МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук управління та адміністрування

Кафедра інформаційних технологій

Бакалаврська кваліфікаційна робота

на тему: Розробка CRM-системи торгівельної компанії

Виконав студент 4 курсу групи К-42
Напрямок 6.05.01.01 Комп'ютерні науки

Кульчицький Ярослав Олександрович

Керівник к.т.н., доцент
Трегубова Ірина Анатоліївна

Рецензент к.ф-м.н., доцент
Вітавецька Лариса Анатоліївна

Одеса 2019

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

БД – база даних

ПЗ – програмне забезпечення

ПП – програмний продукт

СУБД – система управління базами даних

ТЗ – технічне завдання

CLR– CommonLanguageRuntime

IDE – IntegratedDevelopmentEnvironment, інтегроване середовище розробки.

ВСТУП

Актуальність теми роботи обумовлена необхідністю автоматизації рутинних завдань з метою зниження трудових і тимчасових витрат, створенням єдиної бази знань для зберігання досвіду роботи і готових рішень типових проблем, а також загальної оптимізації роботи підрозділу за рахунок більш ефективного розподілу завдань між співробітниками, з огляду на їх досвід і набір знань.

Метою роботиє створення CRM-системи для підвищення ефективності роботи підрозділу по розробці прикладного ПО, автоматизації бізнес-процесів, а також накопичення і документування досвіду роботи.

Для досягнення поставленої мети треба вирішити наступні задачі:

– розкрити теоретичні основи CRM-систем;
– виконати порівняльний огляд існуючих аналогів;
– проаналізувати бізнес-процеси підрозділу по розробці ПО;
– виконати вибір інструментальних засобів з урахуванням вимог до системи;
– здійснити проектування та розробку БД для майбутньої системи;
– створити клієнт для взаємодії зі сховищем даних;
– виконати аналіз ефективності розробленої CRM-системи.

Об’єкт дослідження – є розробка власної CRM-системи.

Предмет дослідження – є теоритичні, методологічні, програмні та прикладні аспекти розробки CRM-системи.

Практична значущість отриманих результатів полягає у тому, що отримані висновки сприятимуть подальшому розвитку новітніх IT технологій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

CRM(CustomerRelationshipManagement – система управління взаємовідносинами з клієнтами) досить популярна тема на протязі останніх 10-15 років. Сьогодні важко уявити ефективно працюючий конкурентний бізнес без налагодженої CRM системи.

У 1987 році Пет Салліван, засновникContactSoftwareInternnationali SalesLogixCorporation, випустив першу комерційну комп'ютерну програму для управління контактами, що отримала назву «ACT!». Ця спеціалізована програма дозволяла продавцям відстежувати розвиток взаємин з клієнтами, обробляти і аналізувати інформацію про ці взаємини. В даний час до двадцятиріччя продукту випущена вже 10-я версія програми («ACT! BySage 2008»), і вона до цих пір залишається в багатьох країнах світу лідером продажів CRM-рішень класу ContactManagement і оперативних CRM-систем.

Наступним витком в історії розвитку CRM ми зобов'язані Джем Ханди (JamHandy). Джем Ханди, був співробітником рекламної служби ChicagoTribune і кілька років він досліджував споживчу поведінку. Підсумком його роботи стала нова модель реклами, заснована на суті продукту, яка і до цього дня використовується маркетологами по всьому світу.

Аудіо-візуальна роз'яснювальна програма Ханди складалася зі спеціальних плакатів і фільмів, які демонструють якість, особливості і конкурентні переваги товарів. Джем говорив зі споживачами на зрозумілій мові – він один з перших почав використовувати сторітеллінг – розповідав захоплюючі історії.

Нова якість терміну «CRM-система» придбав до середини 90-х років. Під CRM-системою стали розуміти наскрізну автоматизацію клієнтоорієнтованих технологій продажів. Сам термін CRM вперше був використаний SiebelSystems, для того щоб відобразити специфіку цього типу корпоративних програмних продуктів.

Основною же стратегією успішного існування та подальшого розвитку сучасних компаній поступово стає ефективне управління взаємовідносинами з клієнтами. Орієнтація підприємства на удосконалення відносин з клієнтами обумовлена низкою тенденцій, зокрема посиленням конкуренції, підвищенням вимог покупців до якості пропонованих продуктів і рівнем сервісу, зниженням ефективності традиційних маркетингових засобів, а також появою нових технологій взаємодії з клієнтами і функціонування підрозділів компанії. Знання своїх клієнтів і задоволення запитів і потреб кожного з них можуть дозволити компанії отримати нові можливості для збуту товарів і послуг і стати ключовим фактором сталого розвитку і джерелом довгострокового конкурентної переваги компанії на ринку.

Західний досвід показує, що висока ефективність роботи з покупцями забезпечується за рахунок прийняття компанією концепції з управління взаємовідносинами з клієнтами, що отримала назву концепції CRM. Концепція дозволяє «інтегрувати» клієнта в сферу організації – фірма отримує максимально можливу інформацію про своїх клієнтів і їхні потреби і, виходячи з цих даних, будує свою організаційну стратегію, яка стосується всіх аспектів її діяльності: виробництва, маркетингу, продажів, обслуговування та інше.

Зростання популярності CRM-систем в світі відображають дві тенденції:

– все більшого значення, яке розробники CRM-продуктів надають потребам своїх користувачів;
– більш чітке розуміння того, як створювати і правильно застосовувати CRM-технології в бізнесі.

Концепція CRM не є винаходом останнього часу і використовувалися з давніх-давен. Найпростіші «CRM-системи» минулого у вигляді Писцовой, комор, боргових книг часто дозволяли не тільки згадати колишню історію спілкування з клієнтами, а й проявити по відношенню до клієнта більше

уваги, дати зрозуміти, що колишні взаємини не забуті. Це вже дозволяло утримувати постійних клієнтів і залучити нових за рахунок «клієнтоорієнтованості» в веденні справ.

Пізніше такий підхід в першу чергу розвивався в сфері послуг і сервісу, особливо в малому бізнесі. CRM навіть називають «технологією дешевого зростання» через доступність і значного ефекту для бізнесу. Саме клієнтоорієнтованість дозволяла підприємствам малого бізнесу вийти зі сфери цінової конкуренції з великими корпораціями, утримати клієнтів і збільшити продажі.

Перша комерційна комп'ютерна програма для управління контактами, що отримала назву «ACT!», Була випущена в 1987 році Петом Салливаном. А сам термін CRM вперше був використаний Siebel Systems для того, щоб відобразити специфіку цього типу корпоративних програмних продуктів. На даний момент в світі існує більше 1000 рішень, які можна віднести до класу CRM і Contact Management.

Незважаючи на те, що комп'ютерні CRM-системи існують на ринку більше двадцяти років, питання про перелік їх функціональних складових все ще відкритий. Хоча визначення CRM еволюціонує, багато фахівців сходяться на думці, що сучасне повнофункціональне CRM-рішення повинно мати 11 основних компонентів з переліку Бартона Голденберга (BartonGoldenberg, засновника і президента компанії ISMInc.):

- управління контактами;
- управління продажами;
- продажу по телефону;
- керування часом;
- підтримка і обслуговування клієнтів;
- управління маркетингом;
- звітність для вищого керівництва;
- інтеграція з іншими системами;

– синхронізація даних;
– управління електронною торгівлею;
– управління мобільними продажами.

Лояльність, прихильність клієнта до компанії важлива складова CRM. Лояльний клієнт – це постійний клієнт, йому подобається продукція і сервіс, що надається в компанії. Саме лояльний клієнт – приносить постійний дохід компанії, підтримує позитивний імідж компанії, приваблює нових клієнтів.

Лояльність можливою завдяки тому, що компанія надає клієнтові не тільки якісну продукцію і сервіс, але і йде на зустріч його особистим інтересам (персоналізація клієнта).

Підвищення лояльності дозволяє збільшити число постійних клієнтів, кількість повторних покупок, знизити витрати на залучення нових клієнтів. допомагає підтримувати контакти одночасно з багатьма клієнтами, не забувати про них, тому такі системи є особливо актуальними в компаніях з великою кількістю клієнтів.

Можна виділити наступні види діяльності потенційних покупців:

а) Компанії виробники продукції.
б) Компанії оптової торгівлі:
    1) алкоголем і продуктами харчування;
    2) автомобілями;
    3) побутовою технікою, комп'ютерами;
    4) меблями;
    5) канцтоварами;
    6) побутовій і промисловій хімієюі.
в) Компанії сфери послуг:
    1) банки і фінансові інститути;
    2) страхові компанії;
    3) софтверні компанії;
    4) консалтингові компанії;

5) юридична допомога;

6) лікувальні та медичні установи.

Ідеологія CRM системи покриває всі ділянки підприємства, співробітники яких якимось чином контактують з клієнтами.

Якість взаємодії співробітника FRONT-OFFICE з клієнтом часто залежить від співробітників BACK-OFFICE, що надають йому інформацію. Наприклад: менеджеру продажів необхідно мати інформацію про оплатах, залишку товару, відвантаженні товару цю інформацію йому повинен надати співробітник бухгалтерії.

Основні інструменти, які включає в себе технологія управління відносини з клієнтами (CRM):

- збір в єдину клієнтську базу всієї накопиченої про клієнтів інформації;
- збір історії взаємин з клієнтами, партнерами і постачальниками;
- обмін інформацією між підрозділами і співробітниками без «інформаційних провалів»;
- автоматизація послідовності робіт - бізнес-процеси - і інтеграція їх у робоче середовище;
- отримання аналітичних звітів;
- прогнозування продажів;
- планування та аналіз ефективності маркетингових заходів;
- контроль задоволеності клієнтів, реєстрація і розбір скарг;
- накопичення знань компанії і управління ними.

Збір в єдину клієнтську базу всієї накопиченої про клієнтів інформації.

Завдання: збереження клієнтської бази та утримання платоспроможних клієнтів.

Рішення за допомогою CRM-системи:

Розширена інформація про клієнта і контактні особи в картці клієнта дозволяє зберегти інформацію про клієнтів в разі звільнення менеджера з

продажу, швидко зв'язатися з клієнтом зручним способом і провести аналіз бази клієнтів за різними аналітичними параметрами.

Сегментація клієнтської бази по стабільності закупівель і за часткою клієнтів в загальних продажах компанії («ABC / XYZ-аналіз» клієнтів) дозволяє в будь-який момент часу виділити ключових (VIP) клієнтів компанії.

Сегментація клієнтської бази по регіону і виду діяльності дозволяє виділити кілька основних сегментів клієнтів і, наприклад, персонально кожному з них запропонувати спеціальні умови покупки або сервісу за допомогою автоматизованої електронної розсилки.

Збір історії взаємин з клієнтами, партнерами і постачальниками.

Завдання: збереження і аналіз історії взаємин.

Рішення за допомогою CRM-системи:

Розширена інформація про контакти з клієнтом за період дозволяє менеджеру з продажу швидко відновити в пам'яті історію спілкування з клієнтом, зрозуміти, хто з колег в даний час працює з цим же клієнтом і з яких питань.

Інформація про заплановані зустрічі менеджера з продажу дозволяє заміщає його співробітникові продовжити контакти з клієнтами під час його відсутності, наприклад через хворобу.

Всі домовленості і будь-яка інша інформація про минулі і планованих зустрічах з клієнтами, телефонних дзвінках, електронних листах або участі клієнта в семінарах компанії, фіксується в системі для подальшого аналізу.

Контроль виконання менеджерами з продажу регламенту по частоті контактів за період з закріпленими за ними клієнтами, в т.ч. за певними типами контактів.

Обмін інформацією між підрозділами і співробітниками без «інформаційних провалів».

Завдання: виключити втрату інформації при передачі всередині компанії і прискорити «інформаційні потоки» компанії.

Рішення за допомогою CRM-системи:

Передача всієї інформації всередині компанії через CRM-систему гарантує її доставку адресату повідомлення.

Механізм передачі інформації про контакт з клієнтом (подію) в CRM-системі дозволяє легко простежити «долю» переданої інформації і вжити заходів, якщо переданий контакт або заявку не відпрацювали в покладений за регламентом термін.

Автоматична передача завдання наступного виконавцю дозволяє максимально скоротити витрати часу на очікування передачі виконаної іншим співробітником або підрозділом роботи наступному виконавцю.

Механізм видачі та контролю виконання доручень дозволяє керівнику оперативно контролювати всі доручені їм співробітникам роботи, причому в зручний для керівника час.

Автоматизація послідовності робіт: бізнес-процеси і інтеграція їх у робоче середовище.

Завдання: помістити «паперові» бізнес-процеси безпосередньо в CRM-систему і зробити їх актуальними, інтегрувати в робоче середовище компанії.

Рішення за допомогою CRM-системи:

Маршрутна карта бізнес-процесу в CRM-системі підкаже співробітнику правильну послідовність дій і не дозволить пропустити важливі стадії робочого процесу. система дозволяє закріпити за кожним етапом бізнес-процесу відповідального співробітника, а також контролера за ходом всього бізнес-процесса.сістема дозволяє в будь-який момент часу виявити відхилення від стандартних параметрів в кожному бізнес-процесі. система дозволяє поширити досвід кращих співробітників компанії через впровадження в роботу єдиних для всіх співробітників бізнес-процесів. Регламенти та інструкції, представлені в CRM-системі у вигляді бізнес-

процесів починають працювати немедленно.система дозволяє контролювати всі видані доручення у зручний для перевантаженого роботою керівника час.

«Управління по інцидентах»: звільнення керівника від рутинних завдань.

Завдання: виявлення і контроль проблемних угод із загального числа поточних операцій компанії, аналіз причин невдачі, перерозподіл ресурсів компанії.

Рішення за допомогою CRM-системи: система дозволяє в будь-який момент часу виявити відхилення від стандартних параметрів на кожному етапі бізнес-процесу, тому що в шаблоні бізнес-процесу задані тимчасові рамки виконання етапу, необхідні дії, результат етапу, ймовірність успіху і відповідальний за етап співробітник. система дозволяє не тільки відслідковувати відхилення від типових параметрів бізнес-процесу, але збирати й аналізувати дані про причини «відмов» клієнтів (неуспішних закритті угод). система дозволяє за допомогою «воронки продажів» швидко відстежити проблемні етапи угод в масштабах компанії, визначити на якому етапі по операціях стався збій, а на якому етапі угоди затрималися довше, ніж передбачалося терміну.

Управління дебіторською заборгованістю: інструменти персональної роботи з клієнтами.

Завдання: зменшення суми простроченої дебіторської заборгованості з використанням даних про історію взаємин з клієнтом

Рішення за допомогою CRM-системи: система дозволяє з мінімальними затратами інформувати клієнтів про наявність простроченої дебіторської заборгованості. система дозволяє використовувати історію взаємин із клієнтами для клієнтоорієнтованого збору боргів система дозволяє вибирати зручні для компанії і клієнта канали комунікацій. система дозволяє працювати з кожним клієнтом персонально.

Клієнт – єдине джерело доходу компаніісистеми дозволяють утримувати існуючих, легше залучати нових і ефективніше працювати з усіма кліентамісістеми потрібні великій кількості компаній різних видів діяльності. CRM – це міжгалузеве рішення.

Впровадження CRM дозволить – збільшити обсяг продажів / прибутку

Ефективна система управління продажами - запорука виживання компанії в умовах кризи, добробуту компанії.Особливо актуальним стає використання CRM-систем в період кризи, в період яких головними завданнями, що стоять перед компаніями, є: збереження клієнтської бази, утримання платоспроможних клієнтів, управління робочим часом (таймменеджмент), підвищення продуктивності праці за допомогою автоматизованих бізнес-процесів, прискорення інформаційних потоків всередині компанії, оптимізація продуктового портфеля компанії, управління відносинами з постачальниками, оптимізація витрат на маркетинг, управління дебіторською заборго женностью, скорочення витрат часу керівника на контроль поточної діяльності компанії, автоматизація рутинних операцій. В таких умовах CRM-система допомагає підвищити ефективність праці співробітників, оптимізувати персонал (перерозподіл по відділах), швидко ввести в роботу нових співробітників, подолати кризу продажів, виділити перспективних клієнтів, відмовитися від неплатоспроможних клієнтів, об'єднати продажу різних підрозділів компанії (перехресні продажі), ефективно працювати з партнерами і знайти нові напрямки розвитку для компанії.

Основні причини для впровадження CRM-системи.

Розрізнена клієнтська база. Дані про клієнтів, партнерів, постачальників, конкурентів зберігаються в різних джерелах. Інформація важкодоступна, немає можливості спільного аналізу даних про клієнтів, при звільненні менеджера дані про клієнтів просто пропадають, так як ніхто не знає, де вони зберігаються і як їх отримати.

Історія спілкування з клієнтами роз'єднана або не реєструється зовсім. Досягнуті в переговорах домовленості забуваються і не виконуються, що викликає негативну реакцію клієнтів. Клієнтів кожен раз перепитують про їх номер телефону або e-mail, про те, що вони замовляли.

Втрата інформації при передачі між підрозділами призводить до збоїв в основних бізнес-процесах компанії. «Інформаційні провали» між співробітниками і підрозділами збільшують кількість скарг і собівартість продажів.

Регламентовані і затверджені бізнес-процеси не автоматизовані і не впроваджені в «робоче середовище» компанії. Вони не завжди виконуються, і неможливо оперативно контролювати хід виконання бізнес-процесів.

Компанії необхідний інструмент прогнозування продажів для оперативного управління даними бізнес-процесом.

Немає можливості аналізу клієнтської бази, побудови комплексних звітів з продажу, закупівель і історії спілкування з клієнтами.

Скарги клієнтів губляться, не розглядаються вчасно, немає можливості отримати звітність за типами скарг за період в розрізі менеджерів компанії.

Знання співробітників компанії зберігаються тільки в головах співробітників, передача знань відбувається від досвідченого співробітника до новачка, займає багато часу і в підсумку призводить до витрат компанії (зниження продажів і т.д.). Співробітники багато часу витрачають на відповіді на типові питання клієнтів.

Паперова звітність складається вручну. Менеджери змушені витрачати кілька годин на тиждень для складання звітів з продажу та бесід з керівником з питань оперативної діяльності.

Рутинні операції, що віднімають багато часу. Складання типового договору або комерційної пропозиції займає у менеджера більше півгодини.

Контроль забирає час. Керівник змушений половину свого робочого дня витрачати на контроль роботи співробітників.

Необгрунтовано великий штат співробітників. Штат відділів маркетингу, продажів і сервісного обслуговування зростає набагато швидше, ніж зростають обороти компанії.

2 АНАЛІЗ ВИБОРУ СИСТЕМИ ДЛЯ ВИКОНАННЯ ЗАВДАННЯ ТА ОСНОВНІ ЕТАПИ РОЗРОБКИ CRM-СИСТЕМИ

2.1 Вибір платформи для розробки CRM-системи

Для виконання поставленої задачі, була обрана мова програмування C#, програмна технологія .NETFramework, інтегроване середовище розробкиMicrosoftVisualStudio та система керуванням базами даних MicrosoftSQLServer.

2.1.1 Опис C#

C# – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольдепід егідою MicrosoftResearch (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 багатощо від своїх попередників – 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++)[1][1)].

Основними перевагами мови C# є:

– підтримка корпорації Microsoft. На відміну від Java, котрій перехід увласність Oracle не пішов на користь, C# добре розвивається завдяки зусиллямMicrosoft;

[1)][1] C-sharp –Вікіпедія . URL: https://uk.wikipedia.org/wiki/C_Sharp. (дата звернення 03.04.2019).

– багато синтаксичного цукру. Синтаксичний цукор – це такі конструкції, які створені для полегшення написання і розуміння коду (особливо якщо це код іншого програміста) і не грають ролі при компіляції;
– середній поріг входження. Синтаксис схожий на C, C ++ або Java полегшує перехід для інших програмістів. Для новичків це також один з найперспективніших мов для вивчення.

З недоліків можна виділити:

– орієнтованість, в основному, тільки на .NET (на Windows платформу);
– безкоштовність тільки для невеликих компанії, учнів і програмістів-одинаків. Для великих команд покупка ліцензій обійдеться недешево. Тому якщо у вас є своя фірма, то доведеться розщедритися.

2.1.2 Опис .NET Framework

Microsoft .NET Framework – програмнатехнологія,запропонована фірмою Microsoft як платформа для створення як звичайних програм, так і вебзастосунків. Багато в чому є продовженням ідей та принципів, покладених в технологію Java. Однією з ідей .NET є сумісність служб, написаних різними мовами. Хоча ця можливість рекламується Microsoft як перевага .NET, платформа Java має таку саму можливість[2][1)].

Програма для .NET Framework, написана на будь-якому підтримуваному мовою програмування, спочатку перекладається компілятором в єдиний для .NET проміжний байт-код Common Intermediate Language (CIL) (раніше називався Microsoft Intermediate Language, MSIL). У термінах .NET виходить збірка, англ. assembly. Потім код або виповнюється віртуальною машиною

[1)][2] .NETFramework – Вікіпедія. URL: https://uk.wikipedia.org/wiki/.NET_Framework. (дата звернення: 04.04.2019).

