

АНОТАЦІЯ

У результаті виконання даної дипломної роботи розроблено автоматизовану інформаційну систему по обслуговуванню клієнтів підприємством, що надає послуги по ремонту мобільних пристроїв.

Для програмної реалізації обрано мову програмування C# , для створення користувацького інтерфейсу – технологію WPF (Windows Presentation Foundation).

Робота має практичне значення для забезпечення підвищення продуктивності та полегшення управління підприємством, що займається ремонтом мобільних пристроїв.

У першому розділі дипломної роботи досліджена предметна область - організація ремонту мобільних пристроїв підприємством. Визначені цілі, які повинна реалізовувати інформаційна система та методи її досягнення. Визначені інструментальні засоби розробки системи: СКБД, середовище розробки, платформа розробки та мова програмування.

У другому розділі описано розроблену інформаційну систему для організації ремонту мобільних пристроїв підприємством. Представлена схема та структура бази даних. Розроблено UML діаграму варіантів використання та класів. Описаний функціонал та призначення окремих частин готового програмного продукту демонструється на тестовому прикладі.

ANNOTATION

As a result of this graduation work, an automated information system for servicing clients by the company providing mobile device repair services has been developed.

For program implementation, the C # programming language is selected, for the creation of the user interface - WPF (Windows Presentation Foundation).

The work is of practical importance to improve productivity and facilitate the management of an enterprise that repairs mobile devices.

In the first section of the thesis the subject area is investigated - the organization of repair of mobile devices by the enterprise. The goals, which should be realized by the information system and methods of its achievement, are defined. Defined tools for system development: DBMS, development environment, development platform and programming language.

The second section describes the developed information system for the organization of repair of mobile devices by the enterprise. The scheme and structure of the database are presented. A UML diagram of usage and classes has been developed. The described functionality and the assignment of individual parts of the finished software product is demonstrated on a test case.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AIC – автоматизована інформаційна система;

АС – автоматизовані системи;

АСК – автоматизована система керування;

ІС – інформаційна система;

БД – база даних;

ОС – операційна система.

СППР – система підтримки прийняття рішень;

АСДС - автоматизована система державної статистики;

АСПР - автоматизована система планових розрахунків;

ІТ – інформаційні технології;

API - прикладний програмний інтерфейс;

ООП – об'єктно-орієнтоване програмування;

CLR – загальномовне середовище виконання;

XAML – розширювана мова розмітки;

UML – уніфікована мова моделювання.

ЗМІСТ

ВСТУП	9
Розділ 1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ	11
1.1.Актуальність предметної області.....	11
1.2.Класифікація інформаційних систем.....	13
1.3.Основні вимоги до інформаційної системи	18
1.4.Підвищення ефективності діяльності підприємства за допомогою автоматизації праці	21
1.5.Опис підприємства «СервісФон».....	24
1.6.Дерево цілей	25
1.7.Інструментальні засоби розробки автоматизованої інформаційної системи.....	26
1.7.1 Комп'ютерна платформа .Net Framework	26
1.7.2 Мова програмування C#.....	32
1.7.3 Технологія WPF	34
1.7.4 DevExpress WPF.....	38
Висновок	40
Розділ 2. ОПИС ОБ'ЄКТУ РОЗРОБКИ.....	41
2.1.Вимоги до програмного та технічного забезпечення	41
2.2.Діаграма варіантів використання автоматизованої інформаційної системи.....	42
2.3.Схема та структура бази даних.....	43
2.4.Діаграми класів автоматизованої інформаційної системи	51
2.4.1 Опис головного класу для роботи з вікнами.....	51
2.4.2 Опис класу для роботи з базою даних	52

2.4.3	Опис класу для обчислення даних, що використовуються при побудові графіків	53
2.4.4	Опис класу, що забезпечує використання команд	53
2.4.5	Опис класу для роботи з головним вікном	54
2.4.6	Опис класу для роботи з вікном входу в систему	55
2.5.	Проектна реалізація автоматизованої інформаційної системи	55
	Висновок	70
	ВИСНОВКИ.....	71
	СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ	72
	ДОДАТКИ.....	74
	Додаток А. Файл DataService.cs	74
	Додаток В. Файл WindowsFactory.cs.....	87

ВСТУП

Сучасне життя неможливе без ефективного керування. Важливою категорією керування є системи обробки інформації, від яких багато в чому залежить ефективність роботи будь-якого підприємства. Дана система повинна забезпечувати одержання загальних звітів за підсумками роботи, дозволяти легко визначати тенденції зміни найважливіших показників, забезпечувати одержання інформації, критичної за часом, без істотних затримок, виконувати точний і повний аналіз даних.

Якщо сучасна людина залишається без мобільного зв'язку, у більшості випадків вона відчуває занепокоєння або паніку. Кожен абонент зробить усе можливе, щоб якнайшвидше підключитися до мережі. Тому не дивно, що у разі пошкодження пристрою людина намагається скоріше знайти майстерню, щоб відремонтувати свій телефон.

Метою дипломної роботи є розробка інформаційної системи для підвищення продуктивності та полегшення управління підприємством, що займається ремонтом мобільних пристроїв. Підвищення ефективності та організації обліку є пріоритетним завданням для цієї системи.

У ході виконання дипломної роботи було виконано наступні завдання:

- досліджено предметну область;
- досліджено принципи роботи підприємства, що займається ремонтом мобільних пристроїв;
- сформовано вимоги до інформаційної системи;
- розроблено схему бази даних;
- розроблено UML діаграму варіантів використання даної АІС ;
- розроблено зручний користувацький інтерфейс для роботи з програмним продуктом;
- написано код даної АІС;

– статистичний аналіз.

Об`єктом дослідження є підприємство, що займається ремонтом мобільних пристроїв, сервісний центр «СервісФон».

Предмет дослідження – підвищення ефективності керування підприємством за рахунок використання АІС.

Результатом виконання дипломної роботи є завершена та готова до використання інформаційна система для підприємства, що займається ремонтом мобільних пристроїв.

Розділ 1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Актуальність предметної області

Наявність мобільного телефону на сьогоднішній день є нормою для кожної цивілізованої людини. Завдяки широким функціональним можливостям даний пристрій вже давно витіснив стаціонарні телефони і став незамінним помічником в повсякденному житті кожного. Багато користувачів часто стикаються з проблемою вибору, від якого залежить не тільки продуктивність і функціональність телефону, але і його надійність. Якщо вибрати неякісний пристрій, то наступною проблемою може стати його ремонт, а також пошук відповідного фахівця.

Очевидним є те, що вартість пристрою в першу чергу обумовлена його функціональністю. Також при купівлі мобільного пристрою потрібно орієнтуватись на особу, що безпосередньо буде ним користуватися. Наприклад, якщо телефон призначений для дитини або літньої людини, перевагу слід віддавати бюджетним моделям, так як дана категорія користувачів не дуже вимоглива до оснащення пристрою, та ймовірність поломки телефона у них доволі мала. За статистикою ремонт мобільних телефонів, які належать дітям і старшим людям відбувається значно рідше, оскільки часто поломки – це результат механічних пошкоджень. Для людей, які займаються бізнесом, більш придатними є телефони, оснащені широкими комунікаційними можливостями, а також ємним телефонним довідником. Основна споживча група мультимедійних телефонів складається переважно з молоді, для якої головним є наявність великого дисплея, ємною батареї, камери з високою роздільною здатністю і інших просунутих функцій.

На сьогодні найбільш популярними є такий вид мобільних телефонів, як смартфони. За своєю функціональністю смартфони можна сміливо порівняти з портативними кишеньковими комп'ютерами, так як вони здатні

виконувати складні обчислювальні операції. А наявність сенсорного екрану дозволило в істотній мірі збільшити екран телефону, що зробило його більш зручним для перегляду фотографій, відео та користування інтернетом. Окремої уваги заслуговує можливість миттєвого виходу через смартфон в інтернет, оскільки з його допомогою можна мати доступ до численних інформаційних ресурсів, без необхідності користуватися стаціонарним комп'ютером.

Ще одним надзвичайно популярним мобільним пристроєм у нас час став планшет. Для багатьох цей клас гаджетів є всього лише іграшкою, яка призначена виключно для розваг. Проте є велика кількість продвинутих користувачів, для яких планшет - це інструмент, що допомагає працювати з офісними документами, дозволяє з легкістю демонструвати презентації, а також дає можливість читати книги піклуючись про природу, що доволі актуально у нас час. На даний момент лідируючі позиції по продажах займають планшети на операційній системі «Android». На сьогоднішній день у планшетів досить апаратної продуктивності і готових додатків, щоб редагувати фото, відео, аудіо, здійснювати віддалене адміністрування і навіть займатися програмуванням.

Основний мінус планшетів - відсутність клавіатури. Це накладає певні обмеження та незручності у використанні пристрою. Звичайно одна справа - ввести з сенсорного екрану пошуковий запит в адресному рядку, інша - написати статтю на кілька сторінок. У другому випадку ноутбук виявиться куди практичніше. Правда варто відзначити, що на допомогу планшетам можуть прийти Bluetooth-клавіатури і різноманітні док-станції з повнорозмірною клавіатурою.

Говорячи про будь-який пристрій можна з великою достовірністю стверджувати, що чим більше функцій він виконує тим більша ймовірність його скорішого виходу з ладу. Сучасні мобільні пристрої по своїй

конструкції і функціональності набагато потужніші ніж багато комп'ютерів п'ять років тому назад, а значить причин поломок може бути безліч.

Декілька років тому при поломці телефону розумілася заміна батареї або шлейфа підключення екрану і клавіатури. Сьогоднішня внутрішня начинка смартфонів і планшетів значно складніша, тому окрім вище вказаних причин з ладу можуть вийти:

- динаміки;
- процесор;
- модулі безпроводного зв'язку(Bluetooth і Wifi);
- мікрофон;
- мікросхема;
- слот для карти пам'яті;
- камера.

Також значне число поломок смартфонів відбувається через збій у програмному забезпеченні. Це зазвичай призводить до того, що смартфон не вмикається, зависає, може самовільно вимикатися. За статистикою телефони були і залишаються одним з найбільш уразливих пристроїв, тому ремонт телефонів займає лідируючі позиції серед інших сервісних послуг.

1.2. Класифікація інформаційних систем

Інформаційна система — сукупність організаційних і технічних засобів для збереження та обробки інформації з метою забезпечення інформаційних потреб користувачів.[5]

Інформаційні системи можуть значно різнитися за типами об'єктів управління, характером та обсягом розв'язуваних завдань і рядом інших ознак:

- за рівнем або сферою діяльності — державні, територіальні (регіональні), галузеві, об'єднань, підприємств або установ, технологічних процесів;
- за рівнем автоматизації процесів управління — інформаційно-пошукові, інформаційно-довідкові, інформаційно-керівні, системи підтримки прийняття рішень, інтелектуальні АС;
- за ступенем централізації обробки інформації — централізовані АС, децентралізовані АС, інформаційні системи колективного використання;
- за ступенем інтеграції функцій — багаторівневі АС з інтеграцією за рівнями управління (підприємство — об'єднання, об'єднання — галузь і т. ін.), багаторівневі АС з інтеграцією за рівнями планування.

Державні АС призначені для вирішення найважливіших народногосподарських проблем країни. На базі використання обчислювальних комплексів та економіко-математичних методів у них складають перспективні та поточні плани розвитку країни, ведуть облік результатів та регулюють діяльність окремих ланцюгів народного господарства, розробляють державний бюджет та контролюють його виконання.[7]

Центральне місце в мережі державних АС належить автоматизованій системі державної статистики (АСДС). Роль та місце АСДС в ієрархії управління визначається тим, що вона є основним джерелом статистичної інформації, дуже потрібної для функціонування усіх державних та регіональних АС.[12]

Серед АС, з якими взаємодіє АСДС, важливе місце належить автоматизованій системі планових розрахунків (АСПР). АСПР функціонує при Міністерстві економіки України і являє собою інформаційну систему, призначену для розробки народногосподарських планів та контролю за їх

виконанням в умовах застосування засобів обчислювальної техніки для збору та обробки інформації.

Процес взаємодії АСДС з АСПР має взаємний характер: статистична інформація, джерелом якої є АСДС, необхідна на всіх етапах складання перспективних і поточних планів розвитку господарства країни. У свою чергу, планова інформація надходить до АСДС і є основою для обліку та аналізу виконання планів і завдань. Взаємодія АСДС та АСПР передбачає також спільний аналіз соціально-економічних проблем розвитку народного господарства. Тому АСДС має повністю задовольнити потреби оптимального планування, проводити економіко-математичний аналіз демографічних процесів у суспільстві, міжгалузевих зв'язків, споживання та прибутків населення, показників діяльності підприємств.[5]

АСДС взаємодіє також з державною інформаційною системою фінансових розрахунків (АСФР) при Міністерстві фінансів України.

АСФР призначена для автоматизації фінансових розрахунків на базі сучасної обчислювальної техніки з формування державного бюджету країни та контролю за його виконанням. При цьому вона використовує статистичну інформацію про випуск і реалізацію продукції, фонди споживання, запаси та витрати фінансових ресурсів і т. ін.

Відомі й інші державні АС, система обробки інформації з цін (АСОІ цін), система управління національним банком (АСУ банк), система обробки науково-технічної інформації (АСО НТІ) і т. ін.

Територіальні (регіональні) АС призначені для управління адміністративно-територіальним регіоном. Сюди належать АС області, міста, району. Ці системи виконують роботи з обробки інформації, яка необхідна для реалізації функцій управління регіоном, формування звітності й видачі оперативних даних місцевим і керівним державним та господарським органам.

Галузеві інформаційні системи управління призначені для управління підвідомчими підприємствами та організаціями. Галузеві АС діють у промисловості та в сільському господарстві, будівництві на транспорті і т. ін. У них розв'язуються задачі інформаційного обслуговування апарату управління галузевих міністерств і їх підрозділів.[5]

Інформаційні системи управління підприємствами (АСУП) або виробничими об'єднаннями (АСУ В) — це системи із застосуванням сучасних засобів автоматизованої обробки даних, економіко-математичних та інших методів для регулярного розв'язування завдань управління виробничо-господарською діяльністю підприємства.

Інформаційні системи управління технологічними процесами (АСУ ТП) керують станом технологічних процесів виробництва. Перша й головна відмінність цих систем від розглянутих раніше полягає передусім у характері об'єкта управління — це різноманітні машини, прилади, обладнання. Друга відмінність полягає у формі передачі інформації. Для АСУ ТП основною формою передачі інформації є сигнал, а в інших АСУ — документи.[7]

Залежно від мети функціонування та завдань, які покладені на АС на етапах збору та змістової обробки даних, розрізняють такі типи АС:

- інформаційно-пошукові;
- інформаційно-довідкові;
- інформаційно-управляючі (управлінські);
- інтелектуальні інформаційні системи та системи підтримки прийняття рішень;
- інформаційно-пошукові системи (ІСП) орієнтовані на розв'язування завдань пошуку інформації. Змістова обробка інформації в таких системах відсутня.

В інформаційно-довідкових системах (ІДС) за результатами пошуку обчислюють значення арифметичних функцій.

Інформаційно-управляючі, або управлінські, системи являють собою організаційно-технічні системи, які забезпечують вироблення рішення на основі автоматизації інформаційних процесів у сфері управління. Отже, ці системи призначені для автоматизованого розв'язування широкого кола завдань управління.

До інформаційних систем нового покоління належать системи підтримки прийняття рішень (СППР) та інформаційні системи, побудовані на штучному інтелекті (інтелектуальні АС).

СППР — це інтерактивна комп'ютерна система, яка призначена для підтримки різних видів діяльності при прийнятті рішень із слабо структурованих або неструктурованих проблем. Інтерес до СППР, як перспективної галузі використання обчислювальної техніки та інструментарію підвищення ефективності праці в сфері управління економікою, постійно зростає.[5]

Штучний інтелект — це штучні системи, створені людиною на базі комп'ютерної техніки, що імітують розв'язування людиною складаних творчих завдань. Створенню інтелектуальних інформаційних систем сприяла розробка в теорії штучного інтелекту логіко-лінгвістичних моделей. Ці моделі дають змогу формалізувати конкретні змістовні знання про об'єкти управління та процеси, що відбуваються в них, тобто ввести в ЕОМ логіко-лінгвістичні моделі поряд з математичними. Логіко лінгвістичні моделі — це семантичні мережі, фрейми, продукувальні системи — іноді об'єднуються терміном «програмно-апаратні засоби в системах штучного інтелекту».[12]

Розрізняють три види інтелектуальних АС:

- інтелектуальні інформаційно-пошукові системи (системи типу «запитання — відповідь»), які в процесі діалогу забезпечують

взаємодію кінцевих користувачів — не програмістів з базами даних та знань професійними мовами користувачів, близьких до природних;

- розрахунково-логічні системи, які дають змогу кінцевим користувачам, що не є програмістами та спеціалістами в галузі прикладної математики, розв'язувати в режимі діалогу з ЕОМ свої задачі з використанням складаних методів і відповідних прикладних програм;
- експертні системи, які дають змогу провадити ефективну комп'ютеризацію областей, у яких знання можуть бути подані в експертній описовій формі, але використання математичних моделей утруднене або неможливе.

