

**КУРСОВОЙ ПРОЕКТ**

**По дисциплине: МДК 01.01 Разработка программных модулей**

**Тема: Разработка системы классов «Автосалон»**

**Специальность 09.02.07 «Информационные системы и программирование»**

|  |  |  |
| --- | --- | --- |
| **Выполнил студент(ка) группы 31ИС-23** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **М.С. Рудько** |
| **Руководитель** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Л.Б. Гусятинер** |

**Москва 2024**



**УТВЕРЖДАЮ**

**Зам. директора КМПО**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С.Ф. Гасанов**

**«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.**

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ**

**По дисциплине: МДК 01.01 Разработка программных модулей**

**Специальность 09.02.07 «Информационные системы и программирование»**

**Студент группы 302ИС-22 Рудько Матвей**

**ТЕМА: Разработка программного модуля информационной**

**системы «Автосалон»**

Дата выдачи задания «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_ 2024 г.

Срок сдачи проекта «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2024 г.

**Москва 2024**

**Содержание**

**ВВЕДЕНИЕ5**

ГЛАВА 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ7

**1.1 Изучение предметной области7**

1.2 Изучение существующих разработок9

ГЛАВА 2. ПРОЕКТНАЯ ЧАСТЬ10

**2.1 Построение диаграммы прецедентов10**

2.2 Построение диаграммы классов10

2.3 Выбор инструментов10

2.4 Проектирование сценария13

2.5 Построение базы данных14

ГЛАВА 3. ЧАСТЬ РЕАЛИЗАЦИИ и арар15

3.1 Начало разработки15

3.2 Описание главного модуля15

3.3 Описание спецификаций модулей16

3.4 Описание модуля «Клиент»17

3.5 Описание модуля «Сотрудник»17

3.6 Описание модуля «Автомобиль»17

3.7 Описание модуля «Заказ»17

3.8 Описание модуля «Модели»17

3.9 Описание модуля «Настройки»17

3.10 Описание модуля «Поиск»18

3.11 Описание модуля «Перечисления»18

3.12 Описание модуля «Валидация»18

3.13 Описание модуля «prelude»18

3.14 Описание модуля «Приостановка программы»18

3.15 Описание файла .env19

3.16 Тестирование19

3.17 Отладка22

ГЛАВА 4. ЭКСПЛУАТАЦИОННАЯ ЧАСТЬ23

5.1 Руководство оператора22

5.2 To-Do List26

ЗАКЛЮЧЕНИЕ27

СПИСОК ЛИТЕРАТУРЫ И ИНТЕРНЕТ-ИСТОЧНИКОВ29

**Введение**

Цель курсового проекта — разработка консольного приложения для управления данными автосалона. Программа позволит эффективно обрабатывать информацию о клиентской базе, автомобилях и заказах, обеспечивая структурированный подход к учёту данных. Такие решения особенно актуальны в условиях роста рынка автомобилей и увеличения объёмов информации.

Для реализации выбран язык Rust, благодаря его высокой производительности, строгой системе управления памятью и поддержке многопоточности. Это обеспечивает безопасность и надёжность обработки данных. Использование асинхронного программирования позволяет обрабатывать множество задач одновременно, что важно для автоматизации процессов.

Приложение включает следующие функции:

* управление базой данных сотрудников: добавление, удаление и редактирование информации;
* управление клиентской базой: добавление, удаление и обновление данных;
* учёт автомобилей: добавление, редактирование и удаление моделей;
* обработка заказов: создание, изменение статуса и удаление;
* поиск и фильтрация данных по критериям.

Проект состоит из четырех частей:

1. **Введение**: анализ актуальности, описание требований и практической значимости.
2. **Архитектура**: структура приложения, использование PostgreSQL, принципы модульности.
3. **Реализация и тестирование**: описание функций, взаимодействие модулей, примеры кода для работы с базой данных и описание процесса тестирования и отладки программы.
4. **Руководство пользователя**: инструкции по установке и использованию, демонстрация функционала. Также представлены планы дальнейшего развития программы.

