МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____Аліна САВЧЕНКО
«_____» _____2021 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ

"МАГІСТРА"

ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ "ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ"

Тема: «Інформаційна система щодо обрахунку заборгованості та бухгалтерського обліку»

Виконавець: Войцехівський Вячеслав Мирославович

Керівник: професор Моржов Володимир Іванович

Нормоконтролер: _____ Ігор РАЙЧЕВ
(підпис)

Київ – 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра Комп'ютерних інформаційних технологій

Галузь знань, спеціальність, освітньо-професійна програма: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології"

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____Аліна САВЧЕНКО

«_____» _____2021р.

ЗАВДАННЯ

на виконання дипломної роботи студента

Войцехівського Вячеслава Мирославовича

(прізвище, ім'я, по батькові)

1. Тема роботи: "Інформаційна система щодо обрахунку заборгованості та бухгалтерського обліку" затверджена наказом ректора від 12.10.2021 р. № 2228/ст.

2. Термін виконання роботи: з 12.10.2021 р. по 31.12.2021 р.

3. Вихідні дані до роботи: теоретичні відомості та основи проектування інформаційних систем, множина відомих архітектур програмних компонентів і додатків та програмних систем (ПС), множина відомих патернів проектування.

4. Зміст пояснювальної записки: вступ, дослідження предметної області, дослідження існуючих рішень на ринку, вибір архітектури та технологій для розробки інформаційної системи, висновок

5. Перелік обов'язкового графічного матеріалу: слайди, скріншоти, презентація

6. Календарний план-графік

№ п/п	Завдання	Термін виконання	Підпис керівника
1.	Отримання завдання на дипломну роботу та побудова плану-графіку виконання робіт.	12.10.2021 – 15.10.2021	
2.	Огляд та аналіз відомих програмних архітектур.	16.10.2021 – 19.10.2021	
3.	Огляд та створення концепції архітектури ПС.	20.10.2021 – 24.10.2021	
4.	Проектування програмної системи.	25.10.2021 – 31.10.2021	
5.	Написання Розділу 1 дипломної роботи.	01.11.2021 – 07.11.2021	
6.	Написання Розділу 2 дипломної роботи.	08.11.2021 – 17.11.2021	
7.	Написання Розділу 3 дипломної роботи. Завершення створення пояснювальної записки дипломної роботи.	18.11.2021 – 01.12.2021	
8.	Оформлення та друк пояснювальної записки.	02.12.2021 – 11.12.2021	
9.	Створення презентації, доповіді та підготовка до захисту дипломної роботи	12.12.2021 – 20.12.2021	

7. Дата видачі завдання: 12.10.2021 р.

Керівник дипломної роботи _____ Володимир МОРЖОВ
(підпис керівника)

Завдання прийняв до виконання _____ Вячеслав ВОЙЦЕХІВСЬКИЙ
(підпис випускника)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Інформаційна система щодо обрахунку заборгованості та бухгалтерського обліку» складається із вступу, трьох розділів, загальних висновків, списку використаних джерел і містить 75 сторінок тексту, 15 рисунків та 6 таблиць. Список використаних джерел містить 15 найменувань.

У дипломній роботі було розглянуто процес розробки програмного забезпечення для обрахунку заборгованості на підприємстві, було виконано порівняння з існуючими рішеннями, розглянуто нові методи та підходи до розробки інформаційної системи.

Метою даної роботи було створення інструменту для полегшення обрахунку заборгованості на підприємстві.

Предметом дослідження є програмне забезпечення яке має виконувати певні задачі в середі його цільового використання.

Об'єктом дослідження є архітектура ПЗ, його взаємодія з елементами як внутрішніми так і зовнішніми.

Ключові слова: КОРИСТУВАЧ, АРХІТЕКТУРА, СИСТЕМА, ПРОГРАМА, ДЕСКТОП, WPF, C#, MSSQL, .NET.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

ПК – Персональний комп’ютер.

ПЗ – Програмне забезпечення.

API – Application Programming Interface – Прикладний програмний інтерфейс.

IDE – Integrated Development Environment – інтегроване середовище розробки програм.

HTML – HyperText Markup Language – мова гіпертекстової розмітки.

WPF – Windows Presentation Foundation – графічна (презентаційна) підсистема.

XML – Extensible Markup Language – Розширювана мова розмітки.

ВСТУП

У наш час українському сучасному підприємству необхідно приймати рішення при невизначених ризиках, які змушують постійно контролювати різнопланові аспекти фінансово – господарської діяльності. Дана діяльність зображена у великій кількості документів, що містять різнорідну інформацію. Якщо дана інформація правильно “переварена” і систематизована вона є такою собі гарантією вдалого керування організацією. Та навпаки відсутність нормальних та точних даних може спричинити максимально погані наслідки для підприємства зрозуміло що, наслідком цього будуть серйозні фінансові збитки.

Ми не будемо брати до такі дії , які були вчинені спеціально в цілях дестабілізувати організацію, а отже виключаючи це, усі помилки під час бухгалтерської діяльності відбуваються через неуважність , як приклад – математичні/арифметичні помилки/погрішності чи через банально недостатнього досвіду ведення бухгалтерії в Україні. Нажаль подібне практично неминуче якщо ми будемо вести облік вручну або якщо будемо користуватись невідомим ПЗ (онлайн калькулятори і тд).

З розвитком ринкової економіки все більше значення приділяється управлінню, оскільки правильна його організація забезпечує успішну діяльність підприємства. За допомогою управління підприємством виробляється стратегія його розвитку, складаються плани та управлінські рішення, виявляються резерви підвищення ефективності виробництва , оцінюються результати діяльності підприємства.

Та навіть в умовах ринкової економіки державні підприємства усе більше хочуть мати надійне, зручне та сучасне програмне забезпечення, бухгалтерський облік має чи не найпоширенішою потребу в максимальній але і зрозумілій простому користувачу цифровізації.

У процесі фінансового-господарської діяльності кожне підприємство вступає в певні економічні взаємовідносини з іншими суб’єктами господарювання, в результаті цього відбуваються господарські операції. Актуальність теми полягає в

тому, що орендарі не приділяють увагу до заборгованості, що призводить до негативних наслідків – прискореного зростання боргу перед орендодавцем, не повернення боргів , втрачання власних коштів, судових процесів.

Сучасні умови господарювання вимагають вирішення цілого ряду облікових проблем . Зокрема, це є теоретичні і методичні аспекти класифікації та відображення заборгованості в системі рахунків бухгалтерського обліку, невизначеність обліку простроченої і безнадійної дебіторської заборгованості та її рефінансування, аналіз заборгованості та автоматизація бухгалтерського обліку взаєморозрахунків.

А отже необхідність розробки програмного забезпечення для обрахунку заборгованості обумовили вибір теми дипломної роботи та її актуальність.

РОЗДІЛ 1.
ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА СУЧАСНИХ МЕТОДІВ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Дослідження предметної області у сфері використання інформаційної системи

Розробка програмного забезпечення завжди розпочинається з дослідження , в нашому випадку з дослідження предметної області що є бухгалтерським обліком та фінансовими операціями в умовах дії Українського законодавства.

Бухгалтерський облік це процес виявлення, вимірювання, реєстрації, накопичення, узагальнення , зберігання та передавання інформації про діяльність підприємства зовнішнім та внутрішнім користувачам для прийняття рішень [1].

Класичні ознаки господарської діяльності, наголошуючи на її суспільно корисному характері, наводить авторитетний український учений В. С . Щербина: «Суспільно корисна господарська діяльність: по -перше, полягає у виробництві продукції, виконанні робіт , наданні послуг не для власних потреб виробника , а для задоволення потреб інших осіб; по -друге, виконується на професійних засадах; по-третє , результати такої діяльності мають реалізовуватися за плату , тобто функціонувати як товар; по-четверте, поєднує як приватні інтереси виробника, так і публічні інтереси (держави, суспільства, значних верств населення тощо)» [1].

Фінансово-господарська діяльність складається з: формування джерел , необхідних для діяльності. Джерела можуть бути фінансовими і матеріальними; – розміщення притягнутих і сформованих засобів :

створення виробничих запасів, витрат на виробництво продукції , реалізації продукції, взаємин з бюджетом у частині оподатковування й інших господарських операціях.

Метод бухгалтерського обліку – ведення обліку фінансово-господарських операцій на основі натуральних вимірників у грошовому вираженні шляхом суцільного, безупинного, документального і взаємозалежного їхнього вираження . Задачами бухгалтерського обліку є:

– формування достовірної інформації про господарські процеси і результати діяльності підприємств необхідної для оперативного керівництва і управління, а також її використання інвесторами, покупцями, податковими, фінансовими , банківськими органами й іншими зацікавленими особами;

– забезпечення контролю за наявністю руху майна і використання матеріальних, трудових і фінансових ресурсів відповідно до затверджених норм, нормативами і кошторисами;

– попередження негативних явищ у фінансово-господарській діяльності, виявлення і мобілізація внутрішньогосподарських ресурсів.

Також оскільки наша система має вести обрахунок заборгованості, що виникає між суб'єктами господарювання ми маємо дослідити такі поняття як борг, пеня, 3% річних та інфляційні втрати і звісно ж досліджуючи ці поняття ми неодмінно будемо звертатись до діючого законодавства .

Пеня - це вид неустойки, що забезпечує виконання грошового зобов'язання і обчислюється у відсотках від суми несвоєчасно виконаного грошового зобов'язання за кожний день прострочення виконання [2] .

Згідно з частиною шостою статті 232 Господарського кодексу України нарахування штрафних санкцій за прострочення виконання зобов'язання, якщо інше не встановлено законом або договором, припиняється через шість місяців від дня, коли зобов'язання мало бути виконано [3]. Таким чином, період часу за який можна нараховувати пеню складає 6 місяців від дня, коли відповідне зобов'язання мало бути виконане, але законом або укладеним сторонами договором може бути передбачено більшу або меншу тривалість цього періоду.

Його перебіг починається з дня, наступного за останнім днем, у який зобов'язання мало бути виконане, і початок такого перебігу не може бути змінений за згодою сторін. До вимог про стягнення пені застосовується спеціальна позовна давності, яка відповідно до пункту 1 частини першої статті 258 ЦК України становить 1 рік [2].

Законом України «Про відповідальність за несвоєчасне виконання грошових зобов'язань » обмежено розмір пені, який можна стягувати з юридичних осіб та фізичних осіб-підприємців і обчислюється від суми простроченого платежу та не може перевищувати подвійної облікової ставки Національного банку України, що діяла у період, за який сплачується пеня. Положення вказаного Закону не розповсюджуються на фізичних осіб, внаслідок чого, при стягненні банками заборгованості за кредитами, дуже часто можна спостерігати як банки намагаються стягнути пеню, яка перевищує суму основного боргу у декілька разів [4].

Крім того, відповідно до статті 625 ЦК України встановлена можливість стягнення інфляційних витрат та 3 відсотків річних за весь час прострочення виконання зобов'язання [2].

Інфляційні нарахування на суму боргу, сплата яких передбачена частиною другою статті 625 ЦК України, не є штрафною санкцією, а виступають способом захисту майнового права та інтересу, який полягає у відшкодуванні матеріальних втрат кредитора від знецінення коштів внаслідок інфляційних процесів за весь час прострочення в їх сплаті [2]. Зазначені нарахування здійснюються окремо за кожен період часу, протягом якого діяв відповідний індекс інфляції, а одержані таким чином результати підсумовуються за весь час прострочення виконання грошового зобов'язання.

Сплата 3 % в річних від простроченої суми (якщо інший їх розмір не встановлений договором або законом), так само як й інфляційні нарахування, не мають характеру штрафних санкцій і є способом захисту майнового права та інтересу кредитора шляхом отримання від боржника компенсації (плати) за користування ним коштами, належними до сплати кредиторові. До вимог про стягнення інфляційних витрат та 3 % річних застосовується загальна позовна

давність тривалістю 3 роки [5].

Що ж , дослідивши предметну область та основні поняття, ми підходимо до перетину сфери господарської діяльності та інформаційних технологій.

Автоматизація бухгалтерського обліку – основа ефективного управління Керівнику українського підприємства сьогодні приходиться приймати рішення в умовах невизначеності і ризику, що змушує його постійно тримати під контролем різні аспекти фінансово – господарської діяльності. Ця діяльність відбита у великій кількості документів, що містять різнорідну інформацію. Грамотно оброблена і систематизована вона є деякою мірою гарантією ефективного управління виробництвом. Навпроти , відсутність достовірних даних може привести до невірного управлінського рішення і, як наслідок, до серйозних збитків. Якщо не брати до уваги навмисні протиправні дії, то всі помилки бухгалтерського обліку відбуваються або через недбайливість (наприклад, арифметичні помилки), або через незнання особливостей ведення бухгалтерського обліку в Україні. Такі помилки практично неминучі при ручному обліку чи при використанні застарілих чи нелегальних версій програмних комплексів.

Гарні інформаційні бухгалтерські системи поза залежністю від їхнього масштабу, програмно – апаратної платформи і вартості повинні забезпечувати якісне ведення обліку, бути надійними і зручними в експлуатації.

1.2. Сучасні методи розробки програмного забезпечення

Революція в програмному забезпеченні викликала низку значних змін у способах роботи програмних продуктів . Зокрема, значно збільшилися інтерактивні можливості програм. Розрізняють структурний і об’єктно-орієнтований підходи до розроблення ПЗ.

Структурний підхід до розроблення систем набув широкого поширення в 80х роках. Цей підхід заснований на двох процес а саме проектування та моделювання ПЗ та ми розгляномо більш сучасний підхід, тобто об’єктно-орієнтований.

Об’єктно-орієнтований підхід набув поширення в 1990-х роках . Асоціація

виробників ПЗ Object Management Group (OMG) затвердила як стандартний засіб моделювання цього підходу мову UML.

Порівняно зі структурним підходом об'єктно -орієнтований підхід більшою мірою орієнтований на дані – він розвивається довкола моделей класів. Зростаюче значення використання в мові UML претендентів сприяє незначному зсуву акцентів від даних до функцій. До найбільш важливих категорій додатків, для яких потрібна об'єктна технологія, відносяться обчислювальна обробка для робочих груп і системи мультимедіа.

Об'єктний підхід до розроблення систем слідує ітеративному процесу з нарощуванням можливостей. Єдина модель конкретизується на етапах аналізу , проектування та реалізації.

Та об'єктно-орієнтований підхід призводить до виникнення низки труднощів: – оскільки етап аналізу проводиться на ще вищому рівні абстракції і якщо серверна частина рішення з реалізації передбачає використання реляційної бази даних, семантичний розрив між концепцією і її реалізацією може бути значним; – керування проектом складно здійснювати; – висока складність рішення, що у свою чергу позначається на таких характеристиках ПЗ, як пристосованість до супроводу і масштабованість. Та навіть незважаючі на такі недоліки все одно, даний метод розробки програмного продукту є чи не найпопулярнішим в даний час а отже ми завжди зможемо знайти інформацію та спеціалістів для супроводу інформаційної системи .

У функціональному аспекті інформаційні бухгалтерські системи повинні , принаймні, безпомилково робити:

- арифметичні розрахунки;
- забезпечувати підготовку , заповнення, перевірку і роздруківку первинних документів;
- здійснювати безпомилкове перенесення даних з однієї друкованої форми в іншу;
- робити нагромадження підсумків і числення відсотків довільного ступеня складності;

У залежності від особливостей обліку на підприємстві бази даних можуть мати різну структуру, але в обов'язковому порядку повинні відповідати структурі прийнятого плану рахунків, що задає основні параметри настроювання системи на конкретну облікову діяльність. Модулі системи, що забезпечують проведення розрахунків, підсумовування підсумків і нарахування відсотків, повинні використовувати розрахункові нормативи, що прийняті в поточний час.

Надійність інформаційної системи в комп'ютерному плані означає захищеність її від випадкових збоїв і в деяких випадках від навмисного псування даних .

Як відомо, сучасні ПК є доволі відкритими , тому не можна сто відсотково гарантувати захист чисто на фізичному рівні. Важливо, щоб після збою "зламані" дані можна було легко відновити , а роботу системи поновити в найкоротший строк . Гарні інформаційні бухгалтерські системи відповідають цим вимогам .

Не менш важливо, щоб фірма – розроблювач бухгалтерської програми мала значний досвід роботи і солідну репутацію. При виборі системи варто враховувати ту обставину, що надалі до продавця прийдеться неодноразово звертатися і за порадою або консультацією, і за заміною застарілої версії на більш свіжу .

Елементи інформаційної комп'ютерної системи бухобліку У неавтоматизованій системі ведення бухгалтерського обліку обробка даних про господарські операції легко прослідкується і звичайно супроводжується документами на паперовому носії інформації – розпорядженнями, дорученнями , рахунками й обліковими регістрами, наприклад нескінченними журналами обліку. Аналогічні документи часто використовуються й у комп'ютерній системі, але в багатьох випадках вони існують тільки в електронній формі. Більш того , основні облікові документи (бухгалтерські книги і журнали) у комп'ютерній системі бухгалтерського обліку являють собою файли даних, прочитати або змінити які без комп'ютера просто не можливо.

1.3. Цільове використання програмного продукту

В першу чергу, даним програмним продуктом будуть користуватися бухгалтера та представники юридичного відділу на підприємстві для обрахунку існуючого боргу , визначення 3 % річних, інфляційних втрат, штрафних санкцій (в тому числі пені).

Основним об'єктом є заборгованість яка утворилась через не своєчасну сплату орендної плати, або інших платежів по договору та створення первинних бухгалтерських документів.

Важливою умовою для користувача програми є хоча б базове володіння ПК, наше ПЗ має бути легким у використанні для користувачів від найнижчого до найвищого рівня володіння ПК.

