|  |  |
| --- | --- |
|  | **Технически университет - София**  **Филиал Пловдив** |

**Дипломна работа**

|  |  |  |
| --- | --- | --- |
| Тема: | ***Информационна система за събиране и обработка на данни от сервизно обслужване на пътна и извън пътна техника*** | |
|  | |  | |
| Студент**:** | | ***Здравко Господинов Иванов*** | |
| Фак №: | | ***370918*** | |
| Специалност: | | ***КСТ*** | |
| Образователнастепен: | | ***БАКАЛАВЪР*** | |
| Факултет: | | ***ФЕА*** | |
| Ръководител: | | ***инж. Николай Николов*** | |

ТУ – София, Филиал Пловдив, 2021 г.

СЪДЪРЖАНИЕ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Увод** | …………………………………………………………… | | | | 6 |
| **Глава 1.** | Структура на постъпващата информация…………….. | | | | 8 |
| **Глава 2.** | Проектиране на структурата на базата данни............... | | | | 10 |
| **2.1.** | | Съставяне на ER модел на базата данни……………. | | | 10 |
| **2.1.1.** | | | Описание на entity типовете……………………... | | 10 |
| **2.1.2.** | | | Описание на типовете връзки……………………. | | 14 |
| **2.1.3.** | | | Обобщение на типове същности и връзки в таблица…………………………………………….. | | 15 |
| **2.1.4.** | | | Обобщен ER модел на предметната област…….. | | 16 |
| **2.2.** | | Описание на релационният модел на базата данни.... | | | 16 |
| **2.2.1.** | | | Таблици с имена на таблици, колони, типове данни иописание………………………………….. | | 17 |
| **2.2.2.** | | | Релационни таблици и връзките между тях | | 29 |
| **Глава 3.** | Обосновка за избор на машина за база данни, език за програмиране и програмна среда……………………... | | | | 33 |
| **3.1.** | | Обосновка за избор на машина за бази данни…….... | | | 33 |
| **3.2.** | | Обосновка за избор на език за програмиране……..... | | | 34 |
| **3.3.** | | Обосновка за избор на програмна среда……………. | | | 34 |
| **Глава 4.** | Описание на основните програмни модули и на взаимодействието между тях………………………….. | | | | 35 |
| **4.1.** | | Описание на форма модул за данни DataModuleUnit. | | | 35 |
| **4.1.1.** | | | Цели логика на формата ………………................. | | 35 |
| **4.1.2.** | | | Съставни части на формата ……………............... | | 37 |
| **4.1.3.** | | | Събития във формата……………………………... | | 38 |
| **4.2.** | | Описание на главна форма MainForm.......................... | | | 38 |
| **4.2.1.** | | | Цел и логика на формата…………………………. | | 38 |
| **4.2.2.** | | | Съставни части на формата……………………… | | 39 |
| **4.2.3.** | | | Събития във формата……………………………... | | 40 |
| **4.3.** | | Описание на форма за добавяне и редактиране на пътна или извънпътна техника EditVehicleForm ….. | | | 40 |
| **4.3.1.** | | | Цел и логика на формата…………………………. | | 41 |
| **4.3.2.** | | | Съставни части на формата……………………… | | 41 |
| **4.3.3.** | | | Събития във формата……………………………... | | 42 |
| **4.4.** | | Описание на форма за добавяне и редактиране на ремонти EditPartForm …………................................... | | | 42 |
| **4.4.1.** | | | Цел и логика на формата…………………………. | | 42 |
| **4.4.2.** | | | Съставни части на формата……………………… | | 43 |
| **4.4.3.** | | | Събития във формата……………………………... | | 44 |
| **4.5.** | | Описание на форма за анализ AnalysisForm …........... | | | 44 |
| **4.5.1.** | | | Цел и логика на формата…………………………. | | 44 |
| **4.5.2.** | | | Съставни части на формата……………………… | | 45 |
| **4.5.3.** | | | Събития във формата……………………………... | | 45 |
| **4.6.** | | Описание на форма за справки Reference.................... | | | 47 |
| **4.6.1.** | | | Цел и логика на формата…………………………. | | 47 |
| **4.6.2.** | | | Съставни части на формата……………………… | | 47 |
| **4.6.3.** | | | Събития във формата……………………………... | | 48 |
| **4.7.** | | Описание на форма за ауторизация LogInForm ……. | | | 48 |
| **4.7.1.** | | | Цел и логика на формата…………………………. | | 48 |
| **4.7.2.** | | | Съставни части на формата……………………… | | 49 |
| **4.7.3.** | | | Събития във формата……………………………... | | 49 |
| **4.8.** | | Описание на форма за създаване на акаунт CreateAccForm................................................................ | | | 49 |
| **4.8.1.** | | | Цел и логика на формата…………………………. | | 49 |
| **4.8.2.** | | | Съставни части на формата……………………… | | 50 |
| **4.8.3.** | | | Събития във формата……………………………... | | 50 |
| **4.9.** | | Описание на форма за редактиране на номенклатура на пътна или извънпътна техника VehicleNomForm.. | | | 51 |
| **4.9.1.** | | | Цел и логика на формата…………………………. | | 51 |
| **4.9.2.** | | | Съставни части на формата……………………… | | 51 |
| **4.9.3.** | | | Събития във формата……………………………... | | 52 |
| **4.10.** | | Описание на форма за редактиране на собственик на пътна или извънпътна техника OwnerForm …....... | | | 53 |
| **4.10.1.** | | | Цел и логика на формата…………………………. | | 53 |
| **4.10.2.** | | | Съставни части на формата……………………… | | 53 |
| **4.10.3.** | | | Събития във формата……………………………... | | 54 |
| **4.11.** | | Описание на форма за редактиране на служители EmployeeForm ……........................................................ | | | 55 |
| **4.11.1.** | | | Цел и логика на формата…………………………. | | 55 |
| **4.11.2.** | | | Съставни части на формата……………………… | | 55 |
| **4.11.3.** | | | Събития във формата……………………………... | | 56 |
| **4.12.** | | Описание на форма за редактиране на сервизи RepairSForm ……........................................................... | | | 56 |
| **4.12.1.** | | | Цел и логика на формата…………………………. | | 56 |
| **4.12.2.** | | | Съставни части на формата……………………… | | 57 |
| **4.12.3.** | | | Събития във формата……………………………... | | 57 |
| **4.13.** | | Описание на форма за редактиране на артикули ItemForm ……................................................................. | | | 58 |
| **4.13.1.** | | | Цел и логика на формата…………………………. | | 58 |
| **4.13.2.** | | | Съставни части на формата……………………… | | 59 |
| **4.13.3.** | | | Събития във формата……………………………... | | 59 |
| **4.14.** | | Описание на форма за редактиране на служители EmployeeForm ……........................................................ | | | 60 |
| **4.14.1.** | | | Цел и логика на формата…………………………. | | 60 |
| **4.14.2.** | | | Съставни части на формата……………………… | | 60 |
| **4.14.3.** | | | Събития във формата……………………………... | | 61 |
| **4.15.** | | Описание на форма за редактиране на позиции заемани от служители PossitionForm …….................. | | | 61 |
| **4.15.1.** | | | Цел и логика на формата…………………………. | | 61 |
| **4.15.2.** | | | Съставни части на формата……………………… | | 62 |
| **4.15.3.** | | | Събития във формата……………………………... | | 62 |
| **4.16.** | | Описание на форма за редактиране на производители ManufacturerForm ……........................ | | | 63 |
| **4.16.1.** | | | Цел и логика на формата…………………………. | | 63 |
| **4.16.2.** | | | Съставни части на формата……………………… | | 63 |
| **4.16.3.** | | | Събития във формата……………………………... | | 64 |
| **4.17.** | | Описание на форма за редактиране на мярки MeasureForm …….......................................................... | | | 64 |
| **4.17.1.** | | | Цел и логика на формата…………………………. | | 64 |
| **4.17.2.** | | | Съставни части на формата……………………… | | 65 |
| **4.17.3.** | | | Събития във формата……………………………... | | 65 |
| **4.18.** | | Описание на форма за редактиране на марки BrandForm …….............................................................. | | | 65 |
| **4.18.1.** | | | Цел и логика на формата…………………………. | | 65 |
| **4.18.2.** | | | Съставни части на формата……………………… | | 66 |
| **4.18.3.** | | | Събития във формата……………………………... | | 66 |
| **4.19.** | | Описание на форма за редактиране на категории CategoryForm ……......................................................... | | | 67 |
| **4.19.1.** | | | Цел и логика на формата…………………………. | | 67 |
| **4.19.2.** | | | Съставни части на формата……………………… | | 67 |
| **4.19.3.** | | | Събития във формата……………………………... | | 68 |
| **4.20.** | | Описание на форма за редактиране на двигатели EngineForm ……............................................................. | | | 68 |
| **4.20.1.** | | | Цел и логика на формата…………………………. | | 68 |
| **4.20.2.** | | | Съставни части на формата……………………… | | 69 |
| **4.20.3.** | | | Събития във формата……………………………... | | 69 |
| **4.21.** | | Описание на форма за редактиране на видове пътна и извънпътна техника TypeForm ……........................ | | | 69 |
| **4.21.1.** | | | Цел и логика на формата…………………………. | | 69 |
| **4.21.2.** | | | Съставни части на формата……………………… | | 70 |
| **4.21.3.** | | | Събития във формата……………………………... | | 70 |
| **Глава 5.** | Ръководство за потребителя…………………………... | | | | 71 |
| **5.1.** | | Вписване в приложението…......................................... | | | 71 |
| **5.2.** | | Добавяне, изтриване и редактиране на пътна или извънпътна техника....................................................... | | | 71 |
| **5.3.** | | Добавяне на пътна или извънпътна техника …......... | | | 72 |
| **5.4.** | | Редактиране на пътна или извънпътна техника.......... | | | 73 |
| **5.5.** | | Изтриване на пътна или извънпътна техника …….... | | | 74 |
| **5.6.** | | Добавяне, изтриване и редактиране на ремонт ……. | | | 74 |
| **5.7.** | | Съставяне на справка..................................................... | | | 74 |
| **5.8.** | | Анализ………………………………………………..... | | | 75 |
| **5.9.** | | Създаване на акаунт .…................................................. | | | 75 |
| **5.10.** | | Таблици за редактиране ……………….......……….... | | | 75 |
| **Използвана литература** | | | | ……………………………………………... | 77 |
| **Приложение** | Структура на базата данни ………………..................... | | | | 78 |
| **Приложение** | SQL скрипт на базата данни…………………………... | | | | 79 |
| **Приложение** | Сорс код на програмното осигуряване……………….. | | | | 86 |

**Увод**

За създаването на информационна управляваща система, включваща в себе си база от данни, е необходимо да се премине през определени етапи. Първо проектирането на база данни, а след това и програмно осигуряване, което ще работи с нея. За създаването на една такава база е необходимо следването на определени процедури, които да анализират и оценят нуждите на програмното осигуряване, кaто след това, ще отговаря на нуждите на потребителите, които ще работят с него. Всяко приложение представлява отделен проблем и се налага да бъде изследван поради вариацията от нужди на потребителите, както и разликите в средите на приложение на програмното осигуряване.

Целта на конкретната дипломна работа е да се разработи приложение, което представлява информационна управляваща система за обслужване на дейността по събиране и обработка на данни от сервизно обслужване на пътна и извън пътна техника. Има се за цел да подпомогне работата на сервизи, лизингодатели или автопаркове, като те могат да се възползват от събраната информация да планират в полза на техния бизнес. Етапите за разработване на приложното програмно осигуряване са:

1. Анализ на изисквания към базата данни спрямо специфичната предметна област
2. Функционален анализ и концептуален дизайн на базата данни – създава се ER модел и по-късно се използват имплементираните в самото приложение на С++ Builder трансакции.
3. Логически дизайн – концептуалния ER модел се надгражда в релационен, който от своя страна е зависим от конкретната машина за бази данни, в този случай се използва InterBase Serverv6.
4. Физически дизайн – определяне на организацията на всички физически параметри.

Приложение е от тип клиент-сървър, което може да се използва като Desktop или Intranet приложение, работещо под ОС MS Windows. Целта на това програмно осигуряване е да опише предметна област, като всичките необходими нейни характеристики се пригодят за съхраняване в база данни. Тези данни ще бъдат използвани и манипулирани по улеснен за потребителите начин.

Програмно осигуряване предлага и запис на направената справка в файл с формат CSV. По този начин справките могат да се отпечатват за по-лесно използване на данните според целите на потребителите, като например следене на най-често повреждана пътна и извънпътна техника, най-често използвани части и т.н.

**Глава 1**

**Структура на постъпващата информация**

Структурата на постъпващата информация се определя от конкретната област, в която се прилага приложението и базата данни. В този случай постъпващата информация представлява постъпващи данни от различни сервизи за обслужване. Това включва:

* Вписване на нова пътна или извънпътна техника
* Вписване на извършваното върху тях сервизно обслужване
* Вписване на извършените услуги и необходими артикули
* Извеждане на списъци с вписаните автомобили и артикули
* Извеждане на кратка аналитична информация

За да може приложението да обхване тези дейности, в базата от данни се пази информация за:

* Пътна и извън пътна техника и техните собственици
* Ремонти заедно с извършените услуги
* Артикули и производителите им
* Сервизи и работници

На базата на наличната информация могат да се правят справки за пътна или извънпътна техника или артикули:

* за определен период от време
* за конкретна определена пътна или извънпътна техника
* за конкретна марка пътна или извън пътна техника
* за конкретно място на провеждане на ремонта

Също така може да се извежда аналитична информация:

* за брой на поправки извършени върху конкретната марка
* за цена на ремонти на конкретна марка
* най-често извършван ремонт

**Глава 2**

**Проектиране на структурата на базата данни**

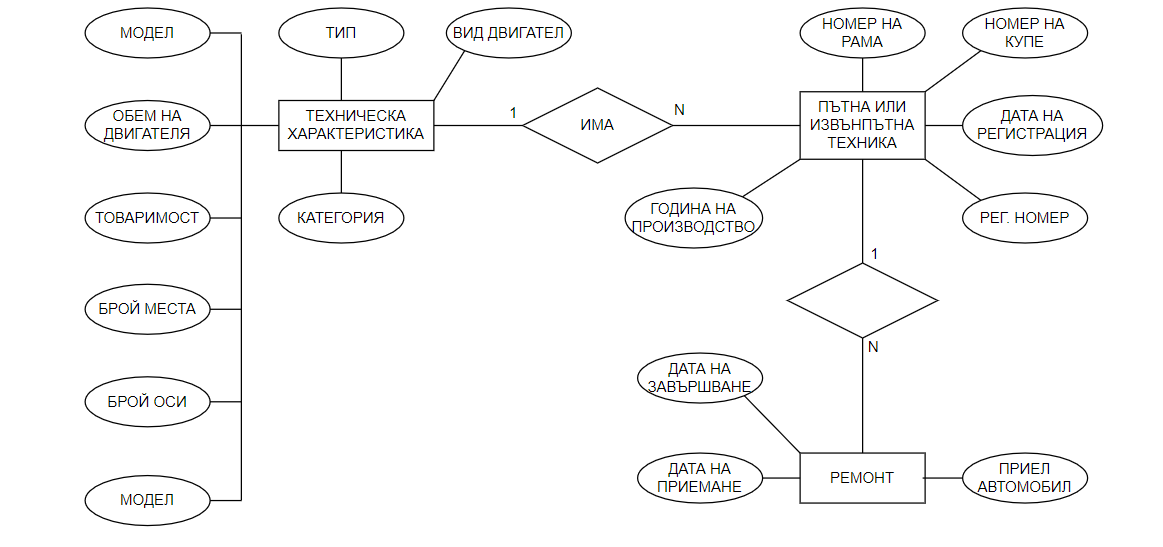
**2.1. Съставяне на ER модел на базата данни**

Първият етап в създаването на всяка една база от данни е съставянето на Entity Relationship (ER) модел. Определят се същности (enitity) на предметната област на приложение и смисловите връзките между тях (relations). Връзките от своя страна са ограничени от структурни ограничения, представляващи броя на същностите, които могат да имат смислова връзка помежду си.

**2.1.1. Описание на типовете същности (entity types)**

За тази предметна област на приложение се състои от пътна и извънпътна техника и техните ремонти извършвани в различни сервизи от различни работници. Ще се разгледа всяко отделно енити по-подробно.

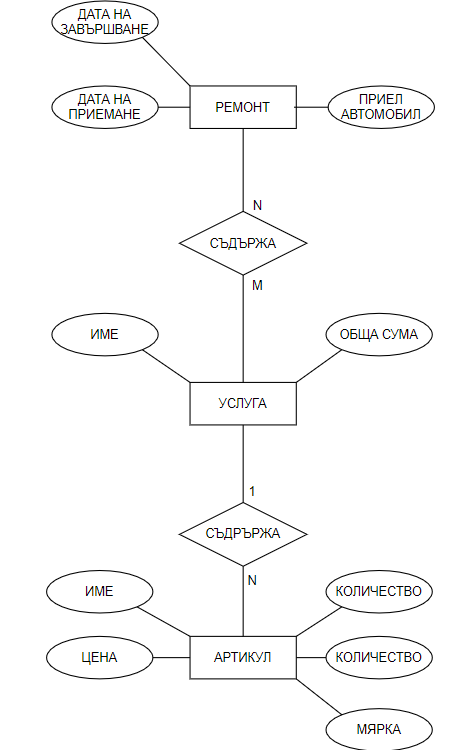
* Entity тип „Пътна или извънпътна техника“ има атрибути:
  + Номер на рама
  + Номер на купе
  + Дата на регистрация
  + Регистрационен номер
  + Година на производство
* Пътната и извънпътна техника има множество технически характеристики, които са прекалено сложни за да бъдат представени като атрибут, затова са изнесени в отделна същност „Техническа характеристика“:
  + Модел
  + Обем на двигателя
  + Товаримост
  + Брой места
  + Брой оси
  + Тип
  + Вид двигател
  + Категория
* Entity „Ремонт“ има следните атрибути:
  + Дата на приемане
  + Дата на завършване
  + Приел автомобила



**Фигура 1**

На фиг. 1 е показана връзката между тези три същности заедно с техните атрибути. Същностите се изобразяват с правоъгълник, връзките с ромб, а атрибутите с елипса.

* Entity „Услуга“ има следните атрибути:
* Име
* Обща сума
* Entity „Артикул“ има следните атрибути:
* Цена
* Количество
* Име
* Мярка
* Производител
* Поради възможност от многократна повтаряемост в последствие е добре атрибутите „Мярка“ и „Производител“ да се представят като отделни същности.
* Entity “Мярка“ има следните атрибути:
* Име
* Entity “Производител“ има следните атрибути:
* Име



**Фигура 2**

На фиг. 2 е показана връзката между същностите „Услуга“ и „Артикул“ заедно с техните атрибути. Показана е също връзката им с същността „Ремонт“

* Entity „Служител“ има роля и на потребител на програмното осигуряване затова има атрибути, които представяват данни за вход в приложението. Има следните атрибути:
* Име
* Личен телефонен номер
* Служебен телефонен номер
* Имейл
* Позиция
* Потребителско име
* Парола
* Отново, поради възможност от многократна повтаряемост, е добре атрибута „Позиция“ да се изведе, като отделна същност с атрибути:
* Име
* Entity „Собственик“ не е задължително да бъде физическо лице и има следните атрибути:
* Име
* Телефонен номер
* Адрес
* Имейл
* МОЛ
* Булстат или ЕГН
* По подобие на същностите „Позиция“, „Мярка“ и „Производител“, част от атрибутите на същност „Техническа характеристика“ са изнесени, като отделни същности:
* Entity “Марка“ с атрибут „Име“
* Entity “Категория“ с атрибут „Име“
* Entity „Тип“ с атрибут „Име“
* Entity “Вид двигател“ с атрибут „Име“
* Entity “Сервиз“ има следните атрибути:
* Име
* Адрес

**2.1.2. Описание на типовете връзки (relationship types)**

Освен различните видове същности и техните атрибути ключова част от изграждането на ER модела са типовете връзки между тях.

На фигура 1 е показана връзката между пътна и извънпътна техника и ремонт. Тя е една от по-съществените връзки. Също така на връзката са показани ограниченията и. 1:N означава, че една пътна или извънпътна техника може да притежава множество различни ремонти.

На фигура 2 е показана друга важна връзка. Между „Ремонт“ и „Услуга“. Физически много ремоти могат да съдържат много услуги и това се се реализира чрез връзката N:M много към много. Този тип връзка налага при преминаването към релационен модел да се създаде допълнителен елемент за реализирането на тази връзка.

Обобщено представяне на смисловите връзки между отделните entity типове от предметната област:

* Всеки собственик може да притежава повече от една пътна или извънпътна техника.
* За една пътна или извънпътна техника се пазят данни само един за собственик.
* Една пътна или извънпътна техника може да има точно една техническа характеристика.
* Една техническа характеристика може да бъде на различни пътни или извънпътни техники.
* Всяка техническа характеристика има точно една марка.
* Една марка може да бъде използвана в различни технически характеристики.
* Всяка техническа характеристика има точно един тип.
* Един тип може да бъде използван в различни технически характеристики.
* Всяка техническа характеристика има точно една категория.
* Една категория може да бъде използвана в различни технически характеристики.
* Всяка техническа характеристика има точно един вид двигател.
* Един тип двигател може да бъде използван в различни технически характеристики.
* Един ремонт може да се съдържа в само една пътна или извън пътна техника
* Всяка пътна или извънпътна техника може да има много ремонти.
* Всеки ремонт има точно един сервиз.
* В един сервиз могат да бъдат направени много различни ремонти.
* Един служител може да извършва много ремонти.
* Един ремонт е извършен точно от един служител.
* Един служител може да изпълнява една длъжност.
* Една длъжност може да се изпълнява от много служители.

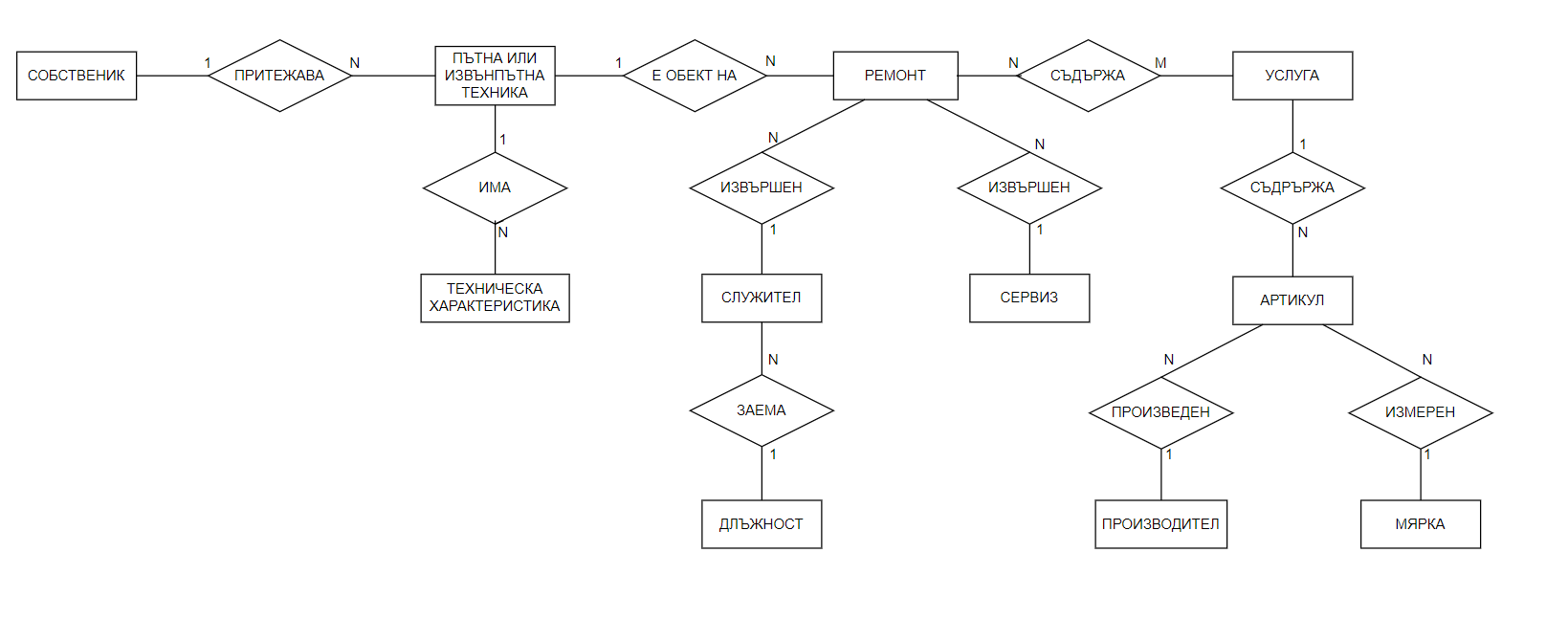
**2.1.3. Обобщение на типове същности и връзки в таблица**

В табл. 1 са посочени всички възможни връзки между entity типовете от предметната област на проложението.

|  |  |  |
| --- | --- | --- |
| Entity | Relationship | Entity |
| Собственик | Притежава | Пътна или извънпътна техника |
| Пътна или извънпътна техника | Има | Техническа характеристика |
| Техническа характеристика | Има | Модел |
| Техническа характеристика | Има | Категория |
| Техническа характеристика | Има | Тип |
| Техническа характеристика | Има | Вид двигател |
| Пътна или извънпътна техника | Съдържа | Ремонт |
| Ремонт | Извършил | Служител |
| Служител | Заема | Позиция |
| Ремонт | Извършен | Сервиз |
| Ремонт | Съдържа | Услуга |
| Услуга | Съдържа | Арикул |
| Артикул | Произведен | Производител |
| Артикул | Измерван | Мярка |

**Таблица 1**

**2.1.4. Обобщен ER модел на предметната област**



**Фигура 3**

Фиг. 3 представлява обобщеният ER модел на предметната област на приложението.

**2.2. Описание на релационният модел на базата данни**

За създаването на релационен модел се използва концептуалния ER модел и се преминава към релационния модел чрез различни правила и стъпки, като релационният модел е много близък по концепция до базата от данни, което го прави и зависим от конкретната машина за бази данни.

Стъпките включват превръщането на същностите в таблици, атрибутите им в колони на таблиците. Като се избира първичен ключ. Различните връзки се изразяват по различни начини, като например връзка 1:1 може да се осъществи чрез външен ключ, таблиците се комбинират или се използва нова таблица, както би се използвала при връзка от типа N:M. 1:N връзката се изразява, като от страна на N връзката се добаи външен ключ към първичния ключ на същността от страна на 1. Връзките М:N се осъществяват с помощта на допълнителна таблица.

Съществуват процедури, които дават представа за това, колко добър и ефективен е дизайна на релационния модел. Това са т.нар. нормални форми на базата данни. На практика, те са три, но в теорията съществуват още две. Те са:

* Първа НФ – данните във всички колони (атрибути) трябва да са атомични (неделими) т.е. да няма релация (таблица) в релация или многостойностен атрибут, който е вграден в самата таблица.
* Втора НФ – отнася се само за първични ключове, включващи повече от един атрибут. Не трябва да има неключови атрибути в таблицата, които да са функционално зависими от първичния ключ.
* Трета НФ (Boyce-Codd NF)–не трябва да има транзитивна зависимост между първичен ключ и неключов атрибут т.е. не трябва да има неключови атрибути, които са функционално определени от други неключови атрибути.

В това приложение са спазени и трите нормални форми на базата данни.

**2.2.1. Структура на релационните таблици**

|  |  |  |
| --- | --- | --- |
| **Таблица OWNER** | | |
| **Атрибути** | **Тип** | **Описание** |
| **ID** | INTEGER | Първичен ключ |
| **NAME** | VARCHAR(64) | Име на собственика |
| **PHONE\_NUMBER** | DECIMAL(12,0) | Телефонен номер на собственика |
| **ADDRESS** | VARCHAR(150) | Адрес на собственика |
| **E\_MAIL** | VARCHAR(64) | Имейл на собственика |
| **MOL** | VARCHAR(64) | Материално отговорно лице |
| **BUL\_EGN** | VARCHAR(12) | Булстат или единен граждански номер |

Тази таблица съдържа данни за всички собственици на пътна или извън пътна техника.

Тя кореспондира с Entity „Собственик“ от ER модела на базата данни.

Таблицата OWNER е приведена и в трите нормални форми, защото:

* Всички атрибути са прости/атомични т.е. няма съставни или многостойностни атрибути.
* Избран е само един атрибут за първичен ключ (ID), а не множество атрибути, което автоматично означава, че базата данни е приведена във втора нормална форма.
* Няма транзитивна зависимост между първичният ключ и неключов атрибут.

|  |  |  |
| --- | --- | --- |
| **Таблица VEHICLE** | | |
| **Атрибути** | **Тип** | **Описание** |
| **ID** | INTEGER | Първичен ключ |
| **PRODUCTION\_YEAR** | DATE | Година на производство на пътната или извън пътна техника |
| **OWNER\_ID** | INTEGER | Собственик (външен ключ) |
| **VEHICLE\_NOMENCLATURE\_ID** | INTEGER | Номенклатура на техниката (външен ключ) |
| **REGISTRATION\_NUMBER** | VARCHAR(16) | Регистрационен номер на техниката |
| **CHASSIS\_NUMBER** | VARCHAR(32) | Номер на купе |
| **FRAME\_NUMBER** | VARCHAR(32) | Номер на рама |
| **REGISTRATION\_DATE** | DATE | Дата на регистрация на техниката |

Тази таблица съдържа данни за всичка пътна или извънпътна техника.

Тя кореспондира с Entity „Пътна или извънпътна техника“ от ER модела на базата данни.

Таблицата VEHICLE е приведена и в трите нормални форми, защото:

* Всички атрибути са прости/атомични т.е. няма съставни или многостойностни атрибути.
* Избран е само един атрибут за първичен ключ (ID), а не множество атрибути, което автоматично означава, че базата данни е приведена във втора нормална форма.
* Няма транзитивна зависимост между първичният ключ и неключов атрибут.

|  |  |  |
| --- | --- | --- |
| **Таблица VEHICLE\_NOMENCLATURE** | | |
| **Атрибути** | **Тип** | **Описание** |
| **ID** | INTEGER | Първичен ключ |
| **BRAND\_ID** | INTEGER | Марка (външен ключ) |
| **MODEL** | VARCHAR(64) | Модел на техниката |
| **CATEGORY\_ID** | INTEGER | Категория (външен ключ) |
| **ENGINE\_TYPE\_ID** | INTEGER | Вид двигател (външен ключ) |
| **TYPE\_VEHICLE\_ID** | INTEGER | Вид на техниката (външен ключ) |
| **ENGINE\_VOLUME** | DECIMAL(4,1) | Обем на двигателя на техниката |
| **LOADABILITY** | VARCHAR(32) | Товаримост на техниката |
| **AXLES\_NUMBER** | INTEGER | Брой оси на техниката |
| **SEATS\_NUMBER** | INTEGER | Брой места на техниката |

Тази таблица служи като номенклатура и съдържа данни за параметрите на различните видове съществуваща пътна и извън пътна техника.

Тя кореспондира с Entity „Техническа характеристика“ от ER модела на базата данни.

Таблицата VEHICLE\_NOMENCLATURE е приведена и в трите нормални форми, защото:

* Всички атрибути са прости/атомични т.е. няма съставни или многостойностни атрибути.
* Избран е само един атрибут за първичен ключ (ID), а не множество атрибути, което автоматично означава, че базата данни е приведена във втора нормална форма.
* Няма транзитивна зависимост между първичният ключ и неключов атрибут.

|  |  |  |
| --- | --- | --- |
| **Таблица BRAND** | | |
| **Атрибути** | **Тип** | **Описание** |
| **ID** | INTEGER | Първичен ключ |
| **NAME** | VARCHAR(32) | Име на марката |

Тази таблица служи като номенклатура и съдържа данни за различните видове марки пътна и извънпътна техника.

Тя кореспондира с Entity „Марка“ от ER модела на базата данни.

Таблицата BRAND е приведена и в трите нормални форми, защото:

* Всички атрибути са прости/атомични т.е. няма съставни или многостойностни атрибути.
* Избран е само един атрибут за първичен ключ (ID), а не множество атрибути, което автоматично означава, че базата данни е приведена във втора нормална форма.
* Няма транзитивна зависимост между първичният ключ и неключов атрибут.

|  |  |  |
| --- | --- | --- |
| **Таблица CATEGORY** | | |
| **Атрибути** | **Тип** | **Описание** |
| **ID** | INTEGER | Първичен ключ |
| **NAME** | VARCHAR(32) | Име на категорията |

Тази таблица служи като номенклатура и съдържа данни за различните видове категории пътна и извънпътна техника. Например „B”, “C”, “T” и т.н.

Тя кореспондира с Entity „Категория“ от ER модела на базата данни.

Таблицата CATEGORY е приведена и в трите нормални форми, защото:

* Всички атрибути са прости/атомични т.е. няма съставни или многостойностни атрибути.
* Избран е само един атрибут за първичен ключ (ID), а не множество атрибути, което автоматично означава, че базата данни е приведена във втора нормална форма.
* Няма транзитивна зависимост между първичният ключ и неключов атрибут.

|  |  |  |
| --- | --- | --- |
| **Таблица TYPE** | | |
| **Атрибути** | **Тип** | **Описание** |
| **ID** | INTEGER | Първичен ключ |
| **NAME** | VARCHAR(32) | Име на вида техника |

Тази таблица служи като номенклатура и съдържа данни за различните типове пътна и извънпътна техника. Например „Комби”, “Cедан”, “Камион” и т.н.

Тя кореспондира с Entity „Тип“ от ER модела на базата данни.

