РЕФЕРАТ

Пояснювальна записка: ___ с., ___ рис., ___ табл., __ дод., ___ джерел.

Об'єкт розробки: інформаційна система електронного документообігу підприємства.

Мета кваліфікаційної роботи: розробка інформаційної системи електронного документообігу підприємства для об’єднання в єдиний документ виробничий цикл всіх його структурних підрозділів.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування програми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження програми, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення виконання даної роботи полягає в розробці інформаційної системи для організації електронного документообігу підприємства, що дозволить підвищити якість та оперативність управління підприємством; об'єднає в єдиний діловодний цикл всіх структурних підрозділів організації, включаючи територіально-віддалені та забезпечить зниження трудових і тимчасових витрат і накладних витрат.

Актуальність теми кваліфікаційної роботи визначається тим, що при наявності регламентації діяльності, перш за все щодо діловодства, інформаційні технології як каталізатор подальшого прогресу суспільства покликані забезпечити доступний і ефективний інструмент автоматизації на основі безпаперового діловодства та документообігу.

Список ключових слів: ДОКУМЕНТООБІГ, ПІДПРИЄМСТВО, КОРЕСПОНДЕНЦІЯ, БАЗА ДАНИХ, ІНФОРМАЦІЙНА СИСТЕМА, ПРОГРАМУВАННЯ.

ABSTRACT

Explanatory note: ___ pp., ___ fig., ___ table, __ appendix, ___ sources.

Object of development: information system of electronic document management of the enterprise.

The purpose of the qualification work: development of an information system of electronic document management of the enterprise to unite in a single document the production cycle of all its structural units.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the scope of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes the existing solutions, selects the platform for development, performs design and development of the program, describes the algorithm and structure of the program, determines the input and output data, provides characteristics of the parameters of hardware, describes the call and download program .

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

The practical significance of this work is to develop an information system for the organization of electronic document management of the enterprise, which will improve the quality and efficiency of enterprise management; will unite in a single office cycle of all structural units of the organization, including remote and provide a reduction in labor and time costs and overhead costs.

The relevance of the topic of qualification work is determined by the fact that in the presence of regulation, especially in office work, information technology as a catalyst for further progress of society is designed to provide an affordable and effective tool for automation based on paperless paperwork and document management.

List of keywords: DOCUMENT FLOW, ENTERPRISE, CORRESPONDENCE, DATABASE, INFORMATION SYSTEM, PROGRAMMING.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

АСУ - Автоматизована система обробки інформації;

БД - бази даних;

ЕД - електронний документ;

ЕДО - електронний документообіг;

ЕЦП - електронний цифровий підпис;

ОС - операційна система;

ПЗ - програмне забезпечення;

ПК - персональний комп'ютер;

СУБД - система управління базами даних;

DSA - Digital Signature Algorithm, алгоритм цифрового підпису.

ЗМІСТ

ВСТУП

Процес управління підприємством в умовах ринкової економіки вимагає інтенсифікації та чіткої організації всього підприємства, аж до кожного співробітника. Таким чином, при наявності регламентації діяльності, перш за все щодо діловодства, інформаційні технології як каталізатор подальшого прогресу суспільства покликані забезпечити доступний і ефективний інструмент автоматизації на основі безпаперового діловодства та документообігу.

Для ефективного менеджменту в діяльності будь-якого підприємства інформаційні потоки повинні представляти собою чітко відстежуваний і керований процес. Типовим інформаційним об'єктом, що фіксує і регламентує діяльність на підприємстві, є документ. Діяльність з організації проходження документів всередині підприємства прийнято називати діловодством даної конкретної організації. В основі діловодства лежить фундаментальне поняття структури інформаційного обміну - документопотік.

Метою кваліфікаційної роботи бакалавра є розробка інформаційної системи електронного документообігу підприємства для об’єднання в єдиний документ виробничий цикл всіх його структурних підрозділів.

Завданням даної кваліфікаційної роботи є розробка інформаційної системи електронного документообігу. Незважаючи на широке розмаїття готових продуктів даної області на ринку необхідно зробити універсальну, недорогу систему для переходу організації з паперового документообігу на електронний. До того ж, більшість компаній збираються розробляти в найближчому майбутньому крос-платформні проекти, а використання засобів «C #» дозволяє запускати програму на будь-якій платформі, що підтримує .Net Framework без повторної компіляції програми. Також ця програма буде використовувати безкоштовну базу даних MySQL, що позбавить організацію від додаткових витрат на комерційні програми.

Ефективність впровадження розробленої автоматизованої інформаційної системи полягає в:

– автоматизації розсилки наказів по підприємству;

– підвищення продуктивності праці відділу, за допомогою зниження навантаження на операторів;

– підвищення відмовостійкості системи, шляхом впровадження технологій баз даних.

Практичне значення виконання даної роботи полягає в розробці інформаційної системи для організації електронного документообігу підприємства, що дозволить підвищити якість та оперативність управління підприємством; об'єднає в єдиний діловодний цикл всіх структурних підрозділів організації, включаючи територіально-віддалені та забезпечить зниження трудових і тимчасових витрат і накладних витрат.

Дана інформаційна система може бути використана на будь-якому підприємстві з аналогічними функціональними вимогами до електронного документообігу.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Під документообігом розуміється комплекс робіт з документами: прийом, реєстрація, розсилка, контроль виконання, формування справ, зберігання і повторне використання документації, довідкова робота. Електронний документообіг (ЕДО) - це єдиний механізм роботи з документами, представленими в електронному вигляді, з реалізацією концепції «безпаперового діловодства».

Існує також таке поняття, як електронний документ (ЕД) - документ, створений за допомогою засобів комп'ютерної обробки інформації, підписаний електронним цифровим підписом (ЕЦП) і збережений на машинному носії у вигляді файлу відповідного формату.

Головне призначення систем електронного документообігу - це організація зберігання електронних документів, а також роботи з ними (зокрема, їхнього пошуку як по атрибутах, так і по змісту).

Під управлінням електронним документообігом у загальному випадку прийнято розуміти організацію руху документів між підрозділами підприємства, групами користувачів або користувачами. При цьому під рухом документів розуміється не їхнє фізичне переміщення (вони найчастіше залишаються на сервері), а передача прав на їх використання з повідомленням конкретних користувачів і контролем.

У системах електронного документообігу також реалізований санкціонований доступ до документів, відстежуються зроблені в них зміни і контролюються всі їхні версії і підверсії.

Для будь-якої організації можна виділити три основні потоки документів: вхідні, внутрішні та вихідні, які і визначають документообіги.

Успішність автоматизації документообігу організації багато в чому буде залежати від правильного вибору, професійного впровадження і забезпечення експлуатації автоматизованої системи. При цьому вибір програмного продукту для автоматизації документообігу стає нагальним завданням для багатьох організацій незалежно від їх організаційно-правових форм. Потреби автоматизації документообігу і досягнення в області нових інформаційних технологій призводять до постійного зростання пропозицій на ринку програмних продуктів. В даний час різними фірмами розробляються, впроваджуються і проходять дослідну експлуатацію велика кількість автоматизованих систем управління документацією, що характеризуються специфічними підходами та комп'ютерно-комунікаційними засобами реалізації.

Споживачами є компанії всіх організаційно-правових форм, які можуть бути класифіковані на основі різних принципів: сферам діяльності, формі власності, правовим положенням та ін. Проте, незважаючи на зовнішні відмінності в діяльності організацій проглядаються спільні риси, подібні форми, методи і технології роботи.

Всіх споживачів можна умовно розділити на наступні категорії:

– великі ієрархічні структури;
– територіальні органи управління;
– комерційні та некомерційні організації середнього і нижчого рівня.

Підготовка та прийняття управлінського рішення з питання автоматизації документообігу є досить складним процесом, що вимагає від керівництва організації обліку багатьох технічних, економічних, організаційних і соціально-психологічних особливостей її розвитку. Разом з цим, автоматизація управління документообігом вимагає певних фінансових вкладень, через що керівництву організації необхідно визначиться - наскільки цей захід фінансово обґрунтовано і в якому обсязі необхідно.

Перш за все, необхідно визначитися з тим, де проходить межа документообігу організації; які саме структурні підрозділи (їх склад) включені

в документообіг, і, виходячи з отриманих даних, визначити необхідний рівень автоматизації.

Незважаючи на те, що, як правило, документообіг організації пронизує всі сфери її діяльності, а з документованої інформацією, так чи інакше, працюють практично всі підрозділи, потреба в автоматизації процесу в кожній організації можуть бути індивідуальні відмінності (відмінна навіть від аналогічної організації). Тому кожна організація повинна визначити сама для себе, на якому рівні вона має намір провести автоматизацію документообігу. Говорячи про необхідність визначення рівня автоматизації документообігу стосовно кожної конкретної організації слід визначитися з тим, що має охопити сам процес:

– всю чи організацію, включаючи її територіально-віддалений підрозділу;

– головний офіс організації цілком;

– ряд структур, активно задіяних в документообігу організації (наприклад, служба кадрів, служба управління, бухгалтерія та ін.);

– структури, що відповідають за організацію документообігу (управління справами, секретаріат, канцелярія тощо);

– спеціальні структури (бухгалтерія, каталог тощо).

Функціональний аналіз діяльності організації, її підрозділів, зон і конкретних робочих місць дозволяє побудувати реальну, властиву даної організації в цілому, і кожного робочого місця в окремо, структуру інформаційних зв'язків. Визначивши необхідний рівень автоматизації, компанія повинна підбирати систему, яка задовольняла б потреби організації в автоматизації, вирішувала б поставлені завдання.

З точки зору повноти охоплення технології обробки інформації і документації в організації, автоматизовані системи документообігу підрозділяються на наступні типи:

– автономні (локальні) системи по обробці документів (наприклад, робоче місце секркетаря керівника);

– системи комплексного документаційного та інформатизаційного менеджменту;

– системи для вирішення прикладних інформаційних завдань (наприклад, текстообробники).

При виборі необхідної системи слід врахувати і ту обставину, для якого типу управлінської структури підбирається система, тому що від цього залежить і тип вибору самої програми.

З огляду на різноманітність організацій, необхідно зауважити, що модель управління інформацією повинна розроблятися для конкретної організації і відображати її специфічне внутрішнє і зовнішнє інформаційне середовище в процесі функціонування. А при виборі методології побудови автоматизованої системи документообігу в конкретній організації, слід врахувати, наскільки вона відповідає прийнятому в ній стилю управління і системі менеджменту.

Оскільки системи електронного документообігу вже досить широко застосовуються на самих різних підприємствах всіх сфер діяльності, то при виборі конкретної системи вельми важливо ознайомитися з досвідом реалізації подібних проектів на аналогічному підприємстві.

Таким чином, при визначенні необхідного рівня автоматизації документообігу необхідно вирішити, що саме організація має намір автоматизувати. Вибираючи для себе систему автоматизації роботи з документами, організація розглядає один з двох варіантів:

– автоматизація діловодства;

– автоматизація документообігу.

Системи автоматизованої обробки документів, як мінімум, повинні виконувати чотири основні функції: реєстрацію (індексування), зберігання та пошук, контроль виконання і підготовки документів.

Автоматизована система управління документацією повинна створити в організації єдиний документований інформаційний простір, що дає користувачам засоби для підвищення ефективності управління та спільної роботи над документами співробітниками організації на будь-якому робочому

місці (в межах їх компетенції), в режимі реального часу.

1.2. Призначення розробки та галузь застосування

Система, що розробляється в рамках даної кваліфікаційної роботи система призначена для:

– підвищення якості та оперативності управління, і як наслідок цього забезпечення конкурентоспроможності підприємства на ринку;

– об'єднання в єдиний діловодний цикл всіх структурних підрозділів організації, включаючи територіально-віддалені;

– забезпечення оперативного, і в той же час розмеженого доступу до інформаційних (документаційних) ресурсів організації;

– зниження трудових, тимчасових витрат і накладних витрат, і як наслідок, отримання економічного ефекту від впровадження даного продукту.

Система призначена для автоматизації всього комплексу робіт з документами та об'єднує в собі три складові цієї роботи:

– документування (підготовка, оформлення, погодження, затвердження, і випуск документу);

– документообіг (рух, пошук, оперативне зберігання і використання документів);

– зберігання (зберігання інформації (документів), при необхідності її пошук для використання в роботі).

Таким чином, з огляду на все вище сказане можна зробити висновок: автоматизація документаційного забезпечення управління організації, незалежно від її організаційно-правових форм, зважаючи на комплексний підхід до вирішення проблем документообігу, підвищує оперативність управління, ефективність роботи її співробітників, а, отже, призводить до підвищення конкурентоспроможності на ринку.

1.3. Підстава для розробки

Підставою для розробки кваліфікаційної роботи на тему «Розробка інформаційної системи для організації електронного документообігу підприємства мовою програмування C#» є наказ по Національному технічному університету «Дніпровська політехніка» від __.__. 2021р. № ____-_.

1.4. Постановка завдання

В даний час ефективність управлінської діяльності, в значній мірі залежить від автоматизації всіх управлінських процесів, Таким чином, питання автоматизації документообігу встають на порядок денний дуже гостро. Важливе значення для споживача має вартість системи, тип і масштаби організації, в яких її можна застосувати, обсяг документообігу. Очевидно, що ефективним є еволюційний підхід до автоматизації управління документообігом, який передбачає на першому етапі автоматизацію спеціалізованих служб (канцелярії, бухгалтерії, кадрів і т.п.); в рамках розвитку системи автоматизації масштаб її впровадження розширюється до рівня організації в цілому, а в подальшому забезпечує перехід до переваг безпаперового діловодства

Завданням даної кваліфікаційної роботи є розробка інформаційної системи електронного документообігу. Незважаючи на широке розмаїття готових продуктів даної області на ринку необхідно зробити універсальну, недорогу систему для переходу організації з паперового документообігу на електронний. До того ж, більшість компаній збираються розробляти в найближчому майбутньому крос-платформні проекти, а використання засобів «C #» дозволить запускати програму на будь-якій платформі, що підтримує .Net Framework без повторної компіляції програми.