Програмне забезпечення Інформаційна система щодо обрахунку заборгованості та бухгалтерського обліку для установи, що будуть подавати документи у такі місця як:

- державна казначейська служба,
- господарські , цивільні та інші суди,
- правоохоронні органи,
- фізичні особи-підприємці,
- інші установи.

Для цього документи мають відповідати вимогам Національного банку України, та мають бути оформлені згідно ДСТУ 4163:2020 [ДСТУ] та має забезпечувати виконання мінімум таких завдань:

– обрахунок заборгованості на задане число;

– обрахунок 3% річних та інфляційних втрат;

– при підрахунку боргу та пені мають бути враховані не тільки сплати а й рішення суду (якщо такі є);

– попередній перегляд і друк рахунку ;

– формування суми платежу прописом;

– конструктор (дизайнер) форми платіжного доручення (квитанцій);

При підготовці платіжних документів за допомогою даної програми забезпечуються:

– запам'ятовування всіх реквізитів платника;

– послідовна нумерація платіжних документів;

Крім того, в майбутньому система дозволить вести облік одержувачів і платників (постачальників і покупців товарів і послуг), реалізуючи наступні функції:

– довідник одержувачів з можливістю фільтрації, пошуку;

– довідник платників з індивідуальними настройками;

– висновок примірників платіжного доручення довільного формату;

– банківські реквізити

– індивідуальні значення поля «Призначення платежу» для кожного одержувача з можливістю додавання в доручення (квитанцію);

– прості налаштування інтерфейсу форми роботи з платіжними дорученнями і квитанціями;

– формування документа «квитанція-сповіщення».

У зв'язку з частим внесенням поправок до податкового законодавства України і як наслідок зміною форм платіжних доручень і квитанцій повідомлень в системі повинен бути реалізовані :

– конструктор форм (шаблонів) платіжних доручень;

– конструктор форм (шаблонів) квитанцій.

Конструктор форм повинен забезпечувати наступні можливості:

– створення нових форм (шаблонів)документів,

– редагування існуючих форм (шаблонів)документів;

– форматування існуючих шаблонів друкарських форм платіжних документів.

Також планується введення таких функцій:

– ведення бази платників і одержувачів з їхніми рахунками банків, платежів і, безпосередньо , платіжних доручень;

– фільтрація списку платіжних доручень за всіма реквізитами (квитанцій);

– автоматична підстановка рахунку і банківських реквізитів при виборі платника або одержувача в ході формування нового доручення (квитанцій);

– пошук платника або одержувача за кодом ОКПО або найменуванням (рекурсивний пошук);

– ведення архіву бази;

– пошук доручення в базі: за номером, сумою, датою ;

– створення нового типу платежу на основі поточного доручення;

– забезпечення макропідстановки в полі «Призначення платежу ».

– адаптивне ведення довідника платежів.

1.4. Дослідження аналогів та існуючих рішень

Проведемо порівняльний аналіз програм для бухгалтерського обліку. На даному етапі досліджуючи аналогічні програми не можна не звернути увагу на мабуть найпопулярнішу програму в Україні на даний момент, а саме «1С:Бухгалтерія », як пишуть самі розробники:

«1С:Бухгалтерія» — це професійний інструмент бухгалтера, за допомогою якого можна вести бухгалтерський та податковий облік, готувати та подавати обов'язкову звітність. Програма об'єднала у собі всі досягнення попередніх версій та нові рішення , засновані на досвіді практичної роботи бухгалтерів сотень тисяч підприємств та організацій. Зрозумілий облік відповідно до законодавства та потреб бізнесу, економія часу при розрахунках податків, оформленні документів та господарських операцій, ефективна підтримка користувачів у поєднанні з високим комфортом роботи — лише деякі ключові особливості «1С:Бухгалтерії 8».

Дійсно, це дуже хороша та популярна програма яка використовується в Україні доволі давно, та нажаль у зв’язку з політичною ситуацією державний сектор не дуже хоче використовувати дане програмне забезпечення, та це не єдина причина чому ми не можемо обрати це рішення,

- По перше це все ж таки продукт іншої компані, яка щомісяця бере кошти за користування.

- По друге, це не найпростіша програма, багатьом бухгалтерам доводиться проходити курси для того аби вміти користуватися її функціями для прикладу див. рис. 1.1.
- По третє, в дані програмі доволі багато функцій та модулів які не можливо видалити або не завантажувати, тож вона є доволі громіздкою та перевантаженою.
- По четверте, в даній програмі не дуже зручно, я б навіть сказав не можливо нормально вести базу орендодавців та їх заборгованостей.

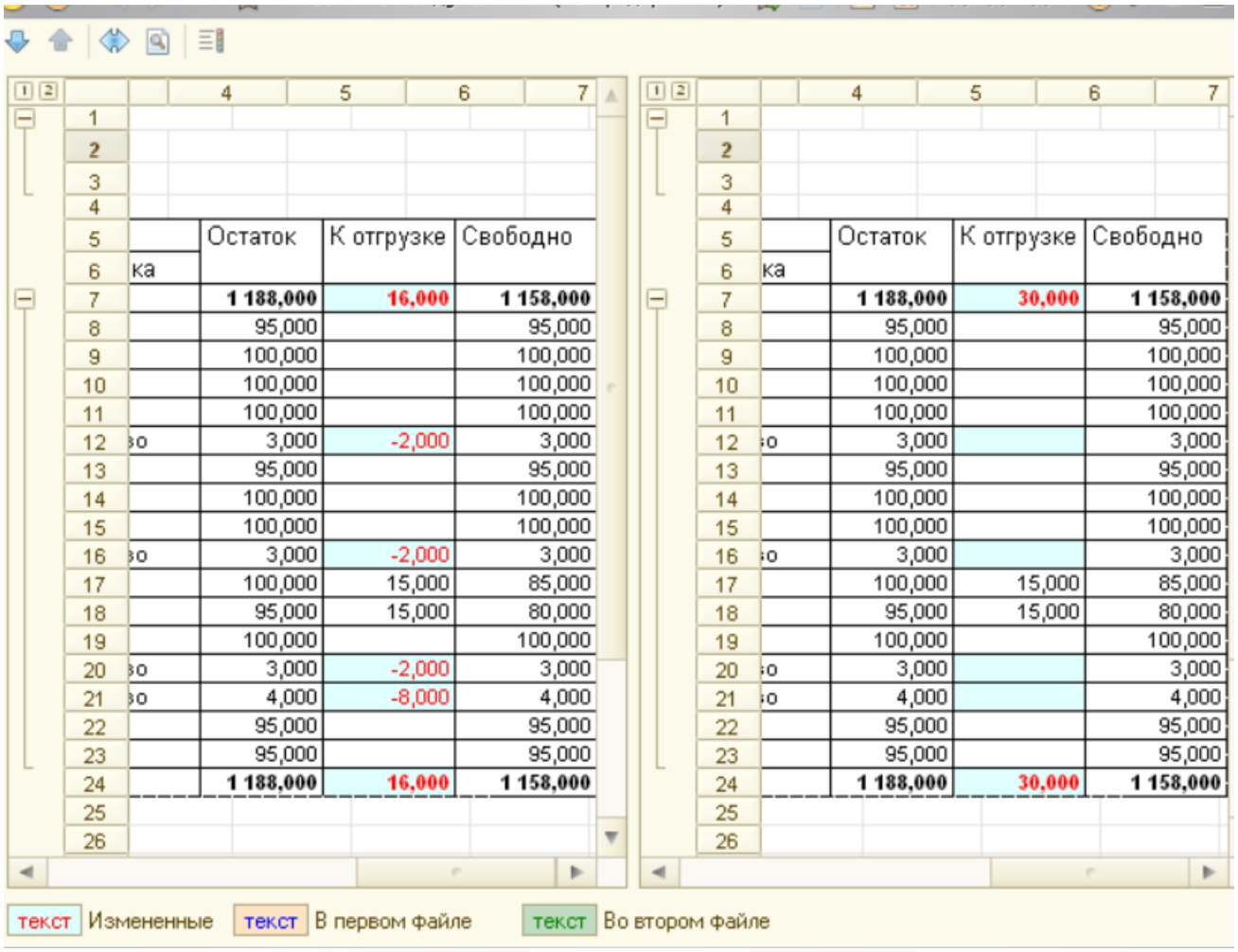

Рис. 1.1. Приклад інтерфейсу 1С Бухгалтерія

Отже ідемо далі, кожного місяця у багатьох орендарів виникає заборгованість що до орендної плати, комунальних платежів і т.д . окрім бухгалтерії дані

обрахунки роблять і юристи при передачі справи до суду аби чітко порахувати заборгованість необхідно врахувати багато чинників, а саме:

- Визначити чітку суму за кожен день прострочення.
- Визначити точний і актуальний індекс інфляції на даний момент.
- Врахувати минулу заборгованість і сплати по ній.
- Мають бути враховані особливості нарахувань з умовами діючого законодавстава (вихідні, святкові дні).

З цим як і юридичному відділу так і відділу бухгалтерії допомагає калькулятор штрафних санкцій Ліга.

Ліга це платна база юридичних та бухгалтерських новин, та окрім цього має модулі для обрахунку заборгованості, основною проблемою є те, що щомісяця в роботі знаходяться сотні боржників і дуже важко для кожного вручну обраховувати суму боргу оскільки заборгованість може бути не за один чи два місяці а за рік-три, а уявіть що вручну треба ввести дані за 24-36 місяців , вибирати дату а ще цю заборгованість необхідно рахувати 2 рази оскільки перший раз рахується борг з урахуванням всієї суми, а другий раз без суми оренди, лише комунальні платежі та інші послуги по договору, оскільки на заборгованість по оренді інфляція вираховується одразу, при виставлені рахунків на оплату боржнику, до того ж кожний місяць це треба робити по новому.

Зрозуміло що це не є зручно , та було б бажано аби все це робилося більш швидше та зручніше, але нажаль усі подібні калькулятори працюють подібним чином, приклад інтерфейсу див. рис. 1.2.

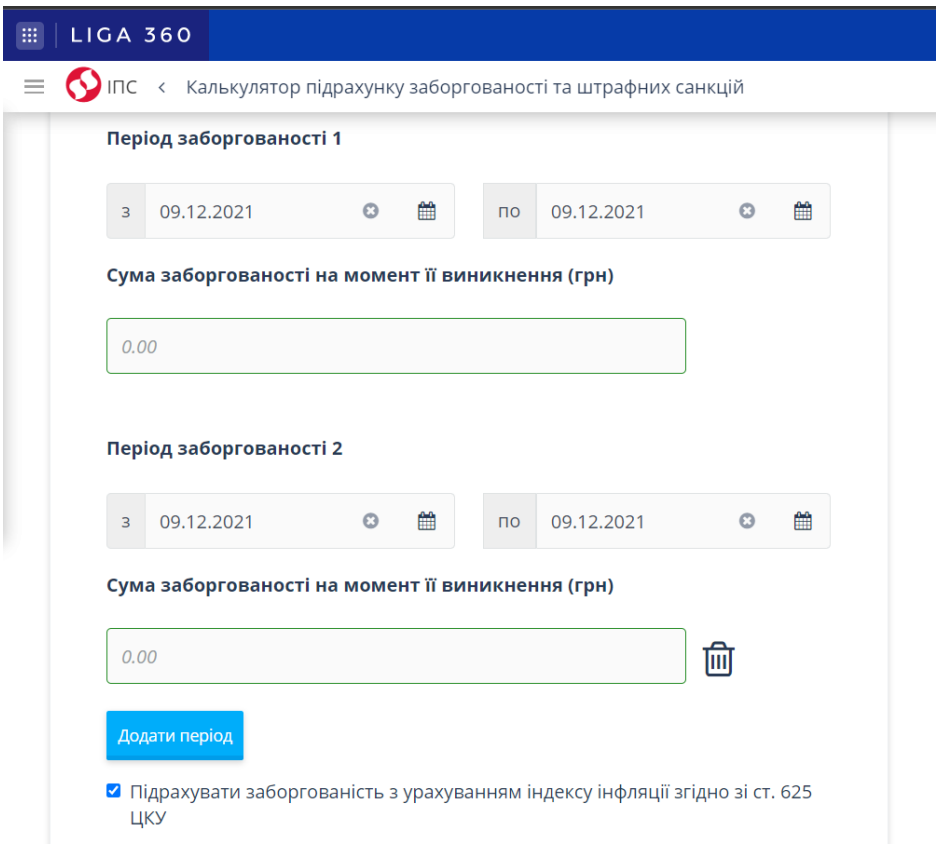

Рис. 1.2. Приклад калькулятора штрафних санкцій Ліга

Окрім цього , це окремий модуль, за який необхідно платити кошти, він умовно безкоштовний але у безкоштовній версії не можна завантажити результат, тільки переглянути , а отже це не підходить і доведеться брати платну версію, що означає щомісячні витрати , що ще раз підтверджує необхідність створення власного програмного продукту.

РОЗДІЛ 2.
ВИБІР ПЛАТФОРМИ ТА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Принцип роботи інформаційної системи

Визначимо, що є інформаційна система, Інформаційна система (у загальному розумінні) — це системи, яка здійснює або в якій відбуваються інформаційні процеси: пошук, збирання, зберігання , передавання й опрацювання інформації.

В інформаційній системі можуть відбуватися одночасно один, два чи кілька процесів.

Опрацювання інформації залежить від змісту вхідної інформації, але під час самого опрацювання інформація не осмислюється, а лише перетворюється згідно з попередньо розробленими алгоритмами.

Інформаційна система (у вузькому розумінні) — це комплекс інформаційних, технічних, програмних та організаційних засобів, необхідних для автоматизованого опрацювання інформації .

В інформаційній системі відбуваються такі процеси:

- введення інформації, отриманої з джерел інформації;
- опрацювання (перетворення) інформації;
- зберігання вхідної і опрацьованої інформації;
- виведення інформації, призначеної для користувача;
- відправка / отримання інформації мережею.

Розробка інформаційної системи передбачає вирішення двох таких завдань:

- наповнення системи даними певної предметної області;
- створення інтерфейсу користувача для отримання необхідної інформації.

Дані в інформаційної системи можуть зберігати в неструктурованому або у структурованому вигляді.

Неструктуровані дані — це звичайні текстові документи (можливо, ілюстровані): статті, реферати, журнали, книги тощо . Системи, в яких зберігають неструктуровані дані, не завжди дають конкретну відповідь на запитання користувача , а можуть видати текст документа або перелік документів, у яких потрібно шукати відповідь.

Структурування даних передбачає задання правил, що визначають їхню форму, тип, розмір, значення тощо.

Бажаючи підкреслити використання електронно-обчислювальної техніки для автоматизації інформаційних процесів, сучасні інформаційні системи часто називають «автоматизованими інформаційними системами». Як інформаційну систему можна розглянути багато об'єктів: телебачення, мережу мобільного зв'язку, цифрові фотоапарати і відеокамери, людине. До інформаційної системи дані надходять від джерела інформації. Ці дані надсилають на зберігання чи певного опрацювання у системі й потім передають споживачеві).

Споживачем може бути людина, пристрій або інша інформаційна система . Між споживачем та власне інформаційною системою може бути встановлено зворотний зв'язок (від споживача до блоку приймання інформації).

Також треба визначити інформаційні потреби, тобто інформація яка буде знаходитись у комп’ютерній системі для здійснення обрахунку заборгованості а також обліку первинних документів підприємства. Він включає в себе опитування користувачів для того, щоб зрозуміти їх потреби. Слід з'ясувати наступне:

– які дані використовуються;

– чи зможе нова система спільно використовувати будь-які з цих даних;

– хто буде вводити дані в базу і в якій формі;

– як часто будуть змінюватися дані ;

– чи достатньо буде однієї бази або буде потрібно кілька баз даних з різними структурами ;

– яка інформація є найбільш чутливою до швидкості її вилучення та зміни.

Після визначення потреб та даних які будуть використовуватися ми переходимо до конкретизації.

Дослідження предметної області показало, що для зберігання бази даних Інформаційної системи щодо обрахунку заборгованості та бухгалтерського обліку досить одного файлу, який буде поміщений на SQL-сервер . База даних імовірно буде не дуже великою , що не знизить швидкість вилучення інформації.

Система повинна містити дані про орендарів. Для підвищення інформативності системи необхідно буде у майбутньому реалізувати у системі Довідник з можливістю фільтрації і пошуку та Довідник платників з індивідуальними налаштуваннями . Наявність таких Довідників дозволяє вести облік. Крім того, на підставі даних цих Довідників формуються платіжні доручення та квитанції.

Наша система повинна мати змогу видати інформацію що до суми сплати орендного платежу в даний місяць для заданої компанії та в разі якщо допущена прострочка по сплаті врахувати це та нарахувати штрафні санкції, 3% річних та інфляційні втрати , зрозуміло що якщо боргу більше як за місяць врахувати минулі суми да додати до боргу на цей місяць, ми повинні мати змогу сформувати рахунок на сплату та дати чек про оплату суми/боргу.

2.2 . Вибір архітектури системи

В основі широкого розповсюдження локальних мереж комп'ютерів лежить відома ідея поділу ресурсів. Висока пропускна здатність локальних мереж забезпечує ефективний доступ з одного вузла локальної мережі до ресурсів, що знаходяться в інших вузлах . Розвиток цієї ідеї призводить до функціонального виділення компонентів мережі. Розумно мати не тільки доступ до ресурсів віддаленого комп'ютера, але також отримувати від цього комп'ютера деякий сервіс, специфічний для ресурсів даного роду і програмні засоби для забезпечення, яких недоцільно дублювати їх в декількох вузлів, тобто відбувається поділ робочих

станцій і серверів локальної мережі.

Робоча станція призначена для безпосередньої роботи користувача або категорії користувачів і володіє ресурсами, відповідними локальним потребам даного користувача.

Сервер локальної мережі повинен володіти ресурсами, відповідними його функціональному призначенню і потребам мережі [6 , с.56].