Common Language Runtime (CLR), або транслюється утилітою NGen.exe в виконуваний код для конкретного цільового процесора. Використання віртуальної машини переважно, оскільки позбавляє розробників від необхідності піклуватися про особливості апаратної частини. У разі використання віртуальної машини CLR вбудований в неї JIT-компілятор «на льоту» (just in time) перетворює проміжний байт-код в машинні коди потрібного процесора. Сучасна технологія динамічної компіляції дозволяє досягти високого рівня швидкодії. Віртуальна машина CLR також сама піклується про базову безпеки, управлінні пам'яттю і системі винятків, позбавляючи розробника від частини роботи.

2.1.3 Опис Microsoft Visual Studio

Microsoft Visual Studio – серія продуктів фірми Microsoft, які включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight[3][1)].

Основними перевагами даної IDEє:

– підтримка безлічі мов при розробці. Visual Studio дозволяє писати код своєю рідною мовою чи будь-яких інших бажаних мовами, використовуючи весь час один і той же інтерфейс (IDE). Більш того, Visual Studio також ще дозволяє створювати Web-сторінки на різних

[1)][3]Microsoft Visual Studio – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio. (датазвернення 11.04.2019).

мовах, але поміщати їх все в один і той же Web-додаток. Єдиним обмеженням є те, що в кожної Web-сторінці можна використовувати тільки якийсь один мову (очевидно, що в іншому випадку проблем при компіляції було б просто не уникнути);

– менше коду для написання. Для створення більшості додатків потрібно пристойну кількість стандартного стереотипного коду, і Web-сторінки ASP.NET тому не виключення. Наприклад, додавання Web-елемента управління, приєднання обробників подій і коригування форматування вимагає установки в розмітці сторінки ряду деталей. У Visual Studio такі деталі встановлюються автоматично;
– Інтуїтивний стиль кодування. За замовчуванням Visual Studio форматує код у міру його введення, автоматично вставляючи необхідні відступи і застосовуючи колірне кодування для виділення елементів типу коментарів. Такі незначні відмінності роблять код більш зручним для читання і менш схильним до помилок. Застосовувані Visual Studio автоматично параметри форматування можна навіть налаштовувати, що дуже зручно у випадках, коли розробник вважає за краще інший стиль розміщення дужок (наприклад, стиль K & R, при якому відкриває дужка розміщується на тому самому рядку, що і оголошення, якому вона передує);
– більш висока швидкість розробки. Багато з функціональних можливостей Visual Studio спрямовані на те, щоб допомагати розробнику робити свою роботу якомога швидше. Зручні функції, на зразок функції IntelliSense (яка вміє перехоплювати помилки і пропонувати правильні варіанти), функції пошуку і заміни (яка дозволяє відшукувати ключові слова як в одному файлі, так і в усьому проекті) і функції автоматичного додавання і видалення

коментарів (яка може тимчасово приховувати блоки коду), дозволяють розробнику працювати швидко і ефективно;

– можливості налагодження. Пропоновані в Visual Studio інструменти налагодження є найкращим засобом для відстеження загадкових помилок і діагностування дивної поведінки. Розробник може виконувати свій код по рядку за раз, встановлювати інтелектуальні точки переривання, при бажанні зберігаючи їх для використання в майбутньому, і в будь-який час переглядати поточну інформацію з пам'яті.

Visual Studio також має і безліч інших функцій: можливість управління проектом; вбудована функція управління вихідним кодом; можливість рефакторізації коду; потужна модель розширюваності. Більш того, в разі використання Visual Studio 2008 Team System розробник отримує розширені можливості для модульного тестування, спільної роботи і управління версіями коду (що значно більше того, що пропонується в більш простих інструментах на кшталт Visual SourceSafe).

Як недолік можна відзначити неможливість відладчика (Microsoft Visual Studio Debugger) відстежувати в коді режиму ядра. Налагодження в Windows в режимі ядра в загальному випадку виконується при використанні WinDbg, KD або SoftICE.

Також було обрано систему керуванням базами даних MicrosoftSQLServer.

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.

Попередні версії засобів розробки Microsoft використовували лише API для надання функціонального доступу до Microsoft SQL Server.

2.1.4 Опис Microsoft SQL Server

Microsoft SQL Server – комерційна система керуванням бази 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[4][1)].

MicrosoftSQLServer 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.

2.2 Основні етапи виконання завдання

ПП будь-якого виду характеризуються життєвим циклом - від моменту виникнення ідеї розробки ПП до моменту відмови від використання ПП. Стадії життєвого циклу програм визначають склад і зміст робіт зі створення програмних продуктів.

[1)][4]MicrosoftSQLServer – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Microsoft_SQL_Server(дата звернення 11.04.2019)

При традиційній розробці програм розрізняють наступні етапи створення програмного продукту:

– аналіз вимог до проекту;
– проектування;
– реалізація;
– тестування продукту;
– впровадження та підтримка.

На етапі аналізу вимог до проекту формулюються цілі та завдання проекту, виділяються базові сутності та взаємозв'язку між ними. Тобто, створюється основа для подальшого проектування системи.

В рамках даного етапу не тільки фіксуються вимоги замовника, а й проводиться їх формування - клієнтам підбирається оптимальне вирішення їхніх проблем, визначається необхідний ступінь автоматизації, виявляються найбільш актуальні для автоматизації бізнес-процеси.

При аналізі вимог визначаються терміни і вартість розробки ПО, формується і підписується ТЗ на розробку програмного забезпечення.

На основі попереднього етапу проводиться проектування системи. Ця методологія проектування поєднує в собі об'єктну декомпозицію, прийоми уявлення фізичної, логічної, а також динамічної та статичної моделей системи.

Під час проектування розробляються проектні рішення щодо вибору платформи, де буде функціонувати система мови або мов реалізації, призначаються вимоги до призначеного для користувача інтерфейсу, визначається найбільш підходяща СУБД. Розробляється функціональна специфікація ПЗ: вибирається архітектура системи, обумовлюються вимоги до апаратного забезпечення, визначається набір орг. заходів, які необхідні для впровадження ПО, а також перелік документів, що регламентують його використання.

Даний етап розробки ПЗ організований відповідно до моделей еволюційного типу життєвого циклу ПО. При розробці застосовуються експериментування і аналіз, будуються прототипи, як цілої системи, так і її частин. Прототипи дають можливість глибше вникнути в проблему і вжити всіх необхідних проектні рішення ще на ранніх етапах проектування. Такі рішення можуть зачіпати різні частини системи: внутрішню організацію, призначений для користувача інтерфейс, розмежування доступу і т.д. У результату етапу реалізації з'являється робоча версія продукту.

Тестування тісно пов'язане з такими етапами розробки програмного забезпечення як проектування і реалізація. У систему вбудовуються спеціальні механізми, які дають можливість виробляти тестування системи на відповідність вимог до неї, перевірку оформлення і наявність необхідного пакету документації.

Результатом тестування є усунення всіх недоліків системи і висновок про її якість.

Впровадження системи зазвичай передбачає наступні кроки:

– установка системи,
– навчання користувачів,
– експлуатація.

До будь-якої розробки додається повний пакет документації, який включає в себе опис системи, керівництва користувачів і алгоритми роботи.

Підтримка функціонування ПО повинна здійснюватися групою технічної підтримки розробника.

3 ПРОЕКТНА ЧАСТИНА

3.1 Проектування структури БД програмного продукта

Один з основних методів проектування структури БД – метод семантичного проектування, який спирається на зміст даних, які будуть зберігатися в БД. Як інструмент семантичного моделювання застосовуються різні варіанти так званих ER-діаграм. Такі діаграми використовують графічні зображення сутностей (Essences) предметної області, їх властивостей (атрибутів), і взаємозв'язків (Relations) між сутностями. Тому даний метод проектування також називається: метод ER-діаграм або метод сутність-зв'язок.

Для того щоб спроектувати базу даних, для початку необхідно визначити для кожного атрибута таблиці його тип даних.Також при проектуванні таблиць необхідно позначити атрибути або їх поєднання, які будуть первинними і зовнішніми ключами таблиці.

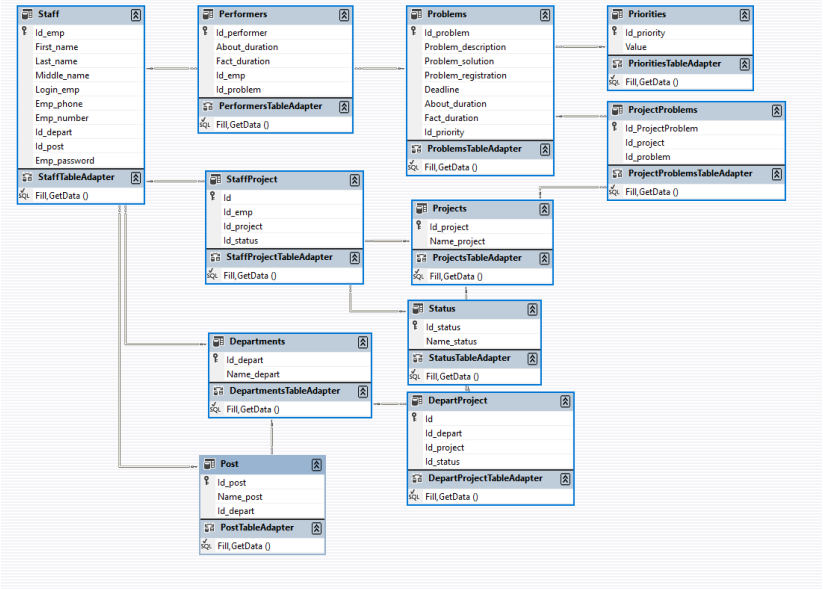

Рисунок 3.1 – Схема БД програмного продукта

Розглянемо структуру таблиць бази даних CRMDBдетальніше.

Співробітники компанії зберігаються у таблиці Staff. Опис даної сутностіприведено в табл. 3.1.

Таблиця 3.1 – Опис сутності Staff

Атрибут	Тип даних	Опис
Id_emp	int	Ідентифікатор співробітника, первинний ключ
First_name	varchar(25)	Ім’я співробітника
Last_name	varchar(25)	Прізвище співробітника
Middle_name	varchar(25)	По батькові
Login_emp	varchar(25)	Логін співробітника
Emp_password	varchar(25)	Пароль співробітника
Emp_phone	varchar(25)	Номер телефону співробітника
Emp_number	varchar(25)	Табельний номер співробітника
Id_depart	int	Індетифікатор відділу, зовнішній ключ
Id_post	int	Ідентифікатор посади, зовнішній ключ

Дані про відділи компанії зберігаються в таблиці Departments. Опис даної сутності приведено в табл. 3.2.

Таблиця 3.2 – Опис сутності Departments

Атрибут	Тип даних	Опис
Id_depart	int	Ідентифікатор відділу, первинний ключ
Name_depart	varchar(50)	Назва відділу

Дані про посади у відділах зберігаються в таблиці Post. Опис даної сутності приведено в табл. 3.3.

Таблиця 3.3 – Опис сутності Post

Атрибут	Тип даних	Опис
Id_post	int	Ідентифікатор посади, первинний ключ
Name_post	varchar(50)	Назва посади
Id_depart	int	Ідентифікатор відділу, зовнішній ключ

Дані про проекти компанії здерігаються в таблиці Projects. Опис даної сутності приведено в табл. 3.4.

Таблиця 3.4– Опис сутності Projects

Атрибут	Тип даних	Опис
Id_project	int	Ідентифікатор проекту, первинний ключ
Name_project	varchar(50)	Назва проекту

Дані про статуси проектів зберігаються в таблиці Status.Опис даної сутності приведено в табл. 3.5.

Таблиця 3.5 – Опис сутності Status

Атрибут	Тип даних	Опис
Id_status	int	Ідентифікатор проекту співробітника, первинний ключ
Name_status	varchar(50)	Значення статусу

Дані про проекти які пов’язані з співробітниками зберігаються в таблиці StaffProject. Опис даної сутності приведено в табл. 3.6.

Таблиця 3.6– Опис сутності StaffProject

Атрибут	Тип даних	Опис
1	2	3
Id	int	Ідентифікатор проекту співробітника, первинний ключ
Id_emp	int	Ідентифікатор співробітника, зовнішній ключ

Продовження таблиці 3.6

1	2	3
Id_project	int	Ідентифікатор проекту, зовнішній ключ
Id_status	int	Ідентифікатор статусу, зовнішній ключ

Дані про проекти які пов’язані з відділами зберігаються в таблиці DepartProject. Опис даної сутності приведено в табл. 3.7.

Таблиця 3.7 – Опис сутності DepartProject

Атрибут	Тип даних	Опис
Id	int	Ідентифікатор проекту відділу, первинний ключ
Id_depart	int	Ідентифікатор відділу, зовнішній ключ
Id_project	int	Ідентифікатор проекту, зовнішній ключ
Id_status	int	Ідентифікатор статусу, зовнішній ключ

Дані про пріоритети запросів зберігаються в таблиці Priorities. Опис даної сутності приведено в табл. 3.8.

Таблиця 3.8 – Опис сутності Priorities

Атрибут	Тип даних	Опис
Id_priority	int	Ідентифікатор пріоритету, первинний ключ
Value	varchar(25)	Значення пріоритету

Дані про запити користувачей зберігаються в таблиці Problems. Опис даної сутності приведено в табл. 3.9.

Таблиця 3.9– Опис сутності Problems

Атрибут	Тип даних	Опис
Id_problem	int	Ідентифікатор запиту, первинний ключ
Problem_description	varchar(200)	Опис запиту
Problem_solution	varchar(100)	Пропоноване рішення
Problem_registration	date	Дата реєстрації запиту
Deadline	date	Крайній термін вирішення запиту
About_duration	int	Приблизна тривалість вирішення запиту
Fact_duration	int	Фактична тривалість вирішення запиту
Id_priority	int	Пріоритет запиту

Дані про виконавців запитів користувачей зберігаються в таблиці Performers. Опис даної сутності приведено в табл. 3.10.

Таблиця 3.10 – Опис сутності Performers

Атрибут	Тип даних	Опис
Id_performer	int	Ідентифікатор виконавця, первинний ключ
Id_emp	int	Ідентифікатор співробітника, зовнішній ключ
Id_problem	int	Ідентифікатор запиту, зовнішній ключ
About_duration	int	Приблизна тривалість вирішення запиту
Fact_duration	int	Фактична тривалість вирішення запиту

Дані про проекти, які пов’язані з запитамизберігаються в таблиці ProjectProblems. Опис даної сутності приведено в табл. 3.11.

Таблиця 3.11 – Опис сутності ProjectProblems

Атрибут	Тип даних	Опис
Id_ProjectProblem	int	Ідентифікатор проекту, який пов’язаний з запитом, первинний ключ
Id_project	int	Ідентифікатор проекту, зовнішній ключ
Id_problem	int	Ідентифікатор запиту, зовнішній ключ

3.2 Розробка представлень БД

Представлення - це збережений результатний набір запиту. Він доступний як віртуальна таблиця, що складається з результатів запиту. На відміну від звичайних таблиць в реляційній БД, розріз даних не є частиною схеми даних: це динамічна, віртуальна таблиця що є результатом обробки даних з бази. Зміна даних в таблицях бази даних змінює їх у відповідних розрізах.Є два види розрізів: віртуальні та матеріалізовані.

Віртуальні представлення насправді не зберігають результат виконання запиту в базі даних, а щоразу динамічно обчислюють. Матеріалізовані представлення – зберігаються, як і звичайні таблиці. Матеріалізовані представлення мають одну проблему: вони повинні б були оновлюватись при кожному оновлені таблиць від яких вони залежать, проте це не так. Відповідно оновлення табличок, від яких вони залежать, буде проходити повільніше. Як вирішення цієї проблеми можна вказати, що матеріалізовані представлення оновлюватимуться щодня (щогодини).

Кінцеві користувачі не повинні знати про фізичну структуру таблиць, тому всі ключі первинні і зовнішні при виведенні на форму додатків повинні бути приховані. Але зовнішні ключі не просто ховаються, а повинні бути представлені відповідної текстовою інформацією, тому для всіх таблиць з зовнішніми ключами необхідно створити представлення які будуть використовуватися для виведення даних з відповідної таблиці на форму клієнтської програми.

3.3 Збережені процедури БД

Збережена процедура – підпрограма, доступна застосункам, які мають доступ до системи керування реляційними базами даних (СКРБД). Такі процедури зберігаються у словнику даних бази.

До типових застосувань збережених процедур належать валідація даних(вбудована до бази даних) та механізми контролю доступу. Крім того, збережені процедури можуть збирати та централізувати логіку, яку спочатку було реалізовано в застосунках. Для збереження часу та пам'яті, об'ємні чи складні обробки, що вимагають виконання декількох операторів SQL, може бути зібрано до збережених процедур, і усі застосунки можуть викликати ці процедури. Можна використовувати вкладені збережені процедури шляхом виклику одних процедур з інших.

В БД створено 9 збережених процедур для видалення даних з таблиць

Процедура DeleteEmp видаляє записи з таблиці Staff:

@Id_emp smallint,

@m bit,

@k tinyint OUTPUT,

@msg varchar(100) output

Параметрипроцедури:

@Id_emp – код співробітника;

@m – параметр визначає дію процедури при наявності запису в підлеглих таблицях;

@k – код помилки, параметр що повертається;

@msg – повідомлення про помилку, параметр що повертається.

Процедура видаляє запис з кодом @Id_emp з таблиці Staff, а так само з пов'язаних з нею записи в підлеглих таблицях, якщо параметр @m = 0.

Якщо @m = 1, то процедура перевіряє наявність записів в дочірніх таблицях. Якщо ж є записи, пов'язані з видаленим записом з видаляємою таблицеюStaff, то запис не видаляється і повертається повідомлення про помилку, інакше видаляється запис тільки з таблиці Staff.

Процедура DeleteDepartment видаляє записи з таблиці Departmets:

@Id_depart smallint,

@m bit,

@k tinyint OUTPUT,

@msg varchar(100) output

Параметрипроцедури:

@Id_depart – код відділу;

@m – параметр визначає дію процедури при наявності запису в підлеглих таблицях;

@k – код помилки, параметр що повертається;

@msg – повідомлення про помилку, параметр що повертається.

Процедура видаляє запис з кодом @Id_depart з таблиці Departments, а так само з пов'язаних з нею записи в підлеглих таблицях, якщо параметр @m = 0.

Якщо @m = 1, то процедура перевіряє наявність записів в дочірніх таблицях. Якщо ж є записи, пов'язані з видаленим записом з видаляємою таблицею Departments, то запис не видаляється і повертається повідомлення про помилку, інакше видаляється запис тільки з таблиці Departments.

Процедура DeletePost видаляє записи з таблиці Post:

@Id_post smallint,

@m bit,

@k tinyint OUTPUT,

@msg varchar(100) output

Параметрипроцедури:

@Id_post – код посади;

@m – параметр визначає дію процедури при наявності запису в підлеглих таблицях;

@k – код помилки, параметр що повертається;

@msg – повідомлення про помилку, параметр що повертається.

Процедура видаляє запис з кодом @Id_post з таблиці Post, а так само з пов'язаних з нею записи в підлеглих таблицях, якщо параметр @m = 0.

Якщо @m = 1, то процедура перевіряє наявність записів в дочірніх таблицях. Якщо ж є записи, пов'язані з видаленим записом з видаляємою таблицею Post, то запис не видаляється і повертається повідомлення про помилку, інакше видаляється запис тільки з таблиці Post.

Процедура DeleteProblem видаляє записи з таблиці Problems:

@Id_problem smallint,

@m bit,

@k tinyint OUTPUT,

@msg varchar(100) output

Параметрипроцедури:

@Id_depart – код запиту;

@m – параметр визначає дію процедури при наявності запису в підлеглих таблицях;

@k – код помилки, параметр що повертається;

@msg – повідомлення про помилку, параметр що повертається.

Процедура видаляє запис з кодом @Id_problem з таблиці Problems, а так само з пов'язаних з нею записи в підлеглих таблицях, якщо параметр @m = 0.

Якщо @m = 1, то процедура перевіряє наявність записів в дочірніх таблицях. Якщо ж є записи, пов'язані з видаленим записом з видаляємою таблицею Problems, то запис не видаляється і повертається повідомлення про помилку, інакше видаляється запис тільки з таблиці Problems.

Процедура DeleteProject видаляє записи з таблиці Projects:

@Id_project smallint,

@m bit,

@k tinyint OUTPUT,

@msg varchar(100) output

Параметрипроцедури:

@Id_project – код проекту;

@m – параметр визначає дію процедури при наявності запису в підлеглих таблицях;

@k – код помилки, параметр що повертається;

@msg – повідомлення про помилку, параметр що повертається.

Процедура видаляє запис з кодом @Id_project з таблиці Projects, а так само з пов'язаних з нею записи в підлеглих таблицях, якщо параметр @m = 0.