В економіці України найпоширенішими є експертні системи. Це системи, які дають змогу на базі сучасних персональних комп'ютерів виявляти, нагромаджувати та коригувати знання з різних галузей народного господарства (предметних областей).

1.3. Основні вимоги до інформаційної системи

Сучасні інформаційні системи не залежно від їхнього масштабу, програмно-апаратної платформи і вартості повинні забезпечувати якісне ведення обліку, бути надійними і зручними в експлуатації.[2]

У функціональному аспекті вимогами до даної інформаційної систем є:

- безпомилкові арифметичні розрахунки;
- забезпечення підготовки , заповнення, роздрукування первинних і звітних документів;
- полегшення доступу до бази даних товарів для прийняття рішення, щодо замовлення;
- спрощення та прискорення обробки даних;

- забезпечення звертання до даних і звітів за минулі періоди (вести архів).

На підприємстві інформаційна система дає можливість уникати ручної технічної праці при оформленні ремонту, обстеження чи продажу/купівлі запчастин, та швидкого доступу до залишків товарів та перевірки стану виконання роботи. Подання статистики підприємства в електронному вигляді теж являється перевагою інформаційної системи .

Для нарахування заробітної плати підприємство використовує дві формули. Для робітника формула 1.1. Для оператора формула 1.2. [3]

$$\text{Зарплата} = \text{Ставка} * \sum \text{ПроведенихРемонтівОбстежень} * 10\% \quad (1.1)$$

$$\text{Зарплата} = \text{Ставка} * \sum \text{ПроданихЗапчастин} * 10\% \quad (1.2)$$

Для того, щоб забезпечити зазначені можливості, інформаційна система повинна мати єдину базу даних по бухгалтерському обліку на підприємстві та обліку запчастин, і дана інформація має легко отримуватись на запит користувача. 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 налаштування системи на конкретну облікову діяльність .

Надійність інформаційної системи в комп'ютерному плані означає захищеність її від випадкових збоїв і в деяких випадках від навмисного псування даних. Як відомо, сучасні персональні комп'ютери є досить відкритими, тому не можна вірогідно гарантувати захист чисто на фізичному рівні. Важливо, щоб після збою зруйновану базу даних можна було легко відновити, а роботу системи відновити в найкоротший термін.

Для збільшення ефективності роботи підприємства створюються АІС, які зменшують витрати на людські ресурси.

Підприємство створює інформаційну систему, для більш ефективної роботи. На підприємстві з економічної точки зору інформаційна система може розглядатися як засіб, який може вільно замінювати робочу силу.[5]

Сучасна технологія передачі даних дозволяє організовувати роботу більш гнучкими способами, підвищуючи здатність реагувати на зміни в ринку. Інформаційна система надає підприємству додаткову гнучкість. [7]

Таким чином, впровадження системи управління на підприємстві створить передумови для якісного поліпшення процесу управлінського планування й контролю діяльності з боку керівництва.

Також клієнт через впровадження на підприємстві автоматизованої інформаційної системи зможе отримати більш зручне та якісне обслуговування, що в свою чергу дозволить зекономити його час та нерви під час ремонту свого мобільного пристрою.

Дана інформаційна система повинна підтримувати три типи користувачів:

- адміністратор;
- робітник;
- оператор.

Адміністратор - це користувач, що має найбільше прав в даній системі він може додавати нових робітників чи операторів і отримувати статистику про роботу підприємства.

Оператор – це користувач, що безпосередньо взаємодіє з клієнтами та приймає замовлення.

Робітник – це користувач, що займається ремонтом пристроїв та за допомогою інформаційної системи оперативно повідомляє про його стан.

1.4. Підвищення ефективності діяльності підприємства за допомогою автоматизації праці

Удосконалення системи управління підприємством - це комплекс заходів, покликаний допомогти керуючому складу підприємства ефективніше використовувати його ресурси, в свій час і швидше знаходити, виправляти недоліки в управлінні.

Якщо підприємство працює не ефективно або не так ефективно, як хотілося б керівництву, проблеми потрібно шукати в системі управління, в технічних виконавцях і в іншому персоналі підприємства. Якщо виправити проблеми в керуючій системі, то більше половини проблем на підприємстві зникнуть самі собою. Таким чином, кошти, витрачені на АІС, в майбутньому неодноразово окупляться, бо не ефективність системи управління на підприємстві - це практично "смертний вирок" цьому підприємству в найближчому майбутньому.[6]

У сучасних умовах поряд з фінансовими, матеріальними, людськими та іншими ресурсами, ефективне управління являє собою цінний ресурс організації. Отже, підвищення ефективності управлінської діяльності стає одним з напрямків вдосконалення діяльності підприємства в цілому. Найбільш очевидним способом підвищення ефективності трудового процесу є його автоматизація.

Управлінська праця відрізняється складністю і різноманіттям, наявністю великого числа форм і видів, багатосторонніми зв'язками з різними явищами і процесами. Це насамперед праця творча та інтелектуальна, в основному пов'язаний з ІТ. Тому автоматизація управлінської діяльності спочатку пов'язувалася тільки з автоматизацією деяких допоміжних, рутинних операцій. Але бурхливий розвиток інформаційних комп'ютерних технологій, вдосконалення технічної платформи і поява принципово нових

класів програмних продуктів привело в наші дні до зміни підходів в автоматизації управління не тільки виробництвом, а й іншими процесами.[4]

Головним напрямком автоматизації діяльності та її радикального вдосконалення, пристосування до сучасних умов, стало масове використання новітньої комп'ютерної і телекомунікаційної техніки, формування на її основі високоефективних ІТ. Засоби і методи прикладної інформатики використовуються в управлінні та маркетингу.

Нові технології, засновані на комп'ютерній техніці, вимагають радикальних змін організаційних структур управління, його регламенту, кадрового потенціалу, системи документації, фіксації переробки і передачі інформації. Особливе значення має впровадження ІТ, значно розширювальної можливості використання компаніями інформаційних ресурсів. Розвиток ІТ зв'язано з організацією системи обробки даних і знань, послідовного їхнього розвитку до рівня інтегрованих автоматизованих систем управління, що охоплюють по вертикалі і горизонталі всі рівні і ланки виробництва.[7]

Згідно визначення, прийнятого ЮНЕСКО, ІТ - це комплекс взаємозалежних, наукових, технологічних, інженерних дисциплін, що вивчають методи ефективної організації праці людей, зайнятих обробкою і зберіганням інформації; обчислювальну техніку і методи організації і взаємодії з людьми і виробничим устаткуванням, практичні додатки, а також пов'язані з усім цим соціальні, економічні та культурні проблеми. Самі ІТ вимагають складної підготовки, великих початкових витрат і наукомісткої техніки.[6]

Основний ефект ІТ проявляють в автоматизованих режимах. При цьому основна мета автоматизованої ІТ - отримувати за допомогою переробки первинних даних інформацію нової якості, на основі якої виробляються оптимальні управлінські рішення. Це досягається за рахунок

інтеграції інформації, забезпечення її актуальності і несуперечності, використання технічних засобів для впровадження та функціонування якісно нових форм інформаційної підтримки діяльності апарату управління.

Автоматизовані ІТ припускають використання комплексу відповідних технічних засобів, що реалізують інформаційний процес і системи управління цим комплексом технічних засобів (як правило, це програмні засоби та організаційно-методичне забезпечення, що погоджує дії персоналу і технічних засобів у єдиний технологічний процес). Оскільки істотну частину технічних засобів для реалізації ІТ займають засоби комп'ютерної техніки, то часто під ІТ, особливо під новими технологіями, розуміють комп'ютерні ІТ.

Інший напрямок вдосконалення систем управління формує необхідність діяти в умовах ринкової економіки, що де все загострюється за рахунок конкуренції товаровиробників. Вона обумовлює підвищені вимоги до професійних якостей фахівців, відповідальності керівників за результати і наслідки прийнятих рішень. Надзвичайно актуальними стають облік тимчасового чинника і організація аналізу матеріальних, товарних, фінансових потоків, пошук обґрунтованих рішень у регулюванні виробничо-господарських і фінансових ситуацій. Всі ці вимоги і проблеми можуть бути легко подолані застосуванням автоматизованих інформаційних систем.[2]

Підготовка фахівців різного профілю (практично всіх професій) повинна передбачати оволодіння ними фундаментальними знаннями теорії і практики, а також умінням активно використовувати ІТ у своїй професійній діяльності. Широке застосування персональних комп'ютерів, засобів комунікації, полегшений доступ до БД і базам знань, використання інтелектуальних технологій і систем забезпечують фахівцеві можливості для виконання аналітичних, прогнозних функцій, підготовки управлінських рішень в сучасному технологічному режимі обробки інформації. Цей

напрямок реалізується на підприємстві насамперед у рамках вдосконалення системи управління персоналом, яке за сучасним підходам має орієнтуватися на методологію соціального менеджменту та ІТ управління.

1.5. Опис підприємства «СервісФон».

СервісФон підприємство, що займається ремонтом мобільних телефонів та планшетів, як на гарантійній основі так і за кошти клієнтів, та продажем запасних частин. Структура підприємства зображена на рис. 1.1.

Дане підприємство складається з 3-х основних відділень:

- відділення для роботи з клієнтом;
- відділення проведення ремонту;
- склад запчастин.

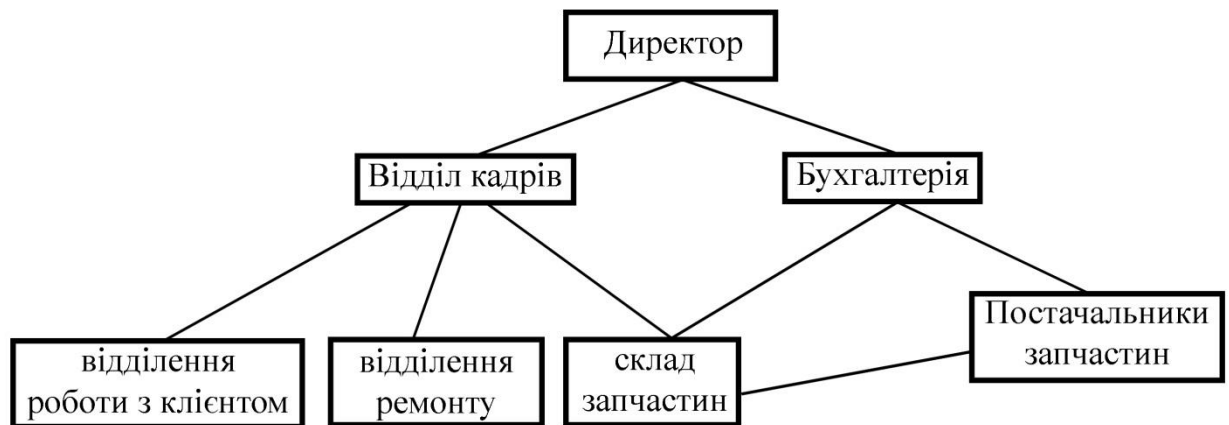


Рис.1.1.Структура підприємства «СервісФон»

Також підприємство містить відділ кадрів, що займається набором професійних робітників та менеджерів для спілкування з клієнтами. Для здійснення бухгалтерського обліку господарської діяльності на підприємстві є окреме відділення бухгалтерія, яка тісно співпрацює з відділом матеріально-технічного постачання та відділом ремонту. Бухгалтерія

отримує від них необхідні для обліку і контролю документи і надає їм обліково-економічну інформацію.

1.6. Дерево цілей

Дерево цілей - це графічне зображення взаємозв'язку і підпорядкованості цілей, що відображає розподіл місії і мети на цілі, під цілі, завдання та окремі дії.

Дерево цілей можна визначити, як цільовий каркас організації, явища чи діяльності. Загальний вигляд дерева цілей показано на рис. 1.2.

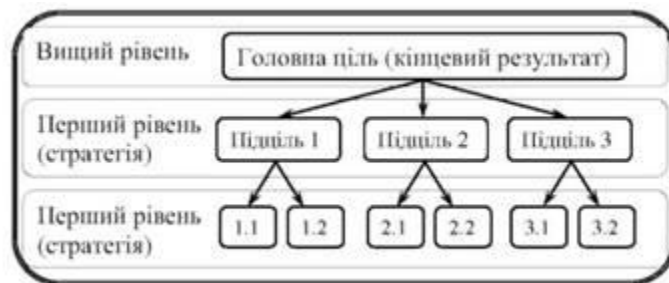


Рис.1.2. Послідовність побудови дерева цілей

Дерево цілей з кількісними показниками, що використовуються в якості одного із засобів при прийнятті рішень, і носить назву дерева рішень. Головна перевага дерева рішень перед іншими методами - можливість пов'язати ставлення цілі з діями, що підлягають реалізації в сьогоденні.

Основна ідея щодо побудови дерева цілей - декомпозиція.

Декомпозиція (розукрупнювання) - це метод розкриття структури системи, при якому за однією ознакою її поділяють на окремі складові.

Декомпозиція використовується для побудови дерева цілей, щоб пов'язати генеральну мету зі способами її досягнення, що сформульовані у вигляді завдань окремим виконавцям.

Розглянемо технологічні засади побудови дерева цілей. Не існує універсальних методів побудови дерева цілей. Способи його побудови

залежать від характеру цілі, обраного методологічного підходу, а також від того, хто розробляє дерево цілей, як він представляє собі поставлені перед ним завдання, як він бачить їхній взаємозв'язок.

Основне правило побудови дерева цілей - це повнота редукції - процес зведення складного явища, процесу або системи до більш простих складових.

Для реалізації цього правила використовують такий системний підхід: ціль вищого рівня є орієнтиром, основою для розробки (декомпозиції) цілей нижчого рівня. Для автоматизованої інформаційної системи по обслуговуванню клієнтів підприємством, що надає послуги по ремонту мобільних пристроїв також розроблене дерево цілей, що зображене на рис. 1.3.

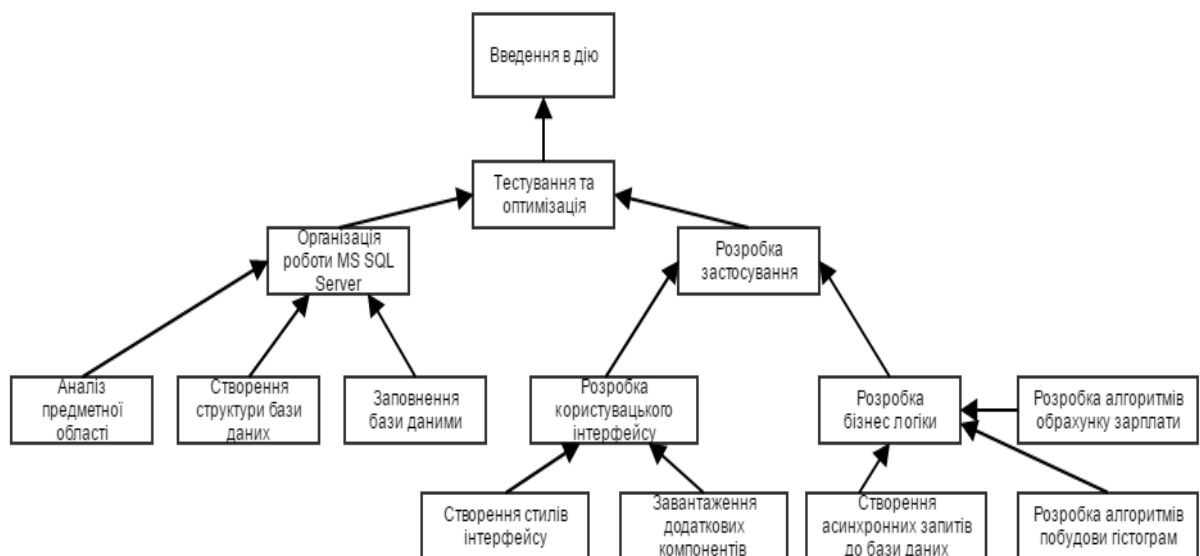


Рис.1.3.Дерево цілей

1.7. Інструментальні засоби розробки автоматизованої інформаційної системи

1.7.1 Комп'ютерна платформа .Net Framework

.NET Framework - це платформа програмування. Взагалі, комп'ютерна платформа - це апаратний або програмний комплекс, що служить основою

для різних обчислювальних систем. Прикладом платформи програмування може бути операційна система комп'ютера. Алгоритмічна мова C# якраз і створена для роботи на платформі .NET.

Розробка програмного забезпечення (ПЗ) на платформах операційних систем (ОС) сімейства Windows мала на меті використання мови програмування C у поєднанні із спеціальними засобами ОС Windows, які називаються скорочено API. Це аббревіатура від Application Programming Interface - інтерфейс прикладного програмування. У цьому інтерфейсі зосереджені великі програмні структури, що дозволяють шляхом їх налаштування на конкретне застосування автоматизувати процес трудомісткого програмування на C.

