**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное автономное образовательное   
учреждение высшего образования   
«Санкт-Петербургский политехнический университет Петра Великого»  
Инстритут среднего профессионального образования**

|  |  |
| --- | --- |
|  | **Допустить к защите** Заместитель директора  по УМР  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Андреев В.А.  (Подпись) (ФИО)  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 г. |
|  |

**курсовой проект**Тема «Приложение по работе с БД автосервиса»

специальность 09.02.07 группа 32919/1

Студент \_Гаврилин В.А. (подпись) (ФИО)

Преподаватель Ильин Ю.П.

(подпись) (ФИО)

Санкт-Петербург

2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc135128504)

[1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ 5](#_Toc135128505)

[1.1. Описание предметной области 5](#_Toc135128506)

[1.2. Анализ методов решения 6](#_Toc135128507)

[1.3. Обзор средств программирования 6](#_Toc135128508)

[1.4. Описание языка C# 9](#_Toc135128509)

[2. ПРАКТИЧЕСКАЯ ЧАСТЬ 14](#_Toc135128510)

[2.1. Постановка задачи 14](#_Toc135128513)

[2.1.1. Основания для разработки 14](#_Toc135128514)

[2.1.2. Назначение программы 14](#_Toc135128515)

[2.2. Проектирование приложения 14](#_Toc135128516)

[2.2.1. Диаграмма прецедентов 14](#_Toc135128517)

[2.2.2. Диаграмма классов 15](#_Toc135128518)

[2.2.3. Диаграмма последовательностей 20](#_Toc135128519)

[2.3. Текст программы 23](#_Toc135128520)

[2.4. Описание программы 23](#_Toc135128521)

[2.4.1. Общие сведения 23](#_Toc135128522)

[2.4.2. Функциональное назначение 23](#_Toc135128523)

[2.4.3. Описание логической структуры 23](#_Toc135128524)

[2.4.4. Используемые технические и программные средства 24](#_Toc135128525)

[2.4.5. Вызов и загрузка 24](#_Toc135128526)

[2.5. Руководство оператора 24](#_Toc135128527)

[2.5.1. Назначение программы 24](#_Toc135128528)

[2.5.2. Выполнение программы и сообщения оператору 24](#_Toc135128529)

[2.6. Программа и методика испытаний 26](#_Toc135128530)

[2.6.1. Объект испытаний 26](#_Toc135128531)

[2.6.2. Цель испытаний 26](#_Toc135128532)

[2.6.3. Требования к программе 26](#_Toc135128533)

[2.6.4. Требования к программной документации 27](#_Toc135128534)

[2.6.5. Средства и порядок испытаний 28](#_Toc135128535)

[2.6.6. Методы испытаний 28](#_Toc135128536)

[2.7. Протокол испытаний 30](#_Toc135128537)

[ЗАКЛЮЧЕНИЕ 35](#_Toc135128538)

[Список использованных источников 36](#_Toc135128539)

[ПРИЛОЖЕНИЕ А. Техническое задание 37](#_Toc135128540)

[1. ВВЕДЕНИЕ 38](#_Toc135128541)

[2. ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ 38](#_Toc135128542)

[3. НАЗНАЧЕНИЕ РАЗРАБОТКИ 38](#_Toc135128543)

[ПРИЛОЖЕНИЕ Б. Результаты испытаний программы 43](#_Toc135128544)

[ПРИЛОЖЕНИЕ В. Блок-схемы 52](#_Toc135128545)

[ПРИЛОЖЕНИЕ Г. Код программы 55](#_Toc135128546)

# ВВЕДЕНИЕ

В наше современное информационное время автомобильные сервисы сталкиваются с огромным объемом данных, требующих учета и эффективной организации. Решение этой задачи становится все более актуальным, и вот почему: для оптимальной работы автосервиса необходимо не только обслуживать автомобили, но и управлять клиентской базой, запасными частями, заказами, расписанием работ, а также отслеживать финансовые показатели и многими другими аспектами.

В этом контексте разработка и использование приложения для работы с базой данных автосервиса становится ключевым фактором в обеспечении эффективной и безошибочной деятельности. Такое приложение позволяет автосервису сосредоточиться на своей основной деятельности, минимизировать временные и финансовые затраты, а также повысить качество обслуживания клиентов.

В данном эссе мы рассмотрим основные преимущества и возможности, которые предлагает приложение для работы с базой данных автосервиса. Мы рассмотрим функциональность такого приложения, а также его влияние на повышение эффективности работы автосервиса, улучшение управления данными и обеспечение высокого уровня обслуживания клиентов.

# ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ

## Описание предметной области

Информационная система автосервиса — это программное обеспечение, которое используется для управления и координации работы автосервиса. Данная система позволяет автоматизировать основные процессы, связанные с управлением автосервиса, в том числе управление работниками, заказами и выполняемыми задачами.

Автосервис — это организация, которая предоставляет свои услуги по ремонту механических транспортных средств. В автосервисе осуществляется единый учет заказов, планирование и распределение работ между сотрудниками, управление ценами, а также координация работы сотрудников автосервиса.

В современном мире автосервис — это не только место, где починить транспортное средство. Автосервис является важным звеном в системе обслуживания, оказывая помощь в ремонте и обслуживании механических транспортных средств. Однако, для успешного функционирования автосервиса необходимо следить за качеством выполняемых работ, их ценами, сроками выполнения и другими параметрами.

Информационная система автосервиса помогает управлять всеми этими процессами, повышает эффективность работы автосервиса и облегчает жизнь клиентам и работникам, которые могут быстро и удобно сдать свой автомобиль на ремонт или посмотреть нудную информацию.

Важным аспектом работы автосервиса является обеспечение высокого уровня качества обслуживания и удовлетворенности клиентов. Информационная система автосервиса позволяет автоматизировать процессы обслуживания клиентов, ускорить поиск необходимой информации и минимизировать ошибки при работе.

В целом, использование информационной системы автосервиса позволяет существенно повысить эффективность работы автосервиса и обеспечить высокий уровень обслуживания клиентов. Благодаря автоматизации многих процессов сотрудники автосервиса могут более эффективно использовать свое время и уделить больше внимания обслуживанию клиентов. Кроме того, использование системы позволяет быстро и точно принимать управленческие решения, основываясь на данных о заказах, работниках и услугах.

Развитие компьютерных технологий в бизнесе, актуально и объективно необходимо.

## Анализ методов решения

Информационная система автосервиса может быть реализована следующими способами:

* ручной способ: Ручного подключения к серверу базы данных, и выполнение SQL скриптов;
* автоматизированный способ: для пользователя подготовлен пользовательский интерфейс, а также запросы, которые пользователь может выполнить;

В данном случае применяется автоматизированный способ, т.к. информационная система предполагает её использование неквалифицированным пользователем.

## Обзор средств программирования

Требования к курсовому проекту включали удобный пользовательский интерфейс, для сравнения я выбрал язык программирования C#.

Язык C#, разработанный компанией Microsoft, один из самых популярных современных языков программирования. C# применяют при работе с программами для ПК, создании сложных веб-сервисов или мобильных приложений. Появившийся как язык для собственных нужд платформы Microsoft .NET, постепенно этот язык стал очень популярным.

Некоторые достоинства языка C#:

* C# имеет поддержку подавляющего большинства продуктов Microsoft;
* Автоматическая «сборка мусора» (не нужно беспокоиться об освобождении памяти после уничтожения объекта);
* Современные функции являются частью C#, он мощный, надежный и масштабируемый в соответствии с современными тенденциями;
* C# имеет более простой и понятный синтаксис по сравнению с Java, что делает его более привлекательным для начинающих разработчиков;
* C# имеет мощный фреймворк .NET, который предоставляет много готовых компонентов для создания приложений.

Недостатки языка C#:

* C# не совместим с другими ОС, кроме Windows, поэтому он не является универсальным языком программирования;
* C# имеет меньшее количество разработчиков и сообществ, чем Java, поэтому вопросы в интернете могут быть решены менее быстро.

Таким образом, средством программирования данного курсового проекта был выбран язык С#, среда разработки Visual Studio 2017.

## Описание языка C#

C# — объектно-ориентированный язык программирования общего назначения. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, исключения.

Язык программирования C# применяется в самых разных сферах. Например, для создания видеоигр (и не только на Unity), микросервисы, бизнес-приложений, функциональных веб-приложений, приложений для Windows.

В таблице 1 приведены примеры ключевых слов.

**Ключевые слова C#:**

*Таблица 1*

|  |  |  |  |
| --- | --- | --- | --- |
| abstract | extern | Null | struct |
| as | false | object | switch |
| base | finally | operator | this |
| bool | fixed | out | throw |
| break | float | override | true |
| byte | for | params | try |
| case | foreach | private | typeof |
| catch | goto | protected | uint |
| const | if | public | ulong |
| continue | implicit | readonly | unchecked |
| decimal | in | ref | unsafe |
| default | int | return | ushort |
| delegate | interface | sbyte | using |
| do | internal | sealed | virtual |
| double | is | short | void |
| else | lock | sizeof | while |
| enum | long | stackalloc |  |
| event | namespace | static |  |
| explicit | new | string |  |

В таблице 2 приведены примеры контекстных ключевых слов.

**Контекстные ключевые слова C#**

*Таблица 2*

|  |  |  |  |
| --- | --- | --- | --- |
| add | equals | join | set |
| ascending | from | let | value |
| async | get | on | var |
| await | global | orderby | where |
| by | group | partial | yield |
| descending | in | remove |  |
| dynamic | into | select |  |

В таблице 3 приведен пример часто используемых в написании кода элементов языка C#.

**Основные алгоритмические конструкции C#**

*Таблица 3*

|  |  |
| --- | --- |
| **Названия конструкций, краткая информация о них** | **Примеры использования** |
| Оператор присваивания, унарные и бинарные операторы.  Оператор присваивания позволяет записывать в переменные любого типа соответствующие типу значения, унарные операторы способны увеличивать или уменьшать значения на единицу, бинарные позволяют выполнять сложение, вычитание, деление, умножение и др. | **Примеры:**  int x = 7; int y = 3;  int z = x + y; //Результат: z = 10  int z1 = y – x; //Результат: z1 = -4  double a = 54; double b = 3;  double c = a \* b; //Результат: c = 162  double c1 = a/b; //Результат: c1 = 18  double c2 = a++; //Результат: c2 = 54; a = 55  double c3 = ++a; //Результат: c3 = 55; a = 55  double c4 = a--; //Результат: c4 = 54; a = 53  double c5 = --a; //Результат: c5 = 53; a = 53  double c6 = a % b //Результат: c6 = 0  string greeting = “Hi, ” + “World!”;  //Результат: greeting = “Hi, World!” |

*Таблица 3. Продолжение*

|  |  |
| --- | --- |
| **Названия конструкций, краткая информация о них** | **Примеры использования** |
| Условные операторы  Известны два популярных условных оператора if и switch. Первый проверяет условие, если оно ложно, то либо выполняется конструкция, описанная в else (в том числе else if), либо пропускается (в случае, если блока else нет). Если условие истинно, выполняются инструкции, описанные в блоке.  Оператор switch более практичен в данной области, если речь идет о сложном выборе, так как предоставляет сразу все условия, и если ни одно не будет удовлетворено, отработает блок default (если он описан). | **Примеры:**  if ( 10 > 7 )  {  Console.Write(“true”);  }  else  {  Console.Write(“false”);  }//Результат: “true”  //Пользователь вводит значение  string menu = Console.ReadLine();  switch(menu)  {  case “1”: Console.Write(“Вы ввели 1.”);  break;  case “2”: Console.Write(“Вы ввели 2.”);  break;  default: Console.Write(“Вы ошиблись при вводе.”);  break;  }  //Результат зависит от указанного пользователем значения |
| Оператор цикла while  Сразу проверяет истинность условия и, если оно истинно, код цикла выполнится. | **Пример:**  int i = 0;  while (i =< 2)  {  Console.WriteLine(i);  i++;  }  //Результат: на экран будут последовательно выведены числа 0, 1, 2  int i = 0;  while (10 < 7)  {  Console.WriteLine(i);  i++;  }  //Результат: пустой вывод (условия ложно) |

*Таблица 3. Продолжение*

|  |  |
| --- | --- |
| **Названия конструкций, краткая информация о них** | **Примеры использования** |
| Оператор цикла do while  В цикле do сначала выполняется код цикла, а потом происходит проверка условия в инструкции while. И пока это условие истинно, цикл повторяется. | **Пример:**  int i = 0;  do  {  Console.WriteLine(i);  i++;  }  while (i =< 2);  //Результат: на экран будут последовательно выведены числа 0, 1, 2  int i = 0;  do  {  Console.WriteLine(i);  i++;  }  while (i > 2);  //Результат: на экран будет выведен только 0 |
| Оператор цикла for  Чаще всего применяется, когда нужно запустить цикл определённое количество раз, однако может быть и использован как цикл while, если указать только условие выхода из цикла. | **Пример:**  for(int i = 0; i=<2; i++)  {  Console.WriteLine(i);  }  //Результат: на экран будут последовательно выведены числа от 0 до 2  int i = 0;  for(; i<=2; )  {  Console.WriteLine(i);  i++;  }  //Результат: на экран будут последовательно выведены числа от 0 до 2 |

*Таблица 3. Продолжение*

|  |  |
| --- | --- |
| **Названия конструкций, краткая информация о них** | **Примеры использования** |
| Оператор цикла foreach  Цикл предназначен для перебора набора или коллекции элементов. | **Пример:**  int[] arr = { 1, 2, 3, 4};  foreach(int current in arr)  {  Console.WriteLine(current);  }  //Результат: на экран будут последовательно выведены числа от 1 до 4 |
| Операторы continue и break  Оператор break используется в случаях, когда требуется выйти из цикла, не дожидаясь его завершения.  Оператор continue используется в тех случаях, когда требуется пропустить текущую итерацию. | **Пример:**  int[] arr = { 1, 2, 3, 4, 5 };  foreach (int current in arr)  {  if (current == 3)  break;  Console.WriteLine(current);  }  //Результат: на экран будут выведены только 1 и 2  int[] arr = { 1, 2, 3, 4, 5 };  foreach (int current in arr)  {  if (current == 1)  continue;  Console.WriteLine(current); }  //Результат: на экран будут выведены значения начиная с 2 до 5 |

# ПРАКТИЧЕСКАЯ ЧАСТЬ



## Постановка задачи

### Основания для разработки

Разработка ведётся в среде программирования Visual Studio 2017 на языке C# на основании задания к курсовому проекту по профессиональному модулю ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем» МДК 01.02 «Прикладное программирование» и утверждена Институтом среднего профессионального образования.