Таблицата TYPE е приведена и в трите нормални форми, защото:

* Всички атрибути са прости/атомични т.е. няма съставни или многостойностни атрибути.
* Избран е само един атрибут за първичен ключ (ID), а не множество атрибути, което автоматично означава, че базата данни е приведена във втора нормална форма.
* Няма транзитивна зависимост между първичният ключ и неключов атрибут.

|  |  |  |
| --- | --- | --- |
| **Таблица ENGINE\_TYPE** | | |
| **Атрибути** | **Тип** | **Описание** |
| **ID** | INTEGER | Първичен ключ |
| **NAME** | VARCHAR(32) | Вид на двигателя |

Тази таблица служи като номенклатура и съдържа данни за различните видове двигатели според това с какво гориво работят. Например „Газ”, “Дизел”, “Бензин” и т.н.

Тя кореспондира с Entity „Вид двигател“ от ER модела на базата данни.

Таблицата ENGINE\_TYPE е приведена и в трите нормални форми, защото:

* Всички атрибути са прости/атомични т.е. няма съставни или многостойностни атрибути.
* Избран е само един атрибут за първичен ключ (ID), а не множество атрибути, което автоматично означава, че базата данни е приведена във втора нормална форма.
* Няма транзитивна зависимост между първичният ключ и неключов атрибут.

|  |  |  |
| --- | --- | --- |
| **Таблица REPAIR** | | |
| **Атрибути** | **Тип** | **Описание** |
| **ID** | INTEGER | Първичен ключ |
| **ACCEPTANCE\_DATE** | DATE | Дата на приемна на техниката за ремонт |
| **COMPLETION\_DATE** | DATE | Дата на приключване на ремонта на техниката |
| **EMPLOYEE\_ID** | INTEGER | Работник извършил ремонта (външен ключ) |
| **REPAIR\_SHOP\_ID** | INTEGER | Сервиз (външен ключ) |
| **VEHICLE\_ID** | INTEGER | Пътна или извънпътна техника (външен ключ) |
| **ACCEPTED\_THE\_VEHICLE** | VARCHAR(128) | Името на човека приел готовата техника след приключване на ремонта |
| **NOTE** | VARCHAR(256) | Бележка при необходимост |

Тази таблица съдържа данни за поправкитите на пътна или извънпътна техника.

Тя кореспондира с Entity „Ремонт“ от ER модела на базата данни.

Таблицата REPAIR е приведена и в трите нормални форми, защото:

* Всички атрибути са прости/атомични т.е. няма съставни или многостойностни атрибути.
* Избран е само един атрибут за първичен ключ (ID), а не множество атрибути, което автоматично означава, че базата данни е приведена във втора нормална форма.
* Няма транзитивна зависимост между първичният ключ и неключов атрибут.

|  |  |  |
| --- | --- | --- |
| **Таблица EMPLOYEE** | | |
| **Атрибути** | **Тип** | **Описание** |
| **ID** | INTEGER | Първичен ключ |
| **FIRST\_NAME** | VARCHAR(32) | Първо име на служител |
| **FAM\_NAME** | INTEGER(32) | Фамилия на служител |
| **PHONE\_NUMBER** | VARCHAR(20) | Личен телефонен номер на служител |
| **SERVICE\_PHONE\_NUMBER** | VARCHAR(20) | Служебен телефонен номер на служител |
| **E\_MAIL** | VARCHAR(64) | Имейл на служител |
| **POSSITION\_ID** | INTEGER | Позиция заемата от работник (външен ключ) |
| **USER\_PASSWORD** | VARCHAR(64) | Потребителско име на акаунт |
| **USERNAME** | VARCHAR(64) | Парола на акаунт |

Тази таблица съдържа данни за работниците отговорни за поправките на пътната и извън пътна техника.

Тя кореспондира с Entity „Служител“ от ER модела на базата данни.

Таблицата EMPLOYEE е приведена и в трите нормални форми, защото:

* Всички атрибути са прости/атомични т.е. няма съставни или многостойностни атрибути.
* Избран е само един атрибут за първичен ключ (ID), а не множество атрибути, което автоматично означава, че базата данни е приведена във втора нормална форма.
* Няма транзитивна зависимост между първичният ключ и неключов атрибут.

|  |  |  |
| --- | --- | --- |
| **Таблица REPAIR\_SHOP** | | |
| **Атрибути** | **Тип** | **Описание** |
| **ID** | INTEGER | Първичен ключ |
| **NAME** | VARCHAR(128) | Име на сервиза |
| **ADDRESS** | VARCHAR(256) | Адрес на сервиза |

Тази таблица съдържа данни за сервизите в които е извършения ремонт на пътната или извънпътна техника.

Тя кореспондира с Entity „Сервиз“ от ER модела на базата данни.

Таблицата REPAIR\_SHOP е приведена и в трите нормални форми, защото:

* Всички атрибути са прости/атомични т.е. няма съставни или многостойностни атрибути.
* Избран е само един атрибут за първичен ключ (ID), а не множество атрибути, което автоматично означава, че базата данни е приведена във втора нормална форма.
* Няма транзитивна зависимост между първичният ключ и неключов атрибут.

|  |  |  |
| --- | --- | --- |
| **Таблица POSSITION** | | |
| **Атрибути** | **Тип** | **Описание** |
| **ID** | INTEGER | Първичен ключ |
| **NAME** | VARCHAR(64) | Име на позицията заемата от служител |

Тази таблица като номенклатура и съдържа данни за различните позиции които служителите могат да заемат. Нампример „Механик“, „Продавач-консултант“ и т.н.

Тя кореспондира с Entity „Позиция“ от ER модела на базата данни.

Таблицата POSSITION е приведена и в трите нормални форми, защото:

* Всички атрибути са прости/атомични т.е. няма съставни или многостойностни атрибути.
* Избран е само един атрибут за първичен ключ (ID), а не множество атрибути, което автоматично означава, че базата данни е приведена във втора нормална форма.
* Няма транзитивна зависимост между първичният ключ и неключов атрибут.

|  |  |  |
| --- | --- | --- |
| **Таблица REPAIR\_SERVICE** | | |
| **Атрибути** | **Тип** | **Описание** |
| **ID** | INTEGER | Първичен ключ |
| **SERVICE\_ID** | INTEGER | Услуга (външен ключ) |
| **REPAIR\_ID** | INTEGER | Ремонт (външен ключ) |
| **TOTAL\_PRICE** | INTEGER | Общата цена на ремонт |

Тази таблица осъществява N:M връзката между таблиците Услуга и Ремонт.

Таблицата REPAIR\_SERVICE е приведена и в трите нормални форми, защото:

* Всички атрибути са прости/атомични т.е. няма съставни или многостойностни атрибути.
* Избран е само един атрибут за първичен ключ (ID), а не множество атрибути, което автоматично означава, че базата данни е приведена във втора нормална форма.
* Няма транзитивна зависимост между първичният ключ и неключов атрибут.

|  |  |  |
| --- | --- | --- |
| **Таблица SERVICE** | | |
| **Атрибути** | **Тип** | **Описание** |
| **ID** | INTEGER | Първичен ключ |
| **NAME** | VARCHAR(64) | Име на услугата |

Тази таблица съдържа данни за различните услуги необходими за даден ремонт. Също може да се приеме, че е таблица номенклатура на услуги.

Тя кореспондира с Entity „Услуга“ от ER модела на базата данни.

Таблицата SERVICE е приведена и в трите нормални форми, защото:

* Всички атрибути са прости/атомични т.е. няма съставни или многостойностни атрибути.
* Избран е само един атрибут за първичен ключ (ID), а не множество атрибути, което автоматично означава, че базата данни е приведена във втора нормална форма.
* Няма транзитивна зависимост между първичният ключ и неключов атрибут.

|  |  |  |
| --- | --- | --- |
| **Таблица ITEM** | | |
| **Атрибути** | **Тип** | **Описание** |
| **ID** | INTEGER | Първичен ключ |
| **ITEM\_NOMENCLATURE\_ID** | INTEGER | Номенклатура на артикули (външен ключ) |
| **SERVICE\_ID** | INTEGER | Услуга (външен ключ) |
| **PRICE** | DECIMAL(16,2) | Цена на артикул |
| **QUANTITY** | DECIMAL(8,2) | Количество на артикул |

Тази таблица съдържа данни за артикулите необходими за извършването на дадена услуга за ремонт.

Тя кореспондира с Entity „Артикул“ от ER модела на базата данни.

Таблицата ITEM е приведена и в трите нормални форми, защото:

* Всички атрибути са прости/атомични т.е. няма съставни или многостойностни атрибути.
* Избран е само един атрибут за първичен ключ (ID), а не множество атрибути, което автоматично означава, че базата данни е приведена във втора нормална форма.
* Няма транзитивна зависимост между първичният ключ и неключов атрибут.

|  |  |  |
| --- | --- | --- |
| **Таблица ITEM\_NOMENCLATURE** | | |
| **Атрибути** | **Тип** | **Описание** |
| **ID** | INTEGER | Първичен ключ |
| **MANUFACTURER\_ID** | INTEGER | Производител (външен ключ) |
| **MEASURE\_ID** | INTEGER | Мярка (външен ключ) |
| **NAME** | VARCHAR(64) | Име на артикула |

Тази таблица изнася част от атрибутите на Entity “Артикул“ в отделно поради тяхната повторяемост.

Таблицата ITEM\_NOMENCLATURE е приведена и в трите нормални форми, защото:

* Всички атрибути са прости/атомични т.е. няма съставни или многостойностни атрибути.
* Избран е само един атрибут за първичен ключ (ID), а не множество атрибути, което автоматично означава, че базата данни е приведена във втора нормална форма.
* Няма транзитивна зависимост между първичният ключ и неключов атрибут.

|  |  |  |
| --- | --- | --- |
| **Таблица MANUFACTURER** | | |
| **Атрибути** | **Тип** | **Описание** |
| **ID** | INTEGER | Първичен ключ |
| **NAME** | VARCHAR(64) | Име на производител |

Тази таблица служи като номенклатура и съдържа данни за производителите.

Тя кореспондира с Entity „Производител“ от ER модела на базата данни.

Таблицата MANUFACTURER е приведена и в трите нормални форми, защото:

* Всички атрибути са прости/атомични т.е. няма съставни или многостойностни атрибути.
* Избран е само един атрибут за първичен ключ (ID), а не множество атрибути, което автоматично означава, че базата данни е приведена във втора нормална форма.
* Няма транзитивна зависимост между първичният ключ и неключов атрибут.

|  |  |  |
| --- | --- | --- |
| **Таблица MEASURE** | | |
| **Атрибути** | **Тип** | **Описание** |
| **ID** | INTEGER | Първичен ключ |
| **NAME** | VARCHAR(64) | Име на мярката |
| **ABBREVIATATION** | VARCHAR(16) | Съкращение на мярката |

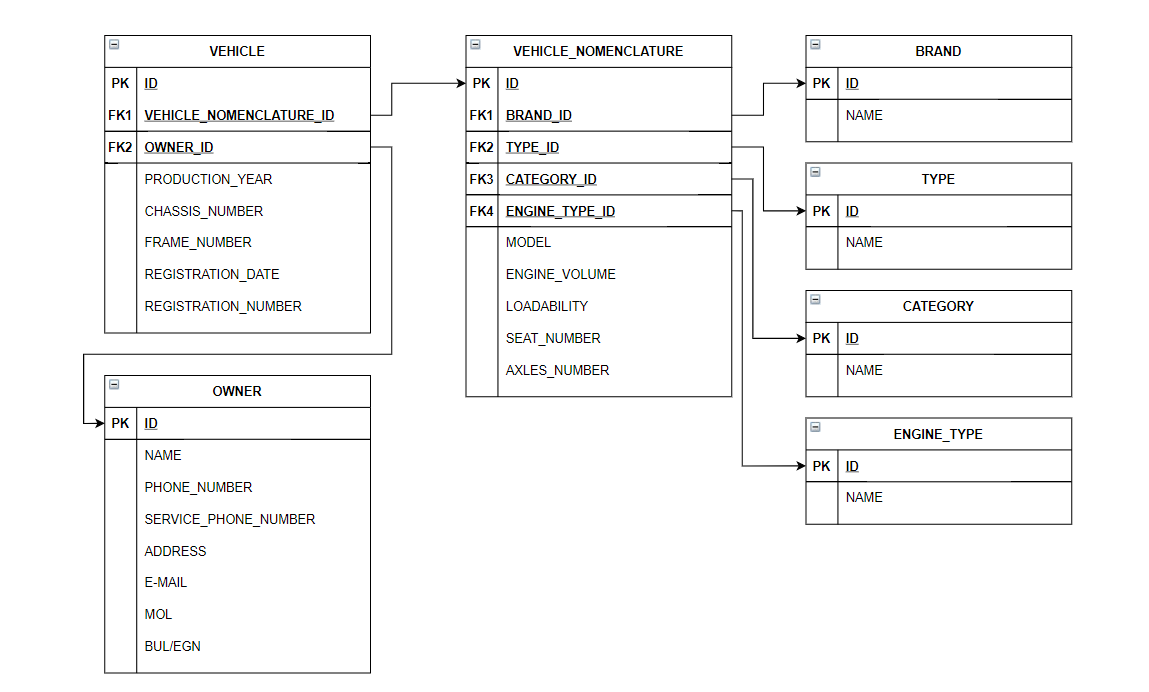
Тази таблица съдържа данни за използваните мерни единици за количество на артикулите.

Тя кореспондира с Entity „Мярка“ от ER модела на базата данни.

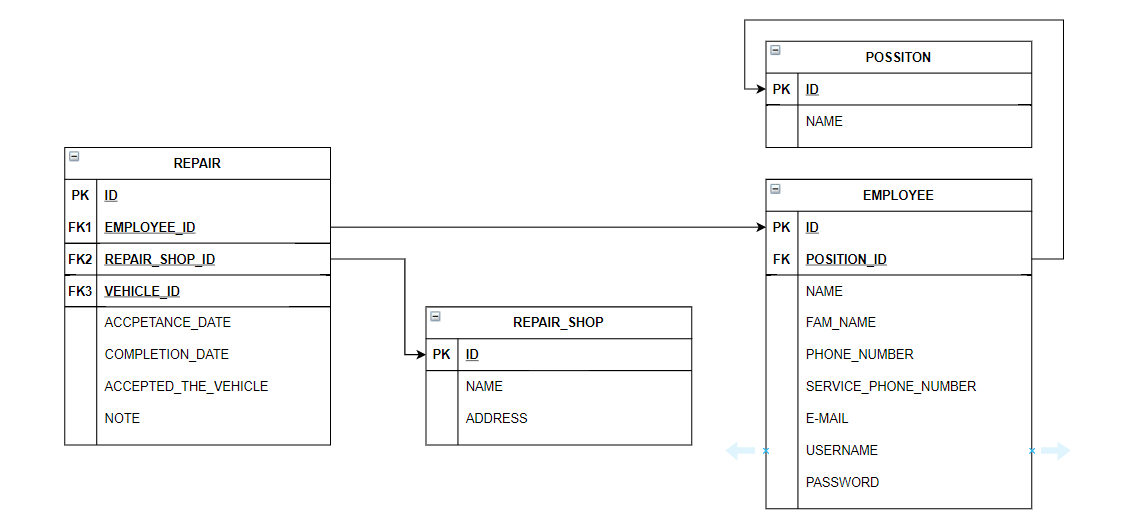
Таблицата MEASURE е приведена и в трите нормални форми, защото:

* Всички атрибути са прости/атомични т.е. няма съставни или многостойностни атрибути.
* Избран е само един атрибут за първичен ключ (ID), а не множество атрибути, което автоматично означава, че базата данни е приведена във втора нормална форма.
* Няма транзитивна зависимост между първичният ключ и неключов атрибут.

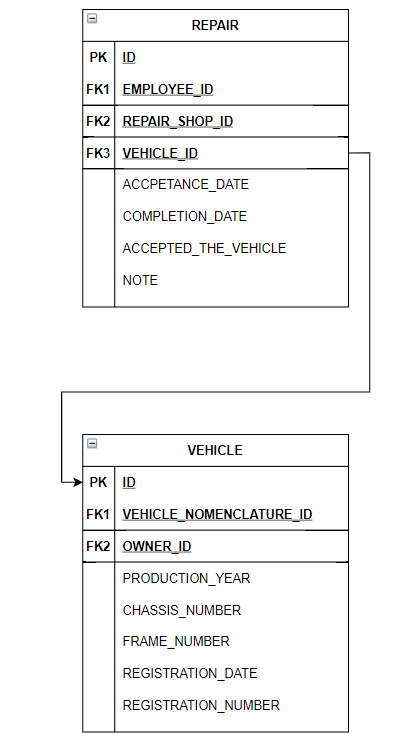
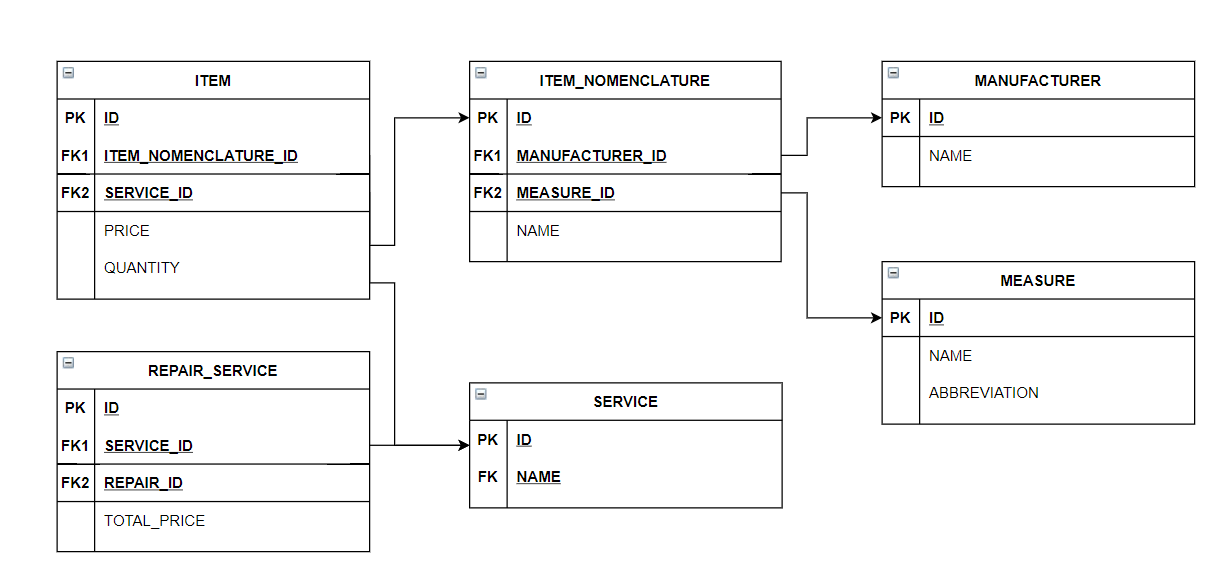
**2.2.2. Връзки между релационните таблици**



**Фигура 4: Връзки на таблица „Пътна и извънпътна техника“ с таблиците „Собственик“ и „Техническа характеристика“. Връзки на таблица „Техническа характеристика“ с таблиците „Марка“, „Тип“, „Катергория“ и „Вид двигател“**

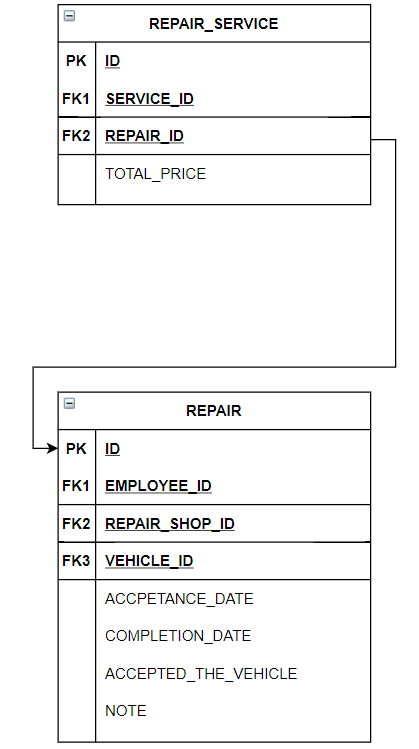


**Фигура 5: Връзки на таблица „Ремонт“ с таблиците „Сервиз“ и „Служител“. Връзка на таблица „Служител“ с таблица „Позиция“.**



**Фигура 6: Връзка на таблица RepairService с таблицата „Услуга“. Връзки на таблицата „Артикул“ с таблиците „Услуга“ и ItemNomenclature. Връзки на таблицата „ItemNomenclature“ с таблиците „Производител“ и „Мярка“**

**Фигура 7: Връзка на таблицита „Ремонт“ с таблица „Пътна и извънпътна техника“**



**Фигура 8: Връзка на таблиците „RepairService“ и „Ремонт“.**

**Глава 3**

**Обосновка за избор на машина за база данни, език за програмиране и програмна среда**

**3.1. Обосновка за избор на машина за бази данни**

Избрана е релационна база данни (relation database management system). Тя се характеризира с по-висока мощност от другите типове бази данни, както и по-голяма стабилност.

Като машина за бази данни е избран InterBase Server DBMSv6.5, защото притежава едни от най-важните характеристики на една система за управление на база данни – ефективност и бързодействие.Други характерни черти на InterBase са: висока производителност, пълен набор от функции, ниска цена, машина за бази данни с минимални системни изисквания. Работи под платформите Windows, Linux, Mac.Вградени са средства за сигурност като потребителски акаунти и различни права на достъп, и криптиране на данните. Машината за бази данни може да работи локално или на отдалечен сървър. Поддържа стандарта SQL 92 и мениджмънт на трансакциите.

Машината за бази данни е InterBase Server DBMSv6.0. Нейни предимства са висока производителност, ниска цена и минимални системни изисквания. Работи под различни платформи като Windows, Linux, Mac и др. Друга силна черта е, че има вградени средства за сигурност като криптиране на данните и потребителски акаунти.

Ако сравним InterBase с други бази данни, то те са прекалено ограничени или скъпи за целта на задачата. Друга силна страна на InterBase е, че е интегрирана за Borland C++ Builder.

**3.2. Обосновка за избор на език за програмиране**

Избран е езикът за програмиране С++. Една от най-големите предимства е, че предлага на програмиста много възможности и висока ефективност. Друг основен плюс е, че е обектно-ориентиран език. Това от своя страна дава много възможности. Например потребителски дефиниране класове заедно с голям набор от готови такива. Разбира се като всеки обектно-ориентиран език притежава основните принципи: полиморфизъм, наследяване, капсулация и абстракция, както и други предимства.

**3.3. Обосновка за избор на програмна среда**

Избрана е средата за програмиране Borland C++ Builder, която е среда за бърза разработка на приложения и като такава предлага много компоненти и функционалности за бързото разработване на софтуер. Това се дължи на библиотеката VCL която съдържа голяма част от компонентите. Както вече беше споменато предимство е, че Borland Builder работи на С++ което му предоставя всички плюсове на езика, също така стила на програмиране е contract-free.

**Глава 4**

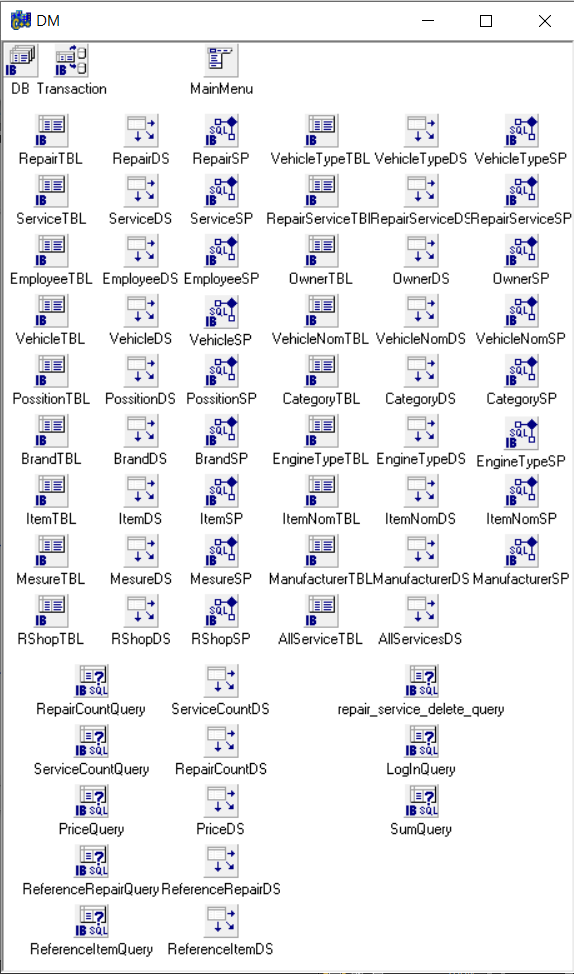
**Описание на основните програмни модули и на взаимодействието между тях**

**4.1. Описание на форма модул за данни DataModuleUnit**

Приложението е създадено чрез интегрираната среда за разработка (IDE) Borland C++ Builder. Поради тази причина, това програмно осигуряване използва концепциите и методите, които средата имплементира. Част от възможностите на средата е модул за данни, който позволява лесното управление на визуалните компоненти, като съдържа само тях в себе си и играе ролята на мост между различните модули.

**4.1.1. Цели логика на DataModuleUnit**

Този модул се използва като контейнер за невизуални компоненти. Използван такъв е компонента TIBDatabase за връзка с базата данни. Този компонент се нуждае от компонента за транзакции към базата, а именно компонента TIBTransaction. За да могат да се използват данните в базата, те се извличат в компоненти от типа TIBTable, като за всяка таблица от база съответства различен компонент от този тип. За да се достъпват данните в табличните компоненти се използва компонента TDataSource. Така имаме пълен достъп до базата данни. Чрез компонентите TIBStoredProc се осъществява достъп до „stotred procedures“ в базата данни, а чрез TIBQuery могат да се изпълняват SQL заявки и да се извличат резултатите им. Компонентите за процедури в това приложение изпълняват функцията на автоматични генератори на ID.



**Фигура 9: Модул DataModuleUnit**

**4.1.2. Съставни части на DataModuleUnit**

Модулът за данни към приложението се състои от следните компоненти:

* DB – компонент за връзка с бази данни
* Transaction – компонент за трансакции към базата данни
* Компоненти за извличане на данни от конкретна таблица – RepairTBL, ServiceTBL, EmployeeTBL, VehicleTBL, PossitionTBL, BrandTBL, ItemTBL, MesureTBL, RShopTBL, VehicleTypeTBL, RepairServiceTBL, OwnerTBL, VehicleNomTBL, CategoryTBL, EngineTypeTBL, ItemNomTBL, ManufacturerTBL и AllServiceTBL.
* Компоненти за достъпване на данни от конкретна таблица – RepairDS, ServiceDS, EmployeeDS, VehicleDS, PossitionDS, BrandDS, ItemDS, MesureDS, RShopDS, VehicleTypeDS, RepairServiceDS, OwnerDS, VehicleNomDS, CategoryDS, EngineTypeDS, ItemNomDS, ManufacturerDS и AllServiceDS.
* Компоненти stored procedures – RepairSP, ServiceSP, EmployeeSP, VehicleSP, PossitionSP, BrandSP, ItemSP, MesureSP, RShopSP, VehicleTypeSP, RepairServiceSP, OwnerSP, VehicleNomSP, CategorySP, EngineTypeSP, ItemNomSP и ManufacturerSP.
* Компоненти от тип заявка:
  + - RepairCountQuery - изпраща зявка, която извлича резултат брой на поправките според марка пътна или извънпътна техника подредени по големина.
    - ServiceCountQuery - аналогично изпраща заявка, която извлича резултат брой на извършваните услуги.
    - PriceQuery - извлича резултат за сума от цена на ремонти по марка пътна или извънпътна техника и най – скъпия извършван ремонт.
    - ReferenceRepairQuery - извлича информация по зададена справка за ремонти в определен период от време или марка, специфична пътна или извън пътна техника или сервиз.
    - ReferenceItemQuery - аналогично извлича информация по зададена справка, но вместо ремонти дава информация за части.
    - Repair\_service\_delete\_query - изтрива всички услуги на ремонт. Използва се поради по-сложното изтриване на дани фигуриращи в N:M връзката между таблиците RepairTBL и ServiceTBL.
    - LogInQuery - проверява дали има съвпадащи потребителско име и парола на регистриран служител.
    - SumQuery - смята сумата на всички артикули за услуга.
* За заявките, които извелчения резултат показва в последствие се създават и компоненти за достъпване на данни – ServiceCountDS, RepairCountDS, PriceDS, ReferenceRepairDS, ReferenceItemDS.
* Един компонент от тип TMainMenu за реализиране на менюто в главната форма.

**4.1.3. Събития на DataModuleUnit**

В модула за данни са реализирани определени събития, които извършват определени процедури. Те са:

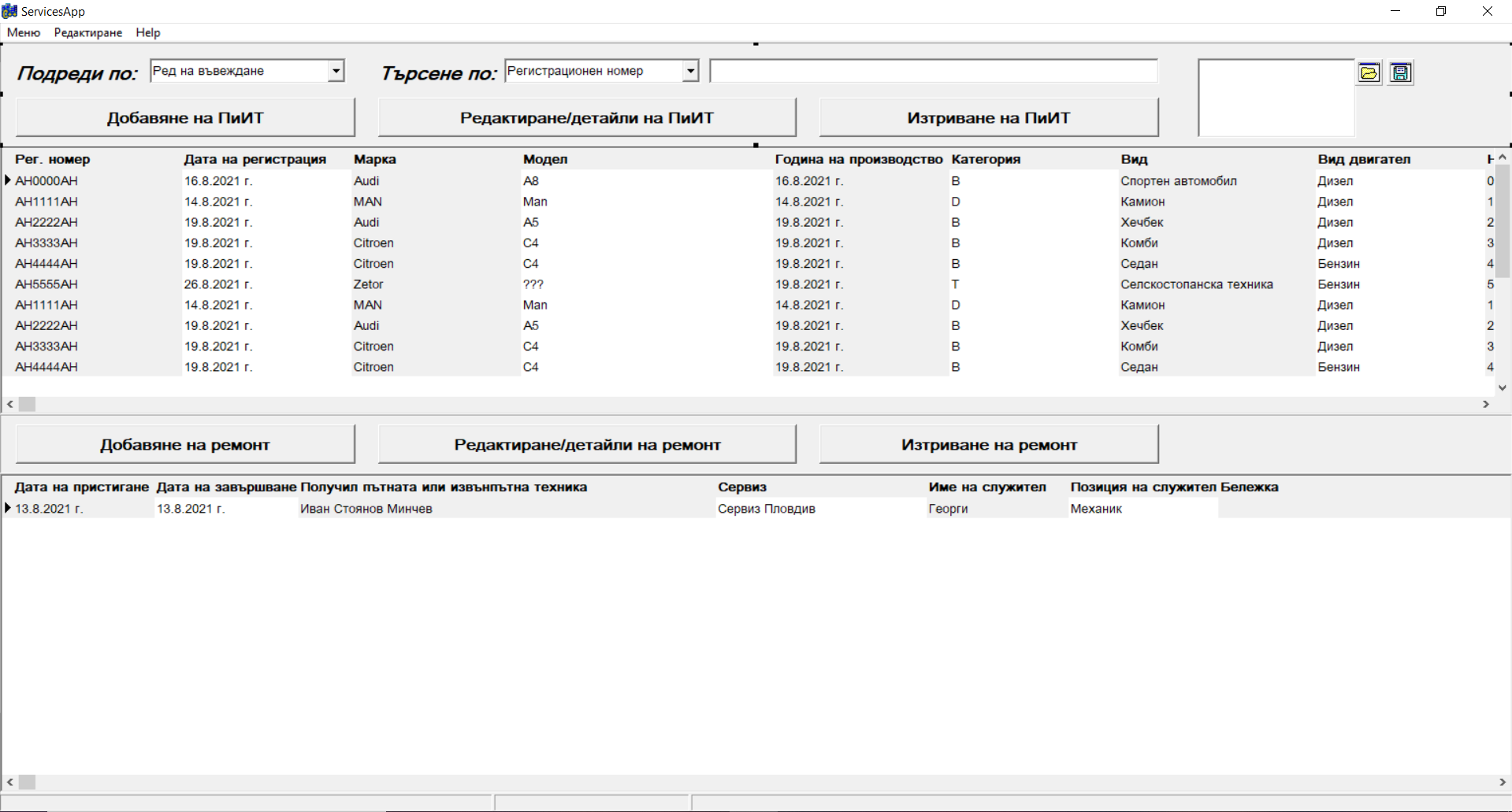
* Всяка TIBTable компонент има реализирано събитие BeforePost(), което се извиква точно преди данни да се запишат и автоматоматично записва ID на записа, който ще бъде записан. Реализира се благодарение на stored procedures.

**4.2. Описание на главна форма MainForm**

Главната форма е първото нещо, което се вижда след стартиране на приложението и след успешно въвеждане на парола и потребителско име. В тази програма, тя показва най-използваната информация и менюто, което се намира в нея, служи за извикване на всички останали форми.

**4.2.1. Цел и логика на главна форма MainForm**

Целта е да изведе информацията от базата данни за всяка пътна и извънпътна техника и съответно при посочване на някоя от тях да се показват извършените ремонти на посочената пътна или извънпътна техника. Също така, има функция за отваряне на отделни форми, от които могат да се добавят, редактират или гледат детайли за посочената техника или ремонт. Също така, съдържа главното меню, което отвежда до почти цялата функционалност на приложението.



