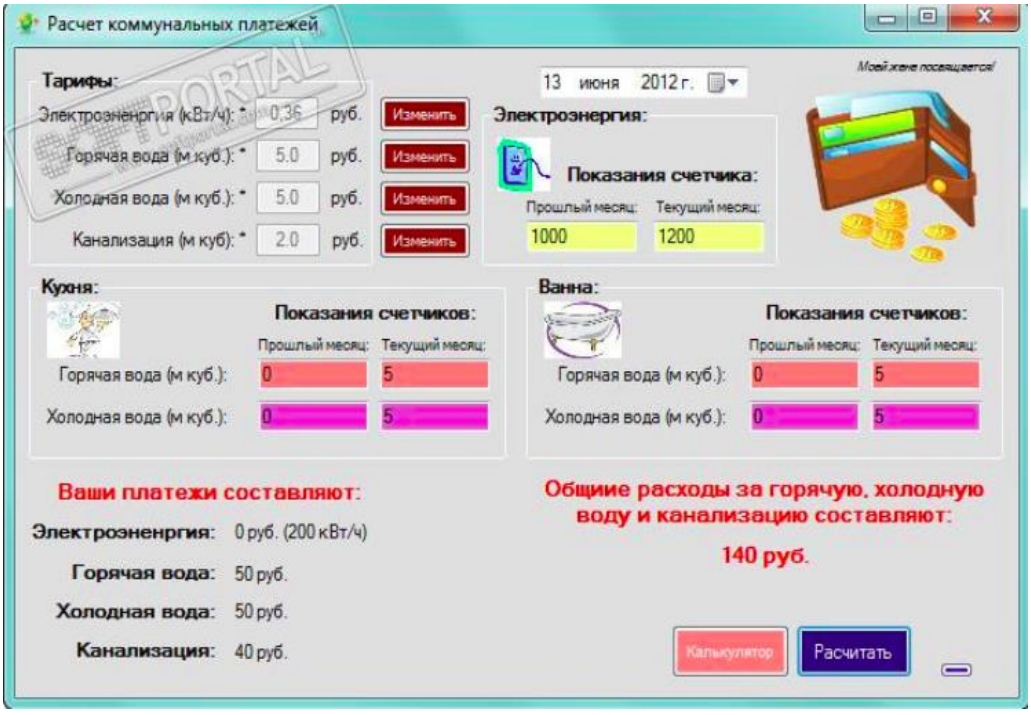

Рисунок 1.1 – Приклад роботи програми “Коммунальные платежи 1.0”.

“1С: Учёт в управляющих компаниях” – зручна і функціональна програма для автоматизації підприємств сфери ЖКГ. Від управління об'єктами інфраструктури, нарахування квартплати до бухгалтерського та податкового обліку [6]. Приклад роботи програми представлено на рисунку 1.2.

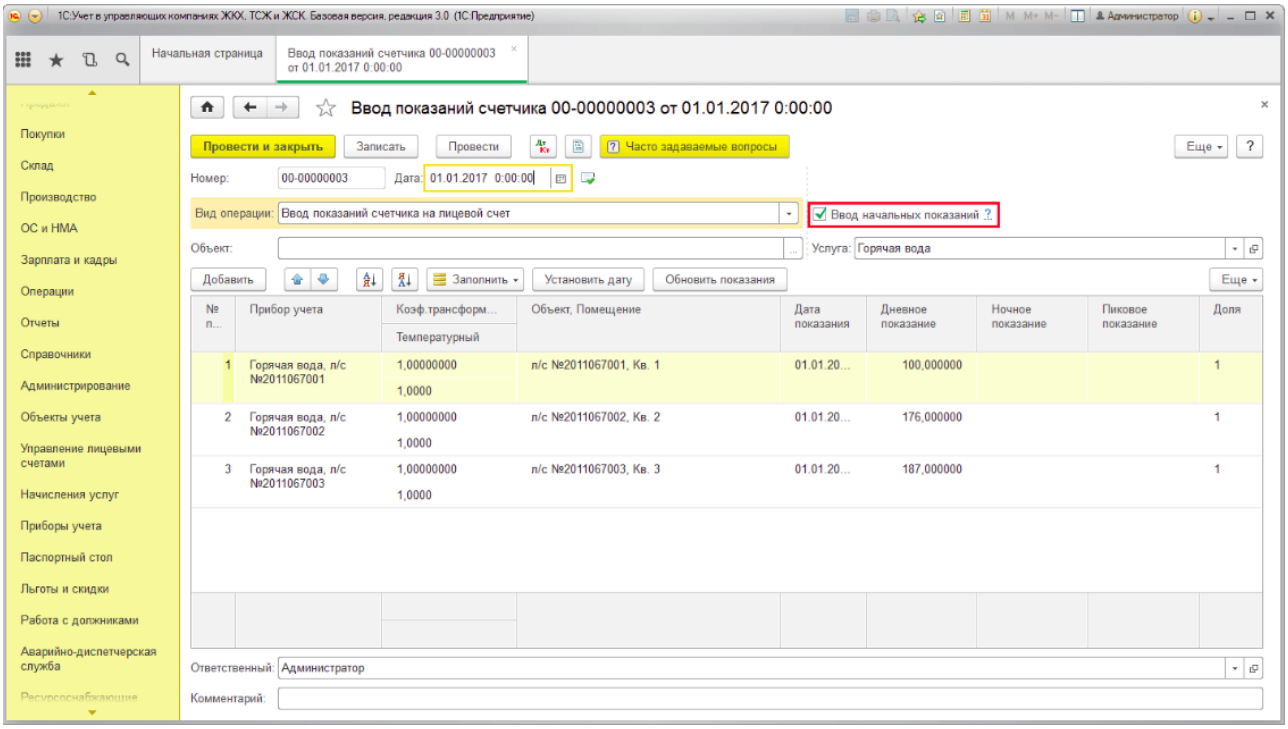

Рисунок 1.2 – Приклад роботи програми “1С: Учёт в управляющих компаниях”

Компанія Універсальної Системи Обліку (УВУ) має безліч проектів, метою яких є автоматизація підприємств різної спрямованості. Комунальні послуги теж необхідний такий проект. Програма обліку комунальних послуг існує, і її можна завантажити прямо зараз. Існує базова версія і розроблена індивідуально. Перша має загальний функціонал, відповідний будь-якої компанії з обслуговування населення у сфері благоустрою. Друга ж, створюється з урахуванням індивідуальних потреб конкретної компанії і відображає в своєму інтерфейсі і функціоналі всі параметри і характеристики, специфіку роботи. Обидва варіанти можуть бути модифіковані та доповнені платними функціями, що не відносяться до основної комплектації. Приклад роботи програми представлено на рисунку 1.3.

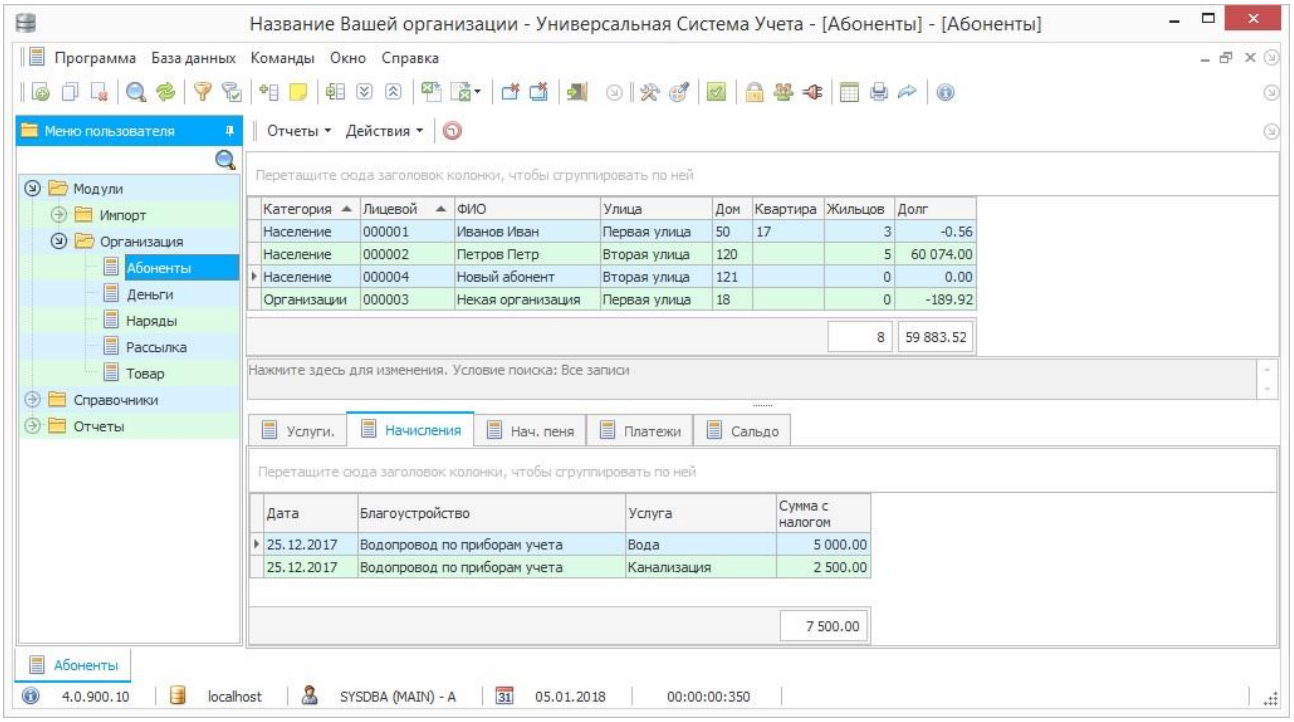

Рисунок 1.3 – Приклад роботи програми “Универсальная Система Учета”

1.3 Формулювання вимог

Відповідно до вищенаведених задач та методів їх розв’язання, було прийнято рішення розробити інформаційну технологію для прогнозування комунальних платежів.

Експлуатація розробленої інформаційної технології передбачається шляхом внесення поточних показників та звернення до бази знань, що містить попередні показники використовуючи web-переглядач, користувачами якого є особи, що бажають тримати під контролем видатки на комунальні послуги. Дана інформаційна технологія повинна виконувати наступні функції:

• створення кабінету коритувача;

• отримання даних про користувачів, при реєстрації в кабінеті;

• створення власного токену;

Вимоги до способів і засобів зв'язку для інформаційного обміну між компонентами системи:

Інформаційний обмін повинен здійснюватися з базою даних та повинен бути гнучким для горизонтального масштабування архітектури.

Вимоги по архітектурі:

Рішення, що розробляється, повинене бути масштабованим та гнучким. Це повинно бути досягнуто за рахунок використання гнучкої мови програмування та динамічної типізації.

Вимоги до інтерфейсу:

Програмна частина повинна бути дружньою до користувача, інтуїтивно зрозумілою та з легко читаємим графічним інтерфейсом, що дозволяє здійснювати в повній мірі всі функціональні можливості інформаційної технології.

Вимоги до апаратних і програмних ресурсів:

Передбачається робота web-переглядача. Запити до бази даних повинні бути побудовані таким чином, щоб забезпечувати максимальну швидкодію інформаційної технології.

Вимоги до програмного забезпечення системи:

Інформаційна технологія для прогнозування комунальних платежів повинна працювати на базі будь-якої десктопної чи мобільної операційної системи з можливістю перегляду web-сторінок.

Усі вищенаведені складові при поєднанні забезпечують повнофункціональне навантаження проекту.

1.4 Висновок

Було проведено аналітичне дослідження по роботі з поняттям прогнозування комунальних платежів та визначено його концепцію.

В даному розділі було проведено аналіз функціональних можливостей програм-аналогів, розглянуто основні принципи прогнозування комунальних платежів; сформульовано мету та задачі подальшого дослідження.