Практично всі сучасні СУБД використовують у своїй роботі технологію «клієнт-сервер» і СУБД MSSQL Server не є винятком. «Клієнт -сервер» – це модель взаємодії комп'ютерів в мережі . Щодо систем баз даних архітектура «клієнт-сервер » цікава і актуальна головним чином тому, що забезпечує просте і відносно дешеве рішення проблеми колективного доступу до баз даних в локальній мережі.

У деякому роді системи баз даних , засновані на архітектурі «клієнт-сервер», є наближенням до розподілених систем баз даних. У мережі один і той же комп'ютер може виконувати роль як клієнта, так і сервера. У загальному випадку, щоб прикладна програма, яка виконується на робочій станції, могла використовувати послугу деякого сервера, як мінімум потрібно деякий інтерфейсний програмний код, що підтримує такого роду взаємодію.

З цього, власне, і випливають основні принципи системної архітектури «клієнт-сервер» [7, с.89]. Система розбивається на дві частини, які можуть виконуватися в різних вузлах мережі, клієнтську і серверну частини.

Прикладна програма або кінцевий користувач взаємодіють з клієнтською частиною системи, яка в найпростішому випадку забезпечує просто міжмережевий інтерфейс. Клієнтська частина системи при потребі звертається по мережі до серверної частини. Термін «сервер баз даних» зазвичай використовують для позначення всієї СУБД, заснованої на архітектурі «клієнт-сервер», включаючи і серверну, і клієнтську частини. Такі системи призначені для зберігання і забезпечення доступу до баз даних. Модель функціонування сервера баз даних наведена на рис. 2.1.

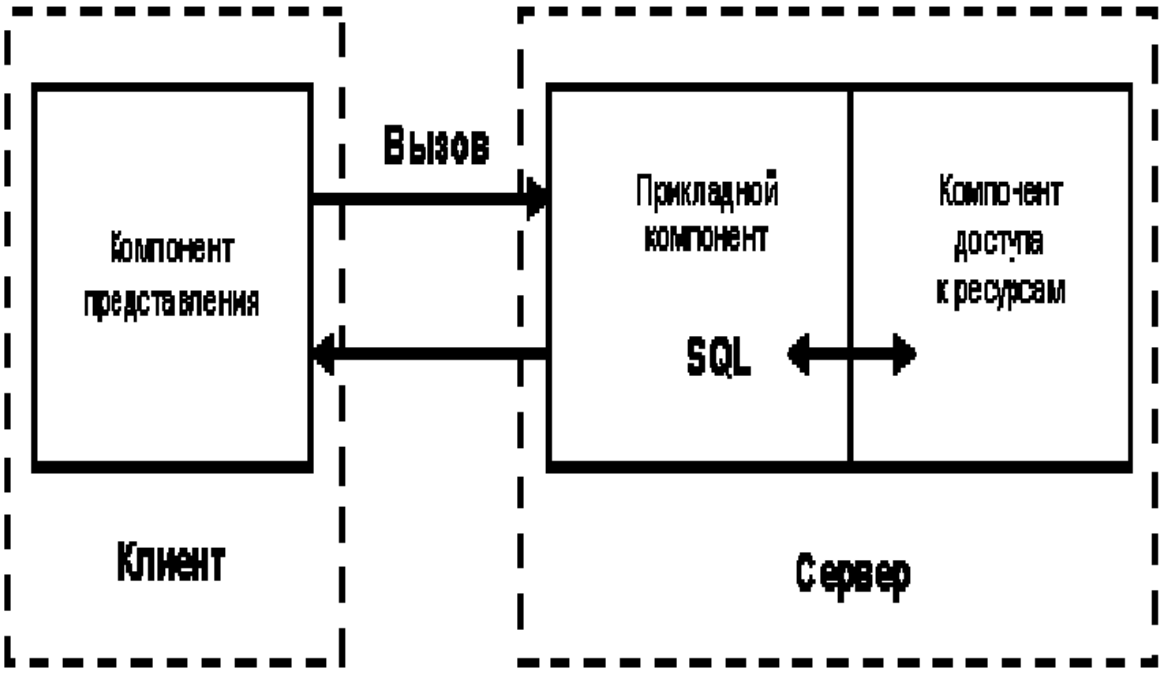

Рис 2.1. Модель функціонування сервера бази даних

Хоча зазвичай одна база даних цілком зберігається у одному вузлі мережі і підтримується одним сервером, сервери баз даних є простим і дешевим наближенням до розподілених баз даних, оскільки загальна база даних доступна для всіх користувачів локальної мережі.

2.3 . Вибір та забезпечення доступу до бази даних

Підключення до баз даних відбувається за допомогою ADO (Active Data Objects) – це високорівнева компонент технології доступу до даних від Microsoft. Даними для ADO можуть бути як звичні таблиці Access або серверні бази MS SQL або Oracle, так і Microsoft Active Directory Service, XML файли і т.п. ADO-технологія , працює через інтерфейс OLE DB [8]. Середовище швидкої розробки додатків .Net Framework забезпечує пряму взаємодію з SQL-сервером MSSQL Server і дозволяє підтримувати повний життєвий цикл інформаційного додатка , починаючи від розробки простого прототипу додатка і закінчуючи його супроводом і модернізацією [9].

Доступ до бази даних від прикладної програми або користувача виробляється шляхом звернення до клієнтської частини системи. Як основний інтерфейс між клієнтської і серверної частинами виступає мова баз даних SQL. Це мова по суті справи є поточним стандартом інтерфейсу СУБД у відкритих системах. Збірна назва SQL-сервер відноситься до всіх серверів баз даних, заснованих на SQL.

Дотримуючись обережності при програмуванні, можна створювати прикладні інформаційні системи в класі SQL серверів. Сервери баз даних , інтерфейс яких базується виключно на мові SQL, мають свої переваги і свої недоліки. Очевидна перевага стандартність інтерфейсу.

В ідеалі, клієнтські частини будь-якої SQL-орієнтованої СУБД могли б працювати з будь-яким SQL-сервером незалежно від того, хто його створив. Недолік теж досить очевидний. При такому високому рівні інтерфейсу між клієнтської і серверної частинами системи на стороні клієнта працює занадто мало програм СУБД . Це нормально, якщо на стороні клієнта використовується малопотужна робоча станція. Але якщо клієнтський комп'ютер має достатню потужність, то часто виникає бажання покласти на нього більше функцій управління базами даних, розвантаживши сервер, який є слабким місцем всієї системи [10, с. 112].

Одним з перспективних напрямків СУБД є гнучка конфігурація системи , при якій розподіл функцій між клієнтською і призначеної для користувача частинами СУБД визначається при установці системи. У типовому випадку на стороні клієнта СУБД працює тільки таке програмне забезпечення , яке не має безпосереднього доступу до баз даних, а звертається для цього до сервера з використанням мови SQL.

СУБД MSSQL Server – це система управління базами даних, що поставляється корпорацією Microsoft для побудови додатків з архітектурою клієнт-сервер довільного масштабу. Сервер MSSQL Server – Система управління реляційними базами даних (РСУБД) розроблена компанією Microsoft. Основна мова запитів - Transact-SQL, створений спільно Microsoft і Sybase. Transact-SQL є реалізацією еталона ANSI/ISO структурованої мови запитів (SQL) з розширеннями.

Використовується для роботи з базами даних розміром від персональних до великих баз даних у масштабі підприємства; конкурує з іншими СУБД у цьому сегменті ринку.

Якщо для розробки систем використана інтегрована середа Visual Studio, то працювати з MSSQL Server можливо через ADO.NET, що забезпечить вам легкий доступ до даних. MSSQL Server підтримує роботу з використовуючи кілька мережевих протоколів: TCP/IP, NetNEUI, IPX/SPX, та забезпечує роботу механізму оптимістичного блокування на рівні запису. Це означає, що сервер блокує тільки ті записи , які реально були змінені користувачем, і не блокує всю сторінку даних цілком. Ця особливість ще більше знижує ймовірність конфліктів при багато користувальницькому режимі. У порівнянні з багатьма іншими СУБД, MSSQL Server надає дуже ефективний механізм тригерів: кожна таблиця може мати велику кількість тригерів, які виконуються автоматично при вставці, зміні або видаленні кожного окремого запису, до або після цих подій.

2.4. Вибір типу програми

Обираючи систему визначення заборгованості та бухгалтерського обліку для підприємства, керівник має вирішити, якою вона має бути — «десктопною» чи онлайновою. Кожен варіант має свої плюси та мінуси. Розгляньмо їх.

Десктопний додаток встановлюється в середовищі конкретної операційної системи. Програма з тією самою назвою , яка запускається у Windows, у Linux або в MacOS, — це, фактично, три цілковито різні програми. І якщо, наприклад, усі ваші менеджери проектів працюють у Windows, а ви вирішили працювати на улюбленому MacBook, увійти в систему ви зможете, або якщо в MacOS встановлено емулятор Windows, або якщо розробник системи передбачив варіант програми для MacOS.

З веб-програмами це обмеження зникає. Власне, вони взагалі не вимагають встановлення — вони вже встановлені на сервері . Для доступу до онлайн-системи потрібний тільки браузер і підключення до Інтернету. Керувати проектами можна

з будь-якого пристрою, до того ж не тільки з комп'ютера, а й із планшета або смартфона. Вам не потрібно шукати комп'ютер, щоб перевірити що-небудь у базі.

Але враховуючи те що, на державних підприємствах майже 99% користувачі ОС Windows, важко сказати що в нашому випадку це щось змінює.

Ідеальних програм не існує: у будь -якій із них виникають помилки, які часом заганяють користувачів у глухий кут. Мінус веб -додатків у тому, що коли помилка критична , робота зупиняється відразу в усіх користувачів системи . Плюс — усунути таку помилку на сервері означає усунути її відразу в усіх.

Ситуація з десктопними додатками складніша: їхня робота залежить від особливостей конкретного комп'ютера (версії ОС, наявності додаткових бібліотек, системних налаштувань тощо). Щоб виправити помилку , службі підтримки доведеться з'ясувати, що викликає неполадку саме на вашому комп'ютері. Це вимагає часу й зусиль як від розробників, так і від вас.

Програми регулярно допрацьовуються й оновлюються . Кожне оновлення усуває знайдені «баги» та додає нові функції. Зазвичай оновлення потрібно встановлювати, щойно вони стають доступними.

В онлайнових додатках встановлення оновлень відбувається без вашої участі й навіть без вашого відома: увійшовши вчергове, ви просто побачите новий інтерфейс. Щоб оновити десктопну програму , вам і всім вашим менеджерам доведеться самостійно завантажити та встановити додаток. Іноді під час цього виникають помилки через особливості операційної системи .

Онлайнові програми «спілкуються» одна з одною за допомогою програмного інтерфейсу (API), об'єднуючись у справжні екосистеми. Дані мігрують з однієї системи в іншу автоматично, усуваючи необхідність вводити вручну ту саму інформацію до різних баз.

Десктопні програми також можуть підключатися до онлайнових баз, але вони вкрай рідко взаємодіють одна з одною : це завдання технічно складніше. Їхнє «спілкування» зазвичай виглядає як операції експорту й імпорту файлів , які виконуються вручну.

Ситуація складається так, що в майбутньому ізольовані програми вимруть як

клас : ніхто не захоче марнувати час на ручний «копіпаст».

Ще рідкісніший випадок — звернення онлайнової програми до десктопної. Онлайнова система керування перекладацькими проектами навряд чи зможе ввійти в базу проектів CAT-програми, встановленої на комп'ютері. Через це десктопні додатки мають набагато менше перспектив інтеграції — а отже, і майбутнього.

Якщо система керування встановлена на вашому комп'ютері, швидкість її роботи залежатиме від потужності його процесора, обсягу операційної пам'яті й налаштувань операційної системи. У веб -додатках ці фактори теж важливі, однак вони враховані із самого початку — у параметрах сервера .

Проблемою іноді стає швидкість з'єднання з Інтернетом . До того ж, можливі затримки з боку сервера, якщо одночасно кілька користувачів надішлють неочікувано багато запитів.

Обов'язкову наявність підключення до Інтернету теж можна вважати недоліком: за його відсутності працювати в онлайновій системі, на відміну від десктопної, неможливо. Проте зараз у світі кількість місць, де ця проблема залишається актуальною , постійно скорочується. Тому розробники онлайнових систем просто не зважають на неї та свідомо залишають цю нішу розробникам десктопних програм.

Інформаційна безпека це найсуперечніша тема. Люди та компанії бояться використовувати веб-програми та зберігати дані в хмарі, хвилюючись за цілісність і конфіденційність інформації . Вважається, що десктопні системи з цього погляду безпечніші: якщо дані фізично розміщені на комп'ютері , то вони під контролем. Проте чи так це насправді? Слід відповісти на кілька простих запитань.

Ваш комп'ютер підключений до Інтернету? Якщо ви не працюєте над проектом підвищеного рівня секретності в підземному бункері, то відповідь буде ствердною. А отже, теоретично ваш комп'ютер доступний ззовні. І якщо ви не експерт із комп'ютерної безпеки, то зловмиснику зламати ваш комп'ютер легше, ніж хмарний сервіс, про захист якого подбали розробники.

Ви напевне встановлюєте оновлення на десктопні програми. Чи перевіряєте ви, що саме відбувається на вашому комп'ютері в момент встановлення ?

Теоретично під час цього розробники можуть, зокрема , завантажити вашу базу даних. Їх стримують не так технічні обмеження, як етичні й репутаційні .

У вас налаштоване резервне копіювання даних на інший диск? Якщо ні, ви ризикуєте втратити їх, оскільки диск може раптово «посипатися». У кращому разі ви звернетеся до спеціалістів для відновлення даних, у гіршому — втратите їх назавжди .

В онлайнових сервісах — зокрема, у системах керування перекладацькими проектами — здійснюється регулярне резервування даних. Копії зазвичай зберігають в окремому дата-центрі, щоб виключити втрату даних унаслідок фізичного знищення дата -центру через потоп, пожежу тощо.

До того ж, багато з тих, хто боїться зберігати дані в хмарі, уже давно так роблять . Найпевніше, ваш поштовий сервер має веб-інтерфейс , а отже, листи й файли з конфіденційною інформацією вже зберігаються на деякому сервері. Існує навіть вірогідність, що ви передаєте файли через онлайн-сховища на кшталт Google Drive, OneDrive або Dropbox. Тобто ви, найімовірніше, уже зберігаєте деякі дані в хмарі. Питання лише в тому, чи дозволите ви додати їх ще в одну систему.

Отже, конфіденційність — це питання довіри не так до технології, як до її розробника.

Простіше кажучи, запитання про конфіденційність розпадається на три запитання:

Чи гарантують розробники сервісу, що дані не буде втрачено?

Чи захистять вони дані від зловмисників?

Чи не викрадуть вони ці дані самі, щоб продати їх конкурентам або скопіювати мій бізнес?

На перші два запитання ми відповіли вище: розробники напевне захистять і збережуть дані краще за вас.

Серйозний бізнес — спокійний бізнес. Розробники заробляють репутацію та довіру, привертають цим багато користувачів , а потім отримують від них платежі за підтримку й обслуговування системи. Тобто розробники зацікавлені в захисті

даних користувачів. Важко навіть сказати , хто зацікавлений у цьому захисті більше — розробники чи користувачі.

Який висновок ми можемо з цього зробити? Десктопні додатки мають перевагу, коли ви обмежені жорсткими рамками та не плануєте за них виходити. Веб-системи гнучкіші, і їх легше розвивати. Та у нашому випадку найзручніше використати саме десктоп додаток, він дасть нам можливість перетворити користувацький ПК на окрему систему, а отже бухгалтеру не треба мати сервер чи щось інше для того аби додаток вдало працював.

2.5. Вибір платформи та технології реалізації

Різні розробники по різному підходять до вибору програм і технологій створення систем. Розглянемо переваги і недоліки різних підходів в реалізації конкретних систем обліку. Однією з технологій є Windows Forms, це гарна але доволі стара технологія.

Насамперед, форми підходять новачкам , тому що вони можуть писати програми у Visual Studio без глибоких знань в об’єктно -орієнтованому програмуванні. Студія передбачає використання стандартних об’єктів (кнопка, надпис, текстове поле, картинка тощо) з інтуїтивно зрозумілими параметрами (висота, ширина і т .п.). Більш того, в студії процес зміни деяких з параметрів, додавання нового об’єкта в форму навіть не потребує написання коду - все це є візуалізованим. Користувач просто перетягує необхідний об’єкт з панелі на форму.

Виконуваний код - класи, що реалізують API для Windows Forms, не залежать від мови розробки. Тобто програміст однаково може використовувати Windows Forms як при написанні ПЗ на C , так і на інших мовах програмування.

Але не буває нічого без недоліків, тож розглянемо Windows Forms з іншої сторони.

Значним недоліком Windows Forms є те, що якщо над проектом працюють не лише програмісти, а й дизайнери, то їм доводеться працювати дуже тісно, щоб

вийшов якісний проект. Мається на увазі, щоб дизайнер малював інтерфейс, віддавав його програмісту, а той, в свою чергу, реалізовував його (відволікаючись від свого безпосереднього завдання), а не програмував логіку програми.

Одним з частих підходів є використання систем сервер-клієнта, в яких в якості клієнта виступає браузер, а як сервер зв'язку з Web-сервера і сервера додатків [5].

