Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

"

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Гриценко Олексій Ігорович*
(ПІБ)

академічної групи *122-18-1*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка програмного забезпечення оптимізації виконання офісних задач засобами ADO.Net, MySQL, WPF*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*проф. Бердник М.Г.*			
розділів:				
спеціальний	*проф. Бердник М.Г.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент	*доц. Шедловський І.А.*			
Нормоконтролер	*Доц. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

"

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

_____ _____

ЗАВДАННЯ
на кваліфікаційну роботу
бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-18-1* (група) *Гриценко Олексій Ігорович* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка програмного забезпечення оптимізації виконання офісних задач засобами ADO.Net, MySQL, WPF*

затверджена наказом ректора НТУ «ДП» від «18» травня 2022 р. № 268-с

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проектно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***17.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ *проф. Бердник М.Г.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Гриценко О.І.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *17.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 82 с., 22 рис., 3 дод., 21 джерел.

Об’єкт розробки: універсальний застосунок «Електронний офіс»

Мета кваліфікаційної роботи: розробка універсального застосунку «Електронний офіс» для об'єднання всіх офісних процесів по обробці та обміну інформацією, в один зручний інтерфейс і максимально автоматизувати їх.

У вступі пояснюється призначення застосунку та галузі, в якій він використовується, підтверджується його актуальність та описується функціонал.

При написанні першої частини визначається мета виконаної роботи та план роботи, складається вимоги до реалізації програми, для визначення технології, яку використовує програмне забезпечення.

Під час написання другої частини проводиться аналіз фактичних рішень, визначається платформа розробки, проектування та розробка програмного забезпечення, визначається структура операційного середовища, вхідні дані та результати.

В економічному розділі визначено трудомісткість розробленого застосунка, проведений підрахунок вартості роботи по створенню застосунка та розраховано час на його створення.

Практичне значення полягає у створенні настільного застосунку, що надає користувачам доступ до документарної бази даних, що надає компанія та переліку працівників, яким можливо надати доступ до конкретних документів

Актуальність застосунку визначається спрощенням процесів передачі інформації, збільшенням попиту на дистанційну взаємодію між працівниками, та прискоренням процесів узгодження документів.

Список ключових слів: Універсальний застосунок, “Електронний офіс”

ABSTRACT

Explanatory note: 82 pp., 22 figs., 3 appendices, 21 sources.

Object of development: universal application "Electronic office"

The purpose of the qualification work: development of a universal application "Electronic Office" to combine all office processes for processing and exchanging information in one user-friendly interface and automate them as much as possible.

The introduction explains the purpose of the application and the industry in which it is used, confirms its relevance and describes the functionality.

When writing the first part, the purpose of the work performed and the work plan are determined, the requirements for the implementation of the program are developed to determine the technology used by the software.

During the writing of the second part, the analysis of actual solutions is performed, the platform of software development, design and development is determined, the structure of the operating environment, input data and results are determined.

The economic section identifies the complexity of the developed application,
the calculation of the cost of work on the creation of the application and the time for its creation.

Of practical importance is the creation of a desktop application that gives users access to the documentary database provided by the company and a list of employees who can be granted access to specific documents

The relevance of the application is determined by the simplification of information transfer processes, increasing the demand for remote interaction between employees, and accelerating the process of reconciling documents.

Keyword list: Universal Application, e-Office

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

DA - Desktop Application;

DB - Database;

ОС - операційна система;

ПК - персональний комп’ютер;

ПЗ - програмне забезпечення;

FE - FrontEnd;

IT – інформаційні технології;

BE - BackEnd.

Вступ

Офіс - являє собою нежитлову територію, яка належить суб’єкту господарювання, де розташований його орган управління, і має конкретну адресу, за якою можна здійснювати поштовий зв'язок. В головному офісі знаходиться керівництво та юридичне представництво компанії.

За необхідністю офіси складаються з декілька частин. В першій частини розміщується приймальні, зони очікування, кімнати для переговорів тощо. В другій частині розташовуються основні робочі підрозділи, що забезпечують аналітичну, фінансову, інформаційну та інші діяльності. Але для малих офісів притаманне наявність лише єдиної частини, для розташування працівників спеціалізованого напрямку.

Задля об'єднання всіх офісних процесів по обробці та обміну інформацією, в один зручний інтерфейс і максимально автоматизувати їх, розроблено систему «Електронний офіс».

Електронний офіс - система всебічного використання в управлінській діяльності засобів обчислювальної техніки і телекомунікацій.

Система електронного офісу зосереджена на об’єднанні інформаційних процесів в єдину систему контролю та автоматизацію організаційних процесів на підприємстві за допомогою обчислювальних засобів. Завдяки цій системі, працівники компанії зосереджують ресурси на вирішені необхідних задач.

Дану систему можливо використовувати як із робочого приміщення офісу у локальній мережі, доповнюючи існуючу систему, так із хмарного середовища, що дозволяє не зосереджуватися на місцезнаходженні працівників, а на їх якості, долучаючи до процесу з інших територіальних розташувань.

Тому проблема, розглянута в даній кваліфікаційній роботі, є актуальною та має широке практичне значення.

Метою даної роботи є розробка системи електронного документування обліку технічного стану об’єктів залізниці.

Для досягнення поставленої мети необхідно вирішити основні завдання:

- аналіз організації робіт з огляду дистанції колії;
- уточнення вимог до процесу документування оглядів;
- розробка алгоритму, бази даних і реалізації програми.

У відповідності до проведеного аналізу поставлені основні функціональні задачі перед системою:

- прискорення часу оформлення звіту про проведений технічний огляд об’єктів залізниці;
- зниження ризиків втрати або псування документів;
- збільшення ефективності ведення обліку несправностей;
- прискорення часу формування звіту про необхідний ремонт.

В цілому, система забезпечує підтримку підвищення якості шляхів залізничного господарства і значно полегшує складання акта комісійного огляду. Розроблену інформаційну систему можуть використовувати всі оператори оглядів технічного стану колії, що задіяні в сфері колійного господарства.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Загальні відомості з предметної галузі

Реалізація інформаційних діяльностей в організаційному процесі управління в контексті автоматизованих систем вимагає розгляду методів моделей «електронного офісу», розроблених наприкінці 80-х років. Електронний офіс організує робочі процеси груп користувачів над спільним рішенням важкої розподіленої роботи по комп'ютерних мережах через засоби обчислювальної техніки.

Як правило, відомі наступні етапи розвитку концепції електронного офісу:

- електронні засоби опрацювання текстів;
- локальні комп’ютерні мережі і автоматизація робіт з документаційного забезпечення управління;
- розвиток телекомунікаційних систем і автоматизовані робочі місця персоналу офісу;
- електронні архіви і сховища даних;
- регіональні і глобальні комп’ютерні мережі.

В таблиці 1 наведено приклади інформаційного обміну та застосування їх в підприємстві.

Вид інформаційного обміну	Застосування
Обмін між організацією та зовнішнім середовищем	- взаємодія з громадянами, іншими органами державного управління та сторонніми організаціями з метою реалізації функцій державного управління
Міжрівневий (вертикальний) обмін інформацією в організації	- низхідні потоки інформації, якими повідомляють підлеглим про поточні завдання, конкретні доручення, зміну пріоритетів та ін.; - висхідні потоки інформації – звіти про виконання завдань, пропозиції з удосконалення технології та ін., за допомогою яких керівництво інформують про поточні та можливі проблеми, про можливі варіанти рішень.
Горизонтальний обмін інформацією	- наради керівників суміжних підрозділів, задіяних у виконанні спільних завдань; - наради керівників підрозділів, які мають - схожі виробничі завдання; - робота у межах робочих груп (управління проектом).
Неформальний обмін інформацією	- обговорення виробничих питань під час неформальних зустрічей (під час обідньої перерви, святкових заходів та ін..); - чутки, основною причиною яких є дефіцит офіційної інформації.

Таблиця 1

Інформаційне середовище - сукупність технічних та програмних засобів для зберігання, обробки, поширення інформації, а також політичної, економічної та культурної реалізації інформаційних систем.У діяльності будь-якої організації важливе місце займає обробка документів, які надані зі зовнішніх ресурсів, узгодження всередині організації, затвердження, передача наступному персоналу, моніторингу виконання, вести роботи з довідкою, збереження. Організація роботи з документації важлива частина процесів управління та прийняття рішень, які значною мірою впливають на ефективність, економічність та надійність інституційного управлінського персоналу, культури роботи керівництво та якісний персонал. Можна визначити кілька типів функціональної взаємодії. Найпростіша модель – від одного до одного (рис 1.1).

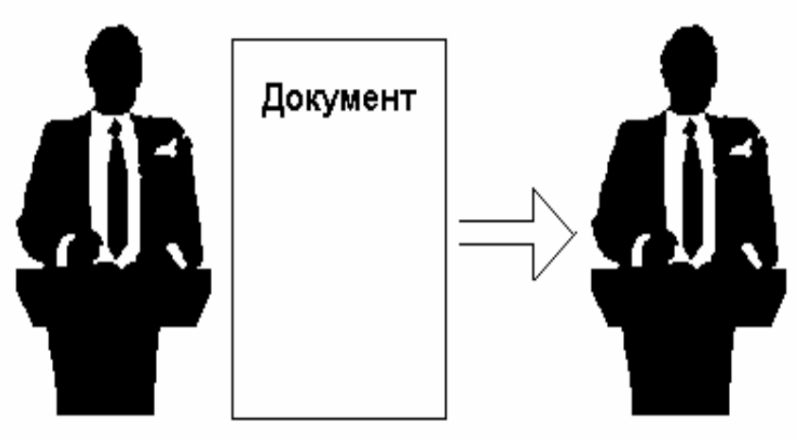

Рис 1.1 Найпростіша функціональна модель використання документа

Наступна функціональна модель–від одного до багатьох(рис 1.2)

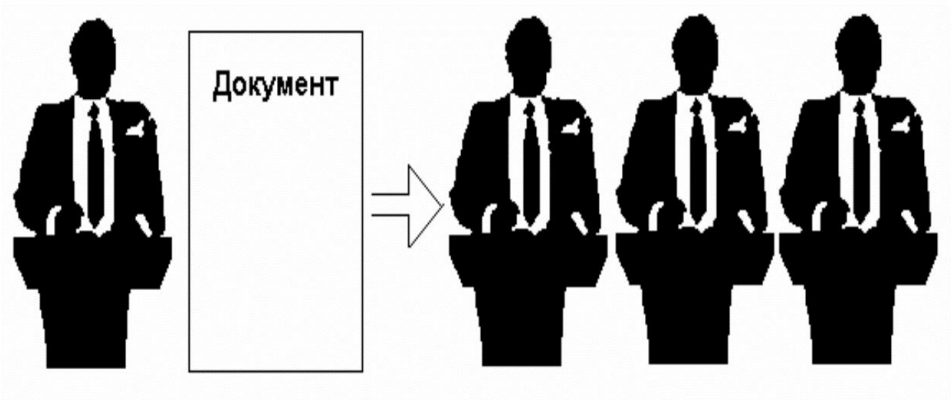

Рис 1.2 Функціональна модель використання документа: від одного до багатьох

Ви можете змоделювати приклади, збільшуючи кількість документів і адміністраторів. Однак як найважливішу особливість документа ми