У першому розділі було виявлено, що для забезпечення ефективної роботи із программами для прогнозування комунальних платежів необхідно забезпечити доступність продукту для індивідуального використання та легкість орієнтування користувача в інтерфейсі продукту. Також, виявлено, що звущення розрахункового періоду та збільшення кількості початкових даних про минулі періоди позитивно впливають на точність прогнозування. Це приводить до висновку, що ряд функцій в наявних рішеннях є надлишковими та здатні збити користувача з пантелику, отже підлягають оптимізації та апроксимації.

Було розглянуто функціональні можливості деаких програм-аналогів й представлено їх порівняльну характеристику.

2 РОЗРОБКА МОДЕЛІ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ДЛЯ ПРОГНОЗУВАННЯ КОМУНАЛЬНИХ ПЛАТЕЖІВ

2.1 Шаблон проектування інформаційної технології для прогнозування комунальних платежів

Стандартним шаблоном проектування в розробці рішень на React.js є паттерн проектування MVC.

Модель–вигляд–контролер (MVC) – шаблон, який використовується під час проектування та розробки програмного забезпечення. Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача [5].

Мета шаблону – гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

У рамках архітектурного шаблону модель–вигляд–контролер (MVC) програма поділяється на три окремі, але взаємопов'язані частини з розподілом функцій між компонентами. Модель (Model) відповідає за зберігання даних і забезпечення інтерфейсу до них. Вигляд (View) відповідальний за представлення цих даних користувачеві. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача (зміна положення курсора миші, натискання кнопки, ввід даних в текстове поле) і передає дані у модель.

Модель є центральним компонентом шаблону MVC і відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель стосується прямого керування даними, логікою та правилами застосунку.

Вигляд може являти собою будь-яке представлення інформації, одержуване на виході, наприклад графік чи діаграму. Одночасно можуть співіснувати кілька виглядів (представлень) однієї і тієї ж інформації, наприклад гістограма для керівництва компанії й таблиці для бухгалтерії.

Контролер одержує вхідні дані й перетворює їх на команди для моделі чи вигляду.

Модель інкапсулює ядро даних і основний функціонал їхньої обробки і не залежить від процесу вводу чи виводу даних.

Вигляд може мати декілька взаємопов'язаних областей, наприклад різні таблиці і поля форм, в яких відображаються дані.

У функції контролера входить відстеження визначених подій, що виникають в результаті дій користувача. Контролер дозволяє структурувати код шляхом групування пов'язаних дій в окремий клас. Наприклад у типовому MVC-проекті може бути користувацький контролер, що містить групу методів, пов'язаних з управлінням обліковим записом користувача, таких як реєстрація, авторизація, редагування профілю та зміна пароля.

Зареєстровані події транслюються в різні запити, що спрямовуються компонентам моделі або об'єктам, відповідальним за відображення даних. Відокремлення моделі від вигляду даних дозволяє незалежно використовувати різні компоненти для відображення інформації. Таким чином, якщо користувач через контролер внесе зміни до моделі даних, то інформація, подана одним або декількома візуальними компонентами, буде автоматично відкоригована відповідно до змін, що відбулися.

2.2 UML діаграми інформацийної технології для проектування комунальних платежів

Для опису структури проекту використаємо деякі діаграми UML. Зокрема: діаграму класів, діаграму компонентів, діаграму діяльності.

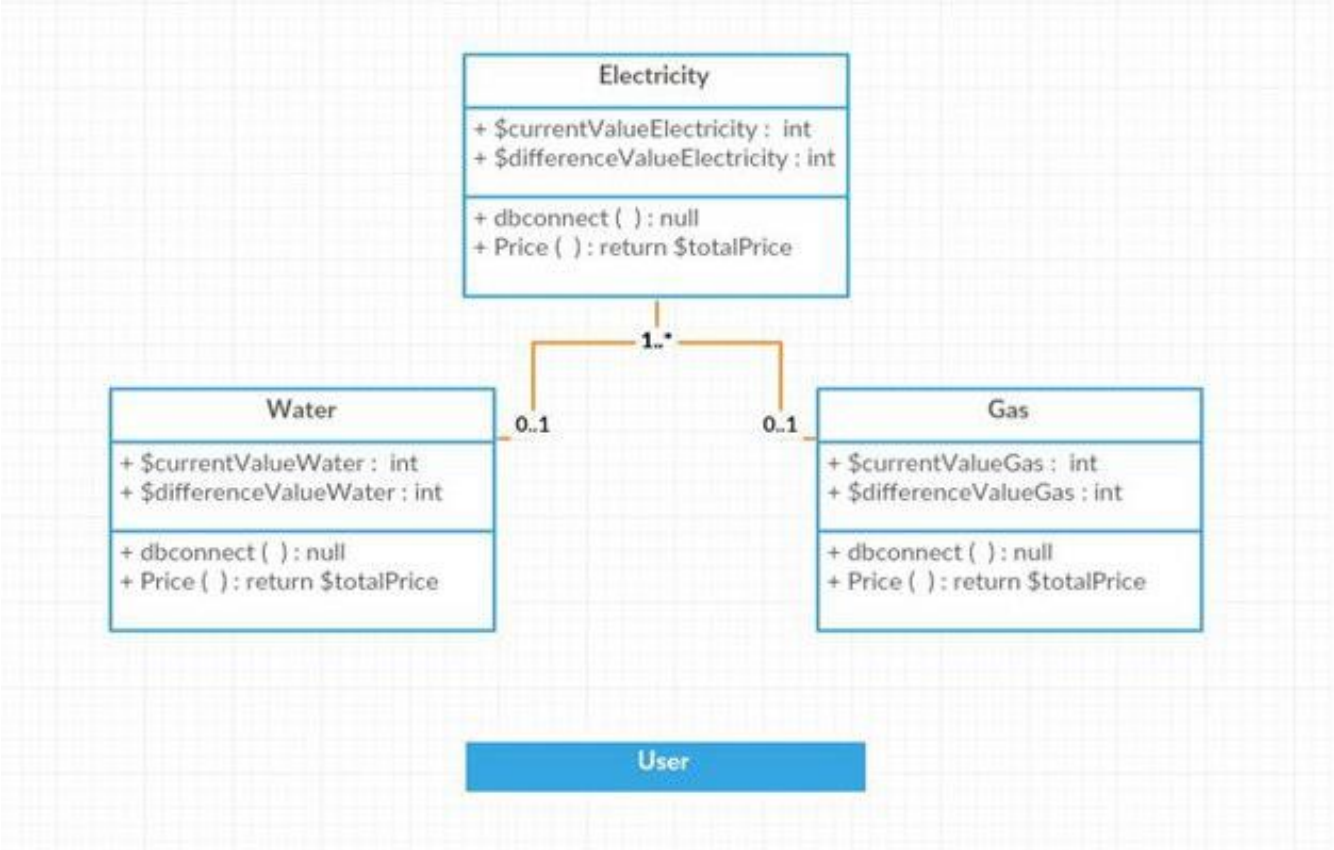

Рисунок 2.1 – Діаграма класів інформаційної технології для прогнозування комунальних платежів

Уніфікована мова моделювання UML (Unified Modeling Language) - це наступник того покоління методів об'єктно-орієнтованого аналізу і проектування, які з'явилися наприкінці 80-х і початку 90-х років. Створення UML фактично почалося в кінці 1994 р., коли Граді Буч і Джеймс Рамбо почали роботу з об'єднання їх методів Booch [Буч-1999] і OMT (Object Modeling Technique) під егідою компанії Rational Software. До кінця 1995 р. вони створили першу специфікацію об'єднаного методу, названого ними Unified Method, версія 0.8. Тоді ж у 1995 р. до них приєднався творець методу OOSE (Object-Oriented Software Engineering) Івар Якобсон. Таким чином, UML є прямим об'єднанням і

уніфікацією методів Буча, Рамбо і Якобсона, однак доповнює їх новими можливостями [7].

UML знаходиться в процесі стандартизації, проведеному консорціумом OMG (Object Management Group), в даний час він прийнятий в якості стандартного мови моделювання і отримав широку підтримку. UML прийнятий на озброєння практично усіма найбільшими компаніями - виробниками програмного забезпечення (Microsoft, IBM, Hewlett-Packard, Oracle, Sybase та ін.) Крім того, практично всі світові виробники CASE-засобів, крім Rational Software (Rational Rose), підтримують UML у своїх продуктах (Paradigm Plus (CA), System Architect (Popkin Software), Microsoft Visual Modeler та ін.)

Діаграма класів вказує типи класів системи і різного роду зв'язки, що виникають між ними. На цих діаграмах вказуються також їхні атрибути, операції(методи) та обмеження, що накладаються на зв'язки між класами (рис. 2.1).

Загальна структурна схема показує, як виглядає модель системи на фізичному рівні. На схемі зображені компоненти програмного забезпечення та зв'язку між ними. При цьому виділяють два типи компонентів: виконувані компоненти і бібліотеки коду [8].

Кожен клас моделі перетвориться в компонент вихідного коду. Після створення вони відразу додаються до загальної структурної схеми. Між окремими компонентами зображують залежності, відповідні залежностям на етапі компіляції або виконання програми (рис. 2.2).

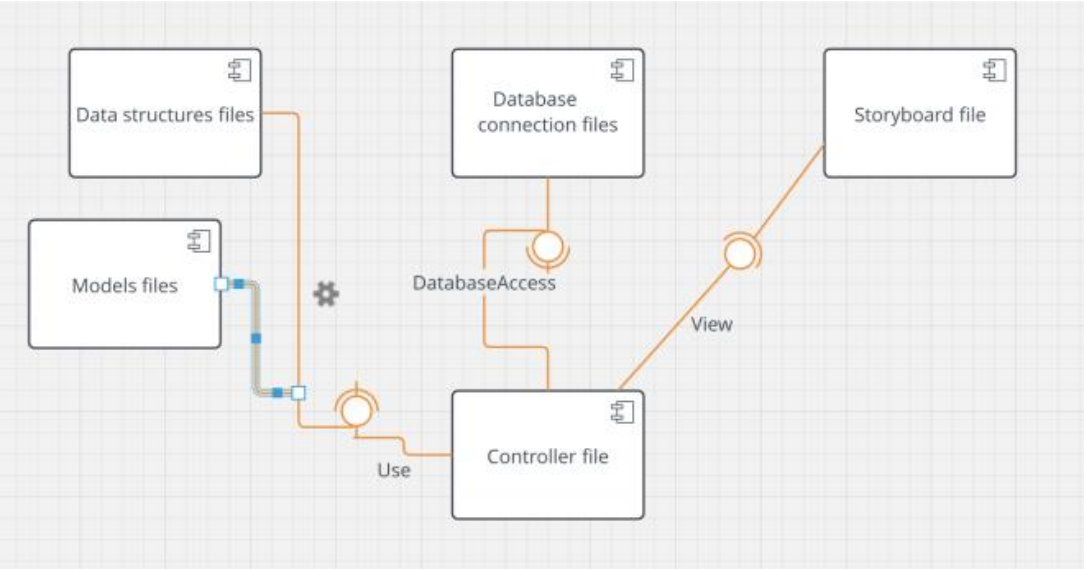

Рисунок 2.2 – Загальна структурна схема

Діаграми діяльності візуалізує граф діяльностей, який є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.

