АНОТАЦІЯ

Дипломна робота на тему «Розробка інформаційної системи аграрного підприємства з використанням мови C# та технології .Net» Миколаєвича Олега Романовича – Тернопільський національний технічний університет імені Івана Пулюя, Факультет комп'ютерно-інформаційних систем і програмної інженерії, Кафедра програмної інженерії, група СПм–62 Тернопіль, 2019.

С. – 81, рис. – 24, табл. – 5, додат. – 5.

Метою дипломної роботи є дослідження, проектування та розробка Web-сервісу керування сільськогосподарськими підприємствами та обміну інформації між ними. Методи та програмні засоби, використані при виконанні розробки системи: мова програмування C# та її бібліотеки, написання серверної частини з довомогою Asp.Net, середовище розробки Windows Presentation Foundation Windows form разом з графічною бібліотекою Silverlight.

Опрацьована вхідна та результуюча інформація, досліджено технології створення динамічних web-сервісів, описана структура наявної інформаційної бази, блок-схеми та кодові взірці основних функцій, проведена розробка програмного забезпеченя, вивчено питання безпеки охорони праці.

Ключові слова: WEB-СЕРВІС, ІНФОРМАЦІЙНА СИСТЕМА, ХМАРНИЙ СЕРВІС, ЯКІСТЬ ОБСЛУГОВУВАННЯ.

ABSTRACT

Diploma thesis on "Distribution of information systems of agrarian enterprise using C # language and .Net technology" Mykolaievich Oleh - Ivan Puliuy Ternopil National Technical University, Faculty of Computer-International Systems and Software Engineering, Department of Software Engineering, SPM Group 62 // Ternopil, 2019.

C. - 81, fig. - 24, tab. - 5, add. - 5.

The methodology of the diploma work is the research, development and development of web-based management services for agricultural enterprises and the exchange of information between them. Methods and software used in advanced mailing systems: C # and its libraries are programmable, written by the Asp.Net server-based server, Windows Presentation Foundation mailing environment Windows uses the Silverlight graphics library.

Get input and output results by exploring the technology of creating dynamic web services, describing the structure of the database used, flowcharts and codes that exist in all official functions, run a developed software program, using employee safety issues.

Keywords: WEB-SERVICE, INFORMATION SYSTEM,CLOUD SERVICE, QUALITY OF SERVICE.

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

IT - Information Technology .

AaaS - Agriculture as a Service, сільське господарство як послуга.

SaaS - software as a service, програмне забезпечення як послуга.

IaaS - Infrastructure-as-a-Service, інфраструктура як послуга.

ГІС - Глобальної інформаційної системи.

FMIS - Farm Management Information System.

WASS - Web-based Agricultural Support System.

DSS - Decision Support System.

UML - Unified Modeling Language, уніфікована мова моделювання.

SOA – Service Oriented Architecture.

CLR - Common Language Runtime.

WPF - Windows Presentation Foundation

QoS - Quality of service, якість обслуговування

Зміст

ВСТУП

Інформація та комунікація відіграють важливу роль у сільському господарстві ще з античності. Фермери, можливо, шукали інформацію один у одного, і вони поширювали інформацію іншим шляхом спілкування, таким чином розвивалися методики. Більшість фермерів досі дотримуються традиційних методів ведення господарства. У машинну епоху з’явилося багато машин, які допомогли фермеру в його вирощуванні. Машини спростили роботу фермера, його обов'язок - звести нанівець працю фермерів, але не має нічого спільного у ефективному збільшенні виробництва та якості продукції. У цьому полягає актуальність використання інформаційних технологій сільського господарства. У цю епоху технологічного прогресу ця ідея має багато спільного у сільському господарстві. Використання ІТ для вдосконалення прийняття рішень у сільському господарстві розглядається як

ідея з величезним потенціалом. IT пов'язаний із світовим світом, і його динаміка змінює наш спосіб життя та соціальну свідомість. На всіх етапах сільського господарства управління інформаційними технологіями має важливе значення для успіху.

Хмарні обчислення стали важливою парадигмою ефективного управління та надання послуг через Інтернет. Конвергенція хмарних обчислень з такими технологіями, як бездротові сенсорні мережі і мобільні обчислення, пропонує нові додатки хмарних сервісів, але для цього потрібно керувати параметрами якості обслуговування для ефективного моніторингу та вимірювання, що надаються. У цій роботі представлена хмарна автономна інформаційна система на основі якості обслуговування для надання пов'язаної з сільським господарством інформації в якості послуги з використанням новітніх хмарних технологій, які керують різними типами пов'язаних з сільським господарством даних на основі різних областей. Пропонована система збирає інформацію від різних користувачів через попередньо сконфігуровані пристрої і керує нею і автоматично надає необхідну інформацію користувачам. Крім того, алгоритм оптимізації зозулі був використаний для ефективного розподілу ресурсів на рівні інфраструктури для ефективного використання ресурсів.

В ході роботи було оцінено продуктивність пропонованого підходу в хмарному середовищі, і експериментальні результати показують, що пропонована система працює краще з точки зору використання ресурсів, часу виконання, вартості та обчислювальної потужності поряд з іншими параметрами якості обслуговування.

Хмарні обчислення стали важливою парадигмою ефективного управління та надання нових додатків в області охорони здоров'я, сільського господарства, освіти, фінансів і т. д. через інтернет. Однак надання спеціалізованих хмарних сервісів, що забезпечують динамічні вимоги (якість обслуговування) і задоволеність користувачів, є великий дослідницької

завданням в хмарних обчисленнях. Оскільки динамізм, неоднорідність і складність додатків швидко ростуть, це робить хмарні системи некерованими в наданні послуг. Щоб подолати ці проблеми, хмарні системи вимагають самостійного управління сервісами. Автономні хмарні обчислювальні системи забезпечують середу, в якій додатки можуть ефективно управлятися шляхом виконання вимог якості обслуговування додатків без участі людини [1] [2].

У цій роботі я запропонував хмарну автономну інформаційну систему, яка забезпечує сільське господарство як послугу через хмарну інфраструктуру і сервіси.

Поява інформаційних і комунікаційних технологій грає важливу роль в сільськогосподарському секторі, надаючи послуги через комп'ютерні сільськогосподарські системи [2]. Але ці сільськогосподарські системи не здатні задовольнити потреби сучасного покоління через відсутність важливих вимог, таких як швидкість обробки, менший обсяг пам'яті для зберігання даних, надійність, доступність, масштабованість і т. д., і навіть ресурси, використовувані в комп'ютерних сільськогосподарських системах, не використовуються ефективно. Щоб вирішити проблему існуючих систем сільського господарства, необхідно розробити хмарну службу, яка могла б легко управляти різними типами пов'язаних з сільським господарством даних на основі різних областей (культура, погода, грунт, шкідливі організми, добрива, продуктивність, іригація, велику рогату худобу і ін. обладнання) за допомогою цих кроків: збирати дані від різних користувачів через попередньо сконфігуровані пристрої, класифікувати зібрані дані по різних класах за допомогою аналізу, зберігати секретну інформацію в хмарному сховище для майбутнього використання і автоматичну діагностику сільського господарства стан справ . Крім того, хмарна автономна інформаційна система також здатна ідентифікувати вимоги якості обслуговування для користувача запиту, і ресурси ефективно розподіляються

для виконання призначеного для користувача запиту на основі цих вимог. Хмарні сервіси можуть значно підвищити надійність, доступність і задоволеність клієнтів.

1 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

1.1. Аналіз вимог до програмної системи

1.1.1 Аналіз придметної області

Сільське господарство - це обробіток ґрунту, плекання сільськогосподарських культур та розведення худоби. Він включає переробку рослинних і тваринних продуктів для людей та їх реалізацію на ринки.

Термін сільське господарство, інакше відомий як землеробство, відноситься до процесу виробництва продуктів харчування, кормів, клітковини та інших бажаних продуктів, які можна отримати шляхом

вирощування певних рослин та вирощування одомашнених тварин (худоби) та є орієнтиром сільського господарства.

Сільськогосподарський сектор є одним з найважливіших галузей економіки через вплив та вплив на інші галузі та повсякденне життя людей. Існує кілька причин, чому цей сектор є основою економіки, наприклад, сільського господарства:

- забезпечує їжею та кормом для годування людей та худоби;
- є джерелом сировини для виробництва;
- є джерелом зайнятості та засобів до існування для значної частини населення світу;
- сприяє національному доходу і є джерелом державних доходів; і
- є основою для економічного розвитку - на місцевому та національному рівні - і сприяє загальному економічному розвитку країни.

Через широкий обсяг впливу на сільськогосподарський сектор він відіграє вирішальну роль в економіці на місцевому, регіональному, національному і навіть глобальному рівнях.

Однак сільський господарство стикається з кількома проблемами, такими як зміни клімату, продовольча безпека та зростання населення. Для вирішення цих проблем та боротьби з ними Європейська Комісія виділяє важливі дослідницькі ресурси на соціальну проблему продовольчої безпеки, сталого сільського господарства, морських та морських досліджень та біоекономіки через програму « Горизонт 2020» . Іншим методом підтримки сільського господарства через ці виклики є використання, успішне впровадження та поєднання цифрових технологій, пов’язаних із промисловістю 4.0.

Сільське господарство та відкриті дані

Світ рухається до четвертої промислової революції - інакше відомої як « Промисловість 4.0», де комп’ютери будуть покращені розумними та автономними системами, що живляться за допомогою даних та машинного

навчання. Для того, щоб пояснити, концепція «Промисловість 4.0» посилається на те, коли на фабриках і машинах розширено бездротове підключення та датчики, підключені до системи, яка може візуалізувати всю виробничу лінію та приймати рішення самостійно.

Сільськогосподарський сектор вже все більше поєднує технології, такі як геолокація, моніторинг стану ґрунтів та навколишнього середовища, Штучний інтелект, хмарні обчислення та Інтернет речей, щоб точно виміряти зміни змінних у посіві та покращити кількість та якість сільськогосподарської продукції. Для успішного впровадження та поєднання цих технологій сільському господарству потрібні дані .

Дані - особливо відкриті дані - відіграватимуть вирішальну роль у наданні допомоги сектору сільського господарства орієнтуватися та процвітати через Промисловість 4.0 та мають потенціал для перетворення сільського господарства та сприяння продовольчій безпеці у всьому світі. Ці набори даних включають дані про погоду, дані про генетику насіння, дані про екологічні умови та дані про ґрунти. Щоб детальніше розповісти про те, як набори даних можуть впливати на сектор сільського господарства, візьмемо приклад даних погоди. Якщо дані про погоду будуть записані відкритими, фермери зможуть спланувати свій сезон посадки та збільшити врожайність, зменшивши ризик заморозків або посухи, що завдають шкоди їхнім урожаям. Крім того, якщо дані про погоду будуть відкриті, фермери можуть оптимізувати свою систему зрошення води, щоб підготуватися до дощових чи сухих днів, а не над водою чи нехтувати поливом своїх культур.

Поява інформаційних і комунікаційних технологій грає важливу роль в сільськогосподарському секторі, надаючи послуги через комп'ютерні сільськогосподарські системи [2]. Але ці сільськогосподарські системи не здатні задовольнити потреби сучасного покоління через відсутність важливих вимог, таких як швидкість обробки, менший обсяг пам'яті для зберігання даних, надійність, доступність, масштабованість і т. д., і навіть

ресурси, використовувані в комп'ютерних сільськогосподарських системах, не використовуються ефективно. Щоб вирішити проблему існуючих систем сільського господарства, необхідно розробити хмарну службу, яка могла б легко управляти різними типами пов'язаних з сільським господарством даних на основі різних областей (культура, погода, грунт, шкідливі організми, добрива, продуктивність, іригація, велику рогату худобу і ін. обладнання) за допомогою цих кроків: збирати дані від різних користувачів через попередньо сконфігурованих пристроїв, класифікувати зібрані дані по різних класах за допомогою аналізу, зберігати секретну інформацію в хмарному сховище для майбутнього використання і автоматичну діагностику сільського господарства стан справ . Крім того, хмарна автономна інформаційна система також здатна ідентифікувати вимоги якості обслуговування для користувача запиту, і ресурси ефективно розподіляються для виконання призначеного для користувача запиту на основі цих вимог. Хмарні сервіси можуть значно підвищити надійність, доступність і задоволеність клієнтів.

1.1.2 Постановка задачі

Необхідно представити Agri-Info як сільськогосподарську послугу для автоматичного управління різними типами даних, пов'язаних із сільським господарством і відносяться до різних областей. Сільськогосподарські дані були класифіковані з використанням механізму класифікації K-NN (k-найближчого сусіда), а для вилучення атрибутів використовується аналіз основних компонентів. Було використовували нечітку логіку для інтерпретації даних по сільському господарству для автоматичної діагностики стану сільського господарства. А також продемонстрували здатність запропонованого методу автономного управління ресурсами,

запровадивши його в хмарну середу на основі моделювання з використанням інструментарію CloudSim разом з його емпіричної оцінкою. Нарешті, ми перевірили Agri-Info, використовуючи хмарну середу, і виміряли варіації. Платформа розробки додатків Azure використовується в якості масштабується хмарного проміжного програмного забезпечення для взаємодії SaaS і IaaS для розгортання веб-служби електронного сільського господарства Agri-Info. Продуктивність Agri-Info також була протестована на хмарному стенді з використанням синтетичних робочих навантажень для різних параметрів якості обслуговування. Потім ми порівняли експериментальні результати пропонованої методики з технікою планування неавтономних ресурсів (без параметрів якості обслуговування). Основним внеском цієї роботи є: використаний алгоритм оптимізації Cuckoo для ефективного розподілу ресурсів на рівні інфраструктури для ефективного використання ресурсів після визначення вимог якості обслуговування, пропонована система збирає інформацію від різних користувачів через попередньо сконфігуровані пристрої і управляє даними в хмарної бази даних і автоматично надає необхідну інформацію користувачам, використовувала платформу розробки додатків Azure для розгортання веб-служби агр - інформація Agri-Info і для підвищення задоволеності клієнтів за рахунок самостійного управління ресурсами. Таким чином, пропонований підхід підвищує задоволеність користувачів за рахунок задоволення їх очікувань і підвищує надійність і доступність хмарних сільськогосподарських послуг.

1.1.3 Аналіз подібних проектних рішень

Сільське господарство в наш час досягло високого рівня. Фермери розглядають господарство як бізнес, а не діяльність з виробництва продуктів харчування для внутрішнього споживання. Як і будь-який інший бізнес, сільське господарство також стикається з такими проблемами, як обмежені ресурси. Але завдяки передовій технології фермери також можуть збільшити

врожайність. Постачальники програмного забезпечення придумали програмне забезпечення для управління фермами, яке допомагає фермерам легко контролювати врожайність ферм. Краса цього програмного забезпечення полягає в тому, що його можна використовувати з будь-яким мобільним гаджетом, будь то андроїд телефон, планшети, iPad або Windows.

Що таке програмне забезпечення для управління фермою? Програмне забезпечення управління фермерським господарством використовується для оптимізації та управління господарськими операціями та виробничою діяльністю. Програмне забезпечення допомагає в автоматизації фермерських заходів, таких як управління записами, зберігання даних, моніторинг та аналіз сільськогосподарської діяльності, а також впорядкування графіків виробництва та роботи. Програмне забезпечення налаштовано на відповідність конкретним фермерським вимогам, оскільки кожна фірма здійснює певні види діяльності, які здійснюються. Варто зазначити, що сільське господарство - це вузькоспеціалізована діяльність, тим більше причин, чому вам слід бути обережними з типом програмного забезпечення, для якого ви влаштовуєтесь. Прогнозуйте та вимірюйте прибуток : Сільське господарство вже не вважається діяльністю минулого часу, а серйозним бізнесом, як будь-який інший бізнес. Серйозний підприємець повинен відслідковувати всі витрати та будь-яку діяльність, що проводиться у господарстві. Програмне забезпечення для управління фермою має функцію, яка допомагає відслідковувати всі фінансові дії для підвищення ефективності та підвищення продуктивності. Розробка планів посіву : Програмне забезпечення забезпечує цілісний вигляд господарства, що дозволяє фермеру ефективно планувати діяльність на фермах. Щоб досягти всебічного уявлення, вам потрібна система, яка збирає, аналізує та генерує звіти, на які можна покластися, щоб прийняти обгрунтоване рішення. Ця функція дає зрозуміти, коли і як проводити сівозміну, найбільш підходящий спосіб боротьби з шкідниками та найкращі добрива, що застосовуються.

Відстеження та вимірювання польових заходів : у господарстві є кілька працівників, які допомагають у щоденній діяльності. Програмне забезпечення допомагає працівникам фермерських господарств відслідковувати відповідну інформацію про ферму, щоб керівник господарства здійснював нагляд. Система також дозволяє відстежувати врожайність, визначати врожаї, які добре справляються в різні сезони для кращого планування. Наявність такої інформації допомагає планувати сезони та придумувати зручний календар, який зберігає важливі дати протягом усього сезону. Управління портфелями ризиків : допомагає вивчити результати роботи на місцях у минулому, таким чином приймаючи обґрунтовані рішення, в який портфель інвестувати, а який уникнути. Система також допомагає зрозуміти вхідні витрати, а також дохідність проекту та ціни продажу. Зовнішні ризики, такі як нестабільні погодні умови, хвороби, шкідники та непередбачувані вимоги ринку, що не підлягають контролю фермерів, є частиною проблем, з якими стикаються сучасні фермери. Ефективне програмне забезпечення для управління фермерським господарством сповіщає фермерів про майбутні ризики, таким чином приймаючи обгрунтоване рішення про те, як обійти проблеми, що насуваються [4].

В даний час існує безліч різних програмних продуктів, які допомагають в процесі роботи підприємства і врахування його діяльності. Вони прискорюють процес обробки, компіляції та зберігання інформації. Кожна програма розробляється і вдосконалюється, і концепція хмарних технологій все ширше використовується.

Для порівняння запропонована архітектура системи управління фермою з використанням характеристик Інтернету, яка фокусується на процедурі ведення сільського господарства і механізмах обміну інформацією між зацікавленими сторонами. Далі, ця архітектура описує метод для кращого управління тільки деякими з завдань фермерів без використання

автономної концепції. Також була представив ALSE (Оцінювач придатності сільськогосподарських земель) для вивчення різних типів земель, щоб знайти підходящі землі для різних типів сільськогосподарських культур шляхом аналізу геоекологічних факторів. ALSE використовувала можливості ГІС (Глобальної інформаційної системи) для оцінки землі з використанням місцевих умов навколишнього середовища через цифрову карту і на основі цієї інформації можуть бути прийняті рішення. Також запропонована FMIS (інформаційна система управління фермою), яка використовується для визначення вимог до точного землеробства для інформаційних систем за допомогою веб-підходу. Було визначино, що управління даними ГІС є ключовою вимогою до точного землеробства. Було вивчено FMIS (Farm Management Information System - Інформаційна система управління фермерськими господарствами) для аналізу динамічних потреб фермерів у поліпшенні процесів прийняття рішень і їх відповідних функціональних можливостей. Далі повідомлено, що ідентифікація процесу, використовуваного для початкового аналізу потреб користувачів, є обов'язковою для фактичного проектування FMIS.