**Фигура 10: Форма MainUnit**

**4.2.2. Съставни части на главна форма MainForm**

Формата се състои от следните компоненти:

* Съдържа два компонента от клас TComboBox, които служат за указване на начина на подреждане и за търсене, за което е необходимо и използването на компонент от клас TEditBox за въвеждане на ключ за търсене. Тези компоненти са: MainSerachCB, MainSortCB
* Всеки един раздел на приложението съдържа компонент от клас TDBGrid, който служи за визуализиране на данните от съответната таблица (функцията за изтриване и добавяне на данни е имплементирана в други форми). Използват се следните компоненти: MainVehicleGrid и MainRepairGrid
* Бутони, шест от тип TButton – MainDeleteRepair, MainEditRepair, MainAddRepair, MainDeleteVehicle, MainEditVehicle, MainAddVehicle
* Статус бар (TStatusBar), който показва кратка информация за работника, който се е влязал в програмата.
* TMemo, TSaveDialog, TOpenDialog – използвани за импорт и експорт

**4.2.3. Събития на главна форма MainForm**

Обработват се следните събития:

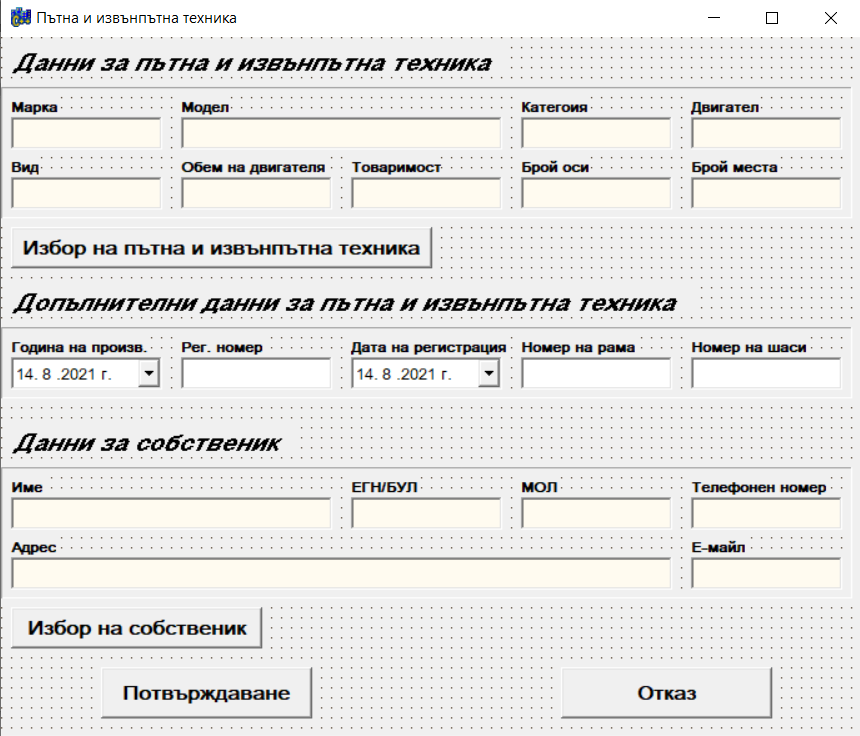
* Събитие OnExit()на всеки един DBGrid компонент – при загуба на фокус данните се запазват за да не се получава загуба на данни.
* Събитие OnChange() на EditBox компонента – настъпва, когато се промени съдържанието на полето за редактиране, за изпъняване на инкрементално търсене.
* Всички събития от тип click, които се случват в компонента за главно меню MainMenu са за показване на друг вид форма или съобщение.
* Събитие OnSelect()на ComboBox – обработва се това събитие, вместо събитието OnClick () на съответния ComboBox, защото стилът на компонента csDropDownList, а това изисква събитие OnSelect (). Събитието настъпва, когато потребителят избере опция от падащото меню на ComboBox. Тогава, в зависимост от това дали ComboBox компонента е за смяна начина на показване на данните или критериий за търсене, се променя редът на показване на записите или критерия за търсене на запис.
* Събитиия при OnClick на бутон – два бутона за отваряне на формата за добавяне на пътна или извънпътна техника в съответно режим за добавяне или режим за редактиране. Два бутона аналогично за ремонт. По един бутон за изтриване на пътна или извънпътна техника или ремонт.

**4.3. Описание на форма за добавяне и редактиране на пътна или извънпътна техника EditVehicleForm**

Понеже данните за пътна и извънпътна техника са сложни и разположени в различни таблици е трудно да бъдат попълнени или редактирани само с използването на грид компонента за това има специална форма с тази функция.

**4.3.1. Цел и логика на форма добавяне и редактиране на пътна или извънпътна техника EditVehicleForm**

Целта на формата е потребителя да може да добавя или редактира пътна или извънпътна техника с възможност да избира от вече съществуващи собственици и технически харескиристики за тях.



**Фигура 11: Форма EditVehicleForm**

**4.3.2. Съставни части на форма за добавяне и редактиране на пътна и извънпътна техника EditVehicleForm**

Тази форма се състои от следните компоненти:

* Компоненти от клас TEditBox за въвеждане на данните – RegNumEdit, FrameEdit ChasseEdit
* Компоненти от клас TDateTimePicker за избор на дата – YearManufactTimePicker и RegDateTimePicker
* Бутони от клас TButton за потвърждение и отказ на промените и отваряне на форми за редактиране от където могат да се изберат собственик и технически характеристики на пътна или извъпътна техника – SubmitButton, CloseButton, OwnerButton, VehicleNomButton

**4.3.3. Събития на форма за добавяне и редактиране на пътна и извънпътна техника EditVehicleForm**

Формата съдържа следните събития:

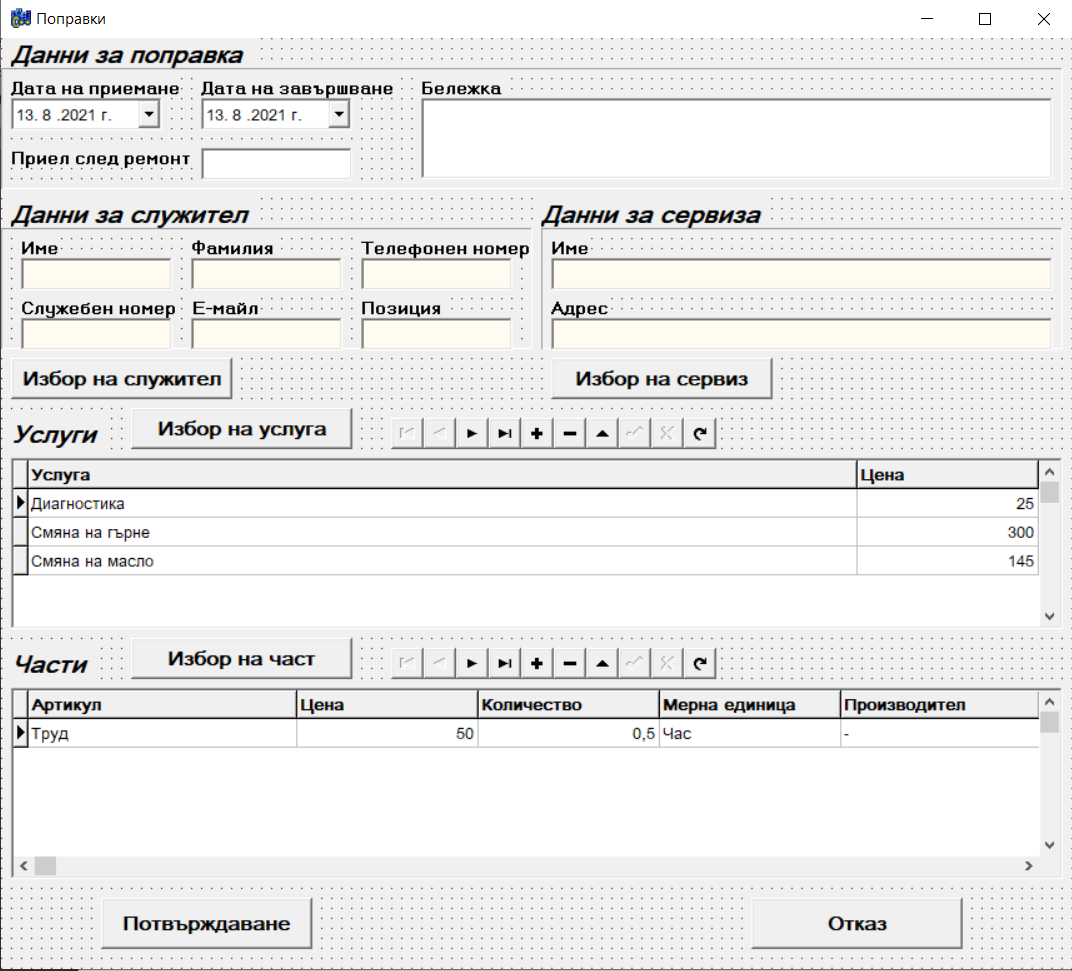
* Събитиe OnClose() – настъпва веднага след затварянето на формата. В него се изчистват всички полета, защото тази форма се отваря, както за добавяне, така и за редактиране. Ако се отвори за редактиране и веднага след това за добавяне, то всички editbox компоненти все още ще имат стойностите, показани при отварянето за редактиране. За да се избегне това всеки път при затварянето на формата се изчистват едит бокс компонентите.
* Събитие OnClick() на бутоните – при кликване бутоните за „Потвърждаване“ и „Отказ“ съответно записват или отказват започната дейност по добавяне или редактиране. А бутоните за избор отварят необходимата форма, като правят видим специфичен бутон в отворената форма за избор на записа.

**4.4 Описание на форма за добавяне и редактиране на ремонти EditPartForm**

Аналогично на формата за пътна и извънпътна техника тази форма помага за добавянето или редактирането на ремонти.

**4.4.1. Цел и логика на форма за редактиране на компоненти EditPartForm**

Целта е потребителя да може лесно да добявя или редактира и избира необходимата информация за ремонт, като услуги, артикули участващи в услугата, място на извършване на ремонта и служител извършил ремонта.



**Фигура 12: Форма EditPartForm**

**4.4.2. Съставни части на форма за добавяне и редактиране на ремонт EditPartForm**

Тази форма се състои от следните компоненти:

* Компонент от клас TEditBox за въвеждане на данни за човека приел готовата пътна или извънпътна техника и компонент от тип TMemo за записване на необходими бележки към ремонта – AccEdit и NoteMemo
* Компоненти от клас TDateTimePicker за избор на дата – AccDatePicker и CompleteDateTimePicker
* Компоненти от тип TDBGrid за показване и манипулиране на избранни данни за услуги и части, като цена и количество на част – ServiceGrid и ItemGrid
* Компоненти от тип TDBNavigator за по-лесно манипулиране на данните в грид компонентите – ServiceNavigator и ItemNavigator
* Бутони от клас TButton за потвърждение и отказ на промените и отваряне на форми за редактиране, от където могат да се изберат собственик и технически характеристики на пътна или извъпътна техника – SubmitButton, CloseButton, SelectItemNomButton, SelectServiceButton, SelectRShopButton, SelectEmployeeButton

**4.4.3. Събития на форма за добавяне и редактиране на пътна и извънпътна техника EditPartForm**

Формата съдържа следните събития:

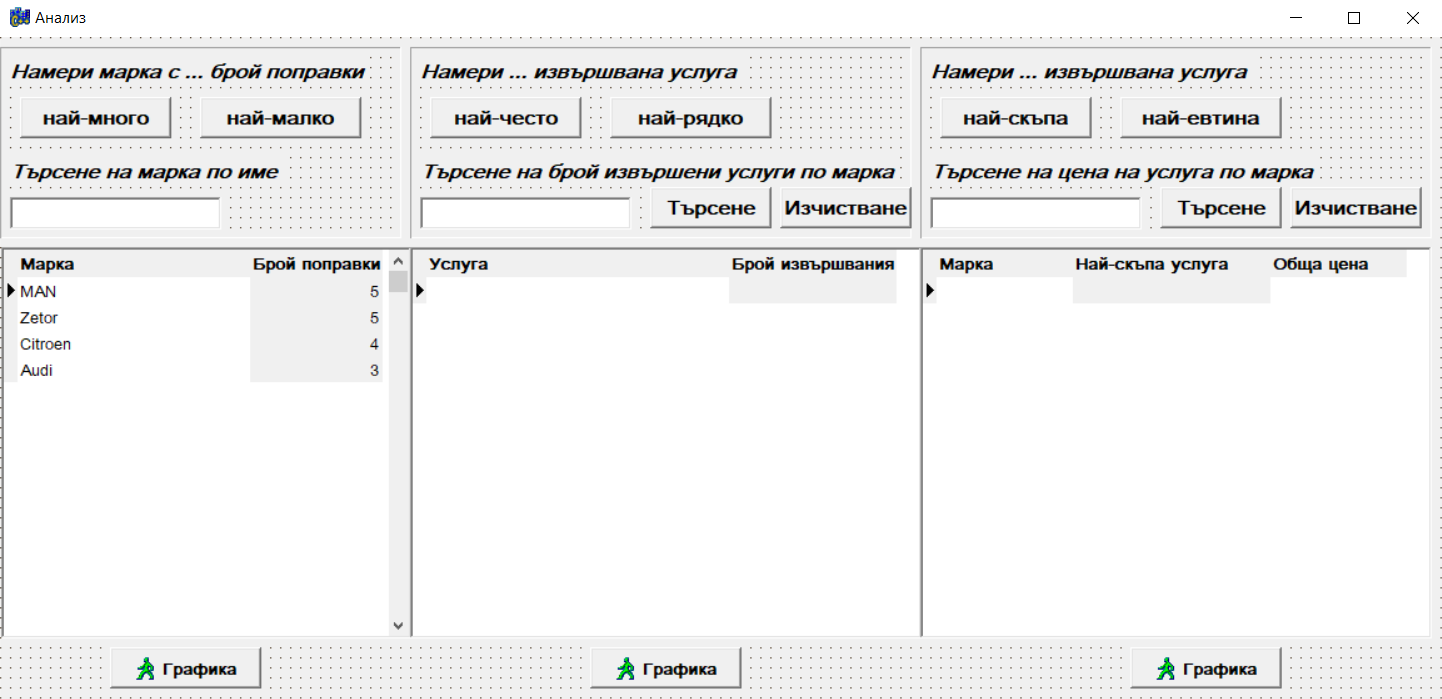
* Събитиe OnClose() – настъпва веднага след затварянето на формата. В него се изчистват всички полета, защото тази форма се отваря, както за добавяне, така и за редактиране. Ако се отвори за редактиране и веднага след това за добавяне, то всички едит бокс компоненти все още ще имат стойностите показани при отварянето за редактиране. За да се избегне това всеки път при затварянето на формата се изчистват едит бокс компонентите.
* Събитие OnClick() на бутоните – при кликване бутоните за „Потвърждаване“ и „Отказ“ съответно записват или отказват започната дейност по добавяне или редактиране. А бутоните за избор отварят необходимата форма, като правят видим специфичен бутон в отворената форма за избор на записа.

**4.5.** **Описание на форма за анализ AnalysisForm**

Тази форма показва кратка аналитична информация за брой поправки групирани по марка, брой извършвани услуги и цена на ремонтите групирани по марка. Също така намиране на минимални и максималните им стойности и показване на графика.

**4.5.1. Цел и логика на форма за анализ Analysis**

Целта е да улесни потребителя, като му предоставя повече синтезирана информация.



**Фигура 13: Форма AnalysisForm**

**4.5.2. Съставни части на форма за анализ AnalysisForm**

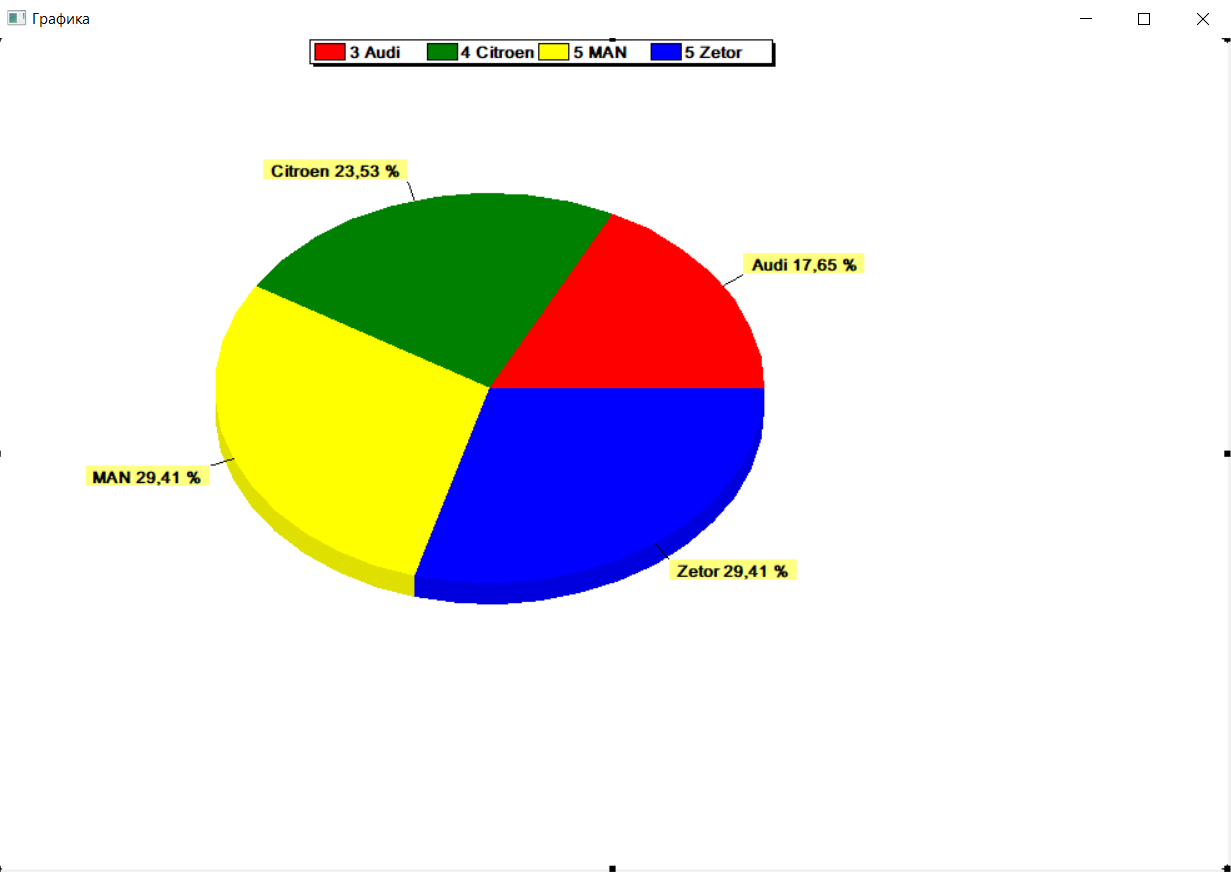
Тази форма се състои от следните компоненти:

* Компоненти от клас TEditBox за въвеждане на ключ за инкрементално търсене – SearchBrandEdit, SearchServiceEdit, SearchPriceEdit
* Компоненти от клас TDBGrid за извеждане на резултата от изпратените заявки – BrandGrid, ServiceGrid, PriceGrid
* Компоненти от клас TButton за маркиране на максимана или минимална стойност също така за търсене по записа в едит компонентата и изчистването на едит компонентата – BrandMaxButton, BrandMinButton, ServiceMaxButton, ServiceMinButton, PriceMaxButton, PriceMinButton, SearchBrandButton, SearchPriceButton
* Компоненти от клас TBitBtn за показване на графиката за съотвентната грид компонента – ServiceGraph, BrandGraph, PriceGraph

**4.5.3. Събития на форма за анализ AnalysisForm**

Формата съдържа следните събития:

* Събитиe OnChange() – на едит компонента, който търси инкрементално
* Събитие OnClick() на бутоните – при кликване бутоните за „Най-много“ или „Най-малко“ съответно маркера посочващ активния запис се премества на този запис, който отговаря съответно на най-много ремоти или най-честа извършвана услуга и т.н. А бутоните „Търсене“ и „Изчистване“ съответно филтрират данните или изчистват филтъра. Бит бутоните отварят нова форма, в която е изобразена графика на съответния грид компонент.



**Фигура 14: Форма ChartForm**

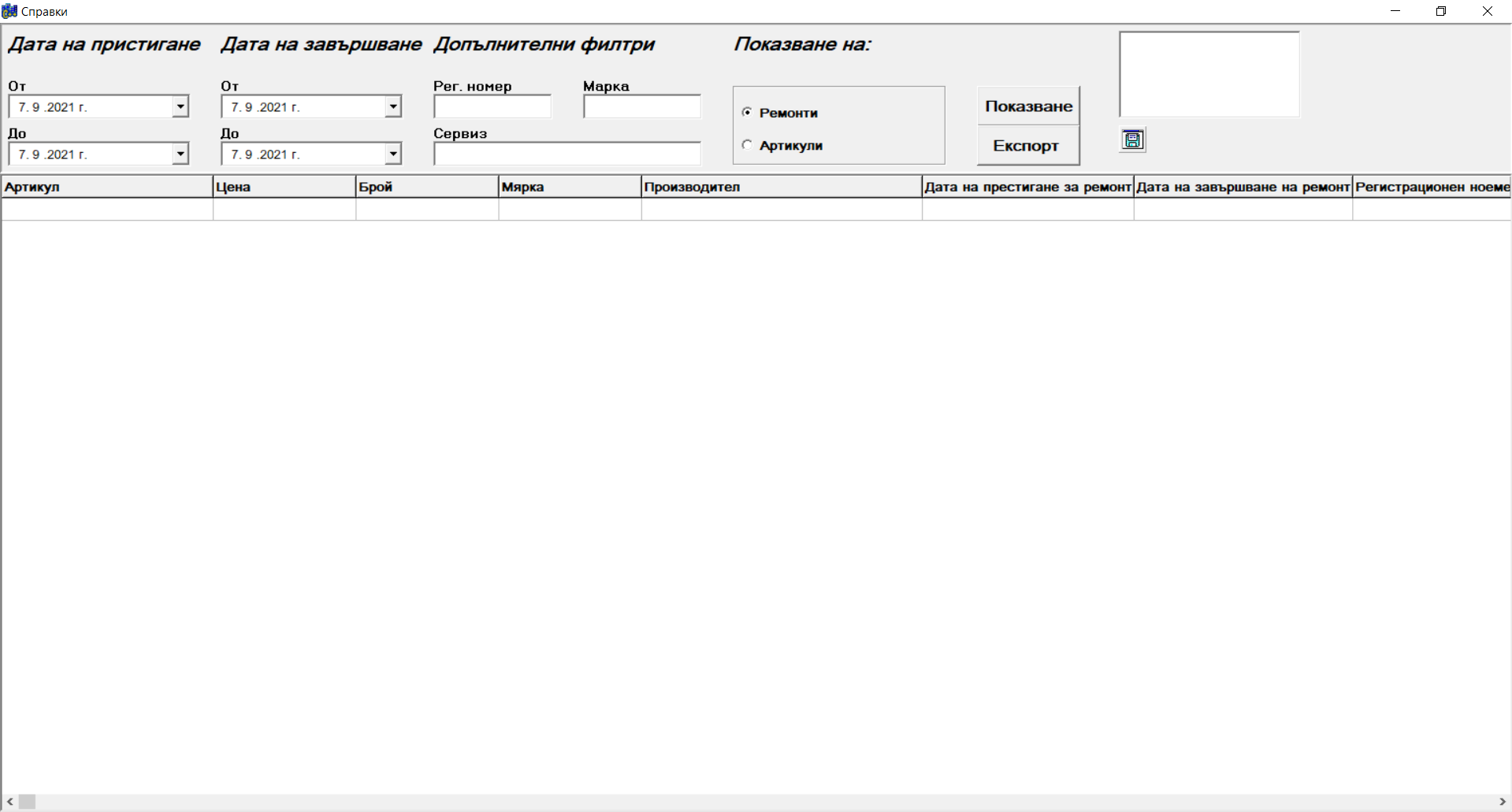
За изобразяване на графиката е използван класа TQRChart и е единствения компонент на формата ChartForm. От своя страна Chart компонента съдържа различни Series в себе си, като всяка от тях представява различните данни.

**4.6.** **Описание на форма за справки ReferenceForm**

Тази форма показва кратка аналитична информация за брой поправки групирани по марка, брой извършвани услуги и цена на ремонтите групирани по марка. Също така намиране на минимални и максималните им стойности и показване на графика.

**4.6.1. Цел и логика на форма за справки ReferenceForm**

Целта на формата е вместо потребителя да пише SQL заявки това да се получава чрез графичния интефейс. Позволява да се показват резултати за части или ремонти. Те могат да бъдат ограничени в избран от потребителя времеви период или по дадена марка, за специфична пътна или извънпътна техника или за специфичен сервиз. Също така има възможността данните да бъдат експортнати до файл с формат CSV.



**Фигура 15: Форма ReferenceForm**

**4.6.2. Съставни части на форма за справки ReferenceForm**

Тази форма се състои от следните компоненти:

* Компоненти от клас TEditBox за въвеждане от потребителя на параметри на заявката (модел, сервиз, регисртационен номер) – RegNumEdit, BrandEdit, RShopEdit
* Компонент от клас TDBGrid за визуализиране на резултата от заявката – ReferenceGrid
* Компонент от клас TRadioGroup служеш за избор между показване на артикули и ремонти - RadioGroup
* Компоненти от клас TButton, „Показване“ служещ за показване на резултат спрямо подадените ограничения и бутон „Експорт“ служещ за експорт – ShowButton и ExportButton
* Компонент от клас TMemo служещ за по-лесно експортване на данните - ExportMemo
* Компонент от клас TSaveDialog служещ за посочване на път за запазване на експортнатия файл - SaveDialog

**4.6.3. Събития на форма за справки ReferenceForm**

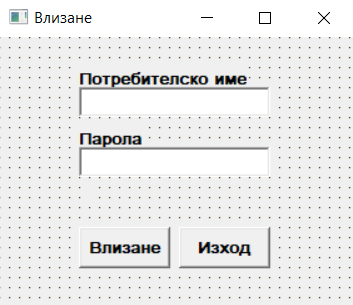
Формата съдържа следните събития:

* Събитиe OnShow() на формата – задават се начални стойности на параметрите на заявките
* Събитие OnClick() на бутоните – при кликване на бутона „Показване“ се визуализира в грид компонента справката според зададените от потребителя ограничения. При натискане на бутона „Експорт“ и задаване на път се създава файл с експортнатите данни.

**4.7.** **Описание на форма за ауторизация LogInForm**

**4.7.1. Цел и логика на форма за ауторизация LogInForm**

Целта на формата е да ограничава достъпа до приложенито, като се допускат само ауторизирани потребители със съществуващ акаунт.



**Фигура 16: Форма LogInForm**

**4.7.2. Съставни части на форма за ауторизация LogInForm**

Тази форма се състои от следните компоненти:

* Компоненти от клас TEditBox за въвеждане на потребителско име и парола – UsernameEdit и PasswordEdit
* Компоненти от клас TButton, „Влизане“ осъществяващ търсенето за съвпадащи потребителско име и парола и „Изход“ за излизане от формата – CanselButton, LogInButton

**4.7.3. Събития на форма за ауторизация LogInForm**

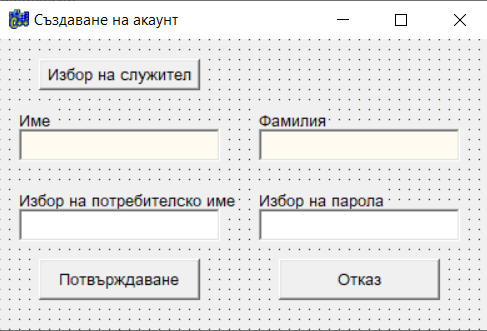
Формата съдържа следните събития:

* Събитиe FormCloseQuery() и OnShow() на формата – благодарение на променлива flag тези две събития гарантират ауторизацията на потребителя да се случва само и единствено при натискане на бутона “Влизане“
* Събитиe OnClick() на TEditBox – при кливане върху някой от едит боксовете да се премахва ако има съобщение за въвеждане на грешно протребитлско име или парола
* Събитие OnClick() на бутоните – при кликване на бутона „Влизане“ се криптират данните в едит боксовете и се търсят съвпадения в базата с данни, като това е единствения начин да се достигне до главната форма на приложението. Бутона „Изход“ е най-лесния начин да се излезе от формата, като при натискането му се терминира приложениетo.

**4.8.** **Описание на форма за създаване на акаунт CreateAccForm**

**4.8.1. Цел и логика на форма за създаване на акаунт CreateAccForm**

Целта на формата е да реализира регистрация на служетли. Чрез регистрацията те могат да имат достъп до функциите на приложението.



**Фигура 17: Форма CreateAccForm**

**4.8.2. Съставни части на форма за създаване на акаунт CreateAccForm**

Тази форма се състои от следните компоненти:

* Компоненти от клас TEditBox за въвеждане на потребителско име и парола и показване на детайли за избрания служител – UsernameEdit и PasswordEdit, EmployeeNameEdit, EmployeeFamEdit
* Компоненти от клас TButton, „Потвърждаване“ осъществяващ записването на потребителското име и парола за избрания служител, бутон „Избор на служител“ реализиращ избора на служител и бутон „Отказ“ за затваряне на формата – SelectEmployeeButton, CanselButton, AddButton

**4.8.3. Събития на форма за създаване на акаунт CreateAccForm**

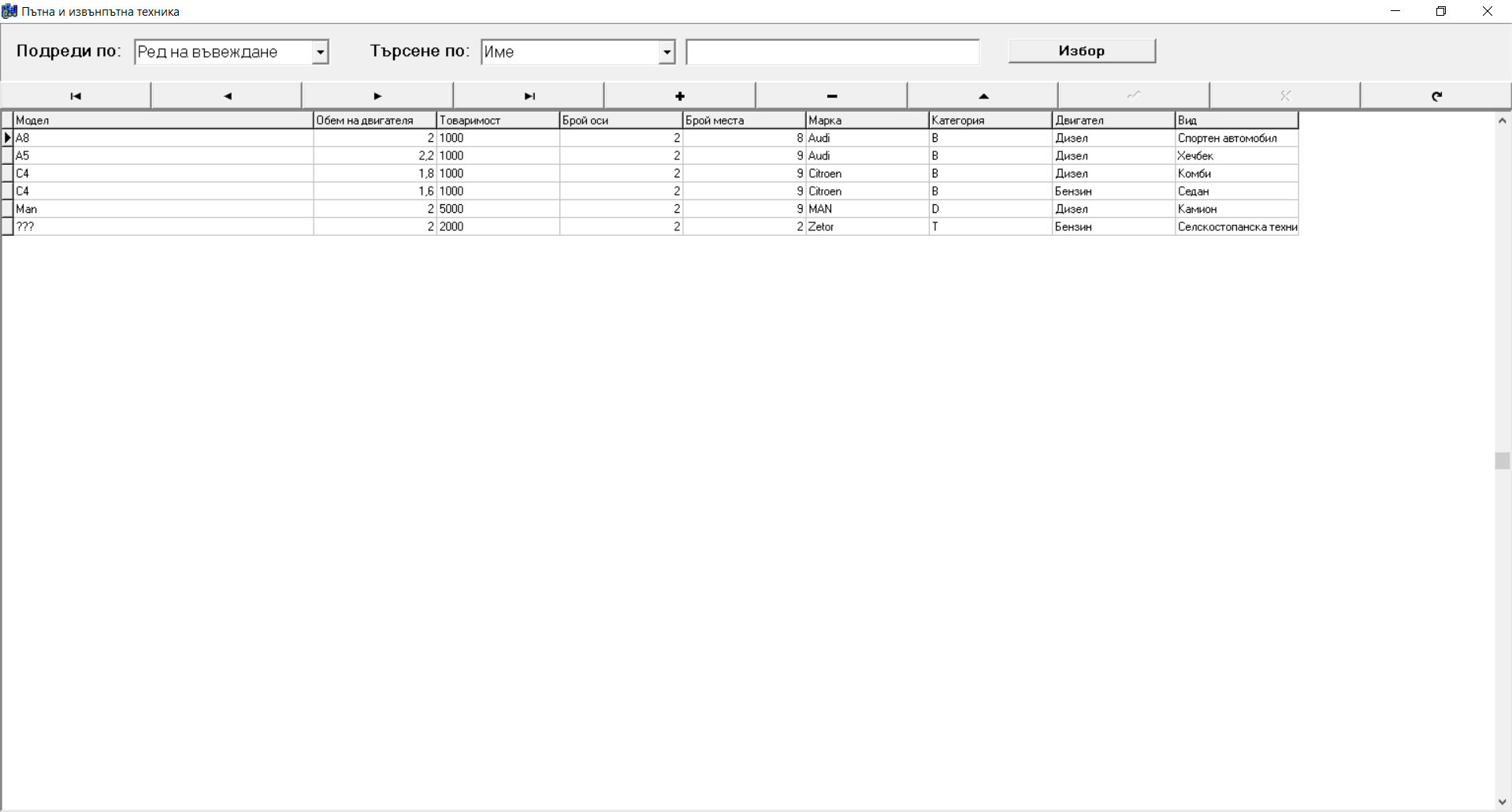
Формата съдържа следните събития:

* Събитие OnClick() на бутоните – при кликване на бутона „Потвърждаване“ се криптират данните и се записват за съответния служител. При кликване на Бутона „Избор на служител“ се отваря необходимата форма, като прави видим специфичен бутон в отворената форма за избор на записа. Бутона „Изход“ служи за излизане от формата.
* Събитие OnFormClose() на формата – изчистват полетата на едит боксовете за да не остават записите при повторно достъпване на формата.

**4.9.** **Описание на форма за редактиране на номенклатура на пътна или извънпътна техника VehicleNomForm**

**4.9.1. Цел и логика на форма за редактиране на номенклатура на пътна или извънпътна техника VehicleNomForm**

Целта на формата е да се редактират данни за номенклатури на пътна и извънпътна техника.