Все заново створене зазвичай дуже недосконале і доробляється в процесі тривалої експлуатації. Потреба відійти від використання безпосередньо в програмуванні засобів API привела до створення досконаліших систем програмування типу, наприклад, Borland C++ Builder, які значно полегшили та зробили важку працю програміста витонченою. Проте життя не стоїть на місці, і мова C на певному етапі перестала забезпечувати потреби програмування. На горизонті з'явилася концепція так званого об'єктно-орієнтованого програмування (ООП), яка дозволяла подивитися на сам процес створення програмного продукту зовсім з іншого боку, даючи програмісту ширші можливості для автоматизації його праці і створення якіснішої програмної продукції.[4]

Основою ООП стали поняття класу та об'єкта. Розробники мови C пішли шляхом додавання до C структури "клас". Вийшла мова C++. Цей процес виявився настільки непростим, що самі розробники дуже пошкодували, що прийняли саме таку концепцію бути на рівні сучасних вимог до процесу створення програмного продукту. У гонитві за швидкістю обробки застосуваннями даних і за потрібною надійністю і безпекою роботи

застосувань розробникам довелося організовувати два види пам'яті при обробці даних: некеровану (у С пам'яттю доводиться управляти вручну) і керовану (в С++ цю функцію бере на себе спеціальне середовище, так звана керована куча, тому управління пам'яттю - автоматичне), організувати спеціальний і досить неприємний апарат покажчиків. [10]

Розробникам довелося будувати апарат переходу між даними з керованої пам'яті в некеровану і навпаки. Легше було поховати С і створити наново іншу мову на новій концепції. Але розробники були зв'язані по руках: дуже багато програмного продукту на С вже працювало в світі, і поставити на ньому хрест означало підірвати виробничий процес багатьох підприємств та організацій. Тому доводилося не тільки піклуватися про збереження С, але і дотримуватися сучасних вимог (створення С++), підтримувати сумісність старих програм при роботі в нових середовищах. Тобто потрібно було тягнути за собою хвости С в нову мову С++, які тільки заважали новій мові та ускладнювали процес розробки програм на цій мові.[11]

Врешті решт, мабуть, у розробників терпець увірвався, і вони створили нову мову під назвою С#, яка враховує нові віяння в програмуванні (ООП) і вільну від недоліків С++. Проте і С++ не виявилася покинутою з причини, відміченої раніше (сумісність і підтримка вже працюючих у світі програм).

Якщо встановити безкоштовний продукт фірми Microsoft .NET 4.0 Framework Software Development Kit(SDK) або середовище Visual Studio 2010, то для програмування на основі платформи .NET стають доступними мови С#, F#, JScript .NET, Visual Basic, С++/CLI. Тут CLI (Common Language Infrastructure, спільномовна інфраструктура) - прив'язка С++ до платформи .NET. Повернемося все таки до платформи .NET, на базі якої функціонує С#.

Ця платформа є програмною платформою для створення застосувань не лише на базі ОС сімейства Windows, але й інших ОС, які створювалися не фірмою Microsoft, як Windows. Це системи Mac OS X, UNIX, Linux.

Платформа забезпечує взаємодію із вже існуючим програмним забезпеченням. Додатки на платформі .NET можна створювати за допомогою багатьох мов програмування, таких як C#, F#, S#, Visual Basic та ін.

Сьогодні фірма Microsoft випускає продукт під назвою Visual Studio (2017), який дає можливість створювати додатки різними мовами на платформі .NET. Усі мови, підтримувані .NET мають спільний виконуючий механізм. Платформа містить в собі велику і, що важливо, спільну для усіх підтримуваних мов бібліотеку базових класів, які забезпечують, наприклад, введення-виведення даних, роботу додатків з графічними об'єктами, створення не лише веб-інтерфейсів, але і звичайних (настільних) і консольних (без графіки) застосувань, роботу з базами даних, дають можливість створювати інтерфейси для роботи з віддаленими об'єктами.

Зокрема, платформа .NET Framework – це кероване середовище виконання, що надає різноманітні служби працюючим в ньому застосуванням. Вона складається з двох основних компонентів: виконавчого середовища спільної мови (Common Language Runtime, CLR), яка є механізмом, управляючим застосуванням, яке виконуються, і бібліотеки класів .NET Framework, яка представляє бібліотеку перевіреного коду, призначену для повторного використання, який розробники можуть викликати зі своїх застосувань.

Служби (точніше - сервіси, а ще точніше - послуги), які платформа .NET Framework надає працюючим застосуванням:

- управління пам'яттю. У багатьох мовах програмування розробники самостійно призначають і виділяють ресурси пам'яті і вирішують питання, пов'язані з часом життя об'єктів. У застосуваннях платформи .NET Framework середовище CLR надає ці сервіси автоматично;

- система спільного типу. У традиційних мовах програмування базові типи визначаються компілятором, що ускладнює взаємодію між мовами. У платформі .NET Framework базові типи визначаються єдиною системою типу .NET Framework, яка називається CTS (Common Type System). При цьому використовуються одні і ті ж базові типи для усіх мов .NET Framework;
- розширена бібліотека класів. Замість того щоб писати багато коду для виконання стандартних низькорівневих операцій програмування, розробники можуть використати легкодоступну бібліотеку типів і членів з бібліотеки класів .NET Framework;
- платформи і технології розробки. Платформа .NET Framework включає бібліотеки для конкретних областей розробки застосувань, наприклад ASP.NET для веб-застосувань, ADO.NET для доступу до даних і Windows Communication Foundation для застосувань, орієнтованих на служби (сервіси);
- взаємодія мов. Мовні компілятори на платформі .NET Framework компілюють додаток не у виконуваний код відразу, а в проміжний код, що називається мовою CIL (Common Intermediate Language), який згодом компілюється під час виконання застосування середовищем CLR. Такий підхід приводить до того, що програми, написані на одній мові, доступні в інших мовах, а розробники можуть зосередитися на створенні додатків на мові, якій віддається перевага, або мовах;
- сумісність версій. За рідкісними виключеннями, додатки, які розробляються за допомогою платформи .NET Framework певної версії, можуть виконуватися без змін на більш пізній версії;
- паралельне виконання. Платформа .NET Framework допомагає у вирішенні конфліктів версій, дозволяючи встановлення декількох

версій середовища CLR на одному комп'ютері. Це означає, що декілька версій додатків також можуть співіснувати, і що застосування може виконуватися на версії платформи .NET Framework, для якої воно було створене;

- налаштування для різних версій. Орієнтуючись на переносність бібліотеки класів платформи .NET Framework, розробники можуть створювати збірки (exe - або dll- файли, призначені для виконання), які працюють на декількох платформах .NET Framework. Наприклад, на .NET Framework, Silverlight, Windows, Phone 7 або Xbox 360.

Якщо ви не розробляєте застосування .NET Framework, але використовуєте їх, ви не повинні володіти якимись спеціальними знаннями про платформу .NET Framework або її роботу.

Якщо використовується ОС Windows, платформа .NET Framework може бути вже встановлена на комп'ютері. Крім того якщо встановлюється додаток, що вимагає платформу .NET Framework, програма встановлення застосування може інсталиувати конкретну версію .NET Framework на вашому комп'ютері. Інколи можна побачити діалогове вікно, яке просить встановити платформу .NET Framework.[11]

Зазвичай, не вимагається видаляти які-небудь версії .NET Framework вже встановлені на вашому комп'ютері, тому що використовуваний додаток може залежати від конкретної версії. У разі видалення якої-небудь версії його виконання може завершитися помилкою. На одному комп'ютері може бути одночасно завантажено декілька версій платформи .NET Framework. Це означає, що не треба видаляти попередні версії для встановлення пізнішої версії.

Розробник може вибрати будь-яку мову програмування, яка підтримує платформу .NET Framework, для створення застосування. Через те, що платформа .NET Framework забезпечує незалежність і взаємодію мов, можна

взаємодіяти з іншими застосуваннями компонентами платформи .NET Framework незалежно від мови, за допомогою якої вони були розроблені.

1.7.2 Мова програмування C#

Мова C# — це багатопарадигмова об'єктно-орієнтована та компонентно-орієнтована мова програмування зі строгою типизацією, розроблена для платформи .NET Framework. Використання мови визначене стандартами ECMA-3344 [14] та ISO/IEC 23270:20065 [15]. Розроблена командою Microsoft Research під керівництвом Андерса Гейлсберга. Синтаксис мови близький до мов C++ та Java.

Деякі риси (наприклад, строга статична типизація), наближують її структуру до Delphi (Object Pascal).Цілі, поставлені при розробці мови C#, були такими:[9]

- C# має бути простою, сучасною, об'єктно-орієнтованою мовою програмування;
- мова має підтримувати безпечні принципи програмування, такі як строга перевірка типів, перевірка меж масиву, виявлення спроб використання неініціалізованих змінних, і автоматичне прибирання сміття;
- можливість розробки програмних компонентів для розподілених систем;
- мова має підтримувати переносимість коду;
- підтримка національних мовних та інших особливостей має бути простою.

Базовий синтаксис C# схожий до інших C-подібних мов, таких як C, C++ та Java, зокрема:

- крапку з комою використовують для позначення кінця інструкції;
- фігурні дужки використовують для формування програмних блоків;

- значення змінним присвоюють за допомогою знаку "=", а для їх порівняння використовують "==";
- квадратні дужки використовують для індексації масивів.

Проте, C# має суттєві відмінності від розглянутих мов- попередників, у тому числі: [8]

- краща переносимість між різними платформами, пов'язана з тим, що мова відповідає специфікації CLI;
- строга типизація робить мову безпечнішою. C# підтримує логічний тип Boolean;
- мова забезпечує використання властивостей у класах, роблячи роботу з об'єктами зручнішою та безпечнішою;
- програма на C# краще структурована завдяки групуванню коду в простори імен;
- обмежене використання вказівників робить безпечнішою роботу з пам'яттю.

Перша версія C# 1.0 вийшла разом з Microsoft Visual Studio .NET у лютому 2002 року. В липні 2015 року випущено версію C# 6.0. Порівняно із першою версією, мову C# суттєво розвинули, доповнивши її рядом вагомих рис, які роблять її сучасною, гнучкою та перспективною для вирішення широкого кола завдань.[1] Існує кілька середовищ програмування, які використовують мову C#, зокрема: Microsoft Visual Studio, MonoDevelop, SharpDevelop тощо.

Найвживанішим з них є інтегроване середовище розробки Microsoft Visual Studio, яке крім C#, підтримує і ряд інших мов програмування, а також забезпечує великий набір класів для розв'язування різних завдань.

1.7.3 Технологія WPF

Технологія WPF (Windows Presentation Foundation) є частина платформи .NET і являє собою підсистему для побудови графічних інтерфейсів для користувачів.

Якщо при створенні традиційних додатків на основі WinForms за малювання елементів управління і графіки відповідали такі частини операційної системи Windows, як User32 і GDI +, то додатки WPF засновані на DirectX. У цьому полягає ключова особливість рендеринга графіки в WPF: використовуючи WPF, значна частина роботи по відображенні графіки, як найпростіших кнопочок, так і складних 3D-моделей, лягає на графічний процесор на відеокарті, що також дозволяє скористатися апаратним прискоренням графіки.[16]

Однією з важливих особливостей є використання мови декларативної розмітки інтерфейсу XAML, заснованого на XML: ви можете створювати насичений графічний інтерфейс, використовуючи або декларативне оголошення інтерфейсу, або код на керованих мовах C # і VB.NET, або поєднувати і те, і інше.[17]

Переваги використання WPF:

- використання традиційних мов .NET-платформи - C # і VB.NET для створення логіки додатка;
- можливість декларативного визначення графічного інтерфейсу за допомогою спеціальної мови розмітки XAML, заснованому на xml і представляє альтернативу програмному створення графіки та елементів управління, а також можливість комбінувати XAML і C # / VB.NET;
- незалежність від дозволу екрану: оскільки в WPF всі елементи вимірюються в незалежних від пристрою одиницях, додатки на WPF легко масштабуються під різні екрани з різним розширенням;

- нові можливості, яких складно було досягти в WinForms, наприклад, створення тривимірних моделей, прив'язка даних, використання таких елементів, як стилі, шаблони, теми і ін;
- хорошу взаємодію з WinForms, завдяки чому, наприклад, в додатках WPF можна використовувати традиційні елементи управління з WinForms.
- багаті можливості по створенню різних додатків: це і мультимедіа, і двовірна і тривимірна графіка, і багатий набір вбудованих елементів управління, а також можливість самим створювати нові елементи, створення анімацій, прив'язка даних, стилі, шаблони, теми і багато іншого;
- апаратне прискорення графіки - незалежно від того, чи працюєте ви з 2D або 3D, графікою або текстом, всі компоненти програми транслуються в об'єкти, зрозумілі Direct3D, і потім візуалізуються за допомогою процесора на відеокарті, що підвищує продуктивність і робить графіку більш плавною;
- створення додатків під безліч ОС сімейства Windows - від Windows XP до Windows 10

У той же час WPF має певні обмеження. Незважаючи на підтримку тривимірної візуалізації, для створення додатків з великою кількістю тривимірних зображень, перш за все ігор, краще використовувати інші засоби - DirectX або спеціальні фреймворки, такі як Monogame або Unity.

Також варто враховувати, що в порівнянні з додатками на Windows Forms обсяг програм на WPF і споживання ними пам'яті в процесі роботи в середньому трохи вище. Але це з лишком компенсується більш широкими графічними можливостями і підвищеною продуктивністю при відображенні

Схематично архітектура WPF зображена на рис. 1.4.

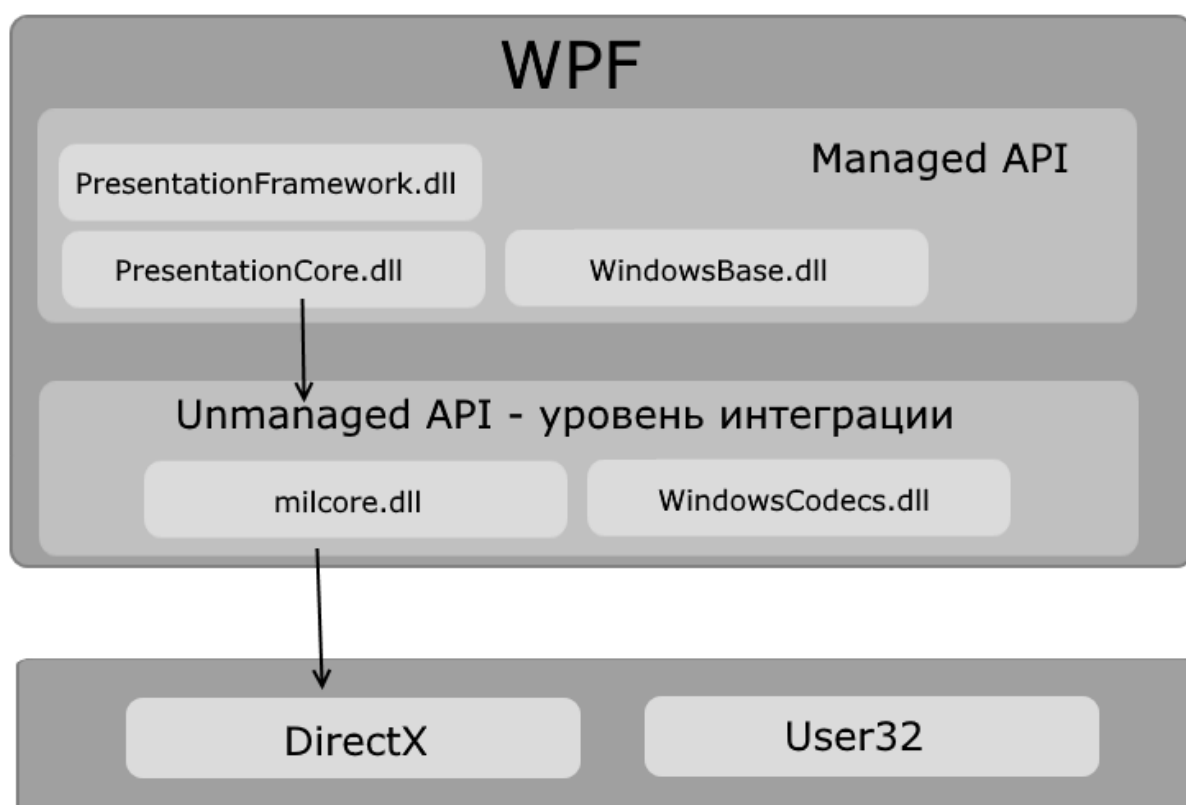


Рис.1.4.Архітектура WPF

Як видно на схемі, WPF розбивається на два рівня: managed API і unmanaged API (рівень інтеграції з DirectX). Managed API (керований API-інтерфейс) містить код, що виконується під управлінням загальномовного середовища виконання .NET - Common Language Runtime. Цей API описує основний функціонал платформи WPF і складається з наступних компонентів:

- PresentationFramework.dll: містить всі основні реалізації компонентів і елементів управління, які можна використовувати при побудові графічного інтерфейсу;
- PresentationCore.dll: містить всі базові типи для більшості класів з PresentationFramework.dll;
- WindowsBase.dll: містить ряд допоміжних класів, які застосовуються в WPF, але можуть також використовуватися і поза даної платформи

Unmanaged API використовується для інтеграції вищого рівня з DirectX;

- `milcore.dll` власне забезпечує інтеграцію компонентів WPF з DirectX. Даний компонент написаний на некерованому коді (C / C++) для взаємодії з DirectX;
- `WindowsCodecs.dll`: бібліотека, яка надає низькорівневу підтримку для зображень в WPF.