Дія є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів. Одна із цих множин, або обидві водночас, можуть бути порожніми. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

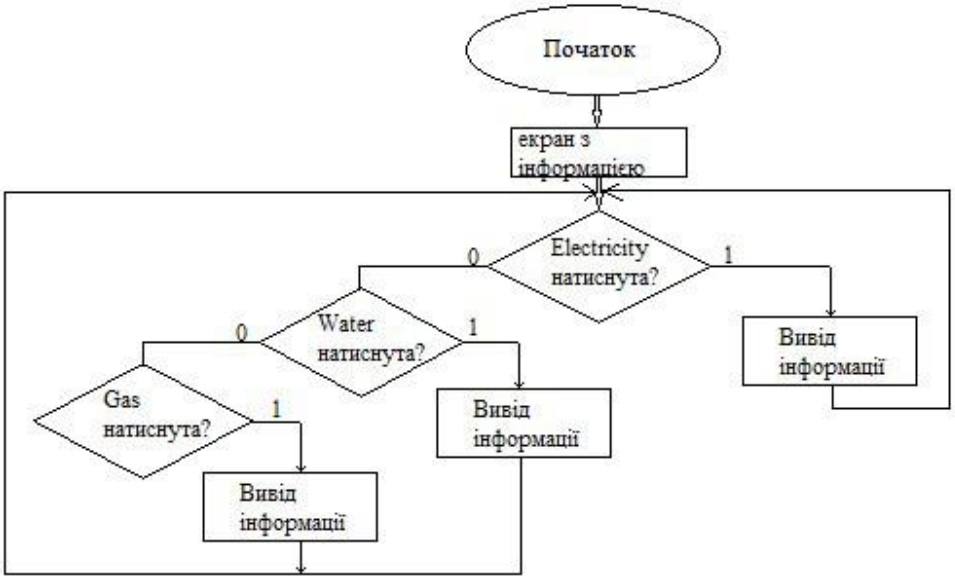

Рисунок 2.3 – Діаграма діяльності інформаційної технології для прогнозування комунальних платежів

2.3 Розробка математичної моделі інформаційної технології для прогнозування комунальних платежів

Інтерполяція — в обчислювальній математиці спосіб знаходження проміжних значень величини за наявним дискретним набором відомих значень.

Багатьом із тих, хто стикається з науковими та інженерними розрахунками часто доводиться оперувати наборами значень, отриманих експериментальним шляхом чи методом випадкової вибірки. Як правило, на підставі цих наборів потрібно побудувати функцію, зі значеннями якої могли б з високою точністю збігатися інші отримувані значення. Така задача називається апроксимацією кривої. Інтерполяцією називають такий різновид апроксимації, при якій крива побудованої функції проходить точно через наявні точки даних.

Існує також близька до інтерполяції задача, що полягає в апроксимації якої-небудь складної функції іншою, простішою функцією. Якщо деяка функція занадто складна для продуктивних обчислень, можна спробувати обчислити її значення в декількох точках, а за ними побудувати, тобто інтерполювати, простішу функцію. Зрозуміло, використання спрощеної функції не дозволяє

одержати такі ж точні результати, які давала б початкова функція. Але, для деяких класів задач, досягнутий виграш у простоті і швидкості обчислень може переважити отриманий огріх у результатах.

Варто також згадати і зовсім інший різновид математичної інтерполяції, відому за назвою «інтерполяція операторів». До класичних робіт з інтерполяції операторів відносяться теорема Рісса-Торіна[en]) і теорема Марцинкевича, що є основою для багатьох інших робіт[9].

Інтерполяційні формули Ньютона — це формули обчислювальної математики, що застосовуються для поліноміальної інтерполяції.

Якщо вузли інтерполяції рівновіддалені і впорядковані за величиною, так що $x_{i+1} - x_i = h = \text{const}$, тобто $x_i = x_0 + ih$, тоді інтерполяційний многочлен можна записати у формі Ньютона.

Інтерполяційні поліноми в формі Ньютона зручно використовувати, якщо точка інтерполяції знаходиться поблизу початку (пряма формула Ньютона) або кінця таблиці (зворотна формула Ньютона).

У випадку рівновіддалених центрів інтерполяції, що знаходяться на одиничній відстані один від одного, справедливою буде коротка форма інтерполяційної формули Ньютона:

$$P_n(x) = \sum_{m=0}^{n} C_x^m \sum_{k=0}^{m} (-1)^{m-k} C_m^k \, f(k)$$

де C_x^m — узагальнені на область дійсних чисел біноміальні коефіцієнти.

Інтерполяційні поліноми в формі Ньютона зручно використовувати, якщо точка інтерполяції знаходиться поблизу початку (пряма формула Ньютона) або кінця таблиці (зворотна формула Ньютона).

Пряма (або перша) інтерполяціонная формула Ньютона, застосовується для інтерполяції вперед:

$$P_n(x) = y_0 + q\Delta y_0 + \frac{q(q-1)}{2!}\Delta^2 y_0 + \ldots + \frac{q(q-1)\ldots(q-n+1)}{n!}\Delta^n y_0$$

де $q = \frac{x - x_0}{h}$, $y_i = f_i$, а вирази виду $\Delta^k y_0$ — кінцеві різниці.

Зворотна (або друга) інтерполяціонная формула Ньютона, застосовується для інтерполяції назад:

$$q = \frac{x - x_n}{h}$$

$$P_n(x) = y_n + q\Delta y_{n-1} + \frac{q(q+1)}{2!}\Delta^2 y_{n-2} + \ldots + \frac{q(q+1)\ldots(q+n-1)}{n!}\Delta^n y_0$$

де

Для роботи даної інформацийної технології необхідно провести інтерполяцію місяцях кожного записаного року, знайти сумму відповідних місяців у доступних роках та знайти середнє арифметичне значення для відповідних блоків.

$$K= \quad \delta min \le \frac{\sum_{j=1}^{n} m_{ji}}{n} \le \delta_{max} \quad ,$$

δmin де — мінімальний індекс діапазону місяців.

δ_{max} — максимальний індекс діапазону місяців.

$\sum_{j=1}^{n}$ — сума відповідних місяців у n роках.

2.4 Розробка структури інформаційної технології для прогнозування комунальних платежів

На вхід подаються дані за поточний розрахунковий період у вибраній категорії енергоносіїв, які потрібно внести до бази знань для формування рахунку за спожиті послуги по завчасно визначених тарифах, та подальшого прогнозування на основі вже доступних даних. Після цього відбувається звернення до бази знань із вибором відповідних даних за визначений розрахунковий період за попередніми роками. Потім відбувається отримання затребуваних даних та їх подальша обробка, після чого відбувається процес прогнозування в основі якого лежить метод інтерполяції Ньютона.

Завершальним є процес візуалізації отриманих результатів у вигляді діаграми та чисельних даних.

Структуру інформаційної технології для прогнозування комунальних платежів зображено на рисунку 2.4.

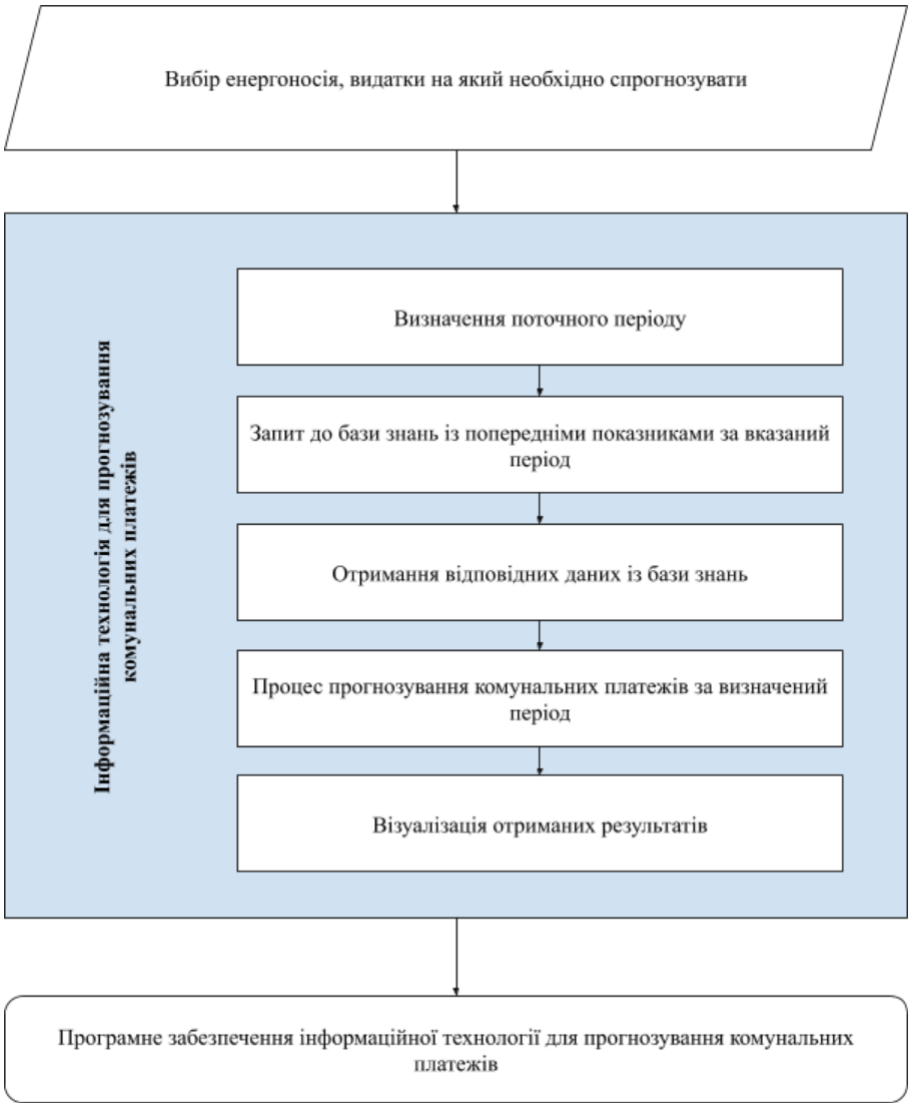

Рисунок 2.4 – Структура інформаційної технології прогнозування комунальних платежів

2.5 Висновок

У другому розділі було розроблену математичну модель інформаційної технології для прогнозування комунальних платежів, досліджено методи, що можуть бути використані для визначення та прогнозування комунальних платежів, для чого було обрано метод інтерполяції доступних даних за визначений період. Також, було розроблено та створено загальну структурну схему, що візуалізує зв’язки між різними компонентами в стореному програмному забезпеченні.

Діаграма класів відображає наявні в програмному забезпеченні класи, які були використані для розробки. Також, розроблена та представлена діаграма діяльності, яка описує алгоритм та принцип роботи розробленого програмного забезпечення.

Даний підхід, на думку магістранта, є одним із найефективніших при вирішенні задач прогнозування комунальних платежів.

3 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ДЛЯ ПРОГНОЗУВАННЯ КОМУНАЛЬНИХ ПЛАТЕЖІВ

3.1 Обгрунтування вибору системи управління базою даних

MongoDB — документо-орієнтована система керування базами даних (СКБД) з відкритим сирцевим кодом, яка не потребує опису схеми таблиць. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними і зручними у формуванні запитів.

Код MongoDB написаний на мові C++ і поширюється в рамках ліцензії AGPLv3.

MongoDB підтримує зберігання документів в JSON-подібному форматі, має досить гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів, підтримує журналювання операцій зі зміни і додавання даних в БД, може працювати відповідно до парадигми Map/Reduce, підтримує реплікацію і побудову відмовостійких конфігурацій [10].

У MongoDB є вбудовані засоби із забезпечення шардінгу (розподіл набору даних по серверах на основі певного ключа), комбінуючи який реплікацією даних можна побудувати горизонтально масштабований кластер зберігання, в якому відсутня єдина точка відмови (збій будь-якого вузла не позначається на роботі БД), підтримується автоматичне відновлення після збою і перенесення навантаження з вузла, який вийшов з ладу. Розширення кластера або перетворення одного сервера на кластер проводиться без зупинки роботи БД простим додаванням нових машин.

При розробці автори виходили з необхідності спеціалізації баз даних, завдяки чому їм вдалося відійти від принципу «один розмір під усе». За рахунок мінімізації семантики для роботи з транзакціями з'являється можливість вирішення цілого ряду проблем, пов'язаних з нестачею продуктивності, причому

горизонтальне масштабування стає простішим. Використовувана модель документів зберігання даних (JSON/BSON) простіше кодується, простіше управляється (у тому числі за рахунок застосування так званого "безсхемного стилю» (англ. schemaless style)), а внутрішнє угруповання релевантних даних забезпечує додатковий виграш в швидкодії. Нереляційний підхід досить зручний для створення баз даних, у яких горизонтальне масштабування означає розгортання на множині машин. Можливість забезпечувати найкращу продуктивність повинна існувати паралельно з підтримкою більшої функціональності, ніж це дозволяє використання пар «ключ-значення» (у чистому вигляді). Технологія баз даних має працювати скрізь, починаючи з серверів користувача та віртуальних машин і закінчуючи хмарними технологіями. MongoDB, на думку розробників, має заповнити розрив між простими сховищами даних типу «ключ-значення» (швидкими і легко масштабованими) і великими РСКБД (зі структурними схемами і потужними запитами).