Для виконання завдання роботи необхідно виконатибули виконані наступні дії:

– розробка логічної моделі програми;
– створення бази даних системи;
– розробка простого и зрозумілого інтерфейсу програми;
– забезпечення доступу до даних БД;
– створення та виконання запитів з БД та вивід їх результатів;
– забезпечення виводу інформації в друкованому виді.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Метою кваліфікаційної роботи бакалавра є розробка інформаційної системи електронного документообігу підприємства для об’єднання в єдиний документ виробничий цикл всіх його структурних підрозділів.

Досягнення поставленої мети можливо за допомогою вирішення функціональних завдань автоматизації документообігу організації, які умовно можна систематизувати за такими областям:

1. Підготовка та оформлення документів:

– підвищення якості та оперативності підготовки документів, що створюються в організації;

– уніфікація процесу роботи з документами на всіх етапах його існування.

2. Організація документообігу та виконання документів:

– виключення дублювання введення інформації про документ на різних етапах роботи з ним (процес реєстрації);

– виключення можливості втрати документа (створення документальної бази організації);

– впорядкування документообігу організації (спрощення схем проходження документів - маршрутизація);

– підвищення оперативності та якості роботи виконавців з документами;

– скорочення термінів виконання та проходження документів;

– своєчасне інформування співробітників і керівництва про надходжені та створені документи (виключення дублювання роботи над одним і тим же документом).

3. Організація контролю виконання документів:

– об'єднання документаційних потоків всіх підрозділів організації (в тому числі територіально-віддалених);

– оперативне отримання інформації про стан виконання і місце знаходження будь-якого документа;

– забезпечення відстеження етапів проходженням документів в підрозділах організації з моменту їх отримання (створення) до завершення роботи з ними (виконання).

4. Організація зберігання документів, пошукова система:

– забезпечення централізованого зберігання текстів документів, підготовлених в електронному вигляді, їх графічних образів і матеріалів до них;

– забезпечення можливості оперативного пошуку і організації логічного зв'язування документів, що відносяться до одного питання;

– забезпечення оперативного пошуку та добірки документів (матеріалів) за тематичним набором реквізитів.

1.5.2. Вимоги до інформаційної безпеки

До розробленої системи пред'являються наступні вимоги до надійності:

– безперервна робота протягом 120 годин (5 діб);

– час відновлення після збою 5 хв;

– час відновлення після відмови одного з елементів системи не повинно перевищувати половини операційного банківського дня;

– при відмові системи, дані, що зберігаються в базах даних, не повинні бути якимось чином бути пошкоджені;

– як система, так і оброблювані в ній дані, повинні бути легко переносимими.

1.5.3. Вимоги до складу та параметрів технічних засобів

Мінімальні системні вимоги для комп'ютера, на якому планується впровадження даної ІС та які дозволять швидко та стабільно працювати з програмою:

– операційна система Microsoft Windows 7;
– процесор з частотою не менше 3 ГГц;
– не менше 3 Гб оперативної пам'яті;
– від 48 Гб на жорсткому диску;
– графічна карта сумісна з DIRECTX 10;
– необхідний вільний простір на диску - 2Гб.

1.5.4. Вимоги до інформаційної та програмної сумісності

Інформаційна система має розроблятися та використовуватися на базі наступних програмних засобів та технологій:

– мова програмування «C #»;
– середовище розробки Visual Studio 2017;
– СКБД Microsoft SQL Server 2017;
– операційна система сімейства Windows;
– пакет MS Office 2007.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Метою кваліфікаційної роботи бакалавра є розробка інформаційної системи електронного документообігу підприємства для об’єднання в єдиний документ виробничий цикл всіх його структурних підрозділів.

Система призначена для автоматизації всього комплексу робіт з документами та об'єднує в собі три складові цієї роботи:

– документування (підготовка, оформлення, погодження, затвердження, і випуск документу);

– документообіг (рух, пошук, оперативне зберігання і використання документів);

– зберігання (зберігання інформації (документів), при необхідності її пошук для використання в роботі).

2.2. Опис застосованих математичних методів

Під час проектування та розробки даної інформаційної системи математичні методи не використовувалися.

2.3. Опис використаних технологій та мов програмування

Завданням даної кваліфікаційної роботи є розробка інформаційної системи електронного документообігу. Незважаючи на широке розмаїття готових продуктів даної області на ринку необхідно зробити універсальну, недорогу систему для переходу організації з паперового документообігу на електронний. До того ж, компанії збираються розробляти в найближчому майбутньому крос-платформні проекти, тож, використання засобів C #

дозволить запускати програму на будь-якій платформі, що підтримує .Net Framework без повторної компіляції програми. Також ця програма буде використовувати безкоштовну базу даних SQL Server 2017, що позбавить організацію від додаткових витрат на комерційні програми.

Бази даних, являють собою реляційні бази даних. Ці бази даних складаються з сукупності об'єктів. Такими об'єктами є:

– таблиці даних - зберігають дані, що становлять основний зміст бази даних;

– ключі - сукупності атрибутів, що утворюють ключі (первинні і зовнішні), призначені для здійснення прискореного пошуку даних і забезпечення обмежень посилальної цілісності;

– індекси - спеціальні таблиці, призначені для швидкого пошуку необхідної інформації в таблицях даних;

– представлення (Views) - пов'язані сукупності підмножин таблиць даних, що надаються користувачам для обмеження їх доступу до таблиць даних. При цьому, до одних таблиць у доступі повністю, а в інших таблицях доступ дозволяється тільки до деяких записів цих таблиць;

– збережені процедури і функції-збережені в базі даних підпрограми на мові SQL, скористатися якими може будь-який користувач, який має на це право;

– тригери - підпрограми, що активізуються при настанні певних подій, наприклад, видалення запису з таблиці, модифікація записів тощо. Тригери є потужним засобом забезпечення цілісності даних;

– користувальницькі типи даних - типи даних, створювані користувачем на підставі базових типів даних СУБД;

– системні таблиці - зберігають всю інформацію про схему бази даних і що розміщені в ній об'єкти.

Вся інформація бази даних може розміщуватися в декількох областях. Областю є файл з розширенням db, в якій зберігається вся база даних або один з її фрагментів. Кожна область характеризується своїм ім'ям і файлом,

відповідним цій галузі. Спочатку база даних займає тільки одну область з ім'ям SYSTEM, якій відповідає базовий файл (root file). Потім, у міру необхідності, простір зовнішньої пам'яті бази даних може розширюватися за рахунок додавання нових областей. Ці файли областей можуть бути розміщені в будь-якому каталозі на будь-якому диску і будь-якому вузлі локальної мережі.

Для створення і реалізації бази даних було обрано СУБД Microsoft SQL Server 2017. Microsoft SQL Server – система керування базами даних, розроблена корпорацією Microsoft [20].

Серед нових можливостей і удосконалень Microsoft SQL Server 2017 слід зазначити появу нових типів даних, а саме – для просторових даних, кращу сумісність з застосунками сторонніх розробників, наприклад Oracle, тіснішу інтеграцію з Office, оптимізовані засоби шифрування даних, засоби управління на основі політик, а також покращені інструменти звітності і аналізу.

Основною використовуваною мовою запитів при роботі з базою даних є мова SQL. Вона є реалізацією стандарту ANSI / ISO по структуруванню мови запитів (SQL) з розширеннями.

SQL (англ. Structured Query Language) – мова структурованих запитів - це універсальна мова для створення, модифікації та керування інформацією, яка входить до складу реляційних баз даних [9].

На сьогоднішній день SQL – це єдиний механізм, який здатний зв’язати прикладне програмне забезпечення та базу даних.

Сама мова запитів володіє кількома видами запитів. Варто відзначити, що будь-який запит SQL має під собою на увазі звернення до бази даних. У зв’язку з цим прийнято виділяти такі види запитів:

– створення, зміна в базі даних нових або вже існуючих об’єктів;
– отримання даних;
– додавання нових даних в таблицю;
– видалення даних;
– звернення до системи керування базами даних.

Розглянемо основні переваги SQL:

– незалежність від існуючої в даній системі СКБД, а саме тексти SQL є універсальними для багатьох СКБД;

– наявність стандартів SQL сприяє «стабілізації» мови;

– декларативність, при роботі з даними, програміст вибирає тільки ту інформацію, яка повинна бути змінена або модифікована.

Перейдемо до недоліків мови SQL:

– складність SQL;

– деяка невідповідність стандартів;

– невідповідність реляційної моделі даних.

Створення проекту проходило на мові програмування «C #» в середовищі розробки Microsoft «Visual Studio 2017», що використовує «.Net Framework» другої версії. Це обумовлено тим, що дана середовище програмування поширюється абсолютно безкоштовно за ліцензією GNU GPL і дозволяє створювати легальні програми з будь-якими іншими ліцензіями. Вона мало чим поступається Microsoft Visual studio 2010 для створення невеликих програм, тому і була обрана в якості середовища розробки. Сама мова C # є гнучким і зручним засобом, дозволяючи створювати в найкоротші терміни пристойні програмні продукти. Вбудовані в мову засоби роботи з пам'яттю також полегшують працю програміста. Але є і недоліки у цієї мови, що виражаються в сповільненій роботі додатків на слабких комп'ютерах. Проблема пов'язана з тим, що Just-in-time компілятор, що застосовується в C #, запускається в момент виконання програми і компілює її з Intermediate Language в командний код частинами. Ця проблема впирається в саму концепцію використання проміжного коду. Однак такий підхід дозволяє використовувати програму, написану на C # в різних операційних системах, що підтримують .Net Framework, як-то: Windows, Linux, FreeBSD та інші. Програмісту не потрібно піклуватися про переносимість і доопрацювання коду для різних платформ. Досить тільки відкомпілювати програму один раз і використовувати в сумісних системах.

2.4. Опис структури системи та алгоритмів її функціонування

2.4.1. Функціональне проектування системи

Однією з основних завдань введення електронного документообігу є перехід від паперових документів до документів, що зберігаються в єдиній базі даних в електронному вигляді. Для цього і служить процедура реєстрації документів. Реєстрація повинна бути побудована таким чином, щоб користувач інтуїтивно міг зрозуміти, які поля обов'язкові для заповнення, а які ні. Отже, необхідно зробити програмну перевірку заповнення полів. До необхідних полів введення при реєстрації документа варто віднести:

– вид документа (акт, довіреність, договір, скарга, заява, інструкція, лист, доручення, наказ, розпорядження, постанова та ін.);

– внутрішній номер документа в організації (необхідно, якщо перекладається в електронний вигляд існуюча база документів організації, що має власні реєстраційні номери);

– контролерів по документу і дорученням;

– виконавців документа і доручення;

– саме доручення (назва і зміст);

– терміни виконання документа і доручення;

– процедуру імпортування готових документів, що знаходяться на різних носіях.

Після заповнення необхідних полів і підтвердження реєстрації документа ці дані повинні бути збережені в таблиці бази даних, а також документам повинен бути присвоєно порядковий номер в системі. Після цього триває подальша робота з програмою.

Також необхідний механізм електронних цифрових підписів для узгодження документів, а також для підтвердження виконавцями їх згоди з текстами документів або доручень.

Для швидкого доступу до інформації, що зберігається в єдиній базі даних документів, необхідна процедура пошуку за різними параметрами. До основних параметрів пошуку в системі документообігу можна віднести:

– порядковий номер документа, присвоєний системою;
– внутрішній номер документа в організації;
– період дати реєстрації;
– особа, що зареєструвала документ;
– вид, потік, а також важливий параметр - текст документа.

Ці основні критерії дозволяють отримати вичерпну інформацію по документам, збереженим в базі даних і швидко знайти потрібний документ.

До наступної необхідної функції варто віднести можливість обміну текстовими повідомленнями всередині організації. У процесі підготовки документів, а також при їх узгодженні потрібно робити досить великий обмін інформацією між різними підрозділами. Для того щоб прискорити цей процес (а він може бути дійсно тривалим, особливо якщо підрозділи територіально віддалені один від одного), в систему документообігу потрібно впровадити можливість обміну повідомленнями між її користувачами. Це дозволить, не користуючись сторонніми програмами, організувати поштове листування всередині компанії. Кожен співробітник, використовуючи адресну книгу (яка бере дані про зареєстрованих користувачів в системі), може відправити лист своєму колезі в організації і отримати від нього відповідь.

Для своєчасного виконання документів і доручень, а також контролю їх виконання, користувачам, зазначеним при реєстрації документа в якості контролерів і виконавців має приходити повідомлення про їх призначення на ці ролі. Ці повідомлення також повинні бути винесені в головне вікно програми і розділені на різні категорії, щоб користувачі могли швидко помітити призначені для них завдання і приступити до виконання.

Як зв'язок з сервером передбачається виконувати різні зазначені запити, які він зможе розпізнати і обробити. Так само, як і в програмі-сервері, передбачається розробити окремий клас для мережевої взаємодії між клієнтом і

сервером. Тут потрібно врахувати всі ті ж варіанти запитів, як і на сервері (відправка, отримання, відправка і отримання), щоб організувати логічний зв'язок цих двох програм.

Для зберігання даних програми необхідно використовувати реляційну базу даних. Посилаються один на одного таблиці повинні мати хоча б один стовпець, званий первинним ключем. Рядки в цьому стовпці унікальні. Вони не можуть повторюватися, не можуть бути порожніми і ці значення дозволяють відокремити один рядок таблиці від будь-якої іншої. Використання реляційної бази даних обумовлено тим, що це дозволяє економити місце на диску, знизити витрати на запис даних, легко маніпулювати окремими даними і логічно розбити на частини все безліч інформації. Для розбиття даних логічно з вищесказаного випливає поділити базу на ці основні таблиці:

– таблиця документів (зберігає номери, дати реєстрації, виконання, а також самі тексти документів);

– таблиця текстових повідомлень (містить відправника, одержувача, тимчасові позначки, реквізити електронного листа);

– таблиця користувачів (дані по користувачах, внесені через сервер документообігу);

– таблиці виконавців, контролерів, а також інші службові таблиці.