В заключении подводятся итоги, анализируются результаты, эффективность инструментов и практическая значимость приложения. Таким образом, проект направлен на создание производительного инструмента для автоматизации работы автосалона.

**Глава 1.**

**Теоретическая часть**

* 1. **Изучение предметной области**

Современный рынок автомобилей требует от автосалонов высокой скорости обслуживания клиентов, точного учета автомобилей и гибкости в управлении заказами. Успешное управление этими процессами становится возможным только благодаря использованию современных информационных систем. Именно поэтому создание консольного приложения для автоматизации деятельности автосалона является актуальной задачей.

Приложение для автосалона предоставляет пользователю удобный интерфейс для работы с базой данных автомобилей, клиентов, заказов и сотрудников. Оно позволяет решать задачи учета, поиска и анализа данных, минимизируя ошибки и упрощая рутинные процессы. Особенно важно, что приложение доступно для использования как небольшими, так и крупными автосалонами благодаря своей гибкости и масштабируемости.

Простота и эффективность интерфейса такого приложения играют ключевую роль. Оно должно быть интуитивно понятным и не требовать от пользователя глубокой технической подготовки. Основной функционал приложения сосредоточен на автоматизации учета автомобилей, заказов и работы с клиентами.

**Цели использования приложения для автосалона:**

1. Автоматизация процессов управления автосалоном.
2. Единый стандарт учета данных о клиентах, заказах и автомобилях.
3. Повышение скорости и точности обработки данных.
4. Упрощение поиска информации по заданным критериям.

**Основные требования к приложению:**

1. Простота использования и интуитивно понятный интерфейс.
2. Высокая производительность при работе с большими объемами данных.
3. Возможность работы с данными клиентов, автомобилей и заказов через единую систему.

В автосалонах ежедневно ведется учет автомобилей, доступных для продажи, обработка заявок от клиентов и оформление заказов. Эти процессы требуют значительных временных и трудозатрат, особенно при использовании традиционных методов, таких как бумажные журналы или электронные таблицы. Автоматизация позволяет не только ускорить выполнение этих задач, но и исключить ошибки, связанные с человеческим фактором.

* 1. **Изучение существующих разработок**

В рамках изучения существующих разработок я провёл поиск консольных приложений, ориентированных на автоматизацию работы автосалонов. Основное внимание было уделено приложениям, которые могли бы выполнять задачи управления сотрудниками, клиентами, автомобилями и заказами. Также были рассмотрены решения, поддерживающие базовые функции, такие как добавление, редактирование, удаление данных и обработка заказов.

Однако в ходе исследования не удалось найти готового решения, полностью соответствующего поставленным задачам.

Исходя из этого, было принято решение о разработке собственного консольного приложения, которое будет включать все необходимые функции и удовлетворять требованиям, сформулированным для данного курсового проекта.

**Глава 2**

**Проектная часть**

**2.1 Построение диаграммы прецедентов**

(ПРИЛОЖЕНИЕ 17) (Диаграмма 1)

**2.2 Построение диаграммы классов**

(ПРИЛОЖЕНИЕ 17) (Диаграмма 2)

**2.3 Выбор инструментов**

При выборе языков программирования, я исходил из своих критериев.

Важность критерия выбрана из: низкая, ниже средней, средняя, выше средней, высокая.

*Таблица 1 -* Критерии выбора языка программирования

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Критерий | Опыт | Простота сопровождения | Понятный синтаксис | Скорость разработки | Перспективы дальнейшего изучения | Скорость выполнения кода |
| Важность критерия | Высокая | Выше среднего | Выше среднего | Средняя | Высокая | Выше среднего |

Исходя из этих критериев, я сравнил 3 языка программирования от 0 до 10 баллов за критерий, где 0 – самый низкий балл и 10 – самый высокий.

*Таблица 2 -* Сравнительный анализ языков программирования, основанный на собственных предпочтениях.