Якщо @m = 1, то процедура перевіряє наявність записів в дочірніх таблицях. Якщо ж є записи, пов'язані з видаленим записом з видаляємою таблицею Projects, то запис не видаляється і повертається повідомлення про помилку, інакше видаляється запис тільки з таблиці Projects.

Процедура DeletePerformer видаляє записи з таблиці Performers:

@Id_performer smallint,

@m bit,

@k tinyint OUTPUT,

@msg varchar(100) output

Параметрипроцедури:

@Id_performer – код виконавця;

@m – параметр визначає дію процедури при наявності запису в підлеглих таблицях;

@k – код помилки, параметр що повертається;

@msg – повідомлення про помилку, параметр що повертається.

Процедура видаляє запис з кодом @Id_performer з таблиці Performers, а так само з пов'язаних з нею записи в підлеглих таблицях, якщо параметр @m = 0.

Якщо @m = 1, то процедура перевіряє наявність записів в дочірніх таблицях. Якщо ж є записи, пов'язані з видаленим записом з видаляємою табли-

цею Performers, то запис не видаляється і повертається повідомлення про помилку, інакше видаляється запис тільки з таблиці Performers.

Процедура DeleteProjectProblem видаляє записи з таблиці ProjectProblems:

@Id_projectProblem smallint,

@m bit,

@k tinyint OUTPUT,

@msg varchar(100) output

Параметрипроцедури:

@Id_projectProblem – код проекту, який пов'язаний з запитом;

@m – параметр визначає дію процедури при наявності запису в підлеглих таблицях;

@k – код помилки, параметр що повертається;

@msg – повідомлення про помилку, параметр що повертається.

Процедура видаляє запис з кодом @Id_projectProblem з таблиці ProjectProblems, а так само з пов'язаних з нею записи в підлеглих таблицях, якщо параметр @m = 0.

Якщо @m = 1, то процедура перевіряє наявність записів в дочірніх таблицях. Якщо ж є записи, пов'язані з видаленим записом з видаляємою таблицею ProjectProblems, то запис не видаляється і повертається повідомлення про помилку, інакше видаляється запис тільки з таблиці ProjectProblems.

Процедура DeleteDepartProject видаляє записи з таблиці DepartProject:

@Id_departProject smallint,

@m bit,

@k tinyint OUTPUT,

@msg varchar(100) output

Параметрипроцедури:

@Id_departProject – код відділу, який пов'язаний з проектом;

@m – параметр визначає дію процедури при наявності запису в підлеглих таблицях;

@k – код помилки, параметр що повертається

@msg – повідомлення про помилку, параметр що повертається.

Процедура видаляє запис з кодом @Id_departProject з таблиці DepartProject, а так само з пов'язаних з нею записи в підлеглих таблицях, якщо параметр @m = 0.

Якщо @m = 1, то процедура перевіряє наявність записів в дочірніх таблицях. Якщо ж є записи, пов'язані з видаленим записом з видаляємою таблицею DepartProject, то запис не видаляється і повертається повідомлення про помилку, інакше видаляється запис тільки з таблиці DepartProject.

Процедура DeleteStaffProject видаляє записи з таблиці StaffProject:

@Id_staffProject smallint,

@m bit,

@k tinyint OUTPUT,

@msg varchar(100) output

Параметрипроцедури:

@Id_ staffProject – код співробітника, який пов'язаний з проектом;

@m – параметр визначає дію процедури при наявності запису в підлеглих таблицях;

@k – код помилки, параметр що повертається

@msg – повідомлення про помилку, параметр що повертається.

Процедура видаляє запис з кодом @Id_staffProject з таблиці StaffProject, а так само з пов'язаних з нею записи в підлеглих таблицях, якщо параметр @m = 0.

Якщо @m = 1, то процедура перевіряє наявність записів в дочірніх таблицях. Якщо ж є записи, пов'язані з видаленим записом з видаляємою таблицею StaffProject, то запис не видаляється і повертається повідомлення про помилку, інакше видаляється запис тільки з таблиці StaffProject.

3.4 Програмний продукт

Як уже згадувалося раніше, додаток розроблялося на мові C# в середовищі розробки MicrosoftVisualStudio.

Підсумковий додаток містить у собі 20 форм для відображення, додавання, зміни та видалення інформації з БД, що викликаються за допомогою вибору пункту рядка меню або натискання накнопку на формі, а також формою авторизації. Перша форма додатку,що з'являється при запуску клієнта, показана на рис. 3.1.

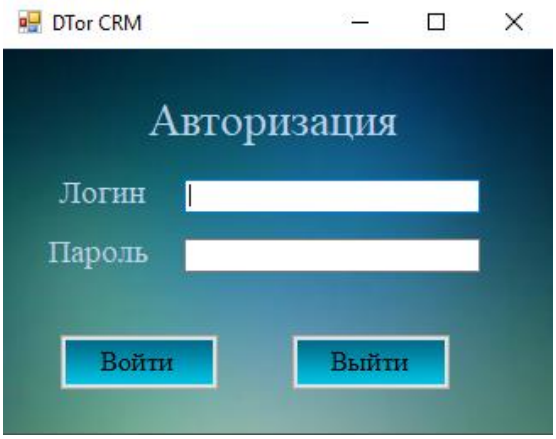

Рисунок 3.1 – Форма авторизації

При на натисканні на кнопку “Войти”користувач буде перенаправлений на головну форму програми. Якщо користувач ввів неправильні дані, то вискочить відповідна помилка. При натисканны на кнопку “Выйти” програма закриється. Головна форма програми показана на рис. 3.2 – 3.4.

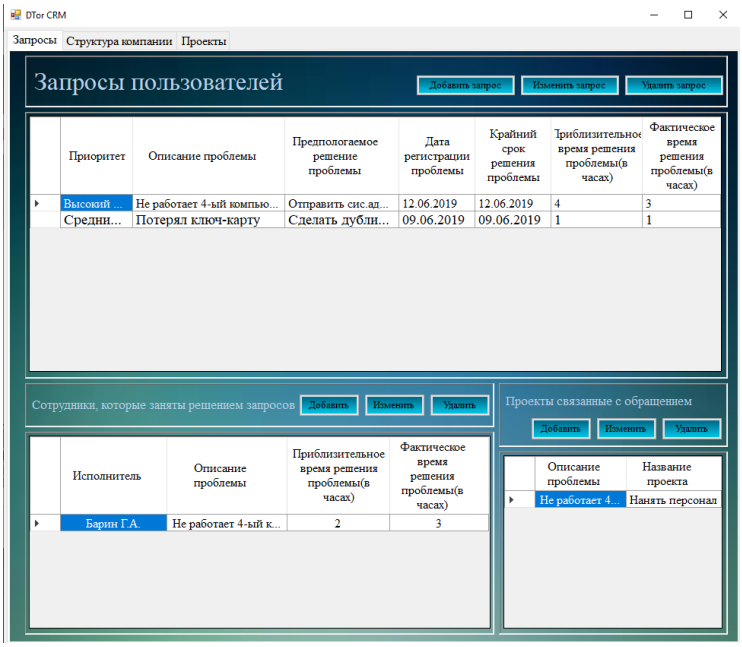

Рисунок 3.2 – Головна форма, вкладка “Запросы”

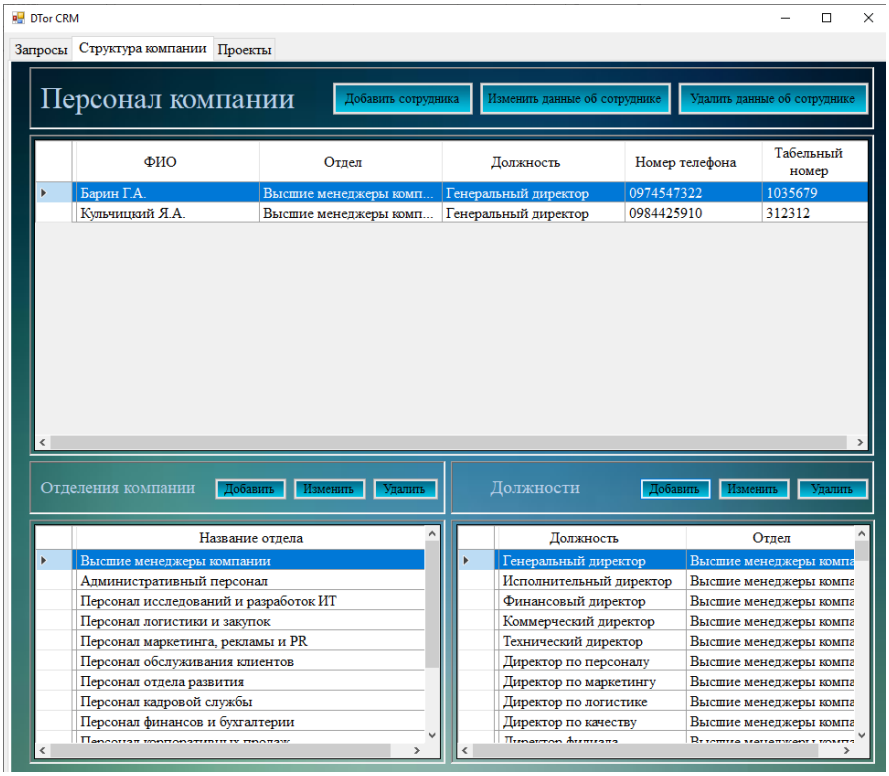

Рисунок 3.3 – Головна форма, вкладка “Структура компании”

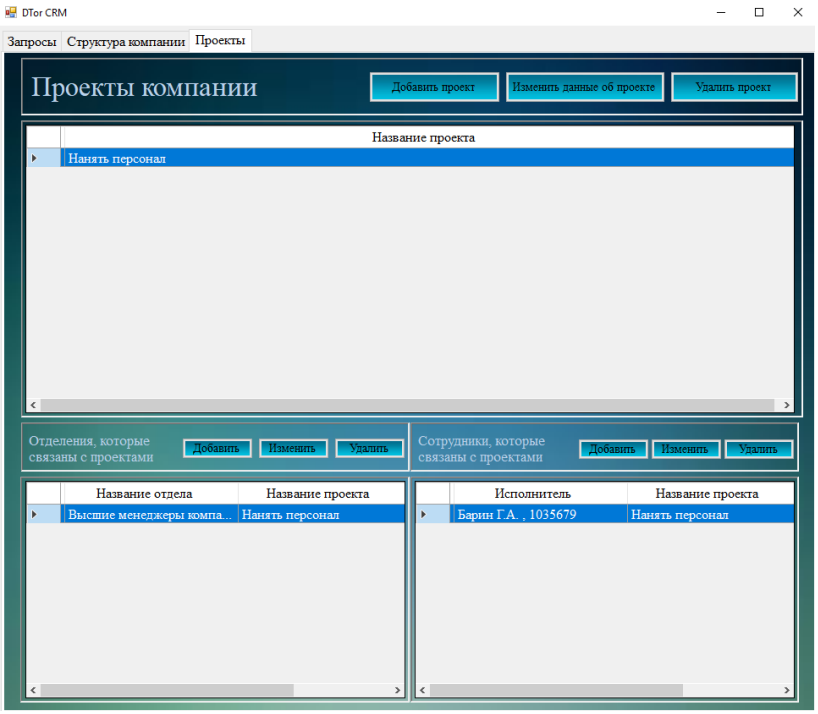

Рисунок 3.4 – Головна форма, вкладка “Проекты”

На головній формі відображені 9 таблиць, по 3 на кожній вкладці з даними запитів користувачів, співробітниками, які заняті запитами, проекти, які пов’язані з запитами, персоналом компанії, відділенями компанії, посадами компанії, проектами, співробітниками, які пов’язані з проектами, відділеннями, які пов’язані з проектами.

Біля кожної таблиці є по 3 кнопки – “Добавить”, “Изменить”,”Удалить”, які відповідають за додавання, оновлення та видалення даних з відповідної таблиці.

При натискані на кнопку “Добавитьсотрудника” біля таблиці “Запросы пользователей” користувач буде перенаправлений на форму додавання даних до таблиціProblems. Форма додавання до таблиці Problems зображена на рис. 3.5.

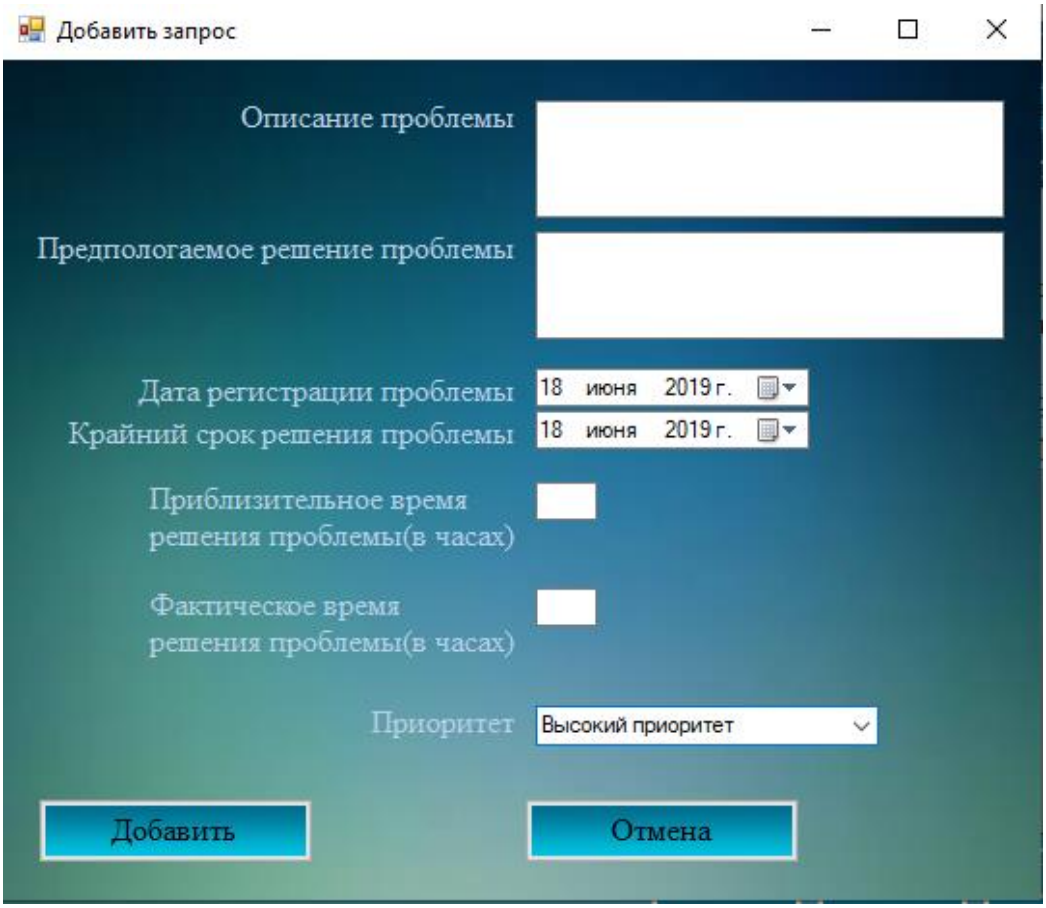

Рисунок 3.5 – Форма додавання даних до таблиці Problems

При натискані на кнопку “Добавить” біля таблиці “Сотрудники, которые заняты решением запросов” користувач буде перенаправлений на форму додавання даних до таблиці Performers.Форма додавання до таблиці Performerszображена на рис. 3.6.

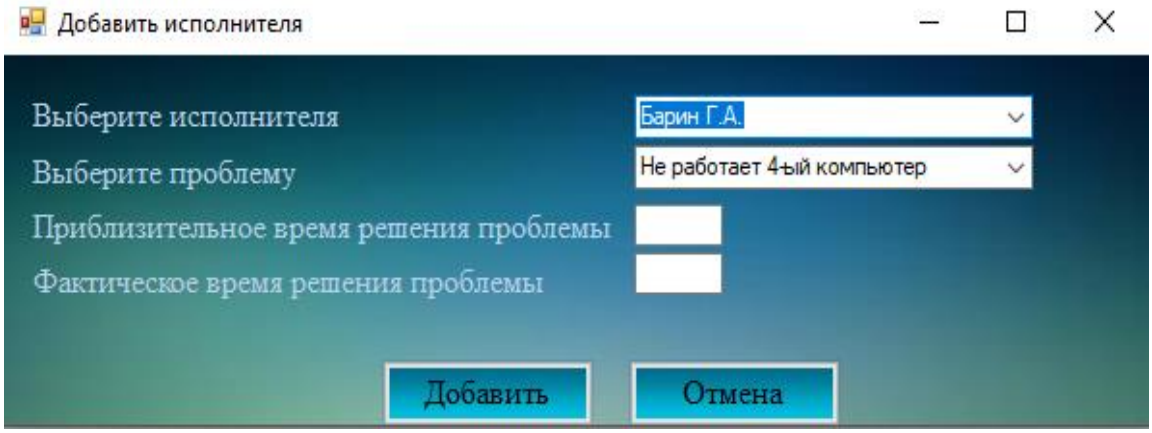

Рисунок 3.6 – Форма додавання данихдо таблиці Performers

При натискані на кнопку “Добавить” біля таблиці “Проекты связанные с обращением” користувач буде перенаправлений на форму додавання даних до таблиці ProjectProblems. Форма додавання до таблиці ProjectProblemszображена на рис. 3.7.

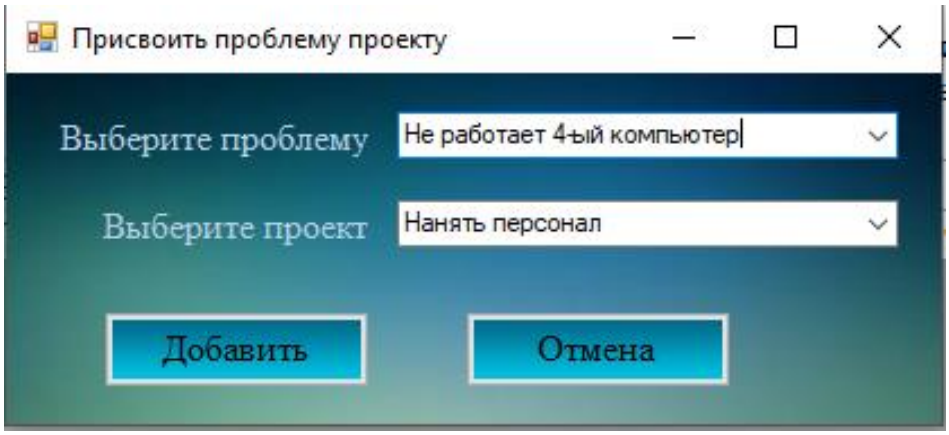

Рисунок 3.7 – Форма додавання даних до таблиці ProjectProblems

При натискані на кнопку “Добавить” біля таблиці “Персонал компании” користувач буде перенаправлений на форму додавання даних до таблиці Staff. Форма додавання до таблиці Staff зображена на рис. 3.8.

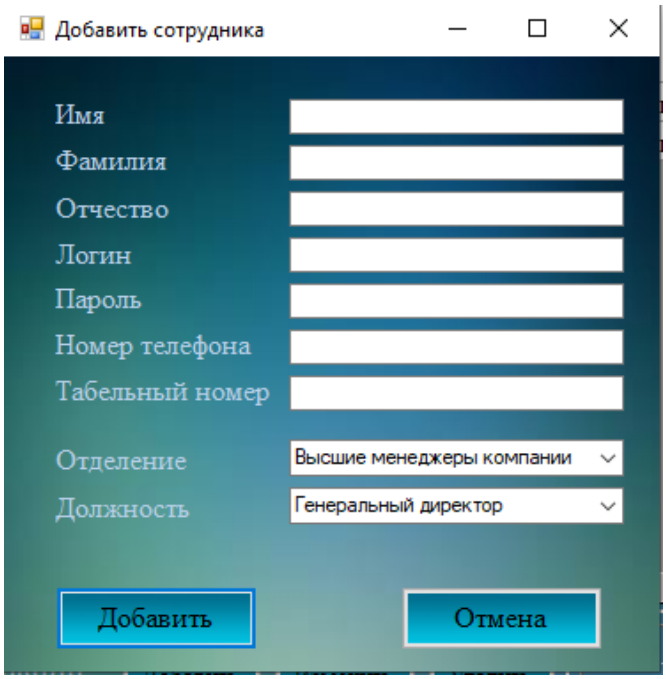

Рисунок 3.8 – Форма додавання данихдо таблиці Staff

При натискані на кнопку “Добавить” біля таблиці “Отделения компании” користувач буде перенаправлений на форму додавання даних до таблиціDepartments. Форма додавання до таблиці Departments зображена на рис. 3.9.