Перед першим запуском системи потрібно відтворити в базі даних структуру таблиць і заповнити її початковими значеннями, необхідними для нормальної роботи програми. Цей процес повинен бути автоматизований і виконуватися через сервер документообігу адміністратором.

2.4.2. Опис логічної структури програми

Розроблена система являє собою додаток з клієнт – серверною архітектурою. На рис. 2.1 представлена серверна частина, яка складається з СКБД Microsoft SQL Server 2017 для бази даних і набору динамічних бібліотек для організації доступу до бази з боку клієнта – постачальника OLE dB.

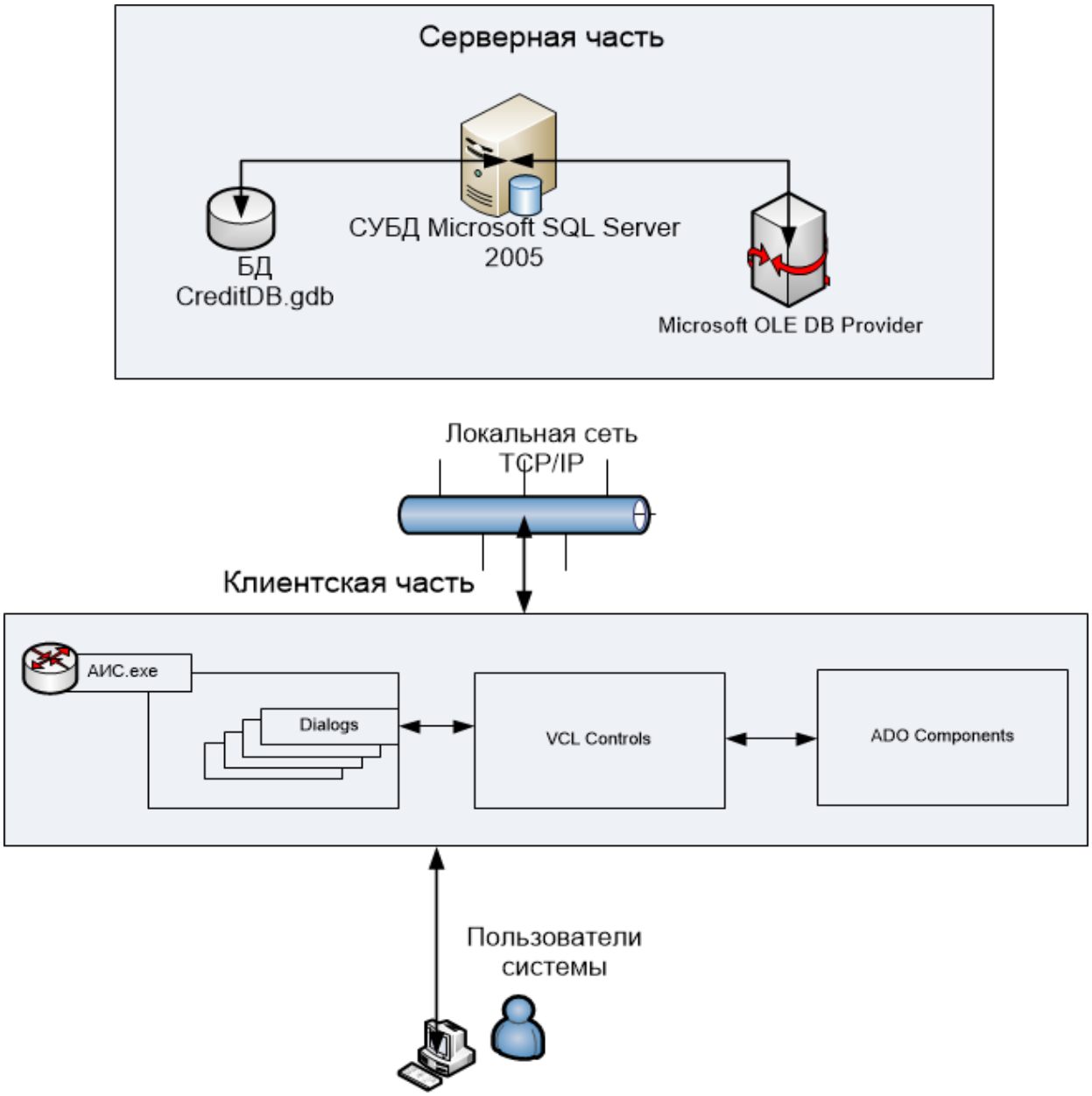

Рис.2.2. Структура функціонування системи

Отже, програма складається з двох основних частин: клієнта і сервера. Сервер з'єднується з базою даних, а клієнти, в свою чергу, підключаються до даного сервера для обміну інформацією, виконання централізованих розрахунків, отриманням і записом даних.

З'єднання клієнтів з сервером здійснюється по протоколу TCP/IP, щоб уникнути втрат важливих даних. До того ж нинішні мережеві технології дозволяють використовувати високошвидкісні канали зв'язку за відносно

невеликі фінансові вкладення. Падіння цін на телекомунікаційне обладнання повсюдно пов'язане з науково-технічним прогресом.

При такій схемі база даних і серверна частина програми електронного документообігу можуть перебувати фізично на різних серверах. Це дозволяє знизити вимоги до обладнання і знизити навантаження на окремі вузли мережевої інфраструктури.

На рис. 2.2 представлена схема взаємодії основних компонентів програми.

Розглянемо склад клієнтської і серверних модулів і принципи взаємодії між ними.

Серверний модуль складається з двох додатків: служби windows і діалогового вікна налаштування властивостей системи.

Всі настройки системи і дані про процеси підприємства зберігаються в базі даних Apriori. Служба Windows - Apriori як і діалогове додаток взаємодіє з БД Apropri за допомогою СУБД SQL Server 2017.

Серверний модуль повинен бути розгорнутий на станції, що виконує роль сервера мережі підприємства.

Розглянемо основні функції серверного модуля:

– обробка клієнтських запитів (реалізована логіка обробки маршрутизації документів, організована система управління повідомленнями між користувачами системи);

– адміністрування системи (настройка з'єднання з БД, реєстрація користувачів, призначення прав користувачам);

– доступ до БД підприємства.

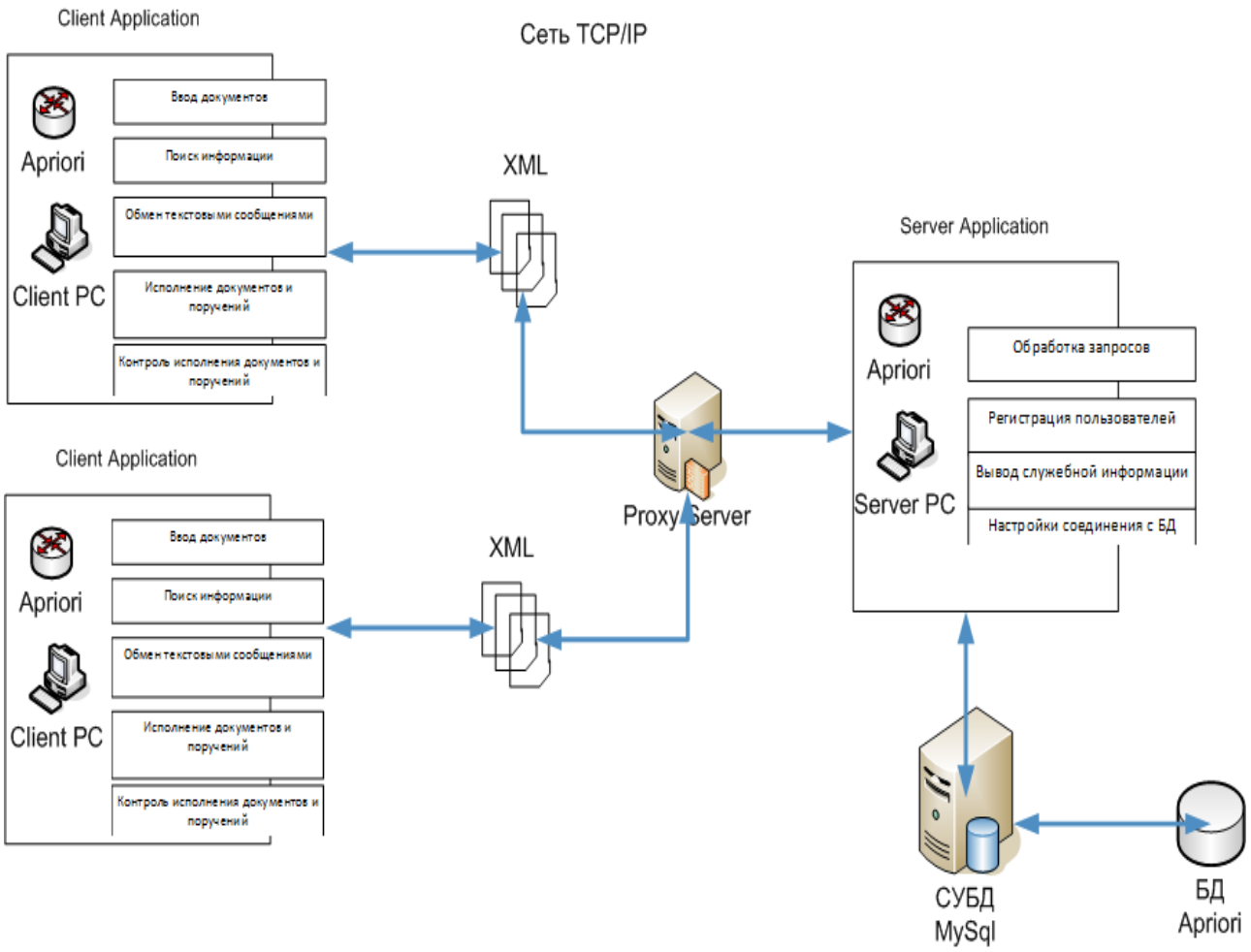

Рис. 2.2. Взаємодія компонентів системи

Клієнтський модуль є окремим діалоговим додатком, який встановлюється на клієнтський ПК. Клієнтський додаток призначений для введення даних від користувача, їх валідацію, відправку на сервер і відображення даних, що прийшли від сервера. Це можуть бути документи, повідомлення, дані необхідні для користувача інтерфейсу (таблицями, діаграмами, повідомленнями, текстових полів та списками). При цьому, клієнтський додаток взаємодіє з сервером за допомогою проксі-сервера мережі. Для обміну даними був обраний формат XML.

2.4.3. Розробка БД системи

База даних складається з ряду необхідних таблиць, таких як таблиця користувачів системи (Users), таблиця текстових повідомлень (Messages), таблиця структури документа (Documents) і інших таблиць, не кажучи вже про більш детальний поділі зазначеної структури. На рис. 2.3 представлена логічна структура бази даних, яка складається з 12 таблиць, наведених до 3-й нормальній формі.

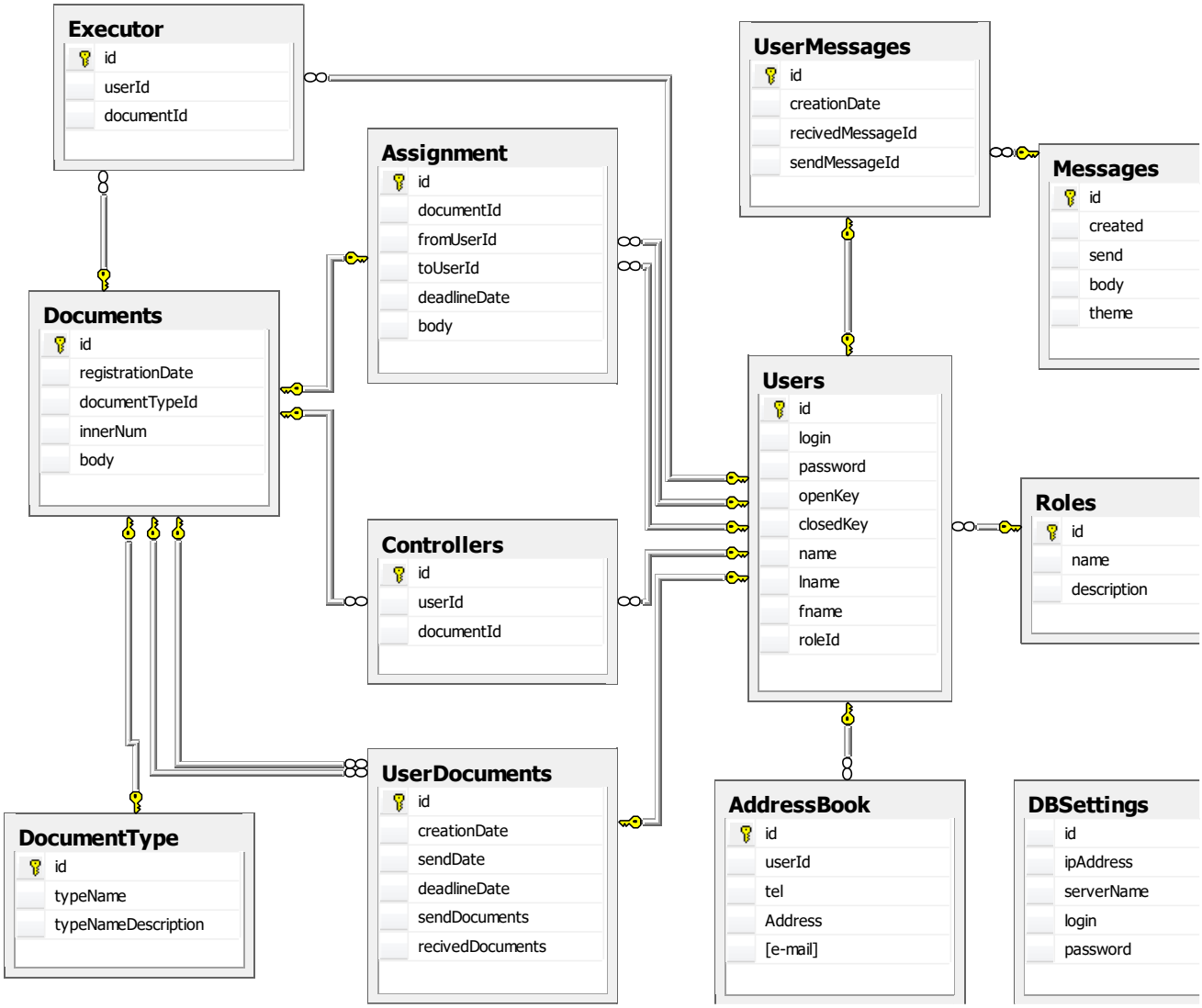

Рис. 2.3. Логічна структура бази даних

Самим документам при реєстрації присвоюється кілька реквізитів, що дозволяють швидко знайти їх за допомогою пошуку в базі даних. Також при реєстрації відділяється сам текст від документа, і це дає можливість виконувати пошук по вмісту документа.