**Фигура 18: Форма VehicleNomForm**

**4.9.2. Съставни части на форма редактиране на номенклатура на пътна или извънпътна техника VehicleNomForm**

Тази форма се състои от следните компоненти:

* Съдържа два компонента от клас TComboBox, които служат за указване на начина на подреждане и за търсене, за което е необходимо и използването на компонент от клас TEditBox за въвеждане на ключ за търсене. Тези компоненти са: VehicleNomSerachCB, VehicleNomSortCB и VehicleNomEdit
* Компонент от тип TDBNavigator за по-лесно манипулиране на данните в грид компонентата – VehicleNomNavigator
* Компонент от клас TDBGrid за визуализиране на данните, като също така позволява да бъдат добавяни, редактирани и изтривани – VehicleNomGrid
* Компонент от клас TButton, който е видим ако формата се извиква от друга форма и служи за избор на данни и невидим когато се използва за редактиране – VehicleNomSelectButton

**4.9.3. Събития на форма за редактиране на номенклатура на пътна или извънпътна техника VehicleNomForm**

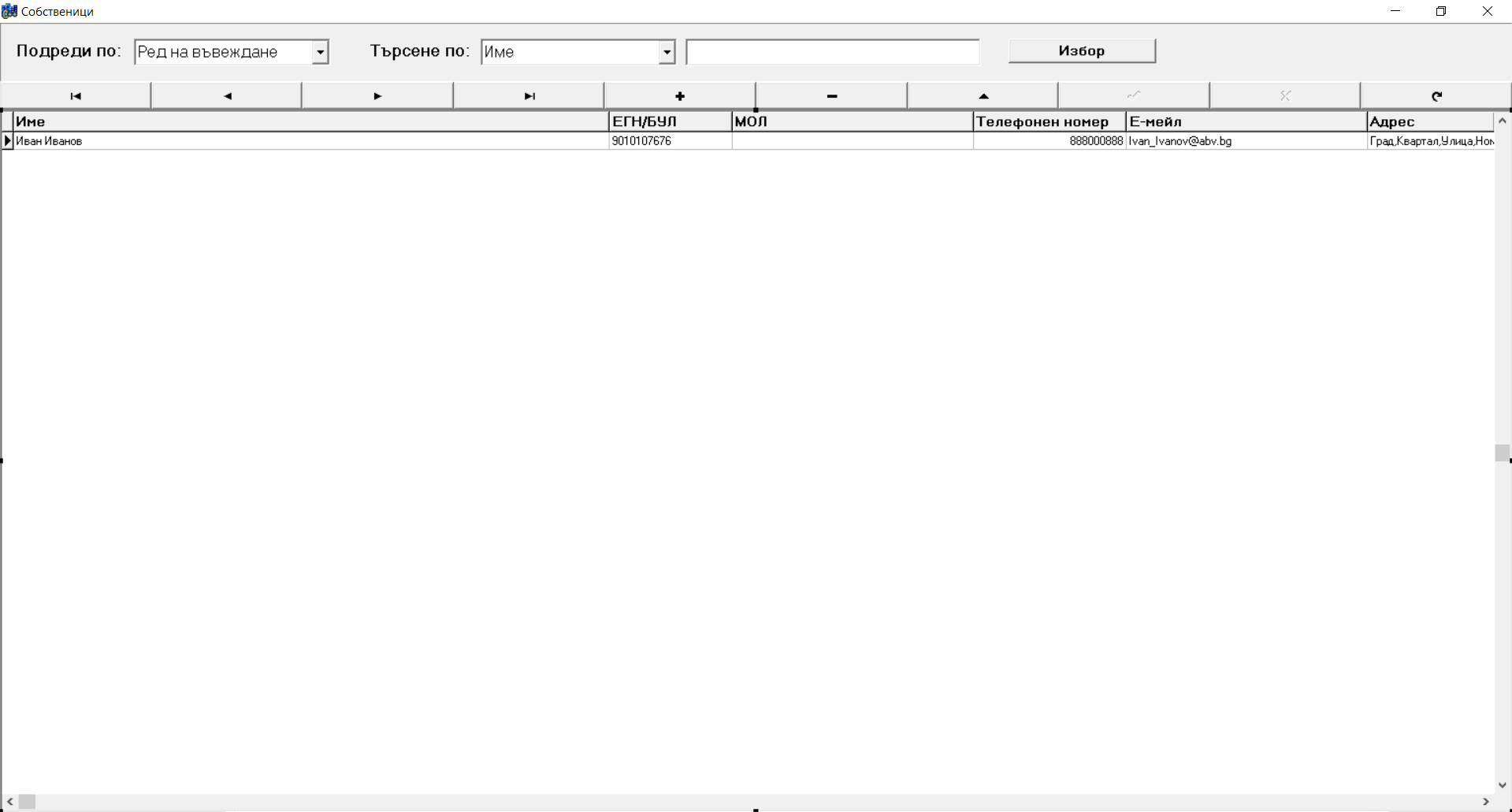
Формата съдържа следните събития:

* Събитие OnExit()на всеки един DBGrid компонент – при загуба на фокус данните се запазват за да не се получава загуба на данни.
* Събитие OnChange() на EditBox компонента – настъпва, когато се промени съдържанието на полето за редактиране, за изпъняване на инкрементално търсене.
* Събитие OnSelect()на ComboBox – обработва се това събитие, вместо събитието OnClick () на съответния ComboBox, защото стилът на компонента csDropDownList, а това изисква събитие OnSelect (). Събитието настъпва, когато потребителят избере опция от падащото меню на ComboBox. Тогава, в зависимост от това дали ComboBox компонента е за смяна начина на показване на данните или критериий за търсене, се променя редът на показване на записите или критерия за търсене на запис.
* Събитие OnClick() на бутона – реализира външен ключ връзките и записва избраните данни в полетата на необходимата форма.
* Събитие OnExit() на формата – прави бутона невидим за да не се вижда при избиране на формата само за редактиране.

**4.10.** **Описание на форма за редактиране на собственик на пътна или извънпътна техника OwnerForm**

**4.10.1. Цел и логика на форма за редактиране на собственик на пътна или извънпътна техника OwnerForm**

Целта на формата е да се редактират данни за собственик на пътна или извънпътна техника.



**Фигура 19: Форма OwnerForm**

**4.10.2. Съставни части на форма редактиране на собственик на пътна или извънпътна техника OwnerForm**

Тази форма се състои от следните компоненти:

* Съдържа два компонента от клас TComboBox, които служат за указване на начина на подреждане и за търсене, за което е необходимо и използването на компонент от клас TEditBox за въвеждане на ключ за търсене. Тези компоненти са: OwnerSerachCB, OwnerSortCB и OwnerEdit
* Компонент от тип TDBNavigator за по-лесно манипулиране на данните в грид компонентата – OwnerNavigator
* Компонент от клас TDBGrid за визуализиране на данните, като също така позволява да бъдат добавяни, редактирани и изтривани – OwnerGrid
* Компонент от клас TButton, който е видим ако формата се извиква от друга форма и служи за избор на данни и невидим когато се използва за редактиране – OwnerSelectButton

**4.10.3. Събития на форма за редактиране на собственик на пътна или извънпътна техника OwnerForm**

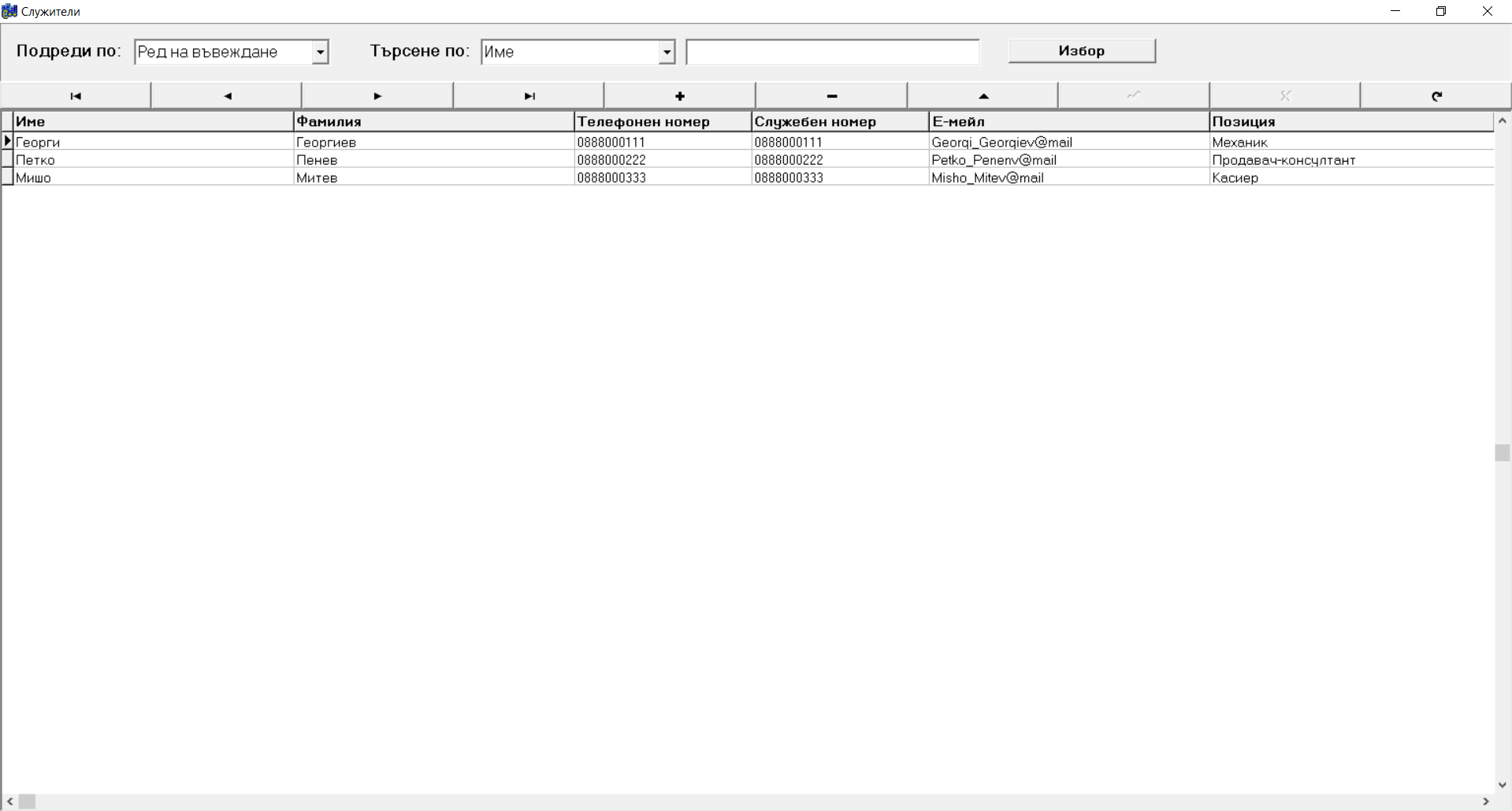
Формата съдържа следните събития:

* Събитие OnExit()на всеки един DBGrid компонент – при загуба на фокус данните се запазват за да не се получава загуба на данни.
* Събитие OnChange() на EditBox компонента – настъпва, когато се промени съдържанието на полето за редактиране, за изпъняване на инкрементално търсене.
* Събитие OnSelect()на ComboBox – обработва се това събитие, вместо събитието OnClick () на съответния ComboBox, защото стилът на компонента csDropDownList, а това изисква събитие OnSelect (). Събитието настъпва, когато потребителят избере опция от падащото меню на ComboBox. Тогава, в зависимост от това дали ComboBox компонента е за смяна начина на показване на данните или критериий за търсене, се променя редът на показване на записите или критерия за търсене на запис.
* Събитие OnClick() на бутона – реализира външен ключ връзките и записва избраните данни в полетата на необходимата форма.
* Събитие OnExit() на формата – прави бутона невидим за да не се вижда при избиране на формата само за редактиране.

**4.11.** **Описание на форма за редактиране на служители EmployeeForm**

**4.11.1. Цел и логика на форма за редактиране на служители EmployeeForm**

Целта на формата е да се редактират данни за служители.



**Фигура 20: Форма EmployeeForm**

**4.11.2. Съставни части на форма редактиране на служители EmployeeForm**

Тази форма се състои от следните компоненти:

* Съдържа два компонента от клас TComboBox, които служат за указване на начина на подреждане и за търсене, за което е необходимо и използването на компонент от клас TEditBox за въвеждане на ключ за търсене. Тези компоненти са: EmployeeSerachCB, EmployeeSortCB и EmployeeEdit
* Компонент от тип TDBNavigator за по-лесно манипулиране на данните в грид компонентата – EmployeeNavigator
* Компонент от клас TDBGrid за визуализиране на данните, като също така позволява да бъдат добавяни, редактирани и изтривани – EmployeeGrid
* Компоненти от клас TButton, който е видим ако формата се извиква от друга форма и служи за избор на данни и невидим когато се използва за редактиране – EmployeeSelectButton, EmployeeSelectAccButton

**4.11.3. Събития на форма за редактиране на служители EmployeeForm**

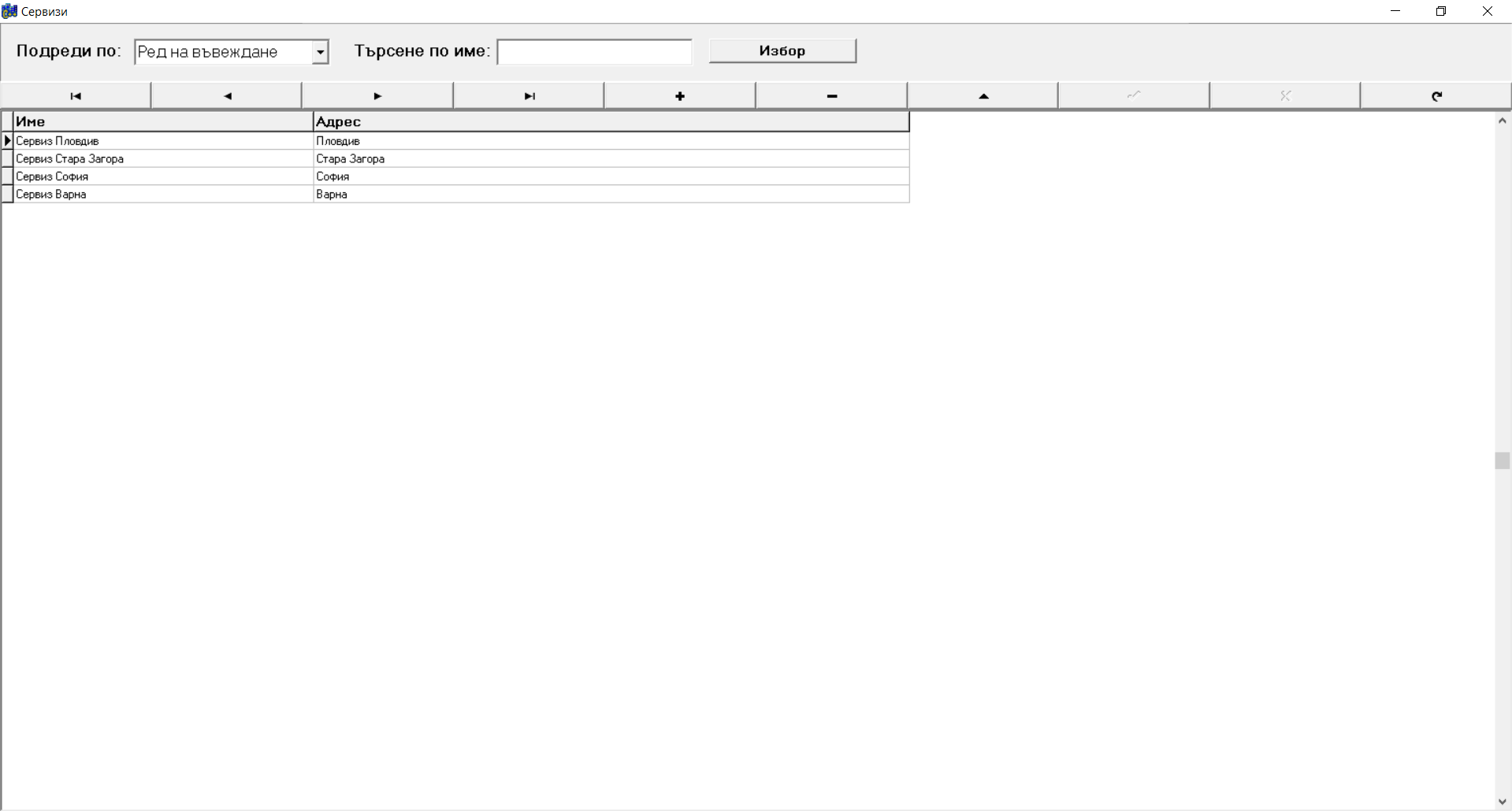
Формата съдържа следните събития:

* Събитие OnExit()на всеки един DBGrid компонент – при загуба на фокус данните се запазват за да не се получава загуба на данни.
* Събитие OnChange() на EditBox компонента – настъпва, когато се промени съдържанието на полето за редактиране, за изпъняване на инкрементално търсене.
* Събитие OnSelect()на ComboBox – обработва се това събитие, вместо събитието OnClick () на съответния ComboBox, защото стилът на компонента csDropDownList, а това изисква събитие OnSelect (). Събитието настъпва, когато потребителят избере опция от падащото меню на ComboBox. Тогава, в зависимост от това дали ComboBox компонента е за смяна начина на показване на данните или критериий за търсене, се променя редът на показване на записите или критерия за търсене на запис.
* Събитие OnClick() на бутоните – реализират външен ключ връзките и записва избраните данни в полетата на необходимата форма.
* Събитие OnExit() на формата – прави бутона невидим за да не се вижда при избиране на формата само за редактиране.

**4.12.** **Описание на форма за редактиране на сервизи RepairSForm**

**4.12.1. Цел и логика на форма за редактиране на сервизи RepairSForm**

Целта на формата е да се редактират данни за сервизи.



**Фигура 21: Форма RepairSForm**

**4.12.2. Съставни части на форма редактиране на сервизи RepairSForm**

Тази форма се състои от следните компоненти:

* Съдържа два компонент от клас TComboBox, които служи за указване на начина на подреждане – RepairSCB
* Компонент от клас TEditBox за въвеждане на ключ за търсене. Тези компоненти са: RepairSEdit
* Компонент от тип TDBNavigator за по-лесно манипулиране на данните в грид компонентата – RepairSNavigator
* Компонент от клас TDBGrid за визуализиране на данните, като също така позволява да бъдат добавяни, редактирани и изтривани – RepairSGrid
* Компонент от клас TButton, който е видим ако формата се извиква от друга форма и служи за избор на данни и невидим когато се използва за редактиране – RepairSSelectButton

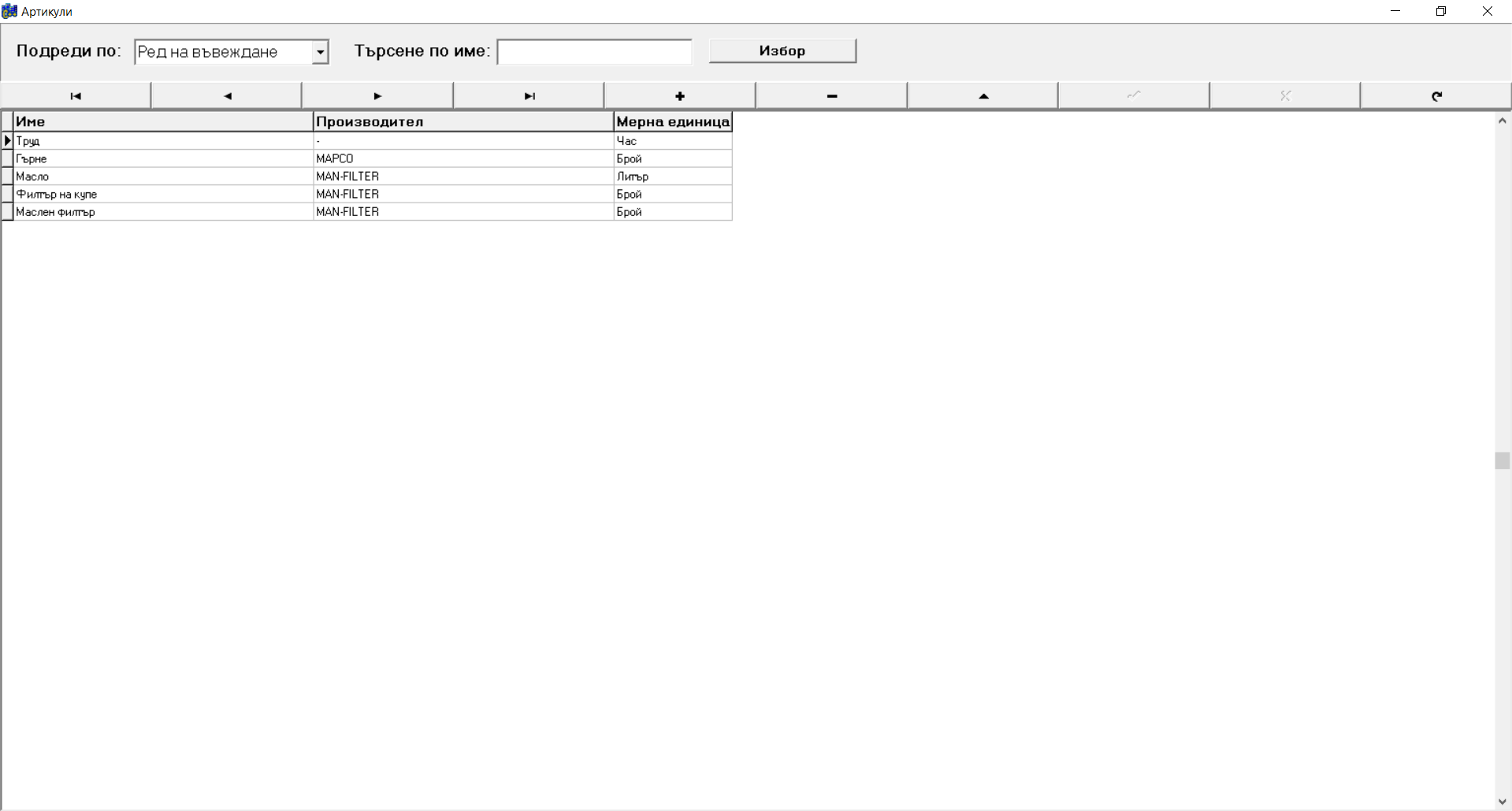
**4.12.3. Събития на форма за редактиране на сервизи RepairSForm**

Формата съдържа следните събития:

* Събитие OnExit()на всеки един DBGrid компонент – при загуба на фокус данните се запазват за да не се получава загуба на данни.
* Събитие OnChange() на EditBox компонента – настъпва, когато се промени съдържанието на полето за редактиране, за изпъняване на инкрементално търсене.
* Събитие OnSelect()на ComboBox – обработва се това събитие, вместо събитието OnClick () на съответния ComboBox, защото стилът на компонента csDropDownList, а това изисква събитие OnSelect (). Събитието настъпва, когато потребителят избере опция от падащото меню на ComboBox. Kомпонента е за смяна начина на показване на данните или критериий или по-точно променя редът на показване на записите.
* Събитие OnClick() на бутона – реализира външен ключ връзките и записва избраните данни в полетата на необходимата форма.
* Събитие OnExit() на формата – прави бутона невидим за да не се вижда при избиране на формата само за редактиране.

**4.13.** **Описание на форма за редактиране на артикули ItemForm**

**4.13.1. Цел и логика на форма за редактиране на артикули ItemForm** Целта на формата е да се редактират данни за ариткули.



**Фигура 22: Форма ItemForm**

**4.13.2. Съставни части на форма редактиране на артикули ItemForm**

Тази форма се състои от следните компоненти:

* Съдържа два компонент от клас TComboBox, които служи за указване на начина на подреждане – ItemCB
* Компонент от клас TEditBox за въвеждане на ключ за търсене. Тези компоненти са: ItemEdit
* Компонент от тип TDBNavigator за по-лесно манипулиране на данните в грид компонентата – ItemNavigator
* Компонент от клас TDBGrid за визуализиране на данните, като също така позволява да бъдат добавяни, редактирани и изтривани – ItemGrid
* Компонент от клас TButton, който е видим ако формата се извиква от друга форма и служи за избор на данни и невидим когато се използва за редактиране – ItemSelectButton

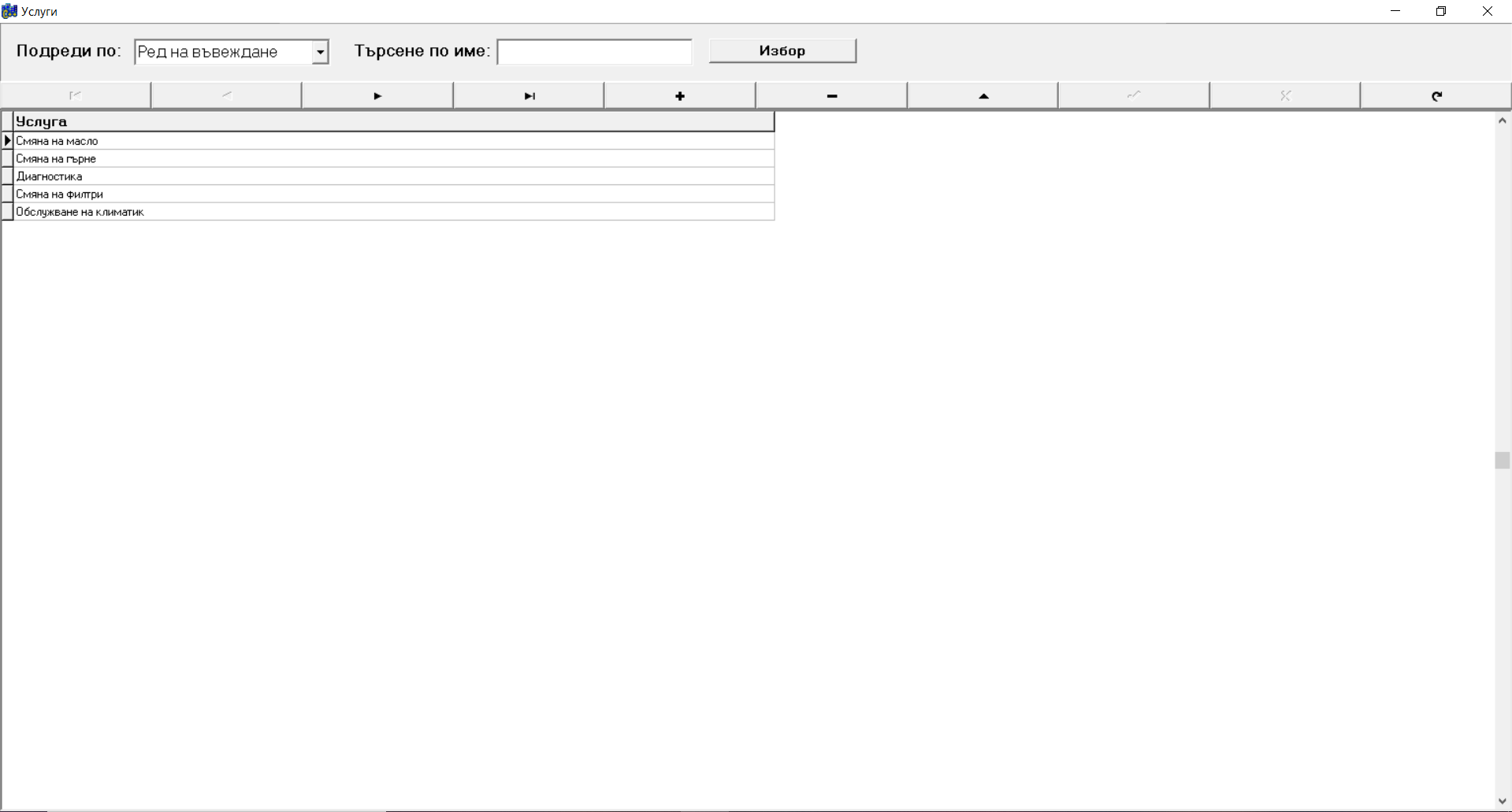
**4.13.3. Събития на форма за редактиране на артикули ItemForm**

Формата съдържа следните събития:

* Събитие OnExit()на всеки един DBGrid компонент – при загуба на фокус данните се запазват за да не се получава загуба на данни.
* Събитие OnChange() на EditBox компонента – настъпва, когато се промени съдържанието на полето за редактиране, за изпъняване на инкрементално търсене.
* Събитие OnSelect()на ComboBox – обработва се това събитие, вместо събитието OnClick () на съответния ComboBox, защото стилът на компонента csDropDownList, а това изисква събитие OnSelect (). Събитието настъпва, когато потребителят избере опция от падащото меню на ComboBox. Kомпонента е за смяна начина на показване на данните или критериий или по-точно променя редът на показване на записите.
* Събитие OnClick() на бутона – реализира външен ключ връзките и записва избраните данни в полетата на необходимата форма.
* Събитие OnExit() на формата – прави бутона невидим за да не се вижда при избиране на формата само за редактиране.

**4.14.** **Описание на форма за редактиране на услуги ServiceForm**

**4.14.1. Цел и логика на форма за редактиране на услуги ServiceForm** Целта на формата е да се редактират данни за услуги.



**Фигура 23: Форма ServiceForm**

**4.14.2. Съставни части на форма редактиране на услуги ServiceForm**

Тази форма се състои от следните компоненти:

* Съдържа два компонент от клас TComboBox, които служи за указване на начина на подреждане – ServiceCB
* Компонент от клас TEditBox за въвеждане на ключ за търсене. Тези компоненти са: ServiceEdit
* Компонент от тип TDBNavigator за по-лесно манипулиране на данните в грид компонентата – ServiceNavigator
* Компонент от клас TDBGrid за визуализиране на данните, като също така позволява да бъдат добавяни, редактирани и изтривани – ServiceGrid
* Компонент от клас TButton, който е видим ако формата се извиква от друга форма и служи за избор на данни и невидим когато се използва за редактиране – ServiceSelectButton

**4.14.3. Събития на форма за редактиране на услуги ServiceForm**

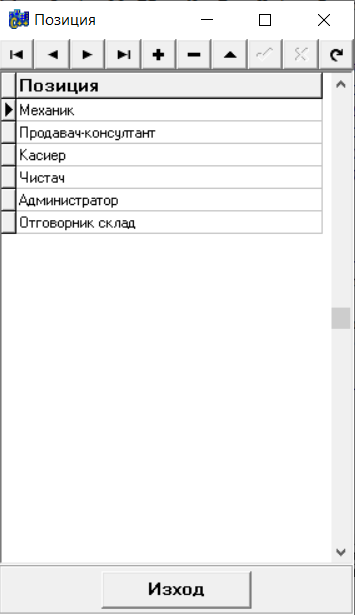
Формата съдържа следните събития:

* Събитие OnExit()на всеки един DBGrid компонент – при загуба на фокус данните се запазват за да не се получава загуба на данни.
* Събитие OnChange() на EditBox компонента – настъпва, когато се промени съдържанието на полето за редактиране, за изпъняване на инкрементално търсене.
* Събитие OnSelect()на ComboBox – обработва се това събитие, вместо събитието OnClick () на съответния ComboBox, защото стилът на компонента csDropDownList, а това изисква събитие OnSelect (). Събитието настъпва, когато потребителят избере опция от падащото меню на ComboBox. Kомпонента е за смяна начина на показване на данните или критериий или по-точно променя редът на показване на записите.
* Събитие OnClick() на бутонa – реализира външен ключ връзките и записва избраните данни в полета тана необходимата форма.
* Събитие OnExit() на формата – прави бутона невидим за да не се вижда при избиране на формата само за редактиране.

**4.15.** **Описание на форма за редактиране на позиции заемани от служители PossitionForm**

**4.15.1. Цел и логика на форма за редактиране на позиции заемани от служители PossitionForm**

Целта на формата е да се редактират данни за позиция заемана от служител.



**Фигура 24: Форма PossitionForm**

**4.15.2. Съставни части на форма редактиране на позиции заемани от служители PossitionForm**

Тази форма се състои от следните компоненти:

* Компонент от тип TDBNavigator за по-лесно манипулиране на данните в грид компонентата – PossitionNavigator
* Компонент от клас TDBGrid за визуализиране на данните, като също така позволява да бъдат добавяни, редактирани и изтривани – PossitionGrid
* Компонент от клас TButton за затваряне на формата – PossitionCloseButton

**4.15.3. Събития на форма за редактиране на позиции заемани от служители PossitionForm**

Формата съдържа следните събития:

* Събитие OnExit()на всеки един DBGrid компонент – при загуба на фокус данните се запазват за да не се получава загуба на данни.
* Събитие OnClick() на бутонa – формата се затваря.

**4.16.** **Описание на форма за редактиране на производители ManufacturerForm**

**4.16.1. Цел и логика на форма за редактиране на производители ManufacturerForm**

Целта на формата е да се редактират данни за прозиводители.



**Фигура 25: Форма ManufacturerForm**

**4.16.2. Съставни части на форма редактиране на производители ManufacturerForm**

Тази форма се състои от следните компоненти:

* Компонент от тип TDBNavigator за по-лесно манипулиране на данните в грид компонентата – ManufacturerNavigator
* Компонент от клас TDBGrid за визуализиране на данните, като също така позволява да бъдат добавяни, редактирани и изтривани – ManufacturerGrid
* Компонент от клас TButton за затваряне на формата – ManufacturerCloseButton

**4.16.3. Събития на форма за редактиране на производители ManufacturerForm**

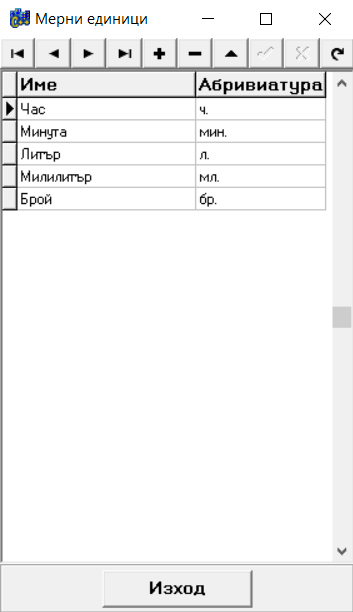
Формата съдържа следните събития:

* Събитие OnExit()на всеки един DBGrid компонент – при загуба на фокус данните се запазват за да не се получава загуба на данни.
* Събитие OnClick() на бутонa – формата се затваря.

