МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук

Кафедра інформаційних технологій

ДИПЛОМНА РОБОТА

Рівень вищої освіти бакалавр

на тему: Розробка автоматизованої системи

складського обліку зернових культур

Виконала студентка 4 курсугрупи К-44

Напрям підготовки 6.050101

Компютерні науки

Зубенко Сергій Сергійович

Керівник асистент

Шуптар Наталія Йосипівна

Консультант к.геогр.н.,доцент

Кузніченко Світлана Дмитрівна

Рецензент к.ф-м.н.,доцент

Зінкевич Яніна Сергіївна

Одеса 2017

МІНІСТЕРСТВО ОСВІТИ І НАУКИУ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Комп'ютерних наук
Кафедра Інформаційних технологій
Рівень вищої освіти бакалавр
Напрям підготовки 6.050101 Комп'ютерні науки
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри _____
_____
"____" _____2018р.

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Зубенко Сергій Сергійович
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка автоматизованої системи складського обліку зернових культур

керівник роботиасистент Шуптар Наталія Йосипівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "____"_____2018р. №____
2. Строк подання студентом роботи травня 2017
3. Вихідні дані до роботи
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз предметної області.
2. Специфікація вимог.
3. Проектні та технічні рішення.
_____
_____

5. Перелік графічного матеріалу
_____
_____
_____
_____

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання“____”_____2016 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту	Термін виконання етапів проекту	Оцінка виконання етапу	
			у %	за 4-х бальною шкалою
1	Аналіз предметної області	15.03-22.03	90	
2	Специфікація вимог	22.03-15.04	90	
3	Проектні та технічні рішення	15.04-30.05	100	
4	Оформлення пояснювальної записки	30.05-10.06	100	
	Інтегральна оцінка виконання етапів календарного плану (як середня по етапам)			

Студент _____ Зубенко С.С.
(підпис) (прізвище та ініціали)

Керівник проекту _____ Шуптар Н.Й.
(підпис) (прізвище та ініціали)

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

ІС – інформаційна система
БД – база даних
СУБД – система управління базами даних
XAML - eXtensible Application Markup Language
WPF - Windows Presentation Foundation
CLR – Common Language Runtime
CLI – Common Language Intermediate

ВСТУП

Впровадження засобів обчислювальної техніки в повсякденне життя людини, доступність інформації, обсяг і швидкість її обробки стають вирішальними факторами розвитку виробничих сил держави, науки, культури та усіх сфер життєдіяльності. Інформація та дані все частіше розглядаються як життєво важливі ресурси, які повинні бути організовані таким чином, щоб ними можна було легко користуватися.

Сучасні господарські відносини, розвиток конкуренції, зумовлюють необхідність нових підходів до матеріального забезпечення господарських процесів на підприємстві, які покликані забезпечити безперервність функціонування господарської одиниці. Збільшення операцій з придбання, укомплектування, відвантаження сировини, продукції, товарів та інших товарно-матеріальних цінностей, вимагає забезпечення відповідними складськими площами. Дана ситуація вимагає розвитку і ефективного управління складським господарством.Саме складське господарство покликане забезпечити якісне зберігання матеріальних цінностей; формування повного асортименту товарів для споживачів; ефективність дослідження і освоєння нових сегментів ринку; організацію виробництва і роботи транспорту; поліпшення використання територій підприємства; зниження простоїв транспортних засобів і транспортних витрат. Таким чином, в сучасних умовах господарювання виникає необхідність в ефективній організації складського обліку на підприємствах України.

Незважаючи на необхідність складського господарства, більшість керівників не приділяють належної уваги організації складського обліку, який призводить до прийняття низькоефективних управлінських рішень, які призводять до низьких показників фінансових результатів діяльності підприємства. Складський облік сьогодні займає вагоме місце в системі організації бухгалтерського обліку, як інформаційної системи управління підприємствомТак, складський облік займає важливе місце у виробничій діяльності підприємства. Він знаходиться на перетині організації бухгалтерського обліку господарських процесів придбання, виробництва, реалізації, і виконує функцію збереження товарно-матеріальних цінностей (сировини, товарів, готової продукції)

Діяльність агро фірми містить у собі велику кількість напрямків, кожен з яких потребує детального документування роботи. Практично кожен бухгалтер у повсякденній роботі зустрічається з різними наборами

документів, такими як, наприклад, накладна, супровідний лист, звіт.На збір та упорядкування усієї документації яка потрібна для складання звітності за відповідний період витрачається багато часу.

Для уникнення подібних проблем, а також для зручності , пропонується створення єдиної складської системи документообігу агро фірми.

В цілому, система має включати, весь документообіг валового збору зернових культур.

1 ХАРАКТЕРИСТИКА ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ ДЛЯ ПРОЕКТУВАННЯ

1.1 Аналіз існуючих аналогів системи

Для розробки оптимального програмного продукту що задовольнить потреби агро фірми «Бургуджи», були проаналізовані існуючи аналоги.

Були розглянуті найбільш популярні системи, а саме: «Контур.Бухгалтерия», «1С Склад», «Парус», «Галактика ERP»та «MicrosoftDynamicsCRM».

1.1.1 Опис «Контур.Бухгалтерия»

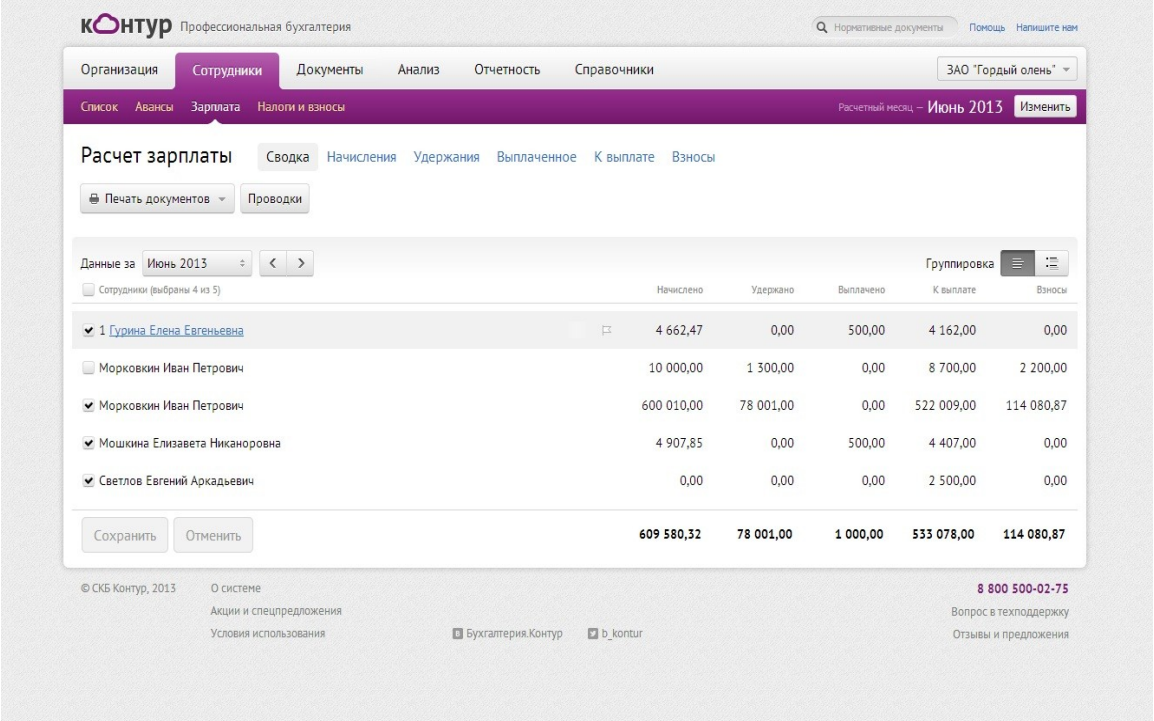

Рисунок 1.1.1 Контур бухгалтерія

Бухгалтерський онлайн сервіс для ІП та ТОВ. Дозволяє вести облік доходів і витрат, формувати і роздруковувати звітність, нагадує, коли її здавати і надає довідкову інформацію. Є можливість інтеграції з інтернет-банком. Дозволяє здавати звітність через Інтернет. Дозволяє організувати спільну роботу з Бухгалетером.

За словами розробників Контур.Бухгалтерії, найбільш затребуваною для користувачів опцією мобільного додатка є виставлення первинних

документів. Для рахунків на оплату така можливість вже була реалізована. А тепер в мобільному додатку можна створювати акти. Це зручно, особливо якщо надаються послуги нароботі чи вдома у замовника: для цього достатньо ввести номер і дату, вибрати контрагента і заповнити список товарів і послуг.

Недоліком такого продукту є необхідність забезпечення постійного підключення до інтернету. Навряд чи можливо вести бухгалтерію на лоні дикої природи або під час сильної грози, але в інших випадках проблем з інтернет-з'єднанням зазвичай не виникає. Звичайно, є ситуації, коли перебої в мережі заважали підприємствам вчасно здати звітність. Але найчастіше це траплялося в регіональному масштабі, і контролюючі органи продовжували термін здачі звітності, або оператор звертався в ИФНС і захищав своїх клієнтів. Для роботи з хмарної бухгалтерією не потрібна висока швидкість інтернету, цілком вистачить популярного сьогодні мобільного 3G.

Програми не завжди коректно працюють в браузерах.Це стосується непопулярних або недавно оновлених рішень при конфлікті версій. Розробники хмарних сервісів регулярно тестують свої продукти на сумісність з популярними браузерами. Так що питання знімається установкою якісної програми: Google Chrome, Opera, Mozilla Firefox.

Інколи користувач вирішує відмовитися від роботи в хмарної бухгалтерії і хоче забрати свої дані. Деякі системи дозволяють вивантажити частину даних, але часом зробити це неможливо. Сьогодні хмарні сервіси мають різні стандарти вивантаження даних, але ситуація розвивається, і питання про стандартизацію буде вирішуватися в найближчому майбутньому.

Деякі користувачі бояться, що хтось може побачити їх бухгалтерію: конкуренти, податкова, шахраї. Однак інформація знаходиться в захищеному дата-центрі в зашифрованому вигляді. Дані знеособлені і представлені у вигляді цифр, які розкидані по різних таблицях. Щоб уникнути втрати або пошкодження, проводиться багаторазове резервне копіювання, копії зберігаються на рознесених серверах. Крім того, відомості нікуди не передаються до державних контролюючих органів, для цього немає законних підстав.

1.1.2 Опис «1С Склад»

Спочаткудодаток «1С: Склад» було створено для автоматизації бухгалтерського та управлінського обліків (включаючи нарахування

зарплати і управління кадрами), але сьогодні цей продукт знаходить своє застосування в областях, далеких від власнебухгалтерських. Технологічна платформа «1С: Склад» являє собою програмну оболонку над базою даних. Використовуються бази на основі DBF-файлів в 7.7, власний формат 1CD з версії 8.0 або СУБД Microsoft SQL Server на будь-який з цих версій. Крім того, з версії 8.1 зберігання даних можливо в PostgreSQL і IBM DB2, а з версії 8.2 додалася і Oracle. Платформа має свій внутрішній мову програмування, що забезпечує, крім доступу до даних, можливість взаємодії з іншими програмами за допомогою OLE і DDE, в версіях 7.7, 8.0 і 8.1 - за допомогою COM-з'єднання. Клієнтська частина платформи функціонує в середовищі MicrosoftWindows, а починаючи з версії 8.3, також в середовищі Linux і MacOSX. Починаючи з версії 8.1, серверна частина платформи в клієнт-серверному варіанті роботи «1С: Склад» може функціонувати на ОС MicrosoftWindows.