2.4.4. Опис компонентів системи

2.4.4.1. Серверна частина

Програма-сервер документообігу виконує кілька функцій:

– обробка запитів клієнтів;
– вивід службової інформації (логів);
– служить посередником між клієнтом і базою даних;
– здійснює процедуру реєстрації користувачів.

Організація руху інформації (вхід/вихід) в системі схематично наведена на рис. 2.4.

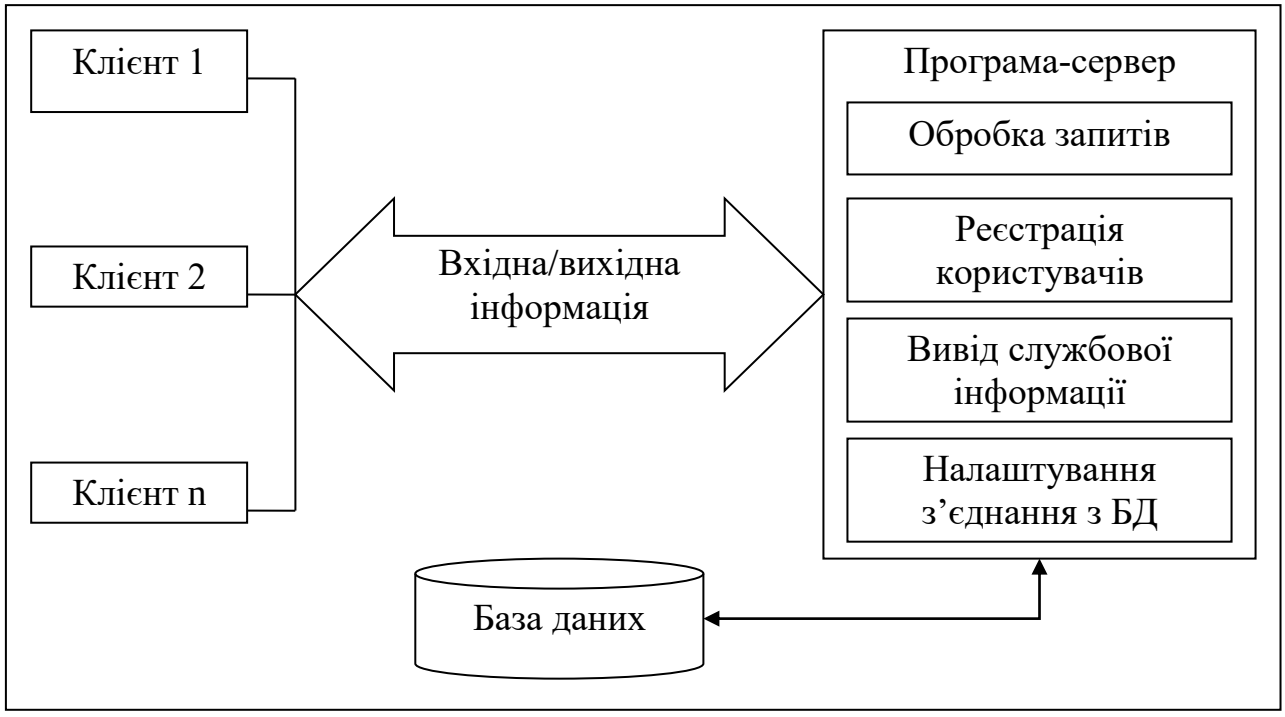

Рис. 2.4. Організація руху інформації (вхід/вихід) в серверній частині системи

Запити до серверної програми повинні надходити по протоколу TCP / IP, так як дані не повинні губитися по шляху проходження. В якості програмних засобів передачі даних було вирішено використовувати набір низькорівневих класів Net Sockets, що дозволяють працювати з керованими сполуками. Так як клієнтів в даній системі може бути кілька (програмно не повинно бути обмежень в кількості клієнтів, їх число обмежується пропускною спроможністю мережі і продуктивністю обладнання), то програма-сервер повинна працювати з ними окремо. Таким чином, планується виділяти клієнтів в самостійні потоки, які будуть народжуватися при отриманні сигналу при новому підключенні і закриватися при від'єднанні користувача.

Необхідно створити кілька можливих типів мережевих запитів, що дозволяють працювати в різних режимах передачі даних: відправка, отримання, відправка і отримання одночасно.

Для дотримання принципів об'єктно-орієнтованого програмування потрібно розділити код за смисловим ознаками в окремі класи. Виходячи з функцій, які виконуються програмою-сервером, можна виділити три основні класи:

– клас інтерфейсу для взаємодії з користувачем;
– клас мережевої взаємодії з клієнтами;
– клас зв'язку з базою даних.

Всі ці класи взаємопов'язані і служать для обробки команд клієнтів.

Спочатку ініціалізується сама програма, потім включаються зазначені основні класи. Мережевий клас отримує команди від клієнта, для їх виконання використовує допоміжний клас взаємодії з базою даних, при необхідності відправляє відповідь клієнту і відображає результати своєї роботи на призначений для користувача інтерфейс. Також можуть використовуватися додаткові структурні ланки.

За допомогою такого поділу на класи реалізується інкапсуляція - це механізм програмування, який об'єднує дані і код в одному блоці, що оберігає їх від втручання ззовні і неправильного використання. «Інкапсуляція дозволяє

об'єднати дані і код в об'єкт і приховати реалізацію об'єкта від користувача. При цьому користувачеві надається тільки специфікація (інтерфейс) об'єкта.

Користувач може взаємодіяти з об'єктом тільки через цей інтерфейс.

Щоб визначити стан сервера і правильність виконання запитів клієнтів необхідно використовувати компонент для запису службової інформації в реальному часі. Цей поповнений список, так званий журнал подій або лог, повинен розташовуватися в основному вікні програми-сервера, щоб найбільш зручним чином відображатиме інформацію для адміністратора системи. Сюди повинні потрапляти, головним чином, результати обробки запитів до бази даних, так як це є дуже вразливим місцем в системі, особливо якщо програма-сервер і сховище даних знаходяться фізично на різних комп'ютерах.

Для скорочення числа додаткових програм в інтерфейс сервера необхідно вбудувати засіб реєстрації користувачів системи. В якості основних ідентифікаційних даних має бути внесено ім'я користувача (логін або псевдонім), справжнє ім'я, прізвище та по батькові, а також пароль. Також необхідно врахувати і додаткові ідентифікатори користувача в організації: номер телефону, адреса знаходження і e-mail. Щоб підвищити рівень безпеки системи пароль потрібно зберігати в базі даних в зашифрованому вигляді. Останній, але важливий елемент в сервері - це параметри настройки з'єднання з базою даних і її стартове розгортання. Щоб сервер міг використовувати як джерело даних довільний доступний вузол мережі, необхідно дати можливість адміністратору задавати параметри підключення до сервера бази даних. У параметри підключення входить IP-адреса комп'ютера, що містить базу даних, ім'я самої бази, логін і пароль користувача, що має доступ до вчинення процедур читання, записи і поновлення таблиць бази даних.

Також до сервера повинна бути приєднана утиліта для початкового створення таблиць в базі і їх заповнення необхідними значеннями. Це дозволить без допомоги сторонніх програм (можливо навіть платних) і знання мови SQL підготувати дану систему документообігу до роботи.

2.4.4.2. Клієнтська частина

Перейдемо до розгляду функціональних особливостей клієнтської частини системи. Так як на сервері відбувається реєстрація користувачів, то клієнт повинен, в першу чергу, пройти процедуру авторизації. Логічно в діалогове вікно авторизації також вбудувати і можливість введення реквізитів з'єднання з сервером. Під цими реквізитами розуміється IP-адреса комп'ютера, на якому розташовується програма-сервер, і порт.

Після відправки введених даних на сервер, клієнт отримує або позитивну відповідь, в разі правильного введення імені користувача і пароля, або повідомлення про відмову доступу. З програмою працювати зможуть тільки користувачі, які пройшли успішно процедуру авторизації.

Після успішного підключення до сервера користувачеві надається головне вікно для роботи з програмою. За своєю суттю це повинно бути SDI (однодокументне) вікно з головним меню і панеллю інструментів. У цьому діалоговому вікні для користувача в ергономічному вигляді надається весь основний функціонал програми-клієнта, а саме:

– введення документів;
– пошук інформації;
– обмін текстовими повідомленнями;
– виконання і контроль доручень і документів;
– відправка запитів на сервер і отримання відповідей.

Організація руху вхідної та вихідної інформації наведено на рис. 2.5.

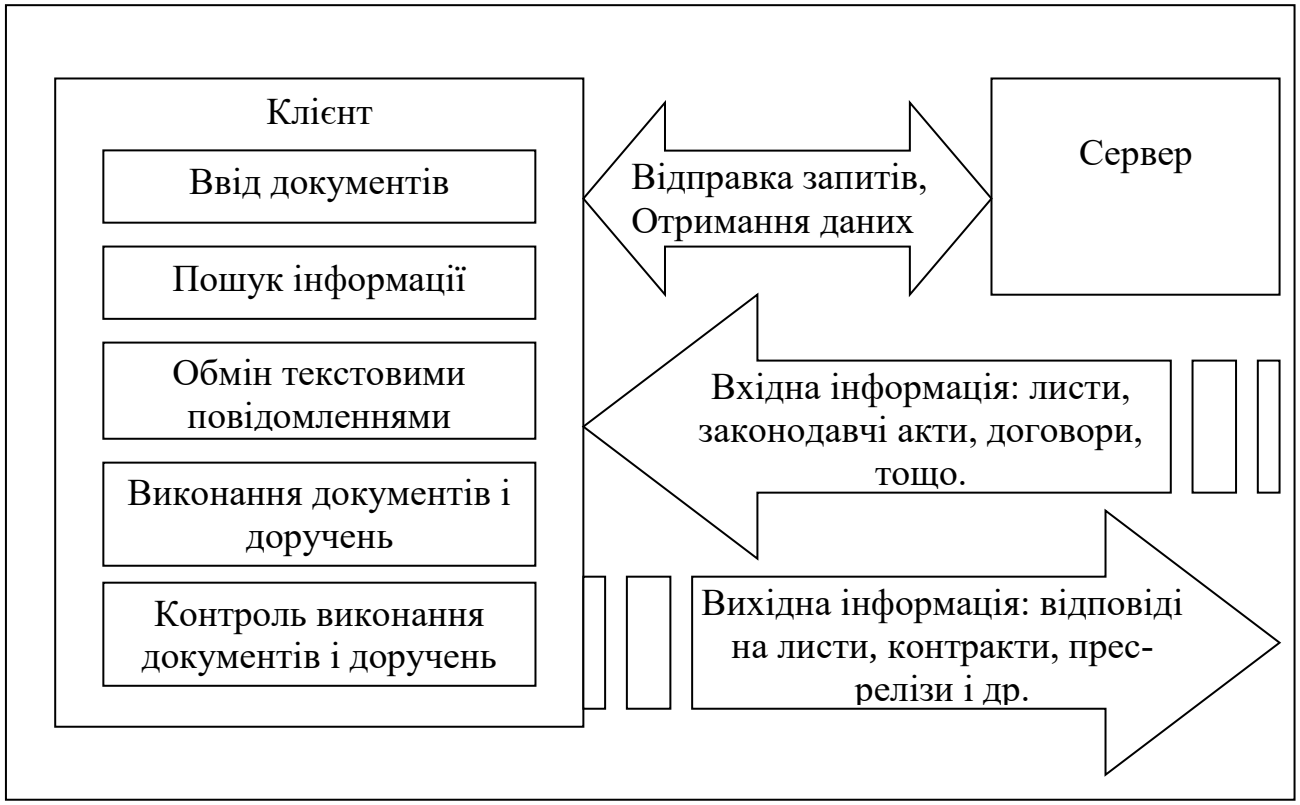

Рис. 2.5. Організація руху вхідної та вихідної інформації в клієнтській частині системи

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Для розроблюваного програмного забезпечення інформаційної системи, метою якої є автоматизація введення, зберігання, обробки інформації та складання звітності, вхідними даними є набір введених користувачем даних, за допомогою діалогових вікон програми. Вхідними даними є всі вносяться оператором дані, необхідні для коректної роботи системи. Це можуть бути документи, повідомлення, дані необхідні для користувача інтерфейсу (таблиці, повідомлення).

До вхідної інформації відносяться: листи, законодавчі акти, договори.

В якості основних ідентифікаційних даних персони виступають: ім'я користувача (логін або псевдонім), справжнє ім'я, прізвище та по батькові, пароль, номер телефону, адреса знаходження і e-mail.