Ще нижче власне знаходяться компоненти операційної системи і DirectX, які відтворюють візуалізацію компонентів програми, або виконують іншу низькорівневу обробку. Зокрема, за допомогою низькорівневого інтерфейсу `Direct3D`, який входить до складу DirectX, відбувається трансляція.

Тут також на одному рівні знаходиться бібліотека `user32.dll`. І хоча вище говорилося, що WPF не використовує цю бібліотеку для рендеринга і візуалізації, проте для ряду обчислювальних задач (що не включають візуалізацію) дана бібліотека продовжує використовуватися.

WPF є частиною .NET і розвивається разом з фреймворком .NET і має ті ж версії. Перша версія WPF 3.0 вийшла разом з .NET 3.0 і операційною системою Windows Vista в 2006 році. З того часу платформа послідовно розвивається. Остання версія WPF 4.6 вийшла паралельно з .NET 4.6 в липні 2015 року, ознаменувавши дев'ятиріччя даної платформи.

1.7.4 DevExpress WPF

DevExpress WPF є програмний пакет, до складу якого входить понад 80 потужних елементів управління і бібліотек для розробки додатків на платформі Windows Presentation Foundation. Оптимізовані інструменти даного рішення дозволяють програмістам розробляти гнучко настроювані застосування, що відповідають конкретним завданням бізнесу.

DevExpress WPF пропонує більш 20 тем для додатків, дозволяє оптимізувати інтерфейс і для десктопів і для тачскріну, а також має структуру Model-View-ViewModel (MVVM) для побудови гнучких програм під WPF і Silverlight.

При розробці програми хочеться сконцентрувати увагу на розв'язання основного завдання, але на практиці значні зусилля можуть піти на рішення типових задач - задач, які вже були вирішені тисячі разів.

Типові елементи призначеного для користувача інтерфейсу - це та частина програми, яка може бути багаторазово використана відразу в декількох проектах. Прості способи взаємодії контролю дозволяють ще на етапі розробки прототипу додатку отримати з коробки миттєву роботу з великими базами даних, touch інтерфейс і багато іншого, дозволяючи надалі сконцентруватися на конкретному завданні.[16]

Компоненти діаграм для WPF дозволяють візуально відобразити складну інформацію за допомогою ієрархічних діаграм і блок-схем (див рис.1). Основні можливості цих компонентів:

- більш ніж 110 різних фігур, що включають в себе як основні елементи, так і елементи блок-схем і SDL діаграм;
- теми діаграм, схожі на ті, що ви можете знайти в Visio: Office, Linear, Integral, Daybreak, Parallel, Sequence і Lines. А також можливість створювати свої власні теми;
- коннектори (прямі, вигнуті і прямокутні);

- 15 стилів для сполучних стрілок;
- автоматичне складання шляху для конекторів за допомогою A * алгоритму;
- автоматичне розміщення фігур;
- редактор діаграм для кінцевого користувача;
- можливість збереження / завантаження діаграм.

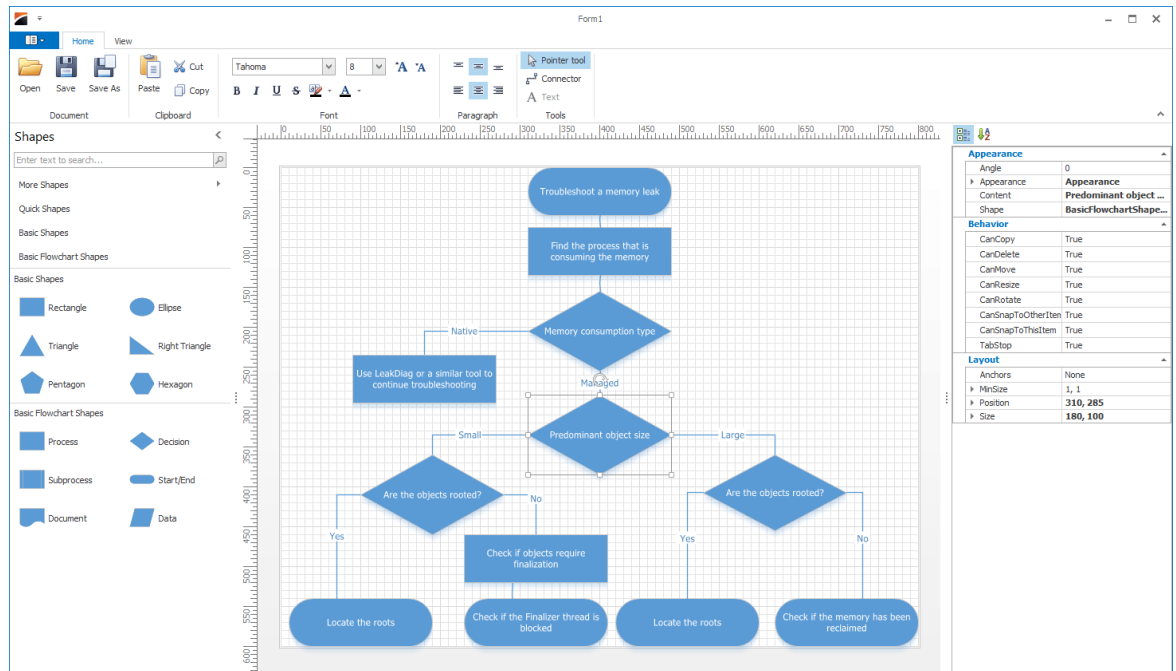


Рис.1.4.Розробка блок-схем за допомогою компонентів DevExpress

Також у цій бібліотеці є можливість створення звітів на основі компонента GridControl. Тепер за допомогою лише кількох рядків коду можна викликати Дизайнер Звіту, зробленого на основі Grid Control і створити звіт з використанням його даних.

Новий компонент TreeMap для платформи WPF дозволить вам візуалізувати табличні або ієрархічно структуровані дані в вигляді вкладених прямокутників, чий розмір залежить від поданих значень. Цей контрол поставляється з наступними вбудованими можливостями:

- кілька алгоритмів розрахунку розміру і положення елементів: «Slice and Dice», «Squarified», «Striped»;
- можливість візуалізувати плоскі (табличні) і ієрархічно структуровані дані;
- можливість автоматично задавати колір елементів, використовуючи один з наступних методів: по елементно за допомогою градієнта, по групах за допомогою градієнта, по палітрі, в залежності від значення елемента;
- інтерактивність: підсвічування елементів при наведенні курсору і можливість вибору елементів при натисканні і програмно;
- гнучко настраюється зовнішній вигляд TreeMap;
- настраюються виринаючі підказки.

Висновок

В даному розділі досліджено предметну область обслуговування клієнтів підприємством, що надає послуги по ремонту мобільних пристроїв, представлені основні вимоги до системи. Досліджені механізми підвищення ефективності підприємства за рахунок впровадження на ньому АІС. Розглянуті інструментальні засоби для розробки автоматизованої інформаційної системи.

Розділ 2. ОПИС ОБ'ЄКТУ РОЗРОБКИ

2.1. Вимоги до програмного та технічного забезпечення

Даний програмний продукт розроблений з використанням сучасних передових технологій розробки десктопних застосунків тому технічні вимоги для цього продукту є досить високі порівняно з іншими продуктами такої категорії застосунків. У таблиці табл.2.1 представлені технічні вимоги для даної програми.

Таблиця 2.1

Мінімальні технічні вимоги для АІС

Оперативна пам'ять	512мб
Процесор	Intel i3
Об'єм вільного простору на диску	200мб
Відеокарта	512мб
Оперційна система	Windows 7/ Windows 10

Також для запуску програмного продукту на робочій машині повинен бути встановлений MS SQL Server 2014. Він може знаходитися на одному комп'ютері, де розміщена основна база даних.

Так як інформаційна система була написана на мові програмування C#, то перед запуском програми потрібно встановити .Net 4.6, без якого програма не зможе коректно працювати.

2.2. Діаграма варіантів використання автоматизованої інформаційної системи

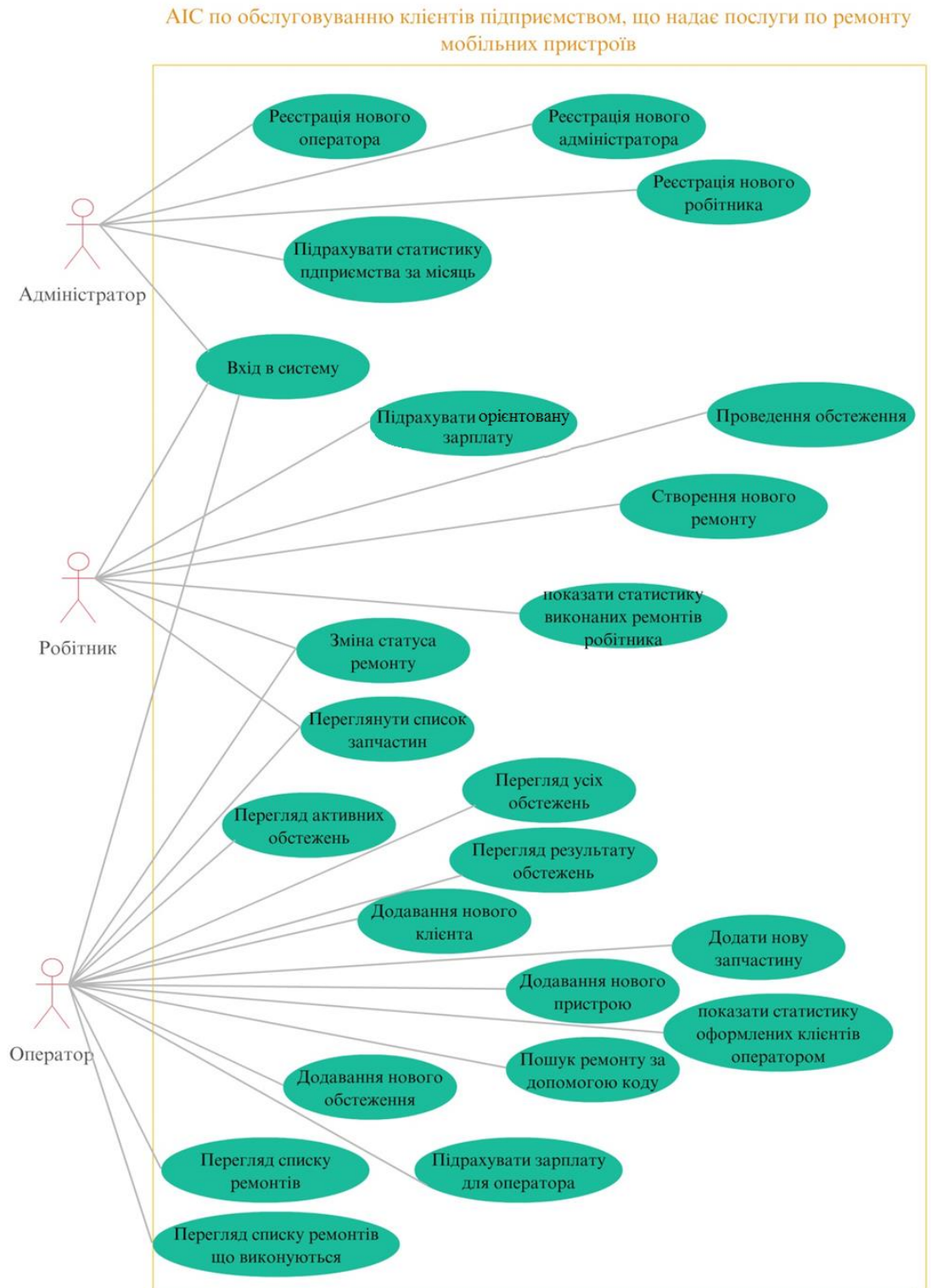


Рис.2.1. Діаграма варіантів використання АІС

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Для підприємства, що займається ремонтом мобільних пристроїв розроблена діаграма варіантів використання зображена на рис. 2.1.

2.3. Схема та структура бази даних

Для створення АІС підприємства, що надає послуги по ремонту мобільних пристроїв була розроблена база даних, схема якої представлена на рис.2.2, за допомогою якої можна досягти поставлених цілей.

База даних складається з 11 таблиць:

- клієнти;
- пристрої;
- частини;
- користувачі;
- виплати;
- персональні дані;
- продажі;
- типи робіт;
- ремонти;
- обстеження;
- пристрої на час ремонту.

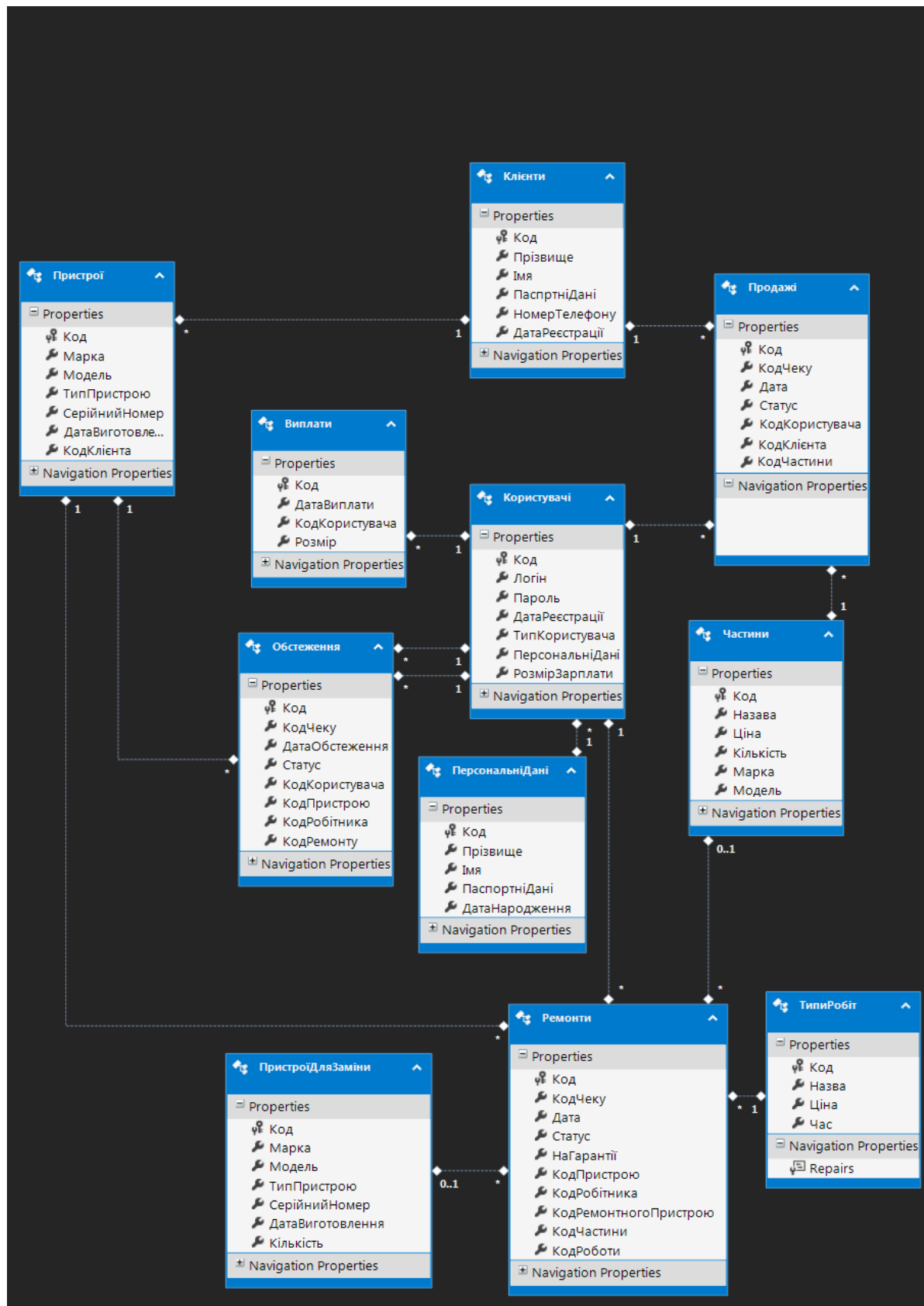


Рис.2.2.Схема зв'язків між таблицями бази даних

Таблиця 2.2

Структура таблиці Клієнти в базі даних

Назва	Тип	Опис
Код	int	Ключове поле
Прізвище	nvarchar(15)	Прізвище клієнта
Ім`я	nvarchar(15)	Ім`я клієнта
Паспортні дані	nvarchar(15)	Серія і код паспорта
Номер телефону	nvarchar(15)	Мобільний номер клієнта
Дата реєстрації	date	Дата реєстрація клієнта

У табл. 2.2 наведено перелік назв полів, їх тип даних та опис їх призначення для таблиці «Клієнти». В даній таблиці зберігається інформація про кожного клієнта, що обслуговується підприємством.

Таблиця 2.3

Структура таблиці Пристрої в базі даних

Назва	Тип	Опис
Код	int	Ключове поле
Марка	nvarchar(50)	Марка виробника
Модель	nvarchar(50)	Модель даного пристрою
Тип пристрою	int	Тип пристрою може бути телефон або планшет
Серійний номер	nvarchar(50)	Унікальний код пристрою
Дата виготовлення	date	Дата виготовлення пристрою
Код клієнта	int	Код власника пристрою

У табл. 2.3 наведено перелік назв полів, їх тип даних та опис їх призначення для таблиці «Пристрої». В даній таблиці міститься інформація про мобільні пристрої, що були на ремонті.