Представлений аналіз мережевих інформаційних систем для сільського господарства і виявив різні проблеми і проблеми, які ще не вирішені в цих системах. Через відсутність автоматизації в існуючій сільськогосподарської системі, система забирає більше часу і важко справляється з динамічними потребами користувача, що призводить до незадоволеності клієнтів. Визначено різні функціональні вимоги FMIS, і інформаційна модель представлена на основі цих вимог для уточнення процесів прийняття рішень. Вони визначили, що складність FMIS збільшується зі збільшенням функціональних вимог, і виявили, що для зменшення складності потрібно автономна система. Була запропонована WASS (Web-based Agricultural Support System - Веб-система сільського господарства) і визначні функціональні можливості (інформація, спільна робота і підтримка

прийняття рішень) і характеристики WASS. Виходячи з характеристик, було розділено WASS на три підсистеми: виробничу, науково-освітню та управлінську

Запропонована до розгляду заснована на державну виконавчу службу структура DSS (Decision Support System - Система підтримки прийняття рішень), в якій просторовий DDS була розроблена для управління водозбірних басейнами і управління врожайністю сільськогосподарських культур на регіональному та фермерському рівні. ГІС використовується для збору і аналізу графічних зображень для прийняття нових правил і рішень для ефективного управління даними. Представили засновану на мобільних обчисленнях середу для агрономів під назвою AgroMobile для вирощування, маркетингу та аналізу зображень культур. Крім того, AgroMobile використовується для виявлення захворювання за допомогою обробки зображень, а також обговорює, як динамічні потреби користувача впливають на продуктивність системи. Запропонував хмарну систему прогнозування захворювань і моніторингу худоби (DFLMS), в якій сенсорні мережі використовувалися для збору інформації та віртуального управління. DFLMS надає ефективний інтерфейс для користувача, але через використовуваного механізму тимчасового зберігання він не може зберігати та видавати дані в базах даних для майбутнього використання. Представив хмарну систему прогнозування погоди для збору і аналізу даних, пов'язаних з погодою, для визначення потреб сільського господарства в різні сезони. Ця система зменшує реплікацію даних і забезпечує балансування навантаження для управління ресурсами. Запропоновану автономну інформаційну систему, засновану на якості обслуговування (Agri-Info), порівняно з існуючими системами сільського господарства, як описано в таблиці 1.1.

Таблиця 1.1. Порівняння існуючих систем сільського господарства із запропонованою системою (Agri-Info)

Сільськогосподарська система	Механізм	Якість обслуговування-обізнаний (параметр)	Домени	Класифікація даних	Управління ресурсами
ALSE	Неавтономний	Так (придатність)	Грунт	Так	Ні
FMIS	Неавтономний	Ні	Шкідники та урожай	Ні	Ні
WASS	Неавтономний	Ні	Продуктивність	WASS	Неавтономний
AgroMobile	Неавтономний	Так (точність даних)	Урожай	AgroMobile	Неавтономний
DFLMS	Неавтономний	Ні	Урожай	DFLMS	Неавтономний

Продовження таблиці 1

Пропонована система (Agri-Info)	Автономний	Так (вартість, час, використання ресурсів, обчислювальна потужність, доступність, пропускна здатність мережі,	Урожай, погода, ґрунт, шкідники, добрива, продуктивність, зрошення, велику рогату худобу та обладнання	Так	Так

		задоволеність клієнтів та затримка			

Всі перераховані вище дослідні роботи були зосереджені на різних областях сільського господарства з різними параметрами якості обслуговування. Жодна з існуючих сільськогосподарських систем не враховує самоврядування ресурсами. Через відсутність автоматизації управління ресурсами послуги стають неефективними, що призводить до незадоволеності клієнтів. Пропонована система являє собою нову автономну інформаційну систему на основі хмарних обчислень, засновану на якості обслуговування, і розглядає різні галузі сільського господарства і автоматично розподіляє ресурси і управляє ними, що не враховується в інших існуючих сільськогосподарських системах.

1.1.4 Пошук актантів та варіантів використання

Актор - багато логічних зв'язаних ролей в UML, виконуючих при сумісній взаємодії з прецедентами чи сутностями (система, підсистема або клас)[5]. Акторами можуть бути люди, інші системи, тригери часу або тригери подій. Актор визначає роль, яку відіграє користувач або будь-яка інша система, яка взаємодіє з предметом. Він може представляти ролі, які грають користувачі людини, зовнішнє обладнання або інші предмети. Актори завжди знаходяться поза системою і безпосередньо взаємодіють з нею, ініціюючи випадок використання, надаючи вхід до неї та / або отримуючи з неї вихід. Хоча один фізичний екземпляр може грати роль декількох різних акторів, актор не обов'язково представляє конкретні фізичні сутності, тобто таймер, який запускає надсилання нагадування електронної пошти.

Основним учасником справи використання є зацікавлена сторона, яка закликає систему надати одну зі своїх послуг. Він має ціль щодо системи - ту,

яку може задовольнити її робота. Основним актором є часто, але не завжди, актор, який викликає випадок використання.

Актори, що підтримують: користувач, який надає послугу (наприклад, інформацію) для системи.

Підтримуючий актор (також відомий як вторинний актор) у випадку використання у зовнішньому акторі, який надає послугу розробленій системі. Це може бути високошвидкісний принтер, веб-служба або люди, які повинні провести деякі дослідження та повернутися до нас[5].

UML (Unified Modeling Language) - мова моделювання, яка використовується розробниками програмного забезпечення . UML можна використовувати для розробки діаграм та надання користувачам (програмістам) готових до використання виразних прикладів моделювання. Деякі засоби UML генерують код мови програми з UML. UML можна використовувати для моделювання системи, незалежної від мови платформи. UML - це графічна мова для візуалізації, уточнення, побудови та документування інформації про програмно-інтенсивні системи. UML дає стандартний спосіб написання системної моделі , що охоплює концептуальні ідеї. З розумінням моделювання, використання та застосування UML можуть зробити процес розробки програмного забезпечення більш ефективним [6].

Ця підсистема надає користувальницький інтерфейс, в якому різні типи користувачів взаємодіють з Agri-Info для надання та отримання корисної інформації про сільське господарство на основі різних доменів. Ми розглянули дев'ять типів інформації в різних галузях сільського господарства: урожай, погода, грунт, шкідники, добрива, продуктивність, зрошення, худобу і обладнання. Користувачі в основному діляться на три категорії:

1. експерт з сільського господарства;
2. фахівець із сільського господарства;
3. фермер.

Експерт по сільському господарству ділиться професійними знаннями, відповідаючи на запити користувачів і оновлюючи базу даних AaaS, грунтуючись на останніх дослідженнях, зроблених в галузі сільського господарства щодо їх області. Сільськогосподарські чиновники - урядовці, які надають останню інформацію про нові сільськогосподарських політиках, схемах і правилах, прийнятих урядом. Фермер є важливою частиною Agri-Info, яка може отримати максимальну вигоду, задаючи свої запити і отримуючи автоматичну відповідь після аналізу.

Діаграма варіантів використання, показана в додатку В, описує важливі функції, які користувач може виконувати за допомогою Agri-Info. В таблиці 1.2 представлені актори та їхні можливості

Таблиця 1.2 Актори системи Agri-Info

Ім’я актора	Опис можливостей актора
Фермер	Можливість реєстрації та аутентифікації в системі, а також діагностику стану сільського господарства
Гість	Реєстрації та аутентифікації в системі. Перегляд незначної кількості інформації
Фахівець із сільського господарстава	Діагностика стану сільського господарства

Продовження таблиці 1.2

Експерт із сільського господарства	Оновлення стану і політики сільського господарства

Діаграма варіантів використання описує взаємодію користувачів (експерт з сільського господарства, сільськогосподарський фахівець і фермер) з Agri-Info. Діаграма послідовності взаємодії об'єктів на основі тимчасової послідовності. Він показує, як об'єкти взаємодіють з іншими в конкретному сценарії варіанти використання. В додатку Г показана схема послідовності дій користувача в Agri-Info. По-перше, була продемонстрована успішна реєстрація користувача. Після виконання завдання аутентифікації і авторизації користувача буде відображатися домашня сторінка користувача. Користувач може написати свій запит щодо інформації про сільське господарство, яка вимагається від будь-якого домену, і потім система автоматично надасть необхідну інформацію після аналізу призначеного для користувача запиту. Всі запити аналізуються на основі їх інформації, запитуваної користувачем, і оновлює базу даних.

Користувачі можуть відслідковувати будь-які дані, пов'язані з їх доменом, і отримувати свої відповіді, не відвідуючи сільськогосподарський довідковий центр. Він об'єднує різні галузі сільського господарства з Agri-Info. Agri-Info не вимагає яких-небудь технічних знань для використання цієї системи. Інформація або запити, отримані від користувача їй, передаються в хмарне сховище для поновлення, а відповідь відправляється назад конкретного користувача на його попередньо налаштованих пристроях (планшетах, мобільних телефонах, ноутбуках і т. д.) через Інтернет.

1.2 Архітетура системи

Архітектура - це базова організація системи, втілена в її компонентах, їх відносинах один з одним і навколишнім середовищем, а також в принципах, що визначають проектування і розробку системи [7]. Для кожної

системи архітектура дуже важлива, оскільки вона охоплює не тільки її структурні аспекти, але також правила її використання та інтеграції з іншими системами, функціональність, продуктивність, гнучкість і надійність. Архітектура також впливає на те, як буде виглядати користувальницький інтерфейс.

1.2.1 Клієнт-серверна архітектура

Клієнт-серверна архітектура це розподілена структура додатків, яка розприділяє задачі або робочі навантаження між постачальниками ресурсу або служби званими серверами, і запитувачів послуг, званими клієнтами. Часто клієнти і сервери обмінюються даними через комп'ютерну мережу на іншому обладнані, але і клієнт і сервер можуть знаходитися на одній системі. На хості сервера запускається одна або декілька серверних програм, які сумісно використовують свої ресурси з клієнтами. Клієнт не розділяє жодних з своїх ресурсів, але він запитує контент або послугу з сервера. Тому клієнти ініціюють сеанси зв'язку з серверами, котрі очікують запитів що входять. Прикладами комп'ютерних додатків, які використовують модель клієнт-сервер, є електрона пошта, сервер друку і всесвітня мережа [3]. Схема архітектури зображена на рисунку 1.1[9]

Рисунок 1.1 – Клієнт – серверна архітектура

Клієнт-серверні додатки діляться на логічних три (рис. 1.2 [8]): призначені для користувача інтерфейсу, доступу до даних і логіки програми,

що працюють з базою даних. Користувач системи взаємодіє з ним через користувацький інтерфейс, база даних зберігає дані, що описують предметну область додатки, а логічний рівень додатки реалізує всі алгоритми, пов'язані з предметною областю.

Рисунок 1.2 - Логічні рівні додатку

1.2.2 Дворівнева архітектура клієнт-сервер

Оди з варіантів клієнт-серверної архітектури є дворівнева класична архітектура. Клієнт-серверний додаток в даному випадку відноситься до інформаційної системи, заснованої на використанні серверів баз даних. Схематично таку архітектуру можна представити так, див. рис. 1.3.

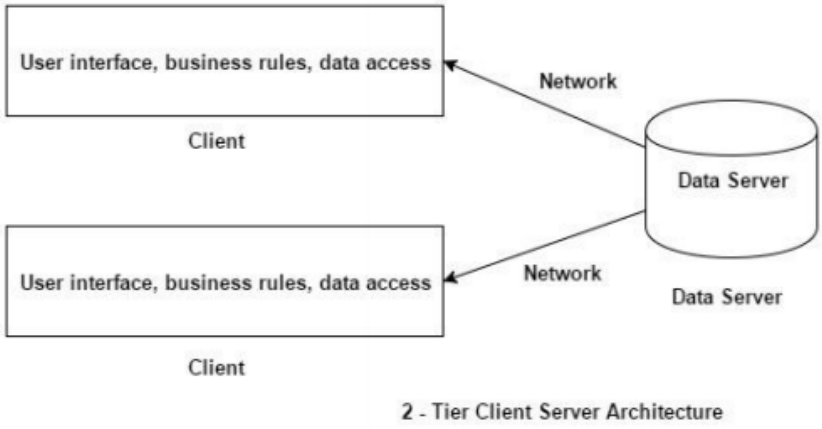


Рисунок 1.3 - Класичне уявлення архітектури клієнт-сервер

Дворівневий клієнт / сервер працює, коли більшість або вся логіка та дані програми розміщуються на сервері. Клієнт інтегрується із шаром презентації та звертається до сервера для виконання конкретних завдань та обробки.

Наприклад, основна програма та дані встановлюються на центральному сервері. Один або кілька клієнтських пристроїв використовує його клієнтський додаток для запиту даних або процесів з сервера. Сервер надсилає необхідні дані або виконує процес для виконання запиту. В іншому дворівневому екземплярі клієнт / сервер, наприклад архітектура резервного копіювання даних, доступ до програми та логіка можуть бути з клієнтським пристроєм, тоді як сервер зберігає та надає основні дані. [8].

1.2.3 Трирівнева клієнт серверна ахітектура

Найбільш поширеною є трирівнева архітектура, в якій інтерфейс призначений для користувача, доступ до даних і програмна логіка розподілені на окремі компоненти системи, які можуть працювати на незалежних комп'ютерах (див. рис. 1.4 [8]).

Рисунок 1.4 - Трирівнева архітектура

Переваги використання трирівневої архітектури:

- Це робить логічне розділення між бізнес-шаром та презентаційним шаром та шаром бази даних;
- Міграція до нових графічних середовищ відбувається швидше;
- Оскільки кожен рівень є незалежним, можна включити паралельний розвиток кожного рівня, використовуючи різні набори розробників;
- Простий у обслуговуванні та розумінні великий проект і складний проект.
- Оскільки прикладний рівень знаходиться між шаром бази даних та шаром презентації, то рівень бази даних буде більш захищеним, а клієнт не матиме прямого доступу до бази даних.
- Опубліковані дані з шару презентації можуть бути перевірені або перевірені на рівні програми перед оновленням до бази даних.
- Безпека бази даних може надаватися на рівні додатків.
- Прикладний рівень або середній рівень або бізнес-шар можуть бути захисним захистом для бази даних.
- Нові правила або нові правила перевірки можуть бути визначені в будь-який час і зміни, внесені до середнього шару, не впливатимуть на рівень презентації.
- Визначте будь-яку логіку один раз у бізнес-шарі, і цю логіку можна поділити між будь-якою кількістю компонентів у презентаційному шарі.
- Ми можемо відображати лише необхідні методи з бізнес-рівня в шарі презентації.
- Ми можемо приховати зайві методи від бізнес-шару в презентаційному шарі.
- Легко застосувати об'єктно-орієнтовану концепцію.
- Легко оновити запити постачальника даних.

Недоліки використання трирівневої архітектури:

- Для реалізації навіть невеликої частини програми це забирає багато часу.

- Потрібна хороша експертиза в об'єктно-орієнтованій концепції (класи та об'єкти).
- Це складніше будувати.

1.2.4 Багаторівнева архітектура клієнт-сервер

Багатошарова архітектура клієнт-сервер - це пряме розширення розділення додатків на рівні програми, логіці програми та доступу до даних. Існують різні варіанти реалізації багатоканальної архітектури. Один з них: різні посилання взаємодіють відповідно до логічної організації програми. Цей тип розподілу називається вертикальним. Його основна особливість - на різних машинах розміщуються логічно рівні компоненти. Інший тип - горизонтальне розділення. При такому типі розподілу клієнт або сервер можуть містити: логічно однорідного модуля, фізично розділені частини і робота можуть відбувається з кожною з частин незалежно. Це робиться для збалансування навантаження. Існують і інші варіанти організації архітектури, наприклад, розподілені як вертикально, так і горизонтально. [10].

1.2.5Сервісно-орієнтована архітектура

SOA (Service Oriented Architecture) - концепція сервіс-орієнтованої архітектури, призначена для вирішення питань інтеграції інформаційної інфраструктури компанії за рахунок побудови архітектури, що дозволяє інтегрувати з максимальною гнучкістю різнорідні додатки. Рис. 1.5 [11].

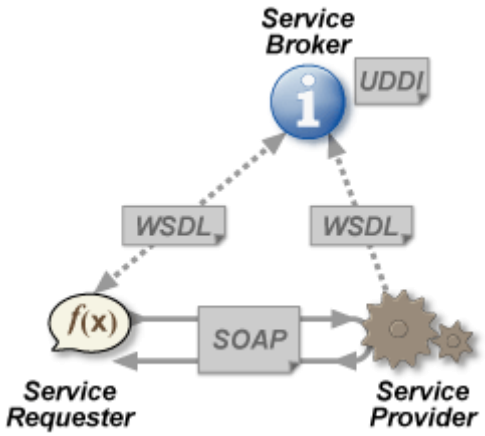


Рисунок 1.5 - Сервісно-орієнтована архітектура

Сервіс-орієнтована архітектура будується за рахунок проектування та розробки сервісів і засобів їх підключення. Сервіс являє собою певну роботу або бізнес-функцію, призначену для забезпечення узгодженої роботи додатків.

SOA не залежить від мов програмування, платформ або протокольних специфікацій, за допомогою яких сервіси розробляються. Зокрема принципами SOA є:

- Архітектура не прив'язана до певної технології
- Незалежність організації системи від використовуваних платформ
- Незалежність організації системи від вживаних мов програмування
- Використання сервісів, незалежних від конкретних додатків, з однаковими інтерфейсами доступу до них
- Організація сервісів як слабосвязанних компонент для побудови систем

Поява SOA обумовлено бажанням перейти від програмування комплексних інформаційних продуктів до можливості «збирання» задовольняє інформаційні потреби системи з різнорідних додатків. Так само не маловажним аспектом SOA є висока гнучкість, яка досягається за рахунок можливості швидкого коректування бізнес-логіки - зміна, що вноситься в бізнес-функцію, в результаті торкнеться всі необхідні додатки.

З точки зору побудови архітектури, SOA включає наступні ключові елементи:

- додаток-клієнт, як правило, відповідає за ініціацію бізнес-процесу і отримує результати його виконання
- сервіс - програмний компонент, який має чітко визначену функціональність і відповідний бізнес-задачі
- репозиторій сервісів, що надає механізми для виявлення сервісів та отримання додаткової інформації про них
- сервісна шина, що забезпечує взаємодію між усіма компонентами архітектури

Так як сама по собі SOA не є продуктом або технологією, але стилем архітектури, розгортання SOA має на увазі використання різних існуючих продуктів і технологій. Наприклад, такі мови як BPEL , розширюють концепцію SOA, надаючи метод об'єднання дрібних сервісів в більш великі бізнес-сервіси, що включаються до складу автоматизованих бізнес-процесів. Таким чином, BPMS визначає правила, згідно з якими будуть викликатися сервіси і передаватися інформацію між ними, що пояснює зростання популярності спільного використання SOA і BPM .