В якості основних ідентифікаційних даних при введенні та реєстрації документа варто віднести:

– вид документа (акт, довіреність, договір, скарга, заява, інструкція, лист, доручення, наказ, розпорядження, постанова та ін.);

– внутрішній номер документа в організації (необхідно, якщо перекладається в електронний вигляд існуюча база документів організації, що має власні реєстраційні номери);

– контролерів по документу і дорученням;

– виконавців документа і доручення;

– саме доручення (назва і зміст);

– терміни виконання документа і доручення;

– процедуру імпортування готових документів, що знаходяться на різних носіях.

Вихідні дані діляться на дві категорії:

1. Візуальні – дані які відображаються на екрані монітора ПК.

2. Друковані – роздруковані форми у вигляді форм звітності.

Вихідні дані поділяються на дві групи:

– у файловому вигляді;

– в електронному, з виведенням на екран діалогового вікна.

У файловому вигляді формуються повідомлення для подальшого подання на друк. Вихідними даними є всі ті дані, які, тим чи іншим чином були отримані з системи.

Дана організація вхідних і вихідних даних обумовлена вимогами поставленої задачі.

Всі настройки системи і дані про процеси підприємства зберігаються в базі даних Apriori. Служба Windows - Apriori як і діалоговий додаток взаємодіє з БД Aproprі за допомогою СУБД SQL Server 2017.

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

Рекомендовані системні вимоги для комп'ютера, які дозволять швидко та стабільно працювати з програмою:

– операційна система Microsoft Windows 10;
– процесор з частотою не менше 4 ГГц;
– не менше 8 Гб оперативної пам'яті;
– від 64 Гб на жорсткому диску;
– графічна карта сумісна з DIRECTX 11/12;
– рекомендований вільний простір на диску - 4Гб.

2.6.2. Використані програмні засоби

Для розробки та проектування інформаційної системи використано мови програмування SQL та C#, а також середовище розробки Microsoft «Visual Studio 2017» та SQL Server 2017

Програма може функціонувати у таких операційних системах: Windows 7, Windows 8, Windows 10 та Linux UBUNTU.

2.6.3. Виклик та завантаження програми

Запуск додатку «Apriori» відбувається за допомогою файлу Apriori.exe після копіювання на ПК директорії проекту «Apriori» . Перед першим запуском системи потрібно відтворити в базі даних структуру таблиць і заповнити її початковими значеннями, необхідними для нормальної роботи програми. Цей процес повинен бути автоматизований і виконуватися через сервер документообігу адміністратором.

2.6.4. Опис інтерфейсу користувача

Так як на сервері відбувається реєстрація користувачів, то клієнт повинен, в першу чергу, пройти процедуру авторизації. Логічно в діалогове вікно авторизації також вбудувати і можливість введення реквізитів з'єднання з сервером. Під цими реквізитами розуміється IP-адреса комп'ютера, на якому розташовується програма-сервер, і порт (рис. 2.6-2.9).

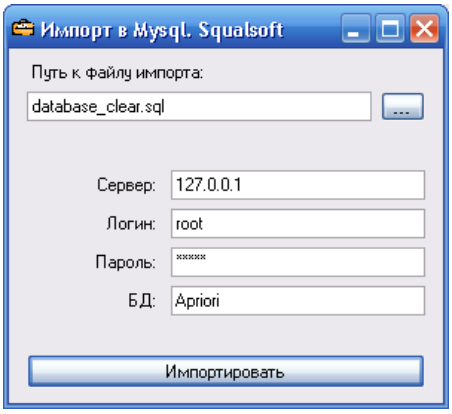

Рис. 2.6. Підключення до серверу БД

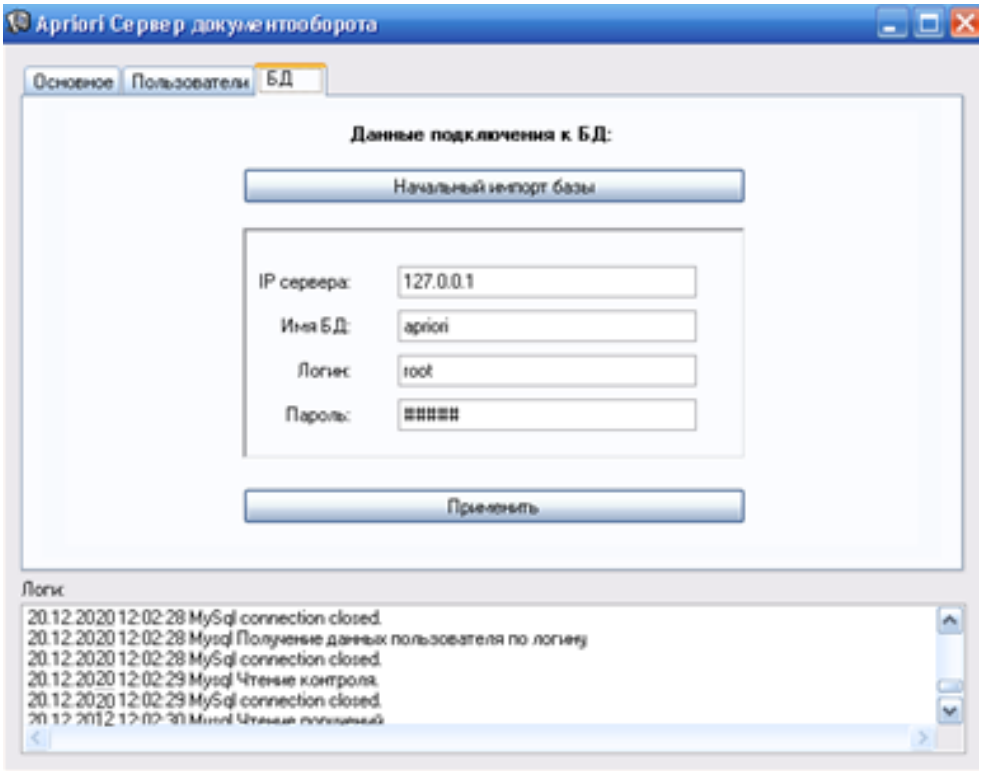

Рис. 2.7. Імпорт БД

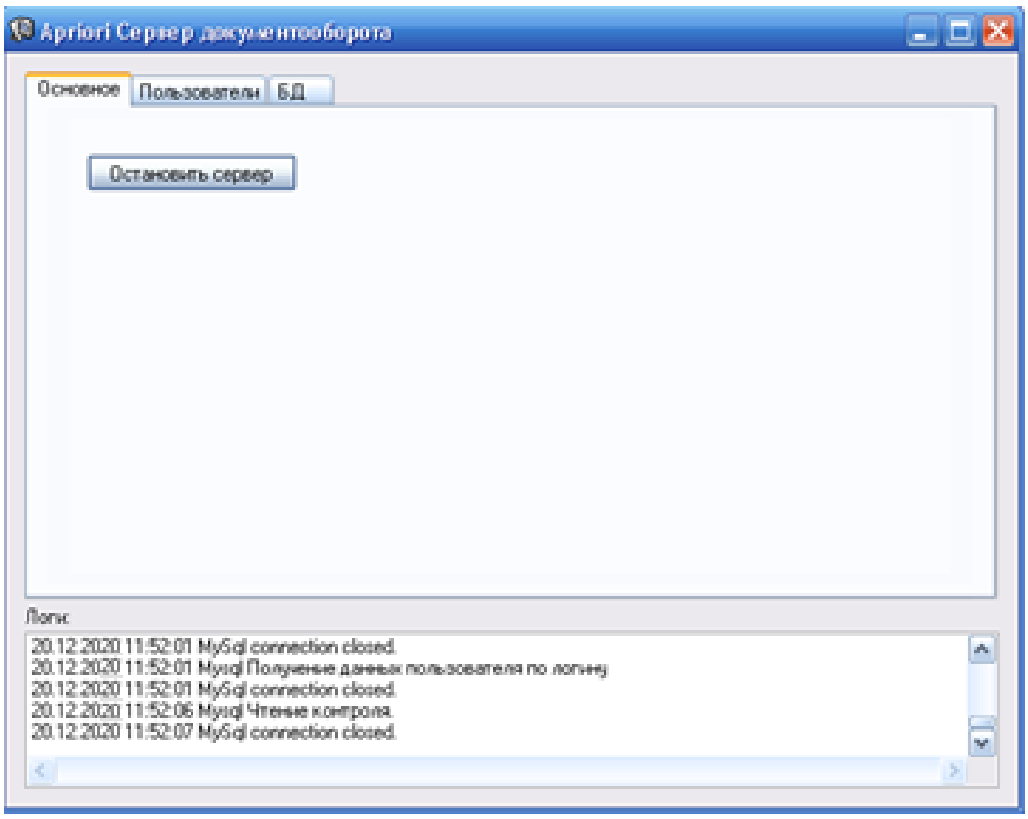

Рис. 2.8. Інформація про зв’язок із сервером БД

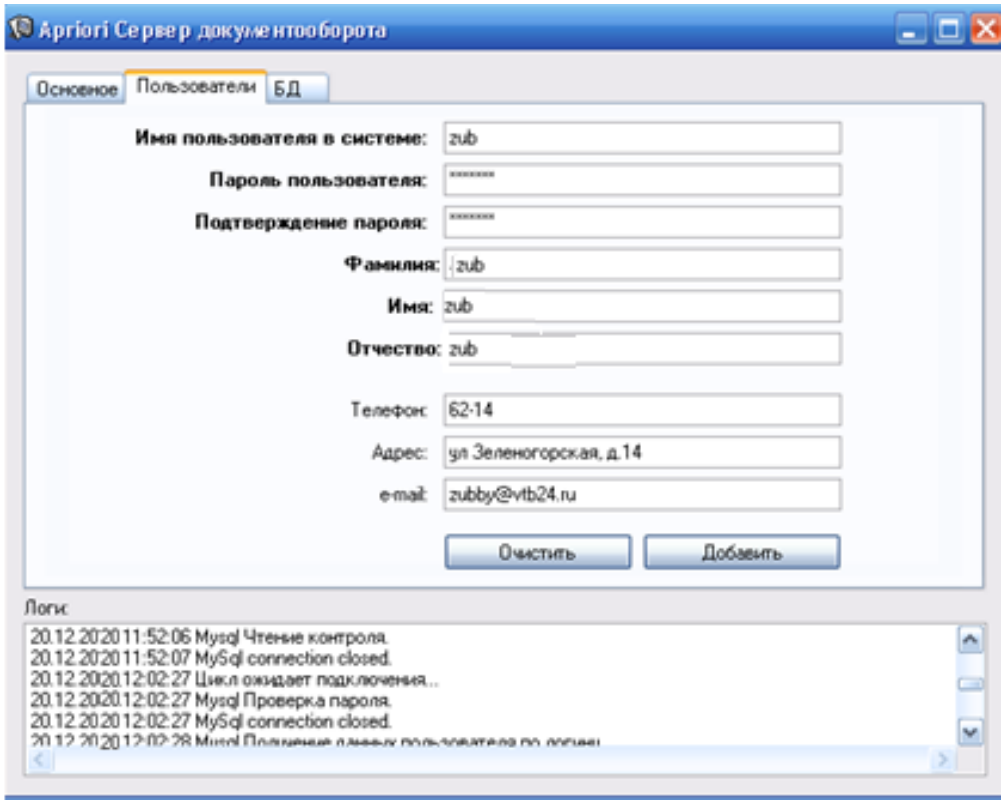

Рис. 2.9. Ввід даних про користувача БД

Для роботи системи необхідно здійснити налаштування з'єднання з базою даних і здійснити імпорт початкових таблиць в базу(рис. 2.10).

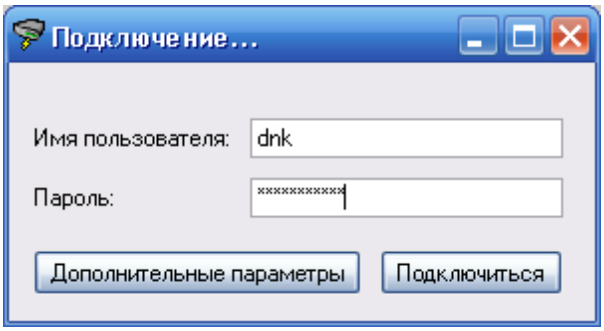

Рис. 2.10. Авторизація та підключення до БД користувача

Після відправки введених даних на сервер, клієнт отримує або позитивну відповідь, в разі правильного введення імені користувача і пароля, або повідомлення про відмову доступу. З програмою працювати зможуть тільки користувачі, які пройшли успішно процедуру авторизації

Після успішного підключення до сервера користувачеві надається головне вікно для роботи з програмою. За своєю суттю це однодокументне вікно з головним меню і панеллю інструментів. У цьому діалоговому вікні для користувача в ергономічному вигляді надається весь основний функціонал програми-клієнта, а саме:

– введення документів;
– пошук інформації;
– обмін текстовими повідомленнями;
– виконання і контроль доручень і документів;
– відправка запитів на сервер і отримання відповідей.

Користувачеві доступно головне вікно програми, яке дозволяє переглядати доручення, контролювати виконання, дивитися вхідну і відправлену кореспонденцію, і видалені файли (рис. 2.11).