Мова HTML орієнтована на представлення даних, а не на їх створення, занадто ускладнює редагування вже створених даних. Це є проблемою багатьох систем заснованих на Web-інтерфейсі, крім того відсутня можливість використання при підготовці платіжних документів технології OLE, що ускладнює вставку складних об'єктів (формул, полів і ін.). Перевагами такого підходу є: – відсутня залежність від операційної системи; – відсутня необхідність в установці і настройці клієнтської частини ; – мова HTML має великі можливості за поданням даних; – відсутність проблем з передачею даних в мережі Інтернет. Альтернативою є використання спеціалізованого програмного забезпечення для створення програмного продукту для обліку та складання первинних бухгалтерських платіжних документів. Даний підхід має наступні переваги: – зручність створення і редагування платіжних документів; – автономність – не потрібне наявність підключення до мережі. Використання спеціалізованого програмного забезпечення для обліку первинних платіжних документів дозволяє [4]:

– мінімізувати обсяг переданння даних;

– реалізувати складні структури документів.

Недоліки спеціалізованого програмного забезпечення:

– можливі обмеження за форматами представлення інформації;

– необхідність установки спеціального клієнтського додатка;

– проблема оновлення версій клієнтів при вдосконаленні програмного забезпечення.

При аналізі аналогів програм для ведення автоматизованого обліку та складання первинних фінансових і бухгалтерських документів на підприємствах, більшість з них вибрали використання спеціалізованого програмного забезпечення ,

так як цей напрямок має значні переваги , такі як: при проектуванні комп’ютерної системи обліку та складання первинних бухгалтерських платіжних документів планувалося мінімізувати обсяг переданої інформації; також, повинна бути врахована автономність, тобто відсутність залежності від наявності зв'язку; зручність і простота у використанні і налаштуванні [4].

Для реалізації Інформаційної системи щодо обрахунку заборгованості та бухгалтерського обліку обрана .Net Framework Windows presentation foundation та мовою C , для створення бази даних СУБД Microsoft SQL Server.

Вважаю що, для того аби дипломна робота була повною ми неодмінно маємо дослідити теоретичні відомості що до доцільності обраної платформи та технології.

Windows Presentation Foundation це система для побудови клієнтських додатків Windows з візуально привабливими можливостями взаємодії з користувачем, графічна (презентаційна) підсистема у складі . NET Framework (починаючи з версії 3.0), що використовує мову XAML [9, с.99].

WPF предустановлена в Windows Vista (. NET Framework 3.0) і Windows 7 (. NET Framework 3.5 SP1). За допомогою WPF можна створювати широкий спектр як автономних, так і запускаються в браузере додатків .

В основі WPF лежить векторна система візуалізації , яка не залежить від дозволу пристроїв виведення і створена з урахуванням можливостей сучасного графічного устаткування. WPF надає засоби для створення візуального інтерфейсу, включаючи мову XAML (Extensible Application Markup Language), елементи управління, прив'язку даних, макети, двомірну і тривимірну графіку, анімацію, стилі, шаблони, документи , текст, мультимедіа та оформлення [10, с.200].

Графічної технологією, яка лежить в основі WPF, є DirectX, на відміну від Windows Forms, де використовується GDI / GDI + [11, с.325]. Продуктивність WPF вище, ніж у GDI + за рахунок використання апаратного прискорення графіки через DirectX.

Також існує урізана версія CLR, яка називається WPF / E, вона ж відома як Silverlight.

XAML являє собою XML, в якому фактично реалізовані класи . NET Framework. Також реалізована модель поділу коду та дизайну, що дозволяє кооперуватися програмісту і дизайнерові. Крім того, є вбудована підтримка стилів елементів, а самі елементи легко розділити на елементи управління другого рівня, які , в свою чергу, розділяються до рівня векторних фігур і властивостей / дій. Це дозволяє легко задати стиль для будь-якого елемента, наприклад , Button (кнопка).

Для роботи з WPF вимагається будь . NET -сумісний мову. У цей список входить безліч мов: C , VB, C + +, Ruby, Python, Delphi (Prism), Lua і багато інших . Для повноцінної роботи може бути використана як Visual Studio, так і Expression Blend. Перша орієнтована на програмування, а друга - на дизайн і дозволяє робити багато речей, не вдаючись до ручного редагування XAML. Приклади цього - анімація , стилізація, стану, створення елементів управління і так далі.

Що таке .NET? .NET – це платформа від Microsoft, яка дозволяє створювати програмні додатки . Перший випуск .NET Framework відбувся в 2002 році. Вважається, що .NET Framework було створено в якості альтернативи платформі Java від компанії Sun. Головна відмінність полягає в тому, що .NET Framework офіційно розрахована на роботу саме з операційними системами родини Microsoft Windows. З того часу вона пройшла довгий шлях від версії 1.0 до 4.8 (18 квітня 2019), і на сьогодні, не дивлячись на появу платформи нового покоління (.NET Core), все ще досить популярна: існує велика кількість програмних продуктів, бібліотек та фреймворків, які написані та розвиваються під .NET Framework.

В 2016 році додатково до .NET Framework було випущено модульну платформу .NET Core, сумісну з різноманітними операційними системами. Іншими словами, вона є кросплатформною . Кросплатформність .NET Core відкрила безліч нових сценаріїв і можливостей її застосування. Це зіграло суттєву роль в просуванні .NET серед розробників і представників бізнесу.

Багато хто вважає, що мова C і платформа .NET – одне і те саме. Звичайно, це не так. Вони, без сумніву, розвиваються озираючись одне на одного, але не мають суворої взаємозалежності. Наприклад, окрім офіційно підтримуваних реалізацій .NET, існують й альтернативні варіанти , як-от Mono, .NET Compact

Framework, .NET Micro Framework та інші. На усіх цих платформах ми можемо використовувати мову C , але до певної міри. З іншого боку, з .NET сумісна не лише C , але й інші мови: F , VB.NET і навіть C++.

Розробники, які знають різні мови, можуть зібратися разом і написати спільний програмний продукт під конкретну .NET-платформу. Елементи цього продукту , написані різними мовами, зможуть комунікувати між собою без жодних проблем. До речі, це пояснює , чому ком'юніті .NET таке велике та різноманітне : воно об'єднує програмістів, які пишуть на різних мовах.

Під .NET створюють не тільки web, але й клієнтські додатки – продукти, які запускаються на персональних комп'ютерах і мобільних пристроях кінцевих користувачів.

Із застосуванням .NET розроблено деякі компоненти операційної системи Windows, серед них – блокнот і калькулятор. Крім того, існує велика кількість додатків під .NET, зроблених індивідуальними розробниками: на цьому ресурсі можна ознайомитися з деякими з них . Розробляються й складніші продукти. Наприклад, для трейдерів – NinjaTrader, Tradesignal. Є ще цікавий додаток для бізнес-аналітиків – Microsoft Power BI, який візуалізує інформацію з будь-якого джерела, спрощує та пришвидшує роботу з великими даними. Для Desktop Client Applications переважно застосовуються технології WPF або Windows Forms – знаючи їх, ви можете створювати складні додатки для стаціонарних комп'ютерів користувачів.

Також існують клієнтські додатки для смартфонів. З допомогою .NET Core і Xamarin.Forms ви можете написати додаток, опублікувати його в Apple або Android Store, і він стане доступним кожному власнику смартфону чи планшету [12].

Саме тому дана технологія та мова програмування були обрані для створення інформаційної системи щодо обрахунку заборгованості та бухгалтерського обліку.

2.6. WPF чи Windows Forms?

Winforms називається Windows Forms. Це графічний користувальницький інтерфейс для настільних додатків Framework.Net. Він має набір керованих бібліотек у рамках .net. Він пропонує розгалужену клієнтську бібліотеку для надання інтерфейсу для доступу до власних елементів графічного інтерфейсу Windows та графіки з керованого коду. WPF скорочується як рамка презентації Windows. Спочатку він був випущений корпорацією Microsoft with.Net Framework 3.0 в 2006 році. Це графічна рамка інтерфейсу користувача для створення програм Windows. WPF – це більше, ніж просто обгортка, це частина рамки .net. Він містить суміш керованого та некерованого коду.

У додатку Windows форми Windows надають обгортку, що складається з набору класів C ++ для розробки програм Windows, і кожен елемент керування у формі програми Windows є конкретним екземпляром класу . Він пропонує різноманітні елементи керування, такі як текстові поля, кнопки, мітки та веб-сторінки , а також параметри для створення спеціального елемента керування. Для цього в Visual Studio є інструмент дизайнера вікон, доступний для управління елементами форми та їх розташування відповідно до потрібного макета для додавання коду для обробки подій.

У формах Windows налаштування додатків – це ще одна функція для створення, зберігання та підтримки інформації. Клас форм Windows може бути розширений за допомогою успадкування для розробки рамки програми , яка забезпечує абстрагування та повторне використання коду . Форми повинні бути компактними з елементами управління на обмеженому розмірі. Форми можна розбити на шматки, упаковані у склади, які можуть автоматично оновлюватись. Проектування програми забезпечує масштабованість та гнучкість з легкістю для налагодження та обслуговування. Форми Windows не можна передавати через межі домену програми.

Основними компонентами архітектури WPF є структура презентації, ядро презентації та mallcore. У WPF елементи інтерфейсу призначені для XAML, тоді як

поведінка може бути реалізована процедурною мовою. З XAML в WPF програмісти можуть працювати паралельно з дизайнерами. WPF – це потужна основа для створення програми для Windows, і вона має чудові функції, такі як прив'язка даних, медіа-сервіси, шаблони, анімації, direct3D та альтернативний ввід.

Розробка додатків WPF може бути здійснена за допомогою таких інструментів Microsoft, як Visual Studio та Expression Blend. VS в основному використовується розробником для створення додатку WPF, тоді як Blend в основному використовується дизайнерами для додатків WPF.

Обидві Winforms проти WPF – це популярний вибір на ринку; Давайте обговоримо деякі основні відмінності Winforms від WPF:

Форми Windows – це не векторний інтерфейс користувача. Тоді як WPF – це векторний графічний шар представленого інтерфейсу на основі графіки. Завдяки векторній основі це дозволяє презентаційному шару плавно масштабувати компоненти інтерфейсу, не маючи жодних проблем із спотворенням розміру.

Форми Windows легше використовувати під час розробки програм , тоді як WPF мало складний у використанні , оскільки для використання елементів керування потрібні добрі знання.

У формах Windows ми можемо налаштувати елементи керування відповідно до вимог. У WPF у нас є стороннє управління, щоб також збагатити можливості додатків.

Форми Windows мають меншу криву навчання. Тоді як WPF має більшу криву навчання, як це потрібно для розуміння повного потоку елементів управління та конструкторської частини.

Форми для Windows менш трудомісткі або менш складні. WPF складніше і витрачає більше часу на отримання речей на місці під час розробки додатків.

Форми Windows не використовуються для розробки нових програм. WPF використовується в основному для розробки нових програм.

Форми Windows – це велика підтримка з точки зору розробників, інтернет - спільноти, бібліотек для будь-якої допомоги при розробці програми для початківців. WPF також має достатню підтримку та бібліотеки для розробки

програм та швидкої підтримки для початківців.

У формах Windows елементи керування важко налаштувати, тоді як у WPF елементи керування можна легко налаштувати так, як це повністю написано з нуля.

Форми Windows погано забезпечують узгодженість. WPF забезпечує більшу узгодженість між програмами.

У формах Windows інтерфейс користувача розроблений за допомогою мови коду ділової логіки. У WPF він використовує XAML як мову розмітки для проектування частини інтерфейсу програми.

Форми Windows в основному базуються на пікселях, тоді як WPF не на основі пікселів, що дозволяє масштабувати частину інтерфейсу для програми.

Windows формує підтримку прив'язки даних обмежено , тоді як WPF повністю підтримує прив'язку даних.

Форми Windows не використовуються з різними темами чи скінами. WPF в основному є шкірним або тематизованим, де різні інтерфейси або теми можуть використовуватися для інтерфейсу користувача.

Форми Windows вимагають менше зусиль для проектування інтерфейсу користувача. WPF вимагає більше зусиль, оскільки більшу частину роботи потрібно виконати самостійно.

Дослідивши віддімінності між обома діалогами, я точно впевнений в тому що WPF є найкращим вибором у даній ситуації.

РОЗДІЛ 3.
ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Проектування моделі бази даних інформаційної системи

На цій стадії ми будемо проводити аналіз об'єктів реального світу, які необхідно зобразити в базі даних і складається з наступних кроків:

– визначення діяльності нашої предметної області .

Тобто мова йде про господарську діяльність підприємства , можна розглядати ведення обліку платіжних документів, виставлення рахунків орендарям, підрахунку існуючої заборгованості;

– ідентифікацію об'єктів , які здійснюють цю функціональну діяльність і ідентифікувати всі сутності і взаємозв'язку між ними.

– ідентифікацію характеристик цих сутностей;

– ідентифікацію взаємозв'язків між сутностями .

Процес «визначення заборгованості та ведення бухгалтерського обліку » ідентифікує такі сутності: Орендар; Тип платежу; Платіж , Документ, Розрахунок.

Сутність Орендар може включати такі характеристики як:

– ідентифікатор підприємства за ЄДРПОУ або ІПН (для фізичних осіб-підприємців),

– адреса об’єкту ,

– сума плати за оренду,

– сума плати за комунальні послуги,

– загальна сума до оплати.

Сутність Борг може включати такі характеристики як:

– ідентифікатор підприємства за ЄДРПОУ або ІПН (для фізичних осіб-підприємців),

– дата формування боргу,

– сума боргу за 3% річних,

– сума боргу за інфляційні втрати,

– сума боргу пені,

– сума загального боргу.

Сутність Рахунок може включати такі характеристики як :

– № розрахункового рахунку,

– тип рахунку,

– ідентифікатор підприємства за ЄДРПОУ або ІПН (для фізичних осіб-підприємців),

– сума до сплати,

– сума боргу.

Сутність Платіж може включати такі характеристики як:

– назва компанії,

– № банківського рахунку,

– сума,

– дата,

– коментар.

Сутність Об’єкт нерухомості може включати такі характеристики як:

–назва об’єкту,

–адреса об’єкту,

–площа.

Сутність Первинний документ може включати такі характеристики як:

– ідентифікатор,

– тип документу.

Наступний етап проектування БД полягає у встановленні відповідності між

сутностями і характеристиками предметної області і відносинами і атрибутами в нотації обраної СУБД. Оскільки кожна сутність реального світу володіє якимись характеристиками, в сукупності утворюють повну картину її прояви, можна поставити їм у відповідність набір відносин (таблиць) і їх атрибутів (полів).

Перерахувавши всі відносини і їх атрибути , вже на цьому етапі можна почати усувати зайві позиції. Кожен атрибут повинен з'являтися тільки один раз і треба вирішити, яке відношення буде власником якого набору атрибутів.

Потім визначаються атрибути, які унікальним чином ідентифікують кожен об'єкт . Це необхідно для того, щоб система могла отримати будь-яку одиничну рядок таблиці.

При проектуванні бази даних, наступним етапом є вироблення правил, які будуть встановлювати і підтримувати цілісність даних.

На наступному етапі встановлюються зв'язки між об'єктами (таблицями і стовпцями) і виробляється дуже важлива операція для виключення надмірності даних – нормалізація таблиць. Існує кілька типів зв'язків: зв'язок «один-до-одного»; зв'язок «один-добагатьох»; зв'язок «багато-до-багатьох». Зв'язок «один-до-одного » являє собою найпростіший вид зв'язку даних, коли первинний ключ таблиці є в той же час зовнішнім ключем, що посилаються на первинний ключ іншої таблиці. Зв'язок «один-до-багатьох » в більшості випадків відображає реальну взаємозв'язок сутностей в предметної області. Вона реалізується вже описаною парою "зовнішній ключ – первинний ключ", тобто коли визначено зовнішній ключ, що посилається на первинний ключ іншої таблиці. Зв'язок «багато-до-багатьох » в явному вигляді в реляційних базах даних не підтримується. Однак є ряд способів непрямої реалізації такого зв'язку, які з успіхом відшкодовують її відсутність. Один з найбільш поширених способів полягає у введенні додаткової таблиці, рядки якої складаються із зовнішніх ключів, що посилаються на первинні ключі двох таблиць. У базі даних , що спроектована для реалізації комп’ютерної системи обіку первинних бухгалтерських документів використовується зв'язок «один-до - багатьох». Після визначення таблиць, полів, індексів і зв'язків між таблицями слід нормалізувати проектовану базу даних. Важливість нормалізації полягає в тому,

що вона дозволяє розбити великі відносини, які містять велику надмірність інформації, на більш дрібні логічні одиниці, що групують тільки дані, об'єднані «по -природі».

Таким чином, ідея нормалізації полягає в наступному. Кожна таблиця в реляційній базі даних задовольняє умові, відповідно до якого в позиції на перетині кожного рядка і стовпця таблиці завжди знаходиться єдине значення, і ніколи не може бути безлічі таких значень. Процес нормалізації включає: – усунення повторюваних груп (приведення до 1НФ); – видалення частково залежних атрибутів (приведення до 2НФ); – видалення транзитивно залежних атрибутів (приведення до 3НФ).

Після проведення всіх вище перерахованих етапів проектування бази даних комп'ютерної системи обліку первинних бухгалтерських документів, схема бази даних представлена на рис. 3.1.

