Київський національний торговельно-економічний університет
Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка програмного забезпечення логістичного підприємства “Нова пошта”»

Студента 4 курсу, 6 групи,
спеціальності 121 «Інженерія програмного забезпечення» _____ Жигуліна Івана Сергійовича
підпис студента

Гарант освітньої програми
кандидат технічних наук,
доцент _____ Цензура Микола Олександрович
підпис керівника

Гарант освітньої програми
кандидат технічних наук,
доцент _____ Цензура Микола Олександрович
підпис керівника

КИЇВ – 2021

Київський національний торговельно-економічний університет

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення та кібербезпеки

Освітній ступінь _____

Спеціальність _____

Затверджую
Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О.В.
"20" жовтня 2020 р.

Завдання
на випускна кваліфікаційна робота студентові

Жигулін Іван Сергійович

1. Тема випускної кваліфікаційної роботи «Розробка програмного забезпечення логістичного підприємства "Нова Пошта"»

Затверджена наказом ректора від " 30 " жовтня 2020 р. № 3225

2. Строк здачі студентом закінченої роботи _____

3. Цільова установка та вихідні дані до роботи

Мета роботи проекту є створення системи управління складським обліком логістичного підприємства «Нова пошта»

Об'єкт дослідженняь розробка програм скадського типу

Предмет дослідження є данні підприємства про посилки, які зберігаються на складі.

4. Консультанти роботи із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Загальна характерістіка діяльності ТОВ «Нова Пошта»

1.2. Постановка задачі

1.3. Аналіз існуючих способів рішення задачі

1.3.1. Сутність бізнес-процесів та їх роль у діяльності підприємства

1.4. Висновки до розділу 1

РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Етапи проектування

2.2. Описання моделі програмної системи в UML-нотації

2.3. Описання діаграми класів

2.4. Описання діаграми діяльності

2.5. Описання діаграми послідовності

2.6. Створення системи складського обліку "Нова пошта"

2.6.1.Характеристика мови програмування

2.7.Висновки до розділу 2

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1.Графічний інтерфейс розробленого додатку

3.2.Архітектура розробленого додатку та інструкція користувача

3.3.Висновки до розділу 3

ВИСНОВКИ ТА ПРОПОЗИЦІ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Додатки

6. Календарний план виконання роботи

№ пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускної кваліфікаційної роботи*	21.09.2020	21.09.2020
2.	*Вступ та перелік літературних джерел*	14.12.2020	14.12.2020
3.	*Розділ 1. «Аналіз предметної області»*	19.02.2021	19.02.2021
4.	*Розділ 2. «Технологічний розділ розробки програмного забезпечення»*	05.03.2021	05.03.2021
5.	*Розділ 3. «Розробка та дослідження методів створення інтерфейсу на прикладі асу складського обліку "Нова пошта»*	10.04.2021	10.04.2021
6.	*Висновки*	24.04.2021	24.04.2021
7.	*Здача випускної кваліфікаційної роботи на кафедру (перша перевірка)*	14.05.2021	14.05.2021
8.	*Підготовка автореферату та презентації доповіді*	17.05.2021	17.05.2021
9.	*Попередній захист випускної кваліфікаційної роботи*	18.05.2021-21.05.2021	18.05.2021-21.05.2021
10.	*Зовнішнє рецензування випускної кваліфікаційної роботи*	02.06.2021	02.06.2021
11.	*Здача прошитого випускної кваліфікаційної роботи на кафедру*	02.06.2021	02.06.2021
12.	*Публічний захист випускної кваліфікаційної роботи*		

7. Дата видачі завдання «___»_____20__ р.

8. Науковий керівник випускної кваліфікаційної роботи
Цензура М.О._____
(прізвище, ініціали, підпис)

9. Гарант освітньої програми Цензура М.О._____
(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент Жигулін І.С._____
(прізвище, ініціали, підпис)

11. Відгук керівника випускної кваліфікаційної роботи

Науковий керівник випускної кваліфікаційної роботи _____
(підпис, дата)

Відмітка про попередній захист _____
(ПІБ, підпис, дата)

12. Висновок про випускну кваліфікаційну робота

Випускна кваліфікаційна робота студента Жигуліна І.С._____
(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми Цензура М.О_____
(прізвище, ініціали, підпис)

Завідувач кафедри Криворучко О.В._____
(підпис, прізвище, ініціали)

«_____»_____20 ___ р.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

БД – База Даних.

ОС – операційна система.

ПЗ – програмне забезпечення.

ПК – персональний комп’ютер.

ТЗ – технічне завдання.

.NET Framework - new enabled technology framework.

GUI - Graphical user interface.

HER - Electronic Health Records.

WCF - Windows Communication Foundation.

WPF - Windows Presentation Foundation.

Застосовувані сегодня методи розробки проектів найчастіше НЕ зважають на необходимость розробки інтерфейсу. Цей недолік может буті наслідком того, что фахівці з розробки інтерфейсів залучаються до проекту Занадто Пізно, коли возможности Поліпшення якості взаємодії между користувачем и продуктом здебільшого Вже загублені. Інтерфейсом зручніше за все займаться самє на початкових стадіях розробки. І если фахівці з інтерфейсів залучаються Вже после того, як програмне забезпечення спроектоване й візначені его інструменти або коли розробка програми Вже почти завершена, то їхні рекомендації могут зажадаті переробки всієї віконаної роботи, что, природно, є неспрійнятнім. Коли бюджет проекту Вже вічерпаній, план почти завершень, перспектива відмові від більшої части або даже Усього дизайну й готового коду, Звичайно, що не может віклікаті ентузіазму в менеджеров проекту.

Актуальність даної тими Полягає в тому, что дана розробка предполагает Різні методи для Спрощення роботи користувача за помощью інтерфейсу складської програми. Передбачення проектування інтерфейсу, что супроводжує загальне проектування системи й підводіть до моделі самого інтерфейсу.

Мета дослідження віпускної кваліфікаційної є розробка моделей, Які дозволяти проаналізуваті, оціністі та Виконати покращення інтерфейсу програмного забезпечення для складського обліку логістичного підприємства «Нова пошта».

Об'єктом дослідження є данні підприємства про посилки, Які зберігаються на складі та об'єм роботи якові віконують працівники на складі.

Предметом дослідження є розробка системи, яка б дозволила автоматізуваті будь-яке підприємство, вірніше складський облік, та дати змогу управляти інтерфейсом програми.

1.Дослідити методи створення (покращення, оптимізації) інтерфейсу.

2. Розробити імітаційну модель роботи інтерфейсу на прикладі АСУ складського обліку «Нова пошта»

3. Обгрунтувати оптимізацію інтерфейсу.

Методи дослідження

Базуються на розробках, як вітчизняних так і зарубіжних вчених в області систем проектування інтерфейсу програм складського типу. Приводяться завдання, для яких звичайно використаються елементи керування, так само приводяться ситуації, під час яких користувач взаємодіє із цими елементами керування.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Загальна характеристика діяльності ТОВ «Нова Пошта»

Заснована у 2001 році «Нова Пошта» - українська компанія, що забезпечує сервіс експрес-доставки документів, вантажів і посилок для фізичних та юридичних осіб.

Група «Нова Пошта» пропонує своїм клієнтам - як юридичним особам, так і приватним особам - повний спектр логістичних і поштових послуг. У Групу входять українські та закордонні підприємства, які схематично представлені на рис. 1.1, зокрема «Нова Пошта», «НП Логістик», «ПОСТ ФІНАНС» і «Нова Пошта Ітернешнл».

«Нова Пошта» - лідер логістичного ринку експрес-перевезень, що забезпечує просту доставку кожному клієнту – у відділення, поштомати, на адресу – і дозволяє тисячам підприємців створювати і розвивати бізнес не лише в Україні, але й за кордоном.

Мережа підприємства нараховує понад 2671 відділень по всій території України, а кількість відправлень перевищила 146 млн., за 2018 рік.

«НП Логістик» - компанія, що надає послуги фулфілменту: зберігання товару на складах, комплектацію та відправку замовлень отримувачу.

«ПОСТ ФІНАНС» - завдяки небанківській фінансовій установи, клієнти компанії здійснювати грошові перекази та операції з електронними грошима.

«Нова Пошта Інтернешнл» розвиває та виходить на міжнародну мережу, щоб надавати клієнтам послуги експрес-доставки не лише в Україні, але й за кордоном [10].

Група «Нова Пошта» працює із дотриманням усіх норм українського законодавства. У сукупності, Група перерахувала в бюджет країни близько 1,5 млрд грн податків і зборів за останні роки.

Рис. 1.1 Структура Групи «Нова Пошта»

Загальний штат працівників компанії перевищує 30 000 осіб.

Місія підприємства – спрощувати життя своїм клієнтам, робивши доставку легкою для життя і бізнесу. Для цього команда «Нова Пошта» впроваджує та удосконалює нові продукти і послуги, орієнтуючись на світові стандарти та кращий міжнародний досвід.

Окрім відправки та отримання посилок та вантажів, у відділеннях «Нова Пошта» можна замовити, додаткові послуги, що розроблені з урахуванням побажань клієнтів.

На рис. 1.2 зображено кількість відділень розташованих у всіх куточках України.

Отже, як показує історія динамічного розвитку, вже понад 18 років компанія пропонує своїм клієнтам зручну, доступну та якісну послугу – доставку вантажів і кореспонденції в будь-яку точку України.

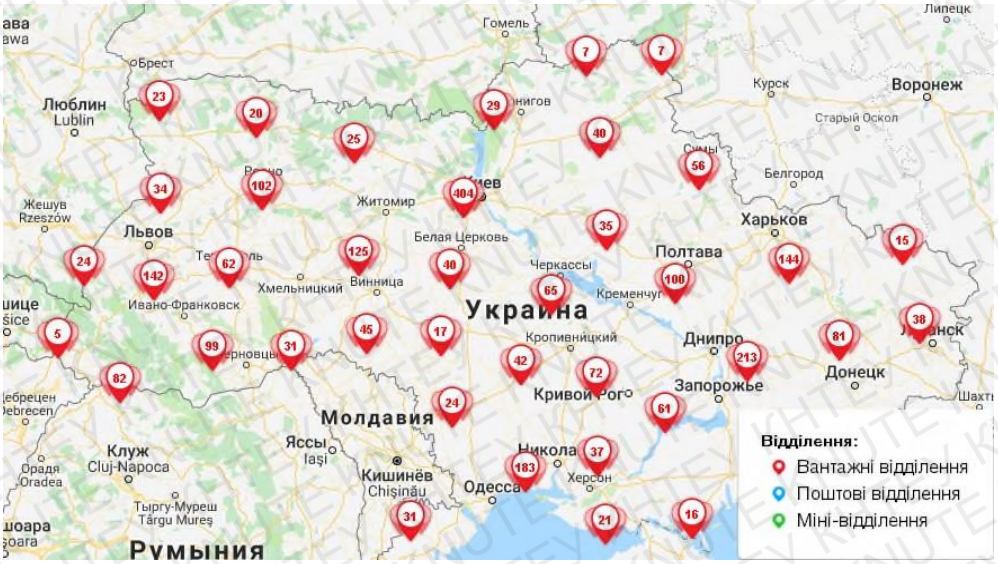

Рис. 1.2 Георгафія розташування відділень ТОВ «Нова Пошта»

Організаційна структура ТОВ «Нова Пошта» дуже разголужена, як це видно з рис. 1.3. Виходячи із даної організаційної структури підприємства ТОВ

«Нова Пошта», можно побачити, що кожен підрозділ самостійний, але безпосередньо взаємно пов'язаний з іншими підрозділами організаційної системи підприємства. Результати роботи будь-якого підрозділу апарату управління оцінюються показниками, що характеризують реалізацію ними своїх цілей і завдань. По кожній підсистемі формуються «ієрархія» послідовності, а також правил роботи, охоплюючи всю організацію від верху до низу.

Далі представимо характеристику кожного з відділів компанії [103]. Фінансово-бухгалтерський відділ в компанії ТОВ «Нова Пошта» працює з різними категоріями і групами клієнтів, розробляючи та втілюючи фінансові системи та інструменти, які забезпечують прибутковість компанії. Їх вплив є визначним при виборі стратегічного напрямку, розробки портфоліо чи планів розширення. Бухгалтерський відділ забезпечує досягнення бізнес-результатів компанії, при веденні точних та своєчасних облікових записів згідно вимог законодавства. На основі даних бухгалтерського, податкового та управлінського обліку, компанія щодня приймає рішення щодо найвигідніших інвестицій, їх окупності та подальших планів.