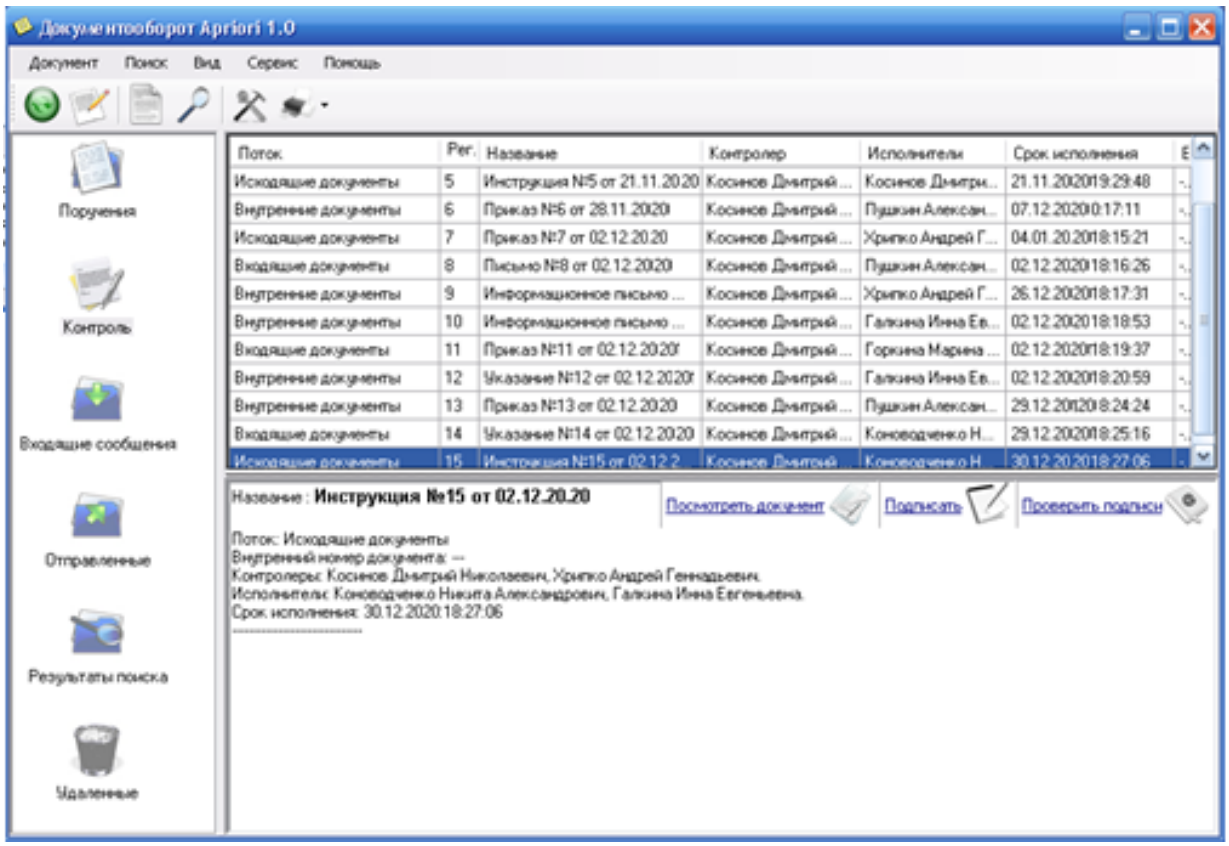

Рис. 2.11. Вікно перегляду списку доручень та перегляду їх змісту

Контроль над дорученнями та листами відбувається, переглянувши відповідну вкладку «Контроль» (рис. 2.12)

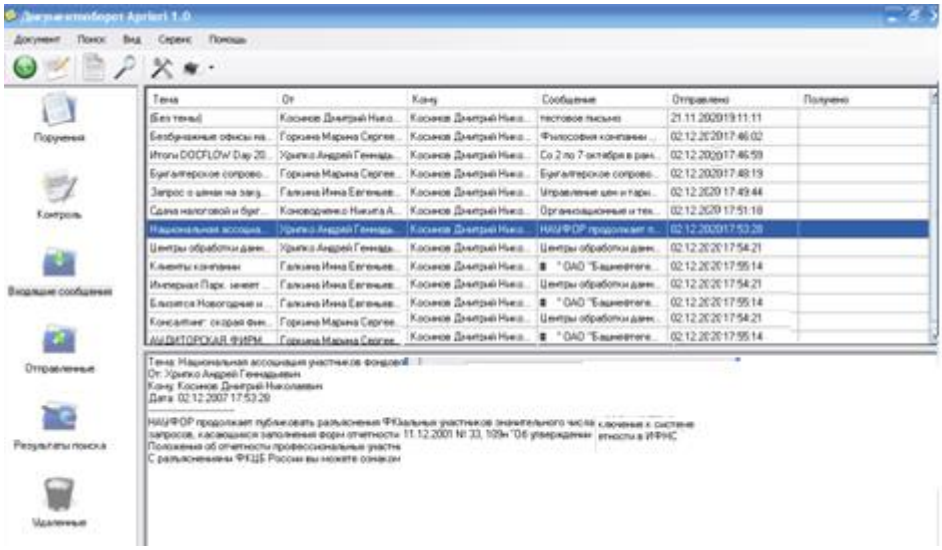

Рис. 2.12. Вкладка «Конторль»

Відіслати текст доручення можна, обравши відповідний пункт меню та

ввівши у відповідне поле текст, призначення та адресата (рис. 2.13).

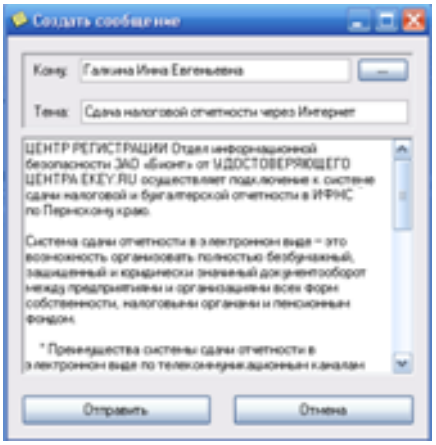

Рис. 2.13. Ввід тексту доручення

Вікно вибору виконавців і контролерів зображено на рис.2.14.

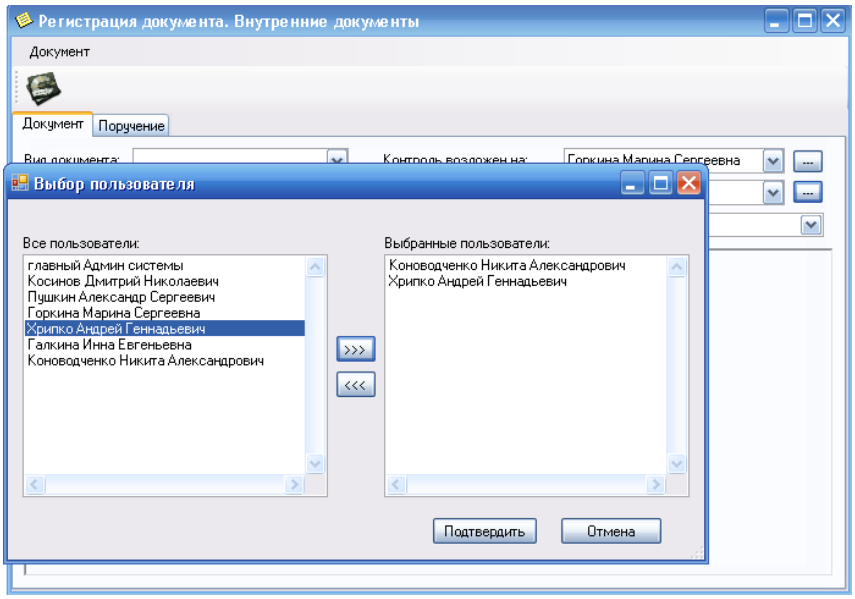

Рис. 2.14. Вибір виконавця і контролера створеного доручення

Для завдання даних про виконавців використовується також інформація з БД про його місцезнаходження (рис. 2.15).

Рис. 2.15. Адресна книга персоналу

На рис. 2.16 - 2.17 представлені вікна для роботи з внутрішніми документами. Система показує номер розпорядження, хто виконавці, терміни виконання.

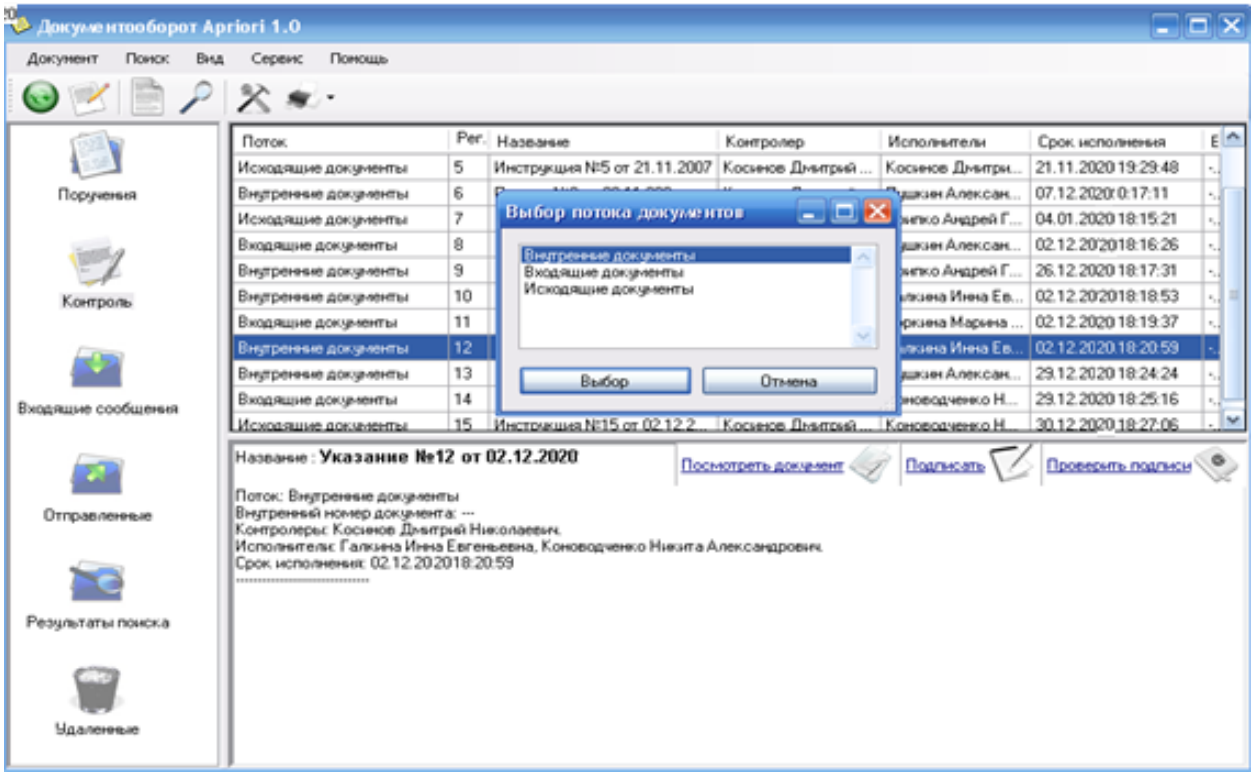

Рис. 2.16. Перегляд потоків переміщення внутрішніх документів

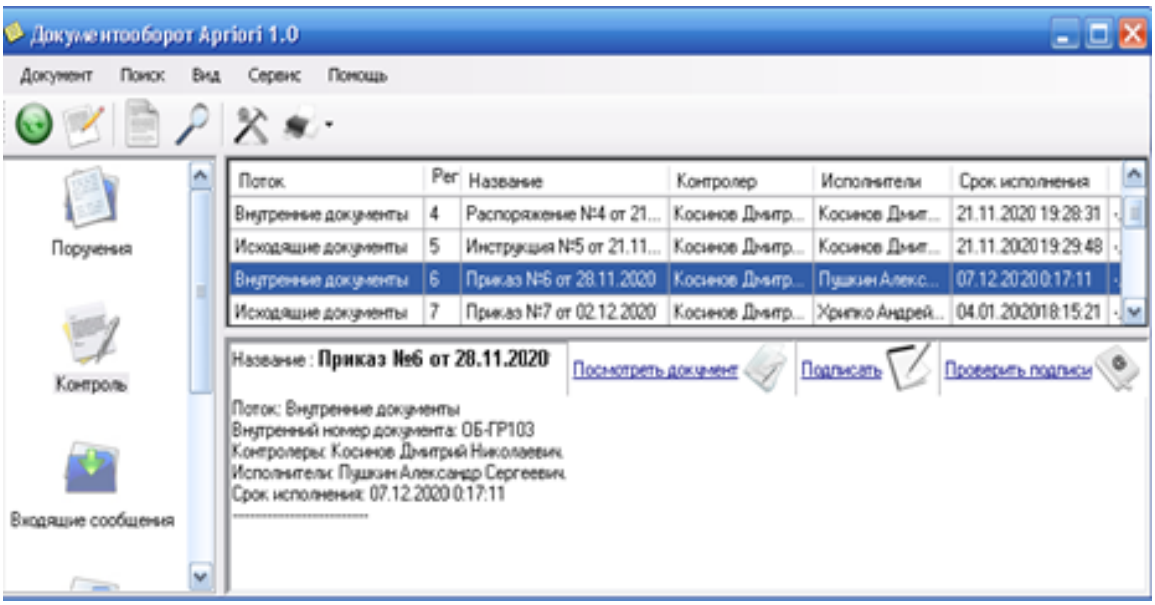

Рис. 2.17. Перегляд змісту внутрішніх документів

На рис. 2.18 – 2.19 показаний приклад вікна створення повідомлення, де документу присвоюється внутрішній номер, виконавці, терміни виконання.

