Министерство образования и науки Российской Федерации
Рязанский государственный радиотехнический университет

Кафедра электронных вычислительных машин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по дисциплине

«***Клиент-серверные приложения баз данных***»

на тему:

Информационная система

«*Аптека*»

Выполнили:

студенты группы

Проверил:

Рязань 2012

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сложно представить современный город без крупной сети аптек. Чаще всего таких сетей гораздо больше одной. Наш проект и будет представлять собой одну из таких сетей.

Эта тема показалась нам наиболее интересной, т.к. она затрагивает важную для жизни сферу, в которой всегда будет спрос. В рамках данного курсового проекта необходимо разработать информационную систему с клиент-серверной архитектурой «Аптека». Она будет позволять администратору просматривать количество препаратов, в случае необходимости оформить его поставку, проверять и изменять информацию о каждом сотруднике, сети. Кассирам она позволяет просматривать остатки товара, продавать его, выдавать дисконтные карты. Также мы не забываем о наших покупателях, которые могут просмотреть наличие/количество нужных им препаратов и информацию о себе (для владельцев дисконтных карт).

Для выполнения поставленной задачи будет использоваться следующее программное обеспечение:

- СУБД: MS SQL Server 2005;
- система программирования: Microsoft Visual C# 2005;
- CASE средства проектирования баз данных: Sybase PowerDesigner 15.

1. АНАЛИЗ ЗАДАЧИ

1.1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ, ВЫЯВЛЕНИЕ НЕОБХОДИМОЙ ПОЛЬЗОВАТЕЛЮ ФУНКЦИОНАЛЬНОСТИ

В данном курсовом проекте рассматривается такая область, как «Аптека». Каждый покупатель должен иметь возможность просмотреть наличие того или иного препарата. Также ему должна быть предоставлена возможность обзора информации о совершённых им покупках.

Кассиру, помимо возможностей покупателя, добавляется возможность продавать товар. Т.е. фактически уменьшать его количества в наличии.

Деятельность администратора шире деятельности продавца, т.к. он имеет возможность редактирования и сохранения сведений всей базы данных «Аптека». Именно от этого человека зависит правильность корректировки информации, доводимой до кассира и покупателя.

В базе данных должна храниться информация о филиалах аптек, о препаратах, их количестве в наличии, их принадлежности к определённой категории, о сотрудниках, которые работают в филиалах, их должности, заработная плата и личная информация, о поставщиках, о покупателях, и соответственно информация о поставках и покупках.

Клиентское приложение должно в удобной для пользователя форме предоставлять информацию для просмотра или редактирования в зависимости его от учетной записи, чему будет способствовать дружественный и довольно простой интерфейс.

1.2. РАЗРАБОТКА ОБЩЕЙ АРХИТЕКТУРЫ ИНФОРМАЦИОННОЙ СИСТЕМЫ

В курсовом проекте применена клиент-серверная архитектура с применением двухзвенной модели DBS (DateBase Server - сервер баз данных). Для этой модели характерно, что функции компьютера клиента ограничиваются функциями представления информации, в то время как прикладные функции обеспечиваются приложением, находящемся на компьютере сервере. При этом приложения реализуются в виде хранимых процедур.

Процедуры обычно хранятся в словаре базы данных и разделяются несколькими клиентами.

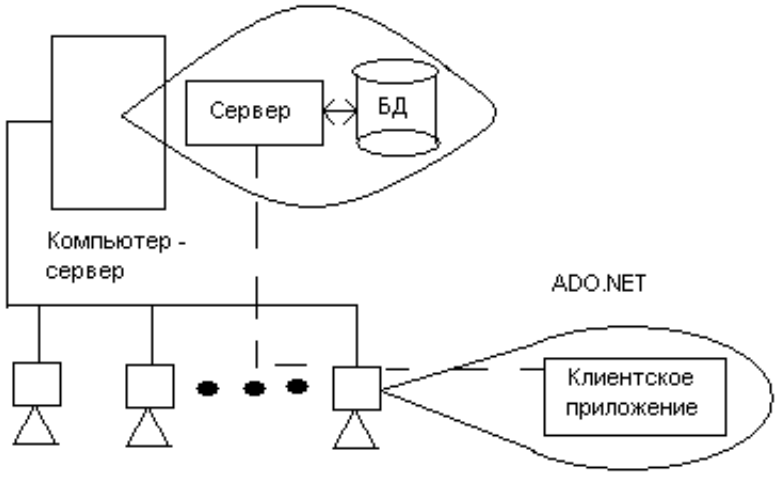

Рис. 1.1. Двухзвенная модель архитектуры клиент-сервер

Достоинствами DBS является возможность хорошего администрирования приложения на этапах разработки, сопровождения и модификации, а также эффективного использования вычислительных и коммуникационных ресурсов.

2. РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ИНФОРМАЦИОННОЙ СИСТЕМЫ

2.1. РАЗРАБОТКА КОЦЕПТУАЛЬНОЙ МОДЕЛИ ДАННЫХ

2.1.1. ВЫЯВЛЕНИЕ СУЩНОСТЕЙ, ИХ АТРИБУТОВ И КЛЮЧЕЙ

В ходе разработки информационной системы были выявлены следующие сущности:

1) Филиалы
   Первичный ключ: ID_Филиала
2) Сотрудники
   Первичный ключ: ID_Сотрудники
3) Должности
   Первичный ключ: ID_ Должности
4) Препараты
   Первичный ключ: ID_Препараты
5) Категории
   Первичный ключ: ID_Категории
6) Склад
   Первичный ключ: ID_Склад
7) Покупатели
   Первичный ключ: ID_Покупатели
8)Покупка
   Первичный ключ: ID_Покупки
9)Поставщики
   Первичный ключ: ID_Поставщики
10)Поставка
   Первичный ключ: ID_Поставки
11)Скидки
   Первичный ключ: ID_Скидки
12)Привилегированные сотрудники
   Первичный ключ: ID_Сотрудники
13)Капитал
   Первичный ключ: ID_Капитал

2.1.2. ВЫЯВЛЕНИЕ СВЯЗЕЙ

В ходе разработки информационной системы были выявлены следующие связи между сущностями:

1)

Рис. 2.1. Связь «Совершаются в»

В один филиал могут быть реализованы разные поставки. Одной поставке соответствует один филиал. Следовательно, имеет место связь 1..N. Каждой поставке должен быть присвоен филиал. Необязательно, что в каждый филиал будет совершена поставка. Следовательно, для связи «Филиалы → Поставки» будет использоваться кардинальность 0,n, а для связи «Поставки → Филиалы» - 1..1.

2)

Рис. 2.2. Связь «Происходят в»

В одном филиале могут происходить разные покупки. Одной покупке соответствует один определённый филиал. Следовательно, имеет место связь 1..N. Каждая покупка происходит в определённом филиале. Необязательно, что в каждом филиале произойдёт покупка. Следовательно, для связи «Филиалы → Покупки» будет использоваться кардинальность 0..N, а для связи «Покупки → Филиалы» - 1..1.

3)

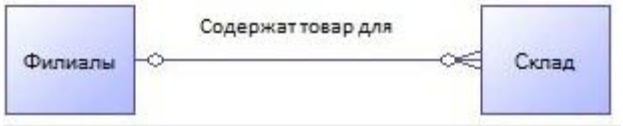

Рис. 2.3. Связь «Содержит товар»

Для филиала приписан только один склад. Склад предназначен для многих филиалов. Следовательно, имеет место связь 1..N. Каждый филиал должен иметь склад. Необязательно, на складе хранятся препараты для каждого филиала. Следовательно, для связи «Филиалы → Склад» будет использоваться кардинальность 1..1, а для связи «Склад → Филиалы» - 0..N.

4)

Рис. 2.4. Связь «Работают в»

Каждый сотрудник должен работать в каком-нибудь филиале. В филиале может не быть того или иного сотрудника. Следовательно, имеет место связь 1..N. Следовательно, для связи «Филиалы → Сотрудники» будет использоваться кардинальность 0..N, а для связи «Сотрудники → Филиалы» - 1..1.

5)

Рис. 2.5. Связь «Занимают»

На одной должности могут работать много сотрудников. Каждый сотрудник может занимать только одну должность. Следовательно, имеет место связь 1..N. Необязательно на каждой должности должен работать сотрудник. Следовательно, для связи «Должности→

Сотрудники» будет использоваться кардинальность 0..N, а для связи «Сотрудники → Должности» - 0..1.

6)

Рис. 2.6. Связь «Содержат»

В одной поставке может быть только один препарат. Каждый препарат может поставляться много раз. Следовательно, имеет место связь 1..N. Необязательно, что поставке будет тот или иной препарат. Но в поставке должен быть один определённый препарат. Следовательно, для связи «Препараты → Поставки» будет использоваться кардинальность 0..N, а для связи «Поставки → Препараты» - 1..1.

7)

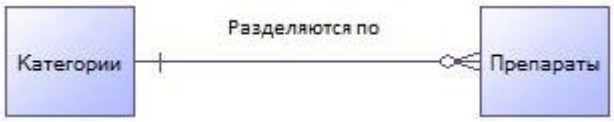

Рис. 2.7. Связь «Разделяются по»

В одной категории может быть несколько препаратов. Каждый препарат может быть приписан только к одной категории. Следовательно, имеет место связь 1..N. Необязательно, что в каждой категории есть препарат. Каждый препарат обязательно в той или иной категории. Следовательно, для связи «Категории → Препараты» будет использоваться кардинальность 0,n, а для связи «Препараты → Категории» - 1..1.

8)

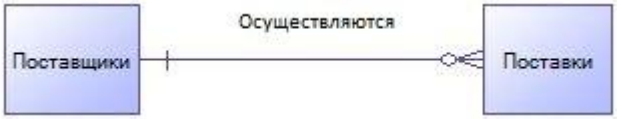

Рис. 2.8. Связь «Осуществляются»

Один поставщик может совершать несколько поставок. Поставка может быть осуществлена одним поставщиком. Следовательно, имеет место связь 1..N. Необязательно, что каждый поставщик осуществит поставку. Каждая поставка совершается поставщиком. Следовательно, для связи «Поставщики → Поставки» будет использоваться кардинальность 0..N, а для связи «Поставки → Поставщики» - 1..1.

9)

Рис. 2.9. Связь «Содержат»

Один препарат может быть куплен несколько раз. В покупке только 1н препарат. Следовательно, имеет место связь 1..N. Необязательно, что в каждой покупке будет тот или иной препарат. Каждая покупка содержит препарат. Следовательно, для связи «Препараты → Покупки» будет использоваться кардинальность 0..N, а для связи «Покупки → Препараты» - 1..1.

10)

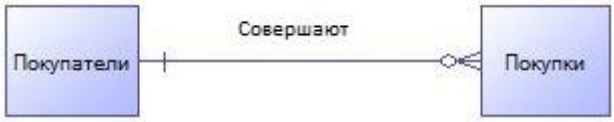

Рис. 2.10. Связь «Совершаются»

Один покупатель может совершить несколько покупок. Покупка одновременно совершается только 1м покупателем. Следовательно, имеет место связь 1..N. Необязательно, что в каждый покупатель совершает покупку. Каждая покупка совершается покупателем. Следовательно, для связи «Покупатели → Покупки» будет использоваться кардинальность 0..N, а для связи «Покупки → Покупатели» - 1..1.

2.1.3. ПОСТРОЕНИЕ CDM

ER-диаграмма, построенная для предметной области «Аптека», на основе выявленных ранее сущностей и связей между ними представлена на рисунке 2.11.

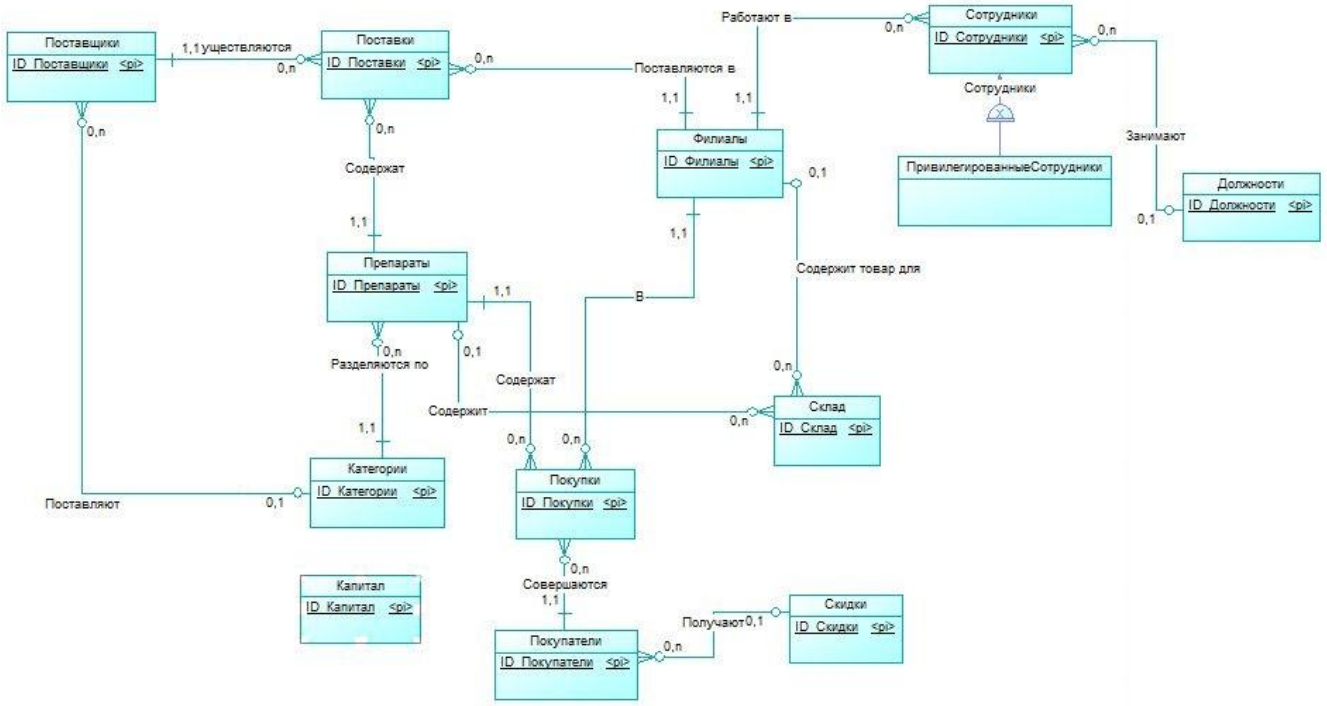

Рис. 2.11. Концептуальная модель данных для предметной области «Аптека».

2.2. РАЗРАБОТКА ЛОГИЧЕСКОЙ МОДЕЛИ ДАННЫХ

2.2.1. ЗАПОЛНЕНИЕ СУЩНОСТЕЙ АТРИБУТАМИ

После перехода к логической модели данных были получены предварительные отношения, которые необходимо заполнить атрибутами:

1. Поставки:
   1) ID_Поставки,
   2) Препараты,
   3) Филиалы,
   4) Поставщики,
   5) Цена поставки,
   6) Количество,
   7) Итого;
2. Поставщики:
   1) ID_Поставщики,
   2) Категории,
   3) ФИО,

4) Адрес,
5) Телефон;

3. Препараты:
   1) ID_Препараты,
   2) Категории,
   3) Название,
   4) Цена;
4. Категории:
   1) ID_Категории,
   2) Название;
5. Покупки:
   1) ID_Покупки,
   2) Филиалы,
   3) Покупатели,
   4) Цена со скидкой,
   5) Количество,
   6) Итого;
6. Капитал:
   1) ID_Капитал;
7. Покупатели:
   1) ID_Покупатели,
   2) Скидки,
   3) ФИО,
   4) Общая сумма покупок;
8. Филиалы:
   1) ID_Филиалы,
   2) Название,
   3) Адрес,
   4) Телефон;
9. Склад:
   1) ID_Склад,
   2) Филиалы,
   3) Препараты,
   4) Количество;
10. Скидки:
    1) ID_Скидки,
    2) Название,
    3) Размер;
11. Привилегированные сотрудники:
    1) ID_Сотрудники,
    2) Должности,
    3) Филиалы,
    4) Логин,
    5) Пароль;
12. Сотрудники:
    1) ID_Сотрудники,
    2) Должности,
    3) Филиалы,
    4) ФИО,
    5) Адрес;
13. Должности:

1) ID_Должности,
2) Название,
3) Зарплата.

2.2.2. ПРОВЕРКА СУЩНОСТЕЙ НА СООТВЕТСТВИЕ НОРМАЛЬНЫМ ФОРМАМ

Для каждого отношения нашей БД построим диаграмму функциональных зависимостей и определим, в какой нормальной форме оно находится.

На рисунке 2.13 показана диаграмма функциональных зависимостей для отношения «Должности».

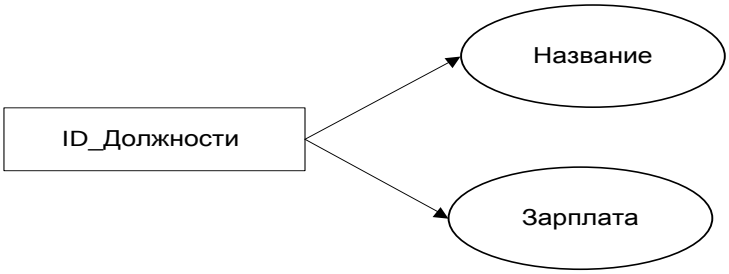

Рис. 2.13. Диаграмма функциональных зависимостей для отношения «Должности»

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

На рисунке 2.14 показана диаграмма функциональных зависимостей для отношения «Сотрудники».

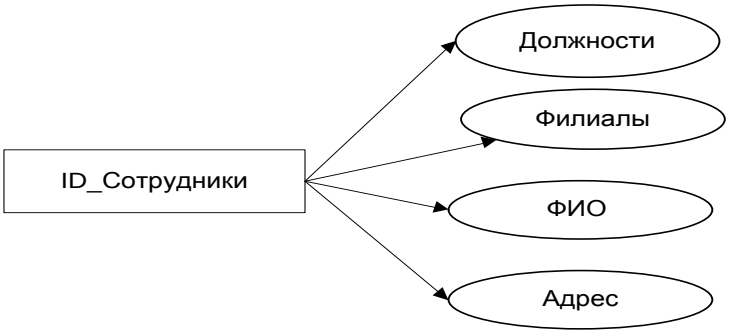

Рис. 2.14. Диаграмма функциональных зависимостей для отношения «Сотрудники»

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

На рисунке 2.15 показана диаграмма функциональных зависимостей для отношения «Привилегированные Сотрудники».

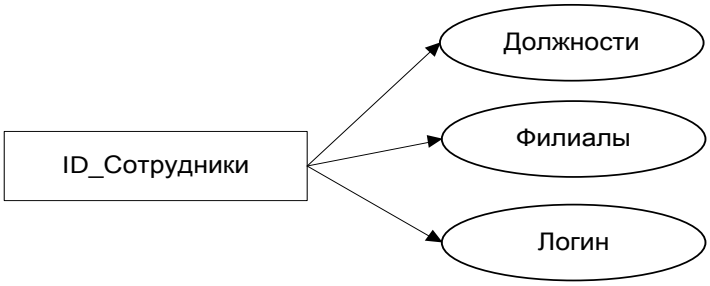

Рис. 2.15. Диаграмма функциональных зависимостей для отношения «Привилегированные Сотрудники»

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

На рисунке 2.16 показана диаграмма функциональных зависимостей для отношения «Филиалы».

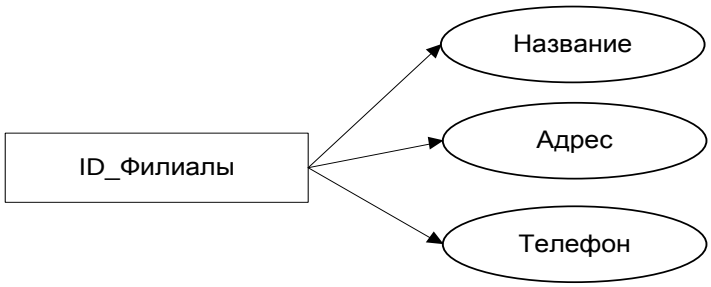

Рис. 2.16. Диаграмма функциональных зависимостей для отношения «Филиалы»

Отношение находится в 1НФ, 2НФ, 3НФ, БКНФ.

На рисунке 2.17 показана диаграмма функциональных зависимостей для отношения «Поставки».

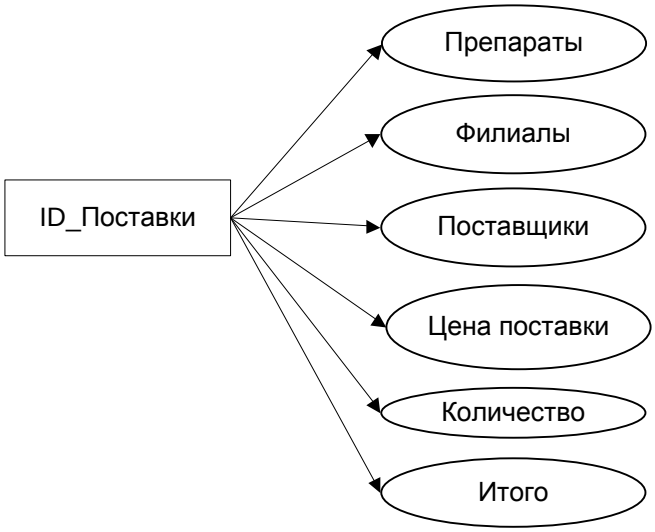

Рис. 2.17. Диаграмма функциональных зависимостей для отношения «Поставки».

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

На рисунке 2.18 показана диаграмма функциональных зависимостей для отношения «Препараты».

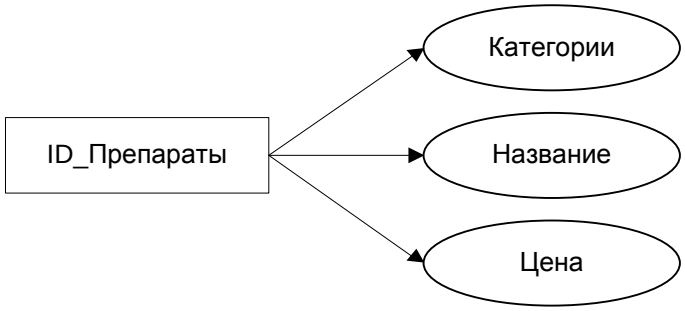

Рис. 2.18. Диаграмма функциональных зависимостей для отношения «Препараты»

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

Диаграмма функциональных зависимостей для отношения «Поставщики» примет вид, показанный на рисунке 2.19.

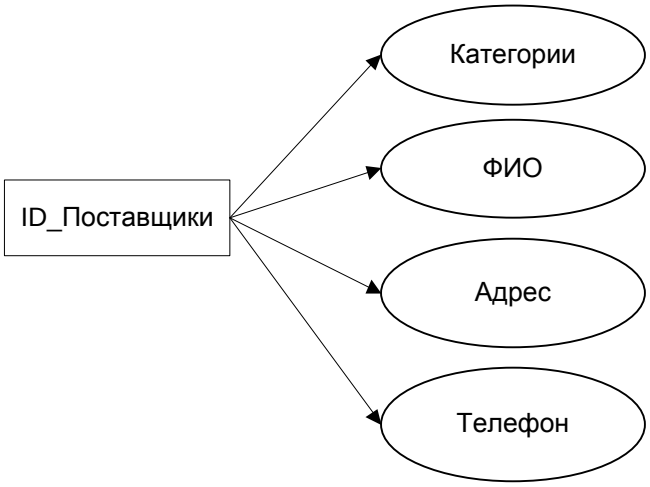

Рис. 2.19. Новая диаграмма функциональных зависимостей для отношения «Поставщики»

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

На рисунке 2.20 показана диаграмма функциональных зависимостей для отношения «Категории».

Рис. 2.20. Диаграмма функциональных зависимостей для отношения «Категории»

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

На рисунке 2.21 показана диаграмма функциональных зависимостей для отношения «Склад».

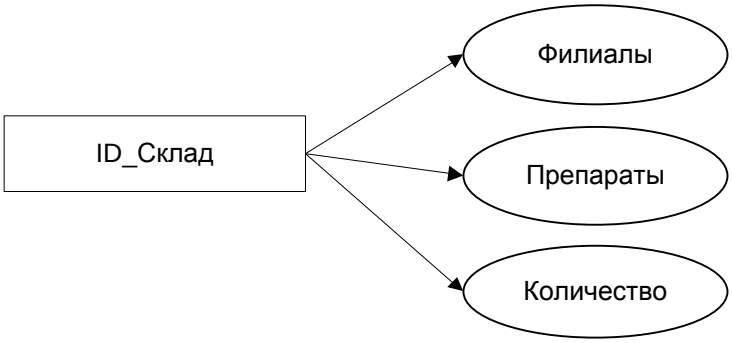

Рис. 2.21. Диаграмма функциональных зависимостей для отношения «Склад»

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

На рисунке 2.22 показана диаграмма функциональных зависимостей для отношения «Покупки».

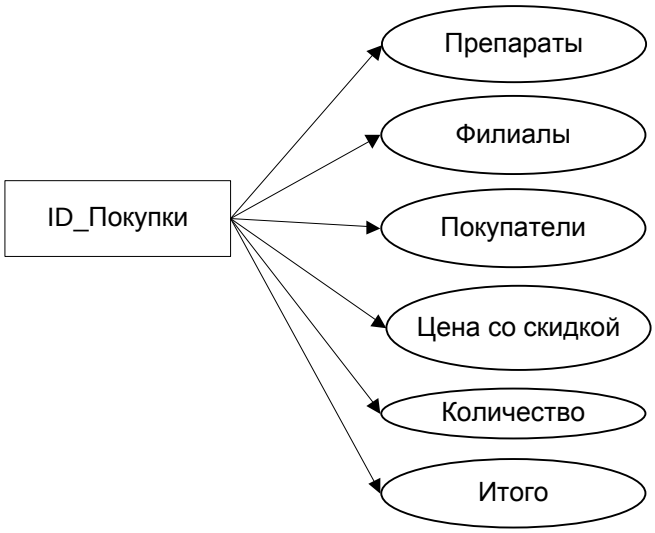

Рис. 2.22. Диаграмма функциональных зависимостей для отношения «Покупки»

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

На рисунке 2.23 показана диаграмма функциональных зависимостей для отношения «Покупатели».

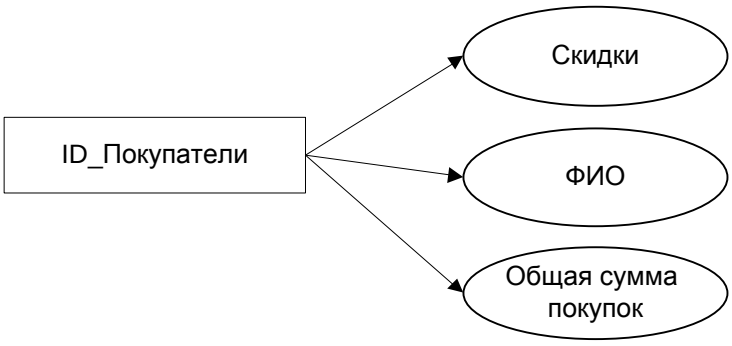

Рис. 2.23. Диаграмма функциональных зависимостей для отношения «Покупатели»

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

На рисунке 2.24 показана диаграмма функциональных зависимостей для отношения «Скидки».

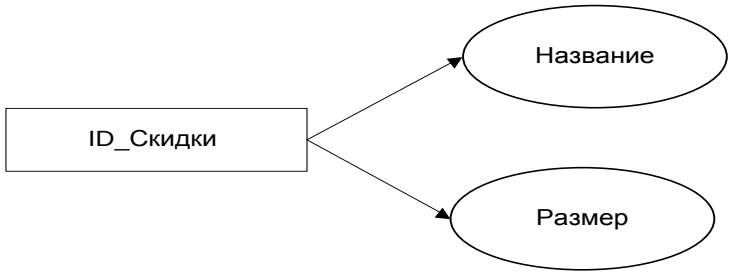

Рис. 2.24. Диаграмма функциональных зависимостей для отношения «Скидки»

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

На рисунке 2.25 показана диаграмма функциональных зависимостей для отношения «Капитал».

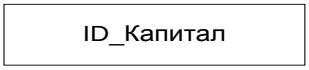

Рис. 2.25. Диаграмма функциональных зависимостей для отношения «Капитал»

Отношение находится в 1НФ, 2НФ, 3НФ, НФБК.

2.2.3. ПОСТРОЕНИЕ LDM

После заполнения атрибутами и нормализации отношений построена следующая логическая модель данных (рис. 2.26).

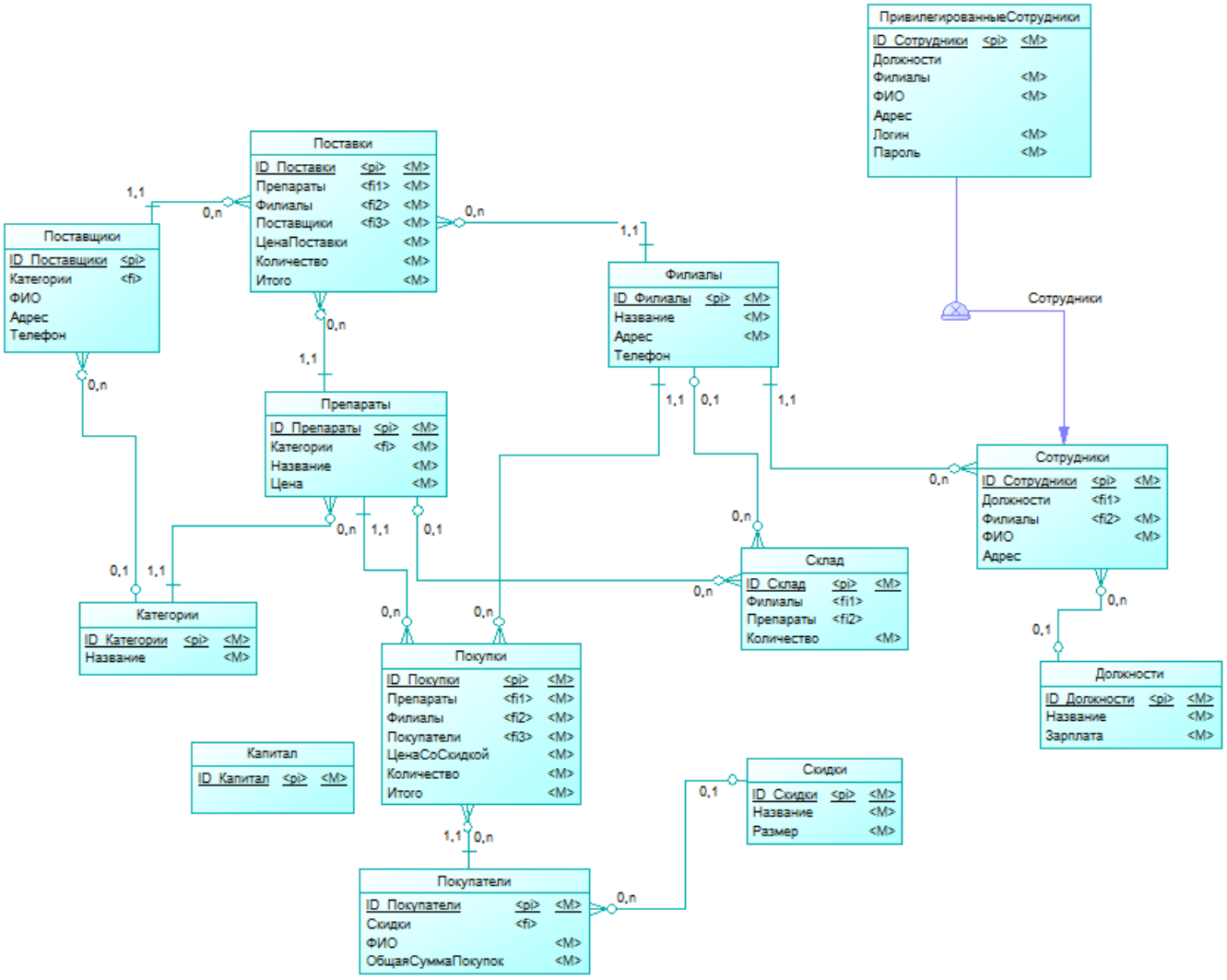

Рис. 2.26. Логическая модель данных для предметной области «Аптека»

2.3. РАЗРАБОТКА ФИЗИЧЕСКОЙ МОДЕЛИ ДАННЫХ

2.3.1. ЗАДАНИЕ ТИПОВ ДАННЫХ ДЛЯ ПОЛЕЙ ТАБЛИЦ

Для полей таблиц были заданы типы, указанные в таблице 1.

Таблица 1

Таблица	Поле	Тип
Поставки	ID_Поставки	int
	Препараты	int
	Филиалы	int
	Поставщики	int
	Цена поставки	int
	Количество	int
	Итого	int
Поставщики	ID_Поставщики	int
	Категории	int
	ФИО	varchar(255)
	Адрес	varchar(255)
	Телефон	int

Таблица	Поле	Тип
Препараты	ID_Препараты	int
	Категории	int
	Название	varchar(25)
	Цена	int
Категории	ID_Категории	int
	Название	varchar(25)
Капитал	ID_Капитал	int
Покупки	ID_Покупки	int
	Препараты	int
	Филиалы	int
	Покупатели	int
	Цена со скидкой	int
	Количество	int
	Итого	int
Покупатели	ID_Покупатели	int
	Скидки	int
	ФИО	varchar(255)
	ОбщаяСуммаПокупок	int
Филиалы	ID_Филиалы	int
	Название	varchar(25)
	Адрес	varchar(255)
	Телефон	int
Склад	ID_Склад	int
	Филиалы	int
	Препараты	int
	Количество	int
Скидки	ID_Скидки	int
	Название	varchar(25)
	Размер	smallint
Привилегированные сотрудники	ID_Сотрудники	int
	Логин	varchar(255)
	Пароль	varchar(30)
Сотрудники	ID_Сотрудники	int
	Должности	int
	Филиалы	int
	ФИО	varchar(255)
	Адрес	varchar(255)
Должности	ID_Должности	int
	Название	varchar(25)
	Зарплата	int

2.3.2. ЗАДАНИЕ ЧАСТНЫХ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ ДАННЫХ

Частные ограничения целостности задаются первичными и внешними ключами, разрешением каскадного обновления. Ограничения целостности для взаимоисключающего законченного наследования будут реализованы на уровне хранимых процедур и путем запрета обычным пользователям БД выполнения запросов INSERT, UPDATE и DELETE напрямую. Это позволит не реализовывать данные ограничения с помощью триггеров, замедляющих работу БД при большой активности пользователей.

2.3.3. ПОСТРОЕНИЕ PDM

Физическая модель приведена на рисунке 2.27.

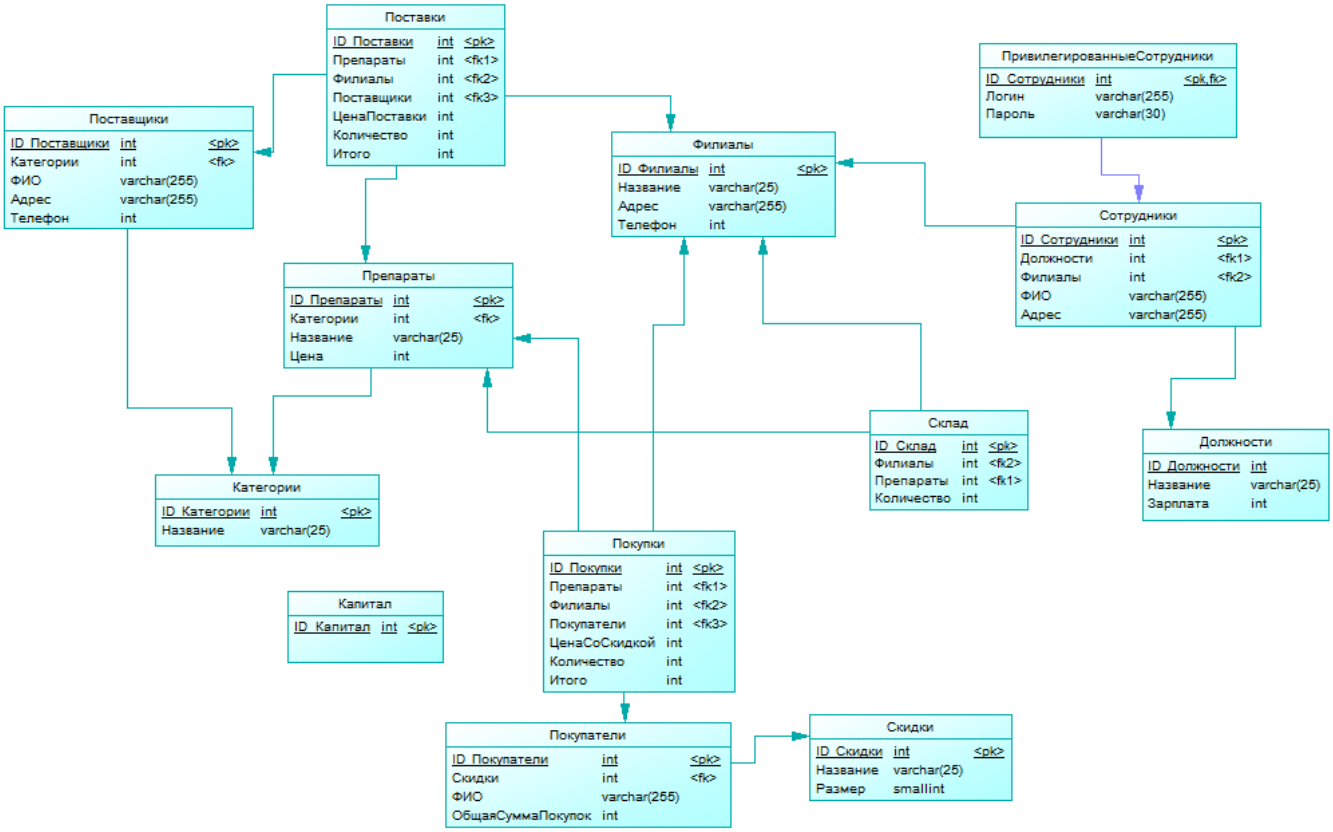

Рис. 2.27. Физическая модель данных для предметной области «Аптека»

2.3.4. ГЕНЕРАЦИЯ SQL-СКРИПТА ДЛЯ СОЗДАНИЯ БАЗЫ ДАННЫХ

SQL-скрипт создания таблиц базы и частных ограничений целостности создан автоматически с помощью Sybase PowerDesigner. Код полученного скрипта, дополненный исходными кодами разработанных в п. 2.4 ХП, представлен в Приложении 1.

2.4. РАЗРАБОТКА ХРАНИМЫХ ПРОЦЕДУР

В данном курсовом проекте в ходе разработки информационной системы были созданы следующие хранимые процедуры.

Процедуры на добавление записей в таблицы.

1) Процедура, добавляющая новый филиал.
   Входные параметры:
   - @Название – название добавляемого филиала;
   - @Адрес – адрес добавляемого филиала;
   - @Телефон – телефон добавляемого филиал.
2) Процедура, добавляющая нового покупателя.
   Входные параметры:
   - @Скидки – название скидки, устанавливаемой для покупателя;
   - @ФИО – фамилия, имя и отчество покупателя;
   - @ОбщаяСуммаПокупок – общая сумма покупок покупателя.
3) Процедура, добавляющая новую категорию.

Входные параметры:

- @Название – название категории.

4) Процедура, добавляющая новую скидку.
   Входные параметры:
   - @Название – название скидки;
   - @Размер – размер скидки в %.
5) Процедура, добавляющая нового сотрудника.
   Входные параметры:
   - @Должности – должность, которую будет занимать сотрудник;
   - @Филиалы – филиал, в котором будет работать сотрудник;
   - @ФИО – фамилия, имя, отчество сотрудник;
   - @Адрес – адрес сотрудника;
   - @Телефон – телефон сотрудника;
   - @Логин – логин сотрудника (администратора или кассира-консультанта);
   - @Пароль – пароль сотрудника (администратора или кассира-консультанта).
6) Процедура, добавляющая новую должность.
   Входные параметры:
   - @Название – название должности;
   - @Зарплата – зарплата, назначаемая сотрудникам, занимающим данную должность.
7) Процедура, добавляющая новый препарат.
   Входные параметры:
   - @Категории – название категории, которой принадлежит добавляемы препарат;
   - @Название – название препарата;
   - @Цена – цена препарата (для продажи).
8) Процедура, добавляющая поставщика.
   Входные параметры:
   - @Категории – категория препаратов, поставляемая поставщиком;
   - @ФИО – фамилия, имя и отчество поставщика;
   - @Адрес – адрес поставщика;
   - @Телефон – телефон поставщика.
9) Процедура, добавляющая новый склад.
   Входные параметры:
   - @Филиалы – филиал, на котором препарат есть в наличии;
   - @Препараты – препарат, который хранится на складе;
   - @Количество – количество препаратов на данном складе.

Процедуры на удаление записей из таблиц.