### Назначение программы

Основное назначение программного продукта заключается в организации программы для работы с базой данных автосервиса: с помощью пользовательского интерфейса пользователю предоставляется возможность взаимодействия с базой данных.

## Проектирование приложения

На этапе проектирования были разработаны диаграмма прецедентов, диаграмма последовательностей и диаграмма классов, отвечающих всем ожиданиям от программы.

### Диаграмма прецедентов

Диаграмма прецедентов характеризует насколько программа будет полезна, а также определяет границы моделируемой предметной области на ранних этапах проектирования. При этом формирует общие требований к поведению проектируемой системы, а не как программа это сделает. Таким образом, диаграмма прецедентов – часть спецификации (рисунок 1).



Рисунок 1. Диаграмма прецедентов

### Диаграмма классов

Программа состоит из тридцати двух классов, шести интерфейсов, и 3 перечислений. Классы Репозитории отвечают за взаимодействие моделей с базой данных. Классы модели описывают за сущности в БД. Классы сервисы выступают прослойкой между репозиториями и контролерами. Классы конфигурации хранят общую информации о приложении и позволяют подключаться к базе данных. Классы контролера являются пользовательским интерфейсом приложения.

Для примера рассмотрим класс AvtoRrepository он содержит в себе одно поле db – объект класса MyApplicationContext, с помощью этого объекта производится обращение к базе данных. Также в классе присутствуют 6 методов: save(Avto,avto), используемый для сохранения объекта класса Avto, findAvtoByNumber(String number), используемый для поиска автомобиля по её номеру, findAvtoById(int id), используемый для поиска автомобиля по её id, delete(Avto avto), используемый для удаления записи об автомобиле из базы данных, findAllAvto(), используемый для получения списка всех автомобилей, хранящихся в базе данных, update(Avto avto), используемый для обновления данных об автомобиле в базе данных.

На рисунках 2–6 показана диаграмма классов проекта.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 2. Диаграмма классов

Изображение выглядит как текст, снимок экрана, число

Автоматически созданное описание

Рисунок 3. Диаграмма классов

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 4. Диаграмма классов

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 5. Диаграмма классов

Изображение выглядит как текст, снимок экрана, книга, Параллельный

Автоматически созданное описание

Рисунок 6. Диаграмма классов

### Диаграмма последовательностей

Диаграмма последовательностей показывает поток событий, происходящих в рамках одного из вариантов использования. Все действующие лица представлены в верхней части диаграммы, а стрелки – сообщения между лицами, необходимые для функционирования программы. На диаграмме объект представлен прямоугольником, от которого проведена прерывистая линия, так называемая линия жизни объекта, что представляет собой фрагмент жизненного цикла объекта в процессе взаимодействия с другими объектами. Диаграмма последовательностей моей программы представлена на рисунке 7.



Рисунок 7. Диаграмма последовательности

### Диаграмма базы данных

Диаграмма базы данных — это визуальное представление структуры и организации базы данных. Она отображает таблицы, атрибуты и связи между ними, а также описывает правила целостности данных. Диаграмма базы данных помогает разработчикам и администраторам баз данных лучше понять структуру и взаимосвязи данных в системе. Она может быть использована для проектирования новой базы данных, документирования существующей или визуализации данных для более наглядного анализа. Схема базы данных для моей программы представлена на рисунке 8.

Изображение выглядит как текст, диаграмма, снимок экрана, План

Автоматически созданное описание

Рисунок 8. Диаграмма последовательности

## Текст программы

Текст программы в соответствии с ГОСТ 19.101–77 (СТ СЭВ 1626-79) и ГОСТ 19.401-79 (СТ СЭВ 3746-82) представляет собой запись программы на исходном языке программирования с необходимыми комментариями. Текст программы представляет собой документ, выполненный машинным способом, и приведен в приложении Г.

## Описание программы

### Общие сведения

Программа для работы с базой данных сети аптек: с помощью пользовательского интерфейса пользователю предоставляется возможность взаимодействия с базой данных.

Программа написана на языке C# и работает под управлением операционной системы Windows 10.

### Функциональное назначение

Программа предназначена для решения бизнес-задач, она должна продемонстрировать навыки в разработке программного обеспечения, приобретённые в процессе изучения предмета.

### Описание логической структуры

Программа написана с использованием паттерна проектирования MVC, а также Service layer.

MVC (Model-View-Controller) — это шаблон проектирования, который используется для разделения компонентов приложения на три основных блока: модель (Model), представление (View) и контроллер (Controller). Модель представляет собой часть приложения, которая отвечает за хранение и управление данными, а также за бизнес-логику приложения. Представление отвечает за отображение данных пользователю, это может быть HTML-страница, графический интерфейс или другой способ визуализации данных. Контроллер обрабатывает запросы от пользователя, взаимодействуя с моделью и представлением, и управляет потоком данных между ними. Использование шаблона MVC позволяет лучше структурировать код приложения, улучшить его поддержку и расширяемость, а также повысить уровень безопасности.

Service layer — это слой (компонент) в архитектуре приложения, который предоставляет сервисы и функциональность для выполнения бизнес-логики и манипуляции с данными приложения. Service layer обычно является прослойкой между контроллером (или другими компонентами UI) и слоем доступа к данным (например, репозиторием или ORM). Основная задача Service layer — это инкапсуляция сложной бизнес-логики и бизнес-правил внутри слоя, чтобы упростить и улучшить поддержку и тестирование кода. Service layer может включать в себя такие функции, как валидация данных, обработка ошибок, транзакционность, работа с уведомлениями и т. д. Service layer позволяет сделать приложение более модульным и масштабируемым, позволяя легко добавлять новые функции и изменять существующие без необходимости изменения других компонентов приложения.

### Используемые технические и программные средства

Для нормального функционирования данной информационной системы необходим компьютер, клавиатура, мышь и следующие технические средства:

* процессор Intel или другой совместимый;
* объем свободной оперативной памяти ~40 Кб;
* объем необходимой памяти на жестком диске ~45Кб;
* стандартный VGA-монитор или совместимый;
* стандартная клавиатура;
* манипулятор «мышь».

### Вызов и загрузка

Программа может быть загружена как с диска, так и с жесткого диска. В последнем случае требуется предварительно переписать программу с диска на жесткий диск.

Исполняемым файлом программы является файл STO.exe. Для его запуска необходимо дважды щелкнуть по исполняемому файлу левой кнопкой мышки.

## Руководство оператора

### Назначение программы

Основное назначение программного продукта – подключение к базе данных и выполнения запросов к ней. Также присутствует возможность создания, чтения, редактирования и удаления данных из базы.

### Выполнение программы и сообщения оператору

Для запуска программы дважды щелкните левой кнопкой мыши по исполняемому файлу STO.exe.

Процесс работы с программой начинается с создания пользователя, для этого требуется нажать на кнопку «регистрация», после чего открывается форма регистрации. В форме регистрации требуется заполнить данные и выбрать роль.

После регистрации требуется ввести логин и пароль на форме для входа и нажать кнопку «Вход». В случае неверно введённых данных будет выведено предупреждение.

После входа в зависимости то роли будет показана одна из 3 форм.

Форма Администратора:

В форме администратора необходимо выбрать сущность, с которой вы хотите взаимодействовать. После чего будет предложено 4 варианта действий: создание записи, редактирование записи, просмотр записей, удаление записей.

При выборе создания записи открывается новая форма для создания. В форме требуется заполнить поля данными, после чего нажать кнопку «сохранить». После чего вас перенесет на начальную форму, а запись будет занесена в базу данных.

При выборе чтения записей, будет открыта новая форма, в которой в табличном виде будут представлены все записи, находящиеся в данной таблице.

При выборе редактирования записи, будет открыта новая форма, в которой будет предложено выбрать запись, которую требуется отредактировать, и поля, после заполнения которых, запись можно будет обновить, нажав на кнопку «Сохранить». После чего вас перенесет на начальную форму, а запись будет изменена в базе данных.

При выборе удаления записи, будет открыта новая форма, в которой потребуется ввести ID записи, которую требуется удалить, после нужно нажать на кнопку «удалить». После чего вас перенесет на начальную форму, а запись будет удалена из базы данных. В случае ввода ID несуществующей записи, будет выведено предупреждение.

Форма Администратора:

В форме администратора есть возможность выполнить 6 запросов к базе данных. Для того чтобы их выполнить 3 запроса, потребуется ввести имя владельца, номер автомобиля или номер заказа, а потом нажать на соответствующую кнопку. Далее появится окно, в которое будут выведен результат выбранного вами запроса. Для выполнения 4 и 5 запроса потребуется выбрать дату или диапазон дат, а потом нажать на соответствующую кнопку. Далее появится окно, в которое будут выведен результат вашего запроса. Для выполнения 6 запроса потребуется указать сразу несколько параметров, а потом нажать на соответствующую кнопку. Далее появится окно, в которое будут выведены результаты запроса.

Для выхода из приложения требуется закрыть все окна с помощью крестика в правом верхнем углу.

## Программа и методика испытаний

### Объект испытаний

Объектом испытаний является программа для работы с базой данных STO.exe.

### Цель испытаний

Целью испытаний является проверка соответствия программы требованиям Технического Задания.

### Требования к программе

* + - 1. Требования к функциональным характеристикам

Разработать программу для создания и работы с собственной БД (с возможностью добавления, удаления и корректировки информации в БД).

База данных должна содержать информацию о работниках автосервиса, услугах по ремонту, заказах на ремонт автомобилей, сроках выполнения заказов.

* Для каждого объекта создать класс с описанием необходимых полей и методов доступа.
* Для реализации поиска информации подготовить соответствующие методы.
* Подготовить форму пользовательского интерфейса для ввода информации об объектах и корректировки данных в случае необходимости.

Необходимо обеспечить возможность поиска информации:

1 версия:

* по фамилии владельца автомобиля;
* по номеру автомобиля;
* по номеру заказа;
* по дате оформления заказа или по диапазону дат;

1. версия:

* по совокупности признаков.
  + - 1. Требования к надежности
* использование лицензированного программного обеспечения;
* проверка программы на наличие вирусов;
* обеспечение обработки исключительных ситуаций.
  + - 1. Требования к составу и параметрам технических средств

Для нормального функционирования данной информационной системы необходим компьютер, клавиатура, мышь и следующие технические средства:

* процессор Intel или другой совместимый;
* объем свободной оперативной памяти ~500 Кб;
* объем необходимой памяти на жестком диске ~20Мб;
* стандартный VGA-монитор или совместимый;
* стандартная клавиатура;
* манипулятор «мышь».
  + - 1. Требования к информационной и программной совместимости

Для полноценного функционирования данной системы необходимо наличие операционной системы Microsoft Windows 10 или совместимой. Язык интерфейса – русский.

* + - 1. Требования к хранению

Программа должна поставляться на системный диск “H:”, в раздел 329191–5, в виде исполняемого (еxе) файла, документации и проекта. На диске должна быть создана папка с названием «Программа для работы с БД Автосервиса Гаврилин В. А. группа 32919/1", в которой будут еще три папки. Папка “EXE” с исполняемым файлом проекта; папка “SOURCE” с исходными текстами для первой и второй версий программы; папка “DOC” с документацией (ТЗ и пояснительной запиской).

### Требования к программной документации

Предварительный состав программной документации:

* «Техническое задание»;
* разрабатываемые программные модули должны быть самодокументированы, т. е. тексты программ должны содержать все необходимые комментарии;
* разрабатываемое программное обеспечение должно включать справочную систему.

### Средства и порядок испытаний

Испытания проводятся в следующем порядке:

1. проверяется наличие и комплектность программной документации (п. 2.6.4);
2. проверяется соответствие требованиям к хранению (п. 2.6.3.5);
3. проверяется соответствие требованиям к функциональным характеристикам (п. 2.6.3.1);
4. проверяется соответствие требованиям к информационной и программной совместимости (п. 2.6.3.4);
5. проверяется соответствие требованиям к надежности (п. 2.6.3.2);
6. проверяется соответствие требованиям к составу и параметрам технических средств (п. 2.6.3.3).