Для реалізації поставленого завдання було проведено огляд різних архітектурних систем, а саме визначення поняття архітектури розподіленої системи, описані такі можливі варіанти архітектури: архітектура клієнт-сервер (дворівнева, трирівнева та багаторівнева), сервісно-орієнтована архітектура.

Найбільш поширеною архітектурою інформаційних систем є клієнт серверна дворівнева архітектура. Вона вважається класичною, оскільки інші види архітектури є її продовженням. Три головні рівні системи з трирівневою архітектурою: інтерфейс користувача, логіка програми та доступ до даних, є самостійними складовими і можуть працювати на різних комп’ютерах. Такий розподіл за рівнями найбільш логічним та зрозумілим. Серед головних переваг даної архітектури є гарна масштабованість системи, надійність та

висока безпека, відсутність необхідного адміністрування клієнтського програмного забезпечення.

1.3. Конструювання програмної системи

1.3.1 Вибір засобів розробки системи

Для розробки програмної системи було вибрано технологію .Net

.NET Framework - це програмна платформа, випущена Microsoft в 2002 році. Платформа заснована на Common Language Runtime (CLR), яка підходить для програмування різних мов.

Функціональність CLR доступна в будь-якій мові програмування. Основна ідея розробки .NET Framework полягала в тому, щоб забезпечити свободу розробника, що дозволяє створювати додатки різних типів і пристроїв.

Другий принцип полягає в тому, щоб зосередитися на системі під керуванням сімейних систем Microsoft Windows. Програма .NET Framework, написана на будь-якому підтримуваному мовою програмування, спочатку перекладається компілятором на загальний проміжний байт-код .NET Intermediate Language (CIL) (раніше називався Microsoft Intermediate Language, MSIL). З точки зору .NET, є колекція, англ. збірка. NGen.exe - це виконуваний код для цільового процесора.

Використання віртуальної машини є кращим, оскільки усуває необхідність для розробників піклуватися про апаратних функціях. При використанні віртуальної машини вбудований JIT-компілятор CLR «на льоту» (якраз вчасно) перетворює проміжний байтовий код в машинні коди потрібного процесора. Сучасна технологія динамічної компіляції забезпечує високий рівень продуктивності. Віртуальна машина CLR також відповідає за

базову безпеку, управління пам'яттю і виключення, виконуючи частину роботи розробника.

Платформа .NET Framework і інфраструктура спільної мови (CLI), розроблена Microsoft і затверджена ISO і ECMA. CLI вказують типи даних .NET, формат метаданих структур, системи виконання байт-коду і багато іншого [12].

.NET є багаторівневим, модульним і ієрархічним. Кожен рівень .NET Framework - це рівень абстракції. Мови .NET є головними і найбільш абстрагованими від інших рівнів. Common Language Runtime (CLR) - це найнижчий рівень. Це важливо, тому що CLR пов'язані з операційним середовищем управління додатками. Платформа .NET Framework. .NET - це ієрархічна структура. Загальна архітектура .NET Framework на рис. 1.6

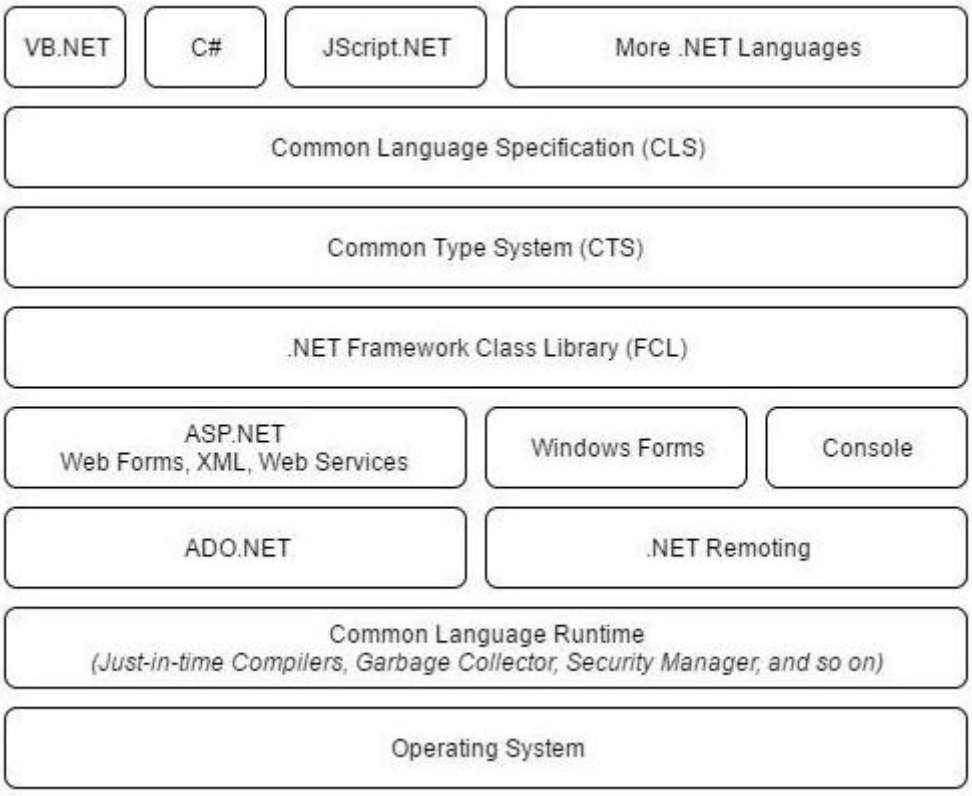


Рисунок 1.6 - Архітектура платформи .NET

Об'єктні класи .NET, доступні всім підтримуваним мов програмування, містяться в бібліотеці Framework Class Library (FcL). У FCL входять класи

Windows Forms, ADO.NET, ASP.NET, Language Integrated Query, Windows Presentation Foundation, Windows Communication Foundation і інші (рис. 1.7[12]). Ядро FCL називається Base Class Library (BCL)

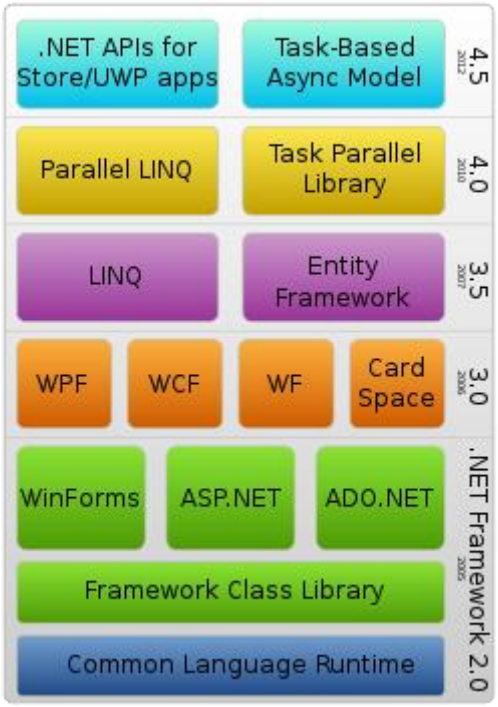


Рисунок 1.7 - Стек технологій .NET Framework

1.3.1.1Засоби створення інтерфейсу користувача

.NET Framework пропонує чотири API для використання при створенні додатків з призначеним для користувача інтерфейсом.

• ASP.NET (System.Web.UI). Призначений для створення тонких клієнтських додатків, які запускаються в стандартному веб-браузері.

• Silverlight. Призначений для створення розширених призначених для користувача інтерфейсів всередині веб-браузера.

• Windows Presentation Foundation (System.Windows). Призначений для створення товстих клієнтських додатків.

• Windows Forms (System.Windows.Forms). Розроблено для підтримки застарілих додатків для товстих клієнтів

У загальному випадку додаток тонкого клієнта зводиться до веб-сайту; додаток товстого клієнта - це програма, яку кінцевий користувач повинен завантажувати або встановлювати на своєму комп'ютері. Успадковане програмне забезпечення – програмне забезпечення, що з деяких причин перестало задовольняти потребам, що змінилися, але продовжує використовуватись через перешкоди, що виникають при спробах його замінити

ASP.NET

Програми, написані з використанням ASP.NET, розміщуються на сервері Windows IIS і доступні практично через усі веб-браузери.

Нижче наведені переваги ASP.NET в порівнянні з технологією товстих клієнтів.

• Немає необхідності розгортати на стороні клієнта.

• Клієнти мають можливість використовувати платформи, відмінні від Windows.

• Простота розгортань оновлень.

Крім того, оскільки більша частина коду, який повинен бути написаний в додатку ASP.NET, виконується на сервері, рівень доступу до даних, за прогнозами, буде виконуватися в тому ж домені додатку - без обмежень безпеки або масштабованості. Навпаки, товстий клієнт, який робить те ж саме, як правило, не так безпечний і масштабується. (Рішення товстого клієнта полягає в створенні проміжного шару між клієнтом і базою даних. Цей проміжний рівень виконується на віддаленому сервері додатків (часто з сервером бази даних) і взаємодіє з товстими клієнтами через WCF, веб-служби або віддалене взаємодія.)

При написанні веб-сторінок ви можете вибирати між традиційним API веб-форм і новим MVC (Model-View-Controller) MVC API. Обидва побудовані на технології ASP.NET. Технологія Web Forms була частиною .NET Framework з самого початку, а MVC був реалізований набагато пізніше

у відповідь на успіх Ruby on Rails і MonoRail. Технологія MVC з'явилася в .NET Framework 4.0 і з тих пір стала більш складною. В цілому, він забезпечує кращу абстракцію програмного забезпечення, ніж веб-форми; це також дозволяє краще контролювати генерується розмітку HTML. Однак є один аспект, в якому MVC програє веб-форм - візуальний конструктор. Таким чином, використання веб-форм більш зручно для створення веб-сторінок, що містять в основному статичний контент.

Обмеження ASP.NET в значній мірі відображають загальні обмеження систем тонких клієнтів:

• інтерфейс веб-браузера значно обмежує можливості;

• підтримувати статус на стороні клієнта (або від імені клієнта) громіздко.

Однак інтерактивність і чутливість додатків, створених за допомогою ASP.NET, можна підвищити за допомогою клієнтських сценаріїв або технологій, таких як AJAX. Працювати з AJAX стало простіше завдяки використанню таких бібліотек, як jQuery.

Типи, призначені для створення програмного забезпечення ASP.NET, знаходяться в просторі імен System.Web.UI і його підпросторах; вони упаковані в збірку System.Web.dll.

Silverlight

Формально Silverlight не є частиною .NET Framework: це окрема платформа, яка містить підмножину ключових засобів .NET Framework, а також підтримує можливість виконання у вигляді плагіна веб-браузера. Графічна модель Silverlight - по суті, підмножина WPF, це дозволяє застосовувати знання та навики роботи з останньою при розробці Silverlight-додатків. Плагін Silverlight доступний як міжплатформений завантажуваний файл для веб-браузерів, що дуже схоже на Macromedia Flash. Flash володіє набагато більшою установною базою, тому домінує в цій галузі. З цієї

причини Silverlight, як правило, використовується для написання сценаріїв, наприклад, в корпоративних мережах.

Windows Presentation Foundation (WPF)

Технологія WPF з'явилася в .NET Framework 3.0 і призначена для створення програмного забезпечення товстих клієнтів. Нижче перераховані переваги WPF в порівнянні з Windows Forms.

• Вона підтримує розвинену графіку, включаючи довільні трансформації, тривимірну візуалізацію і справжню прозорість.

• Первинна одиниця виміру заснована не на пікселях, тому додатки коректно відображаються за будь-якого налаштування DPI (dots per inch - точок на дюйм).

• Вона має велику підтримку динамічного компонування, що означає можливість локалізації додатку без небезпеки того, що елементи будуть перекривати один одного.

• Візуалізація використовує DirectX і є швидкою, отримуючи переваги від апаратного прискорення графіки.

• Користувацькі інтерфейси можуть бути описані декларативно в XAML файлах, які підтримуються незалежно від файлів відокремленого коду - це допомагає відокремити зовнішній вигляд від функціональності [13].

Типи для написання WPF-додатків знаходяться в просторі імен System. Windows і всіх його підпросторах крім System. Windows. Forms.

Windows Forms

Windows Forms - це товстий клієнтський API, аналогічний .NET Framework. У порівнянні з WPF це відносно проста технологія, яка пропонує більшість можливостей, необхідних при написанні типового додатки для Windows. Також важливо підтримувати застарілі програми. Однак у порівнянні з WPF Windows Forms має ряд недоліків.

• Позиції та розміри елементів управління задаються в пікселях, що створює ризик неправильного відображення додатків для клієнтів з настройками DPI, відмінними від налаштувань розробників.

• Нестандартний API малювання GDI + - це інтерфейс, який, незважаючи на свою гнучкість, повільно рендерить великі області (і без подвійної буферизації може мерехтіти).

• Важко домогтися надійності динамічного макета.

Останній пункт - хороша причина віддати перевагу WPF формам Windows Forms. Подібні WPF елементи макета, такі як Grid, дозволяють легко організувати мітки і текстові поля таким чином, щоб вони завжди були вирівняні - навіть при зміні мови локалізації - без неакуратної логіки розташування і мерехтіння. Крім того, вам не потрібно зводити все до найменшого спільного знаменника з точки зору дозволу екрану - елементи макета WPF спочатку були розроблені для підтримки зміни розміру.

З іншого боку, технологія Windows Forms відносно проста в освоєнні і все ще широко підтримується в сторонніх елементах управління [14].

Типи Windows Forms знаходяться в просторах імен System.Windows.Forms (збірка System.Windows.Forms.dll) і System.Drawing (збірка System.Drawing.dll).

1.3.2 Вибір СУБД

Хмарна підсистема

Ця підсистема містить платформу, в якій веб-сервіс для сільського господарства розміщується в хмарі. Сільськогосподарський веб-сервіс дозволяє обробляти інформацію про сільське господарство, що надається користувачами (фахівцем із сільського господарства, сільськогосподарським чиновником і фермером) в різних галузях сільського господарства: урожай, погода, грунт, шкідник, добрива, продуктивність, іригація, худобу і обладнання. Користувачі в основному діляться на три категорії: експерт з

сільського господарства, співробітник по сільському господарству і фермер, як вже обговорювалося. Ці дані зберігаються в хмарному сховищі в різних класах для різних доменів з унікальним ідентифікаційним номером. Інформація постійно відстежується, аналізується і обробляється Агрі-Інфо. Процес аналізу складається з різних підпроцесів: вибір, попередня обробка даних, перетворення, класифікація та інтерпретація. Було розроблено різні класи для кожного домена і підкласів для подальшої категоризації інформації. У сховище сховища призначені для користувача дані класифікуються на основі різних визначених класів кожного домена. Ця інформація далі передається експертам по сільському господарству і працівникам сільського господарства для остаточної перевірки за допомогою попередньо налаштованих пристроїв. Крім того, ряд користувачів можуть використовувати хмарний веб-сервіс для сільського господарства, тому було інтегровано менеджер якості обслуговування і менеджер автономних ресурсів в хмарну підсистему. Менеджер QoS (quality of service) визначає вимоги QoS на підставі кількості та типу запитів користувачів. Грунтуючись на вимогах QoS, менеджер автономних ресурсів визначає вимоги до ресурсів, розподіляє і виконує ресурси на рівні інфраструктури. Монітор продуктивності використовується для перевірки продуктивності системи і її автоматичної підтримки. Якщо система не зможе обробити запит автоматично, система згенерує попередження.

1.3.3 Хмарний веб-сервіс для сільського господарства

Хмарний веб-сервіс для сільського господарства надає призначену для користувача платформу, в якій користувач може отримати доступ до сервісу для сільського господарства. Функціональні аспекти Agri-Info показані на рисунку 1.8. Ми визначили різні функції, що надаються Agri-Info. По-перше, веб-сервіс сільського господарства дозволяє користувачеві створити профіль для взаємодії з Agri-Info.

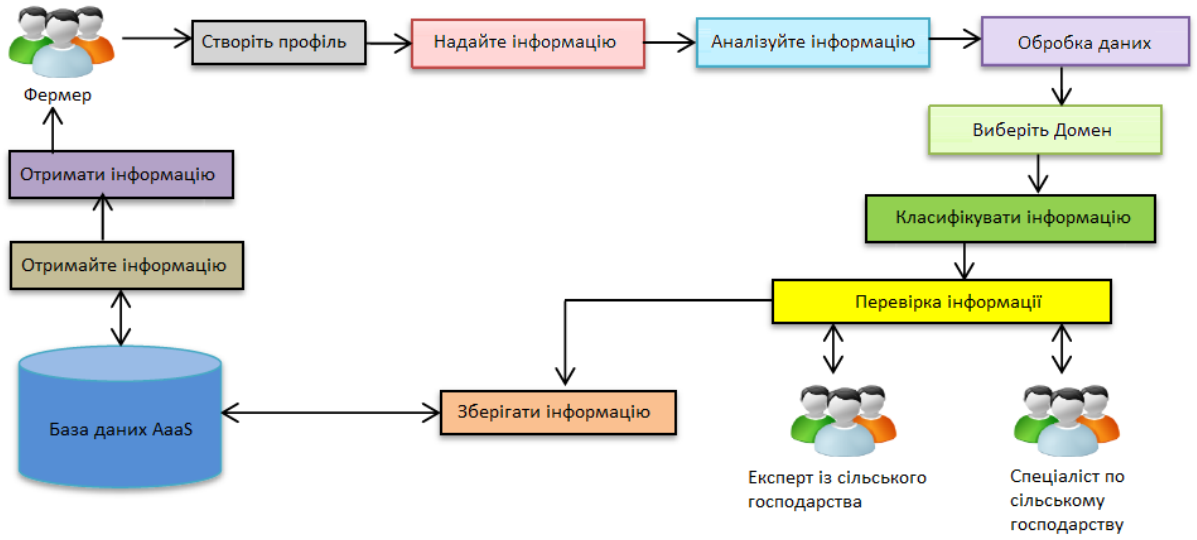


Рисунок 1.8 - Функціональні аспекти Agri-Info

Після створення профілю користувач зобов'язаний надати свої особисті дані разом з інформацією про предметну область. AgriInfo аналізує інформацію, щоб перевірити, чи є дані повними чи ні для подальшої обробки, виконуючи різні перевірки. Далі дані обробляються, надмірність даних видаляється, а дані використовуються для вибору домену, до якого відносяться дані. Інформація класифікується належним чином з унікальним ідентифікаційним номером. Ця інформація далі передається експертам по сільському господарству і працівникам сільського господарства для остаточної перевірки за допомогою попередньо налаштованих пристроїв. Після успішної перевірки інформації вона зберігається в базі даних AaaS. Якщо користувач хоче знати відповідь на свій запит, система автоматично діагностує запит користувача і відправляє відповідь цьому користувачу.