10) Процедура, удаляющая указанный филиал и его склад и увольняющая сотрудников из данного филиала.
    Входные параметры:
    - @ID_Филиалы – номер удаляемого филиала.
11) Процедура, удаляющая указанного покупателя.
    Входные параметры:
    - @ID_Покупатели – номер удаляемого покупателя.
12) Процедура, удаляющая указанную категорию и ее препараты, а также поставщиков данной категории.
    Входные параметры:

- @ID_Категории – номер удаляемой категории.

13) Процедура, удаляющая указанную скидку и покупателей, которым она назначена.
    Входные параметры:
    - @ID_Категории – номер удаляемой скидки.
14) Процедура, увольняющая указанного сотрудника и его логин-пароль (если увольняемый сотрудник – администратор или кассир-консультант).
    Входные параметры:
    - @ID_Сотрудники – номер увольняемого сотрудника.
15) Процедура, удаляющая указанную должность и сотрудников, занимающих ее.
    Входные параметры:
    - @ID_Должности – номер удаляемой должности.
16) Процедура, удаляющая указанный препарат.
    Входные параметры:
    - @ID_Препараты – номер удаляемого препарата.
17) Процедура, удаляющая указанного поставщика.
    Входные параметры:
    - @ID_Поставщики – номер удаляемого поставщика.
18) Процедура, удаляющая указанный склад.
    Входные параметры:
    - @ID_Склад – номер удаляемого склада.

Процедуры на обновление записей в таблицах.

19) Процедура, обновляющая указанный филиал.
    Входные параметры:
    - @ID_Филиалы – номер обновляемого филиала;
    - @Название – новое название филиала;
    - @Адрес – новый адрес филиала;
    - @Телефон – новый телефон филиала.
20) Процедура, обновляющая указанного покупателя.
    Входные параметры:
    - @ID_Покупатели – номер обновляемого покупателя;
    - @Скидки – новый размер скидки покупателя;
    - @ФИО – новые фамилия, имя, отчество покупателя;
    - @ОбщаяСуммаПокупок – новая сумма покупок покупателя.
21) Процедура, обновляющая указанную категорию.
    Входные параметры:
    - @ID_Категории – номер обновляемой категории;
    - @Название – новое название категории.
22) Процедура, обновляющая указанную скидку.
    Входные параметры:
    - @ID_Скидки – номер обновляемой скидки;
    - @Название – новое название скидки;
    - @Размер – новый размер скидки.
23) Процедура, обновляющая указанного сотрудника.
    Входные параметры:
    - @ID_Сотрудники – номер обновляемого сотрудника;
    - @Должности – новая должность сотрудника;
    - @Филиалы – новый филиал сотрудника;
    - @ФИО – новые фамилия, имя и отчество сотрудника;
    - @Адрес – новый адрес сотрудника;

- @Телефон – новый телефон филиала;
- @Логин – новый логин сотрудника (администратора или кассира-консультанта);
- @Пароль – новый пароль сотрудника (администратора или кассира-консультанта).

24) Процедура, обновляющая указанную должность.
Входные параметры:
- @ID_Должности – номер обновляемой должности;
- @Название – новое название должности;
- @Зарплата – новая зарплата, выплачиваемая на данной должности.

25) Процедура, обновляющая указанный препарат.
Входные параметры:
- @ID_Препараты – номер обновляемого препарата;
- @Категории – новая категория препарата;
- @Название – новое название препарата;
- @Цена – новая цена препарата.

26) Процедура, обновляющая указанного поставщика.
Входные параметры:
- @ID_Поставщики – номер обновляемого поставщика;
- @Категории – новая категория препаратов, поставляемая поставщиком;
- @ФИО – новые фамилия, имя, отчество поставщика;
- @Адрес – новый адрес поставщика;
- @Телефон – новый телефон поставщика.

27) Процедура, обновляющая указанный склад.
Входные параметры:
- @ID_Склад – номер обновляемого склада;
- @Филиалы – название нового филиала;
- @Препараты – название нового препарата;
- @Количество – новое количество препарата в филиале.

Процедуры, возвращающие необходимые данные.

28) Процедура, возвращающая число препаратов в заданном филиале на складе.
Входные параметры:
- @Препарат – необходимый препарат;
- @Филиал – необходимый филиал.

Выходные параметры:
- @Количество – число указанных препаратов в указанном филиале.

29) Процедура, возвращающая скидку указанного покупателя.
Входные параметры:
- @Покупатель – указанный покупатель.

Выходные параметры:
- @Скидка – скидка покупателя.

30) Процедура, возвращающая название филиала.
Входные параметры:
- @Филиалы – номер филиала.

Выходные параметры:
- @Название – название филиала.

31) Процедура, возвращающая имя покупателя.
Входные параметры:

- @Покупатели – номер покупателя.

Выходные параметры:

- @ФИО – фамилия, имя, отчество покупателя.

32) Процедура, возвращающая название категории.

Входные параметры:

- @Категории – номер категории.

Выходные параметры:

- @Название – название категории.

33) Процедура, возвращающая название должности.

Входные параметры:

- @Должности – номер должности.

Выходные параметры:

- @Название – название должности.

34) Процедура, возвращающая название препарата.

Входные параметры:

- @Препараты – номер препарата.

Выходные параметры:

- @Название – название препарата.

35) Процедура, возвращающая ФИО поставщика.

Входные параметры:

- @Поставщики – номер поставщика.

Выходные параметры:

- @ФИО – фамилия, имя, отчество поставщика.

36) Процедура, возвращающая цену препарата.

Входные параметры:

- @Препараты – название препарата.

Выходные параметры:

- @Цена – цена препарата.

37) Процедура, возвращающая поставочную цену препарата.

Входные параметры:

- @Препарат – название препарата.

Выходные параметры:

- @Цена – поставочная цена препарата.

38) Процедура, возвращающая размер скидки.

Входные параметры:

- @Скидки – номер скидки.

Выходные параметры:

- @Размер – размер скидки.

39) Процедура, возвращающая общую сумму покупок покупателя.

Входные параметры:

- @Логин – фамилия, имя, отчество покупателя.

Выходные параметры:

- @ОбщаяСуммаПокупок – общая сумма покупок покупателя.

40) Процедура, возвращающая всех поставщиков указанного препарата.

Входные параметры:

- @Препарат – название препарата.

41) Процедура, возвращающая суммарную зарплату всех сотрудников.

Выходные параметры:

- @Итого – суммарная зарплата всех сотрудников.

Процедуры, проверяющие правильность логина и/или пароля.

42) Процедура, проверяющая правильность логина и пароля для сотрудников (администратора и кассира-консультанта).

Входные параметры:

- @Логин – введенный логин сотрудника;
- @Пароль – введенный пароль сотрудника.

Выходные параметры:

- @Код – результат проверки.

43) Процедура, проверяющая правильность логина и пароля для покупателей.

Входные параметры:

- @Логин – введенный логин покупателя.

Выходные параметры:

- @Код – результат проверки.

3. РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ ИНФОРМАЦИОННОЙ СИСТЕМЫ

3.1. ОРГАНИЗАЦИЯ ВЗАИМОДЕЙСТВИЯ КЛИЕНТСКОЙ ПРОГРАММЫ С БД

Для доступа к базе данных использовалась технология ADO.NET. Экземпляры компонентов доступа к БД, применённые в данном проекте, приведены в таблице 2.

Таблица 2

Тип компонента и его описание	Экземпляры компонентов	Назначение
SqlConnection Предоставляет уникальный сеанс связи с источником данных SQL Server.	conn	Установка соединения и проверки правильности введенного имени пользователя и пароля
SqlCommand Представляет инструкцию Transact-SQL или хранимую процедуру, выполняемую над базой данных SQL Server.	cmd	Выполнение запросов к БД или вызов хранимых процедур.
SqlDataAdapter Представляет набор выполняемых над данными команд и подключения базы данных, которые используется для заполнения и обновления базы данных SQL Server.	dataAdapter	Извлечение и сохранение данных в БД
SqlDataReader Предоставляет возможность чтения потока строк только в прямом направлении из базы данных SQL Server	reader	Хранит данные из БД для дальнейшей работы с ними
DataGridView предоставляет настраиваемую таблицу для отображения данных.	dataGridViewBranch	Отображение данных о филиалах
	dataGridViewBuyer	Отображение данных о покупателях
	dataGridViewCategory	Отображение данных о категориях
	dataGridViewDiscount	Отображение данных о скидках
	dataGridViewEmployee	Отображение данных о сотрудниках
	dataGridViewPost	Отображение данных о должностях
	dataGridViewPreparation	Отображение данных о препаратах
	dataGridViewPurchase	Отображение данных о покупках
	dataGridViewSupplier	Отображение данных о поставщиках

	dataGridViewSupply	Отображение данных о поставках
	dataGridViewWarehouse	Отображение данных о складе

3.2. РАЗРАБОТКА ФОРМ

Программа 943_Apteka содержит 44 формы.

При запуске программы на экран выводится форма для авторизации пользователя FormMain (рисунок 3.1). Список компонентов данной формы и их назначение представлены следующим образом:

- buttonConnect – кнопка, открывающая окно подключения к серверу;
- buttonExit – кнопка выхода из программы;
- buttonLogIn – кнопка входа в программу (после выбора режима работы);
- groupBoxLogin – панель для ввода логина и пароля;
- groupBoxUser – панель для выбора режима работы;
- label1 – надпись «Логин»;
- label2 – надпись «Пароль»;
- radioButtonAdmin – радиокнопка для выбора режима «Администратор»;
- radioButtonBuyer – радиокнопка для выбора режима «Покупатель»;
- radioButtonSeller – радиокнопка для выбора режима «Продавец»;
- textBoxLogin – поле для ввода логина;
- textBoxPassword – поле для ввода пароля.

Форма представлена в файлах: FormMain.cs, FormMain.Designer.cs, FormMain.resx.

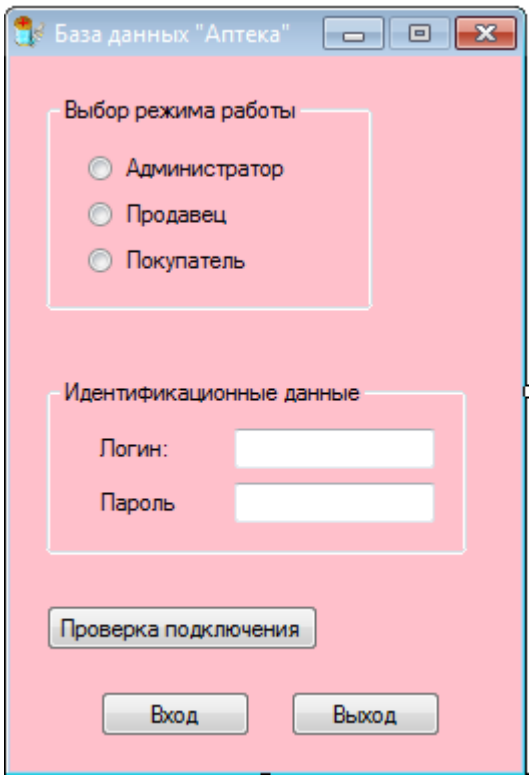

Рис. 3.1. Форма FormMain

Если вход осуществил администратор, то на экране отобразится форма FormAdminPage (рисунок 3.2). Список компонентов данной формы и их назначение представлены следующим образом:

- bindingSourceOrder – источник данных;
- buttonCapital – кнопка, открывающая окно просмотра капитала аптеки;
- buttonExit – кнопка выхода из программы;
- buttonMakeChanges – кнопка, открывающая окно изменения таблицы;
- buttonOrder – кнопка заказа товара;
- buttonReport – кнопка вывода отчета по сотрудникам;
- buttonReturn – кнопка возвращения к главному окну;
- buttonSalary – кнопка выплаты зарплаты;
- buttonViewTable – кнопка, открывающая окно просмотра таблицы;
- comboBoxBranch – выпадающий список, позволяющий выбрать филиал;
- comboBoxEditTables – выпадающий список, позволяющий выбрать таблицу для изменения;
- comboBoxPreparation – выпадающий список, позволяющий выбрать препарат;
- comboBoxSupplier – выпадающий список, позволяющий выбрать поставщика;
- comboBoxViewTables – выпадающий список, позволяющий выбрать таблицу для просмотра;
- groupBoxOrder – панель заказа препарата;
- groupBoxTables – панель изменения таблиц;
- groupBoxViewTables – панель просмотра таблиц;
- label1 – надпись «Количество: »;
- label2 – надпись «Цена: »;
- label3 – надпись «Итого: »;
- radioButtonDelete – радиокнопка удаления записи из таблицы;
- radioButtonInsertInto – радиокнопка добавления записи в таблицу;
- radioButtonUpdate – радиокнопка обновления записи в таблице;
- textBoxCount – поле для ввода количества препаратов;
- textBoxPrice – поле для отображения цены поставки препарата;
- textBoxTotal – поле для отображения стоимости заказа.

Форма представлена в файлах: FormAdminPage.cs, FormAdminPage.Designer.cs, FormAdminPage.resx.

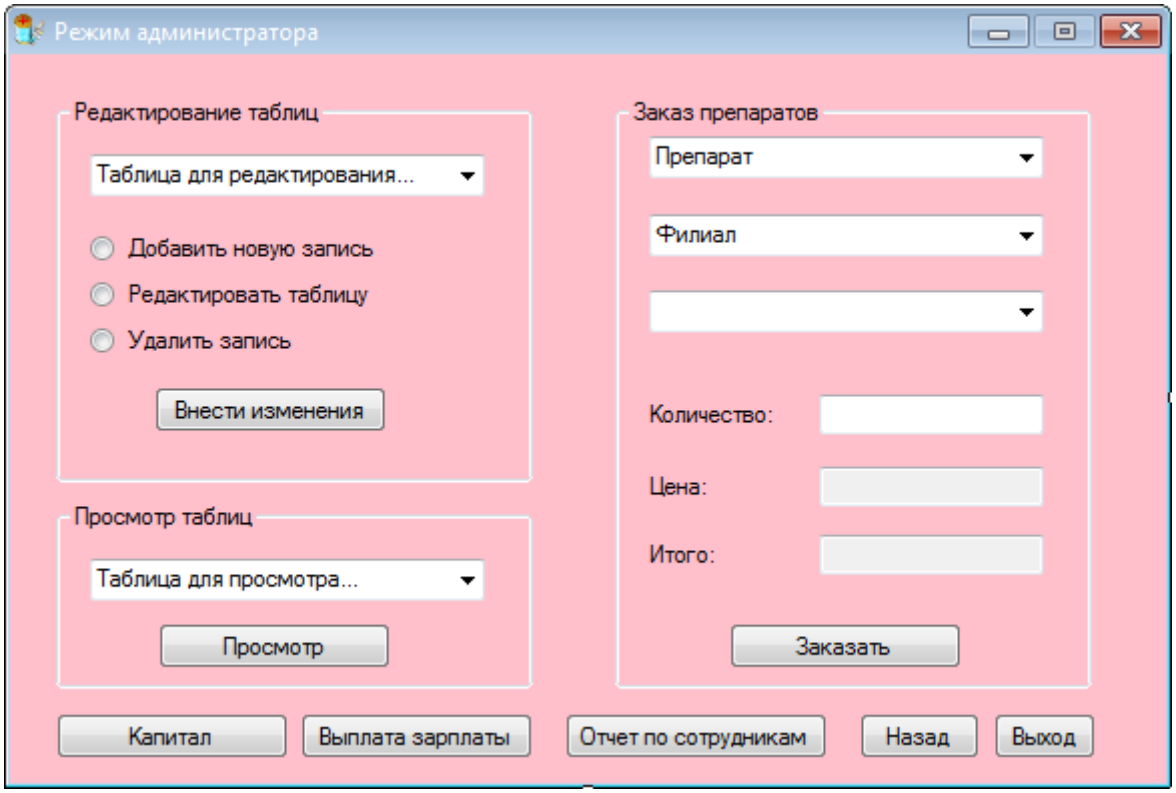

Рис. 3.2. Форма FormAdminPage

При выборе таблицы из выпадающего списка comboBoxEditTables, режима добавления записи (кнопка radioButtonInsertInto) и нажатии на кнопку buttonMakeChanges отобразится одна из форм FormNewBranch, FormNewBuyer, FormNewCategory, FormNewDiscount, FormNewEmployee, FormNewPost, FormNewPreparation, FormNewSupplier, FormNewWarehouse, позволяющие добавить новый филиал, покупателя, категорию, скидку, сотрудника, должность, препарат, поставщика и склад соответственно. Все эти формы аналогичны друг другу, поэтому рассмотрим одну из них, например, FormNewWarehouse (рисунок 3.3). Список компонентов данной формы и их назначение представлены следующим образом:

- buttonAddWarehouse – кнопка, добавляющая новый склад;
- buttonExit – кнопка закрытия формы FormNewWarehouse;
- comboBoxBranch – выпадающий список для выбора филиала;
- comboBoxPreparation – выпадающий список для выбора препарата;
- label3 – надпись «Количество: »;
- textBoxCount – поле для ввода количества препаратов в филиале.

Форма представлена в файлах: FormNewWarehouse.cs, FormNewWarehouse.Designer.cs, FormNewWarehouse.resx.

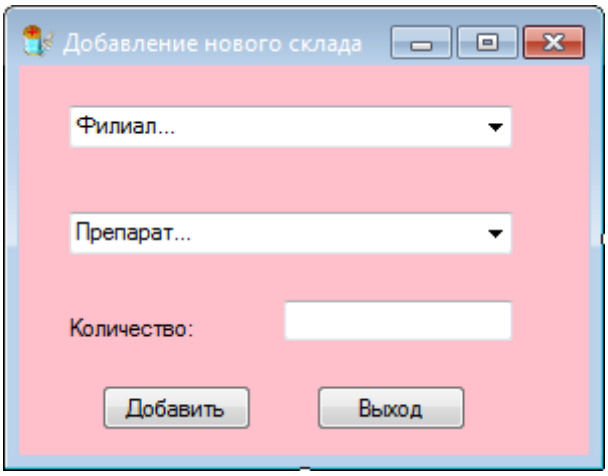

Рис. 3.3. Форма FormNewWarehouse

Вернемся к форме FormAdminPage и на той же панели groupBoxTable выберем радиокнопку radioButtonUpdate, позволяющую редактировать таблицы. В этом случае при нажатии на кнопку buttonMakeChanges откроется одна из форм FormEditBranch, FormEditBuyer, FormEditCategory, FormEditDiscount, FormEditEmployee, FormEditPost, FormEditPreparation, FormEditSupplier, FormEditWarehouse. Все эти формы аналогичны друг другу, поэтому рассмотрим одну из них, например, FormEditPreparation (рисунок 3.4). Список компонентов данной формы и их назначение представлены следующим образом:

- buttonEditPreparation – кнопка, изменяющая информацию о выбранном препарате;
- buttonExit – кнопка закрытия формы FormEditPreparation;
- comboBoxCategory – выпадающий список для выбора категории;
- dataGridViewPreparation – таблица отображения данных о препаратах;
- ID_Препараты – столбец с номером препарата в dataGridViewPreparation;
- label1 – надпись «Категории:»;
- label2 – надпись «Название:»;
- label3 – надпись «ID_Препараты:»;
- label4 – надпись «Выделите нужную строку:»;
- label5 – надпись «Введите нужные значения столбцов:»;
- label6 – надпись «Цена:»;
- textBoxID – поле для отображения номера препарата;
- textBoxName – поле для отображения и ввода названия препарата;
- textBoxPrice – поле для отображения и ввода цены препарата;
- Категории – столбец с категорией препарата в dataGridViewPreparation;
- Название – столбец с названием препарата в dataGridViewPreparation;
- Цена – столбец с ценой препарата в dataGridViewPreparation.

Форма представлена в файлах: FormEditPreparation.cs, FormEditPreparation.Designer.cs, FormEditPreparation.resx.

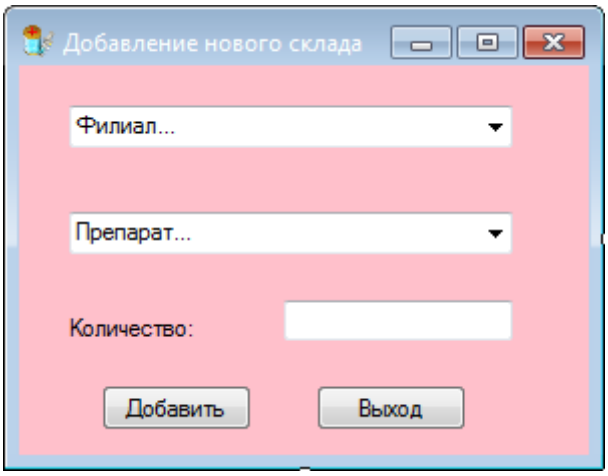

Рис. 3.4. Форма FormEditPreparation

Вернемся к форме FormAdminPage и на той же панели groupBoxTable выберем радиокнопку radioButtonDelete, позволяющую удалять записи из таблиц. В этом случае при нажатии на кнопку buttonMakeChanges откроется одна из форм FormDeleteBranch, FormDeleteBuyer, FormDeleteCategory, FormDeleteDiscount, FormDeleteEmployee, FormDeletePost, FormDeletePreparation, FormDeleteSupplier, FormDeleteWarehouse. Все эти формы аналогичны друг другу, поэтому рассмотрим одну из них, например, FormDeletePost (рисунок 3.5). Список компонентов данной формы и их назначение представлены следующим образом:

- buttonDeletePost – кнопка, удаляющая информацию о выбранной должности;
- buttonExit – кнопка закрытия формы FormDeletePost;
- dataGridViewPost – таблица отображения данных о должностях;
- ID_Должности – столбец с номером должности в dataGridViewPost;
- label1 – надпись «Внимание! Все сотрудники, занимающие удаляемую должность, будут уволены!»;
- label4 – надпись «Выделите удаляемую строку:»;
- Зарплата – столбец с зарплатой на должности в dataGridViewPost;
- Название – столбец с названием должности в dataGridViewPost.

Форма представлена в файлах: FormDeletePost.cs, FormDeletePost.Designer.cs, FormDeletePost.resx.

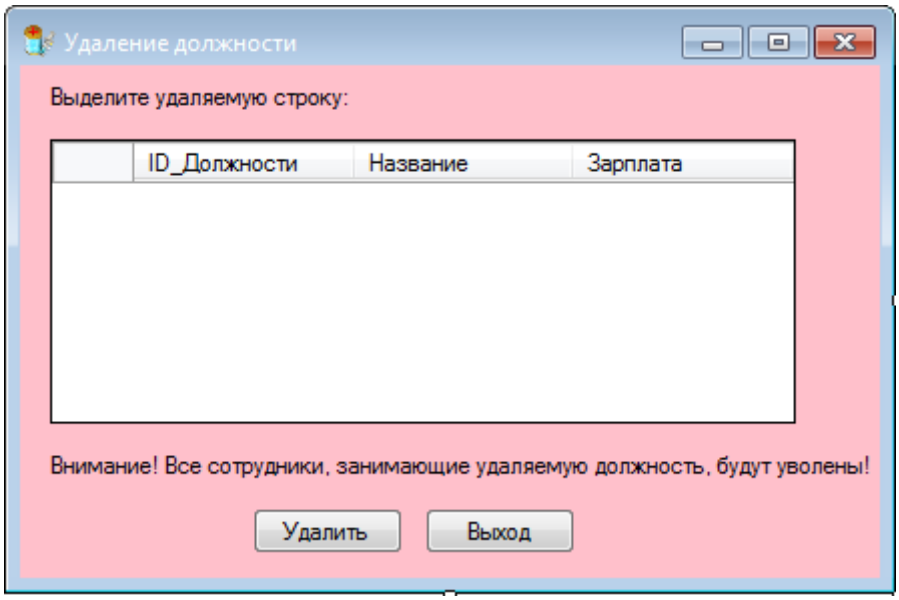

Рис. 3.5. Форма FormDeletePost

Вернемся к форме FormAdminPage, на панели groupBoxViewTables выберем таблицу из списка comboBoxViewTables, нажмем на кнопку buttonViewTable. Откроется одна из форм FormViewBranch, FormViewBuyer, FormViewCategory, FormViewDiscount, FormViewEmployee, FormViewPost, FormViewPreparation, FormViewPurchase, FormViewSupplier, FormViewSupply, FormViewWarehouse. Все эти формы аналогичны друг другу, поэтому рассмотрим одну из них, например, FormViewCategory (рисунок 3.6). Список компонентов данной формы и их назначение представлены следующим образом:

- buttonExit – кнопка закрытия формы FormViewCategory;
- dataGridViewCategory – таблица отображения данных о препаратах;
- ID_Категории – столбец с номером препарата в dataGridViewCategory;
- label1 – надпись «Название»;
- label3 – надпись «ID_Категории»;
- label4 – надпись «Выделите нужную строку:»;
- textBoxID – поле для отображения номера препарата;
- textBoxName – поле для отображения названия категории;
- Название – столбец с названием препарата в dataGridViewCategory.

Форма представлена в файлах: FormViewCategory.cs, FormViewCategory.Designer.cs, FormViewCategory.resx.

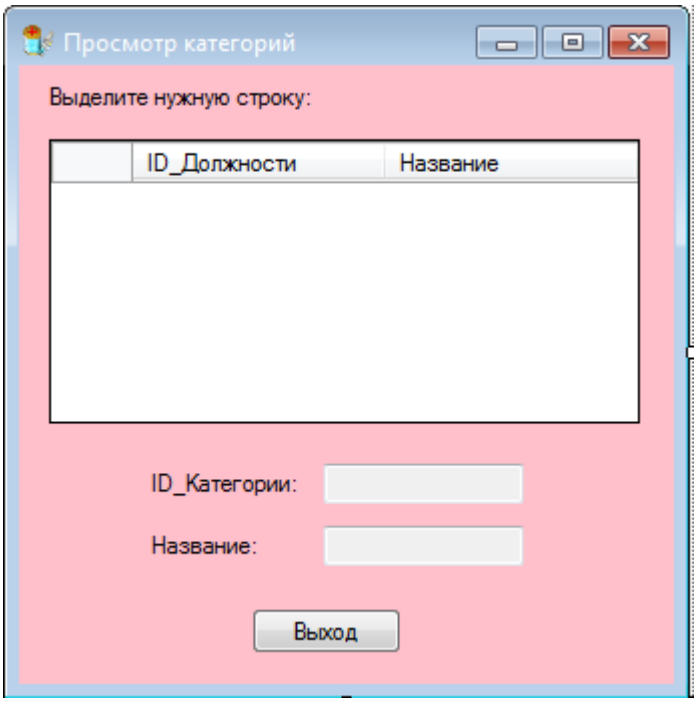

Рис. 3.6. Форма FormViewCategory

Вернемся к форме FormAdminPage, нажмем на кнопку buttonReport. Откроется форма FormReportEmployee (рисунок 3.7). Список компонентов данной формы и их назначение представлены следующим образом:

- crystalReportViewer – компонент просмотра отчетов.

Форма представлена в файлах: FormReportEmployee.cs, FormReportEmployee.Designer.cs, FormReportEmployee.resx.

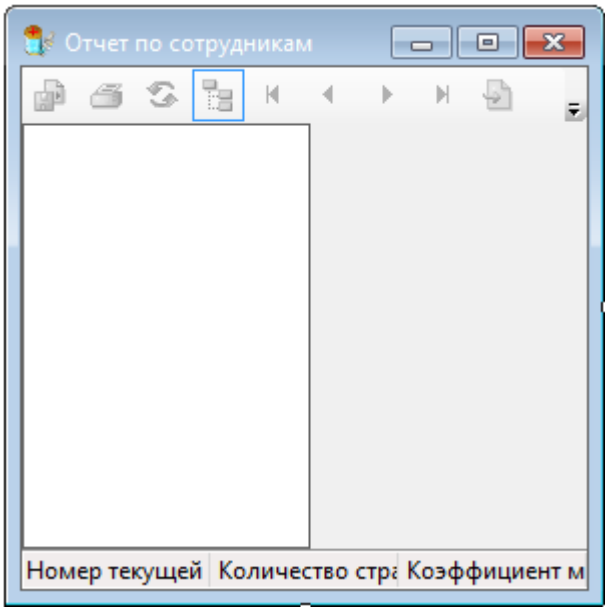

Рис. 3.7. Форма FormReportEmployee

Вернемся к форме FormMain. Если вход осуществил продавец, то на экране отобразится форма FormSellerPage (рисунок 3.8). Список компонентов данной формы и их назначение представлены следующим образом:

- buttonAddBuyer – кнопка, позволяющая добавить нового покупателя;
- buttonExit – кнопка выхода из программы;
- buttonReturn – кнопка возвращения к главному окну;
- buttonSale – кнопка, позволяющая продать препарат;
- buttonViewTable – кнопка для просмотра таблиц;
- comboBoxBranch – выпадающий список для выбора филиала;
- comboBoxBuyer – выпадающий список для выбора покупателя;
- comboBoxDiscount – выпадающий список для выбора скидки;
- comboBoxPreparation – выпадающий список для выбора препарата;
- comboBoxViewTables – выпадающий список для выбора таблицы для просмотра;
- groupBoxBuyers – панель для добавления нового покупателя;
- groupBoxOrder – панель покупки;
- groupBoxViewTables – панель для просмотра таблиц;
- label1 – надпись «Количество:»;
- label2 – надпись «Цена:»;
- label3 – надпись «Итого:»;
- label4 – надпись «Общая сумма покупок:»;
- label5 – надпись «ФИО (полностью)»;
- textBoxCount – поле для ввода количества препаратов;
- textBoxFIO – поле для ввода фамилии, имени, отчества;
- textBoxPrice – поле для отображения цены;
- textBoxSumBuy – поле для ввода общей суммы покупок;
- textBoxTotal – поле для отображения суммарной стоимости покупки.

Форма представлена в файлах: FormSellerPage.cs, FormSellerPage.Designer.cs, FormSellerPage.resx.

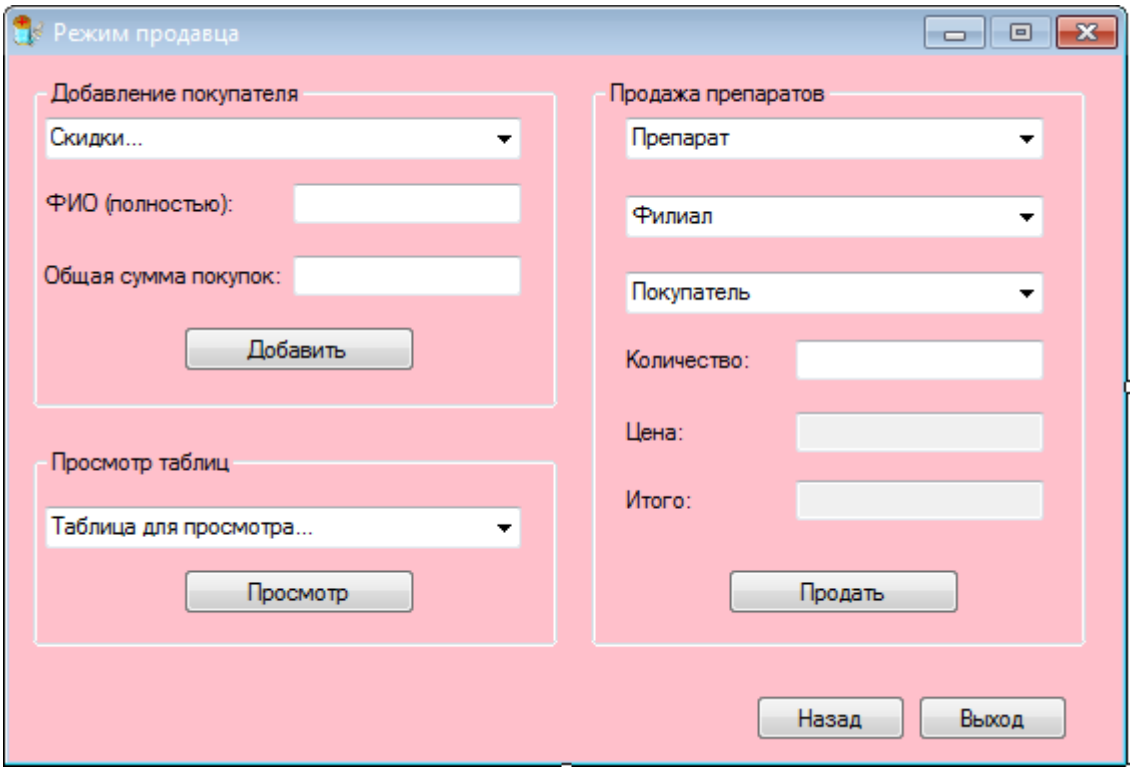

Рис. 3.8. Форма FormSellerPage

Вернемся к форме FormMain. Если вход осуществил покупатель, то на экране отобразится форма FormBuyerPage (рисунок 3.9). Список компонентов данной формы и их назначение представлены следующим образом:

- buttonExit – кнопка выхода из программы;
- buttonReturn – кнопка возврата к главному окну;
- buttonViewTable – кнопка просмотра таблиц;
- groupBoxViewTables – панель просмотра таблиц;
- label1 – надпись «Вы зашли как»;
- lableLogin – надпись «незарегистрированный пользователь»;
- radioButtonBranch – радиокнопка выбора просмотра филиалов;
- radioButtonPreparation – радиокнопка выбора просмотра препаратов;
- radioButtonPurchase – радиокнопка выбора просмотра покупок;
- radioButtonWarehouse – радиокнопка выбора просмотра наличия препаратов в филиалах;
- radioButtonYourself – радиокнопка выбора просмотра информации о себе.

Форма представлена в файлах: FormBuyerPage.cs, FormBuyerPage.Designer.cs, FormBuyerPage.resx.

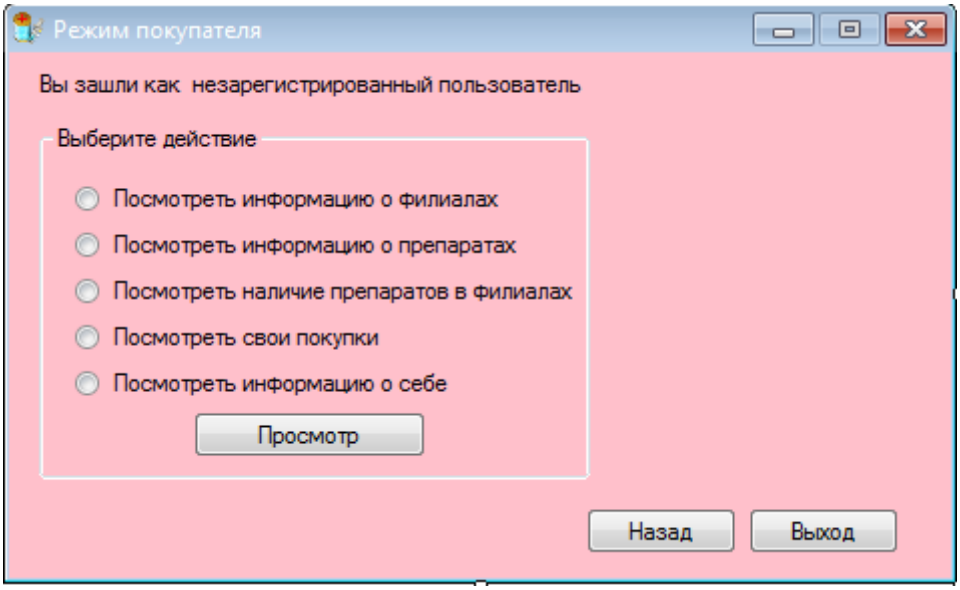

Рис. 3.9. Форма FormBuyerPage

3.3. РАЗРАБОТКА ОТЧЕТОВ

В рамках данной информационной системы был разработан отчет по сотрудникам аптеки с помощью стандартного инструмента для создания отчётов в среде Visual Studio .NET – Crystal Report .NET. Для этого был создан источник данных DataSetReportEmployees. Данные в него заносятся во время выполнения программы из таблиц БД.

В отчёте по сотрудникам аптеки содержится следующая информация:

1) ID;
2) Должность;
3) Филиал;
4) ФИО;
5) Адрес;
6) Телефон.

24.04.2012

ID	Должность	Филиал	ФИО	Адрес	Телефон
1	Директор	Здоровье №1	Иванов Иван Иванович	Перомайский пр-т, 40	767 475
2	Администратор	Здоровье №1	Кузьмин Михаил Павлович	Стройкова 94	949 494
3	Кассир-консультант	Здоровье №1	Кострова Светлана Ильинична	Первомайский пр-т 35	899 878
4	Кассир-консультант	Здоровье №1	Бондарчук Лариса Михайловна	Первомайский пр-т 33	899 869
5	Уборщица	Здоровье №1	Орлова Надежда Константиновна	Первомайский пр-т 29	899 753
8	Кассир-консультант	Здоровье №2	Спиваков Даниил Алексеевич	Солнечная 11	358 974
9	Кассир-консультант	Здоровье №2	Орлов Никадим Кондратович	Солнечная 17	359 785
12	Уборщица	Здоровье №2	Морозова Светлана Сергеевна	Солнечная 30	456 982
17	Кассир-консультант	Здоровье №3	Зазаева Алина Михайловна	Семашко 17	764 412
18	Кассир-консультант	Здоровье №3	Осипов Роман Олегович	Семашко 19	764 967
20	Уборщица	Здоровье №3	Михеева Антонина Степановна	Семашко 20	742 583
23	Кассир-консультант	Здоровье №4	Калинкин Руслан Олегович	Московское шоссе 3	355 696
24	Кассир-консультант	Здоровье №4	Анцупов Алексей Алексеевич	Московское шоссе 35	353 356
25	Кассир-консультант	Здоровье №4	Лапушкин Виктор Борисович	Московское шоссе 33	353 589
27	Уборщица	Здоровье №4	Радугина Людмила Романовна	Московское шоссе 8	353 247
31	Кассир-консультант	Здоровье №5	Лунько Анжелика Олеговна	Зубкова 16	259 631
32	Кассир-консультант	Здоровье №5	Никонорова Людмила Васильевна	Зубкова 35	259 745
35	Уборщица	Здоровье №5	Рамса Галина Эдуардовна	Зубкова 22	478 256

Рис. 3.10. Отчет по сотрудникам аптеки

3.4. РАЗРАБОТКА СЦЕНАРИЯ ИНСТАЛЛЯЦИИ КЛИЕНТСКОЙ ПРОГРАММЫ

Установщик программы написан в Microsoft Visual Studio 2005. Исходный код проекта установщика представлен в приложении 3. Для создания базы данных на сервере необходимо запустить скрипт FootballShipCreateBase.sql (приложение 1).