### Методы испытаний

* + - 1. Проверка программы на способность создания, аутентификации и авторизации пользователя.

Для проверки того, что программа способна сообщить пользователю, что введены неверные данные пользователя, необходимо:

* Запустить программу;
* Ввести данные несуществующего пользователя;
* Нажать кнопку «Вход»;
* Убедиться, что выведено сообщение, что введены неверный логин или пароль.

Для проверки того, что программа способна зарегистрировать нового пользователя, необходимо:

* Запустить программу;
* Нажать кнопка «Регистрация»;
* Заполнить данные о новом пользователе;
* Нажать кнопку «Регистрация»;
* Убедиться, что форма регистрации была закрыта, и снова выведена форма входа.

Для проверки того, что программа способна аутентифицировать и авторизовать пользователя, необходимо:

* Запустить программу;
* Ввести данные существующего пользователя;
* Нажать кнопку «Вход»;
* Убедиться, что форма входа была закрыта, и открыта стартовая форма для соответствующего пользователя.
  + - 1. Проверка программы на способность создавать, читать, редактировать и удалять данные в базе данных.

Для проверки того, что программа способна создать запись в базе данных, необходимо:

* Запустить программу;
* Авторизоваться в программе в качестве администратора;
* Выбрать таблицу для создания записи;
* Нажать на кнопку «Создать запись»;
* Ввести данные в новую форму;
* Нажать кнопку сохранить;
* Убедиться, что форма была закрыта, и в БД появилась новая запись.

Для проверки того, что программа способна читать записи в базе данных, необходимо:

* Запустить программу;
* Авторизоваться в программе в качестве администратора;
* Выбрать таблицу для чтения записи;
* Нажать на кнопку «Посмотреть записи»;
* Убедиться, что была открыта новая форма, в которой выведены записи соответствующей таблицы.

Для проверки того, что программа способна редактировать запись в базе данных, необходимо:

* Запустить программу;
* Авторизоваться в программе в качестве администратора;
* Выбрать таблицу для редактирования записи;
* Нажать на кнопку «Изменить запись»;
* Выбрать запись для редактирования в новой форме;
* Ввести данные в новую форму;
* Нажать кнопку сохранить;
* Убедиться, что форма была закрыта, и в БД данная запись была изменена.

Для проверки того, что программа способна удалить запись в базе данных, необходимо:

* Запустить программу;
* Авторизоваться в программе в качестве администратора;
* Выбрать таблицу для удаления записи;
* Нажать на кнопку «Удалить запись»;
* Ввести ID записи в новую форму;
* Нажать кнопку «Удалить»;
* Убедиться, что форма была закрыта, и в БД данная запись была удалена.

## Протокол испытаний

Результаты испытаний программы представлены в таблице 4, рисунки приведены в приложении Б.

**Результаты испытаний программы**

*Таблица 4.*

|  |  |  |  |
| --- | --- | --- | --- |
| **Проверяемые требования** | **Сообщения программы и вводимые значения** | **Ожидаемые результаты** | **Результаты** |
| **Проверка программы на способность создания, аутентификации и авторизации пользователя.** | | | |
| Способность программы обработки неверно введённых данных пользователя. | Запустить программу;  Ввести данные пользователя  Имя пользователя: «Гаврилин»  Пароль: «12345»  Нажать кнопку «Вход» | Выведено сообщение, что введено неверное имя пользователя или пароль. | Соответствует ожиданиям.  См. рис. 1, 2 |
| Способность программы создать нового пользователя. | Запустить программу;  Нажать на кнопку «Регистрация»  Ввести данные пользователя  Имя пользователя: «Гаврилин»  Пароль: «12345»  Выбрать роль: «Администратор»  Нажать кнопку «Зарегистрироваться» | В базе данных будет создана новая запись о пользователе. | Соответствует ожиданиям.  См. рис. 3,4 |
| Способность программы авторизовать и аутентифицировать пользователя. | Запустить программу;  Ввести данные пользователя  Имя пользователя: «Гаврилин»  Пароль: «12345»  Нажать кнопку «Вход» | Форма входа закроется, и откроется стартовая форма для администратора | Соответствует ожиданиям.  См. рис. 1,5 |
| **Проверка программы на способность создавать, читать, редактировать и удалять данные в базе данных** | | | |
| Способность программы создать запись в базе данных. | Запустить программу;  Авторизоваться в качестве администратора.  Выбрать таблицу для редактирования: «Услуги»  Нажать на кнопку «Создание записи»  Заполнить новую форму и нажать на кнопку «Сохранить» | В базу данных будет добавлена новая запись в таблицу «Услуги». | Соответствует ожиданиям.  См. рис. 6,7,8,9 |

*Таблица 4. Продолжение.*

|  |  |  |  |
| --- | --- | --- | --- |
| **Проверяемые требования** | **Сообщения программы и вводимые значения** | **Ожидаемые результаты** | **Результаты** |
| Способность программы отображать данные из базы данных. | Запустить программу;  Авторизоваться в качестве администратора.  Выбрать таблицу для чтения: «Услуги»  Нажать на кнопку «Посмотреть записи» | Будет выведена форма с записями из таблицы «Услуги» | Соответствует ожиданиям.  См. рис. 10 |
| Способность программы изменять записи в базе данных. | Запустить программу;  Авторизоваться в качестве администратора.  Выбрать таблицу для редактирования: «Услуги»  Нажать на кнопку «Изменить запись»  Выбрать из списка аптеку для редактирования  Заполнить форму и нажать на кнопку «Сохранить» | В базе данных будет изменена соответствующая запись в таблице «Услуги». | Соответствует ожиданиям.  См. рис. 11,12,13 |
| Способность программы удалять записи из базы данных. | Запустить программу;  Авторизоваться в качестве администратора.  Выбрать таблицу для редактирования: «Услуги»  Нажать на кнопку «Удалить запись»  Ввести ID записи  Нажать на кнопку «Удалить» | Запись с указанным ID была удалена из базы данных. | Соответствует ожиданиям.  См. рис. 14, 15,16 |

# ЗАКЛЮЧЕНИЕ

Разработанная в ходе выполнения курсового проекта программа удовлетворяет всем требованиям технического задания, что подтверждается протоколом испытаний.

Разработанная программа может быть использована реализации Информационной системы автосервиса.

# Список использованных источников

1. [Албахари, Дж.](https://www.chitai-gorod.ru/books/authors/albakhari_dzh_albakhari_b/) C# 7.0. Справочник. Полное описание языка / Дж. Албахари, Б. Албахари – Москва: Вильямс, 2018. – 1024 с.
2. Павловская Т.А. C# Программирование на языке высокого уровня. Учебник для вузов. / Т.А. Павловская – СПб: Питер, 2009. – 432 с.
3. Культин, Н. Microsoft Visual C# в задачах и примерах 2-е издание / Н. Культин – СПб: BHV, 2015 – 316 с.
4. Фленов, М.Е. Библия C#, 5-е издание / М.Е. Фленов – СПб: BHV, 2022 – 463 с.

**Электронные ресурсы:**

1. Юридическая фирма Интернет и Право. ГОСТ 19.105-78 Единая система программной документации. Общие требования к программным документам [Электронный ресурс] URL: https://internet-law.ru/gosts/gost/24534/
2. Юридическая фирма Интернет и Право. ГОСТ 19.106-78 Единая система программной документации. Требования к программным документам, выполненным печатным способом [Электронный ресурс] URL: https://internet-law.ru/gosts/gost/32437/
3. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению [Электронный ресурс] Режим доступа: https://www.swrit.ru/doc/espd/19.201-78.pdf
4. ГОСТ 19.301-78 Программа и методика испытаний. Требования к содержанию и оформлению [Электронный ресурс] Режим доступа: https://www.swrit.ru/doc/espd/19.301-79.pdf
5. Юридическая фирма Интернет и Право. ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению [Электронный ресурс] URL: https://internet-law.ru/gosts/gost/14688/
6. ГОСТ 19.401-78 Текст программы. Требования к содержанию и оформлению [Электронный ресурс] Режим доступа: https://www.swrit.ru/doc/espd/19.401-78.pdf

Wikipedia. Сравнение языков C Sharp и Java. [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/%D0%A1%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5\_C\_Sharp\_%D0%B8\_Java

# ПРИЛОЖЕНИЕ А. Техническое задание

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное автономное образовательное учреждение высшего образования  
«Санкт-Петербургский политехнический университет Петра Великого»  
(ФГАОУ ВО «СПбПУ»)  
**Институт среднего профессионального образования**

УТВЕРЖДАЮ  
Председатель ПЦК   
\_\_\_\_\_\_\_\_\_\_\_ Андреев В. А.  
\_\_.\_\_. 2023

Приложение по работе с БД АВтосервиса  
**Техническое задание**  
Листов 6

ПРОВЕРИЛ  
Преподаватель   
\_\_\_\_\_\_\_\_\_\_\_ Ильин Ю.П.  
\_\_.\_\_. 2023

ВЫПОЛНИЛ  
Студент группы 32919/1   
\_\_\_\_\_\_\_\_\_\_\_ Гаврилин В.А.  
\_\_.\_\_. 2023

2023

# ВВЕДЕНИЕ

* 1. Полное наименование программной разработки: «Приложение по работе с БД автосервиса».
  2. Приложение по работе с БД автосервиса: с помощью пользовательского интерфейса пользователю предоставляется возможность взаимодействия с базой данных. БД должна содержать информацию о работниках автосервиса, услугах по ремонту, заказах на ремонт автомобилей, сроках выполнения заказов.
  3. В соответствии с заданием программный продукт состоит из теоретической и практической частей. Теоретическая часть включает подробное описание работы с приложениями, показывающими схему работы системы и алгоритмы отдельных модулей. Практическая часть включает разработку и реализацию программных модулей программного продукта с использованием среды программирования Visual Studio 2017 и языка C#.
  4. Программа предназначена для коммерческих целей: ведения учета в автосервисах.

# ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

2.1. Разработка ведётся на основании задания к курсовому проекту по профессиональному модулю ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем» МДК 01.02 «Прикладное программирование» и утверждена Институтом среднего профессионального образования.

# НАЗНАЧЕНИЕ РАЗРАБОТКИ

* 1. Основное назначение программного продукта заключается в организации приложения по работе с БД автосервиса: с помощью пользовательского интерфейса пользователю предоставляется возможность взаимодействия с базой данных.

4 ТРЕБОВАНИЯ К РАЗРАБОТКЕ

4.1 Требования к функциональным характеристикам:

Разработать программу для создания и работы с собственной БД (с возможностью добавления, удаления и корректировки информации в БД).

БД должна содержать информацию о работниках автосервиса, услугах по ремонту, заказах на ремонт автомобилей, сроках выполнения заказов.

* Для каждого объекта создать класс с описанием необходимых полей и методов доступа
* Для реализации поиска информации подготовить соответствующие методы
* Подготовить форму пользовательского интерфейса для ввода информации об объектах и корректировки данных в случае необходимости

Необходимо обеспечить возможность поиска информации:

**1 версия:**

по фамилии владельца автомобиля;

по номеру автомобиля;

по номеру заказа;

по дате оформления заказа или по диапазону дат;

**2 версия:**

по совокупности признаков

4.2 Требования к надежности:

* использование лицензированного программного обеспечения;
* проверка программы на наличие вирусов;
* организация обработки исключительных ситуаций.

4.3 Требования к составу и параметрам технических средств

Для нормального функционирования данной информационной системы необходим компьютер, клавиатура, мышь и следующие технические средства:

- процессор Intel или другой совместимый;

- объем свободной оперативной памяти ~500 Кб;

- объем необходимой памяти на жестком диске ~20Мб;

- стандартный VGA-монитор или совместимый;

- стандартная клавиатура;

- манипулятор «мышь».

4.4 Требования к информационной и программной совместимости

Для полноценного функционирования данной системы необходимо наличие операционной системы Microsoft Windows 10 или совместимой. Язык интерфейса – русский.

4.5 Требования к хранению

Программа должна поставляться на системный диск “H:”, в раздел 329191–5, в виде исполняемого (еxе) файла, документации и проекта. На диске должна быть создана папка с названием «Программа для работы с БД Автосервиса Гаврилин В. А. группа 32919/1", в которой будут еще три папки. Папка “EXE” с исполняемым файлом проекта; папка “SOURCE” с исходными текстами для первой и второй версий программы; папка “DOC” с документацией (ТЗ и пояснительной запиской).

5 ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

5.1 Предварительный состав программной документации:

- «Техническое задание»;

- разрабатываемые программные модули должны быть самодокументированы, т. е. тексты программ должны содержать все необходимые комментарии;

- разрабатываемое программное обеспечение должно включать справочную систему.

5.2 Перечень материалов пояснительной записки

Введение

1. Теоретические основы разработки
   1. Описание предметной области
   2. Анализ методов решения
   3. Обзор средств программирования
   4. Описание выбранного языка программирования
2. Практическая часть
   1. Постановка задачи
   2. Описание схем
   3. Текст программы
   4. Описание программы
   5. Руководство оператора
   6. Программа и методика испытаний
   7. Протокол испытаний

Заключение

Список использованных источников

Приложения

6 ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

Технико-экономические показатели не рассчитываются.