Департамент логістики. Даний відділ підпорядковує напрям, який найбільш впливає на ефективність роботи ТОВ «Нова Пошта». Оскільки в їх обов'язках є контроль міжміської логістики, міської логістики та термінальної логістики.

ІТ-відділ. Без ІТ – відділу неможливе ефективне управління сучасними інформаційними системами. За його допомогою забезпечується безперервна робота ІТ – інфраструктури і таким чином активне функціонування всього бізнесу.

Основні функції ІТ-відділу:

- дослідження і відстеження новітніх інформаційних технологій;
- забезпечення безперебійної роботи устаткування і користувачів;
- проектування, розробка, впровадження та супровід програмних продуктів;
- регулярне відстеження ситуації в бізнесі для задоволення інформаційних потреб компанії;
- формування потреб в капіталовкладеннях в технологічну інфраструктуру;
- підтримання високого рівня освіченості співробітників в сфері інформаційних технологій

1.2. Постановка задачі

Темою даного дипломного проекту є «Розробка програмного забезпечення логістичного підприємства "Нова Пошта"».

Дана розробка повинна передбачити різні методи для спрощення роботи користувача за допомогою користувальницького інтерфейсу. Також має бути виконано проектування інтерфейсу, що супроводжує загальне проектування системи й підводить до моделі самого інтерфейсу. Необхідно створити діаграми для кращого розуміння предметної області й спрощення проектування всього додатка.

Завдання дослідження :

1. Дослідити методи створення (покращення, оптимізації) інтерфейсу.

2. Розробити імітаційну модель роботи інтерфейсу на прикладі АСУ складського обліку "Нова пошта"

3. Обґрунтувати оптимізацію інтерфейсу.

Відділ складу використовує великі обсяги інформації, формує велику кількість актів та звітності, постійно працює з численними показниками, які потрібно своєчасно та точно підраховувати та аналізувати.

Створення інформаційно-автоматизованої системи управління забезпечить оперативне отримання повної і достовірної інформації та показників споживання ресурсів кожного виду, формуванню планів та звітності по заводу та усіх субспоживачах, пов’язаних між собою, що дасть умови для оптимальної організації управління відділом.

Задачі організаційного управління характеризуються великою складністю, комплексністю й у повному обсязі не можуть вирішуватись ізольовано. Тому, в умовах функціонування всієї автоматизованої системи управління з’являється потреба розробляти підсистему відділу, забезпечивши виконання обов’язкових умов: повноти, своєчасності і достовірної інформації.

1.3. Аналіз існуючих способів рішення задачі

1.3.1. Сутність бізнес-процесів та їх роль у діяльності підприємства

«Нова пошта» – потужне підприємство, види діяльності якого – виготовлення обладнання упаковок, відправка та зберігання посилок, товарів, конструювання обладнання, технологічна підтримка, монтаж та налаштування обладнання. Організація і розподіл обов'язків на «Нова пошта» здійснюється згідно схеми організаційної структури управління.

Стратегічним завданням «Нова пошта» є пошук шляхів подальшої співпраці з іноземними фірмами, розширення сфери співробітництва, заохочення іноземних інвестицій шляхом створення спільних підприємств з іноземними інвестиціями, розширення ринків збуту в Європи.

Підприємство працює надійно і стабільно, має впевненість у завтрашньому дні.

Фактори зручності використання й принципи створення зручного ПЗ.

Одним з важливих показників якості програмного забезпечення є зручність його використання. Воно описується за допомогою таких характеристик, як зрозумілість інтерфейсу користувача, легкість навчання роботі з ним, трудомісткість вирішення певних завдань з його допомогою, продуктивність роботи користувача з ПО, частота появи помилок і скарг на незручності. Для побудови дійсно зручних програм необхідний облік контексту їхнього використання, психології користувачів, необхідності допомагати починаючим користувачам і надавати все потрібне для роботи досвідчених.

Однак самим значимим фактором є те, чи допомагає дана програма вирішувати дійсно значимих для користувачів завдань

Багато програмістів мають технічний або математичний склад розуму. Для таких людей «зрозумілість», «легкість навчання» представляються досить суб'єктивними факторами.

Самі вони досить легко сприймають складні речі, якщо ті представлені в рамках несуперечливої системи понять, як би дико ця система й вхідні в неї поняття не виглядали для стороннього спостерігача.

Вони найчастіше самі вивчають нове ПО за допомогою документації й щиро переконані в тім, що користувачі будуть так само розбиратися з написаної ними програмою. Типовий підхід програміста при розробці інтерфейсу користувача - надати користувачеві ті ж важелі й кнопки, за допомогою яких програмісти самі хотіли б управляти своєю програмою. Основні фактори, за допомогою яких можна оцінити або навіть виміряти зручність використання програми.

• **Адекватність інтерфейсу**. Адекватність користувальницького інтерфейсу програми - це його відповідність тим завданням, які користувачі повинні й хотіли б вирішувати з її допомогою. Ця відповідність має два аспекти: по-перше, всі потрібні користувачам завдання повинні бути розв'язні (якщо це не так - у програми більші проблеми), по-друге, ті дії, які користувачі виконують частіше, повинні вимагати менше зусиль.

• **Продуктивність роботи користувачів.** Це кількість однотипних реальних завдань, які користувач може вирішити за допомогою ПО за одиницю часу.

• **Ефективність запобігання й подолання помилок користувачів.** Цей показник тим краще, ніж рідше користувачі помиляються при роботі з даним інтерфейсом, і чим менше часу й зусиль потрібно для подолання наслідків уже зроблених помилок.

• **Правило доступності.** Система повинна бути настільки зрозумілої, щоб користувач, ніколи раніше що не бачив її, але добре розбирається в предметній області, міг без усякого навчання почати неї використати. Це правило служить деяким ідеалом, до якого треба прагнути, оскільки на практиці досягти такого ступеня зрозумілості майже ніколи не вдається. Проте, усе, що можна зробити для досягнення цього ідеалу, робити потрібно.

• **Правило ефективності.** Система не повинна перешкоджати ефективній роботі досвідчених користувачів, що працюють із нею довгий час.

• **Правило безперервного розвитку.** Система повинна сприяти безперервному росту знань, умінь і навичок користувача й пристосовуватися до його мінливого досвіду

• **Правило підтримки.** Система повинна сприяти більше простому й швидкому рішенню завдань користувача. Це означає, насамперед, що система повинна дійсно вирішувати завдання користувача. Крім того, вона повинна вирішувати їх краще, простіше й швидше, ніж інструменти, що були до її появи, і методи.

• **Принцип структуризації.** Користувальницький інтерфейс повинен бути доцільно структурований. Зв'язані за змістом, родинні його частини повинні бути зв'язані видимим образом, а незалежні - розділені; схожі елементи повинні виглядати схоже, а несхожі - розрізнятися.

• **Принцип простоти.** Найпоширеніші операції повинні виконуватися максимально просто. При цьому повинні бути видимі посилання на більше складні процедури.

• **Принцип видимості.** Всі функції й дані, необхідні для рішення певного завдання, повинні бути видні, коли користувач намагається неї вирішити.

• **Принцип толерантності**. Інтерфейс повинен бути гнучким і терпимим до помилок користувача.

• **Принцип повторного використання**. Варто намагатися використати багаторазово внутрішні й зовнішні компоненти, забезпечуючи тим самим уніфікованість інтерфейсу й подібність між схожими його елементами.

1.4. Висновки до розділу 1.

В даному розділі проведено аналіз логістичного підприємства «Нова пошта», розглянуто принципі діяльності підприємства в цілому, та організацію складського обліку. Одним з важливих показників якості програмного забезпечення є зручність його використання. Воно описується за допомогою таких характеристик, як зрозумілість інтерфейсу користувача, легкість навчання роботі з ним, трудомісткість вирішення певних завдань з його допомогою, продуктивність роботи користувача з ПО, частота появи помилок і скарг на незручності. Отже самим значимим фактором є те, чи допомагає дана програма вирішувати дійсно значимих для користувачів завдань.

РОЗДІЛ 2

ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Етапи проектування

Одним з найбільше технологічних підходів до розробки зручного користувальницького інтерфейсу є проектування, орієнтоване на використання (usage-centered design), запропонований Л. Константайном і Л. Локвуд (L. Constantine, L. Lockwood) [3].

Основною ідеєю цього методу є використання спеціальних моделей, що сприяють адекватному визначенню набору завдань, які необхідно вирішувати користувачам, і способів організації інформації, що дозволяють спростити їхнє рішення.

Список моделей, використовуваних у рамках проектування, орієнтованого на використання, що випливає.

Модель ролей. Ця модель являє собою список ролей користувачів системи. Кожна роль - це група зв'язаних завдань і потреб деякої безлічі користувачів. Модель ролей може визначати зв'язку між ролями (ролі можуть уточнювати один одного, включати один одного або просто бути схожими) і набір з одних-трьох центральних ролей, на яких, в основному, і буде націлене проектування. Крім того, кожна роль може бути постачена профілями, що вказують різні її характеристики стосовно контексту використання системи. Профілі можуть включати наступну інформацію.

Обов'язку — вимоги до знань (про предметну область, про саму систему та ін.), яким користувач у даній ролі, швидше за все, задовольняє.

Уміння — рівень майстерності в роботі із системою.

Взаємодії — типові варіанти взаємодії користувача в цій ролі із системою, включаючи їхню частоту, регулярність, безперервність, концентрацію, інтенсивність, складність, передбачуваність, а також керування взаємодією (чи направляється воно користувачем, або він тільки реагує на дії системи).

Інформація - джерела, обсяг, напрямок передачі й складність інформації при взаємодії із системою. для розв'язуваних завдань. Зручно описувати такі сценарії у вигляді двох послідовностей - устремлінь користувача (не дій, а завдань які він хоче вирішити) і зобов'язань системи у відповідь на ці устремління.

Приклад опису звичайного й сутнісного варіанта використання при роботі наведений на рис. 2.2.

Действия	Реакции
Вставить карту	
	Считать данные
	Запросить PIN
Ввести PIN	
	Проверить PIN
	Вывести меню
Нажать клавишу выдачи денег	
	Запросить сумму
Ввести сумму	
	Вывести сумму
	Запросить подтверждение
Нажать клавишу подтверждения	
	Вернуть карту
	Выдать деньги
	Напечатать чек
	Выдать чек

Задачи	Обязательства
Регистрация в системе	
	Проверка личности
	Предложение набора операций
Выбор операции выдачи денег	
	Выдача денег
	Выдача чека

Рис. 2.2. Опис звичайного (ліворуч) і сутнісного (праворуч) варіанта використання.

У результаті модель завдань являє собою набір перероблених варіантів використання зі зв'язками між ними по узагальненню, розширенню й використанню. Деякі з варіантів використання оголошуються основними - без них програма втратить значну кількість користувачів.

Усяка користувальницька роль при цьому повинна бути пов'язана з одним або декількома варіантами використання.

Модель умісту. Модель умісту користувальницького інтерфейсу описує набір взаємозалежних контекстів взаємодії або робітників просторів із уміщеними в них даними й можливими в їхніх рамках діями.

При побудові цієї моделі потрібно визначити, що ввійде до складу інтерфейсу (які дані й функції), і не вирішувати питання про те, як саме воно буде виглядати. На початковому етапі один контекст взаємодії ставиться у відповідність одному (не допоміжному!) варіанту використання або групі дуже схожих варіантів, для виконання яких знадобиться той самий набір інструментів.

Критерії зручності — специфічні критерії зручності роботи для даної ролі (швидкість реакції, точність вказівок, зручність навігації та ін.).

Функції — специфічні функції, можливості й властивості системи, необхідні або корисні для даної ролі.

Можливі збитки від помилок, які може зробити людина в даній ролі, ризики використання різних функцій.

Приклад моделі ролей для користувачів банкомата наведений на Рис. 2.1.