3.5. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Чтобы запустить программу, выполните двойной щелчок левой клавишей мыши по значку 943_Apteka.exe. На экране появится главное окно программы – форма авторизации пользователей:

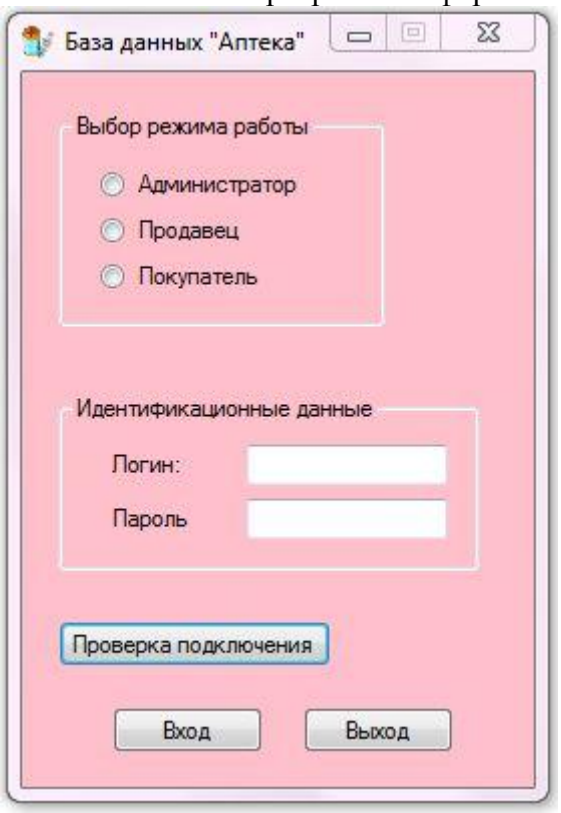

Рис. 3.11. Главное окно программы

Для начала необходимо выполнить подключение к базе данных, нажав на кнопку «Проверка подключения»:

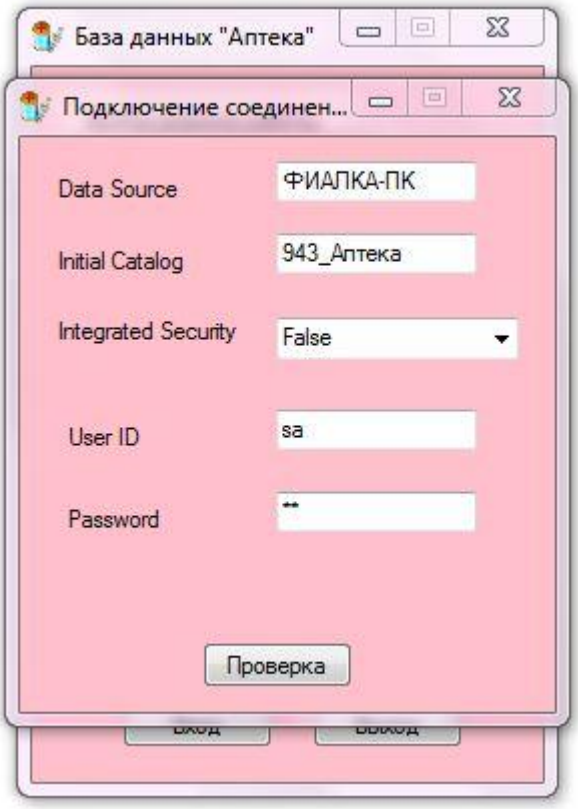

Рис. 3.12. Подключение базы данных

Введем данные и щелкнем на кнопке «Проверка». В случае неверных данных программа выдаст предупреждение:

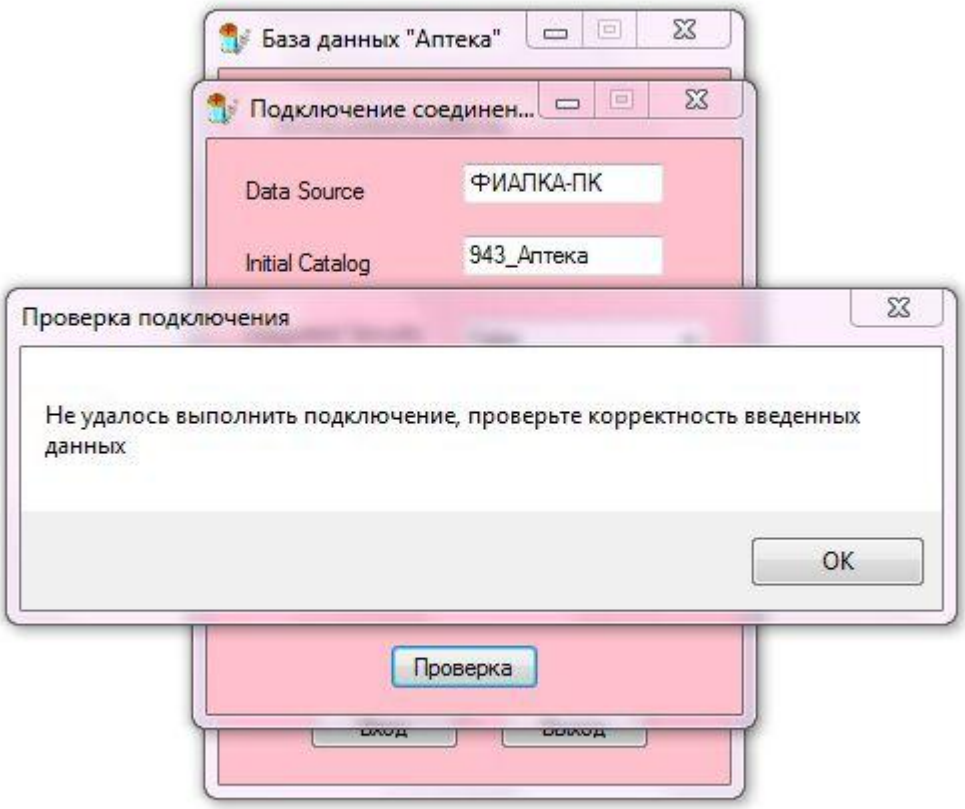

Рис. 3.13. Ошибка подключения к базе данных

В случае корректных данных программа выдаст сообщение об успехе операции:

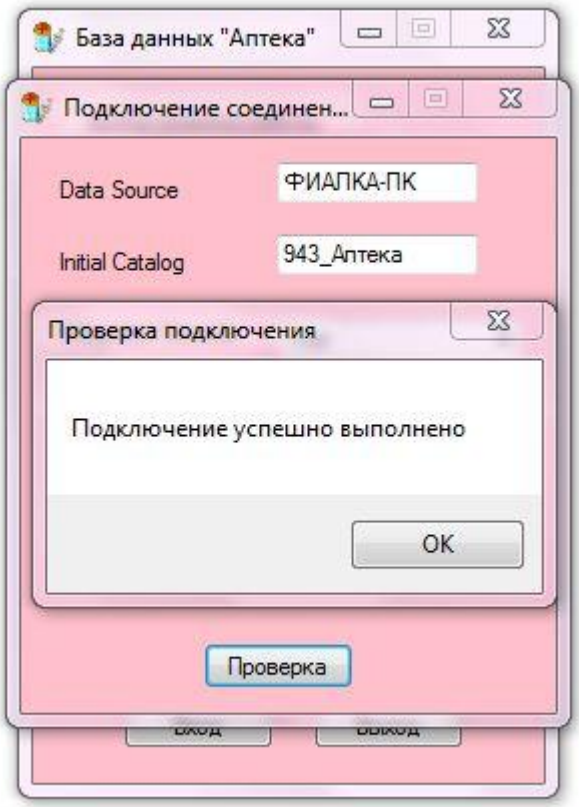

Рис. 3.14. Успешное подключение к базе данных

Закрываем окно подключения и выбираем один из режимов: администратора, продавца или покупателя. В первых двух режимах необходимо ввести логин и пароль. Выберем режим администратора и попробуем зайти в систему без авторизации или с неверной парой логин-пароль, нажав на кнопку «Вход». Программа выдаст следующее предупреждение:

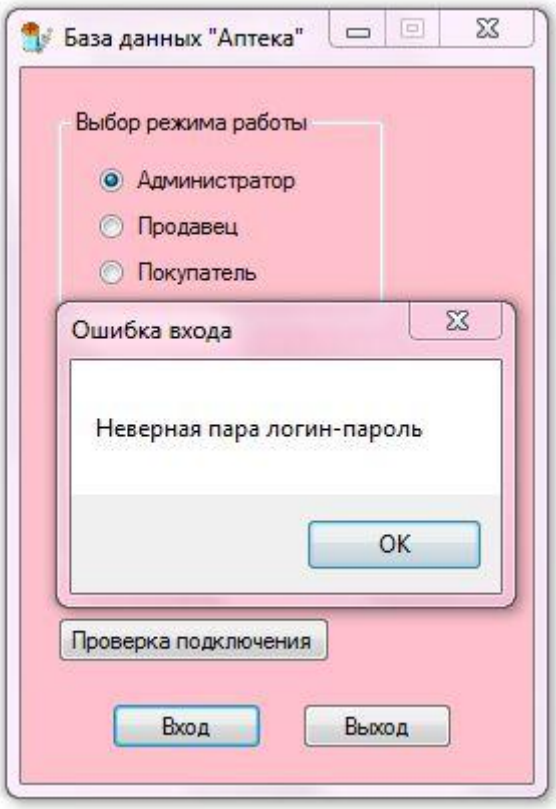

Рис. 3.15. Ошибка входа в программу

Введем логин: Кузьмин и пароль: Михаил. Нажмем на кнопку «Вход». На этот раз вход в программу успешно осуществлен, перед нами окно режима администратора:

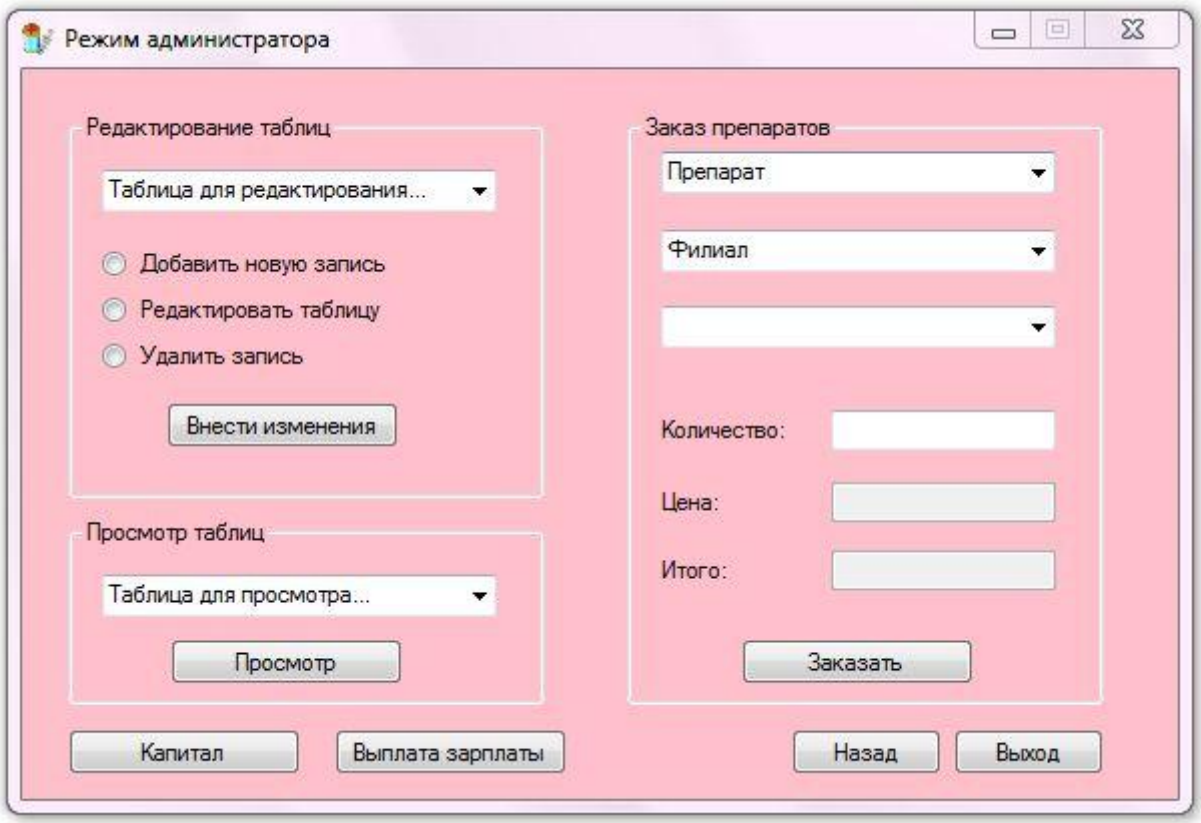

Рис. 3.16. Режим администратора

Добавим новую запись в одну из таблиц. Для этого необходимо выбрать таблицу для редактирования из соответствующего выпадающего списка, нажать на радиокнопку «Добавить новую запись» и щелкнуть на кнопке «Внести изменения». Если забыть выбрать таблицу, программа выдаст предупреждение:

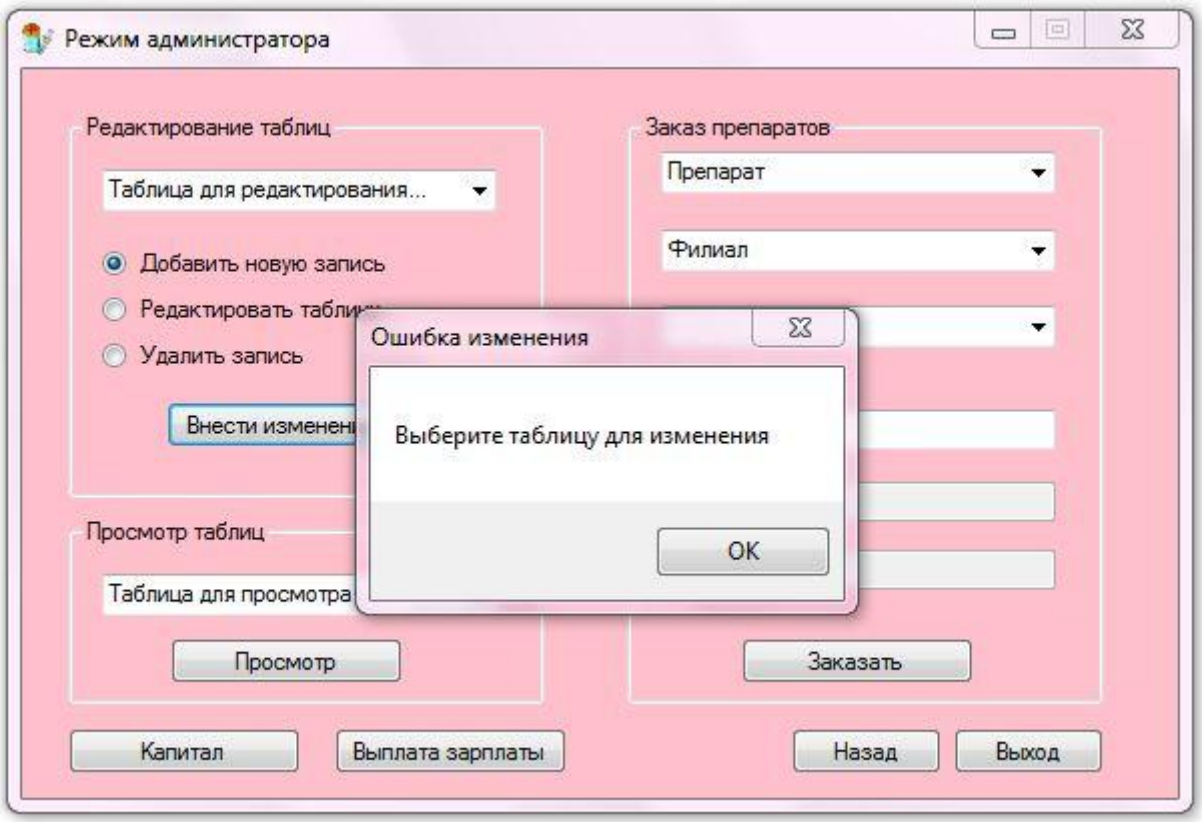

Рис. 3.17. Ошибка изменения таблицы

Попробуем добавить новый филиал. Для этого необходимо заполнить все поля в следующем окне:

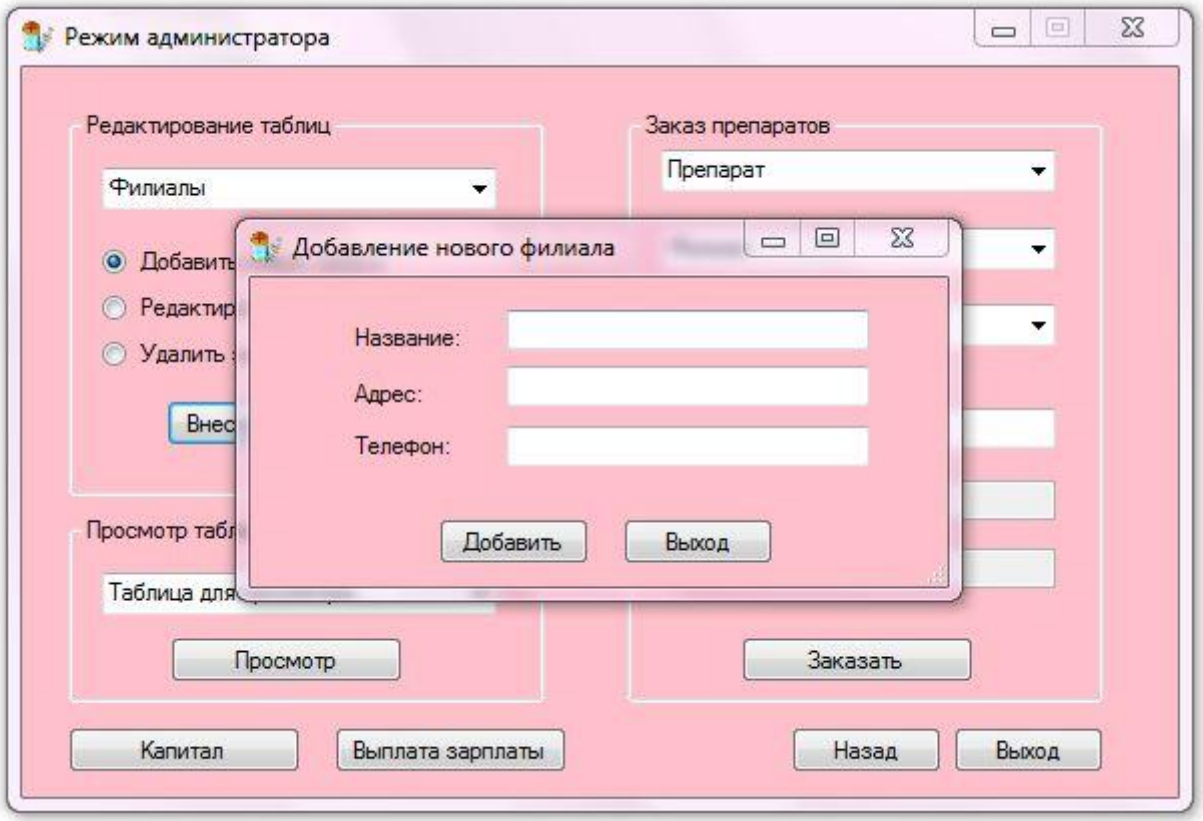

Рис. 3.18. Добавление нового филиала

Если забыть заполнить какое-нибудь поле, то программа выдаст уведомление об ошибке:

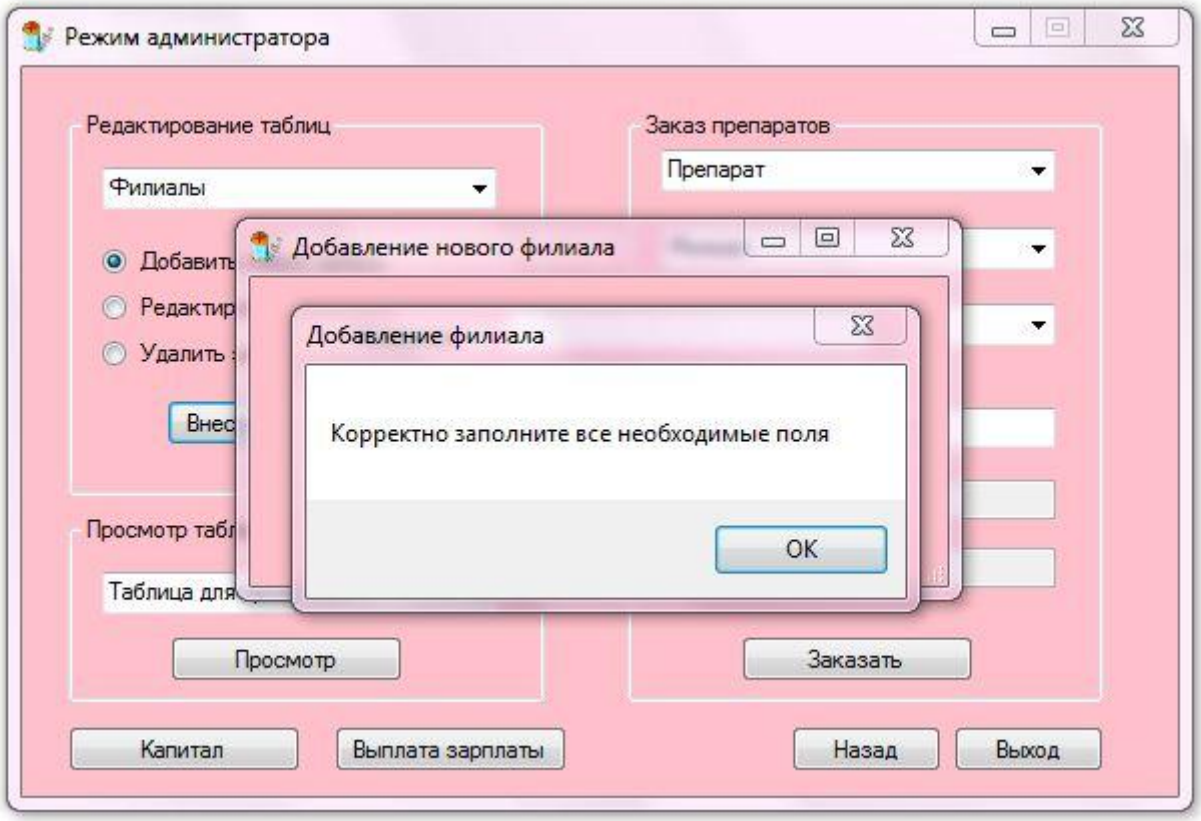

Рис. 3.19. Ошибка добавления филиала

Заполним все поля и щелкнем на кнопке «Добавить»:

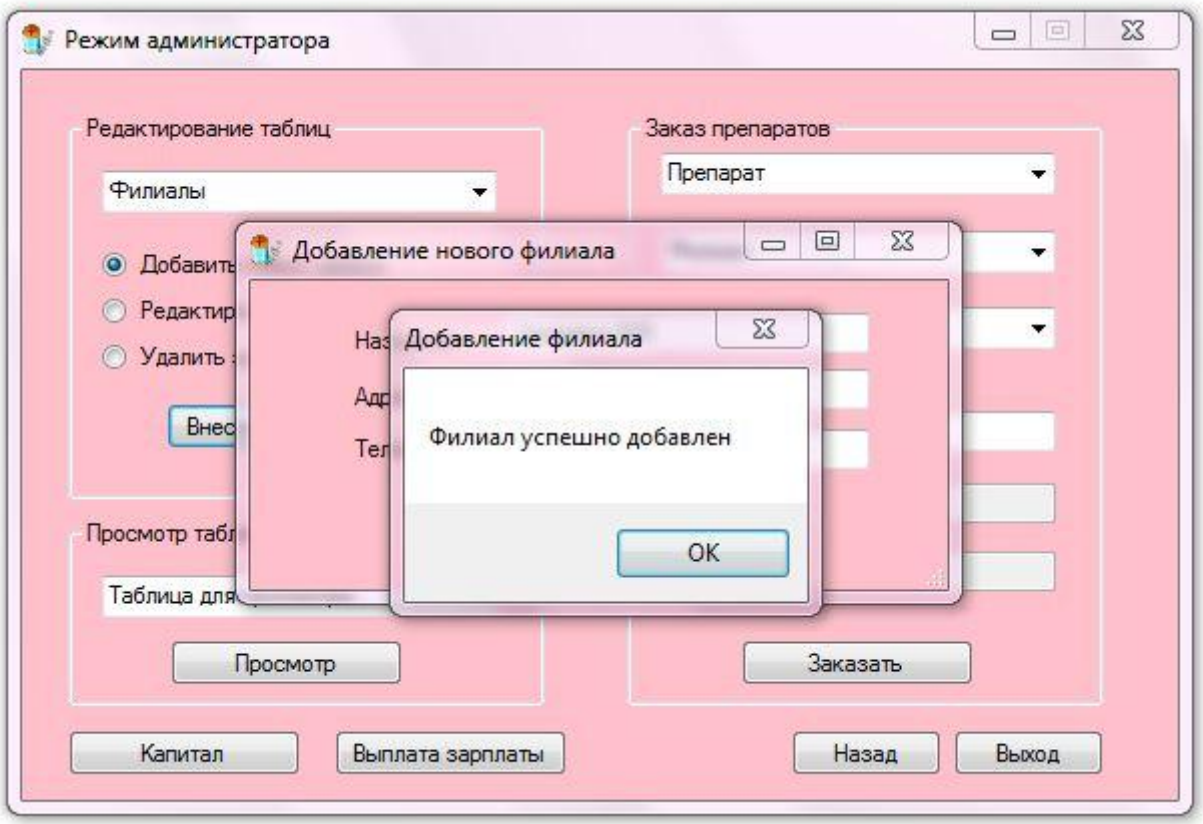

Рис. 3.20. Успешное добавление филиала

Теперь изменим название препарата. Для этого выберем таблицу «Препараты», пункт «Редактировать таблицу», нажмем кнопку «Внести изменения». Откроется следующе диалоговое окно:

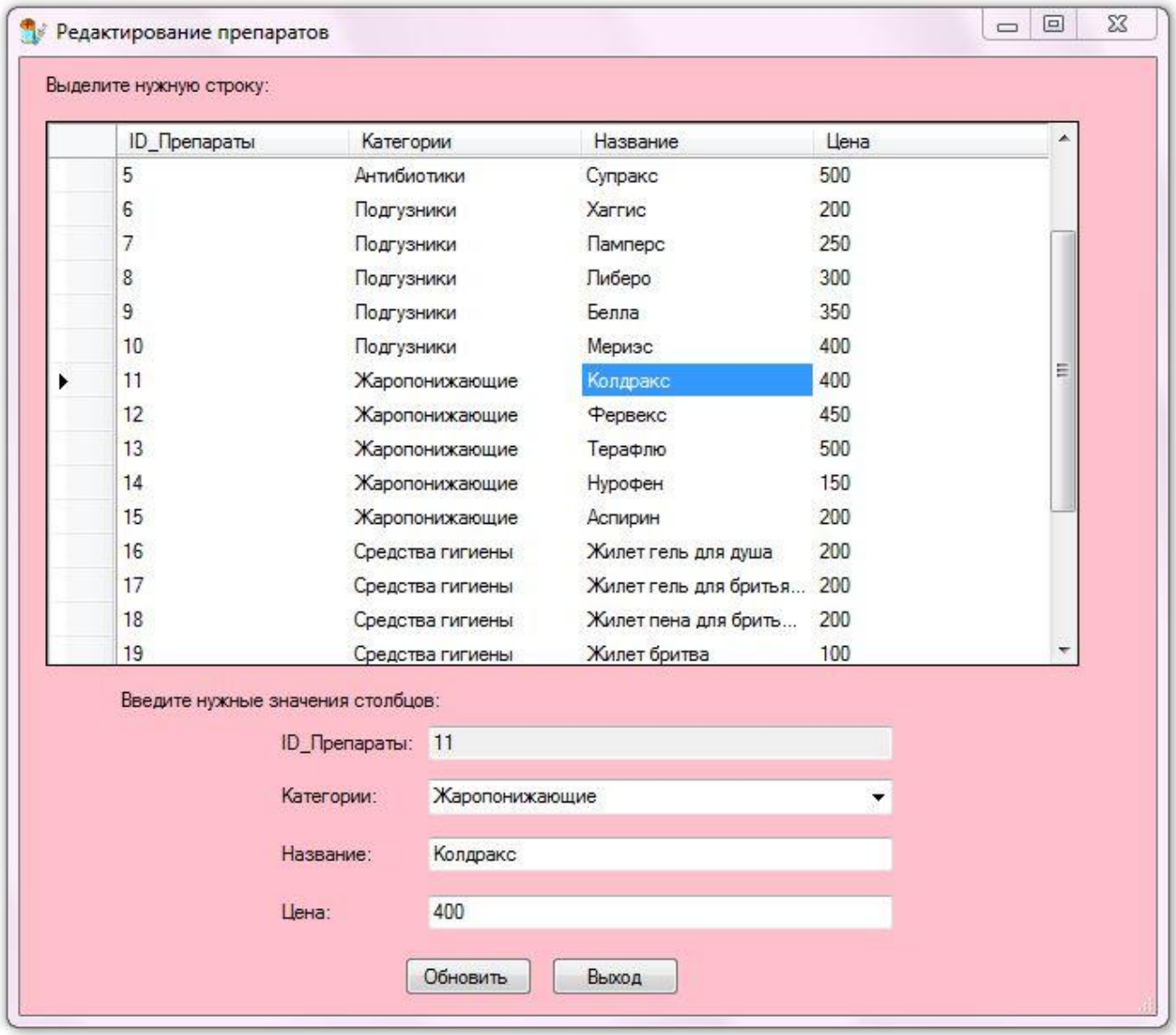

Рис. 3.21. Редактирование препаратов

Изменим в поле «Название» «Колдракс» на «Колдрекс» и нажмем кнопку «Обновить»:

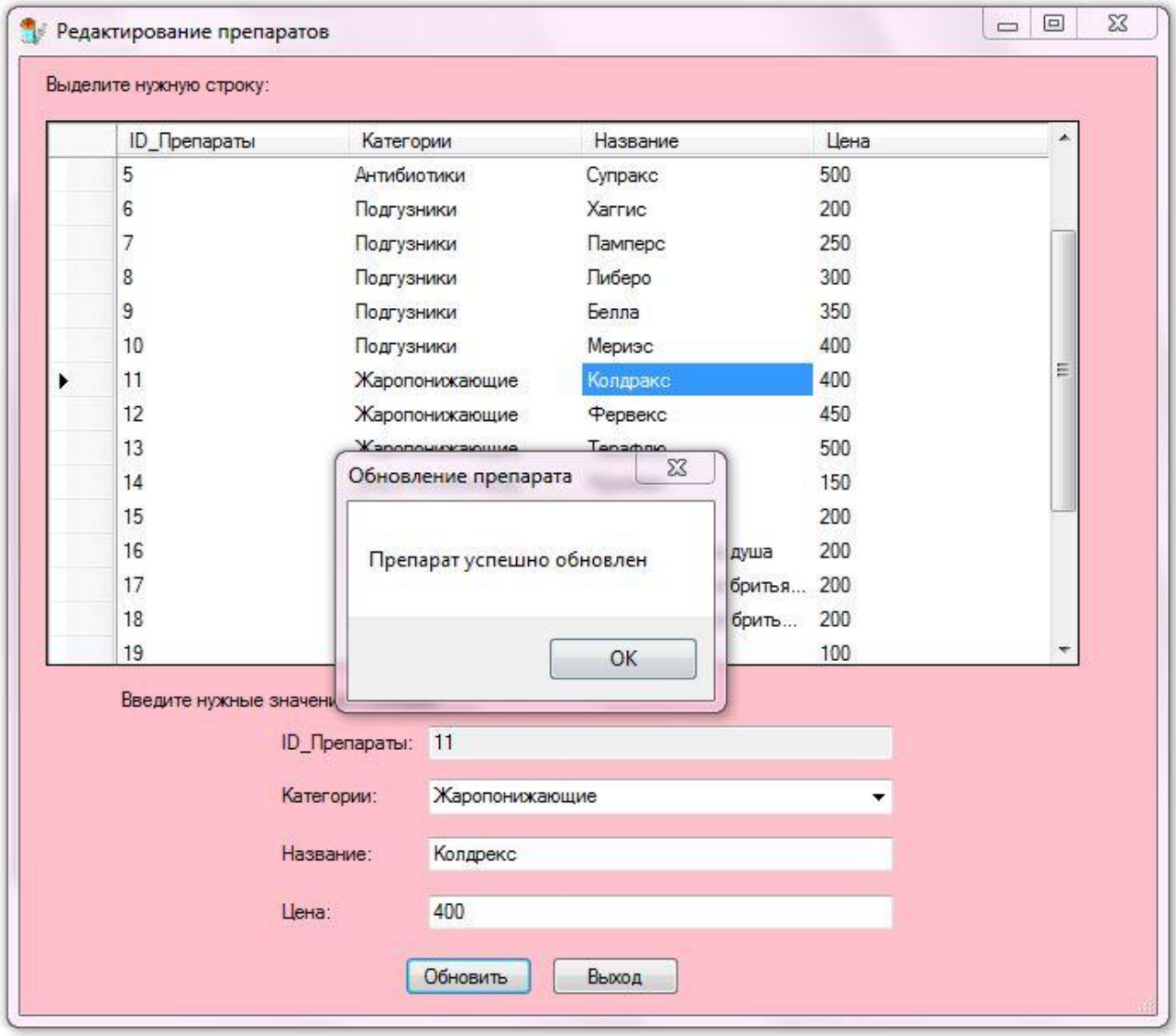

Рис. 3.22. Успешное обновление препарата

Удалим одну из скидок. В выпадающем списке выбираем таблицу «Скидки», щелкаем по радиокнопке «Удалить запись» и нажимаем на кнопку «Внести изменения». Выделяем последнюю строку таблицы и нажимаем «Удалить»:

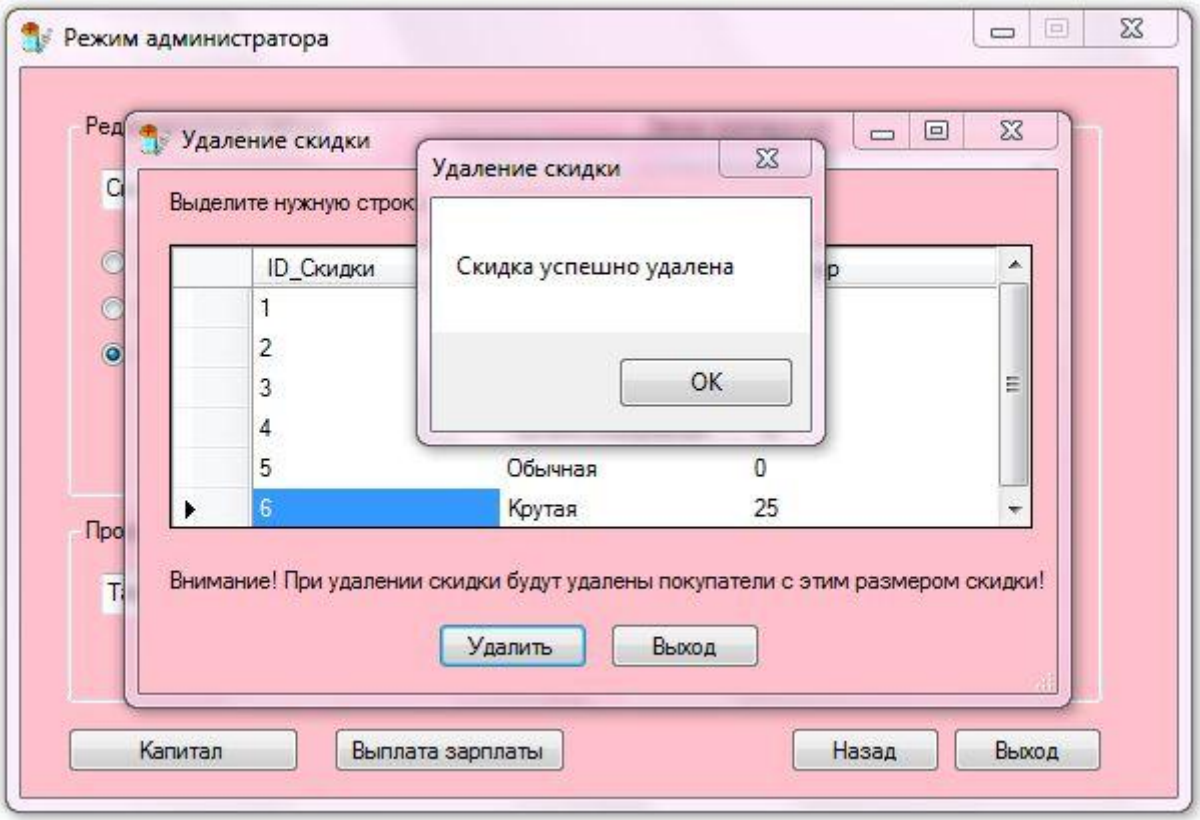

Рис. 3.23. Удаление скидки

Просмотрим информацию о поставщиках. Для этого на панели «Просмотр таблиц» в выпадающем списке выберем таблицу «Поставщики» и щелкнем по кнопке «Просмотр»:

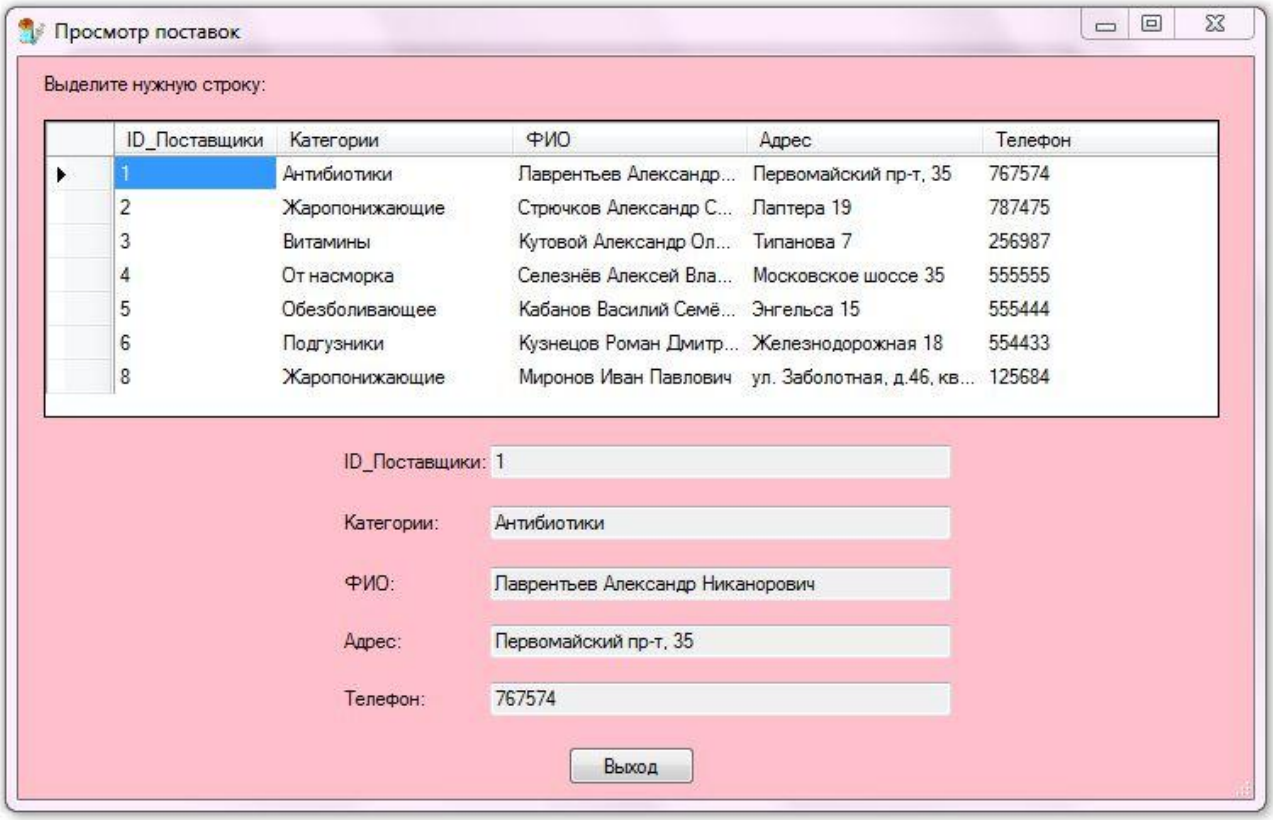

Рис. 3.24. Просмотр поставщиков

При выборе различных строк в полях под таблицей отображается информация о выделенном поставщике. Редактировать эту информацию в данном режиме невозможно.

