МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ И ТЕХНОЛОГИЙ

КУРСОВАЯ РАБОТА (ПРОЕКТ)   
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| канд. техн. наук, доцент |  |  |  | А.В. Бржезовский |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ (ПРОЕКТУ) | | | | |
| РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ АВТОДИЛЕРА | | | | |
| по дисциплине: МЕТОДЫ И СРЕДСТВА ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ И ТЕХНОЛОГИЙ | | | | |
|  | | | | |
|  | | | | |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ гр. № | 4121 |  |  |  | А.В. Ветохина |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

**СОДЕРЖАНИЕ**

[РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ АВТОДИЛЕРА 1](#_Toc185516644)

[ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 3](#_Toc185516645)

[1.1 Исходные данные 3](#_Toc185516646)

[1.2 Возможности бизнеса 3](#_Toc185516647)

[1.3 Бизнес-цели 3](#_Toc185516648)

[1.4 Критерии успеха 4](#_Toc185516649)

[1.5 Положение о концепции 4](#_Toc185516650)

[ТРЕБОВАНИЯ К СИСТЕМЕ 5](#_Toc185516651)

[2.1 Классы пользователей 5](#_Toc185516652)

[2.2 Определение пользовательских требований 6](#_Toc185516653)

[КОНЦЕПТУАЛЬНАЯ МОДЕЛЬ БАЗЫ ДАННЫХ 7](#_Toc185516654)

[ФИЗИЧЕСКАЯ МОДЕЛЬ БАЗЫ ДАННЫХ 10](#_Toc185516655)

[4.1 Хранимые процедуры 10](#_Toc185516656)

[4.2 Триггеры 11](#_Toc185516657)

[4.3 Индексы 12](#_Toc185516658)

[4.4 Ограничения целостности 12](#_Toc185516659)

[ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ 14](#_Toc185516660)

[ПРИЛОЖЕНИЕ А. Создание объектов базы данных 15](#_Toc185516661)

[ПРИЛОЖЕНИЕ Б. Тексты программ 20](#_Toc185516662)

# ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

# 1.1 Исходные данные

В дилерском центре в настоящее время бизнес-процессы, такие как учет проданных автомобилей и автомобилей в наличии, учет сотрудников и многое прочее, ведутся в письменном виде.

Необходимо создать корпоративную информационную систему, для дилерского центра которая будет охватывать все необходимые бизнес-процессы.

# 1.2 Возможности бизнеса

Сотрудники дилерского центра попросили создать такую систему, которая позволила бы автоматизировать учет автомобилей и клиентов. Эта система могла бы включать возможность выбора типа обслуживания (например, плановое ТО, диагностика или ремонт), указания удобной даты и времени, а также возможность выбора дополнительных услуг, таких как мойка или полировка.

Такая инициатива значительно упростит процесс записи для клиентов, позволяя им избежать ожидания в очереди и планировать свои визиты в дилерский центр. Это не только повысит уровень удовлетворенности клиентов, но и увеличит поток записей, что в конечном итоге может привести к росту доходов центра.

# 1.3 Бизнес-цели

БЦ-1: Автоматизация процессов записи на обслуживание, управления запасами и учета продаж для сокращения времени на выполнение рутинных задач;

БЦ-2: Внедрение системы мониторинга запасов запчастей и расходных материалов для снижения затрат и предотвращения дефицита;

БЦ-3: Автоматизация процесса заказа запчастей у поставщиков на основе анализа спроса;

БЦ-4: Внедрение системы управления документами для упрощения хранения и обработки документации.

# 1.4 Критерии успеха

КУ-1: Достижение уровня удовлетворенности пользователей не ниже 80%;

КУ-2: Сокращение времени обработки заказов на обслуживание и записи клиентов на услуги на 30% по сравнению с предыдущими процессами;

КУ-3: Оптимизация процессов, что приведет к снижению количества ошибок и перерасхода ресурсов.

# 1.5 Положение о концепции

Для сотрудников, которые стремятся повысить уровень обслуживания клиентов и оптимизировать процессы продаж, эта информационная система управления взаимоотношениями с клиентами. Является программным обеспечением для автодилеров, которое автоматизирует процесс обработки заявок, отслеживания продаж и управления клиентскими данными, что позволяет значительно улучшить качество обслуживания и увеличить объем продаж.

В отличие от традиционных методов работы с клиентами, которые часто требуют значительных временных затрат и подвержены ошибкам, наш продукт предлагает интеграцию с существующими системами и интуитивно понятный интерфейс, что позволяет сотрудникам более эффективно взаимодействовать с клиентами и принимать обоснованные решения на основе данных.

# ТРЕБОВАНИЯ К СИСТЕМЕ

# 2.1 Классы пользователей

На таблице 1 представлены основные классы пользователей и функциональные требования к системе.

Таблица 1 – Функциональные требования

|  |  |  |
| --- | --- | --- |
| Функция | Класс пользователя | Описание |
| Ф1: Авторизация в системе | Все пользователи системы | Пользователь системы вводит логин и пароль |
| Ф2: Ведение учета клиентской базы | Все пользователи системы | Поиск клиентов в базе данных по фамилии, добавление, редактирование и удаление клиентов |
| Ф3: Управление запасами | Руководитель, бухгалтер и менеджер по продажам | Поиск автомобиля и базе данных по марке и модели, поиск доступных запчастей по названию |
| Ф4: Ведение учета базы сотрудников | Руководитель | Поиск сотрудник в базе данных по фамилии или должности, добавление, редактирование и удаление сотрудника |
| Ф5: Ведение учета обслуживания и ремонта автомобиля | Техник, руководитель и бухгалтер | Регистрация заявок на обслуживание и ремонт, а также запись на ремонт и техническое обслуживание |
| Ф6: Финансовый учет | Бухгалтер и руководитель | Ведение учета всех финансовых операций, связанных с обслуживанием и ремонтом автомобилей. |
| Ф7: Регистрация в системе | Все пользователи системы | Добавление данных в систему |

# 2.2 Определение пользовательских требований

Чтобы описать взаимодействия между пользователями системы и базой данных была разработана use-case диаграмма, представленная на рисунке 1.

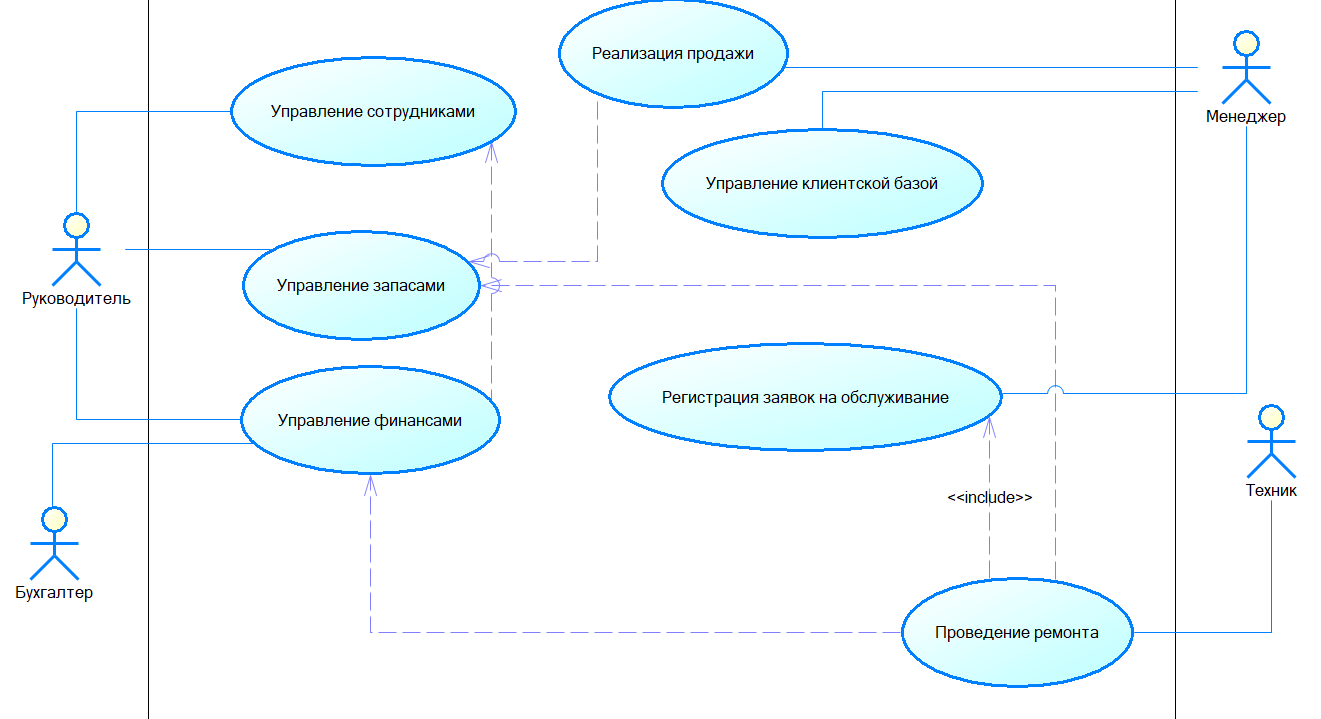


Рисунок 1 – use-case диаграмма

Далее представлены описания вариантов использования каждой функции.

1. Функция авторизация в системе

Краткое описание: прецедент описывает процесс авторизации пользователя в системе, включая ввод учетных данных, проверку их корректности и доступ к функционалу системы.

Актеры: Пользователь

Предусловия: Пользователь должен иметь учетную запись в системе.

Триггер: Пользователь инициирует процесс авторизации через интерфейс системы.

Основной поток событий:

1. Пользователь открывает приложение системы.

2. Система отображает страницу входа с полями для ввода логина и пароля.

3. Пользователь вводит свои учетные данные (логин и пароль).

4. Пользователь нажимает кнопку «Войти».

5. Система проверяет введенные учетные данные на корректность.

*Если учетные данные верны:*

6. Система перенаправляет пользователя на главную страницу приложения.

7. Пользователь получает доступ ко всем функциям системы в соответствии с его правами.

*Если учетные данные неверны:*

6. Система выводит сообщение об ошибке, информируя пользователя о некорректных данных.

7. Пользователь имеет возможность повторно ввести учетные данные.

Альтернативные потоки:

4а. Если пользователь оставил поля логина или пароля пустыми:

1. Система выводит сообщение об ошибке, предлагая заполнить все обязательные поля.

6а. Если пользователь вводит неверный логин или пароль:

1. Система выводит сообщение об ошибке, информируя о некорректных данных и предлагает повторить попытку входа.

Постусловия:

Пользователь успешно авторизован в системе: он имеет доступ к функционалу приложения, а система сохраняет информацию о текущем сеансе пользователя для дальнейшего взаимодействия.

1. Функция ведение учета клиентской базы

Краткое описание: прецедент описывает процесс ведения учета клиентской базы в дилерском центре, включая создание, редактирование, просмотр и удаление записей о клиентах.

Актеры: руководитель, техник, менеджер по продажам, бухгалтер.

Предусловия: пользователь должен быть авторизован в системе и иметь доступ к функционалу управления клиентской базой.

Триггер: пользователь инициирует управление клиентской базой через интерфейс системы.

Основной поток событий:

1. Пользователь открывает приложение системы учета клиентской базы.

2. Система отображает главную страницу с списком клиентов.

3. Пользователь выбирает действие: создать новую запись о клиенте, редактировать существующую запись или удалить запись.

*Если пользователь выбирает создание новой записи:*

4. Система отображает форму для создания новой записи о клиенте.

5. Пользователь заполняет необходимые.

6. Пользователь нажимает кнопку «Создать клиента».

7. Система сохраняет новую запись в базе данных и обновляет список клиентов, отображая нового клиента.

*Если пользователь выбирает редактирование записи:*

4. Система отображает список клиентов.

5. Пользователь выбирает клиента для редактирования и нажимает кнопку «Редактировать».

6. Система открывает форму редактирования с текущими данными клиента.

7. Пользователь вносит изменения и нажимает кнопку «Сохранить изменения».

8. Система обновляет данные клиента в базе данных и обновляет список клиентов.

*Если пользователь выбирает удаление записи:*

4. Система отображает список клиентов.

5. Пользователь выбирает клиента для удаления и нажимает кнопку «Удалить».

6. Система запрашивает подтверждение удаления.

7. Пользователь подтверждает удаление.

8. Система удаляет запись о клиенте из базы данных и обновляет список клиентов.

Альтернативные потоки:

4а. Если пользователь не заполнил обязательные поля при создании записи:

1. Система выводит сообщение об ошибке и предлагает заполнить все обязательные поля.

6а. Если пользователь пытается удалить запись, которая не существует:

1. Система выводит сообщение об ошибке, что запись не найдена.

Постусловия:

Пользователь успешно управляет клиентской базой в системе: новая запись создана, существующая запись изменена или удалена, а список клиентов обновлен в соответствии с выполненными действиями.

1. Функция управление запасами

Краткое описание: Данный прецедент описывает процесс поиска автомобиля по марке и модели, а также поиск доступных запчастей по названию в базе данных дилерского центра.

Актеры: Пользователь (менеджер по продажам, руководитель, бухгалтер)

Предусловия: Пользователь должен быть авторизован в системе и иметь доступ к функционалу поиска автомобилей и запчастей.

Триггер: Пользователь инициирует поиск через интерфейс системы.

Основной поток событий:

1. Пользователь открывает приложение системы.

2. Система отображает главную страницу с меню поиска.

3. Пользователь выбирает действие: поиск автомобиля или поиск запчастей.

*Если пользователь выбирает поиск автомобиля:*

4. Система отображает форму для поиска автомобиля.

5. Пользователь вводит марку и модель автомобиля в соответствующие поля.

6. Пользователь нажимает кнопку «Поиск».

7. Система выполняет запрос к базе данных и отображает список автомобилей, соответствующих введенным критериям.

*Если автомобили найдены:*

8. Система отображает информацию о найденных автомобилях (марка, модель, год выпуска, состояние и т.д.).

*Если автомобили не найдены:*

8. Система выводит сообщение о том, что автомобили не найдены по указанным критериям.

*Если пользователь выбирает поиск запчастей:*

4. Система отображает форму для поиска запчастей.

5. Пользователь вводит название запчасти в соответствующее поле.

6. Пользователь нажимает кнопку «Поиск».

7. Система выполняет запрос к базе данных и отображает список доступных запчастей, соответствующих введенному названию.

*Если запчасти найдены:*

8. Система отображает информацию о найденных запчастях (название, артикул, цена, наличие на складе и т.д.).

*Если запчасти не найдены:*

8. Система выводит сообщение о том, что запчасти не найдены по указанному названию.

Альтернативные потоки:

5а. Если пользователь не заполнил обязательные поля при поиске автомобиля или запчастей:

1. Система выводит сообщение об ошибке и предлагает заполнить все обязательные поля.

7а. Если произошла ошибка при выполнении запроса к базе данных:

1. Система выводит сообщение об ошибке и предлагает повторить попытку позже.

Постусловия:

Пользователь успешно выполнил поиск: информация о найденных автомобилях или запчастях отображена в системе, либо выведено сообщение об отсутствии результатов по указанным критериям.

1. Функция ведение учета базы сотрудников

Краткое описание: функция ведения учета базы сотрудников в дилерском центре включает процессы создания, редактирования, просмотра и удаления записей о сотрудниках. Эта функция обеспечивает эффективное управление информацией о персонале, что способствует улучшению взаимодействия между различными отделами и повышению общей продуктивности работы компании.

Актеры: руководитель, техник, менеджер по продажам, бухгалтер.

Предусловия: пользователь должен быть авторизован в системе и иметь доступ к функционалу управления базой сотрудников. Это гарантирует безопасность данных и ограничивает доступ к информации только для уполномоченных лиц.

Триггер: пользователь инициирует управление базой сотрудников через интерфейс системы, что запускает соответствующие процессы.

Основной поток событий

1. Пользователь открывает приложение системы учета базы сотрудников.

2. Система отображает главную страницу с списком сотрудников.

3. Пользователь выбирает одно из действий: создать новую запись о сотруднике, редактировать существующую запись, удалить запись.

*Если пользователь выбирает создание новой записи:*

4. Система отображает форму для создания новой записи о сотруднике.

5. Пользователь заполняет необходимые поля (ФИО, должность, контактные данные и т.д.).

6. Пользователь нажимает кнопку «Создать сотрудника».

7. Система сохраняет новую запись в базе данных и обновляет список сотрудников, отображая нового сотрудника.

*Если пользователь выбирает редактирование записи:*

4. Система отображает список сотрудников.

5. Пользователь выбирает сотрудника для редактирования и нажимает кнопку «Редактировать».

6. Система открывает форму редактирования с текущими данными сотрудника.

7. Пользователь вносит изменения и нажимает кнопку «Сохранить изменения».

8. Система обновляет данные сотрудника в базе данных и обновляет список сотрудников.

*Если пользователь выбирает удаление записи:*

4. Система отображает список сотрудников.

5. Пользователь выбирает сотрудника для удаления и нажимает кнопку «Удалить».

6. Система запрашивает подтверждение удаления.

7. Пользователь подтверждает удаление.

8. Система удаляет запись о сотруднике из базы данных и обновляет список сотрудников.

Альтернативные потоки

*Если пользователь не заполнил обязательные поля при создании записи:*

1. Система выводит сообщение об ошибке и предлагает заполнить все обязательные поля.

*Если пользователь пытается удалить запись, которая не существует:*

1. Система выводит сообщение об ошибке, что запись не найдена.

Постусловия: пользователь успешно управляет базой сотрудников в системе: новая запись создана, существующая запись изменена или удалена, а список сотрудников обновлен в соответствии с выполненными действиями.

1. Функция ведение учета обслуживания и ремонта автомобиля

Краткое описание: функция ведения учета обслуживания и ремонта автомобиля в дилерском центре включает процессы создания, редактирования, просмотра и удаления записей о проведенных работах. Эта функция обеспечивает систематизацию информации о техническом обслуживании и ремонте автомобилей, что способствует повышению качества услуг и удовлетворенности клиентов.

Актеры: техник, руководитель, бухгалтер.

Предусловия: пользователь должен быть авторизован в системе и иметь доступ к функционалу управления учетом обслуживания и ремонта автомобилей. Это гарантирует защиту данных и ограничивает доступ к информации только для уполномоченных пользователей.

Триггер: пользователь инициирует управление учетом обслуживания и ремонта автомобиля через интерфейс системы, что запускает соответствующие процессы.