наголошуємо на його функції, тобто на можливості використовувати один і той самий документ у багатьох функціях. Ця стаття дозволяє зберегти ряд типів документації в бізнесі. У той же час ця властивість була застосована до основної функції системи управління – контролю

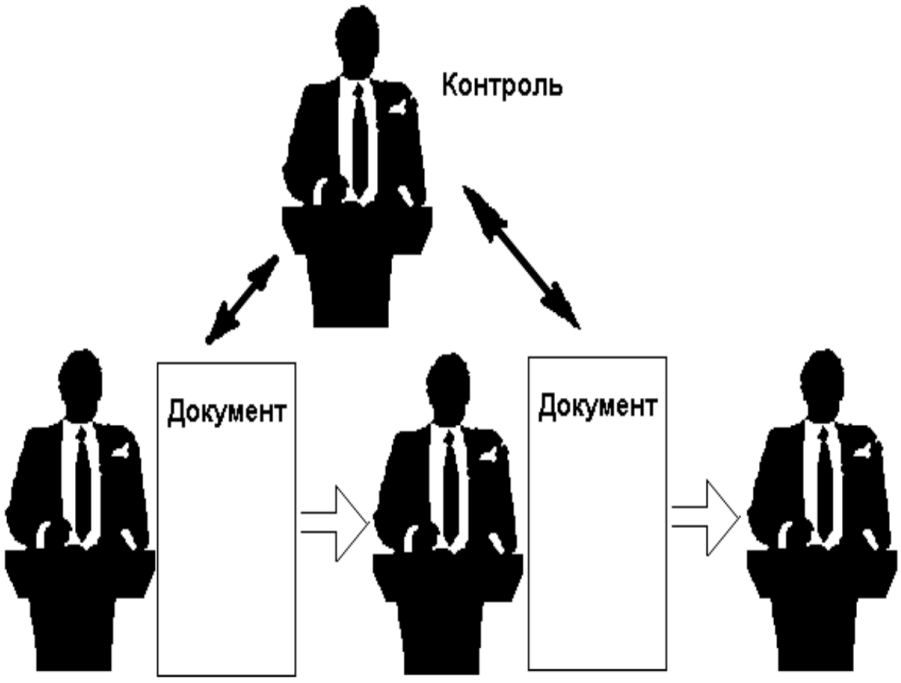

Рис 1.3 Контроль - додаткова функція обробки документа

1.2 Призначення розробки та галузь застосування

Електронний офіс відповідає за основні організаційні та адміністративні обов'язки управління підприємством загалом.

- реалізація можливості ефективно підтримувати і розвивати зв’язки з партнерами, успішно пристосуватися до швидкозмінюваної економічної ситуації;
- включення фірми до інформаційних структур ринкової економіки країни та світу, доступ до комерційних баз даних, проведення електронного маркетингу, рекламних та інформаційних заходів;
- координація діяльності всередині і зовні організації;

- допомога у виробленні і прийнятті ефективних рішень;
- виключення затримок і помилок при опрацюванні інформації, документів.

1.3 Підстава для розробки

Підставами для виконання кваліфікаційної роботи є:

- освітня програма 122 «Комп’ютерні науки»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» №268-с від 18.05.2022р.;
- завдання на кваліфікаційну роботу на тему: “розробка універсального застосунку «Електронний офіс» для об'єднання всіх офісних процесів по обробці та обміну інформацію, в один зручний інтерфейс і максимально автоматизувати їх.”

1.4. Постановка завдання

Завданням кваліфікаційної роботи є розробка «Електронний офіс» засобами ADO.Net, MySQL, WPF для об'єднання всіх офісних процесів по обробці та обміну інформацією, в один зручний інтерфейс і максимально автоматизувати їх.

Програмне забезпечення призначене для полегшення та прискорення взаємодій із обігом інформації всередині підприємства

Функціональні можливості:

1. Реєстрація
2. Зареєструвати працівника в єдину базу
3. Увійти в систему та використати доступний функціонал
4. Дізнатися основні контактні дані адміністрації

5. Переглянути та відредагувати свій профіль
6. Переглянути профілі інших працівників
7. Створити та опублікувати погодження документа
8. Побачити бюджет та всі витрати

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Система повинна бути реалізована як desktop application і працювати у системи Microsoft Windows, також програма повинна мати адаптивний інтерфейс для коректного відображення на різних технічних засобах

Стек технологій:
Фронтенд: WPF;
Бекенд: ADO.Net/MySQL.

1.5.2. Вимоги до інформаційної безпеки

Кожен користувач повинен пройти авторизацію і аутентифікацію перш ніж отримати доступ до функціоналу систему. Авторизація має здійснюватися за допомогою електронної пошти та паролю. У базі даних паролі мають зберігатися у зашифрованому вигляді. Усі поля вводу систему повинні мати функції перевірки даних на валідність. При вводі у поля некоректних даних – система повинна генерувати помилку.

1.5.3 Вимоги до складу та параметрів технічних засобів

Система реалізована в якості desktop application. Вся взаємодія відбувається в середині програми та не потребує зовнішніх зв'язків, або потребує мережу інтернет, якщо доступ до інформації налаштовано на зовнішні мережі.

Для локального використання програми мінімальні параметри для технічного засобу:

- процесор класу Intel(R) Core(TM) i3-2100 CPU @ 3.1GHz ;
- оперативна пам’ять: 4 ГБ;
- дисковий простір 1 ГБ;
- маніпулятор "миша";
- клавіатура.

1.5.4 Вимоги до інформаційної та програмної сумісності

Система має бути реалізована за допомогою мови програмування ADO.Net(бекенд). Фронтенд(інтерфест) має бути створений на базі фреймворка WPF. В якості бази даних буде використана реляційна MySQL. Система повинна працювати в системі Microsoft Windows

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Призначення додатку є прискорення офісних процесів узгодження документів

Функціонал, що реалізує застосунок:

- доступне та просте середовище для взаємодії зі документообігом компанії ;
- контактування працівників фірми з іншими працівниками та адміністрацією;
- швидке створення документів;
- швидке узгодження документообігу.

2.2. Опис застосованих математичних методів

Оскільки особливості робочого місця не передбачають використання арифметичних методів, при розробці цього проекту використання арифметичних методів не передбачається.

2.3. Опис використаних технологій та мов програмування

Зовнішній вигляд було розроблено за допомогою бібліотеки WPF.

Додаток реалізується на мові програмування C#, з використання розвинутої надбудови ADO.Net. Для взаємодії з базою даних, розроблену у вигляді реляційної бази даних MySQL, було використано framework Entity Framework.

Windows Presentation Foundation (WPF) - Аналог WinForms, розробник клієнтського програмного забезпечення Windows і технологія

спілкування з користувачами. За допомогою цієї системи можливе швидке будування функціонального інтерфейсу. За допомогою вбудованих систем розробка та прив’язка елементів UI до Backend частини проекту відбувається у декілька дотиків

ADO.NET (ActiveX Data Object для .NET) - На відміну від технології ADO, яка спеціально розроблена для програм, підключених до сервера, ADO.NET більше зосереджується на роботі в автономному режимі з використанням об’єктів DataSet. Дані об’єкти є локальними копіями взаємопов'язаними таблиць, кожна з котрих має в собі набір строк та стовпців.

MySQL - це безкоштовна система керування зв'язком, розроблена «TcX» для прискорення обробки великих даних. Ця безкоштовна система управління даними (СУБД) впроваджується як альтернатива комерційним системам. MySQL спочатку був схожий на mSQL, але з часом він розширився, і тепер MySQL є однією з найпоширеніших форм керування даними. В основному він використовується для створення потужних веб-сторінок та настільних додатків оскільки має відмінну підтримку різними мовами програмування.

C# (також відомий як C-Sharp) — це мова програмування, зосереджена на об’єктах і безпечна система запису для платформи NET. Під редакцією Андерса Галесберга, Скотта Вілтамута та Пітера Голда під егідою Microsoft Research (керує Microsoft).

Синтаксис C# розташований поруч із C++ і Java. Мова має неабстрактну, типову підтримку полімеризації, полімеризації, перевантаження, індикаторів активності класу, символів, подій, властивостей, зовнішніх елементів, коментарів у форматі XML. Вітаючи більшість своїх попередників - C++, Object Pascal, Module і Smalltalk - C#, з точки зору їх використання, виключає інші моделі, які виявилися проблематичними в розробці програмного забезпечення, такі як C#, на

відміну від C++, не включає застарілий функціонал множинного наслідування класів.

Серед факторів, які сприяють популярності мови C#, є такі:

- підтримка усіх сучасних систем на основі Windows;
- повна інтеграція з системою розробки настільних застосунків WPF;
- велика кількість гнучких налаштувань;
- провірена часом система з великої кількістю літератури.

До недоліків мови C# можна віднести наступне:

- сувора типізація потребує вправного використання даних;
- система JIT-компіляції потребує багато ресурсів під час першого запуску програми;
- прив’язаність до системи Windows.

Найвідоміші системи для розробки:

- WinForms Application;
- WPF;
- ASP.NET;
- .NET Core;
- Xamarin;
- Unity.

Windows Forms Application та WPF – системи для розробки настільних застосунків у середовищі Windows. Windows Forms є більш старою системою, розробленою під старші версії Windows.

ASP.NET - система для розробки веб-застосунків. Має весь необхідний набір інструментів, бібліотек, розширень тощо для впровадження гнучкого, безпечного та легкого до застосування створення веб-застосунків

.NET Core – на основі .Net був побудований новий набір бібліотек для гнучкого вибору систем. Впроваджує доступ до розробки застосунків на

відкритій операційній системі Linux, в основі якої лежать принципи open source

Xamarin – система для розробки мобільних застосунків яка за останні роки розвинулась до конкурентоспроможної на ринку попиту технологією.