7 СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Содержание стадии** | **Содержание этапа** | **Срок 2023 г.** | | **Форма  отчетности** |
| **начало** | **конец** |
| Техническое задание | Составление технического задания | 22.02.23 | 13.03.23 | Техническое задание |
| Эскизный проект | Разработка спецификаций | 14.03.23 | 01.04.23 | Спецификации программного обеспечения |
| Рабочий проект | Проектирование программы | 02.04.023 | 13.04.23 | Схема работы системы и спецификации компонентов |
| Составление программы | 14.04.23 | 21.04.23 | Программная документация |
| Приёмо-сдаточные испытания | 22.04.23 | 09.05.23 | Протокол испытаний (п. 2.7 пояснительной записки) |
| Приёмка | Защита курсового проекта | 10.05.23 | 17.05.23 | Оценка за курсовой проект |

8 ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ

8.1 Порядок контроля

Контроль выполнения должен осуществляться руководителем курсового проекта (преподавателем) в соответствие с п.7.

8.2 Порядок приемки

Приемка должна осуществляться с участием руководителя после проведения приемо-сдаточных испытаний. В результате защиты курсового проекта должна быть выставлена оценка за курсовой проект.

# ПРИЛОЖЕНИЕ Б. Результаты испытаний программы

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 1. Ввод данных не существующего пользователя

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 2. Сообщение об ошибке

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 3. Регистрация нового пользователя

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 4. Запись о новом пользователе в БД

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 5. Стартовая форма для администратора

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 6. Выбор таблицы «Услуги» для редактирования

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 7. Создание записи об услуге

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 8. Запись об услуге в БД

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 10. Чтение записей из БД

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 11. Редактирование Записи в БД



Рисунок 12. Отредактированная запись в БД

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 13. Удаление записи в БД

# ПРИЛОЖЕНИЕ В. Блок-схемы

Изображение выглядит как текст, диаграмма, Технический чертеж, План

Автоматически созданное описание

Рисунок 1. private void registration\_Click(object sender, EventArgs e)



Рисунок 2. Блок-схема метода private void save\_Click(object sender, EventArgs e)



Рисунок 3. Блок-схема для метода private void addMedicins\_Click(object sender, EventArgs e)

# ПРИЛОЖЕНИЕ Г. Код программы

ApplicationContext.cs

using carservice.model;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace carservice.config

{

public class MyApplicationContext : DbContext

{

public DbSet<User> Users => Set<User>();

public DbSet<Order> Orders => Set<Order>();

public DbSet<Carservice> Carservice => Set<Carservice>();

public DbSet<Worker> Workers => Set<Worker>();

public DbSet<Client> Clients => Set<Client>();

public MyApplicationContext()

{

Database.EnsureCreated();

}

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlServer("Server=GAVRILIN;Database=STO911;Trusted\_Connection=True;");

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Client>()

.HasMany(u => u.Orders)

.WithOne(u => u.Client)

.HasForeignKey(u => u.ClientId);

modelBuilder.Entity<Carservice>()

.HasMany(u => u.Orders)

.WithOne(u => u.CarService)

.HasForeignKey(u => u.CarServiceId);

//modelBuilder.Entity<Order>()

// .HasOne<Term>(u => u.Term)

// .WithOne(u => u.Order)

// .HasForeignKey<Term>(u => u.OrderId);

modelBuilder.Entity<Worker>()

.HasMany(u => u.Carservice)

.WithOne(u => u.Worker)

.HasForeignKey(u => u.Idworker);

}

}

}

Session.cs

using carservice.model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace carservice.config

{

class Session

{

private bool valid;

private User currentUser;

private Login formLogin;

private String tableForAdmin;

private String tableForDelete;

public Session(bool valid)

{

this.valid = valid;

}

public bool Valid { get => valid; set => valid = value; }

public User CurrentUser { get => currentUser; set => currentUser = value; }

public Login FormLogin { get => formLogin; set => formLogin = value; }

public string TableForAdmin { get => tableForAdmin; set => tableForAdmin = value; }

public string TableForDelete { get => tableForDelete; set => tableForDelete = value; }

public void setSession(User user, Login login)

{

Valid = true;

CurrentUser = user;

FormLogin = login;

}

}

}

CreateCarserviceForm.cs

using carservice.model;

using carservice.service.orders;

using carservice.service.carservices;

using carservice.service.carservicesOrders;

using carservice.service.worker;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Collections.Specialized.BitVector32;

namespace carservice.controller.admin.Create

{

public partial class CreateCarserviceForm : Form

{

OrdersServiceImpl ordersService = new OrdersServiceImpl();

CarserviceServiceImpl carservicesService = new CarserviceServiceImpl();

CarserviceOrdersService carservicesOrdersService = new CarserviceOrdersServiceImpl();

List<Order> orderssToSave = new List<Order>();

List<Order> allOrderss;

List<Worker> allWorkers;

public int bebra = 2;

WorkerServiceImpl workerService = new WorkerServiceImpl();

public CreateCarserviceForm()

{

InitializeComponent();

setWorkers();

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.MaximizeBox = false;

this.MinimizeBox = false;

save.Enabled = false;

}

private void saveEnable()

{

}

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

{

}

private void setWorkers()

{

allWorkers = workerService.getAll();

if (allWorkers != null)

{

foreach (var worker in allWorkers)

{

WorkerBox.Items.Add(worker.Name);

}

}

}

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

{

saveEnable();

}

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

{

Carservice carservice = new Carservice();

//carservice.Id = bebra;

bebra++;

carservice.Uzel = repairBox.Text;

List<Worker> allWorkers = workerService.getAll();

foreach (var item in allWorkers)

{

if (item.Name == WorkerBox.Text)

{

carservice.Idworker = item.Id;

}

}

carservicesService.save(carservice);

Close();

}

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

{

}

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

{

saveEnable();

}

private void label4\_Click(object sender, EventArgs e)

{

}

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

{

if (repairBox.Text.Length > 0 && WorkerBox.Text.Length > 0)

{

save.Enabled = true;

}

else

{

save.Enabled = false;

}

}

private void repairBox\_KeyPress(object sender, KeyPressEventArgs e)

{

e.KeyChar = '\0';

}

private void WorkerBox\_KeyPress(object sender, KeyPressEventArgs e)

{

e.KeyChar = '\0';

}

}

}

CreateClients.cs

using carservice.model;

using carservice.service.carserviceservice;

using carservice.service.client;

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 carservice.controller.admin.Create

{

public partial class CreateClients : Form

{

ClientServiceImpl clientService = new ClientServiceImpl();

public CreateClients()

{

InitializeComponent();

save.Enabled = false;

}

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

{

}

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

{

if (Name.Text.Length > 0 && AvtoName.Text.Length > 0 && Number.Text.Length > 0 && CarNumber.Text.Length > 0)

{

save.Enabled = true;

}

else

{

save.Enabled = false;

}

}

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

{

Client client = new Client();

client.Name = Name.Text;

client.AvtoName = AvtoName.Text;

client.Telephone = Number.Text;

client.AvtoNumber = CarNumber.Text;

clientService.save(client);

Close();

}

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

{

}

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

{

if (Name.Text.Length > 0 && AvtoName.Text.Length > 0 && Number.Text.Length > 0 && CarNumber.Text.Length > 0)

{

save.Enabled = true;

}

else

{

save.Enabled = false;

}

}

private void Name\_KeyPress(object sender, KeyPressEventArgs e)

{

if ((e.KeyChar >= 'A') && (e.KeyChar <= 'Z'))

return;

if ((e.KeyChar >= 'a') && (e.KeyChar <= 'z'))

return;

if ((e.KeyChar >= 'А') && (e.KeyChar <= 'Я'))

return;

if ((e.KeyChar >= 'а') && (e.KeyChar <= 'я'))

return;

if (e.KeyChar == (char)Keys.Back)

return;

if (e.KeyChar == (char)Keys.Tab)

return;

if (e.KeyChar == (char)Keys.Space)

return;

e.Handled = true;

}

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

{

if (Name.Text.Length > 0 && AvtoName.Text.Length > 0 && Number.Text.Length > 0 && CarNumber.Text.Length > 0)

{

save.Enabled = true;

}

else

{

save.Enabled = false;

}

}

private void Number\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar == (char)Keys.Back)

return;

if (e.KeyChar == (char)Keys.Tab)

return;

if (Number.Text.Length > 11)

{

e.KeyChar = '\0';

return;

}

if ((e.KeyChar >= '0') && (e.KeyChar <= '9'))

return;

if ((e.KeyChar == '+') && (Number.Text.Length == 0))

return;

e.KeyChar = '\0';

}

private void CarNumber\_MaskInputRejected(object sender, MaskInputRejectedEventArgs e)

{

if (Name.Text.Length > 0 && AvtoName.Text.Length > 0 && Number.Text.Length > 0 && CarNumber.Text.Length > 0)

{

save.Enabled = true;

}

else

{

save.Enabled = false;

}

}

private void CarNumber\_TextChanged\_1(object sender, EventArgs e)

{

if (Name.Text.Length > 0 && AvtoName.Text.Length > 0 && Number.Text.Length > 0 && CarNumber.Text.Length > 0)

{

save.Enabled = true;

}

else

{

save.Enabled = false;

}

}

}

}

CreateOrdersForm.cs

using carservice.enums;

using carservice.model;

using carservice.repository;

using carservice.service.worker;

using carservice.service.orders;

using carservice.service.client;

using carservice.service.carservices;

using carservice.service.carservicesOrders;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Net;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using carservice.service.carserviceservice;

namespace carservice.controller.admin.Create

{

public partial class CreateOrdersForm : Form

{

CarserviceServiceImpl carservicesService = new CarserviceServiceImpl();

ClientServiceImpl clientService = new ClientServiceImpl();

CarserviceOrdersServiceImpl carservicesOrdersService = new CarserviceOrdersServiceImpl();

OrdersServiceImpl ordersService = new OrdersServiceImpl();

List<Carservice> CarserviceToSave = new List<Carservice>();

List<Carservice> allCarservice = new List<Carservice>();

public CreateOrdersForm()

{

InitializeComponent();

allCarservice = carservicesService.getAll();

setClients();

setCarservicesList();

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.MaximizeBox = false;

this.MinimizeBox = false;

save.Enabled = false;

}

private void setCarservicesList()

{

List<Carservice> carservice = carservicesService.getAll();

foreach (var item in carservice)

{

CarserviceBox.Items.Add(item.Uzel);

}

}

private void setClients()

{

List<Client> allClients = clientService.getAll();

foreach (var item in allClients)

{

clientBox.Items.Add(item.Name);

}

}

private void saveEnable()

{

}

private void addEnable()

{

}

private void label8\_Click(object sender, EventArgs e)

{

}

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

{

Order orders = new Order();

orders.Cost = Convert.ToInt32(cost.Text);

orders.DateGet= dateTimePicker1.Value;

orders.DateDelivery= dateTimePicker2.Value;

List<Client> allClients = clientService.getAll();

foreach (var item in allClients)

{

if (item.Name == clientBox.Text)

{

orders.ClientId = item.Id;

}

}

List<Carservice> allCarservices = carservicesService.getAll();

foreach (var item in allCarservices)

{

if (item.Uzel == CarserviceBox.Text)

{

orders.CarServiceId = item.Id;

}

}

ordersService.saveOrders(orders);

Close();

}

private void dateTimePicker1\_ValueChanged(object sender, EventArgs e)

{

}

private void typeOfPackaging\_KeyPress(object sender, KeyPressEventArgs e)

{

e.KeyChar = '\0';

}

private void methodOfConsumption\_KeyPress(object sender, KeyPressEventArgs e)

{

e.KeyChar = '\0';

}

private void avtoBox\_KeyPress(object sender, KeyPressEventArgs e)

{

e.KeyChar = '\0';

}

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

{

saveEnable();

}

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

{

saveEnable();

}

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

{

saveEnable();

}

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

{

if (CarserviceBox.Text.Length > 0 && cost.Text.Length > 0 && clientBox.Text.Length > 0 && dateTimePicker1.Text.Length > 0 && dateTimePicker2.Text.Length > 0)

{

save.Enabled = true;

}

else

{

save.Enabled = false;

}

saveEnable();

}

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

{

saveEnable();

}

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

{

addEnable();

}

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

{

saveEnable();

}

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

{

}

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

{

if (CarserviceBox.Text.Length > 0 && cost.Text.Length > 0 && clientBox.Text.Length > 0 && dateTimePicker1.Text.Length > 0 && dateTimePicker2.Text.Length > 0)

{

save.Enabled = true;

}

else

{

save.Enabled = false;

}

}

private void cost\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar == (char)Keys.Back)

return;

if (e.KeyChar == (char)Keys.Tab)

return;

if (cost.Text.Length > 11)

{

e.KeyChar = '\0';

return;

}

if ((e.KeyChar >= '0') && (e.KeyChar <= '9'))

return;

e.KeyChar = '\0';

}

private void clientBox\_KeyPress(object sender, KeyPressEventArgs e)

{

e.KeyChar = '\0';

}

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

{

if (CarserviceBox.Text.Length > 0 && cost.Text.Length > 0 && clientBox.Text.Length > 0 && dateTimePicker1.Text.Length > 0 && dateTimePicker2.Text.Length > 0)

{

save.Enabled = true;