Основной поток событий

1. Пользователь открывает приложение системы учета обслуживания и ремонта автомобилей.

2. Система отображает главную страницу с списком автомобилей и их статусом обслуживания.

3. Пользователь выбирает действие: создать новую запись о проведенном обслуживании или ремонте, редактировать существующую запись, удалить запись.

*Если пользователь выбирает создание новой записи:*

4. Система отображает форму для создания новой записи о проведенном обслуживании или ремонте.

5. Пользователь заполняет необходимые поля (информация об автомобиле, тип услуги, дата проведения работ, стоимость и т.д.).

6. Пользователь нажимает кнопку «Создать запись».

7. Система сохраняет новую запись в базе данных и обновляет список обслуживаемых автомобилей, отображая новую запись.

*Если пользователь выбирает редактирование записи:*

4. Система отображает список записей об обслуживании и ремонте.

5. Пользователь выбирает запись для редактирования и нажимает кнопку «Редактировать».

6. Система открывает форму редактирования с текущими данными записи.

7. Пользователь вносит изменения и нажимает кнопку «Сохранить изменения».

8. Система обновляет данные записи в базе данных и обновляет список записей об обслуживании.

*Если пользователь выбирает удаление записи:*

4. Система отображает список записей об обслуживании и ремонте.

5. Пользователь выбирает запись для удаления и нажимает кнопку «Удалить».

6. Система запрашивает подтверждение удаления.

7. Пользователь подтверждает удаление.

8. Система удаляет запись из базы данных и обновляет список записей об обслуживании.

Альтернативные потоки

*Если пользователь не заполнил обязательные поля при создании записи:*

1. Система выводит сообщение об ошибке и предлагает заполнить все обязательные поля.

*Если пользователь пытается удалить запись, которая не существует:*

1. Система выводит сообщение об ошибке, что запись не найдена.

Постусловия: пользователь успешно управляет учетом обслуживания и ремонта автомобилей в системе: новая запись создана, существующая запись изменена или удалена, а список записей обновлен в соответствии с выполненными действиями. Это обеспечивает актуальность информации о состоянии автомобилей, что критически важно для эффективного управления сервисным обслуживанием.

1. Функция финансовый учет

Краткое описание: функция финансового учета в дилерском центре включает процессы записи, анализа и отчетности по финансовым операциям. Эта функция обеспечивает прозрачность и контроль за финансовыми потоками, что способствует более эффективному управлению ресурсами и принятию обоснованных решений.

Актеры: руководитель, бухгалтер.

Предусловия: пользователь должен быть авторизован в системе и иметь доступ к функционалу управления финансовым учетом. Это гарантирует безопасность данных и ограничивает доступ к информации только для уполномоченных лиц.

Триггер: пользователь инициирует управление финансовым учетом через интерфейс системы, что запускает соответствующие процессы.

Основной поток событий:

1. Пользователь открывает приложение системы финансового учета.

2. Система отображает главную страницу с основными показателями финансовых операций (доходы, расходы, прибыль).

3. Пользователь выбирает действие: создать новую запись о финансовой операции, редактировать существующую запись, удалить запись.

*Если пользователь выбирает создание новой записи:*

4. Система отображает форму для создания новой записи о финансовой операции.

5. Пользователь заполняет необходимые поля (тип операции, сумма, дата, описание и т.д.).

6. Пользователь нажимает кнопку «Создать запись».

7. Система сохраняет новую запись в базе данных и обновляет список финансовых операций, отображая новую запись.

*Если пользователь выбирает редактирование записи:*

4. Система отображает список записей о финансовых операциях.

5. Пользователь выбирает запись для редактирования и нажимает кнопку «Редактировать».

6. Система открывает форму редактирования с текущими данными записи.

7. Пользователь вносит изменения и нажимает кнопку «Сохранить изменения».

8. Система обновляет данные записи в базе данных и обновляет список записей о финансовых операциях.

*Если пользователь выбирает удаление записи:*

4. Система отображает список записей о финансовых операциях.

5. Пользователь выбирает запись для удаления и нажимает кнопку «Удалить».

6. Система запрашивает подтверждение удаления.

7. Пользователь подтверждает удаление.

8. Система удаляет запись из базы данных и обновляет список записей о финансовых операциях.

Альтернативные потоки

*Если пользователь не заполнил обязательные поля при создании записи:*

1. Система выводит сообщение об ошибке и предлагает заполнить все обязательные поля.

*Если пользователь пытается удалить запись, которая не существует:*

1. Система выводит сообщение об ошибке, что запись не найдена.

Постусловия: пользователь успешно управляет финансовым учетом в системе: новая запись создана, существующая запись изменена или удалена, а список записей обновлен в соответствии с выполненными действиями. Это обеспечивает точность и актуальность данных о финансовых операциях, что критически важно для эффективного управления финансами компании.

1. Функция регистрация в системе

Краткое описание: функция регистрации в системе позволяет новым пользователям создавать учетные записи для доступа к функционалу приложения. Этот процесс включает ввод необходимых данных, верификацию информации и создание профиля пользователя, что обеспечивает безопасность и управление доступом к системе.

Актеры: новый пользователь.

Предусловия: пользователь должен иметь доступ к интерфейсу регистрации. Необходимо наличие интернет-соединения для взаимодействия с системой.

Триггер: пользователь инициирует процесс регистрации через интерфейс системы, нажав на кнопку «Регистрация».

Основной поток событий:

1. Пользователь открывает приложение и переходит на страницу регистрации.

2. Система отображает форму для регистрации нового пользователя.

3. Пользователь заполняет необходимые поля (имя, фамилия, электронная почта, пароль и т.д.).

4. Пользователь нажимает кнопку «Зарегистрироваться».

5. Система проверяет корректность введенных данных:

5.1 *Если все поля заполнены правильно и уникальность электронной почты подтверждена*: система продолжает процесс.

5.2 *Если есть ошибки:* система выводит сообщение об ошибке и предлагает исправить данные.

6. Система создает новую учетную запись в базе данных.

7. Система активирует учетную запись пользователя и предоставляет доступ к функционалу системы.

Альтернативные потоки

*Если пользователь не заполнил обязательные поля при регистрации:*

1. Система выводит сообщение об ошибке и предлагает заполнить все обязательные поля.

*Если введенная электронная почта уже используется:*

1. Система выводит сообщение об ошибке, что данная электронная почта уже зарегистрирована.

Постусловия: пользователь успешно зарегистрирован в системе: новая учетная запись создана, данные проверены и сохранены в базе данных, а доступ к функционалу системы.

# КОНЦЕПТУАЛЬНАЯ МОДЕЛЬ БАЗЫ ДАННЫХ

Для определения требований к данным информационной системы автодилера была разработана концептуальная модель данных, которая иллюстрирует логические группы информации в предметной области и связи между ними. Модель была разработана в программной среде PowerDesigner.

На рисунке 2 представлена разработанная концептуальная модель базы данных.

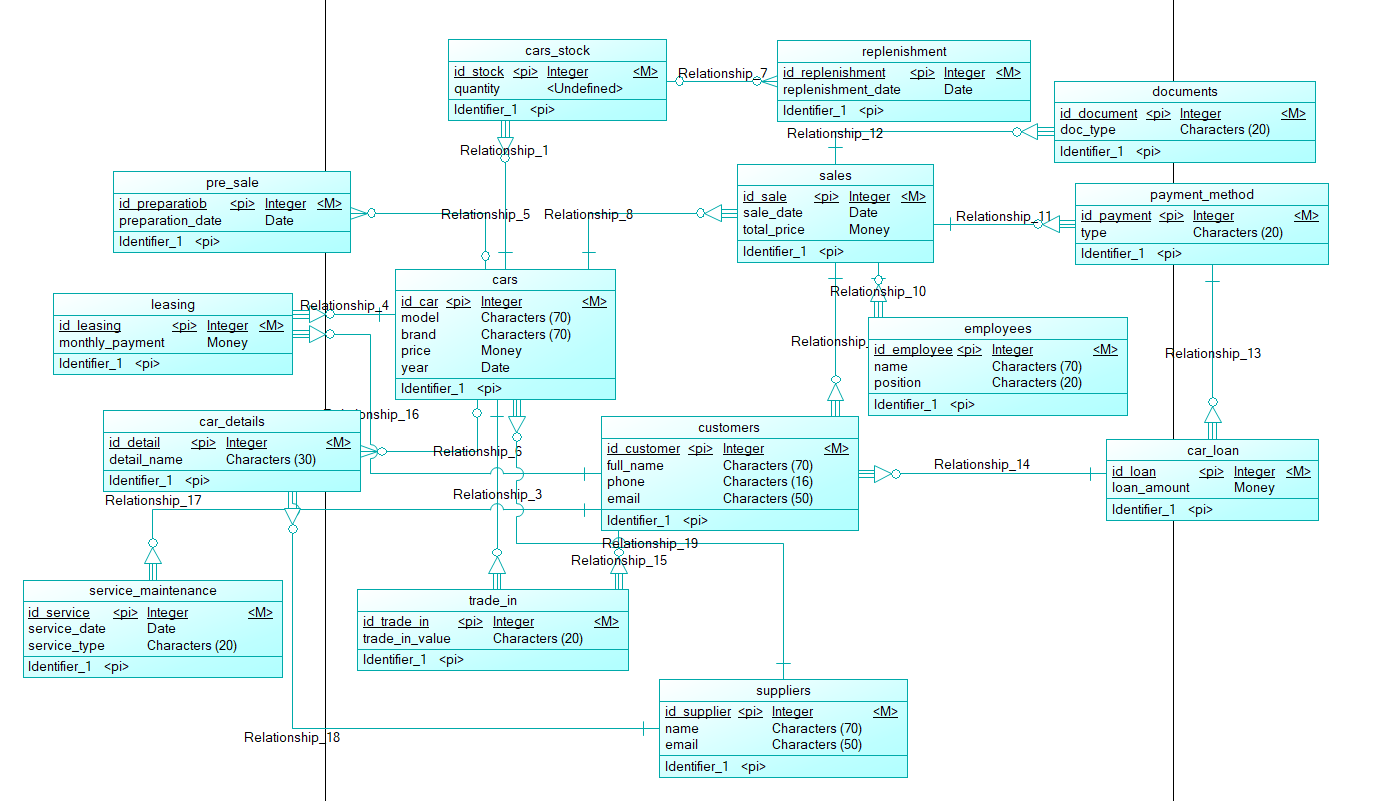


Рисунок 2 – Концептуальная модель данных

Концептуальная модель содержит 15 таблиц. Далее представлено описание каждой из них:

1. cars\_stock: хранит информацию о наличии автомобилей на складе.

Содержит поля: id\_stock: уникальный идентификатор записи; quantity: количество автомобилей на складе.

1. replenishment: отражает данные о пополнении склада.

Поля: id\_replenishment: уникальный идентификатор пополнения; replenishment\_date: дата пополнения.

1. sales: содержит данные о продажах автомобилей.

Поля: id\_sale: уникальный идентификатор продажи; sale\_date: дата продажи; total\_price: общая стоимость продажи.

1. documents: хранит информацию о документах, связанных с продажами.

Поля: id\_document: уникальный идентификатор документа; doc\_type: тип документа.

1. payment\_method: содержит данные о способах оплаты.

Поля: id\_payment: уникальный идентификатор способа оплаты; type: тип оплаты.

1. car\_loan: информация об автокредитах.

Поля: id\_loan: уникальный идентификатор кредита; loan\_amount: сумма кредита.

1. employees: данные о сотрудниках автосалона.

Поля: id\_employee: уникальный идентификатор сотрудника; name: имя сотрудника; position: должность.

1. customers: информация о клиентах.

Поля: id\_customer: уникальный идентификатор клиента; full\_name: полное имя клиента; phone: номер телефона; email: адрес электронной почты.

1. cars: основная информация об автомобилях.

Поля: id\_car: уникальный идентификатор автомобиля; model, brand, price, year: модель, марка, цена и год выпуска автомобиля.

1. leasing: данные об аренде автомобилей.

Поля: id\_leasing: уникальный идентификатор аренды; monthly\_payment: ежемесячный платеж.

1. car\_details: характеристики автомобилей.

Поля: id\_detail: уникальный идентификатор характеристики; detail\_name: название характеристики.

1. service\_maintenance: информация об обслуживании автомобилей.

Поля: id\_service: уникальный идентификатор обслуживания; service\_date: дата обслуживания; service\_type: тип обслуживания.

1. trade\_in: данные о программе обмена старых автомобилей (trade-in).

Поля: id\_trade\_in: уникальный идентификатор обмена; trade\_in\_value: оценочная стоимость автомобиля по программе trade-in.

1. suppliers: Информация о поставщиках автомобилей или запчастей.

Поля: id\_supplier: уникальный идентификатор поставщика; name, email: имя и электронная почта поставщика.

1. pre\_sale: Данные о подготовке автомобилей к продаже.

Поля: id\_preparation: уникальный идентификатор подготовки; preparation\_date: дата подготовки автомобиля к продаже.

Также был разработан словарь данных, результат представлен на рисунке 3.

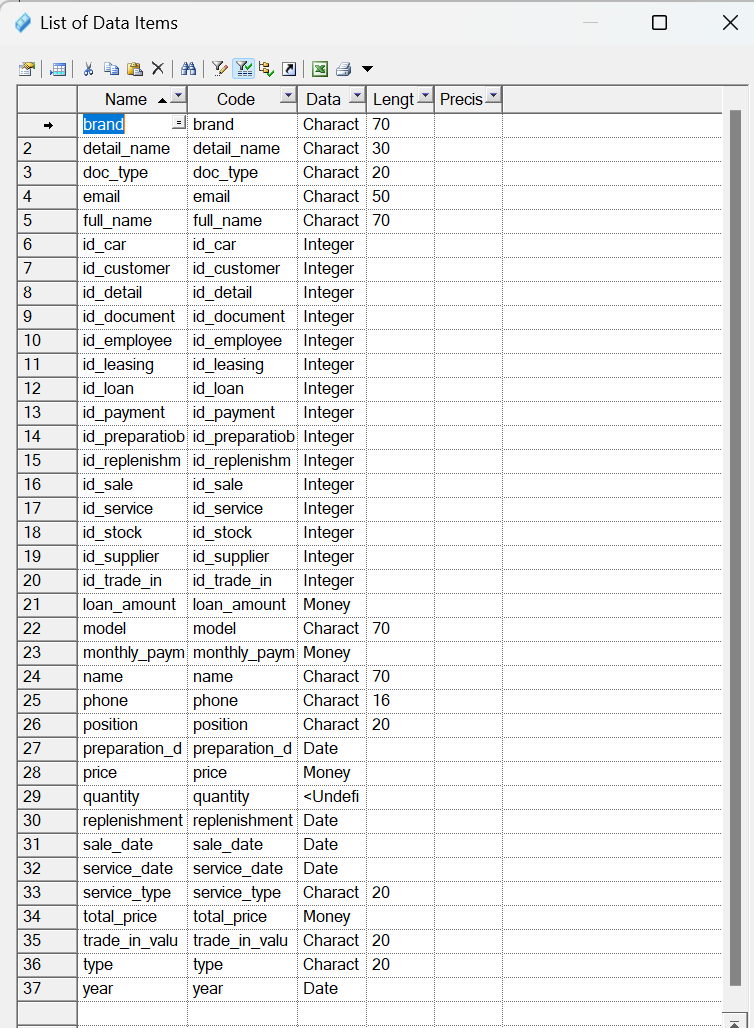


Рисунок 3 – Словарь данных

# ФИЗИЧЕСКАЯ МОДЕЛЬ БАЗЫ ДАННЫХ

Физическая модель базы данных, соответствующая изначальной концептуальной модели, представлена на рисунке 4. SQL-скрипт для реализации этой модели приведён в приложении A.

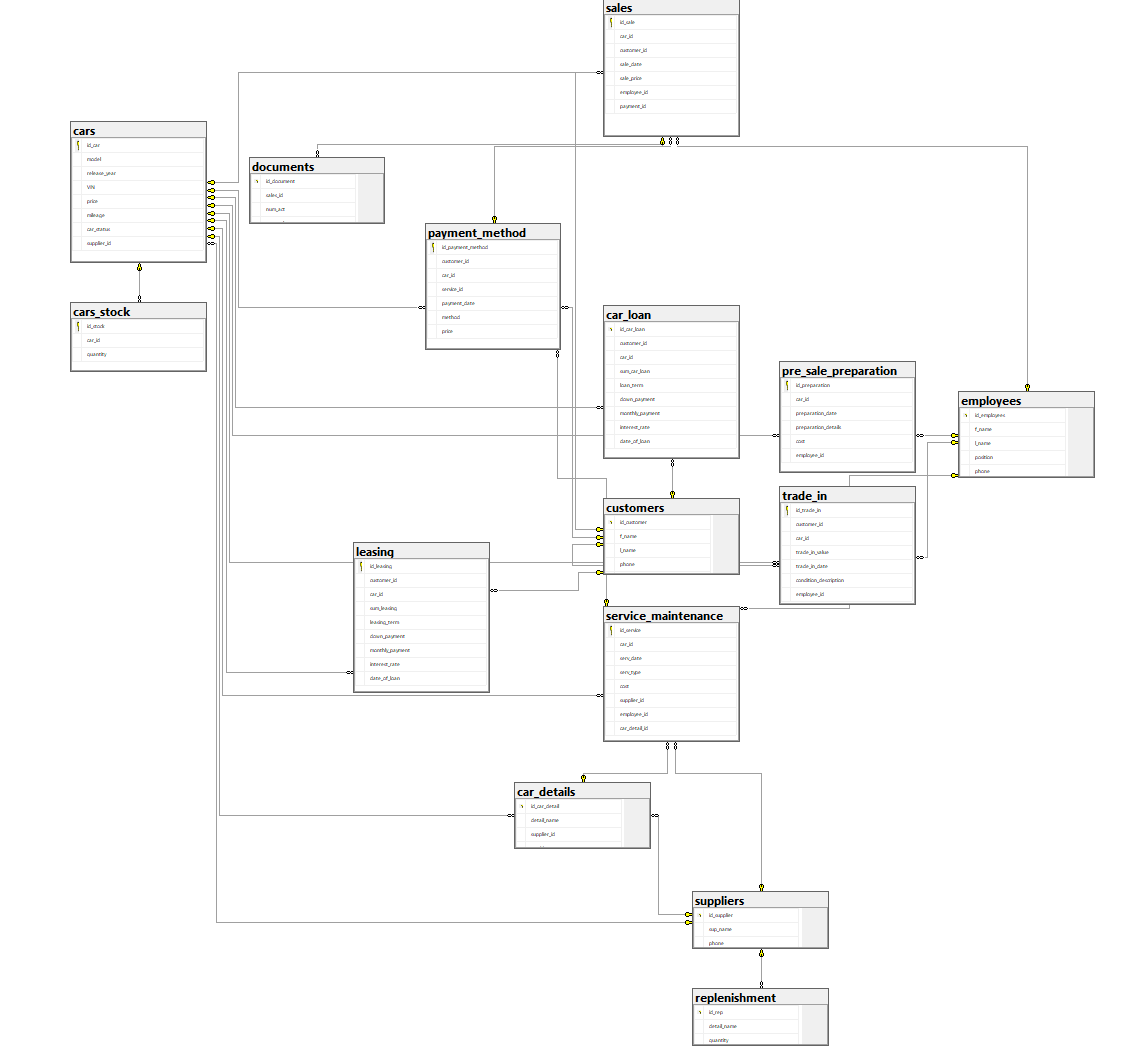


Рисунок 4 – Физическая модель базы данных

# 4.1 Хранимые процедуры

Всего было создано 2 хранимые процедуры. Процедура для получения отчета о продажах. Эта процедура будет генерировать отчет о продажах за определенный период. Ниже представлен код хранимой процедуры.