Основні можливості MongoDB:

- Документо-орієнтоване сховище (проста та потужна JSON-подібна схема даних);
- Досить гнучка мова для формування запитів;
- Динамічні запити;
- Повна підтримка індексів;
- Профілювання запитів;
- Швидкі оновлення «на місці»;
- Ефективне зберігання бінарних даних великих обсягів, наприклад, фото та відео;
- Журналювання операцій, що модифікують дані в БД ;
- Підтримка відмовостійкості і масштабованості: асинхронна реплікація, набір реплік і шардінг;
- Може працювати відповідно до парадигми MapReduce;

СКБД управляє наборами JSON-подібних документів, що зберігаються в бінарному форматі в форматі BSON. Зберігання і пошук файлів в MongoDB відбувається завдяки викликам протоколу GridFS. Подібно до інших документо-орієнтованих СКБД (CouchDB тощо), MongoDB не є реляційною СКБД.

Є докладна і якісна документація, велике число прикладів і драйверів під популярні мови Java, C++, C#, PHP, Python, Perl, Ruby.

При випуску одразу було заявлено, що реліз MongoDB 1.0 готовий до використання у виробництві як одиничний хост, так і у зв'язках master/slave. Код цього релізу досить стабільний і перевірений в промисловій експлуатації протягом 1,5 років. За можливості MongoDB має бути розгорнута мінімум на двох серверах, використовуючи реплікацію Master/Slave. Це забезпечує наявність актуальних даних при виході з ладу однієї з СКБД. MongoDB — продукт досить молодий, і відтак у ньому зустрічаються помилки, з'являються нові можливості тощо. Характерний високий темп розробки (проект пишуть не тільки волонтери, а й компанія людей на повній зайнятості).

JSON (англ. JavaScript Object Notation, укр. запис об'єктів JavaScript, вимовляється джейсон) — це текстовий формат обміну даними між комп'ютерами. JSON базується на тексті, може бути прочитаним людиною. Формат дозволяє описувати об'єкти та інші структури даних. Цей формат головним чином використовується для передачі структурованої інформації через мережу (завдяки процесу, що називають серіалізацією).

Розробив і популяризував формат Дуглас Крокфорд.

JSON знайшов своє головне призначення у написанні веб-програм, а саме при використанні технології AJAX. JSON, що використовується в AJAX, виступає як заміна XML (використовується в AJAX) під час асинхронної передачі структурованої інформації між клієнтом та сервером. При цьому перевагою JSON перед XML є те, що він дозволяє складні структури в атрибутах, займає менше місця і прямо інтерпретується за допомогою JavaScript в об'єкти [11].

3.2 Обґрунтування вибору мови програмування для клієнтської частини

JavaScript (JS) — динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Мова JavaScript використовується для:

- написання сценаріїв веб-сторінок для надання їм інтерактивності;
- створення односторінкових веб-застосунків (ReactJS, AngularJS, Vue.js); програмування на стороні сервера (Node.js);
- стаціонарних застосунків (Electron, NW.js);
- мобільних застосунків (React Native, Cordova);
- сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite чи Apache Jmeter);
- всередині PDF-документів тощо.

Незважаючи на схожість назв, мови Java та JavaScript є двома різними мовами, що мають відмінну семантику, хоча й мають схожі риси в стандартних бібліотеках та правилах іменування. Синтаксис обох мов отриманий «у спадок» від мови C, але семантика та дизайн JavaScript є результатом впливу мов Self та Scheme [12][13].

Таблиця 3.1 – Порівняння мов програмування C++, C#, JavaScript, PYTHON

Можливість	C++	C#	JavaScript	PYTHON
Об’єктно-орієнтована	+	+	+	+
Статистична типізація	+	+	+	-
Динамічна типізація	-	+	+	+
Функціонально-орієнтована	+	+	+	+
Підтримка try/catch	-	+	+	+
Безкоштовність	-	-	+	-
Сумісність різних версій	-	-	+	-

Зважаючи на всі плюси JavaSript, а також на те, що він є у вільному доступі і є велика кількість безкоштовних бібліотек, я вибрав саме його для реалізації програмного забезпечення.

Основні оператори мови програмування, використані в процесі розробки:

- оператор циклічного перебору об’єктів for...in;
- оператор циклу for;
- оператор вибору if;

Цикли використовують для багаторазового виконання однакових команд, які знаходяться в тілі циклу.

Цикл for, цикл з чітко визначеною кількістю ітерацій. З переходом від ітерації до ітерації, змінюється змінна, яка є параметром циклу (на один менше, або на один більше).

Цикл for...in, цикл який виконується стільки разів, скільки є елементів у об’єкті що перебирається.

Оператор вибору if дозволяє нам виконувати, або не виконувати ту чи іншу ділянку коду, в залежності від того чи виконується умова.

Для забезпечення механізму наслідування використовується ключове слово extends.

3.3 Обґрунтування вибору мови програмування для серверної частини

При виборі засобів для розробки програмного проекту необхідно врахувати надзвичайно велику кількість різноманітних аспектів, найбільш важливим із яких є мова програмування, тому що вона в значній мірі визначає інші доступні засоби. При виборі мови програмування основними критеріями були продуктивність програмування, продуктивність роботи додатку та ефективність використання пам’яті.

Web–програмування стрімко розвивається, і перед back–end розробниками постає питання вибору між усталеними важкоатлетами Java, C, Perl і сучасними веб–орієнтованими мовами, такими як Node js, PHP. Наш вибір має величезне значення, накладаючи свій відбиток на роботу програми.

Node.js і PHP – два дуже популярних рішення для веб–розробки. PHP, мова сценаріїв, створений Rasmus Lerdorf в 1994 році, був одним з кращих мов епохи Web 1.0. Красномовним проявом успіху PHP є CMS (системи управління контентом), такі як WordPress, Joomla або Drupal, які забезпечують мільйони блогів і веб–порталів. Node.js є представником більш молодої генерації веб–розробки. На відміну від PHP, Node.js – це не мова, а платформа, яка використовує JavaScript для розробки додатків на стороні сервера. Node.js, запущена в 2009 році, продемонструвала сильні сторони JavaScript в створенні додатків [14].

Переваги Node.js:

1. Fast Server–Side Solution. Node.js використовує цикл подій JavaScript для створення додатків введення / виведення (введення / виведення), які можуть легко обслуговувати кілька паралельних подій. Використовуючи вбудовану асинхронну обробку JavaScript, можна створювати 46 високомасштабуємі серверні рішення, які максимізують використання одного процесора і комп'ютерної пам'яті при одночасному обслуговуванні більш паралельних запитів, ніж звичайні багатопотокові сервери. Ця функціональність робить Node.js відмінним варіантом для асинхронних додатків, керованих даними, і важких робочих процесів з підтримкою введення–виведення, таких як RTA

(додатки реального часу) або SPA (односторінкові додатки), де Node забезпечує відмінну продуктивність під час виконання.

2. Багато популярних клієнтські платформи, такі як Ember, React і Angular, написані на JavaScript, який є основною мовою сучасних браузерів. При використанні серверної частини Node.js у вас є всі переваги однієї мови сценаріїв в стеці розробки додатків. Розробка інтерфейсів і бекенда працюють з однаковими структурами даних JavaScript, функціями, ідіоматичними виразами, все це сприяє прискоренню розробки додатків, виправлення помилок і координації у розробці.

3. Гнучкість в Node.js представляється з декількома жорсткими залежностями, правилами і рекомендаціями, що залишає простір для свободи і творчості при розробці ваших додатків. Node.js не накладає суворі правила, що дозволяють розробникам вибирати кращу архітектуру, шаблони проектування, модулі і функції для нашого проекту. З його допомогою Node.js матиме доступ до тисяч модулів для будь–яких мислимих цілей через репозиторій NPM.

4. Підтримка Node.js серверних подій і WebSockets спрощує реалізацію архітектур pub–sub (publish–subscribe), використовуваних в RTA, і швидко оновлює подання на стороні клієнта. Ті ж функції роблять Node.js придатними для додатків, які обробляють дані з IoT (Internet of Things) та додатків з одиночної сторінкою (SPA), які включають в себе дуже гнучкі, динамічні і важкі функції на стороні клієнта. 47 4. В Node.js простіше впроваджувати веб–служби, в яких серверна частина функціонує як REST API, який забезпечує з'єднання між клієнтом і базою даних і де швидкість операцій CRUD і асинхронна обробка мають велике значення. Недоліки Node.js:

1. Менш ефективний при роботі з додатками з інтенсивним використанням процесора. Node.js – це однопоточні середовище, засноване на подіях, яка не так хороша в обробці інтенсивних операцій ЦП, таких як створення або редагування графіки, звуку і відео, як при управлінні паралельними запитами. При обробці великих файлів або виконанні операцій з графікою додатки Node.js можуть стати

несприйнятливими і млявими. Всякий раз, коли ми працюэмо графічні додатки, багатопотокова серверна середовище є найкращим варіантом, ніж Node.js.

2. Відсутність зрілості. Поряд зі стабільними базовими бібліотеками, такими як HTTP або Crypto, репозиторій Node забезпечує доступ до групи сторонніх модулів, розроблених співтовариством. Ця екосистема модулів ще досить незріла. Важко оцінити якість конкретного модуля до розгортання програми. Більш того, тонкі помилки і неузгодженість версій можуть проникати в додатки, що ускладнює їх підтримку. Легкість публікації ваших власних пакетів в поєднанні з відсутністю надійного механізму затвердження модуля означає, що ви повинні приділяти більше уваги при виборі пакетів, щоб переконатися, що у них були останні дії, такі як виправлення помилок і поновлення [16].

Переваги PHP:

1. База Rich Code. PHP має потужну базу коду, яка включає популярні платформи для створення сайтів (WordPress, Joomla, Drupal) і веб–розробки (Laravel, Symfony), які дозволяють RAPI-privated Application Development. CMS (системи управління контентом), такі як WordPress, спрощують 48 розгортання блогу чи сайту електронної комерції за лічені хвилини і дозволяють не–розробникам налаштовувати їх відповідно до їх потреб. Крім того, існує безліч PHP–додатків, пропонованих платформами хостингу, такими як cPanel, які можна встановити одним клацанням миші. Широта рішень з відкритим вихідним кодом, написана на PHP, і сильне співтовариство, що стоїть за ними, означає, що все, що вам потрібно для розробки додатків, знаходиться під рукою.

2. Кросплатформне рішення. PHP – це дуже кросплатформне рішення для розробки додатків. Його можна запускати практично на будь–якому сервері (Nginx, Apache) і на кожній платформі (Windows, Linux). Це означає, що вам потрібно написати свій код тільки один раз і запустити його в будь–якому місці. У той же час існує широка підтримка хостингу PHP. Ряд хостинг– провайдерів пропонують варіанти спільного розміщення для PHP, але для запуску додатків Node.js необхідний віртуальний сервер з SSH–доступом. Таким чином,

інтеграція і розгортання PHP простіше для невеликих компаній або приватних осіб, які можуть запускати і керувати своїми додатками без будь–яких знань SSH (Secure Shell), консольних команд і технологій сервера Linux, які є кращим варіантом для Node.js.

3. Призначений для Інтернету На відміну від Java або Python і інших мов програмування загального призначення, PHP був розроблений спеціально для роботи в Інтернеті. Саме тому він включає в себе всі необхідні функції для управління HTML, серверами і базами даних (зокрема, MySQL). Завдяки комплексному серверному рішенню, наприклад PHP, також немає необхідності турбуватися про JavaScript в браузері, оскільки всі сторінки можуть бути легко згенеровані і відображені на сервері. Це корисно, якщо ви хочете уникнути перевантаження на стороні клієнта. У той же час важкі серверні рендеринг і регулярні запити до сервера для створення і рендеринга сторінок можуть бути 49 не кращим варіантом для Single Pages Applications (SPA) з багатою клієнтської функціональністю, для якої JavaScript часто є кращим варіантом.