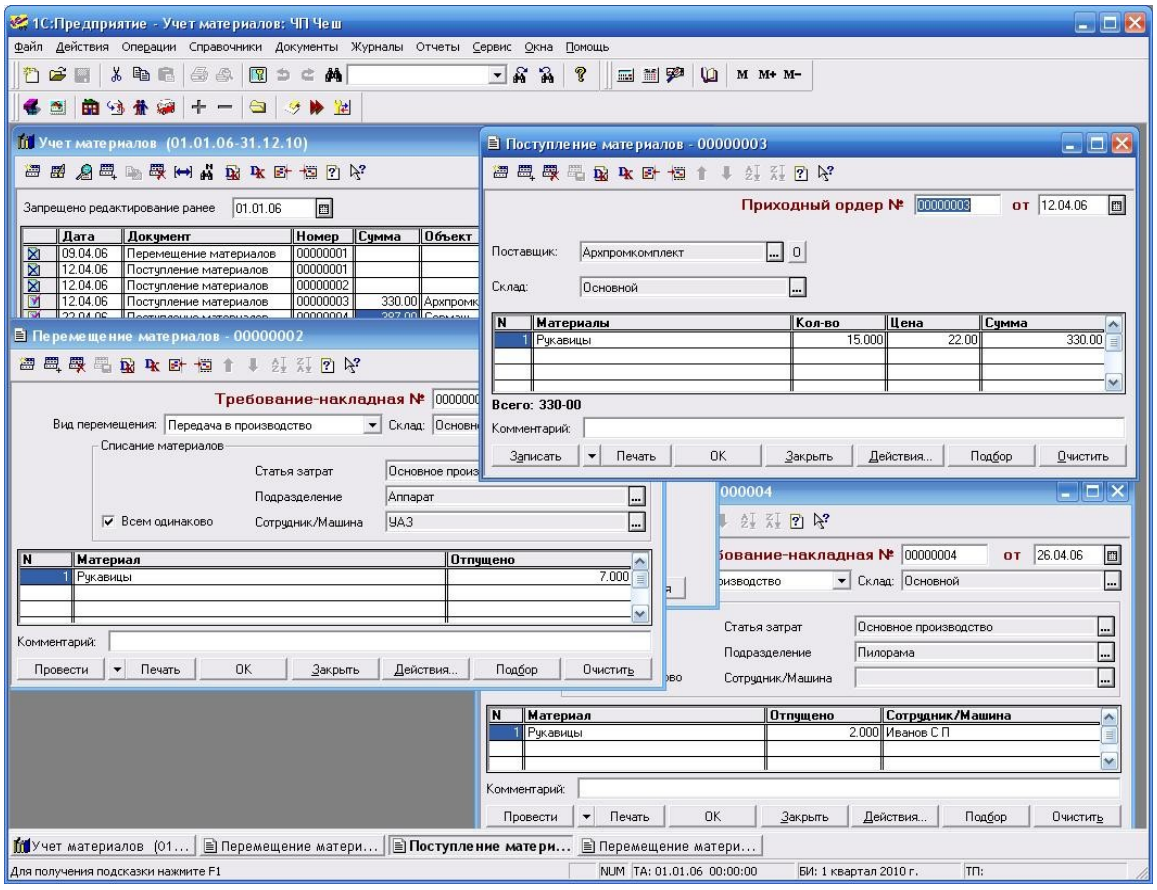

Рисунок 1.3 «1С Склад»

Спочатку «1С: Склад» було призначене для автоматизації бухгалтерського та управлінського обліків (включаючи нарахування

зарплати і управління кадрами), але сьогодні цей продукт знаходить своє застосування в областях, далеких від власнебухгалтерських. Технологічна платформа «1С: Склад» являє собою програмну оболонку над базою даних. Використовуються бази на основі DBF-файлів в 7.7, власний формат 1CD з версії 8.0 або СУБД Microsoft SQL Server на будь-який з цих версій. Крім того, з версії 8.1 зберігання даних можливо в PostgreSQL і IBM DB2, а з версії 8.2 додалася і Oracle. Платформа має свій внутрішній мову програмування, що забезпечує, крім доступу до даних, можливість взаємодії з іншими програмами за допомогою OLE і DDE, в версіях 7.7, 8.0 і 8.1 - за допомогою COM-з'єднання. Клієнтська частина платформи функціонує в середовищі MicrosoftWindows, а починаючи з версії 8.3, також в середовищі Linux і MacOSX. Починаючи з версії 8.1, серверна частина платформи в клієнт-серверному варіанті роботи «1С: Склад» може функціонувати на ОС MicrosoftWindows.

В даний час найпоширенішою програмою для автоматизації бухгалтерського та управлінського обліку в Україні є продукти, компанії 1С "1С: Бухгалтерія", "1С: Склад", "1С: Управління торгівлею". Незважаючи на високу популярність продуктів 1С, ставлення користувачів до них подвійне: багато фахівців вважають програми 1С кращими з розроблених засобів для автоматизації бухобліку та управлінського обліку, також є досить велика кількість фахівців, які вважають програми, розроблені компанією 1С "сирим" продуктом, що має велике кількість недоробок, недоліків і помилок Якщо оцінювати програми, серії "1С" об'єктивно, то їх не можна назвати ні поганими, ні хорошими. Давати оціночну характеристику програмі "1С" без прив'язки до конкретно взятому підприємству просто не можна. Наприклад, для багатьох компаній «1С Склад» - це ідеальне рішення, яке повністю задовольняє їх вимоги щодо автоматизації обліку. Точно також для багатьох компаній дана програма абсолютно не прийнятна, оскільки з її допомогою неможливо автоматизувати їх бізнес-процеси і вирішити поставлені завдання. Оскільки наша компанія пропонує рішення для автоматизації різних торгових точок (магазини, міні-маркети, кіоски), закладів сфери послуг (ресторани, кафе, бари, спорт-зали), промислових об'єктів, розглянемо переваги і недоліки програми 1С саме в такому розрізі. До недоліків рішень на базі 1С відносять: Основне застосування 1С: рішення задач автоматизації податкового та бухгалтерського обліку. Базові продукти 1С орієнтовані в першу чергу на автоматизацію завдань бухгалтерського і податкового обліку ("1С: Бухгалтерія", "1С: Склад", "1С: Управління торгівлею") на

підприємствах всіх сфер діяльності. Для роботи об'єктів роздрібної торгівлі, сфери послуг, спеціалізованих обєктів такі рішення не підходять в принципі.

Багато компаній, що спеціалізуються на розробці під 1С, створюють свої конфігурації, орієнтовані на різні сфери бізнесу. Вартість таких продуктів 1С на порядок вище ніж базові версії 1С, а функціонал необхідно буде доопрацьовувати, доплачувати за налаштування, підтримку і т.д. Звязок одного продукту 1С з іншим (наприклад, зв'язок" 1С: Склад "" 1С: Управління торгівлею") здійснюється за допомогою операцій обміну інформацією (вивантаження / завантаження документів). Дуже багато програм для автоматизації бізнесу мають в своєму складі модулі для обміну даними з платформою 1С. у зв'язку з цим, на торгових об'єктах часто практикується робота двох різних програм. Як правило, бек-офіс - це продукти 1С, а фронт-офіс вирішення інших розробників Істотна різниця між поняттями «готова система» і «платформа». На кожному підприємстві використовується унікальне рішення «1С», реалізоване не налаштуванням конфігурації, а запрограмоване на мові програмування 1С, яка вбудована в платформу. тобто, при покупці будь-якого рішення на основі 1С, ви купуєте тільки "платформу 1С", яку доведеться налаштовувати, дописувати, і допрацьовувати під ваші вимоги. А такі послуги, виливаються в додаткові витрати, і як правило - незаплановані. Для великих підприємств - це нормальна практика, так як для ефективної роботи таких організацій, потрібні продукти, які по впровадженню нових технологічних процесів проектуються індивідуально. У випадку з невеликими підприємствами, такий підхід до автоматизації є трудомістким і витратним, як за часом, так і з матеріальних ресурсів. Налаштування, впровадження і запуск повинні здійснюватись кваліфікованим 1С-програмістом. Всі базові конфігурації "1С" можна назвати "напівфабрикатами", і вони, відповідно, пропонуються за ціною напівфабрикатів. Реальну вартість одного робочого місця можна оцінити тільки тоді, коли закінчаться доопрацювання або повне переписування вихідного коду куплених модулів. У таких випадках Вам не обійтися без послуг компаній, які спеціаліруются на налаштуванні 1С. Або ж доведеться працювати з 1С-програмістом "фрілансером" або наймати на роботу фахівця, здатного написати новий функціональний модуль або знайти і коректно виправити помилки в написаному до нього коді.

Компанія 1С досить часто випускає оновлення до своїх продуктів, що є платним. Це пов'язано, як із змінами в законодавстві України так і з виправлення помилок у роботи цього програмного продукту.

1.1.3 Опис «Парус»

«Па́рус» — серія програмних продуктів ПП «Парус», призначена для автоматизації бізнес-задач підприємств малого та середнього бізнесу, великих корпорацій та холдингів, а також бюджетних установ і організацій.

Програмні продукти «Парус» широко використовуються не лише в державних організаціях й в установах України, а й комерційних компаніях.

Парус-Підприємство 7 — комплексна система для автоматизації управління підприємствами малого та середнього бізнесу різної галузевої спрямованості. Система створена з використанням Visual FoxPro та має модульну структуру. Система дозволяє автоматизувати всі облікові та низку управлінських задач підприємств: управління відносинами з клієнтами (CRM), ведення бухгалтерського обліку, управління логістикою, розрахунок та нарахування заробітної плати, управління персоналом, управління конкурсними закупівлями тощо.

Охоплювати автоматизацію наступних задач: бюджетування, управління фінансами, управління виробництвом, здійснення планування та обліку на виробництві, ведення бухгалтерського та податкового обліку, управління логістикою, розрахунок та нарахування заробітної плати, управління діловими процесами, управління автотранспортом, управління персоналом, ведення обліку виробничих нарядів, управління технічним обслуговуванням і ремонтами, управління конкурсними закупівлями, управління відносинами з клієнтами (CRM), автоматизація call-центрів та ін.

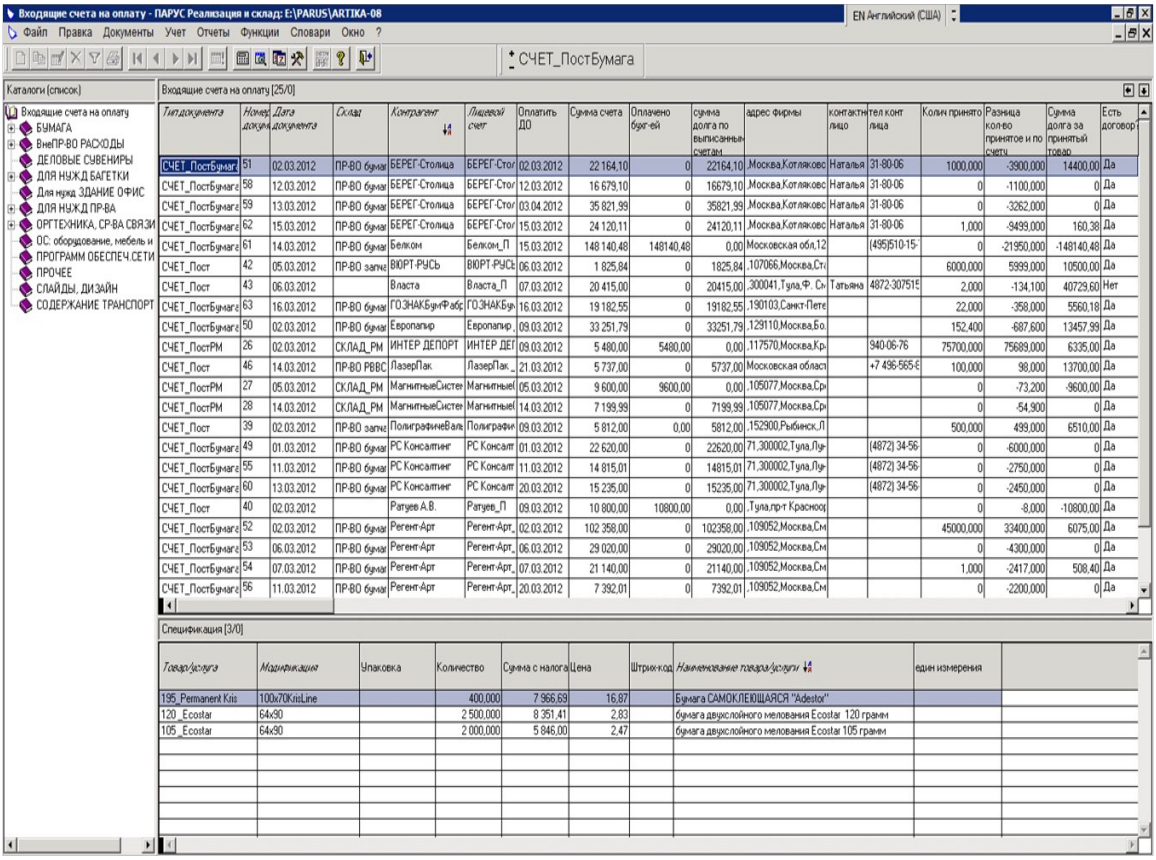

Рисунок 1.4 «Парус»

Крім явних переваг програми «Парус» існують також деякі недоліки і незручності. Наприклад, Парус 4х по своїй суті закрита система і не може бути змінена користувачем. Тільки розробники мають право проводити модифікацію базових модулів і налаштовувати їх до специфіки конкретного підприємства. Така послуга досить дорого коштує, і часто викликає труднощі оновлення версії. Також вартість цієї програми визначається вартістю одного робочого місця, і зі збільшенням кількості користувачів відповідно зростає. Так, при кількості користувачів від 20 до 40 осіб, вартість може бути досить високою. Крім ціни, освоєння програми також може викликати деякі труднощі, тому як правильно працювати там зможуть тільки висококваліфіковані працівники.

1.1.3 Опис «Галактика ERP»

Комплексна система управління підприємством Галактика ERP - є ядром комплексу бізнес-рішень Галактика BusinessSuite, головне призначення якого - виконання в єдиному інформаційному просторі типових

і спеціалізованих завдань управління підприємством, холдингом, групою компаній в умовах сучасної економіки. Система адресована середнім і великим підприємствам і володіє широкою функціональністю для інформаційної підтримки всього спектру завдань стратегічного планування і оперативного управління.

Комплекс Галактика Business Suite на основі передових інформаційних технологій забезпечує рішення:

1) всього спектра управлінських завдань підприємства відповідно до концепції ERP;
2) задач підтримки прийняття управлінських рішень на базі визначення, планування, досягнення і аналізу ключових показників діяльності підприємства.