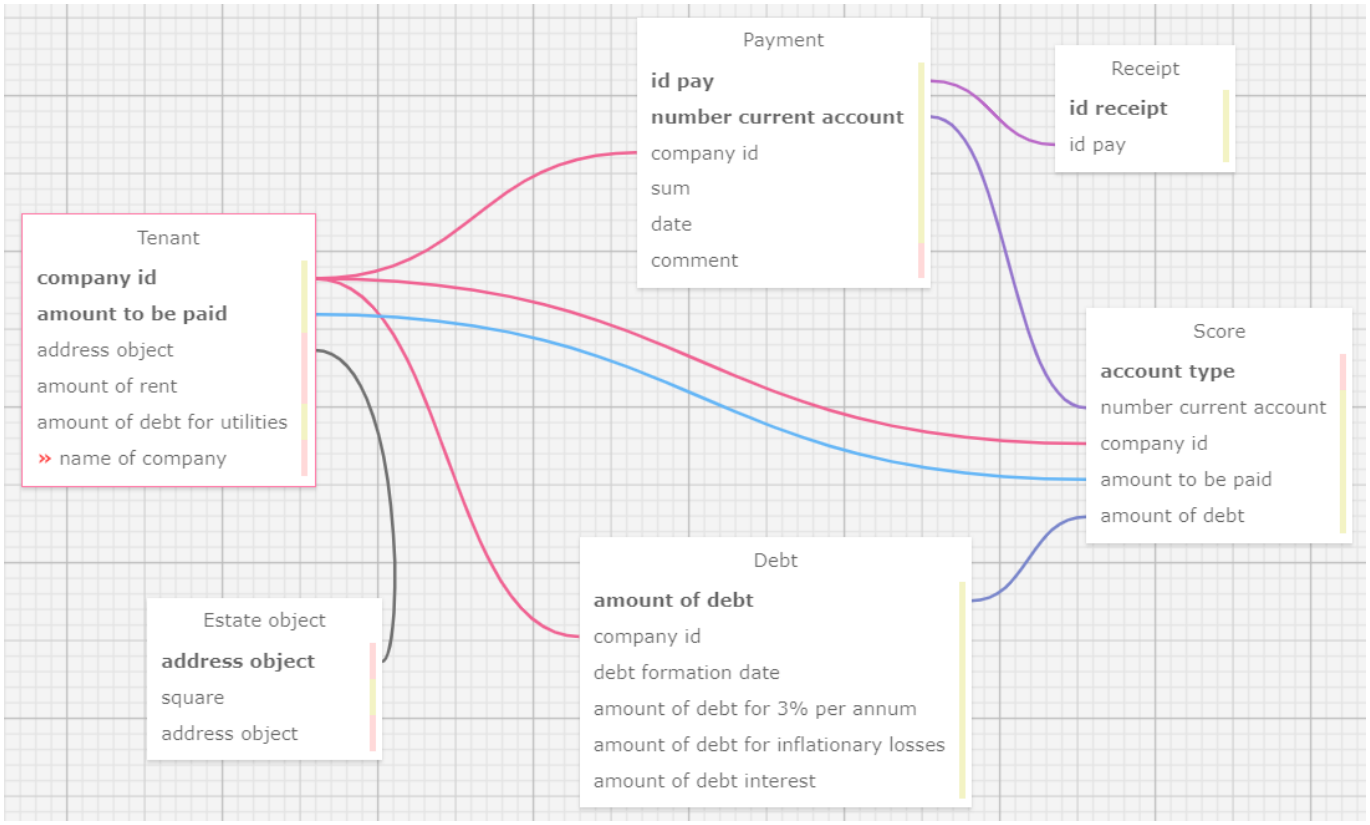

Рис. 3.1. Схема бази даних

3.2. Визначення зв'язків між сутностями бази даних

Виходячи з вимог до розробки комп'ютерної системи обліку первинних бухгалтерських документів , а також враховуючи вимоги до інформації, що зберігається, було отримано ряд таблиць реляційної бази даних. В якості СУБД, для бази даних Інформаційної системи щодо обрахунку заборгованості та бухгалтерського обліку буде використовуватися MSSQL Server, який дозволяє зберігати і обробляти великі обсяги інформації в умовах одночасної роботи з БД множини клієнтських додатків. На підставі врахування всіх вимог до системи створені таблиці бази даних, які мають наступні поля (табл. 3.1 – 3.6).

Таблиця «Орендар» – містить інформацію про всіх орендарів наявних в базі даних підприємства (див. таб. 3.1).

Таблиця 3.1

Сутність «Орендар»

Ім'я поля	Тип
company id	int
address object	varchar (50)
amount of rent	money
amount of debt for utilities	money
amount to be payd	money
name of company	varchar (50)

Таблиця «Об'єкт нерухомості» – містить інформацію про нерухоме майно яке знаходиться на балансі підприємства, майно може бути і не орендоване але просто знаходитьсь в нашій базі даних, наприклад, зараз це майно не орендується, або орендар щойно з'їхав, або закінчився термін договору, під назвою майна розуміється назва корпусу або тип приміщення, наприклад, склад чи корпус Б і тд (див. таб. 3.2).

Таблиця 3.2

Сутність «Об’єкт нерухомості »

Ім'я поля	Тип
address object	varchar (50)
square	float
name of object	varchar (50)

Таблиця «Платіж» – містить інформацію про усі платежі що поступили на рахунок підприємства, важливо що ці платежі будуть підтягуватись лише в тому випадку якщо платник правильно вкаже призначення платежу в інакшому випадку ці сплати будуть просто висіти на банківському рахунку (див. таб. 3.3).

Таблиця 3.3

Сутність «Платіж»

Ім'я поля	Тип
id pay	int
company id	int
number current account	int
sum	money
date	datetime
comment	text

Таблиця «Чек про сплату» – містить інформацію про усі чеки про сплату що були сформовані, важливо те, що ми не можемо сформувати чек про сплату якщо орендарю не був виставлений рахунок, оскільки важливою умовою сплати по договору оренди є виставлення рахунку, тому якщо в цьому місяці чомусь орендарю не виставили рахунок на сплату, сформувати чек не вдасться (див. таб. 3.4).

Таблиця 3.4

Сутність «Чек про сплату»

Ім'я поля	Тип
id receipt	int
Id pay	int

Таблиця «Рахунок» – містить інформацію про усі рахунки на сплату що були створені.

Таблиця 3.5

Сутність «Рахунок»

Ім'я поля	Тип
number current account	int
account type	varchar
company id	int
amount do be pay	money
amount of debt	money

Таблиця «Борг» – містить інформацію про заборгованість орендарів.

Таблиця 3.6

Сутність «Борг»

Ім'я поля	Тип
amount of debt	money
company id	int
debt formation date	datetime
amount of debt for 3% per annum	money
amount of debt for inflationary losses]	money
amount of debt interest	money

Таким чином, цілісність даних бази даних системи встановлена. Заключним етапом проектування бази даних є вирішення питання надійності даних і при необхідності, збереження секретності інформації . Для цього необхідно відповісти на наступні питання : хто буде мати права на використання бази даних, хто буде мати права на модифікацію , вставку і видалення даних, чи потрібно робити розмежування прав доступу у системі; яким чином забезпечити загальний режим захисту інформації [10].

Інформаційна система обрахунку заборгованості та бухгалтерського обліку, не є мережевою і передбачає однокористувальницький режим, тому і установку розмежувань прав доступу проводити не доцільно.

3.3. Проектування інтерфейсу користувачів системи

Основні фактори, за допомогою яких можна оцінити доцільність впровадження інформаційної системи обрахунку заборгованості та бухгалтерського обліку це адекватність інтерфейсу. Адекватність призначеного для користувача інтерфейсу програми – це відповідність тим завданням, які користувачі повинні і хотіли б вирішувати з його допомогою.

Це відповідність має два аспекти [7]:

– по-перше, всі потрібні користувачам завдання повинні бути розв'язані;

– по-друге , ті дії, які користувачі виконують частіше, повинні вимагати менше зусиль.

Продуктивність роботи користувачів. Це кількість однотипних реальних завдань, які користувач може вирішити за допомогою програмного забезпечення за одиницю часу. Швидкість навчання нових користувачів. Це кількість завдань, виконання яких новий користувач самостійно навчається за одиницю часу. Крім того, важливим показником є відповідність навчання частоті виникнення завдань – чим частіше на практиці виникає необхідність вирішити певне завдання, тим швидше користувач повинен навчитися робити це.

Ефективність запобігання та подолання помилок користувачів . Цей показник

тим краще, чим рідше користувачі помиляються при роботі з даним інтерфейсом і чим менше часу і зусиль потрібно для подолання наслідків вже зроблених помилок.

Велике значення має також ризик, пов'язаний з виникненням помилки . Зручне ПЗ не повинно вимагати від користувача серйозних зусиль на вчинення дій, помилки в яких не дуже накладні; але якщо наслідки помилки катастрофічні, необхідно всіляко перешкоджати її здійсненню . Суб'єктивне задоволення користувачів. Цей фактор визначає, наскільки інтерфейс добре сприймався користувачами. Вважається, що суб'єктивне задоволення користувачів майже завжди підвищується в наступних випадках [7]. Якщо інтерфейс програми естетичний і елегантний, тобто побудований на небагатьох і близьких до гармонійного співвідношення між розмірами окремих елементів і відстанями між ними, на м'яких, непомітних кольорах і не ріжучих очей їх поєднаннях , невеликій кількості контрастів, злегка згладжених кутах, на акуратному вирівнюванні окремих елементів.

У роботі системи не повинно виникати довгих пауз, під час яких користувачі не знають, чим зайнятися. Навіть якщо системі потрібно багато часу для виконання якихось дій, які відображаються в цей час картинки і надається додаткова інформація можуть знизити суб'єктивну тривалість очікування.

Правило доступності. Система повинна бути настільки зрозумілою, щоб користувач, ніколи раніше не бачив її, але добре розбирається в предметній області, міг без будь-якого навчання почати її використовувати. Це правило служить деяким ідеалом, до якого треба прагнути, оскільки на практиці досягти такої міри зрозумілості майже ніколи не вдається.

Правило ефективності. Система не повинна перешкоджати ефективній роботі досвідчених користувачів, які працюють з нею довгий час.

Правило підтримки. Система повинна сприяти більш простому і швидкому вирішенню завдань користувача. Це означає, перш за все, що система повинна дійсно вирішувати завдання користувача.

Правило дотримання контексту. Система повинна бути узгоджена з контекстом, в якому їй належить працювати. Це правило вимагає від системи бути

працездатною не «взагалі», а саме в тому оточенні, в якому нею будуть користуватися. У контекст можуть входити специфіка і обсяги вхідних і вихідних даних, тип і цілі організацій, в яких система повинна працювати, рівень користувачів тощо.

Представлені вище правила визначають загальні вимоги, яким повинен задовольняти зручний інтерфейс. Крім них при розробці програмного забезпечення необхідно мати деякі підказки, що дозволяють зробити його більш зручним. Наступні принципи дозволяють знаходити рішення, що підвищують зручність для користувача інтерфейсу [13].

Принцип структуризації. Інтерфейс повинен бути доцільно структурований. Близькі за змістом, родинні його частини повинні бути пов'язані видимим чином, а незалежні – розділені; схожі елементи повинні виглядати схоже, а несхожі – відрізнятися. Принцип простоти. Найбільш поширені операції повинні виконуватися максимально просто. При цьому повинні бути видимі посилання на більш складні процедури.

Принцип видимості. Всі функції і дані, необхідні для вирішення певної задачі, повинні бути видні, коли користувач намагається її вирішити. Принцип зворотного зв'язку. Користувач повинен отримувати повідомлення про дії системи і про важливі події всередині неї. Повідомлення повинні бути інформативними, короткими, однозначними і написаними на мові, зрозумілій користувачеві.

Принцип толерантності. Інтерфейс повинен бути гнучким і терпимим до помилок користувача. Збиток від помилок повинен знижуватися за рахунок можливості скасування і затримки дій і за рахунок розумної інтерпретації будь-яких розумних дій користувача і введених їм даних.

Принцип повторного використання. Слід намагатися використовувати багаторазово внутрішні і зовнішні компоненти, забезпечуючи тим самим уніфікованість інтерфейсу і схожість між його елементами. Схема інтерфейсу, призначеного для користувачів комп’ютерної системи обліку первинних бухгалтерських документів наведена на рис. 3.2.

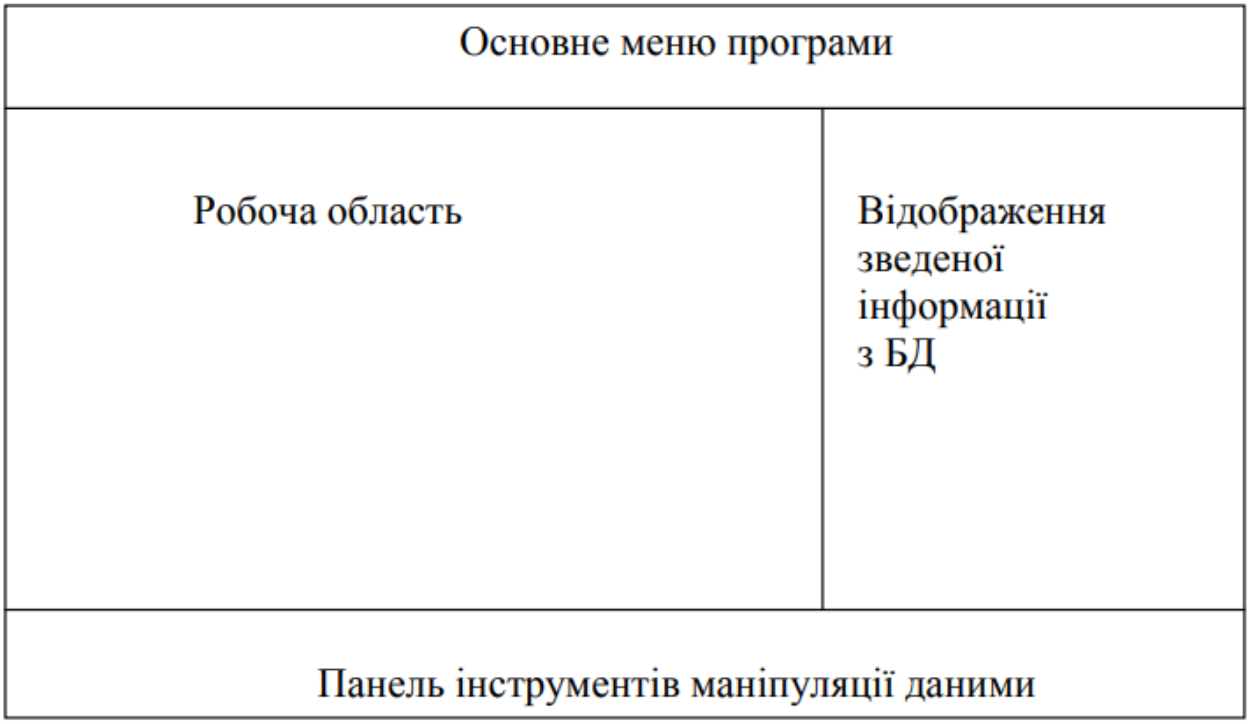

Рис. 3.2. Схема інтерфейсу користувачів компютерної системи

Важливим атрибутом будь-якого програмного продукту є зручний і інтуїтивно зрозумілий інтерфейс, приємний, але ненав'язливий зовнішній вигляд програми. Також важливими елементами є логотип (графічний знак, який ідентифікує компанію) і назва програмного продукту. Ці елементи не відіграють важливої ролі у функціонуванні програми, але є важливою частиною програми, саме яку запам'ятовує клієнт і змушує його повернутися до цього програмного забезпечення.

3.4. Опис розробки та роботи системи

Програма реалізована у вигляді пов'язаних модулів (форм), кожен відповідає за свій конкретний набір дій і описує кожну форму в програмних модулях.

Розпочнемо з налаштування проекту:

По-перше, створимо новий проект програми WPF на C# у Visual Studio, оскільки наш проект вже створений, я зображу лише процес створення нового див. рис.3.3.

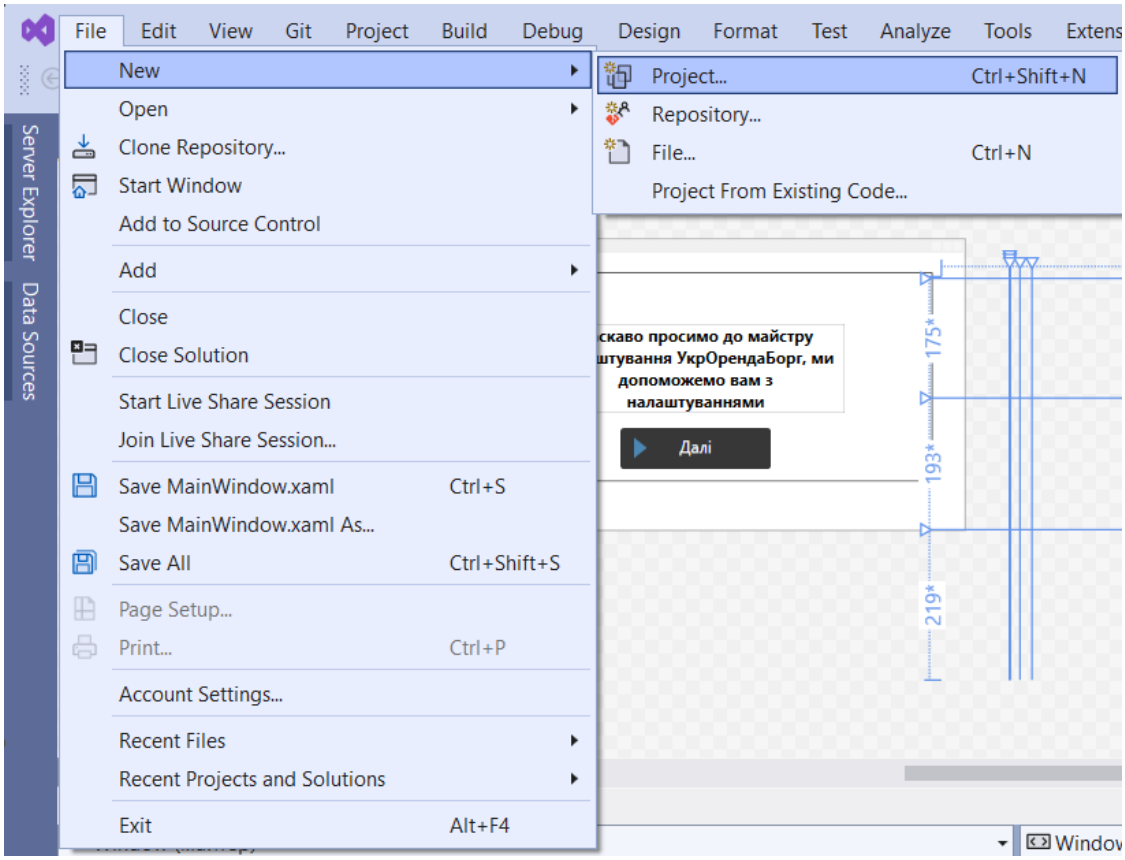

Рис. 3.3. Створення проекту у середі розробки Visual Studio

Додамо пакет NuGet для Entity Framework див. рис.3.4.-3.5. У браузері рішень оберемо вузол проекту, у головному меню виберемо Project > Manege NuGet Packages.