Недоліки PHP:

1. Неефективне поділ проблем (SoC). PHP не дуже підходить для реалізації підходу MVC (Model–View–Controller), який наказує чіткий поділ проблем (SOC) між даними, поведінкою і поданням. MVC є де–факто кращою практикою в веб–розробці, яка забезпечує читаність, зручність обслуговування і масштабованість веб–додатків. Однак в PHP існує тенденція змішувати HTML і синтаксис мови всередині HTML–файлів, що швидко призводить до погано підтримуваного коду, де уявлення і бізнес–логіка не розділені. В результаті важко розширити PHP–додатки з новими функціональними можливостями і керувати програмами з великою базою коду.

2. Застаріла модель клієнт–сервер. PHP слід класичної моделі клієнт–сервер, де кожен запит сторінки ініціює додаток, з'єднання з базою даних, параметри конфігурації і рендеринг HTML. Це робить PHP дещо повільніше в порівнянні з додатком Node.js, яке працює постійно і потребує тільки в ініціалізації один раз. Завдяки цим функціям Node.js більш підходить для

сучасних функцій HTML5, AJAX і WebSockets. Проблема може бути вирішена в PHP через Memcached; проте це не стандартна особливість мови PHP [12].

Таким чином, при виборі між Node.js і PHP ми повинні розглянути, який тип програми ми збираємося створювати, найкращим варіантом для RTA (додатків реального часу), додатки для спільної роботи і SPA (односторінкові додатки) найкраще буде підходити Node.js.

3.4 Обґрунтування вибору програмного інструментарію для розробки клієнтської частини

Для полегшення розробки складної системи і поєднанати різних компонентів будемо використовувати фреймворк. Фреймворк – це готовий до використання комплекс програмних рішень, включаючи дизайн, логіку та базову функціональність системи або підсистеми. Відповідно – програмний фреймворк може містити в собі також допоміжні програми, деякі бібліотеки коду, скрипти. Структура Express.js – це легка і гнучка інфраструктура додатків Node.js, яка надає широкий спектр функцій для створення односторінкових, багатосторінкових і гібридних веб–додатків. Ця структура робить створення веб–сайтів і додатків з використанням Node.js дуже простим. Переваги Express.js, проміжного програмного забезпечення і маршрутизації спрощують процес тестування і покращують продуктивність додатків. За допомогою потужного маршрутизації API розробники можуть виконувати завдання: від створення сервера API REST Express.js до створення маршрутів для простого веб–додатки, а потім перейти до наступного рівня з використанням параметрів маршруту і рядків запиту [13]. API Express.js також використовує вузол вузла Node.js для поширення і установки незліченних сторонніх плагінів. Ви легко можете додати інтеграцію OAuth /соціальні логіни в веб–додаток без особливих проблем, використовуючи це проміжне програмне забезпечення для аутентифікації для підключення. Тим не менше, немає рекомендованого методу організації, який може бути проблемою для початківців і досвідчених розробників, оскільки за

відсутності організації це може привести до нездійсненності проектів. Крім того, є багато ручних трудомістких завдань, які можуть вплинути на використання пам'яті, якщо вони не обробляються професійно.

React — відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників.

React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC, таких як AngularJS. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови веб-застосунків. Як бібліотека інтерфейсу користувача React часто використовується разом з іншими бібліотеками, такими як Redux[17].

У даний час React використовують Khan Academy, Netflix, Yahoo, Airbnb, Sony, Atlassian та інші [14].

AngularJS — JavaScript-фреймворк з відкритим програмним кодом, який розробляє Google. Призначений для розробки односторінкових додатків, що складаються з одної HTML сторінки з CSS і JavaScript. Його мета — розширення браузерних застосунків на основі шаблону Модель-вид-контролер (MVC), а також спрощення їх тестування та розробки.

Фреймворк працює зі сторінкою HTML, що містить додаткові атрибути і пов'язує області вводу або виводу сторінки з моделлю, яка є звичайними змінними JavaScript. Значення цих змінних задаються вручну або отримуються зі статичних або динамічних JSON-даних. За даними служби аналізу JavaScript для Libscore, AngularJS використовується на веб-сайтах Wolfram Alpha, NBC, Walgreens, Intel, Sprint, ABC News та близько 12,000 інших сайтів з 1 мільйона

протестованих у жовтні 2016 року. AngularJS наразі входить до трійки проектів, що набрали найбільшу кількість зірок на GitHub. Angular – це фронтенд частина стеку MEAN, що складається з бази даних MongoDB, програмного каркасу для розробки веб-додатків Express.js, самого Angular.js та платформи Node.js.

AngularJS побудований на переконанні, що декларативне програмування слід використовувати для створення користувацьких інтерфейсів та підключення компонентів програмного забезпечення, тоді як імперативне програмування краще підходить для визначення бізнес-логіки додатка. Фреймворк адаптує та розширює традиційний HTML, щоб представити динамічний вміст через двостороннє зв'язування даних, що дозволяє автоматично синхронізувати моделі та перегляди. Як результат, AngularJS зменшує значення явної DOM-маніпуляції з метою покращення тестування та продуктивності.

Конструктивні цілі AngularJS включають:

- відокремлення DOM-маніпуляцій від логіки додатків, що суттєво впливає на спосіб побудови коду.
- відокремлення клієнтської частину програми від серверної. Це дозволяє розробці працювати паралельно і використовувати повторно обидві сторони.
- проведення розробника через весь шлях створення додатку: від проектування користувацького інтерфейсу, через написання бізнес-логіки, до тестування.

AngularJS реалізує шаблон MVC для відокремлення представлення, даних та логічних компонентів. Використовуючи впровадження залежності, AngularJS традиційно постачає сервісні служби, такі як контролери, залежні від вигляду, для клієнт-серверних веб-додатків. Відповідно, зменшується навантаження на сервер.

AngularJS використовує термін "область видимості" у схожій манері до основ комп'ютерних наук.

Область видимості у комп'ютерних науках описує, коли конкретна прив'язка у програмі є валідною. Специфікація ECMA-262 визначає область

видимості як лексичне середовище, в якому об'єкт функції виконується в клієнтському веб-сценарії.

Як частина архітектури MVC, область видимості формує "Модель". Всі змінні, які визначені в області видимості, можуть також бути доступні у "Контроллері" та "Представленні". В AngularJS "область видимості" є свого роду об'єктом, який сам може бути в або поза областю видимості в будь-якій частині програми, відповідно до звичайних правил області видимості змінних в JavaScript, як і будь-який інший об'єкт. Область видимості здійснює роль клею, який зв'язує "Контроллер" і "Представлення"[19].

AngularJS спроектований з переконанням, що декларативне програмування найкраще пасує для побудови інтерфейсів користувача та опису програмних компонентів, в той час як імперативне програмування пасує для опису бізнес-логіки. Фреймворк адаптує та розширює традиційний HTML, щоб забезпечити двосторонню прив'язку даних для динамічного контенту, що дозволяє автоматично синхронізувати модель та вид. У результаті AngularJS зменшує роль DOM-маніпуляцій з метою підвищення продуктивності і спрощення тестування [15].

Vue.js (читається як "в'ю", з англ. view) — JavaScript-фреймворк що використовує шаблон MVVM для створення інтерфейсів користувача на основі моделей даних, через реактивне зв'язування даних.

Vue використовує синтаксис шаблонів на основі HTML, що дозволяє декларативно зв'язувати рендеринг DOM з основними екземплярами даних в Vue. Всі Vue шаблони валідні HTML, і можуть бути розпарсені браузерами та HTML парсерами. Всередині Vue компілює шаблони в рендерингові функції віртуального DOM. В поєднанні з реактивною системою, Vue здатний розумно обчислити кількість компонентів для ре-рендингу та застосувати мінімальну кількість маніпуляцій з DOM, коли стан додатку зміниться.

В Vue ви можете використовувати синтаксис шаблонів або напряму писати рендерингові функції використовуючи JSX. Для того, щоб це зробити просто замініть шаблон на рендерингову функцію. Рендерингова функція відкриває

можливості для потужних патернів базованих на компонентах - для прикладу, нова транзитна система тепер повністю базована на компонентах, що використовує рендерингові функції всередині.

Одна із найвиразніших особливостей Vue — це ненав'язлива реактивна система. Моделі це просто плоскі JavaScript об'єкти. Це робить керування станами дуже простим та інтуїтивним. Vue надає оптимізований ре-рендеринг з коробки без потреби робити що-небудь додатково. Кожен компонент слідкує за своїми реактивними залежностями під час рендерингу, тому система знає точно коли має відбуватись ре-рендеринг і які компоненти потрібно ре-рендерити.

Vue надає різноманітні шляхи для застосування ефектів переходу, коли елемент додають, оновлюють або видаляють з DOM. Наприклад:

- автоматичне застосування класів для CSS переходів та анімацій
- інтегрування сторонніх бібліотек для CSS анімацій, таких як Animate.css
- використовувати JavaScript для прямих маніпуляцій з DOM під час переходів
- інтегрування сторонніх JavaScript бібліотек анімацій, таких як Velocity.js

Коли елемент, який загорнутий перехідний компонент видаляють чи вставляють, ось що відбувається:

1. Vue автоматично перевірить чи має застосовують до цього елементу CSS анімації та переходи. Якщо так, CSS класи для переходів будуть додані/видалені у відповідний час
2. Якщо перехідний компонент має JavaScript зачіпки, ці зачіпки будуть викликані у відповідний час
3. Якщо ніяких CSS переходів/анімацій не було знайдено та ніяких JavaScript не було надано, DOM операції для додавання і/та видалення відбудеться одразу ж в наступному фреймі.

Vue сам по собі не включає роутингу, та є vue-router пакет, який вирішує це питання. Він підтримує зв'язування вкладенних шляхів з вкладеними

компонентами і пропонує деталізований контроль над переходами. Vue дозволяє створення додатків за допомогою компонетів. Якщо додати vue-router до цього, все що потрібно зробити це зв'язати ваші компоненти з роутами і дозвольте vue-router вирішувати де їх рендерити [20].

Як було зазначено раніше, Angular і React підтримуються і використовуються великими компаніями. Facebook, Instagram і Whatsapp використовують їх на своїх сторінках. Google використовує їх в багатьох своїх проектах: наприклад, новий призначений для користувача інтерфейс Adwords був реалізований за допомогою Angular і Dart. Знову ж, Vue розробляється групою осіб, чия робота підтримується через Patreon і інші засоби спонсорування. Вирішуйте самі, добре це чи погано. Маттіас Гёцке вважає, що невелика команда Vue - це плюс, тому що це веде до більш чистого коду / API, і меншому оверінженерінгу.

Подивимося на статистику: на сторінці команди Angular перераховано 36 осіб, у Vue - 16 осіб, у React сторінки команди немає. На GitHub-е у Angular більше 25 000 зірок і 463 контрибуторів, у React - понад 70 000 зірок і 1000 контрибуторів, і у Vue майже 60 000 зірок і лише 120 контрибуторів. Можете також перевірити сторінку "Github Stars History for Angular, React and Vue". Знову ж, Vue, схоже, дуже добре розвивається. Відповідно до bestof.js, за останні три місяці Angular 2 отримував в середньому 31 зірку в день, React - 74 зірки, Vue - 107 зірок.