}

else

{

save.Enabled = false;

}

}

private void dateTimePicker1\_ValueChanged\_1(object sender, EventArgs e)

{

if (CarserviceBox.Text.Length > 0 && cost.Text.Length > 0 && clientBox.Text.Length > 0 && dateTimePicker1.Text.Length > 0 && dateTimePicker2.Text.Length > 0)

{

save.Enabled = true;

}

else

{

save.Enabled = false;

}

}

private void dateTimePicker2\_ValueChanged(object sender, EventArgs e)

{

if (CarserviceBox.Text.Length > 0 && cost.Text.Length > 0 && clientBox.Text.Length > 0 && dateTimePicker1.Text.Length > 0 && dateTimePicker2.Text.Length > 0)

{

save.Enabled = true;

}

else

{

save.Enabled = false;

}

}

}

}

CreateWorker.cs

using carservice.model;

using carservice.service.carservices;

using carservice.service.carserviceservice;

using carservice.service.worker;

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 carservice.controller.admin.Create

{

public partial class CreateWorker : Form

{

WorkerServiceImpl workerService = new WorkerServiceImpl();

CarserviceServiceImpl carserviceService = new CarserviceServiceImpl();

public CreateWorker()

{

InitializeComponent();

save.Enabled = false;

}

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

{

}

private void dateTimePicker1\_ValueChanged(object sender, EventArgs e)

{

}

private void label6\_Click(object sender, EventArgs e)

{

}

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

{

Worker worker = new Worker();

worker.Name = Name.Text;

worker.Experience = Convert.ToInt32(experience.Text);

worker.Jobtitle = Post.Text;

workerService.save(worker);

Close();

}

private void CreateWorker\_KeyPress(object sender, KeyPressEventArgs e)

{

}

private void Name\_KeyPress(object sender, KeyPressEventArgs e)

{

if ((e.KeyChar >= 'A') && (e.KeyChar <= 'Z'))

return;

if ((e.KeyChar >= 'a') && (e.KeyChar <= 'z'))

return;

if ((e.KeyChar >= 'А') && (e.KeyChar <= 'Я'))

return;

if ((e.KeyChar >= 'а') && (e.KeyChar <= 'я'))

return;

if (e.KeyChar == (char)Keys.Back)

return;

if (e.KeyChar == (char)Keys.Tab)

return;

if (e.KeyChar == (char)Keys.Space)

return;

e.Handled = true;

}

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

{

if (Name.Text.Length > 0 && experience.Text.Length > 0 && Post.Text.Length > 0)

{

save.Enabled = true;

}

else

{

save.Enabled = false;

}

}

private void experience\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar == (char)Keys.Back)

return;

if (e.KeyChar == (char)Keys.Tab)

return;

if (experience.Text.Length > 11)

{

e.KeyChar = '\0';

return;

}

if ((e.KeyChar >= '0') && (e.KeyChar <= '9'))

return;

e.KeyChar = '\0';

}

private void Post\_KeyPress(object sender, KeyPressEventArgs e)

{

if ((e.KeyChar >= 'A') && (e.KeyChar <= 'Z'))

return;

if ((e.KeyChar >= 'a') && (e.KeyChar <= 'z'))

return;

if ((e.KeyChar >= 'А') && (e.KeyChar <= 'Я'))

return;

if ((e.KeyChar >= 'а') && (e.KeyChar <= 'я'))

return;

if (e.KeyChar == (char)Keys.Back)

return;

if (e.KeyChar == (char)Keys.Tab)

return;

if (e.KeyChar == (char)Keys.Space)

return;

e.Handled = true;

}

}

}

DeleteForm.cs

using carservice.model;

using carservice.service.worker;

using carservice.service.orders;

using carservice.service.carservices;

using carservice.service.client;

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 carservice.controller.admin.Delete

{

public partial class DeleteForm : Form

{

ClientServiceImpl clientService = new ClientServiceImpl();

WorkerServiceImpl workerService = new WorkerServiceImpl();

OrdersServiceImpl ordersService = new OrdersServiceImpl();

CarserviceServiceImpl carservicesService = new CarserviceServiceImpl();

public DeleteForm()

{

InitializeComponent();

delete.Enabled = false;

}

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

{

if(deleteId.Text.Length > 0)

{

delete.Enabled = true;

}

else { delete.Enabled = false; }

}

private void deleteId\_KeyPress(object sender, KeyPressEventArgs e)

{

if ((e.KeyChar >= '0') && (e.KeyChar <= '9'))

return;

if (e.KeyChar == (char)Keys.Back)

return;

if (e.KeyChar == (char)Keys.Tab)

return;

e.KeyChar = '\0';

}

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

{

if(Login.Session.TableForDelete == "Услуги")

{

if (carservicesService.delete(Convert.ToInt32(deleteId.Text)))

{

Close();

}

else

{

MessageBox.Show("Запись с данным ID на найдена.");

}

}

if (Login.Session.TableForDelete == "Заказы")

{

if (ordersService.deleteOrders(Convert.ToInt32(deleteId.Text)))

{

Close();

}

else

{

MessageBox.Show("Запись с данным ID на найдена.");

}

}

if (Login.Session.TableForDelete == "Работники")

{

if (workerService.delete(Convert.ToInt32(deleteId.Text)))

{

Close();

}

else

{

MessageBox.Show("Запись с данным ID на найдена.");

}

}

if (Login.Session.TableForDelete == "Клиенты")

{

if (clientService.delete(Convert.ToInt32(deleteId.Text)))

{

Close();

}

else

{

MessageBox.Show("Запись с данным ID на найдена.");

}

}

}

}

}

ReadCarserviceForm.cs

using carservice.service.orders;

using carservice.service.carservices;

using carservice.service.worker;

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 carservice.controller.admin.Read

{

public partial class ReadCarserviceForm : Form

{

CarserviceServiceImpl carservicesService = new CarserviceServiceImpl();

WorkerServiceImpl workerService = new WorkerServiceImpl();

public ReadCarserviceForm()

{

InitializeComponent();

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.MaximizeBox = false;

this.MinimizeBox = false;

dataGridView1.ColumnCount = 3;

dataGridView1.RowCount = carservicesService.getAll().Count == 0 ? 1 : carservicesService.getAll().Count;

dataGridView1.Columns[0].Name = "ID";

dataGridView1.Columns[1].Name = "Ремонтируемый узел";

dataGridView1.Columns[2].Name = "Работник";

int i = 0;

foreach (var carservices in carservicesService.getAll())

{

dataGridView1.Rows[i].Cells[0].Value = carservices.Id;

dataGridView1.Rows[i].Cells[1].Value = carservices.Uzel;

dataGridView1.Rows[i].Cells[2].Value = workerService.findWorkerById(carservices.Idworker).Name;

i++;

}

}

}

}

ReadClientsForm.cs

using carservice.service.carserviceservice;

using carservice.service.client;

using carservice.service.worker;

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 carservice.controller.admin.Read

{

public partial class ReadClientsForm : Form

{

ClientServiceImpl clientService = new ClientServiceImpl();

public ReadClientsForm()

{

InitializeComponent();

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.MaximizeBox = false;

this.MinimizeBox = false;

dataGridView1.RowCount = clientService.getAll().Count == 0 ? 1 : clientService.getAll().Count;

dataGridView1.ColumnCount = 5;

dataGridView1.Columns[0].Name = "ID";

dataGridView1.Columns[1].Name = "ФИО";

dataGridView1.Columns[2].Name = "Название автомобиля";

dataGridView1.Columns[3].Name = "Номер автомобиля";

dataGridView1.Columns[4].Name = "Телефон";

int i = 0;

foreach (var client in clientService.getAll())

{

dataGridView1.Rows[i].Cells[0].Value = client.Id;

dataGridView1.Rows[i].Cells[1].Value = client.Name;

dataGridView1.Rows[i].Cells[2].Value = client.AvtoName;

dataGridView1.Rows[i].Cells[3].Value = client.AvtoNumber;

dataGridView1.Rows[i].Cells[4].Value = client.Telephone;

i++;

}

}

}

}

ReadOrdersForm.cs

using carservice.repository;

using carservice.service.orders;

using carservice.service.client;

using carservice.service.carservices;

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 carservice.controller.admin.Read

{

public partial class ReadOrdersForm : Form

{

OrdersServiceImpl ordersService = new OrdersServiceImpl();

CarserviceServiceImpl carserviceService = new CarserviceServiceImpl();

ClientServiceImpl clientService = new ClientServiceImpl();

public ReadOrdersForm()

{

InitializeComponent();

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.MaximizeBox = false;

this.MinimizeBox = false;

dataGridView1.RowCount = ordersService.getAll().Count == 0 ? 1 : ordersService.getAll().Count;

dataGridView1.ColumnCount = 6;

dataGridView1.Columns[0].Name = "ID";

dataGridView1.Columns[1].Name = "Дата приема";

dataGridView1.Columns[2].Name = "Дата сдачи";

dataGridView1.Columns[3].Name = "Номер клиента";

dataGridView1.Columns[4].Name = "Цена";

dataGridView1.Columns[5].Name = "Услуга";

int i = 0;

foreach(var orders in ordersService.getAll())

{

dataGridView1.Rows[i].Cells[0].Value = orders.Id;

dataGridView1.Rows[i].Cells[1].Value = orders.DateGet.Date;

dataGridView1.Rows[i].Cells[2].Value = orders.DateDelivery.Date;

dataGridView1.Rows[i].Cells[3].Value = clientService.GetById(orders.ClientId).Name;

dataGridView1.Rows[i].Cells[4].Value = orders.Cost;

dataGridView1.Rows[i].Cells[5].Value = carserviceService.getById(orders.CarServiceId).Uzel;

i++;

}

}

}

}

ReadWorkersForm.cs

using carservice.service.carserviceservice;

using carservice.service.orders;

using carservice.service.worker;

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 carservices.controller.admin.Read

{

public partial class ReadWorkersForm : Form

{

WorkerServiceImpl workersService = new WorkerServiceImpl();

public ReadWorkersForm()

{

InitializeComponent();

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.MaximizeBox = false;

this.MinimizeBox = false;

dataGridView1.RowCount = workersService.getAll().Count == 0 ? 1 : workersService.getAll().Count;

dataGridView1.ColumnCount = 4;

dataGridView1.Columns[0].Name = "ID";

dataGridView1.Columns[1].Name = "ФИО";

dataGridView1.Columns[2].Name = "Стаж";

dataGridView1.Columns[3].Name = "Должность";

int i = 0;

foreach (var worker in workersService.getAll())

{

dataGridView1.Rows[i].Cells[0].Value = worker.Id;

dataGridView1.Rows[i].Cells[1].Value = worker.Name;

dataGridView1.Rows[i].Cells[2].Value = worker.Experience;

dataGridView1.Rows[i].Cells[3].Value = worker.Jobtitle;

i++;

}

}

}

}

UpdateCarserviceForm.cs

using carservice.model;

using carservice.service.carservices;

using carservice.service.carservicesOrders;

using carservice.service.orders;

using carservice.service.worker;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml.Linq;

namespace carservice.controller.admin.Update

{

public partial class UpdateCarserviceForm : Form

{

OrdersServiceImpl ordersService = new OrdersServiceImpl();

CarserviceServiceImpl carservicesService = new CarserviceServiceImpl();

CarserviceOrdersService carservicesOrdersService = new CarserviceOrdersServiceImpl();

List<Order> orderssToSave = new List<Order>();

List<Order> allOrderss;

List<Worker> allWorkers;

WorkerServiceImpl workerService = new WorkerServiceImpl();

public UpdateCarserviceForm()

{

InitializeComponent();

setWorkers();

setCarservice();

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.MaximizeBox = false;

this.MinimizeBox = false;

save.Enabled = false;

}

private void saveEnable()

{

}

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

{

}

private void setWorkers()

{

allWorkers = workerService.getAll();

if (allWorkers != null)

{

foreach (var worker in allWorkers)

{

WorkerBox.Items.Add(worker.Name);

}

}

}

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

{

saveEnable();

}

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

{

Carservice carservice = carservicesService.getById(Convert.ToInt32(carserviceBox.Text));

carservice.Uzel = repairBox.Text;

List<Worker> allWorkers = workerService.getAll();

foreach (var item in allWorkers)

{

if (item.Name == WorkerBox.Text)

{

carservice.Idworker = item.Id;

}

}

carservicesService.update(carservice);

Close();

}

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

{

}

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

{

saveEnable();

}

private void label4\_Click(object sender, EventArgs e)

{

}

private void label9\_Click(object sender, EventArgs e)

{

}

private void setCarservice()

{

List<Carservice> products = carservicesService.getAll();

foreach (var item in products)

{

carserviceBox.Items.Add(item.Id);

}

}

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

{

saveEnable();

Carservice carservice = carservicesService.getById(Convert.ToInt32(carserviceBox.Text));

WorkerBox.Text = carservice.Worker.Name;

repairBox.Text = carservice.Uzel;

}

private void carserviceBox\_KeyPress(object sender, KeyPressEventArgs e)

{

e.KeyChar = '\0';

}

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

{

if (repairBox.Text.Length > 0 && WorkerBox.Text.Length > 0)

{

save.Enabled = true;

}

else

{

save.Enabled = false;