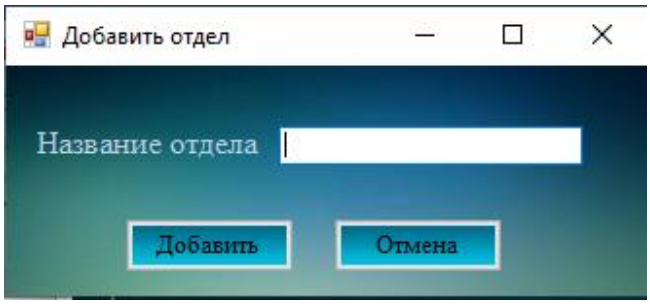

Рисунок 3.9 – Форма додавання даних до таблиці Departments

При натискані на кнопку “Добавить” біля таблиці “Должности” користувач буде перенаправлений на форму додавання даних до таблиціPost. Форма додавання до таблиці Postзображена на рис. 3.10.

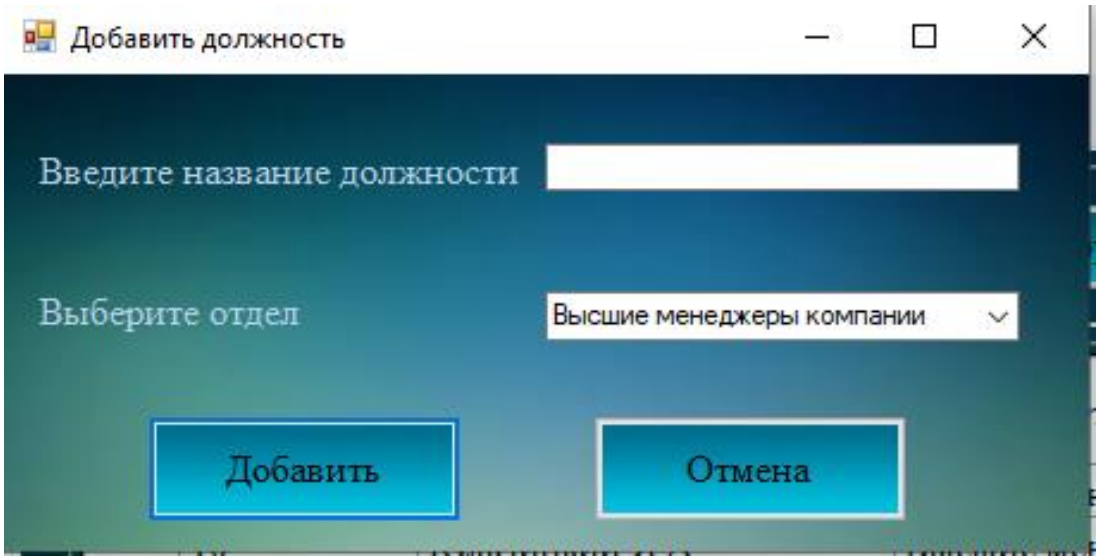

Рисунок 3.10 – Форма додавання до таблиці Post

При натискані на кнопку “Добавить” біля таблиці “Проекты компании” користувач буде перенаправлений на форму додавання даних до таблиціProjects. Форма додавання до таблиці Projectszображена на рис. 3.11.

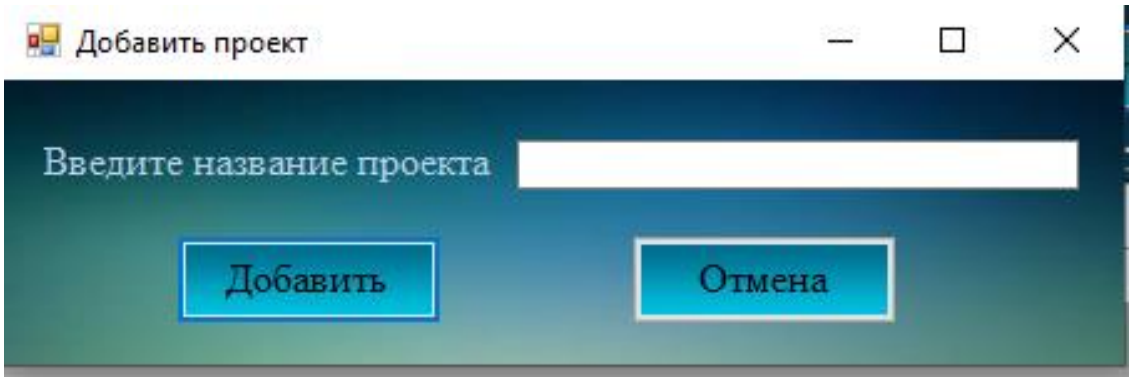

Рисунок 3.11 – Форма додавання до таблиці Projects

При натискані на кнопку “Добавить” біля таблиці “Отделения, которые связанны с проектами” користувач буде перенаправлений на форму додавання даних до таблиціDepartProjects. Форма додавання до таблиці DepartProjectsзображена на рис. 3.12.

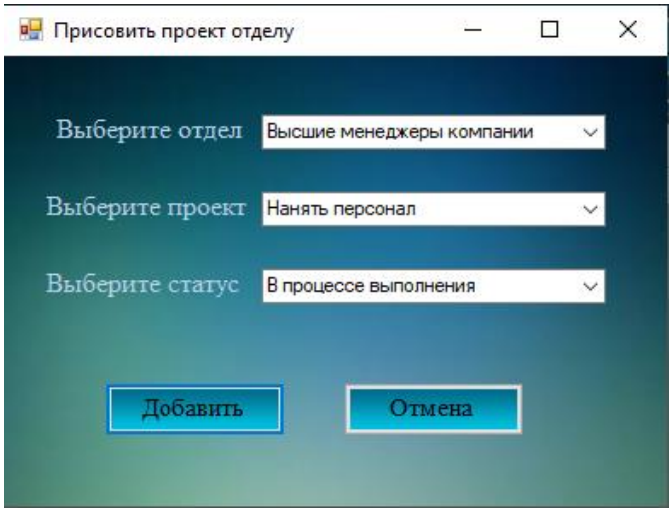

Рисунок 3.12 – Форма додавання до таблиці DepartProjects

При натискані на кнопку “Добавить” біля таблиці “Сотрудники, которые связанны с проектами” користувач буде перенаправлений на форму додавання даних до таблиціStaffProjects. Форма додавання до таблиці StaffProjectsзображена на рис. 3.13.

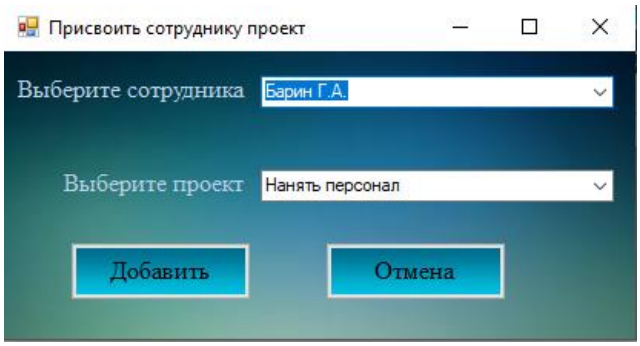

Рисунок 3.13 – Форма додавання до таблиці StaffProjects

При натискані на кнопку “Изменить”біля таблиці “Запросы пользователей” користувач буде перенаправлений на форму оновлення даних таблиціProblems.Форма оновлення до таблиці Problemsзображена на рис. 3.14.

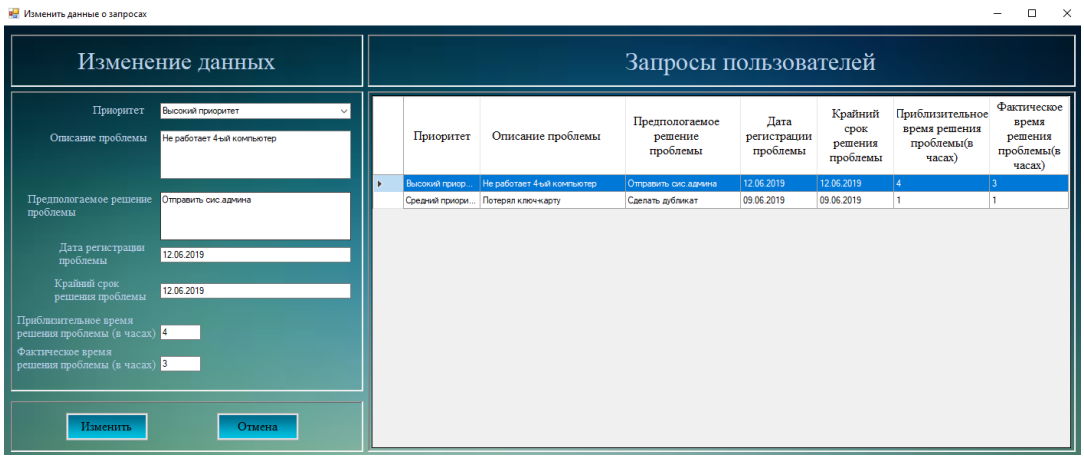

Рисунок 3.14 – Форма оновлення до таблиці Problems

При натискані на кнопку “Изменить”біля таблиці “Сотрудники, которые заняты решением запросов” користувач буде перенаправлений на форму оновлення даних таблиціPerformers. Форма оновлення до таблиці Performerszображена на рис. 3.15.

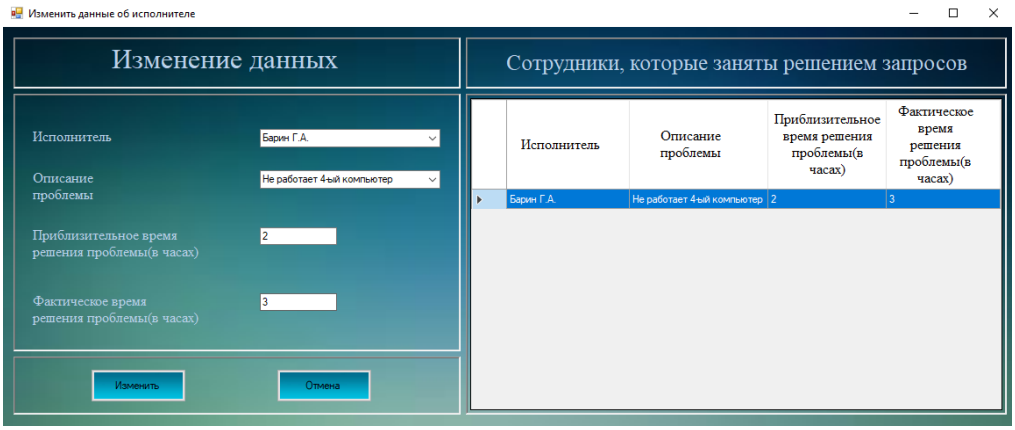

Рисунок 3.15 – Форма «Изменить данные об исполнителе»

При натискані на кнопку “Изменить”біля таблиці “Проекты связазанные с запросами” користувач буде перенаправлений на форму оновлення даних таблиціProjectProblems. Форма оновлення до таблиці ProjectProblemsзображена на рис. 3.16.

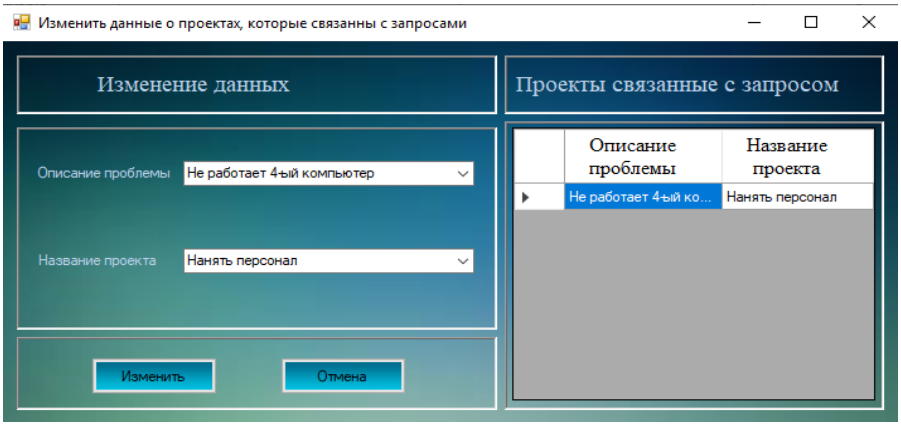

Рисунок 3.16 – Форма оновлення до таблиці ProjectProblems

При натискані на кнопку “Изменить”біля таблиці “Персонал компании” користувач буде перенаправлений на форму оновлення даних таблиціStaff. Форма оновлення до таблиці Staffзображена на рис. 3.17.

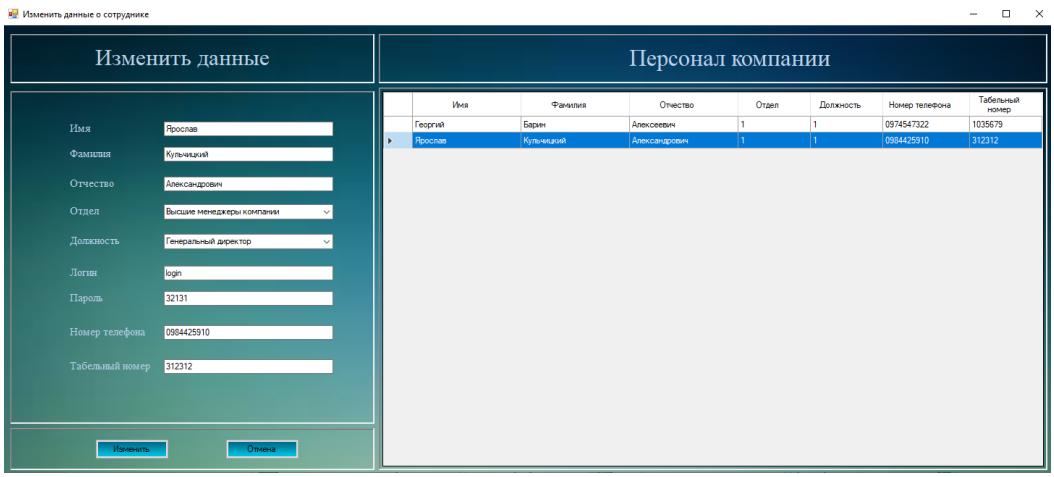

Рисунок 3.17 – Форма оновлення до таблиці Staff

При натискані на кнопку “Изменить”біля таблиці “Отделения компании” користувач буде перенаправлений на форму оновлення даних таблиціDepartments. Форма оновлення до таблиці Departmentsзображена на рис. 3.18.

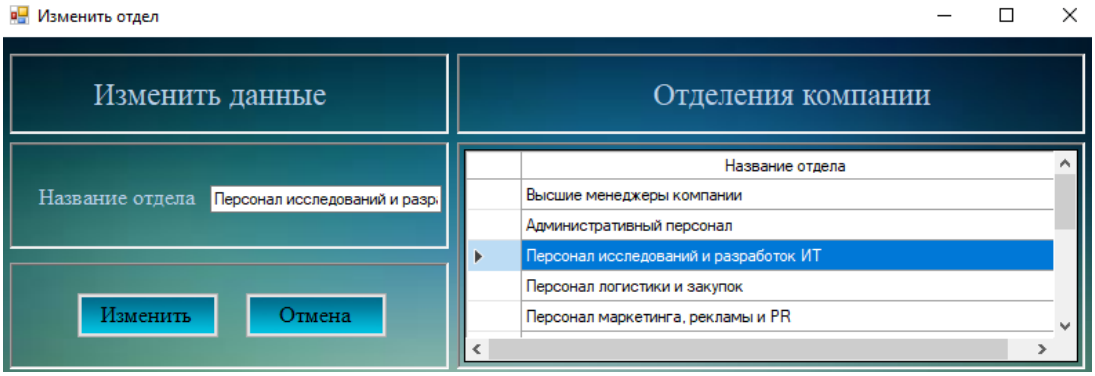

Рисунок 3.18 – Форма оновлення до таблиці Departments

При натискані на кнопку “Изменить”біля таблиці “Должности” користувач буде перенаправлений на форму оновлення даних таблиціPost. Форма оновлення до таблиці Postзображена на рис. 3.19.

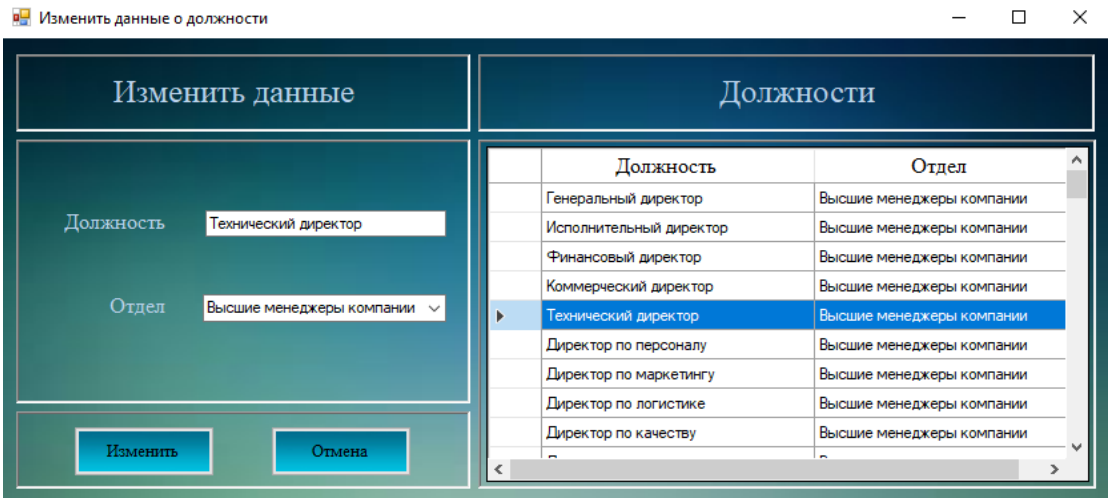

Рисунок 3.19 – Форма «Изменить данные о должности»

При натискані на кнопку “Изменить”біля таблиці “Проекты компании” користувач буде перенаправлений на форму оновлення даних таблиціProjects. Форма оновлення до таблиці Projectsзображена на рис. 3.20.

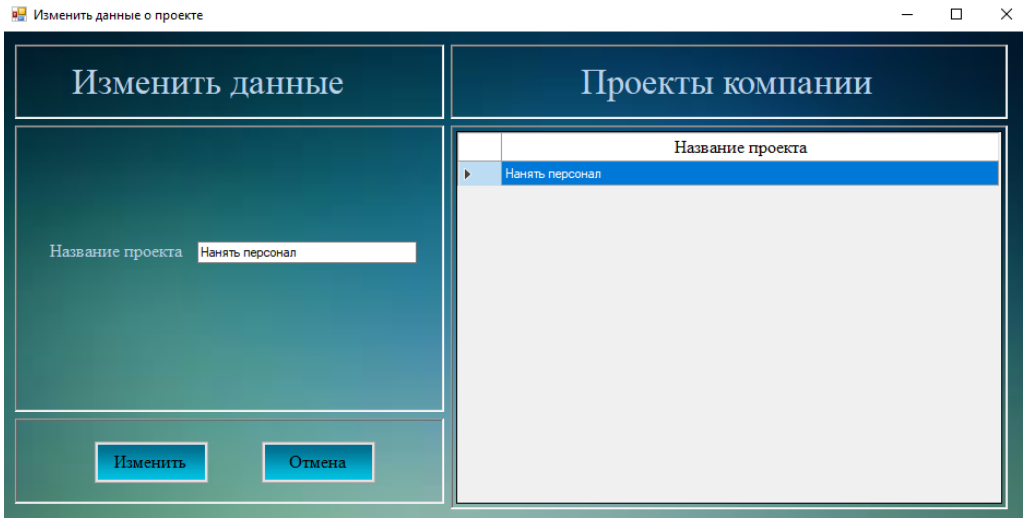

Рисунок 3.20 – Форма «Изменить данные о проекте»

При натискані на кнопку “Изменить”біля таблиці “Отделения, которые связанны с проетками” користувач буде перенаправлений на форму оновлення даних таблиціDepartProjects. Форма оновлення до таблиці DepartProjectsзображена на рис. 3.21.

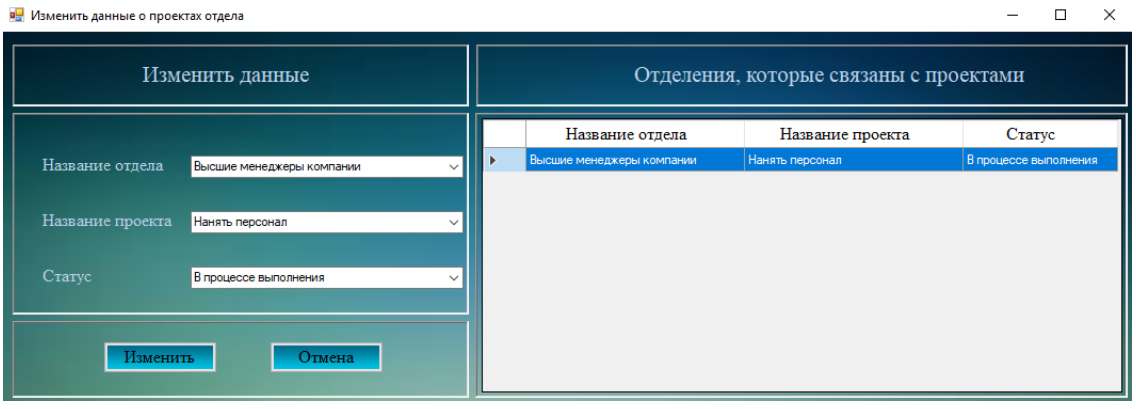

Рисунок 3.21 – Форма «Изменить данные о проектах отдела»

При натискані на кнопку “Изменить”біля таблиці “Сотрудники, которые связанны с проетками” користувач буде перенаправлений на форму оновлення даних таблиціStaffProjects. Форма оновлення до таблиці StaffProjectszображена на рис. 3.22.