Таблиця 2.4

Структура таблиці Частини в базі даних

Назва	Тип	Опис
Код	int	Ключове поле
Назва частини	nvarchar(500)	Назва запчастини її характеристика
Ціна	float	Ціна частини
Кількість	int	Кількість частин що є в наявності
Марка	nchar(50)	Марка для якої підходить запчастина
Модель	nchar(50)	Модель для якої підходить частина

У табл. 2.4 наведено перелік назв полів, їх тип даних та опис їх призначення для таблиці «Частини». В даній таблиці міститься інформації про частини, що використовуються під час ремонту телефонів або планшетів та частини для продажу.

Таблиця 2.5

Структура таблиці Користувачі в базі даних

Назва	Тип	Опис
Код	int	Ключове поле
Логін	nvarchar(15)	Логін для входу в систему
Пароль	nvarchar(15)	Пароль для входу в систему
Дата реєстрації	date	Дата реєстрація користувача
Тип користувача	nvarchar(15)	Тип користувача
Персональні дані	int	Код персональних даних про користувача
Розмір зарплати	float	Розмір зарплати для користувача

У табл. 2.5 наведено перелік назв полів, їх тип даних та опис їх призначення для таблиці «Користувачі». В даній таблиці міститься інформації про користувачів, що можуть здійснювати вхід в систему, та визначає права та можливості використання певних функцій системи.

Таблиця 2.6

Структура таблиці Виплати в базі даних

Назва	Тип	Опис
Код	int	Ключове поле
Дата	date	Дата здійснення виплати
Розмір	float	Розмір здійсненої виплати
Код користувача	int	Код користувача для якого була здійснена виплата

У табл. 2.6 наведено перелік назв полів, їх тип даних та опис їх призначення для таблиці «Виплати». В даній таблиці міститься інформація про витрати, що підприємство виплачує своїм операторам та робітникам.

Таблиця 2.7

Структура таблиці Персональні дані в базі даних

Назва	Тип	Опис
Код	int	Ключове поле
Прізвище	nvarchar(15)	Прізвище користувача
Ім`я	nvarchar(15)	Ім`я користувача
Паспортні дані	nvarchar(15)	Серія і код паспорта користувача
Дата народження	date	Дата народження користувача

У табл. 2.7 наведено перелік назв полів, їх тип даних та опис їх призначення для таблиці «Персональні дані». В даній таблиці міститься

інформації про персональні дані користувачів, що можуть здійснювати вхід в систему.

Таблиця 2.8

Структура таблиці Типи робіт в базі даних

Назва	Тип	Опис
Код	int	Ключове поле
Назва	nvarchar(15)	Назва роботи
Ціна	float	Ціна виконання роботи
Час	time	Час виконання роботи

У табл. 2.8 наведено перелік назв полів, їх тип даних та опис їх призначення для таблиці «Типи робіт». В даній таблиці міститься інформації про типи робіт, які може проводити підприємство.

Таблиця 2.9

Структура таблиці Ремонти в базі даних

Назва	Тип	Опис
Код	int	Ключове поле
Код Чеку	int	Код чеку ремонту
Дата	date	Дата оформлення ремонту
Статус	int	Статус ремонту
Гарантійний ремонт	bit	Телефон знаходиться на гарантії
Код пристрою	int	Код пристрою над яким відбувається ремонт
Код робітника	int	Код робітника що здійснює ремонт
Код запасного пристрою	int	Код запасного пристрою на час ремонту
Код частини	int	Код частини
Код роботи	int	Код роботи,що здійснюється

У табл. 2.9 наведено перелік назв полів, їх тип даних та опис їх призначення для таблиці «Ремонти». В даній таблиці міститься інформація про ремонти, що проводились та ті що зараз виконуються.

Таблиця 2.10

Структура таблиці Пристрої на час ремонту в базі даних

Назва	Тип	Опис
Код	int	Ключове поле
Марка	nvarchar(50)	Марка виробника
Модель	nvarchar(50)	Модель даного пристрою
Тип пристрою	int	Тип пристрою може бути телефон або планшет
Серійний номер	nvarchar(50)	Унікальний код пристрою
Дата виготовлення	date	Дата виготовлення пристрою
Кількість	int	Кількість штук в наявності

У табл. 2.10 наведено перелік назв полів, їх тип даних та опис їх призначення для таблиці «Пристрої на час ремонту». В даній таблиці міститься інформація про мобільні пристрої, що видаються клієнтам на час ремонту.

Таблиця 2.11

Структура таблиці Продажі в базі даних

Назва	Тип	Опис
Код	int	Ключове поле
Код Чеку	int	Код чеку продажі
Дата	date	Дата оформлення продажу
Статус	int	Статус продажі

Продовження Таблиці 2.11

Код користувача	int	Код користувача, що здійснює продаж
Код частини	int	Код частини для продажу
Кількість частин	int	Кількість частин
Код клієнта	int	Код клієнта

У табл. 2.11 наведено перелік назв полів, їх тип даних та опис їх призначення для таблиці «Продажі». В даній таблиці міститься інформація про продані частини.

Таблиця 2.12

Структура таблиці Обстеження в базі даних

Назва	Тип	Опис
Код	int	Ключове поле
Код Чеку	int	Код чеку обстеження
Дата	date	Дата оформлення обстеження
Статус	int	Статус обстеження
Код оператора	int	Код оператора, що оформляє обстеження
Код пристрою	int	Код пристрою
Код робітника	int	Код робітника, що здійснюватиме обстеження
Код ремонту	int	Код ремонту

У табл. 2.12 наведено перелік назв полів, їх тип даних та опис їх призначення для таблиці «Продажі». В даній таблиці міститься інформація про активні обстеження та ті що були здійснені раніше.

2.4. Діаграми класів автоматизованої інформаційної системи

2.4.1 Опис головного класу для роботи з вікнами

Клас `WindowsFactory` призначений для відкривання нових вікон у програмі. Він визначає методи відкриття вікон. За його допомогою синхронізується інформація у всіх вікнах. На рис. 2.3 представлена UML діаграма класу.

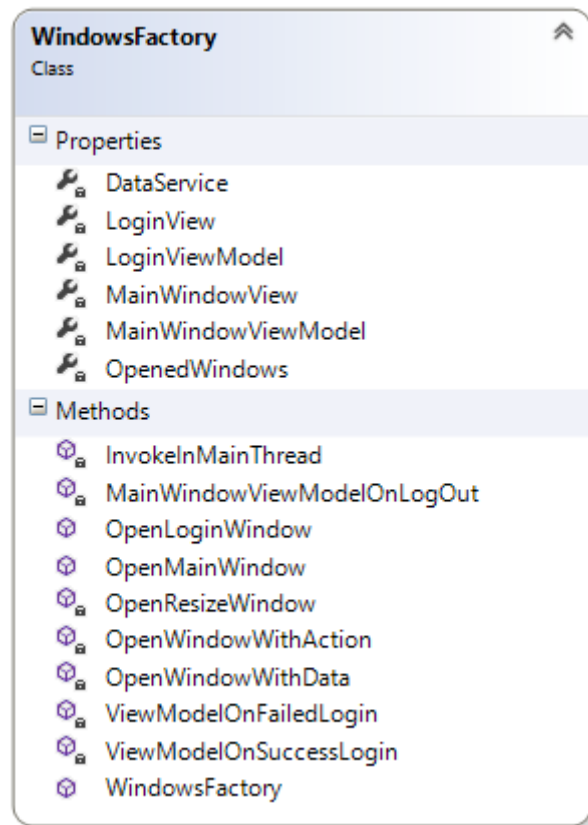


Рис.2.3.UML діаграма класу для роботи з вікнами

Цей клас виконує функцію ядра програми, при запуску програми створюється спочатку екземпляр класу `WindowsFactory` після чого він відкриває вікно входу в систему. Кожне нове вікно додається до списку відкритих вікон за допомогою чого можна дізнатись, яке вікна зараз є відкрити і оперативно закрити усі відкриті вікна.

2.4.2 Опис класу для роботи з базою даних

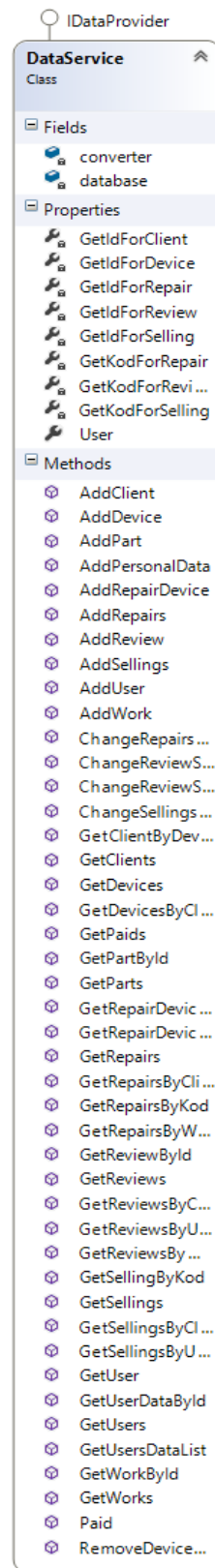


Рис.2.4.UML діаграма класу для роботи з базою даних

Клас DataService призначений для взаємодії з сховищем даних Він реалізує інтерфейс IDataProvider, що забезпечує асинхронну роботу з базою даних. На рис. 2.4 представлена UML діаграма класу.

2.4.3 Опис класу для обчислення даних, що використовуються при побудові графіків

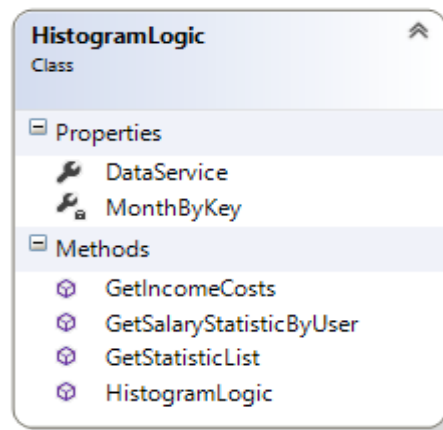


Рис.2.5.UML діаграма класу для обчислення даних, що використовуються при побудові графіків

Клас HistogramLogic призначений для опрацювання даних, що містяться в базі даних і їх використання для побудови гістограм та кругових діаграм. На рис. 2.5 представлена UML діаграма класу.

2.4.4 Опис класу, що забезпечує використання команд

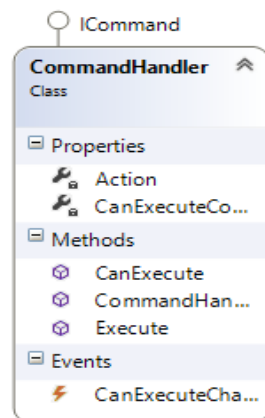


Рис.2.6.UML діаграма класу використання команд

Клас `CommandHandler` призначений для взаємодії користувача системи з самою системою. У технології WPF усі дії з графічним інтерфейсом відбуваються за рахунок інтерфейсу `ICommand`, що і реалізує даний клас. На рис. 2.6 представлена UML діаграма класу.

2.4.5 Опис класу для роботи з головним вікном

Клас `MainWindowViewModel` призначений для роботи з головним вікном програми. Він містить команди, що дозволяють відкривати вікна та доступатися до бази даних. На рис.2.7 представлена UML діаграма класу.

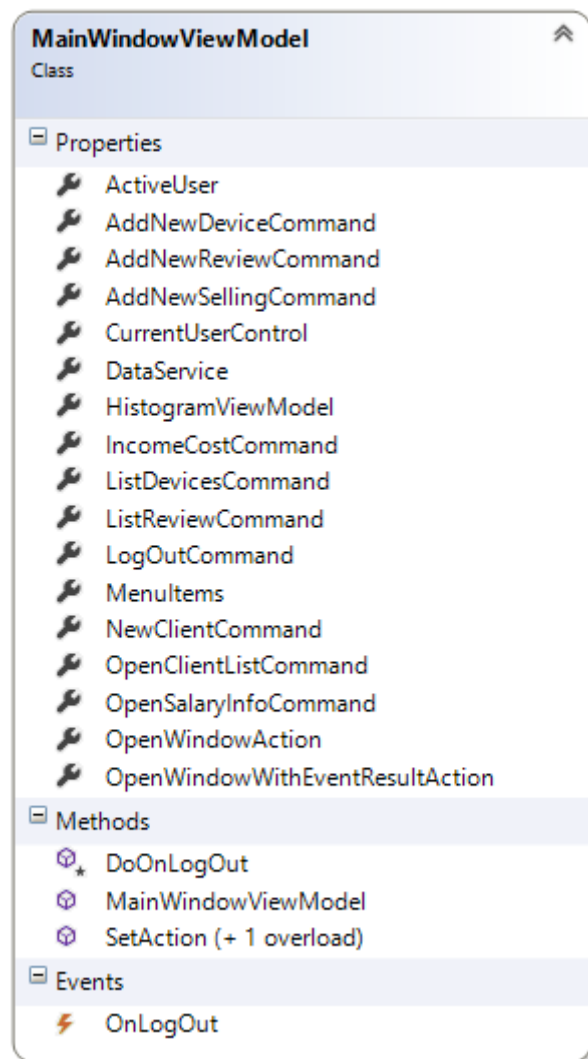


Рис.2.7.UML діаграма класу для роботи з головним вікном

2.4.6 Опис класу для роботи з вікном входу в систему

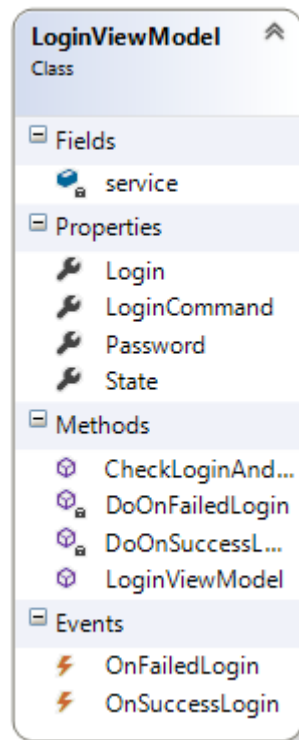


Рис.2.8.UML діаграма класу для роботи з вікном входу в систему

Клас `LoginViewModel` призначений для роботи з вікном входу в систему. Він містить команду, що має доступ до бази даних та перевіряє чи існує такий користувач. У разі знаходження такого користувача спрацьовує подія успішного входу в систему, що відкриває головне вікно програми. В іншому разі спрацьовує подія помилкового входу в систему. На рис. 2.8 представлена UML діаграма класу.

2.5. Проектна реалізація автоматизованої інформаційної системи

Дана автоматизована інформаційна система підтримує три типи користувачів тому для того, щоб ідентифіковано увійти в систему потрібно мати логін і пароль. Ви можете увійти в систему ввівши їх у відповідні текстові поля вікна входу, що зображено на рис. 2.9.

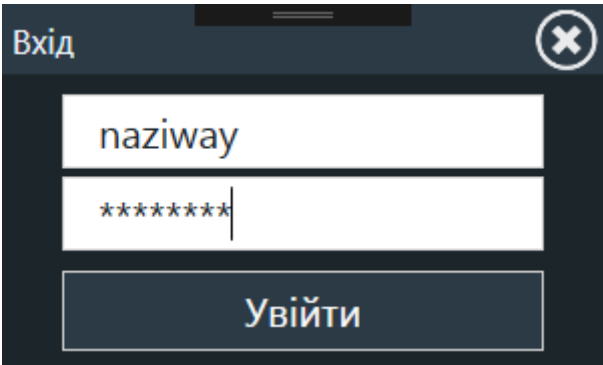


Рис.2.9.Вікно входу в систему

При вході в систему для користувача відкривається головне вікно програми і в залежності від прав доступу йому будуть доступні різні функції системи. На рис.2.10 представлено вигляд головного вікна програми, а на рис.2.11 зображено набори команд для різних типів користувачів.