--Процедура для получения отчета о продажах--

CREATE PROCEDURE GetSalesReport

@StartDate DATETIME,

@EndDate DATETIME

AS

BEGIN

SET NOCOUNT ON;

SELECT sale\_date, sale\_price

FROM sales

WHERE sale\_date BETWEEN @StartDate AND @EndDate;

END;

Процедура для получения информации о клиенте. Эта процедура будет возвращать информацию о клиенте по его идентификатору.

--Процедура для получения информации о клиенте--

CREATE PROCEDURE GetCustomerInfo

@CustomerID INT

AS

BEGIN

SET NOCOUNT ON;

SELECT f\_name, l\_name, phone, email, ph\_address

FROM customers

WHERE id\_customer = @CustomerID;

END;

# 4.2 Триггеры

Триггер на обновление количества автомобилей на складе. Этот триггер будет срабатывать при добавлении записи о продаже автомобиля и автоматически уменьшать количество автомобилей на складе.

--Триггер на обновление количества автомобилей на складе--

CREATE TRIGGER update\_car\_stock\_after\_sale

ON sales

AFTER INSERT

AS

BEGIN

UPDATE cars\_stock

SET quantity = quantity - 1

WHERE id\_stock = (SELECT id\_stock FROM cars WHERE id\_car IN (SELECT id\_car FROM INSERTED));

END;

Триггер на изменение цены автомобиля. Этот триггер будет срабатывать при обновлении информации об автомобиле и автоматически обновлять цену в связанных таблицах.

--Триггер на изменение цены автомобиля--

CREATE TRIGGER update\_price\_in\_loans

ON cars

AFTER UPDATE

AS

BEGIN

UPDATE car\_loan

SET sum\_car\_loan = sum\_car\_loan \* (i.price / d.price)

FROM INSERTED i

JOIN DELETED d ON i.id\_car = d.id\_car;

END;

Триггер для ведения журнала продаж. Этот триггер будет записывать информацию о каждой продаже в отдельную таблицу для аудита.

--Триггер для ведения журнала продаж--

CREATE TRIGGER log\_sales\_audit

ON sales

AFTER INSERT

AS

BEGIN

INSERT INTO sales\_audit (id\_sale, sale\_date, sale\_price)

SELECT id\_sale, sale\_date, sale\_price

FROM INSERTED;

END;

# 4.3 Индексы

Индекс на id\_stock. Этот индекс обеспечит уникальность идентификаторов записей на складе.

--Индекс на id\_stock--

CREATE UNIQUE INDEX idx\_cars\_stock\_id ON cars\_stock (id\_stock);

Индекс на email. Уникальный индекс гарантирует, что адреса электронной почты клиентов не дублируются.

--Индекс на email--

CREATE UNIQUE INDEX idx\_customers\_email ON customers (email);

Индекс на поле внешнего ключа в таблице car\_loan.

--Индекс на поле внешнего ключа в таблице car\_loan--

CREATE INDEX idx\_car\_loan\_customer\_id ON car\_loan (customer\_id);

# 4.4 Ограничения целостности

* Ограничение PRIMARY KEY

Каждая таблица имеет первичный ключ, который уникально идентифицирует каждую запись.

Примеры: в таблице cars первичный ключ — id\_car, в таблице customers — id\_customer, в таблице sales — id\_sale.

Это ограничение гарантирует, что значения в этих столбцах уникальны и не могут быть NULL.

* Ограничение NOT NULL

Некоторые поля имеют ограничение NOT NULL, что означает, что они не могут содержать пустые значения.

Примеры: в таблице cars поля model, release\_year, VIN, price, mileage и car\_status не могут быть NULL; в таблице sales поле sale\_date и sale\_price также не могут быть NULL.

* Ограничение UNIQUE

В таблице cars поле VIN имеет ограничение UNIQUE, что гарантирует уникальность идентификационного номера автомобиля. Это предотвращает возможность дублирования VIN для разных автомобилей.

* Ограничение FOREIGN KEY

Многие таблицы имеют внешние ключи, которые обеспечивают ссылочную целостность между ними.

Примеры: в таблице sales поля car\_id, customer\_id, и employee\_id ссылаются на соответствующие идентификаторы в таблицах cars, customers, и employees; в таблице cars\_stock поле car\_id ссылается на идентификатор автомобиля в таблице cars.

Эти ограничения гарантируют, что значения внешних ключей существуют в родительских таблицах, предотвращая создание «висячих» ссылок.

* Ограничение CHECK

Некоторые поля имеют ограничение CHECK, которое накладывает дополнительные условия на допустимые значения.

Примеры: в таблице cars поле [car\_status] должно принимать только одно из трех значений: 'available', 'sold', 'in\_repair'; в таблице payment\_method поле [method] может принимать только значения: 'cash', 'card', 'car loan', 'leasing'.

Это ограничение помогает предотвратить ввод некорректных данных.

* Ограничение DEFAULT

Некоторые поля имеют значение по умолчанию, которое будет использоваться при отсутствии явного значения.

Примеры: в таблице cars поле [car\_status] по умолчанию устанавливается в 'available'; в таблице payment\_method поле [method] по умолчанию устанавливается в 'card'.

# ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ

# 5.1 Граф переходов между экранными формами

На рисунке 5 представлен граф перехода между формами, он описывает логику переходов.

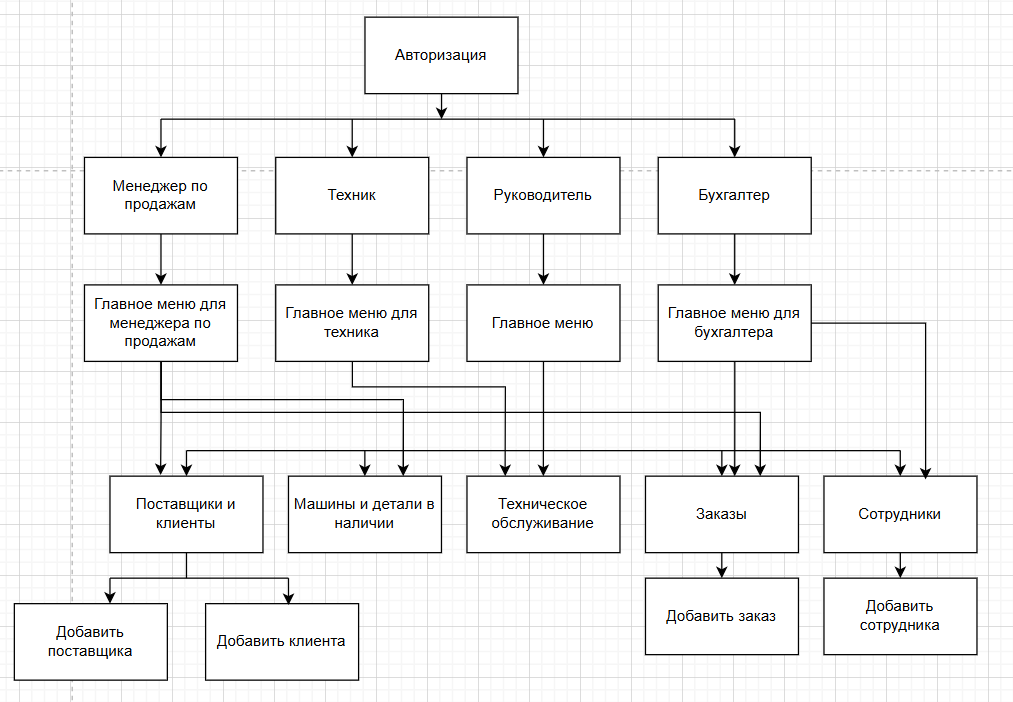


Рисунок 5 – Граф переходов между формами

# 5.2 Описание форм

На рисунке 6 представлена форма авторизации. Авторизация происходит по почте и паролю сотрудника.

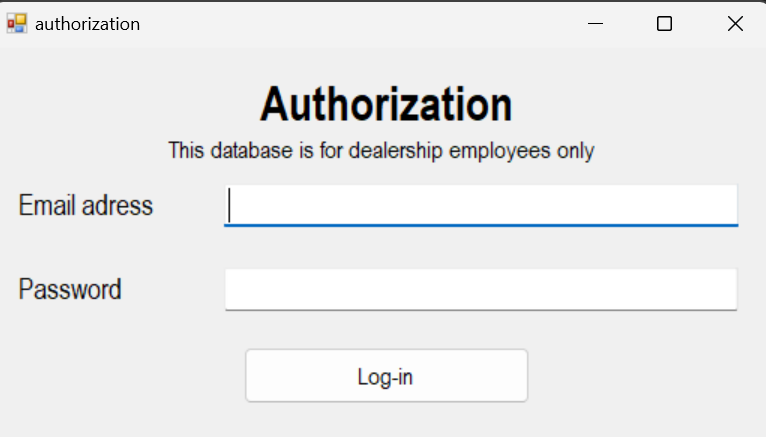


Рисунок 6 – Авторизация

На рисунке 7 представлена заполненная форма авторизации. При вводе некорректных данных, выводится сообщение, как на рисунке 8.

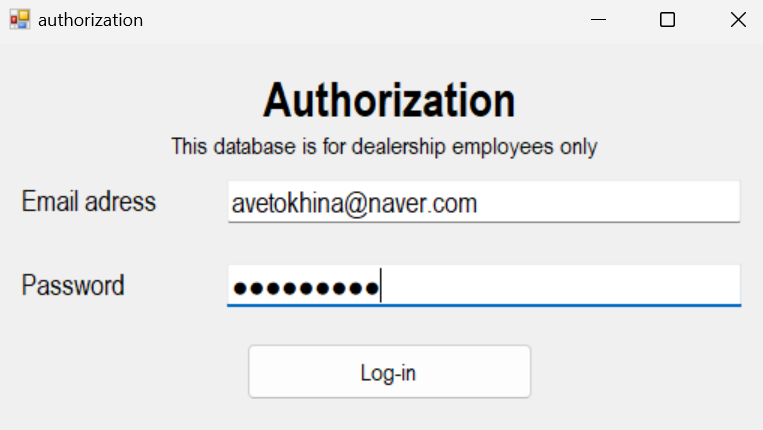


Рисунок 7 – Ввод данных

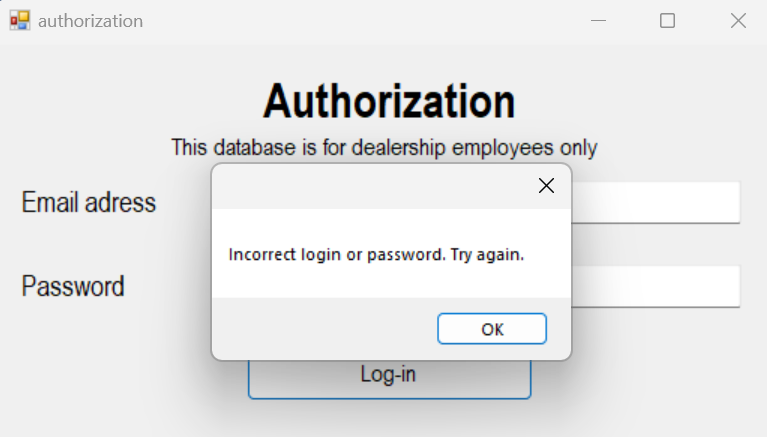


Рисунок 8 – Сообщение о неправильно введенных данных

Был произведен вход под именем руководителя, он обладает всем функционалом системы. Форма главного меню для руководителя представлена на рисунке 9.

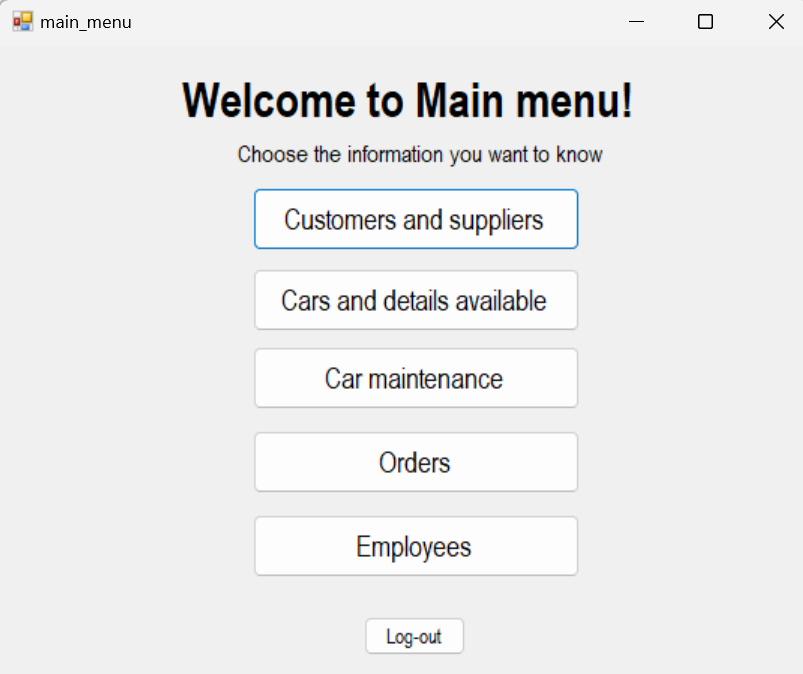


Рисунок 9 – Форма главного меню для руководителя

При попытке выхода из системы, выведется сообщение как на рисунке 10.

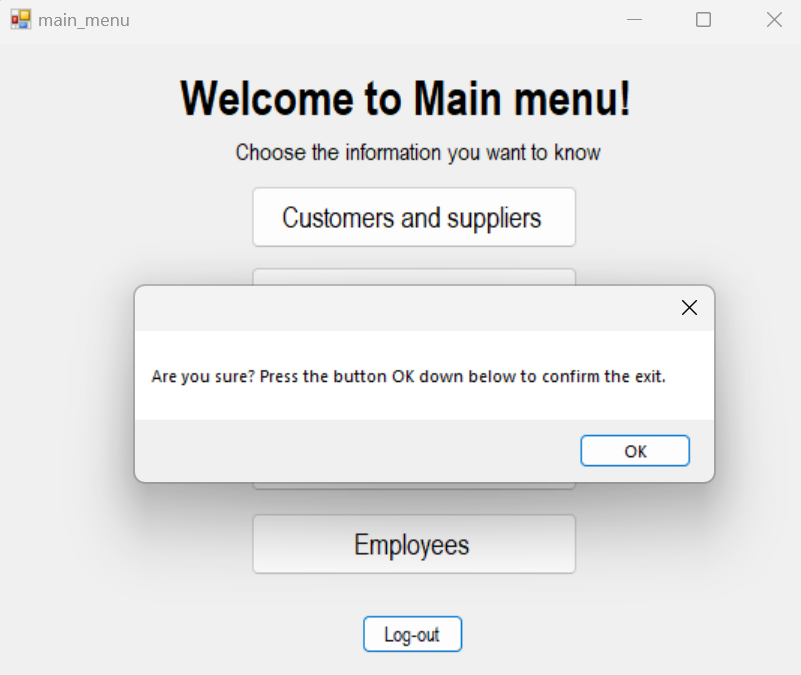


Рисунок 10 – Попытка выхода

Далее, на рисунке 11 представлена форма с поставщиками и клиентами. Можно добавить, обновить информацию о клиентах или поставщиках, найти по фамилии клиента или имени поставщика конкретную запись в базе данных, и удалить клиента или поставщика. При нажатии кнопки «Search» выведутся все данные, как на рисунке 12. Если в поле ввода ввести фамилию клиента, выведется запись только об указанном клиенте, как показано на рисунке 13.