**4.17.** **Описание на форма за редактиране на мярки MeasureForm**

**4.17.1. Цел и логика на форма за редактиране на мярки MeasureForm**

Целта на формата е да се редактират данни за мярки.



**Фигура 26: Форма MeasureForm**

**4.17.2. Съставни части на форма редактиране на мярки MeasureForm**

Тази форма се състои от следните компоненти:

* Компонент от тип TDBNavigator за по-лесно манипулиране на данните в грид компонентата – MeasureNavigator
* Компонент от клас TDBGrid за визуализиране на данните, като също така позволява да бъдат добавяни, редактирани и изтривани – MeasureGrid
* Компонент от клас TButton за затваряне на формата – MeasureCloseButton

**4.17.3. Събития на форма за редактиране на мярки MeasureForm**

Формата съдържа следните събития:

* Събитие OnExit()на всеки един DBGrid компонент – при загуба на фокус данните се запазват за да не се получава загуба на данни.
* Събитие OnClick() на бутонa – формата се затваря.

**4.18.** **Описание на форма за редактиране на марки BrandForm**

**4.18.1. Цел и логика на форма за редактиране на марки BrandForm**

Целта на формата е да се редактират данни за марки.



**Фигура 27: Форма BrandForm**

**4.18.2. Съставни части на форма редактиране на марки BrandForm**

Тази форма се състои от следните компоненти:

* Компонент от тип TDBNavigator за по-лесно манипулиране на данните в грид компонентата – BrandNavigator
* Компонент от клас TDBGrid за визуализиране на данните, като също така позволява да бъдат добавяни, редактирани и изтривани – BrandGrid
* Компонент от клас TButton за затваряне на формата – BrandCloseButton

**4.18.3. Събития на форма за редактиране на марки BrandForm**

Формата съдържа следните събития:

* Събитие OnExit()на всеки един DBGrid компонент – при загуба на фокус данните се запазват за да не се получава загуба на данни.
* Събитие OnClick() на бутонa – формата се затваря.

**4.19.** **Описание на форма за редактиране на категории CategoryForm**

**4.19.1. Цел и логика на форма за редактиране на категории CategoryForm**

Целта на формата е да се редактират данни за категории.



**Фигура 28: Форма CategoryForm**

**4.19.2. Съставни части на форма редактиране на категории CategoryForm**

Тази форма се състои от следните компоненти:

* Компонент от тип TDBNavigator за по-лесно манипулиране на данните в грид компонентата – CategoryNavigator
* Компонент от клас TDBGrid за визуализиране на данните, като също така позволява да бъдат добавяни, редактирани и изтривани – CategoryGrid
* Компонент от клас TButton за затваряне на формата – CategoryCloseButton

**4.19.3. Събития на форма за редактиране на марки категории CategoryForm**

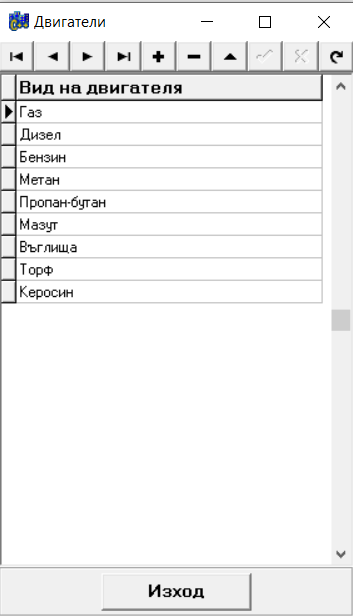
Формата съдържа следните събития:

* Събитие OnExit()на всеки един DBGrid компонент – при загуба на фокус данните се запазват за да не се получава загуба на данни.
* Събитие OnClick() на бутонa – формата се затваря.

**4.20.** **Описание на форма за редактиране на двигатели EngineForm**

**4.20.1. Цел и логика на форма за редактиране на видове двигатели EngineForm**

Целта на формата е да се редактират данни за видове двигатели.



**Фигура 29: Форма EngineForm**

**4.20.2. Съставни части на форма редактиране на видове двигатели EngineForm**

Тази форма се състои от следните компоненти:

* Компонент от тип TDBNavigator за по-лесно манипулиране на данните в грид компонентата – EngineNavigator
* Компонент от клас TDBGrid за визуализиране на данните, като също така позволява да бъдат добавяни, редактирани и изтривани – EngineGrid
* Компонент от клас TButton за затваряне на формата – EngineCloseButton

**4.20.3. Събития на форма за редактиране на видове двигатели EngineForm**

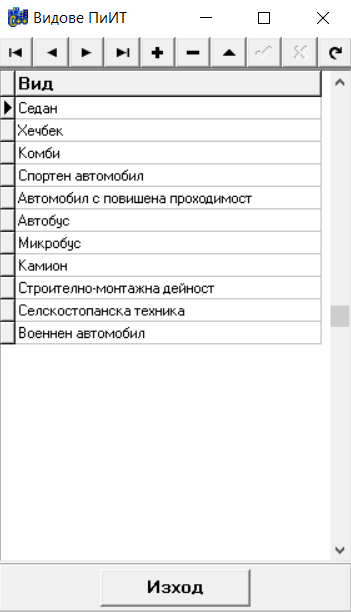
Формата съдържа следните събития:

* Събитие OnExit()на всеки един DBGrid компонент – при загуба на фокус данните се запазват за да не се получава загуба на данни.
* Събитие OnClick() на бутонa – формата се затваря.

**4.21.** **Описание на форма за редактиране на видове пътна и извънпътна техника TypeForm**

**4.21.1. Цел и логика на форма за редактиране на видове пътна и извънпътна техника TypeForm**

Целта на формата е да се редактират данни за видове пътна и извънпътна техника.



**Фигура 30: Форма TypeForm**

**4.21.2. Съставни части на форма редактиране на видове пътна и извънпътна техника TypeForm**

Тази форма се състои от следните компоненти:

* Компонент от тип TDBNavigator за по-лесно манипулиране на данните в грид компонентата – TypeNavigator
* Компонент от клас TDBGrid за визуализиране на данните, като също така позволява да бъдат добавяни, редактирани и изтривани – TypeGrid
* Компонент от клас TButton за затваряне на формата – TypeCloseButton

**4.21.3. Събития на форма за редактиране на видове пътна и извънпътна техника TypeForm**

Формата съдържа следните събития:

* Събитие OnExit()на всеки един DBGrid компонент – при загуба на фокус данните се запазват за да не се получава загуба на данни.
* Събитие OnClick() на бутонa – формата се затваря.

**Глава 5**

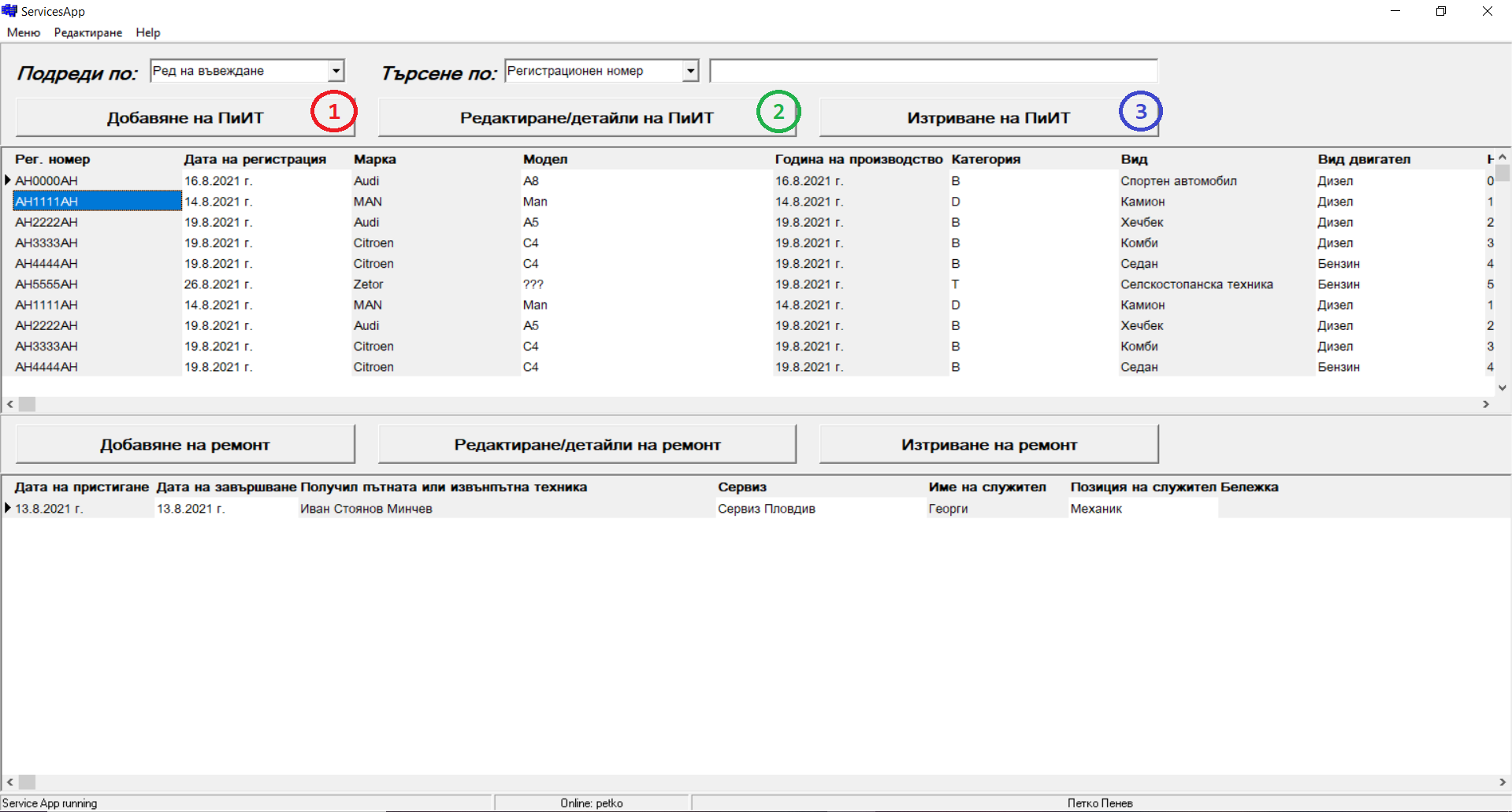
**Ръководство за потребителя**

**5.1. Вписване в приложението**

1. Натиска се иконата за стартиране на приложението за да се отвари формата за вписване.
2. След отварянето на формата се въвеждат в полетата потребителското име и парола.
3. След въвеждане на правилно име и парола се натиска бутона „Вход“ за да се достъпи главната форма на приложението.

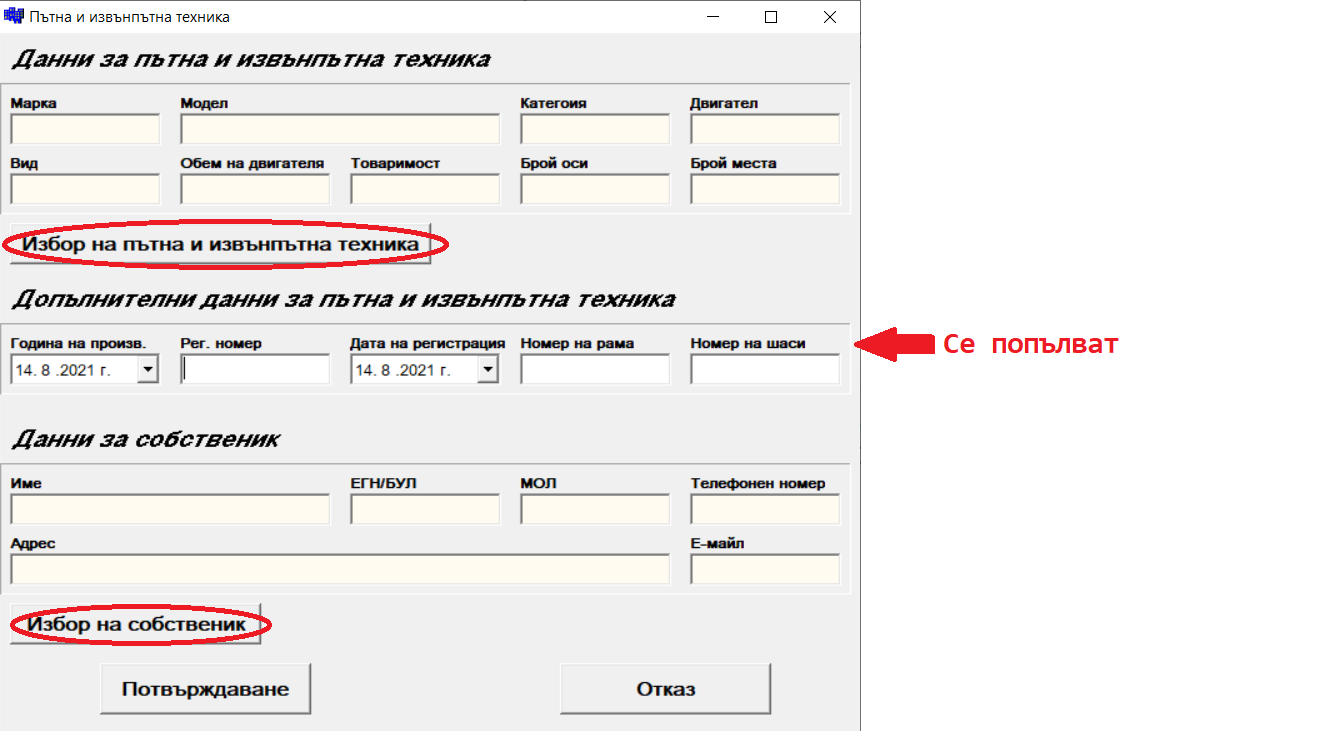
**5.2. Добавяне, изтриване и редактиране на пътна или извънпътна техника**

**Фигура 31**



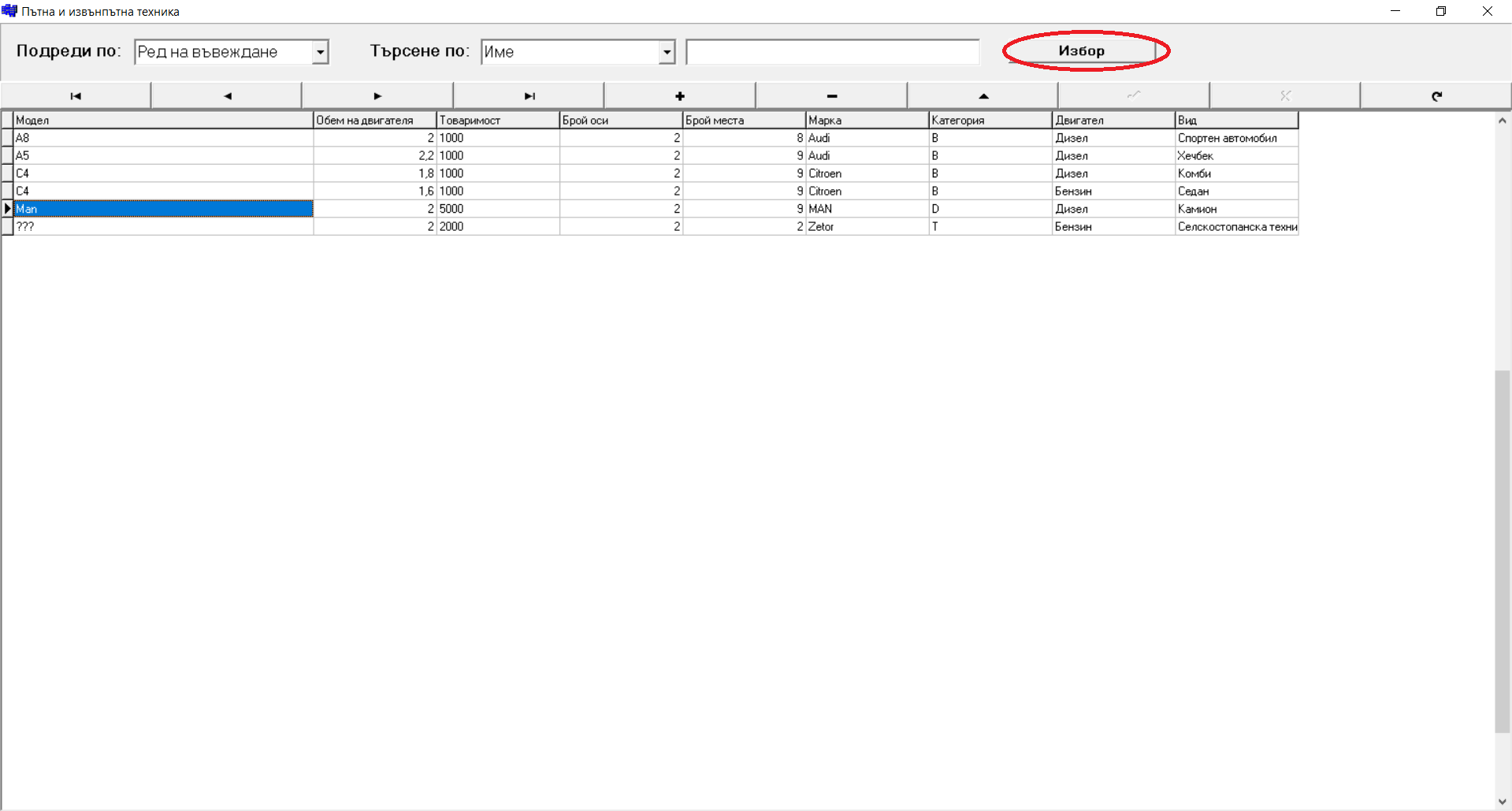
**5.3. Добавяне на пътна или извънпътна техника**

За да се добави нова пътна или извънпътна техника се следват следните стъпки (ПиИТ е съкращение на Пътна или Извънпътна Техника):

1. От главната форма се избира бутона „Добавяне на ПиИТ“ форма (фигура 31 означен с **1**)
2. В новата отворена форма се попъват всички полета в бял цвят, а всички в жълтиникъв се избират чрез натискане на съответния бутон (Фигура 32)

**Фигура 32**

1. След избор на бутон „Избор на пътна или извънпътна техника“ или „Избор на собственик“ се отваря съответната нова форма, от която се маркира полето което искаме да запишем. Полето се маркира чрез кликване, бутоните „↓“ (стелка надолу или нагоре) или със помощта на бутоните от навигацията.
2. Натискаме бутон „Избор“ за да изберем маркираното поле и запишем данните(Фигура 33)



**Фигура 33**

1. Ако необходимите данни не фигурират или трябва да бъдат редактирани:
2. Те се добавят като се натисне бутона „+“ от навигатора или със „↓“ (стелка надолу) се отвори ново поле за попълване.
3. За да се потвърди добавянето се натиска „↓“ (стелка надолу) или от навигатора „✓“.
4. За да се редактира поле се клкива два пъти върху него (double click), променя се стойността на желаната такава и се натиска бутона от навигатора от навигатора „✓“.
5. След попълване на всички данни във формата за попълване (фигура 32) се натиска бутона „Потвърждаване“

**5.4. Редактиране на пътна или извънпътна техника**

За да се редактира пътна или извънпътна техника се следват следните стъпки:

1. Маркира се пътна или извънпътна техника която ще се редактира. Полето се маркира чрез кликване, бутоните „↓“ (стелка надолу или нагоре) или със помощта на бутоните от навигацията.
2. От главната форма се избира бутона „Добавяне на ПиИТ“ форма (фигура 31 означен с **2**)
3. Следващите стъпки са аналогични на стъпки 2 до 6 за добавяне на пътна или извънпътна техника

**5.5. Изтриване на пътна или извънпътна техника**

**За изтриване на пътна или извънпътна техника е необходимо да няма записани рамонти за тях или ако има, те да се изтрият!** За да се изтрие пътна или извънпътна техника се следват следните стъпки:

1. Маркира се пътна или извънпътна техника която ще се редактира. Полето се маркира чрез кликване, бутоните „↓“ (стелка надолу или нагоре) или със помощта на бутоните от навигацията.
2. От главната форма се избира бутона „Добавяне на ПиИТ“ форма (фигура 31 означен с **3**)

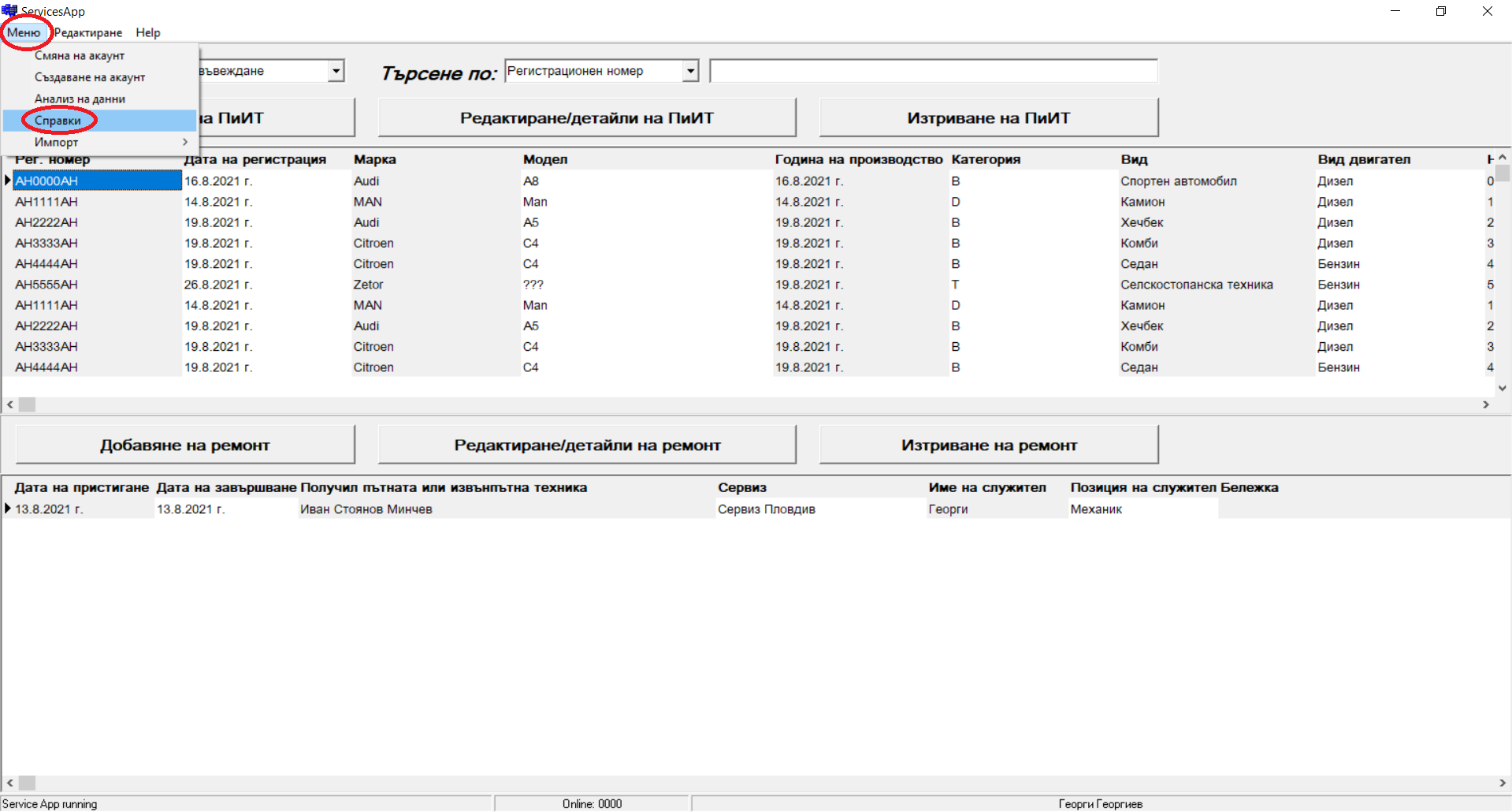
**5.6. Добавяне, изтриване и редактиране на ремонт**

Стъпките за добавяне, изтриване и редактиране на ремонт са аналогични на стъпките за добавяне, изтриване и редактиране на пътна или извънпътна техника използвайки бутоните „Добавяне на ремонт“, „Изтриване на ремонт“ и „Редактиране/детайли на ремонт“. **При изтриване на ремонт данните за общата сума на дадения ремонт ще се загуби!**

**5.7. Съставяне на справка**

За да се състави справка се следват следните стъпки:

**Фигура 34**



1. От главното меню се избира „Меню“.
2. От падащото меню се избира „Справки“ (Фигура 34)
3. Избират се параметрите на справката.
4. Попълват се необходимите полета.
5. Избират се от правилнте дати.
6. Избира се дали да се показва справка за части или за ремонти.
7. Натиска се бутон „Показване“
8. Може справката да се експортне до файл с разширение .CSV.
9. Избира се бутона „Експорт“.
10. Избира се място за експорт.
11. Въвежда се име на файла.
12. Натиска се бутона “Save“ за завършване на експорта.

**5.8. Анализ**

Достъпване и ползване на форма за анализ:

1. От падащото меню показано на фигура 34 се избира „Анализ“.
2. Могат да се използват набора от бутони и полета за филтриране на даните.
3. Чрез бутоните „Графика“ се показва графика за данните над съответния бутон.

**5.9. Създаване на акаунт**

Достъпване и ползване на форма за създаване на акаунт:

1. От падащото меню показано на фигура 34 се избира „Анализ“.
2. Натиска се бутона „Избор на служител“ за да се избере служител на когото ще бъде направен акаунт.
3. Следват се стъпки 4-6 за добавяне на пътна или извънпътна техника.
4. Попълват се в полетата желаните потребителско име и парола.
5. Натиска се бутона „Потвърждаване“ за да се потвърди регистрацията.

**5.10. Таблици за редактиране**

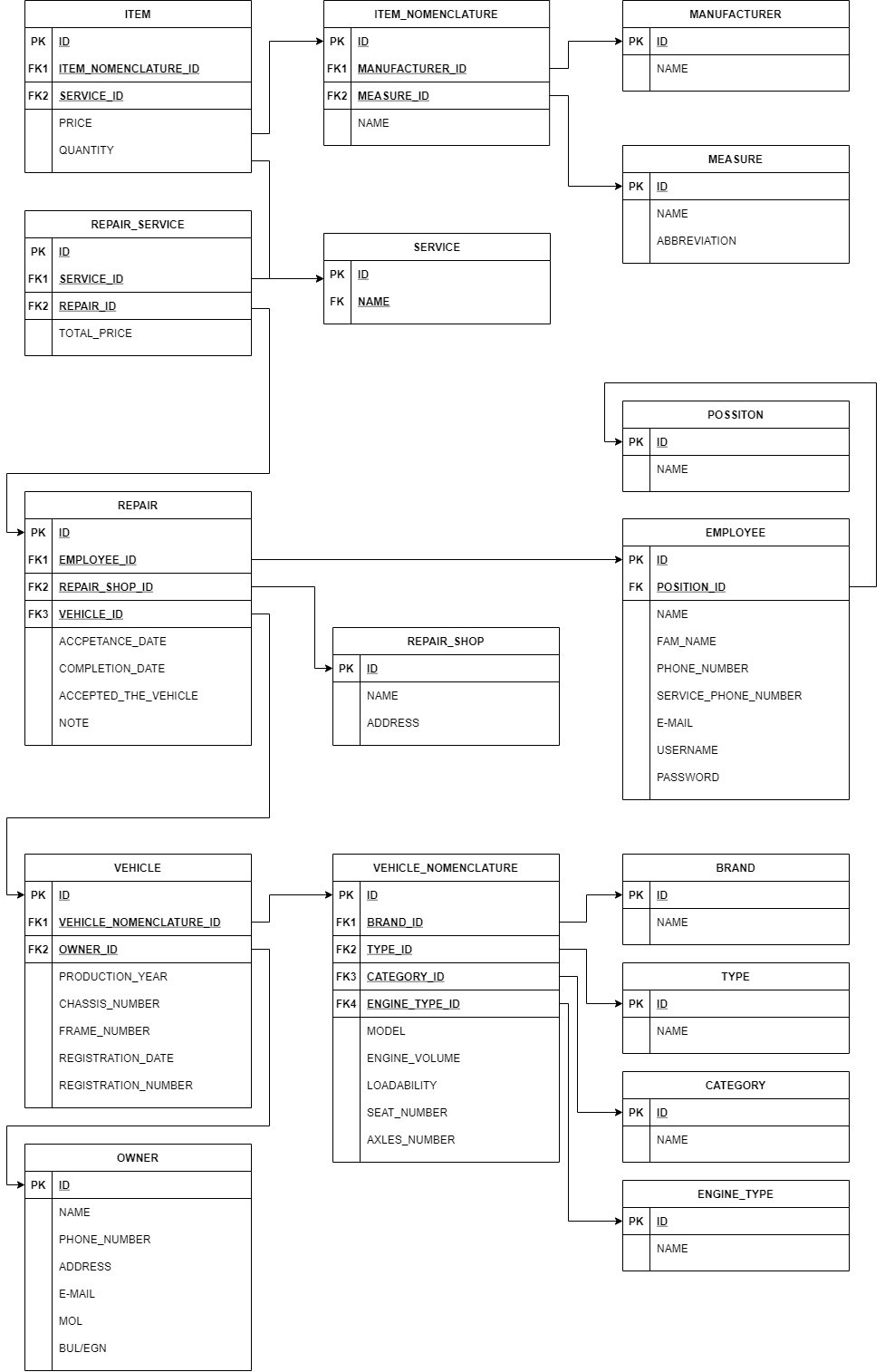
Всички таблици за редактиране следват една и съща логика. Това са таблиците „Пътна и извънпътна техника“, „Собственици“, „Сервизи“, „Служители“, „Позиции на служителите“, „Части“, „Производител“, „Размери“, „Услуги“, „Марки“, „Категории“, „Видове двигатели“ и „Видове пътна и извънпътна техника“.

1. От главното меню се избира „Редактиране“.
2. От падащото меню се избира желаната таблица за редактиране.
3. За добавяне или редактиране на данните се следва точка 5 за добавяне на пътна или извънпътна техника.
4. За изтриване на данни се натиска от навигатора бутона „-„. **За да се изтрият данни в която и да е таблица е необходимо те да не бъдат използвани.**
5. При приключване на редакцията прозореца за редактиране на дадената таблица може просто да се затвори.