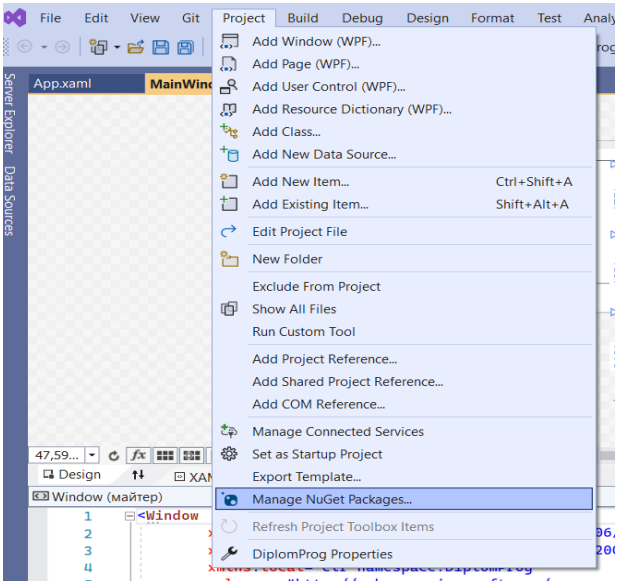

Рис. 3.4. Додавання пакету NuGet до проекту

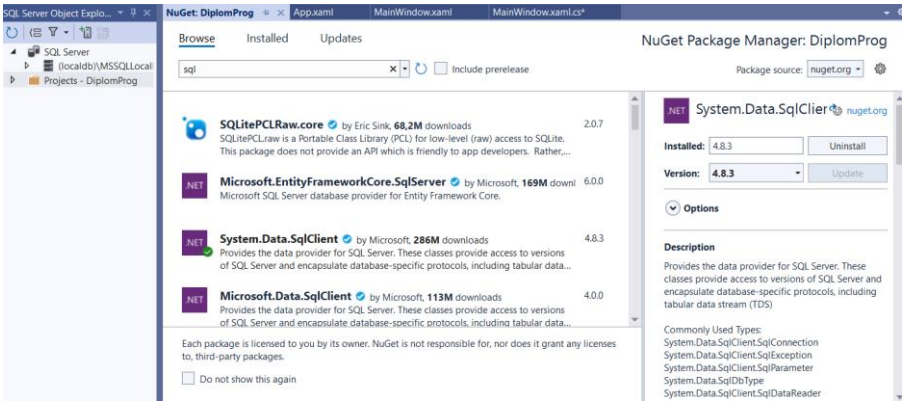

Рис. 3.5. Додавання пакету System.Data.SqlClient та Entity Framework до проекту

Метод підключення та закриття доступу до бази даних які знаходиться в public partial class MainWindow зображено на рис.3.6. – 3.7.

```
1 reference
public DataTable SelectDB(string selectSQL)
{
    DataTable dataTable = new DataTable("dataBase");

    SqlConnection sqlConnection = new SqlConnection("server=DESKTOP-GPCHLMU"+@"\"
                                  +"SQLEXPRESS;Trusted_Connection=Yes;DataBase=BuhOblik;");
    sqlConnection.Open();
    SqlCommand sqlCommand = sqlConnection.CreateCommand();
    sqlCommand.CommandText = selectSQL;
    SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(sqlCommand);
    sqlDataAdapter.Fill(dataTable);
    return dataTable;
}
```

Рис. 3.6. Метод підключення до бази даних

```
private void MainWindow_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{
    SqlConnection sqlConnection = new SqlConnection("server=DESKTOP-GPCHLMU" + @"\"
                                  + "SQLEXPRESS;Trusted_Connection=Yes;DataBase=BuhOblik;");
    sqlConnection.Close();
}
```

Рис. 3.7. Метод закриття доступу до бази даних

Оглянемо основну частину, розпочнемо з вітального вікна див. рис. 3.8. тут ми вітаємось з користувачем, натиснувши далі, ми перейдемо до налаштувань програми.

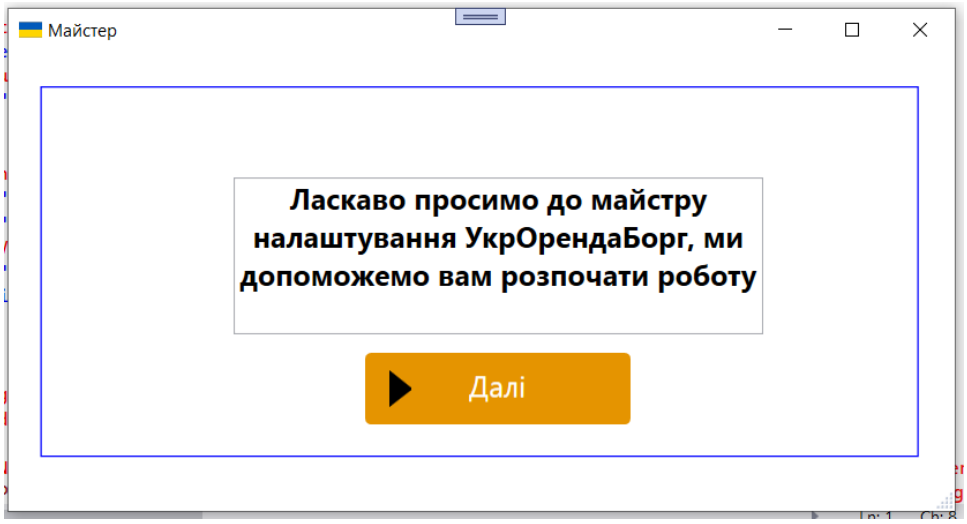

Рис. 3.8. Вітальне вікно програми

У вікні з налаштуваннями (див. рис.3.9.) ми маємо ввести дані своєї компанії аби вони збереглись і в майбутньому підтягувались у платіжні рахунки для орендарів.

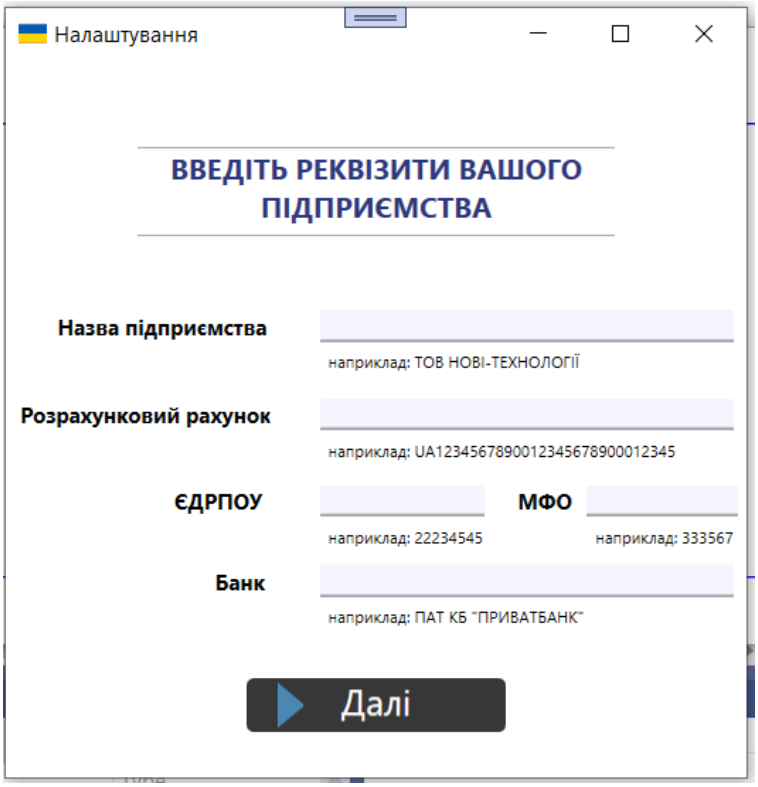

Рис.3.9. Вікно налаштувань

Після налаштування ми переходимо до основного вікна програми, де можемо розрахувати суму заборгованості, формувати рахунку, додавати та переглядати орендарів, вкладка боржники, показує нам, які з орендарів не вчасно вносили платежі і в них сформувався борг, вкладка перевірити оплати дозволяє переглянути які оплати від тих компаній що є в переліку Орендарів нам прийшли.

Тут ми можемо також і зберегти ці дані аби в наступний раз при обрахуванні боргу не вводити наново дані про заборгованість.

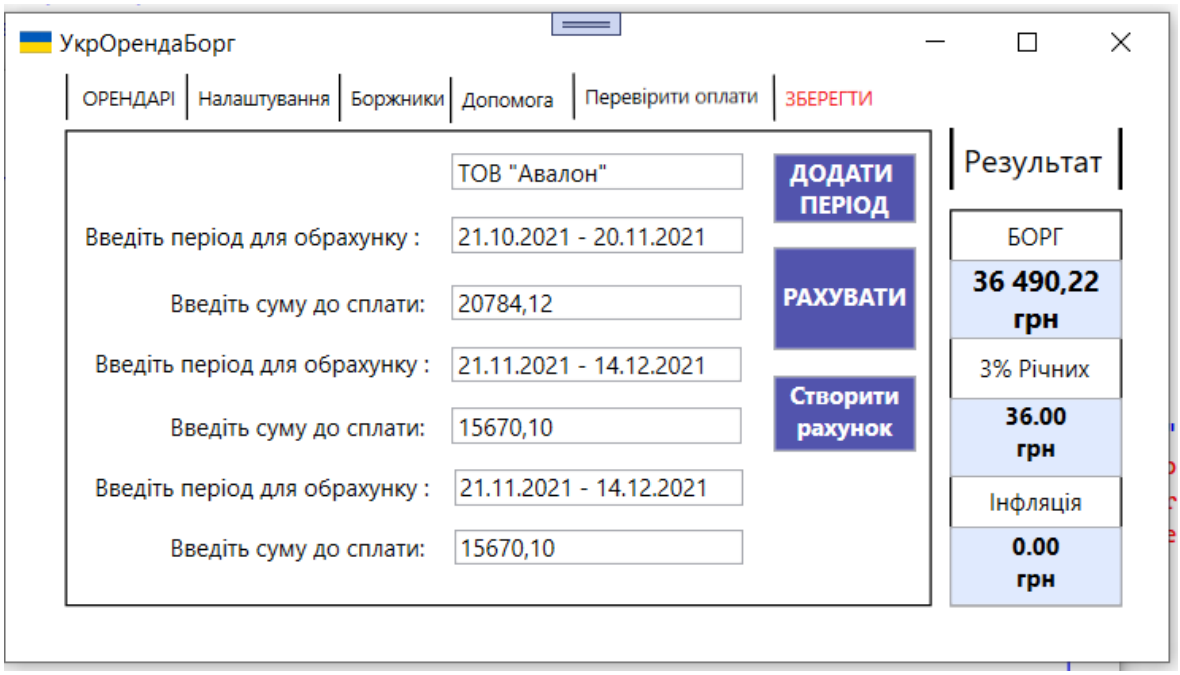

Рис. 3.10. Робоча область програми

Процес додавання нового орендаря в програму зображено на рис. 3.11.

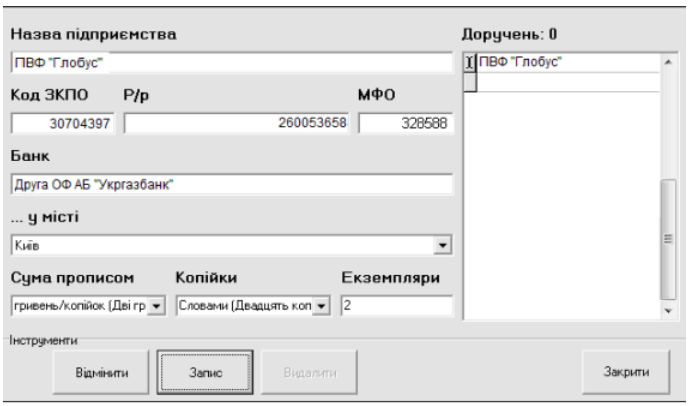

Рис. 3.11. Вікно додавання орендаря

Будь-яке вікно програми запускається в режимі модулю режимі, простіше кажучи, наш юзер не матиме можливості переходити до попереднього вікна, якщо він не закриє старе.

Завдяки подібному інтерфейсу наша система дозволяє не втрачати дані. Однією з фундаментальних можливостей інформаційної системи є роздруківка рахунків, в подальшому буде ще й конструктор шаблонів. Конструктор зможе виконувати функції які необхідні для зміни шаблону документа при внесенні нових поправок в законодавство, або в наслідок, затвердження нових форм фінансових документів.

Модуль який реалізований в даний час не дозволяє вносити такі зміни автоматично не здійснюючи зміни в самому програмному коді системи. Але дозволяэ сформувати та роздрукувати рахунок. Звернутися до Формування рахунку за допомогою Головного меню системи. Треба обрати розділ Головного меню Настройки, обрати створити рахунок (рис. 3.12).

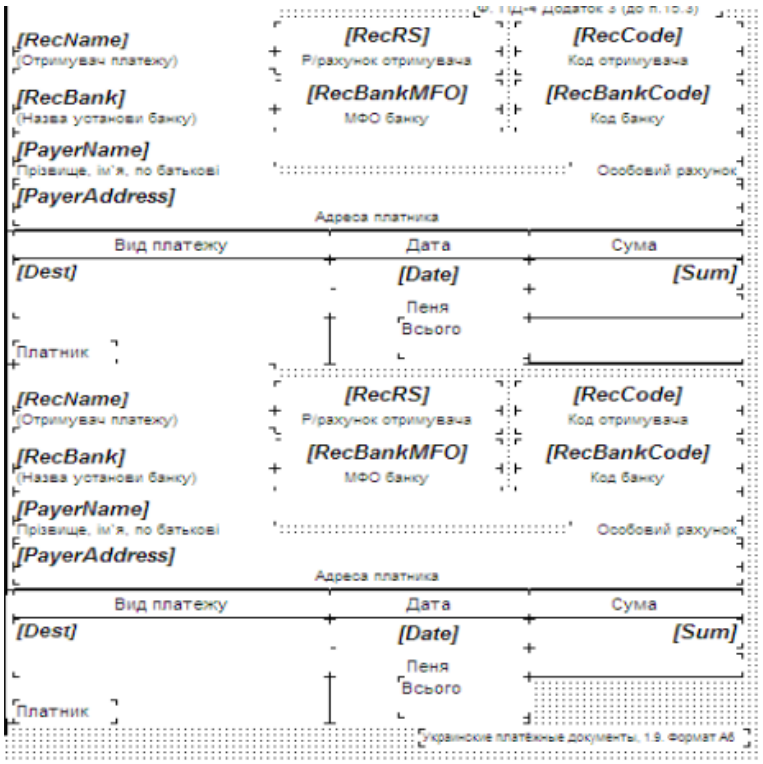

Рис. 3.12. Шаблон рахунку на сплату

Таким чином, розроблена комп’ютерна інформаційна система щодо обрахунку заборгованості та бухгалтерського обліку є закінченою функціональною одиницею, яка оперативно настроюється при необхідності внесення поправок.

ВИСНОВКИ

Застосування інформаційних систем бухгалтерського обліку на підприємстві в умовах перехідної економіки України є однією з найбільш важливих задач. Ситуація така, що сам по собі бухгалтерський облік на підприємстві може розглядатися як внутрішня справа підприємства, а основою для оцінки фінансово-господарської діяльності підприємства з боку держави служить звітність (бухгалтерський баланс і численні інші звітні форми), що повинна щокварталу надаватися в податкову інспекцію по місцеві реєстрації підприємства. Крім того, існують планові і позапланові податкові перевірки, при проведенні яких можуть знадобитися всі бухгалтерські документи, включаючи первинні. Усе це обумовлює широке застосування ІСБО в сучасній Україні.

В даний час існує широкий вибір різних інформаційних систем бухгалтерського обліку. Не слід поділяти їх на погані і гарні, сильні слабкі. Усі вони гарні і їхні можливості знаходять практичне застосування на підприємствах різного розміру, профілю і роду діяльності. При автоматизації варто вибрати необхідну ІСБО, виходячи з задач і наявних ресурсів.

При використанні інформаційних систем бухобліка важливо не просто перевести всю паперову роботу на комп’ютер. Важливо, щоб це збільшило ефективність роботи бухгалтерії і поліпшило контроль над фінансово-господарською діяльністю підприємства, що у свою чергу збільшить ефективність управління підприємством, і, як наслідок, ефективність його роботи.