Unity – одна з найвідоміших систем для розробки ігрових рішень. За свою простоту в використанні, яку можна процитувати фразою: «Easy to learn, hard to master», що в перекладі означає легке вивчення, проте важко оволодіти, породила як найвідоміші хіти, так і нікому не необхідні продукти

2.4. Опис структури системи та алгоритмів її функціонування

Під час аналізу вхідного дизайну та структури даних продукту було досягнуто мети додатку.

Структурі застосунку притаманний мінімалістичній дизайн для зменшення факторів відволікання працівника від не необхідного функціоналу. Указаний мінімалістичній дизайн відображається в візуальному вигляді проекту для працівника(рис. 2.3) виглядає наступним чином:

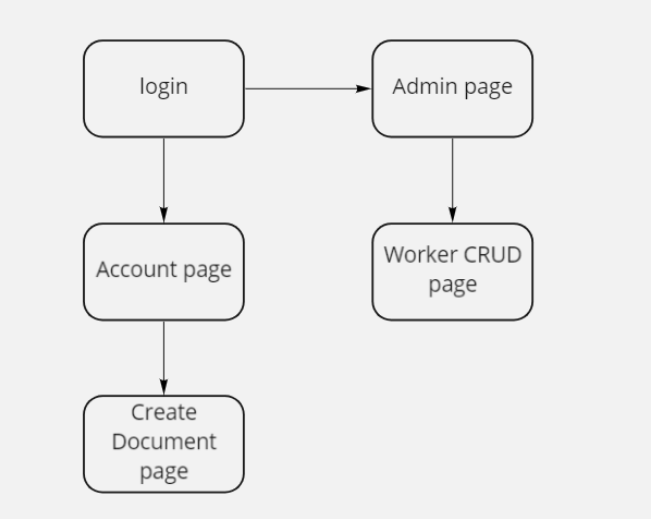

Рис 2.1 Структура настільного додатку компанії «Корса»

Сторінка Login:

- поля для входу у застосунок.

Сторінка Account:

- інформація о працівнику;
- набір документів прив’язаних до працівника;
- функціонал підписання документу;
- функціонал створення документу.

Сторінка Create Document:

- поля для створення документу;
- поля вибору працівників задля підпису .

Сторінка Admin:

- інформація о працівнику;
- поля для створення нового аккаунту працівника.

Сторінка Worker CRUD:

- має необхідний функціонал для операціями над аккаунтами працівників.

Важливо підкреслити, що існують певні правила, наступні можна створити належні умови для використання програмного забезпечення, де здійснюється процес. Хороше програмне забезпечення повинно мати конкретну функцію залежно від обсягу та розміру бізнес-функцій, які компанії повинні вирішувати, та додатків; якісно зроблений і привабливий інтерфейс; для задоволення потреб зручності його використання

На першій сторінці, працівник може увійти у свій аккаунт, системою був заздалегідь був створений аккаунт адміністрації, тому реєстрацію було виключено з функціоналу. А створення нових акаунтів відбувається в середині застосунку користувачем з правами «Адміністратор»

Наступна сторінка змінюється в залежності від посади аунтифікаційного користувача. При аутентифікації адміністрації з’являється сторінка адміністратора. На цій сторінці є необхідний функціонал для взаємодії з аккаунтами працівників.

Якщо проводиться аутентифікація користувача з іншими правами доступу, то він потрапляє на сторінку працівника, на цій сторінці, що також являється аккаунтом користувача, працівник може побачити основні дані

про нього, а головне список документів, з повною інформацією про документ та з позначкою, яка відображає, чи є документ підписаним.

Задля кращого досвіду користування для створення документів була створена додаткова сторінка, з необхідним списком полів для створення документів, з яких головним пунктом є список всіх працівників, яким працівник може призначити новостворений документ.

З цією ж метою було створено сторінку для реєстрації нових користувачів з адміністративного аккаунту. За зразком створення документу, на сторінці вказані всі необхідні поля, та впроваджені всі необхідні засоби правильності вводу даних.

Під час використання додатку локально на комп’ютері, слід тримати всі пов’язані між собою файли в одній папці, що відображатиме структуру файлів застосунку, який буде створено остаточно.

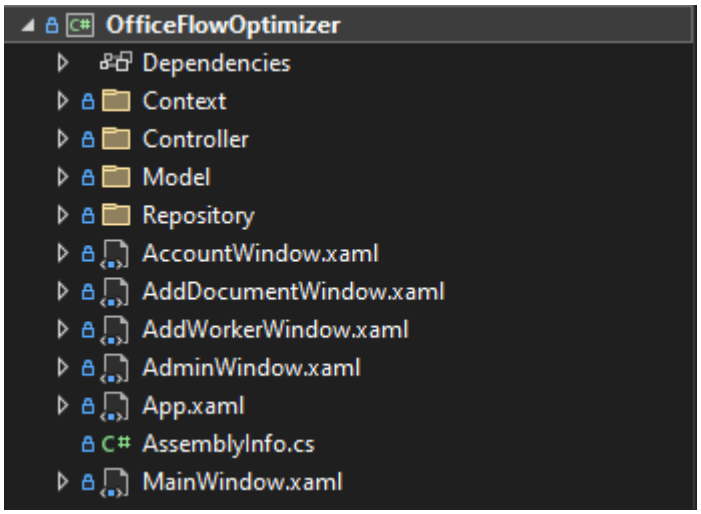

Рис. 2.2. Структура файлової системи

2.5. Обґрунтування та організація вхідних та вихідних даних програми

До вхідних даних можна віднести:

- логін;
- дані о документах;
- дані о працівниках.

До вихідних даних можна віднести:

- сторінки додатку;
- дані о доступних користувачу документах;
- дані о зареєстрованих працівниках.

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

Для серверних технічних засобів рекомендується конфігурація, що забезпечує безперервний доступ до бази даних:

- процесор класу Intel(R) Core(TM) i5-2100 CPU @ 3.1GHz ;
- оперативна пам’ять: 4 ГБ;
- жорстких диск 200 ГБ;
- доступ до мережі Internet;
- маніпулятор "миша";
- клавіатура.

Ці технічні характеристики є лише рекомендованими, тобто якщо технічні засоби не менше зазначених, розроблене програмне забезпечення має працювати відповідно до певних вимог щодо надійності, швидкості обробки даних та безпеки, встановлених замовником.

2.6.2. Використані програмні засоби

Зовнішній вигляд було розроблено за допомогою бібліотеки WPF.

Додаток реалізується на мові програмування C#, з використання розвинутої надбудови ADO.Net. Для взаємодії з базою даних, розроблену у вигляді реляційної бази даних MySQL, було використано framework Entity Framework.

В якості комерційної IDE використовувався редактор коду Microsoft Visual Studio з безкоштовною некомерційною ліцензією, особистого використання. Програмними та технічними засобами для клієнта, бажаними у використанні виступає персональний комп’ютер із системою Microsoft Windows

2.6.3. Виклик та завантаження програми

Для виклику та завантаження необхідно виконати наступний алгоритм дії:

- придбати серверне обладнання та настроїти маршрутизацію до бази даних;
- придбати обладнання на системі Microsoft Windows для основної програми;
- встановити настільний застосунок.

2.6.4. Опис інтерфейсу користувача

Застосунок складається із 5 сторінок вже зазначених раніше. Запускаючи застосунок, працівник спостерігаю сторінку аутентифікації(рис 2.3). Після вводу даних для аутентифікації, відбувається маршрутизація у наступний блок

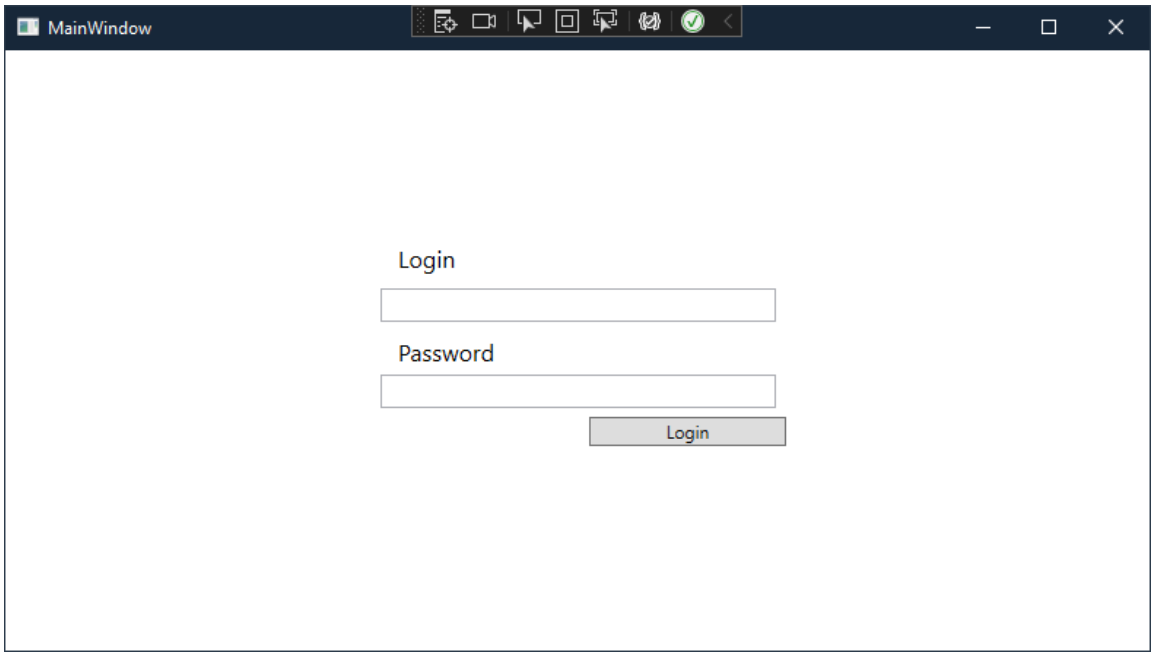

Рис 2.3 Сторінка Login

Працівники із доступом нижче «Адміністратор» потрапляє на сторінку Account(рис 2.4). Де може спостерігати список свої документів, підписати їх та додати новий