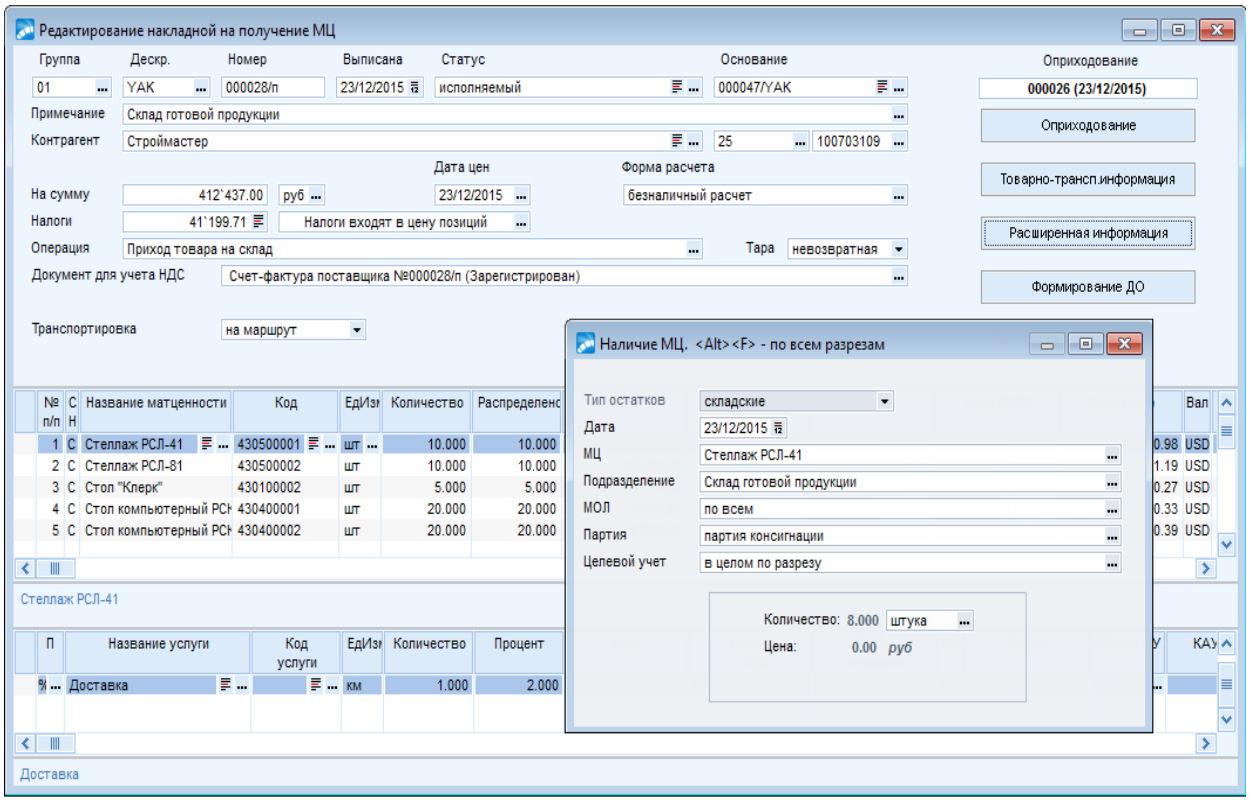

Рисунок 1.5 «ГалактикаERP»

Можливості системи Галактика ERP доповнюються і розширюються за допомогою рішень на платформі Галактика Ranet, таких як Галактика BusinessIntelligence , яке є готовий продукт і може бути легко адаптований під потреби конкретного замовника. Можливості системи Галактика ERP можуть бути розширені і за рахунок створення замовником на платформі

Галактика Ranet власних компонентів інформаційної системи підприємства, які доповнять / замінять модулі системи Галактика ERP.

Система Галактика ERP розроблена для застосування в умовах вітчизняної економіки з її яскраво вираженою специфікою, постійними змінами законодавства. Саме тому Галактика ERP - дієвий інструмент побудови ефективної системи управління незалежно від спадів і підйомів економіки і внутрішніх змін на підприємстві.

Система Галактика ERP має модульну структуру. Кожен модуль призначений для автоматизації окремих, вузьких завдань. Це дає можливість замовникам купувати тільки потрібну конфігурацію. Гнучка модульна система відкриває можливість побудови і використання будь-якої конфігурації, оптимально відповідає конкретним потребам підприємства. При збільшенні кількості користувачів зберігаються продуктивність системи.

В системі реалізована концепція компонентної моделі: все одиниці системи сформовані в компоненти, які взаємодіють між собою через спеціальні інтерфейси, компоненти логічно об'єднані в модулі. Наявність версій у компонентів дозволяє перейти від оновлення системи до оновлення окремих компонентів, що мінімізує витрати замовника.

Апробовані технології розгортання проектів автоматизації сприяють тому, що впровадження erp-системи проходить в стислі терміни, з фіксованим бюджетом і мінімальними для підприємства ризиками. Це дозволяє підприємствам-замовникам швидко окупити витрати на IT. Розробка і впровадження системи Галактика ERP ведеться з використанням самих передових технологій і інструментів.

Галактика ERP - підтримує відкриті стандарти розробки (XML, COM, ActiveX, ODBC), що в свою чергу дає можливість інтеграції системи з будь-яким спеціалізованим або галузевим програмним забезпеченням, офісними додатками. Додаткові можливості для інтеграції Галактики ERP з продуктами сторонніх виробників і побудови глобальних розподілених систем дає реалізація в системі сервіс-орієнтованої архітектури (SOA) і технологій web-сервісів.

До складу системи Галактика ERP включені кошти для централізованої налаштування параметрів системи, її оновлення, установки необхідних додатків. Це істотно підвищує безпеку системи, покращує захист від несанкціонованого доступу та значно полегшує процес її адміністрування.

Недоліки системи, незважаючи на заявлену на першій сторінці свого опису правильну мета роботи підприємства і завдання впровадження системи на ньому, реально галактикане забезпечує виконання цієї мети. Система не є керуючою. Вона не реалізує алгоритмів формування оптимальних запитів на виробництво та / або постачання в залежності від стану попиту, планів, прогнозів або їх комбінації. Впровадження її не приносить конкретної прибутку.

Система не має механізму визначення і контролю процедур виконання конкретних операцій або групи операцій (наприклад, визначення процедури ПОСТАЧАННЯ: спосіб формування заявки - заявка - вибір постачальника - формування замовлення - відстеження його виконання - процедура отримання на склад), що не дозволяє керівнику бути впевненим, що його управлінські рішення виконуються.

Система не має функцій, необхідних для забезпечення діяльності великих корпорацій (Централізоване постачання, розподіл функцій між організаціями, передача повноважень від однієї організації до іншої, взаєморозрахунки всередині корпорації і т.д.)

Система, практично, не є інтегрованою. Більшість модулів практично не пов'язане між собою, а їх зв'язок з фінансами дуже умовна, тому що документи в фінансовому модулі вводяться вручну на підставі первинних документів, що призводить до розбіжності в матеріальному і фінансовому обліку.

Система практично не має аналітики в Головній Книзі (рахунок, субрахунок, код аналітичного обліку, який невідомо як використовується). Дана система обліку не дозволяє на підставі фінансових даних побудувати більш-менш глибокий фінансовий аналіз.

Система не контролює бюджет при введенні оперативних документів і взагалі не має механізмів прогнозування руху грошових коштів, що неприпустимо при управлінні підприємством.

Основним призначенням системи є максимально швидке формування всіх необхідних проводок для закриття місяця і формування звітності до податкової інспекції.

1.2 Опис «MicrosoftDynamicsCRM»

MicrosoftDynamicsCRM - CRM-система, розроблена компанією Microsoft. Це потужний інструмент для управління взаємовідносинами з

клієнтами. Він підвищує продуктивність співробітників всередині і поза організацією та полегшує взаємодію відділів продажів, маркетингу і обслуговування клієнтів за допомогою сучасних технологій, інтегрованих в єдине робочий простір.

Ключові результати використання Microsoft Dynamics CRM: зниження вартості залучення нових клієнтів, висока якість маркетингових даних і можливість аналізу повернення маркетингових інвестицій, скорочення циклу та вартості продажу, управління продажами, збільшення кількості закритих операцій, збільшення продажів існуючим клієнтам, зниження вартості обслуговування клієнтів, підвищення їх задоволеності і лояльності.

Часто перед організаціями постає завдання автоматизації бізнес-процесів, які не знаходять відображення в стандартних IT-рішеннях, таких як ERP, CRM або галузевих інформаційних системах. Прикладами таких процесів можуть служити управління дилерською мережею, робота з громадянами та організаціями в державних структурах, управління корпоративним навчальним процесом, взаємини між компанією і здобувачами в процесі найму і прийому на роботу, управління взаємовідносинами з постачальниками і багато іншого.

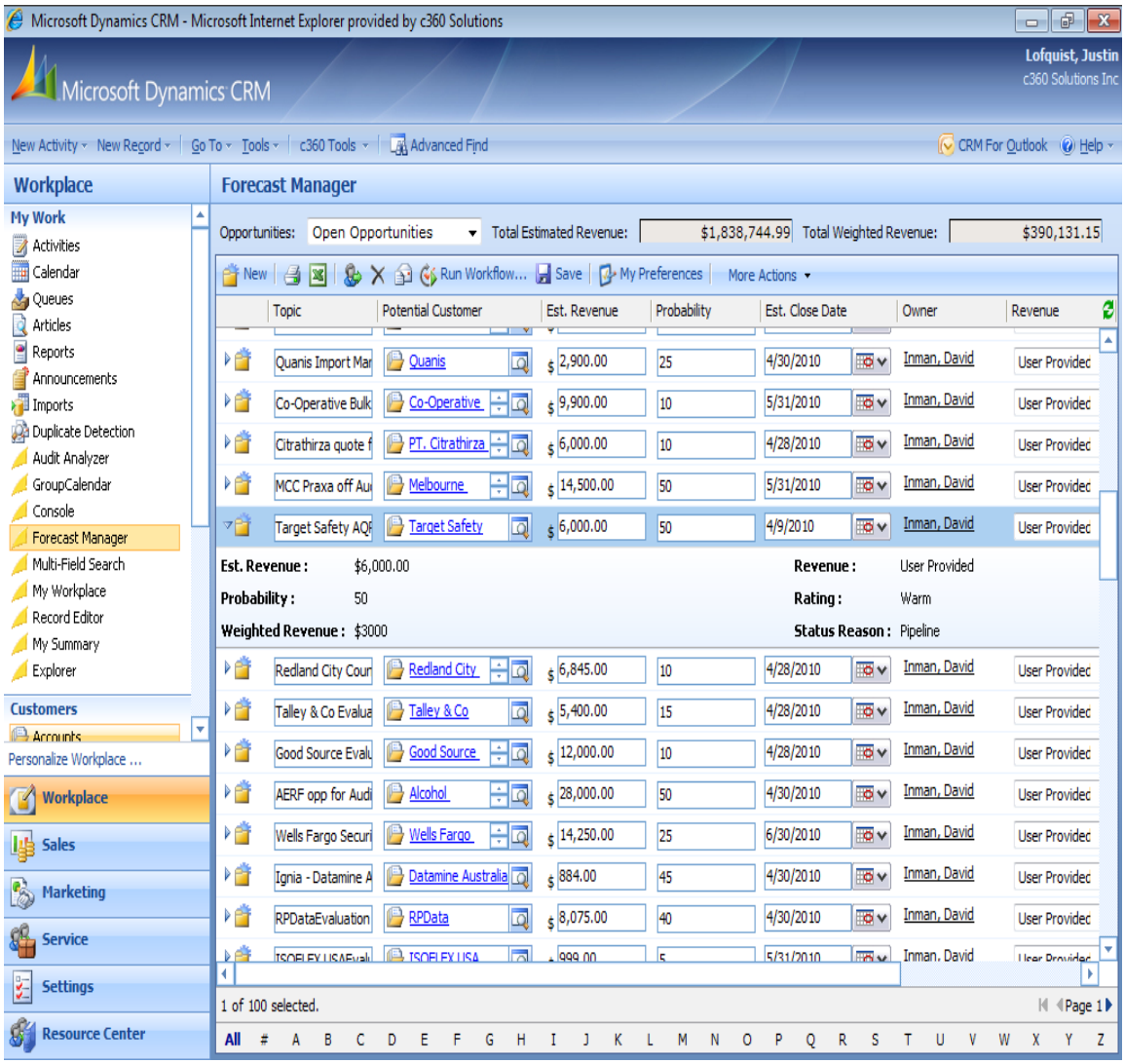

Рисунок 1.6 «MicrosoftDynamicsCRM»

MicrosoftDynamicsCRM охоплює всі галузі менеджменту: виробництво та дистрибуцію, ланцюжки поставок і проекти, фінанси та засоби бізнес-аналізу, взаємовідносини з клієнтами та персоналом.

Ключовими перевагамиMicrosoftDynamics є:

Підвищення ефективності та продуктивності роботи співробітників компанії

- система реалізує стандартні принципи роботи продуктів Microsoft і не вимагає тривалого навчання;

- висока ергономічність рольових користувальницьких інтерфейсів і рольових центрів;

- забезпечити пріоритетність виконання поточних завдань;

- єдиний інтерфейс при доступі з робочого місця і через Інтернет;

- спеціалізовані засоби бізнес-аналізу і звітності дозволяють аналізувати дані на основі збалансованих показників, засобами MicrosoftOfficeExcel і MicrosoftSharePointServer зі службами PerformancePointServices;

- повноцінна робота з MicrosoftDynamics безпосередньо з додатків MicrosoftOffice.