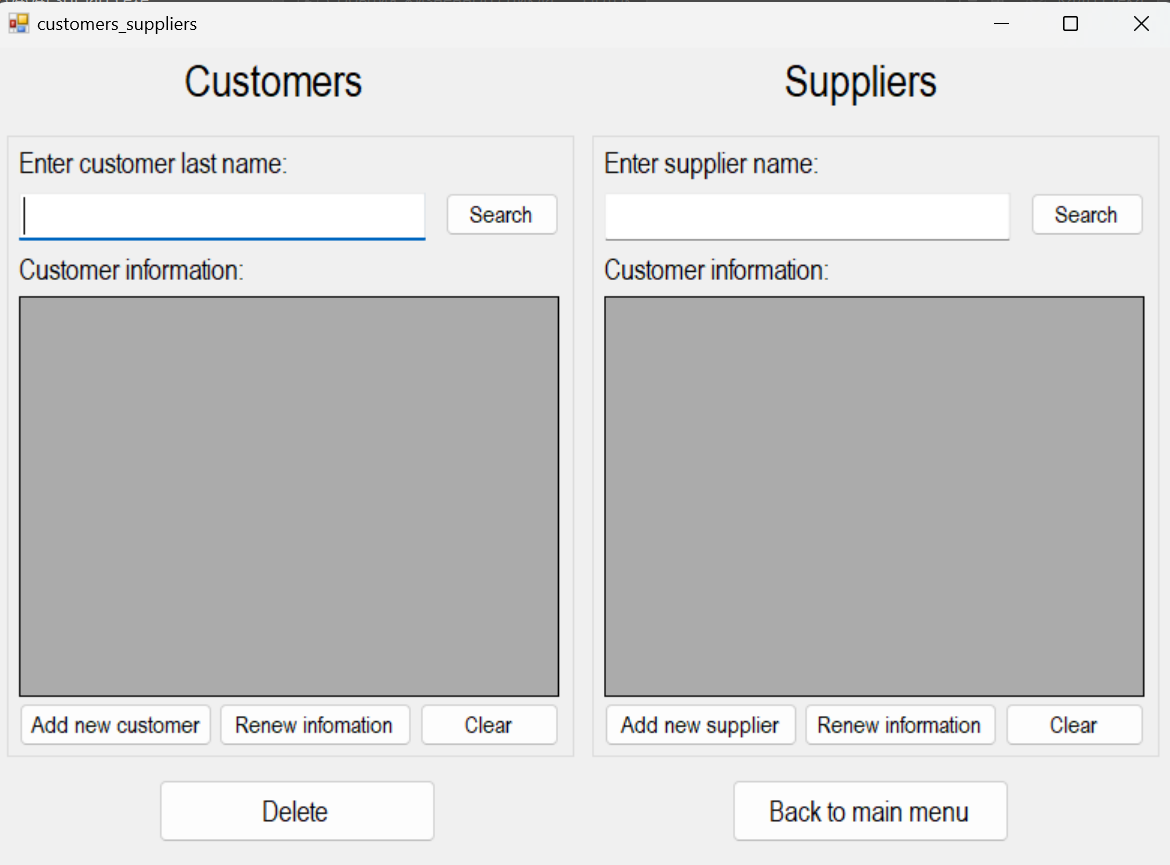


Рисунок 11 – Форма с клиентами и поставщиками

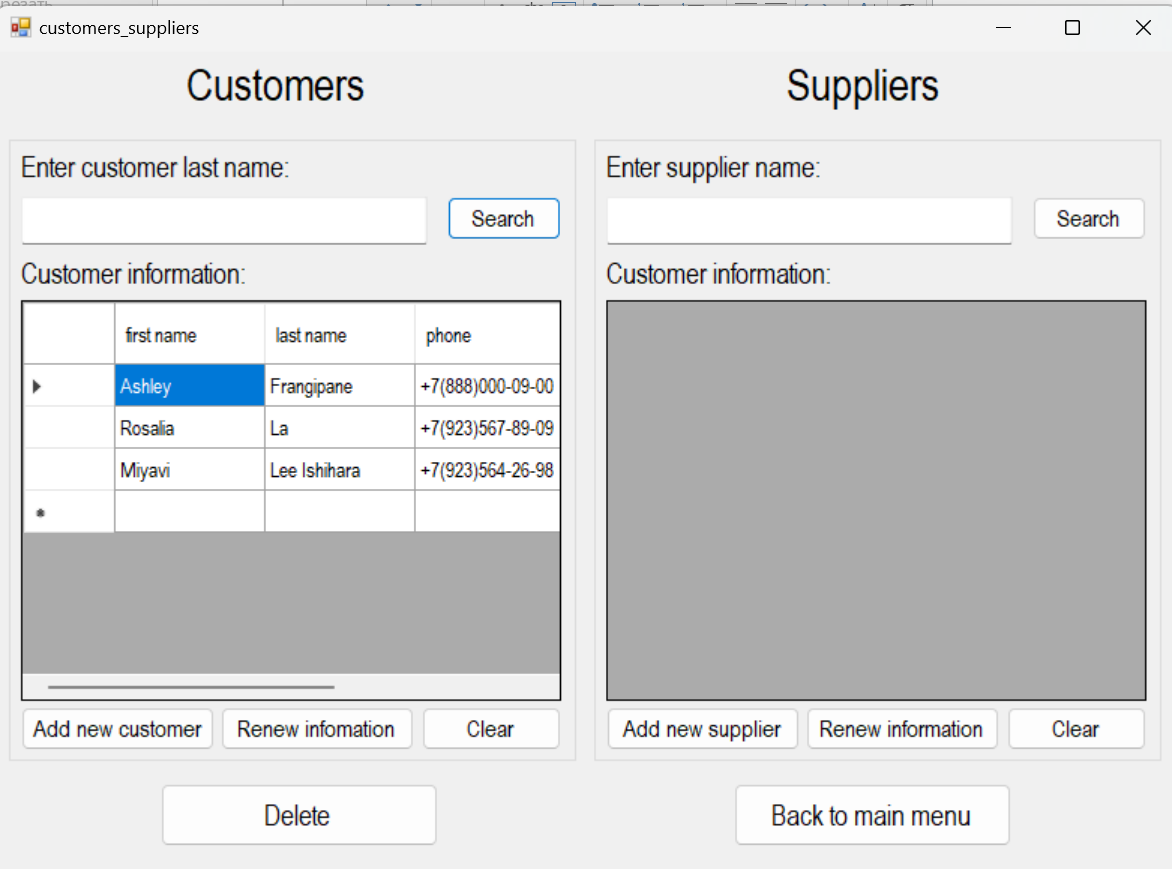


Рисунок 12 – Поиск всех клиентов в базе данных

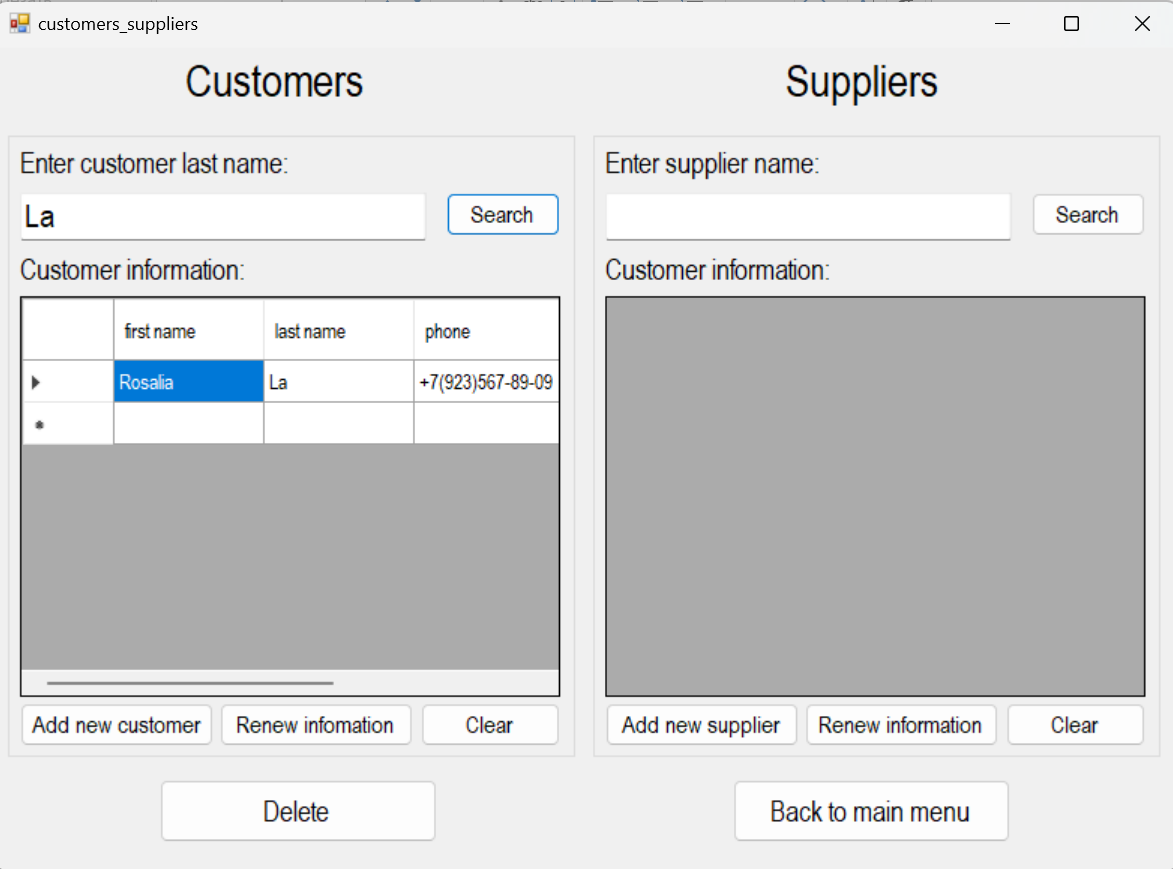


Рисунок 13 – Вывод конкретного клиента

На рисунке 14 представлена еще одна форма – форма добавления нового клиента или обновления данных о существующем клиенте.

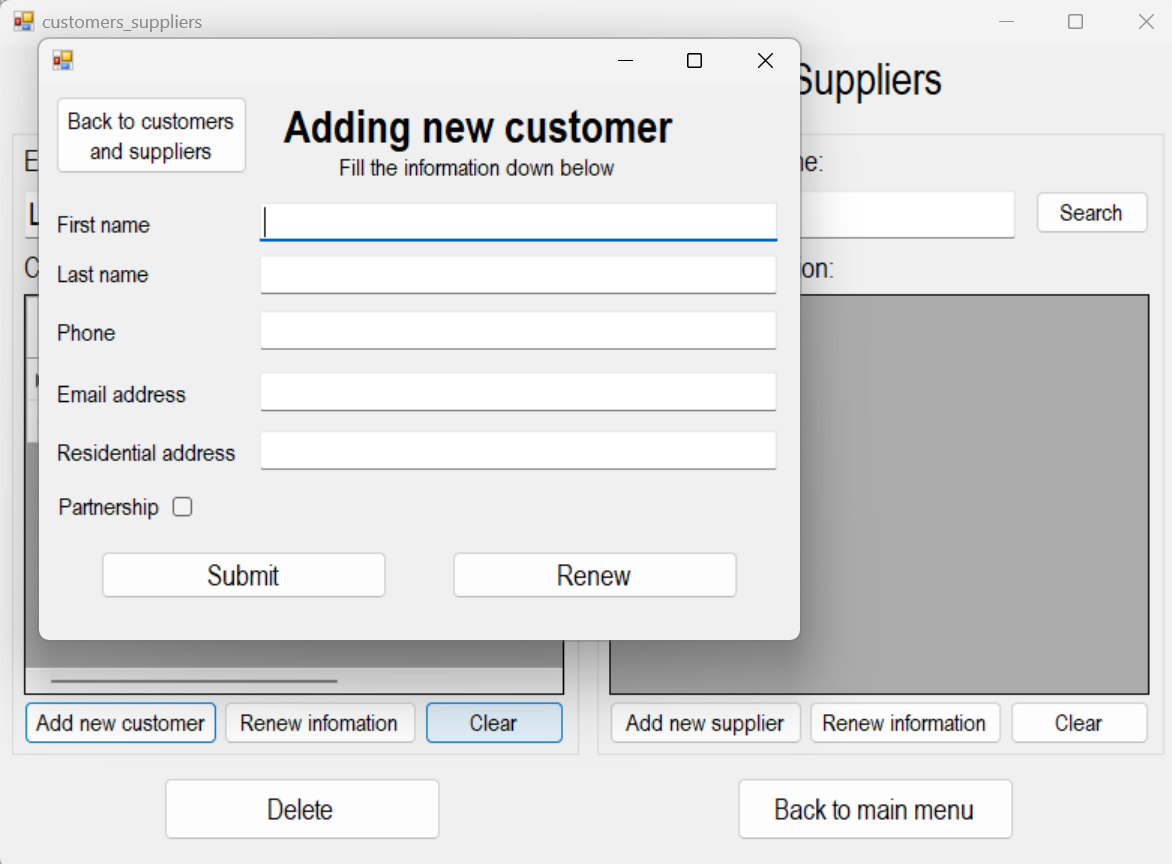


Рисунок 14 – Добавление нового клиента

Для поставщиков реализован такой же функционал, как и у клиентов дилерского центра. Дальше была реализована форма для учета машин и деталей, пример показан на рисунке 15. Машины ищутся по доступности, всего есть три статуса: в наличии, продано и в ремонте, как представлено на рисунке 16. Если ничего не выбирать и сразу нажать кнопку поиска, то выведутся все записи о машинах вне зависимости от их статуса. На рисунке 17 представлены все записи о существовавших когда-либо машинах в базе данных. Но при этом, если все-таки выбрать какой-либо статус, то покажут все записи о машинах, которые находятся в этом статусе. Так, на рисунке 18 продемонстрированы все машины в статусе «в наличии».