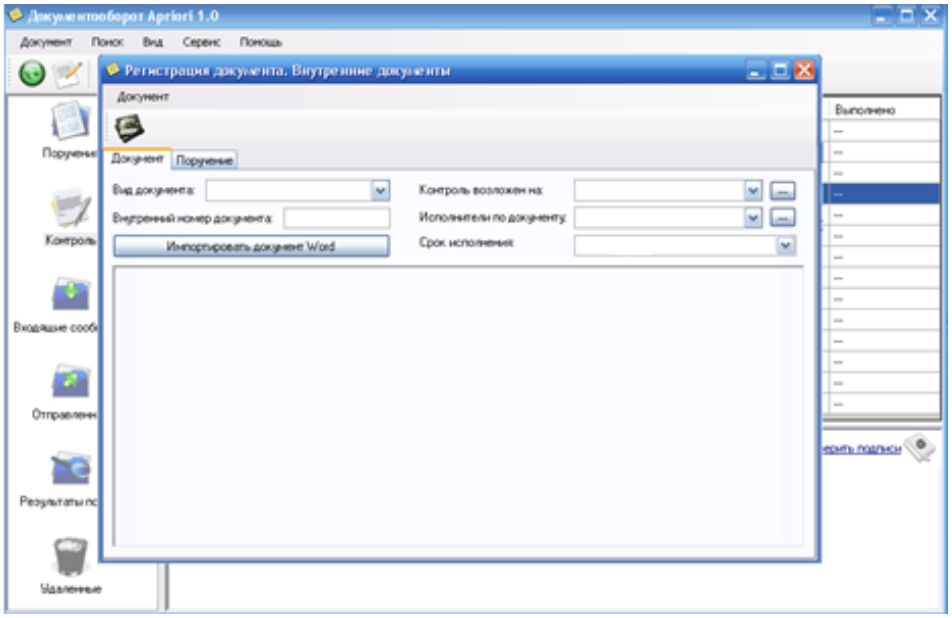

Рис. 2.18. Вікно вводу даних для повідомлення

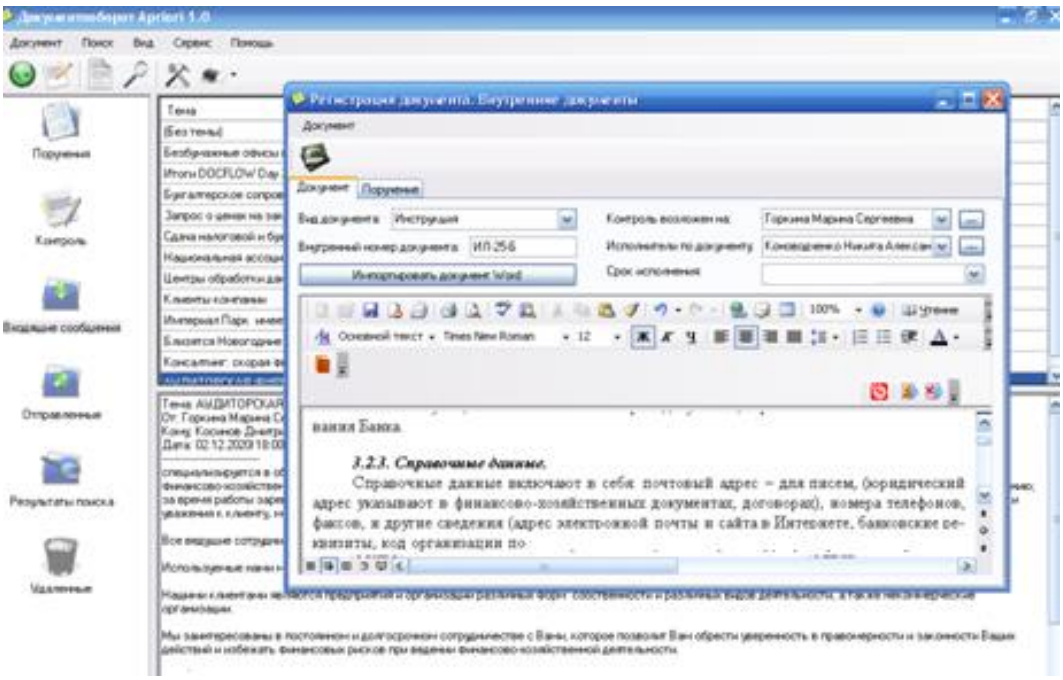

Рис. 2.19. Вікно створення повідомлення

2.6.5. Тестування програми

Для перевірки коректності процесу документообігу в програмі було розроблено кілька тестів:

– множинне підключення клієнтів до сервера (підключався 50 клієнтів в різних вузлах мережі, при цьому серйозного уповільнення обміну запитами не помічалося);

– обмін повідомленнями між користувачами (проводилася відправка повідомлень від усіх користувачів одному, від одного всім, а також обмін повідомленнями один одному. Всі листи були записані в базу даних і доставлялися одержувачам);

– реєстрація документа без доручення та за дорученням одному і декільком контролерам з виконавцями (всі документи потрапляли потрібним контролерам, а доручення виконавцям);

– пошук по всіх можливих реквізитам (видавалися всі необхідні документи, що відповідають критеріям запиту);

– очищення тимчасових файлів на комп'ютері-клієнті, вбудованими в

програму-клієнт засобами (файли віддалялися в штатному режимі, видалення зайнятих системою файлів відкладалося);

– друк документів (всі необхідні документи друкувалися відповідно до вимог на всіх встановлених принтерах в системі).

В процесі тестування серйозних помилок виявлено не було. Помічені можливі внутрішні збої програми MS Word в процесі реєстрації документа, які не залежать від програми-клієнта, засновані на недоліках COM-інтерфейсу. А також зареєстровані можливі збої запису документів великого обсягу в базу даних. Це пов'язано з внутрішніми налаштуваннями сервера баз даних і усувається збільшенням розміру максимального пакета для запису.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані:

1. Передбачуване число операторів – 2000.
2. Коефіцієнт складності програми - 1,25.
3. Коефіцієнт корекції програми в ході її розробки - 0,08.
4. Годинна заробітна плата програміста, грн / год – 30.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,}$$

де:

to - витрати праці на підготовку й опис поставленої задачі (приймається 30);

tu - витрати праці на дослідження алгоритму рішення задачі;

ta - витрати праці на розробку блок-схеми алгоритму;

tn - витрати праці на програмування по готовій блок-схемі;

тотл - витрати праці на налагодження програми на ЕОМ;

tд - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \times C \times (1+ p)\text{ , людино-годин,}$$

де q - передбачуване число операторів;

C - коефіцієнт складності програми;

p - коефіцієнт кореляції програми в ході її розробки.

$$Q = 2000 \cdot 1{,}25 \cdot (1 + 0{,}08) = 2700 \text{ людино-годин.}$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{QB}{(75...85)K}\text{ , людино-годин,}$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі; B=1.2 ... 1.5;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

Витрати праці на розробку алгоритму рішення задачі:

$$t_u = \frac{2700 \cdot 1{,}2}{80 \cdot 0{,}8} = 50\text{ , людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t\alpha = \frac{Q}{(20...25)K} \quad \text{людино-годин.}$$

$$t_a = \frac{2700}{20 \cdot 0{,}8} = 168 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K} \quad \text{людино-годин.}$$

$$t_n = \frac{2700}{25 \cdot 0{,}8} = 135 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

$$t_{отл} = \frac{Q}{(4...5)K} \quad \text{людино-годин.}$$

$$t_{отл} = \frac{2700}{4 \cdot 0{,}8} = 843 \text{ людино-годин.}$$

_ за умови комплексного налагодження завдання:

$$t^{к}_{отл} = 1{,}2 \cdot t_{отл};$$

$$t_{отл} = 843 * 1{,}2 = 1012$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}, \text{ людино-годин,}$$

де *tдр* - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{(15...20)K}, \text{ людино-годин.}$$

$$t_{др} = \frac{2700}{15 \cdot 0{,}8} = 225 \text{ людино-годин,}$$

tдо - трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \cdot t_{др}, \text{ людино-годин.}$$

$$tdo = 0{,}75 \cdot 255 = 168$$

$$t_{\partial} = 225 + 168 = 393 \text{ людино-годин.}$$

Отримуємо трудомісткість розробки програмного забезпечення:

$$t = 30 + 50 + 168 + 135 + 1012 + 393 = 1790 \text{ людино-годин.}$$

3.2. Розрахунок витрати на створення програмного забезпечення

Витрати на створення ПЗ *Кпо* включають витрати на заробітну плату виконавця програми *Зз/п* і витрат машинного часу, необхідного на налагодження програми на ЕОМ

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,}$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,}$$

де: *t* - загальна трудомісткість, людино-годин;

Спр - середня годинна заробітна плата програміста, грн/година

$$З_{ЗП} = 1790 \cdot 30 = 53718 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \times С_{М}, \text{ грн,}$$

де: *тотл* - трудомісткість налагодження програми на ЕОМ, год.

Смч - вартість машино-години ЕОМ, 6 грн/год.

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

$$З_{МВ} = 1012 \times 6 = 6075 \text{ грн.}$$

$$Êïï = 53718 + 6075 = 59793 \text{ ãðí.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ міс.}$$

де: *Bk* - число виконавців;

Fp - місячний фонд робочого часу (при 40 годинному робочому тижні *Fp*=176 годин).

$$T = \frac{1790}{1 \cdot 176} = 10{,}1 \text{ міс.}$$

Визначено трудомісткість розробленої інформаційної системи (1790 люд-год), проведений підрахунок вартості роботи по створенню програми (59793 грн.) та розраховано час на його створення (10,1 міс).

ВИСНОВКИ

Завданням даної кваліфікаційної роботи є розробка інформаційної системи електронного документообігу. Незважаючи на широке розмаїття готових продуктів даної області на ринку, актуальною є розробка універсальної, недорогої системи для переходу організації з паперового документообігу на електронний. До того ж використання засобів «C #» дозволяє запускати програму на будь-якій платформі, що підтримує .Net Framework без повторної компіляції програми. Також ця програма використовує безкоштовну базу даних MySQL, що позбавить організацію від додаткових витрат на комерційні програми.

Ефективність впровадження розробленої автоматизованої інформаційної системи полягає в:

– автоматизації розсилки наказів по підприємству;

– підвищення продуктивності праці відділу, за допомогою зниження навантаження на операторів;

– підвищення відмовостійкості системи, шляхом впровадження технологій баз даних.

Практичне значення виконання даної роботи полягає в розробці інформаційної системи для організації електронного документообігу підприємства, що дозволить підвищити якість та оперативність управління підприємством; об'єднає в єдиний діловодний цикл всіх структурних підрозділів організації, включаючи територіально-віддалені та забезпечить зниження трудових і тимчасових витрат і накладних витрат.

В економічному розділі були проведені обчислення трудомісткості та витрат для розробки програмного забезпечення. Для створення даної інформаційної системи знадобиться 10,1 місяця, трудомісткість розробки ПЗ – 1790 людино-годин, а витрати на її створення програмного забезпечення - 59793 грн.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Благодаров А.В., Гринченко Н.Н., Овечкин Г.В. Клиент-серверные технологии баз данных. Методические указания к лабораторным работам. РГРТУ, Рязань, 2007.

2. Боуман Дж., Эмерсон С., Дарновски М. Практическое руководство SQL; 2000. – 321 с.

3. Васвани Викрам, Полный справочник по MySQL.: пер с англ. – М.: Издательский дом «Вильямс», 2006.

4. Гарсиа-Молина Г. Системы баз данных. Полный курс / Г. Гарсиа-Молина, Дж. Ульман, Дж. Уидом. – М.: Вильямс, 2003. – 1088 с.

5. Глинских К.Т. Мировой рынок систем электронного документооборота. URL: www.jetinfo.ru/2002/8/1/article1.8.2002.html. Дата звернення: 15.01.2021.

6. Грабер М. SQL. Справочное руководство. – М.: Лори, 2007. – 644 с.

7. Грофф Дж. Энциклопедия SQL. 3-е изд. / Дж. Грофф, П. Вайнберг. – М.: Вильямс, 2003. - 896 с.

8. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.

9. Доронина Л.А., Черников Б.В. Новые информационные технологии хранения документов организации: К постановке проблемы // Делопроизводство. 2002. – №1. – с. 33-39.

10. ДСТУ 2394-94 Інформація та документація. Комплектування фонду, бібліографічний опис, аналіз документів. Терміни та визначення. – Чинний від 01.01.1995. - Київ : Держстандарт України, 1994. – 88 с.

11. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Комп’ютерні системи ” / Укладачі О.Г. Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 23с.

12. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 122 «Комп'ютерні науки» галузі знань 12 Інформаційні технології/, Л.М. Коротенко , О.С. Шевцова; Нац. гірн. ун-т. – Д : ДВНЗ НГУ, 2019. – 65 с.

13. Нейгел К., Ивьен Б. DELPHI7 для профессионалов. Пер. с англ. М.: Издательский дом «Вильямс», 2006. – 1376 с.

14. Организация документооборота URL: www.document-circulation.org.ru Дата звернення: 15.01.2021.

15. Офіційний сайт середовища розробки Visual Studio Code URL: https://code.visualstudio.com/docs Дата звернення: 15.01.2021.

16. Пахчанян А. Обзор систем электронного документооборота // Директор информационной службы. 2001. – №2.

17. Сакун Ю. Документы – в оборот // CIO. 2003. – №1.

