**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

**ІНСТИТУТ ПІСЛЯДИПЛОМНОЇ ОСВІТИ (II В.О.)**



**ЗАЛІКОВА РОБОТА**

*з предмету: «Проектний практикум»*

на тему:

**«Інформаційна система обліку товарів на складі»**

**Виконали:**

слухачі групи ПЗС-11

Гринчук Т. А.

Непомнящий І.А.

**Прийняла:**

доц. Левус Є.В.

ЛЬВІВ – 2014

**Тема:** Інформаційна система обліку товарів на складі

**Розробники:** Гринчук Т.А., Непомнящий І.А.

**Терміни виконання:** 01.12.2013 - 07.03.2014

**ЗМІСТ**

1. Передпроектне обстеження предметної області. UML-діаграма класів ..........................4

2. Опис і обґрунтування вибраного методу організації розробки програмної системи .....5

3. Життєвий цикл розробки програмного продукту. Ідентифікація завдань та дій. Детальний графік робіт. ....................................................................................................................6

4. Опис і обґрунтування вибраного методу/моделі розробки програмної системи……....9

5. Створення (Протопипування) програми у складі групи розробників……………….....11

6. Діаграма прецендентів.........................................................................................................14

7. Обгрунтування доцільності розробки. Презентаційні матеріали....................................15

ВИСНОВКИ………………………………………………………………………………………..16

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ..................................................................................17

Додаток А. Передпроектне обстеження

Додаток Б. Специфікація вимог

Додаток В. Графік та плановий бюджет проекту

Додаток Г. Презентація проекту

1. **Передпроектне обстеження предметної області. UML-діаграма класів**

На основі проведеного передпроектного обстеження предметної області групою розробників було створено UML-діаграму класів програмного продукту (рис. 1.1) та оформлено звіт про результати експрес-обстеження, який винесено в Додаток А даної залікової роботи.



Рис. 1.1. UML-діаграма класів

Як видно з діаграми, на даному етапі життєвого циклу виділяємо батьківські класи «Довідники» та «Документи», які будуть мати атрибути та методи, що наслідуються для дочірніх довідників: Товари, Контрагенти та Склади, а також документів: ПрихіднаНакладна, ВидатковаНакладна. Також, окремо виписуємо глобальний об’єкт ЗалишкиТоварів, що залежить від зміни документів ПрихіднаНакладна та ВидатковаНакладна. На цьому етапі ми також проектуємо можливість довільного виду документа мати скільки завгодно стрічок.

1. **Опис і обґрунтування вибраного методу організації розробки програмної системи**

Загалом виділяють три види моделей організації робіт над розробкою програмної системи: модель потоку робіт, модель потоків даних, рольова модель. Для організації роботи над нашим проектом ми використали модель потоку робіт. Ця модель - показує послідовність дій, виконуваних людьми на різних етапах розробки ПЗ. Для кожної дії вказуються входи, виходи (результати) та зв’язки між ними. Розробка проекту ведеться лише двома людьми, потреба в рольовій моделі, а саме у розподілі виконавців на конкретні ролі відпадає, так як обидва виконавці виконують всі стадії проекту у тісній співпраці та без виділення на конкретні спеціалізації. За допомогою моделі потоків робіт зручно зобразити графік робіт, який закріплений за кожним учасником та що від нього вимагається отримати в кінцевому результаті. Характерним для нашого проекту є те, що він має чітко поставлене технічне завдання, яке не буде змінюватись протягом ходу виконання розробки, або якщо й такі зміни будуть – то вони суттєво не вплинуть на процес. Відповідно ми можемо детально описати етапи розробки проекту та на кожному етапі виділити, які роботи потрібно зробити, яку саме інформацію опрацьовуємо і що в кінцевому результаті плануємо досягнути. Ці дані зручно впорядкувати в таблиці схеми та графіки для використання, як і всередині команди так і демонстрації ходу виконання робіт замовнику.