**Използвана литература**

* Документация на машина за бази данни InterBase - http://docs.embarcadero.com/products/interbase/
* Hollingworth J., Borland C++ Builder 6 Developers Guide, Sams Publishing, 2002, ISBN 0-672-32480-6
* Calvert C. & Phillips J., C++ Builder 3 Unleashed, Sams Publishing, 1998, ISBN0-672-312

**Приложение: структура на базата данни**



**Приложение: SQL script**

//---------------------CREATE TABLE----------------------------------------

CREATE TABLE OWNER

(

ID INT NOT NULL,

NAME VARCHAR(64) ,

PHONE\_NUMBER DECIMAL(12) ,

ADDRESS VARCHAR(256),

E\_MAIL VARCHAR(64) ,

MOL VARCHAR(64) ,

BUL\_EGN VARCHAR(12) NOT NULL,

PRIMARY KEY (ID)

);

CREATE TABLE BRAND

(

ID INT NOT NULL,

NAME VARCHAR(32) ,

PRIMARY KEY (ID)

);

CREATE TABLE TYPE\_VEHICLE

(

ID INT NOT NULL,

NAME VARCHAR(32) ,

PRIMARY KEY (ID)

);

CREATE TABLE ENGINE\_TYPE

(

ID INT NOT NULL,

NAME VARCHAR(32) ,

PRIMARY KEY (ID)

);

CREATE TABLE CATEGORY

(

ID INT NOT NULL,

NAME VARCHAR(32) ,

PRIMARY KEY (ID)

);

CREATE TABLE VEHICLE\_NOMENCLATURE (

ID INT NOT NULL,

BRAND\_ID INT,

MODEL VARCHAR(64) ,

CATEGORY\_ID INT,

ENGINE\_TYPE\_ID INT,

TYPE\_VEHICLE\_ID INT,

ENGINE\_VOLUME DECIMAL(4,1),

LOADABILITY VARCHAR(32),

AXLES\_NUMBER INT,

SEATS\_NUMBER INT,

PRIMARY KEY (ID),

FOREIGN KEY (BRAND\_ID) REFERENCES BRAND(ID),

FOREIGN KEY (TYPE\_VEHICLE\_ID) REFERENCES TYPE\_VEHICLE (ID),

FOREIGN KEY (ENGINE\_TYPE\_ID) REFERENCES ENGINE\_TYPE (ID),

FOREIGN KEY (CATEGORY\_ID) REFERENCES CATEGORY (ID)

);

CREATE TABLE VEHICLE

(

ID INT NOT NULL,

PRODUCTION\_YEAR DATE,

OWNER\_ID INT NOT NULL,

VEHICLE\_NOMENCLATURE\_ID INT,

REGISTRATION\_NUMBER VARCHAR(16),

CHASSIS\_NUMBER VARCHAR(64) ,

FRAME\_NUMBER VARCHAR(64) ,

REGISTRATION\_DATE DATE,

PRIMARY KEY (ID),

FOREIGN KEY (OWNER\_ID) REFERENCES OWNER(ID),

FOREIGN KEY (VEHICLE\_NOMENCLATURE\_ID) REFERENCES VEHICLE\_NOMENCLATURE(ID)

);

CREATE TABLE REPAIR\_SHOP

(

ID INT NOT NULL,

NAME VARCHAR(128),

ADDRESS VARCHAR(256),

PRIMARY KEY (ID)

);

CREATE TABLE POSSITION

(

ID INT NOT NULL,

NAME VARCHAR(64),

PRIMARY KEY (ID)

);

CREATE TABLE EMPLOYEE

(

ID INT NOT NULL,

FIRST\_NAME VARCHAR(32),

FAM\_NAME VARCHAR(32),

PHONE\_NUMBER DECIMAL(12),

SERVICE\_PHONE\_NUMBER VARCHAR(12),

E\_MAIL VARCHAR(64),

POSSITION\_ID INT,

USER\_PASSWORD VARCHAR(32),

USERNAME VARCHAR(32),

PRIMARY KEY (ID),

FOREIGN KEY (POSSITION\_ID) REFERENCES POSSITION(ID)

);

CREATE TABLE REPAIR

(

ID INT NOT NULL,

ACCPETANCE\_DATE DATE ,

COMPLETION\_DATE DATE ,

EMPLOYEE\_ID INT ,

REPAIR\_SHOP\_ID INT ,

VEHICLE\_ID INT ,

ACCEPTED\_THE\_VEHICLE VARCHAR(128),

NOTE VARCHAR(256),

PRIMARY KEY (ID),

FOREIGN KEY (REPAIR\_SHOP\_ID) REFERENCES REPAIR\_SHOP(ID),

FOREIGN KEY (VEHICLE\_ID) REFERENCES VEHICLE(ID),

FOREIGN KEY (EMPLOYEE\_ID) REFERENCES EMPLOYEE(ID)

);

CREATE TABLE SERVICE

(

ID INT NOT NULL,

NAME VARCHAR(128),

PRIMARY KEY (ID)

);

CREATE TABLE REPAIR\_SERVICE

(

ID INT NOT NULL,

REPAIR\_ID INT,

SERVICE\_ID INT,

TOTAL\_PRICE DECIMAL(16,2),

PRIMARY KEY (ID),

FOREIGN KEY (REPAIR\_ID) REFERENCES REPAIR(ID),

FOREIGN KEY (SERVICE\_ID) REFERENCES SERVICE(ID)

);

CREATE TABLE MEASURE (

ID INT NOT NULL,

NAME VARCHAR(64),

ABBREVIATION VARCHAR(16) ,

PRIMARY KEY (ID)

);

CREATE TABLE MANUFACTURER

(

ID INT NOT NULL,

NAME VARCHAR(64),

PRIMARY KEY (ID)

);

CREATE TABLE ITEM\_NOMENCLATURE

(

ID INT NOT NULL,

NAME VARCHAR(64),

MANUFACTURER\_ID INT,

MEASURE\_ID INT,

PRIMARY KEY (ID),

FOREIGN KEY (MEASURE\_ID) REFERENCES MEASURE(ID),

FOREIGN KEY (MANUFACTURER\_ID) REFERENCES MANUFACTURER(ID)

);

CREATE TABLE ITEM

(

ID INT NOT NULL,

PRICE DECIMAL(16,2),

QUANTITY DECIMAL(8,2),

ITEM\_NOMENCLATURE\_ID INT,

SERVICE\_ID INT,

PRIMARY KEY (ID),

FOREIGN KEY (SERVICE\_ID) REFERENCES SERVICE(ID),

FOREIGN KEY (ITEM\_NOMENCLATURE\_ID) REFERENCES ITEM\_NOMENCLATURE(ID)

)

//------------------------INDEXES------------------------------------------

CREATE INDEX EMPLOYEETBLFNNDX ON EMPLOYEE(FIRST\_NAME);

CREATE INDEX EMPLOYEETBLFAMNNDX ON EMPLOYEE(FAM\_NAME);

CREATE INDEX EMPLOYEETBLPNNDX ON EMPLOYEE(PHONE\_NUMBER);

CREATE INDEX EMPLOYEETBLSPNNDX ON EMPLOYEE(SERVICE\_PHONE\_NUMBER);

CREATE INDEX EMPLOYEETBLEMNDX ON EMPLOYEE(E\_MAIL);

CREATE INDEX OWNENAMERNDX ON OWNER(NAME);

CREATE INDEX OWNERPNNDX ON OWNER(PHONE\_NUMBER);

CREATE INDEX OWNERADDNDX ON OWNER(ADDRESS);

CREATE INDEX OWNEREMNDX ON OWNER(E\_MAIL);

CREATE INDEX OWNERMOLNDX ON OWNER(MOL);

CREATE INDEX OWNERBENDX ON OWNER(BUL\_EGN);

CREATE INDEX ITEMNOMTBLNAMENDX ON ITEM\_NOMENCLATURE(NAME);

CREATE INDEX SERVICETBLNAMENDX ON SERVICE(NAME);

CREATE INDEX VEHICLERDNDX ON VEHICLE(REGISTRATION\_NUMBER);

CREATE INDEX VEHICLEPYNDX ON VEHICLE(PRODUCTION\_YEAR);

CREATE INDEX VEHICLECNNDX ON VEHICLE(CHASSIS\_NUMBER);

CREATE INDEX VEHICLEFNNDX ON VEHICLE(FRAME\_NUMBER);

CREATE INDEX VEHICLERDATENDX ON VEHICLE(REGISTRATION\_DATE);

CREATE INDEX VEHNOMMODELNDX ON VEHICLE\_NOMENCLATURE(MODEL);

CREATE INDEX VEHNOMENGINEVOLNDX ON VEHICLE\_NOMENCLATURE(ENGINE\_VOLUME);

CREATE INDEX VEHNOMLOADNDX ON VEHICLE\_NOMENCLATURE(LOADABILITY);

CREATE INDEX VEHNOMAXLESNDX ON VEHICLE\_NOMENCLATURE(AXLES\_NUMBER);

CREATE INDEX VEHNOMSEATSNDX ON VEHICLE\_NOMENCLATURE(SEATS\_NUMBER);

CREATE INDEX RSHOPNAMENDX ON REPAIR\_SHOP(NAME);

CREATE INDEX ITEMNAMENDX ON ITEM\_NOMENCLATURE(NAME);

//---------------GENERATORS AND STORED PROCEDURES--------------------------

CREATE GENERATOR VEHICLETBL\_IDGEN;

SET GENERATOR VEHICLETBL\_IDGEN TO 0;

SET TERM ^;

CREATE PROCEDURE VEHICLETBL\_IDSP

RETURNS(ID INTEGER)

AS

BEGIN

ID = GEN\_ID(VEHICLETBL\_IDGEN,1);

END^

CREATE GENERATOR REPAIRTBL\_IDGEN;

SET GENERATOR REPAIRTBL\_IDGEN TO 0;

SET TERM ^;

CREATE PROCEDURE REPAIRTBL\_IDSP

RETURNS(ID INTEGER)

AS

BEGIN

ID = GEN\_ID(REPAIRTBL\_IDGEN,1);

END^

CREATE GENERATOR RSHOPTBL\_IDGEN;

SET GENERATOR RSHOPTBL\_IDGEN TO 0;

SET TERM ^;

CREATE PROCEDURE RSHOPTBL\_IDSP

RETURNS(ID INTEGER)

AS

BEGIN

ID = GEN\_ID(RSHOPTBL\_IDGEN,1);

END^

CREATE GENERATOR SERVICETBL\_IDGEN;

SET GENERATOR SERVICETBL\_IDGEN TO 0;

SET TERM ^;

CREATE PROCEDURE SERVICETBL\_IDSP

RETURNS(ID INTEGER)

AS

BEGIN

ID = GEN\_ID(SERVICETBL\_IDGEN,1);

END^

CREATE GENERATOR REPAIRSERVICETBL\_IDGEN;

SET GENERATOR REPAIRSERVICETBL\_IDGEN TO 0;

SET TERM ^;

CREATE PROCEDURE REPAIRSERVICETBL\_IDSP

RETURNS(ID INTEGER)

AS

BEGIN

ID = GEN\_ID(REPAIRSERVICETBL\_IDGEN,1);

END^

CREATE GENERATOR EMPLOYEETBL\_IDGEN;

SET GENERATOR EMPLOYEETBL\_IDGEN TO 0;

SET TERM ^;

CREATE PROCEDURE EMPLOYEETBL\_IDSP

RETURNS(ID INTEGER)

AS

BEGIN

ID = GEN\_ID(EMPLOYEETBL\_IDGEN,1);

END^

CREATE GENERATOR OWNERTBL\_IDGEN;

SET GENERATOR OWNERTBL\_IDGEN TO 0;

SET TERM ^;

CREATE PROCEDURE OWNERTBL\_IDSP

RETURNS(ID INTEGER)

AS

BEGIN

ID = GEN\_ID(OWNERTBL\_IDGEN,1);

END^

CREATE GENERATOR VEHICLENOMTBL\_IDGEN;

SET GENERATOR VEHICLENOMTBL\_IDGEN TO 0;

SET TERM ^;

CREATE PROCEDURE VEHICLENOMTBL\_IDSP

RETURNS(ID INTEGER)

AS

BEGIN

ID = GEN\_ID(VEHICLENOMTBL\_IDGEN,1);

END^

CREATE GENERATOR POSSITIONTBL\_IDGEN;

SET GENERATOR POSSITIONTBL\_IDGEN TO 0;

SET TERM ^;

CREATE PROCEDURE POSSITIONTBL\_IDSP

RETURNS(ID INTEGER)

AS

BEGIN

ID = GEN\_ID(POSSITIONTBL\_IDGEN,1);

END^

CREATE GENERATOR CATEGORYTBL\_IDGEN;

SET GENERATOR CATEGORYTBL\_IDGEN TO 0;

SET TERM ^;

CREATE PROCEDURE CATEGORYTBL\_IDSP

RETURNS(ID INTEGER)

AS

BEGIN

ID = GEN\_ID(CATEGORYTBL\_IDGEN,1);

END^

CREATE GENERATOR BRANDTBL\_IDGEN;

SET GENERATOR BRANDTBL\_IDGEN TO 0;

SET TERM ^;

CREATE PROCEDURE BRANDTBL\_IDSP

RETURNS(ID INTEGER)

AS

BEGIN

ID = GEN\_ID(BRANDTBL\_IDGEN,1);

END^

CREATE GENERATOR ENGINETYPETBL\_IDGEN;

SET GENERATOR ENGINETYPETBL\_IDGEN TO 0;

SET TERM ^;

CREATE PROCEDURE ENGINETYPETBL\_IDSP

RETURNS(ID INTEGER)

AS

BEGIN

ID = GEN\_ID(ENGINETYPETBL\_IDGEN,1);

END^

CREATE GENERATOR ITEMTBL\_IDGEN;

SET GENERATOR ITEMTBL\_IDGEN TO 0;

SET TERM ^;

CREATE PROCEDURE ITEMTBL\_IDSP

RETURNS(ID INTEGER)

AS

BEGIN

ID = GEN\_ID(ITEMTBL\_IDGEN,1);

END^

CREATE GENERATOR ITEMNOMTBL\_IDGEN;

SET GENERATOR ITEMNOMTBL\_IDGEN TO 0;

SET TERM ^;

CREATE PROCEDURE ITEMNOMTBL\_IDSP

RETURNS(ID INTEGER)

AS

BEGIN

ID = GEN\_ID(ITEMNOMTBL\_IDGEN,1);

END^

CREATE GENERATOR MESURETBL\_IDGEN;

SET GENERATOR MESURETBL\_IDGEN TO 0;

SET TERM ^;

CREATE PROCEDURE MESURETBL\_IDSP

RETURNS(ID INTEGER)

AS

BEGIN

ID = GEN\_ID(MESURETBL\_IDGEN,1);

END^

CREATE GENERATOR MANUFACTURERTBL\_IDGEN;

SET GENERATOR MANUFACTURERTBL\_IDGEN TO 0;

SET TERM ^;

CREATE PROCEDURE MANUFACTURERTBL\_IDSP

RETURNS(ID INTEGER)

AS

BEGIN

ID = GEN\_ID(MANUFACTURERTBL\_IDGEN,1);

END^

CREATE GENERATOR VEHICLETYPETBL\_IDGEN;

SET GENERATOR VEHICLETYPETBL\_IDGEN TO 0;

SET TERM ^;

CREATE PROCEDURE VEHICLETYPETBL\_IDSP

RETURNS(ID INTEGER)

AS

BEGIN

ID = GEN\_ID(VEHICLETYPETBL\_IDGEN,1);

END^

**Приложение: Source code**

**DataModuleUnit.h**

#ifndef DataModuleUnitH

#define DataModuleUnitH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <DB.hpp>

#include <IBCustomDataSet.hpp>

#include <IBDatabase.hpp>

#include <IBTable.hpp>

#include <Menus.hpp>

#include <IBStoredProc.hpp>

#include <IBQuery.hpp>

//---------------------------------------------------------------------------

class TDM : public TDataModule

{

\_\_published: // IDE-managed Components

TIBDatabase \*DB;

TIBTransaction \*Transaction;

TIBTable \*RepairTBL;

TDataSource \*RepairDS;

TMainMenu \*MainMenu;

TMenuItem \*ImportExport1;

TMenuItem \*Export1;

TMenuItem \*Help1;

TMenuItem \*About1;

TIBTable \*EmployeeTBL;

TDataSource \*EmployeeDS;

TIBTable \*RShopTBL;

TDataSource \*RShopDS;

TIBTable \*VehicleTBL;

TDataSource \*VehicleDS;

TMenuItem \*Open1;

TMenuItem \*EmployeesTBL1;

TMenuItem \*OwnersTBL1;

TMenuItem \*PartsTBL1;

TMenuItem \*ServicesTBL1;

TIntegerField \*EmployeeTBLID;

TIBStringField \*EmployeeTBLFIRST\_NAME;

TIBStringField \*EmployeeTBLFAM\_NAME;

TIBStringField \*EmployeeTBLSERVICE\_PHONE\_NUMBER;

TIBStringField \*EmployeeTBLE\_MAIL;

TIntegerField \*EmployeeTBLPOSSITION\_ID;

TIBStringField \*EmployeeTBLUSER\_PASSWORD;

TIBStringField \*EmployeeTBLUSERNAME;

TIBTable \*PossitionTBL;

TDataSource \*PossitionDS;

TStringField \*EmployeeTBLPOSSITION;

TIBTable \*VehicleNomTBL;

TDataSource \*VehicleNomDS;

TIBTable \*OwnerTBL;

TDataSource \*OwnerDS;

TIBTable \*BrandTBL;

TIBTable \*RepairServiceTBL;

TDataSource \*RepairServiceDS;

TDataSource \*BrandDS;

TDataSource \*ServiceDS;

TDataSource \*CategoryDS;

TDataSource \*EngineTypeDS;

TIBTable \*ServiceTBL;

TIBTable \*CategoryTBL;

TIBTable \*EngineTypeTBL;

TIBTable \*ItemTBL;

TIBTable \*ItemNomTBL;

TIBTable \*ManufacturerTBL;

TIBTable \*MesureTBL;

TIBTable \*VehicleTypeTBL;

TDataSource \*ItemDS;

TDataSource \*MesureDS;

TDataSource \*VehicleTypeDS;

TDataSource \*ItemNomDS;

TDataSource \*ManufacturerDS;

TIntegerField \*VehicleTBLID;

TIntegerField \*VehicleTBLOWNER\_ID;

TIntegerField \*VehicleTBLVEHICLE\_NOMENCLATURE\_ID;

TIBStringField \*VehicleTBLCHASSIS\_NUMBER;

TIBStringField \*VehicleTBLFRAME\_NUMBER;

TIntegerField \*VehicleNomTBLID;

TIntegerField \*VehicleNomTBLBRAND\_ID;

TIBStringField \*VehicleNomTBLMODEL;

TIntegerField \*VehicleNomTBLCATEGORY\_ID;

TIntegerField \*VehicleNomTBLENGINE\_TYPE\_ID;

TIntegerField \*VehicleNomTBLTYPE\_VEHICLE\_ID;

TIBBCDField \*VehicleNomTBLENGINE\_VOLUME;

TIBStringField \*VehicleNomTBLLOADABILITY;

TIntegerField \*VehicleNomTBLAXLES\_NUMBER;

TIntegerField \*VehicleNomTBLSEATS\_NUMBER;

TStringField \*VehicleNomTBLBRAND\_NAME;

TStringField \*VehicleNomTBLCATEGORY\_NAME;

TStringField \*VehicleNomTBLENGINE\_NAME;

TStringField \*VehicleNomTBLTYPE\_NAME;

TStringField \*VehicleTBLOWNER\_NAME;

TStringField \*VehicleTBLVEHICLE\_BRAND\_NAME;

TStringField \*VehicleTBLVEHICLE\_MODEL\_NAME;

TStringField \*VehicleTBLVEHICLE\_CATEGORY\_NAME;

TStringField \*VehicleTBLVEHICLE\_ENGINE\_NAME;

TStringField \*VehicleTBLVEHICLE\_TYPE\_NAME;

TMenuItem \*BrandTBL1;

TMenuItem \*N1;

TMenuItem \*N2;

TMenuItem \*N3;

TMenuItem \*N4;

TMenuItem \*N5;

TMenuItem \*N6;

TIntegerField \*OwnerTBLID;

TIBStringField \*OwnerTBLNAME;

TLargeintField \*OwnerTBLPHONE\_NUMBER;

TIBStringField \*OwnerTBLADDRESS;

TIBStringField \*OwnerTBLE\_MAIL;

TIBStringField \*OwnerTBLMOL;

TIBStringField \*OwnerTBLBUL\_EGN;

TIntegerField \*ItemNomTBLID;

TIBStringField \*ItemNomTBLNAME;

TIntegerField \*ItemNomTBLMANUFACTURER\_ID;

TIntegerField \*ItemNomTBLMEASURE\_ID;

TStringField \*ItemNomTBLMANUFACTURER\_NAME;

TStringField \*ItemNomTBLMESURE\_NAME;

TIntegerField \*ServiceTBLID;

TIBStringField \*ServiceTBLNAME;

TIntegerField \*BrandTBLID;

TIBStringField \*BrandTBLNAME;

TIntegerField \*VehicleTypeTBLID;

TIBStringField \*VehicleTypeTBLNAME;

TIntegerField \*EngineTypeTBLID;

TIBStringField \*EngineTypeTBLNAME;

TIntegerField \*ManufacturerTBLID;

TIBStringField \*ManufacturerTBLNAME;

TIntegerField \*MesureTBLID;

TIBStringField \*MesureTBLNAME;

TIBStringField \*MesureTBLABBREVIATION;

TIntegerField \*PossitionTBLID;

TIBStringField \*PossitionTBLNAME;

TIBStoredProc \*VehicleSP;

TMenuItem \*N7;

TIBStoredProc \*RepairSP;

TIBStoredProc \*RShopSP;

TIBStoredProc \*ServiceSP;

TIBStoredProc \*RepairServiceSP;

TIBStoredProc \*EmployeeSP;

TIBStoredProc \*OwnerSP;

TIBStoredProc \*VehicleNomSP;

TIBStoredProc \*PossitionSP;

TIBStoredProc \*CategorySP;

TIBStoredProc \*BrandSP;

TIBStoredProc \*EngineTypeSP;

TIBStoredProc \*ItemSP;

TIBStoredProc \*ItemNomSP;

TIBStoredProc \*MesureSP;

TIBStoredProc \*ManufacturerSP;

TIBStoredProc \*VehicleTypeSP;

TIntegerField \*RepairServiceTBLID;

TIntegerField \*RepairServiceTBLREPAIR\_ID;

TIntegerField \*RepairServiceTBLSERVICE\_ID;

TIBBCDField \*RepairServiceTBLTOTAL\_PRICE;

TIntegerField \*ItemTBLID;

TIBBCDField \*ItemTBLPRICE;

TIBBCDField \*ItemTBLQUANTITY;

TIntegerField \*ItemTBLITEM\_NOMENCLATURE\_ID;

TIntegerField \*ItemTBLSERVICE\_ID;

TStringField \*ItemTBLMANUFACTURER\_NAME\_ITEM;

TStringField \*ItemTBLMEASURE\_NAME\_ITEM;

TMenuItem \*N8;

TDataSource \*AllServicesDS;

TIBTable \*AllServiceTBL;

TStringField \*RepairServiceTBLSERVICE;

TIntegerField \*RepairTBLID;

TIBStringField \*RepairTBLNOTE;

TDateField \*RepairTBLACCPETANCE\_DATE;

TDateField \*RepairTBLCOMPLETION\_DATE;

TIntegerField \*RepairTBLEMPLOYEE\_ID;

TIntegerField \*RepairTBLREPAIR\_SHOP\_ID;

TIntegerField \*RepairTBLVEHICLE\_ID;

TIBStringField \*RepairTBLACCEPTED\_THE\_VEHICLE;

TIBStringField \*VehicleTBLREGISTRATION\_NUMBER;

TDateField \*VehicleTBLREGISTRATION\_DATE;

TDateField \*VehicleTBLPRODUCTION\_YEAR;

TIntegerField \*CategoryTBLID;

TIBStringField \*CategoryTBLNAME;

TIntegerField \*AllServiceTBLID;

TIBStringField \*AllServiceTBLNAME;

TIBStringField \*EmployeeTBLPHONE\_NUMBER;

TIntegerField \*RShopTBLID;

TIBStringField \*RShopTBLNAME;

TIBStringField \*RShopTBLADDRESS;

TStringField \*ItemTBLNAME;

TMenuItem \*N9;

TMenuItem \*N10;

TIBQuery \*LogInQuery;

TIBQuery \*RepairCountQuery;

TDataSource \*RepairCountDS;

TIBStringField \*RepairCountQueryNAME;

TIntegerField \*RepairCountQueryCNT;

TIBQuery \*ServiceCountQuery;

TDataSource \*ServiceCountDS;

TIBStringField \*ServiceCountQueryNAME;

TIntegerField \*ServiceCountQueryCNT;

TIBQuery \*SumQuery;

TIBQuery \*PriceQuery;

TDataSource \*PriceDS;

TIBStringField \*PriceQueryNAME;

TIBBCDField \*PriceQueryMAXI;

TIBBCDField \*PriceQuerySUMM;

TIBQuery \*repair\_service\_delete\_query;

TStringField \*RepairTBLREPAIR\_SHOP\_NAME;

TStringField \*RepairTBLEMPLOYEE\_NAME;

TStringField \*RepairTBLEMPLOYEE\_POSSITION;

TIBQuery \*ReferenceRepairQuery;

TDataSource \*ReferenceRepairDS;

TDataSource \*ReferenceItemDS;

TMenuItem \*N11;

TDateField \*ReferenceRepairQueryACCPETANCE\_DATE;

TDateField \*ReferenceRepairQueryCOMPLETION\_DATE;

TIBStringField \*ReferenceRepairQueryACCEPTED\_THE\_VEHICLE;

TIBStringField \*ReferenceRepairQueryREGISTRATION\_NUMBER;

TIBStringField \*ReferenceRepairQueryNAME;

TIBStringField \*ReferenceRepairQueryNAME1;

TIBStringField \*ReferenceRepairQueryNOTE;

TIBStringField \*ReferenceRepairQueryNAME2;

TIBQuery \*ReferenceItemQuery;

TIBStringField \*ReferenceItemQueryNAME;

TIBBCDField \*ReferenceItemQueryPRICE;

TIBBCDField \*ReferenceItemQueryQUANTITY;

TDateField \*ReferenceItemQueryCOMPLETION\_DATE;

TIBStringField \*ReferenceItemQueryREGISTRATION\_NUMBER;

TIBStringField \*ReferenceItemQueryNAME3;

TIBStringField \*ReferenceItemQueryNAME4;

TIBStringField \*ReferenceItemQueryNAME1;

TIBStringField \*ReferenceItemQueryNAME2;

TDateField \*ReferenceItemQueryACCPETANCE\_DATE;

TMenuItem \*N12;

TMenuItem \*N13;

TMenuItem \*N14;

void \_\_fastcall EmployeesTBL1Click(TObject \*Sender);

void \_\_fastcall OwnersTBL1Click(TObject \*Sender);

void \_\_fastcall PartsTBL1Click(TObject \*Sender);

void \_\_fastcall ServicesTBL1Click(TObject \*Sender);

void \_\_fastcall BrandTBL1Click(TObject \*Sender);

void \_\_fastcall N1Click(TObject \*Sender);

void \_\_fastcall N6Click(TObject \*Sender);

void \_\_fastcall N2Click(TObject \*Sender);

void \_\_fastcall N3Click(TObject \*Sender);

void \_\_fastcall N4Click(TObject \*Sender);

void \_\_fastcall N5Click(TObject \*Sender);

void \_\_fastcall VehicleTBLBeforePost(TDataSet \*DataSet);

void \_\_fastcall N7Click(TObject \*Sender);

void \_\_fastcall RepairTBLBeforePost(TDataSet \*DataSet);

void \_\_fastcall N8Click(TObject \*Sender);

void \_\_fastcall RepairServiceTBLBeforePost(TDataSet \*DataSet);

void \_\_fastcall ItemTBLBeforePost(TDataSet \*DataSet);

void \_\_fastcall About1Click(TObject \*Sender);

void \_\_fastcall BrandTBLBeforePost(TDataSet \*DataSet);

void \_\_fastcall VehicleNomTBLBeforePost(TDataSet \*DataSet);

void \_\_fastcall CategoryTBLBeforePost(TDataSet \*DataSet);

void \_\_fastcall VehicleTypeTBLBeforePost(TDataSet \*DataSet);

void \_\_fastcall EngineTypeTBLBeforePost(TDataSet \*DataSet);

void \_\_fastcall OwnerTBLBeforePost(TDataSet \*DataSet);

void \_\_fastcall AllServiceTBLBeforePost(TDataSet \*DataSet);

void \_\_fastcall MesureTBLBeforePost(TDataSet \*DataSet);

void \_\_fastcall ManufacturerTBLBeforePost(TDataSet \*DataSet);

void \_\_fastcall ItemNomTBLBeforePost(TDataSet \*DataSet);

void \_\_fastcall PossitionTBLBeforePost(TDataSet \*DataSet);

void \_\_fastcall EmployeeTBLBeforePost(TDataSet \*DataSet);

void \_\_fastcall RShopTBLBeforePost(TDataSet \*DataSet);

void \_\_fastcall ItemTBLAfterPost(TDataSet \*DataSet);

void \_\_fastcall N10Click(TObject \*Sender);

void \_\_fastcall Export1Click(TObject \*Sender);

void \_\_fastcall N9Click(TObject \*Sender);

void \_\_fastcall N11Click(TObject \*Sender);

void \_\_fastcall N13Click(TObject \*Sender);

void \_\_fastcall N14Click(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TDM(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TDM \*DM;

//---------------------------------------------------------------------------

#endif

**DataModuleUnit.cpp**

#include <vcl.h>

#pragma hdrstop

#include "DataModuleUnit.h"

#include "EmployeeForm.h"

#include "OwnerForm.h"

#include "PartForm.h"

#include "ServiceForm.h"

#include "BrandForm.h"

#include "CategoryForm.h"

#include "TypeForm.h"

#include "EngineForm.h"

#include "ManufacturerForm.h"

#include "MesureForm.h"

#include "PossitionForm.h"

#include "VehicleForm.h"

#include "RepairForm.h"

#include "LogInForm.h"

#include "CreateAccForm.h"

#include "MainUnit.h"

#include "AnalysisForm.h"

#include "ReferenceForm.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TDM \*DM;

//---------------------------------------------------------------------------

\_\_fastcall TDM::TDM(TComponent\* Owner)

: TDataModule(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::EmployeesTBL1Click(TObject \*Sender)

{

EmployeeFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::OwnersTBL1Click(TObject \*Sender)

{

OwnerFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::PartsTBL1Click(TObject \*Sender)

{

PartFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::ServicesTBL1Click(TObject \*Sender)

{

ServiceFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::BrandTBL1Click(TObject \*Sender)

{

BrandFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::N1Click(TObject \*Sender)

{

CategoryFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::N6Click(TObject \*Sender)

{

TypeFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::N2Click(TObject \*Sender)

{

EngineFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::N3Click(TObject \*Sender)

{

ManufacturerFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::N4Click(TObject \*Sender)

{

MesureFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::N5Click(TObject \*Sender)

{

PossitionFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::VehicleTBLBeforePost(TDataSet \*DataSet)

{

if(VehicleTBL->State == dsInsert)

{

VehicleSP->Prepare();

VehicleSP->ExecProc();

VehicleTBL->FieldByName("ID")->AsInteger =

VehicleSP->ParamByName("ID")->AsInteger;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::N7Click(TObject \*Sender)

{

VehicleNomFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::RepairTBLBeforePost(TDataSet \*DataSet)

{

if(RepairTBL->State == dsInsert)

{

RepairSP->Prepare();

RepairSP->ExecProc();

RepairTBL->FieldByName("ID")->AsInteger =

RepairSP->ParamByName("ID")->AsInteger;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::N8Click(TObject \*Sender)

{

RepairShopFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::RepairServiceTBLBeforePost(TDataSet \*DataSet)

{

if(RepairServiceTBL->State == dsInsert)

{

RepairServiceSP->Prepare();

RepairServiceSP->ExecProc();

RepairServiceTBL->FieldByName("ID")->AsInteger =

RepairServiceSP->ParamByName("ID")->AsInteger;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::ItemTBLBeforePost(TDataSet \*DataSet)

{

if(ItemTBL->State == dsInsert)

{

ItemSP->Prepare();

ItemSP->ExecProc();

ItemTBL->FieldByName("ID")->AsInteger =

ItemSP->ParamByName("ID")->AsInteger;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::About1Click(TObject \*Sender)

{

ShowMessage("Дипломна работа 2021");

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::BrandTBLBeforePost(TDataSet \*DataSet)

{

if(BrandTBL->State == dsInsert)

{

BrandSP->Prepare();

BrandSP->ExecProc();

BrandTBL->FieldByName("ID")->AsInteger =

BrandSP->ParamByName("ID")->AsInteger;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::VehicleNomTBLBeforePost(TDataSet \*DataSet)

{

if(VehicleNomTBL->State == dsInsert)

{

VehicleNomSP->Prepare();

VehicleNomSP->ExecProc();

VehicleNomTBL->FieldByName("ID")->AsInteger =

VehicleNomSP->ParamByName("ID")->AsInteger;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::CategoryTBLBeforePost(TDataSet \*DataSet)

{

if(CategoryTBL->State == dsInsert)

{

CategorySP->Prepare();

CategorySP->ExecProc();

CategoryTBL->FieldByName("ID")->AsInteger =

CategorySP->ParamByName("ID")->AsInteger;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::VehicleTypeTBLBeforePost(TDataSet \*DataSet)

{

if(VehicleTypeTBL->State == dsInsert)

{

VehicleTypeSP->Prepare();

VehicleTypeSP->ExecProc();

VehicleTypeTBL->FieldByName("ID")->AsInteger =

VehicleTypeSP->ParamByName("ID")->AsInteger;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::EngineTypeTBLBeforePost(TDataSet \*DataSet)

{

if(EngineTypeTBL->State == dsInsert)

{

EngineTypeSP->Prepare();

EngineTypeSP->ExecProc();

EngineTypeTBL->FieldByName("ID")->AsInteger =

EngineTypeSP->ParamByName("ID")->AsInteger;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::OwnerTBLBeforePost(TDataSet \*DataSet)

{

if(OwnerTBL->State == dsInsert)

{

OwnerSP->Prepare();

OwnerSP->ExecProc();

OwnerTBL->FieldByName("ID")->AsInteger =

OwnerSP->ParamByName("ID")->AsInteger;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::AllServiceTBLBeforePost(TDataSet \*DataSet)

{

if(AllServiceTBL->State == dsInsert)

{

ServiceSP->Prepare();

ServiceSP->ExecProc();

AllServiceTBL->FieldByName("ID")->AsInteger =

ServiceSP->ParamByName("ID")->AsInteger;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::MesureTBLBeforePost(TDataSet \*DataSet)

{

if(MesureTBL->State == dsInsert)

{

MesureSP->Prepare();

MesureSP->ExecProc();

MesureTBL->FieldByName("ID")->AsInteger =

MesureSP->ParamByName("ID")->AsInteger;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::ManufacturerTBLBeforePost(TDataSet \*DataSet)

{

if(ManufacturerTBL->State == dsInsert)

{

ManufacturerSP->Prepare();

ManufacturerSP->ExecProc();

ManufacturerTBL->FieldByName("ID")->AsInteger =

ManufacturerSP->ParamByName("ID")->AsInteger;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::ItemNomTBLBeforePost(TDataSet \*DataSet)

{

if(ItemNomTBL->State == dsInsert)

{

ItemNomSP->Prepare();

ItemNomSP->ExecProc();

ItemNomTBL->FieldByName("ID")->AsInteger =

ItemNomSP->ParamByName("ID")->AsInteger;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::PossitionTBLBeforePost(TDataSet \*DataSet)

{

if(PossitionTBL->State == dsInsert)

{

PossitionSP->Prepare();

PossitionSP->ExecProc();

PossitionTBL->FieldByName("ID")->AsInteger =

PossitionSP->ParamByName("ID")->AsInteger;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::EmployeeTBLBeforePost(TDataSet \*DataSet)

{

if(EmployeeTBL->State == dsInsert)

{

EmployeeSP->Prepare();

EmployeeSP->ExecProc();

EmployeeTBL->FieldByName("ID")->AsInteger =

EmployeeSP->ParamByName("ID")->AsInteger;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::RShopTBLBeforePost(TDataSet \*DataSet)

{

if(RShopTBL->State == dsInsert)

{

RShopSP->Prepare();

RShopSP->ExecProc();

RShopTBL->FieldByName("ID")->AsInteger =

RShopSP->ParamByName("ID")->AsInteger;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::ItemTBLAfterPost(TDataSet \*DataSet)

{

DM->SumQuery->Close();

DM->SumQuery->ParamByName("PSERVICEID")->AsInteger =

DM->RepairServiceTBL->FieldByName("SERVICE\_ID")->AsInteger;

DM->SumQuery->Open();

DM->RepairServiceTBL->Edit();

DM->RepairServiceTBL->FieldByName("TOTAL\_PRICE")->AsInteger =

DM->SumQuery->FieldByName("SUMM")->AsInteger;

DM->RepairServiceTBL->Post();

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::N10Click(TObject \*Sender)

{

// if(MainFormUnit->MainStatusBar->Panels->Items[1]->Text == "Online: 0000")

CreateAccFormUnit->ShowModal();

// else

// ShowMessage("Нямате правата да създаде акаунт!");

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::Export1Click(TObject \*Sender)