1.5 Використання програмної системи

1.5.1 Розгортання програмної системи та системні вимоги

Метою цієї роботи є демонстрація можливості впровадження та розгортання методу автономного управління ресурсами на реальних хмарних

ресурсах. Вузли в цій системі можуть бути додані або видалені під час виконання автономним диспетчером ресурсів в залежності від вимог. Ключовими компонентами хмарної середовища є: призначений для користувача інтерфейс, веб-сервіс (електронне сільське господарство), Azure, планувальник ресурсів і ресурси. Однак, щоб дати уявлення про хмарної середовищі, в якій реалізована запропонована методика автономного управління ресурсами, ми коротко представимо її роботу таким чином:

1. Споживач хмари відправляє свій запит в призначений для користувача інтерфейс, який містить опис робочого навантаження (кількість призначених для користувача запитів, що підлягають обробці).

2. Хмарна веб-служба Agri-Info (електронне сільське господарство) розгорнута на платформі Azure (використовується в якості масштабується хмарного проміжного програмного забезпечення для взаємодії між SaaS і IaaS).

3. Конфігурація ресурсу ідентифікується для планування кількості робочих навантажень (список відправлених призначених для користувача запитів).

4. Планувальник ресурсів планує ресурси для робочих навантажень на основі алгоритму розподілу ресурсів.

5. Менеджер автономних ресурсів використовує датчики для вимірювання продуктивності системи з точки зору QoS, щоб уникнути недовантаження і перевантаження ресурсів, і оновлена інформація обмінюється між усіма автономними модулями через Effector.

6. Після успішного виконання призначених для користувача запитів це додатково повертає ресурси в пул ресурсів.

7. В кінці автономний блок повертає оновлені дані експерименту разом з обробленим запитом (робочим навантаженням) назад споживачеві хмари.

Було виконано різну кількість експериментів в різних типах верифікації, порівнявши Agri-Info (автономний з підтримкою QoS) з

технікою управління ресурсами, яка не базується на QoS (неавтономної), в якій при розподілі не враховуються параметр QoS і механізм автономного планування. ресурси для обробки запитів користувачів. Експеримент проводився з різною кількістю користувальницьких запитів (1-70) для перевірки вартості виконання, часу виконання, використання ресурсів і затримки . Щоб перевірити Agri-Info, була представлена оцінка продуктивності за допомогою веб-служби, тобто хмари електронного сільського господарства, з урахуванням параметрів QoS на рівні обслуговування.

Ми розробили цю запропоновану систему за такими основними причинами:

• Складно обговорити технічні питання вручну, якщо експерт не присутній в тому ж офісі і особисто керує зустріччю.

• Втрата часу на пошук і перегляд всіх документів, перелічених вручну.

Хмарна веб-служба електронного сільського господарства являє собою глобальну спільноту практиків, в якому користувачі експерти по сільському господарству, співробітники сільського господарства і фермери з усього світу обмінюються інформацією, ідеями та ресурсами, пов'язаними з використанням інформаційні та комунікаційні технології для сталого сільського господарства в різних областях. В електронному сільському господарстві використовуються інформаційні, візуальні і комунікаційні технології в поєднанні з Інтернет-додатками з метою надання поліпшених сільськогосподарських послуг та інформації фермерам і фермерському спільноті. Електронне сільське господарство реалізовано як централізована система в хмарному середовищі. Кілька користувачів використовують цю систему з кількох місць. Користувачі відправляють свій запит в Agri-Info, а Agri-Info дає відповідь на запит, відправлений користувачем. Різні користувачі використовують систему в залежності від прав, наданих користувачеві. Користувачі входять в Agri-Info і отримують доступ до бази

даних. Це комп'ютеризоване управління декількома аспектами, пов'язаними з сільським господарством, для кращого управління різними областями, такими як зернові культури і т. д., що відбуваються в навколишньому середовищі, що призводить до підвищення ефективності роботи і веде до задоволеності клієнтів і довірі до організації завдяки своєчасному і правильному управлінню. Основними цілями розробки хмарного веб-сервісу електронного сільського господарства є:

- Давати пропозиції користувачам за запитами з різних областей сільського господарства.
- Надавати точну інформацію користувачам і економити час користувачів.
- Кілька фермерів можуть спілкуватися один з одним і обговорювати свої проблеми і будь-які питання про будь-якій області.

Інтерфейс хмарної веб-служби електронного сільського господарства показаний на рисунку 1.9.

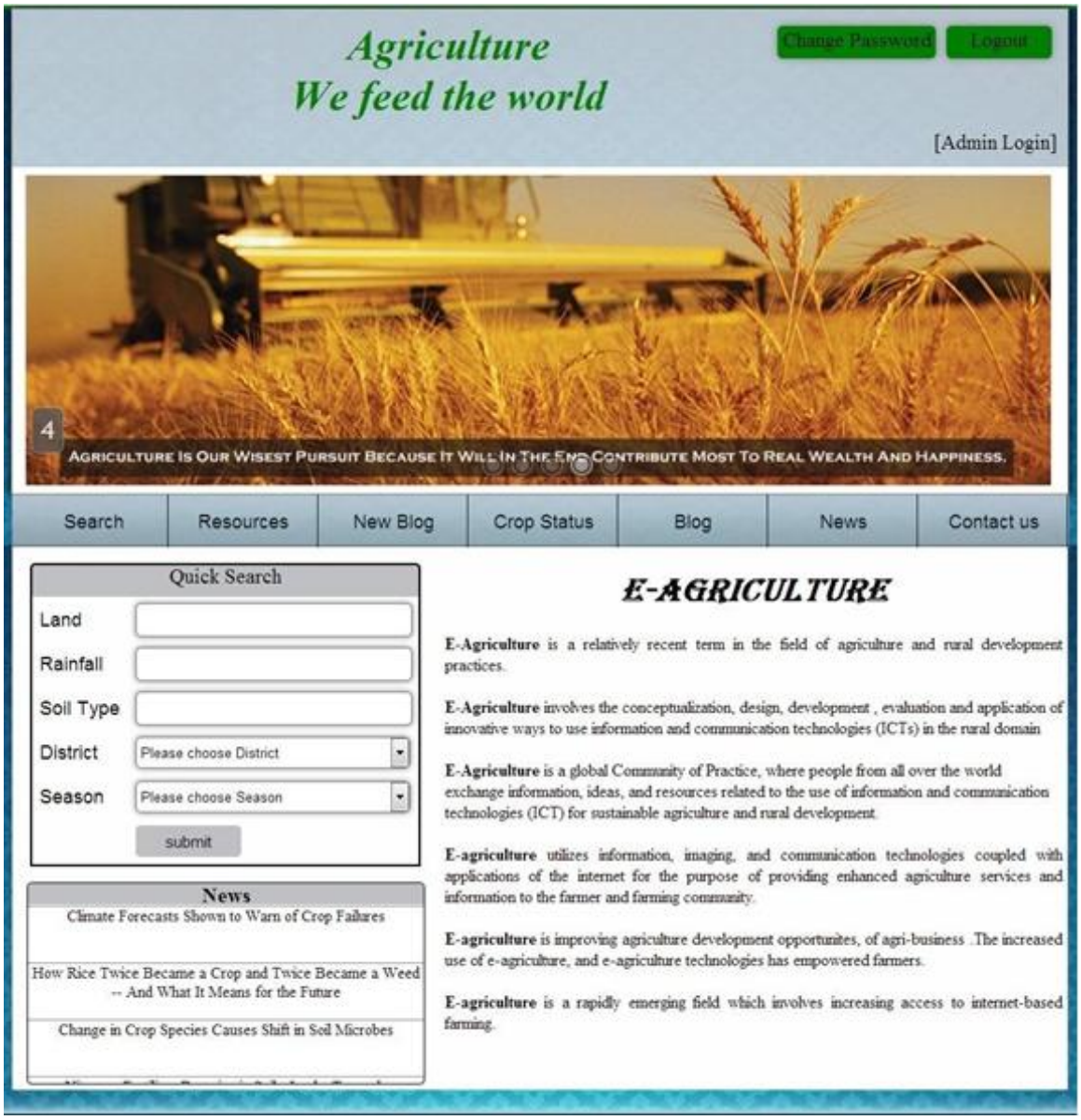


Рисунок 1.9 - Інтерфейс користувача хмарного веб-сервісу електронного сільського господарства

Основними функціями веб-сервісу електронного сільського господарства на основі хмари є: пошук інформації за допомогою швидкого пошуку, перегляд інформації про ресурси (обладнання та інструменти в сільському господарстві), отримання статусу врожаю, останні новини тощо. На рисунку 10 показано пошук урожаю у хмарному електронному секторі Веб-сервіс сільського господарства.

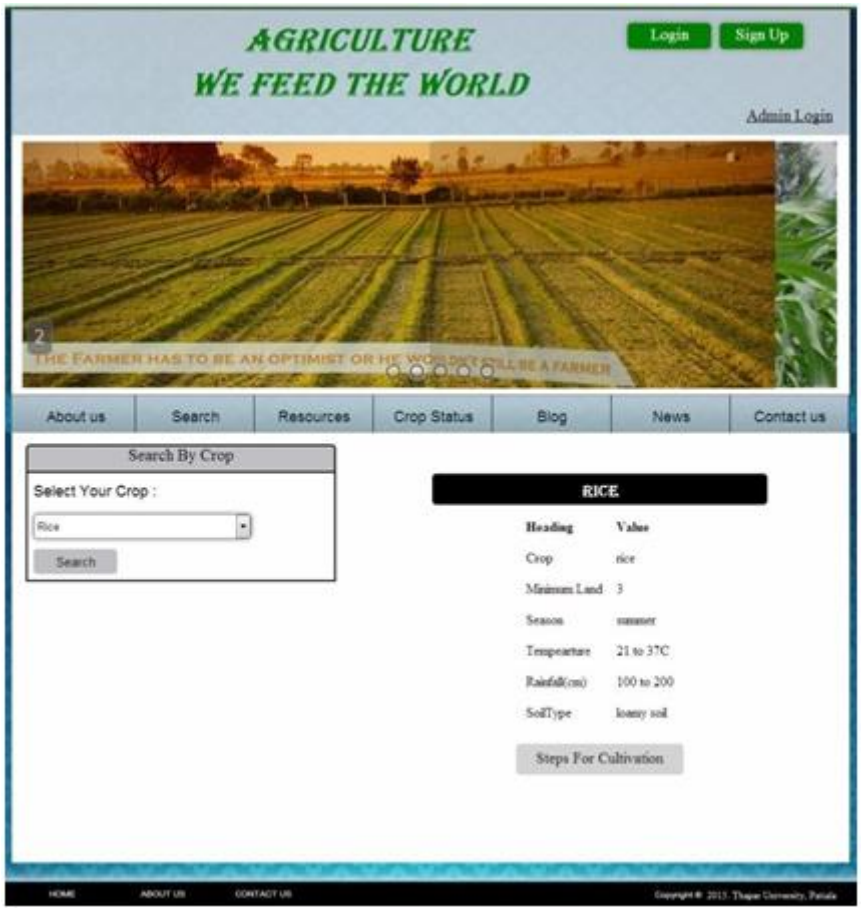


Рисунок 1.10 - Пошук урожаю у хмарній веб-службі електронного сільського господарства

На рисунку 1.11 показана обробка запиту веб-сервісом електронного землеробства на базі хмари та результати після виконання запиту.

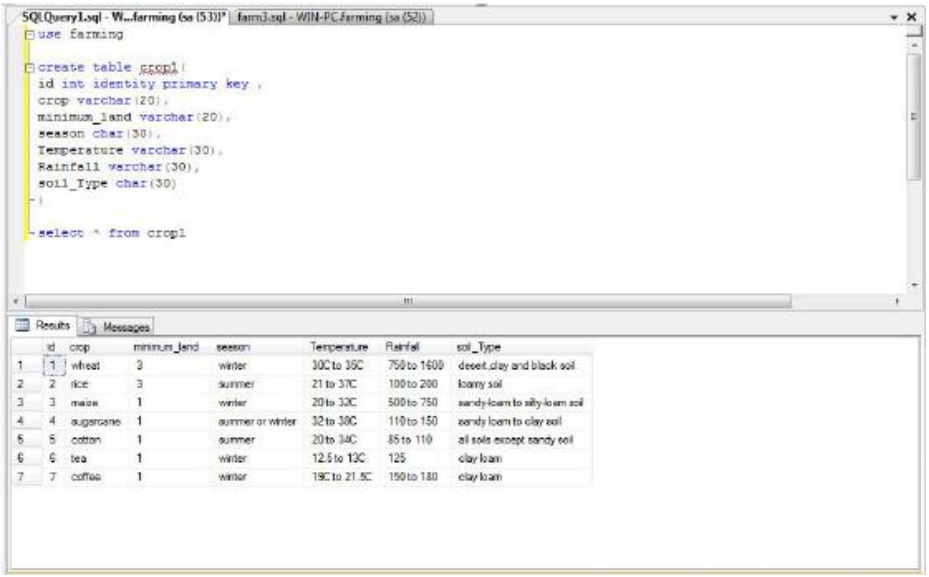


Рисунок 1.11 - Обробка запитів та результати в хмарній веб-службі електронного сільського господарства

2. Тестування програмної системи

Існуюча система сільського господарства управляє ресурсами без урахування концепції автономних параметрів та параметрів якості, що призводить до зниження продуктивності. Для вирішення цієї проблеми було розглянуто чотири параметри QoS для перевірки Agri-Info, що чітко показує, що методи управління автономними ресурсами на основі QoS працюють краще. Зі збільшенням кількості запитів користувачів значення затримки збільшується. Значення затримки в автономній системі, що усвідомлює QoS, є меншою порівняно з неавтономним плануванням ресурсів (неавтономним) для різної кількості запитів користувачів, як показано на рисунку 2.1. Максимальне значення затримки - 220 секунд, а мінімальне значення - в техніці автономного управління ресурсами, усвідомленою QoS, затримка становить 65 секунд.

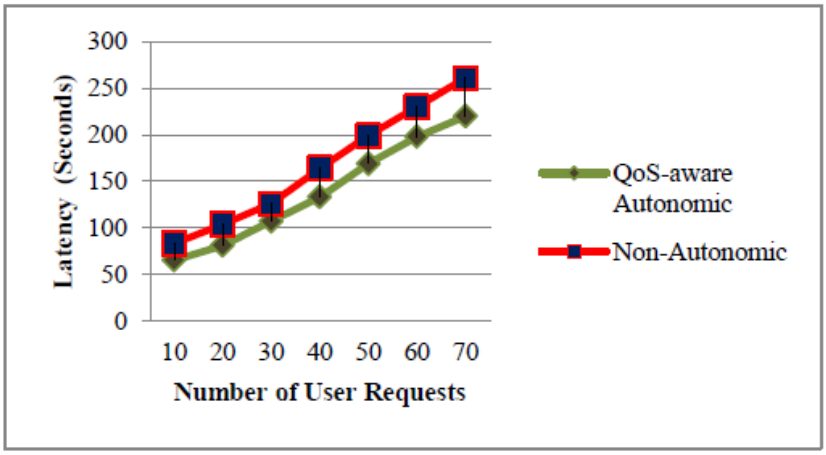


Рисунок 2.1 - Вплив зміни кількості запитів користувачів на затримку

Ми обчислили значення середньої вартості як для технології управління автономними ресурсами, заснованими на хмарі, що базується на QoS (незалежна від QoS), так і для управління ресурсами, що не базуються на QoS, з різною кількістю запитів користувачів, як показано на рисунку 13.

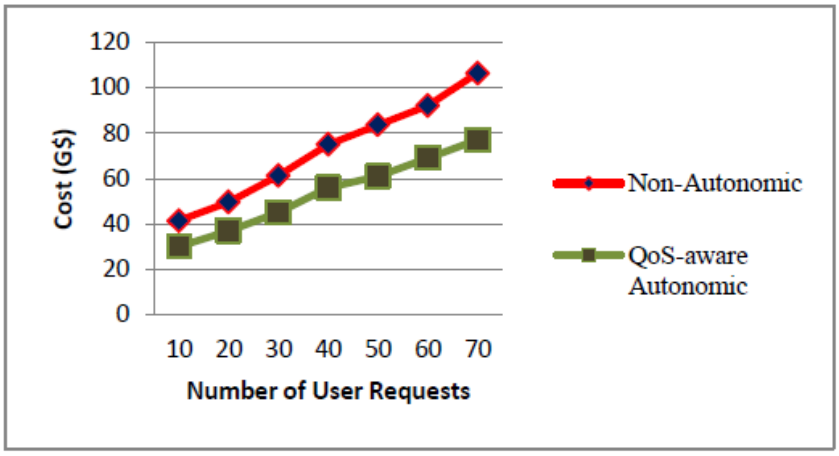


Рисунок 2.2 - Вплив зміни кількості запитів користувачів на середню вартість

Вартість виконання. це доповнення витрат на ресурси та штрафних витрат. Agri-Info визначив різні рівні штрафу на основі вимог QoS. Час затримки - це різниця строку і часу, коли фактично завершено навантаження. Середня вартість зростає зі збільшенням кількості запитів користувачів. QoS-обізнаність в автономії чудово справляється з різною кількістю запитів користувачів. За 70 запитів користувачів середня вартість автономної QoS є на 37,6% меншою, ніж техніка управління ресурсами, що не базуються на QoS.

Як показано на рисунку 2.3, час виконання збільшується зі збільшенням кількості навантажень.

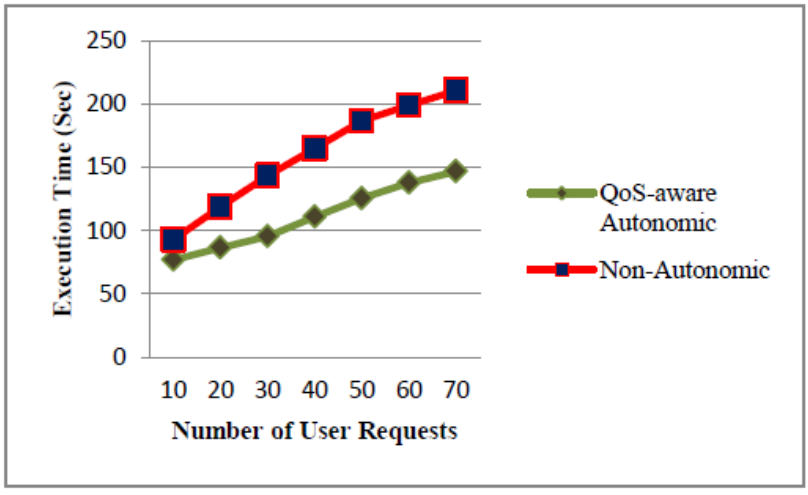


Рисунок 2.3 - Вплив часу виконання зі зміною кількості призначених для користувача запитів

За 45 запитів користувачів час виконання методики управління автономними ресурсами, усвідомленої QoS (QoS-обізнаний) на 33% менше, ніж техніка управління ресурсами на основі QoS. Після 30 запитів користувачів час виконання різко збільшується в техніці управління ресурсами, що не базуються на QoS, але автономія, обізнана з QoS, працює краще, ніж неавтономна методика. Зі збільшенням кількості запитів користувачів зростає відсоток використання ресурсів. Відсоток використання ресурсів у техніці управління автономним управлінням, усвідомленою QoS (автономна QoS) більший порівняно з управлінням ресурсами, заснованим на QoS (неавтономним), при різній кількості запитів користувачів, як показано на рисунку 2.4.