Оперативне управління змінами та розвиток конкурентних переваг:

- забезпечення достовірної інформації для швидкого прийняття правильних рішень;
- зменшення витрат на складання фінансової звітності та аналіз;
- ефективне управління грошовими потоками;
- зручний інструментарій для стратегічного планування;
- підвищення рівня обслуговування клієнтів за рахунок більш ефективної організації процесу продажів;
- оптимізація закупівель та складських запасів;
- мінімізація циклу і гнучке виробниче планування;
- можливість організації внутрішнього і зовнішнього сервісного центру;
- ефективне управління кваліфікацією і розвитком персоналу.

Оптимізація управління територіально розподіленої компанією:

- система підтримує локальні вимоги більше 40 країн і забезпечує роботу більш ніж на 40 мовах;

- консолідація фінансової, операційної та клієнтської інформації в єдиному центрі;

- підтримка необмеженої кількості компаній в територіально розподіленій структурі організації;

- зручний механізм поширення уніфікованих бізнес-процесів на підрозділи і представництва компанії;

- організація оперативного і зручного доступу до важливої інформації через Інтернет, MicrosoftSharePointServer, служби PerformancePointServices і продукти MicrosoftOffice.

Для яких клієнтів підходить MicrosoftDynamics

- середнє або велике підприємство до 10 тис. співробітників;

- потреба в автоматизації: від 20 до 1000 одночасно працюючих користувачів;

- складні і специфічні бізнес-процеси (підприємства з розподіленою структурою, холдинги, дистриб'юторські і виробничі компанії, що працюють в сфері послуг, і т. Д.).

MicrosoftDynamics - це масштабується рішення, яке дозволяє вибрати ефективну конфігурацію, що враховує характеристики каналів зв'язку, архітектуру серверів і робочих станцій: трирівневу конфігурацію, роботу через Інтернет або термінальний доступ. Система підтримує стратегію захищених інформаційних систем (Trustworthy Computing), що гарантує надійність і безперебійну роботу.

2 ВИБІР ТА ОБГРУНТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ, ТА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ ЇЇ РЕАЛІЗАЦІЇ

2.1 Вибір системи управління базами даних

Microsoft SQL Server — комерційна система керування базами даних, що розповсюджується корпорацією Microsoft. Мова, що використовується для запитів — Transact-SQL, створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO щодо структурованої мови запитів SQL із розширеннями. Використовується як для невеликих і середніх за розміром баз даних, так і для великих баз даних масштабу підприємства. Багато років вдало конкурує з іншими системами керування базами даних.

Базовий код MS SQL Server (до версії 7.0) ґрунтувався на коді Sybase SQL Server. Це дозволило Microsoft вийти на ринок баз даних для підприємств, де конкурували Oracle, IBM, і, пізніше, сама Sybase. Microsoft, Sybase і Ashton-Tate спочатку об'єдналися для створення і випуску на ринок першої версії програми, що отримала назву SQL Server 1.0 для OS/2 (близько 1989 року), яка фактично була еквівалентом Sybase SQL Server 3.0 для Unix, VMS та ін. Microsoft SQL Server 4.2 був випущений у 1992 році та входив до складу операційної системи Microsoft OS/2 версії 1.3. Офіційний реліз Microsoft SQL Server версії 4.21 для ОС Windows NT відбувся одночасно з релізом самої Windows NT (версії 3.1). Microsoft SQL Server 6.0 був першою версією SQL Server, створеною виключно для архітектури NT і без участі в процесі розробки Sybase.

До того часу, як вийшла на ринок ОС Windows NT, Sybase і Microsoft розійшлися та створювали вже власні моделі цього програмного продукту. Microsoft намагалася отримати виняткові права на всі версії SQL Server для Windows. Пізніше Sybase змінила назву свого продукту на Adaptive Server Enterprise щоб уникнути плутанини з Microsoft SQL Server. До 1994 року

Microsoft отримала від Sybase три повідомлення про авторські права як натяк на походження Microsoft SQL Server.

Протягом подальших шести років корпорація Microsoft працювала над вдосконаленням вже існуючої версії SQL Server 2000 доки не збудувала зручнішу систему Microsoft SQL Server 2005. Були вдосконалені продуктивність, кліентські інструменти інтегрованого середовища розробки, а також у декількох додаткових системах, що встановлюються разом із SQL Server 2005. Змінено: інструментарій процесів керування сховищами даних (SQL Server Integration Services або SSIS), сервер звітів, сервер OLAP та інтелекутального аналізу даних (Analysis Services), а також декілька технологій повідомлень, особливо Service Broker та Notification Services.

Microsoft робить SQL Server доступним у різноманітних варіантах, які різняться наборами властивостей в залежності від цілей кінцевого користувача. Це такі редакції як:SQL Server Compact Edition (SQL CE). Компактне видання—вкладений механізм бази даних. Завдяки малому обсягу (2 Мб для DLL) має зменшені властивості у порівнянні з іншими варіантами. Розмір бази даних обмежений 4 Гб і не може використовуватися як служба Windows.

SQL Server Express Edition. Раніше відомий під назвою MSDE (Microsoft SQL Server Desktop Engine), Microsoft SQL Server Express є вільно поширюваною версією SQL Server. Дана версія має деякі технічні обмеження, також відсутні графічні інструменти адміністрування. Такі обмеження роблять її непридатною для розгортання великих баз даних. В основному вона використовується у застосунках, при проектуванні, або для самостійного вивчення. Розмір бази даних обмежений 4 Гб, розмір пам'яті, що може бути адресованою—1 Гб, підтримує лише один процесор.

Microsoft SQL Server як мову запитів використовує версію SQL, що отримала назву TRANSACT-SQL, яка є реалізацією SQL-92 (стандарт ISO для SQL) з багатьма розширеннями. T-SQL дозволяє використовувати додатковий синтаксис процедур, що зберігаються і забезпечує підтримку транзакцій (взаємодія бази даних з керуючим застосунком). Microsoft SQL Server та Sybase ASE для взаємодії з мережею використовують протокол рівня застосунка під назвою Tabular Data Stream (TDS, протокол передачі табличних даних).

Microsoft SQL Server також підтримує Open Database Connectivity (ODBC)—інтерфейс взаємодії застосунків з СУБД. Версія SQL Server 2005 надає можливість підключення користувачів через веб-сервер-сервіси, що

використовують протокол SOAP. Це дозволяє клієнтським програмам, не призначеним для Windows, кроссплатформенно з'єднуватися з SQL Server. Microsoft також випустила сертифікований драйвер JDBC, що дозволяє застосункам під керування Java (таким як BEA і IBM Websphere) з'єднуватися з Microsoft SQL Server 2000 і 2005.

SQL Server підтримує дзеркалювання та кластеризацію баз даних. Кластер серверу SQL—це сукупність однаково конфігурованих серверів; така схема допомагає розподілити робоче навантаження між декількома серверами. Усі сервери мають одне віртуальне ім'я, а дані розподіляються за IP-адресами машин кластеру протягом робочого циклу. Також у разі відмови або збою на одному з серверів кластеру доступне автоматичне перенесення навантаження на інший сервер.

SQL Server підтримує надлишкове дублювання даних за трьома сценаріями:

- знімок: Виконується «знімок» бази даних, який сервер відправляє одержувачам.
- історія змін: Всі зміни бази даних безперервно передаються користувачам.
- синхронізація з іншими серверами: Бази даних декількох серверів синхронізуються між собою. Зміни усіх баз даних відбуваються незалежно на кожному сервері, а під час синхронізації відбувається звірка даних. Дублювання такого типу передбачає можливість вирішення протиріч між базами даних.

SQL Server 2005 має вбудовану підтримку .NET Framework. Завдяки цьому, процедури бази даних, що зберігаються, можуть бути написані на будь-якій мові платформи .NET з використанням повного набору бібліотек, доступних для .NET Framework. На відміну від інших процесів, .NET Framework виділяє додаткову пам'ять і будує засоби керування SQL Server, не використовуючи вбудовані засоби Windows. Це підвищує продуктивність порівняно із загальними алгоритмами Windows, оскільки алгоритми розподілу ресурсів спеціально налагоджені для використання у структурах SQL Server.

Microsoft та інші компанії пропонують велику кількість програмних засобів розробки, які дозволяють розробляти застосунки для бізнесу з використанням баз даних Microsoft SQL Server. Microsoft SQL Server 2005 включає також Common Language Runtime (CLR) Microsoft .NET, що

дозволяє застосункам, розробленим на мовах платформи .NET (наприклад, VB.NET або C#), реалізовувати процедури, що зберігаються та різні функції.

До переваг можна віднести незалежність від конкретної СУБД. Не зважаючи на наявність діалектів і відмінностей в синтаксисі, більшість текстів SQL-запитів, що містять, DDL і DML, можуть бути досить легко перенесені з однієї СУБД в іншу. Існують системи, розробники яких спочатку орієнтувалися на застосування щонайменше кількох СУБД (наприклад: система електронного документообігу *Documentum* може працювати як з Oracle, так і з Microsoft SQL Server та IBM DB2). Природно, що при застосуванні деяких специфічних для реалізації можливостей, такого рівня перенесення дуже важко досягти.

Наявність стандартів і наборів тестів для виявлення сумісності та відповідності конкретній реалізації SQL загальноприйнятому стандарту тільки сприяє «стабілізації» мови. Щоправда, слід звернути увагу на той факт, що сам по собі стандарт місцями занадто формалізований і має завеликі розміри, наприклад, Core-частина стандарту SQL:2003 містить понад 1300 сторінок тексту.

Декларативність за допомогою SQL програміст описує лише дані, які потрібно витягнути або модифікувати. Те, яким чином це зробити, вирішує СУБД безпосередньо при обробці SQL-запиту. Не слід вважати, що це повністю універсальний принцип — програміст описує набір даних для вибірки або модифікації, проте йому при цьому корисно уявляти, як СУБД інтерпретуватиме текст його запиту.

До недоліків продукту можна віднести:

1) невідповідність реляційної моделі даних
2) SQL не є істинно реляційною мовою.

Хоча SQL, на етапі створення, була запланована як засіб роботи кінцевого користувача, врешті-решт вона стала настільки складною, що перетворилася на інструмент програміста.

Не зважаючи на наявність міжнародного стандарту ANSI SQL-92, багато компаній, що займаються розробкою СУБД (наприклад, Oracle, Sybase, Microsoft, MySQL), вносять зміни до мови SQL, вживаної в розроблених ними СУБД. Цим, вони створюють передумови відступу від стандарту. Завдяки такій діяльності, для кожної конкретної СУБД з'являються специфічні діалекти мови SQL.

Складність роботи з ієрархічними структурами раніше SQL не пропонувала стандартного способу маніпуляції деревовидними структурами. Деякі постачальники СУБД запропонували свої рішення. Для прикладу, Oracle використовує вираз CONNECT BY. В наш часяк стандарт прийнята рекурсивна конструкція WITH.

Елементи мовиSQL складається з таких елементів мови:

1) положення, які є складовими компонентами заяв і запитів.
2) предикати для тризначної логіки (3VL) (так / ні / невідомо), або логічних значень, які використовують для обмеження наслідків заяв та запитів, або зміни ходу виконання програми.
3) запити для повернення скалярних величин або таблиць, що складаються зі стовпчиків та рядків даних.
4) запити вилучення даних на основі певних критеріїв.
5) заяви впливу на схеми та дані зв'язків.
6) заяви керуванням транзакціями, ходом виконання програми, завданнями та діагностикою.

2.2 Зберігання даних

В даний час одним з найбільш широко поширених форматів для зберігання картографічної інформації є файли формату SHP (ESRI Shapefile). Однак вони не дуже зручні у використанні, тому що в стандарті не визначені механізми пошуку і вибірки об'єктів за певними критеріями. Хоча і самі розробники і сторонні фірми роблять спроби побудови спеціальних індексних файлів для таких цілей. Одним з таких безкоштовно розповсюджуються компонентів є SharpMap. Він дозволяє здійснювати всі необхідні дії при роботі з геоінформаційними даними, однак як показали тести об'єм споживаної пам'яті і продуктивність залишають бажати кращого.

Геоінформаційна система передбачає можливості зміни масштабу карти, а також переміщення і пошуку картографічних об'єктів, що викликає ряд технічних проблем, наприклад, завантаження, ще не завантажених областей, вивантаження неактуальною інформації і т.п. Попередні тести показали неприйнятну продуктивність при використанні бібліотеки SharpMap, тому було вирішено зробити імпорт картографічної інформації в СУБД. В якості СУБД на первинному етапі було вирішено використовувати СУБД - Microsoft SQL Server.

2.3 Призначення, загальна характеристика та об'єкти СУБД MS SQL Server

Система управління базами даних (СУБД) - сукупність програмних і лінгвістичних засобів загального або спеціального призначення, що забезпечують управління створенням і використанням баз даних.