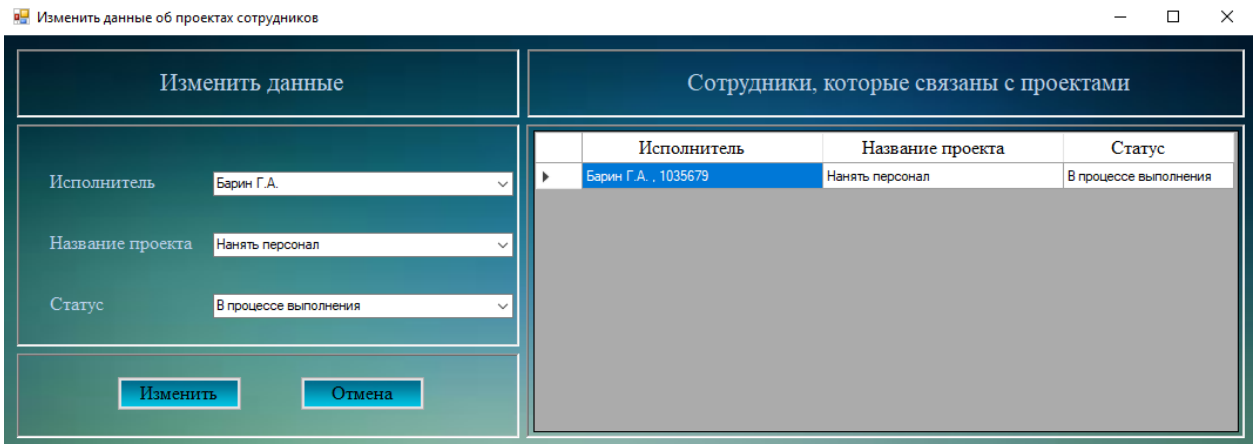

Рисунок 3.22 – Форма оновлення до таблиці StaffProjects

Щоб видалити дані з таблиці потрібно клацнути на строку, потім на кнопку «Удалить».

ВИСНОВКИ

В ході даної роботи була дослідженна система управління взаємовідносинамиза клієнтами та розроблена власна CRM-система для підрозділи по розробці прикладного ПЗ. Для досягнення кінцевого результату були сформульовані і вирішені наступні завдання:

- розкрито теоретичні основи CRM-систем;
- виконано порівняльний огляд існуючих аналогів;
- проаналізовано бізнес-процеси підрозділу по розробці ПО;
- здійснено проектування і розробка БД для майбутньої системи;
- створений клієнт для взаємодії зі сховищем даних;
- виконано аналіз ефективності розробленої CRM-системи.

Основна мета впровадження CRM - це підвищення обсягу продажів і прибутку. CRM-система дозволяє побудувати систему управління продажами, підвищити лояльність клієнтів і збільшити обсяг продажів.

CRM-системи необхідні підприємствам, які хочуть побудувати ефективну систему управління продажами і у яких клієнт - єдине джерело доходу підприємства. Нові технології CRM в роботі відділу продажів, маркетингу і сервісу є запорукою добробуту підприємства. Також CRM необхідні підприємствам, що працюють на ринку з високою конкуренцією, тому що підвищення лояльності клієнтів - це додатковий важіль в конкурентній боротьбі.

Нарешті, CRM-системи будуть корисні в організаціях, у яких процес продажу розтягнутий в часі, і включає кілька етапів, наприклад пошук і залучення клієнта, презентація товару, підготовка та узгодження параметрів угоди, договору, оплата і поставка товару, гарантійне обслуговування та ін.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. C-sharp – Вікіпедія . URL: https://uk.wikipedia.org/wiki/C_Sharp (дата звернення 03.04.2019)
2. .NETFramework – Вікіпедія. URL: https://uk.wikipedia.org/wiki/.NET_Framework (дата звернення: 04.04.2019)
3. MicrosoftVisualStudio – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio(дата звернення 11.04.2019).
4. MicrosoftSQLServer – Вікіпедія URL: https://uk.wikipedia.org/wiki/Microsoft_SQL_Server(дата звернення 11.04.2019).

ДОДАТОК А

Фізична схема бази даних

```
CREATE TABLE Departments(
Id_depart int IDENTITY PRIMARY KEY,
Name_depart varchar(50) NOT NULL
)


CREATE TABLE Post(
Id_post int IDENTITY PRIMARY KEY,
Name_post varchar(50),
Id_depart int REFERENCES Departments
)


CREATE TABLE Staff(
Id_emp int IDENTITY PRIMARY KEY,
First_name varchar(25),
Last_name varchar(25),
Middle_name varchar(25),
Login_emp varchar(25) UNIQUE,
Emp_phone varchar(25) UNIQUE,
Emp_number varchar(25) UNIQUE,
Id_depart int REFERENCES Departments,
Id_post int REFERENCES Post
)


CREATE TABLE Projects(
Id_project int IDENTITY PRIMARY KEY,
Name_project varchar(50)
)


CREATE TABLE DepartProject(
Id int IDENTITY PRIMARY KEY,
Id_depart int REFERENCES Departments,
Id_project int REFERENCES Projects,
Id_status int REFERENCES Status
)


CREATE TABLE StaffProject(
Id int IDENTITY PRIMARY KEY,
Id_emp int REFERENCES Saff,
Id_project int REFERENCES Projects,
Id_status int REFERENCES Status
)


CREATE TABLE Priorities(
Id_priority int IDENTITY PRIMARY KEY,
Value varchar(50)
)


CREATE TABLE Problems(
Id_problem int IDENTITY PRIMARY KEY,
Problem_description varchar(200),
Problem_solution varchar(100),
Problem_registration DATE,
Deadline DATE,
About_duration int,
Fact_duration int,
Id_priorit int REFERENCES Priorities
```

```
)


CREATE TABLE ProjectProblems(
Id_ProjectProblem int IDENTITY PRIMARY KEY,
Id_project int REFERENCES Projects,
Id_problem int REFERENCES Problems
)

CREATE TABLE Performers(
Id_performer int IDENTITY PRIMARY KEY,
About_duration int,
Fact_duration int,
Id_emp int REFERENCES Staff,
Id_problem int REFERENCES Problems
)
```

ДОДАТОК Б

Представленнябазиданих

Б.1 SQL-код создания представленияPostView

```
CREATE VIEW PostView AS
SELECT
      Post.Name_post AS 'Должность',
      Departments.Name_depart As 'Отдел,
      Id_post AS ‘Id’
FROM Post INNER JOIN Departments ON Post.Id_depart =
Departments.Id_depart
```

Б.2 SQL-код создания представленияViewStaff

```
CREATE VIEW ViewStaff AS
SELECT
      (Last_name) + ' ' + LEFT(First_name,1) + '.' +
LEFT(Middle_name,1) + '.' as [ФИО],
      d.Name_depart as [Отдел],
      p.Name_Post as [Должность],
      Emp_phone as [Номер телефона],
      Emp_number as [Табельный номер],
      Id_emp AS [Id]
FROM
      Staff s JOIN Departments d ON s.Id_depart = d.Id_depart JOIN
      Post p ON s.Id_post = p.Id_post
```

Б.3 SQL-код создания представленияViewDepart

```
CREATE VIEW ViewDepart AS
SELECT
      Name_depart AS [Название отдела],
      Id_depart AS [Id]
FROM
Departments
```

Б.4 SQL-код создания представленияViewProblems

```
CREATE VIEW ViewProblems AS
SELECT
      pr.Value AS [Приоритет],
      Problem_description AS [Описание проблемы],
      Problem_solution AS [Предпологаемое решение проблемы],
      convert(char(10), Problem_registration, 104) AS [Дата регистрации
проблемы],
      convert(char(10), Deadline, 104) AS [Крайний срок решения про-
блемы],
      About_duration AS [Приблизительное время решения проблемы(в ча-
сах)],
      Fact_duration AS [Фактическое время решения проблемы(в часах)],
      Id_problem AS ‘Id’
FROM
Problems p JOIN Priorities pr ON p.Id_priority = pr.Id_priority
```

Б.5 SQL-код создания представленияViewPerformers

```
CREATE VIEW ViewPerformers AS
SELECT
```

```
      (s.Last_name) + ' ' + LEFT(s.First_name,1) + '.' +
LEFT(s.Middle_name,1) + '.' as [Исполнитель],
      pr.Problem_description AS [Описание проблемы],
      pr.About_duration AS [Приблизительное время решения проблемы(в
часах)],
      pr.Fact_duration AS [Фактическое время решения проблемы(в ча-
сах)],
      Id_performer AS ‘Id’
FROM
Performers p JOIN Staff s ON p.Id_emp = s.Id_emp JOIN Problems pr ON
p.Id_problem=pr.Id_problem
```

Б.6 SQL-код создания представленияViewProjects

```
CREATE VIEW ViewProjects AS
SELECT
      Name_project AS [Название проекта],
      Id_project AS [Id]
FROM
Projects
```

Б.7 SQL-код создания представленияViewStatus

```
CREATE VIEW ViewStatus AS
SELECT
      Name_status AS [Статус],
      Id_status AS [Id]
FROM
Status
```

Б.8 SQL-код создания представленияViewProjectProblem

```
CREATE VIEW ViewProjectProblem AS
SELECT
      pr.Problem_description AS [Описание проблемы],
      pro.Name_project AS [Название проекта],
      Id_projectProblem AS [Id]
FROM
ProjectProblems pp JOIN Problems pr ON pp.Id_problem = pr.Id_problem
JOIN Projects pro ON pp.Id_project=pro.Id_project
```

Б.9 SQL-код создания представленияViewDepartProject

```
CREATE VIEW ViewDepartProject AS
SELECT
      d.Name_depart AS [Название отдела],
      p.Name_project AS [Название проекта],
      Id AS [Id]
FROM
DepartProject dp JOIN Departments d ON dp.Id_depart=d.Id_depart JOIN
Projects p ON dp.Id_project=p.Id_project
```

Б.10 SQL-код создания представленияViewStaffProject

```
CREATE VIEW ViewStaffProject AS
SELECT
      (s.Last_name) + ' ' + LEFT(s.First_name,1) + '.' +
LEFT(s.Middle_name,1) + '.' as [Исполнитель],
      p.Name_project AS [Название проекта],
      Id AS [Id]
FROM
StaffProject sp JOIN Staff s ON sp.Id_emp=s.Id_emp JOIN Projects p ON
sp.Id_project=p.Id_project
```

ДОДАТОК В

Серверне програмне забезпечення

В.1 Збережена процедура видалення даних з таблиціPost

```
CREATE PROCEDURE DeletePost
@Id_post smallint,
@m bit,
@k tinyint output,
@msg varchar(100) output
AS
BEGIN
      IF @m = 0
      BEGIN
            DELETE Staff WHERE Id_post = @Id_post
DELETE Post WHERE Id_post = @Id_post
SET @k = 0
SET @msg = 'OK'
      END
ELSE
BEGIN
      SET @k = (SELECT k = COUNT(*) FROM Staff
WHERE Id_post = @Id_post)
IF @k = 0
BEGIN
      SET @msg = 'OK'
      DELETE Post WHERE Id_post = @Id_post
END
ELSE
      SET @msg = 'В таблице Staff существует ' + STR(@k,4) +
            ' записей для этой области'
      END
END
```

В.2 Збережена процедура видалення даних з таблиці Departments

```
CREATE PROCEDURE DeleteDepartment
@Id_depart smallint,
@m bit,
@k tinyint output,
@msg varchar(100) output
AS
BEGIN
      IF @m = 0
      BEGIN
                  DELETE DepartProject WHERE Id_depart = @Id_depart
            DELETE Staff WHERE Id_depart = @Id_depart
DELETE Departments WHERE Id_depart = @Id_depart
SET @k = 0
SET @msg = 'OK'
      END
ELSE
BEGIN
      SET @k = (SELECT k = COUNT(*) FROM Staff s JOIN DepartProject dp
on s.Id_depart = dp.Id_depart
WHERE s.Id_depart = @Id_depart)
IF @k = 0
BEGIN
      SET @msg = 'OK'
      DELETE Departments WHERE Id_depart = @Id_depart
END
ELSE
      SET @msg = 'В таблице Departments существует ' + STR(@k,4) +
            ' записей для этого отдела'
      END
END
```

В.3 Збережена процедура видалення даних з таблиці Staff

```
CREATE PROCEDURE DeleteEmp
@Id_emp smallint,
@m bit,
@k tinyint output,
@msg varchar(100) output
AS
BEGIN
	IF @m = 0
	BEGIN
			DELETE StaffProject WHERE Id_emp = @Id_emp
		DELETE Performers WHERE Id_emp = @Id_emp
DELETE Staff WHERE Id_emp = @Id_emp
SET @k = 0
SET @msg = 'OK'
	END
ELSE
BEGIN
	SET @k = (SELECT k = COUNT(*) FROM Performers p JOIN StaffProject
sp on p.Id_emp = sp.Id_emp
WHERE p.Id_emp = @Id_emp)
IF @k = 0
BEGIN
	SET @msg = 'OK'
	DELETE Staff WHERE Id_emp = @Id_emp
END
ELSE
	SET @msg = 'В таблице Staff существует ' + STR(@k,4) +
		' записей для этого сотрудника'
	END
END
```

В.4 Збережена процедура видалення даних з таблиці Problems

```
CREATE PROCEDURE DeleteProblem
@Id_problem smallint,
@m bit,
@k tinyint output,
@msg varchar(100) output
AS
BEGIN
	IF @m = 0
	BEGIN
			DELETE ProjectProblems WHERE Id_problem = @Id_problem
		DELETE Performers WHERE Id_problem = @Id_problem
DELETE Problems WHERE Id_problem = @Id_problem
SET @k = 0
SET @msg = 'OK'
	END
ELSE
BEGIN
	SET @k = (SELECT k = COUNT(*) FROM Performers p JOIN
ProjectProblems pp on p.Id_problem = pp.Id_problem
WHERE p.Id_problem = @Id_problem)
IF @k = 0
BEGIN
	SET @msg = 'OK'
	DELETE Problems WHERE Id_problem = @Id_problem
END
ELSE
	SET @msg = 'В таблице Problem существует ' + STR(@k,4) +
		' записей для этой проблемы'
	END
```

```
END
```

В.5 Збережена процедура видалення даних з таблиці Projects

```
CREATE PROCEDURE DeleteProject
@Id_project smallint,
@m bit,
@k tinyint output,
@msg varchar(100) output
AS
BEGIN
      IF @m = 0
      BEGIN
                  DELETE ProjectProblems WHERE Id_project = @Id_project
            DELETE StaffProject WHERE Id_project = @Id_project
            DELETE DepartProject WHERE Id_project = @Id_project
DELETE Projects WHERE Id_project = @Id_project
SET @k = 0
SET @msg = 'OK'
END
END
```

В.6 Збережена процедура видалення даних з таблиці StaffProject

```
CREATE PROCEDURE DeleteStaffProject
@Id smallint,
@m bit,
@k tinyint output,
@msg varchar(100) output
AS
BEGIN
      IF @m = 0
      BEGIN

DELETE StaffProject WHERE Id = @Id
SET @k = 0
SET @msg = 'OK'
END
END
```

В.7 Збережена процедура видалення даних з таблиці DepartProject

```
CREATE PROCEDURE DeleteDepartProject
@Id smallint,
@m bit,
@k tinyint output,
@msg varchar(100) output
AS
BEGIN
      IF @m = 0
      BEGIN

DELETE DepartProject WHERE Id = @Id
SET @k = 0
SET @msg = 'OK'
END
END
```

В.8 Збережена процедура видалення даних з таблиці ProjectProblem

```
CREATE PROCEDURE DeleteProjectProblem
@Id smallint,
@m bit,
@k tinyint output,
@msg varchar(100) output
AS
BEGIN
      IF @m = 0
      BEGIN

DELETE ProjectProblems WHERE Id_ProjectProblem = @Id
SET @k = 0
SET @msg = 'OK'
END
END
```

В.9 Збережена процедура видалення даних з таблиці ProjectProblem

```
CREATE PROCEDURE DeletePerformer
@Id_performer smallint,
@m bit,
@k tinyint output,
@msg varchar(100) output
AS
BEGIN
      IF @m = 0
      BEGIN

DELETE Performers WHERE Id_performer = @Id_performer
SET @k = 0
SET @msg = 'OK'
END
END
```

ДОДАТОК Г

Лістинг програми

Г.1 Форма авторизації

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace CRMka
{
    public partial class authorization : Form
    {
        private SqlDataReader db_reader;
        private SqlConnection db_con;
        private SqlCommand db_cmd;
        public void SetConnection()
        {
            db_con = new SqlConnection(@"Data Source=DESKTOP-
8MQT1B7\SQLEXPRESS;Initial Catalog=CRMDB;Integrated Security=True;");
        }


        private void ExecuteQuery(string txtQuery)
        {
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            db_cmd.CommandText = txtQuery;
            db_cmd.ExecuteNonQuery();
            db_con.Close();

        }

        private DataSet DS = new DataSet();
        private DataTable DT = new DataTable();

        public authorization()
        {
            InitializeComponent();
        }

        private void ExitButton_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void OkButton_Click(object sender, EventArgs e)
        {
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            string check = "SELECT COUNT(*) FROM Staff WHERE Login_emp='" +
textBox1.Text + "' AND Emp_password='" + textBox2.Text + "'";
            db_cmd.CommandText = check;
            db_reader = db_cmd.ExecuteReader();

            if (Convert.ToInt32(check)!= 0) {
                authorization aut = new authorization();
```