}

}

}

}

UpdateClientForm

using carservice.model;

using carservice.service.carservices;

using carservice.service.carservicesOrders;

using carservice.service.client;

using carservice.service.orders;

using carservice.service.worker;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml.Linq;

namespace carservice.controller.admin.Update

{

public partial class UpdateClientForm : Form

{

OrdersServiceImpl ordersService = new OrdersServiceImpl();

CarserviceServiceImpl carservicesService = new CarserviceServiceImpl();

CarserviceOrdersServiceImpl carservicesOrdersService = new CarserviceOrdersServiceImpl();

ClientServiceImpl clientService = new ClientServiceImpl();

List<Order> orderssToSave = new List<Order>();

List<Order> allOrderss;

List<Worker> allWorkers;

WorkerServiceImpl workerService = new WorkerServiceImpl();

public UpdateClientForm()

{

InitializeComponent();

setClient();

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.MaximizeBox = false;

this.MinimizeBox = false;

save.Enabled = false;

}

private void saveEnable()

{

}

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

{

}

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

{

saveEnable();

}

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

{

Client client = clientService.getByName(ClientBox.Text);

client.Name = Name.Text;

client.AvtoName = AvtoName.Text;

client.Telephone = Number.Text;

client.AvtoNumber = CarNumber.Text;

clientService.update(client);

Close();

}

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

{

}

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

{

if (ClientBox.Text.Length > 0 && Name.Text.Length > 0 && AvtoName.Text.Length > 0 && CarNumber.Text.Length > 0 && Number.Text.Length > 0)

{

save.Enabled = true;

}

else

{

save.Enabled = false;

}

saveEnable();

}

private void label4\_Click(object sender, EventArgs e)

{

}

private void label9\_Click(object sender, EventArgs e)

{

}

private void setClient()

{

List<Client> products = clientService.getAll();

foreach (var item in products)

{

ClientBox.Items.Add(item.Name);

}

}

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

{

saveEnable();

Client client = clientService.getByName(ClientBox.Text);

Name.Text = client.Name;

AvtoName.Text = client.AvtoName;

Number.Text = client.Telephone;

CarNumber.Text = client.AvtoNumber;

}

private void ClientBox\_KeyPress(object sender, KeyPressEventArgs e)

{

e.KeyChar = '\0';

}

private void Name\_KeyPress(object sender, KeyPressEventArgs e)

{

if ((e.KeyChar >= 'A') && (e.KeyChar <= 'Z'))

return;

if ((e.KeyChar >= 'a') && (e.KeyChar <= 'z'))

return;

if ((e.KeyChar >= 'А') && (e.KeyChar <= 'Я'))

return;

if ((e.KeyChar >= 'а') && (e.KeyChar <= 'я'))

return;

if (e.KeyChar == (char)Keys.Back)

return;

if (e.KeyChar == (char)Keys.Tab)

return;

if (e.KeyChar == (char)Keys.Space)

return;

e.Handled = true;

}

private void Number\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar == (char)Keys.Back)

return;

if (e.KeyChar == (char)Keys.Tab)

return;

if (Number.Text.Length > 11)

{

e.KeyChar = '\0';

return;

}

if ((e.KeyChar >= '0') && (e.KeyChar <= '9'))

return;

if ((e.KeyChar == '+') && (Number.Text.Length == 0))

return;

e.KeyChar = '\0';

}

private void CarNumber\_MaskInputRejected(object sender, MaskInputRejectedEventArgs e)

{

if (Name.Text.Length > 0 && AvtoName.Text.Length > 0 && Number.Text.Length > 0 && CarNumber.Text.Length > 0)

{

save.Enabled = true;

}

else

{

save.Enabled = false;

}

}

private void CarNumber\_TabIndexChanged(object sender, EventArgs e)

{

if (ClientBox.Text.Length > 0 && Name.Text.Length > 0 && AvtoName.Text.Length > 0 && CarNumber.Text.Length > 0 && Number.Text.Length > 0)

{

save.Enabled = true;

}

else

{

save.Enabled = false;

}

saveEnable();

}

}

}

UpdateOrderForm .cs

using carservice.model;

using carservice.service.carservices;

using carservice.service.carservicesOrders;

using carservice.service.client;

using carservice.service.orders;

using carservice.service.worker;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml.Linq;

namespace carservice.controller.admin.Update

{

public partial class UpdateOrderForm : Form

{

OrdersServiceImpl ordersService = new OrdersServiceImpl();

CarserviceServiceImpl carservicesService = new CarserviceServiceImpl();

CarserviceOrdersServiceImpl carservicesOrdersService = new CarserviceOrdersServiceImpl();

ClientServiceImpl clientService = new ClientServiceImpl();

List<Order> orderssToSave = new List<Order>();

List<Order> allOrderss;

List<Worker> allWorkers;

WorkerServiceImpl workerService = new WorkerServiceImpl();

public UpdateOrderForm()

{

InitializeComponent();

setOrder();

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.MaximizeBox = false;

this.MinimizeBox = false;

save.Enabled = false;

}

private void saveEnable()

{

}

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

{

}

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

{

saveEnable();

}

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

{

Order orders = ordersService.getById(Convert.ToInt32(OrderBox.Text));

orders.Cost = Convert.ToInt32(cost.Text);

orders.DateGet = dateTimePicker1.Value;

orders.DateDelivery = dateTimePicker2.Value;

List<Client> allClients = clientService.getAll();

foreach (var item in allClients)

{

if (item.Name == clientBox1.Text)

{

orders.ClientId = item.Id;

}

}

List<Carservice> allCarservices = carservicesService.getAll();

foreach (var item in allCarservices)

{

if (item.Uzel == CarserviceBox.Text)

{

orders.CarServiceId = item.Id;

}

}

ordersService.update(orders);

Close();

}

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

{

}

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

{

saveEnable();

}

private void label4\_Click(object sender, EventArgs e)

{

}

private void label9\_Click(object sender, EventArgs e)

{

}

private void setOrder()

{

List<Order> orders = ordersService.getAll();

foreach (var item in orders)

{

OrderBox.Items.Add(item.Id);

}

}

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

{

saveEnable();

Order order = ordersService.getById(Convert.ToInt32(OrderBox.Text));

CarserviceBox.Text = order.CarService.Uzel;

cost.Text = order.Cost.ToString();

clientBox1.Text = order.Client.Name;

dateTimePicker1.Value = order.DateGet;

dateTimePicker2.Value = order.DateDelivery;

}

private void OrderBox\_KeyPress(object sender, KeyPressEventArgs e)

{

e.KeyChar = '\0';

}

private void cost\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar == (char)Keys.Back)

return;

if (e.KeyChar == (char)Keys.Tab)

return;

if (cost.Text.Length > 11)

{

e.KeyChar = '\0';

return;

}

if ((e.KeyChar >= '0') && (e.KeyChar <= '9'))

return;

e.KeyChar = '\0';

}

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

{

if (OrderBox.Text.Length > 0 && CarserviceBox.Text.Length > 0 && cost.Text.Length > 0 && clientBox1.Text.Length > 0 && dateTimePicker1.Text.Length > 0 && dateTimePicker2.Text.Length > 0)

{

save.Enabled = true;

}

else

{

save.Enabled = false;

}

saveEnable();

}

}

}

UpdateWorkerForm.cs

using carservice.model;

using carservice.service.carservices;

using carservice.service.carservicesOrders;

using carservice.service.orders;

using carservice.service.worker;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml.Linq;

namespace carservice.controller.admin.Update

{

public partial class UpdateWorkerForm : Form

{

OrdersServiceImpl ordersService = new OrdersServiceImpl();

CarserviceServiceImpl carservicesService = new CarserviceServiceImpl();

CarserviceOrdersService carservicesOrdersService = new CarserviceOrdersServiceImpl();

List<Order> orderssToSave = new List<Order>();

List<Order> allOrderss;

List<Worker> allWorkers;

WorkerServiceImpl workerService = new WorkerServiceImpl();

public UpdateWorkerForm()

{

InitializeComponent();

setWorker();

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.MaximizeBox = false;

this.MinimizeBox = false;

save.Enabled = false;

}

private void saveEnable()

{

}

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

{

}

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

{

saveEnable();

}

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

{

Worker worker = workerService.GetByName(WorkerBox.Text);

worker.Name = Name.Text;

worker.Experience = Convert.ToInt32(experience.Text);

worker.Jobtitle = Post.Text;

List<Carservice> allcarservice = carservicesService.getAll();

workerService.update(worker);

Close();

}

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

{

}

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

{

if (WorkerBox.Text.Length > 0 && Name.Text.Length > 0 && experience.Text.Length > 0 && Post.Text.Length > 0)

{

save.Enabled = true;

}

else

{

save.Enabled = false;

}

saveEnable();

}

private void label4\_Click(object sender, EventArgs e)

{

}

private void label9\_Click(object sender, EventArgs e)

{

}

private void setWorker()

{

List<Worker> workers = workerService.getAll();

foreach (var item in workers)

{

WorkerBox.Items.Add(item.Name);

}

}

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

{

saveEnable();

Worker worker = workerService.GetByName(WorkerBox.Text);

Name.Text = worker.Name;

experience.Text = worker.Experience.ToString();

Post.Text = worker.Jobtitle;

}

private void WorkerBox\_KeyPress(object sender, KeyPressEventArgs e)

{

e.KeyChar = '\0';

}

private void Name\_KeyPress(object sender, KeyPressEventArgs e)

{

if ((e.KeyChar >= 'A') && (e.KeyChar <= 'Z'))

return;

if ((e.KeyChar >= 'a') && (e.KeyChar <= 'z'))

return;

if ((e.KeyChar >= 'А') && (e.KeyChar <= 'Я'))

return;

if ((e.KeyChar >= 'а') && (e.KeyChar <= 'я'))

return;

if (e.KeyChar == (char)Keys.Back)

return;

if (e.KeyChar == (char)Keys.Tab)

return;

if (e.KeyChar == (char)Keys.Space)

return;

e.Handled = true;

}

private void experience\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar == (char)Keys.Back)

return;

if (e.KeyChar == (char)Keys.Tab)

return;

if (experience.Text.Length > 11)

{

e.KeyChar = '\0';

return;

}

if ((e.KeyChar >= '0') && (e.KeyChar <= '9'))

return;

e.KeyChar = '\0';

}

private void Post\_KeyPress(object sender, KeyPressEventArgs e)

{

if ((e.KeyChar >= 'A') && (e.KeyChar <= 'Z'))

return;

if ((e.KeyChar >= 'a') && (e.KeyChar <= 'z'))

return;

if ((e.KeyChar >= 'А') && (e.KeyChar <= 'Я'))

return;

if ((e.KeyChar >= 'а') && (e.KeyChar <= 'я'))

return;

if (e.KeyChar == (char)Keys.Back)

return;

if (e.KeyChar == (char)Keys.Tab)

return;

if (e.KeyChar == (char)Keys.Space)

return;

e.Handled = true;

}

}

}

StartForm.cs

using carservice.controller.admin.Create;

using carservice.controller.admin.Update;

using carservice.controller.admin.Delete;

using carservice.controller.admin.Read;

using carservices.controller.admin.Read;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Numerics;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using carservice.model;

using carservice.service.carservices;

using carservice.service.orders;

using carservice.service.worker;

using carservice.service.client;

using static System.Collections.Specialized.BitVector32;

namespace carservice.controller.admin

{

public partial class StartForm : Form

{

CarserviceServiceImpl carserviceService = new CarserviceServiceImpl();

ClientServiceImpl clientService = new ClientServiceImpl();

WorkerServiceImpl workerService = new WorkerServiceImpl();

OrdersServiceImpl orderService = new OrdersServiceImpl();

public StartForm()

{

InitializeComponent();

wellcome.Text = "Добро пожаловать, " + Login.Session.CurrentUser.UserName + "!";

table.Items.Add("Услуги");

table.Items.Add("Заказы");

table.Items.Add("Работники");

table.Items.Add("Клиенты");

Create.Enabled = false;

Read.Enabled = false;

Update.Enabled = false;

Delete.Enabled = false;

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.MaximizeBox = false;

this.MinimizeBox = false;

}

private void StartForm\_FormClosed(object sender, FormClosedEventArgs e)

{

Login.Session.FormLogin.Visible = true;

}

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

{

Login.Session.TableForAdmin = table.Text;

if(Login.Session.TableForAdmin == "Услуги")

{

CreateCarserviceForm form = new CreateCarserviceForm();

form.ShowDialog();

}

if (Login.Session.TableForAdmin == "Заказы")

{

CreateOrdersForm form = new CreateOrdersForm();

form.ShowDialog();

}

if (Login.Session.TableForAdmin == "Работники")

{

CreateWorker form = new CreateWorker();

form.ShowDialog();

}

if (Login.Session.TableForAdmin == "Клиенты")

{

CreateClients form = new CreateClients();

form.ShowDialog();

}

}

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

{

if(Create.Text.Length > 0 && Read.Text.Length > 0 && Update.Text.Length > 0 && Delete.Text.Length > 0)

{

Create.Enabled = true;

Read.Enabled = true;

Update.Enabled = true;

Delete.Enabled = true;

}

else

{

Create.Enabled = false;

Read.Enabled = false;

Update.Enabled = false;

Delete.Enabled = false;

}

}

private void table\_KeyPress(object sender, KeyPressEventArgs e)

{

e.KeyChar = '\0';