Рис.2.10.Головне вікно програми

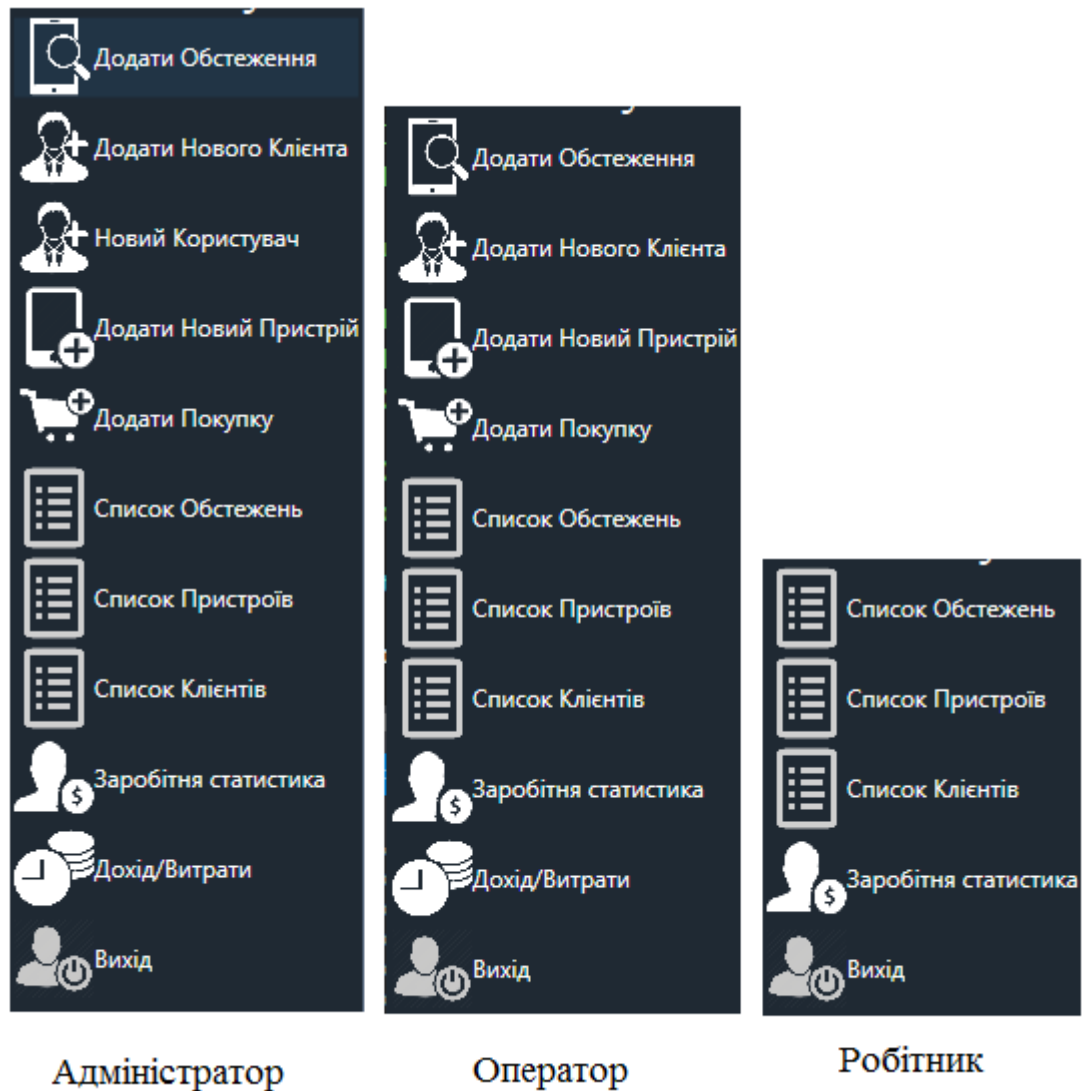


Рис.2.11.Панелі команд користувачів

Головними об'єктами в системі над якими проводяться операції в системі є клієнт і його пристрій. Тому для початку роботи потрібно додати нового клієнта. На панелі команд потрібно натиснути на відповідну піктограму, що зображена на рис.2.12, для відкриття інтерфейсу додавання клієнта

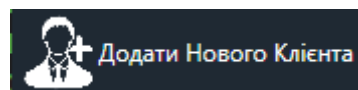


Рис.2.12.Піктограма відкриття вікна для додавання нового клієнта

Після відкриття потрібно коректно заповнити форму та натиснути кнопку «Додати»(див. рис. 2.13)

Після успішного додавання клієнта його можна знайти у списку усіх клієнтів, що зображений на рис. 2.14.

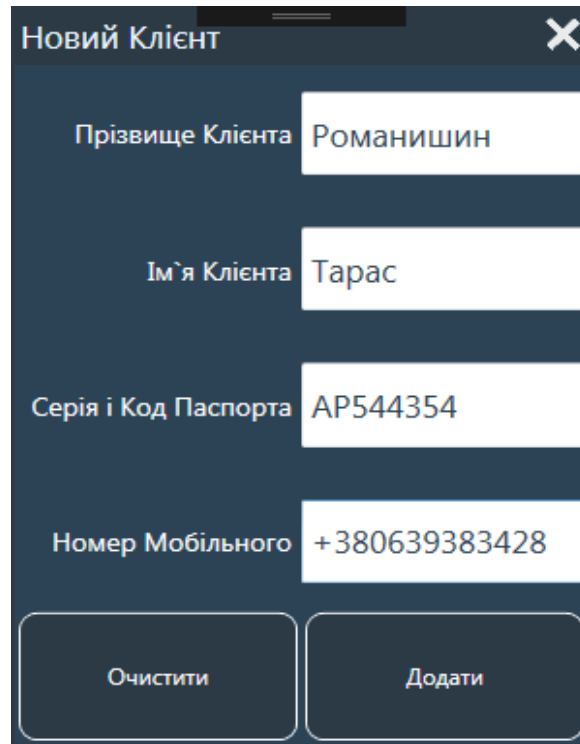
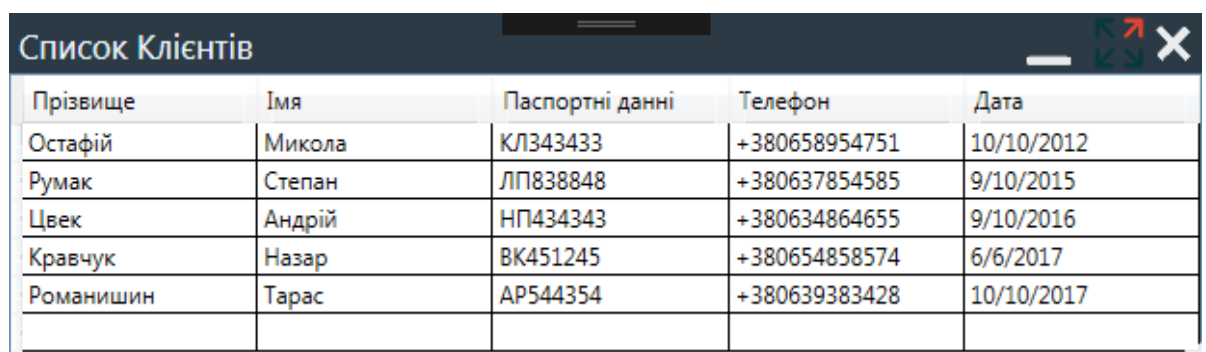


Рис.2.13.Вікно додавання нового клієнта системи



Прізвище	Ім'я	Паспортні данні	Телефон	Дата
Остафій	Микола	КЛ343433	+380658954751	10/10/2012
Румак	Степан	ЛП838848	+380637854585	9/10/2015
Цвек	Андрій	НП434343	+380634864655	9/10/2016
Кравчук	Назар	ВК451245	+380654858574	6/6/2017
Романишин	Тарас	AP544354	+380639383428	10/10/2017

Рис.2.14.Вікно перегляду усіх клієнтів системи

Тепер для клієнта доступна функція купівлі частин у підприємства. Для цього оператору програми потрібно клікнути в панелі команд на піктограму «Додати покупку» (див. рис. 2.15)

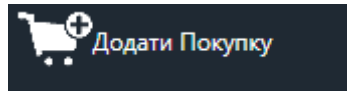


Рис.2.12. Піктограма відкриття вікна для оформлення купівлі

Після чого з'явиться вікно оформлення купівлі, яке зображене на рис. 2.13.

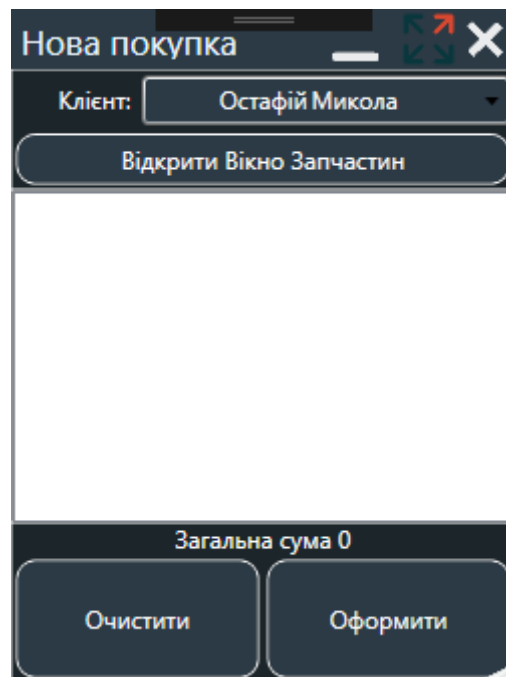


Рис.2.13.Вікно оформлення купівлі

У випадаючий список вікна завантажуються ініціали клієнтів з яких потрібно вибрати одного, для якого буде сформоване замовлення. Також потрібно відкрити вікно частин клікнувши на піктограму, зображену на рис. 2.14.



Рис.2.14.Піктограма відкриття вікон запчастин

Після натиснення з'явиться вікно запчастин(див. рис. 2.15), яке синхронізоване з вікном оформлення покупок. В цьому вікні можна вибрати певні частини та їх кількість при чому дані про вибрані частини автоматично перенесуться у вікні оформлення покупки, де буде підраховано загальну суму купівлі(див. рис. 2.16). Після кліку на кнопку «Оформити» дані про покупку будуть додані в базу даних.

Назва частини	Кількість	Вартість	Дії
Захисне скло	9 шт 3	Сума= 600	+
Акумулятор 2000 mA	3 шт 2	Сума= 1000	+
Динамік	10 шт 1	Сума= 250	+

Рис.2.15.Вікно усіх частин

Назва частини	Кількість	Вартість	Дії
Захисне скло	9 шт 3	Сума= 600	-
Акумулятор 2000 mA	3 шт 2	Сума= 1000	-
Динамік	10 шт 1	Сума= 250	-

Загальна сума 1850

Очистити Оформити

Рис.2.16.Вікно з обраними запчастинами

Для того, щоб провести обстеження або здійснити ремонт потрібно додати новий пристрій. У панелі команд виберіть відповідну піктограму

(див. рис. 2.17), для відкриття вікна додавання нового пристрою (див. рис. 2.18).

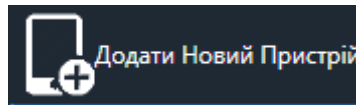


Рис.2.17.Піктограма відкриття вікна додавання нового пристрою

Вікно з темною тематикою з заголовком "Новий Пристрій" та кнопкою закриття (X) у правому верхньому куті. Формуляція містить наступні елементи: поле "Клієнт" з вибором "Остафій Микола"; поле "Марка Пристрою" з текстом "Samsung"; поле "Модель Пристрою" з текстом "Galaxy S5"; поле "Тип Пристрою" з вибором "Телефон"; поле "Серія Телефону" з текстом "5346356356456"; поле "Дата Виготовлення" з текстом "8/30/2015". У нижній частині розташовані дві кнопки: "Очистити" та "Додати".

Рис.2.18.Вікно додавання нового клієнта

Під час заповнення форми нового пристрою для зручності можна використовувати клавішу табуляції. Новий пристрій буде відображений у списку всіх пристроїв(див. рис. 2.19).

Список Пристроїв						
Id	Марка	Модель	Тип	Серійний	Виготовлені	Власник
1	Iphone	5S	Телефон	gfg64564564	10/20/2015	Остафій Микола
2	Iphone	6S	Телефон	tht456564564	10/20/2015	Румак Степан
3	Samsung	S8	Телефон	hyhty7567565	10/20/2015	Цвек Андрій
4	Samsung	5624	Телефон	gjyu56756756	10/20/2015	Остафій Микола
5	Meizu	U10	Телефон	Htyu6575675	10/20/2015	Цвек Андрій
6	Meizu	6S	Телефон	Y6576756755	10/20/2015	Румак Степан
7	Fly	S8	Телефон	Jy876876786	10/20/2015	Цвек Андрій
8	Fly	5624	Телефон	Ku786786767	10/20/2015	Остафій Микола
9	Phone	Phone	Телефон	JFJfGghfdjhgf	10/20/2016	Цвек Андрій
10	Samsung	A16	Телефон	Gf84853945	1/1/0001	Румак Степан
11	Samsung	GR343	Телефон	Gfg454554	1/5/2016	Цвек Андрій
12	Lenovo	K343	Телефон	14565421545	1/1/2015	Остафій Микола
13	Fly	FS504	Телефон	12458158545	1/1/2015	Кравчук Назар
14	Samsung	Galaxy S5	Телефон	53463563564	8/30/2015	Остафій Микола

Рис.2.19.Вікно, що містить список усіх пристроїв

Після того, як пристрій буде додано ви можете оформити для нього обстеження.Для цього у панелі команд потрібно вибрати відповідну піктограму, що відкриває вікно оформлення обстеження, що зображена на рис. 2.20.

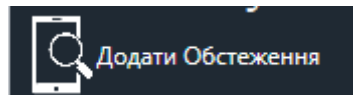


Рис.2.20.Піктограма оформлення обстеження

Для оформлення обстеження у щойно відкритому вікні(див. рис. 2.21) потрібно спочатку вибрати з випадаючого списку клієнта, після чого автоматично завантажиться список пристроїв вибраного клієнта з якого теж потрібно вибрати один для обстеження. І також потрібно вибрати робітника, що буде проводити обстеження

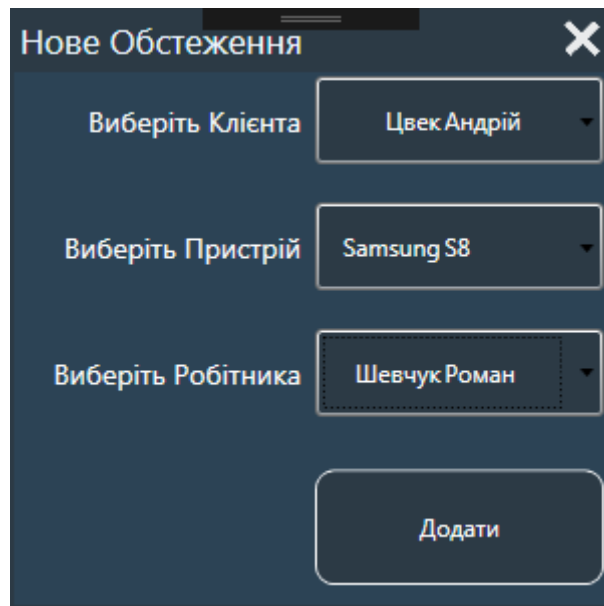


Рис.2.21.Вікно оформлення нового обстеження

Після натиску на кнопку «Додати» обстеження отримає статус «Нове обстеження». Також обстеження буде додано до списку усіх обстежень, яке зображене на рис. 2.22. Для відкриття цього вікна потрібно клікну на піктограму в панелі команд, що зображено на рис. 2.23 Це вікно містить чотири види обстежень, кожне з яких позначається своїм відтінком:

- «Нове обстеження» -червоний;
- «Проводиться обстеження» - жовтий
- «Сформовано список ремонтних робіт» - зелений
- «Обстеження оплачено» - синій

Список Обстежень		
Код замовлення: 382	Дата оформлення: 6/6/2017	Провести обстеження
Код замовлення: 381	Дата оформлення: 6/6/2017	Проводиться обстеження
Код замовлення: 380	Дата оформлення: 6/6/2017	Проводиться обстеження
Код замовлення: 379	Дата оформлення: 6/6/2017	Провести обстеження
Код замовлення: 378	Дата оформлення: 6/6/2017	Проводиться обстеження
Код замовлення: 377	Дата оформлення: 6/6/2017	Проводиться обстеження
Код замовлення: 376	Дата оформлення: 6/6/2017	Провести обстеження
Код замовлення: 375	Дата оформлення: 6/6/2017	Проводиться обстеження
Код замовлення: 374	Дата оформлення: 6/6/2017	Проводиться обстеження
Код замовлення: 373	Дата оформлення: 6/6/2017	Проводиться обстеження
Код замовлення: 372	Дата оформлення: 6/6/2017	Проводиться обстеження
Код замовлення: 371	Дата оформлення: 6/6/2017	Проводиться обстеження
Код замовлення: 370	Дата оформлення: 6/6/2017	Проводиться обстеження
Код замовлення: 369	Дата оформлення: 6/6/2017	Перейти до ремонту
Код замовлення: 368	Дата оформлення: 6/6/2017	Провести обстеження
Код замовлення: 367	Дата оформлення: 6/6/2017	Проводиться обстеження

Рис.2.22.Вікно списку обстежень

Після того як обстеження було успішно додано робітник для якого воно призначене може зайти в список усіх обстежень і клікнути на команду «Провести обстеження»(див. рис. 2.23)

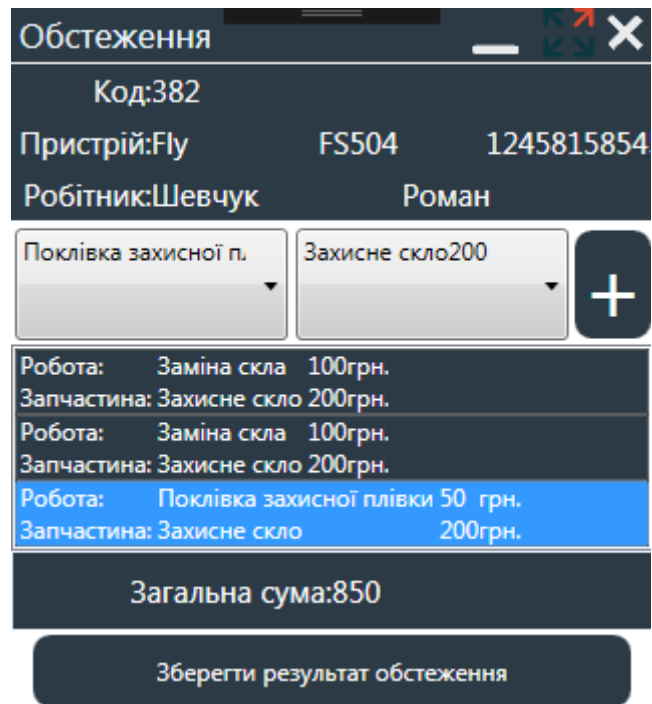
Код замовлення: 382	Дата оформлення: 6/6/2017	Провести обстеження
---------------------	---------------------------	---------------------

Рис.2.23.Обстеження, що має статус «Нове обстеження»

Після натиску на цю команду відкриється вікно проведення обстеження(див. рис. 2.24) відповідно статус обстеження зміниться на «Проводиться Обстеження» та колір елемента списку в вікні обстежень зміниться на жовтий(див. рис. 2.25).