Нажмем на кнопку «Выход» и сделаем заказ. Для этого на панели «Заказ препаратов» необходимо выбрать препарат и филиал из выпадающих списков. После выбора препарата появится возможность выбрать одного из поставщиков, которые его предоставляют, а также в соответствующем поле отобразится цена закупки препарата. Необходимо заполнить поле «Количество», указав число недостающих препаратов. После этого в поле «Итого» отобразится стоимость всей поставки. Поля «Цена» и «Итого» запрещены для редактирования. Если забыть ввести какое-нибудь данное или сделать это некорректно, программа сообщит об этом:

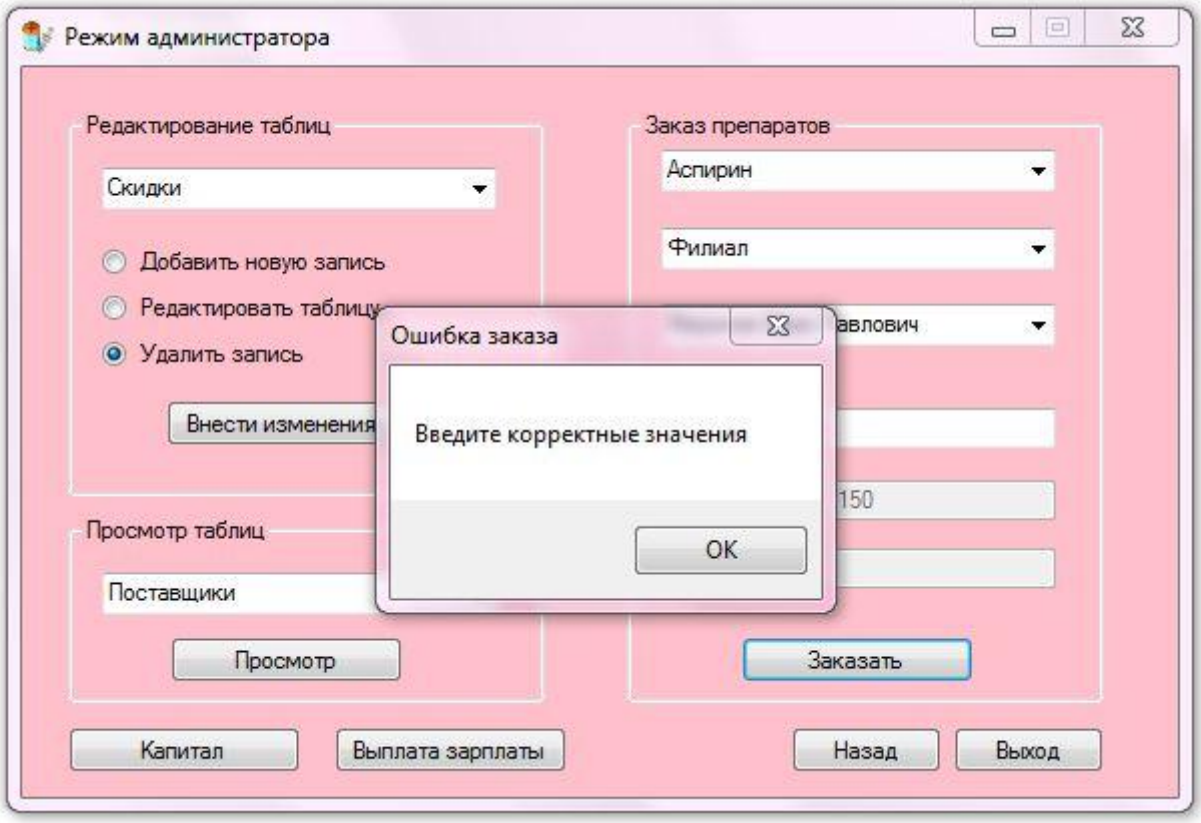

Рис. 3.25. Ошибка заказа

А теперь введем все необходимые значения и нажмем на кнопку «Заказать»:

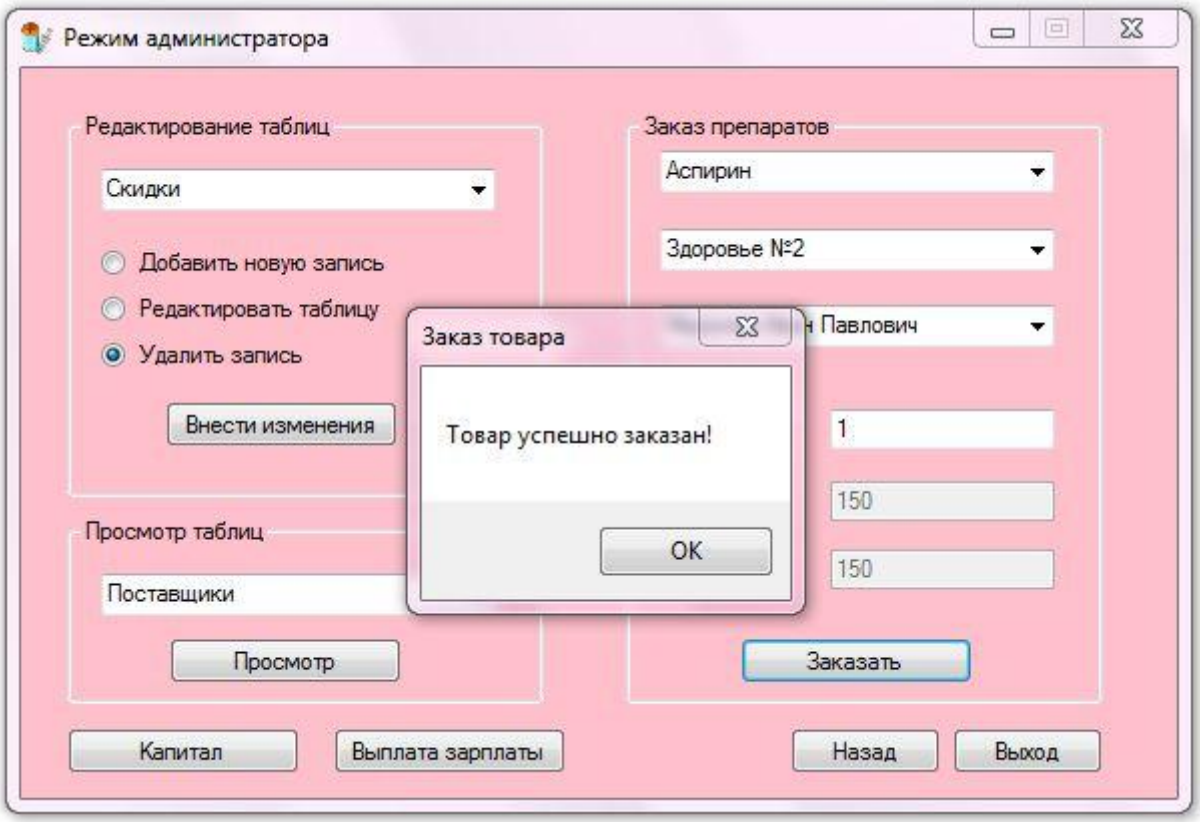

Рис. 3.26. Успешный заказ товара

С помощью нажатия кнопки «Капитал» можно посмотреть сумму денег, которой владеет сеть аптек:

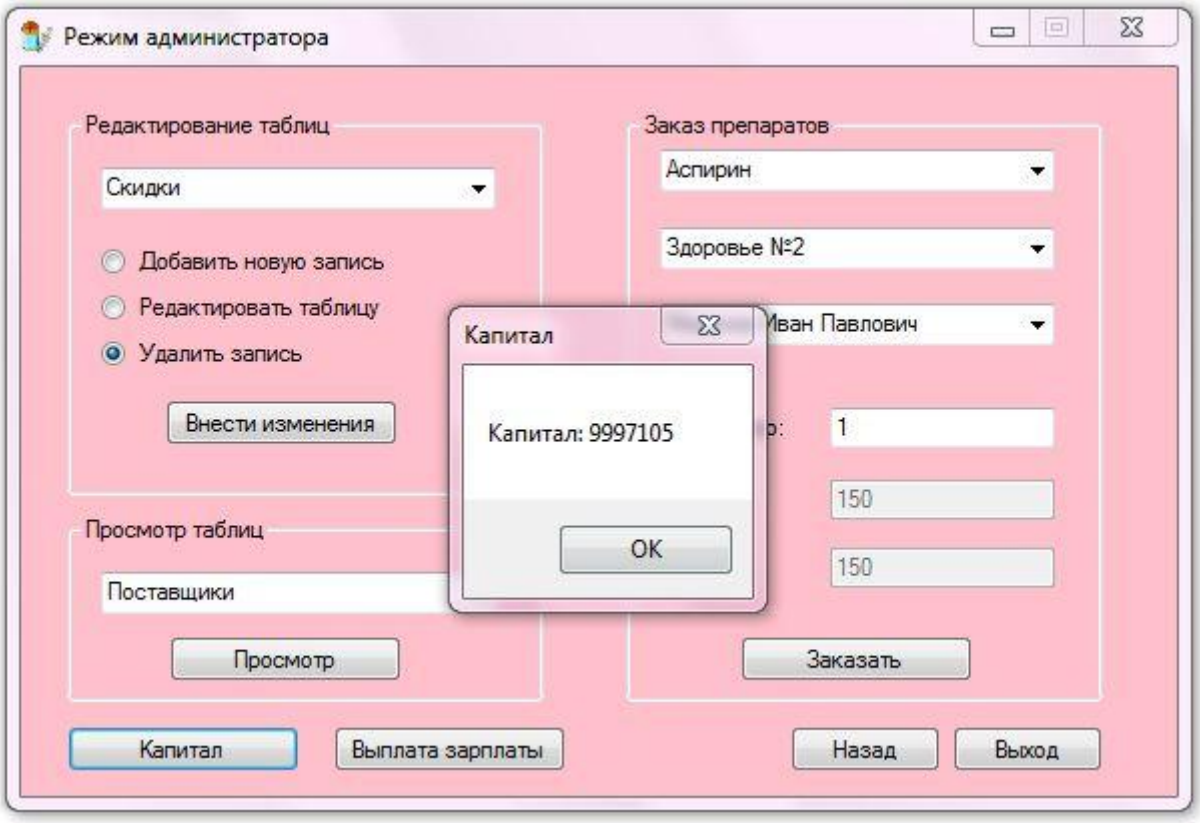

Рис. 3.27. Просмотр капитала

Нажимать на кнопку «Выплата зарплаты» необходимо раз в месяц, выплачивая заработную плату сотрудникам всех филиалов. Сумма капитала при этом уменьшается.

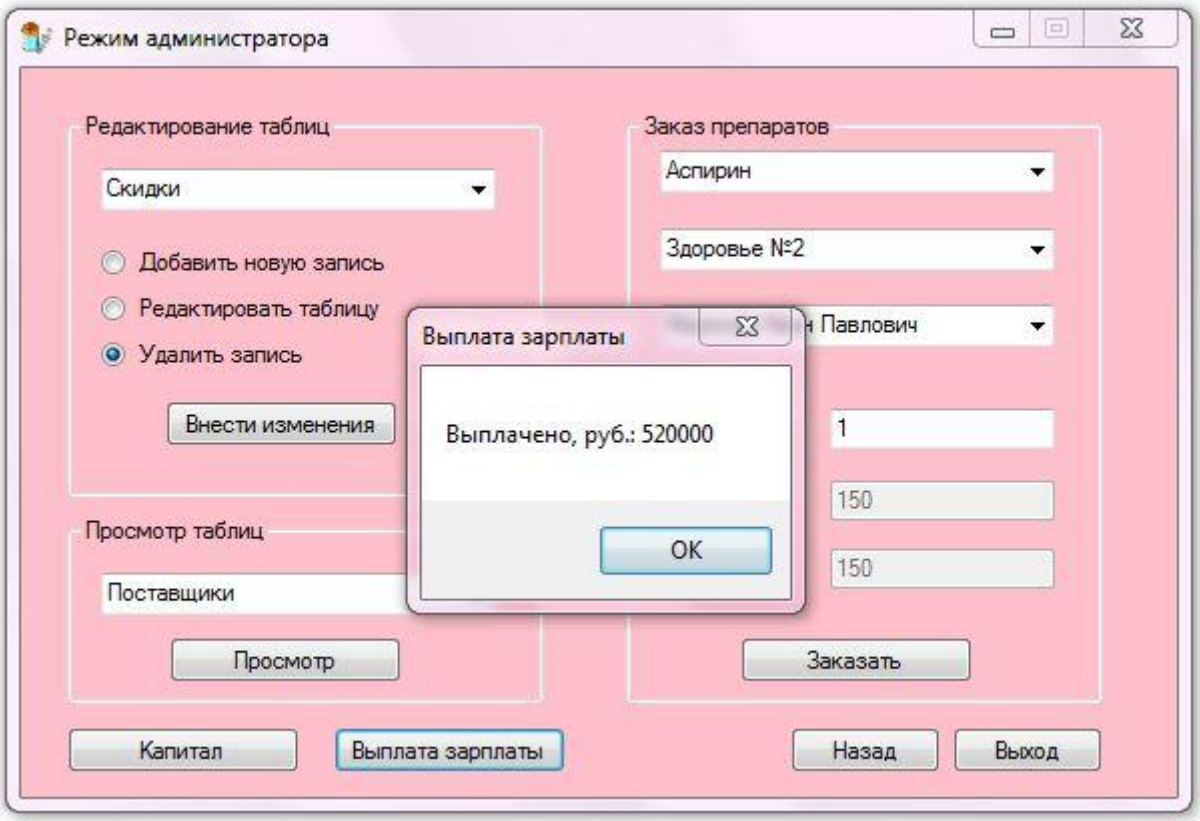

Рис. 3.28. Выплата зарплаты

Нажмем на кнопку «Назад» и вернемся к главному окну программы. Выберем режим работы «Продавец», введем логин и пароль (логин: Кострова, пароль: Светлана) и нажмем на кнопку «Вход». Откроется окно режима продавца:

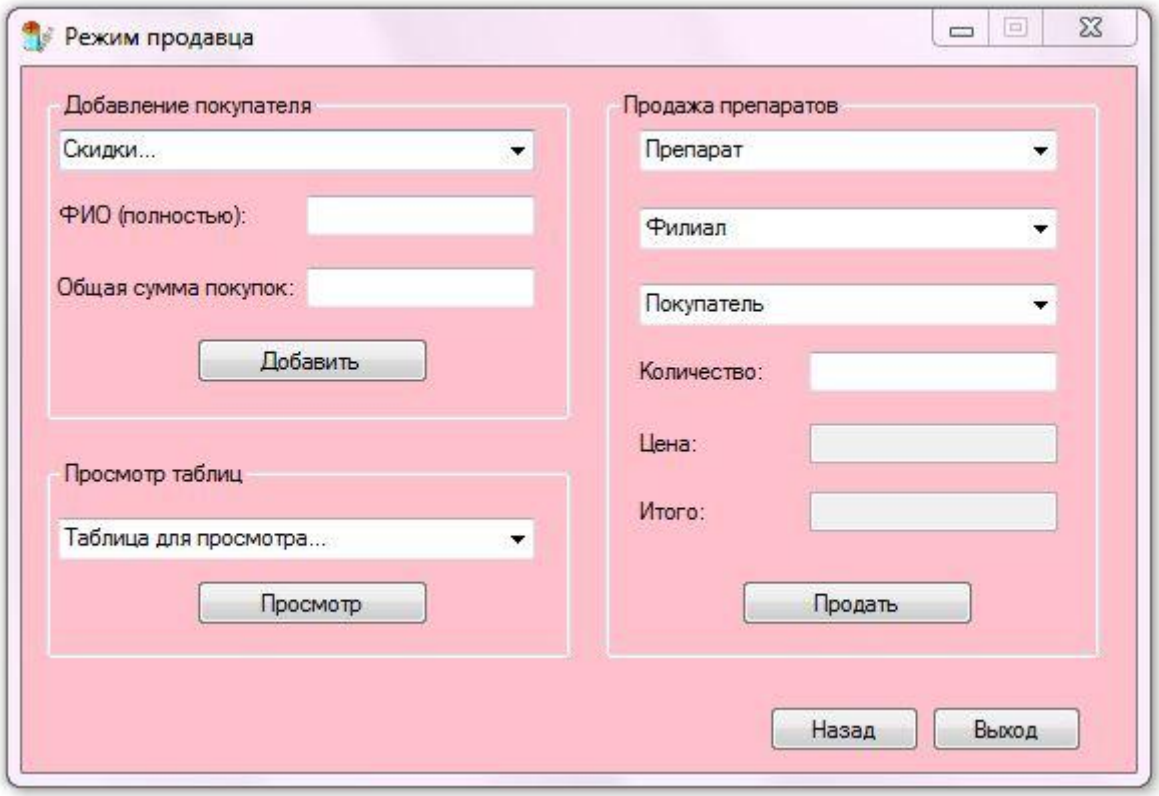

Рис. 3.29. Режим продавца

Чтобы добавить нового покупателя, на панели «Добавление покупателя» введем его ФИО и общую сумму покупок в соответствующие поля, а также назначим ему скидку с помощью выпадающего списка. Щелкнем по кнопке «Добавить». В случае ошибки программа попросит ввести недостающие данные или исправить уже существующие. Эту функцию мы уже описывали при использовании режима администратора.

С просмотром таблиц мы также ознакомились ранее. Заметим лишь, что возможности продавца урезаны по сравнению с администратором. Так, ему позволено просматривать лишь списки покупателей и скидок, а также наличие препаратов на складе. Впрочем, из просмотра склада он может мгновенно переместиться к таблицам филиалов и препаратов с помощью двойного щелчка мышью по соответствующим столбцам:

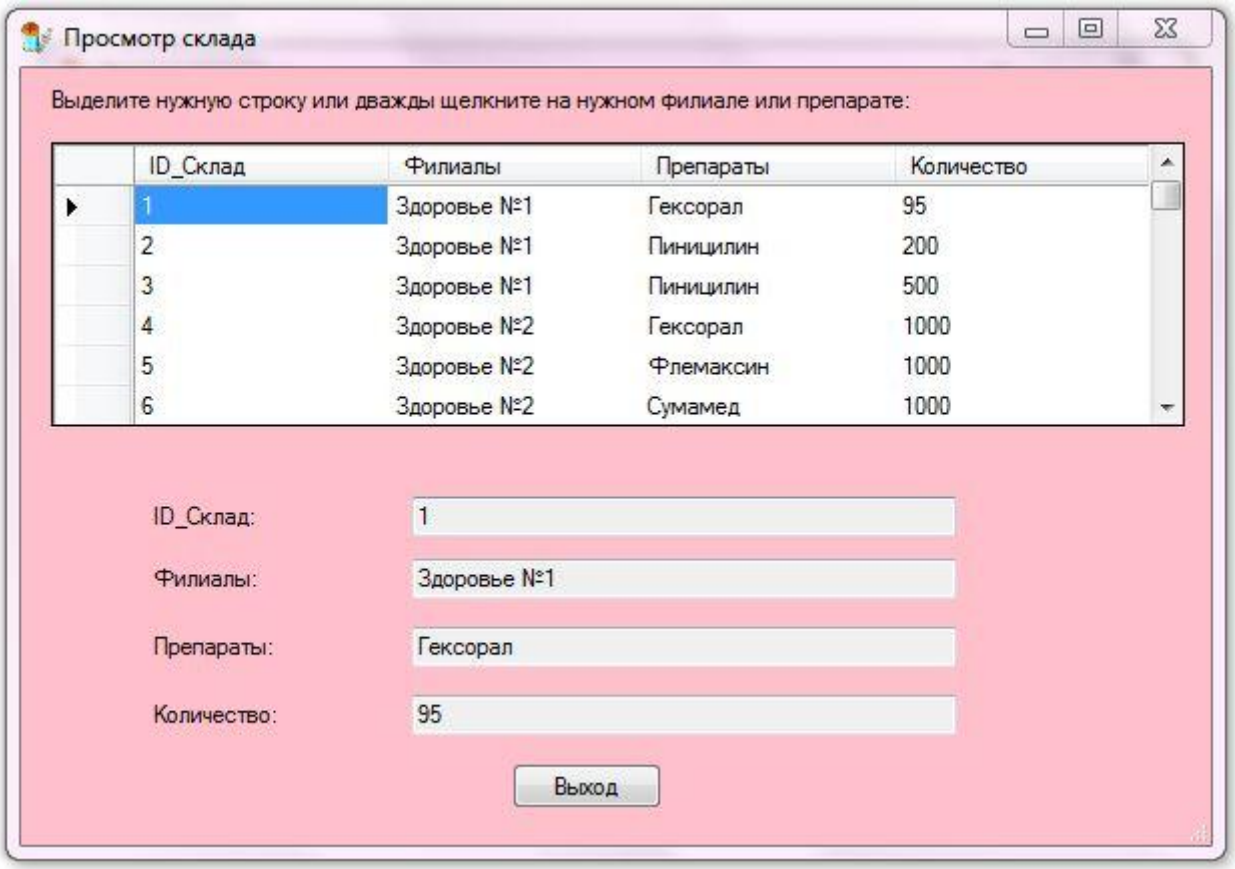

Рис. 3.30. Просмотр склада

Попробуем продать препарат покупателю. Продажа осуществляется аналогично покупке, подробно описанной в режиме администратора: необходимо выбрать препарат, филиал и покупателя в выпадающих списках и указать количество препаратов. Если у покупателя нет дисконтной карты (т.е. он не зарегистрирован в нашей базе данных), в соответствующем списке нужно выбрать «Незарегистрированный покупатель».

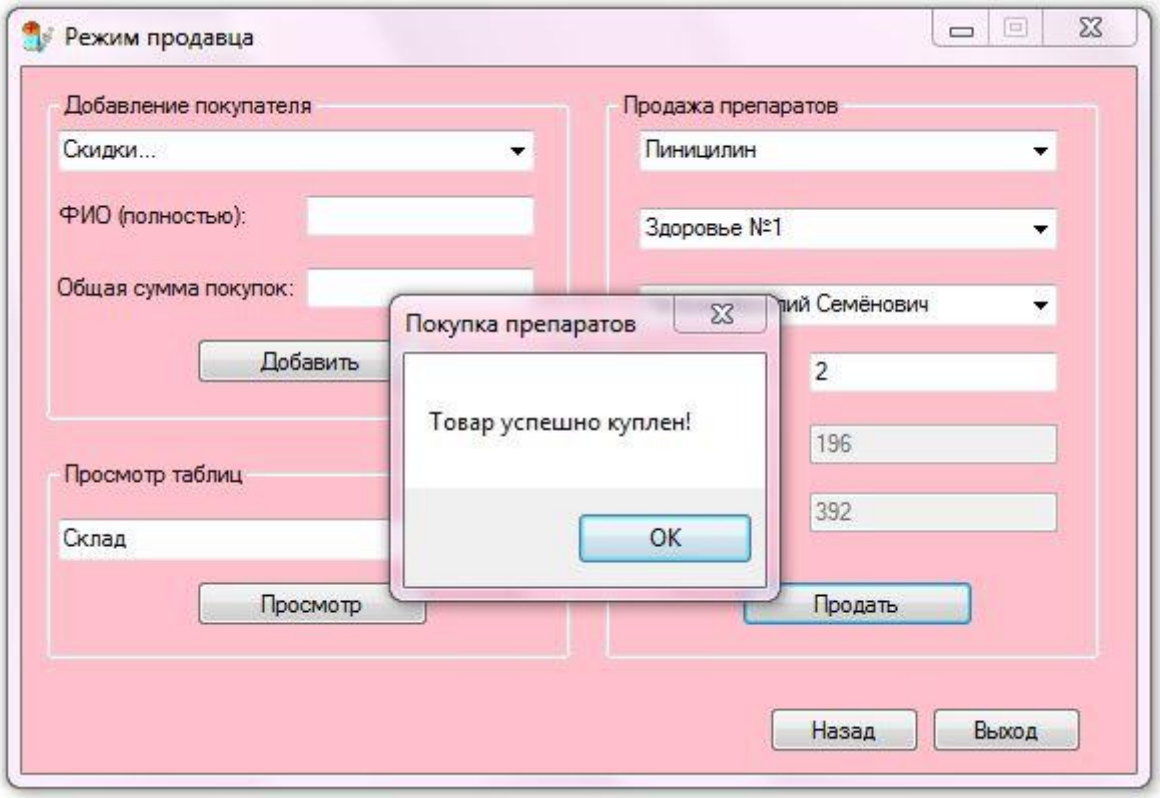

Рис. 3.31. Успешная продажа товара

Нажмем на кнопку «Назад» и вернемся к главному окну программы. Выберем режим покупателя. При этом название «Логин» заменяется на «ФИО», а поле пароля становится недоступным. Если не вводить свои данные, то работа будет осуществляться в режиме незарегистрированного пользователя:

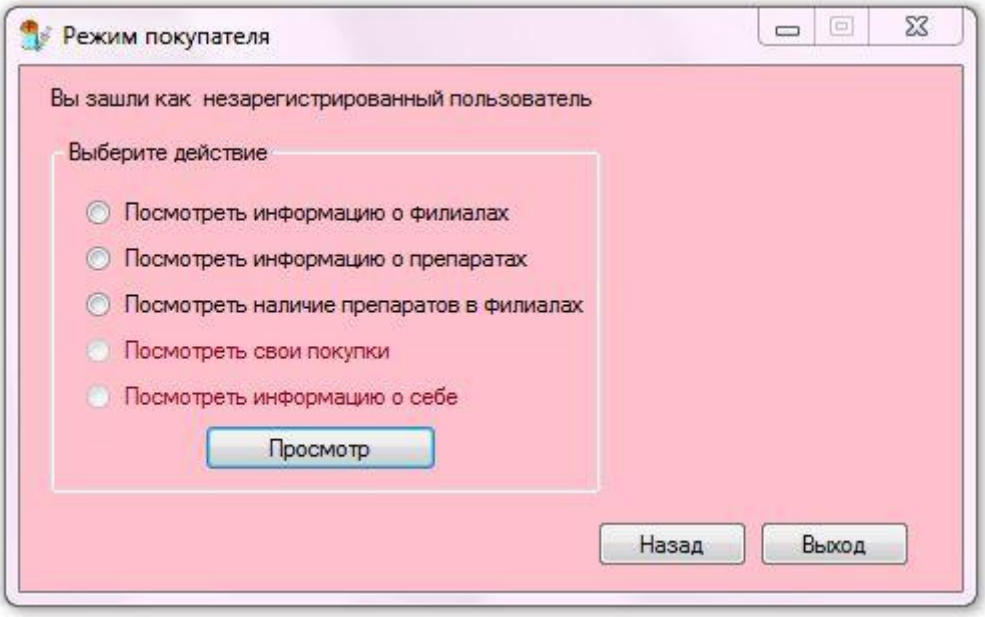

Рис. 3.32. Режим незарегистрированного покупателя

В режиме простого пользователя можно только посмотреть информацию о филиалах, препаратах и о наличии препаратов на складе. Для этого нужно нажать на соответствующую радиокнопку и щелкнуть по кнопке «Просмотр».

Вернемся к главному окну посредством нажатия кнопки «Назад», впишем в поле логина: Латыпкина Ольга Павловна и нажмем на кнопку «Вход». На экране появится следующее окно:

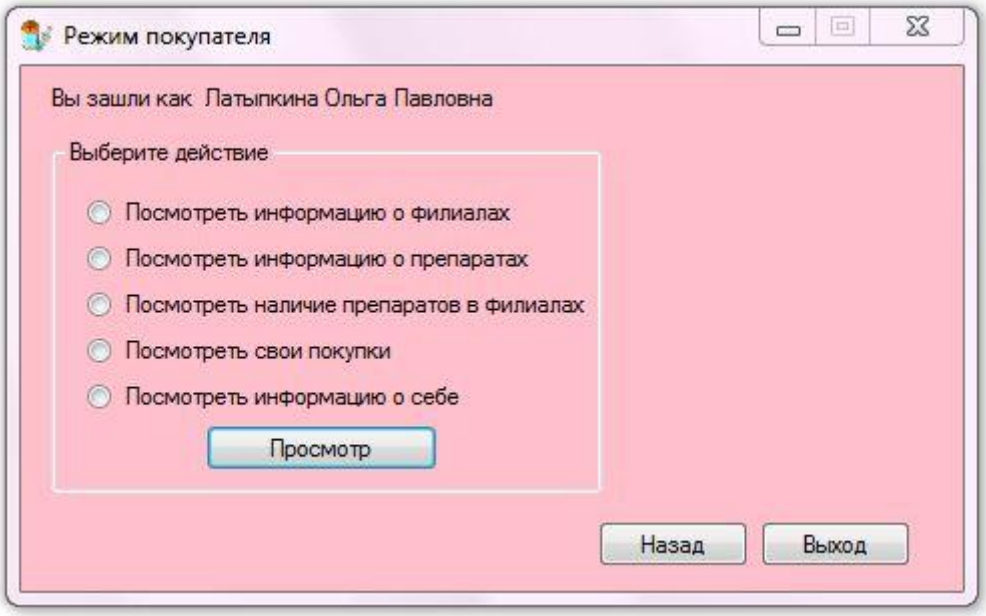

Рис. 3.33. Режим зарегистрированного покупателя

В режиме зарегистрированного покупателя можно просмотреть также свои покупки и информацию о себе:

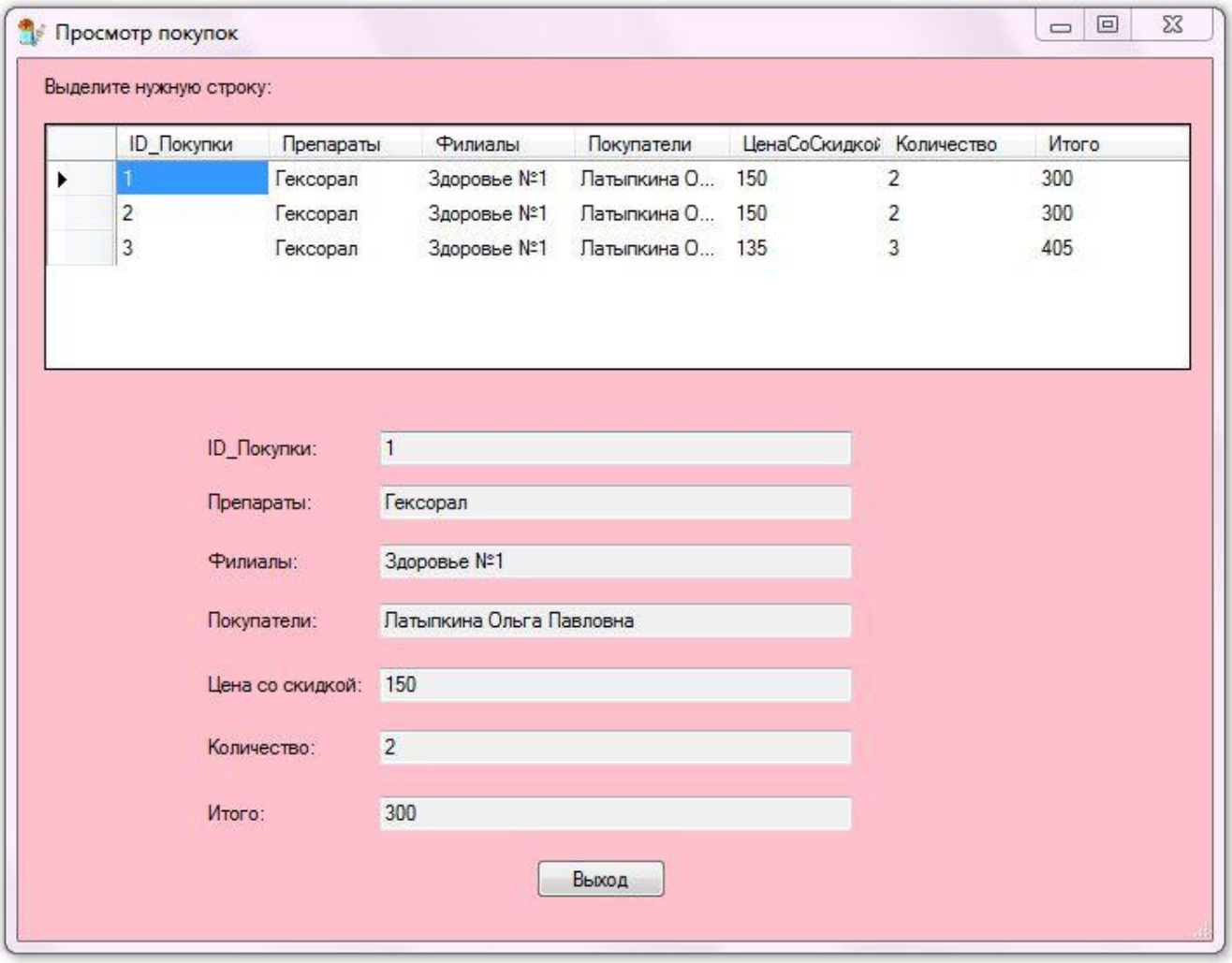

Рис. 3.34. Просмотр покупок конкретного покупателя

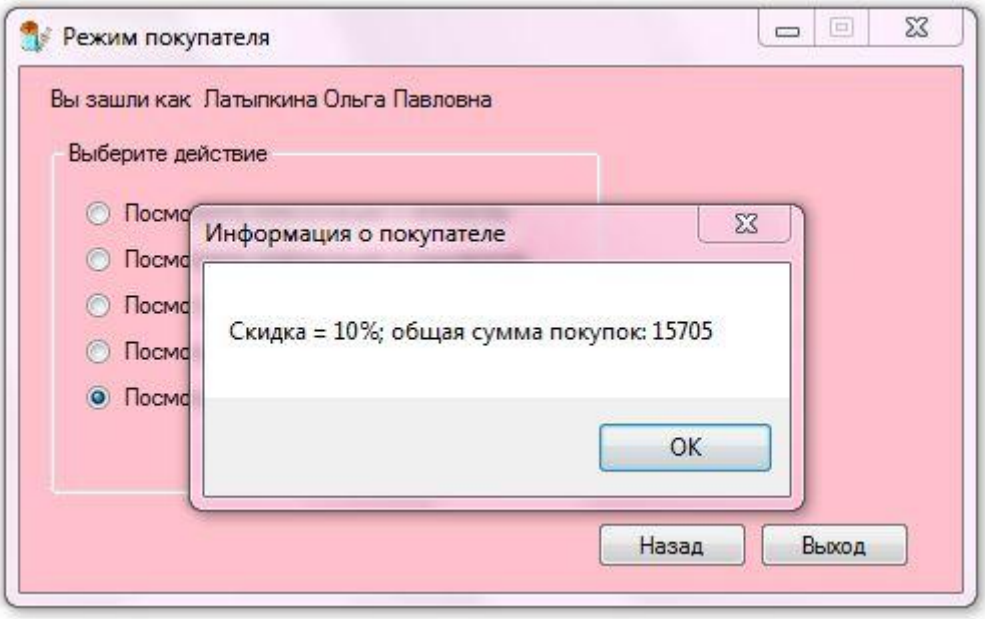

Рис. 3.35. Просмотр информации конкретного покупателя

Закройте окно с помощью кнопки «ОК». Для выхода из программы нажмите на кнопку «Выход».

ЗАКЛЮЧЕНИЕ

В ходе работы над курсовым проектом было выполнено создание БД, организовано взаимодействие с клиентским приложением. Осуществлены операции добавления, удаления и редактирования информации через хранимые процедуры, просмотр данных через запросы к БД.

В результате выполнения курсового проекта получена информационная система, предоставляющая пользователям простой и удобный способ взаимодействия с базой данных. Информационная система легко модифицируема и возможна дальнейшая ее доработка (как базы данных, так и клиентского приложения) для расширения круга решаемых задач. Выполненная работа не только полностью удовлетворяет поставленной задаче, но и расширяет ее возможности.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Вьейра Р. SQL Server 2000. Программирование ч.1 [Текст] / Р. Вьера. – М.: Изд-во "БИНОМ. Лаборатория знаний", 2004. – 736 с.

2. Вьейра Р. SQL Server 2000. Программирование ч.2 [Текст] / Р. Вьера. – М.: Изд-во "БИНОМ. Лаборатория знаний", 2004. – 808 с.

3. Троелсен Э. C# и платформа .NET 3.0. Специальное издание [Текст] / Э. Троелсен – СПб.: Питер, 2008. – 1456 с.

4. Библиотека MSDN для Microsoft Visual Studio [Электронный ресурс]. – Электрон. текст. дан. – Microsoft, 2012 – Режим доступа: http://msdn.microsoft.com/en-us/vstudio/aa718325, свободный.