}

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

{

Login.Session.TableForAdmin = table.Text;

if (Login.Session.TableForAdmin == "Услуги")

{

ReadCarserviceForm form = new ReadCarserviceForm();

form.ShowDialog();

}

if (Login.Session.TableForAdmin == "Заказы")

{

ReadOrdersForm form = new ReadOrdersForm();

form.ShowDialog();

}

if (Login.Session.TableForAdmin == "Работники")

{

ReadWorkersForm form = new ReadWorkersForm();

form.ShowDialog();

}

if (Login.Session.TableForAdmin == "Клиенты")

{

ReadClientsForm form = new ReadClientsForm();

form.ShowDialog();

}

}

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

{

Login.Session.TableForDelete = table.Text;

DeleteForm form = new DeleteForm();

form.ShowDialog();

}

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

{

}

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

{

Login.Session.TableForAdmin = table.Text;

if (Login.Session.TableForAdmin == "Услуги")

{

UpdateCarserviceForm form = new UpdateCarserviceForm();

form.ShowDialog();

}

if (Login.Session.TableForAdmin == "Клиенты")

{

UpdateClientForm form = new UpdateClientForm();

form.ShowDialog();

}

if (Login.Session.TableForAdmin == "Заказы")

{

UpdateOrderForm form = new UpdateOrderForm();

form.ShowDialog();

}

if (Login.Session.TableForAdmin == "Работники")

{

UpdateWorkerForm form = new UpdateWorkerForm();

form.ShowDialog();

}

}

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

{

if (radioButton1.Checked)

{

Client client = clientService.getByName(textBox1.Text);

String result = "Результат: ";

result += client.Name + " " + client.Telephone + " " + client.AvtoNumber + " " + client.AvtoName;

if (result != "Результат: ")

{

MessageBox.Show(result);

}

else

{

MessageBox.Show("Ничего не найдено");

}

}

else if (radioButton2.Checked)

{

List<Client> clients = clientService.getByAvtoNumber(textBox2.Text);

String result = "Результат: ";

foreach (var client in clients)

{

result += client.Name + " " + client.Telephone + " " + client.AvtoNumber + " " + client.AvtoName + "\n";

}

if (result != "Результат: ")

{

MessageBox.Show(result);

}

else

{

MessageBox.Show("Ничего не найдено");

}

}

else if (radioButton3.Checked)

{

Order order = orderService.getById(Convert.ToInt32(textBox3.Text));

String result = "Результат: " + order.Client.Name + " " + order.CarService.Uzel + "\n";

if (result != "Результат: ")

{

MessageBox.Show(result);

}

else

{

MessageBox.Show("Ничего не найдено");

}

}

else if (radioButton4.Checked)

{

List<Order> orders = orderService.findByDate(dateTimePicker1.Value);

String result = "Результат: ";

foreach (var order in orders)

{

result += clientService.GetById(order.ClientId).Name + "\n";

}

if (result != "Результат: ")

{

MessageBox.Show(result);

}

else

{

MessageBox.Show("Ничего не найдено");

}

}

else if (radioButton5.Checked)

{

List<Order> orders = orderService.findByRangeDate(dateTimePicker2.Value, dateTimePicker3.Value);

String result = "Результат: ";

foreach (var order in orders)

{

result += clientService.GetById(order.ClientId).Name + "\n";

}

if (result != "Результат: ")

{

MessageBox.Show(result);

}

else

{

MessageBox.Show("Ничего не найдено");

}

}

else if (radioButton6.Checked)

{

int idForSearch;

try

{

idForSearch = Convert.ToInt32(textBox3.Text);

}

catch

{

idForSearch = 228;

}

List<Order> orders = orderService.findFilteredOrders(textBox1.Text, textBox2.Text, idForSearch, dateTimePicker1.Value, dateTimePicker2.Value, dateTimePicker3.Value);

String result = "Результат: ";

foreach (var order in orders)

{

result += clientService.GetById(order.ClientId).Name + "\n";

}

if (result != "Результат: ")

{

MessageBox.Show(result);

}

else

{

MessageBox.Show("Ничего не найдено");

}

}

}

}

}

StartForm.cs

using carservice.model;

using carservice.service.worker;

using carservice.service.orders;

using carservice.service.carservices;

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 carservice.controller.pharmacist

{

public partial class StartForm : Form

{

OrdersServiceImpl ordersService = new OrdersServiceImpl();

CarserviceServiceImpl carservicesService = new CarserviceServiceImpl();

public StartForm()

{

InitializeComponent();

var allOrderss = ordersService.getAll();

wellcome.Text = "Добро пожаловать, " + Login.Session.CurrentUser.UserName + "!";

allOrderss = ordersService.getAll();

var allCarservicess = carservicesService.getAll();

if (allCarservicess != null)

{

foreach (var carservices in allCarservicess)

{

}

}

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.MaximizeBox = false;

this.MinimizeBox = false;

WhereCanBuy.Enabled = false;

WhereCanBuyCost.Enabled = false;

PostingThisMonth.Enabled = false;

avtoButton.Enabled = false;

}

private void buttonEnabled()

{

if (ordersBox.Text.Length > 0)

{

WhereCanBuy.Enabled = true;

WhereCanBuyCost.Enabled = true;

}

else

{

WhereCanBuy.Enabled = false;

WhereCanBuyCost.Enabled = false;

}

}

//private void WhereCanBuy\_Click(object sender, EventArgs e)

//{

// List<Carservice> carservicess = carservicesService.WhereCanBuy(ordersBox.Text);

// String result = "Результат: ";

// foreach (var pharmasy in carservicess)

// {

// result += pharmasy.Name + " " + pharmasy.Address + "\n";

// }

// if (result != "Результат: ")

// {

// MessageBox.Show(result);

// }

// else

// {

// MessageBox.Show("Ничего не найдено");

// }

//}

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

{

buttonEnabled();

}

private void ordersBox\_KeyPress(object sender, KeyPressEventArgs e)

{

e.KeyChar = '\0';

}

//private void WhereCanBuyCost\_Click(object sender, EventArgs e)

//{

// List<Carservice> carservicess = carservicesService.WhereCanBuyCost(ordersBox.Text);

// String result = "Результат(по возрастанию цены):\n";

// foreach (var pharmasy in carservicess)

// {

// result += pharmasy.Name + " " + pharmasy.Address + "\n";

// }

// if (result != "Результат: ")

// {

// MessageBox.Show(result);

// }

// else

// {

// MessageBox.Show("Ничего не найдено");

// }

//}

private void StartForm\_FormClosed(object sender, FormClosedEventArgs e)

{

Login.Session.FormLogin.Visible = true;

}

private void carservicesBox\_KeyPress(object sender, KeyPressEventArgs e)

{

e.KeyChar = '\0';

}

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

{

if (carservicesBox.Text.Length > 0)

{

PostingThisMonth.Enabled = true;

}

else

{

PostingThisMonth.Enabled = false;

}

if (carservicesBox.Text.Length > 0 && avtoBox.Text.Length > 0)

{

avtoButton.Enabled = true;

}

else

{

avtoButton.Enabled = false;

}

}

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

{

if (carservicesBox.Text.Length > 0 && avtoBox.Text.Length > 0)

{

avtoButton.Enabled = true;

}

else

{

avtoButton.Enabled = false;

}

}

private void avtoBox\_KeyPress(object sender, KeyPressEventArgs e)

{

e.KeyChar = '\0';

}

}

}

StartForm.cs

using carservice.model;

using carservice.service.orders;

using carservice.service.carservices;

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 carservice.controller.boyer

{

public partial class StartForm : Form

{

OrdersServiceImpl ordersService = new OrdersServiceImpl();

CarserviceServiceImpl carservicesService = new CarserviceServiceImpl();

public StartForm()

{

InitializeComponent();

var allOrderss = ordersService.getAll();

wellcome.Text = "Добро пожаловать, " + Login.Session.CurrentUser.UserName + "!";

allOrderss = ordersService.getAll();

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.MaximizeBox = false;

this.MinimizeBox = false;

WhereCanBuy.Enabled = false;

WhereCanBuyCost.Enabled = false;

}

private void buttonEnabled()

{

if(ordersBox.Text.Length > 0)

{

WhereCanBuy.Enabled = true;

WhereCanBuyCost.Enabled = true;

}

else

{

WhereCanBuy.Enabled = false;

WhereCanBuyCost.Enabled = false;

}

}

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

{

buttonEnabled();

}

private void ordersBox\_KeyPress(object sender, KeyPressEventArgs e)

{

e.KeyChar = '\0';

}

private void StartForm\_FormClosed(object sender, FormClosedEventArgs e)

{

Login.Session.FormLogin.Visible = true;

}

}

}

Login.cs

using carservice.config;

using carservice.controller;

using carservice.controller.admin;

using carservice.enums;

using carservice.model;

using carservice.service;

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 carservice

{

public partial class Login : Form

{

private static Session session = new Session(false);

private UserServiseImpl userServise = new UserServiseImpl();

internal static Session Session { get => session; set => session = value; }

public Login()

{

InitializeComponent();

Loging.Enabled = false;

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.MaximizeBox = false;

this.MinimizeBox = false;

}

private void LoginEnable()

{

if (userNameField.Text.Length > 0 && passwordField.Text.Length > 0)

{

Loging.Enabled = true;

}

else

{

Loging.Enabled = false;

}

}

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

{

User user = new User(userNameField.Text, passwordField.Text);

User existingUser = userServise.getUser(userNameField.Text);

if (userServise.validate(existingUser, user))

{

Session.setSession(existingUser, this);

}

else

{

MessageBox.Show("Неверное имя пользователя или пароль!");

userNameField.Text = "";

passwordField.Text = "";

return;

}

controller.admin.StartForm startForm = new controller.admin.StartForm();

startForm.Show();

}

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

{

RegistrationForm registration = new RegistrationForm();

registration.Show();

Session.FormLogin = this;

this.Visible = false;

}

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

{

LoginEnable();

}

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

{

LoginEnable();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

}

}

RegistrationForm.cs

using carservice.config;

using carservice.enums;

using carservice.model;

using carservice.service;

using carservice.service.carservices;

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 carservice.controller

{

public partial class RegistrationForm : Form

{

UserServiseImpl userServise = new UserServiseImpl();

public RegistrationForm()

{

InitializeComponent();

registration.Enabled = false;

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.MaximizeBox = false;

this.MinimizeBox = false;

}

private void registrationEnable()

{

if (userNameField.Text.Length > 0 && passwordField.Text.Length > 0)

{

registration.Enabled = true;

}

else

{

registration.Enabled = false;

}

}

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

{

registrationEnable();

}

private void roleField\_KeyPress(object sender, KeyPressEventArgs e)

{

e.KeyChar = '\0';

}

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

{

registrationEnable();

}

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

{

registrationEnable();

}

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

{

User user = new User();

user.UserName = userNameField.Text;

user.Password = passwordField.Text;

if (userServise.saveUser(user))

{

Close();

}

else

{

MessageBox.Show("Пользователь с таким именем уже существует!");

}

}

private void RegistrationForm\_FormClosed(object sender, FormClosedEventArgs e)

{

Login.Session.FormLogin.Visible = true;

}

private void label4\_Click(object sender, EventArgs e)

{

}

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

{

}

private void label3\_Click(object sender, EventArgs e)

{

}

}

}

TypesOfBody.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace carservice.enums

{

public enum TypesOfBody

{

Primary,

Secondary,

group,

Tertiary

}

}

Avto.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace carservice.model

{

public class Avto

{

private int id;

private String number;

private String brand;

private String body;

Client client;

private int clientId;

public int Id { get => id; set => id = value; }

public string Body { get => body; set => body = value; }

public string Brand { get => brand; set => brand = value; }

public string Number { get => number; set => number = value; }

internal Client Client { get => client; set => client = value; }

public int ClientId { get => clientId; set => clientId = value; }

}

}

Carservice.cs

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace carservice.model

{

public class Carservice

{

private int id;

private int idorder;

private int idworker;

private Worker worker;

private List<Order> orders;

private String uzel;

public int Id { get => id; set => id = value; }

public int Idworker { get => idworker; set => idworker= value; }

internal Worker Worker { get => worker; set => worker = value; }

internal List<Order> Orders { get => orders; set => orders = value; }

public String Uzel { get => uzel; set => uzel = value; }

}

}

Client.cs

using carservice.enums;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace carservice.model

{

public class Client

{

private int id;

private String name;

private String telephone;

private String avtoNumber;

List<Order> orders;

private String avtoName;

internal List<Order> Orders { get => orders; set => orders = value; }

public String AvtoName { get => avtoName; set => avtoName = value; }

public String AvtoNumber { get => avtoNumber; set => avtoNumber = value; }

public int Id { get => id; set => id = value; }

public String Name { get => name; set => name = value; }

public String Telephone { get => telephone; set => telephone = value; }

}

}

Order.cs

using carservice.enums;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace carservice.model