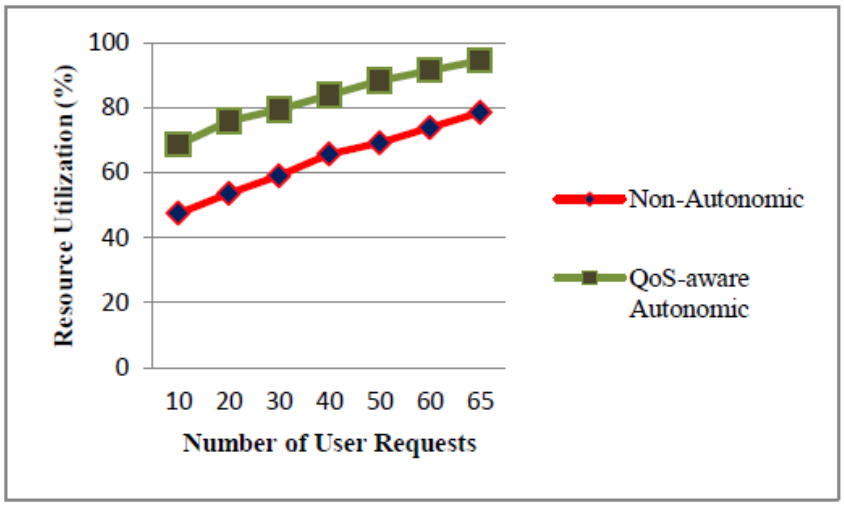


Рисунок 2.4 - Вплив зміни кількості запитів користувачів на використання ресурсів

Максимальний відсоток використання ресурсів становить 94,66% на 65 запитів користувачів у QoS-обізнаному автономному, але QoS-обізнана автономія працює краще, ніж неавтономна методика. Таблиці 2.1 і 2.2 описують порівняння часу виконання, який використовується для обробки різної кількості запитів (малих, середніх та великих) у хмарному середовищі для Agri-Info (QoS-обізнаного автономічного) та не-QoS-методу управління ресурсами (неавтономічний)

Таблиця 2.1. Варіація часу виконання з кількістю запитів користувачів

Кількість запитів користувачів	Час виконання (сек)	
	4 ядра процесора	
	QoS-обізнаний автономний	Неавтономний
Малий (1-20)	10	15
Середній (21-45)	76	100
Великий (46-70)	763	1624

Таблиця 2.2. Варіація часу виконання з кількістю запитів користувачів

Кількість запитів користувачів	Час виконання (сек)	
	8 ядра процесора	
	QoS-обізнаний автономний	Неавтономний
Малий (1-20)	316	158
Середній (21-45)	698	1636
Великий (46-70)	763	1624

У цьому експерименті було розглянуто дві різні хмарні інфраструктури з різною конфігурацією процесора (4-ядерний процесор та 8-ядерний процесор) для вимірювання зміни часу виконання з різною кількістю запитів користувачів. Через обмежену експериментальну оцінку реального хмарного середовища детальний аналіз пропозиції QoS був проведений за допомогою хмарного модельованого середовища за допомогою CloudSim.

2.1 Результати на основі моделювання

Існує безліч методів і технологій для оцінки ефективності управління ресурсами та алгоритмів планування з використанням таких методів оцінки, як аналітичні та імітаційні. Деякі із помітних хмарних інструментів - CloudSim та Cloud Analyst для моделювання. Моделювання дозволяє створювати масштабні віртуальні хмарні середовища та сценарії використання та доступності, які можна повторити та контролювати. Ми вибрали CloudSim [28] для моделювання, оскільки він підтримує як системне, так і моделювання поведінки компонентів хмарної системи, таких як центри

обробки даних, віртуальні машини (VM) та політики планування ресурсів. CloudSim також підтримує моделювання та моделювання середовищ хмарних обчислень, що складаються як з одиночних, так і міжмережевих хмар (федерація Хмар). Крім того, в ньому розкриваються спеціальні інтерфейси для реалізації політики та методики планування розподілу ВМ за сценаріями хмарних обчислень між мережами. CloudSim був встановлений разом з його вимогами за допомогою NetBeans, які надають хмарний сервіс. 3000 незалежних хмарних навантажень (запитів користувачів) були створені випадковим чином у CloudSim як Cloudlets. У таблиці 2.3 наведені характеристики ресурсів та облачок (робочих навантажень), які були використані для всіх експериментів. Хмарні навантаження користувачів моделюються як моделювання незалежних паралельних додатків, що є обчислювальним. Таким чином, залежність даних серед хмарних навантажень у паралельних програмах незначна. Кожне навантаження на хмару є паралельним і тому вважається незалежним від будь-якого іншого хмарного навантаження.

Таблиця 2.3. Параметри планування та їх значення

Параметр	Значення
кількість ресурсів	50-250
Кількість Cloudlets (робочих навантажень)	3000
Пропускна здатність	100 - 1500 B/S
Розмір хмарної робочого навантаження	10000+ (10%–30%) MB
Кількість PE на машину	1
рейтинги PE	100-4000 MIPS
Вартість хмарної робочого навантаження	$3–$5
Розмір пам'яті	2048-12576 MB
розмір файлу	300 + (15%–40%) MB
Вихідний розмір хмарної робочого навантаження	300 + (15%–50%) MB

Була виконана різна кількість експериментів, порівнявши Agri-Info (Autonomic з підтримкою QoS) з технікою управління ресурсами, яка не базується на QoS (неавтономної), в якій не враховуються параметр QoS і механізм автономного планування. Методика планування ресурсів, не заснована на QoS, яка використовується для експериментальної оцінки в цій статті, була розроблена шляхом об'єднання двох традиційних алгоритмів планування ресурсів (Перший Прихід, Перший Сервіс і Круглий Робін), в яких ресурси плануються без урахування параметрів QoS. Експеримент проводився з різною кількістю робочих навантажень (500-3000) для перевірки вартості виконання, часу виконання, використання ресурсів, обчислювальної потужності, доступності, пропускної здатності мережі, задоволеності клієнтів і затримки, а також одного показника продуктивності (кількість пропущених запитів користувачів) для перевірити Agri-Info. Продуктивність всіх параметрів QoS і одна метрика продуктивності були оцінені і в порівнянні з неавтономним плануванням ресурсів. Для перевірки запропонованої нами структури ми провели експерименти на основі моделювання з варіаціями числа представлених робочих навантажень.

Тестовий випадок 1: Наявність: Це відношення середнього часу між відмовою до надійності. Де середній час між відмовою - відношення загальної тривалості роботи до кількості поломок. Де середній час на ремонт - відношення загального простою до кількості поломок

Було опчислено відсоток доступності як для QoS-відомої хмарної автономної інформаційної системи (QoS-обізнаної автономної), так і не-QoS-методики управління ресурсами (неавтономної). Зі збільшенням кількості хмарних навантажень зменшується відсоток доступності. Відсоток доступності автономної QoS-автономії більше порівняно з технікою управління ресурсами, що не базуються на QoS (неавтономна) при різній кількості хмарних навантажень, як показано на Рисунку 2.5 . Максимальний

відсоток доступності становить 89,9% при 500 хмарних робочих навантаженнях .

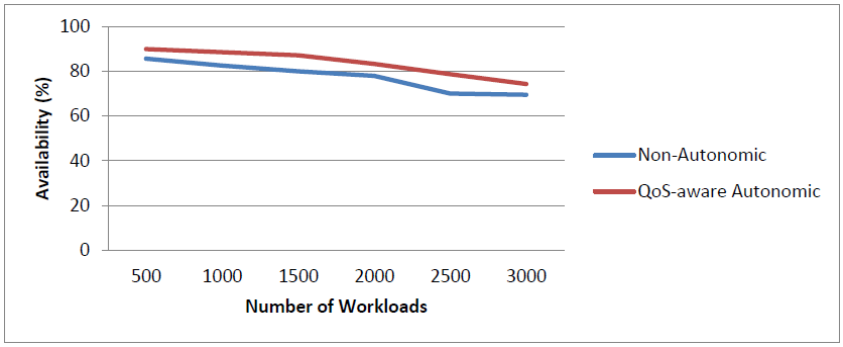


Рисунок 2.5 - Вплив зміни кількості завантажених навантажень на наявність

Тестовий випадок 2: Пропускна здатність мережі

Ми порівняли значення пропускної здатності мережі Qos-відомої хмарної автономної інформаційної системи (QoSaware autonomic) з технікою управління ресурсами на основі QoS (неавтономною), показаної на рисунку 2.6.

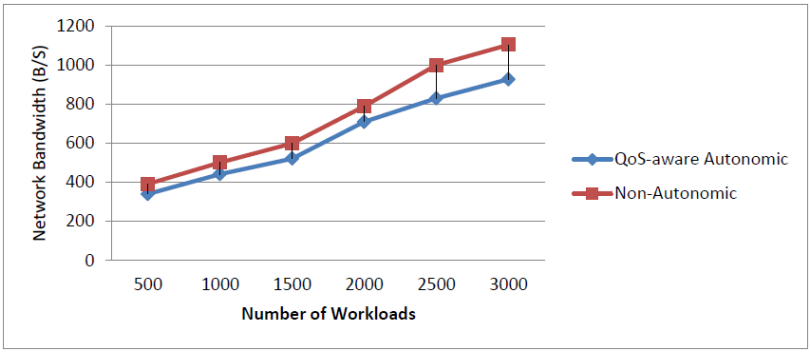


Рисунок 2.6 - Вплив зміни кількості завантажених навантажень на пропускну здатність мережі

Автономія, обізнана з QoS, працює краще, ніж не- автономність після 2000 навантажень. При 3000 робочих навантаженнях пропускна здатність мережі, що використовується в QoS-відомій автономії, на 18,98% менше, ніж техніка управління ресурсами, що не базується на QoS.

Тестовий випадок 3: Задоволення клієнтів: Матриця впевненості та виконання на основі рівня задоволеності хмарним сервісом, як показано в Таблиці 2.4.

Таблиця 2.4. Матриця коденфіційності та завдання

Рівень задоволеності клієнтів	Впевненість (%)
Дуже задоволений	100
Задоволений	75
Нейтральний	50
Незадоволений	25
Повністю незадоволений	0

Було обчислено відсоток задоволеності клієнтів як для хмарної автономної інформаційної системи, заснованої на QoS (автономної QoS), так і для технології управління ресурсами на основі QoS (неавтономної).

Зі збільшенням кількості хмарних навантажень зменшується відсоток доступності. Відсоток задоволеності клієнтів у QoS-обізнаному рівні автономії більший порівняно з технікою управління ресурсами, що не базуються на QoS (неавтономізована) при різній кількості хмарних навантажень, як показано на рисунку 2.7. Максимальний відсоток задоволеності клієнтів становить 90,5% при 500 хмарні навантаження.

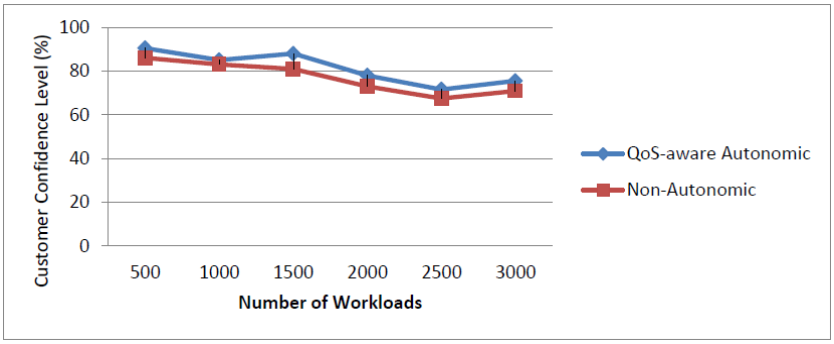


Рисунок 2.7 - Вплив зміни кількості завантажених навантажень на рівень довіри клієнтів

Тестовий випадок 4: Кількість пропущених запитів

Було порівняно значення $RequestsMissed$ QoS-обізнаної хмарної автономної інформаційної системи (QoS-обізнаної автономіки) з технікою управління ресурсами, заснованою на QoS (неавтономною), показано на рисунку 2.8. Автономія, що усвідомлює QoS, працює краще, ніж не -

автономічне, але при 2500 робочих навантаженнях кількість пропущених запитів в обох методах однакова.

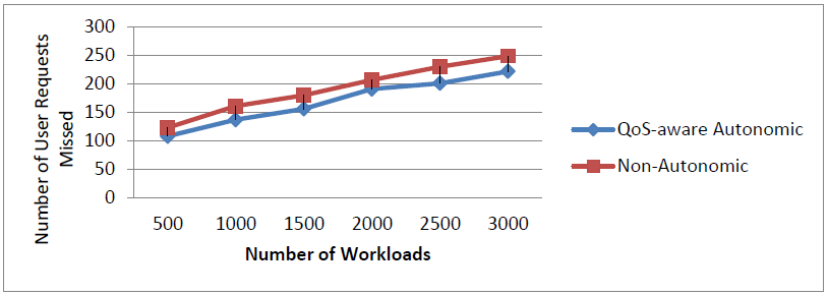


Рисунок 2.8 - Вплив зміни кількості завантажених навантажень на кількість пропущених запитів

Тестовий випадок 5: Затримка

Було обчислено відсоток задоволеності клієнтів як для хмарної автономної інформаційної системи, заснованої на QoS (автономної QoS), так і для технології управління ресурсами на основі QoS (неавтономної). Затримка визначається як різниця у часі навантаження вхідної хмари та часу виходу, виробленого відносно цього робочого навантаження.

Зі збільшенням кількості хмарних навантажень значення затримки зростає. Значення затримки в автономній системі, що усвідомлює QoS, є меншою порівняно з неавтономним плануванням ресурсів (неавтономним) при різній кількості хмарних робочих навантажень, як показано на рисунку 2.9. Максимальне значення затримки - 42,15, а мінімальне значення затримки - 22,26 секунди в QoS-обізнаному автономності.

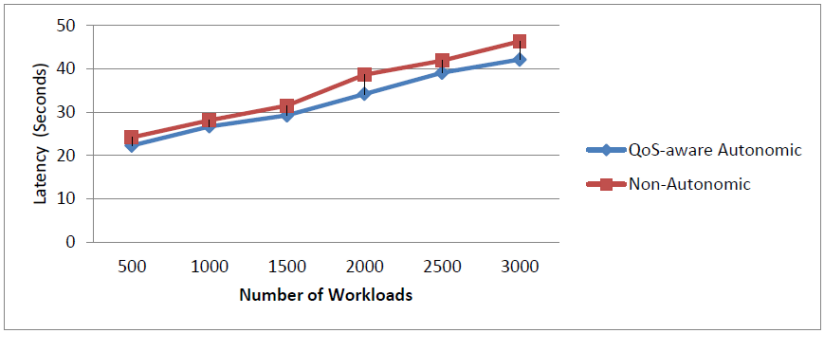


Рисунок 2.9 - Вплив зміни кількості завантажених навантажень на затримку

Тестовий випадок 6: Вартість виконання

Було обчислено значення середньої вартості як для хмарної автономної інформаційної системи, заснованої на QoS (автономної QoS), так і для управління ресурсами на основі QoS (не автономна) з різною кількістю завантажених хмарних даних, як показано на рисунку 2.10. Виконання витрат становить додавання витрат на ресурси та витрат на штраф. Agri-Info визначив різні рівні штрафу на основі вимог QoS. Час затримки - це різниця строку і часу, коли фактично завершено навантаження.

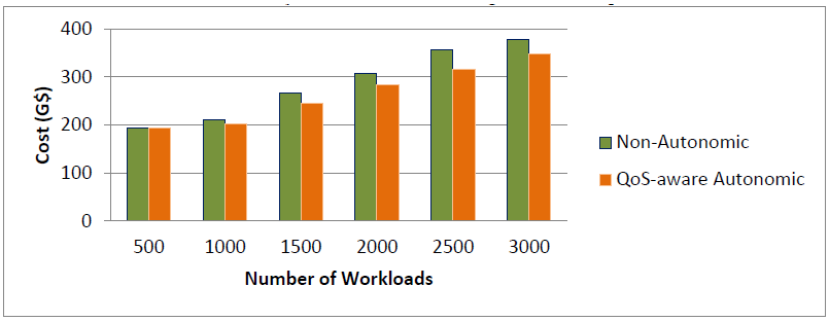


Рисунок 2.10 - Вплив зміни кількості завантажених робіт на середню вартість

Якщо c ∈ C, C - це сума штрафу з різними рівнями, визначеними в Agri-Info. Середня вартість зростає зі збільшенням кількості навантажень. При 500 робочих навантаженнях середня вартість автономної та неавтономної методики, орієнтованої на QoS, майже однакова, але автономія, що обізнана з QoS, справляється чудово при 2000-3000 робочих навантаженнях. При 2500 робочих навантаженнях середня вартість в усвідомленій QoS автономії на 16,66% менше, ніж техніка управління ресурсами, що не базується на QoS.

Тестовий випадок 7: Час виконання

Час виконання визначається як відношення різниці часу закінчення робочого навантаження (WFi) та часу початку завантаження ($WStarti$) до кількості навантажень.

Де n - кількість навантажень, які потрібно виконати. Як показано на рисунку 2.11, час виконання збільшується зі збільшенням кількості навантажень. При 1500 робочих навантаженнях час виконання роботи в

обласній автономній інформаційній системі, заснованій на QoS (QoS-обізнаний) на 21,72% менше, ніж техніка управління ресурсами на основі QoS. Після 1500 робочих навантажень час виконання різко збільшується в техніці управління ресурсами, що не базуються на QoS, але автономія, обізнана з QoS, працює краще, ніж неавтономна методика.

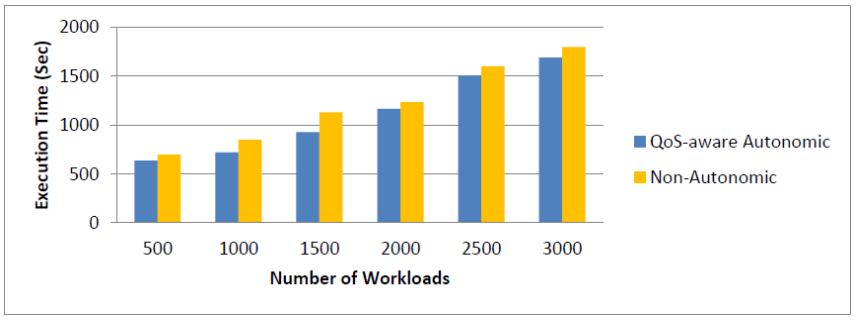


Рисунок 2.11 - Вплив часу виконання при зміні кількості робочих навантажень

Тестовий приклад 8: використання ресурсів

Використання ресурсів визначається як відношення фактичного часу, витраченого ресурсом на виконання робочого навантаження, до загального часу безвідмовної роботи ресурсу для окремого ресурсу.