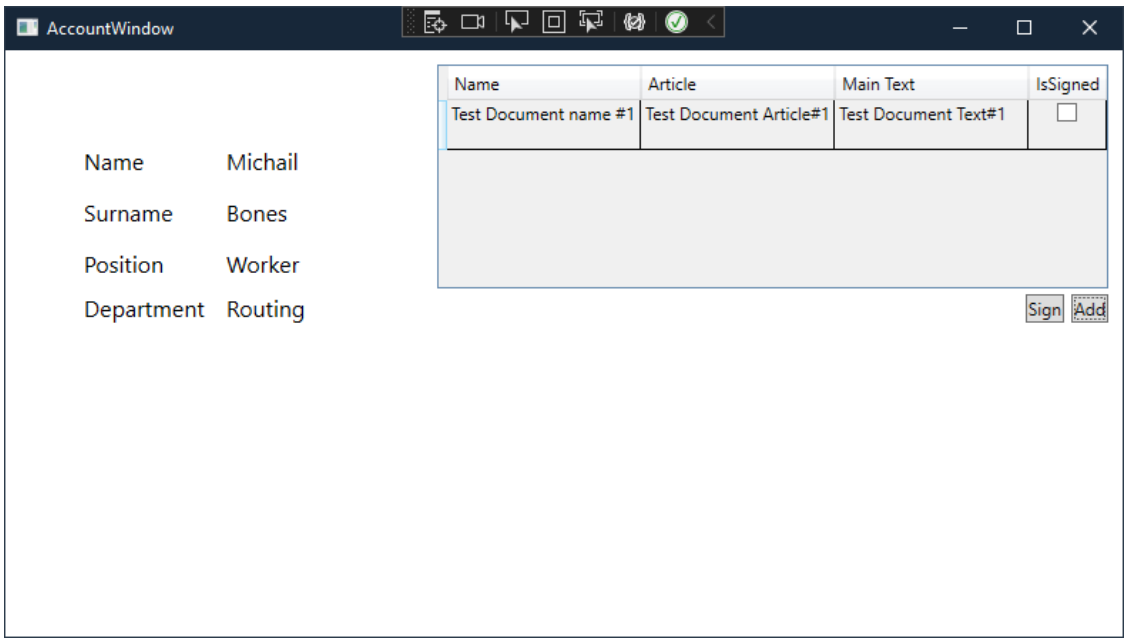

Рис 2.4 Сторінка Account

Демонстрація вигляду підписаного документу у застосунку(рис 2.5)

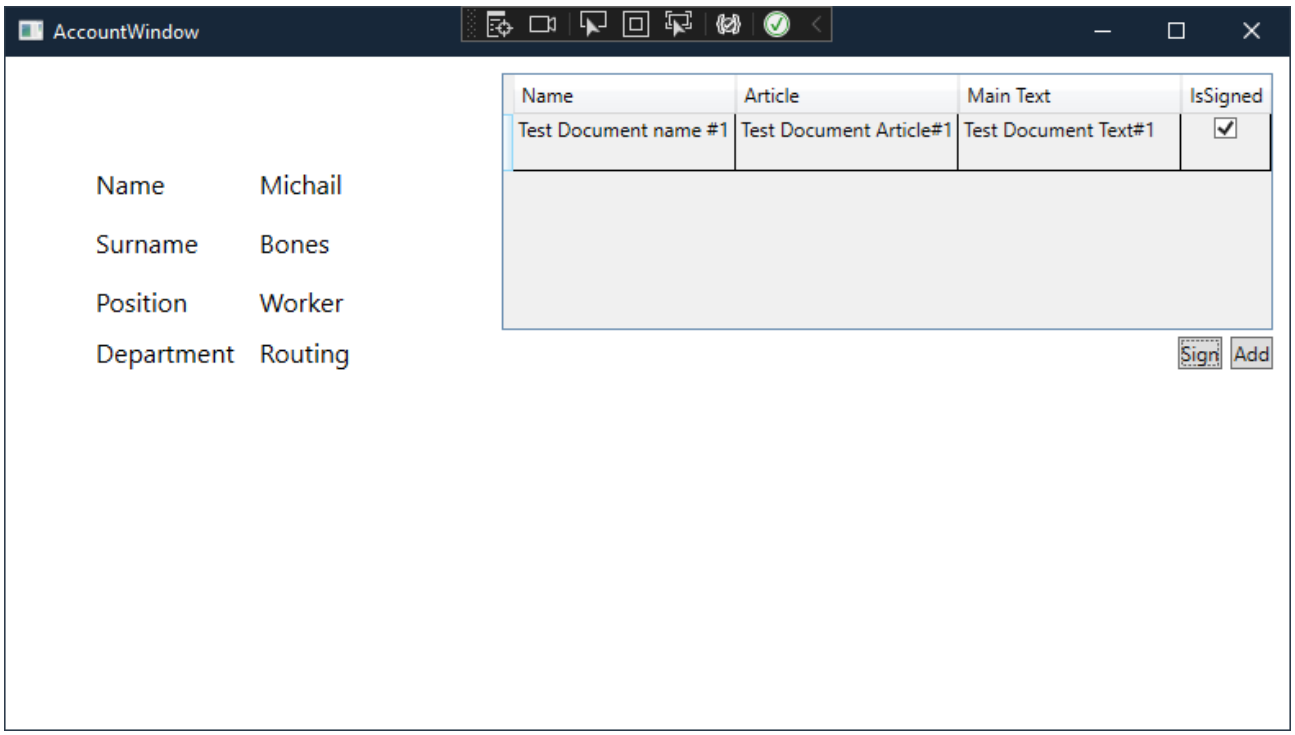

Рис 2.5 підпис документа

Для створення нового документу працівник натискає кнопку «Add» та потрапляє на відповідну сторінку(рис 2.6)

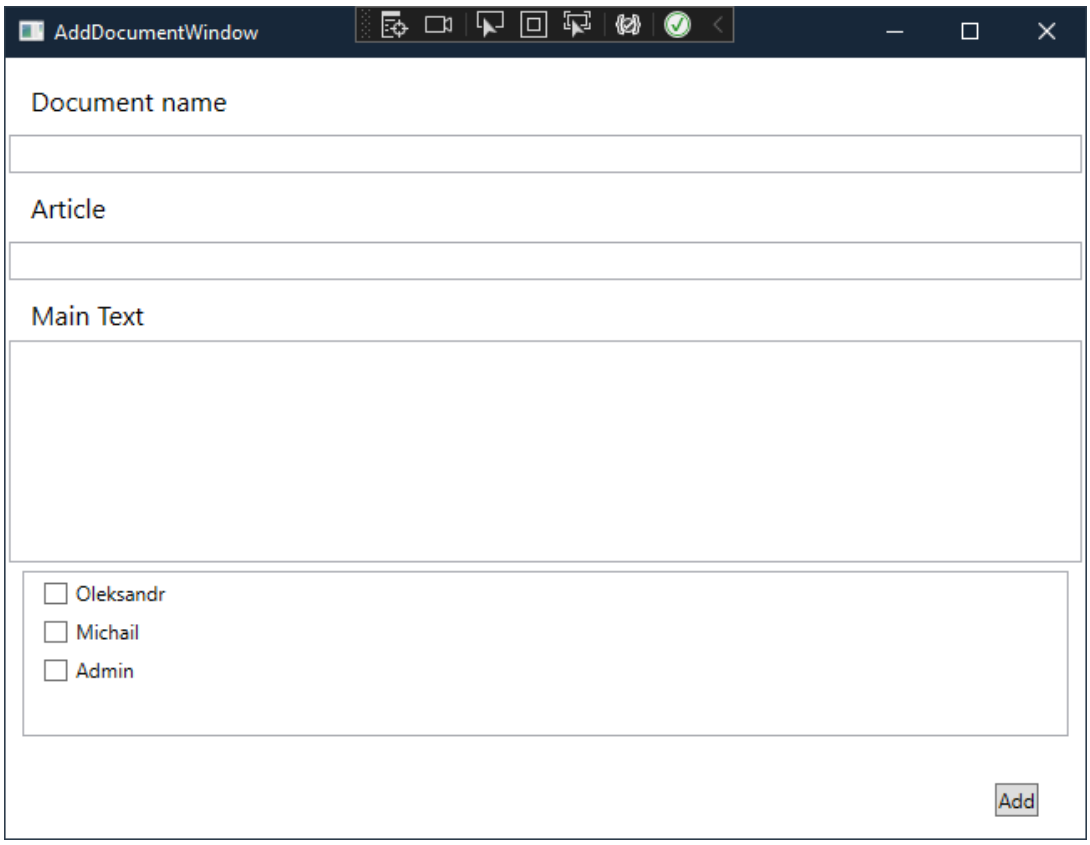

Рис 2.6 Сторінка Створення документу

Після створення документу, даний документ відобразиться у кожного працівника відміченого у полі з працівниками

Працівник з правами «Адміністратор» відповідно буде маршрутизован до сторінки адміністрації(рис 2.7)

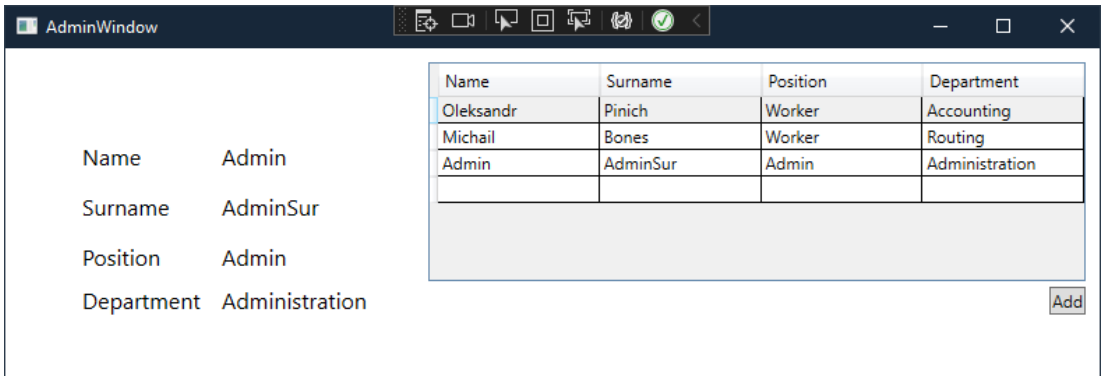

Рис 2.7 Сторінка Адміністрації

Для реєстрації нового працівника адміністратор натискає кнопку «Add», та переходить до сторінки створення нового працівника(рис 2.8)