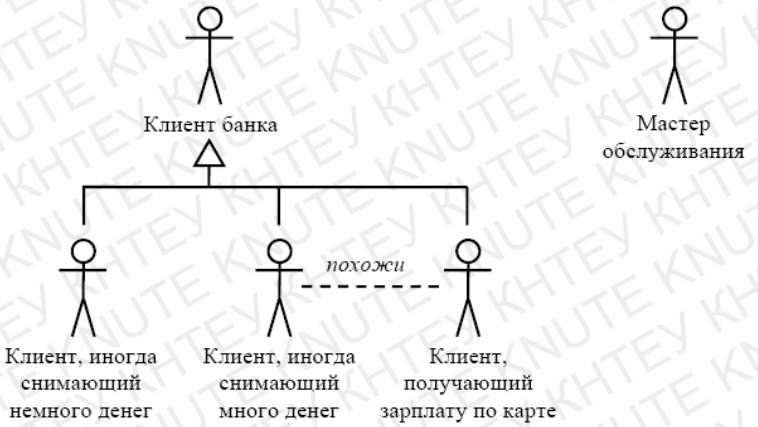

Рис. 2.1. Модель ролей користувачів банкомата.

Модель завдань. Модель завдань при проектуванні користувальницького інтерфейсу будується на основі сутнісних варіантів використання (essential use cases).

Опис сутнісного варіанта використання відрізняється від звичайного тем, що в рамках його сценаріїв виділяються тільки мети й завдання користувача, а не конкретної його дії.

Метою такого виділення є звільнення від неявних припущень про

наявність певних елементів інтерфейсів, що допомагає розробляти їх саме

Засоби для підтримки допоміжних розширювальних варіантів використання звичайно зручно поміщати в контексти розширюваних ними основних варіантів. Спочатку визначається, яка інформація повинна перебувати в заданому контексті для успішного рішення завдань відповідного варіанта використання, потім визначається список необхідних операцій для роботи із цією інформацією. Часто під час обговорення вмісту контексту взаємодії для його подання використають аркуш паперу з наклеєними на нього підписаними стікерами різних квітів (для розрізнення інформаційних елементів і елементів керування). Таке подання зручно для швидкого внесення змін по ходу обговорення. Воно також наочно показує, що розглядається лише прототип вікна або сторінки, а його елементи абстрактні, для них поки не пропонується конкретна форма. Його важко прийняти за «майже готовий» проект вікна, форми або сторінки, що описує підсумкові форму, розташування й кольори елементів інтерфейсу.

Після визначення набору контекстів і їх інформаційного й функціонального вмісту рисується карта навігації між контекстами, що показує можливі переходи між ними. Карта навігації поєднує різні контексти взаємодії в рамках моделі вмісту інтерфейсу.

На рис. 2.3 наведений приклад моделі вмісту вікна пошуку номерів телефонів програми, що реалізує корпоративний телефонний довідник.

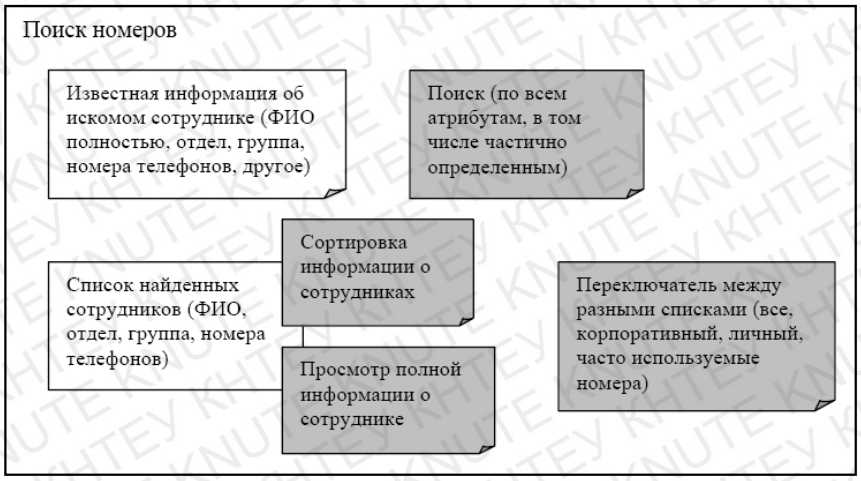

Рис. 2.3. Приклад моделі вмісту контексту взаємодії.

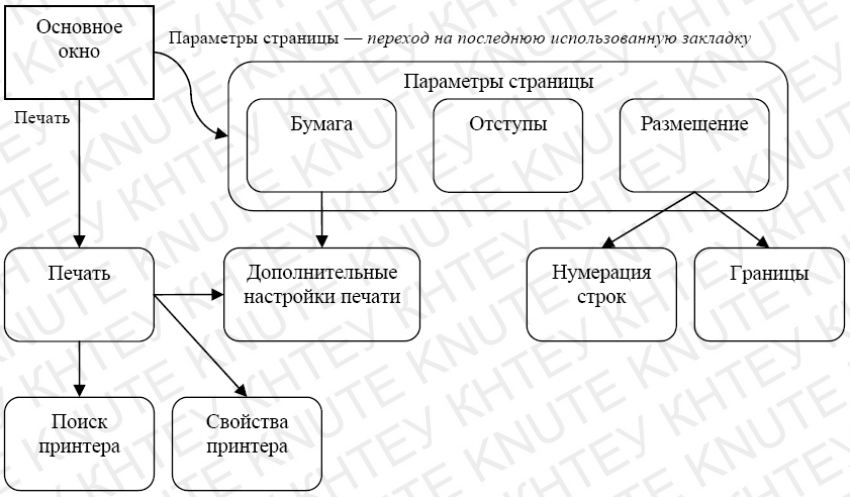

Рис. 2.4. Частина карти навігації редактори Microsoft Word.

Після розробки моделі вмісту всякий основний варіант використання повинен бути підтриманий за допомогою одного або декількох контекстів взаємодії. Чим менше контекстів потрібно використати для виконання одного варіанта використання, тим краще.

Перераховані три види моделей - ролей, завдань і вмісту - є основними. Два види, що залишилися використаються при необхідності.

• Операційна модель описує контекст використання системи й складається із профілів користувальницьких ролей.

• Модель реалізації являє собою візуальний проект інтерфейсу й опис його роботи.

Основні види діяльності в рамках проектування, орієнтованого на використання, що випливають.

• Спільне з користувачами визначення вимог до ПО, з урахуванням побажань

і вимог до його інтерфейсу.

• Розробка моделі предметної області за допомогою користувачів.

• Розробка моделей ролей і завдань за допомогою користувачів.

• Розробка моделі вмісту.

• Розробка візуального проекту інтерфейсу (моделі реалізації).

• Контроль зручності використання проекту інтерфейсу за участю користувачів.

• Проектування об'єктної структури ПО.

• Визначення стандартів і стилю інтерфейсу із залученням користувачів.

• Проектування й розробка довідкової системи й документації.

• Прив'язка інтерфейсу до контексту використання.

• Ітеративна розробка архітектури ПО.

• Ітеративне конструювання ПО з поступовим введенням запланованих функцій.

• Контроль зручності використання готового ПО.

Рис. 2.5. Взаємозв'язки й розподіл діяльностей у часі. Діяльності, у яких залучені користувачі, позначені зірочкою.

Ці діяльності не виконуються строго один за одним у вигляді окремих кроків. Для опису їхнього розподілу в часі використається діаграма, зображена на рис. 2.5.

Одним з важливих показників якості програмного забезпечення є зручність його використання. Воно описується за допомогою таких характеристик, як зрозумілість користувальницького інтерфейсу, легкість навчання роботі з ним, трудомісткість рішення певних завдань із його допомогою, продуктивність роботи користувача з ПО, частота появи помилок і скарг на незручності. Для побудови дійсно зручних програм необхідний

облік контексту їхнього використання, психології користувачів, необхідності допомагати починаючим користувачам і надавати все потрібне для роботи досвідчених.

Однак самим значимим фактором є те, чи допомагає дана програма вирішувати дійсно значимі для користувачів завдання.

Багато програмістів мають технічний або математичний склад розуму. Для таких людей «зрозумілість», «легкість навчання» представляються досить суб'єктивними факторами. Самі вони досить легко сприймають складні речі, якщо ті представлені в рамках несуперечливої системи понять, як би дико ця система й вхідні в неї поняття не виглядали для стороннього спостерігача. Типовий підхід програміста при розробці користувальницького інтерфейсу - надати користувачеві ті ж важелі й кнопки, за допомогою яких програмісти самі хотіли б управляти своєю програмою.

Психологічні й фізіологічні фактори.

Питання зручності використання програмного забезпечення тісно пов'язані з аналогічними питаннями для інших видів інструментів і встаткування, а також предметів побуту. І вирішуються вони на приблизно тій же основі, що й питання типу « чи зручна ця дверна ручка», « чи зручно таке табло спідометра в автомобілі», « чи зручний даний спосіб керування верстатом» та ін.

На рішення цих питань величезний вплив роблять загальні закони психологи й фізіології людини, адже речі зручні або незручні здебільшого не через суб'єктивні переваги, а через те, що будова людського тіла й закони роботи свідомості допомагають або заважають використати їх ефективно.

Фундаментальною основою для визначення зручностей і незручностей розуміння людиною функціонування й способів використання різних предметів є когнітивна психологія, що вивчає будь-які пізнавальні процеси людської свідомості. Психологія використання машин, інструментів, устаткування й предметів побуту в ході практичної діяльності людини звичайно називається інженерною психологією [1,2]. За кордоном виділена особлива наука, що вивчає психологічні, фізіологічні й анатомічні аспекти

взаємодії людини й комп'ютера, що так і називається - взаємодія людини й комп'ютера, Human-Computer Interaction, HCI.

При розгляді завдань побудови зручного ПЗ використають багато інформації з перерахованих дисциплін.

Найбільш важливі для розробки користувальницького інтерфейсу результати цих дисциплін можна сформулювати в такий спосіб.

Швидкісні показники діяльності людини

Час, що людина затрачає на різні дії, пов'язані з роботою на комп'ютері, таке [3,4].

• Натискання на клавішу клавіатури: 0.2-1.25 с.

• Натискання на кнопку миші: 0.1 с.

• Переміщення курсору миші: 1.0-1.5 с.

Значний час затрачається й на дії, які теж постійно необхідно виконувати, хоча вони не мають такого прямого фізичного вираження, як попередні.

• Розпізнавання візуального образа: 0.1 с.

• Переклад погляду й перемикання уваги з одного об'єкта на інший: 0.25 с.

• Вибір із двох альтернатив (ухвалення найпростішого рішення): 1.25 с.

• Перемикання уваги з миші на клавіатуру й назад: 0.36 с.

На даних такого роду заснований GOMS - метод оцінки продуктивності роботи з інтерфейсом (див. далі).

Спостереження показують [5], що більшу частину часу при роботі людини витрачає на інтелектуальну діяльність, пов'язану з визначенням цілей своїх дій, визначенням ланцюжка конкретних дій, яку потрібно зробити, щоб досягти поставлених цілей, виявленням всіх необхідних для цього елементів, розпізнаванням чергового стану системи і його оцінкою з погляду досягнення обраних цілей. Змінюючи користувальницький інтерфейс, можна лише допомогти користувачеві швидше знайти потрібні йому інструменти, виконати потрібні дії й швидше зрозуміти, які ж результати вони дали.

Із наведених даних можна вивести кілька важливих висновків

• Людина сприймає й усвідомлює інформацію, а також робить дії досить повільно в порівнянні з комп'ютером.

Сама «повільність» дій людини і його сприйняттів, а також співвідношення витрат часу на різні дії, повинні враховуватися при проектуванні інтерфейсів, розрахованих на взаємодію з людиною.

Око швидше руки - людини набагато швидше довідається щось, чим робить відповідні дії. Тому, зокрема, людині часто зручніше працювати із системами контекстної підказки, що показують йому можливі варіанти його подальшого уведення, чим набивати весь текст цілком самостійно.

Як окреме спостереження можна помітити, що людина набагато швидше довідається щось, чим згадує, як воно називається. Значно простіше вказати малознайомої людини на фотографії, чим згадати його ім'я й прізвище. Точно також простіше вибрати якийсь елемент із наданого списку, чим набрати його ідентифікатор або імена.