Враховуючи проведену роботу та використання різних сучасних технологій можна зробити висновки що хоч і веб розробка домінує на ринку України та і на світовому ринку в цілому, десктоп додатки ще доволі довго не втратять своєї актуальності, по перше це більш надійна система в якій кожен комп’ютер неначе окреме королівство якому не потрібен ні інтернет ні сторонні застосунки, також це більш швидка і стабільна робота, звичайно можна сказати що пк користувача “не потягне” дане програмне забезпечення, та я вважаю що десктор розробка ще доволі довго буде актуальної, як ми бачимо на на ній дуже вдало можна розробляти

інформаційні системи, але не цим єдиним WPF як і .Net Framework, це сотні десктоп програм, CRM системи, системи складського обліку і багато іншого, а отже дослідивши увесь цикл розробки ПЗ можу стверджувати що дана технологія буде актуальна мінімум 5 років, а отже розробка даної інформаційної системи несе не тільки практичне а і теоретичне застосування як спосіб дослідження технології розробки ПЗ.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ

1. Бухгалтерський облік [Електронний ресурс]. Режим доступу: https://v8.1c.ru/buhv8/ (дата звернення 16.11.2021) – Назва з екрана.
2. Цивільний кодекс України [Електронний ресурс]. Режим доступу: https://zakon.rada.gov.ua/laws/show/435-15#Text (дата звернення 16.11.2021) – Назва з екрана.
3. Господарський кодекс України [Електронний ресурс]. Режим доступу: https://zakon.rada.gov.ua/laws/show/436-15#Text (дата звернення 16.11.2021) – Назва з екрана.
4. Закон України про несвоєчасне виконання зобов'язань [Електронний ресурс]. Режим доступу: https://zakon.rada.gov.ua/laws/show/543/96-%D0%B2%D1%80#Text (дата звернення 16.11.2021) – Назва з екрана.
5. Сплата 3% річних [Електронний ресурс]. Режим доступу: https://wiki.legalaid.gov.ua/index.php/ (дата звернення 16.11.2021) – Назва з екрана.
6. Брауде Эрик Дж. Технология разработки программного обеспечения. – М.: Computer Science, 2004. –655 с.
7. О.М.Ананьєв, В.М. Білик, Я. А. Гончарук. Інформаційні системи і технології в комерційній діяльності. Навчальний посібник . – Львів: Новий Світ-2000, 2006. – 584 с.
8. Карпова Т.С. Базы данных: модели, разработка, реализация. – СПб.: Питер, 2001. – 304 с.
9. Метью Мак-Дональд WPF: Windows Presentation Foundation в. NET 4.0 із прикладами на C # 2010 для професіоналів = Pro WPF in C # 2010: Windows Presentation Foundation with. NET 4.0. - M .: "Вильямс", 2011. - 1024 с.
10. Андерсон, Кріс Основи Windows Presentation Foundation. - СПб. : БХВ-Петербург, 2008. - 432 с.
11. Daniel M. Solis Illustrated WPF. - United States of America: Apress, 2009. - 508 с.

12. Огляд WPF та .Net [Електронний ресурс]. Режим доступу: https://training.epam.ua/#!/News/301?lang=ua (дата звернення 16.11.2021) – Назва з екрана.
13. WPF [Електронний ресурс]. Режим доступу: http://isearch.kiev.ua/uk/news/programs/o-appl/1593-create-your-first-windows-forms-application-in-c-for-example-the-game-qtic-tac-toeq (дата звернення 16.11.2021) – Назва з екрана.
14. ДСТУ [Електронний ресурс]. Режим доступу: https://zakon.help/article/nacionalnii-standart-dstu-41632020-derzhavna?menu=82 (дата звернення 16.11.2021) – Назва з екрана.