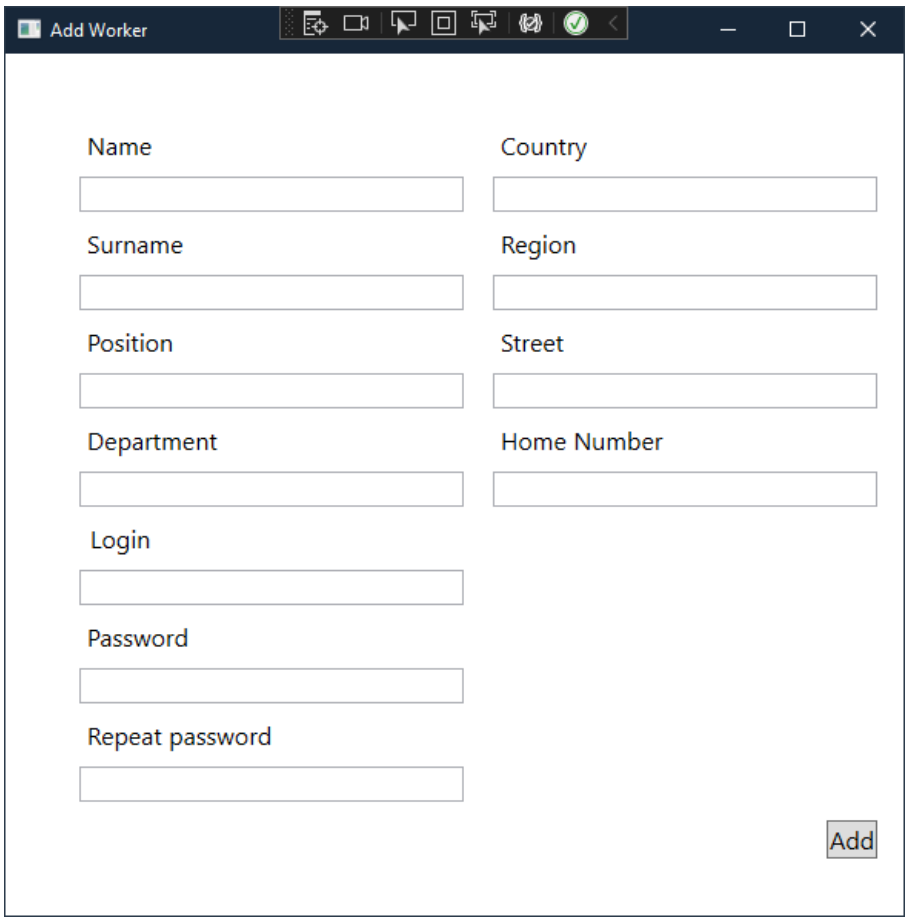

Рис 2.8 Сторінка Реєстрації працівників

Після реєстрації нового працівника, за новими даними доступу людина може зайти за свій робочий аккаунт для виконання роботи

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 1103;
2. коефіцієнт корекції програми в ході її розробки – 0,05;
3. коефіцієнт складності програми – 1,6;

4. годинна заробітна плата програміста– 95 грн/год;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,4;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,4;
7. вартість машино-години ЕОМ – 18 грн/год.

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:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин,} \quad (3.1)$$

де t_o- витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a- витрати праці на розробку блок-схеми алгоритму;

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

$t_{отл}$-витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

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

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p),$$

де q - передбачуване число операторів (3156);

C - коефіцієнт складності програми (1,3);

p - коефіцієнт корекції програми в ході її розробки (0,05).

Звідси умовне число операторів в програмі:

$$Q = 1{,}4 \cdot 1103 \cdot (1 + 0{,}05) = 1159{,}55$$

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

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \text{ людино-годин,}$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

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

$$t_u = (1159{,}55 \cdot 1{,}4) / (75 \cdot 1{,}4) = 15{,}46 \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20\ldots25) \cdot k}, \text{ людино-годин,} \quad (3.2)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.2), отримаємо:

t_a= 1159,55/ (20 · 1,4) = 41,41 людино-годин.

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25)\cdot k}$$, людино-годин.

t_n = 1159,55/ (25 · 1,4) = 33,13 людино-годин.

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)\cdot k}$$, людино-годин.

$t_{отл}$= 1159,55/ (5 · 1,4) = 165,65 чел.-ч.

- за умови комплексного налагодження завдання:

$$t^k_{отл} = 1{,}5 \cdot t_{отл}$$, людино-годин.

$t^k_{отл}$= 1,5 · 165,65 = 248,475 людино-годин.

Витрати праці на підготовку документації визначаються за формулою:

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

де $t_{др}$-трудомісткість підготовки матеріалів і рукопису:

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

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

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

Підставляючи відповідні значення, отримаємо:

$t_{др}$ = 1159,55/ (18 · 1,4) = 46,01 людино-годин.

$t_{до}$ = 0,75 · 46,01 = 34,5 людино-годин.

$t_{д}$ = 46,01 + 34,5 = 80,01 людино-годин.

Повертаючись до формули (3.1), отримуємо повну оцінку трудомісткості розробки програмного забезпечення:

t = 50 + 15,46 + 41,41 + 33,13 + 165,65 + 80,01 = 386 людино-годин.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}$$, грн.

Заробітна плата виконавців визначається за формулою:

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

де: t - загальна трудомісткість, людино-годин;

$C_{ПР}$ - середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 125 грн / год, отримуємо:

$$З_{ЗП} = 386 \cdot 95 = 36\ 670 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot C_{мч}, \text{ грн}, \quad (3.3)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$C_{мч}$ - вартість машино-години ЕОМ, грн/год (18 грн/год).

Підставивши в формулу (3.3) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 165{,}65 \cdot 18 = 2\ 981{,}7 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 194\ 880 + 14\ 881{,}94 = 39\ 651{,}7 \text{ грн.}$$

Очікуваний період створення ПЗ:

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

де B_k- число виконавців (дорівнює 1);

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

Звідси витрати на створення програмного продукту:

$$T = 386 / 1 \cdot 176 \approx 2{,}2 \text{ міс.}$$

Висновок: ПЗ розроблено з метою здійснення забезпечення працівникам доступу до документаційної бази даних, що надається фірмою та осягнення інформації про працівників для більш зручної взаємодії між відділами у робочих процесу. На розробку настільного застосунку піде близько 1606 годин, або **2,2** місяці. Вартість даного застосунку становитиме близько **39 651, 7** грн.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було розроблено універсальний застосунок «Електронний офіс» з використанням простих інструментів та підходів для створення Frontend та Backend частини додатку.

Призначенням даного додатку являється оптимізація офісних процесів. Здійснення створення документів, зворотного зв’язку з працівниками через надані контактні дані. Прискорення узгодження документообігу в середині компанії

Розроблене ПЗ дозволяє:

- доступне та просте середовище для взаємодії зі документообігом компанії ;
- контактування працівників фірми з іншими працівниками та адміністрацією;
- швидке створення документів;
- швидке узгодження документообігу.

Додаток створено за допомогою інструментів Desktop Development, а саме WPF технологій для створення Frontend частини, ADO.Net технологій для написання Backend частини та реляційна база даних MySQL.

Також, в даній кваліфікаційній роботі було визначено трудомісткість розробленого сайту 386 людино-годин, здійснено підрахунок вартості роботи по створенню сайту 39 651, 7 грн. та проведено розрахунок часу на його створення, який склав 2.2 місяців.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 для профессионалов Мэтью Макдональд 290 с

2. WPF 4. Подробное руководство Адам Натан 55 с

3. Язык программирования C# 7 и платформы .NET и .NET Core Филипп Джепикс и Эндрю Троелсен 169 с

4. C# 10 in a Nutshell Джозеф Албахари 391 с

5. ADO.NET in a nutshell Билл Гамильтон. 246 с

6. Microsoft® ADO.NET 4 Step by Step Тим Патрик 84 с

7. Murach's MySQL Джоэл Мурах 178 с

8. High Performance MySQL Барон Шварц, Вадим Ткаченко, и Питер Зайцев 177 с

9. WPF Control Development Unleashed Кевин Хофман и Паван Подила 139 с

10. XAML Unleashed Адам Нэйтан 96 с

11. Кулешов С.Г. Управлінське документознавство: Навч. посібник.- К.: ДАКККіМ, 2003.-57с.

12. Палеха Ю.І. Документаційне забезпечення управління. - К.: МАУП, 1997.- 343 с.

13. Ларьков Н.С. Документоведение:Учебное пособие.-М.:ВостокЗапад, 2006.-427 с.

14. Бездрабко В.В. Управлінське документознавство. Навчальний посібник .- К., 2006.- 208 с.

15. Примірна інструкція з діловодства у міністерствах, інших центральних органах виконавчої влади, Раді міністрів Автономної Республіки Крим, місцевих органах виконавчої влади// Затверджено постановою Кабінету Міністрів України від 17 жовтня 1997 р. N 1153

16. Энциклопедия делопроизводства [Електрон.ресурс].-URL: http://www.termika.ua/.- Загол. з екрану.

17. Методики керування. Системи електронного документообігу [Електрон.ресурс].-URL: http://www.o-fin.ua/.- Загол. з екрану.

18. Романов Д.А., Ильина Т.Н., Логинова А.Ю. Правда об электронном документообороте.-М.: ДМК Пресс, 2002.-224 с

19. Сайт "Професійні ресурси документознавства" www.documentoved.at.ua

20. Сайт «Підключення бази даних» https://netbeans.apache.org/kb/docs/ide/mysql_ru.html

21. Сайт «Створення бази даних» https://hostiq.ua/wiki/create-mysql-database/

22. Сайт «Налаштування WPF» https://help.keenetic.com/hc/ru/articles/213968929-Быстрая-настройка-WPS-для-версий-NDMS-2-11-и-более-ранних-

23. Сайт «Встановлення MySQL Server» https://help.eset.com/protect_install/90/ru-RU/mysql_windows.html

24. Сайт «.Net documentation» https://docs.microsoft.com/en-us/dotnet/core/introduction

25. Сайт «WPF documentation» https://docs.microsoft.com/en-us/dotnet/framework/windows-workflow-foundation/samples/wpf-and-wf-integration-in-xaml

ДОДАТОК А

КОД ПРОГРАМИ

MainWindow.xaml

```
<Window x:Class="OfficeFlowOptimizer.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
        xmlns:local="clr-namespace:OfficeFlowOptimizer"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800">
    <Grid>

        <Label Content="Login" HorizontalAlignment="Left"
Margin="264,127,0,0" VerticalAlignment="Top" FontSize="16" />
        <TextBox Name="LoginTb" HorizontalAlignment="Center"
Margin="0,163,0,0" TextWrapping="Wrap"  VerticalAlignment="Top"
Width="271" FontSize="16" />
        <Label Content="Password" HorizontalAlignment="Left"
Margin="264,191,0,0" VerticalAlignment="Top" FontSize="16" />
        <PasswordBox Name="PasswordTb" HorizontalAlignment="Center"
Margin="0,222,0,0"  VerticalAlignment="Top" Width="271" FontSize="16" />
        <Button Name="LoginBtn" Content="Login" HorizontalAlignment="Left"
Margin="400,251,0,0" VerticalAlignment="Top" Width="135"
Click="LoginBtn_Click" />
        <Label Name="ErrorLable" Content="" HorizontalAlignment="Center"
Margin="0,276,0,0" VerticalAlignment="Top" Width="271" Foreground="Red"
/>
    </Grid>
</Window>
```