Наведені значення для часу переміщення покажчика миші можна уточнити за допомогою закону Фіттса (Fitts) [6] — $T = a + b \cdot \log(D/W)$ (інколи використовується формула $a + b \cdot (D/W)^{½}$). Де D означає відстань, на яку переноситься вказівник, W — лінійний розмір об’єкту, а та b — деякі константи, залежні від людини та пристрою миші. Цей закон говорить, що чим ближче та більше елемент управлення, тим швидше можливо попасти в нього за допомогою мишки.

Із цього треба, що зручніше розташовувати потрібні користувачеві елементи керування ближче до покажчика миші й робити їх більше. Крім того, оскільки миша не можна винести за край екрана, елемент, розташований на краю, сприймається як «нескінченно великий» - потрапити в нього набагато легше, ніж в елемент аналогічного розміру й на тім же відстані від покажчика, але з усіх боків оточений «порожнім» простором.

Люди найчастіше промахуються при першій спробі потрапити покажчиком у потрібне місце, але більшість швидко поправляється, навіть не усвідомлюючи промаху, що відбувся. Якщо система поблажлива до промахів

покажчика, які швидко виправляються, вона зручніше тієї, котра негайно реагує на такі події.

Наприклад, панель завдань в Windows випадає відразу по наведенні на неї покажчика миші, тому при спробах натиснути кнопку, розташовану десь унизу екрана, вона часто з'являється «зненацька», і користувачеві доводиться гаяти час на очікування того, щоб вона зникла назад.

Людині дуже важко довго зберігати зосередженість і не відволікатися від якоїсь діяльності. Найчастіше, мимо волі самої людини, через якийсь час йому в голову приходять різні думки, і погляд сам собою вповза кудись убік.

Це означає, що не можна жадати від людини тривалої уваги до роботи. Саме розуміння того, що необхідно зосередитися, створює в людини стрес і викликає негативні емоції. Тільки в дуже ризикованих ситуаціях, коли від його дій залежить дуже багато чого - піку літака, що перевищує всі норми розігрів хімічного реактора - користувач може зосередитися на значний час, але однаково ми всі надаємо перевагу їх уникати. Такі ситуації повинні явно відзначатися якимись характерними сигналами, і в жодному разі ці ж або схожі сигнали не можна використати для набагато менш серйозних ситуацій, інакше користувачі будуть спантеличені й можуть їх переплутати, що приводить до сумних наслідків.

В інші ж ситуаціях ПО повинне бути готове до постійних перемикань уваги користувача й ненав'язливо допомагати йому відновити фокус уваги на останніх діях, які він виконував, а також згадати їхній контекст - що він взагалі робив, на якому кроці він зараз перебуває, що ще потрібно зробити та ін.

Тому, наприклад, екранні форми, заповнення яких входить в одну процедуру, добре б позначати так, щоб користувач відразу бачив, скільки кроків він уже виконав, і скільки ще залишилося. А поточне поле уведення якось виділяти серед всіх полів уведення на формі.

Для залучення уваги користувачів до якихось повідомлень або елементів керування потрібно пам'ятати правило: спочатку рух, потім яскравий колір,

потім все інше. Увага людини насамперед акцентується на об'єктах, що рухаються, потім - на виділених кольорами, і тільки потім на інших формах виділення.

Кращим способом залучення уваги є поява анімації, ледве менш діючим - яскраві кольорові вікна й повідомлення. Для м'якого, не ріжуче око залучення уваги можна використати виділення за допомогою квітів м'яких відтінків.

Навпаки, поява на екрані або сторінці непотрібних кольорових анімованих картинок є кращим способом відволікти людини від виконуваної їм роботи - ока поневолі переводяться на таку картинку, і потрібне психологічне зусилля і якийсь час, щоб від її відірватися. Деякі люди навіть закривають анімацію рукою, щоб вона не заважала зосередитися на корисній інформації на екрані!

Яскраво виділених елементів на одному екрані не повинне бути багато, більше, ніж один-два, інакше чоловіки перестає звертати увагу на таке виділення.

Інший фактор, пов'язаний з увагою користувачів, - їхня оцінка часу виконання системою заданих дій. Якщо людина очікує, що система буде друкувати документ досить довго, те, пославши його на печатку, він піде налити собі чаю й повернеться на той час, коли, по його оцінці, система повинна закінчити роботу. Тому, перш ніж виконувати довгі операції, потрібно переконатися, що всі необхідні дані від користувача отримані. Інакше й користувач, і система будуть гаяти час, перший - очікуючи на кухні, коли ж система виконає роботу (яку він начебто б запустив), а друга - очікуючи уведення додаткових даних.

По тій же причині варто акуратніше ставитися до індикаторів ступеня виконання, оскільки за їх показниками користувачі часто оцінюють час, що залишився, до кінця виконання завдання.

Людина «розуміє» що-небудь, укладаючи це у свідомості в деяку систему асоціативних зав’язків. Прикладами механізмів, які допомагають у

цьому, є наступні [4].

• Ментальна модель. Цей механізм ілюструється прикладом зі збереженням файлів, про яке розповідалося вище. Користувач розуміє, як працювати із системою, якщо йому пояснюють набір сутностей, у термінах яких вона функціонує, і правила роботи з ними.

Людина, один раз зрозумівши модель дій якоїсь системи, може пам'ятати її досить довго й сприймає більшість подій, що відбуваються в системі, як цілком природні, навіть якщо керуючі нею правила досить складні й неочевидні. Проблема в тім, що моделі роботи більшості систем зараз досить складні, а більшість людей із працею сприймають складні моделі, а, починаючи з деякого рівня складності, не сприймають взагалі. До того ж для навчання людей такої моделі потрібно витрачати додаткові зусилля, або ж потрібні додаткові стимули, щоб люди прочитали документацію, у якій вона пояснюється.

Метафора. Зараз досить часто пояснення того, як працює ПО, будуються навколо тієї або іншої метафори, тобто приводиться якийсь механізм або предмет з реального життя (або літератури), знайомий більшості користувачів, і говориться, що дане ПО працює так само. Це допомагає зрозуміти основні правила його роботи дуже швидко, навіть якщо їх досить багато, на відміну від довгих пояснень моделі. Прикладом метафори служить робочий стіл Windows. Для опису його роботи залучається аналогія з робочим столом офісного службовця, на якому лежать різні документи й інструменти. Цей же приклад досить наочно показує границі застосовності метафори для підвищення зрозумілості інтерфейсу - навряд чи за допомогою цієї метафори можна поширити розуміння роботи Windows за межі роботи з файлами й папками, що лежать на робочому столі. На жаль, дуже рідко є або може бути придумана досить адекватна метафора навіть для частини системи. Ще більш рідко така метафора може охопити систему цілком. Інакше, правила роботи ЗД покриваються набором слабко зв'язаних метафор, і потрібно постійно пам'ятати границі їхньої застосовності, а користувачі можуть заплутатися в

цьому накопиченні. Але навіть у такому випадку метафори допомагають, якщо чітко обмежити зони їхньої дії. Наочність (affordance). Властивість наочності означає, що саме подання інтерфейсу підказує, як з ним треба працювати, використовуючи широко розповсюджені стереотипи й наочні (не обов'язково видимі очами!) зв'язку між елементами керування й керованих об'єктів.

Приклади наочності в ПО й інших областях:

Псевдо трьох вимірні виступаючі кнопки як би пропонують «Натисни мене!». При натисканні вони на час «втоплюються», закріплюючи в такий спосіб наочний асоціативний зв'язок зі звичайними кнопками.

Якщо розташувати вентилі для керування конфорками газової плити у вигляді тієї ж геометричної фігури, як і самі конфорки (звичайно по кутах квадрата), відповідність між вентилями й керованими ними конфорками стає набагато ясніше, ніж при традиційному розташуванні вентилів на одній прямій.

Придумати інтерфейс, що має властивість наочності дуже непросто, однак якщо це вдається, така система майже завжди стає великою подією на ринку.

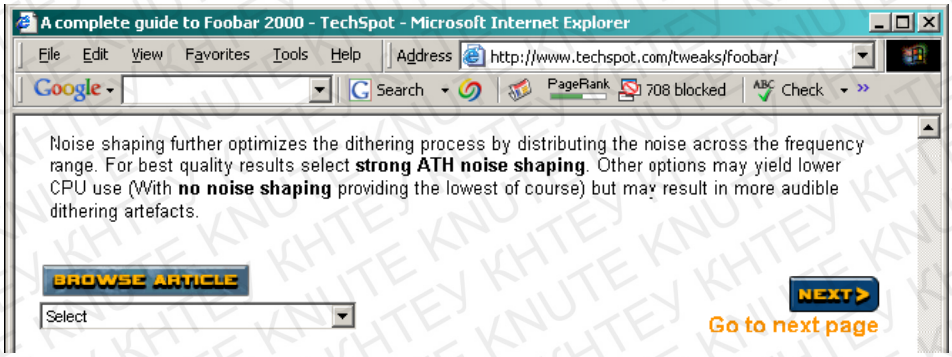

Рис. 2.6. Античність. "Кнопка" Next не натискається.

Набагато частіше, на жаль, можна знайти приклади античності - інтерфейс всім своїм видом «говорить» одне, а використати його треба зовсім інакше, скажемо, намальована кнопка, а натиснути її не можна. Античність необхідно уникати всіма силами. Наприклад, якщо дверна ручка, «пропонуюча»

повернути себе (ручка, за якої зручно хапатися, з одним вільним кінцем, а іншим закріплена на круглій ніжці), не повертається, найчастіше її ненавмисно відламують.

Стандарт. Людина добре сприймає те, що звично. Одним їхній способів «впровадження» потрібних звичок є стандартизація й широке використання стандартних інтерфейсів. Останнім часом стандартам зовнішнього вигляду інтерфейсу в цілому і його окремих елементах надається велике значення в плані забезпечення зручності використання.

Цей підхід дійсно зменшує витрати на навчання роботі з новою програмою, оскільки користувачам легшає орієнтуватися в ній. Іноді вдається виробити гарний стандарт, що значно полегшує життя й користувачам, і розроблювачам. Однак значно підвищити зручність використання, особливо для досвідчених користувачів, тільки за рахунок застосування стандартних елементів інтерфейсу найчастіше не вдається. Це відбувається тому, що стандарт, по-перше, закріплює не всі необхідні деталі, по-друге, далеко не самі зручні, а іноді й зовсім незручні елементи. Наприклад, стандартним засобом для роботи з документами, які не вміщаються в рамки вікна програми, на якийсь час стали смуги прокручування, що реалізують незручний для людини спосіб переміщення по великому просторі. Зіставте свої дії й рухи при розглядані великої карти або схеми з тим, що доводиться вживати при використанні для цього смуг прокручування.

2.2. Описання моделі програмної системи в UML-нотації

Програма складається з візуальних форм-модулів, що дозволяю структурувати програмний продукт за подібними характеристиками функціонального контенту.

Схему функціонування програмного продукту представлено на UseCase UML діаграмі прецедентів програми, що наведена на рисунку 2.7.