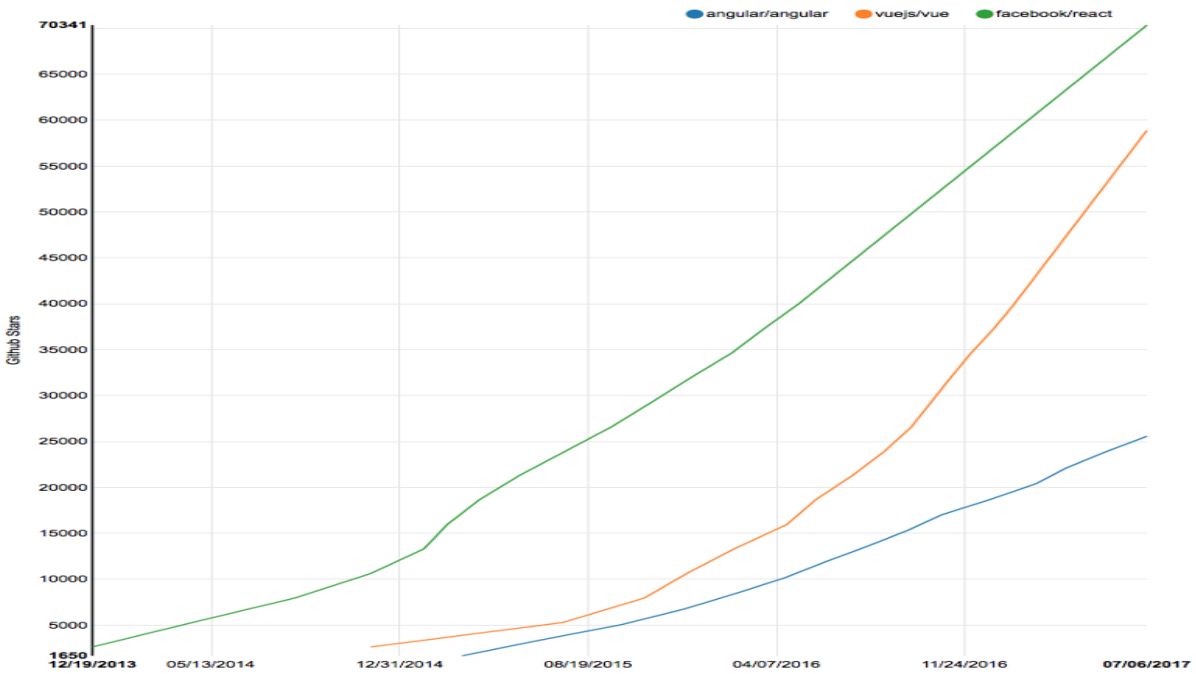

Рисунок 3.1 – Ріст популярності AngularJS, Vue.js та React.js на Github

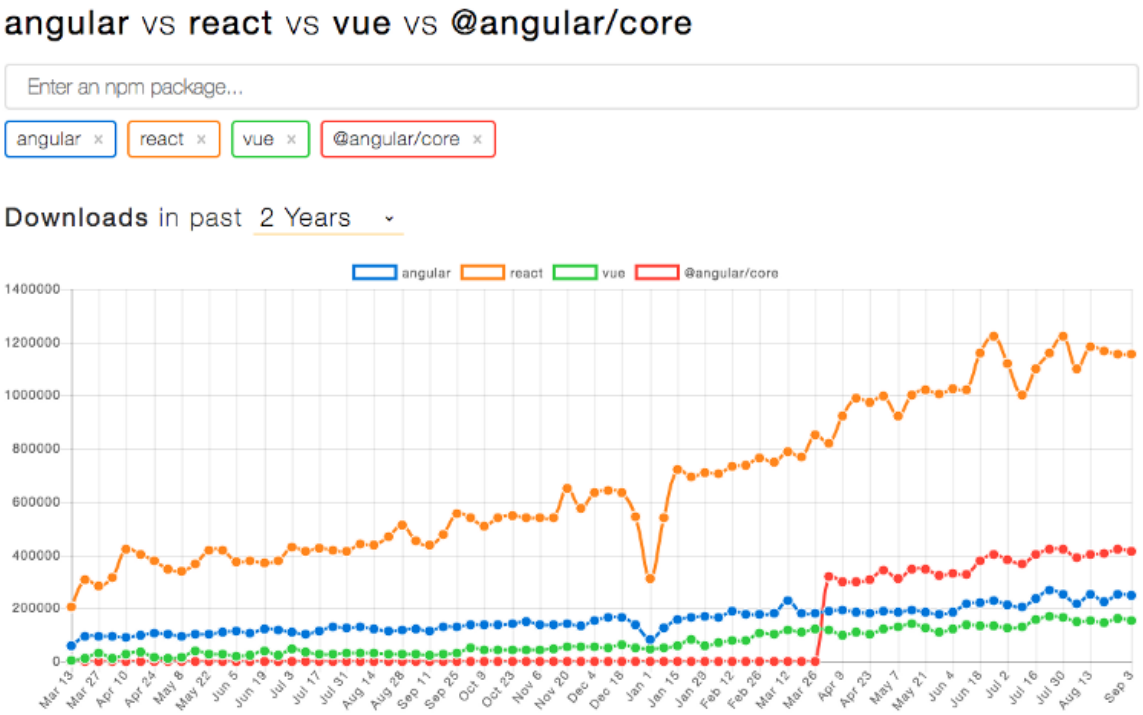

Рисунок 3.2 – Кількість завантажень AngularJS, Vue.js та React.js протягом двох років

Беручи до уваги всі переваги та недоліки, оцінивши сучасні тенденції, було прийнято рішення використати React.js, як фреймфорк для клієнтської частини програмного забезпечення.

JetBrains WebStorm — інтегроване середовище розробки для JavaScript, HTML та CSS від компанії JetBrains, розроблена на основі платформи IntelliJ IDEA. WebStorm є спеціалізованою версією PhpStorm, пропонуючи підмножину з його можливостей. WebStorm постачається з перед-установленим плагінами JavaScript (такими як для Node.js), котрі доступні для PhpStorm безоплатно.

WebStorm підтримує мови JavaScript, CoffeeScript, TypeScript та Dart.

WebStorm забезпечує автодоповнення, аналіз коду на льоту, навігацію по коду, рефакторинг, зневадження та інтеграцію з системами управління версіями. Важливою перевагою інтегрованого середовища розробки WebStorm є робота з проектами (у тому числі, рефакторинг коду JavaScript, що міститься в різних файлах і теках проекту, а також вкладеного в HTML). Підтримується множинна вкладеність (коли в документ на HTML вкладений скрипт на Javascript, в який вкладено інший код HTML, всередині якого вкладений Javascript) — в таких конструкціях підтримується коректний рефакторинг.

Основні можливості:

- Модифікація файлів .css, html, .js з одночасним переглядом результатів (Live Edit, в деяких джерелах ця функціональність називається «редагування файлів на льоту» або «в реальному часі» або «без перезавантаження сторінки»);
- Підтримка HTML5;
- Підтримка JSDoc;
- Підтримка Node.js;
- Можливості Zen Coding і Emmet;
- Знежучування коду на JavaScript;
- Віддалене розгортання за протоколами FTP, SFTP, на монтованих мережевих дисках тощо з можливістю автоматичної синхронізації;
- Інтеграція з системами управління версіями Subversion, Git, GitHub, Perforce, Mercurial, CVS підтримуються з коробки з можливістю побудови списку змін і відкладених змін;
- Інтеграція з системами відстеження жуків.

WebStorm підтримує знежучування застосунків у node.js. Також підтримується повний набір функцій редагування застосунків на javascript — як для виконання на сервері, так і в браузері:

- Автодоповнення;
- Навігація по коду;
- Рефакторінг;
- Перевірка на помилки.

Мови стилів LESS, Sass і SCSS, які розширюють можливості описів стилів у CSS, повністю підтримуються в WebStorm, зокрема, підтримується рефакторинг коду для них, коли треба змінити вираз (наприклад, #a9a9a9) на змінну (наприклад @grey), для того, щоб зробити код читанішим і простіше перевизначати параметри (наприклад, шляхом присвоєння їм значення @grey: #a9a9a9).

Visual Studio Code — засіб для створення, редагування та зневадження сучасних веб-застосунків і програм для хмарних систем. Visual Studio Code розповсюджується безкоштовно і доступний у версіях для платформ Windows, Linux і OS X.

Компанія Microsoft представила Visual Studio Code у квітні 2015 на конференції Build 2015. Це середовище розробки стало першим крос-платформовим продуктом у лінійці Visual Studio.

За основу для Visual Studio Code використовуються напрацювання вільного проекту Atom, що розвивається компанією GitHub. Зокрема, Visual Studio Code є надбудовою над Atom Shell, що використовують браузерний рушій Chromium і Node.js. Примітно, що про використання напрацювань вільного проекту Atom на сайті Visual Studio Code і в прес-релізі і в офіційному блозі не згадується.

Редактор містить вбудований зневаджувач, інструменти для роботи з Git і засоби рефакторингу, навігації по коду, автодоповнення типових конструкцій і контекстної підказки. Продукт підтримує розробку для платформ ASP.NET і Node.js, і позиціонується як легковаге рішення, що дозволяє обійтися без повного

інтегрованого середовища розробки. Серед підтримуваних мов і технологій: JavaScript, C++, C#, TypeScript, jade, PHP, Python, XML, Batch, F#, DockerFile, Coffee Script, Java, HandleBars, R, Objective-C, PowerShell, Luna, Visual Basic, Markdown, JSON, HTML, CSS, LESS і SASS, Haxe.

Sublime Text — швидкий кросплатформенний редактор початкових текстів програм. Підтримує плагіни, розроблені за допомогою мови програмування Python. Sublime Text не є вільним чи відкритим програмним забезпеченням, але деякі його плагіни розповсюджуються з вільною ліцензією, розробляються і підтримуються спільнотою розробників.

Редактор містить різні візуальні теми, з можливістю завантаження додаткових. Користувачі бачать весь свій код в правій частині екрану у вигляді міні-карти, при кліці на яку можна здійснювати навігацію. Є кілька режимів екрану. Один з них включає від 1 до 4 панелей, за допомогою яких можна показувати до чотирьох файлів одночасно. Повноцінний (free modes) режим показує тільки один файл без будь-яких додаткових навколо нього меню.

Виділення стовпців цілком або розстановка кілька покажчиків по тексту, що робить можливим миттєву правку. Покажчики поводяться, ніби кожен з них - єдина в тексті. Команди типу: переміщення на знак, переміщення на рядок, вибірка тексту, переміщення на слово або його частини (CamelCase, розділений дефісом або підкресленням), перехід на початок або кінець рядка тощо, Впливає на всі покажчики незалежно і відразу, дозволяючи правити складноструkурований текст швидко, без використання макрокоманд або регулярних виразів.

Коли користувач набирає код, Sublime Text, в залежності від використовуваної мови, буде пропонувати різні варіанти для завершення запису. Редактор також автоматично завершує створені користувачем змінні.

Sublime Text дозволяє користувачеві збирати програми і запускати їх без необхідності перемикатися на командний рядок. Користувач також може налаштувати свою систему збирання та включити автоматичну збірку програми кожного разу при збереженні коду.

Враховуючи високу гнучкість та інтеллектуальність JetBrains WebStorm, цим продуктом користуються мільйони розробників у цілому світі. Ось чому саме для цієї роботи було обрано JetBrains WebStorm і мову програмування JavaScript.

3.5 Тестування розробленої інформаційної технології для прогнозування комунальних платежів та аналіз результатів її роботи

Розроблена інформаційна технологія для прогнозування комунальних платежів була протестована, що підтвердило коректність її роботи.
Наведемо результати тестування роботи web–додатку. Перед початком роботи з додатком, користувачеві потрібно мати уже створений обліковий запис. Після того як користувач, що бажає використати програмне забезпечення для прогнозування комунальних платежів зайде на сайт, потрібно пройти авторизацію, де необхідно ввести інформацію (електрону пошту, а також пароль від власного кабінету) рисунок 3.3.

Якщо користувач ввів коректні дані, йому відкриється доступ до головного екрану (рисунок 3.4).