AccountWindow.xaml.cs

```
using OfficeFlowOptimizer.Context;
using OfficeFlowOptimizer.Controller;
using OfficeFlowOptimizer.Model;
using System.Linq;
using System.Windows;
using System.Windows.Data;

namespace OfficeFlowOptimizer
```

```
{
    /// <summary>
    /// Interaction logic for AccountWindow.xaml
    /// </summary>
    public partial class AccountWindow : Window
    {
        private OptimizerDbContext _context;
        private WorkerController _workerController;
        private CollectionViewSource documentViewSource;
        private DocumentController _documentController;

        private string _login;
        private string _password;

        public AccountWindow(OptimizerDbContext context, string login, string
password)
        {
            _context = context;
            _login = login;
            _password = password;
            _workerController = new WorkerController(new
Repository.WorkerRepository(_context), new
Repository.DocumentRepository(_context));
            _documentController = new DocumentController(new
Repository.DocumentRepository(_context));
            var worker = _workerController.GetByLogin(_login, _password);

            InitializeComponent();

            NameLable.Content = worker.Name;
            SurnameLable.Content = worker.Surname;
            PositionLable.Content = worker.Position;
            DepartmentLable.Content = worker.Department;
            DocumentGridUpdate();
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
        }

        public void DocumentGridUpdate()
        {
            var worker = _workerController.GetByLogin(_login, _password);
            var Documents = _context.Documents.Local.ToObservableCollection();
```

```
            documentViewSource =

(CollectionViewSource)FindResource(nameof(documentViewSource));
            documentViewSource.Source =
_context.Documents.Local.ToObservableCollection().Where(w =>
w.Workers.All(i => i.Equals(worker))); ;
        }

        private void AddDocumentBtn_Click(object sender, RoutedEventArgs e)
        { OfficeFlowOptimizer/OfficeFlowOptimizer/AddDocumentWindow.xaml
            var addDocumentWindow = new AddDocumentWindow(this, _context);
            addDocumentWindow.Show();
        }

        private void IsSignedBtn_Click(object sender, RoutedEventArgs e)
        {
            Document selectedDocument =
(Document)this.documentDataGrid.SelectedItem;
            selectedDocument.IsSigned = true;
            _documentController.Update(selectedDocument);
            DocumentGridUpdate();
        }
    }
}
```

AddDocumentWindow.xaml

```
using OfficeFlowOptimizer.Context;
using OfficeFlowOptimizer.Controller;
using OfficeFlowOptimizer.Model;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;

namespace OfficeFlowOptimizer
{
    /// <summary>
    /// Interaction logic for AddDocumentWindow.xaml
    /// </summary>
    public partial class AddDocumentWindow : Window
    {
```

```
        private OptimizerDbContext _context;
        private WorkerController _workerController;
        private CollectionViewSource documentViewSource;
        private DocumentController _documentController;
        private AccountWindow _accountWindow;

        public ObservableCollection<CheckedListItem<Worker>> Sections { get;
set; }
        private CheckedListItem<Worker> _selectedSection;

        public CheckedListItem<Worker> SelectedSection
        {
            get { return _selectedSection; }
            set
            {
                _selectedSection = value;
                // OnPropertyChanged("SelectedSection");
            }
        }

        public class CheckedListItem<T> : INotifyPropertyChanged
        {
            public event PropertyChangedEventHandler PropertyChanged;

            private bool isChecked;
            private T item;

            public CheckedListItem()
            { }

            public CheckedListItem(T item, bool isChecked = false)
            {
                this.item = item;
                this.isChecked = isChecked;
            }

            public T Item
            {
                get { return item; }
                set
                {
                    item = value;
                    if (PropertyChanged != null) PropertyChanged(this, new
PropertyChangedEventArgs("Item"));
                }
```

```
        }

        public bool IsChecked
        {
            get { return isChecked; }
            set
            {
                isChecked = value;
                if (PropertyChanged != null) PropertyChanged(this, new
PropertyChangedEventArgs("IsChecked"));
            }
        }
    }

    public AddDocumentWindow(AccountWindow accountWindow,
OptimizerDbContext context)
    {
        _workerController = new WorkerController(new
Repository.WorkerRepository(context), new
Repository.DocumentRepository(context));
        _documentController = new DocumentController(new
Repository.DocumentRepository(context));

        Sections = new ObservableCollection<CheckedListItem<Worker>>();
        foreach (var worker in _workerController.Get())
            Sections.Add(new CheckedListItem<Worker>(worker));

        InitializeComponent();
        _accountWindow = accountWindow;
        _context = context;
        WorkersListCreate();
    }

    public void WorkersListCreate()
    {
        this.workersList.ItemsSource = Sections;
    }

    private void AddDocumentBtn_Click(object sender, RoutedEventArgs e)
    {
        var name = NameTb.Text;
        var article = ArticleTb.Text;
        var mainText = new TextRange(MainTextTb.Document.ContentStart,
MainTextTb.Document.ContentEnd).Text;
        List<Worker> workers = new List<Worker>();
```

```
            foreach (CheckedListItem<Worker> item in workersList.Items)
            {
                if (item.IsChecked == true)
                {
                    workers.Add(item.Item);
                }
            }

            if (name == "")
            {
                ErrorLable.Content = "Name field can not be empty";
                return;
            }
            if (article == "")
            {
                ErrorLable.Content = "Article field can not be empty";
                return;
            }
            if (mainText == "")
            {
                ErrorLable.Content = "Main text field can not be empty";
                return;
            }
            if (workers.Count <= 0)
            {
                ErrorLable.Content = "Pick at least one worker";
                return;
            }

            _documentController.Create(new Model.Document() { Name = name,
Article = article, MainText = mainText, Workers = workers });
            _context.SaveChanges();
            _accountWindow.DocumentGridUpdate();
            this.Close();
        }
    }
}
```

AddWorkerWindow.xaml

```
<Window x:Class="OfficeFlowOptimizer.AddWorkerWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
        mc:Ignorable="d"
        Title="Add Worker" Height="600" Width="600
        ">
    <Grid>
        <Label Content="Name" HorizontalAlignment="Left" Margin="48,44,0,0"
VerticalAlignment="Top" FontSize="16"
RenderTransformOrigin="0.505,0.556" />
        <TextBox Name="NameTb" HorizontalAlignment="Left"
Margin="48,80,0,0" VerticalAlignment="Top" FontSize="16" Width="250"
MaxLength="15" />
        <Label Content="Surname" HorizontalAlignment="Left"
Margin="48,108,0,0" VerticalAlignment="Top" FontSize="16" />
        <TextBox Name="SurnameTb" HorizontalAlignment="Left"
Margin="48,144,0,0" VerticalAlignment="Top" FontSize="16" Width="250"
MaxLength="15" />
        <Label Content="Position" HorizontalAlignment="Left"
Margin="48,172,0,0" VerticalAlignment="Top" FontSize="16" />
        <TextBox Name="PositionTb"  HorizontalAlignment="Left"
Margin="48,208,0,0" VerticalAlignment="Top" FontSize="16" Width="250"
MaxLength="15" />
        <Label Content="Department" HorizontalAlignment="Left"
Margin="48,236,0,0" VerticalAlignment="Top" FontSize="16" />
        <TextBox Name="DepartmentTb"  HorizontalAlignment="Left"
Margin="48,272,0,0" VerticalAlignment="Top" FontSize="16" Width="250"
MaxLength="15" />
        <Label Content="Login" HorizontalAlignment="Left"
Margin="50,300,0,0" VerticalAlignment="Top" FontSize="16" />
        <TextBox Name="LoginTb" HorizontalAlignment="Left"
Margin="48,336,0,0" TextWrapping="Wrap" VerticalAlignment="Top"
Width="250" FontSize="16" MaxLength="15" />
        <Label Content="Password" HorizontalAlignment="Left"
Margin="48,364,0,0" VerticalAlignment="Top" FontSize="16" />
        <PasswordBox Name="PasswordTb" HorizontalAlignment="Left"
Margin="48,400,0,0"  VerticalAlignment="Top" Width="250" FontSize="16"
MaxLength="14" />
        <Label Content="Repeat password" HorizontalAlignment="Left"
Margin="48,428,0,0" VerticalAlignment="Top" FontSize="16" />
        <PasswordBox Name="PasswordRepeatTb" HorizontalAlignment="Left"
Margin="48,464,0,0"  VerticalAlignment="Top" Width="250" FontSize="16"
MaxLength="15" />
        <Button Content="Add" HorizontalAlignment="Left"
Margin="534,499,0,0" VerticalAlignment="Top" FontSize="16"
Click="Button_Click" />
```

```
        <Label Name="ErrorLable" HorizontalAlignment="Left"
Margin="48,507,0,0" VerticalAlignment="Top" Foreground="Red"
Width="400" />
        <Label Content="Country" HorizontalAlignment="Left"
Margin="317,44,0,0" VerticalAlignment="Top" FontSize="16"
RenderTransformOrigin="0.505,0.556" />
        <TextBox Name="CountryTb" HorizontalAlignment="Left"
Margin="317,80,0,0" VerticalAlignment="Top" FontSize="16" Width="250"
MaxLength="15" />
        <Label Content="Region" HorizontalAlignment="Left"
Margin="317,108,0,0" VerticalAlignment="Top" FontSize="16" />
        <TextBox Name="RegionTb" HorizontalAlignment="Left"
Margin="317,144,0,0" VerticalAlignment="Top" FontSize="16" Width="250"
MaxLength="15" />
        <Label Content="Street" HorizontalAlignment="Left"
Margin="317,172,0,0" VerticalAlignment="Top" FontSize="16" />
        <TextBox Name="StreetTb"  HorizontalAlignment="Left"
Margin="317,208,0,0" VerticalAlignment="Top" FontSize="16" Width="250"
MaxLength="15" />
        <Label Content="Home Number" HorizontalAlignment="Left"
Margin="317,236,0,0" VerticalAlignment="Top" FontSize="16" />
        <TextBox Name="HomeNumTb"  HorizontalAlignment="Left"
Margin="317,272,0,0" VerticalAlignment="Top" FontSize="16" Width="250"
MaxLength="15" />
    </Grid>
</Window>
```