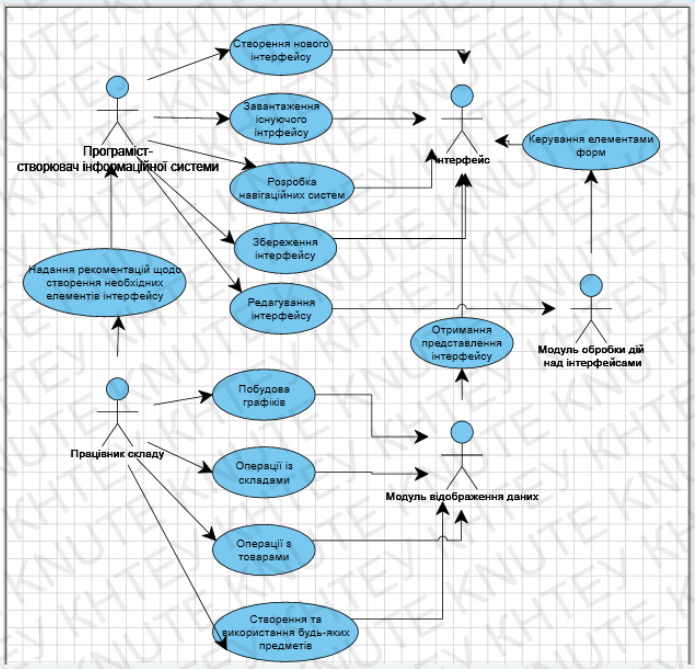

Рис. 2.7. – UseCase UML діаграма прецедентів програми

2.3. Описання діаграми класів

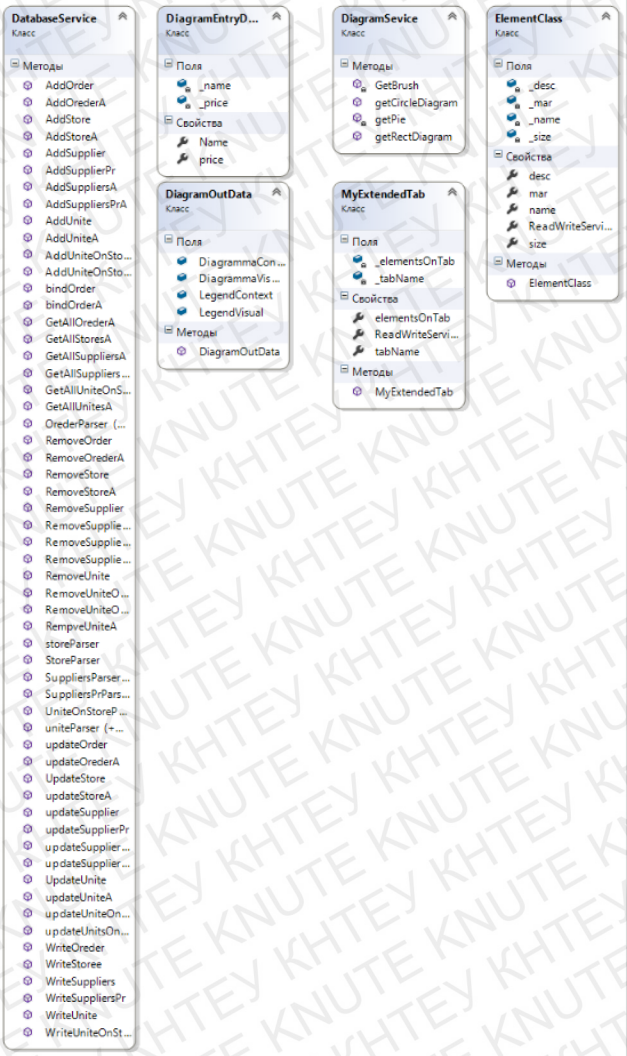

Рис. 2.8. – Діаграма класів

Програма складається з таких класів:

class DatabaseService – призначення для роботи з базою даних.

class DiagramEntryData – призначення для побудови діаграм.

class DiagramOutData – призначення для виводу діаграм.

class DiagramSevice – призначення для графіки.

class MyExtendedTab – призначення для управліннями вкладками (інтерфейсом).

public class ElementClass – призначений для вкладок програми.

2.4. Описання діаграми діяльності

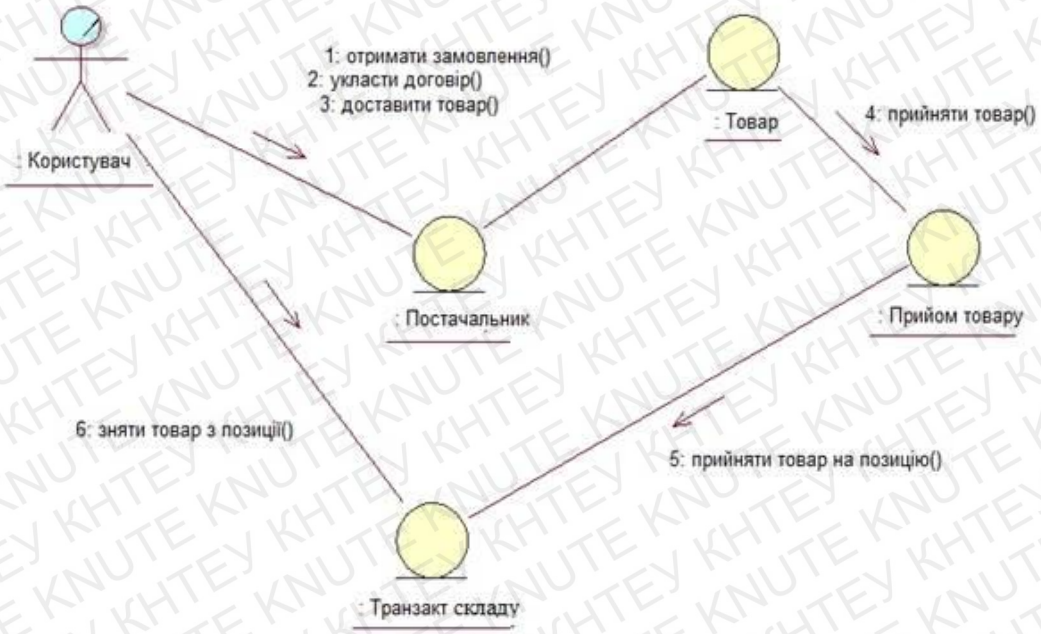

Рис. 2.9. Діаграма діяльності - зберігання товару на складі.

Діаграма діяльності — в UML, візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.

Дія є фундаментальною одиницею визначення поведінки в специфікації.

Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів. Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності.

Що найменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності. Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

2.5. Описання діаграми послідовності

Діаграма послідовності —послідовності відображає взаємодії об'єктів впорядкованих за часом.

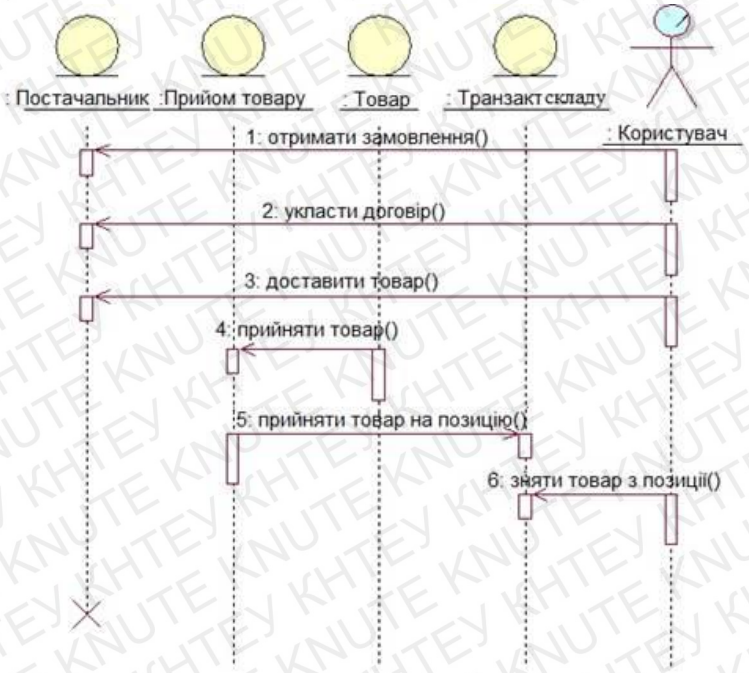

Рис. 2.10. – Діаграма послідовності дій складської програми

2.6. Створення системи складського обліку "Нова пошта"

Системи складського обліку "Нова пошта" повинна забезпечувати надійне зберігання інформації, пошук інформації, що відповідає заданим умовам та її наочне представлення. Уся інформація повинна зберігатися у базі даних.

БД повинна включати в себе наступні дані:

- Відомості про постачальників, які забезпечують завод сировинними ресурсами;
- Відомості про постачальників, які надають готові вироби;
- Повну інформацію щодо наявності всіх ресурсів.
- Система повинна реалізувати автоматизацію наступних функцій:
- Розрахунок фактичного споживання ресурсів кожного з виду: сировина, заготовка, замовлених виробів, мастил, фарб, канцелярських виробів, тощо.
- Формування звітності по використанню матеріалів для виготовлення певних вузлів виробів
- Своєчасне попередження щодо строків постачання сировини;
- Доповнення, вилучення та коригування даних про постачальників
- Можливість пошуку по заданому параметру.

Вхідними даними для даного проекту є данні базі даних SQL, та данні налаштування інтерфейсів.

Данні про позиції елементів, розміру та налаштуванні закладок, зберігаються у текстовому вигляді и мають розширення «.interface»

Приклад:

1|Все__доступные__товары:some desc:0-0:350-300&Все__доступные__поставшики:some desc:678-10:300-290&Товары__поставшика:some desc:360-0:310-300

де:

1 – назва закладки

Всі__доступні__товари: some desc:0-0:350-300 – назва першого елементу,

невеликий опис, його позиція та розмір.

Всі__доступні__постачальники: some desc:678-10:300-290 – назва другого елементу, його позиція та розмір.

Товари__постачальника: some desc:360-0:310-300 – назва третього елементу, його позиція та розмір.

Запустившись програма намагатиметься завантажити той інтерфейс що був заданий за замовчуванням, якщо він був не знайденій то буде завантажено стандартний інтерфейс що знаходиться поряд з файлом розширення .exe і має назву DefaultInterface.interface.

Данні бази даних зберігаються у StoreDB.MDF. База даних складається з 6 таблиць.

SuppliersProducts

OrdersTable

SuppliersTable

UnitsTable

StoreTable

UnitsOnStores

На діаграмі залежності , що зображена на рисунку 2.11 , наочно можна побачити ці таблиці, назви їх полів, та зв’язки між таблицями.

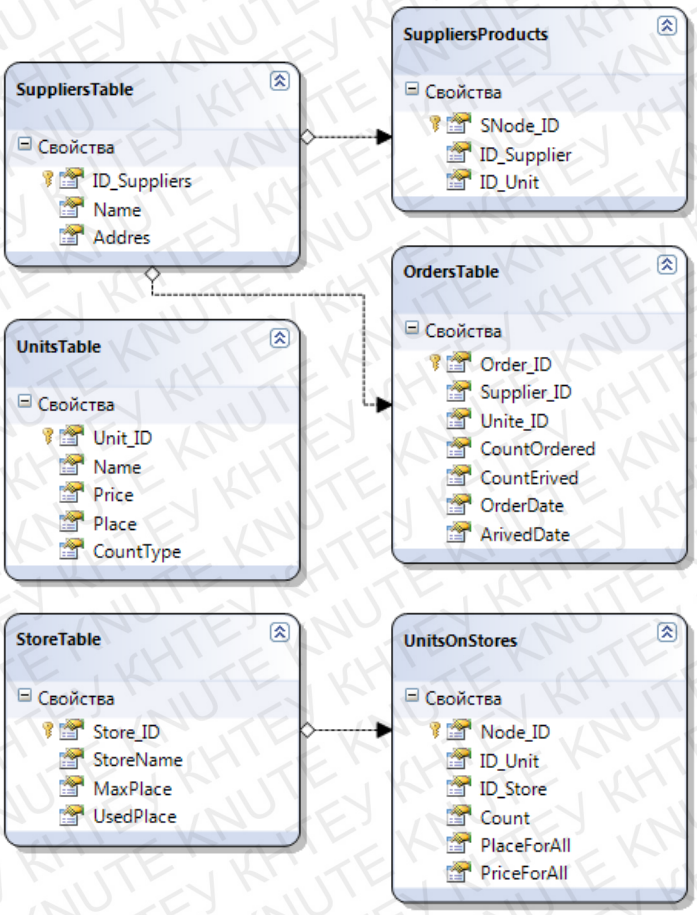

Рис. 2.11. – Діаграма залежності таблиць бази даних

Для запису нових елементів до бази чи видалення, оновлення вже існуючих використовуються методи що описані у статичному класі DatabaseService. Ці методи для зручності поділимо на 3 основні типи, бо є саме три функції різного направлення для кожної таблиці.

Додавання, заповнення полів даних нового елемента та додавання його до таблиці.

Оновлення, редагування вже існуючого елемента таблиці, а точніше його окремих полів. Видалення, видалення все існуючого елемента таблиці.