Код замовлення: 382	Дата оформлення: 6/6/2017	Проводиться обстеження
---------------------	---------------------------	------------------------

Рис.2.25.Обстеження, що має статус «Проводиться Обстеження»



Обстеження

Код:382

Пристрій:Fly FS504 1245815854

Робітник:Шевчук Роман

Поклівка захисної пл. Захисне скло200

+

Робота:	Заміна скла	100грн.
Запчастина:	Захисне скло	200грн.
Робота:	Заміна скла	100грн.
Запчастина:	Захисне скло	200грн.
Робота:	Поклівка захисної плівки	50 грн.
Запчастина:	Захисне скло	200грн.

Загальна сума:850

Зберегти результат обстеження

Рис.2.24.Вікно проведення обстеження

У вікні проведення обстежень міститься:

- код замовлення;
- марка пристрою;
- модель пристрою;
- ім'я та прізвище робітника;
- випадаючий список з видами робіт;
- випадаючий список з частинами для пристрою;
- список вибраних робіт та частин
- поле де підраховується загальна сума;
- кнопка для збереження результатів обстеження.

Робітник досліджуючи пристрій, і знаходячи поломки вибирає відповідну роботу та запчастину при необхідності та додає роботу до списку результатів обстеження. Система автоматично підраховує загальну суму

майбутнього ремонту. Після того, як робітник закінчив обстеження він натискає на кнопку «Зберегти результати обстеження». Цією дією робітник змінює статус обстеження на «Сформовано ремонт», колір обстеження стає зелений(див. рис. 2.26), доступною стає команда для оператора «Перейти до ремонту».

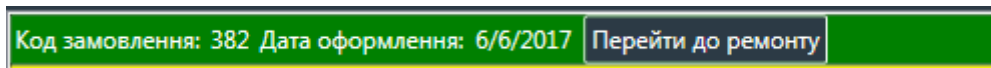


Рис.2.26.Обстеження, що має статус «Сформовано список ремонтних робіт»

Тепер коли обстеження стало зеленого кольору оператор може перейти до списку результатів обстеження натиснувши на кнопку «Перейти до ремонту» й відкривши вікно результатів обстеження(див. рис. 2.27)

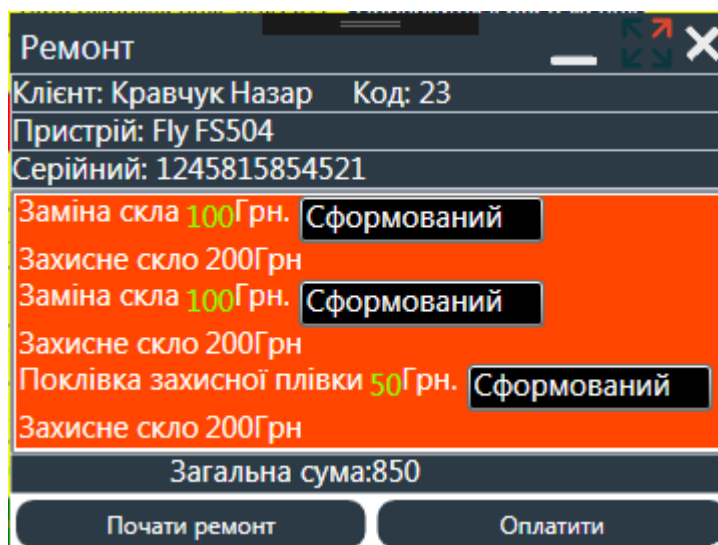


Рис.2.27. Вікно результатів обстеження

У вікні результатів обстеження клієнт може зробити вибір або ремонтувати пристрій або оплатити обстеження. Якщо він вирішує оплатити обстеження то потрібно клікнути на кнопку «Оплатити» і статус обстеження зміниться на «Обстеження оплачено» і колір обстеження в вікні список обстежень стане синім(див. рис. 2.28).

Код замовлення: 382 Дата оформлення: 6/6/2017 [Перейти до ремонту](#)

Рис.2.28.Обстеження, що має статус «Обстеження оплачено»

Якщо ж користувач вирішить здійснювати ремонт то потрібно натиснути на кнопку «Почати ремонт». Ремонт отримає статус «Новий». Під час ремонту робітник може міняти статус кожної виконаної роботи за допомогою випадаючого списку, і зберігати зміни(див. рис. 2.29)

Ремонт
Клієнт: Кравчук Назар Код: 23
Пристрій: Fly FS504
Серійний: 1245815854521
Заміна скла 100Грн. Виконується
Захисне скло 200Грн.
Заміна скла 100Грн. Новий
Захисне скло 200Грн.
Поклівка захисної плівки 50Грн. Виконаний
Захисне скло 200Грн.
Загальна сума:850
Завершити ремонт
Зберегти зміни ремонту

Рис.2.29.Вікно зміни статусів ремонту

При завершенні ремонту статус всіх робіт повинен бути «Виконаний» і тільки тоді клієнт може забрати свій пристрій та оплатити ремонт.

Для кожного користувача системи та підприємства в цілому проводяться статистичні обчислення які відображаються в різних діаграмах.

На головному вікні програми розміщена гістограма (див. рис. 2.30), яка показує прибутки підприємства за останні п`ять місяців з:

- обстежень;
- ремонтів;
- продажів.



Рис.2.30.Гістограма прибутків підприємства за п'ять останніх місяців

Також кожен користувач системи може побачити гістограму своєї заробітної плати. Для цього потрібно клікнути на піктограму, що вказана на рис. 2.31. та відкрити вікно заробітної статистики (див. рис. 2.32).

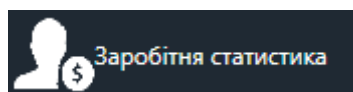


Рис.2.31.Піктограма відкриття вікна статистики заробітної плати

Також можна переглянути глобальну статистику підприємства, яка зображається у вигляді кругової діаграми. Щоб її переглянути потрібно клікнути на піктограму, що зображена на рис. 2.33 , та перейти в вікно витрати прибутки(див. рис. 2.34).

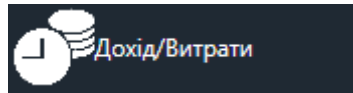


Рис.2.33.Піктограма відкриття вікна доходів та витрат

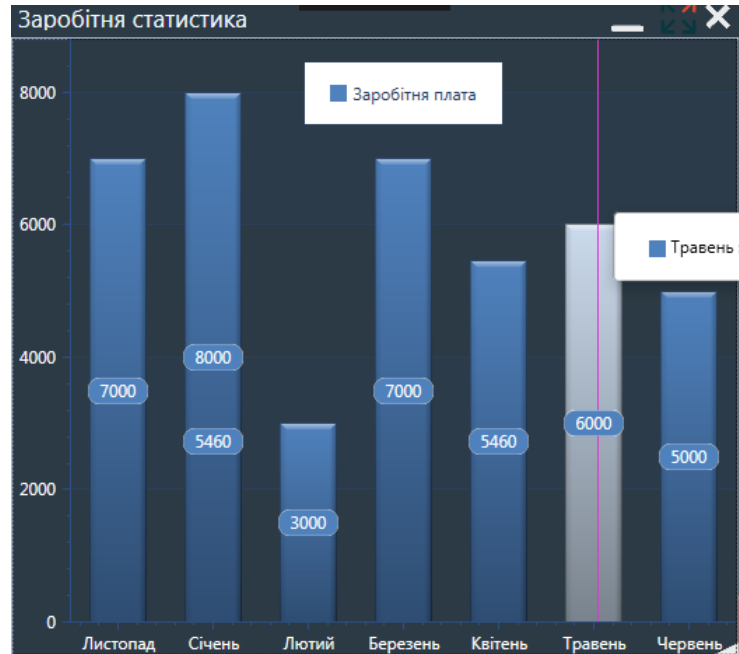


Рис.2.32.Вікно статистики зарплати

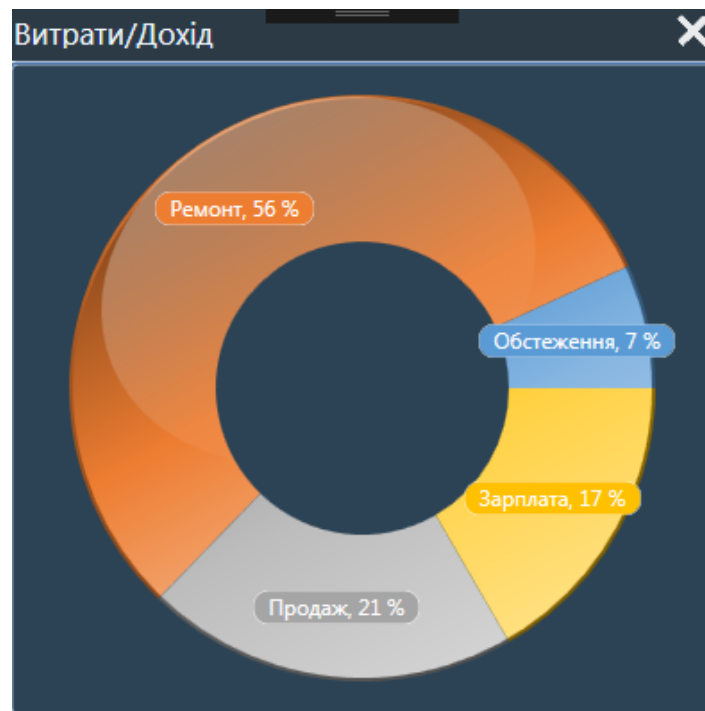
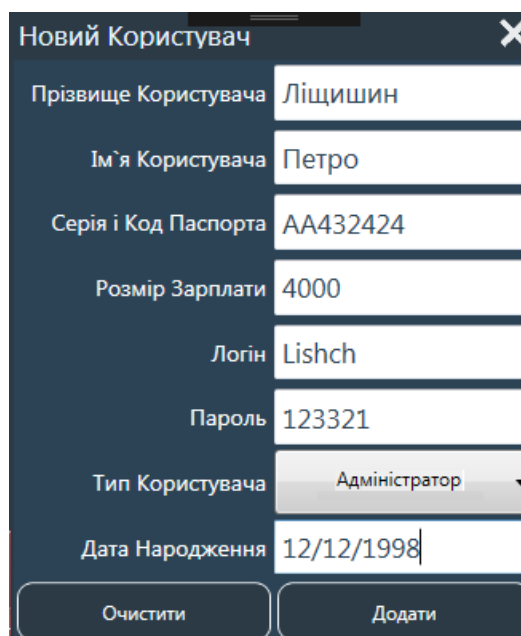


Рис.2.34.Вікно витрат та прибутків

Для адміністратора як для користувача з найбільшими правами доступна функція додавання нових користувачів системи. Для цього потрібно клікнути на відповідну іконку та заповнити форму нового користувача, що зображена на рис. 2.35



Прізвище Користувача	Ліщишин
Ім'я Користувача	Петро
Серія і Код Паспорта	AA432424
Розмір Зарплати	4000
Логін	Lishch
Пароль	123321
Тип Користувача	Адміністратор
Дата Народження	12/12/1998

Очистити Додати

Рис.2.35.Вікно додавання нового користувача

Висновок

У другому розділі дипломної роботи описано вимоги до технічного та програмного забезпечення.

Наведено UML діаграму варіантів використання інформаційної системи та діаграми класів з описом їхнього призначення.

Розроблено і описано структуру бази даних, зображено схему зв'язків між таблицями. Представлено усі можливості ІС. На тестовому прикладі програми продемонстровано роботу ІС.

ВИСНОВКИ

В процесі виконання дипломної роботи було розроблено автоматизовану інформаційну систему для обслуговування клієнтів підприємством, що надає послуги по ремонту мобільних пристроїв.

АІС розроблена і повністю відповідає поставленим вимогам:

- проведення безпомилкових арифметичних розрахунків;
- забезпечення підготовки , заповнення, роздруківки первинних і звітних документів;
- полегшення доступу до бази даних товарів для прийняття рішення, щодо замовлення;
- спрощення та прискорення обробки даних;
- забезпечення звертання до даних і звітів за минулі періоди (вести архів).

Дана тема являється актуальною у наш час. Використання мови програмування C#, технології розробки користувацького інтерфейсу WPF та бібліотеки компонентів DevExpress, виявилось доцільним та раціональним. Завдяки цим інструментам час розробки програми зменшився, а якість та вигляд програмного продукту значно покращилися.

Програма протестована і може успішно використовуватися на підприємстві.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Коноваленко І.В. Програмування мовою С# - Тернопіль, 2016. - 300 с.
2. Береза А.М. Основи створення інформаційних систем. Навчальний посібник. 2-ге видання, 2001. – 156 с.
3. Тарасевич В. М. Економічна теорія , 2006. - 107 с.
4. Кравець П.О. Об'єктно-орієнтоване програмування. – Видавництво Львівської політехніки, 2012. – 200 с.
5. Вовчак І. Інформаційні системи та комп'ютерні технології в менеджменті. - Тернопіль: Карт-бланш, 2001. - 286 с.
6. Гончарук А. Г. Формування механізму управління ефективністю підприємства — Одеса, 2010. — 120 с.
7. Інформаційні системи і технології в обліку й аудиті: Навчальний посібник – 2016. – 124 с.
8. Daniel Solis. Illustrated C# 2012. - 320 с.
9. Шилд, Герберт. Довідник по С#, 2007. – 50 с.
10. Кузнєцов М.С. Об'єктно-орієнтоване програмування з використанням UML та мови С++: Навч. посібник. – Дніпропетровськ: НМетАУ, 2003. – 90 с.
11. Rudolf Pecinovsky. OOP: Learn Object Oriented Thinking and Programming / Rudolf Pecinovsky – Eva & Tomas Bruckner Publishing , 2013. – 527 с.
12. Ситник В.Ф. Основи інформаційних систем, 2001.- 220 с.
13. А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд Язык программирования С#. Классика Computers Science. 4-е издание = C# Programming Language (Covering C# 4.0), 4th Ed. — СПб.: «Питер», 2012. — 784 с.
14. Standard ECMA-334. C# Language Specification. 4th Edition. Ecma International. – June 2006

15. International standart ISO/IEC 23270:2006. Information technology – Programming languages – C#. Second edition. – ISO/IEC. - 2006
16. Matthew MacDonald, Pro WPF in C# 2010: Windows Presentation Foundation in .NET 4, 2010. – 217 с.
17. Адам Натан WPF 4 Подробное руководство, 2012. – 245с.