AddWorkerWindow.xaml.cs

```
using OfficeFlowOptimizer.Context;
using OfficeFlowOptimizer.Controller;
using System.Linq;
using System.Windows;

namespace OfficeFlowOptimizer
{
    /// <summary>
    /// Interaction logic for AddWorkerWindow.xaml
    /// </summary>
    public partial class AddWorkerWindow : Window
    {
        private OptimizerDbContext _context;
        private WorkerController _workerController;
        private AdminWindow _adminWindow;
        private AddressController _addressController;
```

```
        public AddWorkerWindow(OptimizerDbContext context, AdminWindow
admintWindow)
        {
            _context = context;
            _workerController = new WorkerController(new
Repository.WorkerRepository(_context), new
Repository.DocumentRepository(_context));
            _addressController = new AddressController(new
Repository.AddressRepository(_context));
            InitializeComponent();
            _adminWindow = admintWindow;
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            var name = NameTb.Text;
            var surname = SurnameTb.Text;
            var position = PositionTb.Text;
            var department = DepartmentTb.Text;
            var login = LoginTb.Text;
            var password = PasswordTb.Password;
            var repeatPassword = PasswordRepeatTb.Password;

            var country = CountryTb.Text;
            var region = RegionTb.Text;
            var street = StreetTb.Text;
            var homeNum = HomeNumTb.Text;

            if (name == "")
            {
                ErrorLable.Content = "Name field can not be empty";
                return;
            }
            if (surname == "")
            {
                ErrorLable.Content = "Surname field can not be empty";
                return;
            }
            if (position == "")
            {
                ErrorLable.Content = "Position field can not be empty";
                return;
            }
            if (department == "")
            {
```

```
    ErrorLable.Content = "Department field can not be empty";
    return;
}
if (login == "")
{
    ErrorLable.Content = "Login field can not be empty";
    return;
}
if (country == "")
{
    ErrorLable.Content = "Country field can not be empty";
    return;
}
if (region == "")
{
    ErrorLable.Content = "Region field can not be empty";
    return;
}
if (street == "")
{
    ErrorLable.Content = "Street field can not be empty";
    return;
}
if (homeNum == "")
{
    ErrorLable.Content = "Home Number field can not be empty";
    return;
}
foreach (var worker in _context.Workers)
    if (login == worker.Login)
    {
        ErrorLable.Content = "Login is already taken";
        return;
    }
if (password == "")
{
    ErrorLable.Content = "Password field can not be empty";
    return;
}
if (repeatPassword == "")
{
    ErrorLable.Content = "Repeat Password field can not be empty";
    return;
}
```

```
            if (!password.Equals(repeatPassword))
            {
                ErrorLable.Content = "Password and Repeat Password fields doesn't
math";
                return;
            }
            _addressController.Create(new Model.Address()
            {
                Country = country,
                Region = region,
                Street = street,
                HomeNum = homeNum
            });
            _context.SaveChanges();
            _workerController.Create(new Model.Worker()
            {
                Name = name,
                Surname = surname,
                Position = position,
                Department = department,
                Login = login,
                Password = password,
                AddressId = _context.Addresses.Where(x => x.Country ==
country).Where(x => x.Region == region).Where(x => x.Street ==
street).Where(x => x.HomeNum == homeNum).FirstOrDefault().Id
            }); ;
            _context.SaveChanges();
            _adminWindow.WorkerGridUpdate();
            this.Close();
        }
    }
}
```

AdminWindow.xaml.cs

```
using OfficeFlowOptimizer.Context;
using OfficeFlowOptimizer.Controller;
using System.Windows;
using System.Windows.Data;

namespace OfficeFlowOptimizer
{
    /// <summary>
    /// Interaction logic for AdminWindow.xaml
    /// </summary>
    public partial class AdminWindow : Window
    {
```

```
        private OptimizerDbContext _context;
        private WorkerController _workerController;
        private CollectionViewSource workerViewSource;

        public AdminWindow(OptimizerDbContext context, string login, string
password)
        {
            _context = context;
            _workerController = new WorkerController(new
Repository.WorkerRepository(_context), new
Repository.DocumentRepository(_context));
            var worker = _workerController.GetByLogin(login, password);

            InitializeComponent();
            NameLable.Content = worker.Name;
            SurnameLable.Content = worker.Surname;
            PositionLable.Content = worker.Position;
            DepartmentLable.Content = worker.Department;
            WorkerGridUpdate();
        }

        private void AddWorkerBtn_Click(object sender, RoutedEventArgs e)
        {
            var addWindow = new AddWorkerWindow(_context, this);
            addWindow.Show();
        }

        public void WorkerGridUpdate()
        {
            workerViewSource =
                (CollectionViewSource)FindResource(nameof(workerViewSource));
            workerViewSource.Source =
_context.Workers.Local.ToObservableCollection();
        }
    }
}
```

AdminWindow.xaml

```
<Window x:Class="OfficeFlowOptimizer.AdminWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
        mc:Ignorable="d"
```

```
      Title="AdminWindow" Height="450" Width="800">
   <Window.Resources>
      <CollectionViewSource x:Key="workerViewSource" />
      <CollectionViewSource x:Key="workerDocumentsViewSource"
                     Source="{Binding Documents, Source={StaticResource
workerViewSource}}" />
   </Window.Resources>
   <Grid>
      <Label Content="Name" HorizontalAlignment="Left" Margin="50,62,0,0"
VerticalAlignment="Top" FontSize="16" />
      <Label Name="NameLable" Content="" HorizontalAlignment="Left"
Margin="150,62,0,0" VerticalAlignment="Top" FontSize="16" />
      <Label Content="Surname" HorizontalAlignment="Left"
Margin="50,98,0,0" VerticalAlignment="Top" FontSize="16" />
      <Label Name="SurnameLable" Content="" HorizontalAlignment="Left"
Margin="150,98,0,0" VerticalAlignment="Top" FontSize="16" />
      <Label Content="Position" HorizontalAlignment="Left"
Margin="50,134,0,0" VerticalAlignment="Top" FontSize="16" />
      <Label Name="PositionLable" Content="" HorizontalAlignment="Left"
Margin="150,134,0,0" VerticalAlignment="Top" FontSize="16" />
      <Label Content="Department" HorizontalAlignment="Left"
Margin="50,165,0,0" VerticalAlignment="Top" FontSize="16" />
      <Label Name="DepartmentLable" Content=""
HorizontalAlignment="Left" Margin="150,165,0,0" VerticalAlignment="Top"
FontSize="16" />
      <Grid Margin="293,0,0,210">
         <DataGrid x:Name="documentDataGrid"
AutoGenerateColumns="False"
        EnableRowVirtualization="True"
        ItemsSource="{Binding Source={StaticResource workerViewSource}}"
        Margin="10,10,10,34"
RowDetailsVisibilityMode="VisibleWhenSelected">
            <DataGrid.Columns>
               <DataGridTextColumn Binding="{Binding Name}"
                     Header="Name" Width="*"
                     IsReadOnly="True" />
               <DataGridTextColumn Binding="{Binding Surname}"
Header="Surname"
                     Width="*"
                     IsReadOnly="True" />
               <DataGridTextColumn Binding="{Binding Position}"
Header="Position"
                     Width="*"
                     IsReadOnly="True" />
```

```
                <DataGridTextColumn Binding="{Binding Department}"
Header="Department"
                                    Width="*"
                                    IsReadOnly="True" />
            </DataGrid.Columns>
        </DataGrid>
        <Grid>
            <Button Name="AddWorkerBtn" HorizontalAlignment="Right"
VerticalAlignment="Bottom" Content="Add" Click="AddWorkerBtn_Click"
Margin="0,0,10,10" />
        </Grid>
    </Grid>
  </Grid>
</Window>
```

AdminWindow.xaml.cs

```
using OfficeFlowOptimizer.Context;
using OfficeFlowOptimizer.Controller;
using System.Windows;
using System.Windows.Data;

namespace OfficeFlowOptimizer
{
  /// <summary>
  /// Interaction logic for AdminWindow.xaml
  /// </summary>
  public partial class AdminWindow : Window
  {
    private OptimizerDbContext _context;
    private WorkerController _workerController;
    private CollectionViewSource workerViewSource;

    public AdminWindow(OptimizerDbContext context, string login, string
password)
    {
      _context = context;
      _workerController = new WorkerController(new
Repository.WorkerRepository(_context), new
Repository.DocumentRepository(_context));
      var worker = _workerController.GetByLogin(login, password);

      InitializeComponent();
      NameLable.Content = worker.Name;
      SurnameLable.Content = worker.Surname;
      PositionLable.Content = worker.Position;
```

```
            DepartmentLable.Content = worker.Department;
            WorkerGridUpdate();
        }

        private void AddWorkerBtn_Click(object sender, RoutedEventArgs e)
        {
            var addWindow = new AddWorkerWindow(_context, this);
            addWindow.Show();
        }

        public void WorkerGridUpdate()
        {
            workerViewSource =
                (CollectionViewSource)FindResource(nameof(workerViewSource));
            workerViewSource.Source =
_context.Workers.Local.ToObservableCollection();
        }
    }
}
```

App.xaml

```
<Application x:Class="OfficeFlowOptimizer.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             StartupUri="MainWindow.xaml">
    <Application.Resources>
    </Application.Resources>
</Application>
```

App.xaml.cs

```
using System.Windows;

namespace OfficeFlowOptimizer
{
    /// <summary>
    /// Interaction logic for App.xaml
    /// </summary>
    public partial class App : Application
    {
    }
}
```

AssemblyInfo.cs

```
using System.Windows;
```

```
[assembly: ThemeInfo(
   ResourceDictionaryLocation.None, //where theme specific resource
dictionaries are located
                                    //(used if a resource is not found in the page,
                                    // or application resource dictionaries)
   ResourceDictionaryLocation.SourceAssembly //where the generic resource
dictionary is located
                                              //(used if a resource is not found in the page,
                                              // app, or any theme specific resource dictionaries)
)]
```

OfficeFlowOptimizer.csproj

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>WinExe</OutputType>
    <TargetFramework>net6.0-windows</TargetFramework>
    <Nullable>enable</Nullable>
    <UseWPF>true</UseWPF>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference
Include="Microsoft.AspNetCore.Identity.EntityFrameworkCore"
Version="6.0.5" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Sqlite"
Version="6.0.5" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer"
Version="6.0.5" />
    <PackageReference Include="Microsoft.Extensions.Configuration"
Version="6.0.1" />
  </ItemGroup>

  <ItemGroup>
    <Page Update="AddDocumentWindow.xaml">
      <Generator> MSBuild:UpdateDesignTimeXaml</Generator>
    </Page>
  </ItemGroup>