1. **Життєвий цикл розробки програмного продукту. Ідентифікація завдань та дій. Детальний графік робіт.**

Життєвий цикл розробки нашого продукту складається з наступних етапів:

1. **Специфікація вимог.** Провівши передпроектне обстеження діяльності підприємства-замовника, опрацювавши відповідну документацію, проконсультувавшись з керівниками даного підприємства та опитавши відповідальних працівників, було створено детальну специфікацію вимог до програмного продукту, яка міститься в Додатку Б.
2. **Аналіз.** Основним завданням програмного продукту є створення ефективного інструменту управлінського обліку товарів на складах підприємства. Проаналізувавши поточне програмне забезпечення, яке на даний працює на підприємстві, ми, група розробників, зробили висновок, що оптимальним шляхом досягнення поставлених цілей буде доробка та модифікація інформаційної бази «1С: Бухгалтерія» в частині управлінського обліку товарів, інтегрувавши нові модулі та структури даних в поточну систему. Альтернативний варіант, який розглядався було: впровадження нової інформаційної системи, яка би містила в собі як і бухгалтерський так і фінансовий обліки, але після ретельних розрахунків вартості, виявилось, що цей варіант не є економічно виправданим.
3. **Проектування.** Для виконання проекту в архітектуру інформаційної системи, перш за все додаємо регістр «Товари», на якому будуть накопичуватись залишки товарів в розрізі типів обліку, номенклатурних одиниць, складів та серій (рис. 3.1).

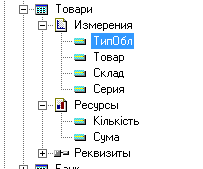


Рис. 3.1. Регістр «Товари»

Також додаємо в систему документи: ТовариПрихід – купівля товарів у постачальника та оприходування їх на склад, ТовариВідвантаження – продажа товарів покупцю та списання залишку, ТовариПовернення – повернення товарів покупцем, ТовариПоверненняПостачальнику – відповідно повернення купленого товару назад постачальнику, АктІнвентаризації – інвентаризація та введення початкових залишів товарів (рис. 3.2).

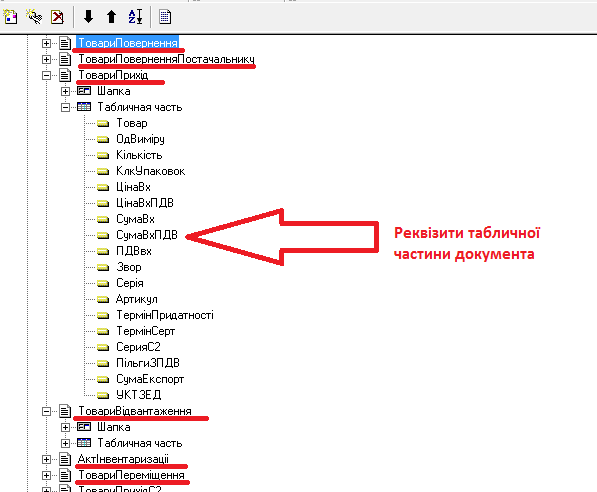


Рис. 3.2. Нові документи

Для того щоб користувачам системи було зручно аналізувати поточний стан залишків на руху товарів на складах, потрібно доробити звіти (рис. 3.3), відповідно до специфікації.

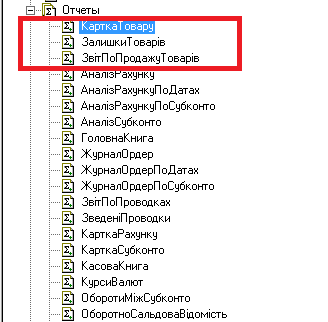


Рис. 3.3. Нові звіти

Концептуальна діаграма взаємозв’язків цих об’єктів між собою, була раніше зображена в розділі 1, на рис. 1.1 (сторінка 4).