- управління даними у зовнішній пам'яті (на дисках);
- керування даними в оперативній пам'яті з використанням дискового кешу;
- журнализация змін, резервне копіювання і відновлення бази даних після збоїв;
- підтримка мов БД (мова визначення даних, мова маніпулювання даними);

У загальних рисах, базу даних можна розглядати з двох точок зору - користувача і системи бази даних. Користувачі бачать базу даних як набір логічно пов'язаних даних, а для системи баз даних це просто послідовність байтів, які зазвичай зберігаються на диску. Хоча це два повністю різні погляди, між ними є щось спільне: система баз даних повинна надавати не тільки інтерфейс, що дозволяє користувачам створювати бази даних і витягувати або модифікувати дані, але також системні компоненти для управління збереженими даними. Тому система баз даних повинна надавати такі можливості:

- фізичну незалежність даних;
- логічну незалежність даних;
- оптимізацію запитів;
- цілісність даних;
- управління паралелізмом;
- резервне копіювання і відновлення;
- безпеку баз даних.
- різноманітні інтерфейси;

Microsoft SQL Server - система керування базами даних (РСУБД), розроблена корпорацією Microsoft. Основний використовуваний мову запитів - Transact-SQL, створений спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO по структурованого мови запитів (SQL) з розширеннями. Використовується для роботи з базами даних розміром від персональних до великих баз даних масштабу підприємства; конкурує з іншими СУБД в цьому сегменті ринку.

SQL – мова реляційної бази даних. Мова реляційної бази даних в системі SQL Server називається Transact-SQL. Це різновид самої значимої на сьогоднішній день мови бази даних - мови SQL (Structured Query Language - мова структурованих запитів). Походження мови SQL тісно пов'язане з проектом, званим System R, розробленого і реалізованого компанією IBM ще на початку 80-х років минулого століття. За допомогою цього проекту було продемонстровано, що, використовуючи теоретичні основи роботи Едгара Ф. Кодда, можливе створення системи реляційних баз даних.

На відміну від традиційних мов програмування, таких як C #, C ++ і Java, мова SQL є безліч-орієнтованим (set-oriented). Розробники мови також називають її запис-орієнтованим (record-oriented). Це означає, що в мові SQL можна запитувати дані з декількох рядків однієї або декількох таблиць, використовуючи всього лише одну інструкцію. Це одне з найбільш важливих переваг мови SQL, що дозволяє використовувати цю мову на логічно більш високому рівні, ніж традиційні мови програмування.

Іншою важливою властивістю мови SQL є її непроцедурність. Будь-яка програма, написана на процедурній мові (C #, C ++, Java), крок за кроком описує, як виконувати певне завдання. На противагу цьому, мову SQL, як і будь-яка інша непроцедурна мова, описує, що хоче користувач. Таким чином, відповідальність за знаходження відповідного способу для задоволення запиту користувача лежить на системі.

Мова SQL містить два под'язика: мова опису даних DDL (Data Definition Language) і мову обробки даних DML (Data Manipulation Language). Інструкції мови DDL також застосовуються для опису схем таблиць баз даних. Мова DDL містить три загальні інструкції SQL: CREATE, ALTER і DROP. Ці інструкції використовуються для створення, зміни та видалення, відповідно, об'єктів баз даних, таких як бази даних, таблиці, стовпці та індекси.

На відміну від мови DDL, мова DML охоплює всі операції з маніпулювання даними. Для маніпулювання базами даних завжди застосовуються чотири загальні операції: вилучення, вставка, видалення і модифікування даних (SELECT, INSERT, DELETE, UPDATE).

2.4 Entity Framework - технологія роботи з базою даних

Для побудови баз даних був використаний Entity Framework. Він є продовженням технології Microsoft ActiveX Data і надає можливість роботи з

базами даних через об'єктно-орієнтована код C #. Цей підхід надає ряд істотних переваг: немає необхідності ає турбуватися про код доступу до даних, немає необхідностізнати деталі роботи СУБД SQL Server і синтаксису мови запитів T-SQL, замість цього ви працюєте з таблицями бази даних як з класами C #, з полями цих таблиць - як з властивостями класів, а синтаксис SQL-запитів, який в ADO.NET раніше потрібно було вставляти в код C # у вигляді команд, замінений на більш зручний підхід з LINQ. Entity Framework бере на себе обов'язки по перетворенню коду C # в SQL-інструкції. При роботі з Entity Framework надаються величезні можливості по створенню моделі бази даних за допомогою інтегрованого середовища розробки (IDE) Visual Studio.

2.4.1 Структура Entity Framework

Модель EDM. Entity Framework акцентує свою увагу на моделюванні, в якому можна побачити використання діаграми ER (entity-relationship, "сутність-відношення"), підхід з використанням логічного і фізичного проектування шарів і багато іншого. Ядром Entity Framework є модель EDM, суть якої полягає в зберіганні сутностей (entity) в вигляді суворо типізованих класів, а не у вигляді об'єктів схеми бази даних. Модель EDM (рис. 3.1) дозволяє забезпечити зв'язок між сутнісними класами в коді і таблицями бази даних.

Архітектура Entity Framework в абстрактному сенсі заснована на шарах (layers): робочий, віддалений і сполучний.

Класи з кодом сутностей містяться в робочому шарі, в якому працюють програмісти. Залежно від того, який підхід ви використовуєте (Code-First або DB-First), робочий шар може бути змодельований або за допомогою графічного дизайнера Visual Studio, або за допомогою коду. Після цього у програмістів з'являється широкий інструментарій для роботи з Entity Framework. Синтаксис робочого шару описується за допомогою мови Conceptual Schema Definition Language (CSDL).

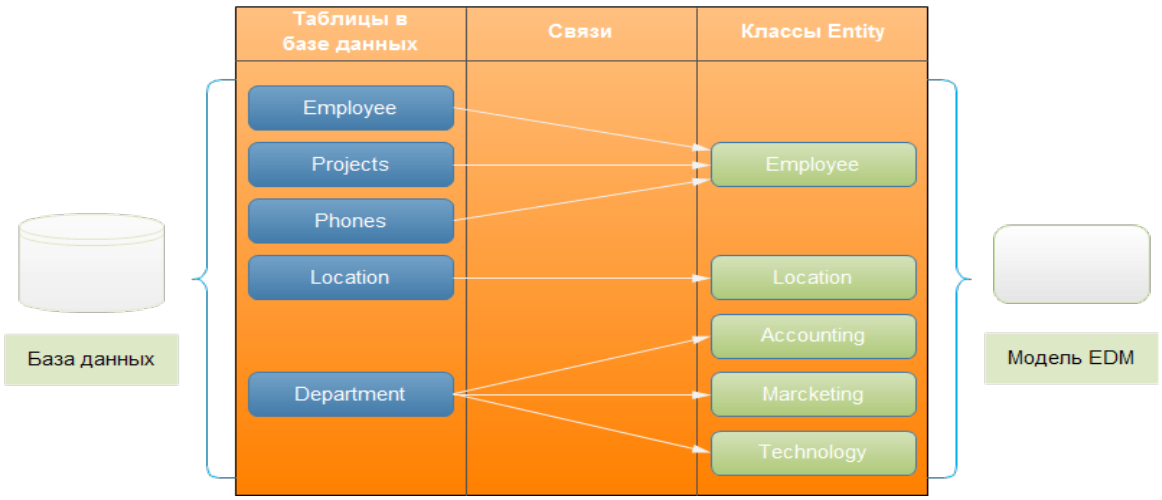

Рисунок 2.1 – Модель EDM

Віддалений шар визначає таблиці, стовпці, рядки, відносини між таблицями бази даних. Синтаксис віддаленого шару описується за допомогою мови Store Schema Definition Language (SSDL).

Сполучний шар визначає відповідність між робочим і віддаленим шарами, він пов'язує властивості сутнісного класу в робочому шарі за допомогою стовпців таблиці бази даних в віддаленому шарі. Керувати цим шаром (тобто деталями прив'язки) можна з вікна Mapping Details знаходиться в інструментах дизайнера Visual Studio або за допомогою анотацій Fluent API, якщо ви працюєте з підходом Code-First. Мова Mapping Specification Language (MSL) визначає синтаксис сполучного шару.

Важливо відзначити, що мови CSDL, SSDL і MSL мають синтаксис XML, але при цьому використовують різну семантику.

Файли Entity Framework

Всі файли, які використовуються в Entity Framework засновані на синтаксисі XML. Використання XML робить файли простими й універсальними для інших додатків. Також XML-файли читабельні для людини - ви можете в будь-який момент відкрити і переглянути вміст цих файлів. Проте, кожен елемент Entity Framework використовує різні файли XML з різним розширенням.

Після того як створено новий додаток, який спирається на Entity Framework і додано сутнісні класи бази даних, можна побачити результуючі файли в основній папці проекту (рис. 3.2). У середовищі Visual Studio 2012 створюється єдиний файл Entity Data Model XML (.EDMX), хоча в більш

старих версіях Visual Studio можливо буде створено декілька файлів (один для кожної сутності).

Файл EDMX містить в собі кілька секцій, написаних на мовах CSDL, SSDL і MSL, що представляють різні верстви в архітектурі Entity Framework. Щоб відкрити вихідний код цього файлу в Visual Studio, клацніть по ньому правою кнопкою миші і виберіть в контекстному меню Open With ..., після чого в діалоговому вікні виберіть варіант XML Editor.

2.4.2 Підходи з проектування бази

Database-First підходить для проектувальників баз даних - спочатку ви створюєте базу даних за допомогою різних інструментів (наприклад, SQL Server Management Studio), а потім генеруєте EDMX-модель бази даних (надає зручний графічний інтерфейс для взаємодії з базою даних у вигляді діаграм і об'єктну модель у вигляді класів C #). В даному випадку вам потрібно працювати з SQL Server і добре знати синтаксис T-SQL, але при цьому не потрібно розбиратися в C #.

Model-First. Підходить для архітекторів - спочатку ви створюєте графічну модель EDMX в Visual Studio (в фоновому режимі створюються класи C # моделі), а потім генеруєте на основі діаграми EDMX базу даних. При цьому підході не потрібно знати ні деталей T-SQL ні синтаксису C #.

Code-First. Підходить для програмістів - при даному підході модель EDMX взагалі не використовується і ви вручну налаштовуєте класи C # об'єктної моделі (даний підхід підтримує як генерацію сутнісних класів з існуючої бази даних, так і створення бази даних зі створеної вручну моделі об'єктів C #). Очевидно, що це підходить для програмістів, добре знайомих з синтаксисом C #.

У проекті при роботі з Entity Framework був обраний підхід Code-First.

Створення бази даних Code-First

Основу функціональності Entity Framework складають класи, що знаходяться в просторі імен System.Data.Entity. Серед усього набору класів цього простору імен слід виділити наступні:

- DbContext: визначає контекст даних, який використовується для взаємодії з базою даних.
- DbModelBuilder: зіставляє класи на мові C # з сутностями в базі даних.
- DbSet / DbSet <TEntity>: представляє набір сутностей, що зберігаються в базі даних.

У будь-якому додатку, що працює з БД через Entity Framework, нам потрібен буде контекст (клас похідний від DbContext) і набір даних DbSet, через який ми зможемо взаємодіяти з таблицями з БД. У нашому випадку таким контекстом є клас dbContext.

У конструкторі цього класу викликається конструктор базового класу, в який передається рядок "myMapDB" - це ім'я майбутньої рядка підключення до бази даних. В принципі ми можемо не використовувати конструктор, тоді в цьому випадку рядок підключення носила б ім'я самого класу контексту даних.

Для установки підключення зазвичай використовується файл конфігурації програми. У проектах для десктопних додатків файл конфігурації називається App.config (як в нашому випадку), в проектах веб-додатків - web.config. У нашому випадку це файл App.config. Після додавання Entity Framework він виглядає приблизно наступним чином:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
<configSections>
<section name="entityFramework"
type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection,
EntityFramework, Version=6.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089" requirePermission="false" />
</configSections>
<startup>
<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
</startup>
<entityFramework>
<defaultConnectionFactory
type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory,
EntityFramework">
<parameters>
<parameter value="mssqllocaldb" />
</parameters>
</defaultConnectionFactory>
<providers>
<provider invariantName="System.Data.SqlClient"
type="System.Data.Entity.SqlServer.SqlProviderServices,
EntityFramework.SqlServer" />
```