</Project>
```

OptimizerDbContext.cs

```
using Microsoft.EntityFrameworkCore;
using OfficeFlowOptimizer.Model;
```

```
using System.Collections.Generic;

namespace OfficeFlowOptimizer.Context
{
   public class OptimizerDbContext : DbContext
   {
      public OptimizerDbContext()
      {
         Database.EnsureDeleted();
         Database.EnsureCreated();
      }

      public OptimizerDbContext(DbContextOptions options) : base(options)
      {
         Database.EnsureDeleted();
         Database.EnsureCreated();
      }

      protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
      {
         optionsBuilder.UseSqlServer(
            "Data Source=(localdb)\\MSSQLLocalDB;Initial
Catalog=Office;Integrated Security=True;Connect
Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=Re
adWrite;MultiSubnetFailover=False");
         base.OnConfiguring(optionsBuilder);
      }

      protected override void OnModelCreating(ModelBuilder modelBuilder)
      {
         modelBuilder.Entity<Address>().HasData(
            new Address(1, "Ukraine", "Dnipro", "Korolenko", "12"));
         modelBuilder.Entity<Document>().HasData(

               new List<Document>() {
                  new Document(1, "Cargo delivering", "From Dnipro To Kyiv",
                  "Cargo 12 tons myst be delivered from Dnipro to Kyiv due
wednesdey")
               });
         modelBuilder.Entity<Worker>().HasData(
            new Worker[] {
            new Worker(){Id= 1,Name = "Oleksa123ndr",Surname = "Pinich",
Position = "Worker", Department =  "Accounting", Login=  "worker",Password
= "worker", AddressId = 1 },
```

```
            new Worker(2,"Michail", "Bones", "Worker","Routing",
"BonesMch31", "Amisdufhwb22"),
            new Worker(3,"Admin", "AdminSur", "Admin","Administration",
"admin", "admin"),
            });
        base.OnModelCreating(modelBuilder);
    }

    //Fields would be created automaticly, but compiler .Net 8 doesn't know it
so add "= null!" to say "Compile it anyway"
    public DbSet<Worker> Workers { get; set; } = null!;

    public DbSet<Document> Documents { get; set; } = null!;
    public DbSet<Address> Addresses { get; set; } = null!;
  }
}
```

AddressController.cs

```
using OfficeFlowOptimizer.Model;
using OfficeFlowOptimizer.Repository;
using System.Collections.Generic;

namespace OfficeFlowOptimizer.Controller
{
  public class AddressController
  {
    private AddressRepository _repository;

    public AddressController(AddressRepository repository)
    {
      _repository = repository;
    }

    public IEnumerable<Address> Get()
    {
      IEnumerable<Address> addresses = _repository.GetAll();
      return addresses;
    }

    public Address Get(int id)
    {
      Address address = _repository.GetById(id);
      return address;
    }
```

```
        public void Create(Address address)
        {
            if (address == null)
            {
                return;
            }

            _repository.Add(address);
        }

        public void Delete(Address address)
        {
            _repository.Delete(address);
        }

        public void Update(Address address)
        {
            _repository.Update(address);
        }
    }
}
```

DocumentController.cs

```
using OfficeFlowOptimizer.Model;
using OfficeFlowOptimizer.Repository;
using System.Collections.Generic;

namespace OfficeFlowOptimizer.Controller
{
    public class DocumentController
    {
        private DocumentRepository _repository;

        public DocumentController(DocumentRepository repository)
        {
            _repository = repository;
        }

        public IEnumerable<Document> Get()
        {
            IEnumerable<Document> documents = _repository.GetAll();
            return documents;
        }
```

```
        public Document Get(int id)
        {
            Document document = _repository.GetById(id);
            return document;
        }

        public void Create(Document document)
        {
            if (document == null)
            {
                return;
            }

            _repository.Add(document);
        }

        public void Delete(Document document)
        {
            _repository.Delete(document);
        }

        public void Update(Document document)
        {
            _repository.Update(document);
        }
    }
}
```

WorkerController.cs

```
using OfficeFlowOptimizer.Model;
using OfficeFlowOptimizer.Repository;
using System.Collections.Generic;
using System.Linq;

namespace OfficeFlowOptimizer.Controller
{
    public class WorkerController
    {
        private WorkerRepository _repository;
        private DocumentRepository _documentRepository;

        public WorkerController(WorkerRepository repository,
DocumentRepository documentRepository)
        {
```

```
            _repository = repository;
            _documentRepository = documentRepository;
        }

        public List<Worker> Get()
        {
            List<Worker> workers = _repository.GetAll().ToList();
            return workers;
        }

        public Worker Get(int id)
        {
            Worker worker = _repository.GetById(id);
            return worker;
        }

        public Worker GetByLogin(string login, string password)
        {
            IEnumerable<Worker> workers = _repository.GetAll();
            var worker = workers.Where(x => x.Login == login && x.Password ==
password).FirstOrDefault();
            return worker;
        }

        public void Create(Worker worker)
        {
            if (worker == null)
            {
                return;
            }

            _repository.Add(worker);
        }

        public void Delete(Worker worker)
        {
            _repository.Delete(worker);
        }

        public void Update(Worker worker)
        {
            _repository.Update(worker);
        }
    }
}
```

Address.cs

```
namespace OfficeFlowOptimizer.Model
{
   public class Address
   {
      public Address()
      {
      }

      public Address(int id, string country, string region, string street, string
homeNum)
      {
         Id = id;
         Country = country;
         Region = region;
         Street = street;
         HomeNum = homeNum;
      }

      public int Id { get; set; }
      public string Country { get; set; }
      public string Region { get; set; }
      public string Street { get; set; }
      public string HomeNum { get; set; }
   }
}
```

Document.cs

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace OfficeFlowOptimizer.Model
{
   public class Document
   {
      public Document()
      {
      }

      public Document(int id, string name, string article, string mainText)
      {
         Id = id;
         Name = name;
         Article = article;
         MainText = mainText;
```

```
        }

        [Key]
        public int Id { get; set; }

        public string Name { get; set; }
        public string Article { get; set; }
        public string MainText { get; set; }
        public bool IsSigned { get; set; }

        public List<Worker> Workers { get; set; } = new List<Worker>();
    }
}
```

Worker.cs

```
namespace OfficeFlowOptimizer.Model
{
    public class Worker
    {
        public void AddExtra(string about, string phone)
        {
            About = about;
            Phone = phone;
        }

        public Worker(int id, string name, string surname, string position, string
department, string login, string password)
        {
            Name = name;
            Surname = surname;
            Position = position;
            Id = id;
            Login = login;
            Password = password;
            Department = department;
        }

        public Worker()
        {
        }

        public int Id { get; set; }
        public string Name { get; set; }
        public string Surname { get; set; }
        public string Position { get; set; }
```

```
        public string Department { get; set; }
        public string? About { get; set; }
        public string? Phone { get; set; }
        public string Login { get; set; }
        public string Password { get; set; }

        public int AddressId { get; set; }
    }
}
```

AddressRepository.cs

```
using OfficeFlowOptimizer.Context;
using OfficeFlowOptimizer.Model;
using System.Collections.Generic;
using System.Linq;

namespace OfficeFlowOptimizer.Repository
{
    public class AddressRepository
    {
        private OptimizerDbContext _context;

        public AddressRepository(OptimizerDbContext context)
        {
            _context = context;
        }

        public IEnumerable<Address> GetAll()
        {
            return _context.Addresses.ToList();
        }

        public Address GetById(int id)
        {
            return _context.Addresses.Where(x => x.Id == id).FirstOrDefault();
        }

        public void Add(Address Address)
        {
            _context.Addresses.Add(Address);
            _context.SaveChanges();
        }

        public void Delete(Address Address)
```

```
        {
            _context.Addresses.Remove(Address);
            _context.SaveChanges();
        }

        public void Update(Address Address)
        {
            _context.Addresses.Update(Address);
            _context.SaveChanges();
        }
    }
}
```

DocumentRepository.cs

```
using OfficeFlowOptimizer.Context;
using OfficeFlowOptimizer.Model;
using System.Collections.Generic;
using System.Linq;

namespace OfficeFlowOptimizer.Repository
{
    public class DocumentRepository
    {
        private OptimizerDbContext _context;

        public DocumentRepository(OptimizerDbContext context)
        {
            _context = context;
        }

        public IEnumerable<Document> GetAll()
        {
            return _context.Documents.ToList();
        }

        public Document GetById(int id)
        {
            return _context.Documents.Where(x => x.Id == id).FirstOrDefault();
        }

        public void Add(Document Document)
        {
            _context.Documents.Add(Document);
            _context.SaveChanges();
        }
```

```
        public void Delete(Document Document)
        {
            _context.Documents.Remove(Document);
            _context.SaveChanges();
        }

        public void Update(Document Document)
        {
            _context.Documents.Update(Document);
            _context.SaveChanges();
        }
    }
}
```

WorkerRepository.cs

```
using OfficeFlowOptimizer.Context;
using OfficeFlowOptimizer.Model;
using System.Collections.Generic;
using System.Linq;

namespace OfficeFlowOptimizer.Repository{
    public class WorkerRepository{
        private OptimizerDbContext _context;

        public WorkerRepository(OptimizerDbContext context){
            _context = context;
        }

        public IEnumerable<Worker> GetAll(){
            return _context.Workers.ToList();
        }

        public Worker GetById(int id) {
            return _context.Workers.Where(x => x.Id == id).FirstOrDefault();
        }

        public void Add(Worker worker){
            _context.Workers.Add(worker);
            _context.SaveChanges();
        }

        public void Delete(Worker worker){
            _context.Workers.Remove(worker);
            _context.SaveChanges();
```

```
        }

        public void Update(Worker worker){
            _context.Workers.Update(worker);
            _context.SaveChanges();
        }
    }
}
```

У даному додатку було вказано весь об’єм даних з проекту. Для реалізації проекту, його необхідно створювати як WPF проект. Повний список також надано на електронному носії

ДОДАТОК Б

ВІДГУК
керівника економічного розділу
на кваліфікаційну роботу бакалавра
на тему:
«Розробка програмного забезпечення оптимізації виконання офісних задач засобами ADO.Net, MySQL, WPF»
студента групи 122-18-1 Гриценко Олексія Ігоровича

Керівник економічного розділу
доцент каф. ПЕП та ПУ, к.е.н **Л. В. Касьяненко**

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом Гриценко.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом Гриценко.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
OfficeFlowOptimizer.zip	Архів. Містить коди програми.
Презентація	
Презентація Гриценко.ppt	Презентація кваліфікаційної роботи.