Зі збільшенням кількості хмарних робочих навантажень відсоток використання ресурсів збільшується. Відсоток використання ресурсів в хмарної автономної інформаційної системи з підтримкою QoS (автономної з урахуванням QoS) більше в порівнянні з управлінням ресурсами без урахування QoS (неавтоматичним) при різній кількості хмарних робочих навантажень, як показано на рисунку 2.12. Максимум відсоток використання ресурсів становить 91,67 % при 3000 хмарних робочих навантаженнях при автономній роботі з підтримкою QoS, але автономна робота з урахуванням QoS працює краще, ніж неавтономна техніка.

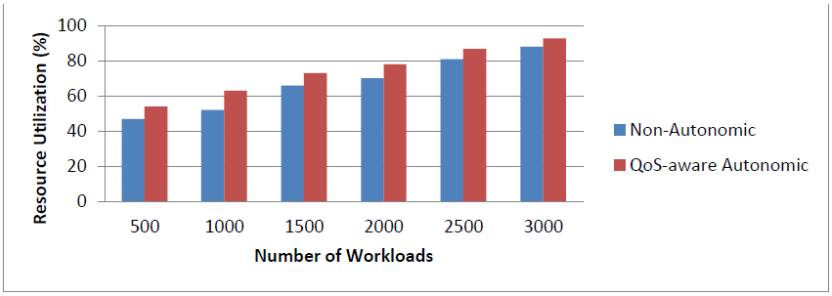


Рисунок 2.12 - Вплив зміни кількості робочих навантажень, представлених при використанні ресурсів

Тестовий приклад 9: обчислювальна потужність

Ми вирахували значення обчислювальної потужності як для хмарної автономної інформаційної системи, заснованої на QoS (автономна підтримка QoS), так і для технології управління ресурсами, яка не базується на QoS (неавтономна), з різною кількістю хмарних робочих навантажень. Обчислювальна потужність - це відношення фактичного часу використання ресурсу до очікуваного часу використання ресурсу.

Зі збільшенням кількості хмарних робочих навантажень значення обчислювальної потужності зменшується. Хмарна автономна інформаційна система з підтримкою QoS (автономна з підтримкою QoS) працює краще, ніж неавтономна диспетчеризація ресурсів при різній кількості хмарних робочих навантажень, як показано на рисунок 2.13.

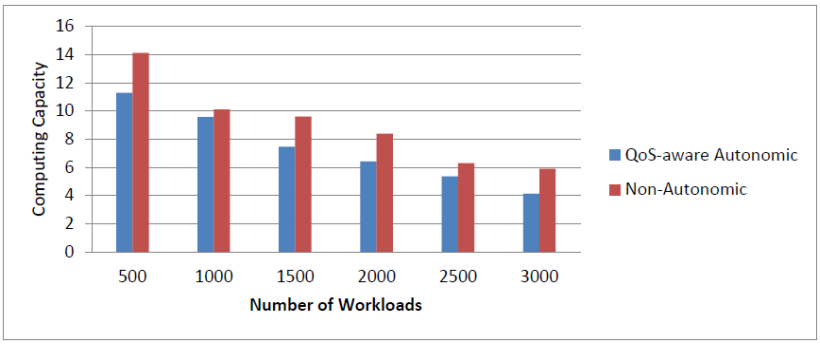


Рисунок 2.13 - Вплив зміни кількості робочих навантажень на обчислювальну потужність

Мінімальне значення обчислювальної потужності становить 4,13 при 3000 хмарних робочих навантаженнях, а максимальне значення - 11,29 при 500 робочих навантаженнях в автономній системі з підтримкою QoS.

Висновки по тестуваню

Продуктивність хмарної автономної інформаційної системи (Agri-Info) з підтримкою QoS була протестована в хмарному середовищі моделювання, а також на хмарному стенді для аналізу значень різних параметрів QoS з різною кількістю робочих навантажень і ресурсів. Продуктивність Agri-Info була оцінена з точки зору пропускної здатності мережі, вартості виконання, часу виконання, доступності, задоволеності клієнтів, обчислювальної потужності, використання ресурсів і затримки в моделюється хмарної середовищі. Результати експериментів показали, що Agri-Info виконує однакову кількість робочих навантажень при мінімальній пропускній спроможності мережі, вартості виконання, часу виконання, кількості пропущених призначених для користувача запитів і часу очікування, а також максимальному використанні ресурсів, доступності та задоволеності клієнтів. При 3000 робочих навантаженнях пропускна здатність мережі, яка використовується в Agri-Info, на 18,98% менше, ніж методика управління ресурсами, не заснована на QoS. Максимальне значення затримки становить 42,15, а мінімальне значення затримки становить 22,26 секунди в Agri-Info. Вартість виконання дозволяє оцінити вибір ресурсу, тоді як тривалість виконання робочого навантаження оцінюється за часом виконання. Agri-Info скорочує час виконання до 21,72% в порівнянні з технікою управління ресурсами, яка не базується на QoS (неавтономних), і знижує вартість виконання до 16,66% в порівнянні з неавтономні. Максимальний відсоток використання ресурсів становить 91,67% при 3000 робочих навантаженнях в Agri-Info, але він працює краще, ніж неавтономний метод.

Крім того, було проведено емпіричне дослідження, щоб продемонструвати, що техніку управління автономним управлінням ресурсами на реальних хмарних ресурсах також можлива. Експеримент проводився з різною кількістю запитів користувачів (1-70) для перевірки вартості виконання, часу виконання, використання ресурсів та затримки в

реальному хмарному середовищі. У цьому тестувані розглянули дві різні хмарні інфраструктури з різною конфігурацією процесора (4-ядерний процесор та 8-ядерний процесор) для вимірювання зміни часу виконання з різною кількістю запитів користувачів (малі, середні та великі) у вигляді запитів користувачів. Agri-Info скорочує час виконання на 33% в порівнянні з технікою управління ресурсами, що не базуються на QoS, і зменшує вартість виконання до 37,6% порівняно з неавтономною. Тестувальні результати повідомляють, що Agri-Info, використовуючи хмарну інфраструктуру з 4-ядерним процесором, працює краще, ніж Agri-Info, використовуючи хмарну інфраструктуру з 8-ядерним процесором для невеликої кількості запитів користувачів (1-20) і для середнього (21-45) та великого (46 -70) кількість запитів користувачів Agri-Info краще працює з хмарною інфраструктурою 8-ядерного процесора. З усіх експериментальних результатів, автономна інформаційна система, заснована на хмарі QoS (Agri-Info), працює краще, ніж техніка управління ресурсами, заснована на QoS, як у моделюванні, так і в реальному хмарному середовищі

3. ОРГАНІЗАЦІЙНА-ЕКОНОМІЧНА ЧАСТИНА

3.1 Загальний підхід до визначення економічної ефективності розробки

Обов’язковою складовою частиною будь-якого інженерного проекту, в тому числі софтверного, є фінансові витрати на різних етапах виконання робіт. Відповідно, важливо вірно здійснити фінансову оцінку передбачуваних витрат, продуктивність, корисність та, в результаті, економічну ефективність проекту.

Наукові розробки та дослідження, на відміну від корпоративних, не завжди супроводжують за мету отримання прибутку або ж іншої матеріальної вигоди. В багатьох випадках, проекти наукового спрямування не є економічно вигідними. Однак, вони є рушійною силою прогресу, дослідженням незвіданих галузей та проблем, які, у свою чергу, майже завжди впливають на майбутні різнопланові розробки. Тому дуже часто наукова діяльність стимулюється зовнішніми інвестиціями та підтримкою держаних інститутів, міжнародних грантів. В плані використання результатів досліджень та на основі отриманих моделей можна робити прогнози по впровадженню нових методик та принципів організації роботи діагностичних установ. Різноманітні медичні центри зможуть скористатися на практиці розробленою системою для покращення існуючих та отримання нових діагностичних методик.

Оцінка вартості дослідницьких розробок базується на витратному підході: використанні первісної вартості об'єктів, виходячи з фактичних витрат на розробку та доведення до комерційного використання з урахуванням амортизації. Так, як результати роботи у вигляді математичних моделей та реалізованого на їх основі ПЗ не буде використовуватися в комерційних цілях та не підлягатиме продажу, а становить наукову та

інтелектуальну цінність, то доходу від продажі ПЗ та розробки як такого не передбачається. Іншими словами, всі вкладені кошти та витрати на розробку даного рішення є не взаємоокупними, що несуть лише витрати у кількості залучених ресурсів та матеріальних засобів.

Згідно Статті 8 Закону № 3792-12 передбачено, що твори наукового характеру та комп'ютерні програми є об'єктами авторського права [15]. У разі реєстрації виконаної роботи як інтелектуальної власності та авторського права у державній службі інтелектуальної власності України необхідно буде сплатити збори за державну реєстрацію (382,5 грн.).

Для отримання відмінних результатів експериментів та доцільності розробки такого спеціалізованого ПЗ потрібні відповідні затрати на дослідження та розробку. Це і становитиме основу витрат, які будуть здійснені протягом підготовки та виконання реалізації даного рішення. Уявно модель витрат можна поділити на дві основні частини: витрати, пов'язанані на дослідження предметної області, побудову математичних моделей, отримання попередніх результатів експериментів, та частину реалізації програмної системи, архітектури та тестування.

Враховуючи залежність якості кінцевого продукту від кваліфікації програмістів, потрібно сконцентрувати увагу на якості та результативності розробки. Для цього потрібно провести ґрунтовний аналіз предметної області, залучити найновіші та інноваційні технології, провести ґрунтовне тестування та оцінку результатів розробки. Для забезпечення хорошої результативності при розробці доводиться йти на додаткові заходи заохочення та стимулювання у вигляді преміювання працівників, підтримання наукового дослідження досвідом іноземних науковців, дорогоцінних лабораторних дослідів.

До створення ПЗ можуть бути залучені позаштатні програмісти як зареєстровані, так і не зареєстровані підприємцями. В обох випадках співпраця з ними здійснюється на підставі цивільно-правового договору,

найчастіше – договорами підряду. Щодо оподаткування виплат за договором підряду, то все залежить від того, чи зареєстрований виконавець підприємцем. Важливим етапом розробки ПЗ є його тестування, яке виконується тестувальником за певну винагороду. Якщо тестувальники не перебувають з підприємством у трудових відносинах, оплата виконується на основі договору підряду на виконання робіт з тестування ПЗ. Сам же результат розробки не оподатковується, адже не є комерційним проектом і не спрямований на продаж.

3.2 Розрахунок вартості процесу розробки та оцінка економічної ефективності проекту

Для оцінки нематеріальних активів використовують міжнародні стандарти оцінки розрахунку вартості об’єктів інтелектуальної власності, розроблені TIAVIS (TheInternationalAssetsVabulationStandartsCommitie).

Виконання розробки програмного забезпечення з огляду економічної моделі можна виконувати двома способами: процедурним та об’єктно-орієнтованим. Обидва підходи потребують залучення ресурсів у вигляді програмісті-розробників, тестувальників, керівника проекту, наукового ресурсу. Різниця виникає в самій схемі розробки, тривалості періоду розробки та відповідній вартості. Процедурний підхід для розробки ПЗ в основі якого лежать процедури і функції передбачає розробку ПЗ як монолітного композиту, що в подальшому, як правило, вимагає великих витрат на супровід та модернізацію. Об’єктно-орієнтований підхід, що ґрунтується на основі об’єктів певних класів, що описують певну область, описують певну поведінку (методи) та володіють властивостями (атрибутами), орієнтовані на варіанти використання та покроковий процес розробки [16].

Для початку робіт необхідно скласти технічне завдання на розробку, яке є основним документом, що регламентує подальшу роботу, та містить докладний опис необхідних функцій програми, інтерфейс, технології, інше. Вартість складання технічного завдання переважно складає до 10% від планованої вартості розробки. Роботу зі складання технічного завдання веде керівник проекту разом із програмістами та консультуючись із замовником.

Усі програмісти, що працюють у штаті підприємства-розробника мають встановлено певний посадовий оклад. Місячний оклад, денна заробітна плата, трудомісткість (днів) і основна заробітна плата кожного учасника техпроцесу представлено у таблиці 3.1. Всі суми наведені в національній валюті – в гривні.

Таблиця 3.1 – Розрахункова вартість технологічного процесу розробки

Посада	Місячний оклад, грн.	Денна зар. плата, грн.	Об'єктно-орієнтований підхід		Процедурний підхід	
			Днів	Сума, грн.	Днів	Сума, грн.
Керівник	12100,00	550,00	15	8250,00	16	8800,00
Програміст	11660,00	530,00	21	11130,00	25	13250 ,00
Тестувальник	9680,00	440,00	7	3080,00	7	3080,00
Бізнес аналітик	10450,00	475,00	10	4750,00	10	4750,00
Додаткова зар. плата 20%			53	5442,00	58	5976,00
Фонд оплати праці 36,77%			10005,117		10986,876	
Всього витрат на зар. плату			42657,12		46842,88	
Військовий збір 1,5%			639,86		702,64	
Єдиний соціальний внесок 3.6%			1535,66		1686,34	
ПДВ, 15%			6398,57		7026,43	
Всього			**51222,21**		**54571,95**	

Згідно вимог та прорахованої кількості необхідних ресурсів на виконання, розробку, тестування та дослідницьку роботу було отримано

основні часові рамки роботи над проектом. Так для об’єктно-орієнтованого підходу загальна тривалість роботи над ПЗ становить 53 робочих днів (під робочим днем розуміється 8-ми годинний робочий день), що включає роботу програміста, який, в свою чергу, являється і керівником розробки, роботу тестувальника та наукового працівника. Сума витрат на заробітню плату становить 51222,21 гривень включаючи всі види додаткових оплат. Для процедурного підходу до розробки суми дещо більші, адже затрачається більше часу на розробку. Так, при використанні процедурного підходу сумарна тривалість часу розробки становить 58 робочі дні, та витрати у вигляді виплат заробітної плати становлять 54571,95 гривень.

Витрати на науково-дослідницьку роботу та здійснення розробки програмних продуктів і об’єктно-орієнтованим, і процедурним способом включають:

Основна заробітна плата:

$ЗП_{осн\ 1} = 27210$ грн; $ЗП_{осн\ 2} = 29880$ грн.

Додаткова заробітна плата обчислюється як $ЗП_{дод} = 0,2 \cdot ЗП_{осн}$.

$ЗП_{дод\ 1} = 0,2 \cdot 27210 = 5442$ грн; $ЗП_{дод\ 2} = 0,2 \cdot 29880 = 5976$ грн.

Нарахування на фонд оплати праці (ФОП):

$ФОП_{ЄСВ} = 0,3677 \cdot ФЗП$

$ФОП_{ЄСВ1} = 0,3677 \cdot 32652 = 10005,117$ грн;

$ФОП_{ЄСВ1} = 0,3677 \cdot 35856 = 10986,876$ грн.

Всього витрат:

$В_{ЗП1} = ЗП_1 + ФОП_{ЄСВ1} + ЗП_{дод1} = 51222,21$ грн;

$В_{ЗП2} = ЗП_2 + ФОП_{ЄСВ2} + ЗП_{дод2} = 37130,97$ грн.

З цієї суми утримуються обов’язкові відрахування на заробітну плату: єдиний соціальний внесок, який складає 3,6% від суми нарахованої заробітної плати та податок на доходи фізичних осіб, який складає 15% від суми нарахованої заробітної плати, зменшеної на суму єдиного внеску на

загальнообов'язкове соціальне страхування та податкової соціальної пільги, військовий збір у розмірі 1,5%, від суми нарахувань.

До окремих витрат також відносяться витрати на куповані вироби (матеріальне забезпечення) та спец обладнання для підтримки експерименту, накладні витрати. Витрати, що будуть супроводжувати проект розробки, порівнюватимемо в двох можливих підходах розробки.

Матеріальні витрати визначаються як добуток кількості витрачених матеріалів та їх ціни (формула 3.1).

$$M_{Bi} = q_i \cdot p_i \,, \tag{3.1}$$

де q_i– кількість витраченого матеріалу i-го виду; p_i– ціна матеріалу i-го виду.

Матеріальні витрати в рамках проекту наведені в таблиці 3.2. Загальна сума матеріальних витрат становить 1535 гривень.

Таблиця 3.2 – Матеріальні витрати

Найменування ресурсу	Кількість, шт.	Ціна одиниці, грн	Загальна сума, грн
Флешки	4	227,00	908,00
Папір для друку А4, арк	500	0,158	79,00
Тонер для принтера	1	60,00	60,00
Дошка для записів	1	450,00	450,00
Перманентний маркер	4	12,00	48,00
Всього			1535,00

Розрахунок витрат на електроенергію одиниці обладнання визначаються за формулою:

$$З_в = W * T * S, \tag{3.2}$$

де W – необхідна потужність, кВт; T – кількість годин роботи обладнання; S – вартість кіловат-години електроенергії, S = 2,50 грн/кВт·год.

$З_{в1}$ =0.7 * 424 * 2.50 = 742 грн;

$З_{в2}$ =0.7 * 464 * 2.50 = 812 грн;

Розрахунок суми амортизаційних відрахувань. Комп'ютери та оргтехніка належать до четвертої групи основних фондів. Для цієї групи річна норма амортизації дорівнює 60 %, вартість яких перевищує 1000 грн. і визначається:

$$A = \frac{C_Б \cdot N_A \cdot T_{ФАК}}{T_{ГОД}} \quad (3.3)$$

де $C_Б$ – балансова вартість обладнання, грн; N_A – норма амортизаційних відрахувань в рік, %; $T_{ФАК}$ – фактичний час роботи обладнання по написанню програми, год; $T_{ГОД}$ – річний робочий фонд часу, год.

У даній формулі норма відрахувань на амортизацію рівна N_A = 0,6. Балансова вартість обладнання вказана в таблиці 3.3 і рівна $C_Б$ = 22589 гривень. Річний робочий фонд часу приймемо за $T_{ГОД}$ = 2120 годин. З них реальний фактичний робочий час становить $T_{ФАК}$ = 424 години згідно об'єктно-орієнтованого підходу та $T_{ФАК}$ = 464 годин згідно процедурного підходу.

Згідно вищезгаданої формули витрати на амортизацію становлять 1943,5 гривень та 2135,3 гривень для кожного підходу відповідно.

Таблиця 3.3 – Перелік необхідного обладнання

Найменування	Кількість, шт	Ціна, грн	Сума, грн	
Комп'ютер	2	9545,00	19090,00	
Принтер	1	3499,00	3499,00	
Середовища розробки	2	безкоштовно	Безкоштовно	
Операційна система (Linux)	2	безкоштовно	Безкоштовно	
Всього більше 1000 грн.			22589,00	
Всього витрат на амортизацію			1943,5	2135,3
Всього			24532,5	24724,3

Накладні витрати пов’язані з обслуговуванням виробництва, утриманням апарату управління та створення необхідних умов праці та закупівлю ресурсів та обладнання для розробки наведені в таблиці 3.3.