{

public class Order

{

private int id;

private DateTime dateGet;

private DateTime dateDelivery;

private int clientId;

private Client client;

private int cost;

private int carServiceId;

private Carservice carService;

public int Id { get => id; set => id = value; }

public int Cost { get => cost; set => cost = value; }

public DateTime DateGet { get => dateGet; set => dateGet = value; }

public DateTime DateDelivery { get => dateDelivery; set => dateDelivery = value; }

internal Client Client { get => client; set => client = value; }

public int ClientId { get => clientId; set => clientId = value; }

public int CarServiceId { get => carServiceId; set => carServiceId = value; }

internal Carservice CarService { get => carService; set => carService = value; }

}

}

Term.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace carservice.model

{

public class Term

{

private int id;

private int dateofreceipt;

private int deliverydate;

private int orderId;

private Order order;

public int Id { get => id; set => id = value; }

public int Dateofreceipt { get => dateofreceipt; set => dateofreceipt = value; }

public int Deliverydate { get => deliverydate; set => deliverydate = value; }

public int OrderId { get => orderId; set => orderId = value; }

internal Order Order { get => order; set => order = value; }

}

}

User.cs

using carservice.enums;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace carservice.model

{

public class User

{

private int id;

private String userName;

private String password;

public User()

{

}

public User(string userName, string password)

{

this.userName = userName;

this.password = password;

}

public int Id { get => id; set => id = value; }

public string UserName { get => userName; set => userName = value; }

public string Password { get => password; set => password = value; }

}

}

Worker.cs

using carservice.model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace carservice.model

{

public class Worker

{

private int id;

private String name;

private int experience;

private String jobtitle;

private List<Carservice> carservice;

public int Id { get => id; set => id = value; }

public String Name { get => name; set => name = value; }

public int Experience { get => experience; set => experience = value; }

public String Jobtitle { get => jobtitle; set => jobtitle = value; }

public List<Carservice> Carservice { get; set; }

}

}

CarserviceOrdersRepository.cs

using carservice.config;

using carservice.model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace carservice.repository

{

internal class CarserviceOrdersRepository

{

MyApplicationContext db = new MyApplicationContext();

public void save(Worker carservicesOrders)

{

db.Workers.Add(carservicesOrders);

db.SaveChanges();

}

}

}

CarserviceRepository.cs

using Microsoft.EntityFrameworkCore;

using carservice.config;

using carservice.model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace carservice.repository

{

class CarserviceRepository

{

MyApplicationContext db = new MyApplicationContext();

public void save(Carservice carservice)

{

db.Carservice.Add(carservice);

db.SaveChanges();

}

public void update(Carservice carservice)

{

db.Carservice.Update(carservice);

db.SaveChanges();

}

public Carservice findCarserviceById(int id)

{

return db.Carservice.Include(c => c.Worker).Where(p => p.Id == id).First();

}

public void delete(Carservice carservice)

{

db.Carservice.Remove(carservice);

db.SaveChanges();

}

public List<Carservice> FindAllCarservice()

{

return db.Carservice.Include(c => c.Orders).ToList();

}

}

}

ClientRepository.cs

using carservice.config;

using carservice.model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace carservice.repository

{

internal class ClientRepository

{

MyApplicationContext db = new MyApplicationContext();

public void save(Client client)

{

db.Clients.Add(client);

db.SaveChanges();

}

public void update(Client client)

{

db.Clients.Update(client);

db.SaveChanges();

}

public List<Client> findClientByNumber(String number)

{

return (from client in db.Clients

where number.Equals(client.AvtoNumber)

select client).ToList();

}

public Client findClientByName(String name)

{

try

{

return db.Clients.Where(p => p.Name == name).First();

}

catch

{

return null;

}

}

public Client findClientById(int id)

{

return db.Clients.Where(p => p.Id == id).First();

}

public void delete(Client client)

{

db.Clients.Remove(client);

db.SaveChanges();

}

public List<Client> findAllClients()

{

return db.Clients.ToList();

}

}

}

OrdersRepository.cs

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata.Conventions.Internal;

using carservice.config;

using carservice.model;

using carservice.service.carservices;

using Remotion.Linq.Clauses.ExpressionVisitors;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using static System.Collections.Specialized.BitVector32;

using carservice.service.carserviceservice;

namespace carservice.repository

{

class OrdersRepository

{

MyApplicationContext db = new MyApplicationContext();

CarserviceServiceImpl carserviceService = new CarserviceServiceImpl();

public void save(Order order)

{

db.Orders.Add(order);

db.SaveChanges();

}

public void update(Order order)

{

db.Orders.Update(order);

db.SaveChanges();

}

public void CheckLinks(Order order)

{

List<Order> allOrders = findAllOrders();

List<Carservice> allCarservices = carserviceService.getAll();

if (allOrders.Count > 0)

{

foreach (var item in allCarservices)

{

}

}

}

public Order findOrderById(int id)

{

return db.Orders.Include(o => o.Client).Include(o => o.CarService).Where(p => p.Id == id).First();

}

public void delete(Order order)

{

db.Orders.Remove(order);

db.SaveChanges();

}

public List<Order> findByDate(DateTime date)

{

return (from order in db.Orders

where date.Date == order.DateGet.Date

select order).ToList();

}

public List<Order> findByRangeDate(DateTime date1, DateTime date2)

{

return (from order in db.Orders

where date1.Date <= order.DateGet.Date && order.DateGet.Date <= date2

select order).ToList();

}

public List<Order> findAllOrders()

{

try

{

return db.Orders.Include(c => c.Client).ToList();

} catch { return null; }

}

public List<Order> findFilteredOrders(String name, String number, int id, DateTime date, DateTime date1, DateTime date2)

{

return (from order in db.Orders

where (name != "" ? order.Client.Name.Equals(name) : true) && (number != "" ? order.Client.AvtoNumber.Equals(number) : true) && (id != 228 ? order.Id == id : true) && ((date1.Date <= order.DateGet.Date && order.DateGet.Date <= date2) || (date.Date == order.DateGet.Date))

select order).Distinct().ToList();

}

}

}

UserRepository.cs

using carservice.config;

using carservice.model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace carservice.repository

{

class UserRepository

{

MyApplicationContext db = new MyApplicationContext();

public User findUserByUserName(String UserName)

{

try

{

return db.Users.Where(p => p.UserName == UserName).ToList().First();

}

catch (Exception e)

{

return null;

}

}

public void save(User user)

{

db.Users.Add(user);

db.SaveChanges();

}

}

}

WorkerRepository.cs

using carservice.config;

using carservice.model;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace carservice.repository

{

internal class WorkerRepository

{

MyApplicationContext db = new MyApplicationContext();

public void save(Worker worker)

{

db.Workers.Add(worker);

db.SaveChanges();

}

public void update(Worker worker)

{

db.Workers.Update(worker);

db.SaveChanges();

}

public Worker findWorkerByName(String name)

{

try

{

return db.Workers.Include(c => c.Carservice).Where(p => p.Name == name).First();

}

catch

{

return null;

}

}

public Worker findWorkerById(int id)

{

return db.Workers.Where(p => p.Id == id).First();

}

public void delete(Worker worker)

{

db.Workers.Remove(worker);

db.SaveChanges();

}

public List<Worker> findAllWorkers()

{

return db.Workers.ToList();

}

}

}

CarserviceOrdersService.cs

using carservice.model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace carservice.service.carservicesOrders

{

internal interface CarserviceOrdersService

{

bool saveCarservicesOrders(Worker carservicesOrders);

}

}

CarserviceOrdersServiceImpl.cs

using carservice.model;

using carservice.repository;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace carservice.service.carservicesOrders

{

internal class CarserviceOrdersServiceImpl : CarserviceOrdersService

{

CarserviceOrdersRepository carservicesOrdersRepository = new CarserviceOrdersRepository();

public bool saveCarservicesOrders(Worker carservicesOrders)

{

carservicesOrdersRepository.save(carservicesOrders);

return true;

}

}

}

CarserviceService.cs

using carservice.model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace carservice.service.carserviceservice

{

interface CarserviceService

{

bool save(Carservice carservice);

List<Carservice> getAll();

bool delete(int id);

}

}

CarserviceServiceImpl.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Linq.Expressions;

using System.Text;

using System.Threading.Tasks;

using carservice.model;

using carservice.repository;

using carservice.service.carserviceservice;

namespace carservice.service.carservices

{

class CarserviceServiceImpl : CarserviceService

{

CarserviceRepository carservicesRepository = new CarserviceRepository();

public bool save(Carservice carservices)

{

carservicesRepository.save(carservices);

return true;

}

public bool update(Carservice carservices)

{

try {

carservicesRepository.update(carservices);

return true;

}

catch

{ return false; }

}

public List<Carservice> getAll()

{

return carservicesRepository.FindAllCarservice();

}

public Carservice getById(int id)

{

return carservicesRepository.findCarserviceById(id);

}

public bool delete(int id)

{

//try

//{

carservicesRepository.delete(carservicesRepository.findCarserviceById(id));

return true;

//}

//catch

//{

// return false;

//}

}

}

}

ClientService.cs

using carservice.model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace carservice.service.client

{

interface ClientService

{

bool save(Client client);

List<Client> getAll();

Client getByName(String Name);

Client GetById(int id);

bool delete(int id);

List<Client> getByAvtoNumber(string number);

}

}

ClientServiceImpl.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using carservice.model;

using carservice.repository;

namespace carservice.service.client

{

class ClientServiceImpl : ClientService

{

ClientRepository clientRepository = new ClientRepository();

public bool delete(int id)

{

try

{

clientRepository.delete(clientRepository.findClientById(id));

return true;

}

catch

{

return false;

}

}

public List<Client> getByAvtoNumber(string number)

{

return clientRepository.findClientByNumber(number);

}

public List<Client> getAll()

{

return clientRepository.findAllClients();

}

public Client getByName(String Name)

{

return clientRepository.findClientByName(Name);

}

public bool save(Client client)

{

clientRepository.save(client);

return true;

}

public bool update(Client client)

{

clientRepository.update(client);

return true;

}

public Client GetById(int id)

{

return clientRepository.findClientById(id);

}

}

}

OrdersService.cs

using carservice.model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace carservice.service.orders

{

interface OrdersService

{

bool saveOrders(Order orders);

List<Order> getAll();

bool deleteOrders(int id);

List<Order> findByDate(DateTime date);

List<Order> findByRangeDate(DateTime date1, DateTime date2);

List<Order> findFilteredOrders(String name, String number, int id, DateTime date, DateTime date1, DateTime date2);

}

}

OrdersServiceImpl.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using carservice.model;

using carservice.repository;

namespace carservice.service.orders

{

class OrdersServiceImpl : OrdersService

{

OrdersRepository ordersRepository = new OrdersRepository();

public bool saveOrders(Order orders)

{

ordersRepository.save(orders);

return true;

}

public bool update(Order orders)

{

ordersRepository.update(orders);

return true;

}

public List<Order> getAll()

{

return ordersRepository.findAllOrders();

}

public Order getById(int Id)

{

return ordersRepository.findOrderById(Id);

}

public List<Order> findByRangeDate(DateTime date1, DateTime date2)

{

return ordersRepository.findByRangeDate(date1, date2);

}

public List<Order> findByDate(DateTime date)

{

return ordersRepository.findByDate(date);

}

public List<Order> findFilteredOrders(String name, String number, int id, DateTime date, DateTime date1, DateTime date2)

{

return ordersRepository.findFilteredOrders(name, number, id, date, date1, date2);

}

public bool deleteOrders(int id)

{

//try

//{

ordersRepository.delete(ordersRepository.findOrderById(id));

return true;

//}

//catch

//{

// return false;

//}

}

}

}

UserSerrvise.cs

using carservice.model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace carservice.service

{

interface UserSerrvise

{

bool validate(User existingUser, User user);

bool saveUser(User user);

User getUser(String userName);

}

}

UserServiseImpl.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using carservice.model;

using carservice.repository;

namespace carservice.service

{

class UserServiseImpl : UserSerrvise

{

UserRepository userRepository = new UserRepository();

public bool saveUser(User user)

{

if (userRepository.findUserByUserName(user.UserName) != null)

{

return false;

}

userRepository.save(user);

return true;

}

public User getUser(String userName)

{

try

{

return userRepository.findUserByUserName(userName);

}

catch

{

return null;

}

}

public bool validate(User existingUser, User userForValidate)

{

if (existingUser != null && existingUser.Password == userForValidate.Password)

{

return true;

}

return false;

}

}

}

WorkerService.cs

using carservice.model;

using carservice.repository;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace carservice.service.worker

{

interface WorkerService

{

bool save(Worker worker);

List<Worker> getAll();

bool delete(int id);

Worker getWorkerByNumber(String name);

Worker findWorkerById(int id);

Worker GetByName(String Name);

}

}

WorkerServiceImpl.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using carservice.model;

using carservice.repository;

namespace carservice.service.worker

{

class WorkerServiceImpl : WorkerService

{

WorkerRepository workerRepository = new WorkerRepository();

public bool save(Worker worker)

{

workerRepository.save(worker);

return true;

}

public bool update(Worker worker)

{

workerRepository.update(worker);

return true;

}

public List<Worker> getAll()

{

return workerRepository.findAllWorkers();

}

public bool delete(int id)

{

workerRepository.delete(workerRepository.findWorkerById(id));

return true;

}

public Worker getWorkerByNumber(string name)

{

return workerRepository.findWorkerByName(name);

}

public Worker findWorkerById(int id)

{

return workerRepository.findWorkerById(id);

}

public Worker GetByName(String Name)

{

return workerRepository.findWorkerByName(Name);

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace carservice

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Login());

}

}

}