2.6.1. Характеристика мови програмування

C# - об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом

Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов C++, Delphi, Модула і Smalltalk — C#, спираючись на практику їхньоговикористання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем: так, C# не підтримує множинне спадкування класів (на відміну від C++) або виведення типів (на відміну Haskell).

C# є дуже близьким родичем мови програмування Java. Мова Java була створена компанією Sun Microsystems, коли глобальний розвиток інтернету поставив задачу роззосереджених обчислень. Взявши за основу популярну мову C++, Java виключила з неї потенційно небезпечні речі (типу вказівників без контролю виходу за межі). Для роззосереджених обчислень була створена концепція віртуальної машини та машинно-незалежного байт-коду, свого роду посередника між вихідним текстом програм і апаратними інструкціями комп'ютера чи іншого інтелектуального пристрою.

Java набула чималої популярності, і була ліцензована також і компанією Microsoft. Але з плином часу Sun почала винуватити Microsoft, що та при створенні свого клону Java робить її сумісною виключно з платформою Windows, чим суперечить самій концепції машинно-незалежного середовища виконання і порушує ліцензійну угоду. Microsoft відмовилася піти назустріч вимогам Sun, і тому з'ясування стосунків набуло статусу судового процесу.

Суд визнав позицію Sun справедливою, і зобов'язав Microsoft відмовитися від поза ліцензійного використання Java.

У цій ситуації в Microsoft вирішили, користуючись своєю вагою на ринку, створити свій власний аналог Java, мови, в якій корпорація стане

повновладним господарем. Ця новостворена мова отримала назву C#. Вона успадкувала від Java концепції віртуальної машини (середовище .NET), байт-коду (MSIL) і більшої безпеки вихідного коду програм, плюс врахувала досвід використання програм на Java.

Нововведенням C# стала можливість легшої взаємодії, порівняно з мовами-попередниками, з кодом програм, написаних на інших мовах, що є важливим при створенні великих проектів. Якщо програми на різних мовах виконуються на платформі .NET, .NET бере на себе клопіт щодо сумісності програм (тобто типів даних, за кінцевим рахунком).

Станом на сьогодні C# визначено флагманською мовою корпорації Microsoft, бо вона найповніше використовує нові можливості .NET. Решта мов програмування, хоч і підтримуються, але визнані такими, що мають спадкові прогалини щодо використання .NET.

C# розроблявся як мова програмування прикладного рівня для CLR і, як такий, залежить, перш за все, від можливостей самої CLR. Це стосується, перш за все, системи типів C#. Присутність або відсутність тих або інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльована у відповідні конструкції CLR. Так, з розвитком CLR від версії 1.1 до 2.0 значно збагатився і сам C#; подібної взаємодії слід чекати і надалі. (Проте ця закономірність буде порушена з виходом C# 3.0, що є розширеннями мови, що не спираються на розширення платформи .NET.) CLR надає C#, як і всім іншим .NET-орієнтованим мовам, багато можливостей, яких позбавлені «класичні» мови програмування. Наприклад, збірка сміття не реалізована в самому C#, а проводиться CLR для програм, написаних на C# точно так, як і це робиться для програм на VB.NET, J# тощо.

XAML (англ. eXtensible Application Markup Language - розширювана мова розмітки додатків; вимовляється [Замле] або [земл]) - заснований на XML мова розмітки для декларативного програмування додатків,

розроблений Microsoft.

Модель додатків Vista включає об'єкт Application. Його набір властивостей, методів і подій дозволяє об'єднати веб-документи в зв'язаний додаток. Об'єкт Application контролює виконання програми і генерує події для користувальницького коду. Документи додатка пишуться на XAML. Втім, за допомогою XAML описується, перш за все, користувальницький інтерфейс. Логіка програми, як і раніше управляється процедурних кодом (C #, VB і т.д.). XAML може використовуватися як для браузер-базованих додатків, так і для локальних настільних додатків.

XAML включає основні чотири категорії елементів: панелі, елементи управління, елементи, пов'язані з документом і графічні фігури. Заявлено 7 класів панелей, які задають принципи відображення вкладених в них елементів. Для завдання положення елементів щодо меж батьківської панелі використовуються атрибути на кшталт властивостей в об'єктно-орієнтованих мовах. Подібний синтаксис не дуже в'яжеться з рекомендаціями CSS, але буде звичний програмістам настільних додатків.

Додатки, оголошені в XAML, можуть включати безліч сторінок. Елемент управління PageViewer дозволяє розбивати зміст на сторінки і забезпечує навігацію по них. Елемент ContextMenu допомагає в створенні навігаційних меню додатка. Код процедурного мови може бути розміщений безпосередньо у файлі XAML або ж призначений при складанні проекту.

XAML широко використовується в. NET Framework 3.0, в особливості в Windows Presentation Foundation (WPF), Windows Workflow Foundation (WWF) і Silverlight. У WPF XAML використовується як мова розмітки користувальницького інтерфейсу, для визначення елементів користувальницького інтерфейсу, прив'язки даних, підтримки подій та інших властивостей. У WF, за допомогою XAML можна визначати послідовності виконуваних дій (workflows).

XAML файли можна створювати і редагувати за допомогою інструментів візуального конструювання, таких як: Microsoft Expression Blend, Microsoft

Visual Studio, WPF visual designer.

Також, їх можна створювати за допомогою стандартного текстового редактора, редактора коду такого як: XAMLPad, або графічного редактора, такого як Vectropy. Все створене або реалізоване в XAML може бути виражене за допомогою більш традиційних. NET мов, таких як: C # або Visual Basic.NET. Однак, ключовим аспектом технології є зменшення складності використовуваних для обробки XAML інструментів, так як XAML заснований на XML. В результаті цього з'являється безліч продуктів, що створюють засновані на XAML додатка. Оскільки XAML базується на XML, у розробників і дизайнерів існує можливість одночасно працювати над вмістом без необхідності компіляції.

. NET Framework - програмна платформа, випущена компанією Microsoft в 2002 році. Основою платформи є виконуюча середу Common Language Runtime (CLR), здатна виконувати як звичайні програми, так і серверні веб-додатки. . NET Framework підтримує створення програм, написаних на різних мовах програмування.

Вважається, що платформа. NET Framework з'явилася відповіддю компанії Microsoft на набрала на той час велику популярність платформу Java компанії Sun Microsystems (нині належить Oracle). Хоча. NET є патентованої технологією корпорації Microsoft та офіційно розрахована на роботу під операційними системами сімейства Microsoft Windows, існують незалежні проекти, що дозволяють запускати програми. NET на деяких інших операційних системах. В основі WPF лежить векторна система візуалізації, яка не залежить від дозволу пристрої виведення і створена з урахуванням можливостей сучасного графічного устаткування. WPF надає засоби для створення візуального інтерфейсу, включаючи мову XAML, елементи управління, прив'язку даних, макети, двомірну і тривимірну графіку, анімацію, стилі, шаблони, документи, текст, мультимедіа та оформлення.

Графічної технологією, яка лежить в основі WPF, є DirectX, на відміну від Windows Forms, де використовується GDI / GDI +.

Продуктивність WPF вище, ніж у GDI + за рахунок використання апаратного прискорення графіки через DirectX. Також існує урізана версія CLR, яка називається WPF / E, вона ж відома як Silverlight.

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

2.7. Висновки до розділу 2.

В даному розділі проведено аналіз системи проектування, яка містить велику кількість інформації та виконує велику кількість функцій. Незалежно від її призначення, система повинна бути забезпечена засобом представлення інформації, яку вона зберігає та відображати інформацію про поточний режим роботи. Це виконується за допомогою списків вибори. Ці списки вибору називаються меню. Меню є головною формою для навігації в системі, і вони створюються, для того щоб допомогти користувачеві розібратися в ментальній моделі системи.

Меню повинно швидко розпізнаватися користувачем. Воно повинно надавати інформацію користувачам про доступні опції та інформацію про поточний стан системи.

Новачкам і недосвідчені користувачам системи може здатися складною для вивчення. Інформація в меню часто повинна запам'ятовуватися й бути інтегрована на серії вікон. Якщо кожне меню відділений друг від друга, то взаємодія з ними може бути важко для сприйняття. Слова й фрази можуть бути не коректо сприйняті через недосвідченість користувача.

Графічні й веб - системи дуже меню орієнтовані. Вони змінюються за формою й застосовуються в різних завданнях. У графічних системах вони використаються, щоб показувати команди й властивості, застосовувані до об'єктів, документам і вікнам. При виборі елемента графічного меню, може відкритися інше меню й показується яке-небудь вікно або виконується яку-небудь операція. Меню складається з панелей, так само існують спливаючі меню (pop-up, pull - down), багаторівневі, окремі й іконочні.

3.1. Графічний інтерфейс розробленого додатку

Даний програмний продукт – орієнтована система має назву «Розробка програмного забезпечення логістичного підприємства "Нова Пошта"».

Відділ складу підприємства використовує великі обсяги інформації, яку необхідно обробляти, формує велику кількість актів та звітності, постійно працює з численними показниками, які потрібно своєчасно та точно підраховувати та аналізувати.

Для написання даного програмного продукту було використано програмне забезпечення у вигляді середовища розробки Microsoft Visual Studio. Графічний інтерфейс можливо налаштовувати по бажанню користувача, на рисунку 3.1. бачимо базовий графічний інтерфейс, а на рисунку 3.2. інтерфейс налаштовано за потреб оператора.

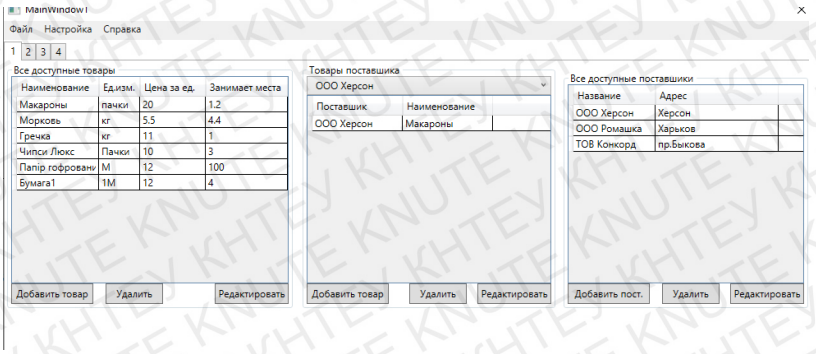

Рис. 3.1. – Головне вікно програм з базовим не налаштованим інтерфейсом.

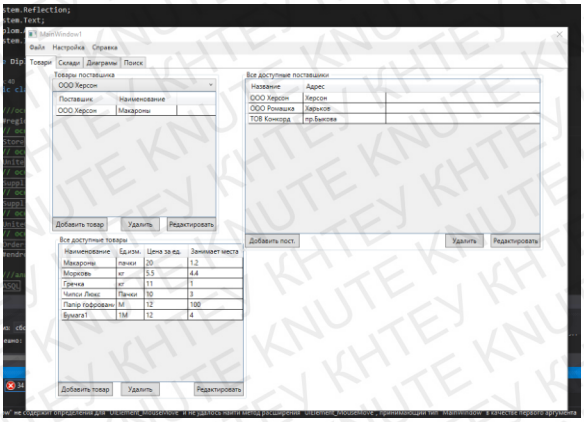

Рис. 3.2. – Головне вікно програм з налаштуваннями оператора.

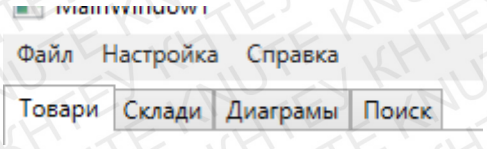

Рис. 3.3 – Налаштоване меню та вкладки програми

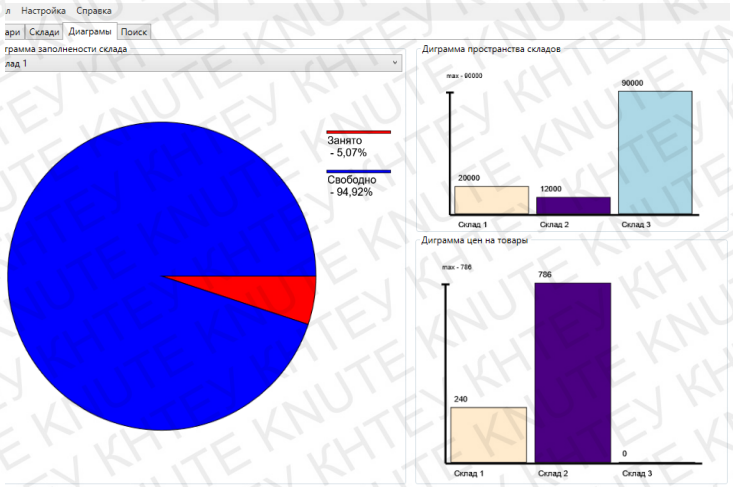

Рис. 3.4 – Графічний інтерфейс вкладки «Діаграми»

3.2. Архітектура розробленого додатку та інструкція користувача.