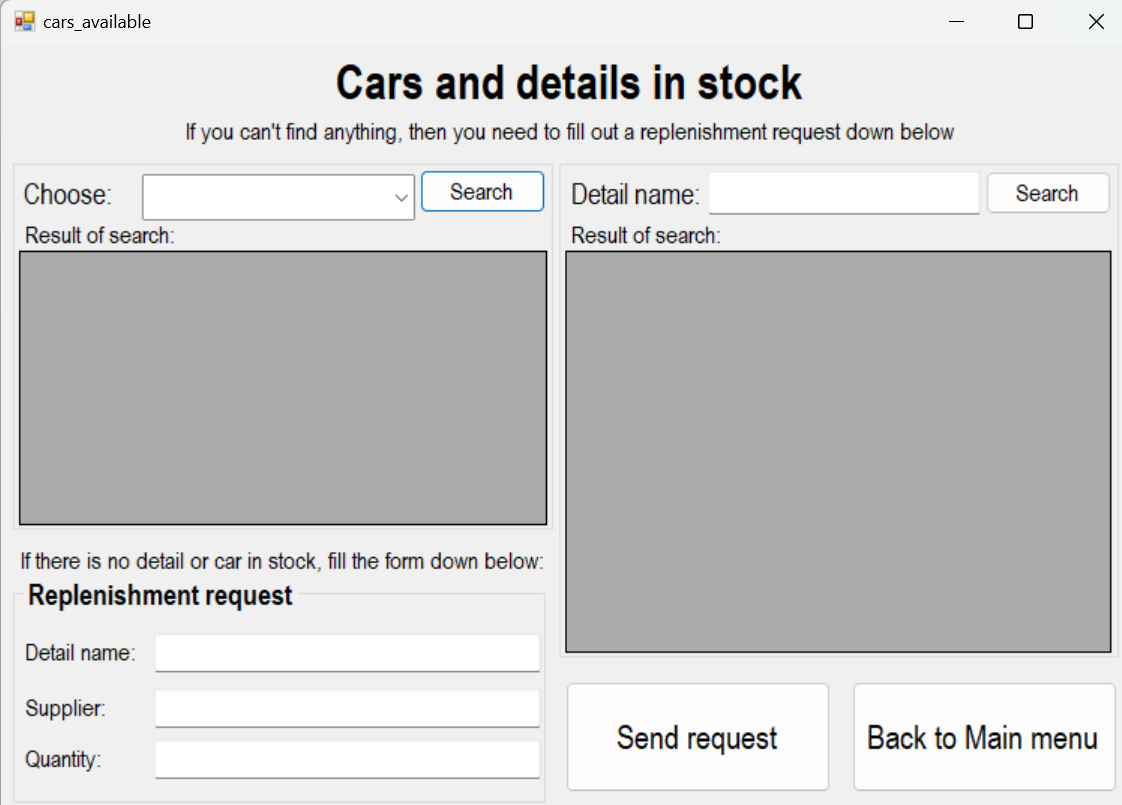


Рисунок 15 – Форма машин и деталей в наличии

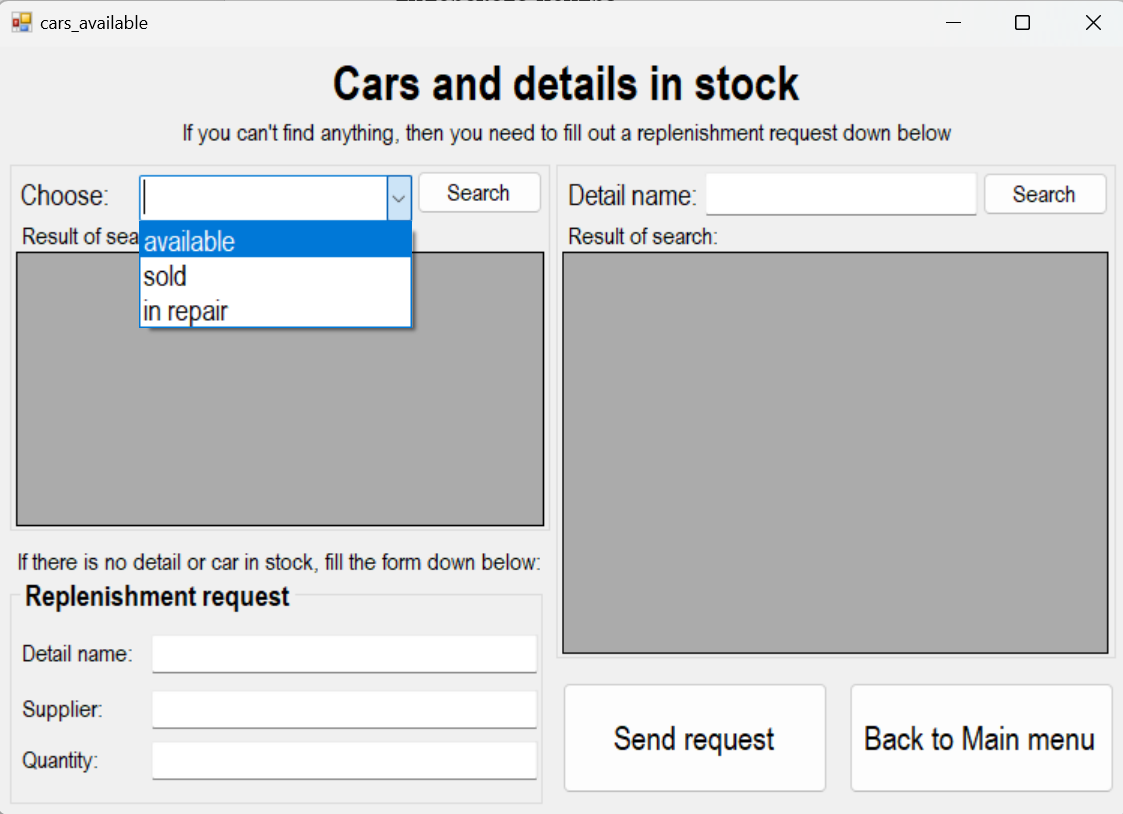


Рисунок 16 – Выбор статуса машины

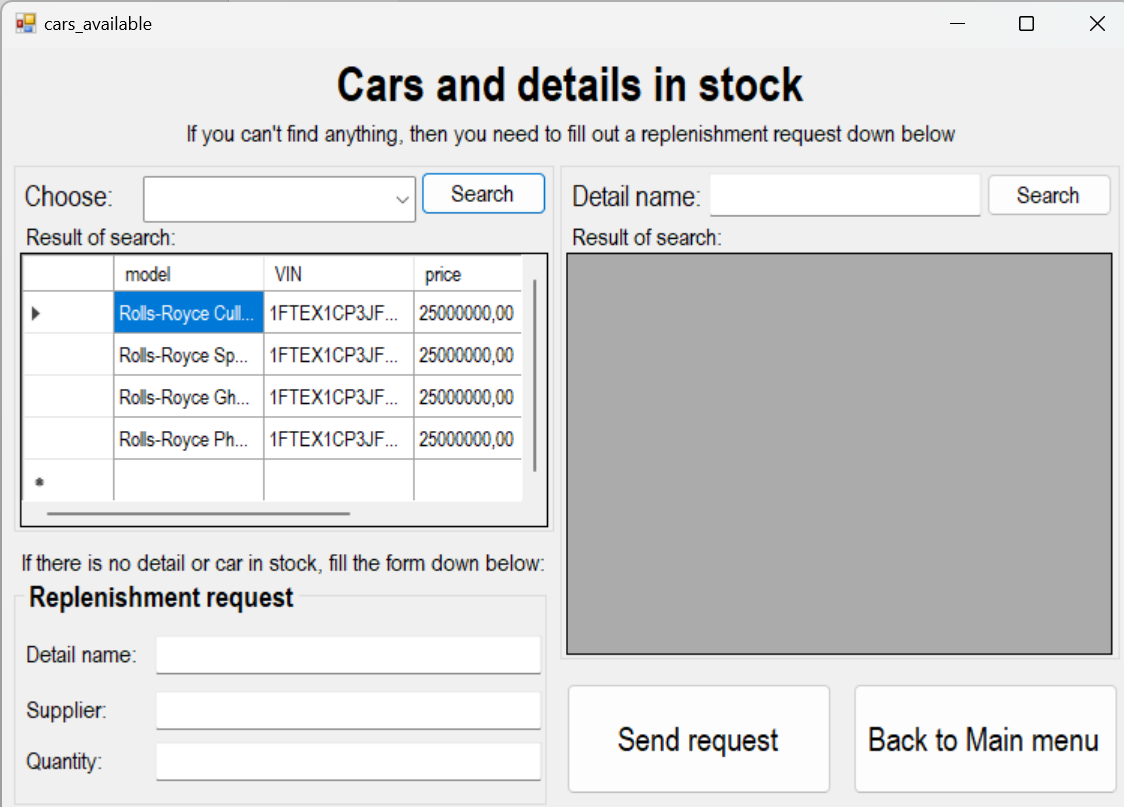


Рисунок 17 – Все записи о машинах

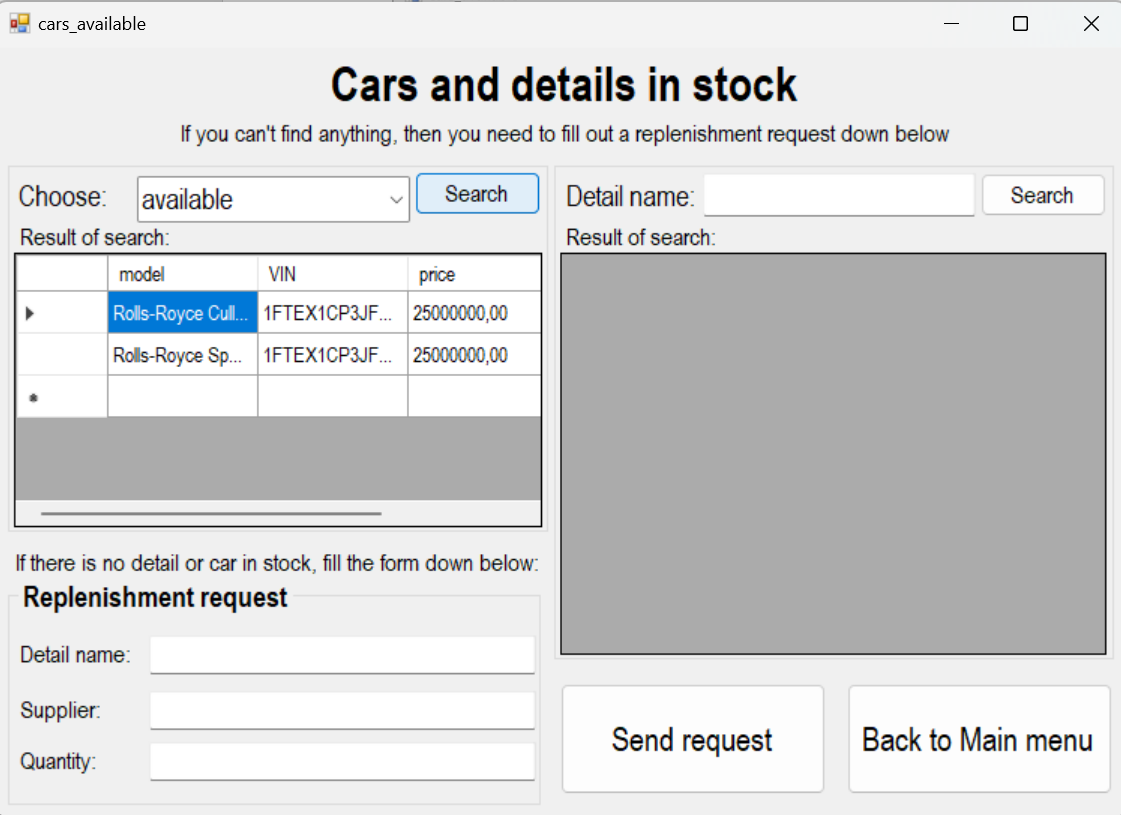


Рисунок 18 – Доступные машины

Поиск деталей осуществляется по названию детали, если ничего не ввести – то выведется список всех деталей в наличии. Считается, что если запись есть – то деталь в наличии. На рисунке 19 показаны все детали в наличии в дилерском центре, на рисунке 20 – конкретная деталь.

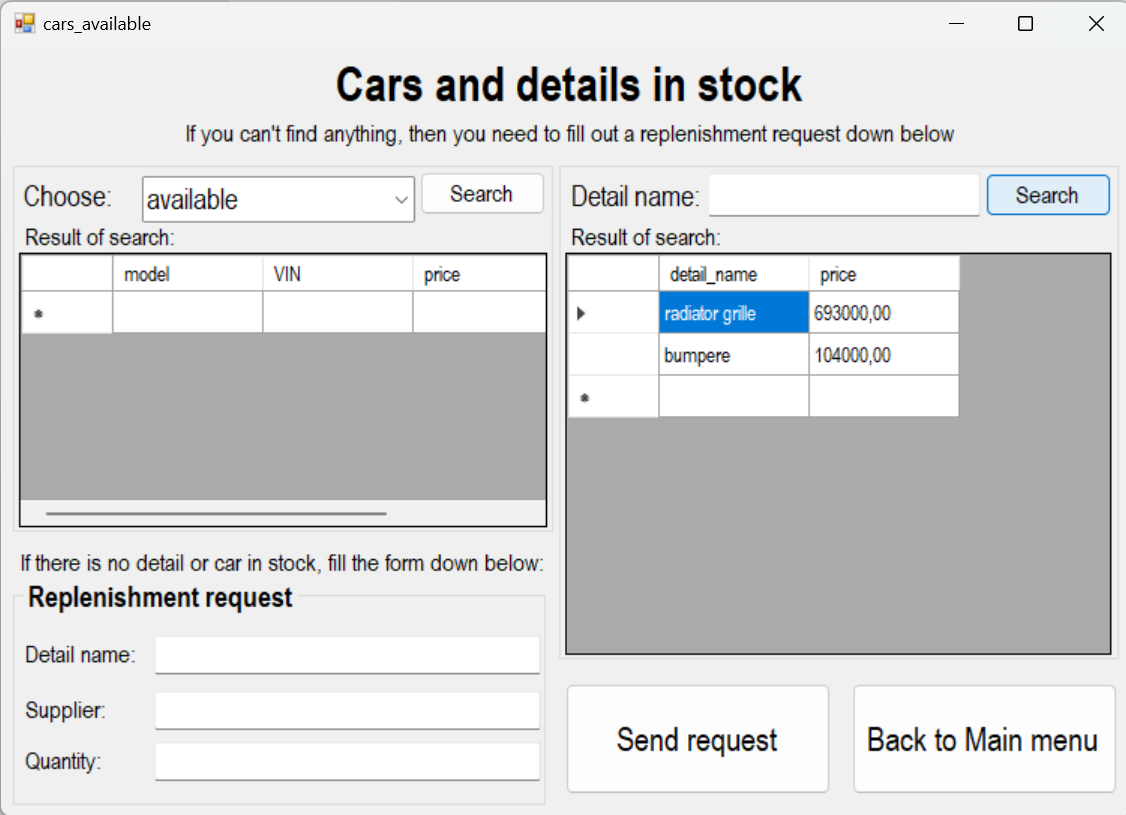


Рисунок 19 – Детали в наличии

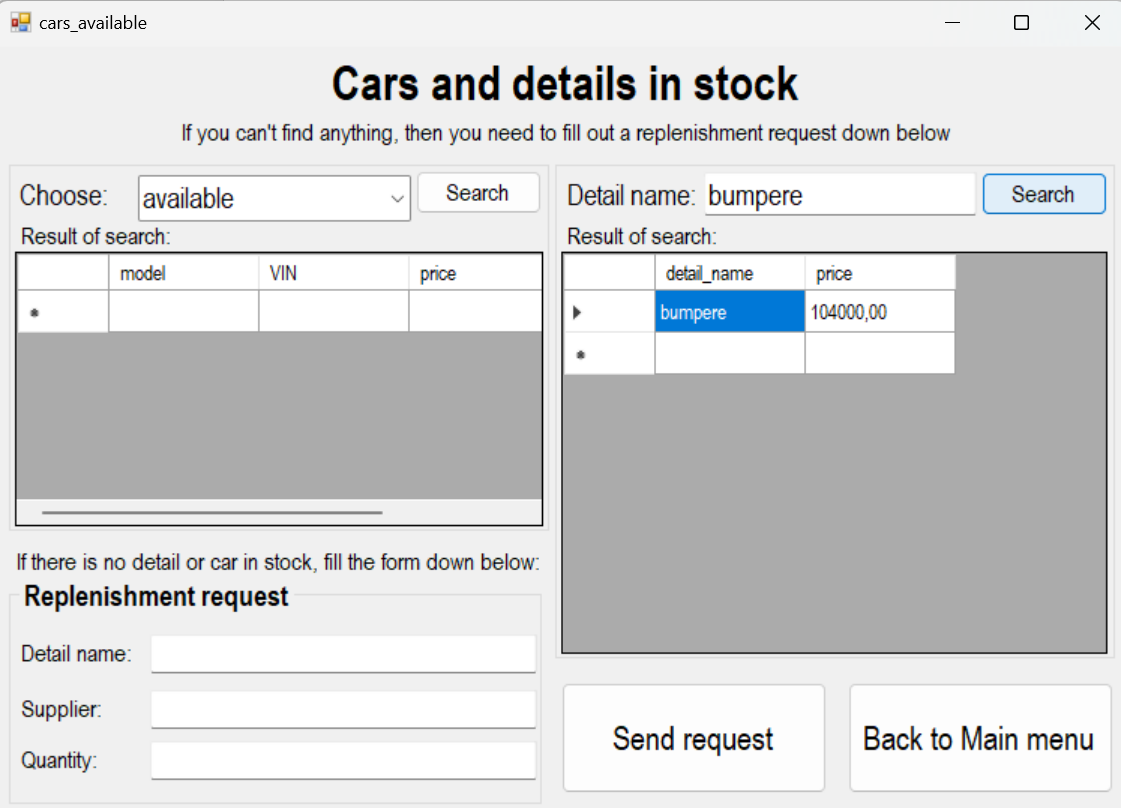


Рисунок 20 – Конкретная деталь в наличии

Если нужная деталь не найдена, в этой же форме можно оставить заявку на закупку детали. Для этого нужно указать название детали, поставщика и количество закупаемых деталей.

На рисунке 21 представлена форма для техобслуживания автомобилей. Слева в окне сотрудник дилерского центра может записать машину на какое-либо техобслуживание, а справа представлена вся информация про существующие записи на техобслуживание. Поиск реализован по названию сервиса или по модели машины, как показано на рисунке 22.

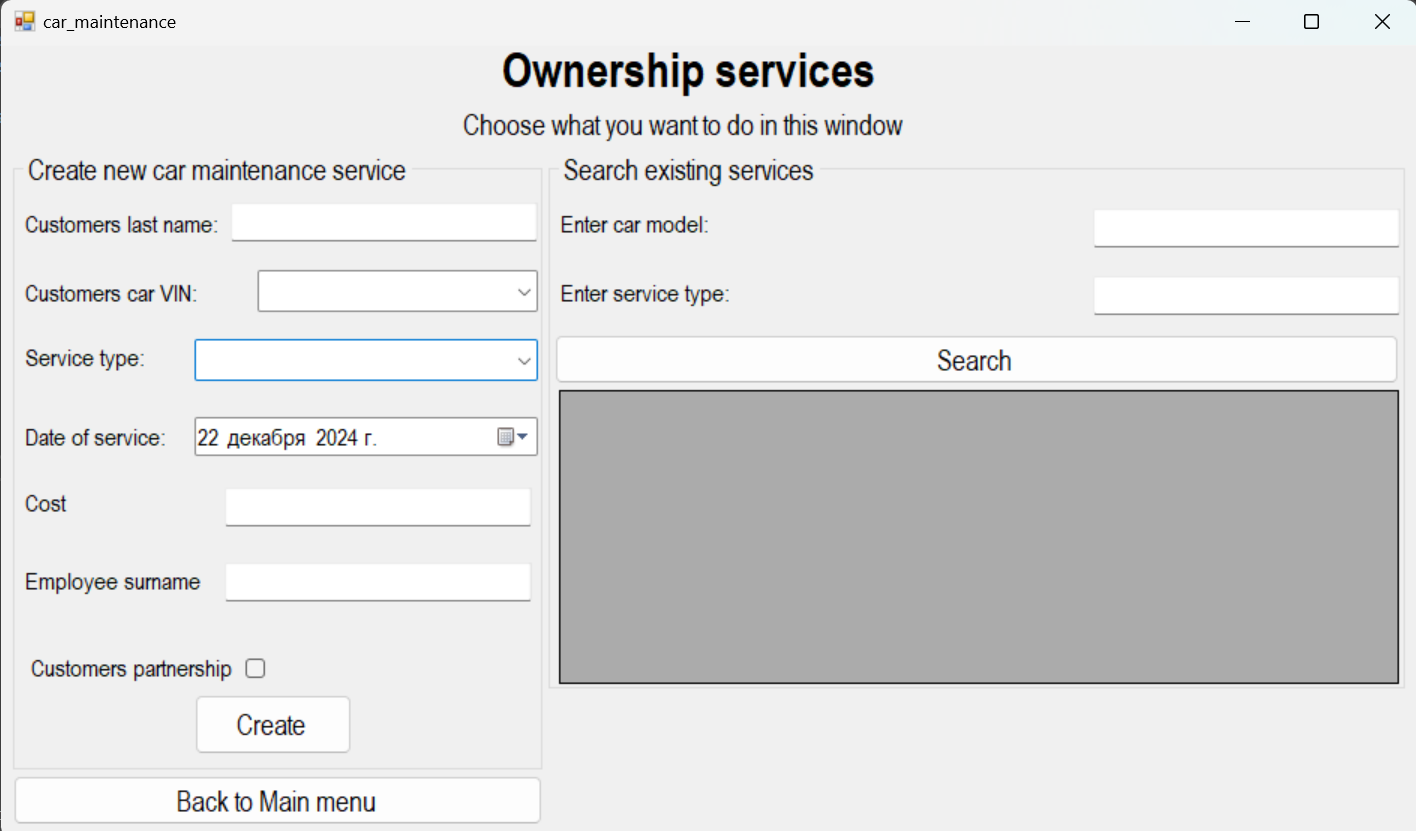


Рисунок 21 – Форма техобслуживания автомобиля

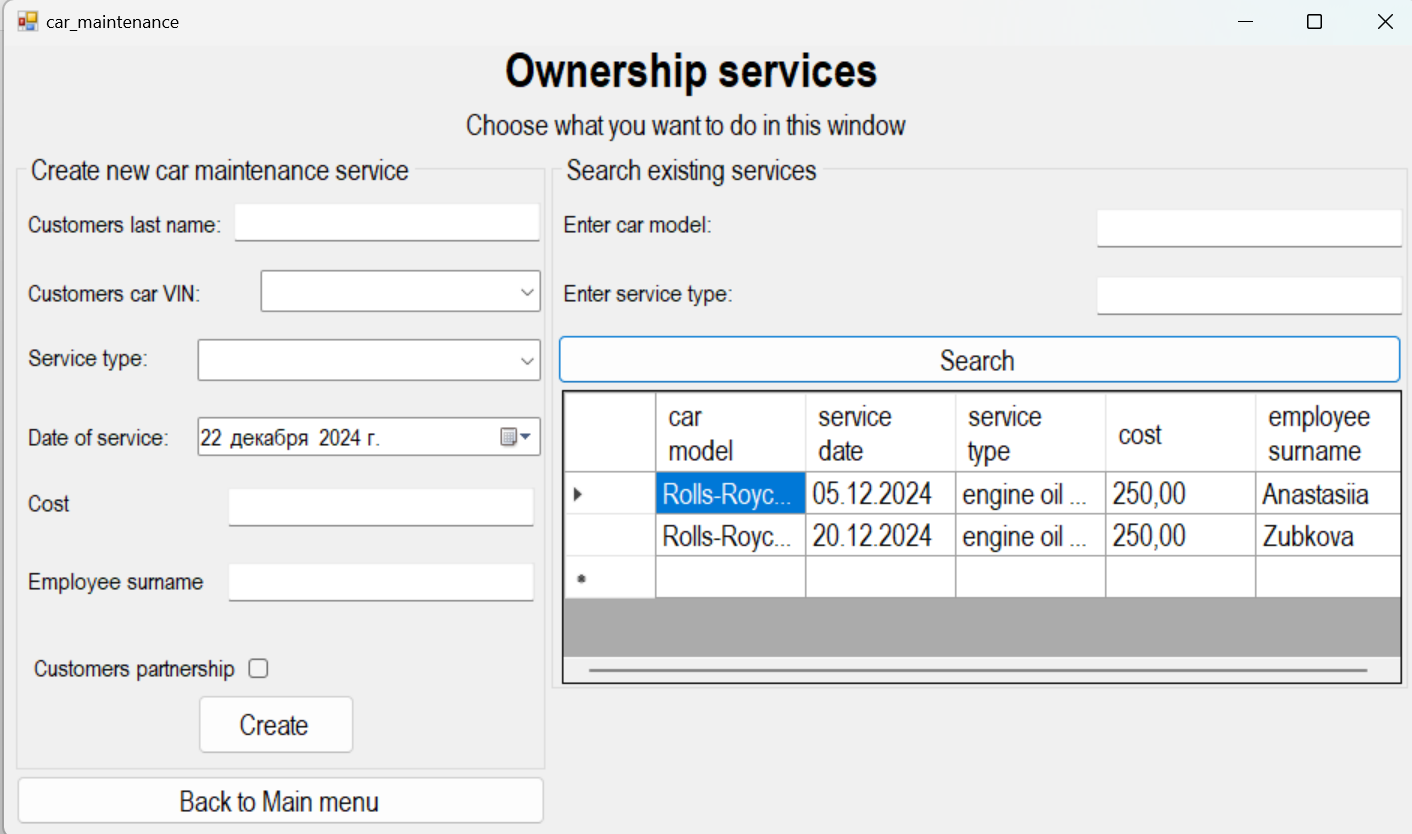


Рисунок 22 – Все записи о техобслуживании

На рисунке 23 представлена форма с заказами машин. Поиск осуществляется по фамилии покупателя. В таблице отображается модель машины, фамилия клиента, дата продажи, цена, фамилия сотрудника, который продал машину, статус платежа и метод оплаты. На рисунке 24 представлен пример заказа. Так же в этом окне можно создать новый заказ, нажав на соответствующую кнопку. После нажатия появляется форма заказа, как на рисунке 25. Если хотя бы одно поле не заполнено, выводится сообщение как на рисунке 26.