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий/Язык программирования | C | Python | Rust |
| Опыт | 2 | 6 | 10 |
| Простота сопровождения | 3 | 6 | 10 |
| Понятный синтаксис | 6 | 8 | 10 |
| Скорость разработки | 2 | 5 | 10 |
| Перспективы дальнейшего изучения | 5 | 10 | 10 |
| Скорость выполнения кода | 10 | 4 | 10 |
| Итого баллов | 28 | 39 | 60 |

По результатам сравнения я выбрал язык программирования Rust.

При выборе базы данных, я исходил из своих критериев.

Важность критерия выбрана из: низкая, ниже средней, средняя, выше средней, высокая.

*Таблица 3 -* Критерии выбора языка программирования

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Критерий | Опыт | Производительность при обработке данных | Поддержка сложных операций и запросов | Надёжность и соответствие ACID | Расширяемость и функциональные возможности | Масштабируемость и работа с большими данными |
| Важность критерия | Высокая | Выше среднего | Средняя | Средняя | Выше среднего | Высокая |

Исходя из этих критериев, я сравнил 3 языка программирования от 0 до 10 баллов за критерий, где 0 – самый низкий балл и 10 – самый высокий.

*Таблица 4 -* Сравнительный анализ баз данных, основанный на собственных предпочтениях.

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий/Язык программирования | МуSQL | PostgreSQL | MongoDB |
| Опыт | 5 | 10 | 8 |
| Производительность при обработке данных | 7 | 10 | 8 |
| Поддержка сложных операций и запросов | 6 | 10 | 5 |
| Надёжность и соответствие ACID | 8 | 10 | 6 |
| Расширяемость и функциональные возможности | 7 | 10 | 8 |
| Масштабируемость и работа с большими данными | 7 | 10 | 8 |
| Итого баллов | 40 | 60 | 43 |

По результатам сравнения я выбрал базу данных PostgreSQL.

**2.4 Проектирование сценария**

В данном разделе приведен сценарий использования программы пользователем.

(ПРИЛОЖЕНИЕ 17) (Схема 1)

Ниже приведу пример одного из множества вариантов использования приложения. Задача: нужно удалить работника с id1 8, т.к. его уволили.

Пользователь запускает программу, выбирает «войти» (у него уже есть своя учетная запись) и вводит свой id. Затем выбирает «настройки», «работник» и «редактировать». Вводит 8 (это id работника, которого нужно удалить), программа проверяет под каким id зашел пользователь, т.к. невозможно удалить учетную запись под которой он зашел. Если id не совпадают, то пользователь вводит пароль для удаления сотрудника и происходит удаление (если ввели неправильный пароль, то программа возвращает в главное меню и удаление не происходит). Затем программа выводит его в главное меню, и он нажимает «выход».

id1 - аббревиатура от английского слова identifier, которое переводится как «идентификатор». Индивидуальный номер.

**2.5 Построение базы данных**

Построим схему базы данных. В моей базе данных 4 таблицы: клиент, сотрудник, автомобиль, заказ.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | ***Клиент*** | | id | | Имя | | Телефон | | Email | | Бюджет | | |  | | --- | | ***Сотрудник*** | | id | | Имя | | Должность | | Зарплата | | Дата трудоустройства | | |  | | --- | | ***Машина*** | | id | | Марка | | Модель | | Год | | Цена | | Пробег | | Состояние | | |  | | --- | | ***Заказ*** | | id | | id машины | | id покупателя | | id сотрудника | | Дата создания | | Цена | | Актуальность | |

Из выше указанных таблиц, делаем вывод, что:

1) Для каждого клиента необходимо заполнить поля «Имя», «Телефон и «Бюджет». Email является опциональным.

2) Для каждого работника необходимо заполнить поля «Имя», «Должность», «Зарплата» и «Дата трудоустройства».

3) Для каждой машины необходимо заполнить поля «Марка», «Модель», «Год», «Цена», «Пробег» и «Состояние».