Залежно від організаційно-правової форми діяльності господарюючого суб’єкта, накладні витрати можуть становити 20–60 % від суми основної та додаткової заробітної плати працівників. Нехай вона буде дорівнювати 30%, що становить 9795,6 грн для об’єктно-орієнтованого і 10756,3 грн для процедурного підходу розробки.

Проведемо розрахунок вартості створюваного програмного продукту. Вартість продукції включає у собі собівартість і планований прибуток. Найважливішим моментом для розробника, з економічної точки зору, є процес встановлення ціни.

Можна отримати загальні значення витрат на розробку та реалізацію проекту, враховуючи всі вище описані затрати та нарахування. Цей вид витрат складається з сум витрат на оплату праці (всього витрати на оплату праці), матеріальні затрати, затрати на електроенергію, накладні витрати, витрати на обладнання, враховуючи амортизації обладнання на час виконання проекту.

Собівартість продукції – це сума грошових витрат підприємства (фірми) на виробництво і збут одиниці продукції, виконання робіт та надання послуг.[17]

Повна собівартість програмного продукту дорівнює сумі усіх витрат на його виробництво: 87613,05 грн використовуючи об’єктно-орієнтований підхід, 92170,85 грн при процедурному підході розробки.

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Економічна ефективність (Е) полягає у відношенні результату виробництва до затрачених ресурсів:

$$E = \frac{П}{Св} \tag{3.4}$$

де *П* – прибуток, *П= В - Св; Св* – собівартість.

У випадку даної розробки, маючи некомерційний проект без економічно корисного результату, можна прогнозувати, що економічна ефективність прямує до 0 у обох випадках. Однак це не є причиною для негативного економічного висновку щодо даного проекту, адже такого плану розробки приносять користь у вигляді інтелектуальних ресурсів, і, переважно, фінансуються або виконуються на замовлення організацій, зацікавлених в отриманні результатів досліджень та розробок. Фінансування не можна вважати отриманим доходом від реалізації. Однак за надходження коштів на реалізацію ззовні можна вважати ефективність проекту рівною 1 (E = 1), що означає перекриття витрат на розробку у повній мірі, тобто фінансування.

Якщо ринкова вартість програмного продукту рівна прийнятій, то економічна ефективність визначається встановленим рівнем прибутку. Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень ($T_{ок}$):

$$T_{ок} = \frac{1}{E} \tag{3.5}$$

У нашому випадку прямого прибутку не існує. Прибуток можна прогнозувати на підприємствах, організаціях чи відомствах, що зацікавлені в дослідженні. Окупність же для розробки даного ПЗ за ефективності рівній одиниці можна вважати теж рівній 1 згідно формули 3.4.

Виходячи із експертних оцінок і складності програми, приймемо величину витрат на супровід і модернізацію програмного забезпечення,

створеного за об’єктно-орієнтованим методом 25% (21903,26 грн) від початкових витрат, а за процедурним – 30% (27651,26 грн).

Однак варто зауважити, що розробка спрямована на короткотривалу підтримку та не прогнозує модернізації. У разі ж необхідності розробки такого ж або суміжного ПЗ можна вважати за доцільно розпочинати розробку з початкових етапів, що потягне за собою нові витрати у повній мірі. Тобто у разі короткотермінової підтримки все ж за доцільніше обирати об’єктно-орієнтовану модель розробки, адже вартість короткотермінової підтримки згідно цієї моделі не вплине значно на сукупну вартість розробки.

Здана в експлуатацію система не завжди цілком завершена, її треба змінювати протягом терміну експлуатації. Внаслідок змін система стає більш складною і погано керованою. Об’єктно-орієнтоване представлення програми дозволяє навіть середньому програмісту швидко і ефективно супроводжувати і модернізувати програми, що значно скорочує подальші витрати на супровід і модернізацію [17].

Сумарні дані економічного розрахунку розробки даного проекту наведені в таблиці 3.4.

Таблиця 3.4 – Загальні витрати

Вид витрат	Об’єктно-орієнтований підхід, грн	Процедурний підхід, грн
Зарплата основна	27210,00	29880,00
Зарплата додаткова	5442,00	5976,00
Фонд заробітної плати	32652,00	35856,00
Відрахування на ФОП	10005,12	10986,88
Разом на оплату праці	51222,21	54571,95
Матеріальні витрати	1535,00	1535,00
Електроенергія	742	812
Амортизація	1943,50	2135,30

Накладні витрати	9795,60	10756,30
Обладнання	22589,00	22589,00
Разом на ін. витрати	36390,84	37598,9
Собівартість	87613,05	92170,85
Прибуток	Відсутній	Відсутній
Вартість розробленого ПЗ	**87613,05**	**92170,85**
Модернізація і супровід	21903,26	27651,26
Загальні витрати на розробку	109516,30	119822,11
Економія ($ЗВ_1$-$ЗВ_2$)	**10305,81**	

Загальна вартість пропонованих робіт становить 119822,11 гривень для процедурного і 109516,30 гривень для об’єктно-орієнтованого підходів розробки. В даному випадку реалізації проекту варто вибрати об’єктно орієнтований підхід для розробки даного ПЗ, адже фінансово це більш вигідно. Також у даній методиці розробки кращі часові рамки та перспективи підтримки і модернізації. У оцінці вартості продукту варто враховувати можливість фінансування та спонсорування проекту, що дозволить гнучко та ефективно підійти до розробки та організації праці.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Тема магістерської роботи пов’язана із розробкою інформаційної системи аграрного підприємства. Приміщення, в якому проводилася розробка, маючи площу 30м^2 та об’єм 92м^3, містить 4 комп’ютеризовані робочі місця. Для зберігання документів використовуються дві шафи. Приміщення містить два вікна та одні двері, які розташовані по середині, між робочими місцями.

Розглянемо відповідність характеристик комп’ютеризованого робочого місця розробника нормативним. Площа, на якій розташовується одне робоче місце з ВДТ, повинна становити не менше 6,0 кв м. В нашому випадку площа становить 7,5 кв м.

При розміщенні робочих столів з персональними комп’ютерами слід дотримувати:

- відстань між бічними поверхнями персональних комп’ютерів 1,2 м;
- відстань від тильної поверхні одного персонального комп’ютера до екрана іншого – 2,5 м;

- робочі місця з ВДТ розміщуються на відстані не менше 1 м від стіни зі світловими прорізами (вікнами);
- прохід між рядами робочих місць має бути не меншим за 1 м.

В даному випадку робочі місця розміщено так, що в поле зору робітника не може попасти екранна сторона дисплея, окрім його власного. Також відстань між бічними поверхнями, відстань від стін із світловими прорізами та проходи між рядами задовольняють нормам.

Рекомендовані розміри столу для робочого місця з ВДТ становлять: висота – 725 мм, ширина – 600-1400 мм, глибина 800-1000 мм. В даному приміщенні використовуються робочі столи з розмірами: ширина – 1200 мм, глибина – 800 мм.

Комп'ютеризовані робочі місця розміщені рядами вздовж стіни з вікнами. Це дає змогу виключити дзеркальне відбиття на екрані ВДТ джерел природного світла (вікон) та потрапляння останніх у поле зору операторів, що погіршує їх зорову роботу.

Відповідно до ДСН 3.3.6.042-99 роботи, що виконуються користувачами ЕОМ, відносяться до легких фізичних робіт – категорії Іа. У виробничих приміщеннях на робочих місцях з ВДТ мають забезпечуватись оптимальні значення параметрів мікроклімату.

Згідно НПАОП 0.00-7.15-18 приміщення, що розглядається, повинне мати природне і штучне освітлення.

Природне освітлення приміщення відбувається за системою однобічного бічного освітлення. Природне світло проникає у приміщення через два світлові віконні отвори, які мають регулювальні пристрої для відкривання. Також наявні штори (жалюзі) з можливістю захисту працюючих від прямого попадання сонячних променів і регулювання рівня освітленості в приміщенні. Вікна приміщення орієнтовані на північний схід. Оскільки будинок розташований у відносній віддаленості від прилеглих будівель, то які-небудь перешкоди природному освітленню розглянутого приміщення

відсутні. Всередині приміщення стіни обклеєні світлими шпалерами, стеля побілена, у якості підлогового покриття використаний лінолеум світло-жовтого кольору.

У приміщенні присутні внутрішні джерела постійного шуму – вентилятори блоків ЕОМ, дисководи. Зовнішніми джерелами шуму і вібрації в приміщенні є проїжджаючі транспортні засоби. Шум погіршує умови праці здійснюючи шкідливу дію на організм людини. Працюючі в умовах тривалої шумової дії випробовують дратівливість, головні болі, запаморочення, зниження пам'яті, підвищену стомлюваність, зниження апетиту, болі у вухах і т. Такі порушення в роботі ряду органів і систем організму людини можуть викликати негативні зміни в емоційному стані людини аж до стресових ситуацій. Під впливом шуму знижується концентрація уваги, порушуються фізіологічні функції, з'являється втомленість у зв'язку з підвищеними енергетичними витратами і нервово-психічною напругою, погіршується мовна комутація. Все це знижує працездатність людини і її продуктивність, якість і безпеку праці. Однак фактичний обмірюваний рівень шуму в робочій зоні склав 37 дБА, що задовольняє нормативному рівню шуму (не повинен перевищувати 50 дБА згідно з ДСН 3.3.6.037- 99).

Джерелом електромагнітного випромінювання в сучасному офісі є візуальні дисплейні термінали. Нормування електромагнітного випромінювання ВДТ здійснюється згідно положень ДСанПіН 3.3.2-007-98.

Приміщення відноситься до зони П-IIа згідно з ДНАОП 0.00-1.31-99 до категорії пожежної небезпеки В. Горючі рідини, пил та волокна у приміщенні не використовуються і не виділяються.

Ймовірними причинами виникнення пожежі можуть бути несправність електрообладнання, короткі замикання внаслідок виходу з ладу електроустаткування, порушення правил протипожежної безпеки тощо.

Для своєчасного попередження пожеж та підвищення оперативності реагування у приміщенні використовується такий комплекс заходів:

- обов’язковий інструктаж персоналу з питань охорони праці;
- зокрема, правила пожежної безпеки у приміщеннях з ЕОМ;
- заборона використання відкритого вогню у приміщенні;
- наявність системи автоматичної пожежної сигналізації з димовими пожежними сповіщувачами;
- наявність шляхів евакуації при виникненні пожежі;
- розміщення схеми евакуації людей при пожежі і ознайомлення з нею персоналу.

В даному розділі було проаналізовано основні вимоги, які ставляться до приміщення в цілому, робочого місця користувача комп’ютером, освітлення, шуму, електромагнітних випромінювань, пожежної безпеки.

Можна стверджувати, що при розробці даної програмної системи було дотримано усіх необхідних вимог та норм охорони праці.

4.2 Підвищення стійкості роботи підприємства аграрного комплексу в воєнний час

Стійкість роботи об'єкта — це здатність його в надзвичайних ситуаціях випускати продукцію у запланованому обсязі, необхідної номенклатури і відповідної якості, а у випадку впливу на об'єкт уражаючих факторів, стихійних лих та виробничих аварій — у мінімально короткі строки відновити своє виробництво.

На основі вивчення факторів, які впливають на стійкість роботи об'єктів, і оцінки стійкості елементів і галузей виробництва проти уражаючих факторів ядерної, хімічної і біологічної зброї, стихійних лих і виробничих аварій, необхідно завчасно організувати і провести організаційні, інженерно-технічні й технологічні заходи для підвищення стійкості роботи.

Здійснення організаційних заходів передбачає завчасну підготовку всіх структур цивільного захисту, служб і формувань до надзвичайних ситуацій.

Вжиттям технологічних заходів підвищується стійкість роботи об'єктів шляхом змінювання технологічних процесів, режимів, можливих в умовах надзвичайних ситуацій. Інженерно-технічні заходи мають забезпечити підвищену стійкість виробничих споруд, технологічних ліній, устаткування, комунікацій об'єкта до впливу уражаючих факторів під час надзвичайних ситуацій.

При проведенні цих заходів необхідно враховувати конкретні умови об'єкта народного господарства. Проте є загальні організаційні інженерно-технічні заходи, які мають проводитись на всіх об'єктах.

Стійкість роботи сільськогосподарського підприємства у воєнний час – його здатність виробляти плановий об’єм продукції, а у випадку ушкоджень, викликаних дією уражальних чинників воєнного часу, – відновлювати виробництво у стислий термін. Основними шляхами забезпечення стійкої роботи сільськогосподарського підприємства у воєнному стані є:

1. Забезпечення захисту людей та їх життєдіяльності. Створення на об'єкті надійної системи оповіщення про загрозу нападу противника, радіоактивне забруднення, хімічне і біологічне зараження, загрозу стихійного лиха і виробничої аварії. Організація розвідки і спостереження за радіоактивним забрудненням, хімічним і біологічним зараженням; гідрометеорологічне спостереження за рівнем води, напрямком і швидкістю вітру, рухом і поширенням хмари радіоактивного забруднення.

2. Захист цінного й унікального устаткування. Захистити цінне і унікальне устаткування можна завдяки проведенню інженерно-технічних заходів, щоб зменшити небезпеку пошкодження і руйнування цінного й унікального устаткування, станків з програмним керуванням, шліфувальних, токарних, розточних, зубофрезер-них, пресових станків, автоматичних конвеєрних ліній та іншого устаткування. Варіантами такого захисту є розміщення зазначеного устаткування в заглиблених приміщеннях а також використання спеціальних захисних пристосувань, закріплення станків на

фундаментах, застосування контрфорсів для підвищення стійкості проти перекидання обладнання.

3. Стійкість роботи галузі рослинництва. Планування і проведення заходів захисту сільськогосподарських рослин, урожаю в різних надзвичайних ситуаціях. Встановлення надійної взаємодії зі станцією захисту рослин, радіологічною і агрохімічною лабораторією для організації спостереження за зараженістю посівів сільськогосподарських культур та ґрунтів, відбір необхідних проб та їх аналіз. Впровадження у виробництво високо урожайних, стійких проти небезпечних хвороб і шкідників сільськогосподарських культур. Підготовка техніки і хімічних засобів захисту сільськогосподарських культур від біологічних засобів ураження. Розробка заходів збирання урожаю в умовах обмеженості забезпечення людьми, технікою, паливом і мастилами, порушення міжгалузевих зв'язків, технології доведення урожаю до кондиції.

4. Стійкість роботи тваринництва. Підготовка до проведення ветеринарно санітарних заходів, спрямованих на зниження втрат тварин від сучасних засобів ураження. Завчасна підготовка приміщень для утримання тварин. Розробка заходів захисту тварин на пасовищах. Створення запасів кормів і організація забезпечення водою. Організація ветеринарної розвідки в господарстві, відбір необхідних проб та їх аналіз. Розробка заходів евакуації тварин із зон можливих руйнувань, катастрофічного затоплення, районів хімічного зараження, підготовка місць для евакуації тварин. Підготовка до постійної готовності спеціальної техніки для обробки тварин, а також пристосування для цієї мети іншої техніки, наявної в господарстві.

5. Підвищення стійкості мереж комунального господарства. Для забезпечення стійкості роботи об'єктів повинні проводитись інженерно-технічні заходи на мережах комунального господарства з метою захисту джерел тепла із заглибленням у ґрунт комунікацій. Котельні слід розміщувати в спеціальному окремо розміщеному приміщенні. Теплова

мережа має будуватися за кільцевою системою з прокладанням труб у спеціальних каналах зі з'єднанням паралельних ділянок. Для відключення пошкоджених ділянок мають бути встановлені запірно-регулючі засувки, вентилі та ін. Ці пристосування необхідно розміщувати в оглядових колодязях, на території, що не завалюється при руйнуванні будівель.

6. Забезпечення стійкості роботи паливно-енергетичного комплексу і водопостачання. Створення резерву енергетичних потужностей за рахунок автономних пересувних електростанцій, а також місцевих джерел електроенергії. Підготовка автономних електростанцій до роботи за спеціальним режимом (графіком) для забезпечення технологічних процесів виробництва, для яких неможливі тривалі перерви в електропостачанні. З метою попередження аварій на електричних мережах необхідно установити автоматичну систему відключення при виникненні перенапруги. Повітряні лінії електропостачання замінити на підземно-кабельні. Створення необхідних запасів (резервів) паливно-мастильних матеріалів та інших видів палива й організація їх безпечного зберігання. Щоб не допустити зупинки підприємства через дефіцит палива, необхідно підготуватись для роботи на різних видах палива: нафта, вугілля, газ.

Для підвищення стійкості забезпечення водою слід провести такі заходи. Необхідно створити основні і резервні джерела водопостачання. Як резервне джерело краще мати артезіанську свердловину, яку необхідно підключити до системи водопостачання. Крім того, воду можна брати з близько розміщеної природної водойми або спорудити штучну водойму чи резервуари з обладнанням пристроїв для збору і перекачування води. Всі ділянки водопостачання повинні бути заглиблені в ґрунт з обладнанням пожежних гідрантів і пристроїв для відключення пошкоджених ділянок. Локальні мережі водопостачання окремих великих підприємств варто з'єднати із загальноміською системою водопостачання в єдине кільце.

7. Стійкість роботи автотранспортної та іншої техніки, технологічного обладнання і механізмів. Організація своєчасного оповіщення гаража, технологічного парку, їх керівників, водіїв, механізаторів про загрозу надзвичайної ситуації. Підготовка автотранспортної техніки до проведення робіт в умовах радіоактивного забруднення, хімічного біологічного зараження і світломаскування. Пристосування і використання всіх видів транспортних засобів для евакуації населення і перевезення потерпілих. Розробка заходів з метою пристосування автотранспортної, іншої техніки для виконання завдань. Розробка пристосувань і технологічних процесів для відбору потужностей тракторів і автомобілів з метою приведення в дію електрогенераторів і технологічного обладнання, насосів для подачі води до місця споживання зі свердловин, відкритих водойм і шахтних колодязів. Підготовка всієї техніки для проведення рятувальних та інших невідкладних робіт у надзвичайних умовах мирного і воєнного часу.

8. Розробка пристосувань і технологічних процесів для відбору потужностей тракторів і автомобілів з метою приведення в дію електрогенераторів і технологічного обладнання, насосів для подачі води до місця споживання зі свердловин, відкритих водойм і шахтних колодязів. Підготовка всієї техніки для проведення рятувальних та інших невідкладних робіт у надзвичайних умовах мирного і воєнного часу.

9. Забезпечення стійкого постачання об'єкта. Для забезпечення виробництва продукції необхідні електроенергія, паливо, мастила, засоби захисту рослин, міндобрива, профілактичні й лікувальні препарати ветеринарної медицини, запасні частини, сировина та інші матеріально-технічні засоби. Забезпечення об'єктів цими ресурсами дасть можливість випускати необхідну продукцію в надзвичайних умовах мирного і воєнного часу. Тому повинні проводитись такі заходи, які б забезпечили стійкість постачання і сприяли підвищенню захисту мережі електро-, водо-, газопостачання, транспортних комунікацій і джерел постачання всім

необхідним для забезпечення функціонування галузей сільського господарства в надзвичайних умовах. З метою попередження аварій на електричних мережах необхідно встановити автоматичну систему відключення перенапруги. Повітряні лінії електропостачання слід замінити на підземно-кабельні.