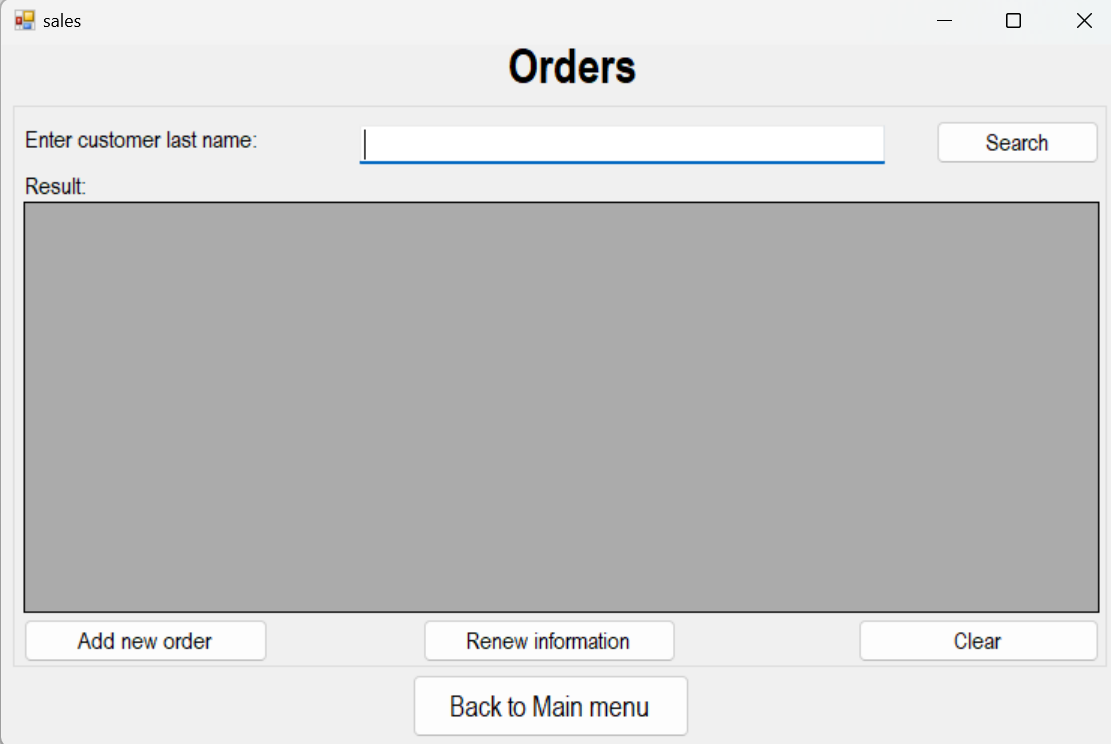


Рисунок 23 – Форма заказов

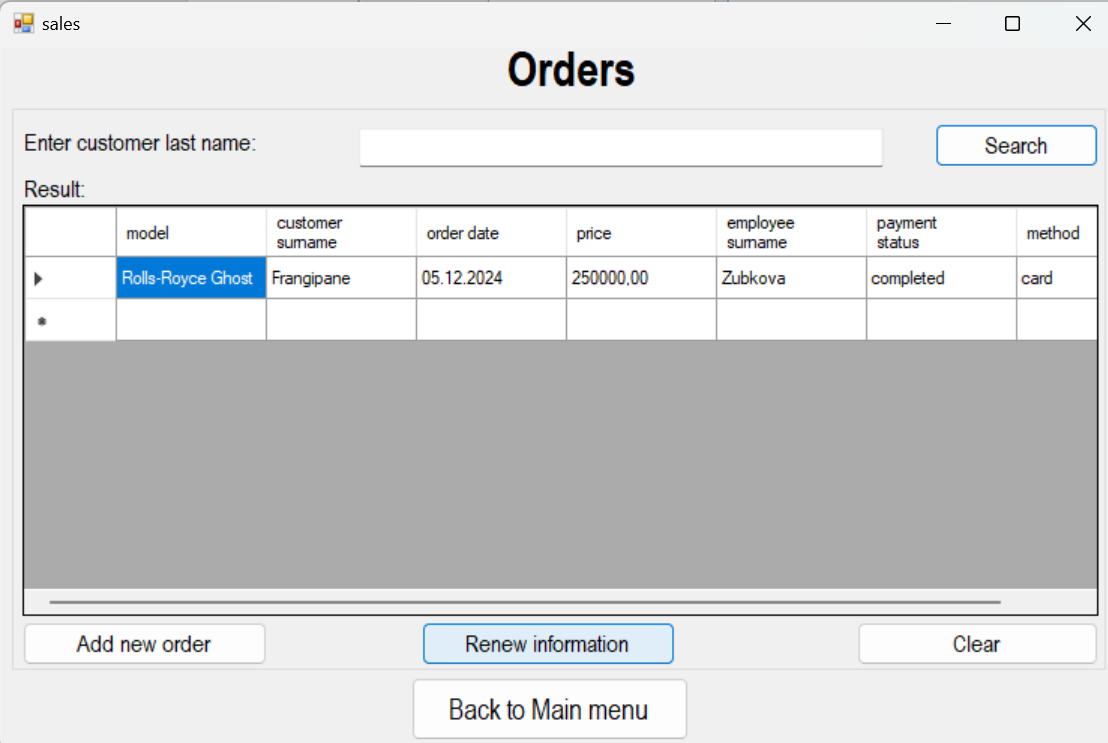


Рисунок 24 – Все заказы

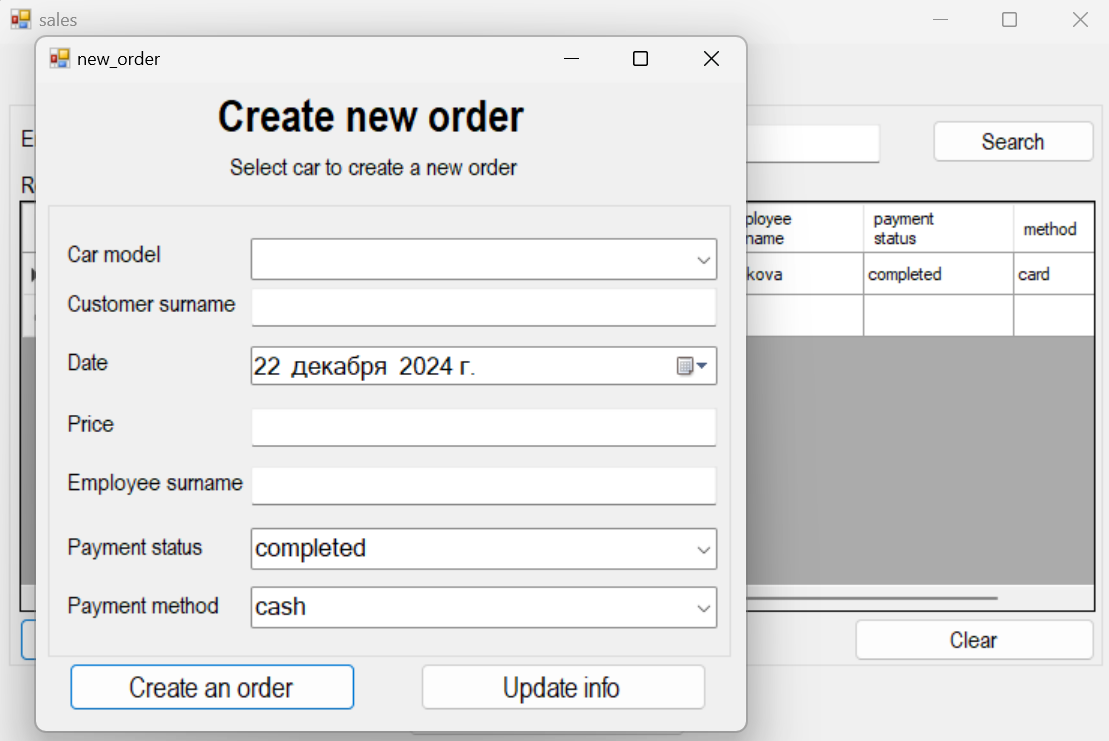


Рисунок 25 – Создание нового заказа

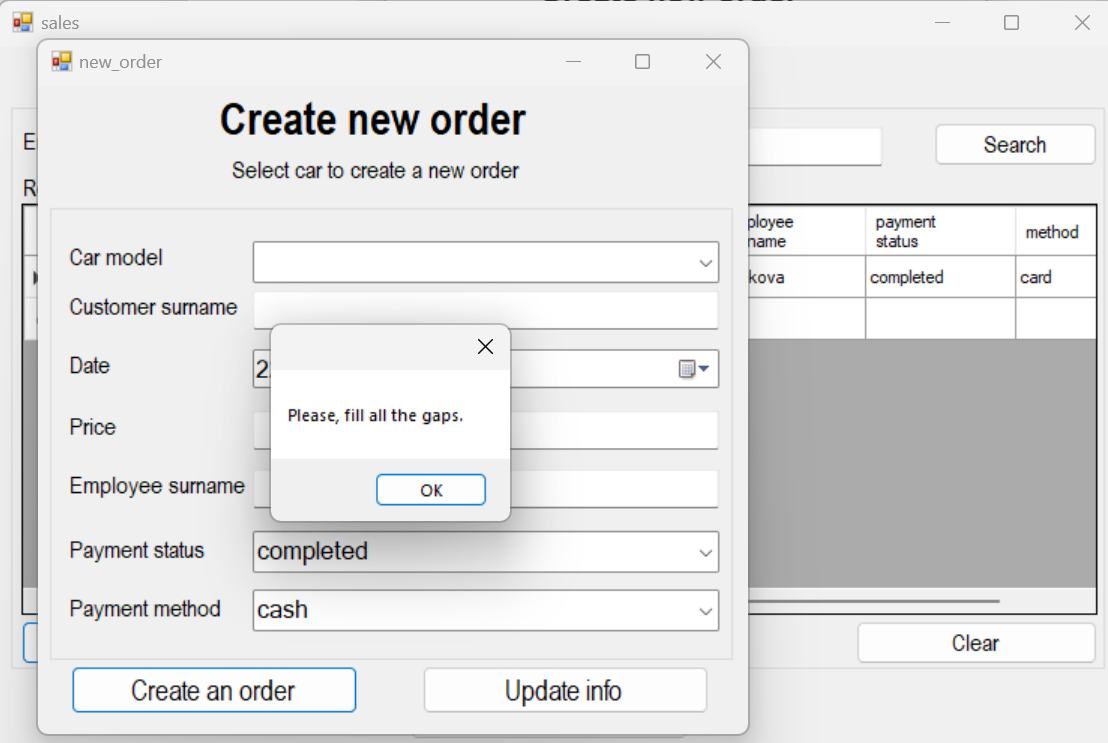


Рисунок 26 – Сообщение о необходимости заполнить все поля

На рисунке 27 представлена форма со всеми сотрудниками дилерского центра. На рисунке 28 представлены все записи о сотрудниках, в этой форме также можно добавить нового сотрудника или удалить запись о нем. На рисунке 29 представлена форма добавления сотрудника.

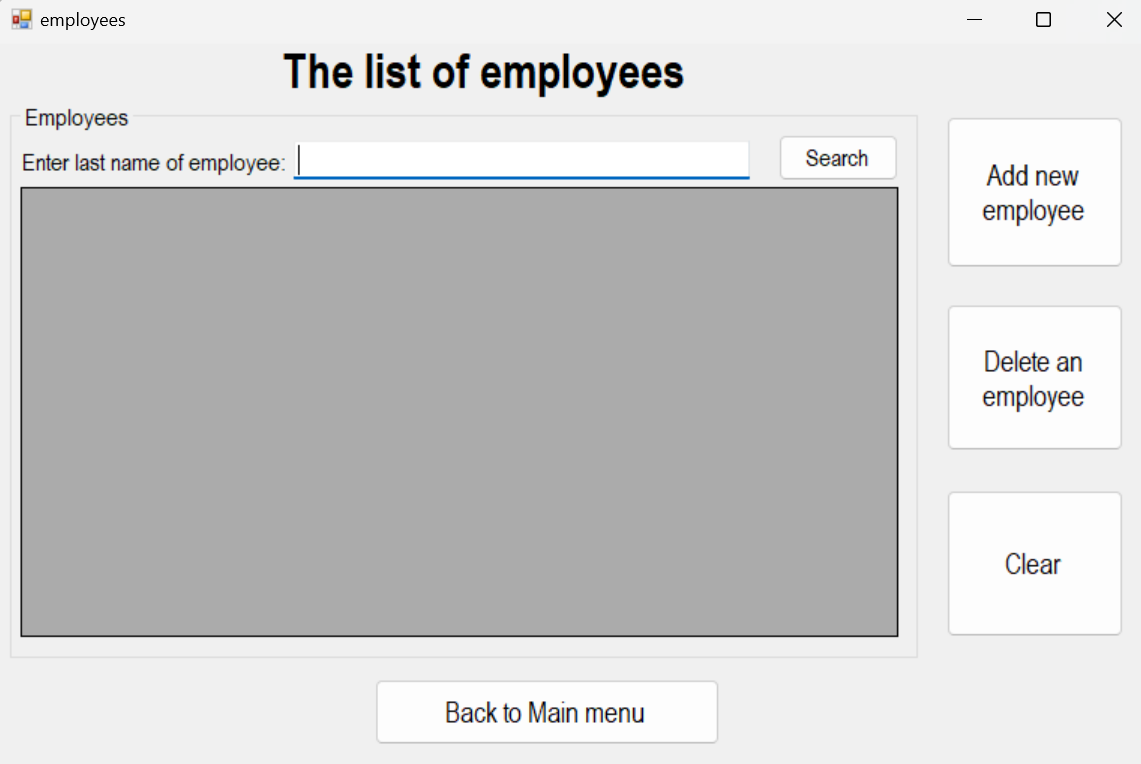


Рисунок 27 – Форма с сотрудниками

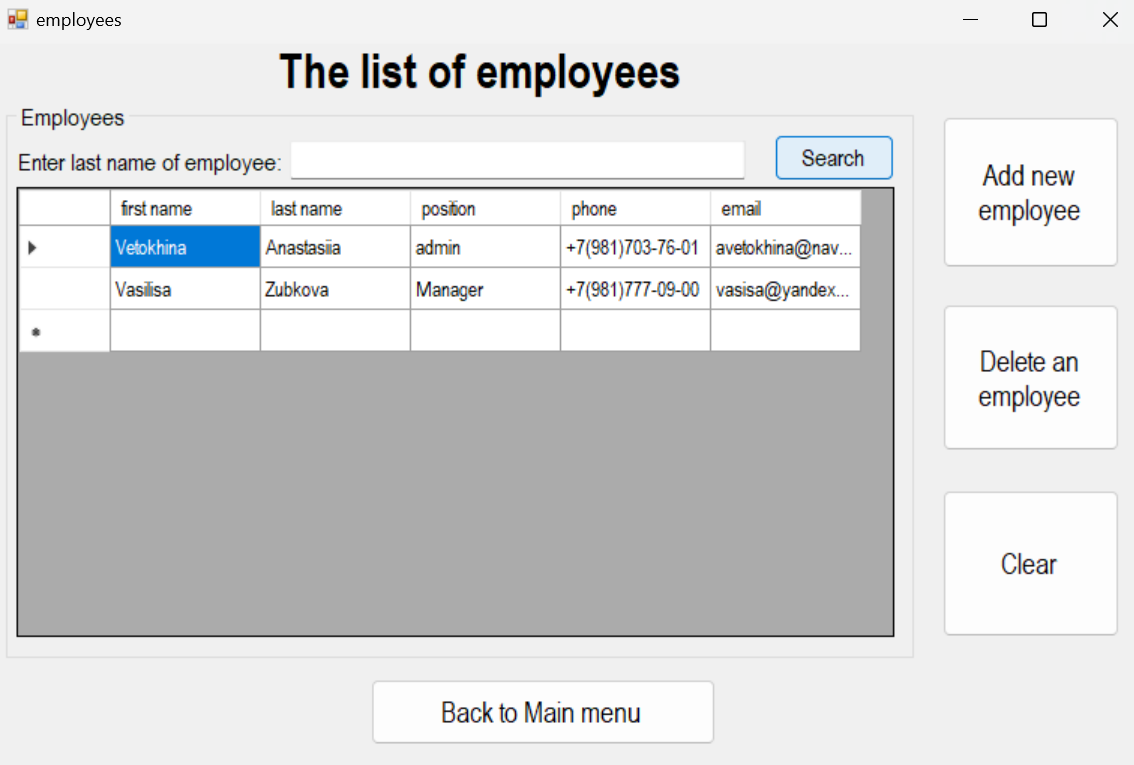


Рисунок 28 – Все данные о сотрудниках

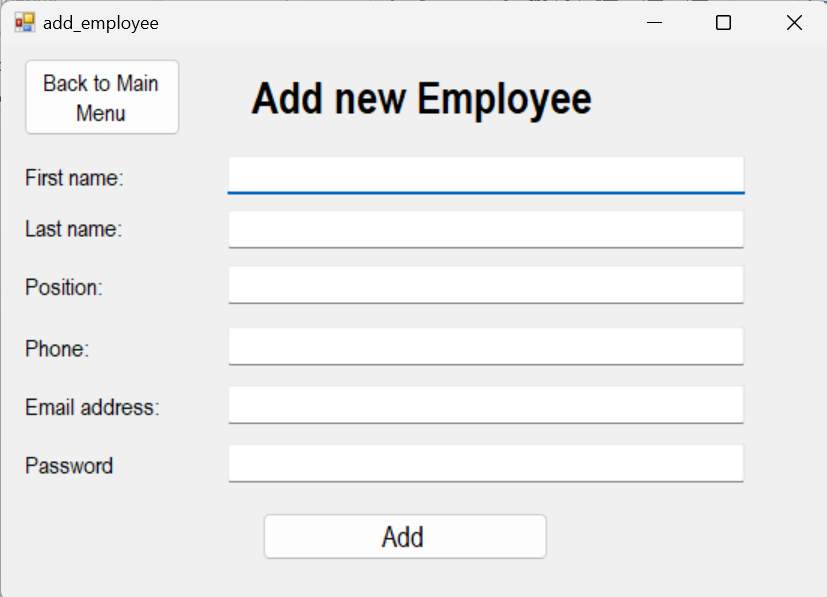


Рисунок 29 – Добавление нового сотрудника в базу данных

На рисунке 30 представлен пример главного меню для менеджера по продажам.