ПРИЛОЖЕНИЕ 1: SQL-СКРИПТ ДЛЯ СОЗДАНИЯ БД

```
USE master
CREATE DATABASE [943_Аптека]
  ON
(NAME = '943_Аптека_Data',
FILENAME = 'C:\943_Аптека_Data.MDF' ,
SIZE = 2, FILEGROWTH = 10%)
  LOG ON
(NAME = '943_Аптека_Log',
FILENAME = 'C:943_Аптека_Log.LDF' ,
SIZE = 1, FILEGROWTH = 10%)
go
USE 943_Аптека
/*==============================================================*/
/* DBMS name:      Microsoft SQL Server 2005                    */
/* Created on:     09.04.2012 19:09:13                          */
/*==============================================================*/


/*==============================================================*/
/* Table: Должности                                             */
/*==============================================================*/
create table Должности (
   ID_Должности         int                  identity,
   Название             varchar(25)          not null,
   Зарплата             int                  not null
)
go

alter table Должности
   add constraint PK_ДОЛЖНОСТИ primary key nonclustered (ID_Должности)
go

/*==============================================================*/
/* Table: Капитал                                               */
/*==============================================================*/
create table Капитал (
   ID_Капитал           int                  identity
)
go

alter table Капитал
   add constraint PK_КАПИТАЛ primary key nonclustered (ID_Капитал)
go

/*==============================================================*/
/* Table: Категории                                             */
/*==============================================================*/
create table Категории (
   ID_Категории         int                  identity,
   Название             varchar(25)          not null
)
go

alter table Категории
   add constraint PK_КАТЕГОРИИ primary key nonclustered (ID_Категории)
go

/*==============================================================*/
```

```
/* Table: Покупатели                                            */
/*==============================================================*/
create table Покупатели (
   ID_Покупатели        int                  identity,
   Скидки               int                  null,
   ФИО                  varchar(255)         not null,
   ОбщаяСуммаПокупок    int                  not null
)
go

alter table Покупатели
   add constraint PK_ПОКУПАТЕЛИ primary key nonclustered (ID_Покупатели)
go

/*==============================================================*/
/* Table: Покупки                                               */
/*==============================================================*/
create table Покупки (
   ID_Покупки           int                  identity,
   Препараты            int                  not null,
   Филиалы              int                  not null,
   Покупатели           int                  not null,
   ЦенаСоСкидкой        int                  not null,
   Количество           int                  not null,
   Итого                int                  not null
)
go

alter table Покупки
   add constraint PK_ПОКУПКИ primary key nonclustered (ID_Покупки)
go

/*==============================================================*/
/* Table: Поставки                                              */
/*==============================================================*/
create table Поставки (
   ID_Поставки          int                  identity,
   Препараты            int                  not null,
   Филиалы              int                  not null,
   Поставщики           int                  not null,
   ЦенаПоставки         int                  not null,
   Количество           int                  not null,
   Итого                int                  not null
)
go

alter table Поставки
   add constraint PK_ПОСТАВКИ primary key nonclustered (ID_Поставки)
go

/*==============================================================*/
/* Table: Поставщики                                            */
/*==============================================================*/
create table Поставщики (
   ID_Поставщики        int                  identity,
   Категории            int                  null,
   ФИО                  varchar(255)         not null,
   Адрес                varchar(255)         null,
   Телефон              int                  null
)
go
```

```
alter table Поставщики
   add constraint PK_ПОСТАВЩИКИ primary key nonclustered (ID_Поставщики)
go

/*==============================================================*/
/* Table: Препараты                                             */
/*==============================================================*/
create table Препараты (
   ID_Препараты         int                  identity,
   Категории            int                  not null,
   Название             varchar(25)          not null,
   Цена                 int                  not null
)
go

alter table Препараты
   add constraint PK_ПРЕПАРАТЫ primary key nonclustered (ID_Препараты)
go

/*==============================================================*/
/* Table: ПривилегированныеСотрудники                           */
/*==============================================================*/
create table ПривилегированныеСотрудники (
   ID_Сотрудники        int                  not null,
   Должности            int                  not null,
   Филиалы              int                  not null,
   Логин                varchar(255)         not null,
   Пароль               varchar(30)          not null
)
go

alter table ПривилегированныеСотрудники
   add constraint PK_ПРИВИЛЕГИРОВАННЫЕСОТРУДНИКИ primary key nonclustered
(ID_Сотрудники)
go

/*==============================================================*/
/* Table: Скидки                                                */
/*==============================================================*/
create table Скидки (
   ID_Скидки            int                  identity,
   Название             varchar(25)          not null,
   Размер               smallint             not null
)
go

alter table Скидки
   add constraint PK_СКИДКИ primary key nonclustered (ID_Скидки)
go

/*==============================================================*/
/* Table: Склад                                                 */
/*==============================================================*/
create table Склад (
   ID_Склад             int                  identity,
   Филиалы              int                  null,
   Препараты            int                  null,
   Количество           int                  not null
)
go

alter table Склад
```

```
   add constraint PK_СКЛАД primary key nonclustered (ID_Склад)
go

/*==============================================================*/
/* Table: Сотрудники                                            */
/*==============================================================*/
create table Сотрудники (
   ID_Сотрудники        int                  identity,
   Должности            int                  null,
   Филиалы              int                  not null,
   ФИО                  varchar(255)         not null,
   Адрес                varchar(255)         null
)
go

alter table Сотрудники
   add constraint PK_СОТРУДНИКИ primary key nonclustered (ID_Сотрудники)
go

/*==============================================================*/
/* Table: Филиалы                                               */
/*==============================================================*/
create table Филиалы (
   ID_Филиалы           int                  identity,
   Название             varchar(25)          not null,
   Адрес                varchar(255)         not null,
   Телефон              int                  null
)
go

alter table Филиалы
   add constraint PK_ФИЛИАЛЫ primary key nonclustered (ID_Филиалы)
go

alter table Покупатели
   add constraint FK_ПОКУПАТЕ_ПОКУПАТЕЛ_СКИДКИ foreign key (Скидки)
      references Скидки (ID_Скидки)
         on update cascade
go

alter table Покупки
   add constraint FK_ПОКУПКИ_ПОКУПКИПО_ПОКУПАТЕ foreign key (Покупатели)
      references Покупатели (ID_Покупатели)
         on update cascade
go

alter table Покупки
   add constraint FK_ПОКУПКИ_ПОКУПКИПР_ПРЕПАРАТ foreign key (Препараты)
      references Препараты (ID_Препараты)
         on update cascade
go

alter table Покупки
   add constraint FK_ПОКУПКИ_ПОКУПКИФИ_ФИЛИАЛЫ foreign key (Филиалы)
      references Филиалы (ID_Филиалы)
         on update cascade
go

alter table Поставки
   add constraint FK_ПОСТАВКИ_ПОСТАВКИП_ПОСТАВЩИ foreign key (Поставщики)
      references Поставщики (ID_Поставщики)
         on update cascade
```

```
go

alter table Поставки
   add constraint FK_ПОСТАВКИ_ПОСТАВКИП_ПРЕПАРАТ foreign key (Препараты)
      references Препараты (ID_Препараты)
         on update cascade
go

alter table Поставки
   add constraint FK_ПОСТАВКИ_ПОСТАВКИФ_ФИЛИАЛЫ foreign key (Филиалы)
      references Филиалы (ID_Филиалы)
         on update cascade
go

alter table Поставщики
   add constraint FK_ПОСТАВЩИ_ПОСТАВЩИК_КАТЕГОРИ foreign key (Категории)
      references Категории (ID_Категории)
         on update cascade
go

alter table Препараты
   add constraint FK_ПРЕПАРАТ_ПРЕПАРАТЫ_КАТЕГОРИ foreign key (Категории)
      references Категории (ID_Категории)
         on update cascade
go

alter table ПривилегированныеСотрудники
   add constraint FK_ПРИВИЛЕГ_СОТРУДНИК_СОТРУДНИ foreign key (ID_Сотрудники)
      references Сотрудники (ID_Сотрудники)
         on update cascade
go

alter table Склад
   add constraint FK_СКЛАД_СКЛАДПРЕП_ПРЕПАРАТ foreign key (Препараты)
      references Препараты (ID_Препараты)
         on update cascade
go

alter table Склад
   add constraint FK_СКЛАД_СКЛАДФИЛИ_ФИЛИАЛЫ foreign key (Филиалы)
      references Филиалы (ID_Филиалы)
         on update cascade
go

alter table Сотрудники
   add constraint FK_СОТРУДНИ_СОТРУДНИК_ДОЛЖНОСТ foreign key (Должности)
      references Должности (ID_Должности)
         on update cascade
go

alter table Сотрудники
   add constraint FK_СОТРУДНИ_СОТРУДНИК_ФИЛИАЛЫ foreign key (Филиалы)
      references Филиалы (ID_Филиалы)
         on update cascade
go
----------
CREATE PROC HPAddBranch (@Название char(30),
                         @Адрес char(30),
                         @Телефон int)
AS
INSERT INTO Филиалы
VALUES(@Название, @Адрес, @Телефон)
```

```
----------
CREATE PROC HPAddBuyer (@Скидки char(30),
                                    @ФИО char(255),
                                    @ОбщаяСуммаПокупок int)
AS
DECLARE @ID_Скидки int
SELECT @ID_Скидки = ID_Скидки
      FROM Скидки
            WHERE Название = @Скидки

INSERT INTO Покупатели
VALUES(@ID_Скидки, @ФИО, @ОбщаяСуммаПокупок)
----------
CREATE PROC HPAddCategory(@Название char(30))
AS
INSERT INTO Категории
VALUES(@Название)
----------
CREATE PROC HPAddDiscount (@Название char(30),
                                          @Размер int)
AS
INSERT INTO Скидки
VALUES(@Название, @Размер)
----------
CREATE PROC HPAddEmployee (@Должности char(50),
                                    @Филиалы char(50),
                                    @ФИО char(255),
                                    @Адрес char(255),
                                    @Телефон int,
                                    @Логин char(255),
                                    @Пароль char(30))
AS
DECLARE @ID_Должности int
DECLARE @ID_Филиалы int

SELECT @ID_Должности = ID_Должности
      FROM Должности
            WHERE Название = @Должности
SELECT @ID_Филиалы = ID_Филиалы
      FROM Филиалы
            WHERE Название = @Филиалы

INSERT INTO Сотрудники
VALUES(@ID_Должности, @ID_Филиалы, @ФИО, @Адрес, @Телефон)

IF @Должности = 'Администратор' OR @Должности = 'Кассир-консультант'
BEGIN
      DECLARE @ID_Сотрудники int
      SELECT @ID_Сотрудники = ID_Сотрудники
            FROM Сотрудники
                  WHERE Должности = @ID_Должности AND Филиалы = @ID_Филиалы
      INSERT INTO ПривилегированныеСотрудники
      VALUES(@ID_Сотрудники, @Логин, @Пароль)
END
-----------
CREATE PROC HPAddPost(@Название char(30),
                        @Зарплата int)
AS
INSERT INTO Должности
VALUES(@Название, @Зарплата)
-----------
CREATE PROC HPAddPreparation (@Категории char(50),
```

```
                                          @Название char(50),
                                          @Цена int)
AS
DECLARE @ID_Категории int

SELECT @ID_Категории = ID_Категории
      FROM Категории
            WHERE Название = @Категории

INSERT INTO Препараты
VALUES(@ID_Категории, @Название, @Цена)
------------
CREATE PROC HPAddSupplier (@Категории char(50),
                        @ФИО char(30),
                        @Адрес char(30),
                        @Телефон int)
AS

DECLARE @ID_Категории int
SELECT @ID_Категории = ID_Категории
      FROM Категории
            WHERE @Категории = Название

INSERT INTO Поставщики
VALUES(@ID_Категории, @ФИО, @Адрес, @Телефон)
------------
CREATE PROC HPAddWarehouse (@Филиалы char(50),
                                    @Препараты char(50),
                                    @Количество int)
AS
DECLARE @ID_Филиалы int
DECLARE @ID_Препараты int

SELECT @ID_Филиалы = ID_Филиалы
      FROM Филиалы
            WHERE Название = @Филиалы
SELECT @ID_Препараты = ID_Препараты
      FROM Препараты
            WHERE Название = @Препараты
INSERT INTO Склад
VALUES(@ID_Филиалы, @ID_Препараты, @Количество)
-----------
CREATE PROC HPCheckAdmin (@Логин char(30),
                                          @Пароль char(30),
                                          @Код int OUTPUT)
AS
IF EXISTS (SELECT *
                  FROM ПривилегированныеСотрудники
                        WHERE @Логин = Логин AND @Пароль = Пароль)
      SET @Код = 0
ELSE
      SET @Код = 1
-----------
CREATE PROC HPCheckLogin (@Логин char(255),
                                          @Код int OUTPUT)
AS
IF EXISTS (SELECT *
                  FROM Покупатели
                        WHERE @Логин = ФИО)
BEGIN
      SELECT @Код = ID_Покупатели
            FROM Покупатели
```

```
                  WHERE @Логин = ФИО
END
ELSE
      SET @Код = 0
-----------
CREATE PROC HPDeleteBranch(@ID_Филиалы int)

AS
DELETE
      FROM Сотрудники
            WHERE Филиалы = @ID_Филиалы

DELETE
      FROM Склад
            WHERE Филиалы = @ID_Филиалы

DELETE
      FROM Филиалы
            WHERE ID_Филиалы = @ID_Филиалы
-----------
CREATE PROC HPDeleteBuyer(@ID_Покупатели int)
AS
DELETE
      FROM Покупатели
            WHERE ID_Покупатели = @ID_Покупатели
-----------
CREATE PROC HPDeleteCategory(@ID_Категории int)
AS
DELETE
      FROM Поставщики
            WHERE Категории = @ID_Категории

DELETE
      FROM Препараты
            WHERE Категории = @ID_Категории

DELETE
      FROM Категории
            WHERE ID_Категории = @ID_Категории
----------
CREATE PROC HPDeleteDiscount(@ID_Скидки int)
AS

DELETE
      FROM Покупатели
            WHERE Скидки = @ID_Скидки

DELETE
      FROM Скидки
            WHERE ID_Скидки = @ID_Скидки
----------
CREATE PROC HPDeleteEmployee(@ID_Сотрудники int)
AS
DELETE
      FROM Сотрудники
            WHERE ID_Сотрудники = @ID_Сотрудники

IF EXISTS (SELECT *
                  FROM ПривилегированныеСотрудники
                        WHERE ID_Сотрудники = @ID_Сотрудники)
      BEGIN
            DELETE
```

```
                  FROM ПривилегированныеСотрудники
                        WHERE ID_Сотрудники = @ID_Сотрудники
      END
---------
CREATE PROC HPDeletePost(@ID_Должности int)
AS
DELETE
      FROM Сотрудники
            WHERE Должности = @ID_Должности

DELETE
      FROM Должности
            WHERE ID_Должности = @ID_Должности
-----------
CREATE PROC HPDeletePreparation(@ID_Препараты int)
AS
DELETE
      FROM Препараты
            WHERE ID_Препараты = @ID_Препараты
-----------
CREATE PROC HPDeleteSupplier(@ID_Поставщики int)
AS
DELETE
      FROM Поставщики
            WHERE ID_Поставщики = @ID_Поставщики
----------
CREATE PROC HPDeleteWareHouse(@ID_Склад int)
AS
DELETE
      FROM Склад
            WHERE ID_Склад = @ID_Склад
----------
CREATE PROC HPGiveCountPreparation (@Препарат1 char(30),
                                                      @Филиал1 char(30),
                                                      @Количество1 int OUTPUT)
AS
DECLARE @ID_Препараты int
DECLARE @ID_Филиалы int

SELECT @ID_Препараты = ID_Препараты
      FROM Препараты
            WHERE @Препарат1 = Название

SELECT @ID_Филиалы = ID_Филиалы
      FROM Филиалы
            WHERE @Филиал1 = Название

SELECT @Количество1 = Количество
      FROM Склад
            WHERE @ID_Препараты = Препараты AND @ID_Филиалы = Филиалы
------------
CREATE PROC HPGiveDiscount (@Покупатель char(255),
                                          @Скидка int OUTPUT)
AS
SELECT @Скидка = Размер
      FROM Скидки
            WHERE ID_Скидки IN (SELECT Скидки
                                                FROM Покупатели
                                                      WHERE ФИО = @Покупатель)
-------------
CREATE PROC HPGiveDiscountBuyer (@Логин char(255),
                                                @Скидки int OUTPUT)
```

```
AS
SELECT @Скидки = Размер
      FROM Скидки
            WHERE ID_Скидки IN
                              (SELECT Скидки
                                    FROM Покупатели
                                          WHERE @Логин = ФИО)
-------------
CREATE PROC HPGiveNameBranch (@Филиалы int,
                                          @Название char(30) OUTPUT)
AS
SELECT @Название = Название
      FROM Филиалы
            WHERE ID_Филиалы = @Филиалы
--------------
CREATE PROC HPGiveNameBuyer (@Покупатели int,
                                          @ФИО char(255) OUTPUT)
AS
SELECT @ФИО = ФИО
      FROM Покупатели
            WHERE ID_Покупатели = @Покупатели
--------------
CREATE PROC HPGiveNameCategory (@Категории int,
                                                @Название char(30) OUTPUT)
AS
SELECT @Название = Название
      FROM Категории
            WHERE ID_Категории = @Категории
------------
CREATE PROC HPGiveNamePost (@Должности int,
                                          @Название char(30) OUTPUT)
AS
SELECT @Название = Название
      FROM Должности
            WHERE ID_Должности = @Должности
------------
CREATE PROC HPGiveNamePreparation (@Препараты int,
                                                      @Название char(30) OUTPUT)
AS
SELECT @Название = Название
      FROM Препараты
            WHERE ID_Препараты = @Препараты
----------
CREATE PROC HPGiveNameSupplier (@Поставщики int,
                                                @ФИО char(255) OUTPUT)
AS
SELECT @ФИО = ФИО
      FROM Поставщики
            WHERE ID_Поставщики = @Поставщики
----------
CREATE PROC HPGivePricePreparation (@Препарат char(30),
                                                      @Цена int OUTPUT)
AS
SELECT @Цена = Цена
      FROM Препараты
            WHERE Название = @Препарат
----------
CREATE PROC HPGivePricePreparationForSupplier (@Препарат char(30),
                                                      @Цена int OUTPUT)
AS
SELECT @Цена = Цена*3/4
      FROM Препараты
```

```
            WHERE Название = @Препарат
----------
CREATE PROC HPGiveSizeDiscount (@Скидки int,
                                                  @Размер int OUTPUT)
AS
SELECT @Размер = Размер
      FROM Скидки
            WHERE ID_Скидки = @Скидки
---------
CREATE PROC HPGiveSumBuyer  (@Логин1 char(255),
                                          @ОбщаяСуммаПокупок int OUTPUT)
AS
SELECT @ОбщаяСуммаПокупок = ОбщаяСуммаПокупок
      FROM Покупатели
            WHERE @Логин1 = ФИО

----------
CREATE PROC HPGiveSupplierFromPreparation (@Препарат char(30))
AS
SELECT ФИО
      FROM Поставщики
            WHERE Категории IN
                                    (SELECT Категории
                                          FROM Препараты
                                                WHERE Название = @Препарат)
------------
CREATE PROC HPGiveTotalSalary (@Итого int OUTPUT)

AS
DECLARE @Зарплата table(id int identity, summa int, primary key(id))
INSERT INTO @Зарплата
      SELECT Зарплата
            FROM Должности

DECLARE @КоличествоДолжности table(id int identity, count1 int, primary key(id))
INSERT INTO @КоличествоДолжности
SELECT COUNT(Должности)
      FROM Сотрудники
            GROUP BY Должности

SET @Итого = 0
DECLARE @number int
DECLARE @n1 int
DECLARE @n2 int
SET @number = 1

WHILE (SELECT COUNT(id) FROM @Зарплата) >= @number
      BEGIN
            SELECT @n1 = summa
                  FROM @Зарплата
                        WHERE id = @number
            SELECT @n2 = count1
                  FROM @КоличествоДолжности
                        WHERE id = @number
            SET @Итого = @Итого + @n1 * @n2
            SET @number = @number + 1
      END

UPDATE Капитал
SET Сумма = Сумма - @Итого
--------------
CREATE PROC HPMakeOrder (@Препарат char(30),
```

```
                                    @Филиал char(30),
                                    @Поставщик char(255),
                                    @Цена int,
                                    @Количество int,
                                    @Итого int)
AS
DECLARE @ID_Препараты int
DECLARE @ID_Филиалы int
DECLARE @ID_Поставщики int
DECLARE @Капитал int

SELECT @ID_Препараты = ID_Препараты
      FROM Препараты
            WHERE Название = @Препарат

SELECT @ID_Филиалы = ID_Филиалы
      FROM Филиалы
            WHERE Название = @Филиал

SELECT @ID_Поставщики = ID_Поставщики
      FROM Поставщики
            WHERE ФИО = @Поставщик

INSERT INTO Поставки
VALUES (@ID_Препараты, @ID_Филиалы, @ID_Поставщики, @Цена, @Количество, @Итого)

UPDATE Капитал
SET Сумма = Сумма - @Итого

IF EXISTS (SELECT *
                  FROM Склад
                        WHERE @ID_Филиалы = Филиалы AND @ID_Препараты = Препараты)
BEGIN
      UPDATE Склад
            SET Количество = Количество + @Количество
                  WHERE @ID_Филиалы = Филиалы AND @ID_Препараты = Препараты
END
ELSE
BEGIN
      INSERT INTO Склад
            VALUES (@ID_Филиалы, @ID_Препараты, @Количество)
END
----------
CREATE PROC HPMakeSale(@Препарат char(30),
                                    @Филиал char(30),
                                    @Покупатель char(255),
                                    @Цена int,
                                    @Количество int,
                                    @Итого int)
AS
DECLARE @ID_Препараты int
DECLARE @ID_Филиалы int
DECLARE @ID_Покупатели int

SELECT @ID_Препараты = ID_Препараты
      FROM Препараты
            WHERE Название = @Препарат

SELECT @ID_Филиалы = ID_Филиалы
      FROM Филиалы
            WHERE Название = @Филиал
```

```
SELECT @ID_Покупатели = ID_Покупатели
      FROM Покупатели
            WHERE ФИО = @Покупатель

INSERT INTO Покупки
VALUES (@ID_Препараты, @ID_Филиалы, @ID_Покупатели, @Цена, @Количество, @Итого)

UPDATE Покупатели
SET ОбщаяСуммаПокупок = ОбщаяСуммаПокупок + @Итого
WHERE @ID_Покупатели = ID_Покупатели AND ФИО <> 'Незарегистрированный покупатель'

UPDATE Капитал
SET Сумма = Сумма + @Итого

UPDATE Склад
SET Количество = Количество - @Количество
WHERE @ID_Филиалы = Филиалы AND @ID_Препараты = Препараты
--------
CREATE PROC HPUpdateBranch(@ID_Филиалы int,
                                    @Название char(30),
                                    @Адрес char(255),
                                    @Телефон int)
AS
UPDATE Филиалы
SET Название = @Название
WHERE ID_Филиалы = @ID_Филиалы

UPDATE Филиалы
SET Адрес = @Адрес
WHERE ID_Филиалы = @ID_Филиалы

UPDATE Филиалы
SET Телефон = @Телефон
WHERE ID_Филиалы = @ID_Филиалы
-------
CREATE PROC HPUpdateBuyer(@ID_Покупатели int,
                                        @Скидки int,
                                        @ФИО char(30),
                                        @ОбщаяСуммаПокупок int)
AS
UPDATE Покупатели
SET Скидки = @Скидки
WHERE ID_Покупатели = @ID_Покупатели

UPDATE Покупатели
SET ФИО = @ФИО
WHERE ID_Покупатели = @ID_Покупатели

UPDATE Покупатели
SET ОбщаяСуммаПокупок = @ОбщаяСуммаПокупок
WHERE ID_Покупатели = @ID_Покупатели
--------
CREATE PROC HPUpdateCategory(@ID_Категории int,
                                    @Название char(30))
AS
UPDATE Категории
SET Название = @Название
WHERE ID_Категории = @ID_Категории
--------
CREATE PROC HPUpdateDiscount(@ID_Скидки int,
                                    @Название char(30),
                                    @Размер int)
```

```
AS
UPDATE Скидки
SET Название = @Название
WHERE ID_Скидки = @ID_Скидки

UPDATE Скидки
SET Размер = @Размер
WHERE ID_Скидки = @ID_Скидки
--------
CREATE PROC HPUpdateEmployee(@ID_Сотрудники int,
                                        @Должности int,
                                        @Филиалы int,
                                        @ФИО char(255),
                                        @Адрес char(255),
                                        @Телефон int,
                                        @Логин char(30),
                                        @Пароль char(30))
AS
UPDATE Сотрудники
SET Должности = @Должности
WHERE ID_Сотрудники = @ID_Сотрудники

UPDATE Сотрудники
SET Филиалы = @Филиалы
WHERE ID_Сотрудники = @ID_Сотрудники

UPDATE Сотрудники
SET ФИО = @ФИО
WHERE ID_Сотрудники = @ID_Сотрудники

UPDATE Сотрудники
SET Адрес = @Адрес
WHERE ID_Сотрудники = @ID_Сотрудники

UPDATE Сотрудники
SET Телефон = @Телефон
WHERE ID_Сотрудники = @ID_Сотрудники

DECLARE @НазваниеДолжности char(50)
SELECT @НазваниеДолжности = Название
      FROM Должности
            WHERE @Должности = ID_Должности

IF @НазваниеДолжности = 'Администратор' OR @НазваниеДолжности = 'Кассир-консультант'
      BEGIN
            IF NOT EXISTS (SELECT *
                                    FROM ПривилегированныеСотрудники
                                          WHERE ID_Сотрудники = @ID_Сотрудники)
                  BEGIN
                        INSERT INTO ПривилегированныеСотрудники
                        VALUES(@ID_Сотрудники, @Логин, @Пароль)
                  END
            ELSE
                  BEGIN
                        UPDATE ПривилегированныеСотрудники
                              SET Логин = @Логин
                                    WHERE ID_Сотрудники = @ID_Сотрудники
                        UPDATE ПривилегированныеСотрудники
                              SET Пароль = @Пароль
                                    WHERE ID_Сотрудники = @ID_Сотрудники
                  END
      END
```

```
----------
CREATE PROC HPUpdatePost(@ID_Должности int,
                                    @Название char(30),
                                    @Зарплата int)
AS
UPDATE Должности
SET Название = @Название
WHERE ID_Должности = @ID_Должности

UPDATE Должности
SET Зарплата = @Зарплата
WHERE ID_Должности = @ID_Должности
--------
CREATE PROC HPUpdatePreparation(@ID_Препараты int,
                                                  @Категории int,
                                                  @Название char(30),
                                                  @Цена int)
AS
UPDATE Препараты
SET Категории = @Категории
WHERE ID_Препараты = @ID_Препараты

UPDATE Препараты
SET Название = @Название
WHERE ID_Препараты = @ID_Препараты

UPDATE Препараты
SET Цена = @Цена
WHERE ID_Препараты = @ID_Препараты
---------
CREATE PROC HPUpdateSupplier(@ID_Поставщики int,
                                          @Категории int,
                                          @ФИО char(255),
                                          @Адрес char(255),
                                          @Телефон int)
AS
UPDATE Поставщики
SET Категории = @Категории
WHERE ID_Поставщики = @ID_Поставщики

UPDATE Поставщики
SET ФИО = @ФИО
WHERE ID_Поставщики = @ID_Поставщики

UPDATE Поставщики
SET Адрес = @Адрес
WHERE ID_Поставщики = @ID_Поставщики

UPDATE Поставщики
SET Телефон = @Телефон
WHERE ID_Поставщики = @ID_Поставщики
---------
CREATE PROC HPUpdateWareHouse(@ID_Склад int,
                                          @Филиалы int,
                                          @Препараты int,
                                          @Количество int)
AS
UPDATE Склад
SET Филиалы = @Филиалы
WHERE ID_Склад = @ID_Склад

UPDATE Склад
```

```
SET Препараты = @Препараты
WHERE ID_Склад = @ID_Склад

UPDATE Склад
SET Количество = @Количество
WHERE ID_Склад = @ID_Склад
----------
```

ПРИЛОЖЕНИЕ 2: ИСХОДНЫЙ ТЕКСТ КЛИЕНТСКОЙ ПРОГРАММЫ

ConnectionForm.cs

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

namespace _43_Apteka
{
    public partial class ConnectionForm : Form
    {
        public ConnectionForm()
        {
            InitializeComponent();
            comboBoxIntegratedSecurity.SelectedItem =
comboBoxIntegratedSecurity.Items[1];
        }

        private void comboBoxItegratedSecurity_SelectedIndexChanged(object sender,
EventArgs e)
        {
            if (comboBoxIntegratedSecurity.SelectedItem ==
comboBoxIntegratedSecurity.Items[1])
            {
                panelIDPassword.Visible = false;
            }
            else
            {
                panelIDPassword.Visible = true;
            }
        }

        private void buttonTest_Click(object sender, EventArgs e)
        {
            string dataSource, initialCatalog, integratedSecurity, userID = "",
password = "";
            try
            {
                dataSource = textBoxDataSource.Text.ToString();
                initialCatalog = textBoxInitialCatalog.Text.ToString();
                if (comboBoxIntegratedSecurity.SelectedItem ==
comboBoxIntegratedSecurity.Items[0])
                {
                    integratedSecurity = "false";
                    userID = textBoxUserID.Text.ToString();
                    password = textBoxPassword.Text.ToString();
                }
                else
                {
                    integratedSecurity = "true";
                }
            }
            catch
```

```
            {
                return;
            }
            FormMain.conn.ConnectionString = "Data Source=" + dataSource +
                                ";Initial Catalog=" + initialCatalog + ";";
            if (comboBoxIntegratedSecurity.SelectedItem ==
comboBoxIntegratedSecurity.Items[0])
            {
                FormMain.conn.ConnectionString += "User ID=" + userID +
                                    ";Password=" + password +";";
            }
            else
            {
                FormMain.conn.ConnectionString += "Integrated Security=" +
integratedSecurity + ";";
            }
            try
            {
                FormMain.conn.Open();
                MessageBox.Show("Подключение успешно выполнено", "Проверка
подключения", MessageBoxButtons.OK);
                buttonTest.Visible = false;
                buttonCancel.Visible = true;
            }
            catch (System.Data.SqlClient.SqlException ex1)
            {
                MessageBox.Show("Не удалось выполнить подключение, проверьте
корректность введенных данных", "Проверка подключения", MessageBoxButtons.OK);
            }
        }
    }
}
```

FormAdminPage.cs

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

namespace _43_Apteka
{
    public partial class FormAdminPage : Form
    {
        public FormAdminPage()
        {
            InitializeComponent();
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "SELECT Название FROM Препараты";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                comboBoxPreparation.Items.Add(reader["Название"].ToString().Trim());
            }
            reader.Close();

            cmd.CommandText = "SELECT Название FROM Филиалы";
```

```
            reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                comboBoxBranch.Items.Add(reader["Название"].ToString().Trim());
            }
            reader.Close();
        }

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

        private void buttonMakeChanges_Click(object sender, EventArgs e)
        {
            try
            {
                string table = comboBoxEditTables.SelectedItem.ToString();
                if (radioButtonInsertInto.Checked == true)
                {
                    switch (table)
                    {
                        case "Должности":
                            FormNewPost formNewPost = new FormNewPost();
                            formNewPost.ShowDialog();
                            break;
                        case "Категории":
                            FormNewCategory formNewCategory = new FormNewCategory();
                            formNewCategory.ShowDialog();
                            break;
                        case "Покупатели":
                            FormNewBuyer formNewBuyer = new FormNewBuyer();
                            formNewBuyer.ShowDialog();
                            break;
                        case "Поставщики":
                            FormNewSupplier formNewSupplier = new FormNewSupplier();
                            formNewSupplier.ShowDialog();
                            break;
                        case "Препараты":
                            FormNewPreparation formNewPreparation = new
FormNewPreparation();
                            formNewPreparation.ShowDialog();
                            break;
                        case "Скидки":
                            FormNewDiscount formNewDiscount = new FormNewDiscount();
                            formNewDiscount.ShowDialog();
                            break;
                        case "Склад":
                            FormNewWarehouse formNewWarehouse = new
FormNewWarehouse();
                            formNewWarehouse.ShowDialog();
                            break;
                        case "Сотрудники":
                            FormNewEmployee formNewEmployee = new FormNewEmployee();
                            formNewEmployee.ShowDialog();
                            break;
                        case "Филиалы":
                            FormNewBranch formNewBranch = new FormNewBranch();
                            formNewBranch.ShowDialog();
                            break;
                    }
                }
```

```
                else
                {
                    if (radioButtonUpdate.Checked == true)
                    {
                        switch (table)
                        {
                            case "Должности":
                                FormEditPost formEditPost = new FormEditPost();
                                formEditPost.ShowDialog();
                                break;
                            case "Категории":
                                FormEditCategory formEditCategory = new
FormEditCategory();
                                formEditCategory.ShowDialog();
                                break;
                            case "Покупатели":
                                FormEditBuyer formEditBuyer = new FormEditBuyer();
                                formEditBuyer.ShowDialog();
                                break;
                            case "Поставщики":
                                FormEditSupplier formEditSupplier = new
FormEditSupplier();
                                formEditSupplier.ShowDialog();
                                break;
                            case "Препараты":
                                FormEditPreparation formEditPreparation = new
FormEditPreparation();
                                formEditPreparation.ShowDialog();
                                break;
                            case "Скидки":
                                FormEditDiscount formEditDiscount = new
FormEditDiscount();
                                formEditDiscount.ShowDialog();
                                break;
                            case "Склад":
                                FormEditWarehouse formEditWarehouse = new
FormEditWarehouse();
                                formEditWarehouse.ShowDialog();
                                break;
                            case "Сотрудники":
                                FormEditEmployee formEditEmployee = new
FormEditEmployee();
                                formEditEmployee.ShowDialog();
                                break;
                            case "Филиалы":
                                FormEditBranch formEditBranch = new FormEditBranch();
                                formEditBranch.ShowDialog();
                                break;
                        }
                    }
                    else
                    {
                        if (radioButtonDelete.Checked == true)
                        {
                            switch (table)
                            {
                                case "Должности":
                                    FormDeletePost formDeletePost = new
FormDeletePost();
                                    formDeletePost.ShowDialog();
                                    break;
                                case "Категории":
```

```
                                    FormDeleteCategory formDeleteCategory = new
FormDeleteCategory();
                                    formDeleteCategory.ShowDialog();
                                    break;
                                case "Покупатели":
                                    FormDeleteBuyer formDeleteBuyer = new
FormDeleteBuyer();
                                    formDeleteBuyer.ShowDialog();
                                    break;
                                case "Поставщики":
                                    FormDeleteSupplier formDeleteSupplier = new
FormDeleteSupplier();
                                    formDeleteSupplier.ShowDialog();
                                    break;
                                case "Препараты":
                                    FormDeletePreparation formDeletePreparation = new
FormDeletePreparation();
                                    formDeletePreparation.ShowDialog();
                                    break;
                                case "Скидки":
                                    FormDeleteDiscount formDeleteDiscount = new
FormDeleteDiscount();
                                    formDeleteDiscount.ShowDialog();
                                    break;
                                case "Склад":
                                    FormDeleteWarehouse formDeleteWarehouse = new
FormDeleteWarehouse();
                                    formDeleteWarehouse.ShowDialog();
                                    break;
                                case "Сотрудники":
                                    FormDeleteEmployee formDeleteEmployee = new
FormDeleteEmployee();
                                    formDeleteEmployee.ShowDialog();
                                    break;
                                case "Филиалы":
                                    FormDeleteBranch formDeleteBranch = new
FormDeleteBranch();
                                    formDeleteBranch.ShowDialog();
                                    break;
                            }
                        }
                    }
                }
            }
            catch (NullReferenceException ex)
            {
                MessageBox.Show("Выберите таблицу для изменения", "Ошибка изменения",
MessageBoxButtons.OK);
            }
        }

        private void buttonViewTable_Click(object sender, EventArgs e)
        {
            try
            {
                string table = comboBoxViewTables.SelectedItem.ToString();
                switch (table)
                {
                    case "Должности":
                        FormViewPost formViewPost = new FormViewPost();
                        formViewPost.ShowDialog();
                        break;
```

```
                    case "Категории":
                        FormViewCategory formViewCategory = new FormViewCategory();
                        formViewCategory.ShowDialog();
                        break;
                    case "Покупатели":
                        FormViewBuyer formViewBuyer = new FormViewBuyer();
                        formViewBuyer.ShowDialog();
                        break;
                    case "Покупки":
                        FormViewPurchase formViewPurchase = new FormViewPurchase();
                        formViewPurchase.ShowDialog();
                        break;
                    case "Поставки":
                        FormViewSupply formViewSupply = new FormViewSupply();
                        formViewSupply.ShowDialog();
                        break;
                    case "Поставщики":
                        FormViewSupplier formViewSupplier = new FormViewSupplier();
                        formViewSupplier.ShowDialog();
                        break;
                    case "Препараты":
                        FormViewPreparation formViewPreparation = new
FormViewPreparation();
                        formViewPreparation.ShowDialog();
                        break;
                    case "Скидки":
                        FormViewDiscount formViewDiscount = new FormViewDiscount();
                        formViewDiscount.ShowDialog();
                        break;
                    case "Склад":
                        FormViewWarehouse formViewWarehouse = new
FormViewWarehouse();
                        formViewWarehouse.ShowDialog();
                        break;
                    case "Сотрудники":
                        FormViewEmployee formViewEmployee = new FormViewEmployee();
                        formViewEmployee.ShowDialog();
                        break;
                    case "Филиалы":
                        FormViewBranch formViewBranch = new FormViewBranch();
                        formViewBranch.ShowDialog();
                        break;
                }
            }
            catch (NullReferenceException ex)
            {
                MessageBox.Show("Выберите таблицу для просмотра", "Ошибка просмотра",
MessageBoxButtons.OK);
            }
        }

        private void comboBoxPreparation_SelectedIndexChanged(object sender,
EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPGivePricePreparationForSupplier @Препарат,
@Цена OUTPUT";
            cmd.Parameters.Add("@Препарат", SqlDbType.Char).Value =
comboBoxPreparation.SelectedItem.ToString().Trim();
            cmd.Parameters.Add(new SqlParameter("@Цена", SqlDbType.Int));
            cmd.Parameters["@Цена"].Direction = ParameterDirection.Output;
            cmd.ExecuteNonQuery();
```

```
            textBoxPrice.Text = cmd.Parameters["@Цена"].Value.ToString();

            cmd.CommandText = "EXECUTE HPGiveSupplierFromPreparation @Препарат1";
            cmd.Parameters.Add("@Препарат1", SqlDbType.Char).Value =
comboBoxPreparation.SelectedItem.ToString().Trim();
            SqlDataAdapter dataAdapter = new SqlDataAdapter(cmd);
            DataTable dt = new DataTable();
            dataAdapter.Fill(dt);

            bindingSourceOrder.DataSource = dt;
            comboBoxSupplier.DisplayMember = "ФИО".Trim();
        }

        private void buttonOrder_Click(object sender, EventArgs e)
        {
            try
            {
                SqlCommand cmd = FormMain.conn.CreateCommand();
                cmd.CommandText = "SELECT Сумма FROM Капитал WHERE ID_Капитал = 1";
                SqlDataReader reader = cmd.ExecuteReader();
                int capital = 0;
                while (reader.Read())
                {
                    capital = Convert.ToInt32(reader["Сумма"].ToString());
                }
                reader.Close();
                if (capital > Convert.ToInt32(textBoxTotal.Text.ToString()))
                {
                    cmd = FormMain.conn.CreateCommand();
                    cmd.CommandText = "EXECUTE HPMakeOrder @Препарат, @Филиал,
@Поставщик, @Цена, @Количество, @Итого";
                    cmd.Parameters.Add("@Препарат", SqlDbType.Char).Value =
comboBoxPreparation.SelectedItem.ToString().Trim();
                    cmd.Parameters.Add("@Филиал", SqlDbType.Char).Value =
comboBoxBranch.SelectedItem.ToString().Trim();
                    cmd.Parameters.Add("@Поставщик", SqlDbType.VarChar, 255).Value =
((DataRowView)comboBoxSupplier.SelectedItem).Row.ItemArray[0].ToString().Trim();
                    cmd.Parameters.Add("@Цена", SqlDbType.Int).Value =
textBoxPrice.Text.ToString();
                    cmd.Parameters.Add("@Количество", SqlDbType.Int).Value =
textBoxCount.Text.ToString();
                    cmd.Parameters.Add("@Итого", SqlDbType.Int).Value =
textBoxTotal.Text.ToString();
                    cmd.ExecuteNonQuery();
                    MessageBox.Show("Товар успешно заказан!", "Заказ товара",
MessageBoxButtons.OK);
                }
                else
                {
                    MessageBox.Show("Недостаточно средств для покупки", "Ошибка
заказа", MessageBoxButtons.OK);
                }
            }
            catch (FormatException ex)
            {
                MessageBox.Show("Введите корректные значения", "Ошибка заказа",
MessageBoxButtons.OK);
            }
            catch (NullReferenceException ex)
            {
                MessageBox.Show("Введите корректные значения", "Ошибка заказа",
MessageBoxButtons.OK);
```

```
            }
        }

        private void textBoxCount_TextChanged(object sender, EventArgs e)
        {
            try
            {
                int total = Convert.ToInt32(textBoxPrice.Text) *
Convert.ToInt32(textBoxCount.Text);
                textBoxTotal.Text = total.ToString();
            }
            catch (FormatException ex)
            {
                textBoxTotal.Text = "";
            }
        }

        private void buttonCapital_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "SELECT Сумма FROM Капитал";
            SqlDataReader reader = cmd.ExecuteReader();
            int capital = 0;
            while (reader.Read())
            {
                capital = Convert.ToInt32(reader["Сумма"].ToString());
            }
            reader.Close();
            MessageBox.Show("Капитал: " + capital.ToString(), "Капитал",
MessageBoxButtons.OK);
        }

        private void buttonSalary_Click(object sender, EventArgs e)
        {
            try
            {
                SqlCommand cmd = FormMain.conn.CreateCommand();
                cmd = FormMain.conn.CreateCommand();
                cmd.CommandText = "EXECUTE HPGiveTotalSalary @Итого OUTPUT";
                cmd.Parameters.Add(new SqlParameter("@Итого", SqlDbType.Int));
                cmd.Parameters["@Итого"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                MessageBox.Show("Выплачено, руб.: " +
cmd.Parameters["@Итого"].Value.ToString(), "Выплата зарплаты", MessageBoxButtons.OK);
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка выплаты зарплаты","Выплата зарплаты",
MessageBoxButtons.OK);
            }

        }

    }
}
```

FormBuyerPage.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
```

```
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace _43_Apteka
{
    public partial class FormBuyerPage : Form
    {
        private string login;
        private int code;
        public FormBuyerPage()
        {
            InitializeComponent();
        }
        public FormBuyerPage(string login)
        {
            InitializeComponent();
            this.login = login;
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPCheckLogin @Логин, @Код OUTPUT";
            cmd.Parameters.Add("@Логин", SqlDbType.Char).Value = login;
            cmd.Parameters.Add(new SqlParameter("@Код", SqlDbType.Int));
            cmd.Parameters["@Код"].Direction = ParameterDirection.Output;
            cmd.ExecuteNonQuery();
            code = Convert.ToInt32(cmd.Parameters["@Код"].Value.ToString().Trim());
            if (code != 0)
            {
                labelLogin.Text = login;
                radioButtonPurchase.Enabled = true;
                radioButtonYorself.Enabled = true;
            }

        }

        private void buttonViewTable_Click(object sender, EventArgs e)
        {
            if (radioButtonBranch.Checked == true)
            {
                FormViewBranch formViewBranch = new FormViewBranch();
                formViewBranch.ShowDialog();
            }
            if (radioButtonPreparation.Checked == true)
            {
                FormViewPreparation formViewPreparation = new FormViewPreparation();
                formViewPreparation.ShowDialog();
            }
            if (radioButtonWarehouse.Checked == true)
            {
                FormViewWarehouse formViewWarehouse = new FormViewWarehouse();
                formViewWarehouse.ShowDialog();
            }
            if (radioButtonPurchase.Checked == true)
            {
                FormViewPurchase formViewPurchase = new FormViewPurchase(code);
                formViewPurchase.ShowDialog();
            }
            if (radioButtonYorself.Checked == true)
            {
                SqlCommand cmd = FormMain.conn.CreateCommand();
```

```
                cmd.CommandText = "EXECUTE HPGiveDiscountBuyer @Логин, @Скидки
OUTPUT";
                cmd.Parameters.Add("@Логин", SqlDbType.Char).Value = login;
                cmd.Parameters.Add(new SqlParameter("@Скидки", SqlDbType.Int));
                cmd.Parameters["@Скидки"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                string discount = cmd.Parameters["@Скидки"].Value.ToString().Trim();

                cmd.CommandText = "EXECUTE HPGiveSumBuyer @Логин1, @ОбщаяСуммаПокупок
OUTPUT";
                cmd.Parameters.Add("@Логин1", SqlDbType.Char).Value = login;
                cmd.Parameters.Add(new SqlParameter("@ОбщаяСуммаПокупок",
SqlDbType.Int));
                cmd.Parameters["@ОбщаяСуммаПокупок"].Direction =
ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                string sum =
cmd.Parameters["@ОбщаяСуммаПокупок"].Value.ToString().Trim();

                MessageBox.Show("Скидка = " + discount + "%; общая сумма покупок: " +
sum, "Информация о покупателе", MessageBoxButtons.OK);
            }
        }

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

FormDeleteBranch.cs

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

namespace _43_Apteka
{
    public partial class FormDeleteBranch : Form
    {
        public FormDeleteBranch()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Филиалы";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell name = new DataGridViewTextBoxCell();
                DataGridViewCell address = new DataGridViewTextBoxCell();
                DataGridViewCell telephone = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Филиалы"].ToString();
```

```
                name.Value = reader["Название"].ToString();
                address.Value = reader["Адрес"].ToString();
                telephone.Value = reader["Телефон"].ToString();
                row.Cells.AddRange(id, name, address, telephone);
                dataGridViewBranch.Rows.Add(row);
            }
            reader.Close();
        }

        private void buttonDeleteBranch_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPDeleteBranch @ID_Филиалы";
            cmd.Parameters.Add("@ID_Филиалы", SqlDbType.Int).Value =
dataGridViewBranch[0, dataGridViewBranch.CurrentRow.Index].Value.ToString();
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Филиал успешно удален", "Обновление филиала",
MessageBoxButtons.OK);
                this.Close();
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка обновления филиала", "Обновление филиала",
MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Обновление филиала",
MessageBoxButtons.OK);
            }
        }
    }
}
```

FormDeleteBuyer.cs

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

namespace _43_Apteka
{
    public partial class FormDeleteBuyer : Form
    {
        public FormDeleteBuyer()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Покупатели";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell discount = new DataGridViewTextBoxCell();
```

```
                DataGridViewCell fio = new DataGridViewTextBoxCell();
                DataGridViewCell sumBuy = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Покупатели"].ToString();
                discount.Value = reader["Скидки"].ToString();
                fio.Value = reader["ФИО"].ToString();
                sumBuy.Value = reader["ОбщаяСуммаПокупок"].ToString();
                row.Cells.AddRange(id, discount, fio, sumBuy);
                dataGridViewBuyer.Rows.Add(row);
            }
            reader.Close();

            for (int i = 0; i < dataGridViewBuyer.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveSizeDiscount @Скидки, @Размер
OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Скидки", SqlDbType.Int).Value =
dataGridViewBuyer[1, i].Value.ToString();
                cmd.Parameters.Add("@Размер", SqlDbType.Int);
                cmd.Parameters["@Размер"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewBuyer[1, i].Value =
cmd.Parameters["@Размер"].Value.ToString();
            }
        }

        private void buttonDeleteBuyer_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPDeleteBuyer @ID_Покупатели";
            cmd.Parameters.Add("@ID_Покупатели", SqlDbType.Int).Value =
dataGridViewBuyer[0, dataGridViewBuyer.CurrentRow.Index].Value.ToString();
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Покупатель успешно удален", "Удаление покупателя",
MessageBoxButtons.OK);
                this.Close();
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка удаления покупателя", "Удаление покупателя",
MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Удаление
покупателя", MessageBoxButtons.OK);
            }
        }
    }
}
```