```
                aut.Close();
                MainForm mf = new MainForm();
                mf.Show();
            }
            else
            {
                MessageBox.Show("Неверный логин или пароль");
            }
            db_con.Close();
        }
    }
}
```

Г.2 Головна форма

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace CRMka
{
    public partial class MainForm : Form
    {
        private SqlCommand problemDelCmd;
        private SqlCommand performerDelCmd;
        private SqlCommand projprobDelCmd;
        private SqlCommand staffDelCmd;
        private SqlCommand departDelCmd;
        private SqlCommand postDelCmd;
        private SqlCommand projectDelCmd;
        private SqlCommand departprojDelCmd;
        private SqlCommand staffprojDelCmd;

        public void SetConnection()
        {
            db_con     =     new     SqlConnection(@"Data     Source=DESKTOP-
8MQT1B7\SQLEXPRESS;Initial Catalog=CRMDB;Integrated Security=True;");
        }

        private void ExecuteQuery(string txtQuery)
        {
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            db_cmd.CommandText = txtQuery;
            db_cmd.ExecuteNonQuery();
            db_con.Close();
        }


        private SqlConnection db_con;
        private SqlCommand db_cmd;
        private DataSet DS = new DataSet();
        private DataTable DT = new DataTable();

        public MainForm()
        {
            InitializeComponent();
            init_problemDelCmd();
```

```
            init_performerDelCmd();
            init_projprobDelCmd();
            init_staffDelCmd();
            init_departDelCmd();
            init_postDelCmd();
            init_projectDelCmd();
            init_departprojDelCmd();
            init_staffprojDelCmd();
        }
        private void init_problemDelCmd()
        {
            problemDelCmd = new System.Data.SqlClient.SqlCommand();
            problemDelCmd.Connection                                    =
this.viewProblemsTableAdapter.Connection;
            problemDelCmd.CommandText = "DeleteProblem";
            problemDelCmd.CommandType                   =                   Sys-
tem.Data.CommandType.StoredProcedure;
            problemDelCmd.Parameters.Add("@Id_problem",                     Sys-
tem.Data.SqlDbType.SmallInt);
            problemDelCmd.Parameters.Add("@m", System.Data.SqlDbType.Bit);
            problemDelCmd.Parameters.Add("@k",                              Sys-
tem.Data.SqlDbType.TinyInt).Direction                   =                   Sys-
tem.Data.ParameterDirection.InputOutput;
            problemDelCmd.Parameters.Add("@msg",                            Sys-
tem.Data.SqlDbType.VarChar,           100).Direction          =           Sys-
tem.Data.ParameterDirection.InputOutput;
        }

        private void init_performerDelCmd()
        {
            performerDelCmd = new System.Data.SqlClient.SqlCommand();
            performerDelCmd.Connection                                      =
this.viewPerformersTableAdapter.Connection;
            performerDelCmd.CommandText = "DeletePerformer";
            performerDelCmd.CommandType                  =                  Sys-
tem.Data.CommandType.StoredProcedure;
            performerDelCmd.Parameters.Add("@Id_performer",                 Sys-
tem.Data.SqlDbType.SmallInt);
            performerDelCmd.Parameters.Add("@m", System.Data.SqlDbType.Bit);
            performerDelCmd.Parameters.Add("@k",                            Sys-
tem.Data.SqlDbType.TinyInt).Direction                   =                   Sys-
tem.Data.ParameterDirection.InputOutput;
            performerDelCmd.Parameters.Add("@msg",                          Sys-
tem.Data.SqlDbType.VarChar,           100).Direction          =           Sys-
tem.Data.ParameterDirection.InputOutput;
        }

        private void init_projprobDelCmd()
        {
            projprobDelCmd = new System.Data.SqlClient.SqlCommand();
            projprobDelCmd.Connection                                       =
this.viewProjectProblemTableAdapter.Connection;
            projprobDelCmd.CommandText = "DeleteProjectProblem";
            projprobDelCmd.CommandType                   =                  Sys-
tem.Data.CommandType.StoredProcedure;
            projprobDelCmd.Parameters.Add("@Id",                            Sys-
tem.Data.SqlDbType.SmallInt);
            projprobDelCmd.Parameters.Add("@m", System.Data.SqlDbType.Bit);
            projprobDelCmd.Parameters.Add("@k",                             Sys-
tem.Data.SqlDbType.TinyInt).Direction                   =                   Sys-
tem.Data.ParameterDirection.InputOutput;
            projprobDelCmd.Parameters.Add("@msg",                           Sys-
tem.Data.SqlDbType.VarChar,           100).Direction          =           Sys-
tem.Data.ParameterDirection.InputOutput;
        }

        private void init_staffDelCmd()
        {
            staffDelCmd = new System.Data.SqlClient.SqlCommand();
            staffDelCmd.Connection = this.viewStaffTableAdapter.Connection;
            staffDelCmd.CommandText = "DeleteEmp";
```

```
            staffDelCmd.CommandType                    =                    Sys-
tem.Data.CommandType.StoredProcedure;
            staffDelCmd.Parameters.Add("@Id_emp",                           Sys-
tem.Data.SqlDbType.SmallInt);
            staffDelCmd.Parameters.Add("@m", System.Data.SqlDbType.Bit);
            staffDelCmd.Parameters.Add("@k",                                Sys-
tem.Data.SqlDbType.TinyInt).Direction                 =                    Sys-
tem.Data.ParameterDirection.InputOutput;
            staffDelCmd.Parameters.Add("@msg",  System.Data.SqlDbType.VarChar,
100).Direction = System.Data.ParameterDirection.InputOutput;
        }

        private void init_departDelCmd()
        {
            departDelCmd = new System.Data.SqlClient.SqlCommand();
            departDelCmd.Connection = this.viewDepartTableAdapter.Connection;
            departDelCmd.CommandText = "DeleteDepartment";
            departDelCmd.CommandType                   =                    Sys-
tem.Data.CommandType.StoredProcedure;
            departDelCmd.Parameters.Add("@Id_depart",                       Sys-
tem.Data.SqlDbType.SmallInt);
            departDelCmd.Parameters.Add("@m", System.Data.SqlDbType.Bit);
            departDelCmd.Parameters.Add("@k",                               Sys-
tem.Data.SqlDbType.TinyInt).Direction                 =                    Sys-
tem.Data.ParameterDirection.InputOutput;
            departDelCmd.Parameters.Add("@msg",                             Sys-
tem.Data.SqlDbType.VarChar,           100).Direction          =           Sys-
tem.Data.ParameterDirection.InputOutput;
        }

        private void init_postDelCmd()
        {
            postDelCmd = new System.Data.SqlClient.SqlCommand();
            postDelCmd.Connection = this.postViewTableAdapter.Connection;
            postDelCmd.CommandText = "DeletePost";
            postDelCmd.CommandType = System.Data.CommandType.StoredProcedure;
            postDelCmd.Parameters.Add("@Id_post",                           Sys-
tem.Data.SqlDbType.SmallInt);
            postDelCmd.Parameters.Add("@m", System.Data.SqlDbType.Bit);
            postDelCmd.Parameters.Add("@k",                                 Sys-
tem.Data.SqlDbType.TinyInt).Direction                 =                    Sys-
tem.Data.ParameterDirection.InputOutput;
            postDelCmd.Parameters.Add("@msg",   System.Data.SqlDbType.VarChar,
100).Direction = System.Data.ParameterDirection.InputOutput;
        }

        private void init_projectDelCmd()
        {
            projectDelCmd = new System.Data.SqlClient.SqlCommand();
            projectDelCmd.Connection                                          =
this.viewProjectsTableAdapter.Connection;
            projectDelCmd.CommandText = "DeleteProject";
            projectDelCmd.CommandType                  =                    Sys-
tem.Data.CommandType.StoredProcedure;
            projectDelCmd.Parameters.Add("@Id_project",                     Sys-
tem.Data.SqlDbType.SmallInt);
            projectDelCmd.Parameters.Add("@m", System.Data.SqlDbType.Bit);
            projectDelCmd.Parameters.Add("@k",                              Sys-
tem.Data.SqlDbType.TinyInt).Direction                 =                    Sys-
tem.Data.ParameterDirection.InputOutput;
            projectDelCmd.Parameters.Add("@msg",                            Sys-
tem.Data.SqlDbType.VarChar,           100).Direction          =           Sys-
tem.Data.ParameterDirection.InputOutput;
        }

        private void init_departprojDelCmd()
        {
            departprojDelCmd = new System.Data.SqlClient.SqlCommand();
            departprojDelCmd.Connection                                       =
this.viewDepartProjectTableAdapter.Connection;
            departprojDelCmd.CommandText = "DeleteDepartProject";
```

```
            departprojDelCmd.CommandType                  =                  Sys-
tem.Data.CommandType.StoredProcedure;
            departprojDelCmd.Parameters.Add("@Id",                           Sys-
tem.Data.SqlDbType.SmallInt);
            departprojDelCmd.Parameters.Add("@m", System.Data.SqlDbType.Bit);
            departprojDelCmd.Parameters.Add("@k",                            Sys-
tem.Data.SqlDbType.TinyInt).Direction                  =                  Sys-
tem.Data.ParameterDirection.InputOutput;
            departprojDelCmd.Parameters.Add("@msg",                          Sys-
tem.Data.SqlDbType.VarChar,          100).Direction          =          Sys-
tem.Data.ParameterDirection.InputOutput;
        }

        private void init_staffprojDelCmd()
        {
            staffprojDelCmd = new System.Data.SqlClient.SqlCommand();
            staffprojDelCmd.Connection                                       =
this.viewStaffProjectTableAdapter.Connection;
            staffprojDelCmd.CommandText = "DeleteStaffProject";
            staffprojDelCmd.CommandType                  =                   Sys-
tem.Data.CommandType.StoredProcedure;
            staffprojDelCmd.Parameters.Add("@Id",                            Sys-
tem.Data.SqlDbType.SmallInt);
            staffprojDelCmd.Parameters.Add("@m", System.Data.SqlDbType.Bit);
            staffprojDelCmd.Parameters.Add("@k",                             Sys-
tem.Data.SqlDbType.TinyInt).Direction                  =                  Sys-
tem.Data.ParameterDirection.InputOutput;
            staffprojDelCmd.Parameters.Add("@msg",                           Sys-
tem.Data.SqlDbType.VarChar,          100).Direction          =          Sys-
tem.Data.ParameterDirection.InputOutput;
        }


        private void MainForm_Load(object sender, EventArgs e)
{
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet4.ViewStaffProject". При необходимости она может быть перемещена
или удалена.

this.viewStaffProjectTableAdapter3.Fill(this.cRMDBDataSet4.ViewStaffProject);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet4.ViewDepartProject". При необходимости она может быть перемеще-
на или удалена.

this.viewDepartProjectTableAdapter3.Fill(this.cRMDBDataSet4.ViewDepartProject
);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.ViewPerformers". При необходимости она может быть перемещена
или удалена.

this.viewPerformersTableAdapter1.Fill(this.cRMDBDataSet2.ViewPerformers);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.ViewProblems". При необходимости она может быть перемещена или
удалена.

this.viewProblemsTableAdapter1.Fill(this.cRMDBDataSet2.ViewProblems);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Departments". При необходимости она может быть перемещена или
удалена.

this.departmentsTableAdapter.Fill(this.cRMDBDataSet2.Departments);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.ViewStaffProject". При необходимости она может быть перемещена
или удалена.

this.viewStaffProjectTableAdapter1.Fill(this.cRMDBDataSet2.ViewStaffProject);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.ViewDepartProject". При необходимости она может быть перемеще-
на или удалена.
```

```
this.viewDepartProjectTableAdapter1.Fill(this.cRMDBDataSet2.ViewDepartProject
);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.ViewProjects". При необходимости она может быть перемещена или
удалена.

this.viewProjectsTableAdapter1.Fill(this.cRMDBDataSet2.ViewProjects);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.PostView". При необходимости она может быть перемещена или
удалена.
            this.postViewTableAdapter1.Fill(this.cRMDBDataSet2.PostView);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.ViewDepart". При необходимости она может быть перемещена или
удалена.
            this.viewDepartTableAdapter1.Fill(this.cRMDBDataSet2.ViewDepart);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.ViewStaff". При необходимости она может быть перемещена или
удалена.
            this.viewStaffTableAdapter1.Fill(this.cRMDBDataSet2.ViewStaff);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.ViewProjectProblem". При необходимости она может быть переме-
щена или удалена.

this.viewProjectProblemTableAdapter1.Fill(this.cRMDBDataSet2.ViewProjectProbl
em);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet1.ViewPerformers". При необходимости она может быть перемещена
или удалена.

this.viewPerformersTableAdapter.Fill(this.cRMDBDataSet1.ViewPerformers);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet1.ViewPerformers". При необходимости она может быть перемещена
или удалена.

this.viewPerformersTableAdapter.Fill(this.cRMDBDataSet1.ViewPerformers);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet.ViewStaffProject". При необходимости она может быть перемещена
или удалена.

this.viewStaffProjectTableAdapter.Fill(this.cRMDBDataSet.ViewStaffProject);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet.ViewDepartProject". При необходимости она может быть перемещена
или удалена.

this.viewDepartProjectTableAdapter.Fill(this.cRMDBDataSet.ViewDepartProject);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet.ViewProjects". При необходимости она может быть перемещена или
удалена.

this.viewProjectsTableAdapter.Fill(this.cRMDBDataSet.ViewProjects);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet.PostView". При необходимости она может быть перемещена или уда-
лена.
            this.postViewTableAdapter.Fill(this.cRMDBDataSet.PostView);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet.ViewDepart". При необходимости она может быть перемещена или
удалена.
            this.viewDepartTableAdapter.Fill(this.cRMDBDataSet.ViewDepart);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet.ViewStaff". При необходимости она может быть перемещена или
удалена.
            this.viewStaffTableAdapter.Fill(this.cRMDBDataSet.ViewStaff);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet.Staff". При необходимости она может быть перемещена или удале-
на.
            this.staffTableAdapter.Fill(this.cRMDBDataSet.Staff);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet.ViewProjectProblem". При необходимости она может быть перемеще-
на или удалена.
```

```
this.viewProjectProblemTableAdapter.Fill(this.cRMDBDataSet.ViewProjectProblem
);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet.ViewPerformers". При необходимости она может быть перемещена
или удалена.

this.viewPerformersTableAdapter.Fill(this.cRMDBDataSet1.ViewPerformers);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet.ViewProblems". При необходимости она может быть перемещена или
удалена.

this.viewProblemsTableAdapter.Fill(this.cRMDBDataSet.ViewProblems);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet.Problems". При необходимости она может быть перемещена или уда-
лена.
            this.problemsTableAdapter.Fill(this.cRMDBDataSet.Problems);
}


        private void DeleteProblemButton_Click(object sender, EventArgs e)
        {
            if       (MessageBox.Show(("Хотитеудалитьзапрос:        "        +
(string)dataGridView1.CurrentRow.Cells[1].Value) + "?",
                "Удалениезапроса",    MessageBoxButtons.OKCancel)    ==    Sys-
tem.Windows.Forms.DialogResult.OK)
            {
                problemDelCmd.Parameters["@Id_problem"].Value                =
(int)(dataGridView2.CurrentRow.Cells[4].Value);
                problemDelCmd.Parameters["@m"].Value = 1;
                problemDelCmd.Parameters["@k"].Value = System.DBNull.Value;
                problemDelCmd.Parameters["@msg"].Value = System.DBNull.Value;
                System.Data.ConnectionState  previousConnectionState  =  prob-
lemDelCmd.Connection.State;
                if       (((problemDelCmd.Connection.State       &       Sys-
tem.Data.ConnectionState.Open) != System.Data.ConnectionState.Open))
                {
                    problemDelCmd.Connection.Open();
                }
                try
                {
                    problemDelCmd.ExecuteNonQuery();
                    if   (Convert.ToByte(problemDelCmd.Parameters["@k"].Value)
> 0)
{
                        if
(MessageBox.Show((string)(problemDelCmd.Parameters["@msg"].Value)   +   "\nВсе
равно удалить?",
                            "Удаление       из       связаных       таблиц",
MessageBoxButtons.OKCancel) == System.Windows.Forms.DialogResult.OK)
{
                            problemDelCmd.Parameters["@m"].Value = 0;
                            problemDelCmd.ExecuteNonQuery();
                        }
                    }
                }
                finally
                {
                    if        (previousConnectionState        ==        Sys-
tem.Data.ConnectionState.Closed)
                        problemDelCmd.Connection.Close();
                }

this.viewProblemsTableAdapter.Fill(cRMDBDataSet.ViewProblems);
            }
        }

        private void DeletePerformerButton_Click(object sender, EventArgs e)
        {
            if          (MessageBox.Show(("Хотитеудалить          "          +
(string)dataGridView2.CurrentRow.Cells[1].Value) + "?",
```

```
                "Удалениеобласти",    MessageBoxButtons.OKCancel)    ==    Sys-
tem.Windows.Forms.DialogResult.OK)
            {
                performerDelCmd.Parameters["@Id_performer"].Value            =
(int)(dataGridView2.CurrentRow.Cells[4].Value);
                performerDelCmd.Parameters["@m"].Value = 1;
                performerDelCmd.Parameters["@k"].Value = System.DBNull.Value;
                performerDelCmd.Parameters["@msg"].Value        =        Sys-
tem.DBNull.Value;
                System.Data.ConnectionState  previousConnectionState  =  per-
formerDelCmd.Connection.State;
                if      (((performerDelCmd.Connection.State      &      Sys-
tem.Data.ConnectionState.Open) != System.Data.ConnectionState.Open))
                {
                    performerDelCmd.Connection.Open();
                }
                try
                {
                    performerDelCmd.ExecuteNonQuery();
                    if                                                  (Con-
vert.ToByte(performerDelCmd.Parameters["@k"].Value) > 0)
{
                        if
(MessageBox.Show((string)(performerDelCmd.Parameters["@msg"].Value)  +  "\nВсе
равно удалить?",
                            "Удаление       из       связаных       таблиц",
MessageBoxButtons.OKCancel) == System.Windows.Forms.DialogResult.OK)
{
                            performerDelCmd.Parameters["@m"].Value = 0;
                            performerDelCmd.ExecuteNonQuery();
                        }
                    }
                }
                finally
                {
                    if        (previousConnectionState        ==        Sys-
tem.Data.ConnectionState.Closed)
                        performerDelCmd.Connection.Close();
                }

this.viewPerformersTableAdapter.Fill(cRMDBDataSet1.ViewPerformers);

            }
        }

        private  void  DeleteProjProblemButton_Click(object  sender,  EventArgs
e)
        {
            if          (MessageBox.Show(("Хотитеудалить          "          +
(string)dataGridView3.CurrentRow.Cells[1].Value) + "?",
                "Удалениеобласти",    MessageBoxButtons.OKCancel)    ==    Sys-
tem.Windows.Forms.DialogResult.OK)
            {
                projprobDelCmd.Parameters["@Id"].Value                       =
(short)(dataGridView3.CurrentRow.Cells[0].Value);
                projprobDelCmd.Parameters["@m"].Value = 1;
                projprobDelCmd.Parameters["@k"].Value = System.DBNull.Value;
                projprobDelCmd.Parameters["@msg"].Value         =         Sys-
tem.DBNull.Value;
                System.Data.ConnectionState  previousConnectionState  =  pro-
jprobDelCmd.Connection.State;
                if       (((projprobDelCmd.Connection.State       &       Sys-
tem.Data.ConnectionState.Open) != System.Data.ConnectionState.Open))
                {
                    projprobDelCmd.Connection.Open();
                }
                try
                {
                    projprobDelCmd.ExecuteNonQuery();
                    if ((byte)projprobDelCmd.Parameters["@k"].Value > 0)
{
```

```
                        if
(MessageBox.Show((string)(projprobDelCmd.Parameters["@msg"].Value)  +  "\nВсе
равно удалить?",
                            "Удаление       из       связаных       таблиц",
MessageBoxButtons.OKCancel) == System.Windows.Forms.DialogResult.OK)
{
                            projprobDelCmd.Parameters["@m"].Value = 0;
                            projprobDelCmd.ExecuteNonQuery();
                        }
                    }
                }
                finally
                {
                    if        (previousConnectionState        ==        Sys-
tem.Data.ConnectionState.Closed)
                        projprobDelCmd.Connection.Close();
                }

this.viewProjectProblemTableAdapter.Fill(cRMDBDataSet.ViewProjectProblem);
            }
        }

        private void DeleteEmpButton_Click(object sender, EventArgs e)
        {
            if       (MessageBox.Show(("Хотитеудалитьданныеоб:       "       +
(string)dataGridView4.CurrentRow.Cells[0].Value) + "?",
                "Удалениезаписиобсотруднике",  MessageBoxButtons.OKCancel)  ==
System.Windows.Forms.DialogResult.OK)
            {
                staffDelCmd.Parameters["@Id_emp"].Value                      =
(short)(dataGridView4.CurrentRow.Cells[5].Value);
                staffDelCmd.Parameters["@m"].Value = 1;
                staffDelCmd.Parameters["@k"].Value = System.DBNull.Value;
                staffDelCmd.Parameters["@msg"].Value = System.DBNull.Value;
                System.Data.ConnectionState  previousConnectionState  =  staff-
DelCmd.Connection.State;
                if        (((staffDelCmd.Connection.State        &        Sys-
tem.Data.ConnectionState.Open) != System.Data.ConnectionState.Open))
                {
                    staffDelCmd.Connection.Open();
                }
                try
                {
                    staffDelCmd.ExecuteNonQuery();
                    if ((byte)staffDelCmd.Parameters["@k"].Value > 0)
{
                        if
(MessageBox.Show((string)(staffDelCmd.Parameters["@msg"].Value) + "\nВсе рав-
но удалить?",
                            "Удаление       из       связаных       таблиц",
MessageBoxButtons.OKCancel) == System.Windows.Forms.DialogResult.OK)
{
                            staffDelCmd.Parameters["@m"].Value = 0;
                            staffDelCmd.ExecuteNonQuery();
                        }
                    }
                }
                finally
                {
                    if        (previousConnectionState        ==        Sys-
tem.Data.ConnectionState.Closed)
                        staffDelCmd.Connection.Close();
                }
                this.viewStaffTableAdapter.Fill(cRMDBDataSet.ViewStaff);
            }
        }

        private void DeleteDepartButton_Click(object sender, EventArgs e)
        {
            if       (MessageBox.Show(("Хотитеудалитьданныеоб:       "       +
(string)dataGridView5.CurrentRow.Cells[0].Value) + "?",
```

```
                "Удалениеданныхоботделе", MessageBoxButtons.OKCancel) == Sys-
tem.Windows.Forms.DialogResult.OK)
            {
                departDelCmd.Parameters["@Id_depart"].Value                    =
(short)(dataGridView5.CurrentRow.Cells[1].Value);
                departDelCmd.Parameters["@m"].Value = 1;
                departDelCmd.Parameters["@k"].Value = System.DBNull.Value;
                departDelCmd.Parameters["@msg"].Value = System.DBNull.Value;
                System.Data.ConnectionState  previousConnectionState  =  de-
partDelCmd.Connection.State;
                if       (((departDelCmd.Connection.State       &       Sys-
tem.Data.ConnectionState.Open) != System.Data.ConnectionState.Open))
                {
                    departDelCmd.Connection.Open();
                }
                try
                {
                    departDelCmd.ExecuteNonQuery();
                    if ((byte)departDelCmd.Parameters["@k"].Value > 0)
{
                        if
(MessageBox.Show((string)(departDelCmd.Parameters["@msg"].Value)   +   "\nВсе
равно удалить?",
                            "Удаление       из       связаных       таблиц",
MessageBoxButtons.OKCancel) == System.Windows.Forms.DialogResult.OK)
{
                            departDelCmd.Parameters["@m"].Value = 0;
                            departDelCmd.ExecuteNonQuery();
                        }
                    }
                }
                finally
                {
                    if        (previousConnectionState        ==        Sys-
tem.Data.ConnectionState.Closed)
                        departDelCmd.Connection.Close();
                }
                this.viewDepartTableAdapter.Fill(cRMDBDataSet.ViewDepart);
            }
        }

        private void DeletePostButton_Click(object sender, EventArgs e)
        {
            if       (MessageBox.Show(("Хотитеудалитьданныеоб:       "       +
(string)dataGridView6.CurrentRow.Cells[0].Value) + "?",
                "Удалениеданныхобдолжности",   MessageBoxButtons.OKCancel)   ==
System.Windows.Forms.DialogResult.OK)
            {
                postDelCmd.Parameters["@Id_post"].Value                        =
(short)(dataGridView6.CurrentRow.Cells[1].Value);
                postDelCmd.Parameters["@m"].Value = 1;
                postDelCmd.Parameters["@k"].Value = System.DBNull.Value;
                postDelCmd.Parameters["@msg"].Value = System.DBNull.Value;
                System.Data.ConnectionState  previousConnectionState  =  post-
DelCmd.Connection.State;
                if        (((postDelCmd.Connection.State        &        Sys-
tem.Data.ConnectionState.Open) != System.Data.ConnectionState.Open))
                {
                    postDelCmd.Connection.Open();
                }
                try
                {
                    postDelCmd.ExecuteNonQuery();
                    if ((byte)postDelCmd.Parameters["@k"].Value > 0)
{
                        if
(MessageBox.Show((string)(postDelCmd.Parameters["@msg"].Value) + "\nВсе равно
удалить?",
                            "Удаление       из       связаных       таблиц",
MessageBoxButtons.OKCancel) == System.Windows.Forms.DialogResult.OK)
{
```

```
                            postDelCmd.Parameters["@m"].Value = 0;
                            postDelCmd.ExecuteNonQuery();
                        }
                    }
                }
                finally
                {
                    if        (previousConnectionState        ==        Sys-
tem.Data.ConnectionState.Closed)
                        postDelCmd.Connection.Close();
                }
                this.postViewTableAdapter.Fill(cRMDBDataSet.PostView);
            }
        }

        private void DeleteProjectButton_Click(object sender, EventArgs e)
        {
            if       (MessageBox.Show(("Хотитеудалитьданныеоб:       "       +
(string)dataGridView7.CurrentRow.Cells[0].Value) + "?",
                "Удалениеданныхобпроекте", MessageBoxButtons.OKCancel) == Sys-
tem.Windows.Forms.DialogResult.OK)
            {
                projectDelCmd.Parameters["@Id_project"].Value                =
(short)(dataGridView7.CurrentRow.Cells[1].Value);
                projectDelCmd.Parameters["@m"].Value = 1;
                projectDelCmd.Parameters["@k"].Value = System.DBNull.Value;
                projectDelCmd.Parameters["@msg"].Value = System.DBNull.Value;
                System.Data.ConnectionState  previousConnectionState  =  pro-
jectDelCmd.Connection.State;
                if       (((projectDelCmd.Connection.State       &       Sys-
tem.Data.ConnectionState.Open) != System.Data.ConnectionState.Open))
                {
                    projectDelCmd.Connection.Open();
                }
                try
                {
                    projectDelCmd.ExecuteNonQuery();
                    if ((byte)projectDelCmd.Parameters["@k"].Value > 0)
{
                        if
(MessageBox.Show((string)(projectDelCmd.Parameters["@msg"].Value)   +   "\nВсе
равно удалить?",
                            "Удаление       из       связаных       таблиц",
MessageBoxButtons.OKCancel) == System.Windows.Forms.DialogResult.OK)
{
                            projectDelCmd.Parameters["@m"].Value = 0;
                            projectDelCmd.ExecuteNonQuery();
                        }
                    }
                }
                finally
                {
                    if        (previousConnectionState        ==        Sys-
tem.Data.ConnectionState.Closed)
                        projectDelCmd.Connection.Close();
                }

this.viewProjectsTableAdapter.Fill(cRMDBDataSet.ViewProjects);
            }
        }

        private void DeleteDepartProjButton_Click(object sender, EventArgs e)
        {
            if       (MessageBox.Show(("Хотитеудалитьданныеоб       "       +
(string)dataGridView8.CurrentRow.Cells[0].Value) + " отделе?",
                "Удалениеданныхоботделе",  MessageBoxButtons.OKCancel)  ==  Sys-
tem.Windows.Forms.DialogResult.OK)
            {
                departprojDelCmd.Parameters["@Id"].Value                     =
(short)(dataGridView8.CurrentRow.Cells[3].Value);
                departprojDelCmd.Parameters["@m"].Value = 1;
```

```
                departprojDelCmd.Parameters["@k"].Value        =        Sys-
tem.DBNull.Value;
                departprojDelCmd.Parameters["@msg"].Value       =        Sys-
tem.DBNull.Value;
                System.Data.ConnectionState  previousConnectionState  =  de-
partprojDelCmd.Connection.State;
                if      (((departprojDelCmd.Connection.State      &      Sys-
tem.Data.ConnectionState.Open) != System.Data.ConnectionState.Open))
                {
                    departprojDelCmd.Connection.Open();
                }
                try
                {
                    departprojDelCmd.ExecuteNonQuery();
                    if ((byte)departprojDelCmd.Parameters["@k"].Value > 0)
{
                        if
(MessageBox.Show((string)(departprojDelCmd.Parameters["@msg"].Value) + "\nВсе
равно удалить?",
                            "Удаление       из       связаных       таблиц",
MessageBoxButtons.OKCancel) == System.Windows.Forms.DialogResult.OK)
{
                            departprojDelCmd.Parameters["@m"].Value = 0;
                            departprojDelCmd.ExecuteNonQuery();
                        }
                    }
                }
                finally
                {
                    if        (previousConnectionState        ==        Sys-
tem.Data.ConnectionState.Closed)
                        departprojDelCmd.Connection.Close();
                }

this.viewDepartProjectTableAdapter.Fill(cRMDBDataSet.ViewDepartProject);
            }
        }

        private void DeleteStaffProjButton_Click(object sender, EventArgs e)
        {
            if       (MessageBox.Show(("Хотитеудалитьданныеоб       "       +
(string)dataGridView9.CurrentRow.Cells[0].Value) + " сотрудника?",
                "Удалениеданныхоботделе", MessageBoxButtons.OKCancel)  ==  Sys-
tem.Windows.Forms.DialogResult.OK)
            {
                staffprojDelCmd.Parameters["@Id"].Value                     =
(short)(dataGridView9.CurrentRow.Cells[3].Value);
                staffprojDelCmd.Parameters["@m"].Value = 1;
                staffprojDelCmd.Parameters["@k"].Value = System.DBNull.Value;
                staffprojDelCmd.Parameters["@msg"].Value        =        Sys-
tem.DBNull.Value;
                System.Data.ConnectionState  previousConnectionState  =  staff-
projDelCmd.Connection.State;
                if      (((staffprojDelCmd.Connection.State      &      Sys-
tem.Data.ConnectionState.Open) != System.Data.ConnectionState.Open))
                {
                    staffprojDelCmd.Connection.Open();
                }
                try
                {
                    staffprojDelCmd.ExecuteNonQuery();
                    if ((byte)staffprojDelCmd.Parameters["@k"].Value > 0)
{
                        if
(MessageBox.Show((string)(staffprojDelCmd.Parameters["@msg"].Value)  +  "\nВсе
равно удалить?",
                            "Удаление       из       связаных       таблиц",
MessageBoxButtons.OKCancel) == System.Windows.Forms.DialogResult.OK)
{
                            staffprojDelCmd.Parameters["@m"].Value = 0;
                            staffprojDelCmd.ExecuteNonQuery();
```

```
                    }
                }
            }
            finally
            {
                if (previousConnectionState == System.Data.ConnectionState.Closed)
                    staffprojDelCmd.Connection.Close();
            }

this.viewStaffProjectTableAdapter.Fill(cRMDBDataSet.ViewStaffProject);
        }
    }

    private void AddEmpButton_Click(object sender, EventArgs e)
    {
        AddEmpForm addEmp = new AddEmpForm();
        addEmp.Show();
    }

    private void AddProblemButton_Click(object sender, EventArgs e)
    {
        AddProblemForm addProblem = new AddProblemForm();
        addProblem.Show();
    }

    private void AddPerformerButton_Click(object sender, EventArgs e)
    {
        addPerformer addPerformer = new addPerformer();
        addPerformer.Show();
    }

    private void AddProjProblemButton_Click(object sender, EventArgs e)
    {
        addProjProb addProjProb = new addProjProb();
        addProjProb.Show();
    }

    private void AddDepartButton_Click(object sender, EventArgs e)
    {
        addDepart addDepart = new addDepart();
        addDepart.Show();
    }

    private void AddPostButton_Click(object sender, EventArgs e)
    {
        addPost addPost = new addPost();
        addPost.Show();
    }

    private void AddProjectButton_Click(object sender, EventArgs e)
    {
        addProject addProject = new addProject();
        addProject.Show();
    }

    private void AddDepartProjButton_Click(object sender, EventArgs e)
    {
        addDepartProj addDepartProj = new addDepartProj();
        addDepartProj.Show();
    }

    private void AddStaffProjButton_Click(object sender, EventArgs e)
    {
        addStaffProj addStaffProj = new addStaffProj();
        addStaffProj.Show();
    }

    private void EditProblemButton_Click(object sender, EventArgs e)
    {
        editProblem editProblem = new editProblem();
```

```
            editProblem.Show();
        }

        private void EditPerformerButton_Click(object sender, EventArgs e)
        {
            editPerformer editPerformer = new editPerformer();
            editPerformer.Show();
        }

        private void EditProjProblemButton_Click(object sender, EventArgs e)
        {
            editProjProb editProjProb = new editProjProb();
            editProjProb.Show();
        }

        private void EditEmpButton_Click(object sender, EventArgs e)
        {
            editEmp editEmp = new editEmp();
            editEmp.Show();
        }

        private void EditDepartButton_Click(object sender, EventArgs e)
        {
            editDepart editDepart = new editDepart();
            editDepart.Show();
        }

        private void EditPostButton_Click(object sender, EventArgs e)
        {
            editPost editPost = new editPost();
            editPost.Show();
        }

        private void EditProjectButton_Click(object sender, EventArgs e)
        {
            editProject editProject = new editProject();
            editProject.Show();
        }

        private void EditDepartProjButton_Click(object sender, EventArgs e)
        {
            editDepartProj editDepartProj = new editDepartProj();
            editDepartProj.Show();
        }

        private void EditStaffProjButton_Click(object sender, EventArgs e)
        {
            editStaffProject editStaffProject = new editStaffProject();
            editStaffProject.Show();
}
    }
}
```