4) Для каждого заказа необходимо заполнить поля «id машины», «id покупателя», «id сотрудника», «Дата создания», «Цена» и «Актуальность».

5) Вводить id не нужно, т.к. с новой записью он автоматически увеличивается на 1.

**ГЛАВА 3**

**ЧАСТЬ РЕАЛИЗАЦИИ**

**3.1 Начало разработки**

Для реализации проекта «Автосалон» будет использована специализированная интегрированная среда разработки (IDE) от компании JetBrains, предназначенная для работы с языком программирования Rust.

RustRover имеет глубокую интеграцию с языком Rust и имеет множество встроенных инструментов для упрощения написания кода.

Для начала в файле Cargo.toml в разделе «dependencies» нужно написать библиотеки, которые будут использованы в проекте:

(ПРИЛОЖЕНИЕ 1)

Определение функции в main.rs, которая будет подключаться к базе данных PostgreSQL и возвращать объект подключения:

(ПРИЛОЖЕНИЕ 2)

Переменная *connection* представляет соединение с базой данных, которая обрабатывает входящие и исходящие данные.

Переменная *client* используется для выполнения запросов к базе данных.

**3.2 Описание главного модуля**

В главный модуль подключаются 7 основных модулей: модуль автомобиля, модуль клиента, модуль сотрудника, модуль заказа, модуль моделей, модуль настроек и модуль поиска. Так же в этом модуле происходит подключение к базе данных, создание таблиц (если они не созданы) и создание учетной записи или вход в учетную запись сотрудника.

Кроме того, подключаются 4 вспомогательных модуля: модуль перечислений, модуль prelude (для удобного импорта модулей), модуль валидации и модуль приостановления программы.

Главный модуль объединяет работу всех основных и вспомогательных модулей, обеспечивая их взаимодействие и инкапсулируя общую логику программы.

Данный модуль состоит из одного файла.

(ПРИЛОЖЕНИЕ 3)

Блок-схема главного модуля:

(ПРИЛОЖЕНИЕ 17) (Блок схема 1)

**3.3 Описание спецификаций к модулям**

Всего разработано 11 модулей, 7 из которых основных, а 4 второстепенных.

7 основных:

* Модуль автомобиля
* Модуль клиента
* Модуль сотрудника
* Модуль заказа
* Модуль моделей
* Модуль настроек
* Модуль поиска

4 второстепенных:

* Модуль перечислений
* Модуль prelude (для удобного импорта модулей)
* Модуль валидации
* Модуль приостановки программы

**3.4 Описание модуля «Клиент»**

В этом модуле описаны функции, которые относятся к клиенту.

(ПРИЛОЖЕНИЕ 4)

**3.5 Описание модуля «Сотрудник»**

В этом модуле описаны функции, которые относятся к сотруднику.

(ПРИЛОЖЕНИЕ 5)

**3.6 Описание модуля «Автомобиль»**

В этом модуле описаны функции, которые относятся к автомобилю.

(ПРИЛОЖЕНИЕ 6)

**3.7 Описание модуля «Заказ»**

В этом модуле описаны функции, которые относятся к заказу.

(ПРИЛОЖЕНИЕ 7)

**3.8 Описание модуля «Модели»**

В этом модуле описаны модели.

(ПРИЛОЖЕНИЕ 8)

**3.9 Описание модуля «Настройки»**

В этом модуле представлены функции для манипуляций с клиентом, сотрудником, машиной и заказом.

(ПРИЛОЖЕНИЕ 9)

**3.10 Описание модуля «Поиск»**

В этом модуле представлены функции поиска покупателей, сотрудников, машин и заказов по id.

(ПРИЛОЖЕНИЕ 10)

**3.11 Описание модуля «Перечисления»**

В этом модуле представлено перечисление возможных доменов электронной почты.

(ПРИЛОЖЕНИЕ 11)

**3.12 Описание модуля «Валидация»**

В этом модуле представлены функции валидации, вывода текста, преобразование типов и вывода старых значений при редактировании объекта.