1. **Реалізація.** Розробка проекту здійснена на платформі 1С 7.7. Приклад модуля документа ТовариПрихід, з фрагментом коду проведення документа по таблиці регістра залишків зображено на рис. 3.4:

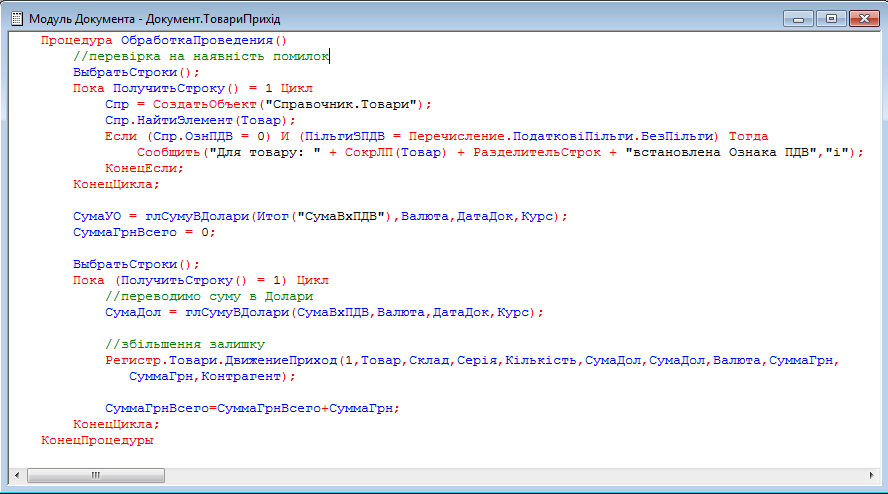


Рис. 3.4. Модуль проводок документа ТовариПрихід

1. **Тестування та дослідна експлуатація.** Тестування проекту попередньо здійснюється командою розробників, після чого проект запускається на етап «Дослідної експлуатації» на стороні замовника. Таким чином виявляються помилки і надсилаються на виправленні команді розробника, який зобов’язаний в найкоротший термін їх виправити.
2. **Промислова експлуатація і супровід.** На цьому етапі ми запускаємо відтестований проект в «Промислову експлуатацію» для роботи на «живих даних». Всі помилки, які виникають на даному етапі також ліквідовує розробник, додаткові побажання користувачів реалізовуються на їхнє звернення за додаткову плату, яка не входить у вартість проекту.

Графік робіт та плановий бюджет проекту наведено в Додатку В.

1. **Опис і обґрунтування вибраного методу/моделі розробки програмної системи**

В загальному випадку, життєвий цикл визначається моделлю й описується у формі методології (методу). Модель або парадигма життєвого циклу визначає загальну організацію і, як правило, основні його фази та принципи переходу між ними. Методологія (метод) визначає комплекс робіт, їх детальний зміст і рольову відповідальність спеціалістів на всіх етапах вибраної моделі.

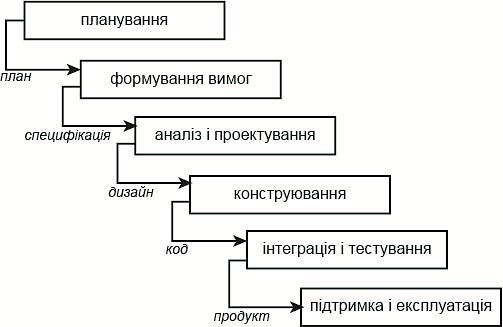
Життєвий цикл програмного забезпечення супроводжується розробленням, обігом та використанням програмної документації.

Модель життєвого циклу - це структура, що складається із процесів, робіт та задач, які включають в себе розробку, експлуатацію і супровід програмного продукту; охоплює життя системи від визначення вимог до неї до припинення її використання. На сьогодні найбільшого розповсюдження набули дві моделі:

- каскадна модель;

- спіральна модель.