Г.3 Форма додавання до таблиці Departments

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace CRMka
{
    public partial class addDepart : Form
```

```
    {
        public void SetConnection()
        {
            db_con = new SqlConnection(@"Data Source=DESKTOP-
8MQT1B7\SQLEXPRESS;Initial Catalog=CRMDB;Integrated Security=True;");
        }

        private void ExecuteQuery(string txtQuery)
        {
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            db_cmd.CommandText = txtQuery;
            db_cmd.ExecuteNonQuery();
            db_con.Close();
        }


        private SqlConnection db_con;
        private SqlCommand db_cmd;
        private DataSet DS = new DataSet();
        private DataTable DT = new DataTable();


        public addDepart()
        {
            InitializeComponent();
        }

        private void AddButton_Click(object sender, EventArgs e)
        {
            MainForm mf = new MainForm();
            string add = "insert into Departments(Name_depart) values ('" +
textBox1.Text + "');";
            ExecuteQuery(add);
            Close();
            mf.Refresh();
        }

        private void CancelButton_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```

Г.4 Форма додавання до таблиці DepartProject

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace CRMka
{
    public partial class addDepartProj : Form
    {
        public void SetConnection()
        {
            db_con = new SqlConnection(@"Data Source=DESKTOP-
8MQT1B7\SQLEXPRESS;Initial Catalog=CRMDB;Integrated Security=True;");
        }
```

```
        private void ExecuteQuery(string txtQuery)
        {
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            db_cmd.CommandText = txtQuery;
            db_cmd.ExecuteNonQuery();
            db_con.Close();
        }


        private SqlConnection db_con;
        private SqlCommand db_cmd;
        private DataSet DS = new DataSet();
        private DataTable DT = new DataTable();


        public addDepartProj()
        {
            InitializeComponent();
        }

        private void AddDepartProj_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet4.Status". При необходимости она может быть перемещена или уда-
лена.
            this.statusTableAdapter.Fill(this.cRMDBDataSet4.Status);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Projects". При необходимости она может быть перемещена или
удалена.
            this.projectsTableAdapter.Fill(this.cRMDBDataSet2.Projects);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Departments". При необходимости она может быть перемещена или
удалена.

this.departmentsTableAdapter.Fill(this.cRMDBDataSet2.Departments);

        }

        private void AddButton_Click(object sender, EventArgs e)
        {
            MainForm mf = new MainForm();
            string add = "insert into DepartProject(Id_depart, Id_project,
Id_status) values ('" + comboBox1.Text + "','" + comboBox2.Text + "','" +
comboBox3.Text + "');";
            ExecuteQuery(add);
            Close();
            mf.Refresh();
        }

        private void CancelButton_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```

Г.5 Форма додавання до таблиці Staff

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
using System.Data.SqlClient;

namespace CRMka
{
    public partial class AddEmpForm : Form
    {
        public void SetConnection()
        {
            db_con = new SqlConnection(@"Data Source=DESKTOP-
8MQT1B7\SQLEXPRESS;Initial Catalog=CRMDB;Integrated Security=True;");
        }

        private void ExecuteQuery(string txtQuery)
        {
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            db_cmd.CommandText = txtQuery;
            db_cmd.ExecuteNonQuery();
            db_con.Close();
        }


        private SqlConnection db_con;
        private SqlCommand db_cmd;
        private DataSet DS = new DataSet();
        private DataTable DT = new DataTable();

        public AddEmpForm()
        {
            InitializeComponent();
        }

        private void AddButton_Click(object sender, EventArgs e)
        {
            MainForm mf = new MainForm();
            string add = "insert into
Staff(First_name,Last_name,Middle_name,Login_emp,Emp_password,Emp_phone,
Emp_Number, Id_depart, Id_post) values ('" + textBox1.Text + "','" +
textBox2.Text + "', '" + textBox3.Text + "', '" + textBox4.Text + "', '" +
textBox5.Text + "', '" + textBox6.Text + "', '" + textBox7.Text + "', '" +
comboBox1.Text + "', '" + comboBox2.Text + "');";
            ExecuteQuery(add);
            Close();
            mf.Refresh();
        }

        private void CancelButton_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void AddEmpForm_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.PostView". При необходимости она может быть перемещена или
удалена.
            this.postViewTableAdapter.Fill(this.cRMDBDataSet2.PostView);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.ViewDepart". При необходимости она может быть перемещена или
удалена.
            this.viewDepartTableAdapter.Fill(this.cRMDBDataSet2.ViewDepart);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Post". При необходимости она может быть перемещена или удале-
на.
            this.postTableAdapter.Fill(this.cRMDBDataSet2.Post);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Departments". При необходимости она может быть перемещена или
удалена.

this.departmentsTableAdapter.Fill(this.cRMDBDataSet2.Departments);
```

```
        }
    }
}
```

Г.6 Форма додавання до таблиці Performers

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace CRMka
{
    public partial class addPerformer : Form
    {

        public void SetConnection()
        {
            db_con = new SqlConnection(@"Data Source=DESKTOP-
8MQT1B7\SQLEXPRESS;Initial Catalog=CRMDB;Integrated Security=True;");
        }

        private void ExecuteQuery(string txtQuery)
        {
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            db_cmd.CommandText = txtQuery;
            db_cmd.ExecuteNonQuery();
            db_con.Close();
        }


        private SqlConnection db_con;
        private SqlCommand db_cmd;
        private DataSet DS = new DataSet();
        private DataTable DT = new DataTable();

        public addPerformer()
        {
            InitializeComponent();
        }

        private void AddPerformerButton_Click(object sender, EventArgs e)
        {
            string add = "insert into
Performers(Id_emp,Id_problem,About_duration,Fact_duration) values ('" +
comboBox1.Text + "','" + comboBox2.Text + "', '" + textBox2.Text + "', '" +
textBox3.Text +  "');";
            ExecuteQuery(add);
            Close();
        }

        private void CancelButton_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void AddPerformer_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet4.ViewStaff". При необходимости она может быть перемещена или
удалена.
```

```
            this.viewStaffTableAdapter.Fill(this.cRMDBDataSet4.ViewStaff);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Problems". При необходимости она может быть перемещена или
удалена.
            this.problemsTableAdapter.Fill(this.cRMDBDataSet2.Problems);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Staff". При необходимости она может быть перемещена или удале-
на.
            this.staffTableAdapter.Fill(this.cRMDBDataSet2.Staff);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.ViewProblems". При необходимости она может быть перемещена или
удалена.

this.viewProblemsTableAdapter.Fill(this.cRMDBDataSet2.ViewProblems);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.ViewPerformers". При необходимости она может быть перемещена
или удалена.

this.viewPerformersTableAdapter.Fill(this.cRMDBDataSet2.ViewPerformers);

        }
    }
}
```

Г.7 Форма додавання до таблиці Post

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace CRMka
{
    public partial class addPost : Form
    {

        public void SetConnection()
        {
            db_con = new SqlConnection(@"Data Source=DESKTOP-
8MQT1B7\SQLEXPRESS;Initial Catalog=CRMDB;Integrated Security=True;");
        }

        private void ExecuteQuery(string txtQuery)
        {
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            db_cmd.CommandText = txtQuery;
            db_cmd.ExecuteNonQuery();
            db_con.Close();
        }


        private SqlConnection db_con;
        private SqlCommand db_cmd;
        private DataSet DS = new DataSet();
        private DataTable DT = new DataTable();



        public addPost()
        {
            InitializeComponent();
        }
```

```
        private void AddPost_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Departments". При необходимости она может быть перемещена или
удалена.

this.departmentsTableAdapter.Fill(this.cRMDBDataSet2.Departments);

        }

        private void AddButton_Click(object sender, EventArgs e)
        {
            MainForm mf = new MainForm();
            string add = "insert into Post(Name_post, Id_depart) values ('" +
textBox1.Text + "','" + comboBox1.Text + "');";
            ExecuteQuery(add);
            Close();
            mf.Refresh();
        }

        private void CancelButton_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```

Г.8 Форма додавання до таблиці Problems

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace CRMka
{
    public partial class AddProblemForm : Form
    {
        public void SetConnection()
        {
            db_con = new SqlConnection(@"Data Source=DESKTOP-
8MQT1B7\SQLEXPRESS;Initial Catalog=CRMDB;Integrated Security=True;");
        }

        private void ExecuteQuery(string txtQuery)
        {
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            db_cmd.CommandText = txtQuery;
            db_cmd.ExecuteNonQuery();
            db_con.Close();
        }


        private SqlConnection db_con;
        private SqlCommand db_cmd;
        private DataSet DS = new DataSet();
        private DataTable DT = new DataTable();


        public AddProblemForm()
```

```
        {
            InitializeComponent();
        }

        private void AddProblemForm_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Priorities". При необходимости она может быть перемещена или
удалена.
            this.prioritiesTableAdapter.Fill(this.cRMDBDataSet2.Priorities);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.PostView". При необходимости она может быть перемещена или
удалена.
            this.postViewTableAdapter.Fill(this.cRMDBDataSet2.PostView);

        }

        private void addButton_Click(object sender, EventArgs e)
        {
            MainForm mf = new MainForm();
            string add = "insert into
Problems(Problem_description,Problem_solution,Problem_registration,Deadline,A
bout_duration,Fact_duration, Id_priority) values ('" + textBox1.Text + "','"
+ textBox2.Text + "', '" + Convert.ToString(dateTimePicker1.Value) + "', '" +
Convert.ToString(dateTimePicker2.Value) + "', '" +textBox5.Text + "','" +
textBox6.Text + "','" + comboBox1.Text + "');";
            ExecuteQuery(add);
            Close();
            mf.Refresh();

        }

        private void cancelButton_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```

Г.9 Форма додавання до таблиці Projects

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace CRMka
{
    public partial class addProject : Form
    {

        public void SetConnection()
        {
            db_con = new SqlConnection(@"Data Source=DESKTOP-
8MQT1B7\SQLEXPRESS;Initial Catalog=CRMDB;Integrated Security=True;");
        }

        private void ExecuteQuery(string txtQuery)
        {
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            db_cmd.CommandText = txtQuery;
```