ДОДАТКИ

Додаток А. Файл DataService.cs

```

using DatabaseService.Extension;
using Model;
using Shared;
using Shared.Enum;
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Threading.Tasks;

namespace DatabaseService
{
    public class DataService : IDataProvider
    {
        MobiDoc database = new MobiDoc();
        public User User { get; set; }
        ConverterToSystemStructure converter = new ConverterToSystemStructure();
        private int GetKodForSelling => database.Sellings.ToList().LastOrDefault()?.Kod + 1 ?? 1;
        private int GetIdForSelling => database.Sellings.ToList().LastOrDefault()?.Id + 1 ?? 1;
        private int GetKodForReview => database.Reviews.ToList().LastOrDefault()?.Kod + 1 ?? 1;
        private int GetIdForReview => database.Reviews.ToList().LastOrDefault()?.Id + 1 ?? 1;
        private int GetKodForRepair => database.Repairs.ToList().LastOrDefault()?.Kod + 1 ?? 1;
        private int GetIdForRepair => database.Repairs.ToList().LastOrDefault()?.Id + 1 ?? 1;
        private int GetIdForClient => database.Clients.ToList().LastOrDefault()?.Id + 1 ?? 1;
        private int GetIdForDevice => database.Devices.ToList().LastOrDefault()?.Id + 1 ?? 1;
        public bool Paid(Paid paid)
        {
            try
            {
                database.Table.Add(paid.Convert());
                database.SaveChangesAsync();
            }
            catch (Exception e)

```

```

    {
        return false;
    }
    return true;
}

public List<Paid> GetPaid()
{
    return database.Table.Select(converter.Convert).ToList();
}

public User GetUser(string login, string password)
{
    User findUser = null;
    try
    {
        findUser = GetUsers().FirstOrDefault(user => user.Login == login && user.Password
== password);    }
    catch (Exception e)
    {
        throw new UserNotFoundException();
    }
    return findUser;
}

public bool AddUser(User user)
{
    try
    {
        database.Users.Add(user.Convert());
        database.SaveChangesAsync();
    }
    catch (Exception e)
    {
        return false;
    }
    return true;
}

public List<Client> GetClients()

```



```

{
    return database.Clients.Select(converter.Convert).ToList();
}

public List<Device> GetDevices()
{
    var clients = GetClients();
    var devices = database.Devices;
    var list = new List<Device>();
    foreach (var device in devices)
    {
        var item = new Device
        {
            Id = device.Id,
            DeviceType = (DeviceType)device.DeviceType,
            ManufactureDate = device.ManufactureDate,
            Marka = device.Marka,
            Model = device.Model,
            SerialNumber = device.SerialNumber,
            Client = clients.First(client => client.Id == device.ClientId)
        };
        list.Add(item);
    }
    return list;
}

public List<RepairDevice> GetRepairDevices()
{
    return database.RepairDevices.Select(converter.Convert).ToList();
}

public List<Part> GetParts()
{
    return database.Parts.Select(converter.Convert).ToList();
}

public List<Work> GetWorks()
{
    return database.Works.Select(converter.Convert).ToList();
}

```

```

public List<User> GetUsers()
{
    var userData = GetUsersDataList();
    var users = database.Users;
    var list = new List<User>();
    foreach (var user in users)
    {
        list.Add(new User
        {
            Id = user.Id,
            Login = user.Login,
            Password = user.Password,
            RegistrationDate = user.RegistrationDate,
            UserType = (UserType)user.UserType,
            UserData = userData.First(data => data.Id == user.PersonalDataId)
        });
    }
    return list;
}

public List<UserData> GetUsersDataList()
{
    return database.PersonalData.Select(converter.Convert).ToList();
}

public List<Device> GetDevicesByClientId(int id)
{
    return GetDevices().Where(devicee => devicee.Client.Id == id).ToList();
}

public bool AddDevice(Device device)
{
    try
    {
        device.Id = GetIdForDevice;
        database.Devices.Add(device.Convert());
        database.SaveChanges();
    }
    catch (Exception e)

```

```

    {
        return false;
    }
    return true;
}

public RepairDevice GetRepairDevicesById(int id)
{
    return GetRepairDevices().First(device => device.Id == id);
}

public bool AddRepairDevice(RepairDevice device)
{
    try
    {
        database.RepairDevices.Add(device.Convert());
        database.SaveChangesAsync();
    }
    catch (Exception e)
    {
        return false;
    }
    return true;
}

public bool AddClient(Client client)
{
    try
    {
        client.Id = GetIdForClient;
        database.Clients.Add(client.Convert());
        database.SaveChangesAsync();
    }
    catch (Exception e)
    {
        return false;
    }
    return true;
}

```

```

    }
    public Client GetClientByDeviceId(int id)
    {
        return GetClients().First(client => client.Id == id);
    }
    public UserData GetUserDataById(int id)
    {
        return GetUsersDataList().First(data => data.Id == id);
    }
    public bool AddPersonalData(UserData userData)
    {
        try
        {
            database.PersonalData.Add(userData.Convert());
            database.SaveChangesAsync();
        }
        catch (Exception e)
        {
            return false;
        }
        return true;
    }
    public Part GetPartById(int id)
    {
        return GetParts().First(part => part.Id == id);
    }
    public bool AddPart(Part part)
    {
        try
        {
            database.Parts.Add(part.Convert());
            database.SaveChangesAsync();
        }
        catch (Exception e)
        {
            return false;
        }
    }

```

```

    }
    return true;
}

public Work GetWorkById(int id)
{
    return GetWorks().First(work => work.Id == id);
}

public bool AddWork(Work work)
{
    try
    {
        database.Works.Add(work.Convert());
        database.SaveChangesAsync();
    }
    catch (Exception e)
    {
        return false;
    }
    return true;
}

public List<Selling> GetSellings()
{
    var users = GetUsers();
    var clients = GetClients();
    var part = GetParts();
    var list = new List<Selling>();

    foreach (var sellingse in database.Sellings)
    {
        var selling = new Selling
        {
            Id = sellingse.Id,
            Part = part.First(p => p.Id == sellingse.PartId),
            User = users.First(p => p.Id == sellingse.UserId),
            Client = clients.First(p => p.Id == sellingse.ClientId),
            Kod = sellingse.Kod,

```

```

        OrderDate = sellingse.OrderDate,
        Status = (SellingStatus)sellingse.Status
    };
    list.Add(selling);
}
return list;
}

public List<Selling> GetSellingByKod(int kod)
{
    return GetSellings().Where(selling => selling.Kod == kod).ToList();
}

public List<Selling> GetSellingsByClientId(int clientId)
{
    return GetSellings().Where(selling => selling.Client.Id == clientId).ToList();
}

public List<Selling> GetSellingsByUserId(int userId)
{
    return GetSellings().Where(selling => selling.User.Id == userId).ToList();
}

public bool RemoveDeviceById(Device device)
{
    throw new NotImplementedException();
}

public List<Review> GetReviews()
{
    var users = GetUsers();
    var devices = GetDevices();
    var list = new List<Review>();

    foreach (var review in database.Reviews)
    {
        var item = new Review
        {
            Id = review.Id,
            Kod = review.Kod,
            OrderDate = review.OrderDate,

```

```

        Status = (ReviewStatus)review.Status,
        Worker = users.First(user => user.Id == review.WorkerId),
        Device = devices.First(devicee => devicee.Id == review.DeviceId),
        User = users.First(userr => userr.Id == review.UserId)
    };
    if (review.RepairKod != null)
        item.Repair = GetRepairs().First(repair => repair.Kod == review.RepairKod);
    list.Add(item);
}
return list;
}

public List<Repair> GetRepairs()
{
    var repairDevices = GetRepairDevices();
    var workers = GetUsers();
    var devices = GetDevices();
    var parts = GetParts();
    var works = GetWorks();
    var list = new List<Repair>();
    foreach (var repair in database.Repairs)
    {
        var item = new Repair
        {
            Id = repair.Id,
            IsWarranty = repair.IsWarranty,
            Kod = repair.Kod,
            OrderDate = repair.OrderDate,
            Status = (RepairStatus)repair.Status,
            Worker = workers.First(user => user.Id == repair.WorkerId),
            Device = devices.First(devicee => devicee.Id == repair.DeviceId),
            Work = works.First(work => work.Id == repair.WorkId)
        };
        if (repair.PartId != null)
            item.Part = parts.First(part => part.Id == repair.PartId);
        if (repair.RepairDeviceId != null)
            item.RepairDevice = repairDevices.First(device => device.Id ==
repair.RepairDeviceId);

```

```

        list.Add(item);
    }
    return list;
}
public List<Review> GetReviewById(int id)
{
    return GetReviews().Where(review => review.Id == id).ToList();
}
public List<Review> GetReviewsByClientId(int clientId)
{
    return GetReviews().Where(review => review.Device.Client.Id == clientId).ToList();
}
public List<Review> GetReviewsByUserId(int userId)
{
    return GetReviews().Where(review => review.User.Id == userId).ToList();
}
public List<Review> GetReviewsByWorkerId(int workerId)
{
    return GetReviews().Where(review => review.Worker.Id == workerId).ToList();
}
public List<Repair> GetRepairsByClientId(int clientId)
{
    return GetRepairs().Where(repair => repair.Device.Client.Id == clientId).ToList();
}
public List<Repair> GetRepairsByKod(int kod)
{
    return GetRepairs().Where(repair => repair.Kod == kod).ToList();
}
public List<Repair> GetRepairsByWorkerId(int workerId)
{
    return GetRepairs().Where(repair => repair.Worker.Id == workerId).ToList();
}
public async Task<int> ChangeSellingsStatusByKod(int kod, SellingStatus newStatus)
{
    await database.Sellings.Where(sellings => sellings.Kod == kod)

```



```

        .ForEachAsync(sellings => sellings.Status = (int)newStatus);
    try
    {
        return await database.SaveChangesAsync();
    }
    catch (Exception)
    {
        return await Task.FromResult(-1);
    }
}

public async Task<int> AddSellings(List<Selling> sellings)
{
    int kod = GetKodForSelling;
    int id = GetIdForSelling;
    try
    {
        foreach (var selling in sellings)
        {
            selling.User = User;
            database.Sellings.Add(selling.Convert(id++, kod));
        }
        var a = await database.SaveChangesAsync();
        return a;
    }
    catch (Exception)
    {
        return await Task.FromResult(-1);
    }
}

public Review AddReview(Review review)
{
    int kod = GetKodForReview;
    int id = GetIdForReview;
    review.User = User;
    try
    {
        var modReview = review.Convert(id, kod);
        database.Reviews.Add(modReview);
    }
}

```

```

        database.SaveChanges();
        review.Kod = modReview.Kod;
        review.Id = modReview.Id;
        return review;
    }
    catch (Exception)
    {
        return null;
    }
}

public int AddRepairs(List<Repair> repairs)
{
    int kod = GetKodForRepair;
    int id = GetIdForRepair;
    try
    {
        foreach (var repair in repairs)
        {
            database.Repairs.Add(repair.Convert(id++, kod));
        }
        database.SaveChanges();
    }
    catch (Exception)
    {
        return -1;
    }
    return kod;
}

public async Task<int> ChangeReviewStatusById(int id, ReviewStatus newStatus)
{
    await database.Reviews.Where(reviews => reviews.Id == id)
        .ForEachAsync(reviews =>
        {
            reviews.Status = (int)newStatus;
        });
    try

```

```

        {
            return await database.SaveChangesAsync();
        }
        catch (Exception)
        {
            return await Task.FromResult(-1);
        }
    }

    public async Task<int> ChangeReviewStatusAndSetRefToRepairByKod(int id,
ReviewStatus newStatus, int kodRepair)
    {
        await database.Reviews.Where(reviews => reviews.Id == id)
            .ForEachAsync(reviews =>
            {
                reviews.Status = (int)newStatus;
                reviews.RepairKod = kodRepair;
            });
        try
        {
            return await database.SaveChangesAsync();
        }
        catch (Exception)
        {
            return await Task.FromResult(-1);
        }
    }

    public async Task<int> ChangeRepairsStatusByKod(int id, RepairStatus newStatus)
    {
        await database.Repairs.Where(repairs => repairs.Id == id)
            .ForEachAsync(repairs =>
            {
                repairs.Status = (int)newStatus;
            });
        try
        {
            return await database.SaveChangesAsync();
        }
        catch (Exception)
        {
            return await Task.FromResult(-1);
        }
    }
}

```

Додаток В. Файл WindowsFactory.cs

```

using DatabaseService;
using GraduateWork.Base;
using Model;
using Shared.Enum;
using System;
using System.Collections.Generic;
using System.Windows;
using UserControls;
using UserControls.AddingControl;
using UserControls.ConvertControl;
using UserControls.InformationControl;
using UserControls.Lists;
using ViewModel;
using ViewModel.Lists;
using ViewModel.ResourceAdd;
namespace GraduateWork
{
    public class WindowsFactory
    {
        private LoginViewModel LoginViewModel { get; set; } = new LoginViewModel();
        private LoginView LoginView { get; set; }
        private MainWindowViewModel MainWindowViewModel { get; set; }
        private ResizeBaseView MainWindowView { get; set; }
        private DataService DataService { get; set; }
        private List<Window> OpenedWindows { get; } = new List<Window>();
        public WindowsFactory()
        {
            DataService = new DataService();
        }
        public void OpenLoginWindow()
        {
            LoginViewModel.OnSuccessLogin += ViewModelOnSuccessLogin;
            LoginViewModel.OnFailedLogin += ViewModelOnFailedLogin;
            LoginView = new LoginView(LoginViewModel);
            InvokeInMainThread(LoginView.Show);    }
    }
}

```

```

public void OpenMainWindow()
{
    MainWindowViewModel = new MainWindowViewModel(DataService);
    MainWindowViewModel.OnLogOut += MainWindowViewModelOnLogOut;
    MainWindowViewModel.SetAction(OpenResizeWindow);
    InvokeInMainThread(() =>
    {
        MainWindowView = new ResizeBaseView(new
MainWindowView(MainWindowViewModel), "Mobi Doc", 600, 900);
        MainWindowView.Show();
    });
}
private void OpenResizeWindow(Shared.Enum.OpenWindow windowType)
{
    Window view = new BaseView();
    switch (windowType)
    {
        case OpenWindow.NewSelling:
            var sellingViewModel = new SellingCreatorViewModel(DataService);
            sellingViewModel.SetAction(OpenWindowWithAction);
            var sellingView = new SellingCreatorUserControl
            {
                DataContext = sellingViewModel,
                RemoveBoxItemCommand = sellingViewModel.RemoveBoxItemCommand
            };
            view = new ResizeBaseView(sellingView, "Нова покупка", 600, 380);
            break;
        case OpenWindow.ListClient:
            var clientViewModel = new ClientListViewModel(DataService);
            view = new ResizeBaseView(new ClientListUserControl { DataContext =
clientViewModel }, "Список Клієнтів", 400, 500);
            break;
        case OpenWindow.SalaryInfo:
            var salaryViewModel = new SalaryViewModel(DataService);
            view = new ResizeBaseView(new SalaryUserControl { DataContext =
salaryViewModel }, "Заробітня статистика", 500, 300);

```

```

        break;
    case OpenWindow.ListReview:
        var reviewsViewModel = new ReviewsViewModel(DataService);
        reviewsViewModel.SetAction(OpenWindowWithData);
        view = new ResizeBaseView(new ReviewListUserControl { DataContext =
reviewsViewModel }, "Список Обстежень", 500, 500);
        break;
    case OpenWindow.ListDevices:
        var deviceListViewModel = new DeviceListViewModel(DataService);
        view = new ResizeBaseView(new DeviceListUserControl { DataContext =
deviceListViewModel }, "Список Пристроїв", 500, 500);
        break;
    case OpenWindow.NewReview:
        var viewModel = new NewReviewControl { DataContext = new
NewReviewViewModel(DataService) };
        view = new BaseView(viewModel, "Нове Обстеження", 300, 300);
        break;
    case OpenWindow.SummaryItstatistics:
        var incomeCostUserControl = new IncomeCostsUserControl { DataContext = new
IncomeCostsViewModel(DataService) };
        view = new BaseView(incomeCostUserControl, "Витрати/Дохід", 400, 400);
        break;
    case OpenWindow.NewClient:
        var clientUserControl = new NewClientUserControl { DataContext = new
NewClientViewModel(DataService) };
        view = new BaseView(clientUserControl, "Новий Клієнт", 400, 300);
        break;
    case OpenWindow.NewUser:
        var userControl = new NewUserUserControl { DataContext = new
NewUserViewModel(DataService) };
        view = new BaseView(userControl, "Новий Користувач", 400, 330);
        break;
    case OpenWindow.NewDevice:
        var deviceUserControl = new NewDeviceUserControl { DataContext = new
NewDeviceViewModel(DataService) };
        view = new BaseView(deviceUserControl, "Новий Пристрій", 400, 300);

```

```

        break;
        default: throw new InvalidOperationException();
    }
    OpenedWindows.Add(view);
    InvokeInMainThread(view.Show);
}
private void OpenWindowWithData(Shared.Enum.OpenWindow windowType, object data)
{
    Window view;
    switch (windowType)
    {
        case OpenWindow.ReviewToOrder:
            view = new ResizeBaseView(new ReviewToOrderView { DataContext = new
ReviewToOrderViewModel(DataService, data as Review) }, "Обстеження", 500, 300);
            break;
        case OpenWindow.RepairInfo:
            view = new ResizeBaseView(new RepairUserControl { DataContext = new
RepairViewModel(DataService, data as List<Repair>) }, "Ремонт", 500, 300);
            break;
        default: throw new InvalidOperationException();
    }
    OpenedWindows.Add(view);
    InvokeInMainThread(view.Show);
}
private void OpenWindowWithAction(OpenWindow windowType, Action<object> action)
{
    Window view;
    switch (windowType)
    {
        case OpenWindow.ListParts:

            var listPartViewModel = new ListPartViewModel(DataService);
            var listPart = new ListPart
            {
                DataContext = listPartViewModel,
                Command = listPartViewModel.Command
            }

```

```

        };
        listPartViewModel.SetAction(action);
        view = new ResizeBaseView(listPart, "Список Запчастин", 500, 500);
        break;
        default: throw new InvalidOperationException();
    }
    OpenedWindows.Add(view);

    InvokeInMainThread(view.Show);
}
private void InvokeInMainThread(Action action)
{
    Application.Current.Dispatcher.Invoke(action);
}
private void MainWindowViewModelOnLogOut(object sender, EventArgs e)
{
    OpenLoginWindow();
    InvokeInMainThread(MainWindowView.Close);
}
#region Process Response Login
private void ViewModelOnSuccessLogin(object sender, User user)
{
    LoginViewModel.OnSuccessLogin -= ViewModelOnSuccessLogin;
    LoginViewModel.OnFailedLogin -= ViewModelOnFailedLogin;
    DataService.User = user;
    OpenMainWindow();
    InvokeInMainThread(LoginView.Close);
}
private void ViewModelOnFailedLogin(object sender, EventArgs e)
{
    MessageBox.Show($"Failed Login Or Password");
}
#endregion
}
}

```