{

LogInFormUnit->ShowModal();

MainFormUnit->Close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::N9Click(TObject \*Sender)

{

AnalysisFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::N11Click(TObject \*Sender)

{

ReferenceFromUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::N13Click(TObject \*Sender)

{

AnsiString text,s[13];

int i,j,lines,column;

if(MainFormUnit->OpenDialog->Execute())

{

MainFormUnit->Memo->Lines->LoadFromFile(MainFormUnit->OpenDialog->FileName);

lines = MainFormUnit->Memo->Lines->Count;

for(i=1;i<lines;i++)

{

text = MainFormUnit->Memo->Lines->Strings[i];

column = 1;

for(j=1;j<=text.Length();j++)

{

if(text[j]==';')

{

s[column]=text.SubString(1,j-1);

text = text.Delete(1,j);

j=1;

column++;

}

}

DM->VehicleTBL->Append();

DM->VehicleTBL->FieldByName("REGISTRATION\_NUMBER")->AsString = s[1];

DM->VehicleTBL->FieldByName("REGISTRATION\_DATE")->AsDateTime = StrToDateTime(s[2].Delete(10,3));

DM->VehicleTBL->FieldByName("VEHICLE\_BRAND\_NAME")->AsString = s[3];

DM->VehicleTBL->FieldByName("VEHICLE\_MODEL\_NAME")->AsString = s[4];

DM->VehicleTBL->FieldByName("PRODUCTION\_YEAR")->AsDateTime = StrToDateTime(s[5].Delete(10,3));

DM->VehicleTBL->FieldByName("VEHICLE\_CATEGORY\_NAME")->AsString = s[6];

DM->VehicleTBL->FieldByName("VEHICLE\_TYPE\_NAME")->AsString = s[7];

DM->VehicleTBL->FieldByName("VEHICLE\_ENGINE\_NAME")->AsString = s[8];

DM->VehicleTBL->FieldByName("CHASSIS\_NUMBER")->AsString = s[9];

DM->VehicleTBL->FieldByName("FRAME\_NUMBER")->AsString = s[10];

DM->VehicleTBL->FieldByName("OWNER\_NAME")->AsString = s[11];

DM->VehicleTBL->FieldByName("OWNER\_id")->AsInteger = StrToInt(s[12]);

DM->VehicleTBL->FieldByName("VEHICLE\_NOMENCLATURE\_ID")->AsInteger = StrToInt(s[13]);

DM->VehicleTBL->Post();

}

}

else

ShowMessage("Има грешка при избирането на файл!");

}

//---------------------------------------------------------------------------

void \_\_fastcall TDM::N14Click(TObject \*Sender)

{

DM->VehicleTBL->First();

while(!DM->VehicleTBL->Eof)

{

MainFormUnit->Memo->Lines->Add(

DM->VehicleTBL->FieldByName("REGISTRATION\_NUMBER")->AsString + ";" +

DM->VehicleTBL->FieldByName("REGISTRATION\_DATE")->AsString + ";" +

DM->VehicleTBL->FieldByName("VEHICLE\_BRAND\_NAME")->AsString + ";" +

DM->VehicleTBL->FieldByName("VEHICLE\_MODEL\_NAME")->AsString + ";" +

DM->VehicleTBL->FieldByName("PRODUCTION\_YEAR")->AsString + ";" +

DM->VehicleTBL->FieldByName("VEHICLE\_CATEGORY\_NAME")->AsString + ";" +

DM->VehicleTBL->FieldByName("VEHICLE\_TYPE\_NAME")->AsString + ";" +

DM->VehicleTBL->FieldByName("VEHICLE\_ENGINE\_NAME")->AsString + ";" +

DM->VehicleTBL->FieldByName("CHASSIS\_NUMBER")->AsString + ";" +

DM->VehicleTBL->FieldByName("FRAME\_NUMBER")->AsString + ";" +

DM->VehicleTBL->FieldByName("OWNER\_NAME")->AsString + ";" +

DM->VehicleTBL->FieldByName("OWNER\_ID")->AsString + ";" +

DM->VehicleTBL->FieldByName("VEHICLE\_NOMENCLATURE\_ID")->AsString);

DM->VehicleTBL->Next();

}

if(MainFormUnit->SaveDialog->Execute())

MainFormUnit->Memo->Lines->SaveToFile(MainFormUnit->SaveDialog->FileName + ".CSV");

else

ShowMessage("Има грешка при избирането на файл!");

}

//---------------------------------------------------------------------------

**MainUnit.h**

#ifndef MainUnitH

#define MainUnitH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <DBGrids.hpp>

#include <Grids.hpp>

#include <ExtCtrls.hpp>

#include <ComCtrls.hpp>

#include <Buttons.hpp>

#include <Dialogs.hpp>

//---------------------------------------------------------------------------

class TMainFormUnit : public TForm

{

\_\_published: // IDE-managed Components

TDBGrid \*MainVehicleGrid;

TDBGrid \*MainRepairGrid;

TPanel \*MainVehiclePanel;

TLabel \*MainLabelSort;

TLabel \*MainLabelSearch;

TEdit \*MainEdit;

TComboBox \*MainSortCB;

TComboBox \*MainSearchCB;

TButton \*MainAddVehicleButton;

TButton \*MainEditVehicleButton;

TButton \*MainDeleteVehicle;

TPanel \*MainRepairPanel;

TButton \*MainRepairAddButton;

TButton \*MainEditRepirButton;

TButton \*MainDeleteRepair;

TStatusBar \*MainStatusBar;

TMemo \*Memo;

TSaveDialog \*SaveDialog;

TOpenDialog \*OpenDialog;

void \_\_fastcall MainSortCBSelect(TObject \*Sender);

void \_\_fastcall MainEditChange(TObject \*Sender);

void \_\_fastcall MainAddVehicleButtonClick(TObject \*Sender);

void \_\_fastcall MainRepairAddButtonClick(TObject \*Sender);

void \_\_fastcall MainEditVehicleButtonClick(TObject \*Sender);

void \_\_fastcall MainEditRepirButtonClick(TObject \*Sender);

void \_\_fastcall MainDeleteRepairClick(TObject \*Sender);

void \_\_fastcall MainDeleteVehicleClick(TObject \*Sender);

void \_\_fastcall FormShow(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TMainFormUnit(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TMainFormUnit \*MainFormUnit;

//---------------------------------------------------------------------------

#endif

**MainUnit.cpp**

#include <vcl.h>

#include <stdio.h>

#pragma hdrstop

#include "MainUnit.h"

#include "DataModuleUnit.h"

#include "EditVehicleForm.h"

#include "EditPartForm.h"

#include "LogInForm.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TMainFormUnit \*MainFormUnit;

//---------------------------------------------------------------------------

\_\_fastcall TMainFormUnit::TMainFormUnit(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TMainFormUnit::MainSortCBSelect(TObject \*Sender)

{

switch(MainSortCB->ItemIndex)

{

case 0: DM->VehicleTBL->IndexName ="RDB$PRIMARY11";break;

case 1: DM->VehicleTBL->IndexName ="VEHICLERDNDX";break;

case 2: DM->VehicleTBL->IndexName ="VEHICLERDATENDX";break;

case 3: DM->VehicleTBL->IndexName ="VEHICLEPYNDX";break;

case 4: DM->VehicleTBL->IndexName ="VEHICLEFNNDX";break;

case 5: DM->VehicleTBL->IndexName ="VEHICLECNNDX";break;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TMainFormUnit::MainEditChange(TObject \*Sender)

{

{

AnsiString FieldName ="REGISTRATION\_NUMBER";

TLocateOptions Opts;

Opts.Clear();

Opts << loPartialKey;

Opts << loCaseInsensitive;

switch(MainSearchCB->ItemIndex)

{

case 0: FieldName ="REGISTRATION\_NUMBER";break;

case 1: FieldName ="REGISTRATION\_DATE";break;

case 2: FieldName ="PRODUCTION\_YEAR";break;

case 3: FieldName ="CHASSIS\_NUMBER";break;

case 4: FieldName ="FRAME\_NUMBER";break;

}

DM->VehicleTBL->Locate(FieldName, MainEdit->Text, Opts);

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TMainFormUnit::MainAddVehicleButtonClick(TObject \*Sender)

{

DM->VehicleTBL->Append();

EditVehicleFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TMainFormUnit::MainRepairAddButtonClick(TObject \*Sender)

{

DM->RepairTBL->Append();

DM->RepairTBL->Post();

DM->RepairTBL->Edit();

RepairEditFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TMainFormUnit::MainEditVehicleButtonClick(TObject \*Sender)

{

AnsiString FieldName ="ID";

TLocateOptions Opts;

Opts.Clear();

Opts << loPartialKey;

Opts << loCaseInsensitive;

DM->VehicleNomTBL->Locate(FieldName,

DM->VehicleTBL->FieldByName("VEHICLE\_NOMENCLATURE\_ID")->AsInteger, Opts);

EditVehicleFormUnit->EVFUBrandEdit->Text =

DM->VehicleNomTBL->FieldByName("BRAND\_NAME")->AsString;

EditVehicleFormUnit->EVFUModelEdit->Text =

DM->VehicleNomTBL->FieldByName("MODEL")->AsString;

EditVehicleFormUnit->EVFUCategoryEdit->Text =

DM->VehicleNomTBL->FieldByName("CATEGORY\_NAME")->AsString;

EditVehicleFormUnit->EVFUEngineTypeEdit->Text =

DM->VehicleNomTBL->FieldByName("ENGINE\_NAME")->AsString;

EditVehicleFormUnit->EVFUTypeEdit->Text =

DM->VehicleNomTBL->FieldByName("TYPE\_NAME")->AsString;

EditVehicleFormUnit->EVFUEngineVolumeEdit->Text =

DM->VehicleNomTBL->FieldByName("ENGINE\_VOLUME")->AsString;

EditVehicleFormUnit->EVFULoadabilityEdit->Text =

DM->VehicleNomTBL->FieldByName("LOADABILITY")->AsString;

EditVehicleFormUnit->EVFUAxlesEdit->Text =

DM->VehicleNomTBL->FieldByName("AXLES\_NUMBER")->AsString;

EditVehicleFormUnit->EVFUSeatsEdit->Text =

DM->VehicleNomTBL->FieldByName("SEATS\_NUMBER")->AsString;

EditVehicleFormUnit->EVFUYearManufactTimePicker->DateTime =

DM->VehicleTBL->FieldByName("PRODUCTION\_YEAR")->AsDateTime;

EditVehicleFormUnit->EVFURegNumEdit->Text =

DM->VehicleTBL->FieldByName("REGISTRATION\_NUMBER")->AsString;

EditVehicleFormUnit->EVFURegDateTimePicker->DateTime =

DM->VehicleTBL->FieldByName("REGISTRATION\_DATE")->AsDateTime;

EditVehicleFormUnit->EVFUFrameEdit->Text =

DM->VehicleTBL->FieldByName("FRAME\_NUMBER")->AsString;

EditVehicleFormUnit->EVFUChasseEdit->Text =

DM->VehicleTBL->FieldByName("CHASSIS\_NUMBER")->AsString;

DM->OwnerTBL->Locate(FieldName,

DM->VehicleTBL->FieldByName("OWNER\_ID")->AsInteger, Opts);

EditVehicleFormUnit->EVFUNameEdit->Text =

DM->OwnerTBL->FieldByName("NAME")->AsString;

EditVehicleFormUnit->EVFUEgnBulEdit->Text =

DM->OwnerTBL->FieldByName("BUL\_EGN")->AsString;

EditVehicleFormUnit->EVFUMolEdit->Text =

DM->OwnerTBL->FieldByName("MOL")->AsString;

EditVehicleFormUnit->EVFUTelNumEdit->Text =

DM->OwnerTBL->FieldByName("PHONE\_NUMBER")->AsString;

EditVehicleFormUnit->EVFUEmailEdit->Text =

DM->OwnerTBL->FieldByName("E\_MAIL")->AsString;

EditVehicleFormUnit->EVFUAddressEdit->Text =

DM->OwnerTBL->FieldByName("ADDRESS")->AsString;

DM->VehicleTBL->Edit();

EditVehicleFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TMainFormUnit::MainEditRepirButtonClick(TObject \*Sender)

{

AnsiString FieldName ="ID";

TLocateOptions Opts;

Opts.Clear();

Opts << loPartialKey;

Opts << loCaseInsensitive;

if(!(DM->RepairTBL->Locate("VEHICLE\_ID",

DM->VehicleTBL->FieldByName("ID")->AsInteger, Opts)))

ShowMessage("Няма избран ремонт!");

//without this if the program crashes if you try to edit or delete repair that dont exist

else

{

DM->EmployeeTBL->Locate(FieldName,

DM->RepairTBL->FieldByName("EMPLOYEE\_ID")->AsInteger, Opts);

RepairEditFormUnit->REFUFirstNameEdit->Text =

DM->EmployeeTBL->FieldByName("FIRST\_NAME")->AsString;

RepairEditFormUnit->REFUFamNameEdit->Text =

DM->EmployeeTBL->FieldByName("FAM\_NAME")->AsString;

RepairEditFormUnit->REFUPhNumEdit->Text =

DM->EmployeeTBL->FieldByName("PHONE\_NUMBER")->AsString;

RepairEditFormUnit->REFUSrPhNumEdit->Text =

DM->EmployeeTBL->FieldByName("SERVICE\_PHONE\_NUMBER")->AsString;

RepairEditFormUnit->REFUEmailEdit->Text =

DM->EmployeeTBL->FieldByName("E\_MAIL")->AsString;

RepairEditFormUnit->REFUPossitionEdit->Text =

DM->EmployeeTBL->FieldByName("POSSITION")->AsString;

DM->RShopTBL->Locate(FieldName,

DM->RepairTBL->FieldByName("REPAIR\_SHOP\_ID")->AsInteger, Opts);

RepairEditFormUnit->REFUShopNameEdit->Text =

DM->RShopTBL->FieldByName("NAME")->AsString;

RepairEditFormUnit->REFUShopAddressEdit->Text =

DM->RShopTBL->FieldByName("ADDRESS")->AsString;

RepairEditFormUnit->REFUAccDateTimePicker->DateTime=

DM->RepairTBL->FieldByName("ACCPETANCE\_DATE")->AsDateTime;

RepairEditFormUnit->REFUComplDateTimePicker->DateTime=

DM->RepairTBL->FieldByName("COMPLETION\_DATE")->AsDateTime;

RepairEditFormUnit->REFUAccEdit->Text=

DM->RepairTBL->FieldByName("ACCEPTED\_THE\_VEHICLE")->AsString;

RepairEditFormUnit->REFUNoteMemo->Text=

DM->RepairTBL->FieldByName("NOTE")->AsString;

RepairEditFormUnit->ShowModal();

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TMainFormUnit::MainDeleteRepairClick(TObject \*Sender)

{

TLocateOptions Opts;

Opts.Clear();

Opts << loPartialKey;

Opts << loCaseInsensitive;

if(!(DM->RepairTBL->Locate("VEHICLE\_ID",

DM->VehicleTBL->FieldByName("ID")->AsInteger, Opts)))

ShowMessage("Няма избран ремонт!");

else

DM->repair\_service\_delete\_query->ParamByName("PREPAIRID")->AsInteger =

DM->RepairTBL->FieldByName("ID")->AsInteger;

DM->repair\_service\_delete\_query->Close();

DM->repair\_service\_delete\_query->Prepare();

DM->repair\_service\_delete\_query->ExecSQL();

DM->RepairTBL->Delete();

}

//---------------------------------------------------------------------------

void \_\_fastcall TMainFormUnit::MainDeleteVehicleClick(TObject \*Sender)

{

AnsiString FieldName ="ID";

TLocateOptions Opts;

Opts.Clear();

Opts << loPartialKey;

Opts << loCaseInsensitive;

if(DM->VehicleTBL->Locate(FieldName,

DM->RepairTBL->FieldByName("VEHICLE\_ID")->AsInteger, Opts))

ShowMessage("Моля изтрийте ремонтите първо!");

else

DM->VehicleTBL->Delete();

}

//---------------------------------------------------------------------------

void \_\_fastcall TMainFormUnit::FormShow(TObject \*Sender)

{

LogInFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

**EditVehicleForm.h**

#ifndef EditVehicleFormH

#define EditVehicleFormH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <DBCtrls.hpp>

#include <dbcgrids.hpp>

#include <DBGrids.hpp>

#include <Grids.hpp>

#include <ExtCtrls.hpp>

#include <ComCtrls.hpp>

//---------------------------------------------------------------------------

class TEditVehicleFormUnit : public TForm

{

\_\_published: // IDE-managed Components

TLabel \*EVFUYearManufactLabel;

TLabel \*EVFURegNumLabel;

TEdit \*EVFURegNumEdit;

TLabel \*EVFURegDateLabel;

TLabel \*EVFUFrameLabel;

TEdit \*EVFUFrameEdit;

TLabel \*EVFUChasseLabel;

TEdit \*EVFUChasseEdit;

TButton \*EVFUSelectVehicleNomButton;

TButton \*EVFUSelectOwnerButton;

TButton \*EVFUSubmitButton;

TBevel \*EVFUBevel1;

TLabel \*EVFULabel1;

TLabel \*EVFUBrandLabel;

TEdit \*EVFUBrandEdit;

TEdit \*EVFUModelEdit;

TLabel \*EVFUModelLabel;

TLabel \*EVFUCategoryLabel;

TLabel \*EVFUEngineTypeLabel;

TLabel \*EVFUTypeLabel;

TEdit \*EVFUCategoryEdit;

TEdit \*EVFUEngineTypeEdit;

TEdit \*EVFUTypeEdit;

TLabel \*EVFUEngineVolumeLabel;

TEdit \*EVFUEngineVolumeEdit;

TEdit \*EVFULoadabilityEdit;

TEdit \*EVFUAxlesEdit;

TEdit \*EVFUSeatsEdit;

TLabel \*EVFULoadabilityLabel;

TLabel \*EVFUAxlesLabel;

TLabel \*EVFUSeatsLabel;

TBevel \*EVFUBevel2;

TLabel \*EVFULabel2;

TLabel \*EVFULabel3;

TBevel \*Bevel1;

TEdit \*EVFUNameEdit;

TLabel \*EVFUNameLabel;

TEdit \*EVFUTelNumEdit;

TLabel \*EVFUTelNumLabel;

TEdit \*EVFUEmailEdit;

TLabel \*EVFUEmailLabel;

TEdit \*EVFUMolEdit;

TLabel \*EVFUMolLabel;

TEdit \*EVFUEgnBulEdit;

TLabel \*EVFUEgnBulLabel;

TEdit \*EVFUAddressEdit;

TLabel \*EVFUAddressLabel;

TButton \*EVFUCloseButton;

TDateTimePicker \*EVFUYearManufactTimePicker;

TDateTimePicker \*EVFURegDateTimePicker;

void \_\_fastcall EVFUSelectVehicleNomButtonClick(TObject \*Sender);

void \_\_fastcall EVFUSubmitButtonClick(TObject \*Sender);

void \_\_fastcall EVFUSelectOwnerButtonClick(TObject \*Sender);

void \_\_fastcall EVFUCloseButtonClick(TObject \*Sender);

void \_\_fastcall FormClose(TObject \*Sender, TCloseAction &Action);

private: // User declarations

public: // User declarations

\_\_fastcall TEditVehicleFormUnit(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TEditVehicleFormUnit \*EditVehicleFormUnit;

//---------------------------------------------------------------------------

#endif

**EditVehicleForm.cpp**

include <vcl.h>

#pragma hdrstop

#include "EditVehicleForm.h"

#include "MainUnit.h"

#include "DataModuleUnit.h"

#include "OwnerForm.h"

#include "VehicleForm.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TEditVehicleFormUnit \*EditVehicleFormUnit;

//---------------------------------------------------------------------------

\_\_fastcall TEditVehicleFormUnit::TEditVehicleFormUnit(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TEditVehicleFormUnit::EVFUSelectVehicleNomButtonClick(TObject \*Sender)

{

VehicleNomFormUnit->VehicleNomButtonSelect->Visible = true;

VehicleNomFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TEditVehicleFormUnit::EVFUSubmitButtonClick(TObject \*Sender)

{

if(

EVFUModelEdit->Text.IsEmpty() ||

EVFURegNumEdit->Text.IsEmpty() ||

EVFUEgnBulEdit->Text.IsEmpty()

)

ShowMessage("Моля попълнете всички полета");

else{

DM->VehicleTBL->FieldByName("PRODUCTION\_YEAR")->AsDateTime =

EVFUYearManufactTimePicker->DateTime;

DM->VehicleTBL->FieldByName("REGISTRATION\_NUMBER")->AsString =

EVFURegNumEdit->Text;

DM->VehicleTBL->FieldByName("REGISTRATION\_DATE")->AsDateTime =

EVFURegDateTimePicker->DateTime;

DM->VehicleTBL->FieldByName("FRAME\_NUMBER")->AsString =

EVFUFrameEdit->Text;

DM->VehicleTBL->FieldByName("CHASSIS\_NUMBER")->AsString =

EVFUChasseEdit->Text;

DM->VehicleTBL->FieldByName("OWNER\_ID")->AsInteger=

DM->OwnerTBL->FieldByName("ID")->AsInteger;

DM->VehicleTBL->FieldByName("VEHICLE\_NOMENCLATURE\_ID")->AsInteger=

DM->VehicleNomTBL->FieldByName("ID")->AsInteger;

DM->VehicleTBL->Post();

EditVehicleFormUnit->Close();

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TEditVehicleFormUnit::EVFUSelectOwnerButtonClick(TObject \*Sender)

{

OwnerFormUnit->OwnerButtonSelect->Visible = true;

OwnerFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TEditVehicleFormUnit::EVFUCloseButtonClick(TObject \*Sender)

{

EditVehicleFormUnit->Close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TEditVehicleFormUnit::FormClose(TObject \*Sender,

TCloseAction &Action)

{

if(DM->VehicleTBL->State == dsEdit || DM->VehicleTBL->State == dsInsert)

DM->VehicleTBL->Cancel();

EVFUYearManufactTimePicker->DateTime = Now();

EVFUTypeEdit->Text = "";

EVFUTelNumEdit->Text = "";

EVFUSeatsEdit->Text = "";

EVFURegNumEdit->Text = "";

EVFURegDateTimePicker->DateTime = Now();

EVFUNameEdit->Text = "";

EVFUMolEdit->Text = "";

EVFUModelEdit->Text = "";

EVFULoadabilityEdit->Text = "";

EVFUFrameEdit->Text = "";

EVFUEngineVolumeEdit->Text = "";

EVFUEngineTypeEdit->Text = "";

EVFUEmailEdit->Text = "";

EVFUEgnBulEdit->Text = "";

EVFUChasseEdit->Text = "";

EVFUCategoryEdit->Text = "";

EVFUBrandEdit->Text = "";

EVFUAxlesEdit->Text = "";

EVFUAddressEdit->Text = "";

}

//---------------------------------------------------------------------------

**EditPartForm.h**

#ifndef EditPartFormH

#define EditPartFormH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <DBGrids.hpp>

#include <Grids.hpp>

#include <DBCtrls.hpp>

#include <ExtCtrls.hpp>

#include <ComCtrls.hpp>

//---------------------------------------------------------------------------

class TRepairEditFormUnit : public TForm

{

\_\_published: // IDE-managed Components

TDBGrid \*REFUServniceGrid;

TDBGrid \*REFUPartGrid;

TLabel \*REFULabel1;

TLabel \*REFUAccDateLabel;

TEdit \*REFUAccEdit;

TLabel \*REFUNoteLabel;

TLabel \*REFUComplDateLabel;

TLabel \*REFUAccLabel;

TBevel \*REFUBevel1;

TLabel \*REFULabel3;

TEdit \*REFUShopNameEdit;

TEdit \*REFUShopAddressEdit;

TLabel \*REFUShopNameLabel;

TLabel \*REFUShopAddressLabel;

TButton \*REFUSelectEmployeeButton;

TEdit \*REFUFirstNameEdit;

TLabel \*REFULabel2;

TEdit \*REFUFamNameEdit;

TEdit \*REFUPhNumEdit;

TEdit \*REFUSrPhNumEdit;

TEdit \*REFUEmailEdit;

TBevel \*REFUBevel2;

TBevel \*REFUBevel3;

TButton \*REFUSelectShopButton;

TLabel \*REFULabel4;

TLabel \*REFULabel5;

TButton \*REFUSelectService;

TButton \*REFUSelectItemNom;

TLabel \*REFUFirstNameLabel;

TLabel \*REFUFamNameLabel;

TLabel \*REFUPhNumLabel;

TLabel \*REFUSrPhNumLabel;

TLabel \*REFUEmailLabel;

TButton \*REFUSubmitButton;

TMemo \*REFUNoteMemo;

TLabel \*REFUPossitionLabel;

TEdit \*REFUPossitionEdit;

TDateTimePicker \*REFUAccDateTimePicker;

TDateTimePicker \*REFUComplDateTimePicker;

TDBNavigator \*DBNavigator1;

TDBNavigator \*DBNavigator2;

TButton \*REFUCloseButton;

void \_\_fastcall REFUSubmitButtonClick(TObject \*Sender);

void \_\_fastcall REFUSelectEmployeeButtonClick(TObject \*Sender);

void \_\_fastcall FormClose(TObject \*Sender, TCloseAction &Action);

void \_\_fastcall REFUSelectServiceClick(TObject \*Sender);

void \_\_fastcall REFUSelectShopButtonClick(TObject \*Sender);

void \_\_fastcall REFUSelectItemNomClick(TObject \*Sender);

void \_\_fastcall REFUCloseButtonClick(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TRepairEditFormUnit(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TRepairEditFormUnit \*RepairEditFormUnit;

//---------------------------------------------------------------------------

#endif

**EditPartForm.cpp**

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "EditPartForm.h"

#include "DataModuleUnit.h"

#include "EmployeeForm.h"

#include "ServiceForm.h"

#include "RepairForm.h"

#include "ItemForm.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TRepairEditFormUnit \*RepairEditFormUnit;

//---------------------------------------------------------------------------

\_\_fastcall TRepairEditFormUnit::TRepairEditFormUnit(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TRepairEditFormUnit::REFUSubmitButtonClick(TObject \*Sender)

{

if(

REFUAccEdit->Text.IsEmpty() ||

REFUFirstNameEdit->Text.IsEmpty() ||

REFUShopNameEdit->Text.IsEmpty()

)

ShowMessage("Моля попълнете всички полета");

else{

DM->RepairTBL->Edit();

DM->RepairTBL->FieldByName("ACCPETANCE\_DATE")->AsDateTime =

REFUAccDateTimePicker->DateTime;

DM->RepairTBL->FieldByName("COMPLETION\_DATE")->AsDateTime =

REFUComplDateTimePicker->DateTime;

DM->RepairTBL->FieldByName("ACCEPTED\_THE\_VEHICLE")->AsString =

REFUAccEdit->Text;

DM->RepairTBL->FieldByName("NOTE")->AsString =

RepairEditFormUnit->REFUNoteMemo->Text;

DM->RepairTBL->FieldByName("Employee\_ID")->AsInteger=

DM->EmployeeTBL->FieldByName("ID")->AsInteger;

DM->RepairTBL->FieldByName("Repair\_Shop\_ID")->AsInteger=

DM->RShopTBL->FieldByName("ID")->AsInteger;

DM->RepairTBL->Post();

RepairEditFormUnit->Close();

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TRepairEditFormUnit::REFUSelectEmployeeButtonClick(

TObject \*Sender)

{

EmployeeFormUnit->EmployeeButtonSelect->Visible = true;

EmployeeFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TRepairEditFormUnit::FormClose(TObject \*Sender,

TCloseAction &Action)

{

if(DM->RepairTBL->State == dsEdit || DM->RepairTBL->State == dsInsert)

DM->RepairTBL->Cancel();

if(

REFUAccEdit->Text.IsEmpty() ||

REFUFirstNameEdit->Text.IsEmpty() ||

REFUShopNameEdit->Text.IsEmpty()

)

DM->RepairTBL->Delete();

REFUFirstNameEdit->Text = "";

REFUFamNameEdit->Text = "";

REFUPhNumEdit->Text = "";

REFUSrPhNumEdit->Text = "";

REFUEmailEdit->Text = "";

REFUPossitionEdit->Text = "";

REFUShopNameEdit->Text = "";

REFUShopAddressEdit->Text = "";

REFUAccEdit->Text = "";

REFUNoteMemo->Text = "";

REFUAccDateTimePicker->DateTime = Now();

REFUComplDateTimePicker->DateTime = Now();

}

//---------------------------------------------------------------------------

void \_\_fastcall TRepairEditFormUnit::REFUSelectServiceClick(

TObject \*Sender)

{

ServiceFormUnit->ServiceButtonSelect->Visible = true;

ServiceFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TRepairEditFormUnit::REFUSelectShopButtonClick(

TObject \*Sender)

{

RepairShopFormUnit->RShopButtonSelect->Visible = true;

RepairShopFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TRepairEditFormUnit::REFUSelectItemNomClick(TObject \*Sender)

{

ItemNomFormUnit->ItemNomButtonSelect->Visible = true;

ItemNomFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TRepairEditFormUnit::REFUCloseButtonClick(TObject \*Sender)

{

RepairEditFormUnit->Close();

}

//---------------------------------------------------------------------------

**AnalysisForm.h**

#ifndef AnalysisFormH

#define AnalysisFormH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <DBGrids.hpp>

#include <Grids.hpp>

#include <DBCtrls.hpp>

#include <Mask.hpp>

#include <ExtCtrls.hpp>

#include <Buttons.hpp>

//---------------------------------------------------------------------------

class TAnalysisFormUnit : public TForm

{

\_\_published: // IDE-managed Components

TDBGrid \*SearchBrandGrid;

TButton \*BrandMaxButton;

TButton \*BrandMinButton;

TLabel \*SearchBrandLabel1;

TLabel \*SearchBrandLabel2;

TDBGrid \*SearchServiceGrid;

TButton \*SearchServiceButton;

TEdit \*SearchBrandEdit;

TLabel \*SearchServiceLabel1;

TEdit \*SearchServiceEdit;

TLabel \*SearchServiceLabel2;

TButton \*ClearServiceButton;

TButton \*ServiceMaxButton;

TButton \*ServiceMinButton;

TDBGrid \*SearchPriceGrid;

TLabel \*SearchPriceLabel1;

TLabel \*SearchPriceLabel2;

TEdit \*SearchPriceEdit;

TButton \*SearchPriceButton;

TButton \*ClearPriceButton;

TButton \*PriceMaxButton;

TButton \*PriceMinButton;

TBevel \*ServiceBevel;

TBevel \*BrandBevel;

TBevel \*PriceBevel;

TBitBtn \*BitBtn1;

TBitBtn \*BitBtn2;

TBitBtn \*BitBtn3;

void \_\_fastcall BrandMaxButtonClick(TObject \*Sender);

void \_\_fastcall BrandMinButtonClick(TObject \*Sender);

void \_\_fastcall SearchServiceButtonClick(TObject \*Sender);

void \_\_fastcall SearchBrandEditChange(TObject \*Sender);

void \_\_fastcall ClearServiceButtonClick(TObject \*Sender);

void \_\_fastcall ServiceMaxButtonClick(TObject \*Sender);

void \_\_fastcall ServiceMinButtonClick(TObject \*Sender);

void \_\_fastcall PriceMaxButtonClick(TObject \*Sender);

void \_\_fastcall PriceMinButtonClick(TObject \*Sender);

void \_\_fastcall ClearPriceButtonClick(TObject \*Sender);

void \_\_fastcall SearchPriceButtonClick(TObject \*Sender);

void \_\_fastcall FormShow(TObject \*Sender);

void \_\_fastcall BitBtn1Click(TObject \*Sender);

void \_\_fastcall BitBtn2Click(TObject \*Sender);

void \_\_fastcall BitBtn3Click(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TAnalysisFormUnit(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TAnalysisFormUnit \*AnalysisFormUnit;

//---------------------------------------------------------------------------

#endif

**AnalysisForm.cpp**

#include <vcl.h>

#include <limits.h>

#pragma hdrstop

#include "AnalysisForm.h"

#include "DataModuleUnit.h"

#include "ChartForm.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TAnalysisFormUnit \*AnalysisFormUnit;

//---------------------------------------------------------------------------

\_\_fastcall TAnalysisFormUnit::TAnalysisFormUnit(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TAnalysisFormUnit::BrandMaxButtonClick(TObject \*Sender)

{

DM->RepairCountQuery->First();

}

//---------------------------------------------------------------------------

void \_\_fastcall TAnalysisFormUnit::BrandMinButtonClick(TObject \*Sender)

{

DM->RepairCountQuery->Last();

}

//---------------------------------------------------------------------------

void \_\_fastcall TAnalysisFormUnit::SearchServiceButtonClick(TObject \*Sender)

{

DM->ServiceCountQuery->Close();

if(SearchServiceEdit->Text.IsEmpty())

DM->ServiceCountQuery->ParamByName("PBRANDNAME")->AsString = '%';

else

DM->ServiceCountQuery->ParamByName("PBRANDNAME")->AsString = SearchServiceEdit->Text;

DM->ServiceCountQuery->Open();

}

//---------------------------------------------------------------------------

void \_\_fastcall TAnalysisFormUnit::SearchBrandEditChange(TObject \*Sender)

{

AnsiString FieldName ="NAME";

TLocateOptions Opts;

Opts.Clear();

Opts << loPartialKey;

Opts << loCaseInsensitive;

DM->RepairCountQuery->Locate(FieldName, SearchBrandEdit->Text, Opts);

}

//---------------------------------------------------------------------------

void \_\_fastcall TAnalysisFormUnit::ClearServiceButtonClick(

TObject \*Sender)

{

DM->ServiceCountQuery->Close();

DM->ServiceCountQuery->ParamByName("PBRANDNAME")->AsString = '%';

DM->ServiceCountQuery->Open();

SearchServiceEdit->Text = "";

}

//---------------------------------------------------------------------------

void \_\_fastcall TAnalysisFormUnit::ServiceMaxButtonClick(TObject \*Sender)