FormDeleteCategory.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
```

```
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace _43_Apteka
{
    public partial class FormDeleteCategory : Form
    {
        public FormDeleteCategory()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Категории";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell name = new DataGridViewTextBoxCell();
                DataGridViewRow row = new DataGridViewRow();

                id.Value = reader["ID_Категории"].ToString();
                name.Value = reader["Название"].ToString();
                row.Cells.AddRange(id, name);
                dataGridViewCategory.Rows.Add(row);
            }
            reader.Close();
        }

        private void buttonDeletePost_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPDeleteCategory @ID_Категории";
            cmd.Parameters.Add("@ID_Категории", SqlDbType.Int).Value =
dataGridViewCategory[0, dataGridViewCategory.CurrentRow.Index].Value.ToString();
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Категория успешно удалена", "Удаление категории",
MessageBoxButtons.OK);
                this.Close();
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка удаления категории", "Удаление категории",
MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Удаление категории",
MessageBoxButtons.OK);
            }
        }
    }
}
```

FormDeleteDiscount.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
```

```
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace _43_Apteka
{
    public partial class FormDeleteDiscount : Form
    {
        public FormDeleteDiscount()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Скидки";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell name = new DataGridViewTextBoxCell();
                DataGridViewCell size = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Скидки"].ToString();
                name.Value = reader["Название"].ToString();
                size.Value = reader["Размер"].ToString();
                row.Cells.AddRange(id, name, size);
                dataGridViewDiscount.Rows.Add(row);
            }
            reader.Close();
        }

        private void buttonDeleteDiscount_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPDeleteDiscount @ID_Скидки";
            cmd.Parameters.Add("@ID_Скидки", SqlDbType.Int).Value =
dataGridViewDiscount[0, dataGridViewDiscount.CurrentRow.Index].Value.ToString();
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Скидка успешно удалена", "Удаление скидки",
MessageBoxButtons.OK);
                this.Close();
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка удаления скидки", "Удаление скидки",
MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Удаление скидки",
MessageBoxButtons.OK);
            }
        }
    }
}
```

FormDeleteEmployee.cs

```
using System;
using System.Collections.Generic;
```

```
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace _43_Apteka
{
    public partial class FormDeleteEmployee : Form
    {
        public FormDeleteEmployee()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Сотрудники";
            SqlDataReader reader = cmd.ExecuteReader();
            int j = 0;
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell post = new DataGridViewTextBoxCell();
                DataGridViewCell branch = new DataGridViewTextBoxCell();
                DataGridViewCell fio = new DataGridViewTextBoxCell();
                DataGridViewCell address = new DataGridViewTextBoxCell();
                DataGridViewCell telephone = new DataGridViewTextBoxCell();
                DataGridViewCell login = new DataGridViewTextBoxCell();
                DataGridViewCell password = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Сотрудники"].ToString();
                post.Value = reader["Должности"].ToString();
                branch.Value = reader["Филиалы"].ToString();
                fio.Value = reader["ФИО"].ToString();
                address.Value = reader["Адрес"].ToString();
                telephone.Value = reader["Телефон"].ToString();
                login.Value = "";
                password.Value = "";
                row.Cells.AddRange(id, post, branch, fio, address, telephone, login,
password);
                dataGridViewEmployee.Rows.Add(row);
                j++;
            }
            reader.Close();

            for (int i = 0; i < dataGridViewEmployee.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNamePost @Должности, @Название1
OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Должности", SqlDbType.Int).Value =
dataGridViewEmployee[1, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название1", SqlDbType.VarChar,
30));
                cmd.Parameters["@Название1"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewEmployee[1, i].Value =
cmd.Parameters["@Название1"].Value.ToString();
            }

            for (int i = 0; i < dataGridViewEmployee.RowCount; i++)
            {
```

```
                cmd.CommandText = "EXECUTE HPGiveNameBranch @Филиалы, @Название
OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Филиалы", SqlDbType.Int).Value =
dataGridViewEmployee[2, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название", SqlDbType.VarChar,
30));
                cmd.Parameters["@Название"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewEmployee[2, i].Value =
cmd.Parameters["@Название"].Value.ToString();
            }

            cmd.CommandText = "SELECT * FROM ПривилегированныеСотрудники";
            reader = cmd.ExecuteReader();
            j = 0;
            while (reader.Read())
            {
                string id = reader["ID_Сотрудники"].ToString();
                string login = reader["Логин"].ToString();
                string password = reader["Пароль"].ToString();
                for (int i = 0; i < dataGridViewEmployee.RowCount; i++)
                {
                    if (dataGridViewEmployee[0,
i].Value.ToString().Trim().CompareTo(id.Trim()) == 0)
                    {
                        dataGridViewEmployee[6, i].Value = login;
                        dataGridViewEmployee[7, i].Value = password;
                        j = i + 1;
                        break;
                    }
                }
            }
            reader.Close();
        }

        private void buttonEditEmployee_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPDeleteEmployee @ID_Сотрудники";
            cmd.Parameters.Add("@ID_Сотрудники", SqlDbType.Int).Value =
dataGridViewEmployee[0, dataGridViewEmployee.CurrentRow.Index].Value.ToString();
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Сотрудник успешно удален", "Удаление сотрудника",
MessageBoxButtons.OK);
                this.Close();
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка удаления сотрудника", "Удаление сотрудника",
MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Удаление
сотрудника", MessageBoxButtons.OK);
            }
        }
    }
}
```

FormDeletePost.cs

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

namespace _43_Apteka
{
    public partial class FormDeletePost : Form
    {
        public FormDeletePost()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Должности";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell name = new DataGridViewTextBoxCell();
                DataGridViewCell salary = new DataGridViewTextBoxCell();
                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Должности"].ToString();
                name.Value = reader["Название"].ToString();
                salary.Value = reader["Зарплата"].ToString();
                row.Cells.AddRange(id, name, salary);
                dataGridViewPost.Rows.Add(row);
            }
            reader.Close();
        }

        private void buttonEditPost_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPDeletePost @ID_Должности";
            cmd.Parameters.Add("@ID_Должности", SqlDbType.Int).Value =
dataGridViewPost[0, dataGridViewPost.CurrentRow.Index].Value.ToString();
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Должность успешно удалена", "Удаление должности",
MessageBoxButtons.OK);
                this.Close();
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка удаления должности", "Удаление должности",
MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Удаление должности",
MessageBoxButtons.OK);
            }
        }
```

```
    }
}
```

FormDeletePreparation.cs

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

namespace _43_Apteka
{
    public partial class FormDeletePreparation : Form
    {
        public FormDeletePreparation()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Препараты";
            SqlDataReader reader = cmd.ExecuteReader();
            int j = 0;
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell category = new DataGridViewTextBoxCell();
                DataGridViewCell name = new DataGridViewTextBoxCell();
                DataGridViewCell price = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Препараты"].ToString();
                category.Value = reader["Категории"].ToString();
                name.Value = reader["Название"].ToString();
                price.Value = reader["Цена"].ToString();
                row.Cells.AddRange(id, category, name, price);
                dataGridViewPreparation.Rows.Add(row);
                j++;
            }
            reader.Close();

            for (int i = 0; i < dataGridViewPreparation.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNameCategory @Категории, @Название
OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Категории", SqlDbType.Int).Value =
dataGridViewPreparation[1, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название", SqlDbType.VarChar,
50));
                cmd.Parameters["@Название"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewPreparation[1, i].Value =
cmd.Parameters["@Название"].Value.ToString();
            }
        }

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

```
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPDeletePreparation @ID_Препараты";
            cmd.Parameters.Add("@ID_Препараты", SqlDbType.Int).Value =
dataGridViewPreparation[0,
dataGridViewPreparation.CurrentRow.Index].Value.ToString();
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Препарат успешно удален", "Обновление удален",
MessageBoxButtons.OK);
                this.Close();
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка удаления препарата", "Удаление препарата",
MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Удаление препарата",
MessageBoxButtons.OK);
            }
        }
    }
}
```

FormDeleteSupplier.cs

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

namespace _43_Apteka
{
    public partial class FormDeleteSupplier : Form
    {
        public FormDeleteSupplier()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Поставщики";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell category = new DataGridViewTextBoxCell();
                DataGridViewCell fio = new DataGridViewTextBoxCell();
                DataGridViewCell address = new DataGridViewTextBoxCell();
                DataGridViewCell telephone = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Поставщики"].ToString();
                category.Value = reader["Категории"].ToString();
                fio.Value = reader["ФИО"].ToString();
                address.Value = reader["Адрес"].ToString();
                telephone.Value = reader["Телефон"].ToString();
```

```
                row.Cells.AddRange(id, fio, address, telephone);
                dataGridViewSupplier.Rows.Add(row);
            }
            reader.Close();

            for (int i = 0; i < dataGridViewSupplier.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNameCategory @Категории, @Название
OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Категории", SqlDbType.Int).Value =
dataGridViewSupplier[1, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название", SqlDbType.VarChar,
50));
                cmd.Parameters["@Название"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewSupplier[1, i].Value =
cmd.Parameters["@Название"].Value.ToString();
            }
        }

        private void buttonDeleteSupplier_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPDeleteSupplier @ID_Поставщики";
            cmd.Parameters.Add("@ID_Поставщики", SqlDbType.Int).Value =
dataGridViewSupplier[0, dataGridViewSupplier.CurrentRow.Index].Value.ToString();
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Поставщик успешно удален", "Удаление поставщика",
MessageBoxButtons.OK);
                this.Close();
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка удаления поставщика", "Удаление поставщика",
MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Удаление
поставщика", MessageBoxButtons.OK);
            }
        }
    }
}
```

FormDeleteWarehouse.cs

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

namespace _43_Apteka
{
```

```
    public partial class FormDeleteWarehouse : Form
    {
        public FormDeleteWarehouse()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Склад";
            SqlDataReader reader = cmd.ExecuteReader();
            int j = 0;
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell branch = new DataGridViewTextBoxCell();
                DataGridViewCell preparation = new DataGridViewTextBoxCell();
                DataGridViewCell count = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Склад"].ToString();
                branch.Value = reader["Филиалы"].ToString();
                preparation.Value = reader["Препараты"].ToString();
                count.Value = reader["Количество"].ToString();
                row.Cells.AddRange(id, branch, preparation, count);
                dataGridViewWarehouse.Rows.Add(row);
                j++;
            }
            reader.Close();

            for (int i = 0; i < dataGridViewWarehouse.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNameBranch @Филиалы, @Название
OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Филиалы", SqlDbType.Int).Value =
dataGridViewWarehouse[1, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название", SqlDbType.VarChar,
30));
                cmd.Parameters["@Название"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewWarehouse[1, i].Value =
cmd.Parameters["@Название"].Value.ToString();
            }

            for (int i = 0; i < dataGridViewWarehouse.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNamePreparation @Препараты,
@Название1 OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Препараты", SqlDbType.Int).Value =
dataGridViewWarehouse[2, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название1", SqlDbType.VarChar,
30));
                cmd.Parameters["@Название1"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewWarehouse[2, i].Value =
cmd.Parameters["@Название1"].Value.ToString();
            }
        }

        private void buttonDeleteWarehouse_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPDeleteWareHouse @ID_Склад";
```

```
            cmd.Parameters.Add("@ID_Склад", SqlDbType.Int).Value =
dataGridViewWarehouse[0, dataGridViewWarehouse.CurrentRow.Index].Value.ToString();
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Склад успешно удален", "Удаление склада",
MessageBoxButtons.OK);
                this.Close();
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка удаления склада", "Удаление склада",
MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Удаление склада",
MessageBoxButtons.OK);
            }
        }
    }
}
```

FormEditBranch.cs

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

namespace _43_Apteka
{
    public partial class FormEditBranch : Form
    {
        public FormEditBranch()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Филиалы";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell name = new DataGridViewTextBoxCell();
                DataGridViewCell address = new DataGridViewTextBoxCell();
                DataGridViewCell telephone = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Филиалы"].ToString().Trim();
                name.Value = reader["Название"].ToString().Trim();
                address.Value = reader["Адрес"].ToString().Trim();
                telephone.Value = reader["Телефон"].ToString().Trim();
                row.Cells.AddRange(id, name, address, telephone);
                dataGridViewBranch.Rows.Add(row);
            }
            reader.Close();
            textBoxID.Text = dataGridViewBranch[0, 0].Value.ToString();
```

```
            textBoxName.Text = dataGridViewBranch[1, 0].Value.ToString();
            textBoxAddress.Text = dataGridViewBranch[2, 0].Value.ToString();
            textBoxTelephone.Text = dataGridViewBranch[3, 0].Value.ToString();
        }

        private void dataGridViewBranch_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBoxID.Text = dataGridViewBranch[0,
dataGridViewBranch.CurrentRow.Index].Value.ToString();
            textBoxName.Text = dataGridViewBranch[1,
dataGridViewBranch.CurrentRow.Index].Value.ToString();
            textBoxAddress.Text = dataGridViewBranch[2,
dataGridViewBranch.CurrentRow.Index].Value.ToString();
            textBoxTelephone.Text = dataGridViewBranch[3,
dataGridViewBranch.CurrentRow.Index].Value.ToString();
        }

        private void buttonEditBranch_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPUpdateBranch @ID_Филиалы, @Название, @Адрес,
@Телефон";
            cmd.Parameters.Add("@ID_Филиалы", SqlDbType.Int).Value = textBoxID.Text;
            cmd.Parameters.Add("@Название", SqlDbType.Char).Value = textBoxName.Text;
            cmd.Parameters.Add("@Адрес", SqlDbType.Char).Value = textBoxAddress.Text;
            cmd.Parameters.Add("@Телефон", SqlDbType.Int).Value =
textBoxTelephone.Text;
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Филиал успешно обновлен", "Обновление филиала",
MessageBoxButtons.OK);
                this.Close();
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка обновления филиала", "Обновление филиала",
MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Обновление филиала",
MessageBoxButtons.OK);
            }
            catch (FormatException ex)
            {
                MessageBox.Show("Корректно заполните все необходимые поля",
"Обновление филиала", MessageBoxButtons.OK);
            }
        }
    }
}
```

FormEditBuyer.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace _43_Apteka
{
    public partial class FormEditBuyer : Form
    {
        int[] discounts = new int[150];
        public FormEditBuyer()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Покупатели";
            SqlDataReader reader = cmd.ExecuteReader();
            int j = 0;
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell discount = new DataGridViewTextBoxCell();
                DataGridViewCell fio = new DataGridViewTextBoxCell();
                DataGridViewCell sumBuy = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Покупатели"].ToString();
                discount.Value = reader["Скидки"].ToString();
                discounts[j] = Convert.ToInt32(discount.Value);
                fio.Value = reader["ФИО"].ToString().Trim();
                sumBuy.Value = reader["ОбщаяСуммаПокупок"].ToString().Trim();
                row.Cells.AddRange(id, discount, fio, sumBuy);
                dataGridViewBuyer.Rows.Add(row);
                j++;
            }
            reader.Close();

            for (int i = 0; i < dataGridViewBuyer.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveSizeDiscount @Скидки, @Размер
OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Скидки", SqlDbType.Int).Value =
dataGridViewBuyer[1, i].Value.ToString();
                cmd.Parameters.Add("@Размер", SqlDbType.Int);
                cmd.Parameters["@Размер"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewBuyer[1, i].Value =
cmd.Parameters["@Размер"].Value.ToString().Trim();
            }

            cmd.CommandText = "SELECT Размер FROM Скидки";
            reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                comboBoxDiscount.Items.Add(reader["Размер"].ToString());
            }
            reader.Close();

            textBoxID.Text = dataGridViewBuyer[0, 0].Value.ToString();
            comboBoxDiscount.SelectedItem = comboBoxDiscount.Items[0];
            textBoxFIO.Text = dataGridViewBuyer[2, 0].Value.ToString();
            textBoxSumBuy.Text = dataGridViewBuyer[3, 0].Value.ToString();
        }
```

```
        private void buttonEditBuyer_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPUpdateBuyer @ID_Покупатели, @Скидки, @ФИО,
@ОбщаяСуммаПокупок";
            cmd.Parameters.Add("@ID_Покупатели", SqlDbType.Int).Value =
textBoxID.Text;
            cmd.Parameters.Add("@Скидки", SqlDbType.Int).Value =
comboBoxDiscount.SelectedIndex + 1;
            cmd.Parameters.Add("@ФИО", SqlDbType.Char).Value = textBoxFIO.Text;
            cmd.Parameters.Add("@ОбщаяСуммаПокупок", SqlDbType.Int).Value =
textBoxSumBuy.Text;
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Покупатель успешно обновлен", "Обновление
покупателя", MessageBoxButtons.OK);
                this.Close();
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка обновления покупателя", "Обновление
покупателя", MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Обновление
покупателя", MessageBoxButtons.OK);
            }
            catch (FormatException ex)
            {
                MessageBox.Show("Корректно заполните все необходимые поля",
"Обновление покупателя", MessageBoxButtons.OK);
            }
        }

        private void dataGridViewBuyer_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBoxID.Text = dataGridViewBuyer[0,
dataGridViewBuyer.CurrentRow.Index].Value.ToString();
            comboBoxDiscount.SelectedIndex =
discounts[dataGridViewBuyer.CurrentRow.Index] - 1;
            textBoxFIO.Text = dataGridViewBuyer[2,
dataGridViewBuyer.CurrentRow.Index].Value.ToString();
            textBoxSumBuy.Text = dataGridViewBuyer[3,
dataGridViewBuyer.CurrentRow.Index].Value.ToString();
        }
    }
}
```

FormEditCategory.cs

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

```
using System.Data.SqlClient;

namespace _43_Apteka
{
    public partial class FormEditCategory : Form
    {
        public FormEditCategory()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Категории";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell name = new DataGridViewTextBoxCell();
                DataGridViewRow row = new DataGridViewRow();

                id.Value = reader["ID_Категории"].ToString();
                name.Value = reader["Название"].ToString().Trim();
                row.Cells.AddRange(id, name);
                dataGridViewCategory.Rows.Add(row);
            }
            reader.Close();
            textBoxID.Text = dataGridViewCategory[0, 0].Value.ToString();
            textBoxName.Text = dataGridViewCategory[1, 0].Value.ToString();
        }

        private void dataGridViewCategory_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBoxID.Text = dataGridViewCategory[0,
dataGridViewCategory.CurrentRow.Index].Value.ToString();
            textBoxName.Text = dataGridViewCategory[1,
dataGridViewCategory.CurrentRow.Index].Value.ToString();
        }

        private void buttonEditPost_Click(object sender, EventArgs e)
        {
            if (textBoxName.Text == "")
            {
                MessageBox.Show("Корректно заполните все необходимые поля",
"Обновление категории", MessageBoxButtons.OK);
                return;
            }
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPUpdateCategory @ID_Категории, @Название";
            cmd.Parameters.Add("@ID_Категории", SqlDbType.Int).Value =
textBoxID.Text;
            cmd.Parameters.Add("@Название", SqlDbType.Char).Value = textBoxName.Text;
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Категория успешно обновлена", "Обновление
категории", MessageBoxButtons.OK);
                this.Close();
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка обновления категории", "Обновление
категории", MessageBoxButtons.OK);
            }
```

```
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Обновление
категории", MessageBoxButtons.OK);
            }
        }
    }
}
```

FormEditDiscount.cs

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

namespace _43_Apteka
{
    public partial class FormEditDiscount : Form
    {
        public FormEditDiscount()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Скидки";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell name = new DataGridViewTextBoxCell();
                DataGridViewCell size = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Скидки"].ToString();
                name.Value = reader["Название"].ToString();
                size.Value = reader["Размер"].ToString();
                row.Cells.AddRange(id, name, size);
                dataGridViewDiscount.Rows.Add(row);
            }
            reader.Close();
            textBoxID.Text = dataGridViewDiscount[0, 0].Value.ToString().Trim();
            textBoxName.Text = dataGridViewDiscount[1, 0].Value.ToString().Trim();
            textBoxSize.Text = dataGridViewDiscount[2, 0].Value.ToString().Trim();
        }

        private void dataGridViewDiscount_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBoxID.Text = dataGridViewDiscount[0,
dataGridViewDiscount.CurrentRow.Index].Value.ToString().Trim();
            textBoxName.Text = dataGridViewDiscount[1,
dataGridViewDiscount.CurrentRow.Index].Value.ToString().Trim();
            textBoxSize.Text = dataGridViewDiscount[2,
dataGridViewDiscount.CurrentRow.Index].Value.ToString().Trim();
        }

        private void buttonEditDiscount_Click(object sender, EventArgs e)
```

```
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPUpdateDiscount @ID_Скидки, @Название,
@Размер";
            cmd.Parameters.Add("@ID_Скидки", SqlDbType.Int).Value =
textBoxID.Text.Trim();
            cmd.Parameters.Add("@Название", SqlDbType.Char).Value =
textBoxName.Text.Trim();
            cmd.Parameters.Add("@Размер", SqlDbType.Int).Value =
textBoxSize.Text.Trim();
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Скидка успешно обновлена", "Обновление скидки",
MessageBoxButtons.OK);
                this.Close();
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка обновления скидки", "Обновление скидки",
MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Обновление скидки",
MessageBoxButtons.OK);
            }
            catch (FormatException ex)
            {
                MessageBox.Show("Корректно заполните все необходимые поля",
"Обновление скидки", MessageBoxButtons.OK);
            }
        }
    }
}
```

FormEditEmployee.cs

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

namespace _43_Apteka
{
    public partial class FormEditEmployee : Form
    {
        int[] posts = new int[150];
        int[] branches = new int[150];

        public FormEditEmployee()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Сотрудники";
            SqlDataReader reader = cmd.ExecuteReader();
            int j = 0;
```

```
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell post = new DataGridViewTextBoxCell();
                DataGridViewCell branch = new DataGridViewTextBoxCell();
                DataGridViewCell fio = new DataGridViewTextBoxCell();
                DataGridViewCell address = new DataGridViewTextBoxCell();
                DataGridViewCell telephone = new DataGridViewTextBoxCell();
                DataGridViewCell login = new DataGridViewTextBoxCell();
                DataGridViewCell password = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Сотрудники"].ToString();
                post.Value = reader["Должности"].ToString();
                posts[j] = Convert.ToInt32(post.Value);
                branch.Value = reader["Филиалы"].ToString();
                branches[j] = Convert.ToInt32(branch.Value);
                fio.Value = reader["ФИО"].ToString().Trim();
                address.Value = reader["Адрес"].ToString().Trim();
                telephone.Value = reader["Телефон"].ToString();
                login.Value = "";
                password.Value = "";
                row.Cells.AddRange(id, post, branch, fio, address, telephone, login,
password);
                dataGridViewEmployee.Rows.Add(row);
                j++;
            }
            reader.Close();

            for (int i = 0; i < dataGridViewEmployee.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNamePost @Должности, @Название1
OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Должности", SqlDbType.Int).Value =
dataGridViewEmployee[1, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название1", SqlDbType.VarChar,
30));
                cmd.Parameters["@Название1"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewEmployee[1, i].Value =
cmd.Parameters["@Название1"].Value.ToString().Trim();
            }

            for (int i = 0; i < dataGridViewEmployee.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNameBranch @Филиалы, @Название
OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Филиалы", SqlDbType.Int).Value =
dataGridViewEmployee[2, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название", SqlDbType.VarChar,
30));
                cmd.Parameters["@Название"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewEmployee[2, i].Value =
cmd.Parameters["@Название"].Value.ToString().Trim();
            }

            cmd.CommandText = "SELECT * FROM ПривилегированныеСотрудники";
            reader = cmd.ExecuteReader();
            j = 0;
```

```
            while (reader.Read())
            {
                string id = reader["ID_Сотрудники"].ToString();
                string login = reader["Логин"].ToString();
                string password = reader["Пароль"].ToString();
                for (int i = 0; i < dataGridViewEmployee.RowCount; i++)
                {
                    if (dataGridViewEmployee[0,
i].Value.ToString().Trim().CompareTo(id.Trim()) == 0)
                    {
                        dataGridViewEmployee[6, i].Value = login.Trim();
                        dataGridViewEmployee[7, i].Value = password.Trim();
                        j = i + 1;
                        break;
                    }
                }
            }
            reader.Close();

            cmd.CommandText = "SELECT Название FROM Должности";
            reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                comboBoxPost.Items.Add(reader["Название"].ToString().Trim());
            }
            reader.Close();

            cmd.CommandText = "SELECT Название FROM Филиалы";
            reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                comboBoxBranch.Items.Add(reader["Название"].ToString().Trim());
            }
            reader.Close();

            textBoxID.Text = dataGridViewEmployee[0, 0].Value.ToString();
            comboBoxBranch.SelectedItem = comboBoxBranch.Items[0];
            comboBoxPost.SelectedItem = comboBoxPost.Items[0];
            textBoxFIO.Text = dataGridViewEmployee[3, 0].Value.ToString();
            textBoxAddress.Text = dataGridViewEmployee[4, 0].Value.ToString();
            textBoxTelephone.Text = dataGridViewEmployee[5, 0].Value.ToString();
            textBoxLogin.Text = dataGridViewEmployee[6, 0].Value.ToString();
            textBoxPassword.Text = dataGridViewEmployee[7, 0].Value.ToString();
        }

        private void buttonEditEmployee_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPUpdateEmployee @ID_Сотрудники, @Должности,
@Филиалы, @ФИО, @Адрес, @Телефон, @Логин, @Пароль";
            cmd.Parameters.Add("@ID_Сотрудники", SqlDbType.Int).Value =
textBoxID.Text;
            cmd.Parameters.Add("@Должности", SqlDbType.Int).Value =
comboBoxPost.SelectedIndex + 1;
            cmd.Parameters.Add("@Филиалы", SqlDbType.Int).Value =
comboBoxBranch.SelectedIndex + 1;
            cmd.Parameters.Add("@ФИО", SqlDbType.Char).Value = textBoxFIO.Text;
            cmd.Parameters.Add("@Адрес", SqlDbType.Char).Value = textBoxAddress.Text;
            cmd.Parameters.Add("@Телефон", SqlDbType.Int).Value =
textBoxTelephone.Text;
            if (panelIDPassword.Enabled == true)
            {
```

```
                cmd.Parameters.Add("@Логин", SqlDbType.VarChar, 30).Value =
textBoxLogin.Text;
                cmd.Parameters.Add("@Пароль", SqlDbType.VarChar, 30).Value =
textBoxPassword.Text;
            }
            else
            {
                cmd.Parameters.Add("@Логин", SqlDbType.VarChar, 30).Value = "";
                cmd.Parameters.Add("@Пароль", SqlDbType.VarChar, 30).Value = "";
            }
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Сотрудник успешно обновлен", "Обновление
сотрудника", MessageBoxButtons.OK);
                this.Close();
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка обновления сотрудника", "Обновление
сотрудника", MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Обновление
сотрудника", MessageBoxButtons.OK);
            }
            catch (FormatException ex)
            {
                MessageBox.Show("Корректно заполните все необходимые поля",
"Обновление сотрудника", MessageBoxButtons.OK);
            }
        }

        private void dataGridViewEmployee_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBoxID.Text = dataGridViewEmployee[0,
dataGridViewEmployee.CurrentRow.Index].Value.ToString();
            comboBoxPost.SelectedIndex = posts[dataGridViewEmployee.CurrentRow.Index]
- 1;
            comboBoxBranch.SelectedIndex =
branches[dataGridViewEmployee.CurrentRow.Index] - 1;
            textBoxFIO.Text = dataGridViewEmployee[3,
dataGridViewEmployee.CurrentRow.Index].Value.ToString();
            textBoxAddress.Text = dataGridViewEmployee[4,
dataGridViewEmployee.CurrentRow.Index].Value.ToString();
            textBoxTelephone.Text = dataGridViewEmployee[5,
dataGridViewEmployee.CurrentRow.Index].Value.ToString();
            textBoxLogin.Text = dataGridViewEmployee[6,
dataGridViewEmployee.CurrentRow.Index].Value.ToString();
            textBoxPassword.Text = dataGridViewEmployee[7,
dataGridViewEmployee.CurrentRow.Index].Value.ToString();

            for (int i = 0; i < dataGridViewEmployee.RowCount; i++)
            {
                if
((comboBoxPost.SelectedItem.ToString().Trim().CompareTo("Администратор") == 0) ||
(comboBoxPost.SelectedItem.ToString().Trim().CompareTo("Кассир-консультант") == 0))
                {
                    panelIDPassword.Enabled = true;
                }
```

```
                else
                {
                    panelIDPassword.Enabled = false;
                }
            }
        }
    }
}
```

FormEditPost.cs

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

namespace _43_Apteka
{
    public partial class FormEditPost : Form
    {
        public FormEditPost()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Должности";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell name = new DataGridViewTextBoxCell();
                DataGridViewCell salary = new DataGridViewTextBoxCell();
                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Должности"].ToString().Trim();
                name.Value = reader["Название"].ToString().Trim();
                salary.Value = reader["Зарплата"].ToString().Trim();
                row.Cells.AddRange(id, name, salary);
                dataGridViewPost.Rows.Add(row);
            }
            reader.Close();
            textBoxID.Text = dataGridViewPost[0, 0].Value.ToString();
            textBoxName.Text = dataGridViewPost[1, 0].Value.ToString();
            textBoxSalary.Text = dataGridViewPost[2, 0].Value.ToString();
        }

        private void dataGridViewPost_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBoxID.Text = dataGridViewPost[0,
dataGridViewPost.CurrentRow.Index].Value.ToString();
            textBoxName.Text = dataGridViewPost[1,
dataGridViewPost.CurrentRow.Index].Value.ToString();
            textBoxSalary.Text = dataGridViewPost[2,
dataGridViewPost.CurrentRow.Index].Value.ToString();
        }

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

```
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPUpdatePost @ID_Должности, @Название,
@Зарплата";
            cmd.Parameters.Add("@ID_Должности", SqlDbType.Int).Value =
textBoxID.Text;
            cmd.Parameters.Add("@Название", SqlDbType.Char).Value = textBoxName.Text;
            cmd.Parameters.Add("@Зарплата", SqlDbType.Int).Value =
textBoxSalary.Text;
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Должность успешно обновлена", "Обновление
должности", MessageBoxButtons.OK);
                this.Close();
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка обновления должности", "Обновление
должности", MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Обновление
должности", MessageBoxButtons.OK);
            }
            catch (FormatException ex)
            {
                MessageBox.Show("Корректно заполните все необходимые поля",
"Обновление должности", MessageBoxButtons.OK);
            }
        }
    }
}
```

FormEditPreparation.cs

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

namespace _43_Apteka
{
    public partial class FormEditPreparation : Form
    {
        int[] categories = new int[150];
        public FormEditPreparation()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Препараты";
            SqlDataReader reader = cmd.ExecuteReader();
            int j = 0;
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell category = new DataGridViewTextBoxCell();
```

```
                DataGridViewCell name = new DataGridViewTextBoxCell();
                DataGridViewCell price = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Препараты"].ToString();
                category.Value = reader["Категории"].ToString();
                categories[j] = Convert.ToInt32(category.Value);
                name.Value = reader["Название"].ToString();
                price.Value = reader["Цена"].ToString();
                row.Cells.AddRange(id, category, name, price);
                dataGridViewPreparation.Rows.Add(row);
                j++;
            }
            reader.Close();

            for (int i = 0; i < dataGridViewPreparation.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNameCategory @Категории, @Название
OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Категории", SqlDbType.Int).Value =
dataGridViewPreparation[1, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название", SqlDbType.VarChar,
50));
                cmd.Parameters["@Название"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewPreparation[1, i].Value =
cmd.Parameters["@Название"].Value.ToString().Trim();
            }

            cmd.CommandText = "SELECT Название FROM Категории";
            reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                comboBoxCategory.Items.Add(reader["Название"].ToString().Trim());
            }
            reader.Close();

            textBoxID.Text = dataGridViewPreparation[0, 0].Value.ToString().Trim();
            comboBoxCategory.SelectedItem = comboBoxCategory.Items[0];
            textBoxName.Text = dataGridViewPreparation[2, 0].Value.ToString().Trim();
            textBoxPrice.Text = dataGridViewPreparation[3,
0].Value.ToString().Trim();
        }

        private void buttonEditPreparation_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPUpdatePreparation @ID_Препараты, @Категории,
@Название, @Цена";
            cmd.Parameters.Add("@ID_Препараты", SqlDbType.Int).Value =
textBoxID.Text;
            cmd.Parameters.Add("@Категории", SqlDbType.Int).Value =
comboBoxCategory.SelectedIndex + 1;
            cmd.Parameters.Add("@Название", SqlDbType.Char).Value = textBoxName.Text;
            cmd.Parameters.Add("@Цена", SqlDbType.Int).Value = textBoxPrice.Text;
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Препарат успешно обновлен", "Обновление препарата",
MessageBoxButtons.OK);
                this.Close();
```

```
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка обновления препарата", "Обновление
препарата", MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Обновление
препарата", MessageBoxButtons.OK);
            }
            catch (FormatException ex)
            {
                MessageBox.Show("Корректно заполните все необходимые поля",
"Обновление препарата", MessageBoxButtons.OK);
            }
        }

        private void dataGridViewPreparation_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBoxID.Text = dataGridViewPreparation[0,
dataGridViewPreparation.CurrentRow.Index].Value.ToString().Trim();
            comboBoxCategory.SelectedIndex =
categories[dataGridViewPreparation.CurrentRow.Index] - 1;
            textBoxName.Text = dataGridViewPreparation[2,
dataGridViewPreparation.CurrentRow.Index].Value.ToString().Trim();
            textBoxPrice.Text = dataGridViewPreparation[3,
dataGridViewPreparation.CurrentRow.Index].Value.ToString().Trim();
        }
    }
}
```

FormEditSupplier.cs

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

namespace _43_Apteka
{
    public partial class FormEditSupplier : Form
    {
        int[] categories = new int[150];
        public FormEditSupplier()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Поставщики";
            SqlDataReader reader = cmd.ExecuteReader();
            int j = 0;
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell category = new DataGridViewTextBoxCell();
                DataGridViewCell fio = new DataGridViewTextBoxCell();
```

```
                DataGridViewCell address = new DataGridViewTextBoxCell();
                DataGridViewCell telephone = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Поставщики"].ToString();
                category.Value = reader["Категории"].ToString();
                categories[j] = Convert.ToInt32(category.Value);
                fio.Value = reader["ФИО"].ToString();
                address.Value = reader["Адрес"].ToString();
                telephone.Value = reader["Телефон"].ToString();
                row.Cells.AddRange(id, category, fio, address, telephone);
                dataGridViewSupplier.Rows.Add(row);
                j++;
            }
            reader.Close();

            for (int i = 0; i < dataGridViewSupplier.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNameCategory @Категории, @Название
OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Категории", SqlDbType.Int).Value =
dataGridViewSupplier[1, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название", SqlDbType.VarChar,
50));
                cmd.Parameters["@Название"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewSupplier[1, i].Value =
cmd.Parameters["@Название"].Value.ToString().Trim();
            }

            cmd.CommandText = "SELECT Название FROM Категории";
            reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                comboBoxCategory.Items.Add(reader["Название"].ToString().Trim());
            }
            reader.Close();

            textBoxID.Text = dataGridViewSupplier[0, 0].Value.ToString();
            comboBoxCategory.SelectedItem = comboBoxCategory.Items[0];
            textBoxFIO.Text = dataGridViewSupplier[2, 0].Value.ToString().Trim();
            textBoxAddress.Text = dataGridViewSupplier[3, 0].Value.ToString().Trim();
            textBoxTelephone.Text = dataGridViewSupplier[4,
0].Value.ToString().Trim();
        }

        private void dataGridViewSupplier_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBoxID.Text = dataGridViewSupplier[0,
dataGridViewSupplier.CurrentRow.Index].Value.ToString().Trim();
            comboBoxCategory.SelectedIndex =
categories[dataGridViewSupplier.CurrentRow.Index] - 1;
            textBoxFIO.Text = dataGridViewSupplier[2,
dataGridViewSupplier.CurrentRow.Index].Value.ToString().Trim();
            textBoxAddress.Text = dataGridViewSupplier[3,
dataGridViewSupplier.CurrentRow.Index].Value.ToString().Trim();
            textBoxTelephone.Text = dataGridViewSupplier[4,
dataGridViewSupplier.CurrentRow.Index].Value.ToString().Trim();
        }
```

```
        private void buttonEditSupplier_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPUpdateSupplier @ID_Поставщики, @Категории,
@ФИО, @Адрес, @Телефон";
            cmd.Parameters.Add("@ID_Поставщики", SqlDbType.Int).Value =
textBoxID.Text;
            cmd.Parameters.Add("@Категории", SqlDbType.Int).Value =
comboBoxCategory.SelectedIndex + 1;
            cmd.Parameters.Add("@ФИО", SqlDbType.Char).Value = textBoxFIO.Text;
            cmd.Parameters.Add("@Адрес", SqlDbType.Char).Value = textBoxAddress.Text;
            cmd.Parameters.Add("@Телефон", SqlDbType.Int).Value =
textBoxTelephone.Text;
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Поставщик успешно обновлен", "Обновление
поставщика", MessageBoxButtons.OK);
                this.Close();
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка обновления поставщика", "Обновление
поставщика", MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Обновление
поставщика", MessageBoxButtons.OK);
            }
            catch (FormatException ex)
            {
                MessageBox.Show("Корректно заполните все необходимые поля",
"Обновление поставщика", MessageBoxButtons.OK);
            }
        }
    }
}
```

FormEditWarehouse.cs

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

namespace _43_Apteka
{
    public partial class FormEditWarehouse : Form
    {
        int[] branches = new int[50];
        int[] preparations = new int[50];
        public FormEditWarehouse()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Склад";
```

```
            SqlDataReader reader = cmd.ExecuteReader();
            int j = 0;
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell branch = new DataGridViewTextBoxCell();
                DataGridViewCell preparation = new DataGridViewTextBoxCell();
                DataGridViewCell count = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Склад"].ToString();
                branch.Value = reader["Филиалы"].ToString();
                branches[j] = Convert.ToInt32(branch.Value);
                preparation.Value = reader["Препараты"].ToString();
                preparations[j] = Convert.ToInt32(preparation.Value);
                count.Value = reader["Количество"].ToString();
                row.Cells.AddRange(id, branch, preparation, count);
                dataGridViewWarehouse.Rows.Add(row);
                j++;
            }
            reader.Close();

          for (int i = 0; i < dataGridViewWarehouse.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNameBranch @Филиалы, @Название
OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Филиалы", SqlDbType.Int).Value =
dataGridViewWarehouse[1, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название", SqlDbType.VarChar,
30));
                cmd.Parameters["@Название"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewWarehouse[1, i].Value =
cmd.Parameters["@Название"].Value.ToString().Trim();
            }

            for (int i = 0; i < dataGridViewWarehouse.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNamePreparation @Препараты,
@Название1 OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Препараты", SqlDbType.Int).Value =
dataGridViewWarehouse[2, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название1", SqlDbType.VarChar,
30));
                cmd.Parameters["@Название1"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewWarehouse[2, i].Value =
cmd.Parameters["@Название1"].Value.ToString().Trim();
            }

            cmd.CommandText = "SELECT Название FROM Филиалы";
            reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                comboBoxBranch.Items.Add(reader["Название"].ToString().Trim());
            }
            reader.Close();

            cmd.CommandText = "SELECT Название FROM Препараты";
            reader = cmd.ExecuteReader();
```

```
            while (reader.Read())
            {
                comboBoxPreparation.Items.Add(reader["Название"].ToString().Trim());
            }
            reader.Close();

            textBoxID.Text = dataGridViewWarehouse[0, 0].Value.ToString();
            comboBoxBranch.SelectedItem = comboBoxBranch.Items[0];
            comboBoxPreparation.SelectedItem = comboBoxPreparation.Items[0];
            textBoxCount.Text = dataGridViewWarehouse[3, 0].Value.ToString();
        }

        private void buttonEditWarehouse_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPUpdateWareHouse @ID_Склад, @Филиалы,
@Препараты, @Количество";
            cmd.Parameters.Add("@ID_Склад", SqlDbType.Int).Value = textBoxID.Text;
            cmd.Parameters.Add("@Филиалы", SqlDbType.Int).Value =
comboBoxBranch.SelectedIndex + 1;
            cmd.Parameters.Add("@Препараты", SqlDbType.Int).Value =
comboBoxPreparation.SelectedIndex + 1;
            cmd.Parameters.Add("@Количество", SqlDbType.Int).Value =
textBoxCount.Text;
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Склад успешно обновлен", "Обновление склада",
MessageBoxButtons.OK);
                this.Close();
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка обновления склада", "Обновление склада",
MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Обновление склада",
MessageBoxButtons.OK);
            }
            catch (FormatException ex)
            {
                MessageBox.Show("Корректно заполните все необходимые поля",
"Обновление склада", MessageBoxButtons.OK);
            }
        }

        private void dataGridViewWarehouse_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBoxID.Text = dataGridViewWarehouse[0,
dataGridViewWarehouse.CurrentRow.Index].Value.ToString();
            comboBoxBranch.SelectedIndex =
branches[dataGridViewWarehouse.CurrentRow.Index] - 1;
            comboBoxPreparation.SelectedIndex =
preparations[dataGridViewWarehouse.CurrentRow.Index] - 1;
            textBoxCount.Text = dataGridViewWarehouse[3,
dataGridViewWarehouse.CurrentRow.Index].Value.ToString();
        }
    }
}
```