18. Microsoft SQL Server. Эффективная работа Вишневский Алексей, серия: Эффективная работа, Изд.: Питер 2009 г.

ДОДАТОК А

КОД ПРОГРАМИ

Main.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using WpfApp1.Sections;
namespace WpfApp1
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            Home_page home_Page = new Home_page();
            Frame_main.Navigate(home_Page);
        }
```

```
private void Exit_Click(object sender, RoutedEventArgs e)
{
    this.Close();
}
private void WindowMinimize_Click(object sender, RoutedEventArgs e)
{
    this.WindowState = WindowState.Minimized;
}

private void Grid_MouseDown(object sender, MouseButtonEventArgs e)
{
    try
    {
        this.DragMove();
    }
    catch
    {
    }
}
private void bt_section_1_Click(object sender, RoutedEventArgs e)
{
    Section_1 section_1 = new Section_1();
    Frame_main.Navigate(section_1);
}
private void bt_section_2_Click(object sender, RoutedEventArgs e)
{
    Section_2 section_2 = new Section_2();
    Frame_main.Navigate(section_2);
}
private void bt_section_3_Click(object sender, RoutedEventArgs e)
```

```
{
    Section_3_theory section_3 = new Section_3_theory();
    Frame_main.Navigate(section_3);
}
private void Menu_Click(object sender, RoutedEventArgs e)
{
    if (Section_1.mark >= 5)
    {
        bt_section_2.IsEnabled = true;
        if (Section_2.mark >= 5)
        {
            bt_section_3.IsEnabled = true;
        }
        else
        {
            bt_section_3.IsEnabled = false;
        }
    }
    else
    {
        bt_section_2.IsEnabled = false;
    }
}
private void bt_introduction_Click(object sender, RoutedEventArgs e)
{
    Introduction introduction = new Introduction();
    Frame_main.Navigate(introduction);
}
private void bt_cval_Click(object sender, RoutedEventArgs e)
{
```

```
            Pracktice pracktice = new Pracktice();
            Frame_main.Navigate(pracktice);
        }
    }
}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace man2
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace m
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            Form F5 = new Form5();
            F5.Show();
            this.Hide();
        }
        private void button4_Click(object sender, EventArgs e)
        {
            Form F2 = new Form2();
            F2.Show();
            this.Hide();
        }
```

```
    private void button5_Click(object sender, EventArgs e)
    {
        Form F3 = new Form3();
        F3.Show();
        this.Hide();
    }
    private void button3_Click(object sender, EventArgs e)
    {
        Form F4 = new Form4();
        F4.Show();
        this.Hide();
    }
    private void button2_Click(object sender, EventArgs e)
    {
        Form F6 = new Form6();
        F6.Show();
        this.Hide();
    }
    private void button6_Click(object sender, EventArgs e)
    {
        Application.Exit();
    }
    private void button1_Click_1(object sender, EventArgs e)
    {
        Form F7 = new Form7();
        F7.Show();
        this.Hide();
    }

}
```

```
}
```

Form10.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace mani2
{
   public partial class Form10 : Form
   {
      public Form10()
      {
         InitializeComponent();
      }

      private void Form10_Load(object sender, EventArgs e)
      {
      }

      private void button1_Click(object sender, EventArgs e)
      {
         Form F7 = new Form7();
         F7.Show();
         this.Hide();
```

```
        }
    }
}
```

Form11.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace man2018
{
    public partial class Form11 : Form
    {
        public Form11()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            Form F7 = new Form7();
            F7.Show();
            this.Hide();
        }
    }
}
```

Form2.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace man2
{
   public partial class Form2 : Form
   {
      public Form2()
      {
         InitializeComponent();
      }
      private void Form2_Load(object sender, EventArgs e)
      {
      }
      private void button2_Click(object sender, EventArgs e)
      {
         Form F7 = new Form7();
         F7.Show();
         this.Hide();
      }
      private void button1_Click(object sender, EventArgs e)
      {
         Form F3 = new Form3();
```

```
            F3.Show();
            this.Hide();
        }
    }
}
```

Form3.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace man2
{
    public partial class Form3 : Form
    {
        public Form3()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Form F7 = new Form7();
            F7.Show();
```

```
            this.Hide();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Form F2 = new Form2();
            F2.Show();
            this.Hide();
        }
    }
}
```

Data.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.IO;
using System.Windows;
using System.Windows.Media;
using static Modeling.wElementCreater;
namespace Modeling
{      static class Data
    {       public static event Action ElementsChanged;
        //налаштвання за замовчанням. Існують тільки тут.
        public static string DataBasePathDefault { get; } =
Environment.CurrentDirectory + "\\Sources\\Elements.xml";
        //public
        public static readonly double dataScale;
        public static readonly char splitSymbol;
        public static readonly double maxSpeedRatio;
```

```
public static readonly double minSpeedRatio;
public static readonly Brush groundColor;
public static readonly Brush grassColor;
public static readonly Brush skyColor;
public static readonly Brush normalBorderBrush;
public static readonly Brush chosentBorderBrush;
public static readonly Brush guidesForceBrush;
public static readonly Brush guidesBackBrush;
public static readonly Brush promptsForeground;
public static readonly List<Element> Elements;
   //Інкапсулюються властивостями
private static string dataBasePath;
private static double minimalCalculateTime;
private static int maxFramesCount;
private static double scrollStep;
private static double minScale;
private static double translateFields;
private static double averageSpeedRatio;
   //Властивості
public static string DataBasePath
{          get
   {               return dataBasePath;
   }
   set
   {               trySaveData();
      dataBasePath = value;
      Elements.Clear();
      if (ElementsChanged != null)
         ElementsChanged();
      loadData();
```

```
    }       }
public static double MinimalCalculateTime {
    get { return minimalCalculateTime; }
    set
    {               if (value < MinTimeDefault)
            minimalCalculateTime = MinTimeDefault;
        else
            minimalCalculateTime = value;
        isSettingsSave = false;
    }       }
public static int MaxFramesCount {
    get { return maxFramesCount; }
    set
    {               if (value < MaxFramesCountDefault)
            maxFramesCount = MaxFramesCountDefault;
        else
            maxFramesCount = value;
        isSettingsSave = false;
    }       }
public static double ScrollStep
{           get { return scrollStep; }
    set
    {               if (value < 1)
            scrollStep = 1;
        else
            if (value > 10)
                scrollStep = 10;
        else
            scrollStep = value;
        isSettingsSave = false;
```

```
    }       }
public static double MinScale
{
    get { return minScale; }
    set
    {           if (value <= 0)
            minScale = MinScaleDefault;
        else
            if (value > 1)
                minScale = 1;
        else
            minScale = value;
        isSettingsSave = false;
    }       }
public static double TranslateFields
{           get { return translateFields; }
    set
    {           if (value < 0)
            translateFields = 0;
        else
            translateFields = value;
        isSettingsSave = false;
    }       }
public static double AverageSpeedRatio
{           get { return averageSpeedRatio; }
    set
    {           if (value < minSpeedRatio)
            averageSpeedRatio = minSpeedRatio;
        else
            if (value > maxSpeedRatio)
```

```
                averageSpeedRatio = maxSpeedRatio;
            else
                averageSpeedRatio = value;
            isSettingsSave = false;
        }       }
    //private
    private static readonly string[] explanationPaths;
    private static Stack<Window> windowHistory;
    private static string settingsPath;
    private static bool isLoaded;
    private static bool isDataSave { get; set; }
    private static bool isSettingsSave { get; set; }
    //TODO: абсолютний шлях до теорії.
    static Data()
    {
        splitSymbol = ' ';
        dataScale = 100;
        maxSpeedRatio = 5;
        minSpeedRatio = 0.2;
        groundColor = new SolidColorBrush( Color.FromRgb(93, 31, 0));
        grassColor = new SolidColorBrush(Colors.Green);
        skyColor = new SolidColorBrush(Color.FromRgb(0, 190, 255));
        normalBorderBrush = new SolidColorBrush(Colors.Black);
        chosentBorderBrush = new SolidColorBrush(Color.FromRgb(0,0,150));
        guidesForceBrush = new SolidColorBrush(Colors.Black);
        guidesBackBrush = new SolidColorBrush(Colors.Blue);
        promptsForeground = new SolidColorBrush(Colors.BlanchedAlmond);
        Elements = new List<Element>();
        windowHistory = new Stack<Window>();
```

```
        explanationPaths =new string[]{ Environment.CurrentDirectory +
"\\Sources\\Explanations\\"+wExplanation.ExplanationType.MenuMain.ToString()+"
.html",
                                        Environment.CurrentDirectory +
"\\Sources\\Explanations\\"+wExplanation.ExplanationType.MenuModeling.ToString
()+".html",
                                        Environment.CurrentDirectory +
"\\Sources\\Explanations\\"+wExplanation.ExplanationType.ElementCreater.ToStrin
g()+".html",
                                        Environment.CurrentDirectory +
"\\Sources\\Explanations\\"+wExplanation.ExplanationType.Modeling.ToString()+".
html"};
    }
    //Ініціалізація. Повинна бути виконана при старті програми.
    public static void initialize()
    {           try
       {              /
          using (StreamReader sr = new StreamReader(settingsPath))
          {                  Dictionary<String, String> dat = new Dictionary<String,
String>();
             bool isCorrect = true;
             string[] temp = sr.ReadLine().Split(' ',':');
             dat.Add(temp[0], temp[temp.Length-1]);
             temp = sr.ReadLine().Split(' ', ':');
             dat.Add(temp[0], temp[temp.Length - 1]);
             temp = sr.ReadLine().Split(' ', ':');
             dat.Add(temp[0], temp[temp.Length - 1]);
             temp = sr.ReadLine().Split(' ', ':');
             dat.Add(temp[0], temp[temp.Length - 1]);
             temp = sr.ReadLine().Split(' ', ':');
```

```
            dat.Add(temp[0], temp[temp.Length - 1]);
            temp = sr.ReadLine().Split(' ', ':');
            dat.Add(temp[0], temp[temp.Length - 1]);
            temp = sr.ReadLine().Split(' ', ':');
            dat.Add(temp[0], temp[temp.Length - 1]);
            MinimalCalculateTime = double.Parse(dat["MinimalCalculateTime"]);
            if (double.Parse(dat["MinimalCalculateTime"]) !=
MinimalCalculateTime)
               isCorrect = false;
            MaxFramesCount = int.Parse(dat["MaxFramesCount"]);
            if (double.Parse(dat["MaxFramesCount"]) != MaxFramesCount)
               isCorrect = false;
            MinScale = double.Parse(dat["MinScale"]);
            if (double.Parse(dat["MinScale"]) != MinScale)
               isCorrect = false;
            TranslateFields = double.Parse(dat["TranslateFields"]);
            if (double.Parse(dat["TranslateFields"]) != TranslateFields)
               isCorrect = false;
            ScrollStep = double.Parse(dat["ScrollStep"]);
            if (double.Parse(dat["ScrollStep"]) != ScrollStep)
               isCorrect = false;
            AverageSpeedRatio = double.Parse(dat["AverageSpeedRatio"]);
            if (double.Parse(dat["AverageSpeedRatio"]) != AverageSpeedRatio)
               isCorrect = false;
                        if (dat["DataBasePath"].StartsWith("\\"))
               temp[0] = Environment.CurrentDirectory + dat["DataBasePath"];
            DataBasePath = temp[0];
            if (!isDataSave)
            {                   DataBasePath = DataBasePathDefault;
               isCorrect = false;
```

```
            }
            //Якщо досі isCorrect - true, то налаштування у файлі коректні
            if (isCorrect)
               isSettingsSave = true;
            isDataSave = true;            }                    }
        catch (Exception e)
        {            if (e is IndexOutOfRangeException|| e is FormatException || e is
OverflowException || e is ArgumentNullException)
            MessageBox.Show("Файл: "+settingsPath+ " має не вірний формат.
else
            MessageBox.Show(e.Message, "Помилка", MessageBoxButton.OK,
MessageBoxImage.Error);
           //Заносимо дефолтні значення
           MinimalCalculateTime = MinTimeDefault;
           MaxFramesCount = MaxFramesCountDefault;
           MinScale = MinScaleDefault;
           TranslateFields = TranslateFieldsDefault;
           ScrollStep = TranslateFieldsDefault;
           AverageSpeedRatio = AverageSpeedRatioDefault;
           DataBasePath = DataBasePathDefault;
        }
        finally
        {            //Програма завантажилась
           isLoaded = true;
        }       }
     private static void loadData()
     {          try
        {            using (FileStream fs = new FileStream(DataBasePath,
FileMode.Open))
           {                DataSet ds = new DataSet();
```

```
            ds.ReadXml(fs, XmlReadMode.InferTypedSchema);
            fs.Close();
            ElementCreater.createElements(ds.Tables[0]);//завжди одна таблиця
            isDataSave = true;
        }          }
      catch
              }       }
    {          if (!isDataSave && isLoaded)
            saveData();
    }
    public static void trySaveSettings()//зберегти налаштування з
підтвердженням
    {          if (!isSettingsSave )
                    saveSettings();
    }
    public static void saveChange()//зберегти усе без підтверджень
    {          saveData();
      saveSettings();
    }
    private static void saveSettings()
    {          using (StreamWriter sw = new StreamWriter(settingsPath))
      {              sw.WriteLine("MinimalCalculateTime: " +
minimalCalculateTime.ToString());
            sw.WriteLine("MaxFramesCount: " + maxFramesCount.ToString());
            sw.WriteLine("MinScale: " + MinScale.ToString());
            sw.WriteLine("TranslateFields: " + TranslateFields.ToString());
            sw.WriteLine("ScrollStep: " + ScrollStep.ToString());
            sw.WriteLine("AverageSpeedRatio: " + AverageSpeedRatio.ToString());
            if(dataBasePath.Contains(Environment.CurrentDirectory))
```

```
                sw.WriteLine("DataBasePath: " +
DataBasePath.Replace(Environment.CurrentDirectory,""));
            else
                sw.WriteLine("DataBasePath: " + DataBasePath);
            isSettingsSave = true;
        }       }
```

Form5.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace man2
{
   public partial class Form3 : Form
   {
      public Form3()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {
         Form F7 = new Form7();
```

```
            F7.Show();
            this.Hide();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Form F2 = new Form2();
            F2.Show();
            this.Hide();
        }
    }
}
```

Form9.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace man2018
{
    public partial class Form11 : Form
    {
        public Form11()
        {
            InitializeComponent();
```

```
        }
        private void button1_Click(object sender, EventArgs e)
        {
            Form F7 = new Form7();
            F7.Show();
            this.Hide();
        }
    }
}
```

Form2.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace man2
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }
        private void Form2_Load(object sender, EventArgs e)
        {
        }
```

```
        private void button2_Click(object sender, EventArgs e)
        {
            Form F7 = new Form7();
            F7.Show();
            this.Hide();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            Form F3 = new Form3();
            F3.Show();
            this.Hide();
        }
    }
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_ doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_.ppt	Презентація кваліфікаційної роботи