(ПРИЛОЖЕНИЕ 12)

**3.13 Описание модуля prelude (для удобного импорта модулей)**

В этом модуле представлены организация и упрощения импорта модулей и функций в файлы для лучшей читаемости и понимания кода.

(ПРИЛОЖЕНИЕ 13)

**3.14 Описание модуля приостановки программы**

В этом модуле представлены функции для приостановления выполнения кода программы.

(ПРИЛОЖЕНИЕ 14)

**3.15 Описание файла .env**

В этом файле находятся данные базы данных.

(ПРИЛОЖЕНИЕ 15)

**3.16 Тестирование**

**Тест 1.** Проверка ввода имени при создании работника

Действия: При вводе имени я нажму клавишу «Enter», тем самым пропущу это поле.

Ожидаемый результат: программа попросит повторить ввод.

Результат: (ПРИЛОЖЕНИЕ 16) (Рисунок 1)

Итог: Ожидаемый результат совпал с фактическим результатом

**Тест 2.** Проверка на создание заказа

Действия: Я создаю заказ, заранее не выбрав покупателя.

Ожидаемый результат: Ошибка, т.к. не был выбрал покупатель. Результат: (ПРИЛОЖЕНИЕ 16) (Рисунок 2)

Итог: Ожидаемый результат совпал с фактическим результатом

**Тест 3.** Проверка неверного ввода email при создании клиента

Действия: При создании нового клиента я введу неправильный email.

Ожидаемый результат: Ошибка, программа попросит ввести корректный email.

Результат: (ПРИЛОЖЕНИЕ 16) (Рисунок 3)

Итог: Ожидаемый результат совпал с фактическим результатом

**Тест 4.** Удаление сотрудника

Действия: Я удаляю сотрудника

Ожидаемый результат: Программа попросит пароль:

* 1 тест. Пароль введен неверно, программа вернет меня в главное меню и сотрудник не будет удален.
* 2 тест. Пароль введен верно, сотрудник удален.

**1 тест.** Результат при неверном вводе пароля:

(ПРИЛОЖЕНИЕ 16) (Рисунок 4)

**2 тест**. Результат при правильном вводе пароля:

Таблица «Сотрудники» до удаления:

(ПРИЛОЖЕНИЕ 16) (Рисунок 5)

(ПРИЛОЖЕНИЕ 16) (Рисунок 6)

Таблица «Сотрудники» после удаления:

(ПРИЛОЖЕНИЕ 16) (Рисунок 7)

Итог: Ожидаемый результат совпал с фактическим результатом

**Тест 5.** Закрытие заказа

Действия: Заказ выполнили – его нужно закрыть (т.к. он больше неактуальный).

* 1 тест. Я введу несуществующий id.
* 2 тест. Я введу существующий id.
* 3 тест. Я введу буквы вместо цифр.

Ожидаемый результат:

* 1 тест. Ошибка, программа вернет в главное меню.
* 2 тест. Программа выполнит успешное изменения заказа и сделает его неактуальным.
* 3 тест. Ошибка, программа попросит ввести корректный id.

Результат:

1 тест.

(ПРИЛОЖЕНИЕ 16) (Рисунок 8)

2 тест.

(ПРИЛОЖЕНИЕ 16) (Рисунок 9)

(ПРИЛОЖЕНИЕ 16) (Рисунок 10)

(ПРИЛОЖЕНИЕ 16) (Рисунок 11)

3 тест.

(ПРИЛОЖЕНИЕ 16) (Рисунок 12)

Итог: Ожидаемый результат совпал с фактическим результатом

Для тестирования программы я провёл 5 ручных тестов, тщательно проверяя ключевые функциональные модули. Каждый тест охватывал различные сценарии использования, включая обработку корректных и некорректных данных. Все тесты завершились успешно — ожидаемый результат совпал с фактическим во всех случаях.