FormMain.cs

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

namespace _43_Apteka
{
    public partial class FormMain : Form
    {
        public static SqlConnection conn = new SqlConnection();

        public FormMain()
        {
            InitializeComponent();
        }

        private void buttonConnect_Click(object sender, EventArgs e)
        {
            ConnectionForm conForm = new ConnectionForm();
            conForm.ShowDialog();
        }

        private void buttonLogIn_Click(object sender, EventArgs e)
        {
            string login = textBoxLogin.Text;
            string password = textBoxPassword.Text;
            try
            {
                if (radioButtonAdmin.Checked == true)
                {
                    SqlCommand cmd = FormMain.conn.CreateCommand();
                    cmd.CommandText = "EXECUTE HPCheckAdmin @Логин, @Пароль, @Код
OUTPUT";
                    cmd.Parameters.Add("@Логин", SqlDbType.Char).Value = login;
                    cmd.Parameters.Add("@Пароль", SqlDbType.Char).Value = password;
                    cmd.Parameters.Add(new SqlParameter("@Код", SqlDbType.Int));
                    cmd.Parameters["@Код"].Direction = ParameterDirection.Output;
                    cmd.ExecuteNonQuery();
                    int code =
Convert.ToInt32(cmd.Parameters["@Код"].Value.ToString().Trim());
                    if (code == 0)
                    {
                        this.Hide();
                        FormAdminPage formAdminPage = new FormAdminPage();
                        if (formAdminPage.ShowDialog() == DialogResult.Cancel)
                        {
                            this.Show();
                        }
                    }
                    else
                    {
                        MessageBox.Show("Неверная пара логин-пароль", "Ошибка входа",
MessageBoxButtons.OK);
                    }
```

```
                }
                else
                {
                    if (radioButtonSeller.Checked == true)
                    {
                        SqlCommand cmd = FormMain.conn.CreateCommand();
                        cmd.CommandText = "EXECUTE HPCheckAdmin @Логин, @Пароль, @Код
OUTPUT";
                        cmd.Parameters.Add("@Логин", SqlDbType.Char).Value = login;
                        cmd.Parameters.Add("@Пароль", SqlDbType.Char).Value =
password;
                        cmd.Parameters.Add(new SqlParameter("@Код", SqlDbType.Int));
                        cmd.Parameters["@Код"].Direction = ParameterDirection.Output;
                        cmd.ExecuteNonQuery();
                        int code =
Convert.ToInt32(cmd.Parameters["@Код"].Value.ToString().Trim());
                        if (code == 0)
                        {
                            this.Hide();
                            FormSellerPage formSellerPage = new FormSellerPage();
                            if (formSellerPage.ShowDialog() == DialogResult.Cancel)
                            {
                                this.Show();
                            }
                        }
                        else
                        {
                            MessageBox.Show("Неверная пара логин-пароль", "Ошибка
входа", MessageBoxButtons.OK);
                        }
                    }
                    else
                    {
                        this.Hide();
                        FormBuyerPage formBuyerPage = new FormBuyerPage(login);
                        if (formBuyerPage.ShowDialog() == DialogResult.Cancel)
                        {
                            this.Show();
                        }
                    }
                }
            }
            catch (InvalidOperationException ex)
            {
                this.buttonConnect_Click(sender, e);
                this.Show();
                this.buttonLogIn_Click(sender, e);
            }

        }

        private void buttonAddBranch_Click(object sender, EventArgs e)
        {
            this.Hide();
            FormNewBranch formNewBranch = new FormNewBranch();
            formNewBranch.Show();
        }

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

```
        private void radioButtonBuyer_CheckedChanged(object sender, EventArgs e)
        {
            label1.Text = "ФИО";
            label2.Enabled = false;
            textBoxPassword.Enabled = false;
        }

        private void radioButtonAdmin_CheckedChanged(object sender, EventArgs e)
        {
            label1.Text = "Логин";
            label2.Enabled = true;
            textBoxPassword.Enabled = true;
        }
    }
}
```

FormNewBranch.cs

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

namespace _43_Apteka
{
    public partial class FormNewBranch : Form
    {
        public FormNewBranch()
        {
            InitializeComponent();
        }

        private void buttonAddBranch_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPAddBranch @Название, @Адрес, @Телефон";
            cmd.Parameters.Add("@Название", SqlDbType.Char).Value = textBoxName.Text;
            cmd.Parameters.Add("@Адрес", SqlDbType.Char).Value = textBoxAddress.Text;
            cmd.Parameters.Add("@Телефон", SqlDbType.Int).Value =
textBoxTelephon.Text;
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Филиал успешно добавлен", "Добавление филиала",
MessageBoxButtons.OK);
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка добавления филиала", "Добавление филиала",
MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Добавление филиала",
MessageBoxButtons.OK);
            }
```

```
            catch (FormatException ex)
            {
                MessageBox.Show("Корректно заполните все необходимые поля",
"Добавление филиала", MessageBoxButtons.OK);
            }
        }
    }
}
```

FormNewBuyer.cs

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

namespace _43_Apteka
{
    public partial class FormNewBuyer : Form
    {
        public FormNewBuyer()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT Название FROM Скидки";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                comboBoxBuyer.Items.Add(reader["Название"].ToString());
            }
            reader.Close();
        }

        private void buttonAddBuyer_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPAddBuyer @Скидки, @ФИО, @ОбщаяСуммаПокупок";
            cmd.Parameters.Add("@Скидки", SqlDbType.Char).Value =
comboBoxBuyer.SelectedItem.ToString();
            cmd.Parameters.Add("@ФИО", SqlDbType.Char).Value = textBoxFIO.Text;
            cmd.Parameters.Add("@ОбщаяСуммаПокупок", SqlDbType.Int).Value =
textBoxSumBuy.Text;
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Покупатель успешно добавлен", "Добавление
покупателя", MessageBoxButtons.OK);
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка добавления покупателя", "Добавление
покупателя", MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Добавление
покупателя", MessageBoxButtons.OK);
```

```
            }
            catch (FormatException ex)
            {
                MessageBox.Show("Корректно заполните все необходимые поля",
"Добавление покупателя", MessageBoxButtons.OK);
            }
        }

        private void buttonExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

FormNewCategory.cs

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

namespace _43_Apteka
{
    public partial class FormNewCategory : Form
    {
        public FormNewCategory()
        {
            InitializeComponent();
        }

        private void buttonAddPost_Click(object sender, EventArgs e)
        {
            if (textBoxName.Text == "")
            {
                MessageBox.Show("Корректно заполните все необходимые поля",
"Добавление категории", MessageBoxButtons.OK);
                return;
            }
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPAddCategory @Название";
            cmd.Parameters.Add("@Название", SqlDbType.Char).Value = textBoxName.Text;
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Категория успешно добавлена", "Добавление
категории", MessageBoxButtons.OK);
                this.Close();
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка добавления категории", "Добавление
категории", MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
```

```
                MessageBox.Show("Ошибка подключения к серверу", "Добавление
категории", MessageBoxButtons.OK);
            }
        }

        private void buttonExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

FormNewDiscount.cs

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

namespace _43_Apteka
{
    public partial class FormNewDiscount : Form
    {
        public FormNewDiscount()
        {
            InitializeComponent();
        }

        private void buttonAddDiscount_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPAddDiscount @Название, @Размер";
            cmd.Parameters.Add("@Название", SqlDbType.Char).Value = textBoxName.Text;
            cmd.Parameters.Add("@Размер", SqlDbType.Int).Value = textBoxSize.Text;
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Скидка успешно добавлена", "Добавление скидки",
MessageBoxButtons.OK);
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка добавления скидки", "Добавление скидки",
MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Добавление скидки",
MessageBoxButtons.OK);
            }
            catch (FormatException ex)
            {
                MessageBox.Show("Корректно заполните все необходимые поля",
"Добавление скидки", MessageBoxButtons.OK);
            }
        }
```

```
        private void buttonExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

FormNewEmployee.cs

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

namespace _43_Apteka
{
    public partial class FormNewEmployee : Form
    {
        public FormNewEmployee()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT Название FROM Должности";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                comboBoxPost.Items.Add(reader["Название"].ToString());
            }
            reader.Close();
            cmd.CommandText = "SELECT Название FROM Филиалы";
            reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                comboBoxBranch.Items.Add(reader["Название"].ToString());
            }
            reader.Close();
        }

        private void buttonAddEmployee_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPAddEmployee @Должности, @Филиалы, @ФИО,
@Адрес, @Телефон, @Логин, @Пароль";
            cmd.Parameters.Add("@Должности", SqlDbType.Char).Value =
comboBoxPost.SelectedItem.ToString().Trim();
            cmd.Parameters.Add("@Филиалы", SqlDbType.Char).Value =
comboBoxBranch.SelectedItem.ToString().Trim();
            cmd.Parameters.Add("@ФИО", SqlDbType.Char).Value = textBoxFIO.Text;
            cmd.Parameters.Add("@Адрес", SqlDbType.Char).Value = textBoxAddress.Text;
            cmd.Parameters.Add("@Телефон", SqlDbType.Int).Value =
textBoxTelephon.Text;
            if
((comboBoxPost.SelectedItem.ToString().Trim().CompareTo("Администратор") == 0) ||
(comboBoxPost.SelectedItem.ToString().Trim().CompareTo("Продавец") == 0))
            {
                cmd.Parameters.Add("@Логин", SqlDbType.Char).Value =
textBoxLogin.Text;
```

```
                cmd.Parameters.Add("@Пароль", SqlDbType.Char).Value =
textBoxPassword.Text;
            }
            else
            {
                cmd.Parameters.Add("@Логин", SqlDbType.Char).Value = "";
                cmd.Parameters.Add("@Пароль", SqlDbType.Char).Value = "";
            }
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Сотрудник успешно добавлен", "Добавление
сотрудника", MessageBoxButtons.OK);
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка добавления сотрудника", "Добавление
сотрудника", MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Добавление
сотрудника", MessageBoxButtons.OK);
            }
            catch (FormatException ex)
            {
                MessageBox.Show("Корректно заполните все необходимые поля",
"Добавление сотрудника", MessageBoxButtons.OK);
            }
        }

        private void comboBoxPost_SelectedIndexChanged(object sender, EventArgs e)
        {
            if
((comboBoxPost.SelectedItem.ToString().Trim().CompareTo("Администратор") == 0) ||
(comboBoxPost.SelectedItem.ToString().Trim().CompareTo("Продавец") == 0))
            {
                panelLoginID.Enabled = true;
            }
            else
            {
                panelLoginID.Enabled = false;
            }
        }

        private void buttonExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }

    }
}
```

FormNewPost.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
```

```
using System.Windows.Forms;
using System.Data.SqlClient;

namespace _43_Apteka
{
    public partial class FormNewPost : Form
    {
        public FormNewPost()
        {
            InitializeComponent();
        }

        private void buttonAddPost_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPAddPost @Название, @Зарплата";
            cmd.Parameters.Add("@Название", SqlDbType.Char).Value = textBoxName.Text;
            cmd.Parameters.Add("@Зарплата", SqlDbType.Int).Value =
textBoxSalary.Text;
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Должность успешно добавлена", "Добавление
должности", MessageBoxButtons.OK);
                this.Close();
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка добавления должности", "Добавление
должности", MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Добавление
должности", MessageBoxButtons.OK);
            }
            catch (FormatException ex)
            {
                MessageBox.Show("Корректно заполните все необходимые поля",
"Добавление должности", MessageBoxButtons.OK);
            }
        }
    }
}
```

FormNewPreparation.cs

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

namespace _43_Apteka
{
    public partial class FormNewPreparation : Form
    {
        public FormNewPreparation()
```

```
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT Название FROM Категории";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                comboBoxCategory.Items.Add(reader["Название"].ToString());
            }
            reader.Close();
        }

        private void buttonAddPreparation_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPAddPreparation @Категории, @Название,
@Цена";
            cmd.Parameters.Add("@Категории", SqlDbType.Char).Value =
comboBoxCategory.SelectedItem.ToString();
            cmd.Parameters.Add("@Название", SqlDbType.Char).Value = textBoxName.Text;
            cmd.Parameters.Add("@Цена", SqlDbType.Int).Value = textBoxPrice.Text;
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Препарат успешно добавлен", "Добавление препарата",
MessageBoxButtons.OK);
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка добавления препарата", "Добавление
препарата", MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Добавление
препарата", MessageBoxButtons.OK);
            }
            catch (FormatException ex)
            {
                MessageBox.Show("Корректно заполните все необходимые поля",
"Добавление препарата", MessageBoxButtons.OK);
            }
        }

        private void buttonExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

FormNewSupplier.cs

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

```
namespace _43_Apteka
{
    public partial class FormNewSupplier : Form
    {
        public FormNewSupplier()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT Название FROM Категории";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                comboBoxCategory.Items.Add(reader["Название"].ToString());
            }
            reader.Close();
        }

        private void buttonAddEmployee_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPAddSupplier @Категории, @ФИО, @Адрес,
@Телефон";
            cmd.Parameters.Add("@Категории", SqlDbType.Char).Value =
comboBoxCategory.SelectedItem.ToString();
            cmd.Parameters.Add("@ФИО", SqlDbType.Char).Value = textBoxFIO.Text;
            cmd.Parameters.Add("@Адрес", SqlDbType.Char).Value = textBoxAddress.Text;
            cmd.Parameters.Add("@Телефон", SqlDbType.Int).Value =
textBoxTelephon.Text;
            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Поставщик успешно добавлен", "Добавление
поставщика", MessageBoxButtons.OK);
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка добавления поставщика", "Добавление
поставщика", MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Добавление
поставщика", MessageBoxButtons.OK);
            }
            catch (FormatException ex)
            {
                MessageBox.Show("Корректно заполните все необходимые поля",
"Добавление поставщика", MessageBoxButtons.OK);
            }
        }

        private void buttonExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

FormNewWarehouse.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
namespace _43_Apteka
{
    public partial class FormNewWarehouse : Form
    {
        public FormNewWarehouse()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT Название FROM Филиалы";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                comboBoxBranch.Items.Add(reader["Название"].ToString());
            }
            reader.Close();
            cmd.CommandText = "SELECT Название FROM Препараты";
            reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                comboBoxPreparation.Items.Add(reader["Название"].ToString());
            }
            reader.Close();
        }



        private void buttonAddWarehouse_Click(object sender, EventArgs e)
        {SqlCommand cmd = FormMain.conn.CreateCommand();
        cmd.CommandText = "EXECUTE HPAddWarehouse @Филиалы, @Препараты, @Количество";

            cmd.Parameters.Add("@Филиалы", SqlDbType.Char).Value =
comboBoxBranch.SelectedItem.ToString().Trim();
            cmd.Parameters.Add("@Препараты", SqlDbType.Char).Value =
comboBoxPreparation.SelectedItem.ToString().Trim();
            cmd.Parameters.Add("@Количество", SqlDbType.Int).Value =
textBoxCount.Text;

            try
            {
                cmd.ExecuteNonQuery();
                MessageBox.Show("Препарат успешно добавлен на склад", "Добавление
препарата на склад", MessageBoxButtons.OK);
            }
            catch (SqlException ex)
            {
                MessageBox.Show("Ошибка добавления препарата на склад", "Добавление
препарата на склад", MessageBoxButtons.OK);
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show("Ошибка подключения к серверу", "Добавление препарата
на склад", MessageBoxButtons.OK);
            }
            catch (FormatException ex)
```

```
            {
                MessageBox.Show("Корректно заполните все необходимые поля",
"Добавление препарата на склад", MessageBoxButtons.OK);
            }
        }

        private void buttonExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

FormSellerPage.cs

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

namespace _43_Apteka
{
    public partial class FormSellerPage : Form
    {
        int price = 0;
        public FormSellerPage()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT Название FROM Скидки";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                comboBoxDiscount.Items.Add(reader["Название"].ToString().Trim());
            }
            reader.Close();

            cmd.CommandText = "SELECT Название FROM Препараты";
            reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                comboBoxPreparation.Items.Add(reader["Название"].ToString().Trim());
            }
            reader.Close();

            cmd.CommandText = "SELECT Название FROM Филиалы";
            reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                comboBoxBranch.Items.Add(reader["Название"].ToString().Trim());
            }
            reader.Close();

            cmd.CommandText = "SELECT ФИО FROM Покупатели";
            reader = cmd.ExecuteReader();
            while (reader.Read())
            {
```

```
                comboBoxBuyer.Items.Add(reader["ФИО"].ToString().Trim());
            }
            reader.Close();
        }

        private void buttonAddBuyer_Click(object sender, EventArgs e)
        {
            try
            {
                SqlCommand cmd = FormMain.conn.CreateCommand();
                cmd.CommandText = "EXECUTE HPAddBuyer @Скидки, @ФИО,
@ОбщаяСуммаПокупок";
                cmd.Parameters.Add("@Скидки", SqlDbType.Char).Value =
comboBoxBuyer.SelectedItem.ToString();
                cmd.Parameters.Add("@ФИО", SqlDbType.Char).Value = textBoxFIO.Text;
                cmd.Parameters.Add("@ОбщаяСуммаПокупок", SqlDbType.Int).Value =
textBoxSumBuy.Text;
                try
                {
                    cmd.ExecuteNonQuery();
                    MessageBox.Show("Покупатель успешно добавлен", "Добавление
покупателя", MessageBoxButtons.OK);
                }
                catch (SqlException ex)
                {
                    MessageBox.Show("Ошибка добавления покупателя", "Добавление
покупателя", MessageBoxButtons.OK);
                }
                catch (InvalidOperationException ex)
                {
                    MessageBox.Show("Ошибка подключения к серверу", "Добавление
покупателя", MessageBoxButtons.OK);
                }
            }
            catch (NullReferenceException ex)
            {
                MessageBox.Show("Заполните все необходимые поля", "Ошибка добавления
покупателя", MessageBoxButtons.OK);
            }
        }

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

        private void buttonViewTable_Click(object sender, EventArgs e)
        {
            try
            {
                string table = comboBoxViewTables.SelectedItem.ToString();
                switch (table)
                {
                    case "Покупатели":
                        FormViewBuyer formViewBuyer = new FormViewBuyer();
                        formViewBuyer.ShowDialog();
                        break;
                    case "Скидки":
                        FormViewDiscount formViewDiscount = new FormViewDiscount();
                        formViewDiscount.ShowDialog();
                        break;
                    case "Склад":
```

```
                        FormViewWarehouse formViewWarehouse = new
FormViewWarehouse();
                        formViewWarehouse.ShowDialog();
                        break;
                }
            }
            catch (NullReferenceException ex)
            {
                MessageBox.Show("Выберите таблицу для просмотра", "Ошибка просмотра",
MessageBoxButtons.OK);
            }
        }

        private void buttonSale_Click(object sender, EventArgs e)
        {
            try
            {
                SqlCommand cmd = FormMain.conn.CreateCommand();
                cmd.Parameters.Clear();
                cmd.CommandText = "EXECUTE HPGiveCountPreparation @Препарат1,
@Филиал1, @Количество1 OUTPUT";
                cmd.Parameters.Add("@Препарат1", SqlDbType.Char).Value =
comboBoxPreparation.SelectedItem.ToString().Trim();
                cmd.Parameters.Add("@Филиал1", SqlDbType.Char).Value =
comboBoxBranch.SelectedItem.ToString().Trim();
                cmd.Parameters.Add(new SqlParameter("@Количество1", SqlDbType.Int));
                cmd.Parameters["@Количество1"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                int count =
Convert.ToInt32(cmd.Parameters["@Количество1"].Value.ToString());
                if (count > Convert.ToInt32(textBoxCount.Text.ToString()))
                {
                    cmd = FormMain.conn.CreateCommand();
                    cmd.CommandText = "EXECUTE HPMakeSale @Препарат, @Филиал,
@Покупатель, @Цена, @Количество, @Итого";
                    cmd.Parameters.Add("@Препарат", SqlDbType.Char).Value =
comboBoxPreparation.SelectedItem.ToString().Trim();
                    cmd.Parameters.Add("@Филиал", SqlDbType.Char).Value =
comboBoxBranch.SelectedItem.ToString().Trim();
                    cmd.Parameters.Add("@Покупатель", SqlDbType.VarChar, 255).Value =
comboBoxBuyer.SelectedItem.ToString().Trim();
                    cmd.Parameters.Add("@Цена", SqlDbType.Int).Value =
textBoxPrice.Text.ToString();
                    cmd.Parameters.Add("@Количество", SqlDbType.Int).Value =
textBoxCount.Text.ToString();
                    cmd.Parameters.Add("@Итого", SqlDbType.Int).Value =
textBoxTotal.Text.ToString();
                    cmd.ExecuteNonQuery();
                    MessageBox.Show("Товар успешно куплен!", "Покупка препаратов",
MessageBoxButtons.OK);
                }
                else
                {
                    MessageBox.Show("На складе недостаточно препаратов!", "Покупка
препаратов", MessageBoxButtons.OK);
                }
            }
            catch (FormatException ex)
            {
                MessageBox.Show("Введите корректные значения", "Ошибка покупки",
MessageBoxButtons.OK);
            }
```

```
            catch (NullReferenceException ex)
            {
                MessageBox.Show("Заполните все требуемые поля", "Ошибка покупки",
MessageBoxButtons.OK);
            }
        }

        private void textBoxCount_TextChanged(object sender, EventArgs e)
        {
            try
            {
                int total = Convert.ToInt32(textBoxPrice.Text) *
Convert.ToInt32(textBoxCount.Text);
                textBoxTotal.Text = total.ToString();
            }
            catch (FormatException ex)
            {
                textBoxTotal.Text = "";
            }
        }

        private void comboBoxPreparation_SelectedIndexChanged(object sender,
EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPGivePricePreparation @Препарат, @Цена
OUTPUT";
            cmd.Parameters.Add("@Препарат", SqlDbType.Char).Value =
comboBoxPreparation.SelectedItem.ToString().Trim();
            cmd.Parameters.Add(new SqlParameter("@Цена", SqlDbType.Int));
            cmd.Parameters["@Цена"].Direction = ParameterDirection.Output;
            cmd.ExecuteNonQuery();
            textBoxPrice.Text = cmd.Parameters["@Цена"].Value.ToString();
            price = Convert.ToInt32(cmd.Parameters["@Цена"].Value.ToString());
        }

        private void comboBoxBuyer_SelectedIndexChanged(object sender, EventArgs e)
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "EXECUTE HPGiveDiscount @Покупатель, @Скидка OUTPUT";
            cmd.Parameters.Add("@Покупатель", SqlDbType.Char).Value =
comboBoxBuyer.SelectedItem.ToString().Trim();
            cmd.Parameters.Add(new SqlParameter("@Скидка", SqlDbType.Int));
            cmd.Parameters["@Скидка"].Direction = ParameterDirection.Output;
            cmd.ExecuteNonQuery();
            int discount =
Convert.ToInt32(cmd.Parameters["@Скидка"].Value.ToString());
            int newPrice = price * (100 - discount) / 100;
            textBoxPrice.Text = newPrice.ToString();
        }
    }
}
```

FormViewBranch.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
```

```
using System.Windows.Forms;
using System.Data.SqlClient;

namespace _43_Apteka
{
    public partial class FormViewBranch : Form
    {
        public FormViewBranch()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Филиалы";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell name = new DataGridViewTextBoxCell();
                DataGridViewCell address = new DataGridViewTextBoxCell();
                DataGridViewCell telephone = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Филиалы"].ToString();
                name.Value = reader["Название"].ToString().Trim();
                address.Value = reader["Адрес"].ToString().Trim();
                telephone.Value = reader["Телефон"].ToString();
                row.Cells.AddRange(id, name, address, telephone);
                dataGridViewBranch.Rows.Add(row);
            }
            reader.Close();
            textBoxID.Text = dataGridViewBranch[0, 0].Value.ToString();
            textBoxName.Text = dataGridViewBranch[1, 0].Value.ToString();
            textBoxAddress.Text = dataGridViewBranch[2, 0].Value.ToString();
            textBoxTelephone.Text = dataGridViewBranch[3, 0].Value.ToString();
        }

        private void dataGridViewBranch_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBoxID.Text = dataGridViewBranch[0,
dataGridViewBranch.CurrentRow.Index].Value.ToString();
            textBoxName.Text = dataGridViewBranch[1,
dataGridViewBranch.CurrentRow.Index].Value.ToString();
            textBoxAddress.Text = dataGridViewBranch[2,
dataGridViewBranch.CurrentRow.Index].Value.ToString();
            textBoxTelephone.Text = dataGridViewBranch[3,
dataGridViewBranch.CurrentRow.Index].Value.ToString();
        }
    }
}
```

FormViewBuyer.cs

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

```
namespace _43_Apteka
{
    public partial class FormViewBuyer : Form
    {
        public FormViewBuyer()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Покупатели";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell discount = new DataGridViewTextBoxCell();
                DataGridViewCell fio = new DataGridViewTextBoxCell();
                DataGridViewCell sumBuy = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Покупатели"].ToString();
                discount.Value = reader["Скидки"].ToString();
                fio.Value = reader["ФИО"].ToString().Trim();
                sumBuy.Value = reader["ОбщаяСуммаПокупок"].ToString();
                row.Cells.AddRange(id, discount, fio, sumBuy);
                dataGridViewBuyer.Rows.Add(row);
            }
            reader.Close();

            for (int i = 0; i < dataGridViewBuyer.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveSizeDiscount @Скидки, @Размер
OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Скидки", SqlDbType.Int).Value =
dataGridViewBuyer[1, i].Value.ToString();
                cmd.Parameters.Add("@Размер", SqlDbType.Int);
                cmd.Parameters["@Размер"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewBuyer[1, i].Value =
cmd.Parameters["@Размер"].Value.ToString();
            }

            textBoxID.Text = dataGridViewBuyer[0, 0].Value.ToString();
            textBoxDiscount.Text = dataGridViewBuyer[1, 0].Value.ToString();
            textBoxFIO.Text = dataGridViewBuyer[2, 0].Value.ToString();
            textBoxSumBuy.Text = dataGridViewBuyer[3, 0].Value.ToString();
        }

        private void dataGridViewBuyer_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBoxID.Text = dataGridViewBuyer[0,
dataGridViewBuyer.CurrentRow.Index].Value.ToString();
            textBoxDiscount.Text = dataGridViewBuyer[1,
dataGridViewBuyer.CurrentRow.Index].Value.ToString();
            textBoxFIO.Text = dataGridViewBuyer[2,
dataGridViewBuyer.CurrentRow.Index].Value.ToString();
            textBoxSumBuy.Text = dataGridViewBuyer[3,
dataGridViewBuyer.CurrentRow.Index].Value.ToString();
        }
    }
}
```

FormViewCategory.cs

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

namespace _43_Apteka
{
    public partial class FormViewCategory : Form
    {
        public FormViewCategory()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Категории";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell name = new DataGridViewTextBoxCell();
                DataGridViewRow row = new DataGridViewRow();

                id.Value = reader["ID_Категории"].ToString();
                name.Value = reader["Название"].ToString().Trim();
                row.Cells.AddRange(id, name);
                dataGridViewCategory.Rows.Add(row);
            }
            reader.Close();
            textBoxID.Text = dataGridViewCategory[0, 0].Value.ToString();
            textBoxName.Text = dataGridViewCategory[1, 0].Value.ToString();
        }

        private void dataGridViewCategory_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBoxID.Text = dataGridViewCategory[0,
dataGridViewCategory.CurrentRow.Index].Value.ToString();
            textBoxName.Text = dataGridViewCategory[1,
dataGridViewCategory.CurrentRow.Index].Value.ToString();
        }
    }
}
```

FormViewDiscount.cs

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

namespace _43_Apteka
{
```

```
    public partial class FormViewDiscount : Form
    {
        public FormViewDiscount()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Скидки";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell name = new DataGridViewTextBoxCell();
                DataGridViewCell size = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Скидки"].ToString();
                name.Value = reader["Название"].ToString().Trim();
                size.Value = reader["Размер"].ToString();
                row.Cells.AddRange(id, name, size);
                dataGridViewDiscount.Rows.Add(row);
            }
            reader.Close();
            textBoxID.Text = dataGridViewDiscount[0, 0].Value.ToString();
            textBoxName.Text = dataGridViewDiscount[1, 0].Value.ToString();
            textBoxSize.Text = dataGridViewDiscount[2, 0].Value.ToString();
        }

        private void dataGridViewDiscount_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBoxID.Text = dataGridViewDiscount[0,
dataGridViewDiscount.CurrentRow.Index].Value.ToString();
            textBoxName.Text = dataGridViewDiscount[1,
dataGridViewDiscount.CurrentRow.Index].Value.ToString();
            textBoxSize.Text = dataGridViewDiscount[2,
dataGridViewDiscount.CurrentRow.Index].Value.ToString();
        }
    }
}
```

FormViewEmployee.cs

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

namespace _43_Apteka
{
    public partial class FormViewEmployee : Form
    {
        public FormViewEmployee()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Сотрудники";
            SqlDataReader reader = cmd.ExecuteReader();
```

```
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell post = new DataGridViewTextBoxCell();
                DataGridViewCell branch = new DataGridViewTextBoxCell();
                DataGridViewCell fio = new DataGridViewTextBoxCell();
                DataGridViewCell address = new DataGridViewTextBoxCell();
                DataGridViewCell telephone = new DataGridViewTextBoxCell();
                DataGridViewCell login = new DataGridViewTextBoxCell();
                DataGridViewCell password = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Сотрудники"].ToString();
                post.Value = reader["Должности"].ToString();
                branch.Value = reader["Филиалы"].ToString();
                fio.Value = reader["ФИО"].ToString().Trim();
                address.Value = reader["Адрес"].ToString().Trim();
                telephone.Value = reader["Телефон"].ToString();
                login.Value = "";
                password.Value = "";
                row.Cells.AddRange(id, post, branch, fio, address, telephone, login,
password);
                dataGridViewEmployee.Rows.Add(row);
            }
            reader.Close();

            for (int i = 0; i < dataGridViewEmployee.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNamePost @Должности, @Название1
OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Должности", SqlDbType.Int).Value =
dataGridViewEmployee[1, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название1", SqlDbType.VarChar,
30));
                cmd.Parameters["@Название1"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewEmployee[1, i].Value =
cmd.Parameters["@Название1"].Value.ToString().Trim();
            }

            for (int i = 0; i < dataGridViewEmployee.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNameBranch @Филиалы, @Название
OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Филиалы", SqlDbType.Int).Value =
dataGridViewEmployee[2, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название", SqlDbType.VarChar,
30));
                cmd.Parameters["@Название"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewEmployee[2, i].Value =
cmd.Parameters["@Название"].Value.ToString().Trim();
            }

            cmd.CommandText = "SELECT * FROM ПривилегированныеСотрудники";
            reader = cmd.ExecuteReader();
            int j = 0;
            while (reader.Read())
            {
                string id = reader["ID_Сотрудники"].ToString();
```

```
                string login = reader["Логин"].ToString().Trim();
                string password = reader["Пароль"].ToString().Trim();
                for (int i = 0; i < dataGridViewEmployee.RowCount; i++)
                {
                    if (dataGridViewEmployee[0,
i].Value.ToString().Trim().CompareTo(id.Trim()) == 0)
                    {
                        dataGridViewEmployee[6, i].Value = login;
                        dataGridViewEmployee[7, i].Value = password;
                        j = i + 1;
                        break;
                    }
                }
            }
            reader.Close();

            textBoxID.Text = dataGridViewEmployee[0, 0].Value.ToString();
            textBoxBranch.Text = dataGridViewEmployee[1, 0].Value.ToString();
            textBoxPost.Text = dataGridViewEmployee[2, 0].Value.ToString();
            textBoxFIO.Text = dataGridViewEmployee[3, 0].Value.ToString();
            textBoxAddress.Text = dataGridViewEmployee[4, 0].Value.ToString();
            textBoxTelephone.Text = dataGridViewEmployee[5, 0].Value.ToString();
            textBoxLogin.Text = dataGridViewEmployee[6, 0].Value.ToString();
            textBoxPassword.Text = dataGridViewEmployee[7, 0].Value.ToString();
        }

        private void dataGridViewEmployee_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBoxID.Text = dataGridViewEmployee[0,
dataGridViewEmployee.CurrentRow.Index].Value.ToString();
            textBoxPost.Text = dataGridViewEmployee[1,
dataGridViewEmployee.CurrentRow.Index].Value.ToString();
            textBoxBranch.Text = dataGridViewEmployee[2,
dataGridViewEmployee.CurrentRow.Index].Value.ToString();
            textBoxFIO.Text = dataGridViewEmployee[3,
dataGridViewEmployee.CurrentRow.Index].Value.ToString();
            textBoxAddress.Text = dataGridViewEmployee[4,
dataGridViewEmployee.CurrentRow.Index].Value.ToString();
            textBoxTelephone.Text = dataGridViewEmployee[5,
dataGridViewEmployee.CurrentRow.Index].Value.ToString();
            textBoxLogin.Text = dataGridViewEmployee[6,
dataGridViewEmployee.CurrentRow.Index].Value.ToString();
            textBoxPassword.Text = dataGridViewEmployee[7,
dataGridViewEmployee.CurrentRow.Index].Value.ToString();

            for (int i = 0; i < dataGridViewEmployee.RowCount; i++)
            {
                if ((textBoxPost.Text.ToString().Trim().CompareTo("Администратор") ==
0) || (textBoxPost.Text.ToString().Trim().CompareTo("Кассир-консультант") == 0))
                {
                    panelIDPassword.Enabled = true;
                }
                else
                {
                    panelIDPassword.Enabled = false;
                }
            }
        }
    }
}
```

FormViewPost.cs

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

namespace _43_Apteka
{
    public partial class FormViewPost : Form
    {
        public FormViewPost()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Должности";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell name = new DataGridViewTextBoxCell();
                DataGridViewCell salary = new DataGridViewTextBoxCell();
                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Должности"].ToString();
                name.Value = reader["Название"].ToString().Trim();
                salary.Value = reader["Зарплата"].ToString();
                row.Cells.AddRange(id, name, salary);
                dataGridViewPost.Rows.Add(row);
            }
            reader.Close();
            textBoxID.Text = dataGridViewPost[0, 0].Value.ToString();
            textBoxName.Text = dataGridViewPost[1, 0].Value.ToString();
            textBoxSalary.Text = dataGridViewPost[2, 0].Value.ToString();
        }

        private void dataGridViewPost_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBoxID.Text = dataGridViewPost[0,
dataGridViewPost.CurrentRow.Index].Value.ToString();
            textBoxName.Text = dataGridViewPost[1,
dataGridViewPost.CurrentRow.Index].Value.ToString();
            textBoxSalary.Text = dataGridViewPost[2,
dataGridViewPost.CurrentRow.Index].Value.ToString();
        }
    }
}
```

FormViewPreparation.cs

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

```
using System.Data.SqlClient;

namespace _43_Apteka
{
    public partial class FormViewPreparation : Form
    {
        public FormViewPreparation()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Препараты";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell category = new DataGridViewTextBoxCell();
                DataGridViewCell name = new DataGridViewTextBoxCell();
                DataGridViewCell price = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Препараты"].ToString();
                category.Value = reader["Категории"].ToString();
                name.Value = reader["Название"].ToString().Trim();
                price.Value = reader["Цена"].ToString();
                row.Cells.AddRange(id, category, name, price);
                dataGridViewPreparation.Rows.Add(row);
            }
            reader.Close();

            for (int i = 0; i < dataGridViewPreparation.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNameCategory @Категории, @Название
OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Категории", SqlDbType.Int).Value =
dataGridViewPreparation[1, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название", SqlDbType.VarChar,
50));
                cmd.Parameters["@Название"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewPreparation[1, i].Value =
cmd.Parameters["@Название"].Value.ToString().Trim();
            }

            textBoxID.Text = dataGridViewPreparation[0, 0].Value.ToString();
            textBoxCategory.Text = dataGridViewPreparation[1, 0].Value.ToString();
            textBoxName.Text = dataGridViewPreparation[2, 0].Value.ToString();
            textBoxPrice.Text = dataGridViewPreparation[3, 0].Value.ToString();
        }

        private void dataGridViewPreparation_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBoxID.Text = dataGridViewPreparation[0,
dataGridViewPreparation.CurrentRow.Index].Value.ToString();
            textBoxCategory.Text = dataGridViewPreparation[1,
dataGridViewPreparation.CurrentRow.Index].Value.ToString();
            textBoxName.Text = dataGridViewPreparation[2,
dataGridViewPreparation.CurrentRow.Index].Value.ToString();
            textBoxPrice.Text = dataGridViewPreparation[3,
dataGridViewPreparation.CurrentRow.Index].Value.ToString();
        }
```

```
    }
}
```

FormViewPurchase.cs

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

namespace _43_Apteka
{
    public partial class FormViewPurchase : Form
    {
        public FormViewPurchase()
        {
            InitializeComponent();
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "SELECT * FROM Покупки";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell preparation = new DataGridViewTextBoxCell();
                DataGridViewCell branch = new DataGridViewTextBoxCell();
                DataGridViewCell buyer = new DataGridViewTextBoxCell();
                DataGridViewCell price = new DataGridViewTextBoxCell();
                DataGridViewCell count = new DataGridViewTextBoxCell();
                DataGridViewCell total = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Покупки"].ToString();
                preparation.Value = reader["Препараты"].ToString();
                branch.Value = reader["Филиалы"].ToString();
                buyer.Value = reader["Покупатели"].ToString();
                price.Value = reader["ЦенаСоСкидкой"].ToString();
                count.Value = reader["Количество"].ToString();
                total.Value = reader["Итого"].ToString();
                row.Cells.AddRange(id, preparation, branch, buyer, price, count,
total);
                dataGridViewPurchase.Rows.Add(row);
            }
            reader.Close();

            for (int i = 0; i < dataGridViewPurchase.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNamePreparation @Препараты,
@Название OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Препараты", SqlDbType.Int).Value =
dataGridViewPurchase[1, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название", SqlDbType.VarChar,
30));
                cmd.Parameters["@Название"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewPurchase[1, i].Value =
cmd.Parameters["@Название"].Value.ToString().Trim();
```

```
            }

            for (int i = 0; i < dataGridViewPurchase.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNameBranch @Филиалы, @Название1
OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Филиалы", SqlDbType.Int).Value =
dataGridViewPurchase[2, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название1", SqlDbType.VarChar,
30));
                cmd.Parameters["@Название1"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewPurchase[2, i].Value =
cmd.Parameters["@Название1"].Value.ToString().Trim();
            }

            for (int i = 0; i < dataGridViewPurchase.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNameBuyer @Покупатели, @ФИО OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Покупатели", SqlDbType.Int).Value =
dataGridViewPurchase[3, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@ФИО", SqlDbType.VarChar, 255));
                cmd.Parameters["@ФИО"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewPurchase[3, i].Value =
cmd.Parameters["@ФИО"].Value.ToString().Trim();
            }
            if (dataGridViewPurchase.RowCount > 0)
            {
                textBoxID.Text = dataGridViewPurchase[0, 0].Value.ToString();
                textBoxPreparation.Text = dataGridViewPurchase[1,
0].Value.ToString();
                textBoxBranch.Text = dataGridViewPurchase[2, 0].Value.ToString();
                textBoxBuyer.Text = dataGridViewPurchase[3, 0].Value.ToString();
                textBoxPrice.Text = dataGridViewPurchase[4, 0].Value.ToString();
                textBoxCount.Text = dataGridViewPurchase[5, 0].Value.ToString();
                textBoxTotal.Text = dataGridViewPurchase[6, 0].Value.ToString();
            }
        }

        public FormViewPurchase(int code)
        {
            InitializeComponent();
            SqlCommand cmd = FormMain.conn.CreateCommand();
            cmd.CommandText = "SELECT * FROM Покупки";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                if (Convert.ToInt32(reader["Покупатели"].ToString().Trim()) == code)
                {
                    DataGridViewCell id = new DataGridViewTextBoxCell();
                    DataGridViewCell preparation = new DataGridViewTextBoxCell();
                    DataGridViewCell branch = new DataGridViewTextBoxCell();
                    DataGridViewCell buyer = new DataGridViewTextBoxCell();
                    DataGridViewCell price = new DataGridViewTextBoxCell();
                    DataGridViewCell count = new DataGridViewTextBoxCell();
                    DataGridViewCell total = new DataGridViewTextBoxCell();

                    DataGridViewRow row = new DataGridViewRow();
                    id.Value = reader["ID_Покупки"].ToString();
```