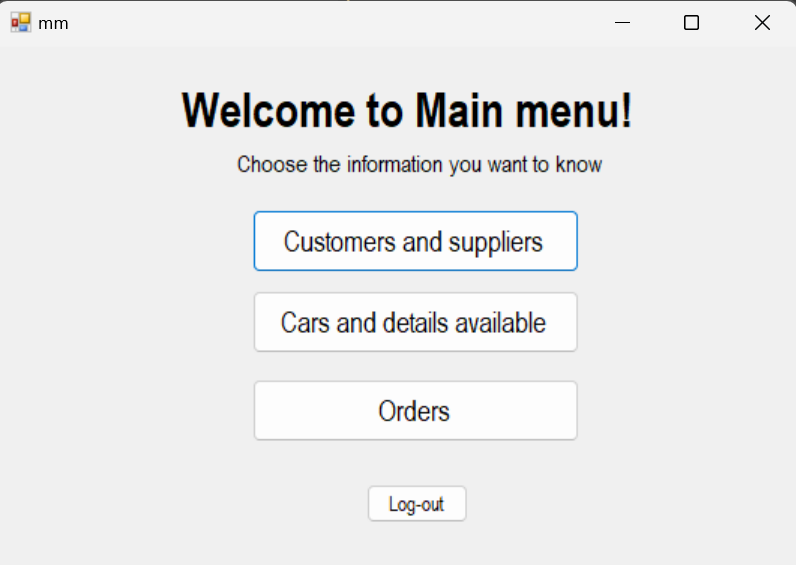


Рисунок 30 – Форма главного меню для менеджера по продажам

# ПРИЛОЖЕНИЕ А. Создание объектов базы данных

Создание таблиц базы данных, файл cardealer\_tables.sql

use cardealer

go

-- Таблица автомобилей

CREATE TABLE cars (

id\_car INT PRIMARY KEY,

model VARCHAR(50) NOT NULL,

release\_year INT NOT NULL,

VIN VARCHAR(17) NOT NULL UNIQUE,

price DECIMAL(10, 2) NOT NULL,

mileage INT NOT NULL,

[car\_status] varchar(20) NOT NULL check ([car\_status] in ('available', 'sold', 'in\_repair')) default 'available',

supplier\_id int,

foreign key (supplier\_id) references suppliers(id\_supplier)

);

-- Таблица клиентов

CREATE TABLE customers (

id\_customer INT PRIMARY KEY,

f\_name VARCHAR(50) NOT NULL,

l\_name VARCHAR(50) NOT NULL,

phone VARCHAR(16),

email VARCHAR(100),

ph\_address VARCHAR(255)

);

select \* from employees;

-- Таблица сотрудников

CREATE TABLE employees (

id\_employees INT PRIMARY KEY,

f\_name VARCHAR(50) NOT NULL,

l\_name VARCHAR(50) NOT NULL,

position VARCHAR(50),

phone VARCHAR(16),

email VARCHAR(100),

pass\_word varchar(50)

);

-- Таблица продаж

CREATE TABLE sales (

id\_sale INT PRIMARY KEY,

car\_id INT,

customer\_id INT,

payment\_id int,

sale\_date DATE NOT NULL,

sale\_price DECIMAL(10, 2) NOT NULL,

employee\_id int,

FOREIGN KEY (car\_id) REFERENCES cars(id\_car),

FOREIGN KEY (customer\_id) REFERENCES customers(id\_customer),

foreign key (employee\_id) references employees(id\_employees),

foreign key (payment\_id) references payment\_method(id\_payment\_method)

);

-- Таблица поставщиков

CREATE TABLE suppliers (

id\_supplier INT PRIMARY KEY,

sup\_name VARCHAR(100) NOT NULL,

phone VARCHAR(16),

email VARCHAR(100)

);

-- Таблица запасов

CREATE TABLE cars\_stock (

id\_stock INT PRIMARY KEY,

car\_id INT,

quantity INT NOT NULL,

FOREIGN KEY (car\_id) REFERENCES cars(id\_car)

);

-- Таблица сервисного обслуживания

CREATE TABLE service\_maintenance (

id\_service INT PRIMARY KEY,

car\_id INT,

serv\_date DATE NOT NULL,

serv\_type varchar(100),

cost DECIMAL(10, 2),

supplier\_id int,

employee\_id int,

car\_detail\_id int,

FOREIGN KEY (car\_id) REFERENCES cars(id\_car),

foreign key (supplier\_id) references suppliers(id\_supplier),

foreign key (employee\_id) references employees(id\_employees),

foreign key (car\_detail\_id) references car\_details(id\_car\_detail)

);

--комплектующие детали--

create table car\_details (

id\_car\_detail int primary key,

detail\_name varchar(20),

supplier\_id int,

car\_id int,

foreign key (supplier\_id) references suppliers(id\_supplier),

foreign key (car\_id) references cars(id\_car)

);

create table payment\_method (

id\_payment\_method int primary key,

customer\_id int,

car\_id int,

service\_id int,

payment\_date date,

[method] varchar(10) not null check ([method] in ('cash','card','car loan','leasing')) default 'card',

price DECIMAL(10, 2) NOT NULL,

foreign key (customer\_id) references customers(id\_customer),

foreign key (car\_id) references cars(id\_car),

foreign key (service\_id) references service\_maintenance(id\_service)

);

create table car\_loan (

id\_car\_loan int primary key,

customer\_id int,

car\_id int,

sum\_car\_loan DECIMAL(10, 2) NOT NULL,

loan\_term int,

down\_payment DECIMAL(10, 2) NOT NULL,

monthly\_payment DECIMAL(10, 2) NOT NULL,

interest\_rate DECIMAL(5, 2) NOT NULL,

date\_of\_loan date,

foreign key (customer\_id) references customers(id\_customer),

foreign key (car\_id) references cars(id\_car),

);

create table leasing (

id\_leasing int primary key,

customer\_id int,

car\_id int,

sum\_leasing DECIMAL(10, 2) NOT NULL,

leasing\_term int,

down\_payment DECIMAL(10, 2) NOT NULL,

monthly\_payment DECIMAL(10, 2) NOT NULL,

interest\_rate DECIMAL(5, 2) NOT NULL,

date\_of\_loan date,

foreign key (customer\_id) references customers(id\_customer),

foreign key (car\_id) references cars(id\_car)

);

create table documents (

id\_document int primary key,

sales\_id int,

num\_act int,

num\_sales\_contract int,

num\_car\_maint int,

foreign key (sales\_id) references sales(id\_sale)

);

-- Таблица предпродажной подготовки

CREATE TABLE pre\_sale\_preparation (

id\_preparation INT PRIMARY KEY,

car\_id INT,

preparation\_date DATE NOT NULL,

preparation\_details VARCHAR(255),

cost DECIMAL(10, 2),

employee\_id INT,

FOREIGN KEY (car\_id) REFERENCES cars(id\_car),

FOREIGN KEY (employee\_id) REFERENCES employees(id\_employees)

);

-- Таблица trade-in

CREATE TABLE trade\_in (

id\_trade\_in INT PRIMARY KEY,

customer\_id INT,

car\_id INT,

trade\_in\_value DECIMAL(10, 2) NOT NULL,

trade\_in\_date DATE NOT NULL,

condition\_description VARCHAR(255),

employee\_id INT,

FOREIGN KEY (customer\_id) REFERENCES customers(id\_customer),

FOREIGN KEY (car\_id) REFERENCES cars(id\_car),

FOREIGN KEY (employee\_id) REFERENCES employees(id\_employees)

);

create table replenishment (

id\_rep int identity(1,1) primary key,

detail\_name varchar(20),

quantity int,

supplier\_id int,

sup\_name VARCHAR(100),

FOREIGN KEY (supplier\_id) REFERENCES suppliers(id\_supplier)

);

create table orders (

id\_order int identity(1,1) primary key,

car\_model VARCHAR(50),

cus\_surname VARCHAR(50),

date\_ord date,

price decimal(10,2),

em\_sur VARCHAR(50)

);

Созданные индексы, хранимые процедуры и триггеры, файл triggers\_etc.sql

use cardealer

go

--Триггер на обновление количества автомобилей на складе--

CREATE TRIGGER update\_car\_stock\_after\_sale

ON sales

AFTER INSERT

AS

BEGIN

UPDATE cars\_stock

SET quantity = quantity - 1

WHERE id\_stock = (SELECT id\_stock FROM cars WHERE id\_car IN (SELECT id\_car FROM INSERTED));

END;

--Триггер на изменение цены автомобиля--

CREATE TRIGGER update\_price\_in\_loans

ON cars

AFTER UPDATE

AS

BEGIN

UPDATE car\_loan

SET sum\_car\_loan = sum\_car\_loan \* (i.price / d.price)

FROM INSERTED i

JOIN DELETED d ON i.id\_car = d.id\_car;

END;

--Триггер для ведения журнала продаж--

CREATE TRIGGER log\_sales\_audit

ON sales

AFTER INSERT

AS

BEGIN

INSERT INTO sales\_audit (id\_sale, sale\_date, sale\_price)

SELECT id\_sale, sale\_date, sale\_price

FROM INSERTED;

END;

--stored procedure--

--Процедура для получения отчета о продажах--

CREATE PROCEDURE GetSalesReport

@StartDate DATETIME,

@EndDate DATETIME

AS

BEGIN

SET NOCOUNT ON;

SELECT sale\_date, sale\_price

FROM sales

WHERE sale\_date BETWEEN @StartDate AND @EndDate;

END;

--Процедура для получения информации о клиенте--

CREATE PROCEDURE GetCustomerInfo

@CustomerID INT

AS

BEGIN

SET NOCOUNT ON;

SELECT f\_name, l\_name, phone, email, ph\_address

FROM customers

WHERE id\_customer = @CustomerID;

END;

--Индекс на id\_stock--

CREATE UNIQUE INDEX idx\_cars\_stock\_id ON cars\_stock (id\_stock);

--Индекс на email--

CREATE UNIQUE INDEX idx\_customers\_email ON customers (email);

--Индекс на поле внешнего ключа в таблице car\_loan--

CREATE INDEX idx\_car\_loan\_customer\_id ON car\_loan (customer\_id);

Вставка тестовых данных в таблицы, файл insertion\_test\_cardealer.sql

use cardealer

go

insert into employees values (1,'Vetokhina','Anastasiia','admin','+7(981)703-76-01','avetokhina@naver.com','mycheetah')

insert into employees values ('Vetokhina','Anastasiia','admin','+7(981)703-76-01','avetokhina@naver.com','mycheetah')

select \*from customers

select \*from employees

select \*from suppliers

select \*from cars

select \*from car\_details

select \*from replenishment

select \*from orders

insert into car\_details values ('radiator grille',2,4,'693000');

insert into car\_details values ('bumpere',2,4,'104000');

insert into cars values ('Rolls-Royce Cullinan','2024','1FTEX1CP3JFA13968','25000000','0','available',2)

insert into cars values ('Rolls-Royce Ghost','2024','1FTEX1CP3JFA13969','25000000','100','sold',2);

insert into cars values ('Rolls-Royce Spectre','2024','1FTEX1CP3JFA13970','25000000','200','available',2);

insert into cars values ('Rolls-Royce Phantom','2024','1FTEX1CP3JFA13971','25000000','0','sold',2)

insert into suppliers values ('Good people','+7(921)345-08-09','wearegoodpeople@yandex.ru')

insert into customers values ('Rosalia','La','+7(921)345-08-09','wearegoodpeople@yandex.ru','')

# ПРИЛОЖЕНИЕ Б. Тексты программ

Форма главного меню для администратора, main\_menu.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace sql\_lab3

{

public partial class main\_menu : Form

{

public main\_menu()

{

InitializeComponent();

label1.Text = "Welcome to Main menu!";

label2.Text = "Choose the information you want to know";

button1.Text = "Customers and suppliers";

button2.Text = "Cars and details available";

button3.Text = "Car maintenance";

button4.Text = "Orders";

button5.Text = "Employees";

button6.Text = "Log-out";

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void button6\_Click(object sender, EventArgs e)

{

MessageBox.Show("Are you sure? Press the button OK down below to confirm the exit.");

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

customers\_suppliers form = new customers\_suppliers();

form.Show();

this.Hide();

}

private void button2\_Click(object sender, EventArgs e)

{

cars\_available form = new cars\_available();

form.Show();

this.Hide();

}

private void button3\_Click(object sender, EventArgs e)

{

car\_maintenance form = new car\_maintenance();

form.Show();

this.Hide();

}

private void button4\_Click(object sender, EventArgs e)

{

orders form = new orders();

form.Show();

this.Hide();

}

private void button5\_Click(object sender, EventArgs e)

{

employees form = new employees();

form.Show();

this.Hide();

}

private void label2\_Click(object sender, EventArgs e)

{

}

}

}

Форма технического обслуживания, car\_maintenance.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace sql\_lab3

{

public partial class car\_maintenance : Form

{

DataSet ds = new DataSet();

public car\_maintenance()

{

InitializeComponent();

label1.Text = "Ownership services";

label2.Text = "Customers last name:";

label3.Text = "Customers car VIN:";

label4.Text = "Service type:";

label5.Text = "Date of service:";

button1.Text = "Create";

label6.Text = "Enter car model:";

label7.Text = "Enter service type:";

button2.Text = "Search";

button3.Text = "Back to Main menu";

checkBox1.Text = "Customers partnership";

label8.Text = "Choose what you want to do in this window";

label9.Text = "Cost";

label10.Text = "Employee surname";

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void label2\_Click(object sender, EventArgs e)

{

}

private void button3\_Click(object sender, EventArgs e)

{

main\_menu form = new main\_menu();

form.Show();

this.Hide();

}

private void button2\_Click(object sender, EventArgs e)

{

string model = textBox2.Text;

string serv\_type = textBox3.Text;

string connectionString = "Data Source=DESKTOP-7E43SNE\\SQLEXPRESS;Initial Catalog=cardealer;Integrated Security=SSPI;";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT c.model AS [car model], sm.serv\_date AS [service date]," +

" sm.serv\_type AS [service type], sm.cost, e.l\_name as [employee surname] FROM " +

"service\_maintenance sm JOIN cars c ON sm.car\_id = c.id\_car JOIN employees e " +

"ON sm.employee\_id = e.id\_employees";

using (SqlCommand cmd = new SqlCommand(query, connection))

{

cmd.Parameters.AddWithValue("@model", (object)model ?? DBNull.Value);

cmd.Parameters.AddWithValue("@serv\_type", (object)serv\_type ?? DBNull.Value);

using (SqlDataAdapter adapter = new SqlDataAdapter(cmd))

{

ds.Clear();

adapter.Fill(ds, "service\_maintenance");

dataGridView1.DataSource = ds.Tables["service\_maintenance"];

}

}

}

}

catch (SqlException ex)

{

MessageBox.Show("Database error^ " + ex.Message);

}

catch (Exception ex)

{

MessageBox.Show("An error occured: " + ex.Message);

}

}

private void button1\_Click(object sender, EventArgs e)

{

string connectionString = "Data Source=DESKTOP-7E43SNE\\SQLEXPRESS;Initial Catalog=cardealer;Integrated Security=SSPI;";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string l\_name = textBox1.Text;

string vin = comboBox2.Text;

string service\_type = comboBox1.Text;

DateTime date\_serv = dateTimePicker1.Value;

string cost = textBox4.Text;

string employee\_last\_name = textBox5.Text;

string checkExistenceQuery = "select count(\*) from customers where l\_name = @l\_name";

using (SqlCommand checkEmail = new SqlCommand(checkExistenceQuery, connection))

{

checkEmail.Parameters.AddWithValue("@l\_name", l\_name);

int existingEmail = (int)checkEmail.ExecuteScalar();

if (existingEmail <= 0)

{

MessageBox.Show("Customer does not exist in this database. Add new customer.");

return;

}

}

string checkExistenceQueryVIN = "select count(\*) from cars where VIN = @vin";

using (SqlCommand checkEmail = new SqlCommand(checkExistenceQueryVIN, connection))