ДОДАТКИ

Додаток А

Програмний код

```
namespace DiplomProg
{
   /// <summary>
   /// Interaction logic for MainWindow.xaml
   /// </summary>
   public partial class MainWindow : Window
   {
      public MainWindow()
      {
         InitializeComponent();

         DataTable dt_user = SelectDB("SELECT * FROM [dbo].[Tenant]");

         for (int i = 0; i < dt_user.Rows.Count; i++)
         {
            MessageBox.Show(dt_user.Rows[i][0] + "|" + dt_user.Rows[i][1] + "|" +
dt_user.Rows[i][2]); // виводимо дані

         }



         this.Closing += MainWindow_Closing;


      }
      private void MainWindow_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
      {
         SqlConnection sqlConnection = new SqlConnection("server=DESKTOP-
GPCHLMU" + @"\"
                              +
"SQLEXPRESS;Trusted_Connection=Yes;DataBase=BuhOblik;");
         sqlConnection.Close();
      }

      private void Button_Click(object sender, RoutedEventArgs e)
      {
         MyDependencyClass.SetCurrentDuration(MyWindow, "Duration: 101m");
         SetingWindow setingWindow = new SetingWindow();
```

```
            setingWindow.Show();
        }

        public DataTable SelectDB(string selectSQL)
        {
            DataTable dataTable = new DataTable("dataBase");

            SqlConnection sqlConnection = new SqlConnection("server=DESKTOP-
GPCHLMU"+@"\"

+"SQLEXPRESS;Trusted_Connection=Yes;DataBase=BuhOblik;");
            sqlConnection.Open();
            SqlCommand sqlCommand = sqlConnection.CreateCommand();
            sqlCommand.CommandText = selectSQL;
            SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(sqlCommand);
            sqlDataAdapter.Fill(dataTable);
            return dataTable;
        }


    }

    public class MyDependencyClass : DependencyObject
    {
        public static readonly DependencyProperty CurrentDurationProperty;

        public static void SetCurrentDuration(DependencyObject DepObject, string value)
        {
            DepObject.SetValue(CurrentDurationProperty, value);
        }

        public static string GetCurrentDuration(DependencyObject DepObject)
        {
            return (string)DepObject.GetValue(CurrentDurationProperty);
        }

        static MyDependencyClass()
        {
            PropertyMetadata MyPropertyMetadata = new PropertyMetadata("Duration: 0m");

            CurrentDurationProperty =
DependencyProperty.RegisterAttached("CurrentDuration",
                                                typeof(string),
                                                typeof(MyDependencyClass),
```

```
                                        MyPropertyMetadata);
        }
    }

}

<Window
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="clr-namespace:DiplomProg"
        xmlns:av="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
mc:Ignorable="av" x:Class="DiplomProg.MainWindow"
        x:Name="майстер"
        Title="Майстер" Height="356" Width="670"
        WindowStartupLocation="CenterScreen"
        local:MyDependencyClass.CurrentDuration="" Icon="/Прапор.jpg"
          ResizeMode ="CanResizeWithGrip">

    <Window.Resources>
        <Style TargetType="{x:Type Button}">
            <Setter Property="Background" Value="#373737" />
            <Setter Property="Foreground" Value="White" />
            <Setter Property="FontSize" Value="15" />
            <Setter Property="FontFamily" Value="./#Segoe UI" />
            <Setter Property="SnapsToDevicePixels" Value="True" />

            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="{x:Type Button}">
                        <Border CornerRadius="4" Background="{TemplateBinding
Background}">
                            <Grid>
                                <Path x:Name="PathIcon" Width="15" Height="25" Stretch="Fill"
Fill="#4C87B3" HorizontalAlignment="Left" Margin="17,0,0,0" Data="F1 M
30.0833,22.1667L 50.6665,37.6043L 50.6665,38.7918L 30.0833,53.8333L
30.0833,22.1667 Z "/>
                                <ContentPresenter x:Name="MyContentPresenter"
Content="{TemplateBinding Content}" HorizontalAlignment="Center"
VerticalAlignment="Center" Margin="0,0,0,0" />
                            </Grid>
                        </Border>

                        <ControlTemplate.Triggers>
```

```
                    <Trigger Property="IsMouseOver" Value="True">
                        <Setter Property="Background" Value="#E59400" />
                        <Setter Property="Foreground" Value="White" />
                        <Setter TargetName="PathIcon" Property="Fill" Value="Black" />
                    </Trigger>

                    <Trigger Property="IsPressed" Value="True">
                        <Setter Property="Background" Value="OrangeRed" />
                        <Setter Property="Foreground" Value="White" />
                    </Trigger>
                </ControlTemplate.Triggers>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>

<StackPanel x:Key="MyStackPanel">
    <TextBlock x:Name="MainContent" Text="Далі" FontSize="20"
RenderTransformOrigin="0.5,0.5" >
        <TextBlock.RenderTransform>
            <TransformGroup>
                <ScaleTransform/>
                <SkewTransform/>
                <RotateTransform Angle="0.423"/>
                <TranslateTransform/>
            </TransformGroup>
        </TextBlock.RenderTransform>
    </TextBlock>
</StackPanel>
</Window.Resources>

<Grid Margin="646,34,-807,-187">
    <Grid.RowDefinitions>
        <RowDefinition Height="0*"/>
        <RowDefinition Height="175*"/>
        <RowDefinition Height="193*"/>
        <RowDefinition Height="219*"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="94*"/>
        <ColumnDefinition Width="0*"/>
        <ColumnDefinition Width="0*"/>
        <ColumnDefinition Width="0*"/>
        <ColumnDefinition Width="15*"/>
```

```
            <ColumnDefinition Width="17*"/>
            <ColumnDefinition Width="835*"/>
        </Grid.ColumnDefinitions>
        <Button x:Name="MyWindow" Height="50" Content="{StaticResource
MyStackPanel}" VerticalAlignment="Top" Margin="-399,34,295,0" Grid.Row="2"
Click="Button_Click" />
        <TextBox HorizontalContentAlignment="Center" HorizontalAlignment="Left"
Height="109" Margin="-490,53,0,0" Grid.RowSpan="3" FontSize ="20" FontWeight =
"Bold"  TextWrapping="Wrap" Text="Ласкаво просимо до майстру налаштування
УкрОрендаБорг, ми допоможемо вам розпочати роботу" VerticalAlignment="Top"
Width="367"/>
        <Border  BorderBrush="Blue" BorderThickness="1" HorizontalAlignment="Left"
Height="257" Margin="-624,-10,0,0" Grid.RowSpan="3" VerticalAlignment="Top"
Width="609"/>
    </Grid>
</Window>

public partial class WorkMain : Window
    {
        public WorkMain()
        {
            InitializeComponent();
        }

        private void TextBox_TextChanged(object sender, TextChangedEventArgs e)
        {

        }

        private void TextBox_TextChanged_1(object sender, TextChangedEventArgs e)
        {

        }
    }

<Window x:Class="DiplomProg.Borzhnikixaml"
        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:DiplomProg"
        mc:Ignorable="d"
        Title="УкрОрендаБорг" Height="394" Width="205" Background="#FF828EF6"
ShowInTaskbar="False" Icon="/Прапор.jpg">
```

```
    <Grid>

    </Grid>
</Window>

<Window x:Class="DiplomProg.WorkMain"
        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:DiplomProg"
        mc:Ignorable="d"
        Title="УкрОрендаБорг" Height="342" Width="601" Icon="/Прапор.jpg">
    <Grid Margin="4,0,10,11">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="177*"/>
            <ColumnDefinition Width="8*"/>
            <ColumnDefinition Width="14*"/>
            <ColumnDefinition Width="388*"/>

        </Grid.ColumnDefinitions>
        <Label Content="Налаштування" HorizontalAlignment="Left" Height="24"
VerticalAlignment="Top" Width="95" FontWeight="Normal" FontSize="10"
Grid.ColumnSpan="2" Margin="88,1,0,0"/>
        <Label Content="Боржники" HorizontalAlignment="Left" Margin="164,1,0,0"
VerticalAlignment="Top" Width="80" FontSize="10" BorderThickness="1,0,0,0"
BorderBrush="Black" Grid.ColumnSpan="4" Height="24"/>
        <Label Content="Допомога" HorizontalAlignment="Left" Margin="26,2,0,0"
VerticalAlignment="Top" Width="74" FontSize="10" BorderThickness="1,0,0,0"
BorderBrush="Black" Grid.Column="3"/>
        <Label Content="Перевірити оплати" HorizontalAlignment="Left"
Margin="88,0,0,0" VerticalAlignment="Top" Width="98" FontSize="10"
BorderThickness="1,0,0,0" BorderBrush="Black" RenderTransformOrigin="0.52,0.535"
Grid.Column="3">
            <Label.RenderTransform>
                <TransformGroup>
                    <ScaleTransform/>
                    <SkewTransform/>
                    <RotateTransform Angle="-0.408"/>
                    <TranslateTransform X="0.004" Y="-0.014"/>
                </TransformGroup>
            </Label.RenderTransform>
        </Label>
```

```
        <Border BorderBrush="Black" BorderThickness="1" HorizontalAlignment="Left"
Height="240" Margin="26,29,0,0" VerticalAlignment="Top" Width="437"
Grid.ColumnSpan="4">
            <Label Content=""/>
        </Border>
        <TextBox HorizontalAlignment="Left" Height="18" Margin="27,41,0,0"
TextWrapping="Wrap" Text="ТОВ &amp;quot;Авалон&amp;quot;" VerticalAlignment="Top"
Width="147" TextChanged="TextBox_TextChanged" Grid.Column="3"/>
        <Label Content="Введіть період для обрахунку :" HorizontalAlignment="Left"
Margin="31,69,0,0" VerticalAlignment="Top" Grid.ColumnSpan="4"/>
        <Label Content="Результат" HorizontalAlignment="Left" Margin="278,28,0,0"
VerticalAlignment="Top" Height="31" Width="87" BorderThickness="2,0,2,0"
BorderBrush="Black" FontSize="16" Grid.Column="3"/>
        <TextBox HorizontalAlignment="Left" Height="18" Margin="27,73,0,0"
TextWrapping="Wrap" Text="21.10.2021 - 20.11.2021" VerticalAlignment="Top"
Width="147" TextChanged="TextBox_TextChanged" Grid.Column="3"/>
        <Label Content="Введіть суму до сплати:  " HorizontalAlignment="Left"
Margin="74,103,0,0" VerticalAlignment="Top" Grid.ColumnSpan="4"/>
        <TextBox HorizontalAlignment="Left" Height="18" Margin="27,107,0,0"
TextWrapping="Wrap" Text="20784,12" VerticalAlignment="Top" Width="146"
TextChanged="TextBox_TextChanged" Grid.Column="3"/>
        <TextBox HorizontalAlignment="Left" Height="35" Margin="189,41,0,0"
TextWrapping="Wrap" Text="ДОДАТИ &amp;#xD;&amp;#xA;ПЕРІОД"
VerticalAlignment="Top" Width="72" TextChanged="TextBox_TextChanged"
TextAlignment="Center" Background="#FF5254AD" Foreground="White"
FontWeight="Bold" Grid.Column="3"/>
        <Label Content="Введіть період для обрахунку :" HorizontalAlignment="Left"
Margin="36,133,0,0" VerticalAlignment="Top" Grid.ColumnSpan="4"/>
        <Label Content="Введіть суму до сплати:  " HorizontalAlignment="Left"
Margin="74,165,0,0" VerticalAlignment="Top" Grid.ColumnSpan="4"/>
        <TextBox HorizontalAlignment="Left" Height="18" Margin="27,138,0,0"
TextWrapping="Wrap" Text="21.11.2021 - 14.12.2021" VerticalAlignment="Top"
Width="146" TextChanged="TextBox_TextChanged" Grid.Column="3"/>
        <TextBox HorizontalAlignment="Left" Height="18" Margin="27,169,0,0"
TextWrapping="Wrap" Text="15670,10" VerticalAlignment="Top" Width="146"
TextChanged="TextBox_TextChanged" Grid.Column="3"/>
        <Label Content="Введіть період для обрахунку :" HorizontalAlignment="Left"
Margin="36,196,0,0" VerticalAlignment="Top" Grid.ColumnSpan="4"/>
        <Label Content="Введіть суму до сплати:  " HorizontalAlignment="Left"
Margin="74,226,0,0" VerticalAlignment="Top" Grid.ColumnSpan="4"/>
        <TextBox HorizontalAlignment="Left" Height="18" Margin="28,230,0,0"
TextWrapping="Wrap" Text="15670,10" VerticalAlignment="Top" Width="146"
TextChanged="TextBox_TextChanged" Grid.Column="3"/>
```

```
        <TextBox HorizontalAlignment="Left" Height="18" Margin="28,200,0,0"
TextWrapping="Wrap" Text="21.11.2021 - 14.12.2021" VerticalAlignment="Top"
Width="146" TextChanged="TextBox_TextChanged" Grid.Column="3"/>
        <Border BorderBrush="Black" BorderThickness="1" HorizontalAlignment="Left"
Height="200" Margin="278,69,0,0" VerticalAlignment="Top" Width="87"
Grid.Column="3"/>
        <TextBox HorizontalAlignment="Left" Height="40" Margin="278,94,0,0"
TextWrapping="Wrap" Text="36 490,22&#xD;&#xA;грн" VerticalAlignment="Top"
Width="87" RenderTransformOrigin="0.502,0.476" TextAlignment="Center"
BorderThickness="1,1,1,1" Background="#FFE0EAFE" FontWeight="Bold"
FontSize="14" Grid.Column="3"/>
        <Label Content="БОРГ" HorizontalAlignment="Left" Margin="302,69,0,0"
VerticalAlignment="Top" Height="25" Grid.Column="3"/>
        <Label Content="3% Річних" HorizontalAlignment="Left" Margin="288,136,0,0"
VerticalAlignment="Top" Grid.Column="3"/>
        <TextBox HorizontalAlignment="Left" Height="40" Margin="278,164,0,0"
TextWrapping="Wrap" Text="36.00&#xA;грн" VerticalAlignment="Top" Width="87"
RenderTransformOrigin="0.502,0.476" TextAlignment="Center"
BorderThickness="1,1,1,1" Background="#FFE0EAFE" FontWeight="Bold"
Grid.Column="3"/>
        <Label Content="Інфляція" HorizontalAlignment="Left" Margin="293,203,0,0"
VerticalAlignment="Top" Grid.Column="3"/>
        <TextBox HorizontalAlignment="Left" Height="40" Margin="278,229,0,0"
TextWrapping="Wrap" Text="0.00&#xA;грн" VerticalAlignment="Top" Width="87"
RenderTransformOrigin="0.502,0.476" TextAlignment="Center"
BorderThickness="1,1,1,1" Background="#FFE0EAFE" FontWeight="Bold"
TextChanged="TextBox_TextChanged_1" Grid.Column="3"/>
        <TextBox HorizontalAlignment="Left" Height="52" Margin="189,88,0,0"
TextWrapping="Wrap" Text="&#xD;&#xA;РАХУВАТИ" VerticalAlignment="Top"
Width="72" TextChanged="TextBox_TextChanged" TextAlignment="Center"
Background="#FF5254AD" Foreground="White" FontWeight="Bold"
Grid.Column="3"/>
        <TextBox HorizontalAlignment="Left" Height="38" Margin="189,153,0,0"
TextWrapping="Wrap" Text="Створити&#xD;&#xA;рахунок"
VerticalAlignment="Top" Width="72" TextChanged="TextBox_TextChanged"
TextAlignment="Center" Background="#FF5254AD" Foreground="White"
FontWeight="Bold" Grid.Column="3"/>
        <Label Content="ЗБЕРЕГТИ" HorizontalAlignment="Left" Margin="189,1,0,0"
VerticalAlignment="Top" Width="57" FontSize="10" BorderThickness="1,0,0,0"
BorderBrush="Black" RenderTransformOrigin="0.5,0.5" Grid.Column="3"
Foreground="#FFF31212">
            <Label.RenderTransform>
                <TransformGroup>
                    <ScaleTransform/>
```

```
                <SkewTransform/>
                <RotateTransform Angle="-0.408"/>
                <TranslateTransform/>
              </TransformGroup>
            </Label.RenderTransform>
        </Label>
        <Label Content=" ОРЕНДАРІ" HorizontalAlignment="Left" Margin="26,1,0,0"
VerticalAlignment="Top" Width="62" FontSize="10" BorderThickness="1,0,1,0"
BorderBrush="Black"/>

    </Grid>
</Window>

<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="auto"/>
        <RowDefinition Height="auto"/>
        <RowDefinition Height="*"/>
    </Grid.RowDefinitions>
    <Grid x:Name="existingCustomerGrid" Grid.Row="1" HorizontalAlignment="Left"
Margin="5" Visibility="Visible" VerticalAlignment="Top" Background="AntiqueWhite"
DataContext="{StaticResource customerViewSource}">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto" MinWidth="233"/>
            <ColumnDefinition Width="Auto" MinWidth="397"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
        <Label Content="Customer ID:" Grid.Row="0" Style="{StaticResource Label}"/>
        <TextBox x:Name="customerIDTextBox" Grid.Row="0" Style="{StaticResource
CustTextBox}"
                 Text="{Binding CustomerID, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Company Name:" Grid.Row="1" Style="{StaticResource
Label}"/>
        <TextBox x:Name="companyNameTextBox" Grid.Row="1"
Style="{StaticResource CustTextBox}"
```

```
            Text="{Binding CompanyName, Mode=TwoWay,
NotifyOnValidationError=true, ValidatesOnExceptions=true}"/>
        <Label Content="Contact Name:" Grid.Row="2" Style="{StaticResource Label}"/>
        <TextBox x:Name="contactNameTextBox" Grid.Row="2" Style="{StaticResource
CustTextBox}"
            Text="{Binding ContactName, Mode=TwoWay,
NotifyOnValidationError=true, ValidatesOnExceptions=true}"/>
        <Label Content="Contact title:" Grid.Row="3" Style="{StaticResource Label}"/>
        <TextBox x:Name="contactTitleTextBox" Grid.Row="3" Style="{StaticResource
CustTextBox}"
            Text="{Binding ContactTitle, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Address:" Grid.Row="4" Style="{StaticResource Label}"/>
        <TextBox x:Name="addressTextBox" Grid.Row="4" Style="{StaticResource
CustTextBox}"
            Text="{Binding Address, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="City:" Grid.Column="1" Grid.Row="0" Style="{StaticResource
Label}"/>
        <TextBox x:Name="cityTextBox" Grid.Column="1" Grid.Row="0"
Style="{StaticResource CustTextBox}"
            Text="{Binding City, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Country:" Grid.Column="1" Grid.Row="1"
Style="{StaticResource Label}"/>
        <TextBox x:Name="countryTextBox" Grid.Column="1" Grid.Row="1"
Style="{StaticResource CustTextBox}"
            Text="{Binding Country, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Fax:" Grid.Column="1" Grid.Row="2" Style="{StaticResource
Label}"/>
        <TextBox x:Name="faxTextBox" Grid.Column="1" Grid.Row="2"
Style="{StaticResource CustTextBox}"
            Text="{Binding Fax, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Phone:" Grid.Column="1" Grid.Row="3" Style="{StaticResource
Label}"/>
        <TextBox x:Name="phoneTextBox" Grid.Column="1" Grid.Row="3"
Style="{StaticResource CustTextBox}"
            Text="{Binding Phone, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Postal Code:" Grid.Column="1" Grid.Row="4"
VerticalAlignment="Center" Style="{StaticResource Label}"/>
```

```
        <TextBox x:Name="postalCodeTextBox" Grid.Column="1" Grid.Row="4"
Style="{StaticResource CustTextBox}"
                 Text="{Binding PostalCode, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Region:" Grid.Column="1" Grid.Row="5" Style="{StaticResource
Label}"/>
        <TextBox x:Name="regionTextBox" Grid.Column="1" Grid.Row="5"
Style="{StaticResource CustTextBox}"
                 Text="{Binding Region, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
    </Grid>
    <Grid x:Name="newCustomerGrid" Grid.Row="1" HorizontalAlignment="Left"
VerticalAlignment="Top" Margin="5" DataContext="{Binding
RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type Window}},
Path=newCustomer, UpdateSourceTrigger=Explicit}" Visibility="Collapsed"
Background="CornflowerBlue">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto" MinWidth="233"/>
            <ColumnDefinition Width="Auto" MinWidth="397"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
        <Label Content="Customer ID:" Grid.Row="0" Style="{StaticResource Label}"/>
        <TextBox x:Name="add_customerIDTextBox" Grid.Row="0"
Style="{StaticResource CustTextBox}"
                 Text="{Binding CustomerID, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Company Name:" Grid.Row="1" Style="{StaticResource
Label}"/>
        <TextBox x:Name="add_companyNameTextBox" Grid.Row="1"
Style="{StaticResource CustTextBox}"
                 Text="{Binding CompanyName, Mode=TwoWay,
NotifyOnValidationError=true, ValidatesOnExceptions=true }"/>
        <Label Content="Contact Name:" Grid.Row="2" Style="{StaticResource Label}"/>
        <TextBox x:Name="add_contactNameTextBox" Grid.Row="2"
Style="{StaticResource CustTextBox}"
                 Text="{Binding ContactName, Mode=TwoWay,
NotifyOnValidationError=true, ValidatesOnExceptions=true}"/>
```

```
        <Label Content="Contact title:" Grid.Row="3" Style="{StaticResource Label}"/>
        <TextBox x:Name="add_contactTitleTextBox" Grid.Row="3"
Style="{StaticResource CustTextBox}"
                 Text="{Binding ContactTitle, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Address:" Grid.Row="4" Style="{StaticResource Label}"/>
        <TextBox x:Name="add_addressTextBox" Grid.Row="4" Style="{StaticResource
CustTextBox}"
                 Text="{Binding Address, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="City:" Grid.Column="1" Grid.Row="0" Style="{StaticResource
Label}"/>
        <TextBox x:Name="add_cityTextBox" Grid.Column="1" Grid.Row="0"
Style="{StaticResource CustTextBox}"
                 Text="{Binding City, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Country:" Grid.Column="1" Grid.Row="1"
Style="{StaticResource Label}"/>
        <TextBox x:Name="add_countryTextBox" Grid.Column="1" Grid.Row="1"
Style="{StaticResource CustTextBox}"
                 Text="{Binding Country, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Fax:" Grid.Column="1" Grid.Row="2" Style="{StaticResource
Label}"/>
        <TextBox x:Name="add_faxTextBox" Grid.Column="1" Grid.Row="2"
Style="{StaticResource CustTextBox}"
                 Text="{Binding Fax, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Phone:" Grid.Column="1" Grid.Row="3" Style="{StaticResource
Label}"/>
        <TextBox x:Name="add_phoneTextBox" Grid.Column="1" Grid.Row="3"
Style="{StaticResource CustTextBox}"
                 Text="{Binding Phone, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Postal Code:" Grid.Column="1" Grid.Row="4"
VerticalAlignment="Center" Style="{StaticResource Label}"/>
        <TextBox x:Name="add_postalCodeTextBox" Grid.Column="1" Grid.Row="4"
Style="{StaticResource CustTextBox}"
                 Text="{Binding PostalCode, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Region:" Grid.Column="1" Grid.Row="5" Style="{StaticResource
Label}"/>
        <TextBox x:Name="add_regionTextBox" Grid.Column="1" Grid.Row="5"
Style="{StaticResource CustTextBox}"
```

```
            Text="{Binding Region, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
    </Grid>
    <Grid x:Name="newOrderGrid" Grid.Row="1" HorizontalAlignment="Left"
VerticalAlignment="Top" Margin="5" DataContext="{Binding Path=newOrder,
Mode=TwoWay}" Visibility="Collapsed" Background="LightGreen">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto" MinWidth="233"/>
            <ColumnDefinition Width="Auto" MinWidth="397"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
        <Label Content="New Order Form" FontWeight="Bold"/>
        <Label Content="Employee ID:"  Grid.Row="1" Style="{StaticResource Label}"/>
        <TextBox x:Name="add_employeeIDTextBox" Grid.Row="1"
Style="{StaticResource CustTextBox}"
             Text="{Binding EmployeeID, Mode=TwoWay,
NotifyOnValidationError=true, ValidatesOnExceptions=true}"/>
        <Label Content="Order Date:"  Grid.Row="2" Style="{StaticResource Label}"/>
        <DatePicker x:Name="add_orderDatePicker" Grid.Row="2"
HorizontalAlignment="Right" Width="120"
            SelectedDate="{Binding OrderDate, Mode=TwoWay,
NotifyOnValidationError=true, ValidatesOnExceptions=true,
UpdateSourceTrigger=PropertyChanged}"/>
        <Label Content="Required Date:" Grid.Row="3" Style="{StaticResource Label}"/>
        <DatePicker x:Name="add_requiredDatePicker" Grid.Row="3"
HorizontalAlignment="Right" Width="120"
             SelectedDate="{Binding RequiredDate, Mode=TwoWay,
NotifyOnValidationError=true, ValidatesOnExceptions=true,
UpdateSourceTrigger=PropertyChanged}"/>
        <Label Content="Shipped Date:"  Grid.Row="4"  Style="{StaticResource Label}"/>
        <DatePicker x:Name="add_shippedDatePicker"  Grid.Row="4"
HorizontalAlignment="Right" Width="120"
            SelectedDate="{Binding ShippedDate, Mode=TwoWay,
NotifyOnValidationError=true, ValidatesOnExceptions=true,
UpdateSourceTrigger=PropertyChanged}"/>
        <Label Content="Ship Via:"  Grid.Row="5" Style="{StaticResource Label}"/>
```

```
        <TextBox x:Name="add_ShipViaTextBox"  Grid.Row="5" Style="{StaticResource
CustTextBox}"
                 Text="{Binding ShipVia, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Freight"  Grid.Row="6" Style="{StaticResource Label}"/>
        <TextBox x:Name="add_freightTextBox" Grid.Row="6" Style="{StaticResource
CustTextBox}"
                 Text="{Binding Freight, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
    </Grid>
    <DataGrid x:Name="ordersDataGrid" SelectionUnit="Cell" SelectionMode="Single"
AutoGenerateColumns="False" CanUserAddRows="false" IsEnabled="True"
EnableRowVirtualization="True" Width="auto" ItemsSource="{Binding
Source={StaticResource customerOrdersViewSource}}" Margin="10,10,10,10"
Grid.Row="2" RowDetailsVisibilityMode="VisibleWhenSelected">
        <DataGrid.Columns>
            <DataGridTemplateColumn>
                <DataGridTemplateColumn.CellTemplate>
                    <DataTemplate>
                        <Button Content="Delete" Command="{StaticResource
DeleteOrderCommand}" CommandParameter="{Binding}"/>
                    </DataTemplate>
                </DataGridTemplateColumn.CellTemplate>
            </DataGridTemplateColumn>
            <DataGridTextColumn x:Name="customerIDColumn" Binding="{Binding
CustomerID}" Header="Customer ID" Width="SizeToHeader"/>
            <DataGridTextColumn x:Name="employeeIDColumn" Binding="{Binding
EmployeeID}" Header="Employee ID" Width="SizeToHeader"/>
            <DataGridTextColumn x:Name="freightColumn" Binding="{Binding Freight}"
Header="Freight" Width="SizeToHeader"/>
            <DataGridTemplateColumn x:Name="orderDateColumn" Header="Order Date"
Width="SizeToHeader">
                <DataGridTemplateColumn.CellTemplate>
                    <DataTemplate>
                        <DatePicker SelectedDate="{Binding OrderDate, Mode=TwoWay,
NotifyOnValidationError=true, ValidatesOnExceptions=true,
UpdateSourceTrigger=PropertyChanged}"/>
                    </DataTemplate>
                </DataGridTemplateColumn.CellTemplate>
            </DataGridTemplateColumn>
            <DataGridTextColumn x:Name="orderIDColumn" Binding="{Binding OrderID}"
Header="Order ID" Width="SizeToHeader"/>
            <DataGridTemplateColumn x:Name="requiredDateColumn" Header="Required
Date" Width="SizeToHeader">
```

```
            <DataGridTemplateColumn.CellTemplate>
               <DataTemplate>
                  <DatePicker SelectedDate="{Binding RequiredDate, Mode=TwoWay,
NotifyOnValidationError=true, ValidatesOnExceptions=true,
UpdateSourceTrigger=PropertyChanged}"/>
               </DataTemplate>
            </DataGridTemplateColumn.CellTemplate>
         </DataGridTemplateColumn>
         <DataGridTextColumn x:Name="shipAddressColumn" Binding="{Binding
ShipAddress}" Header="Ship Address" Width="SizeToHeader"/>
         <DataGridTextColumn x:Name="shipCityColumn" Binding="{Binding
ShipCity}" Header="Ship City" Width="SizeToHeader"/>
         <DataGridTextColumn x:Name="shipCountryColumn" Binding="{Binding
ShipCountry}" Header="Ship Country" Width="SizeToHeader"/>
         <DataGridTextColumn x:Name="shipNameColumn" Binding="{Binding
ShipName}" Header="Ship Name" Width="SizeToHeader"/>
         <DataGridTemplateColumn x:Name="shippedDateColumn" Header="Shipped
Date" Width="SizeToHeader">
            <DataGridTemplateColumn.CellTemplate>
               <DataTemplate>
                  <DatePicker SelectedDate="{Binding ShippedDate, Mode=TwoWay,
NotifyOnValidationError=true, ValidatesOnExceptions=true,
UpdateSourceTrigger=PropertyChanged}"/>
               </DataTemplate>
            </DataGridTemplateColumn.CellTemplate>
         </DataGridTemplateColumn>
         <DataGridTextColumn x:Name="shipPostalCodeColumn" Binding="{Binding
ShipPostalCode}" Header="Ship Postal Code" Width="SizeToHeader"/>
         <DataGridTextColumn x:Name="shipRegionColumn" Binding="{Binding
ShipRegion}" Header="Ship Region" Width="SizeToHeader"/>
         <DataGridTextColumn x:Name="shipViaColumn" Binding="{Binding ShipVia}"
Header="Ship Via" Width="SizeToHeader"/>
      </DataGrid.Columns>
   </DataGrid>
</Grid>
```

Створення таблиць бази даних

```
CREATE TABLE [Tenant] (
  [company id] int IDENTITY (1, 1) ,
  [address object] varchar (50),
  [amount of rent] money ,
  [amount of debt for utilities] money ,
  [amount to be paid] money ,
  [name of company] varchar (50),
```

```
 PRIMARY KEY ([company id])
) ON [PRIMARY]
GO

CREATE TABLE [Estate object] (
  [address object] varchar (50),
  [square] float,
  [name of object] varchar (50),
 PRIMARY KEY ([address object])
) ON [PRIMARY]
GO

CREATE TABLE [Payment] (
  [id pay] int IDENTITY (1, 1) ,
  [company id] int ,
  [number current account] int ,
  [sum] money ,
  [date] datetime ,
  [comment] text ,
 PRIMARY KEY ([id pay])
) ON [PRIMARY]
GO

CREATE TABLE [ Receipt] (
  [id receipt] int IDENTITY (1, 1) ,
  [id pay] int ,
 PRIMARY KEY ([id receipt])
) ON [PRIMARY]
GO

CREATE TABLE [Score] (
  [ number current account] int IDENTITY (1, 1) ,
  [account type] varchar ,
  [company id] int ,
  [amount do be pay] money ,
  [amount of debt] money ,
PRIMARY KEY ([ number current account])
) ON [PRIMARY]
GO

CREATE TABLE [Debt] (
  [amount of  debt] money ,
  [company id] int ,
  [debt formation date] datetime ,
```

```
  [amount of debt for 3% per annum] money ,
  [amount of debt for inflationary losses] money ,
  [amount of debt interest] money ,
 PRIMARY KEY ([amount of  debt])
) ON [PRIMARY]
GO

ALTER TABLE [Tenant] ADD FOREIGN KEY ([address object]) REFERENCES
[Estate object] ([address object]);

ALTER TABLE [Payment] ADD FOREIGN KEY ([company id]) REFERENCES
[Tenant] ([company id]);

ALTER TABLE [ Receipt] ADD FOREIGN KEY ([id pay]) REFERENCES [Payment]
([id pay]);

ALTER TABLE [Score] ADD FOREIGN KEY ([amount of debt]) REFERENCES [Debt]
([amount of  debt]);

 ALTER TABLE [Debt] ADD FOREIGN KEY ([company id]) REFERENCES [Tenant]
                                ([company id]);
```