```
                    preparation.Value = reader["Препараты"].ToString();
                    branch.Value = reader["Филиалы"].ToString();
                    buyer.Value = reader["Покупатели"].ToString();
                    price.Value = reader["ЦенаСоСкидкой"].ToString();
                    count.Value = reader["Количество"].ToString();
                    total.Value = reader["Итого"].ToString();
                    row.Cells.AddRange(id, preparation, branch, buyer, price, count,
total);
                    dataGridViewPurchase.Rows.Add(row);
                }
            }
            reader.Close();

            for (int i = 0; i < dataGridViewPurchase.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNamePreparation @Препараты,
@Название OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Препараты", SqlDbType.Int).Value =
dataGridViewPurchase[1, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название", SqlDbType.VarChar,
30));
                cmd.Parameters["@Название"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewPurchase[1, i].Value =
cmd.Parameters["@Название"].Value.ToString().Trim();
            }

            for (int i = 0; i < dataGridViewPurchase.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNameBranch @Филиалы, @Название1
OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Филиалы", SqlDbType.Int).Value =
dataGridViewPurchase[2, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название1", SqlDbType.VarChar,
30));
                cmd.Parameters["@Название1"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewPurchase[2, i].Value =
cmd.Parameters["@Название1"].Value.ToString().Trim();
            }

            for (int i = 0; i < dataGridViewPurchase.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNameBuyer @Покупатели, @ФИО OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Покупатели", SqlDbType.Int).Value =
dataGridViewPurchase[3, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@ФИО", SqlDbType.VarChar, 255));
                cmd.Parameters["@ФИО"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewPurchase[3, i].Value =
cmd.Parameters["@ФИО"].Value.ToString().Trim();
            }
            if (dataGridViewPurchase.RowCount > 0)
            {
                textBoxID.Text = dataGridViewPurchase[0, 0].Value.ToString();
                textBoxPreparation.Text = dataGridViewPurchase[1,
0].Value.ToString();
                textBoxBranch.Text = dataGridViewPurchase[2, 0].Value.ToString();
                textBoxBuyer.Text = dataGridViewPurchase[3, 0].Value.ToString();
```

```
                textBoxPrice.Text = dataGridViewPurchase[4, 0].Value.ToString();
                textBoxCount.Text = dataGridViewPurchase[5, 0].Value.ToString();
                textBoxTotal.Text = dataGridViewPurchase[6, 0].Value.ToString();
            }
        }

        private void dataGridViewPurchase_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBoxID.Text = dataGridViewPurchase[0,
dataGridViewPurchase.CurrentRow.Index].Value.ToString();
            textBoxPreparation.Text = dataGridViewPurchase[1,
dataGridViewPurchase.CurrentRow.Index].Value.ToString();
            textBoxBranch.Text = dataGridViewPurchase[2,
dataGridViewPurchase.CurrentRow.Index].Value.ToString();
            textBoxBuyer.Text = dataGridViewPurchase[3,
dataGridViewPurchase.CurrentRow.Index].Value.ToString();
            textBoxPrice.Text = dataGridViewPurchase[4,
dataGridViewPurchase.CurrentRow.Index].Value.ToString();
            textBoxCount.Text = dataGridViewPurchase[5,
dataGridViewPurchase.CurrentRow.Index].Value.ToString();
            textBoxTotal.Text = dataGridViewPurchase[6,
dataGridViewPurchase.CurrentRow.Index].Value.ToString();
        }
    }
}
```

FormViewSupplier.cs

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

namespace _43_Apteka
{
    public partial class FormViewSupplier : Form
    {
        public FormViewSupplier()
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Поставщики";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell category = new DataGridViewTextBoxCell();
                DataGridViewCell fio = new DataGridViewTextBoxCell();
                DataGridViewCell address = new DataGridViewTextBoxCell();
                DataGridViewCell telephone = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Поставщики"].ToString();
                category.Value = reader["Категории"].ToString();
                fio.Value = reader["ФИО"].ToString().Trim();
                address.Value = reader["Адрес"].ToString().Trim();
```

```
                telephone.Value = reader["Телефон"].ToString();
                row.Cells.AddRange(id, category, fio, address, telephone);
                dataGridViewSupplier.Rows.Add(row);
            }
            reader.Close();

            for (int i = 0; i < dataGridViewSupplier.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNameCategory @Категории, @Название
OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Категории", SqlDbType.Int).Value =
dataGridViewSupplier[1, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название", SqlDbType.VarChar,
50));
                cmd.Parameters["@Название"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewSupplier[1, i].Value =
cmd.Parameters["@Название"].Value.ToString().Trim();
            }

            textBoxID.Text = dataGridViewSupplier[0, 0].Value.ToString();
            textBoxCategory.Text = dataGridViewSupplier[1, 0].Value.ToString();
            textBoxFIO.Text = dataGridViewSupplier[2, 0].Value.ToString();
            textBoxAddress.Text = dataGridViewSupplier[3, 0].Value.ToString();
            textBoxTelephone.Text = dataGridViewSupplier[4, 0].Value.ToString();
        }

        private void dataGridViewSupplier_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBoxID.Text = dataGridViewSupplier[0,
dataGridViewSupplier.CurrentRow.Index].Value.ToString();
            textBoxCategory.Text = dataGridViewSupplier[1,
dataGridViewSupplier.CurrentRow.Index].Value.ToString();
            textBoxFIO.Text = dataGridViewSupplier[2,
dataGridViewSupplier.CurrentRow.Index].Value.ToString();
            textBoxAddress.Text = dataGridViewSupplier[3,
dataGridViewSupplier.CurrentRow.Index].Value.ToString();
            textBoxTelephone.Text = dataGridViewSupplier[4,
dataGridViewSupplier.CurrentRow.Index].Value.ToString();
        }
    }
}
```

FormViewSupply.cs

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

namespace _43_Apteka
{
    public partial class FormViewSupply : Form
    {
        public FormViewSupply()
```

```
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Поставки";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell preparation = new DataGridViewTextBoxCell();
                DataGridViewCell branch = new DataGridViewTextBoxCell();
                DataGridViewCell supplier = new DataGridViewTextBoxCell();
                DataGridViewCell price = new DataGridViewTextBoxCell();
                DataGridViewCell count = new DataGridViewTextBoxCell();
                DataGridViewCell total = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Поставки"].ToString();
                preparation.Value = reader["Препараты"].ToString();
                branch.Value = reader["Филиалы"].ToString();
                supplier.Value = reader["Поставщики"].ToString();
                price.Value = reader["ЦенаПоставки"].ToString();
                count.Value = reader["Количество"].ToString();
                total.Value = reader["Итого"].ToString();
                row.Cells.AddRange(id, preparation, branch, supplier, price, count,
total);
                dataGridViewSupply.Rows.Add(row);
            }
            reader.Close();

            for (int i = 0; i < dataGridViewSupply.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNamePreparation @Препараты,
@Название OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Препараты", SqlDbType.Int).Value =
dataGridViewSupply[1, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название", SqlDbType.VarChar,
30));
                cmd.Parameters["@Название"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewSupply[1, i].Value =
cmd.Parameters["@Название"].Value.ToString().Trim();
            }

            for (int i = 0; i < dataGridViewSupply.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNameBranch @Филиалы, @Название1
OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Филиалы", SqlDbType.Int).Value =
dataGridViewSupply[2, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название1", SqlDbType.VarChar,
30));
                cmd.Parameters["@Название1"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewSupply[2, i].Value =
cmd.Parameters["@Название1"].Value.ToString().Trim();
            }

            for (int i = 0; i < dataGridViewSupply.RowCount; i++)
            {
```

```
                cmd.CommandText = "EXECUTE HPGiveNameSupplier @Поставщики, @ФИО
OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Поставщики", SqlDbType.Int).Value =
dataGridViewSupply[3, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@ФИО", SqlDbType.VarChar, 255));
                cmd.Parameters["@ФИО"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewSupply[3, i].Value =
cmd.Parameters["@ФИО"].Value.ToString().Trim();
            }

            if (dataGridViewSupply.RowCount > 0)
            {
                textBoxID.Text = dataGridViewSupply[0, 0].Value.ToString();
                textBoxPreparation.Text = dataGridViewSupply[1, 0].Value.ToString();
                textBoxBranch.Text = dataGridViewSupply[2, 0].Value.ToString();
                textBoxSupplier.Text = dataGridViewSupply[3, 0].Value.ToString();
                textBoxPrice.Text = dataGridViewSupply[4, 0].Value.ToString();
                textBoxCount.Text = dataGridViewSupply[5, 0].Value.ToString();
                textBoxTotal.Text = dataGridViewSupply[6, 0].Value.ToString();
            }
        }

        private void dataGridViewSupply_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBoxID.Text = dataGridViewSupply[0,
dataGridViewSupply.CurrentRow.Index].Value.ToString();
            textBoxPreparation.Text = dataGridViewSupply[1,
dataGridViewSupply.CurrentRow.Index].Value.ToString();
            textBoxBranch.Text = dataGridViewSupply[2,
dataGridViewSupply.CurrentRow.Index].Value.ToString();
            textBoxSupplier.Text = dataGridViewSupply[3,
dataGridViewSupply.CurrentRow.Index].Value.ToString();
            textBoxPrice.Text = dataGridViewSupply[4,
dataGridViewSupply.CurrentRow.Index].Value.ToString();
            textBoxCount.Text = dataGridViewSupply[5,
dataGridViewSupply.CurrentRow.Index].Value.ToString();
            textBoxTotal.Text = dataGridViewSupply[6,
dataGridViewSupply.CurrentRow.Index].Value.ToString();
        }
    }
}
```

FormViewWarehouse.cs

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

namespace _43_Apteka
{
    public partial class FormViewWarehouse : Form
    {
        public FormViewWarehouse()
```

```
        {
            SqlCommand cmd = FormMain.conn.CreateCommand();
            InitializeComponent();
            cmd.CommandText = "SELECT * FROM Склад";
            SqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                DataGridViewCell id = new DataGridViewTextBoxCell();
                DataGridViewCell branch = new DataGridViewTextBoxCell();
                DataGridViewCell preparation = new DataGridViewTextBoxCell();
                DataGridViewCell count = new DataGridViewTextBoxCell();

                DataGridViewRow row = new DataGridViewRow();
                id.Value = reader["ID_Склад"].ToString();
                branch.Value = reader["Филиалы"].ToString();
                preparation.Value = reader["Препараты"].ToString();
                count.Value = reader["Количество"].ToString();
                row.Cells.AddRange(id, branch, preparation, count);
                dataGridViewWarehouse.Rows.Add(row);
            }
            reader.Close();

            for (int i = 0; i < dataGridViewWarehouse.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNameBranch @Филиалы, @Название
OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Филиалы", SqlDbType.Int).Value =
dataGridViewWarehouse[1, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название", SqlDbType.VarChar,
30));
                cmd.Parameters["@Название"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewWarehouse[1, i].Value =
cmd.Parameters["@Название"].Value.ToString().Trim();
            }

            for (int i = 0; i < dataGridViewWarehouse.RowCount; i++)
            {
                cmd.CommandText = "EXECUTE HPGiveNamePreparation @Препараты,
@Название1 OUTPUT";
                cmd.Parameters.Clear();
                cmd.Parameters.Add("@Препараты", SqlDbType.Int).Value =
dataGridViewWarehouse[2, i].Value.ToString();
                cmd.Parameters.Add(new SqlParameter("@Название1", SqlDbType.VarChar,
30));
                cmd.Parameters["@Название1"].Direction = ParameterDirection.Output;
                cmd.ExecuteNonQuery();
                dataGridViewWarehouse[2, i].Value =
cmd.Parameters["@Название1"].Value.ToString().Trim();
            }

            textBoxID.Text = dataGridViewWarehouse[0, 0].Value.ToString();
            textBoxBranch.Text = dataGridViewWarehouse[1, 0].Value.ToString();
            textBoxPreparation.Text = dataGridViewWarehouse[2, 0].Value.ToString();
            textBoxCount.Text = dataGridViewWarehouse[3, 0].Value.ToString();
        }

        private void dataGridViewWarehouse_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
```

```
            textBoxID.Text = dataGridViewWarehouse[0,
dataGridViewWarehouse.CurrentRow.Index].Value.ToString();
            textBoxBranch.Text = dataGridViewWarehouse[1,
dataGridViewWarehouse.CurrentRow.Index].Value.ToString();
            textBoxPreparation.Text = dataGridViewWarehouse[2,
dataGridViewWarehouse.CurrentRow.Index].Value.ToString();
            textBoxCount.Text = dataGridViewWarehouse[3,
dataGridViewWarehouse.CurrentRow.Index].Value.ToString();
        }

        private void dataGridViewWarehouse_CellDoubleClick(object sender,
DataGridViewCellEventArgs e)
        {
            switch (dataGridViewWarehouse.CurrentCell.ColumnIndex)
            {
                case 1:
                    FormViewBranch formViewBranch = new FormViewBranch();
                    formViewBranch.ShowDialog();
                    break;
                case 2:
                    FormViewPreparation formViewPreparation = new
FormViewPreparation();
                    formViewPreparation.ShowDialog();
                    break;
            }
        }
    }
}
```

FormReportEmployee.cs

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

namespace _43_Apteka
{
  public partial class FormReportEmployee : Form
  {
    public FormReportEmployee()
    {
      InitializeComponent();

      SqlConnection _Connection = FormMain.conn;
      DataSetReportEmployees ds = new DataSetReportEmployees();
      SqlCommand com = _Connection.CreateCommand();
      com.CommandText = "SELECT ID_Сотрудники AS ID, Должности.Название AS
Должность,"
        + " Филиалы.Название AS Филиал, ФИО, Сотрудники.Адрес, Сотрудники.Телефон"
        + " FROM (Сотрудники INNER JOIN Должности ON Сотрудники.Должности =
Должности.ID_Должности)"
        + " INNER JOIN Филиалы ON Сотрудники.Филиалы = Филиалы.ID_Филиалы";
      SqlDataAdapter dataAdapter = new SqlDataAdapter(com);
      DataTable dt = new DataTable("DataTableEmployees");
      dataAdapter.Fill(dt);
```

```
        ds.Tables.Add(dt);

        Employees report = new Employees();
        report.SetDataSource(ds.Tables[1]);
        crystalReportViewer.ReportSource = report;
        crystalReportViewer.Refresh();
      }
    }
}
```

ПРИЛОЖЕНИЕ 3: СЦЕНАРИЙ ИНСТАЛЛЯЦИИ ПРОГРАММЫ

```
"DeployProject"
{
"VSVersion" = "3:800"
"ProjectType" = "8:{978C614F-708E-4E1A-B201-565925725DBA}"
"IsWebType" = "8:FALSE"
"ProjectName" = "8:943_AptekaSetup"
"LanguageId" = "3:1033"
"CodePage" = "3:1252"
"UILanguageId" = "3:1033"
"SccProjectName" = "8:"
"SccLocalPath" = "8:"
"SccAuxPath" = "8:"
"SccProvider" = "8:"
    "Hierarchy"
    {
        "Entry"
        {
        "MsmKey" = "8:_008A162E36EB45419D3B99FA33EA5CEB"
        "OwnerKey" = "8:_UNDEFINED"
        "MsmSig" = "8:_UNDEFINED"
        }
        "Entry"
        {
        "MsmKey" = "8:_37EC5B91C09E4521A61EC12C0D60E4C4"
        "OwnerKey" = "8:_UNDEFINED"
        "MsmSig" = "8:_UNDEFINED"
        }
        "Entry"
        {
        "MsmKey" = "8:_UNDEFINED"
        "OwnerKey" = "8:_37EC5B91C09E4521A61EC12C0D60E4C4"
        "MsmSig" = "8:_UNDEFINED"
        }
    }
    "Configurations"
    {
        "Debug"
        {
        "DisplayName" = "8:Debug"
        "IsDebugOnly" = "11:TRUE"
        "IsReleaseOnly" = "11:FALSE"
        "OutputFilename" = "8:Debug\\943_AptekaSetup.msi"
        "PackageFilesAs" = "3:2"
        "PackageFileSize" = "3:-2147483648"
        "CabType" = "3:1"
        "Compression" = "3:2"
        "SignOutput" = "11:FALSE"
        "CertificateFile" = "8:"
        "PrivateKeyFile" = "8:"
        "TimeStampServer" = "8:"
        "InstallerBootstrapper" = "3:2"
        }
        "Release"
        {
        "DisplayName" = "8:Release"
        "IsDebugOnly" = "11:FALSE"
        "IsReleaseOnly" = "11:TRUE"
```

```
        "OutputFilename" = "8:Release\\943_AptekaSetup.msi"
        "PackageFilesAs" = "3:2"
        "PackageFileSize" = "3:-2147483648"
        "CabType" = "3:1"
        "Compression" = "3:2"
        "SignOutput" = "11:FALSE"
        "CertificateFile" = "8:"
        "PrivateKeyFile" = "8:"
        "TimeStampServer" = "8:"
        "InstallerBootstrapper" = "3:2"
        }
    }
    "Deployable"
    {
        "CustomAction"
        {
        }
        "DefaultFeature"
        {
        "Name" = "8:DefaultFeature"
        "Title" = "8:"
        "Description" = "8:"
        }
        "ExternalPersistence"
        {
            "LaunchCondition"
            {
                "{A06ECF26-33A3-4562-8140-
9B0E340D4F24}:_549A8AA1F0424539820FA84F9182EF78"
                {
                "Name" = "8:.NET Framework"
                "Message" = "8:[VSDNETMSG]"
                "Version" = "8:2.0.50727"
                "AllowLaterVersions" = "11:FALSE"
                "InstallUrl" = "8:http://go.microsoft.com/fwlink/?LinkId=9832"
                }
            }
        }
        "File"
        {
            "{1FB2D0AE-D3B9-43D4-B9DD-
F88EC61E35DE}:_008A162E36EB45419D3B99FA33EA5CEB"
            {
            "SourcePath" = "8:..\\..\\..\\943_Apteka\\943_Apteka\\Icon\\anti-
virusoldschool_2906.ico"
            "TargetName" = "8:anti-virusoldschool_2906.ico"
            "Tag" = "8:"
            "Folder" = "8:_7C6FFC6550874FEAB708FBC5B25F0C20"
            "Condition" = "8:"
            "Transitive" = "11:FALSE"
            "Vital" = "11:TRUE"
            "ReadOnly" = "11:FALSE"
            "Hidden" = "11:FALSE"
            "System" = "11:FALSE"
            "Permanent" = "11:FALSE"
            "SharedLegacy" = "11:FALSE"
            "PackageAs" = "3:1"
            "Register" = "3:1"
            "Exclude" = "11:FALSE"
            "IsDependency" = "11:FALSE"
            "IsolateTo" = "8:"
            }
```

```
            "{9F6F8455-1EF1-4B85-886A-
4223BCC8E7F7}:_37EC5B91C09E4521A61EC12C0D60E4C4"
            {
            "AssemblyRegister" = "3:1"
            "AssemblyIsInGAC" = "11:FALSE"
            "AssemblyAsmDisplayName" = "8:943_Apteka, Version=1.0.0.0,
Culture=neutral, processorArchitecture=MSIL"
                "ScatterAssemblies"
                {
                    "_37EC5B91C09E4521A61EC12C0D60E4C4"
                    {
                    "Name" = "8:943_Apteka.exe"
                    "Attributes" = "3:512"
                    }
                }
            "SourcePath" =
"8:..\\..\\..\\943_Apteka\\943_Apteka\\bin\\Release\\943_Apteka.exe"
            "TargetName" = "8:"
            "Tag" = "8:"
            "Folder" = "8:_7C6FFC6550874FEAB708FBC5B25F0C20"
            "Condition" = "8:"
            "Transitive" = "11:FALSE"
            "Vital" = "11:TRUE"
            "ReadOnly" = "11:FALSE"
            "Hidden" = "11:FALSE"
            "System" = "11:FALSE"
            "Permanent" = "11:FALSE"
            "SharedLegacy" = "11:FALSE"
            "PackageAs" = "3:1"
            "Register" = "3:1"
            "Exclude" = "11:FALSE"
            "IsDependency" = "11:FALSE"
            "IsolateTo" = "8:"
            }
        }
        "FileType"
        {
        }
        "Folder"
        {
            "{1525181F-901A-416C-8A58-
119130FE478E}:_644B1EFAD4124C6C8A6A0D41AB8EDF96"
            {
            "Name" = "8:#1916"
            "AlwaysCreate" = "11:FALSE"
            "Condition" = "8:"
            "Transitive" = "11:FALSE"
            "Property" = "8:DesktopFolder"
                "Folders"
                {
                }
            }
            "{3C67513D-01DD-4637-8A68-
80971EB9504F}:_7C6FFC6550874FEAB708FBC5B25F0C20"
            {
            "DefaultLocation" = "8:[ProgramFilesFolder][Manufacturer]\\[ProductName]"
            "Name" = "8:#1925"
            "AlwaysCreate" = "11:FALSE"
            "Condition" = "8:"
            "Transitive" = "11:FALSE"
            "Property" = "8:TARGETDIR"
                "Folders"
```

```
                {
                }
            }
            "{1525181F-901A-416C-8A58-
119130FE478E}:_DFF9E5BB20B547AA8BA672ECC838778E"
            {
            "Name" = "8:#1919"
            "AlwaysCreate" = "11:FALSE"
            "Condition" = "8:"
            "Transitive" = "11:FALSE"
            "Property" = "8:ProgramMenuFolder"
                "Folders"
                {
                }
            }
        }
        "LaunchCondition"
        {
        }
        "Locator"
        {
        }
        "MsiBootstrapper"
        {
        "LangId" = "3:1033"
        }
        "Product"
        {
        "Name" = "8:Microsoft Visual Studio"
        "ProductName" = "8:943_AptekaSetup"
        "ProductCode" = "8:{28E4E3B7-F5E9-4099-AF5C-DD5060CFBDC3}"
        "PackageCode" = "8:{4341330E-A262-45F7-BC40-DE7EDD8351E1}"
        "UpgradeCode" = "8:{A1ACADC0-8634-41CE-8076-AB3AA4C4D255}"
        "RestartWWWService" = "11:FALSE"
        "RemovePreviousVersions" = "11:FALSE"
        "DetectNewerInstalledVersion" = "11:TRUE"
        "InstallAllUsers" = "11:FALSE"
        "ProductVersion" = "8:1.0.0"
        "Manufacturer" = "8:Julia Bazhura"
        "ARPHELPTELEPHONE" = "8:89537414025"
        "ARPHELPLINK" = "8:"
        "Title" = "8:943_AptekaSetup"
        "Subject" = "8:"
        "ARPCONTACT" = "8:Default Company Name"
        "Keywords" = "8:"
        "ARPCOMMENTS" = "8:"
        "ARPURLINFOABOUT" = "8:"
        "ARPPRODUCTICON" = "8:"
        "ARPIconIndex" = "3:0"
        "SearchPath" = "8:"
        "UseSystemSearchPath" = "11:TRUE"
        "TargetPlatform" = "3:0"
        "PreBuildEvent" = "8:"
        "PostBuildEvent" = "8:"
        "RunPostBuildEvent" = "3:0"
        }
        "Registry"
        {
            "HKLM"
            {
                "Keys"
                {
```

```
                    "{60EA8692-D2D5-43EB-80DC-
7906BF13D6EF}:_02F93E58B1984E9D940D339A8C2EFF46"
                    {
                    "Name" = "8:Software"
                    "Condition" = "8:"
                    "AlwaysCreate" = "11:FALSE"
                    "DeleteAtUninstall" = "11:FALSE"
                    "Transitive" = "11:FALSE"
                        "Keys"
                        {
                            "{60EA8692-D2D5-43EB-80DC-
7906BF13D6EF}:_D7A0938B0F8A4A68BC27CE47701653A4"
                            {
                            "Name" = "8:[Manufacturer]"
                            "Condition" = "8:"
                            "AlwaysCreate" = "11:FALSE"
                            "DeleteAtUninstall" = "11:FALSE"
                            "Transitive" = "11:FALSE"
                                "Keys"
                                {
                                }
                                "Values"
                                {
                                }
                            }
                        }
                        "Values"
                        {
                        }
                    }
                }
            }
            "HKCU"
            {
                "Keys"
                {
                    "{60EA8692-D2D5-43EB-80DC-
7906BF13D6EF}:_6B4B24B774A44DBE899584C9BF4FEB04"
                    {
                    "Name" = "8:Software"
                    "Condition" = "8:"
                    "AlwaysCreate" = "11:FALSE"
                    "DeleteAtUninstall" = "11:FALSE"
                    "Transitive" = "11:FALSE"
                        "Keys"
                        {
                            "{60EA8692-D2D5-43EB-80DC-
7906BF13D6EF}:_07D2DFE1875A476A88D155BCC8751CC2"
                            {
                            "Name" = "8:[Manufacturer]"
                            "Condition" = "8:"
                            "AlwaysCreate" = "11:FALSE"
                            "DeleteAtUninstall" = "11:FALSE"
                            "Transitive" = "11:FALSE"
                                "Keys"
                                {
                                }
                                "Values"
                                {
                                }
                            }
                        }
```

```
                    "Values"
                    {
                    }
                }
            }
            "HKCR"
            {
                "Keys"
                {
                }
            }
            "HKU"
            {
                "Keys"
                {
                }
            }
            "HKPU"
            {
                "Keys"
                {
                }
            }
        }
        "Sequences"
        {
        }
        "Shortcut"
        {
            "{970C0BB2-C7D0-45D7-ABFA-
7EC378858BC0}:_CED635E01CC9478AAD532B1B990C5439"
            {
            "Name" = "8:943_Apteka.exe"
            "Arguments" = "8:"
            "Description" = "8:"
            "ShowCmd" = "3:1"
            "IconIndex" = "3:0"
            "Transitive" = "11:FALSE"
            "Target" = "8:_37EC5B91C09E4521A61EC12C0D60E4C4"
            "Folder" = "8:_DFF9E5BB20B547AA8BA672ECC838778E"
            "WorkingFolder" = "8:_7C6FFC6550874FEAB708FBC5B25F0C20"
            "Icon" = "8:_008A162E36EB45419D3B99FA33EA5CEB"
            "Feature" = "8:"
            }
            "{970C0BB2-C7D0-45D7-ABFA-
7EC378858BC0}:_DF6E171162424145BAA46B9FA385F3B9"
            {
            "Name" = "8:943_Apteka.exe"
            "Arguments" = "8:"
            "Description" = "8:"
            "ShowCmd" = "3:1"
            "IconIndex" = "3:0"
            "Transitive" = "11:FALSE"
            "Target" = "8:_37EC5B91C09E4521A61EC12C0D60E4C4"
            "Folder" = "8:_644B1EFAD4124C6C8A6A0D41AB8EDF96"
            "WorkingFolder" = "8:_7C6FFC6550874FEAB708FBC5B25F0C20"
            "Icon" = "8:_008A162E36EB45419D3B99FA33EA5CEB"
            "Feature" = "8:"
            }
        }
        "UserInterface"
```

```
        {
            "{DF760B10-853B-4699-99F2-
AFF7185B4A62}:_58E7DAB348BC4BB0A14118B0AF03F6A6"
            {
            "Name" = "8:#1901"
            "Sequence" = "3:2"
            "Attributes" = "3:2"
                "Dialogs"
                {
                    "{688940B3-5CA9-4162-8DEE-
2993FA9D8CBC}:_59D0C3FD105341B5B57A338411DE501B"
                    {
                    "Sequence" = "3:100"
                    "DisplayName" = "8:Progress"
                    "UseDynamicProperties" = "11:TRUE"
                    "IsDependency" = "11:FALSE"
                    "SourcePath" = "8:<VsdDialogDir>\\VsdAdminProgressDlg.wid"
                        "Properties"
                        {
                            "BannerBitmap"
                            {
                            "Name" = "8:BannerBitmap"
                            "DisplayName" = "8:#1001"
                            "Description" = "8:#1101"
                            "Type" = "3:8"
                            "ContextData" = "8:Bitmap"
                            "Attributes" = "3:4"
                            "Setting" = "3:1"
                            "UsePlugInResources" = "11:TRUE"
                            }
                            "ShowProgress"
                            {
                            "Name" = "8:ShowProgress"
                            "DisplayName" = "8:#1009"
                            "Description" = "8:#1109"
                            "Type" = "3:5"
                            "ContextData" = "8:1;True=1;False=0"
                            "Attributes" = "3:0"
                            "Setting" = "3:0"
                            "Value" = "3:1"
                            "DefaultValue" = "3:1"
                            "UsePlugInResources" = "11:TRUE"
                            }
                        }
                    }
                }
            }
            "{DF760B10-853B-4699-99F2-
AFF7185B4A62}:_5E284867D92C42DCA6E1071B72C65195"
            {
            "Name" = "8:#1901"
            "Sequence" = "3:1"
            "Attributes" = "3:2"
                "Dialogs"
                {
                    "{688940B3-5CA9-4162-8DEE-
2993FA9D8CBC}:_D7F3E1513AF642898FFC205356A34F73"
                    {
                    "Sequence" = "3:100"
                    "DisplayName" = "8:Progress"
                    "UseDynamicProperties" = "11:TRUE"
                    "IsDependency" = "11:FALSE"
```

```
                    "SourcePath" = "8:<VsdDialogDir>\\VsdProgressDlg.wid"
                        "Properties"
                        {
                            "BannerBitmap"
                            {
                            "Name" = "8:BannerBitmap"
                            "DisplayName" = "8:#1001"
                            "Description" = "8:#1101"
                            "Type" = "3:8"
                            "ContextData" = "8:Bitmap"
                            "Attributes" = "3:4"
                            "Setting" = "3:1"
                            "UsePlugInResources" = "11:TRUE"
                            }
                            "ShowProgress"
                            {
                            "Name" = "8:ShowProgress"
                            "DisplayName" = "8:#1009"
                            "Description" = "8:#1109"
                            "Type" = "3:5"
                            "ContextData" = "8:1;True=1;False=0"
                            "Attributes" = "3:0"
                            "Setting" = "3:0"
                            "Value" = "3:1"
                            "DefaultValue" = "3:1"
                            "UsePlugInResources" = "11:TRUE"
                            }
                        }
                    }
                }
            }
            "{DF760B10-853B-4699-99F2-
AFF7185B4A62}:_6767E38AE5A24287AB51D8278FF86D06"
            {
            "Name" = "8:#1900"
            "Sequence" = "3:1"
            "Attributes" = "3:1"
                "Dialogs"
                {
                    "{688940B3-5CA9-4162-8DEE-
2993FA9D8CBC}:_118C23203BA244279E16B846EB5BD3D3"
                    {
                    "Sequence" = "3:300"
                    "DisplayName" = "8:Confirm Installation"
                    "UseDynamicProperties" = "11:TRUE"
                    "IsDependency" = "11:FALSE"
                    "SourcePath" = "8:<VsdDialogDir>\\VsdConfirmDlg.wid"
                        "Properties"
                        {
                            "BannerBitmap"
                            {
                            "Name" = "8:BannerBitmap"
                            "DisplayName" = "8:#1001"
                            "Description" = "8:#1101"
                            "Type" = "3:8"
                            "ContextData" = "8:Bitmap"
                            "Attributes" = "3:4"
                            "Setting" = "3:1"
                            "UsePlugInResources" = "11:TRUE"
                            }
                        }
                    }
```

```
                    "{688940B3-5CA9-4162-8DEE-
2993FA9D8CBC}:_65EA97DD784649819699330467C90D00"
                    {
                    "Sequence" = "3:200"
                    "DisplayName" = "8:Installation Folder"
                    "UseDynamicProperties" = "11:TRUE"
                    "IsDependency" = "11:FALSE"
                    "SourcePath" = "8:<VsdDialogDir>\\VsdFolderDlg.wid"
                        "Properties"
                        {
                            "BannerBitmap"
                            {
                            "Name" = "8:BannerBitmap"
                            "DisplayName" = "8:#1001"
                            "Description" = "8:#1101"
                            "Type" = "3:8"
                            "ContextData" = "8:Bitmap"
                            "Attributes" = "3:4"
                            "Setting" = "3:1"
                            "UsePlugInResources" = "11:TRUE"
                            }
                            "InstallAllUsersVisible"
                            {
                            "Name" = "8:InstallAllUsersVisible"
                            "DisplayName" = "8:#1059"
                            "Description" = "8:#1159"
                            "Type" = "3:5"
                            "ContextData" = "8:1;True=1;False=0"
                            "Attributes" = "3:0"
                            "Setting" = "3:0"
                            "Value" = "3:1"
                            "DefaultValue" = "3:1"
                            "UsePlugInResources" = "11:TRUE"
                            }
                        }
                    }
                    "{688940B3-5CA9-4162-8DEE-
2993FA9D8CBC}:_F9868AC17EAD457AAEBCCEDB08E36C45"
                    {
                    "Sequence" = "3:100"
                    "DisplayName" = "8:Welcome"
                    "UseDynamicProperties" = "11:TRUE"
                    "IsDependency" = "11:FALSE"
                    "SourcePath" = "8:<VsdDialogDir>\\VsdWelcomeDlg.wid"
                        "Properties"
                        {
                            "BannerBitmap"
                            {
                            "Name" = "8:BannerBitmap"
                            "DisplayName" = "8:#1001"
                            "Description" = "8:#1101"
                            "Type" = "3:8"
                            "ContextData" = "8:Bitmap"
                            "Attributes" = "3:4"
                            "Setting" = "3:1"
                            "UsePlugInResources" = "11:TRUE"
                            }
                            "CopyrightWarning"
                            {
                            "Name" = "8:CopyrightWarning"
                            "DisplayName" = "8:#1002"
                            "Description" = "8:#1102"
```

```
                            "Type" = "3:3"
                            "ContextData" = "8:"
                            "Attributes" = "3:0"
                            "Setting" = "3:1"
                            "Value" = "8:#1202"
                            "DefaultValue" = "8:#1202"
                            "UsePlugInResources" = "11:TRUE"
                            }
                            "Welcome"
                            {
                            "Name" = "8:Welcome"
                            "DisplayName" = "8:#1003"
                            "Description" = "8:#1103"
                            "Type" = "3:3"
                            "ContextData" = "8:"
                            "Attributes" = "3:0"
                            "Setting" = "3:2"
                            "Value" = "8:The installer will guide you through the
steps required to install 943_Apteka on your computer."
                            "DefaultValue" = "8:#1203"
                            "UsePlugInResources" = "11:TRUE"
                            }
                        }
                    }
                }
            }
            "{DF760B10-853B-4699-99F2-
AFF7185B4A62}:_8FF1AC5FF9A5476694595D3244C2BDE8"
            {
            "Name" = "8:#1900"
            "Sequence" = "3:2"
            "Attributes" = "3:1"
                "Dialogs"
                {
                    "{688940B3-5CA9-4162-8DEE-
2993FA9D8CBC}:_3099DB88EFDC4F84A936C2CED1669673"
                    {
                    "Sequence" = "3:300"
                    "DisplayName" = "8:Confirm Installation"
                    "UseDynamicProperties" = "11:TRUE"
                    "IsDependency" = "11:FALSE"
                    "SourcePath" = "8:<VsdDialogDir>\\VsdAdminConfirmDlg.wid"
                        "Properties"
                        {
                            "BannerBitmap"
                            {
                            "Name" = "8:BannerBitmap"
                            "DisplayName" = "8:#1001"
                            "Description" = "8:#1101"
                            "Type" = "3:8"
                            "ContextData" = "8:Bitmap"
                            "Attributes" = "3:4"
                            "Setting" = "3:1"
                            "UsePlugInResources" = "11:TRUE"
                            }
                        }
                    }
                    "{688940B3-5CA9-4162-8DEE-
2993FA9D8CBC}:_B7486A3EFA964B09BD2EF215EA55E1DC"
                    {
                    "Sequence" = "3:100"
                    "DisplayName" = "8:Welcome"
```

```
                    "UseDynamicProperties" = "11:TRUE"
                    "IsDependency" = "11:FALSE"
                    "SourcePath" = "8:<VsdDialogDir>\\VsdAdminWelcomeDlg.wid"
                        "Properties"
                        {
                            "BannerBitmap"
                            {
                            "Name" = "8:BannerBitmap"
                            "DisplayName" = "8:#1001"
                            "Description" = "8:#1101"
                            "Type" = "3:8"
                            "ContextData" = "8:Bitmap"
                            "Attributes" = "3:4"
                            "Setting" = "3:1"
                            "UsePlugInResources" = "11:TRUE"
                            }
                            "CopyrightWarning"
                            {
                            "Name" = "8:CopyrightWarning"
                            "DisplayName" = "8:#1002"
                            "Description" = "8:#1102"
                            "Type" = "3:3"
                            "ContextData" = "8:"
                            "Attributes" = "3:0"
                            "Setting" = "3:1"
                            "Value" = "8:#1202"
                            "DefaultValue" = "8:#1202"
                            "UsePlugInResources" = "11:TRUE"
                            }
                            "Welcome"
                            {
                            "Name" = "8:Welcome"
                            "DisplayName" = "8:#1003"
                            "Description" = "8:#1103"
                            "Type" = "3:3"
                            "ContextData" = "8:"
                            "Attributes" = "3:0"
                            "Setting" = "3:1"
                            "Value" = "8:#1203"
                            "DefaultValue" = "8:#1203"
                            "UsePlugInResources" = "11:TRUE"
                            }
                        }
                    }
                    "{688940B3-5CA9-4162-8DEE-
2993FA9D8CBC}:_F284A32D38FB4083A8A457F2EB68DD2F"
                    {
                    "Sequence" = "3:200"
                    "DisplayName" = "8:Installation Folder"
                    "UseDynamicProperties" = "11:TRUE"
                    "IsDependency" = "11:FALSE"
                    "SourcePath" = "8:<VsdDialogDir>\\VsdAdminFolderDlg.wid"
                        "Properties"
                        {
                            "BannerBitmap"
                            {
                            "Name" = "8:BannerBitmap"
                            "DisplayName" = "8:#1001"
                            "Description" = "8:#1101"
                            "Type" = "3:8"
                            "ContextData" = "8:Bitmap"
                            "Attributes" = "3:4"
```

```
                            "Setting" = "3:1"
                            "UsePlugInResources" = "11:TRUE"
                            }
                        }
                    }
                }
            }
            "{DF760B10-853B-4699-99F2-
AFF7185B4A62}:_97AFD64F22C64416B44D57C4EFDDA638"
            {
            "Name" = "8:#1902"
            "Sequence" = "3:1"
            "Attributes" = "3:3"
                "Dialogs"
                {
                    "{688940B3-5CA9-4162-8DEE-
2993FA9D8CBC}:_4627A7E01E3048498827B817AEEDF75D"
                    {
                    "Sequence" = "3:100"
                    "DisplayName" = "8:Finished"
                    "UseDynamicProperties" = "11:TRUE"
                    "IsDependency" = "11:FALSE"
                    "SourcePath" = "8:<VsdDialogDir>\\VsdFinishedDlg.wid"
                        "Properties"
                        {
                            "BannerBitmap"
                            {
                            "Name" = "8:BannerBitmap"
                            "DisplayName" = "8:#1001"
                            "Description" = "8:#1101"
                            "Type" = "3:8"
                            "ContextData" = "8:Bitmap"
                            "Attributes" = "3:4"
                            "Setting" = "3:1"
                            "UsePlugInResources" = "11:TRUE"
                            }
                            "UpdateText"
                            {
                            "Name" = "8:UpdateText"
                            "DisplayName" = "8:#1058"
                            "Description" = "8:#1158"
                            "Type" = "3:15"
                            "ContextData" = "8:"
                            "Attributes" = "3:0"
                            "Setting" = "3:2"
                            "Value" = "8:Thank you for your choice!"
                            "DefaultValue" = "8:#1258"
                            "UsePlugInResources" = "11:TRUE"
                            }
                        }
                    }
                }
            }
            "{2479F3F5-0309-486D-8047-
8187E2CE5BA0}:_A25E9B14F6ED4D2093B8D54A8D81F7AC"
            {
            "UseDynamicProperties" = "11:FALSE"
            "IsDependency" = "11:FALSE"
            "SourcePath" = "8:<VsdDialogDir>\\VsdBasicDialogs.wim"
            }
            "{DF760B10-853B-4699-99F2-
AFF7185B4A62}:_DB06A2B169B0470CA6DD76F67BAC8093"
```

```
            {
            "Name" = "8:#1902"
            "Sequence" = "3:2"
            "Attributes" = "3:3"
                "Dialogs"
                {
                    "{688940B3-5CA9-4162-8DEE-
2993FA9D8CBC}:_80A4D467B7714B24BF12D58570077DC4"
                    {
                    "Sequence" = "3:100"
                    "DisplayName" = "8:Finished"
                    "UseDynamicProperties" = "11:TRUE"
                    "IsDependency" = "11:FALSE"
                    "SourcePath" = "8:<VsdDialogDir>\\VsdAdminFinishedDlg.wid"
                        "Properties"
                        {
                            "BannerBitmap"
                            {
                            "Name" = "8:BannerBitmap"
                            "DisplayName" = "8:#1001"
                            "Description" = "8:#1101"
                            "Type" = "3:8"
                            "ContextData" = "8:Bitmap"
                            "Attributes" = "3:4"
                            "Setting" = "3:1"
                            "UsePlugInResources" = "11:TRUE"
                            }
                        }
                    }
                }
            }
            "{2479F3F5-0309-486D-8047-
8187E2CE5BA0}:_DF2B90D833FD47679B58DEB59A507E79"
            {
            "UseDynamicProperties" = "11:FALSE"
            "IsDependency" = "11:FALSE"
            "SourcePath" = "8:<VsdDialogDir>\\VsdUserInterface.wim"
            }
        }
        "MergeModule"
        {
        }
        "ProjectOutput"
        {
        }
        "VJSharpPlugin"
        {
        }
    }
}
```