Запустившись програма намагатиметься завантажити той інтерфейс що був заданий за замовчуванням. Якщо він був не знайдений, то буде завантажено стандартний інтерфейс що знаходиться поряд з файлом розширення .exe і має назву DefaultInterface.interface, що представлено на рисунку 3.5.

- У головному вікні ми бачимо:

•По центру область закладок, в даному випадку їх 4

•Зверху головне меню

•Файл, де є такі пункти меню,

•Створити новій інтерфейс

•Завантажити інтерфейс

•Зберегти інтерфейс

•Використовувати за замовчуванням

•Вихід

•Налаштування, де є такі пункти меню,

•Змінити режим налаштування інтерфейсу, перейти до стану коли можна переміщати елементи на екрані

•Налаштування закладок

•Довідка, де не нема жодного підменю, та при натисканні перейдемо до вікна перегляду довідки.

Кожен елемент на першій вкладці має декілька кнопок, а саме по 3 кожний, які виконують три базові операції редагування таблиць бази даних.

Елементи що можуть бути на вкладках доволі фіксовані, нижче представлені вони всі.

На елементі «Усі доступні товари», що представлено на рисунку 3.5, відображено таблицю UnitsTable. Там перечисленні всі товари з якими може працювати склад. Цю таблицю можна по елементно оновлювати, додавати нові чи видаляти вже існуючі записи.

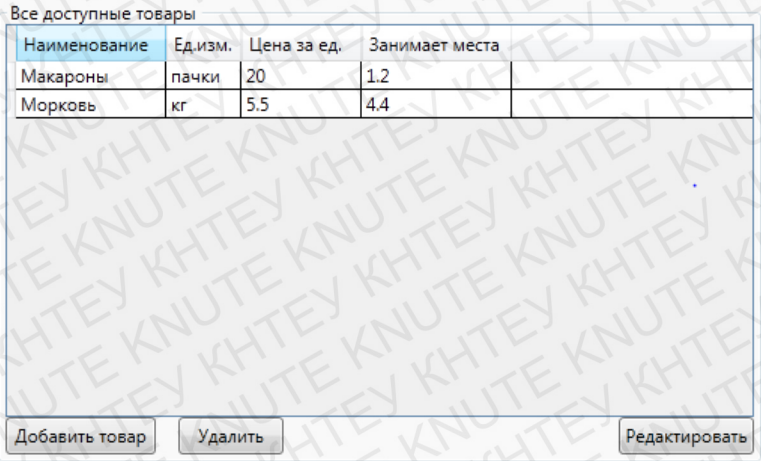

Рис. 3.5 – Елемент «Усі доступні товари»

На елементі «Товари постачальника», що представлено на рисунку 3.6, відображено таблицю SuppliersProducts, яка показує які товари постачає конкретній постачальник. Один постачальник може постачати навіть один і той самий товар двічі, наприклад за різну ціну. Цю таблицю можна по елементно оновлювати, додавати нові чи видаляти вже існуючі записи.

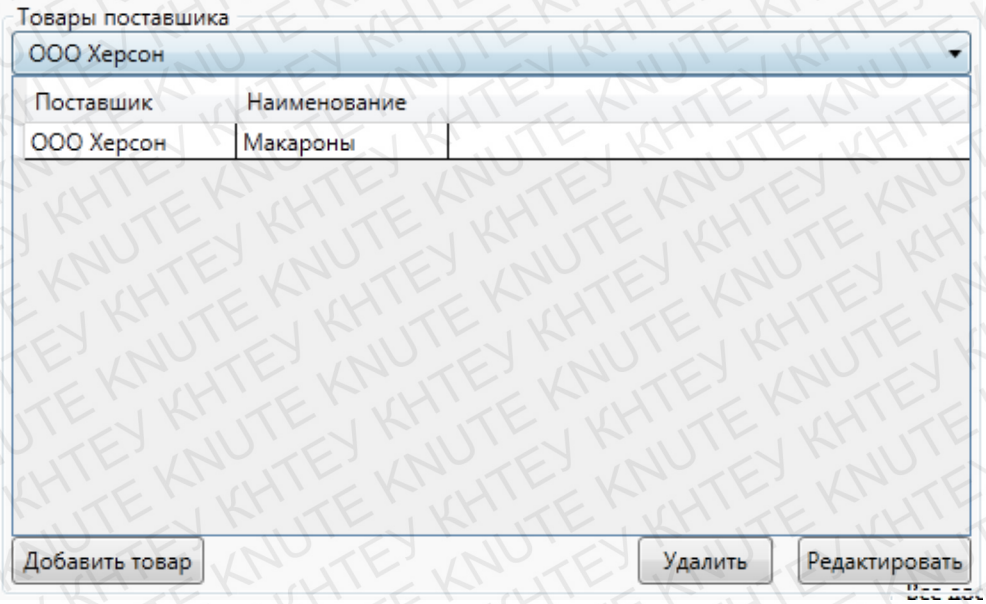

Рис. 3.6. – Елемент « Товари постачальника»

На елементі «Усі доступні постачальники», що представлено на рисунку 3.7, відображено таблицю SuppliersTable, яка показує які постачальники працюють зі складом. Кожен постачальник має назву, та адресу.

Цю таблицю можна по елементно оновлювати, додавати нові чи видаляти вже існуючі записи.

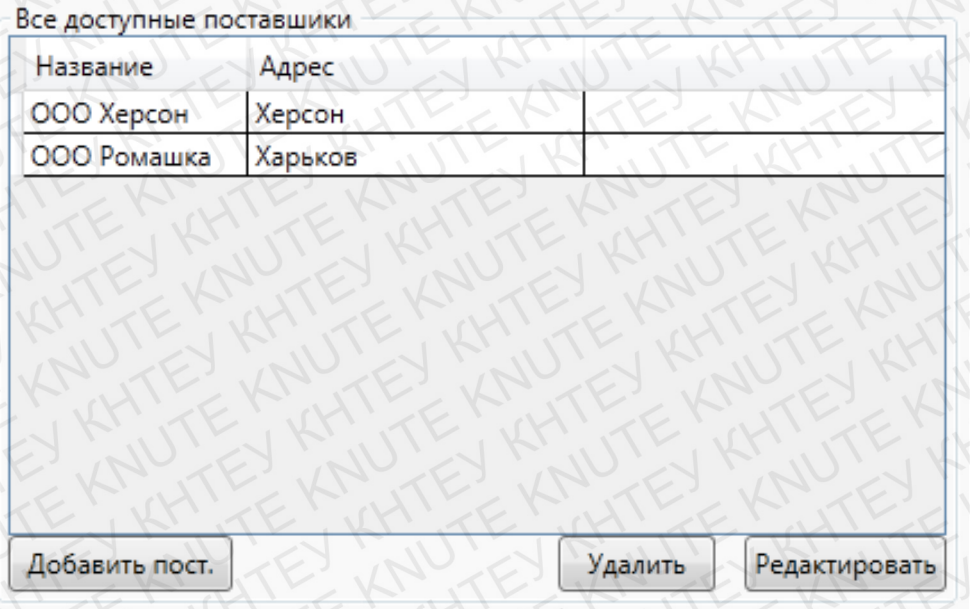

Рис. 3.7. – Елемент «Усі доступні постачальники»

На елементі «Товари у сховищі», що представлено на рисунку 3.8, відображено таблицю UnitsOnStores , яка показує які товари на якому складі в якій кількості за якою ціною, тобто відображено данні всіх полів таблиці UnitsOnStores, у зручному на наочному вигляді. Цю таблицю можна по елементно оновлювати, додавати нові записи, але тільки через таблицю замовлення, яка описана нижче чи видаляти вже існуючі записи.

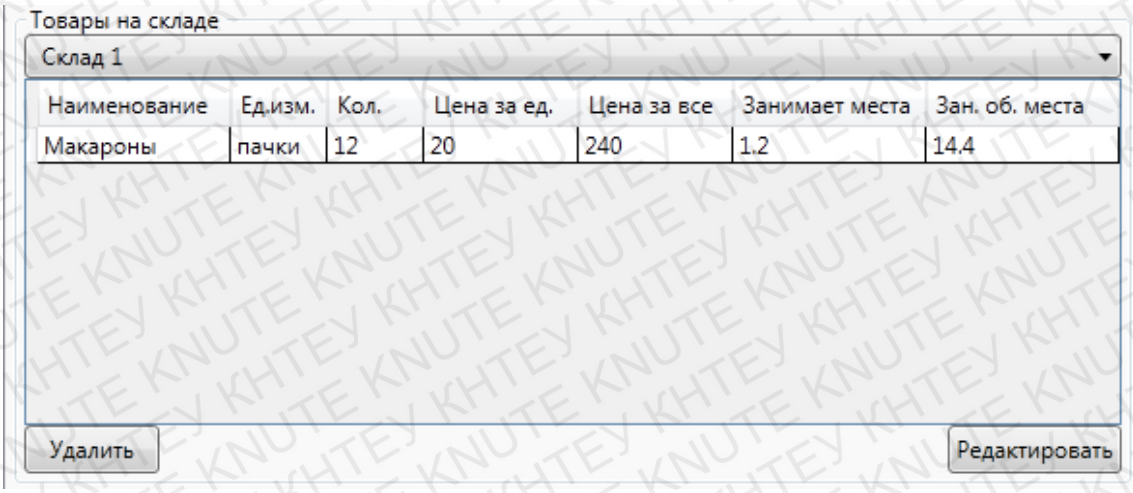

Рис. 3.8. – Елемент «Товари у сховищі»

На елементі «існуючі склади», що представлено на рисунку 3.9, відображено таблицю StoreTable, яка показує які склади існують, вказана їх

максимальний вміст та скільки простору вже зайнято. Цю таблицю можна по елементно оновлювати, додавати нові чи видаляти вже існуючі записи

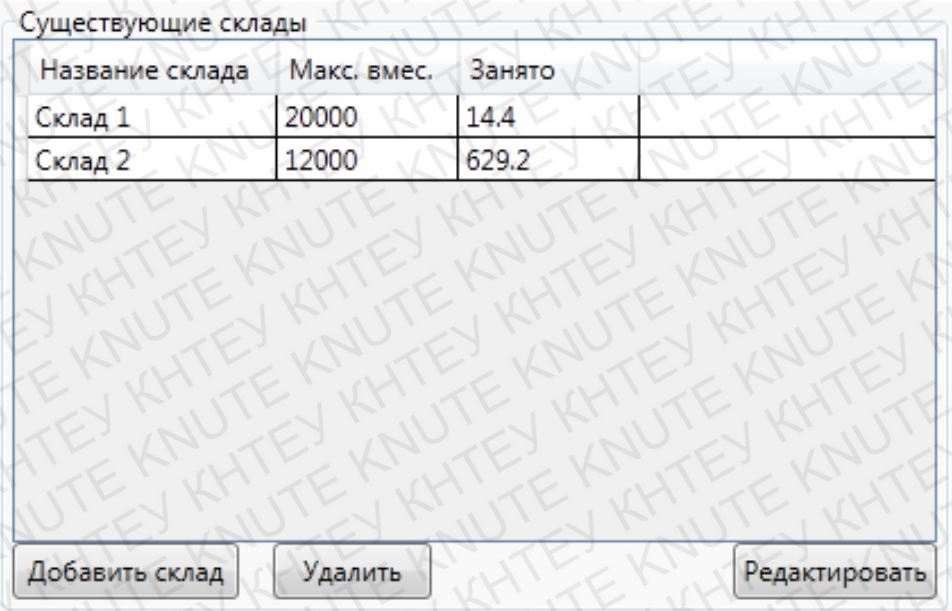

Рис. 3.9. – Елемент «існуючі склади»

На елементі «замовлення», що представлено на рисунку 3.10, відображено таблицю OrdersTable , яка показує які замовлення, на які тори і коли були сформовані, якщо в полях «прийнято» та дата прибуття пусто, то замовлення ще не було виконано. Цю таблицю можна по елементно оновлювати, додавати нові чи видаляти вже існуючі записи, прийнятій товар треба призначити на якийсь з існуючих складів.