{

checkEmail.Parameters.AddWithValue("@vin", vin);

int existingEmail = (int)checkEmail.ExecuteScalar();

if (existingEmail <= 0)

{

MessageBox.Show("VIN does not exist in this database. Add new car.");

return;

}

}

string checkExistenceQueryEmployee = "select count(\*) from employees where l\_name = @employee\_last\_name";

using (SqlCommand checkEmail = new SqlCommand(checkExistenceQueryEmployee, connection))

{

checkEmail.Parameters.AddWithValue("@employee\_last\_name", employee\_last\_name);

int existingEmail = (int)checkEmail.ExecuteScalar();

if (existingEmail <= 0)

{

MessageBox.Show("Employee does not exist in this database. Add new employee.");

return;

}

}

string insertQuery = "INSERT INTO service\_maintenance (car\_id, serv\_date, serv\_type, customer\_id, cost, employee\_id) SELECT c.id\_car, @date\_serv, @service\_type, cu.id\_customer, @cost, e.id\_employees FROM cars c JOIN customers cu ON cu.l\_name = @l\_name JOIN employees e ON e.l\_name = @employee\_last\_name WHERE c.vin = @vin;";

using (SqlCommand insertCommand = new SqlCommand(insertQuery, connection))

{

insertCommand.Parameters.AddWithValue("@vin", vin);

insertCommand.Parameters.AddWithValue("@date\_serv", date\_serv);

insertCommand.Parameters.AddWithValue("@service\_type", service\_type);

insertCommand.Parameters.AddWithValue("@l\_name", l\_name);

insertCommand.Parameters.AddWithValue("@cost", cost);

insertCommand.Parameters.AddWithValue("@employee\_last\_name", employee\_last\_name);

insertCommand.ExecuteNonQuery();

}

MessageBox.Show("The request has been created succesfully.");

}

}

}

}

Форма машин и деталей в наличии, cars\_available.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace sql\_lab3

{

public partial class cars\_available : Form

{

DataSet ds = new DataSet();

SqlConnection cnn;

public cars\_available()

{

InitializeComponent();

label1.Text = "Cars and details in stock";

label2.Text = "Choose:";

label3.Text = "Detail name:";

button1.Text = "Search";

button2.Text = "Search";

label4.Text = "Result of search:";

label5.Text = "Result of search:";

button3.Text = "Back to Main menu";

label6.Text = "If there is no detail or car in stock, fill the form down below:";

label7.Text = "Detail name:";

label8.Text = "Supplier:";

label9.Text = "Quantity:";

button4.Text = "Send request";

comboBox1.Items.Add("available");

comboBox1.Items.Add("sold");

comboBox1.Items.Add("in repair");

label10.Text = "If you can't find anything, then you need to fill out a replenishment request down below";

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

private void button4\_Click(object sender, EventArgs e)

{

string connectionString = "Data Source=DESKTOP-7E43SNE\\SQLEXPRESS;Initial Catalog=cardealer;Integrated Security=SSPI;";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string detail\_name = textBox3.Text;

string sup\_name = textBox4.Text;

string quantity = textBox5.Text;

string insertQuery = "insert into replenishment (detail\_name,sup\_name,quantity)" + "values (@detail\_name, @sup\_name, @quantity)";

using (SqlCommand insertCommand = new SqlCommand(insertQuery, connection))

{

insertCommand.Parameters.AddWithValue("@detail\_name", detail\_name);

insertCommand.Parameters.AddWithValue("@sup\_name", sup\_name);

insertCommand.Parameters.AddWithValue("@quantity", quantity);

insertCommand.ExecuteNonQuery();

}

MessageBox.Show("Replenishment request was created successfully.");

textBox3.Clear();

textBox4.Clear();

textBox5.Clear();

}

}

private void button3\_Click(object sender, EventArgs e)

{

main\_menu form = new main\_menu();

form.Show();

this.Hide();

}

private void button1\_Click(object sender, EventArgs e)

{

string car\_status = comboBox1.Text;

string connectionString = "Data Source=DESKTOP-7E43SNE\\SQLEXPRESS;Initial Catalog=cardealer;Integrated Security=SSPI;";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query;

if (string.IsNullOrWhiteSpace(car\_status))

{

query = "SELECT model, VIN, price, mileage FROM cars";

}

else

{

query = "SELECT model, VIN, price, mileage FROM cars WHERE car\_status = @car\_status";

}

using (SqlDataAdapter adapter = new SqlDataAdapter(query, connection))

{

if (!string.IsNullOrWhiteSpace(car\_status))

{

adapter.SelectCommand.Parameters.AddWithValue("@car\_status", car\_status);

}

ds.Clear();

adapter.Fill(ds, "cars");

dataGridView1.DataSource = ds.Tables["cars"];

}

}

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void button2\_Click(object sender, EventArgs e)

{

string detail\_name = textBox2.Text;

string connectionString = "Data Source=DESKTOP-7E43SNE\\SQLEXPRESS;Initial Catalog=cardealer;Integrated Security=SSPI;";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query;

if (string.IsNullOrWhiteSpace(detail\_name))

{

query = "SELECT detail\_name, price FROM car\_details";

}

else

{

query = "SELECT detail\_name, price FROM car\_details WHERE detail\_name = @detail\_name";

}

using (SqlDataAdapter adapter = new SqlDataAdapter(query, connection))

{

if (!string.IsNullOrWhiteSpace(detail\_name))

{

adapter.SelectCommand.Parameters.AddWithValue("@detail\_name", detail\_name);

}

ds.Clear();

adapter.Fill(ds, "car\_details");

dataGridView2.DataSource = ds.Tables["car\_details"];

}

}

}

}

}

Форма заказа, orders.cs

using System;

using System.Collections;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace sql\_lab3

{

public partial class orders : Form

{

DataSet ds = new DataSet();

public orders()

{

InitializeComponent();

label1.Text = "Orders";

label2.Text = "Enter customer last name:";

button1.Text = "Search";

label3.Text = "Result:";

button9.Text = "Back to Main menu";

button7.Text = "Add new order";

button4.Text = "Renew information";

button3.Text = "Clear";

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void button9\_Click(object sender, EventArgs e)

{

main\_menu form = new main\_menu();

form.Show();

this.Hide();

}

private void button3\_Click(object sender, EventArgs e)

{

ArrayList Empty = new ArrayList();

dataGridView1.DataSource = Empty;

}

private void button7\_Click(object sender, EventArgs e)

{

new\_order form = new new\_order();

form.Show();

//this.Hide();

}

private void button4\_Click(object sender, EventArgs e)

{

new\_order form = new new\_order();

form.Show();

}

private void button1\_Click(object sender, EventArgs e)

{

string cus\_surname = textBox1.Text;

string connectionString = "Data Source=DESKTOP-7E43SNE\\SQLEXPRESS;Initial Catalog=cardealer;Integrated Security=SSPI;";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query;

if (string.IsNullOrWhiteSpace(cus\_surname))

{

//query = "SELECT car\_model as model, cus\_surname as [customer surname], date\_ord as [order date], price, em\_sur as [employee surname], payment\_status as [payment status] FROM orders";

query = "SELECT o.car\_model AS model, o.cus\_surname AS [customer surname], o.date\_ord AS [order date], " +

"o.price, o.em\_sur AS [employee surname], o.payment\_status AS [payment status], pm.method FROM orders o " +

"JOIN payment\_method pm ON o.payment\_method\_id = pm.id\_payment\_method";

}

else

{

query = "SELECT o.car\_model AS model, o.cus\_surname AS [customer surname], o.date\_ord AS [order date], " +

"o.price, o.em\_sur AS [employee surname], o.payment\_status AS [payment status], pm.method FROM orders o " +

"JOIN payment\_method pm ON o.payment\_method\_id = pm.id\_payment\_method WHERE o.cus\_surname = @cus\_surname";

}

using (SqlDataAdapter adapter = new SqlDataAdapter(query, connection))

{

if (!string.IsNullOrWhiteSpace(cus\_surname))

{

adapter.SelectCommand.Parameters.AddWithValue("@cus\_surname", cus\_surname);

}

ds.Clear();

adapter.Fill(ds, "orders");

dataGridView1.DataSource = ds.Tables["orders"];

}

}

}

}

}

Форма авторизации, authorization.cs

using Microsoft.SqlServer.Management.Smo;

using System;

using System.Collections;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.ListView;

namespace sql\_lab3

{

public partial class authorization : Form

{

//SqlConnection cnn;

public authorization()

{

InitializeComponent();

//cnn = new SqlConnection(@"Data Source=DESKTOP-7E43SNE\SQLEXPRESS;Integrated Security=True;Connect Timeout=30;Encrypt=True;Trust Server Certificate=True;Application Intent=ReadWrite;Multi Subnet Failover=False");

//cnn.Open();

label1.Text = "Authorization";

label2.Text = "Password";

label3.Text = "Email adress";

button1.Text = "Log-in";

label4.Text = "This database is for dealership employees only";

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

string connectionString = "Data Source=DESKTOP-7E43SNE\\SQLEXPRESS;Initial Catalog=cardealer;Integrated Security=SSPI;";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string login = textBox1.Text;

string password = textBox2.Text;

SqlCommand cmd = new SqlCommand("SELECT position FROM employees WHERE email = @login AND pass\_word = @password", connection);

cmd.Parameters.AddWithValue("@login", login);

cmd.Parameters.AddWithValue("@password", password);

string position = (string)cmd.ExecuteScalar();

if (position == "admin")

{

main\_menu form = new main\_menu();

form.Show();

this.Hide();

}

else if (position == "Manager")

{

mm form = new mm();

form.Show();

this.Hide();

}

else

{

MessageBox.Show("Incorrect login or password. Try again.");

}

////string query = "select count(\*) from customers where email = @login";

//string query = "select count(\*) from employees where email = @login and pass\_word = @password";

//using (SqlCommand command = new SqlCommand(query, connection))

//{

// //command.Parameters.AddWithValue("@login", login);

// //command.Parameters.AddWithValue("@password", password);

// command.Parameters.Clear();

// command.CommandText = query;

// command.Parameters.AddWithValue("@login", login);

// command.Parameters.AddWithValue("@password", password);

// int count = (int)command.ExecuteScalar();

// if (count > 0)

// {

// main\_menu form = new main\_menu();

// form.Show();

// this.Hide();

// }

// else

// {

// MessageBox.Show("Incorrect login or password. Try again.");

// }

//}

}

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

textBox2.UseSystemPasswordChar = true;

}

}

}

Форма сотрудников, employees.cs

using System;

using System.Collections;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace sql\_lab3

{

public partial class employees : Form

{

DataSet ds = new DataSet();

SqlConnection cnn;

public employees()

{

InitializeComponent();

label1.Text = "The list of employees";

label2.Text = "Enter last name of employee:";

button1.Text = "Search";

button2.Text = "Add new employee";

button3.Text = "Delete an employee";

button4.Text = "Back to Main menu";

button5.Text = "Clear";

cnn = new SqlConnection(@"Data Source=DESKTOP-7E43SNE\\SQLEXPRESS;Initial Catalog=cardealer;Integrated Security=SSPI;");

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void groupBox1\_Enter(object sender, EventArgs e)

{

}

private void button4\_Click(object sender, EventArgs e)

{

main\_menu form = new main\_menu();

form.Show();

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

string l\_name = textBox1.Text;

string connectionString = "Data Source=DESKTOP-7E43SNE\\SQLEXPRESS;Initial Catalog=cardealer;Integrated Security=SSPI;";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query;

if (string.IsNullOrWhiteSpace(l\_name))

{

query = "SELECT f\_name as [first name], l\_name as [last name], position, phone, email FROM employees";

}

else

{

query = "SELECT f\_name as [first name], l\_name as [last name], position, phone, email FROM employees WHERE l\_name = @l\_name";

}

using (SqlDataAdapter adapter = new SqlDataAdapter(query, connection))

{

if (!string.IsNullOrWhiteSpace(l\_name))

{

adapter.SelectCommand.Parameters.AddWithValue("@l\_name", l\_name);

}

ds.Clear();

adapter.Fill(ds, "employees");

dataGridView1.DataSource = ds.Tables["employees"];

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

add\_employee form = new add\_employee();

form.Show();

this.Hide();

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void button3\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

string firstNameToDelete = dataGridView1.SelectedRows[0].Cells["first name"].Value.ToString();

//int IdEmployeetToDelete = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["first name"].Value);

string deleteQuery = "DELETE FROM employees WHERE f\_name = @firstNameToDelete";

string connectionString = "Data Source=DESKTOP-7E43SNE\\SQLEXPRESS;Initial Catalog=cardealer;Integrated Security=SSPI;";

using (SqlConnection cnn = new SqlConnection(connectionString))

{

cnn.Open(); // Открываем соединение

using (SqlCommand cmd = new SqlCommand(deleteQuery, cnn))

{

// Добавляем параметры

cmd.Parameters.AddWithValue("@firstNameToDelete", firstNameToDelete);

cmd.ExecuteNonQuery(); // Выполняем команду

}

// Обновляем таблицу DataGridView

ds.Clear();

SqlDataAdapter da = new SqlDataAdapter("SELECT \* FROM employees", cnn);

da.Fill(ds, "employees");

dataGridView1.DataSource = ds.Tables["employees"];

}

}

else

{

MessageBox.Show("Choose the row for deleting.");

}

}

private void button5\_Click(object sender, EventArgs e)

{

ArrayList Empty = new ArrayList();

dataGridView1.DataSource = Empty;

}

}

}

Форма главного меню для менеджера по продажам, mm.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace sql\_lab3

{

public partial class mm : Form

{

public mm()

{

InitializeComponent();

label1.Text = "Welcome to Main menu!";

label2.Text = "Choose the information you want to know";

button1.Text = "Customers and suppliers";

button2.Text = "Cars and details available";

button4.Text = "Orders";

button6.Text = "Log-out";

}

private void button6\_Click(object sender, EventArgs e)

{

MessageBox.Show("Are you sure? Press the button OK down below to confirm the exit.");

this.Close();

}

private void label2\_Click(object sender, EventArgs e)

{

}

private void button5\_Click(object sender, EventArgs e)

{

}

private void button4\_Click(object sender, EventArgs e)

{

orders form = new orders();

form.Show();

this.Hide();

}

private void button3\_Click(object sender, EventArgs e)

{

}

private void button2\_Click(object sender, EventArgs e)

{

cars\_available form = new cars\_available();

form.Show();

this.Hide();

}

private void button1\_Click(object sender, EventArgs e)

{

customers\_suppliers form = new customers\_suppliers();

form.Show();

this.Hide();

}

private void label1\_Click(object sender, EventArgs e)

{

}

}

}

Форма регистрации, registration.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace sql\_lab3

{

public partial class registration : Form

{

public registration()

{

InitializeComponent();

label2.Text = "Adding new customer";

button1.Text = "Submit";

label3.Text = "First name";

label4.Text = "Last name";

label5.Text = "Phone";

label6.Text = "Email address";

label7.Text = "Residential address";

button2.Text = "Back to customers and suppliers";

checkBox1.Text = "Partnership";

label8.Text = "Fill the information down below";

button3.Text = "Renew";

}

private void label2\_Click(object sender, EventArgs e)

{

}

private void registration\_Load(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

string connectionString = "Data Source=DESKTOP-7E43SNE\\SQLEXPRESS;Initial Catalog=cardealer;Integrated Security=SSPI;";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string f\_name = textBox1.Text;

string l\_name = textBox2.Text;

string phone = textBox3.Text;

string email = textBox4.Text;

string ph\_address = textBox5.Text;

string checkExistenceQuery = "select count(\*) from customers where phone = @phone";

using (SqlCommand checkPhone = new SqlCommand(checkExistenceQuery,connection))

{

checkPhone.Parameters.AddWithValue("@phone", phone);

int existingPhone = (int)checkPhone.ExecuteScalar();

if (existingPhone > 0)

{

MessageBox.Show("Client with this phone number is already exist in this database.");

return;

}

}

string insertQuery = "insert into customers (f\_name,l\_name,phone,email,ph\_address)" + "values (@f\_name, @l\_name, @phone, @email, @ph\_address)";

using (SqlCommand insertCommand = new SqlCommand(insertQuery,connection))

{

insertCommand.Parameters.AddWithValue("@f\_name",f\_name);

insertCommand.Parameters.AddWithValue("@l\_name", l\_name);

insertCommand.Parameters.AddWithValue("@phone", phone);

insertCommand.Parameters.AddWithValue("@email", email);

insertCommand.Parameters.AddWithValue("@ph\_address", ph\_address);

insertCommand.ExecuteNonQuery();

}

MessageBox.Show("Client has been added to the database successfully.");

}

customers\_suppliers form = new customers\_suppliers();

form.Show();

this.Hide();

}

private void Button2\_Click(object sender, EventArgs e)

{

customers\_suppliers form = new customers\_suppliers();

form.Show();

this.Hide();

}

private void button3\_Click(object sender, EventArgs e)

{

string connectionString = "Data Source=DESKTOP-7E43SNE\\SQLEXPRESS;Initial Catalog=cardealer;Integrated Security=SSPI;";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string f\_name = textBox1.Text;

string l\_name = textBox2.Text;

string phone = textBox3.Text;

string email = textBox4.Text;

string ph\_address = textBox5.Text;

if (string.IsNullOrWhiteSpace(f\_name) || string.IsNullOrWhiteSpace(l\_name) || string.IsNullOrWhiteSpace(phone) || string.IsNullOrWhiteSpace(email) || string.IsNullOrWhiteSpace(ph\_address))

{

MessageBox.Show("Please, fill all the gaps.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

string updateQuery = "UPDATE customers SET f\_name = @f\_name, l\_name = @l\_name, phone = @phone, email = @email, ph\_address = @ph\_address where l\_name = @l\_name";

using (SqlCommand cmd = new SqlCommand(updateQuery, connection))

{

cmd.Parameters.AddWithValue("@f\_name", f\_name);

cmd.Parameters.AddWithValue("@l\_name", l\_name);

cmd.Parameters.AddWithValue("@phone", phone);

cmd.Parameters.AddWithValue("@email", email);

cmd.Parameters.AddWithValue("@ph\_address", ph\_address);

try

{

int rowsAffected = cmd.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("The data update succesfully.", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("Failed to update data. Please make sure the customer exists.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

catch (Exception ex)

{

MessageBox.Show("An error occurred while updating data: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

if (connection.State == ConnectionState.Open)

{

connection.Close();

}

}

}

}

}

}