Однією з перших почала застосовуватися каскадна модель, де кожна робота виконується один раз і в такому порядку, який подано на рис. 4.1

Рис. 4.1. Каскадна модель

Тобто вважається, що кожна робота має бути виконана настільки ретельно, що після її закінчення і переходу до наступного етапу, повертатися до попереднього не буде потреби. Розробник перевіряє проміжний результат відомими методами верифікації і фіксує його як готовий еталон для наступного процесу.

Згідно з даною моделлю роботи і завдання процесу розроблення зазвичай виконуються послідовно, як це наведено у схемі.

Проте допоміжні і організаційні процеси (контроль вимог, керування якістю і ін.), як правило, виконуються разом з процесами розробки ПЗ. У даній моделі повернення до початкового процесу передбачається після супроводження і виправлення помилок.

Особливість такої моделі полягає у фіксації послідовних процесів розроблення програмного продукту. В її основу покладена модель фабрики, де продукт проходить стадії від задуму до виробництва, потім його передають замовнику у вигляді готового виробу, де заміна не передбачена, хоча можна подати інший подібний виріб. Недоліки цієї моделі такі:

– процес створення ПС не завжди вкладається в таку жорстку форму і послідовність дій;

– не враховуються змінювані потреби користувачів, нестабільні умови зовнішнього середовища, які впливають на зміни вимог до ПС під час ї розроблення;

Оскільки дана модель, в силу своїх особливостей, добре підходить для розробки ПЗ в галузі бухгалтерського обліку, нарахування ЗП та всіх галузях де продукт розробляється кожен раз по подібній схемі, вибираємо її.

1. **Створення (Протопипування) програми у складі групи розробників**

Розробка програми велася згідно запланованого графіка виконання, зображеного в Додатку В. Процес розробки вівся на термінальному сервері, таким чином, що учасники проекту мали змогу бачити результати роботи один-одного. Хід виконання завдань розміщався в програмі баг-трекері Mantis (рис. 5.1).



Рис. 5.1. Вигляд задачі в програмі програмі баг-трекері Mantis

У ході проекту були розроблені форми та макети документів та звітів, ось деякі з них:

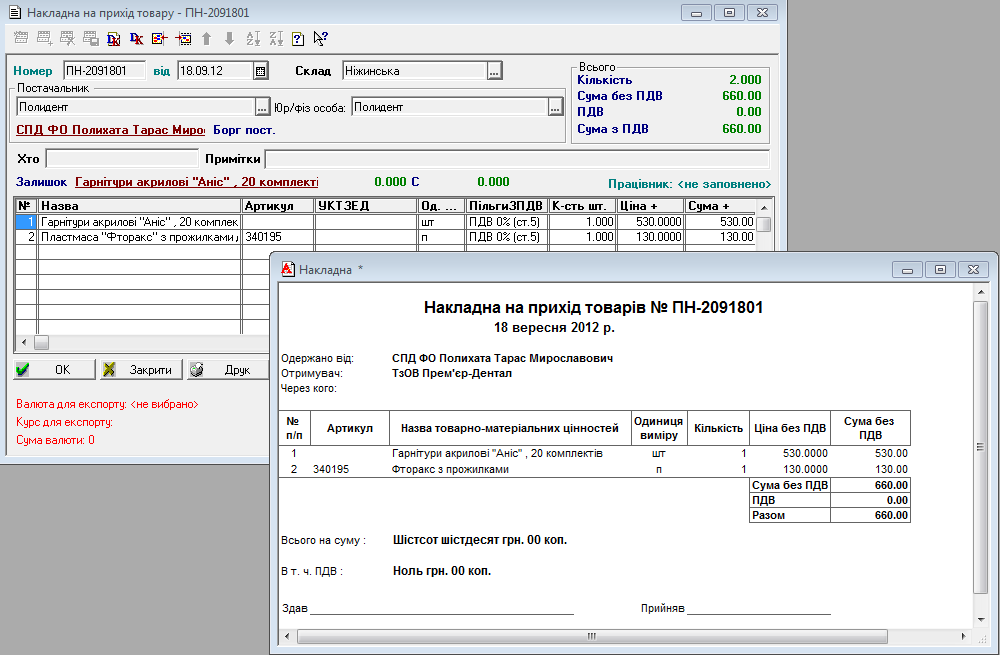


Рис. 5.2. Прихідна накладна

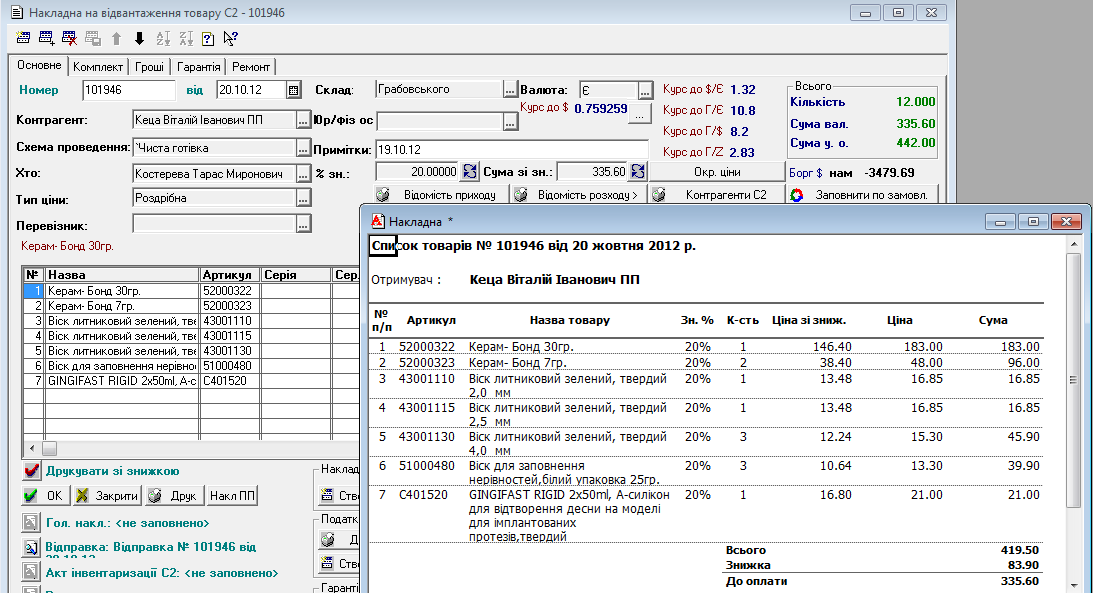


Рис. 5.3. Видаткова накладна

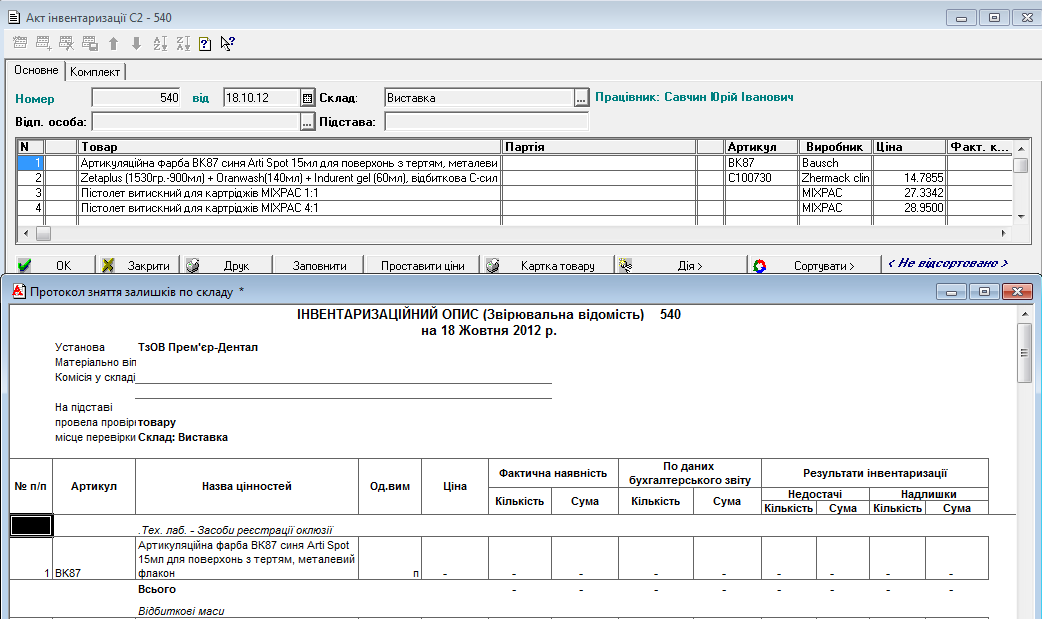


Рис. 5.4. Акт інвентаризації

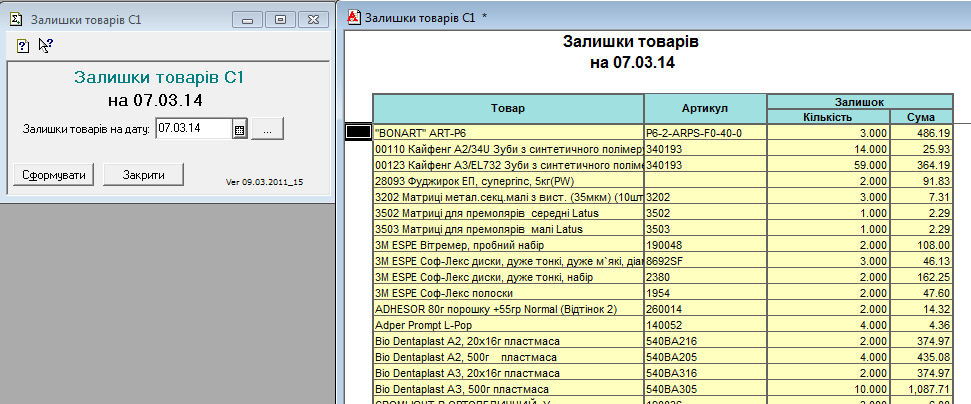


Рис. 5.5. Залишки товарів

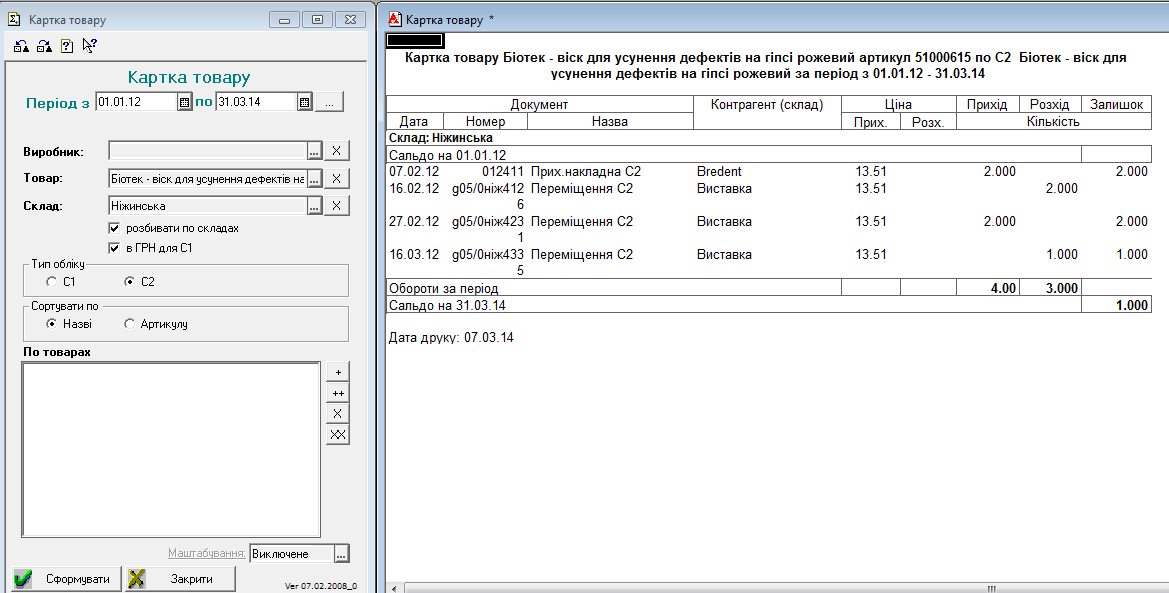


Рис. 5.6. Картка товару

1. **Діаграма прецендентів**

Процес опрацювання документації користувачем зручно зобразити на наступній діаграмі прецендентів (рис. 6.1):

Рис. 6.1. Діаграма прецендентів

Як бачимо, типовий користувач системи (не з повними правами) має можливості: проводити документи, друкувати їх та видаляти. Кожне проведення веде за собою перевірку залишку та чи дата документа оперативна. Якщо одна з цих перевірок не виконалась – документ не проводиться. Видаляти також можна лише документи оперативно (тільки поточної дати), інакше не виконається перевірка на дату документа і видалення не відбудеться. При проведенні є можливість також роздрукувати макет документа (якщо цього захоче користувач).