10. Забезпечення збереження й відновлення будівель і споруд. Оцінка можливих ступенів руйнування будівель і споруд господарства, населеного пункту. Визначення обсягу невідкладних ремонтних робіт, потреби в будівельних матеріалах. Розрахунок сил і засобів для проведення невідкладних ремонтних та інших робіт, а також знезаражування приміщень, виробничих ділянок і території. Створення і підготовка спеціальних формувань для ремонтно-відновних, будівельних та інших робіт на об'єкті.

11. Забезпечення надійності системи управління і зв'язку. Організація захищеного пункту управління, оснащення його засобами зв'язку, які б дали можливість швидко доводити сигнали ЦЗ до всіх виробничих підрозділів і населення у місцях проживання. Розробка документів, які регламентують чіткі дії персоналу для забезпечення сталої роботи об'єкта в надзвичайних умовах. Підготовка необхідного резерву кадрів спеціалістів, механізаторів і керівних працівників для зміни тим, які будуть мобілізовані.

Висновок

В ході виконання даної магістерської роботи було розроблено інформаційну систему аграрного підприємства з використанням мови C# та технології .Net.

В дані роботі було визначено засоби розробки, а саме було використано Asp.Net для розробки серверної частини і Windows Presentation Foundation Windows form разом з графічною бібліотекою Silverlight для розробки клієнтської частини.

Аналіз отриманих значень показників ефективності для даного продукту показує, що в цілому він досить надійний і прибутковий, і має всі шанси стати одним з програмних продуктів в даній області застосування.

Узагальнюючим показником, що характеризує діяльність підприємства, є прибуток. Якщо проект здатний приносити прибуток, то його на даному етапі можна вважати успішним.

Виходячи з усього вищесказаного пропонований проект автоматизованої інформаційної системи для сільськогогосподаоських підприємств можна вважати ефективним, а його розробку вигідною.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sukhpal Singh and Inderveer Chana, “QoS-aware Autonomic Resource Management in Cloud Computing: A Systematic Review”,“ACM Computing Surveys” , Volume 48, Issue 6, pp. 1-39, 2015

2. Rajkumar Buyya, Rodrigo N. Calheiros, and Xiaorong Li. “Autonomic cloud computing: Open challenges and architectural elements”, In Proceeding of the Third International Conference on Emerging Applications of Information Technology (EAIT). (2012), 3-10, IEEE.

3. ScienceDirect [Електронний ресурс]– Режим доступу: https://www.sciencedirect.com/science/article/pii/S0168169912002219?via%3Dihub

4. Raimo Nikkilä, Ilkka Seilonen, and Kari Koskinen, “Software architecture for farm management information systems in precision agriculture”, Computers and Electronics in Agriculture, 70(2) (2010): 328-336

5. unified-am [Електронний ресурс]– Режим доступу: http://www.unified-am.com/UAM/UAM/guidances/guidelines/uam_logical_role_A100F4F1.html

6. Вікіпедія [Електронний ресурс]– Режим доступу: https://ru.wikipedia.org/wiki/UML

7. Архітектура системи (проекту) [Електронний ресурс]– Режим доступу: http://lib.mdpu.org.ua/e-book/vstup/L6.htm

8. Савельева, Е. А. Методические материалы к курсу «Технологии распределенных систем» / Е. А. Савельева. – Алматы: Научно- издательский центр КазНТУ, 2010. – 90 с.

9. Technology, TechnoSpirituality, Inspiration [Електронний ресурс]– Режим доступу: https://technospirituality.com/2019/05/computer-architecture-primer/

10. Guru99 [Електронний ресурс]– Режим доступу: https://www.guru99.com/n-tier-architecture-system-concepts-tips.html

11. Веб-служба [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/Веб-служба.

12. .NET Framework [Електронний ресурс] – Режим доступу: 88 https://ru.wikipedia.org/wiki/.NET_Framework#.D0.90.D1.80.D1.85.D0.B8.D1.82.D0.B5.D0.BA.D1.82.D1.83.D1.80.D0.B0_.NET.

13. Торстейсон П. Архитектура .NET и программирование на Visual C++ / П. Торстейсон, Р. Оберг. – Москва: Издательский дом «Вильямс», 2002. - 656 с.

14. Албахари Дж. C# 5.0. Справочник. Полное описание языка / Дж.Албахари, Б. Албахари. – Москва: Издательский дом «Вильямс», 2002. – 1008 с.

15. Законодавство України [Електронний ресурс] – Режим доступу: https://zakon.rada.gov.ua/laws/show/3792-12

16. Tutorialspoint [Електронний ресурс] – Режим доступу: https://www.tutorialspoint.com/system_analysis_and_design/system_analysis_and_design_object_oriented_approach.htm

17. Економіка підприємства, навчальний посібник / В. С. Рижиков, В. А. Панков, В. В. Ровенська, Є. О. Підгора – Київ, видавничий дім слово 2004 р., с 257

Додатки

Додаток А

Міністерство освіти і науки України

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп’ютерно-інформаційних систем і програмної інженерії

Кафедра програмної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедрою

програмної інженерії

докт. фіз.-мат. наук, професор Петрик М. Р.

“___ “_____ 201_ р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської роботи

«Розробка інформаційної системи аграрного підприємства з використанням мови C# та технології .Net»

Миколаєвич Олег Романович СПм-62

Керівник роботи

канд. фіз.-мат. наук, доцент Бойко І. В.

“___” _____ 201_р.

_____

Виконавець

студент групи СПм-62

Миколаєвич Олег Романович

“___” _____ 201_р.

_____

м. Тернопіль – 2019

Зміст

1 Підстави для розробки

Розробка проводиться у відповідності до графіку навчального плану на 2019 рік , та згідно наказу на розробку дипломної роботи.

Тема роботи: «Розробка інформаційної системи аграрного підприємства з використанням мови C# та технології.Net».

2 Призначення розробки

Для ефективного управління сільськогосподарського підприємства потрібно швидко отримати актуальну інформацію про процеси які виконуються на підприємстві.

Метою виконання дипломної роботи є система, яка дозволить ефективно управляти фермою, вести облік на підприємстві, перегляд робочого процесу, відображення розташування техніки .

Для реалізації серверної частини буде використано платформу .NET Core 2.0 та каркас ASP.NET, який використовує сучасні підходи для реалізації архітектури системи, що дозволить зменшити витрати та спростити її супровід. Для клієнтської частини – Windows Presentation Foundation (WPF) разом з графічною бібліотекою Silverlight.

3 Вимоги до програмного продукту

3.1 Функціональні характеристики

Програмний продукт має забезпечити виконання наступних дій:

а) авторизація в системі:

1) створення облікового запису;
2) вхід зареєстрованих користувачів;

б) користувацька частина:

1) додавання та видалення інформації;
2) ведення обліку на складах;
3) обмін інформації між підприємствами;
4) відображення інформації з IoT;
5) публікація постів та організація подій;
6) відображення інформації про процеси на підпідприємстві;

3.2 Склад та параметри технічних засобів

Планується розміщення компонентів системи у хмарній платформі Azure. Для серверної складової можна використати базовий план другої категорії, який включає:

- 2 ядра процесора;
- 3 ГБ оперативної пам’яті;
- 10 ГБ дискового простору.

Для клієнтської частини буде достатньо базового плану першої категорії, який включає:

- 1 ядро процесора;
- 1.75 ГБ оперативної пам’яті;
- 10 ГБ дискового простору.

3.3 Інформаційна та програмна сполучність

Програмний продукт повинен коректно функціонувати в різних операційних системах . Програмний продукт має назву «Agri-Info» і для серверної частини повинен бути застосований каркас ASP.NET Core 2.0 на основі мови програмування C# та з використанням не реляційної бази даних. Клієнтська частина повинна бути реалізована на основі каркасу Windows

Presentation Foundation (WPF) разом з графічною бібліотекою Silverlight та відобразити усі функціональні вимоги у зручному користувацькому інтерфейсі.

4 Стадії розробки

- аналіз предметної області;
- пошук акторів та варіантів використання;
- вибір та проектування бази даних;
- встановлення необхідного ПЗ;
- розробка серверної частини;
- розробка клієнтської частини;
- розгортання програмної системи;
- тестування програмної системи;
- оформлення супровідної документації;
- здача проекту.

5 Програмна документація

Для програмного продукту повинні бути розроблені наступні документи:

- Технічне завдання;
- Пояснювальна записка.

6 Порядок контролю та приймання

Розроблений програмний продукт має виконувати всі вимоги, що складаються з перерахованих у п. 3.1 характеристик.

Приймання проводиться спеціально створеною комісією в термін до “__” _____ 2019 р.

Додаток Б

Матеріали наукової конференції Тернопільського національного технічного університету імені Івана Пулюя. Тернопіль 2019.

УДК 004.422.83

О. Миколаєвич, І. Бойко, канд. фіз-мат.наук, доцент.

Тернопільський національний технічний університет імені Івана Пулюя, Україна

РОЗРОБКА ІНЙОРМАЦІЙНОЇ СИСТЕМИ АГРАРНОГО ПІДПРИЄМСТВА З ВИКОРИСТАННЯМ МОВИ C# ТА ТЕХНОЛОГІЇ .NET

O. Mikolaievich, I. Boyko, Ph.D, Assoc.Prof.

DEVELOPMENT OF AGRICULTURAL ENTERPRISE INFORMATION SYSTEM USING C # LANGUAGE AND .NET TECHNOLOGY

Інформаційна система — сукупність організаційних і технічних засобів для збереження та обробки інформації з метою забезпечення інформаційних потреб користувачів. Основними завданнями такої інформаційної системи є збір та обробка інформації.

Враховуючи те що при зборі інформації буде використовуватися реляційна модель бази даних найкращим варіантом буде використання системи керування базами даних Microsoft SQL Server. Microsoft SQL Server це компактний багатопотоковий сервер баз даних, який характеризується високою швидкістю, стійкістю і простотою використання.

Для побудови інтерфейсу використовується Windows Presentation Foundation з використанням .NET. Програмна технологія Microsoft .NET є платформою для створення і звичайних програм, так і веб застосунків. Також платформа підтримує багато мов програмування і є кросплатформовою, що є великим плюсом, тому що програмісту дається на вибір багато мов і він може вибрати для себе ту, якою найкраще володіє. Для реалізації інформаційної системи для розподілу навчального навантаження було взято мову. C# — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Microsoft .NET — програмна технологія, запропонована фірмою Microsoft. Основою платформи є загальномовне середовище виконання Common Language Runtime (CLR), яка підходить для різних мов програмування. Функціональні можливості CLR доступні в будь-яких мовах програмування, що використовують цю середу.

За допомогою таких потужних елементів розробки створюється інформаційна система, покликана підвищити продуктивність праці сільськогосподарських підприємств, справити позитивний вплив на покращення роботи всієї робочої ланки, та покращити стабільну передачу інформацію про виконання робіт на підприємстві.

Література

1. Руководства по SQL Server - Інформація. [Електронний ресурс] // – 2019–

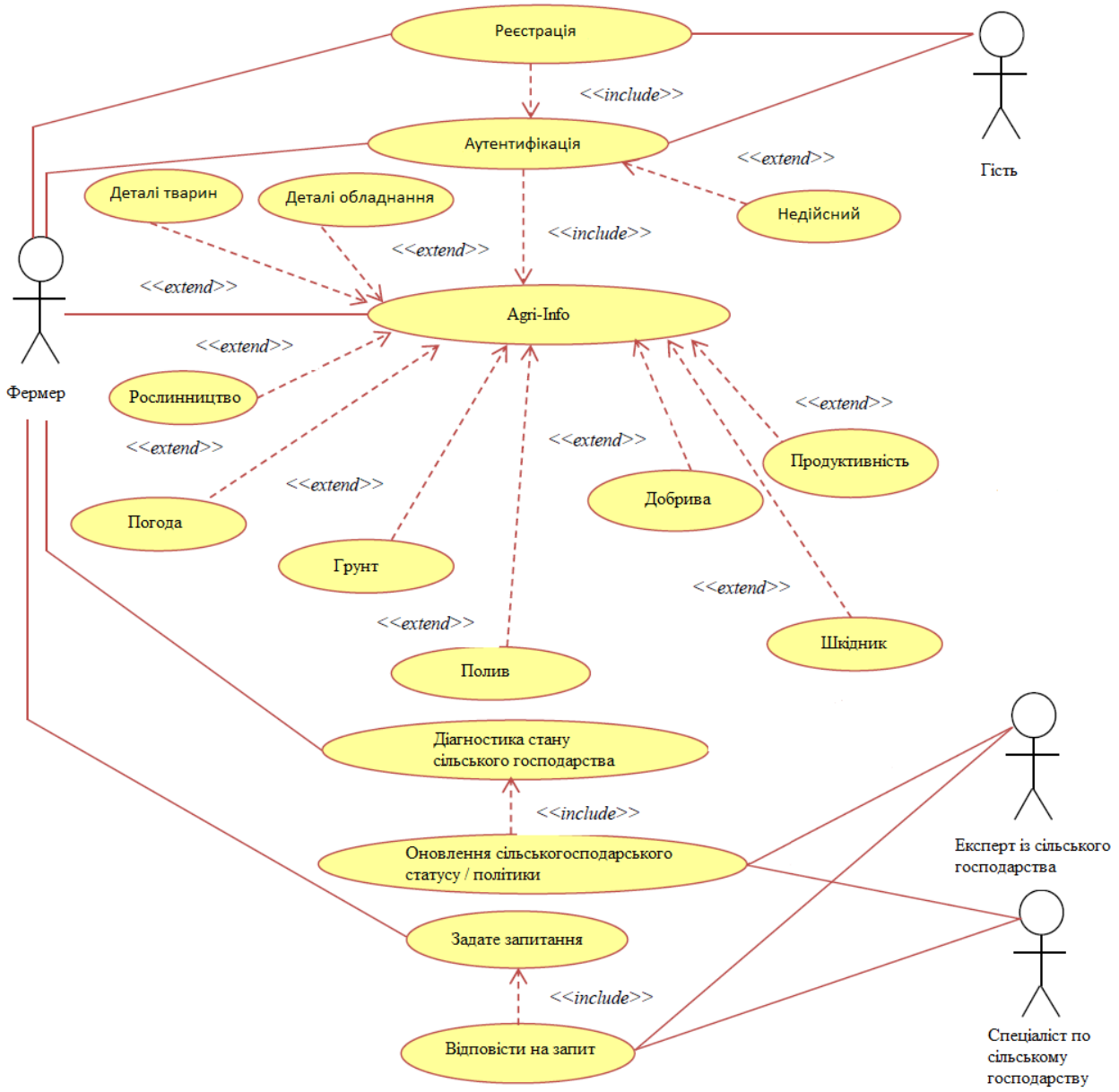
Режим доступу: https://docs.microsoft.com/ru-ru/sql/sql-server/tutorials-for-sql-server-

2016?view=sql-server-ver15 Заголовок з екрану.

2. .NET - Інформація. [Електронний ресурс] // – 2019 – Режим доступу: https://uk.wikipedia.org/wiki/.NET Заголовок з екрану.

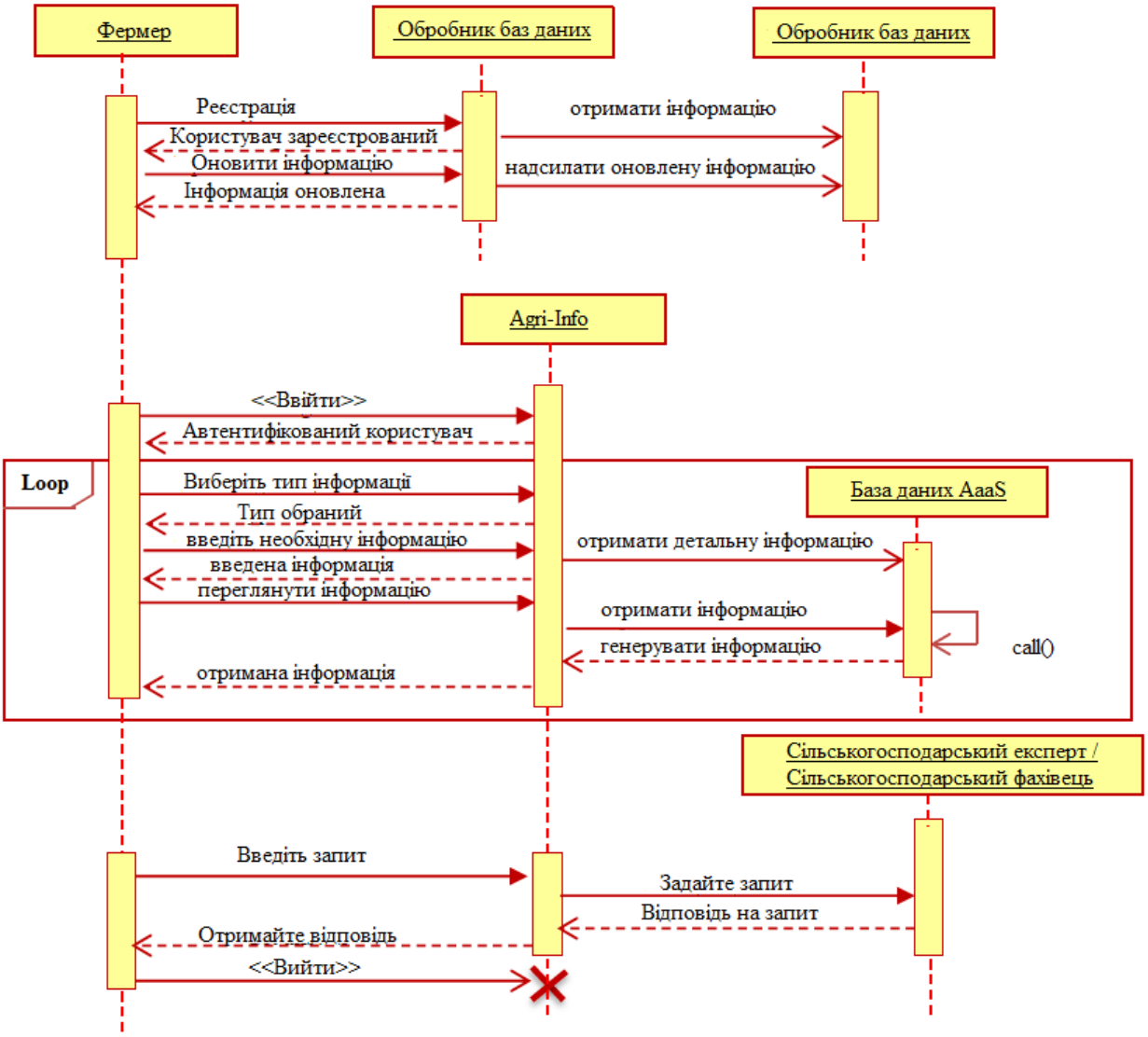
Додаток В

Діаграма варіантів використання



Додаток Г

Схема послідовності дій користувача в Agri-Info



Додаток Д

Диск