В ходе тестирования была также проверена работа основных функций, таких как добавление данных, их выборка, поиск, а также корректность взаимодействия между модулями. Кроме того, были проведены проверки на устойчивость программы к вводу некорректных данных и стабильность при выполнении последовательности операций. Это подтвердило надёжность и корректность работы программы.

**3.17 Отладка программы**

При написании кода я допустил ошибку, которая проявилась во время выполнения программы. Для проверки работоспособности приложения я ввел команду «cargo run», но компилятор выдал ошибку.

Компилятор указал, что после вызова «.await?» отсутствует знак «;». Это привело к тому, что синтаксический анализатор Rust не смог корректно завершить выражение. Конкретное сообщение компилятора выглядело так:

(ПРИЛОЖЕНИЕ 16)

Причина ошибки - метод «.await» завершает асинхронный вызов, и в синтаксисе Rust каждая инструкция должна завершаться точкой с запятой, если это не последний вызов в блоке. Отсутствие «;» нарушило синтаксическую структуру, так как компилятор ожидал завершения выражения.

Следуя подсказке компилятора «(help: add ';' here)», я добавил недостающую точку с запятой после «.await?». После внесения изменений программа успешно скомпилировалась и запустилась.

**4 ГЛАВА**

**ЭКСПЛУАТАЦИОННАЯ ЧАСТЬ**

**4.1 Руководство оператора**

**АННОТАЦИЯ**

В данном программном документе приведено руководство оператора по применению и эксплуатации консольной программы «Автосалон», предназначенной для облегчения работы автосалонов.

В данном программном документе, в разделе «Назначение программы» указаны сведения о назначении программы и информация, достаточная для понимания функций программы и её эксплуатации.

В разделе «Условия выполнения программы» указаны условия, необходимые для выполнения программы (минимальный состав аппаратных и программных средств и т.п.).

В данном программном документе, в разделе «Выполнение программы» указана последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы, приведено описание функций, формата и возможных вариантов команд, с помощью которых оператор осуществляет загрузку и управляет выполнением программы, а также ответы программы на эти команды.

**4.1.1 Назначение программы**

Специальное программное обеспечение «Автосалон» предназначено для автоматизации и упрощения ключевых процессов управления в автосалонах. Оно предоставляет удобные инструменты для работы с базами данных сотрудников, клиентов, автомобилей и заказов, что значительно облегчает организацию и контроль всех этапов работы.

С помощью программы можно быстро добавлять и редактировать данные о клиентах, формировать и отслеживать заказы, управлять информацией о сотрудниках, а также вести учёт автомобилей, включая их поиск и обновление данных.

Программное обеспечение также включает модули для анализа данных и формирования отчётов, что позволяет эффективно оценивать текущие показатели работы автосалона и принимать управленческие решения на основе актуальной информации.

Благодаря простому интерфейсу и логичной структуре, «Автосалон» подходит как для опытных пользователей, так и для начинающих, обеспечивая высокий уровень удобства и надёжности.

**4.1.2 Назначение функций**

1. «Войти» – войти в свою учетную запись сотрудника.
2. «Создать сотрудника» – создание нового сотрудника.
3. «Создать заказ» – создание нового заказа (сначала необходимо создать клиента, если он первый раз или выбрать уже существующего).
4. «Закрыть заказ» – закрытие заказа.
5. «Добавить клиента» - добавление клиента.
6. «Выбрать клиента» - выбор клиента.
   1. «Вывести всех клиентов» – вывод всех клиентов.
   2. «Найти клиента по фильтрам» - ввод определенных фильтров и вывод клиентов.
7. «Подобрать автомобиль» - подбор автомобиля.
   1. «По бюджету» - Выбор автомобиля по бюджету.
      1. «От» - ввод от какой суммы будут выведены машины.
      2. «До» - ввод от какой суммы будут выведены машины.

7.2 «По определенным фильтрам» - ввод определенных фильтров для поиска автомобилей.