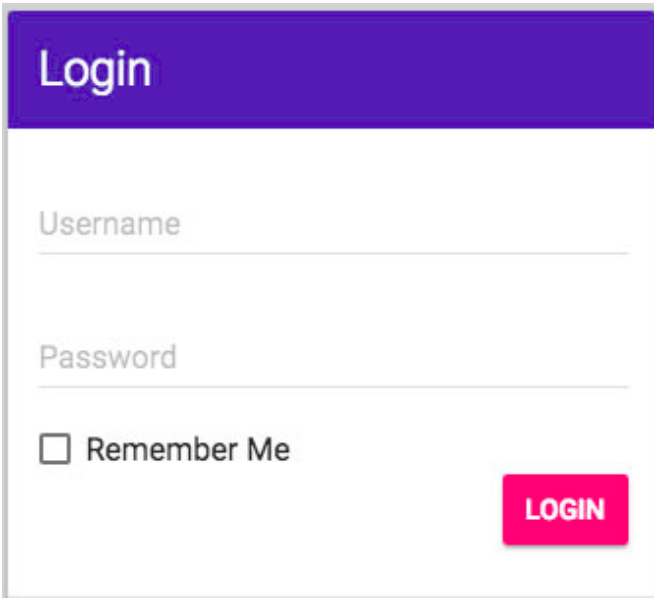

Рисунок 3.3 – Авторизація користувача

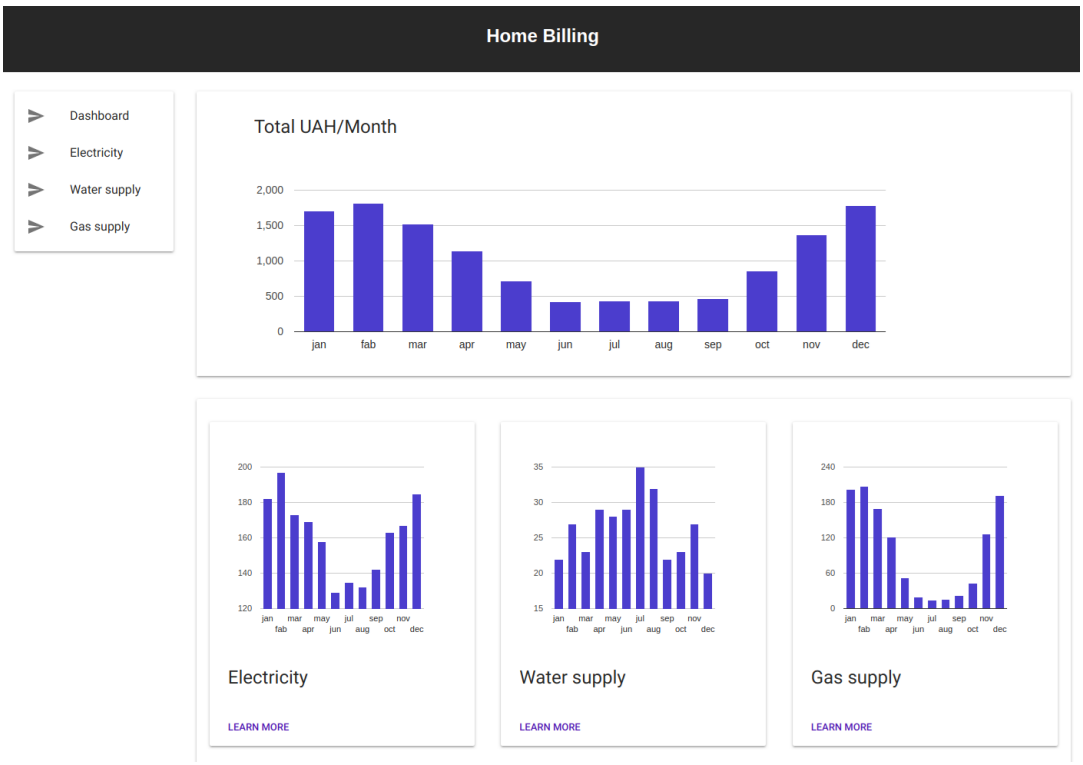

Рисунок 3.4 – Головний екран

Для отримання детальної інформації про конкретну категорію, необхідно обрати пункт меню, наприклад, Electricity (рис. 3.5).

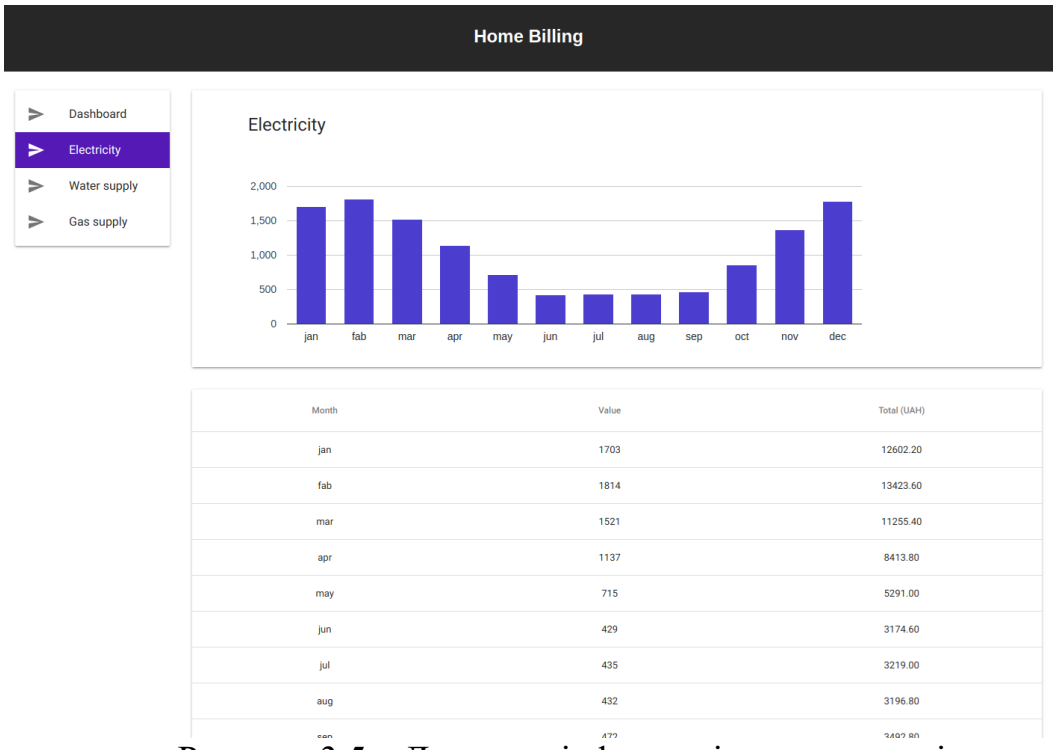

Рисунок 3.5 – Детальна інформація про категорію

Для внесення нових даних до програмного забезпечення, необхідно натиснути на кнопку “ADD NEW VALUE” (рис. 3.6), яка активує вікно додавання нових даних (рис. 3.7).

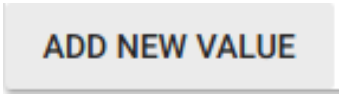

Рисунок 3.6 – Кнопка для визову вікна додавання нових даних

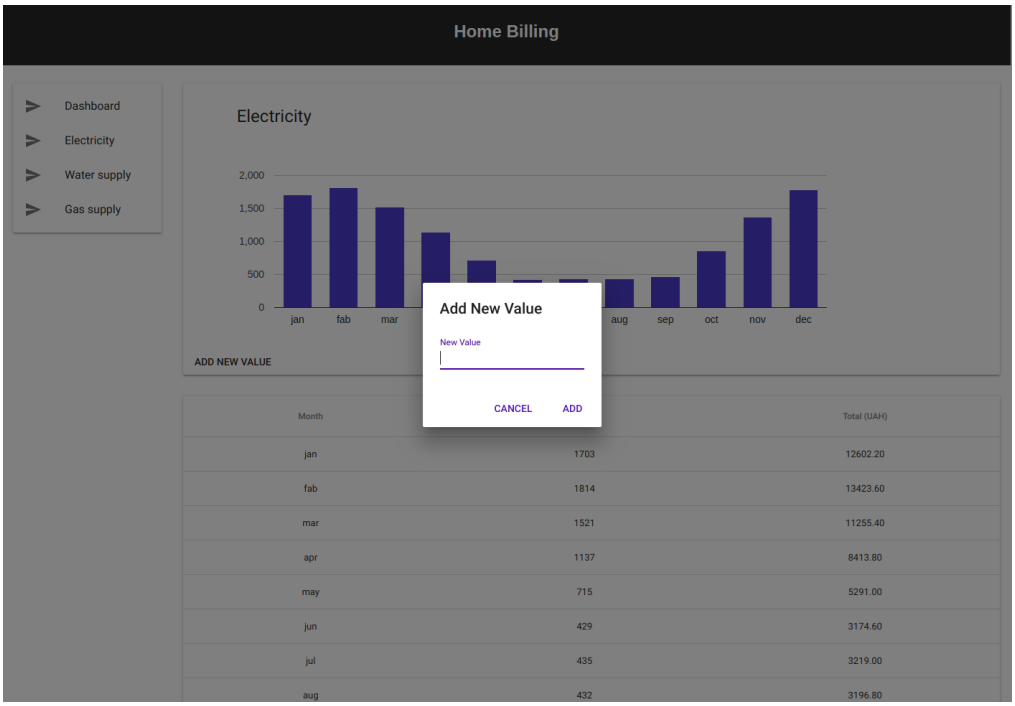

Рисунок 3.7 – Вікна додавання нових даних

У результаті багаторазового тестування розробленої інформаційної технології для прогнозування комунальних платежів та порівняння із програмами-аналогами було визначено, що розроблена інформаційна технологія для прогнозування комунальних платежів дещо збільшила точність прогнозування комунальних платежів, а також не займає додаткового місця на обчислювальному пристрої користувача та не вимагає високої розрахункової потужності, тому що дана інформаційна технологія для прогнозування комунальних платежів була виконана у вигляді клієнт-серверного веб застосунку, що не вимагає значних ресурсів для користування розродбенною інформаційною технологією для прогнозування комунальних платежів, лише

веб-переглядач із доступом до мережі Інтернет.

Результати порівняльного аналізу роботи розробленої інформаційної технології для прогнозування комунальних платежів із програмами-аналогами наведені у таблиці 3.2.

Таблиця 3.2 – Результати порівняльного аналізу

	Навантаження на процесор, %	Використання пам’яті (Мб)	Зручність інтерфейсу користувача	Похибка
“Коммунальные платежи 1.0”	29	393	незручний, складний у освоєнні	Не прогнозує
“1С: Учёт в управляющих компаниях”	34	427	незручний, складний у освоєнні	Не прогнозує
“Универсальная Система Учета”	27	342	зручний та інтуїтивно зрозумілий	5-5.4%
Розроблена програма	17	156	зручний та інтуїтивно зрозумілий	4.9-5.2%

3.6 Висновок

В даному розділі описано етап реалізації інформаційної технології для прогнозування комунальних платежів, а також його тестуваня. Здійснено обґрунтування вибору мови програмування, середовища та фреймворку для розробки додатку. В результаті здійснення аналізу було обрано мову JS. Було

протестовано розроблений додаток. При тестуванні проводилось декілька запусків програмного забезпечення. Програмне забезпечення працює в режимі реального часу та на різних апаратних і програмних платформах у відповідності з задачами проектування. Результати, які отримано під час тестування роботи програми, відповідають результатам, які слідувало очікувати.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Такими експертами будуть Колодний В.В. та Косесницький О.К.

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями за 5-ти бальною шкалою.

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 4.1.

Таблиця 4.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	1. Експерт 1	2. Експерт 2
	Бали, виставлені експертами:	
1	4	4
2	3	3
3	3	4
4	4	3
5	3	3
6	4	4
7	4	3
8	4	4
9	3	3
10	4	3
11	3	4
12	3	4

Сума балів	$СБ_1 = 43$	$СБ_2 = 42$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{2} = 42,5$	

Отже, з отриманих даних таблиці 4.1 видно, що нова розробка має високий рівень комерційного потенціалу.

4.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи.

Для розробки нового програмного продукту необхідні такі витрати.
Основна заробітна плата для розробників визначається за формулою (4.1):

$$З_о = \frac{М}{Т_р} \cdot t, \quad (4.1)$$

де М- місячний посадовий оклад конкретного розробника; $Т_р$ - кількість робочих днів у місяці, $Т_р = 22$ дні; t - число днів роботи розробника, t = 50 днів.