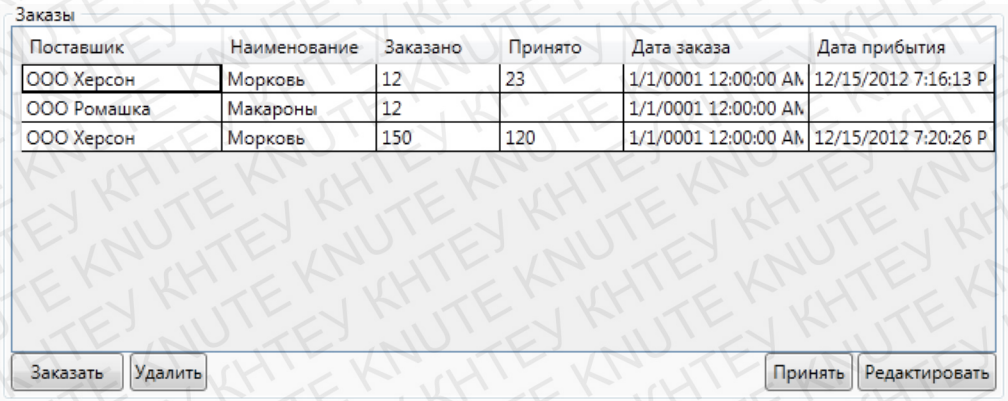

Рис. 3.10. – Елемент «замовлення»

На елементі «діаграма наповненості сховища », що представлено на рисунку 3.11, є комбо бокс де користувач вибирає склад, який буде

оцінюватись. На круговій діаграмі бачимо графічне відношення зайнятого місця на складі до вільного.

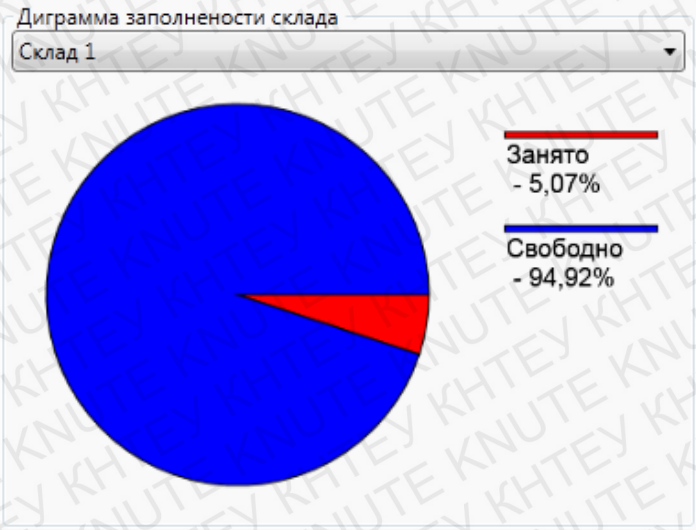

Рис. 3.11. – Елемент «діаграма наповненості сховища»

На елементі «діаграма простору сховищ », що представлено на рисунку 3.12, ми бачимо графічне відношення між максимальним вмістом усіх складів одночасно, де найвищою точкою є вміст найбільшого серед складів.

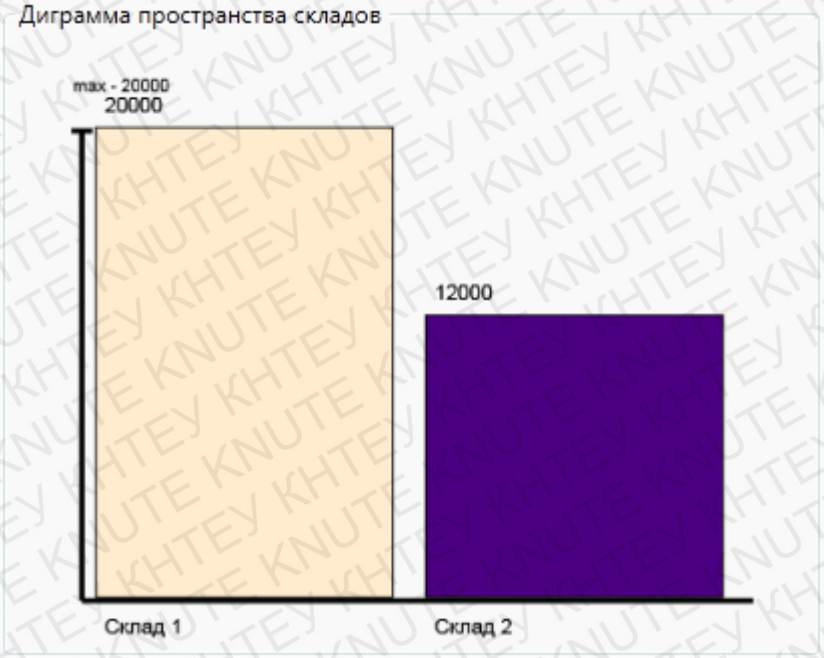

Рис. 3.12. – Елемент «діаграма простору сховищ»

На елементі «діаграма цін на товари», що представлено на рисунку 3.13, ми бачимо графічне відношення між цінами за всю партію товару, серед усіх

товарів з якими працює програма, які зараз є на всіх складах. Цифра що відображається це сума за всі однакові найменування з усіх складів.

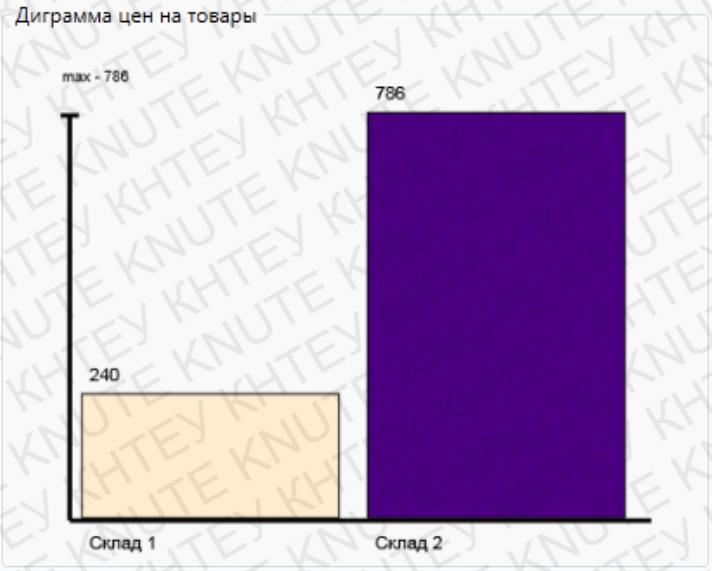

Рис. 3.13. – Елемент «діаграма цін на товари»

Усі елементі що користувач бачить на екрані створені за допомогою візуального редактору Потім файли що описують карту розташування елементів та закладок, і розташовує отриману раніше інформацію на закладках. Щоб всі елементи можна було привести до одного типу, кожен елюент згрупований у GroupBox. Імена закладок та елементи на них користувач може налаштовувати у будь який час. Коли користувач змінює налаштування, програма очищає дані всіх закладок і поново створює закладки з урахуванням змін, після чого завантажує на закладки елементи згідно з новою картою елементів.

3.3. Висновки до розділу 3.

В даному розділі розглянуто графічний інтерфейс програми, а також розроблено інструкцію користувача. При написанні проекту працездатність коду на кожному етапі перевірялась. Результати роботи можна перевірити за показниками ефективності та звітами з підприємства стосовно координатних змін, що сталися

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

У даній роботі був розроблений інтерфейс програмного забезпечення,

для дипломної роботи «Розробка програмного забезпечення логістичного підприємства "Нова Пошта"», у якому був створений прототип користувацького інтерфейсу системи.

Дана програма повністю відповідає поставленому завданню. Після виконання оптимізації інтерфейсу керування програмою стало більш оптимальним, про що свідчить статистика показників ефективності підприємства.

Програма наділена практичним значенням, що визначається достатньо високою універсальністю та полягає у комплексному використанні методів розробки інтерфейсу.

Досліджено методи створення (покращення, оптимізації) інтерфейсу. Описується робота з кольорами довідкою й присвячена оцінці користувальницького інтерфейсу. Система містить велику кількість інформації й виконує велику кількість функцій.

Практична реалізація користувальницьких інтерфейсів із застосування всіх існуючих правил проектування користувальницького інтерфейсу такого рівня, при якому користувач працював із програмним продуктом на такому рівні, щоб всі функції, виконувані в програмі він виконував за допомогою інтерфейсу не замислюючись про ці функції. На основі досліджень розроблено імітаційну модель функціонування АСУ складського обліку логістичного підприємства "Нова Пошта". Створення інформаційної системи управління забезпечило оперативне отримання повної і достовірної інформації та показників споживання ресурсів кожного виду, формуванню планів та звітності по заводу та усіх субспоживачах, пов'язаних між собою, що надало умови для оптимальної організації управління відділом.

					КНТЕУ 121 6-10.БР		
Изм.	Лист	№ докум	Подпись	Дата			
Зав.кафедри		Криворучко О.В.			СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ		
Керівник		Цензура М.О.			*Розробка програмного забезпечення логістичного підприємства "Нова пошта"*	Стадія	Аркуш	Аркушів
Гарант		Цензура М.О.				ВП	48	49
Розробив		Жигулін І.С.						
					Висновки та пропозиції	*Факультет інформаційних технологій, 4 курс, 6 група*		

1. Алексей В. Технология XSLT. - BHV - Санкт-Петербург. - 544 с.

2. Алексей Дубовцев. Microsoft .Net в подлинеке. - БВХ-Петербург, 2004. - 704с.

3. Герберт Шилдт “ Полное руководство C# 4.0” Вильямс 2011 1056 с.

4. Джеффри Рихтер. Программирование на платформе Microsoft .Net Framework. - М: Русская Редакция, 2002. - 512 с.

5. Зовнішнє незалежне оцінювання навчальних досягнень випускників загальноосвітніх навчальних закладів. 2011 р.: Інформаційні матеріали/ Український центр оцінювання якості освіти: Уклад.: І. Л. Лікарчук (наук. ред.) та ін. — К., 2007. — 288 с.

6. Зиборов В. В. "Visual C# 2010 на примерах (+ CD-ROM)" БХВ-Петербург ISBN: 978-5-9775-0698-4 2011 - 380 с.

7. Кент Бек. Экстремальное программировнаие. - Питер, 2002. - 224 с.

8. Кристиан Нейлгел, Билл Ивьен, Джей Глинн, Карли Уотсон, Морган Скиннер, Аллен Джонс. C# 2005 для профессионалов. -М.: Издательский дом "Вильямс", 2006. - 1376 с.

9. Мартин Ф. Архитектура корпоративных программных приложений. - М.: Издательский дом "Вильямс", 2004. - 544 с.

10. Мартин Фаулер. Архитектура Корпоративных программных приложений. - М.: Издательский дом "Вильямс", 2004. - 544 с.

11. Трей Нэш "C# 2010: ускоренный курс для профессионалов" Диалектика-Вильямс ISBN: 978-5-8459-1638-9 2010 – 284 с.

12. Эндрю Тройлсен “ Язык програмированния C# 2010 5-издание” Вильямс 2011 1392 с.

Генеральний директор

Регіональний директор

Відділ логістики

IT-відділ

Відділ служби доставки

Відділ по роботі з

Головний

Фінансисти

Керівник

Керівник служби доставки

Менеджер з TOP-клієнтів

Фахівці

Менеджерслужби доставки

Оператор служби доставки

Менеджер з продажу

Бухгалтер

Відділ термінальної

Логісти

Менеджер з VIP-клієнтів

Бухгалтер відділу 2

Відділ міжміської

Логісти

Водій

Бухгалтер відділу 3