```
            db_cmd.ExecuteNonQuery();
            db_con.Close();
        }


        private SqlConnection db_con;
        private SqlCommand db_cmd;
        private DataSet DS = new DataSet();
        private DataTable DT = new DataTable();


        public addProject()
        {
            InitializeComponent();
        }

        private void addButton_Click(object sender, EventArgs e)
        {
            MainForm mf = new MainForm();
            string add = "insert into Projects(Name_project) values ('" +
textBox1.Text + "');";
            ExecuteQuery(add);
            Close();
            mf.Refresh();
        }

        private void cancelButton_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```

Г.10 Форма додавання до таблиці ProjectProblem

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace CRMka
{
    public partial class addProjProb : Form
    {

        public void SetConnection()
        {
            db_con = new SqlConnection(@"Data Source=DESKTOP-
8MQT1B7\SQLEXPRESS;Initial Catalog=CRMDB;Integrated Security=True;");
        }

        private void ExecuteQuery(string txtQuery)
        {
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            db_cmd.CommandText = txtQuery;
            db_cmd.ExecuteNonQuery();
            db_con.Close();
        }


        private SqlConnection db_con;
```

```
        private SqlCommand db_cmd;
        private DataSet DS = new DataSet();
        private DataTable DT = new DataTable();


        public addProjProb()
        {
            InitializeComponent();
        }

        private void AddProjProb_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Projects". При необходимости она может быть перемещена или
удалена.
            this.projectsTableAdapter.Fill(this.cRMDBDataSet2.Projects);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Problems". При необходимости она может быть перемещена или
удалена.
            this.problemsTableAdapter.Fill(this.cRMDBDataSet2.Problems);

        }

        private void AddButton_Click(object sender, EventArgs e)
        {
            MainForm mf = new MainForm();
            string add = "insert into ProjectProblems(Id_problem, Id_project)
values ('" + comboBox1.Text + "','" + comboBox2.Text + "');";
            ExecuteQuery(add);
            Close();
            mf.Refresh();
        }

        private void CancelButton_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```

Г.11 Форма додавання до таблиці StaffProject

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace CRMka
{
    public partial class addStaffProj : Form
    {
        public void SetConnection()
        {
            db_con = new SqlConnection(@"Data Source=DESKTOP-
8MQT1B7\SQLEXPRESS;Initial Catalog=CRMDB;Integrated Security=True;");
        }

        private void ExecuteQuery(string txtQuery)
        {
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            db_cmd.CommandText = txtQuery;
```

```
            db_cmd.ExecuteNonQuery();
            db_con.Close();
        }


        private SqlConnection db_con;
        private SqlCommand db_cmd;
        private DataSet DS = new DataSet();
        private DataTable DT = new DataTable();



        public addStaffProj()
        {
            InitializeComponent();
        }

        private void AddButton_Click(object sender, EventArgs e)
        {
            MainForm mf = new MainForm();
            string add = "insert into StaffProject(Id_emp, Id_project) values
('" + comboBox1.Text + "','" + comboBox2.Text + "');";
            ExecuteQuery(add);
            Close();
            mf.Refresh();
        }

        private void CancelButton_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void AddStaffProj_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet4.ViewStaff". При необходимости она может быть перемещена или
удалена.
            this.viewStaffTableAdapter.Fill(this.cRMDBDataSet4.ViewStaff);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Projects". При необходимости она может быть перемещена или
удалена.
            this.projectsTableAdapter.Fill(this.cRMDBDataSet2.Projects);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Staff". При необходимости она может быть перемещена или удале-
на.
            this.staffTableAdapter.Fill(this.cRMDBDataSet2.Staff);

        }
    }
}
```

Г.12 Форма оновлення таблиці Departments

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace CRMka
{
    public partial class editDepart : Form
    {
        public void SetConnection()
```

```
        {
            db_con = new SqlConnection(@"Data Source=DESKTOP-
8MQT1B7\SQLEXPRESS;Initial Catalog=CRMDB;Integrated Security=True;");
        }

        private void ExecuteQuery(string txtQuery)
        {
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            db_cmd.CommandText = txtQuery;
            db_cmd.ExecuteNonQuery();
            db_con.Close();
        }


        private SqlConnection db_con;
        private SqlCommand db_cmd;
        private DataSet DS = new DataSet();
        private DataTable DT = new DataTable();

        public editDepart()
        {
            InitializeComponent();
        }

        private void EditDepart_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.ViewDepart". При необходимости она может быть перемещена или
удалена.
            this.viewDepartTableAdapter.Fill(this.cRMDBDataSet2.ViewDepart);

        }

        private void CancelButton_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void EditButton_Click(object sender, EventArgs e)
        {
            string txtQuery = "UPDATE Departments SET Name_depart = '" +
textBox1.Text +  "' WHERE Id_depart = " +
dataGridView1.SelectedRows[0].Cells[0].Value.ToString() + ";";
            ExecuteQuery(txtQuery);
        }

        private void DataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBox1.Text =
dataGridView1.SelectedRows[0].Cells[1].Value.ToString();
        }
    }
}
```

Г.13 Форма оновлення таблиці DepartProject

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
```

```
namespace CRMka
{
    public partial class editDepartProj : Form
    {
        public void SetConnection()
        {
            db_con = new SqlConnection(@"Data Source=DESKTOP-
8MQT1B7\SQLEXPRESS;Initial Catalog=CRMDB;Integrated Security=True;");
        }

        private void ExecuteQuery(string txtQuery)
        {
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            db_cmd.CommandText = txtQuery;
            db_cmd.ExecuteNonQuery();
            db_con.Close();
        }


        private SqlConnection db_con;
        private SqlCommand db_cmd;
        private DataSet DS = new DataSet();
        private DataTable DT = new DataTable();

        public editDepartProj()
        {
            InitializeComponent();
        }

        private void EditDepartProj_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet4.ViewDepartProject". При необходимости она может быть перемеще-
на или удалена.

this.viewDepartProjectTableAdapter1.Fill(this.cRMDBDataSet4.ViewDepartProject
);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet3.Status". При необходимости она может быть перемещена или уда-
лена.
            this.statusTableAdapter.Fill(this.cRMDBDataSet3.Status);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Projects". При необходимости она может быть перемещена или
удалена.
            this.projectsTableAdapter.Fill(this.cRMDBDataSet2.Projects);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Departments". При необходимости она может быть перемещена или
удалена.

this.departmentsTableAdapter.Fill(this.cRMDBDataSet2.Departments);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.ViewDepartProject". При необходимости она может быть перемеще-
на или удалена.

this.viewDepartProjectTableAdapter.Fill(this.cRMDBDataSet2.ViewDepartProject)
;

        }

        private void CancelButton_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void EditButton_Click(object sender, EventArgs e)
        {
            string txtQuery = "UPDATE DepartProject SET Id_depart = '" +
comboBox1.Text + "', Id_project = '"+ comboBox2.Text + "', Id_status = '" +
```

```
comboBox3.Text +"' WHERE Id = " +
dataGridView1.SelectedRows[0].Cells[0].Value.ToString() + ";";
            ExecuteQuery(txtQuery);
        }
    }
}
```

Г.14 Форма оновлення таблиці Staff

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace CRMka
{
    public partial class editEmp : Form
    {
        public void SetConnection()
        {
            db_con = new SqlConnection(@"Data Source=DESKTOP-
8MQT1B7\SQLEXPRESS;Initial Catalog=CRMDB;Integrated Security=True;");
        }

        private void ExecuteQuery(string txtQuery)
        {
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            db_cmd.CommandText = txtQuery;
            db_cmd.ExecuteNonQuery();
            db_con.Close();
        }


        private SqlConnection db_con;
        private SqlCommand db_cmd;
        private DataSet DS = new DataSet();
        private DataTable DT = new DataTable();

        public editEmp()
        {
            InitializeComponent();
        }

        private void EditEmp_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Staff". При необходимости она может быть перемещена или удале-
на.
            this.staffTableAdapter.Fill(this.cRMDBDataSet2.Staff);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Post". При необходимости она может быть перемещена или удале-
на.
            this.postTableAdapter.Fill(this.cRMDBDataSet2.Post);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Departments". При необходимости она может быть перемещена или
удалена.

this.departmentsTableAdapter.Fill(this.cRMDBDataSet2.Departments);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.ViewStaff". При необходимости она может быть перемещена или
удалена.
```

```
            this.viewStaffTableAdapter.Fill(this.cRMDBDataSet2.ViewStaff);

        }

        private void CancelButton_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void EditButton_Click(object sender, EventArgs e)
        {
            string txtQuery = "UPDATE Staff SET First_name = '" +
textBox1.Text + "',Last_name = '" + textBox2.Text + "',Middle_name = '" +
textBox3.Text + "',Id_depart = '" + comboBox1.Text + "',Id_post = '" +
comboBox2.Text + "',Login_emp = '" + textBox4.Text + "',Emp_password = '" +
textBox5.Text + "',Emp_phone = '" + textBox6.Text + "',Emp_number = '" +
textBox7.Text + "' WHERE Id_emp = " +
dataGridView1.SelectedRows[0].Cells[0].Value.ToString() + ";";
            ExecuteQuery(txtQuery);
            Close();
        }

        private void DataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBox1.Text =
dataGridView1.SelectedRows[0].Cells[1].Value.ToString();
            textBox2.Text =
dataGridView1.SelectedRows[0].Cells[2].Value.ToString();
            textBox3.Text =
dataGridView1.SelectedRows[0].Cells[3].Value.ToString();
            textBox4.Text =
dataGridView1.SelectedRows[0].Cells[6].Value.ToString();
            textBox5.Text =
dataGridView1.SelectedRows[0].Cells[7].Value.ToString();
            textBox6.Text =
dataGridView1.SelectedRows[0].Cells[8].Value.ToString();
            textBox7.Text =
dataGridView1.SelectedRows[0].Cells[9].Value.ToString();
        }
    }
}
```

Г.15 Форма оновлення таблиці Performers

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace CRMka
{
    public partial class editPerformer : Form
    {
        public void SetConnection()
        {
            db_con = new SqlConnection(@"Data Source=DESKTOP-
8MQT1B7\SQLEXPRESS;Initial Catalog=CRMDB;Integrated Security=True;");
        }

        private void ExecuteQuery(string txtQuery)
        {
            SetConnection();
```

```
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            db_cmd.CommandText = txtQuery;
            db_cmd.ExecuteNonQuery();
            db_con.Close();
        }


        private SqlConnection db_con;
        private SqlCommand db_cmd;
        private DataSet DS = new DataSet();
        private DataTable DT = new DataTable();

        public editPerformer()
        {
            InitializeComponent();
        }

        private void EditPerformer_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet4.ViewStaff". При необходимости она может быть перемещена или
удалена.
            this.viewStaffTableAdapter.Fill(this.cRMDBDataSet4.ViewStaff);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Problems". При необходимости она может быть перемещена или
удалена.
            this.problemsTableAdapter.Fill(this.cRMDBDataSet2.Problems);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Staff". При необходимости она может быть перемещена или удале-
на.
            this.staffTableAdapter.Fill(this.cRMDBDataSet2.Staff);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.ViewProblems". При необходимости она может быть перемещена или
удалена.

this.viewProblemsTableAdapter.Fill(this.cRMDBDataSet2.ViewProblems);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.ViewPerformers". При необходимости она может быть перемещена
или удалена.

this.viewPerformersTableAdapter.Fill(this.cRMDBDataSet2.ViewPerformers);

        }

        private void CancelButton_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void EditButton_Click(object sender, EventArgs e)
        {
            string txtQuery = "UPDATE Performers SET Id_emp = '" +
comboBox1.Text + "',Id_problem = '" + comboBox2.Text + "',About_duration= '"
+ textBox1.Text + "',Fact_duration = '" + textBox2.Text + " WHERE
Id_performer = " + dataGridView1.SelectedRows[0].Cells[0].Value.ToString() +
";";
            ExecuteQuery(txtQuery);
            Close();
        }

        private void DataGridView2_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBox1.Text =
dataGridView1.SelectedRows[0].Cells[3].Value.ToString();
            textBox2.Text =
dataGridView1.SelectedRows[0].Cells[4].Value.ToString();
        }
    }
}
```

Г.16 Форма оновлення таблиці Post

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace CRMka
{
    public partial class editPost : Form
    {
        public void SetConnection()
        {
            db_con = new SqlConnection(@"Data Source=DESKTOP-
8MQT1B7\SQLEXPRESS;Initial Catalog=CRMDB;Integrated Security=True;");
        }

        private void ExecuteQuery(string txtQuery)
        {
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            db_cmd.CommandText = txtQuery;
            db_cmd.ExecuteNonQuery();
            db_con.Close();
        }


        private SqlConnection db_con;
        private SqlCommand db_cmd;
        private DataSet DS = new DataSet();
        private DataTable DT = new DataTable();
        public editPost()
        {
            InitializeComponent();
        }

        private void EditPost_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Departments". При необходимости она может быть перемещена или
удалена.

this.departmentsTableAdapter.Fill(this.cRMDBDataSet2.Departments);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.ViewDepart". При необходимости она может быть перемещена или
удалена.
            this.viewDepartTableAdapter.Fill(this.cRMDBDataSet2.ViewDepart);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.PostView". При необходимости она может быть перемещена или
удалена.
            this.postViewTableAdapter.Fill(this.cRMDBDataSet2.PostView);

        }

        private void CancelButton_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void EditButton_Click(object sender, EventArgs e)
        {
```

```
            string txtQuery = "UPDATE Post SET Name_post = '" + textBox1.Text
+ "', Id_depart = '" + comboBox1.Text + "' WHERE Id_post = " +
dataGridView1.SelectedRows[0].Cells[0].Value.ToString() + ";";
            ExecuteQuery(txtQuery);
        }

        private void DataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBox1.Text =
dataGridView1.SelectedRows[0].Cells[1].Value.ToString();
        }
    }
}
```

Г.17 Форма оновлення таблиці Problems

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace CRMka
{
    public partial class editProblem : Form
    {
        public void SetConnection()
        {
            db_con = new SqlConnection(@"Data Source=DESKTOP-
8MQT1B7\SQLEXPRESS;Initial Catalog=CRMDB;Integrated Security=True;");
        }

        private void ExecuteQuery(string txtQuery)
        {
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            db_cmd.CommandText = txtQuery;
            db_cmd.ExecuteNonQuery();
            db_con.Close();
        }


        private SqlConnection db_con;
        private SqlCommand db_cmd;
        private DataSet DS = new DataSet();
        private DataTable DT = new DataTable();


        public editProblem()
        {
            InitializeComponent();
        }

        private void EditProblem_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Priorities". При необходимости она может быть перемещена или
удалена.
            this.prioritiesTableAdapter.Fill(this.cRMDBDataSet2.Priorities);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.ViewProblems". При необходимости она может быть перемещена или
удалена.
```

```
this.viewProblemsTableAdapter.Fill(this.cRMDBDataSet2.ViewProblems);

        }

        private void DataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBox2.Text =
dataGridView1.SelectedRows[0].Cells[2].Value.ToString();
            textBox3.Text =
dataGridView1.SelectedRows[0].Cells[3].Value.ToString();
            textBox4.Text =
dataGridView1.SelectedRows[0].Cells[4].Value.ToString();
            textBox5.Text =
dataGridView1.SelectedRows[0].Cells[5].Value.ToString();
            textBox6.Text =
dataGridView1.SelectedRows[0].Cells[6].Value.ToString();
            textBox7.Text =
dataGridView1.SelectedRows[0].Cells[7].Value.ToString();
        }

        private void EditButton_Click(object sender, EventArgs e)
        {
            string txtQuery = "UPDATE Problems SET Id_Priority = '" +
comboBox1.Text + "',Problem_description = '" + textBox2.Text +
"',Problem_solution= '" + textBox3.Text + "',Problem_registration = '" +
textBox4.Text + "',Deadline = '" + textBox5.Text + "',About_duration = " +
textBox6.Text + ",Fact_duration = " + textBox7.Text + " WHERE Id_problem = "
+ dataGridView1.SelectedRows[0].Cells[0].Value.ToString() + ";";
            ExecuteQuery(txtQuery);
            Close();
        }
    }
}
```

Г.18 Форма оновлення таблиці Projects

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace CRMka
{
    public partial class editProject : Form
    {
        public void SetConnection()
        {
            db_con = new SqlConnection(@"Data Source=DESKTOP-
8MQT1B7\SQLEXPRESS;Initial Catalog=CRMDB;Integrated Security=True;");
        }

        private void ExecuteQuery(string txtQuery)
        {
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            db_cmd.CommandText = txtQuery;
            db_cmd.ExecuteNonQuery();
            db_con.Close();
        }
```

```
        private SqlConnection db_con;
        private SqlCommand db_cmd;
        private DataSet DS = new DataSet();
        private DataTable DT = new DataTable();

        public editProject()
        {
            InitializeComponent();
        }

        private void EditProject_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.ViewProjects". При необходимости она может быть перемещена или
удалена.

this.viewProjectsTableAdapter.Fill(this.cRMDBDataSet2.ViewProjects);

        }

        private void CancelButton_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void EditButton_Click(object sender, EventArgs e)
        {
            string txtQuery = "UPDATE Projects SET Name_project = '" +
textBox1.Text + "' WHERE Id_project = " +
dataGridView1.SelectedRows[0].Cells[0].Value.ToString() + ";";
            ExecuteQuery(txtQuery);
        }

        private void DataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBox1.Text =
dataGridView1.SelectedRows[0].Cells[1].Value.ToString();
        }
    }
}
```

Г.19 Форма оновлення таблиці ProjectProblem

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace CRMka
{
    public partial class editProjProb : Form
    {

        public void SetConnection()
        {
            db_con = new SqlConnection(@"Data Source=DESKTOP-
8MQT1B7\SQLEXPRESS;Initial Catalog=CRMDB;Integrated Security=True;");
        }

        private void ExecuteQuery(string txtQuery)
        {
```

```
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            db_cmd.CommandText = txtQuery;
            db_cmd.ExecuteNonQuery();
            db_con.Close();
        }


        private SqlConnection db_con;
        private SqlCommand db_cmd;
        private DataSet DS = new DataSet();
        private DataTable DT = new DataTable();

        public editProjProb()
        {
            InitializeComponent();
        }

        private void EditProjProb_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Projects". При необходимости она может быть перемещена или
удалена.
            this.projectsTableAdapter.Fill(this.cRMDBDataSet2.Projects);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.Problems". При необходимости она может быть перемещена или
удалена.
            this.problemsTableAdapter.Fill(this.cRMDBDataSet2.Problems);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet2.ViewProjectProblem". При необходимости она может быть переме-
щена или удалена.

this.viewProjectProblemTableAdapter.Fill(this.cRMDBDataSet2.ViewProjectProble
m);

        }

        private void CancelButton_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void EditButton_Click(object sender, EventArgs e)
        {
            string txtQuery = "UPDATE ProjectProblems SET Id_problem = '" +
comboBox1.Text + "',Id_project = '" + comboBox2.Text + "' WHERE
Id_ProjectProblem = " +
dataGridView1.SelectedRows[0].Cells[0].Value.ToString() + ";";
            ExecuteQuery(txtQuery);
            Close();
        }
    }
}
```

Г.20 Форма оновлення таблиці StaffProject

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace CRMka
```

```
{
    public partial class editStaffProject : Form
    {
        public void SetConnection()
        {
            db_con = new SqlConnection(@"Data Source=DESKTOP-
8MQT1B7\SQLEXPRESS;Initial Catalog=CRMDB;Integrated Security=True;");
        }

        private void ExecuteQuery(string txtQuery)
        {
            SetConnection();
            db_con.Open();
            db_cmd = db_con.CreateCommand();
            db_cmd.CommandText = txtQuery;
            db_cmd.ExecuteNonQuery();
            db_con.Close();
        }


        private SqlConnection db_con;
        private SqlCommand db_cmd;
        private DataSet DS = new DataSet();
        private DataTable DT = new DataTable();

        public editStaffProject()
        {
            InitializeComponent();
        }

        private void EditStaffProject_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet4.Status". При необходимости она может быть перемещена или уда-
лена.
            this.statusTableAdapter.Fill(this.cRMDBDataSet4.Status);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet4.Projects". При необходимости она может быть перемещена или
удалена.
            this.projectsTableAdapter.Fill(this.cRMDBDataSet4.Projects);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet4.ViewStaff". При необходимости она может быть перемещена или
удалена.
            this.viewStaffTableAdapter.Fill(this.cRMDBDataSet4.ViewStaff);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
"cRMDBDataSet4.ViewStaffProject". При необходимости она может быть перемещена
или удалена.

this.viewStaffProjectTableAdapter.Fill(this.cRMDBDataSet4.ViewStaffProject);

        }

        private void CancelButton_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void EditButton_Click(object sender, EventArgs e)
        {
            string txtQuery = "UPDATE StaffProject SET Id_emp = '" +
comboBox1.Text + "', Id_project = '" + comboBox2.Text + "', Id_status = '" +
comboBox3.Text + "' WHERE Id = " +
dataGridView1.SelectedRows[0].Cells[0].Value.ToString() + ";";
            ExecuteQuery(txtQuery);
        }
    }
}
```