1. «Добавить автомобиль» - создание нового автомобиля.
2. «Поиск» - поиск клиента, сотрудника, автомобиля и заказа по id.
3. «Настройки» - редактирование или удаление клиента, сотрудника, автомобиля или заказа.
4. «Выход» - программа прекращает свою работу.

**4.1.3 Условия выполнения программы**

Операционная система: Windows 98/XP/7/8/10 /macOS/Linux.

Процессор (CPU): Любой.

Оперативная память (RAM): 512 МБ.

Видеоадаптер: Любой.

Свободное место на жёстком диске: ~ 50Мб.

Установленный на компьютер язык программирования Rust (https://www.rust-lang.org/tools/install).

Установленный на компьютер Cargo.

Установленная на компьютер PostgreSQL (https://www.postgresql.org/download/).

**4.1.4 Выполнение программы**

Программу можно запустить 2 способами:

1. Создать файл .env
2. Вставить: DATABASE\_URL=postgres://username:password@localhost/database\_name

Вместо username, password и database\_name ввести свои значения

1 способ через консоль:

* 1. В консоль написать cargo build
  2. В консоль написать cargo run

2 способ использовать файл .exe:

* 1. В консоль написать cargo build –release
  2. Перейти по следуюзему пути:

название\_проекта/target/release/название\_проекта.exe

**4.2 To-Do List**

1. При входе в учетную запись сотрудника добавить ввод пароля.

2. Добавить программу скидок для покупателей.

3. Улучшить систему подсчета зарплаты сотрудников.

4. Добавить новые ограничения для создания заказов.

**Заключение**

В рамках курсового проекта была разработана программа «Автосалон», предназначенная для автоматизации ключевых процессов управления в автосалоне. Целью данного проекта являлось создание консольного приложения, обеспечивающего удобное управление базами данных сотрудников, клиентов, автомобилей и заказов. В процессе реализации программы были достигнуты все поставленные задачи, включая:

1. Управление базой данных работников, включая добавление, удаление и обновление.
2. Поиск и фильтрация данных по заданным критериям.
3. Учет автомобилей, включая добавление новых моделей, редактирование и удаление.

**Основные практические результаты**

1. **Функциональность управления базами данных:**

* Программа позволяет эффективно работать с данными сотрудников, клиентов и автомобилей.
* Реализована система управления заказами, включая их создание, редактирование и закрытие.

1. **Удобство использования:**

* Программа имеет простой и интуитивно понятный интерфейс, подходящий как для опытных пользователей, так и для начинающих.
* Встроенная система поиска и валидации данных позволяет минимизировать ошибки при работе с программой.

1. **Надёжность и безопасность:**

* Обеспечена обработка некорректного ввода для повышения устойчивости программы.

**Предложения по улучшению:**

1. **Интеграция графического интерфейса:**

* Добавление графического интерфейса для повышения удобства работы пользователей и расширения возможностей программы.

1. **Расширение функциональности:**

* Внедрение аналитических модулей для автоматического анализа данных о продажах, клиентах и заказах.

**Пути внедрения**

Разработанное приложение «Автосалон» может быть внедрено в рабочую среду автосалонов без значительных затрат. Для начала эксплуатации достаточно установить программу на сервер или рабочие станции и провести обучение сотрудников. Программа не требует значительных вычислительных ресурсов, что делает её доступной для большинства компаний.

Итогом работы над проектом стало создание функционального, безопасного и удобного консольного приложения, которое соответствует современным требованиям управления бизнес-процессами в автосалонах. Реализация предложенных доработок и дальнейшая модернизация программы позволят ещё больше расширить её возможности и повысить эффективность работы автосалонов.

**Список литературы и интернет-источников**

<https://www.infoconnector.ru/devices/glossary/c-terms/id-identifikator>

https://www.jetbrains.com/rust/

https://www.jetbrains.com/rust/whatsnew/2024-3/

<https://doc.rust-lang.ru/book/>

<https://metanit.com/rust/tutorial/>

<https://github.com/madmattheww/casino-Rust>

<https://github.com/madmattheww/rust_toDo_postgres>