```
</providers>
</entityFramework>
<connectionStrings>
<add name="myMapDB" connectionString="data source=(LocalDB)\v12.0;Initial
Catalog=MapDB;Integrated Security=True;"
providerName="System.Data.SqlClient"/>
</connectionStrings>
</configuration>
```

Все підключення до джерел даних встановлюються в секції connectionStrings, а кожне окреме підключення представляє елемент add. У конструкторі класу контексту dbContext ми передаємо в якості назви підключення рядок "myMapDB", тому дане назва вказується в атрибуті name = "myMapDB".

Налаштування рядка підключення задає атрибут connectionString. В даному випадку ми встановлюємо назву бази даних, з якою будемо взаємодіяти - MapDB.mdf.

2.5Вибір мови програмування

C# (вимовляється *Сі-шарп*, також відомий як *ЦЭ-решітка*) — 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 їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

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 позаліцензійного 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# розроблялась як мова програмування прикладного рівня для CLR і тому вона 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.

До принципово важливих рішень, які були реалізовані, можна віднести наступні:

- компонентно-орієнтований підхід до програмування;
- властивості як засіб інкапсуляції даних);
- обробка подій (маються розширення, в тому числі в частині обробки винятків, зокрема, оператор try);
- уніфікована система типізації (відповідає ідеології Microsoft .NET в цілому);
- делегати (delegate - розвиток покажчика на функцію в мовах C і C ++);
- індексатори (indexer - оператори індексу для звернення до елементам класи-контейнера);
- перевантажені оператори;
- оператор foreach (обробка всіх елементів класів-колекцій);
- механізми boxing і unboxing для перетворення типів;
- Атрибути (засіб оперування метаданими в COM-моделі);
- прямокутні масиви (набір елементів з доступом за номером індексу і однаковою кількістю стовпців і рядків).

Наразі, C # успішно конкурує з Java і C ++ попопулярності. Розглянемо подібності цих мов.

Спершу перерахуємо подібності мов програмування C # iJava. Обидві мови об'єктно-орієнтовані і припускаютьєдиність успадкування. Так само особливостями, які роблятьсхожими мови програмування C # і Java, є механізмиінтерфейсів, обробки виняткових ситуацій, ниток (threads). обидва мови мають строгу типізацію і динамічне завантаження коду за виконанні програми.

Від мови програмування C ++, мовою C # успадкованімеханізми: «перевантажені» оператори, небезпечні арифметичніоперації з плаваючою точкою і безліч інших особливостейсинтаксису.

Виходячи з особливостей мови програмування C #, сформулюємоосновні переваги даної мови.

- Мова програмування C # претендує на справжню об'єктнуорієнтованість (будь-яка мовна сутність претендує на те, щоббути об'єктом);

- Компонентно-орієнтований підхід до програмування,сприяє меншій машинно-архітектурної залежності;результуючого програмного коду, гнучкості, переносимості і легкостіповторного використання (фрагментів) програм;

- Орієнтація на безпеку коду (в порівнянні з C і C ++);

- Уніфікована система типізації;

- Розширена підтримка подієво-орієнтованого програмування.

Незважаючи на переваги, мова C # має деякі недоліки, такі як:

- Досить складний синтаксис (75% з Java, 10% з C ++, 5% зVisual Basic);

- Мало свіжих концептуальних ідей (менш ніж10% конструкцій мови);

- Відносно невисока продуктивність (набагато повільніше,ніж мова C, але порівняємо з Java);

- не є крос-платформною мовою. Так як C # розроблений компаніюMicrosoft, то і працює він тільки під операційною системою Windows,хоча в даний момент вже розробляється крос-платформна версія даної мови.

З огляду на об'єктно-орієнтований дизайн, C # є хорошим вибором для швидкого конструювання різних компонентів – відвисокорівневої бізнес логіки до системних додатків,що використовують низькорівневий код. Також слід зазначити, що C #є і Web орієнтованою мовою - використовуючи прості вбудованіконструкції мови компоненти можуть бути перетворені в Web сервіси.

Додатковими можливостями мови C # - використання Webтехнологій, таких як: XML (Extensible Markup Language) і SOAP (SimpleObject Access Protocol). Середовище розробки Web сервісів дозволяєпрограмісту дивитися Web додатки, як на рідні C # об'єкти, щодає можливість розробникам співвіднести наявні Web сервіси з їхзнаннями в об'єктно-орієнтованому програмуванні.

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

Якщо при створенні традиційних додатків на основі WinForms за отрисовку елементів управління і графіки відповідали такі частини ОС 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, лягати на графічний процесор на відеокарті, що також дозволяє скористатися апаратним прискоренням графіки.

Однією з важливих особливостей є використання мови декларативною розмітки інтерфейсу XAML, заснованого на XML: можливе створєння насиченого графічного інтерфейсу, використовуючи або декларативне оголошення інтерфейсу, або код на керованих мовах C # і VB.NET, або поєднувати і те, і інше.

Переваги 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 дозволом.
- Нові можливості, яких складно було досягти в WinForms, наприклад, створення тривимірних моделей, прив'язка даних, використання таких елементів, як стилі, шаблони, теми і ін.
- Гарна взаємодія 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 двомірна 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, все компоненти програми транслюються в об'єкти, зрозумілі Direct3D, і потім візуалізуються за допомогою процесора на відеокарті, що підвищує продуктивність, робить графіком більш плавною.
- Створення додатків під безліч ОС сімейства Windows - від Windows XP до Windows 10.

3 ПРОЕКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ СКЛАДСЬКОГО ОБЛІКУ ЗЕРНОВИХ КУЛЬТУР

3.1 Життєвий цикл інформаційної системи

Життєвий цикл інформаційної системи є базовими елементом концепції проектного аналізу. Життєвий цикл інформаційної системи – це час від першої затрати до останньої вигоди проекту. Він відображає розвиток проекту, роботи, які проводяться на різних стадіях підготовки, реалізації та експлуатації проекту. До поняття життєвого циклу інформаційної системи входить визначення різних стадій розробки й реалізації проекту.

Життєвий цикл інформаційної системи являє собою певну схему або алгоритм, за допомогою якого відбувається встановлення певної послідовності дій при розробці та впровадженні проекту.

Ступінь деталізації і термінологія опису відповідних процедур залежать від характеру проекту, предметної культури, поставлених завдань, наявних ресурсів і, можливо, уподобань та смаків проектного аналітика.

Головне в процесі виділення фаз, стадій та етапів проекту полягає у позначенні деяких контрольних точок, під час проходження яких використовується додаткова (зовнішня) інформація і визначаються або оцінюються можливі напрями проекту. В будь-якому разі, прийнятий поділ відображає взаємодію проекту з середовищем (діючий механізм регулювання економіки країни, політики держави, існуюче становище в економіці тощо).

Реалізація проекту вимагає виконання певної кількості різноманітних заходів і робіт, які для зручності розгляду можна поділити на дві групи: основна діяльність і діяльність із забезпечення проекту. Такий поділ є поділом процесу реалізації проекту на фази і стадії, оскільки ці діяльності часто зберігаються в часі.

До основної діяльності звичайно відносять аналіз проблеми, формування цілей проекту, базове та детальне проектування, виконання будівельно-монтажних і пуско-налагоджувальних робіт, здавання проекту, експлуатацію проекту, ремонт, обслуговування та демонтаж обладнання тощо.

Діяльність по забезпеченню проекту, в свою чергу, може бути поділена на організаційну, правову, кадрову, фінансову, матеріально-технічну, комерційну та інформаційну.

Чіткого й однозначного розподілу цих робіт у логічній послідовності та у часі за можливою кількістю проектів не існує (відповідно і фаз та етапів виконання проекту), оскільки визначальними є цілі й умови проекту.

Узарубіжній літературі з аналізу та управління проектами використовуються різні підходи при поділі реалізації проекту на фази. Так, у Німеччині переважає підхід, що ґрунтується на основній діяльності, аналізі проблем, розробці концепції та детальному поданні проекту, використання результатів його реалізації, ліквідації об'єктів проекту.

У публікаціях деяких російських авторів пропонується розглядати три фази проекту – концептуальну, контрактну і фазу реалізації проекту. З огляду на запропоноване розрізнення концептуальна фаза має такі стадії: розробка концепції проекту, оцінка життєдіяльності проекту, планування проекту, розробка вимог до проекту, вибір і придбання земельної ділянки. Контрактна фаза включає вироблення кваліфікаційних вимог, підготовку попереднього завдання на проектування, заяву про наміри, добір потенційних виконавців, оформлення контракту з обраними виконавцями, вибір і затвердження остаточного варіанту проекту. Фаза реалізації проекту має дві стадії – детальне проектування та поставки; будівництво або інсталяція .

Програмою промислового розвитку ООН (UNIDO) запропоновано своє бачення проекту як циклу, що складається з трьох окремих фаз – передінвестиційної, інвестиційної та експлуатаційної.

Передінвестиційна фаза має такі стадії: визначення інвестиційних можливостей, аналіз альтернативних варіантів і попередній вибір проекту – попереднє техніко-економічне обґрунтування, висновок по проекту і рішення про інвестування.

Фаза експлуатації розглядається як у довгостроковому, так і в короткостроковому планах. У короткостроковому плані вивчається можливе виникнення проблем, пов'язаних із застосуванням обраної технології, функціонуванням обладнання або з кваліфікацією персоналу.Уплані, довгостроковому, до розгляду береться обрана стратегія та сукупні витрати на виробництво і маркетинг, а також надходження від продажу.

Універсальним підходом до визначення робіт, які відносяться до різних фаз і стадій ЦП, є підхід Всесвітнього банку.

На рис. 3.1 показано шість стадій, які відіграють важливу роль у більшості проектів. Це ідентифікація, розробка, експертиза, переговори, реалізація та завершальна оцінка [9].

Ці стадії об'єднані в дві фази: фаза проектування – перші три стадії; фаза впровадження – останні три стадії. Розглянемо їх докладніше далі на рис. 3.1.

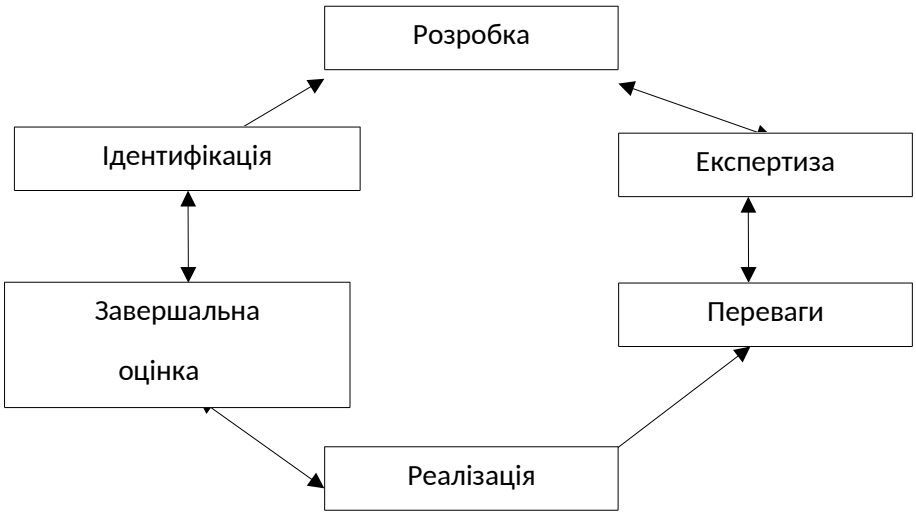

Рисунок 3.1 – Стадії життєвого циклу інформаційної системи

Перша стадія циклу – ідентифікація – стосується вибору або генерування таких ґрунтовних ідей, які можуть забезпечити виконання важливих завдань розвитку. На цій стадії слід скласти перелік всіх можливих ідей, придатних для досягнення цілей економічного розвитку. На подальших стадіях циклу проекту ці та інші ідеї буде уточнено і піддано дедалі ретельнішому аналізові в міру просування по стадіях проекту з метою остаточного визначення тієї комбінації заходів, що найкращим чином забезпечить досягнення цілей проекту. Ідеї, відображені на першій стадії, повинні відповідати деяким широким критеріям здорового глузду, а саме умовам, що прибуток від реалізації проекту перевищить витрати на його здійснення.

Таким чином, перша стадія циклу проекту виходить з чіткого формулювання цілей і тим самим утворює місток поміж аналізом здійснимості проекту. Завдання аналізу економічної політики полягає у встановленні пріоритетних цілей економічного розвитку та дослідженні тих змін у політиці й керівництві, які потрібні для виконання цих завдань. Аналіз можливості здійснення проекту передбачає оцінку цих завдань шляхом порівняння альтернативних засобів їх виконання та вибір найвигідніших варіантів.