1. **Обгрунтування доцільності розробки. Презентаційні матеріали.**

Основна ціль проекту є автоматизація управлінського обліку товарів на складах, що суттєво відрізняється від бухгалтерського і подальша їхня інтеграція в одній базі даних. Вартість проекту становить 28750 грн. Альтернативою цього проекту, могла би бути інтеграція аналогічної системи, але з вже готовим функціоналом. Схожий функціонал має програмний продукт: «1С: Управління виробничим підприємством», але його ціна без додаткових ліцензій і вартості інтеграції його на підприємстві становить понад 40 тис. грн. В такому випадку робимо однозначний висновок, що наш проект є доцільним, як з функціональної так економічної точки зору.

Презентація проекту міститься у Додатку Г.

**ВИСНОВКИ**

На даній проектній роботі ми навчилися моделювати процес командної розробки програмного продукту на всіх фазах його життєвого циклу. Також поглибили свої навики у складанні проектної документації, оформленні презентацій до проекту, вирішенні конфліктів у групах.

Дана робота допомогла нам відчути нам різні ситуації розробки програмного забезпечення, які є близькими до реальних, що допоможе в майбутньому на практиці краще організовувати робочий процес групи ІТ-фахівців.

**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. Якобсон А., Буч Г., Рамбо Дж. Унифицированный процесс разработки программного обеспечения. – СПб.: Питер, 2002. – 498 с.:ил.

2. Гарсиа-Молина Г., Ульман Дж., Уидом Дж. Системы баз данных. Полный курс.: Пер.с англ. – М.: Издательский дом “Вильямс”, 2004. – 1088с.: ил.

3. Кузнецов С.Д. Концептуальное проектирование реляционных баз данных с использованием языка UML // Новые публикации, 15.03.2008. – www/CITForum.ru.

4. Буч Г., Рамбо Дж., Якобсон А Язык UML: Руководство пользователя. – М.: ДМК, 2000. – 356 с.: ил.

5. Мюллер Р.Дж. Базы данных и UML. Проектирование / Первод. с англ. Е. Молодцова. – М.: Издательство “Лори”,2002. – 432с.: ил.

6. Гордеев А.В., Молчанов А.Ю. Системное программное обеспечение. – С.-Пб.:Питер, 2001. – 740 с.: ил.