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 4.2.

Таблиця 4.2 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	5800	263,63	7	1845,41
Інженер-програміст	3800	172,72	50	8636
Всього:				10481,41

Розрахуємо додаткову заробітну плату:

$$Здод= 0,1 \cdot 10481,41=1048,14 \text{ (грн.)}$$

Нарахування на заробітну плату операторів НЗП розраховується як 37,5…40% від суми їхньої основної та додаткової заробітної плати:

$$Нзп= (Зо+Зр) \cdot \frac{\beta}{100}, \text{ (4 .2)}$$

$$Нзп= (10481,41+1048,14) \cdot \frac{36,3}{100} = 4185,22(\text{грн.}).$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

$$А=\frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12}, \quad (4.3)$$

де Ц – балансова вартість обладнання, грн; $Н_а$ – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%); Т – Термін використання (Т=3 міс.).

Таблиця 4.3 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань,грн
Персональний комп’ютер	10000	25	3	625
Всього:				625

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

$$K = \sum_1^n Н_i \cdot Ц_i \cdot К_i, \quad (4.4)$$

де n – кількість комплектуючих;

$Н_i$ - кількість комплектуючих і-го виду;

$Ц_i$ – покупна ціна комплектуючих і-го виду, грн;

$К_i$ – коефіцієнт транспортних витрат (приймемо $К_i$ = 1,1).

Таблиця 4.4 - Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	шт.	170	1	170
Пачка паперу	уп.	120	1	120
Ручка	шт.	10	1	10
Всього з урахуванням транспортних витрат				330

Витрати на силову електроенергію розраховуються за формулою:

$$В_е=В\cdot П\cdot Ф\cdot К_п\ ; \quad (4.5)$$

де В – вартість 1кВт-години електроенергії (В=1,7 грн/кВт); П – установлена потужність комп'ютера (П=0,6кВт); Ф – фактична кількість годин роботи комп'ютера (Ф=195 год.); $К_п$ – коефіцієнт використання потужності ($К_п< 1$, $К_п$ = 0,8).

$$Ве= 1,7\cdot 0,6\cdot 195\cdot 0,8 =159,12 \text{ (грн.)}$$

Розрахуємо інші витрати $B_{ін}$. Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

$$В_{ін} = (1..3)\cdot (З_о + З_р). \quad (4.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 * (10481{,}41 + 1048{,}14) = 11529{,}55 \text{ (грн)}.$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в$$

В = 10481,41+1048,14+4185,22+159,12+625+330+11529,55= 28358,44 (грн.)

Розрахуємо загальну вартість наукової роботи$В_{заг}$за формулою:

$$В_{заг} = \frac{В_{ін}}{\alpha} \quad (4.7)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{28358,44}{1} = 28358,44$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta} \quad (4.8)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{28358,44}{0,9} = 31509,37 \text{ (грн.)}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \Delta N)_i \quad (4.9)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році; N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;ΔN – покращення основного кількісного показника діяльності

підприємства від впровадження результатів розробки; $П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки; n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

У результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшаться на 15 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 15 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 300 користувачів, протягом другого року – на 250 користувачів, протягом третього року – 150 користувачів. Реалізація інформаційної технології до впровадження результатів наукової розробки складала 600 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 250 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_1 = 15 \cdot 600 + (250 + 15) \cdot 300 = 88500 \text{грн.}$$

Протягом другого року:

$$\Delta П_2 = 15 \cdot 600 + (250 + 15) \cdot (300 + 200) = 154750 \text{грн.}$$

Протягом третього року:

$$\Delta П_3 = 15 \cdot 600 + (250 + 15) \cdot (300 + 250 + 150) = 194500 \text{грн.}$$

4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $E_{абс}$вкладених інвестицій розраховується за формулою:

$$E_{абс} = (ПП - PV),$$

(4.10)

де$\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 4.1.

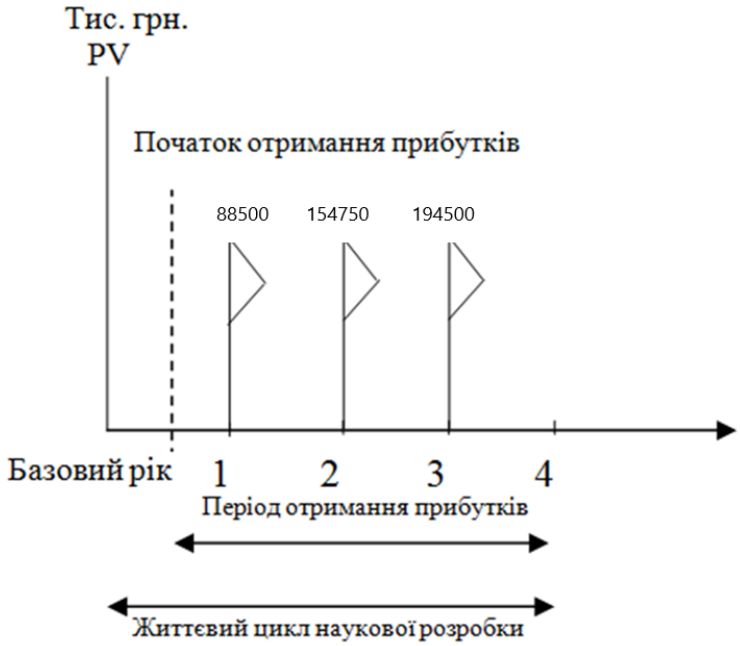

Рисунок 4.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою:

$$ПП = \sum_1^m \frac{\Delta П_i}{(1+\tau)^t} \quad (4.11)$$

де$\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в

країні; для України цей показник знаходиться на рівні 0,1; t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{31509,37}{(1+0,1)^0} + \frac{88500}{(1+0,1)^2} + \frac{154750}{(1+0,1)^3} + \frac{194500}{(1+0,1)^4} = 353761,93(грн.)$$

Тоді розрахуємо $E_{абс}$:

$$E_{абс} = 353761,93 - 31509,37 = 322252,56 \text{ грн.}$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$ за формулою:

$$E_в = \sqrt[T]{1 + \frac{E_{абс}}{PV}} - 1 \qquad (4.12)$$

де $E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн; PV – теперішня вартість інвестицій PV = ЗВ, грн; $T_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$E_в = \sqrt[3]{1 + \frac{322252,56}{31509,37}} - 1 = 1,23, \text{ або } 123\ \%$$

Далі, розраховану величина $E_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2019 році в Україні d = 0,2; f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0,2 + 0,1 = 0,3$$

Оскільки $E_в$ = 123% > $\tau_{мін}$ = 0,3 = 30%, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

4.5 Висновок

Термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{\text{ок}} = \frac{1}{\text{Е}_{\text{в}}},$$

$$T_{\text{ок}} = \frac{1}{1{,}23} = 0{,}81\text{року}.$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи було реалізовано інформаційну технологію для прогнозування комунальних платежів, яка включає в себе базу даних, що містить інформацію про показники різних сфер комунальних послуг за деякий проміжок часу, web–додаток (клієнтська, серверна частина), який дозволяє оперувати даними та отримувати візуалізовані графіки обробленої інформації.

Проведене дослідження показало, що створення програмного забезпечення для прогнозування комунальних платежів являється актуальную в силу необхідності збереження та обробки показників лічильників, спрощення розрахунків та економії часу.

Проаналізовано існуючі реалізації програмного забезпечення для прогнозування комунальних платежів. Існуючі аналоги інформаційної технології для прогнозування комунальних платежів потребують значних витрат коштів на впровадження та супроводження, а також відсутність контролю власних інвестицій. Існуючі рішення створені для потреб, здебільшого, комунальних підприємств та великих домогосподарств, та не є можливими для приватного використання.

Було протестовано розроблена інформаційна теїнологія. При тестуванні проводилось декілька запусків програми так, щоб програмне забезпечення вірно проводило збереження даних та разрахунки.

Програмне забезпечення працює в режимі реального часу, процес збереження зчитування та обробки даних відбувається у зручному для приватного користувача вигляді. У відповідності з задачами проектування програмного забезпечення показало коректність і адекватність роботи програмного забезпечення при здійсненні користувачем всіх доступних опцій для роботи з ним.

Результати виконання магістерської дипломної роботи повністю відповідають технічному завданню.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Перегуда А. В. Аналіз методів екстраполяції для прогнозування комунальних платежів. В матеріалах конференції «The results of scientific mind's development: 2019», Сеул, 2019 [Електронний ресурс]

Режим доступу: https://ojs.ukrlogos.in.ua/index.php/conferences/issue/archive Дата звернення: Грудень 2019

2. Константиновская Л. В. Прогнозирование (элементарный справочник), 2018. – 75с

3. M. Kuhn and K. Johnson, Applied Predictive Modeling // Springer Science+Business Media, 2013. – 463.

4. Методи і моделі прогнозування, їх класифікація – Електрон.текст.дані. – 2017. – Режим доступу http://lubbook.org/book_524_glava_8_3.Metodi_%D1%96_model%D1%96_prognozuva.html

5. 1С: Учёт в управляющих компаниях– Електрон.текст.дані. – 2017. – Режим доступу https://vgkh.ru/jsk/jkh/

6. Коммунальные платежи 1.0 – Електрон.текст.дані. – 2017. – Режим доступу http://www.softportal.com/software-25417-kommunalnie-platezshi.html

7. Джошуа Блох. Java. Эффективное программирование = Effective Java. — М.: Лори, 2002. — 224 с. — ISBN 5-85582-169-2.

8. Кей С. Хорстманн. Java. Библиотека профессионала, том 1. Основы. 10-е издание = Core Java. Volume I - Fundamentals (Tenth Edition). — М.: «Вильямс», 2017. — 864 с. — ISBN 978-5-8459-2084-3.

9. Герберт Шилдт. C++: The Complete Reference. — 4-е изд. — М.: Вильямс, 2011. — С. 800. — ISBN 978-5-8459-0489-8.

10. David Beazley, Guido Van Rossum. Python: Essential Reference. — New Riders Publishing, 1999.

11. Итан Браун. Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript = Web Development with Node and Express / Итан Браун. — Санкт-Петербург: Питер, 2017. — 336 с. — ISBN 978-1-491-94930-6.

12. Brad Green, Shyam Seshadri. AngularJS. — O'Reilly Media, 2013. — 196 p. — ISBN 978-1449344856.

13. David Papich. React.JS Essential. — O'Reilly Media, 2014. — 203 p. — ISBN 978-1849743281.

14. Адам Фримен. ASP.NET MVC 4 с примерами на C# 5.0 для профессионалов, 4-е издание = Pro ASP.NET MVC 4, 4th edition. — М.: «Вильямс», 2013. — 688 с. — ISBN 978-5-8459-1867-3.

15. Джон Скит. C# для профессионалов: тонкости программирования, 3-е издание, новый перевод = C# in Depth, 3rd ed.. — М.: «Вильямс», 2014. — 608 с. — ISBN 978-5-8459-1909-0.

16. Брюс Эккель «Философия Java» — Спб.: «Питер», 2015. — 1168 с. — ISBN 978-5-496-01127-3.

17. А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд. Язык программирования C#. Классика Computers Science. 4-е издание = C# Programming Language (Covering C# 4.0), 4th Ed. — СПб.: «Питер», 2012. — 784 с. — ISBN 978-5-459-00283-6.

18. Гаджинский А.М. Логистика: Учебник для высших и средних специальных учебных заведений. М.: ИВЦ"Маркетинг", 2000. 375 с.

19. Порівняння мов програмування [Електронний ресурс] – режим доступа: https://ru.wikipedia.org/wiki/Сравнение_языков_программирования

20. Васильев А. Н. − Java. «Объектно-ориентированое програмирование» — Питер, 2012. – 395 с. – ISBN 978-5-459-01050-3.

21. Eclipse основні відомості [Електронний ресурс] – режим доступа: https://ru.wikipedia.org/wiki/Eclipse_(среда_разработки)