Після того, як проект пройшов першу стадію циклу (ідентифікацію), необхідно прийняти рішення, чи варто продовжувати розгляд ідеї. Розпочинається стадія розробки. Для цього потрібне послідовне уточнення проекту за всіма його параметрами, а саме за його технічними

характеристиками, врахування його впливу на довколишнє середовище, ефективності та фінансової здійснимості, прийнятності з соціальних і культурних міркувань, а також масштабності організаційних заходів.

Розробка проекту включає звуження кола запропонованих на першій стадії циклу ідей шляхом детальнішого їх вивчення. Можливе проведення кількох типів досліджень, у тому числі попереднє інженерне проектування, аналіз економічної та фінансової здійснимості, розгляд систем адміністративного управління, які необхідні для успішного здійснення проекту та подальшої його експлуатації, оцінка альтернативних варіантів під поглядом захисту навколишнього середовища, оцінка впливу проекту на місцеве населення та його най вразливіші групи тощо. Чим більше ми знаємо про різні підходи до управління проектом, тим більше можливості маємо забракувати невдалі варіанти й приступити до детального вивчення обраного проекту.

Експертиза забезпечує остаточну оцінку всіх аспектів проекту перед запитом чи рішенням про його фінансування. На заключному етапі розробки проекту готується детальне обґрунтування його доцільності та здійснимості із зазначенням тих компонентів проекту, які дадуть максимальний прибуток. На стадії експертизи увага, як правило, зосереджується на оптимальному варіанті.Провадиться докладне вивчення фінансово-економічної ефективності, факторів невизначеності й ризиків, а також окремих змін у керівництві або політиці, які можуть вплинути на успіх здійснення проекту.

На стадії переговорів інвестор і замовник, який хоче одержати фінансування під проект, докладають зусиль для того, щоб дійти згоди щодо заходів, необхідних для забезпечення успіху проекту. Досягнуті домовленості потім оформлюються як документально застережені юридичні зобов’язання. Після проведення переговорів складається протокол намірів, меморандум або інші документи, що відображають досягнуті домовленості.

Під реалізацієюпроекту розуміють виконання необхідних робіт для досягнення його цілей. На стадії реалізації провадиться контроль і нагляд за всіма видами робіт чи діяльності в міру розвитку проекту. Порядок проведення контролю та інспекції має бути погоджено на стадії переговорів.

Настадії завершальної оцінки визначається ступінь досягнення цілей проекту, із набутого досвіду робляться висновки для його використання в подальших проектах. У перебігу цієї стадії треба порівняти фактичні результати проекту із запланованими.

3.2 Проектування інформаційної системи

Для проектування інформаційної системи була вибрана методологія IDEF3, оскільки вона служить для опису логіки і послідовності виконання процесів, а також представлення об'єктів, що беруть участь в одному процесі спільно. У моделюванні бізнес-процесів методологія використовується для аналізу завершеності процедур обробки інформації.

IDEF3 – спосібописупроцесів звикористаннямструктурованогометоду, що дозволяєекспертув предметнійобластіподати станречейяк упорядкованупослідовністьподійзодночаснимописомоб'єктів, що маютьбезпосереднє відношеннядо процесу.

IDEF3 є технологією, що добре пристосована для збору даних.На відмінувідбільшості технологіймоделюваннябізнес-процесів, IDEF3не маєжорсткихсинтаксичних абосемантичнихобмежень, які роблятьнезручнимописнеповнихабонецілісних систем. Крім того, автор моделі (системний аналітик) позбавлений необхідності змішувати свої власні припущення про функціонування системи з експертними твердженнями з метою заповненняпрогалин у описі предметної області. На рис.3.2 зображено приклад опису процесу з використання методології.

Зв'язки виділяють істотні взаємовідносини між діями. Усі зв'язки в IDEF3 є односпрямованими, і хоча стрілка може починатися або закінчуватися на будь-якій стороні блоку, що позначає дію, діаграми IDEF3 зазвичай організовуються зліва направо таким чином, що стрілки починаються на правій і закінчуються на лівій стороні блоків.

Зв'язок типу «тимчасове передування». Як видно з назви, зв'язки цього типу показують, що вихідна дія повинна повністю завершитися, перш ніж почнеться виконання кінцевої дії. Зв'язок повинен бути поіменован таким чином, щоб людині, хто переглядає модель, була зрозуміла причина її появи. У багатьох випадках завершення однієї дії ініціює початок виконання іншої.

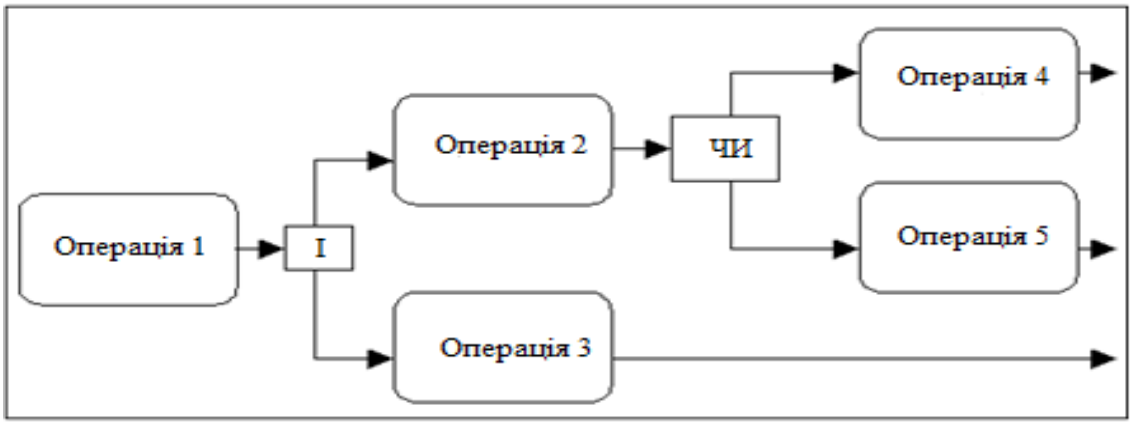

Рисунок 3.2 – Зразок побудови IDEF3 діаграми

IDEF3-моделювання органічно доповнює традиційне моделювання з використанням стандарту методології IDEF0. В даний час воно набуває все більшого поширення як цілком життєздатний шлях побудови моделей проектованих систем для подальшого аналізу імітаційними методами. Імітаційне тестування часто використовують для оцінки експлуатаційних якостей, що розробляється.

Основою моделі IDEF3 служить так званий сценарій бізнес-процесу, який виділяє послідовність дій або під процесів аналізованої системи. Оскільки сценарій визначає призначення і межі моделі, досить важливим є підбір відповідного найменування для позначення дій. Для підбору необхідного імені застосовуються стандартні рекомендації щодо кращого використання дієслів і віддієслівних іменників, наприклад «обробити замовлення клієнта» або «застосувати новий дизайн».

Також важливим для системного аналітика є розуміння мети моделювання –набору питань, відповідями на які буде служити модель, меж моделювання – які частини системи увійдуть, а які не будуть відображені в моделі, і цільової аудиторії – для кого розробляється модель.

Зв'язок типу «об'єктний потік».Одна з причин використання зв'язку типу «об'єктний потік» полягає в тому, що деякий об'єкт, який є результатом виконання вихідного дії, необхідний для виконання кінцевого дії. Позначення такого зв'язку відрізняється від зв'язку тимчасового передування подвійною стрілкою. Найменування потокових зв'язків повинні чітко ідентифікувати об'єкт, який передається з їх допомогою.

Зв'язок типу «нечітке відношення». Зв'язки цього типу використовуються для виділення відносин між діями, які неможливо описати з використанням попередніх або об'єктних зв'язків. Значення кожної такого

зв'язку має бути визначено, оскільки зв'язку типу «нечітке відношення» самі по собі не припускають ніяких обмежень.

При проектуванні дипломної роботи, розробляючи модель IDEF3, було виявлено наступні процеси, що відбуваються в наступній послідовності з моменту входу працівника в систему, включаючи обслуговування клієнта, до моменту запуску программи.

3.3 Проектування діаграми потоків даних

У відповідність з розглянутої методологією модель аналізованої інформаційної системи визначається як ієрархія діаграм потоків даних DFD, що описують процес перетворення інформації від введення в систему до видачі інформації користувачеві.

Діаграми потоків даних використовуються для опису руху документів і обробки інформації як додаток до методології функціонального моделювання IDEF0. На відміну від методології IDEF0, стрілки на діаграмах DFD показують лише те, як об'єкти (включаючи дані) рухаються від однієї роботи до іншої. Діаграма потоків даних DFD – це граф, на якому показано рух значень даних від їх джерел через перетворюють їх процеси до їх споживачів в інших об'єктах.

Діаграми верхніх рівнів ієрархії (контекстні діаграми) відображають зв'язок основного процесу системи із зовнішніми сутностями, які визначаються відповідними входами і виходами. Контекстні діаграми деталізуються за допомогою діаграм нижнього рівня. Така декомпозиція триває, створюючи багаторівневу ієрархію діаграм, до тих пір, поки не буде досягнутий такий рівень декомпозиції, на якому процеси стають елементарними і деталізувати їх далі неможливо.

Джерела інформації (зовнішні сутності) породжують інформаційні потоки (потоки даних), які переносять інформацію до процесів. Ті в свою чергу перетворюють інформацію і породжують нові потоки, які переносять інформацію до інших процесів, сховищ даних або зовнішніх сутностей – споживачам інформації.

Основними елементами моделі, яка поєднує діаграми потоків даних, є: процеси, зовнішні сутності, сховища даних, потік даних.

Діаграми потоків даних будуються за ієрархічним принципом. Першим кроком при побудові ієрархії діаграм є побудова контекстної діаграми.

Контекстна діаграма визначає межі моделі. Як правило, вона має зіркоподібну топологію, в центрі якої знаходиться головний процес, сполучений з приймачами та джерелами інформації, які є зовнішнім оточенням модельованої інформаційної системи. Включення зовнішніх сутностей в контекстну діаграму не скасовує вимоги методології чітко визначити мету, область і єдину точку зору на модельовану систему.

Для головного процесу, присутнього на контекстній діаграмі, проводиться декомпозиція. На першому рівні ієрархії показуються основні внутрішні процеси системи і відповідні їм зовнішні сутності, сховища і потоки даних. Декомпозиція процесів закінчується, коли досягнуто необхідний ступінь деталізації або відображаються на черговому рівні діаграм процеси є елементарними і не можуть бути розбиті на більш дрібні.

3.4 Проектування автоматизованої системи складського обліку зернових культур

Для автоматизованої системи складського обліку зернових культур буде розроблено два види користувачів: адміністратор та звичайний гість. Всі вони матимуть різні функції та привілеї. Також розмежування доступу використовується на багатьох програмних продуктах і служить, як гарант надійності та безпеки користування.

Першим, найголовнішим користувачем системи, є адміністратор. Після вводу унікального паролю, він матиме такі можливості користування:

1) Додавання та редагування у системі інформації обліку зернових культур на складах компанії;
2) Створення нових позицій у системі обліку;

Користувачем системи, якій матиме обмежений доступ до перегляду інформації та інших функцій, буде звичайних гість. Його функції користування матиме такий вигляд:

1) Можливість перегляду інформації;
2) Можливість редагування інформації без права збереження;

4. ПРАКТИЧНА РЕАЛІЗАЦІЯ

4.1 Постановка завдання для проектування

Мета даної дипломної роботи – розробити систему складського обліку зернових культур за допомогою якої користувач матиме можливість ознайомлюватися з усією необхідною йому інформацією.

Для реалізації поставленої мети були виділені основні задачі розробки програмного продукту:

- вибір програмнихзасобів, за допомогою яких буде реалізовуватись система складського обліку;
- аналіз існуючих аналогів та збір необхідні матеріали, які дозволили би створити всі необхідні умови для зручної роботи користувачів з даним програмним продуктом;
- створення надійної функціональної оболонки програмного продукту, яка дозволяла би працювати системі без помилок, а також з максимальною швидкістю;

Розроблена та готова до запуску на комп’ютері система повинна мати в собі такі важливі характеристики, як:

- дружній інтерфейс, який повинен бути зрозумілим, як новачку так і постійному користувачу з легкістю переміщуватись у рамках системи;
- надійна система безпеки, яка буде захищати базу даних від вірусів та від стороннього втручання;

4.2Система складського обліку

На головному вікні програмногу продукту ми маємо можливість бачити декілька груп управління та вікно з інформацією (рис.4.1).

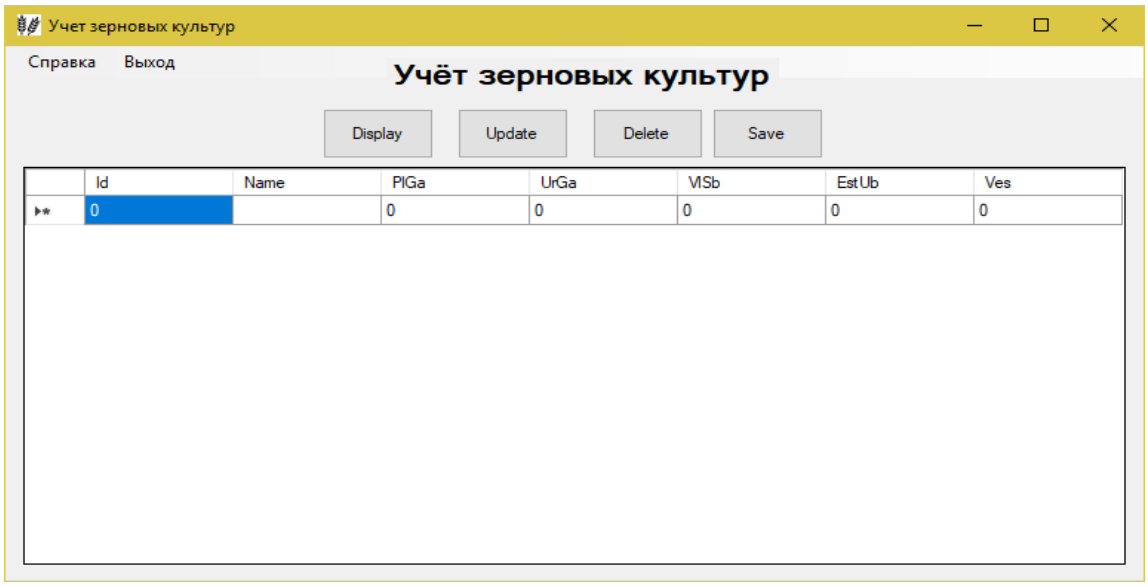

Рисунок 4.1 Система складського обліку

Рисунок 4.2 Кнопки управління

Кнопка Displayвідповідає за показ інформації з бази даних у головному вікні програми.

Код:

```
privatevoid Refresh()
      {
StudentDataClasses1DataContext SDCD1 = newStudentDataClasses1DataContext(@"Data
Source=(LocalDB)\v12.0;AttachDbFilename=C:\Users\zuben\Desktop\C#\FP1\bin\Debug\abc.mdf;Integrated
Security=False;Connect Timeout=30");
Zerno SI = newZerno();
var query = from q in SDCD1.Zerno
select q;
          dataGridView1.DataSource = query;// Attaching the all data with Datagridview
      }

privatevoid button3_Click(object sender, EventArgs e)
{
Refresh();
      }
```

Кнопка Updatevідповідає за редагування окремого поля окремої позиції та зберігання змінених даних.

Код:

```
privatevoid button2_Click(object sender, EventArgs e)
      {
int iid = 0;
StudentDataClasses1DataContext SDCD1 = newStudentDataClasses1DataContext(@"Data
Source=(LocalDB)\v12.0;AttachDbFilename=C:\Users\zuben\Desktop\C#\FP1\bin\Debug\abc.mdf;Integrated
Security=False;Connect Timeout=30");
Zerno SI = newZerno();
int rowindex = dataGridView1.CurrentRow.Index;   // here rowindex will get through currentrow prop-
erty of datagridview.
iid = Convert.ToInt32(dataGridView1.Rows[rowindex].Cells[0].Value);
var update = from s1 in SDCD1.Zerno
where s1.Id == iid
select s1;
foreach (var v in update)
        {
          v.Id = Convert.ToInt32(dataGridView1.Rows[rowindex].Cells[0].Value);
          v.Name = Convert.ToString(dataGridView1.Rows[rowindex].Cells[1].Value);
          v.PlGa = Convert.ToDecimal(dataGridView1.Rows[rowindex].Cells[2].Value);
          v.UrGa = Convert.ToDecimal(dataGridView1.Rows[rowindex].Cells[3].Value);
          v.VlSb = Convert.ToDecimal(dataGridView1.Rows[rowindex].Cells[4].Value);
          v.EstUb = Convert.ToDecimal(dataGridView1.Rows[rowindex].Cells[5].Value);
          v.Ves = Convert.ToDecimal(dataGridView1.Rows[rowindex].Cells[6].Value);
SDCD1.SubmitChanges(); // here will submitchanges function call and queries will automatic call.
        }
MessageBox.Show("Updated");
Refresh();// refresh the data gridview.
}
```

Кнопка Deleteвідповідає за видалення виділених позицій з бази даних.

Код:

```
privatevoid button1_Click(object sender, EventArgs e)
      {
int iid = 0;
StudentDataClasses1DataContext SDCD1 = newStudentDataClasses1DataContext(@"Data
Source=(LocalDB)\v12.0;AttachDbFilename=C:\Users\zuben\Desktop\C#\FP1\bin\Debug\abc.mdf;Integrated
Security=False;Connect Timeout=30");
Zerno SI = newZerno();
int rowindex = dataGridView1.CurrentRow.Index;  // here rowindex will get through currentrow property
of datagridview.
iid = Convert.ToInt32(dataGridView1.Rows[rowindex].Cells[0].Value);
var delete = from p in SDCD1.Zerno
where p.Id == iid// match the ecords.
select p;
SDCD1.Zerno.DeleteAllOnSubmit(delete);// DeleteAllOnSubmit function will call and queries will auto-
matic call thats the data context class handle it.
SDCD1.SubmitChanges();
rowindex = 0;

MessageBox.Show("deleted");
Refresh();
```

```
        }
```

Кнопка Saveвідповідає за збереження усіх змін які були створенні за період роботи.

Код:

```
privatevoid SaveButton_Click(object sender, EventArgs e)
        {
StudentDataClasses1DataContext SDCD1 = newStudentDataClasses1DataContext(@"Data Source=(LocalDB)\v12.0;AttachDbFilename=C:\Users\zuben\Desktop\C#\FP1\bin\Debug\abc.mdf;Integrated Security=False;Connect Timeout=30");
Zerno SI = newZerno();
int rowindex = dataGridView1.CurrentRow.Index;  // here rowindex will get through currentrow property of datagridview.

            SI.Id = Convert.ToInt32(dataGridView1.Rows[rowindex].Cells[0].Value);
            SI.Name = Convert.ToString(dataGridView1.Rows[rowindex].Cells[1].Value);
            SI.PlGa = Convert.ToDecimal(dataGridView1.Rows[rowindex].Cells[2].Value);
            SI.UrGa = Convert.ToDecimal(dataGridView1.Rows[rowindex].Cells[3].Value);
            SI.VlSb = Convert.ToDecimal(dataGridView1.Rows[rowindex].Cells[4].Value);
            SI.EstUb = Convert.ToDecimal(dataGridView1.Rows[rowindex].Cells[5].Value);
            SI.Ves = Convert.ToDecimal(dataGridView1.Rows[rowindex].Cells[6].Value);
SDCD1.Zerno.InsertOnSubmit(SI);//InsertOnSubmit queries will automatic call thats the data context class handle it.
SDCD1.SubmitChanges();
MessageBox.Show("Saved");
rowindex = 0;
        }
```

Також в програмному продукті є кнопка з інформацією для користувачів, нативснувши на котру з’явиться файл блокноту з поясненнями щодо програмного продукту (рис. 4.3).

Код:

```
privatevoidсправкаToolStripMenuItem_Click(object sender, EventArgs e)
        {
System.Diagnostics.Process.Start(@"C:\Users\zuben\Desktop\C#\FP1\spravka.txt");
}
```

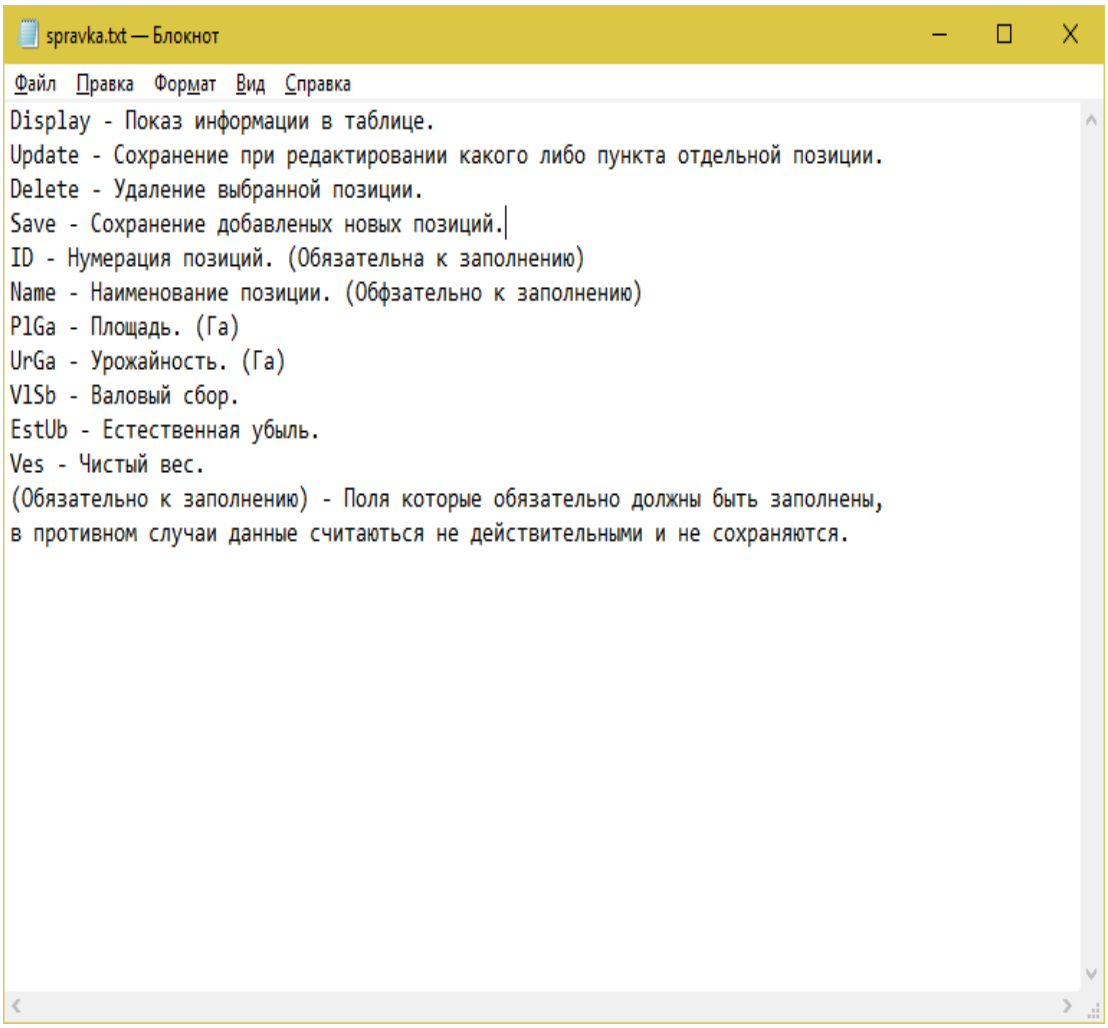

Рисунок 4.3 Справка

Діаграми потоків даних використовуються для опису руху документів і обробки інформації як додаток до методології функціонального моделювання IDEF0. На відміну від методології IDEF0, стрілки на діаграмах DFD показують лише те, як об'єкти (включаючи дані) рухаються від однієї роботи до іншої.

В наступному рисунку зображена діаграма потоків даних яку ми використували (рис. 4.4). В данній діаграммі ми можемо побачи дві бази данних які синхронізуються при кожному натиснені кнопки Save.

Таким чином при синхронізації даних друга база даних завжди наповняєть інформацією і завжди є резервною копією основної бази даних.

Резервна копія захищена від редагування але доступна до чтіння.

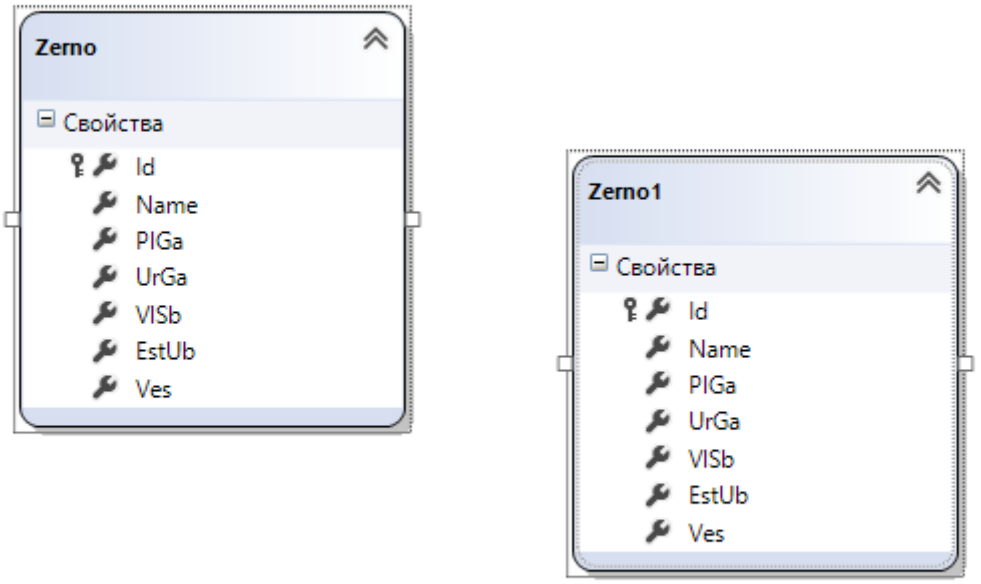

Рисунок 4.4 IDEF0 Діаграма

ВИСНОВКИ

У ході виконання дипломного проекту було створено повнофункціональну системускладського обліку зернових культур для АФ «Бургуджи».

При розробці даного програмного продукту були проаналізовані сучасні версії програмного забезпечення та найактуальніші засоби розробки програмних продуктів.

Розроблена програма задовольняє всім вимогам, поставленим на етапі постановки завдання, а саме:

1) обрано програмні засоби, за допомогою яких реалізовувано системускладського обліку зернових культур;

2) зроблено аналіз існуючих аналогів та зібрані необхідні матеріали, щодо умовзручної роботи користувачів з даним програмним продуктом;

3) використовуючи методологію моделювання IDEF3 спроектувано роботу системи;

4) створено надійну функціональну оболонку програмного продукту, яка дозволяє працювати системі без помилок, а також з максимальною швидкістю.

У ході написання даного дипломного проекту, програмний продукт неодноразово тестувався робітниками АФ «Бургуджи» і піддавався множинним доопрацюванням.

ПЕРЕЛІК ПОСИЛАНЬ

1. Попов Є.Ю. MYSQL для початківців[Електроний ресурс] – Режим доступу: http://www.evgeniypopov.ru/

2. Бейлі Моррісо. Вивчаємо MySql // Комп`ютерналітература. 2010 рік.

3. Галактика ERP- https://www.galaktika.ru/erp/galaktika-erp.html

4. 1С Склад, Контур бухгалтерія [Електроний ресурс] - http://erp-project.com.ua/index.php/uk/korisni-materiali/statti/avtomatizatsiya/190-presentatin

5. Язык программирования C#. Классика Computers Science. 4-е изд. **Автори:** А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд

6. Изучаем C#. 3-е изд. Авторы: Э. Стиллмен, Дж. Грин

7. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. **Автор:** Дж. Рихтер

8. Microsoft Visual C#. Step by Step **Автор:** ДжонШарп

9. C# 6.0 in a Nutshell: The Definitive Reference**Автор:** ДжозефАлбахари, БенАлбахари

10. C# 6.0 and the .NET 4.6 Framework **Автор:** Andrew Troelsen, Philip Japikse

11. Beginning C# Object-Oriented Programming**Автор:** Dan Clark