{

DM->ServiceCountQuery->First();

}

//---------------------------------------------------------------------------

void \_\_fastcall TAnalysisFormUnit::ServiceMinButtonClick(TObject \*Sender)

{

DM->ServiceCountQuery->Last();

}

//---------------------------------------------------------------------------

void \_\_fastcall TAnalysisFormUnit::PriceMaxButtonClick(TObject \*Sender)

{

DM->PriceQuery->First();

}

//---------------------------------------------------------------------------

void \_\_fastcall TAnalysisFormUnit::PriceMinButtonClick(TObject \*Sender)

{

DM->PriceQuery->Last();

}

//---------------------------------------------------------------------------

void \_\_fastcall TAnalysisFormUnit::ClearPriceButtonClick(TObject \*Sender)

{

DM->PriceQuery->Close();

DM->PriceQuery->ParamByName("PBRANDNAME")->AsString = '%';

DM->PriceQuery->Open();

SearchPriceEdit->Text = "";

}

//---------------------------------------------------------------------------

void \_\_fastcall TAnalysisFormUnit::SearchPriceButtonClick(TObject \*Sender)

{

DM->PriceQuery->Close();

if(SearchServiceEdit->Text.IsEmpty())

DM->PriceQuery->ParamByName("PBRANDNAME")->AsString = '%';

else

DM->PriceQuery->ParamByName("PBRANDNAME")->AsString = SearchPriceEdit->Text;

DM->PriceQuery->Open();

}

//---------------------------------------------------------------------------

void \_\_fastcall TAnalysisFormUnit::FormShow(TObject \*Sender)

{

DM->RepairCountQuery->Close();

DM->RepairCountQuery->Open();

DM->ServiceCountQuery->Close();

DM->ServiceCountQuery->ParamByName("PBRANDNAME")->AsString = '%';

DM->ServiceCountQuery->Open();

DM->PriceQuery->Close();

DM->PriceQuery->ParamByName("PBRANDNAME")->AsString = '%';

DM->PriceQuery->Open();

}

//---------------------------------------------------------------------------

void \_\_fastcall TAnalysisFormUnit::BitBtn1Click(TObject \*Sender)

{

ChartFormUnit->QRDBChart1->Series[0]->Active = false;

ChartFormUnit->QRDBChart1->Series[1]->Active = false;

ChartFormUnit->QRDBChart1->Series[2]->Active = false;

ChartFormUnit->QRDBChart1->Series[0]->Active = true;

ChartFormUnit->Show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TAnalysisFormUnit::BitBtn2Click(TObject \*Sender)

{

ChartFormUnit->QRDBChart1->Series[0]->Active = false;

ChartFormUnit->QRDBChart1->Series[1]->Active = false;

ChartFormUnit->QRDBChart1->Series[2]->Active = false;

ChartFormUnit->QRDBChart1->Series[1]->Active = true;

ChartFormUnit->Show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TAnalysisFormUnit::BitBtn3Click(TObject \*Sender)

{

ChartFormUnit->QRDBChart1->Series[0]->Active = false;

ChartFormUnit->QRDBChart1->Series[1]->Active = false;

ChartFormUnit->QRDBChart1->Series[2]->Active = false;

ChartFormUnit->QRDBChart1->Series[2]->Active = true;

ChartFormUnit->Show();

}

//---------------------------------------------------------------------------

**ChartForm.h**

#ifndef ChartFormH

#define ChartFormH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <Chart.hpp>

#include <DbChart.hpp>

#include <ExtCtrls.hpp>

#include <QRTEE.hpp>

#include <QuickRpt.hpp>

#include <Series.hpp>

#include <TeEngine.hpp>

#include <TeeProcs.hpp>

//---------------------------------------------------------------------------

class TChartFormUnit : public TForm

{

\_\_published: // IDE-managed Components

TQRDBChart \*QRDBChart1;

TQRChart \*QRChart;

TPieSeries \*Series1;

TPieSeries \*Series2;

TPieSeries \*Series3;

void \_\_fastcall FormClose(TObject \*Sender, TCloseAction &Action);

private: // User declarations

public: // User declarations

\_\_fastcall TChartFormUnit(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TChartFormUnit \*ChartFormUnit;

//---------------------------------------------------------------------------

#endif

**ChartForm.cpp**

#include <vcl.h>

#pragma hdrstop

#include "ChartForm.h"

#include "DataModuleUnit.h"

#include "AnalysisForm.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TChartFormUnit \*ChartFormUnit;

//---------------------------------------------------------------------------

\_\_fastcall TChartFormUnit::TChartFormUnit(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TChartFormUnit::FormClose(TObject \*Sender,

TCloseAction &Action)

{

AnalysisFormUnit->Show();

}

//---------------------------------------------------------------------------

**ReferenceForm.h**

#ifndef ReferenceFormH

#define ReferenceFormH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <ComCtrls.hpp>

#include <DBGrids.hpp>

#include <Grids.hpp>

#include <ExtCtrls.hpp>

#include <Dialogs.hpp>

//---------------------------------------------------------------------------

class TReferenceFromUnit : public TForm

{

\_\_published: // IDE-managed Components

TDBGrid \*ReferenceGrid;

TPanel \*Panel1;

TDateTimePicker \*AcceptanceToDateTimePicker;

TDateTimePicker \*AcceptanceFromDateTimePicker;

TDateTimePicker \*CompletionFromDateTimePicker;

TDateTimePicker \*CompletionToDateTimePicker;

TEdit \*BrandEdit;

TEdit \*RegNumEdit;

TEdit \*RShopEdit;

TLabel \*AcceptanceDateToLabel;

TLabel \*AcceptanceDateFromLabel;

TLabel \*CompltionDateFromLabel;

TLabel \*CompletionDateToLabel;

TLabel \*BrandLabel;

TLabel \*RShopLabel;

TLabel \*RegNumLabel;

TLabel \*MessageLabel1;

TLabel \*MessageLabel2;

TLabel \*MessageLabel3;

TLabel \*MessageLabel4;

TButton \*ShowButton;

TButton \*ExportButton;

TMemo \*ExportMemo;

TSaveDialog \*SaveDialog;

TRadioGroup \*RadioGroup;

void \_\_fastcall FormShow(TObject \*Sender);

void \_\_fastcall ShowButtonClick(TObject \*Sender);

void \_\_fastcall ExportButtonClick(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TReferenceFromUnit(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TReferenceFromUnit \*ReferenceFromUnit;

//---------------------------------------------------------------------------

#endif

**ReferenceForm.cpp**

#include <vcl.h>

#pragma hdrstop

#include "ReferenceForm.h"

#include "DataModuleUnit.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TReferenceFromUnit \*ReferenceFromUnit;

//---------------------------------------------------------------------------

\_\_fastcall TReferenceFromUnit::TReferenceFromUnit(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TReferenceFromUnit::FormShow(TObject \*Sender)

{

if(!(ReferenceGrid->DataSource == DM->ReferenceRepairDS))

ReferenceGrid->DataSource == DM->ReferenceRepairDS;

DM->ReferenceRepairQuery->Close();

DM->ReferenceRepairQuery->ParamByName("PBRANDNAME")->AsString = '%';

DM->ReferenceRepairQuery->ParamByName("PREGNUM")->AsString = '%';

DM->ReferenceRepairQuery->ParamByName("PRSHOPNAME")->AsString = '%';

DM->ReferenceRepairQuery->ParamByName("PADFROM")->AsDateTime = Now();

DM->ReferenceRepairQuery->ParamByName("PADTO")->AsDateTime = Now();

DM->ReferenceRepairQuery->ParamByName("PCDFROM")->AsDateTime = Now();

DM->ReferenceRepairQuery->ParamByName("PCDTO")->AsDateTime = Now();

DM->ReferenceRepairQuery->Open();

DM->ReferenceItemQuery->Close();

DM->ReferenceItemQuery->ParamByName("PBRANDNAME")->AsString = '%';

DM->ReferenceItemQuery->ParamByName("PREGNUM")->AsString = '%';

DM->ReferenceItemQuery->ParamByName("PRSHOPNAME")->AsString = '%';

DM->ReferenceItemQuery->ParamByName("PADFROM")->AsDateTime = Now();

DM->ReferenceItemQuery->ParamByName("PADTO")->AsDateTime = Now();

DM->ReferenceItemQuery->ParamByName("PCDFROM")->AsDateTime = Now();

DM->ReferenceItemQuery->ParamByName("PCDTO")->AsDateTime = Now();

DM->ReferenceItemQuery->Open();

BrandEdit->Text="";

RegNumEdit->Text="";

RShopEdit->Text="";

}

//---------------------------------------------------------------------------

void \_\_fastcall TReferenceFromUnit::ShowButtonClick(TObject \*Sender)

{

if(RadioGroup->ItemIndex == 0)

{

DM->ReferenceRepairQuery->Close();

if(!BrandEdit->Text.IsEmpty())

DM->ReferenceRepairQuery->ParamByName("PBRANDNAME")->AsString =

BrandEdit->Text;

if(!RegNumEdit->Text.IsEmpty())

DM->ReferenceRepairQuery->ParamByName("PREGNUM")->AsString =

RegNumEdit->Text;

if(!RShopEdit->Text.IsEmpty())

DM->ReferenceRepairQuery->ParamByName("PRSHOPNAME")->AsString =

RShopEdit->Text;

DM->ReferenceRepairQuery->ParamByName("PADFROM")->AsDateTime =

AcceptanceFromDateTimePicker->DateTime;

DM->ReferenceRepairQuery->ParamByName("PADTO")->AsDateTime =

AcceptanceToDateTimePicker->DateTime;

DM->ReferenceRepairQuery->ParamByName("PCDFROM")->AsDateTime =

CompletionFromDateTimePicker->DateTime;

DM->ReferenceRepairQuery->ParamByName("PCDTO")->AsDateTime =

CompletionToDateTimePicker->DateTime;

DM->ReferenceRepairQuery->Open();

if(!(ReferenceGrid->DataSource == DM->ReferenceRepairDS))

ReferenceGrid->DataSource = DM->ReferenceRepairDS;

}

else

{

DM->ReferenceItemQuery->Close();

if(!BrandEdit->Text.IsEmpty())

DM->ReferenceItemQuery->ParamByName("PBRANDNAME")->AsString =

BrandEdit->Text;

if(!RegNumEdit->Text.IsEmpty())

DM->ReferenceItemQuery->ParamByName("PREGNUM")->AsString =

RegNumEdit->Text;

if(!RShopEdit->Text.IsEmpty())

DM->ReferenceItemQuery->ParamByName("PRSHOPNAME")->AsString =

RShopEdit->Text;

DM->ReferenceItemQuery->ParamByName("PADFROM")->AsDateTime =

AcceptanceFromDateTimePicker->DateTime;

DM->ReferenceItemQuery->ParamByName("PADTO")->AsDateTime =

AcceptanceToDateTimePicker->DateTime;

DM->ReferenceItemQuery->ParamByName("PCDFROM")->AsDateTime =

CompletionFromDateTimePicker->DateTime;

DM->ReferenceItemQuery->ParamByName("PCDTO")->AsDateTime =

CompletionToDateTimePicker->DateTime;

DM->ReferenceItemQuery->Open();

if(!(ReferenceGrid->DataSource == DM->ReferenceItemDS))

ReferenceGrid->DataSource = DM->ReferenceItemDS;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TReferenceFromUnit::ExportButtonClick(TObject \*Sender)

{

if(RadioGroup->ItemIndex == 0){

DM->ReferenceRepairQuery->First();

while(!DM->ReferenceRepairQuery->Eof)

{

ExportMemo->Lines->Add(

DM->ReferenceRepairQuery- >FieldByName("ACCPETANCE\_DATE")->AsString + ";" +

DM->ReferenceRepairQuery->FieldByName("COMPLETION\_DATE")->AsString + ";" +

DM->ReferenceRepairQuery->FieldByName("ACCEPTED\_THE\_VEHICLE")->AsString + ";" +

DM->ReferenceRepairQuery->FieldByName("REGISTRATION\_NUMBER")->AsString + ";" +

DM->ReferenceRepairQuery->FieldByName("NAME")->AsString + ";" +

DM->ReferenceRepairQuery->FieldByName("NAME1")->AsString + ";" +

DM->ReferenceRepairQuery->FieldByName("NAME2")->AsString + ";" +

DM->ReferenceRepairQuery->FieldByName("NOTE")->AsString);

DM->ReferenceRepairQuery->Next();

}

}

else{

DM->ReferenceItemQuery->First();

while(!DM->ReferenceItemQuery->Eof)

{

ExportMemo->Lines->Add(

DM->ReferenceItemQuery->FieldByName("NAME")->AsString + ";" +

DM->ReferenceItemQuery->FieldByName("PRICE")->AsString + ";" +

DM->ReferenceItemQuery->FieldByName("QUANTITY")->AsString + ";" +

DM->ReferenceItemQuery->FieldByName("NAME1")->AsString + ";" +

DM->ReferenceItemQuery->FieldByName("NAME2")->AsString + ";" +

DM->ReferenceItemQuery->FieldByName("ACCPETANCE\_DATE")->AsString + ";" +

DM->ReferenceItemQuery->FieldByName("COMPLETION\_DATE")->AsString + ";" +

DM->ReferenceItemQuery->FieldByName("REGISTRATION\_NUMBER")->AsString + ";" +

DM->ReferenceItemQuery->FieldByName("NAME3")->AsString + ";" +

DM->ReferenceItemQuery->FieldByName("NAME4")->AsString);

DM->ReferenceItemQuery->Next();

}

}

if(SaveDialog->Execute())

ExportMemo->Lines->SaveToFile(SaveDialog->FileName + ".CSV");

else

ShowMessage("Има грешка при избирането на файла!");

}

//---------------------------------------------------------------------------

**LogInForm.h**

#ifndef LogInFormH

#define LogInFormH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

//---------------------------------------------------------------------------

class TLogInFormUnit : public TForm

{

\_\_published: // IDE-managed Components

TEdit \*UsernameEdit;

TLabel \*UsernameLabel;

TLabel \*PasswordLabel;

TEdit \*PasswordEdit;

TLabel \*MessageLabel;

TButton \*LogInButton;

TButton \*CanselButton;

void \_\_fastcall LogInButtonClick(TObject \*Sender);

void \_\_fastcall CanselButtonClick(TObject \*Sender);

void \_\_fastcall PasswordEditClick(TObject \*Sender);

void \_\_fastcall FormShow(TObject \*Sender);

void \_\_fastcall FormCloseQuery(TObject \*Sender, bool &CanClose);

private:

bool flag; // User declarations

public: // User declarations

\_\_fastcall TLogInFormUnit(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TLogInFormUnit \*LogInFormUnit;

//---------------------------------------------------------------------------

#endif

**LogInForm.cpp**

#include <vcl.h>

#pragma hdrstop

#include "LogInForm.h"

#include "DataModuleUnit.h"

#include "MainUnit.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TLogInFormUnit \*LogInFormUnit;

//---------------------------------------------------------------------------

\_\_fastcall TLogInFormUnit::TLogInFormUnit(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TLogInFormUnit::LogInButtonClick(TObject \*Sender)

{

flag = true;

AnsiString key = "j8bKDwVgVgMGXkFRGkJ3KKk8kNHu7S";

AnsiString user;

AnsiString pass;

AnsiString statusbar;

user = UsernameEdit->Text.SetLength(30).c\_str();

statusbar = UsernameEdit->Text.SetLength(30).c\_str();

pass = PasswordEdit->Text.SetLength(30).c\_str();

int i;

for(i=1;i<user.Length() + 1;i++)

user[i] = user[i] + key[i];

for(i=1;i<pass.Length() + 1;i++)

pass[i] = pass[i] + key[i];

DM->LogInQuery->Close();

DM->LogInQuery->ParamByName("PUSER")->AsString = user;

DM->LogInQuery->ParamByName("PPASS")->AsString = pass;

DM->LogInQuery->Open();

if(DM->LogInQuery->FieldByName("ID")->IsNull)

{

MessageLabel->Caption = "Грешно потребителско име или парола!";

return;

}

MainFormUnit->MainStatusBar->Panels->Items[0]->Text = "Service App running";

MainFormUnit->MainStatusBar->Panels->Items[1]->Text = "Online: " + statusbar;

MainFormUnit->MainStatusBar->Panels->Items[2]->Text =

DM->LogInQuery->FieldByName("FIRST\_NAME")->AsString + " " +

DM->LogInQuery->FieldByName("FAM\_NAME")->AsString + " ";

Close();

UsernameEdit->Text="";

PasswordEdit->Text="";

}

//---------------------------------------------------------------------------

void \_\_fastcall TLogInFormUnit::CanselButtonClick(TObject \*Sender)

{

Application->Terminate();

}

//---------------------------------------------------------------------------

void \_\_fastcall TLogInFormUnit::PasswordEditClick(TObject \*Sender)

{

MessageLabel->Caption = "";

}

//---------------------------------------------------------------------------

void \_\_fastcall TLogInFormUnit::FormShow(TObject \*Sender)

{

bool flag = false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TLogInFormUnit::FormCloseQuery(TObject \*Sender,

bool &CanClose)

{

if(!flag)

CanClose = false;

}

//---------------------------------------------------------------------------

**CreateAccForm.h**

#ifndef CreateAccFormH

#define CreateAccFormH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

//---------------------------------------------------------------------------

class TCreateAccFormUnit : public TForm

{

\_\_published: // IDE-managed Components

TEdit \*EmployeeNameEdit;

TButton \*SelectEmployeeButton;

TLabel \*EmployeeNameLabel;

TEdit \*EmployeeFamEdit;

TLabel \*EmployeeFamLabel;

TEdit \*UsernameEdit;

TLabel \*Label3;

TLabel \*Label4;

TEdit \*PasswordEdit;

TButton \*Button1;

TButton \*Button2;

void \_\_fastcall SelectEmployeeButtonClick(TObject \*Sender);

void \_\_fastcall Button1Click(TObject \*Sender);

void \_\_fastcall FormClose(TObject \*Sender, TCloseAction &Action);

void \_\_fastcall Button2Click(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TCreateAccFormUnit(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TCreateAccFormUnit \*CreateAccFormUnit;

//---------------------------------------------------------------------------

#endif

**CreateAccForm.cpp**

#include <vcl.h>

#pragma hdrstop

#include "CreateAccForm.h"

#include "EmployeeForm.h"

#include "DataModuleUnit.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TCreateAccFormUnit \*CreateAccFormUnit;

//---------------------------------------------------------------------------

\_\_fastcall TCreateAccFormUnit::TCreateAccFormUnit(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TCreateAccFormUnit::SelectEmployeeButtonClick(

TObject \*Sender)

{

EmployeeFormUnit->EmpolyeeButtonSelectAcc->Visible = true;

EmployeeFormUnit->ShowModal();

}

//---------------------------------------------------------------------------

void \_\_fastcall TCreateAccFormUnit::Button1Click(TObject \*Sender)

{

AnsiString key = "j8bKDwVgVgMGXkFRGkJ3KKk8kNHu7S";

AnsiString user;

AnsiString pass;

user = UsernameEdit->Text.SetLength(30).c\_str();

pass = PasswordEdit->Text.SetLength(30).c\_str();

int i;

for(i=1;i<user.Length() + 1;i++)

user[i] = user[i] + key[i];

for(i=1;i<pass.Length() + 1;i++)

pass[i] = pass[i] + key[i];

DM->EmployeeTBL->Edit();

DM->EmployeeTBL->FieldByName("USERNAME")->AsString = user;

DM->EmployeeTBL->FieldByName("USER\_PASSWORD")->AsString = pass;

DM->EmployeeTBL->Post();

CreateAccFormUnit->Close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TCreateAccFormUnit::FormClose(TObject \*Sender,

TCloseAction &Action)

{

UsernameEdit->Text="";

PasswordEdit->Text="";

}

//---------------------------------------------------------------------------

void \_\_fastcall TCreateAccFormUnit::Button2Click(TObject \*Sender)

{

CreateAccFormUnit->Close();

}

//---------------------------------------------------------------------------

**BrandForm.h**

#ifndef BrandFormH

#define BrandFormH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <DBCtrls.hpp>

#include <DBGrids.hpp>

#include <ExtCtrls.hpp>

#include <Grids.hpp>

//---------------------------------------------------------------------------

class TBrandFormUnit : public TForm

{

\_\_published: // IDE-managed Components

TDBNavigator \*BrandDBNavigator;

TDBGrid \*BrandGrid;

TPanel \*BrandPanel;

TButton \*BrandButton;

void \_\_fastcall BrandGridExit(TObject \*Sender);

void \_\_fastcall BrandButtonClick(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TBrandFormUnit(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TBrandFormUnit \*BrandFormUnit;

//---------------------------------------------------------------------------

#endif

**BrandForm.cpp**

#include <vcl.h>

#pragma hdrstop

#include "BrandForm.h"

#include "DataModuleUnit.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TBrandFormUnit \*BrandFormUnit;

//---------------------------------------------------------------------------

\_\_fastcall TBrandFormUnit::TBrandFormUnit(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TBrandFormUnit::BrandGridExit(TObject \*Sender)

{

if(DM->BrandTBL->State==dsInsert || DM->BrandTBL->State==dsEdit)

DM->BrandTBL->Post();

}

//---------------------------------------------------------------------------

void \_\_fastcall TBrandFormUnit::BrandButtonClick(TObject \*Sender)

{

BrandFormUnit->Close();

}

//---------------------------------------------------------------------------

**CategoryForm.h**

#ifndef CategoryFormH

#define CategoryFormH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <DBCtrls.hpp>

#include <DBGrids.hpp>

#include <ExtCtrls.hpp>

#include <Grids.hpp>

//---------------------------------------------------------------------------

class TCategoryFormUnit : public TForm

{

\_\_published: // IDE-managed Components

TDBNavigator \*CategoriesDBNavigator;

TDBGrid \*CategoriesGrid;

TPanel \*CategoriesPanel;

TButton \*CategoriesButton;

void \_\_fastcall CategoriesButtonClick(TObject \*Sender);

void \_\_fastcall CategoriesGridExit(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TCategoryFormUnit(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TCategoryFormUnit \*CategoryFormUnit;

//---------------------------------------------------------------------------

#endif

**CategoryForm.cpp**

#include <vcl.h>

#pragma hdrstop

#include "CategoryForm.h"

#include "DataModuleUnit.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TCategoryFormUnit \*CategoryFormUnit;

//---------------------------------------------------------------------------

\_\_fastcall TCategoryFormUnit::TCategoryFormUnit(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TCategoryFormUnit::CategoriesButtonClick(TObject \*Sender)

{

CategoryFormUnit->Close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TCategoryFormUnit::CategoriesGridExit(TObject \*Sender)

{

if(DM->CategoryTBL->State==dsInsert || DM->CategoryTBL->State==dsEdit)

DM->CategoryTBL->Post();

}

//---------------------------------------------------------------------------

**EmployeeForm.h**

#ifndef EmployeeFormH

#define EmployeeFormH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <DBGrids.hpp>

#include <Grids.hpp>

#include <ExtCtrls.hpp>

#include <ComCtrls.hpp>

#include <DBCtrls.hpp>

//---------------------------------------------------------------------------

class TEmployeeFormUnit : public TForm

{

\_\_published: // IDE-managed Components

TDBGrid \*EmployeeGrid;

TPanel \*EmployeePanel;

TEdit \*EmployeeEdit;

TComboBox \*EmployeeSortCB;

TLabel \*EmployeeLabelSort;

TLabel \*EmployeeLabelSearch;

TComboBox \*EmployeeSearchCB;

TDBNavigator \*EmployeeDBNavigator;

TButton \*EmployeeButtonSelect;

TButton \*EmpolyeeButtonSelectAcc;

void \_\_fastcall EmployeeSortCBSelect(TObject \*Sender);

void \_\_fastcall EmployeeGridExit(TObject \*Sender);

void \_\_fastcall EmployeeEditChange(TObject \*Sender);

void \_\_fastcall EmployeeButtonSelectClick(TObject \*Sender);

void \_\_fastcall FormClose(TObject \*Sender, TCloseAction &Action);

void \_\_fastcall EmpolyeeButtonSelectAccClick(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TEmployeeFormUnit(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TEmployeeFormUnit \*EmployeeFormUnit;

//---------------------------------------------------------------------------

#endif

**EmployeeForm.cpp**

#include <vcl.h>

#pragma hdrstop

#include "EmployeeForm.h"

#include "DataModuleUnit.h"

#include "EditPartForm.h"

#include "CreateAccForm.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TEmployeeFormUnit \*EmployeeFormUnit;

//---------------------------------------------------------------------------

\_\_fastcall TEmployeeFormUnit::TEmployeeFormUnit(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TEmployeeFormUnit::EmployeeSortCBSelect(TObject \*Sender)

{

switch(EmployeeSortCB->ItemIndex)

{

case 0: DM->EmployeeTBL->IndexName ="RDB$PRIMARY16";break;

case 1: DM->EmployeeTBL->IndexName ="EMPLOYEETBLFNNDX";break;

case 2: DM->EmployeeTBL->IndexName ="EMPLOYEETBLFAMNNDX";break;

case 3: DM->EmployeeTBL->IndexName ="EMPLOYEETBLPNNDX";break;

case 4: DM->EmployeeTBL->IndexName ="EMPLOYEETBLSPNNDX";break;

case 5: DM->EmployeeTBL->IndexName ="EMPLOYEETBLEMNDX";break;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TEmployeeFormUnit::EmployeeGridExit(TObject \*Sender)

{

if(DM->EmployeeTBL->State==dsInsert || DM->EmployeeTBL->State==dsEdit)

DM->EmployeeTBL->Post();

}

//---------------------------------------------------------------------------

void \_\_fastcall TEmployeeFormUnit::EmployeeEditChange(TObject \*Sender)

{

AnsiString FieldName ="FIRST\_NAME";

TLocateOptions Opts;

Opts.Clear();

Opts << loPartialKey;

Opts << loCaseInsensitive;

switch(EmployeeSearchCB->ItemIndex)

{

case 0: FieldName ="FIRST\_NAME";break;

case 1: FieldName ="FAM\_NAME";break;

case 2: FieldName ="PHONE\_NUMBER";break;

case 3: FieldName ="SERVICE\_PHONE\_NUMBER";break;

case 4: FieldName ="E\_MAIL";break;

}

DM->EmployeeTBL->Locate(FieldName, EmployeeEdit->Text, Opts);

}

//---------------------------------------------------------------------------

void \_\_fastcall TEmployeeFormUnit::EmployeeButtonSelectClick(

TObject \*Sender)

{

RepairEditFormUnit->REFUFirstNameEdit->Text =

DM->EmployeeTBL->FieldByName("FIRST\_NAME")->AsString;

RepairEditFormUnit->REFUFamNameEdit->Text =

DM->EmployeeTBL->FieldByName("FAM\_NAME")->AsString;

RepairEditFormUnit->REFUPhNumEdit->Text =

DM->EmployeeTBL->FieldByName("PHONE\_NUMBER")->AsString;

RepairEditFormUnit->REFUSrPhNumEdit->Text =

DM->EmployeeTBL->FieldByName("SERVICE\_PHONE\_NUMBER")->AsString;

RepairEditFormUnit->REFUEmailEdit->Text =

DM->EmployeeTBL->FieldByName("E\_MAIL")->AsString;

RepairEditFormUnit->REFUPossitionEdit->Text =

DM->EmployeeTBL->FieldByName("POSSITION")->AsString;

EmployeeFormUnit->Close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TEmployeeFormUnit::FormClose(TObject \*Sender,

TCloseAction &Action)

{

if(EmployeeButtonSelect->Visible == true)

EmployeeButtonSelect->Visible = false;

if(EmpolyeeButtonSelectAcc->Visible == true)

EmpolyeeButtonSelectAcc->Visible = false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TEmployeeFormUnit::EmpolyeeButtonSelectAccClick(

TObject \*Sender)

{

CreateAccFormUnit->EmployeeNameEdit->Text =

DM->EmployeeTBL->FieldByName("FIRST\_NAME")->AsString;

CreateAccFormUnit->EmployeeFamEdit->Text =

DM->EmployeeTBL->FieldByName("FAM\_NAME")->AsString;

EmployeeFormUnit->Close();

}

//---------------------------------------------------------------------------

**EngineForm.h**

#ifndef EngineFormH

#define EngineFormH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <DBCtrls.hpp>

#include <DBGrids.hpp>

#include <ExtCtrls.hpp>

#include <Grids.hpp>

//---------------------------------------------------------------------------

class TEngineFormUnit : public TForm

{

\_\_published: // IDE-managed Components

TDBNavigator \*EngineDBNavigator;

TDBGrid \*EngineGrid;

TPanel \*EnginePanel;

TButton \*EngineButton;

void \_\_fastcall EngineGridExit(TObject \*Sender);

void \_\_fastcall EngineButtonClick(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TEngineFormUnit(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TEngineFormUnit \*EngineFormUnit;

//---------------------------------------------------------------------------

#endif

**EngineForm.cpp**

#include <vcl.h>

#pragma hdrstop

#include "EngineForm.h"

#include "DataModuleUnit.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TEngineFormUnit \*EngineFormUnit;

//---------------------------------------------------------------------------

\_\_fastcall TEngineFormUnit::TEngineFormUnit(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TEngineFormUnit::EngineGridExit(TObject \*Sender)

{

if(DM->EngineTypeTBL->State==dsInsert || DM->EngineTypeTBL->State==dsEdit)

DM->EngineTypeTBL->Post();

}

//---------------------------------------------------------------------------

void \_\_fastcall TEngineFormUnit::EngineButtonClick(TObject \*Sender)

{

EngineFormUnit->Close();

}

//---------------------------------------------------------------------------

**ItemForm.h**

#ifndef ItemFormH

#define ItemFormH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <DBCtrls.hpp>

#include <DBGrids.hpp>

#include <ExtCtrls.hpp>

#include <Grids.hpp>

//---------------------------------------------------------------------------

class TItemNomFormUnit : public TForm

{

\_\_published: // IDE-managed Components

TPanel \*ItemNomPanel;

TLabel \*ServiceLabelSort;

TLabel \*ItemNomLabelSearch;

TEdit \*ItemNomEdit;

TComboBox \*ItemNomSortCB;

TButton \*ItemNomButtonSelect;

TDBGrid \*ItemNomGrid;

TDBNavigator \*ItemNomDBNavigator;

void \_\_fastcall ItemNomSortCBSelect(TObject \*Sender);

void \_\_fastcall ItemNomEditChange(TObject \*Sender);

void \_\_fastcall FormClose(TObject \*Sender, TCloseAction &Action);

void \_\_fastcall ItemNomButtonSelectClick(TObject \*Sender);

void \_\_fastcall ItemNomGridExit(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TItemNomFormUnit(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TItemNomFormUnit \*ItemNomFormUnit;

//---------------------------------------------------------------------------

#endif

**ItemForm.cpp**

#include <vcl.h>

#pragma hdrstop

#include "ItemForm.h"

#include "DataModuleUnit.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TItemNomFormUnit \*ItemNomFormUnit;

//---------------------------------------------------------------------------

\_\_fastcall TItemNomFormUnit::TItemNomFormUnit(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TItemNomFormUnit::ItemNomSortCBSelect(TObject \*Sender)

{

switch(ItemNomSortCB->ItemIndex)

{

case 0: DM->ItemNomTBL->IndexName ="RDB$PRIMARY40";break;

case 1: DM->ItemNomTBL->IndexName ="ITEMNAMENDX";break;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TItemNomFormUnit::ItemNomEditChange(TObject \*Sender)

{

TLocateOptions Opts;

Opts.Clear();

Opts << loPartialKey;

Opts << loCaseInsensitive;

DM->ServiceTBL->Locate("NAME", ItemNomEdit->Text, Opts);

}

//---------------------------------------------------------------------------

void \_\_fastcall TItemNomFormUnit::FormClose(TObject \*Sender,

TCloseAction &Action)

{

if(ItemNomButtonSelect->Visible == true)

ItemNomButtonSelect->Visible = false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TItemNomFormUnit::ItemNomButtonSelectClick(TObject \*Sender)

{

DM->ItemTBL->Append();

DM->ItemTBL->FieldByName("SERVICE\_ID")->AsInteger=

DM->ServiceTBL->FieldByName("ID")->AsInteger;

DM->ItemTBL->FieldByName("ITEM\_NOMENCLATURE\_ID")->AsInteger=

DM->ItemNomTBL->FieldByName("ID")->AsInteger;

DM->ItemTBL->FieldByName("PRICE")->AsInteger= 0;

DM->ItemTBL->FieldByName("QUANTITY")->AsInteger= 0;

DM->ItemTBL->Post();

ItemNomFormUnit->Close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TItemNomFormUnit::ItemNomGridExit(TObject \*Sender)

{

if(DM->ItemNomTBL->State==dsInsert || DM->ItemNomTBL->State==dsEdit)

DM->ItemNomTBL->Post();

}

//---------------------------------------------------------------------------

**ManufacturerForm.h**

#ifndef ManufacturerFormH

#define ManufacturerFormH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <DBCtrls.hpp>

#include <DBGrids.hpp>

#include <ExtCtrls.hpp>

#include <Grids.hpp>

//---------------------------------------------------------------------------

class TManufacturerFormUnit : public TForm

{

\_\_published: // IDE-managed Components

TDBNavigator \*ManufacturerDBNavigator;

TDBGrid \*ManufacturerGrid;

TPanel \*ManufacturerPanel;

TButton \*ManufacturerButton;

void \_\_fastcall ManufacturerGridExit(TObject \*Sender);

void \_\_fastcall ManufacturerButtonClick(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TManufacturerFormUnit(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TManufacturerFormUnit \*ManufacturerFormUnit;

//---------------------------------------------------------------------------

#endif

**ManufacturerForm.cpp**

#include <vcl.h>

#pragma hdrstop

#include "ManufacturerForm.h"

#include "DataModuleUnit.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TManufacturerFormUnit \*ManufacturerFormUnit;

//---------------------------------------------------------------------------

\_\_fastcall TManufacturerFormUnit::TManufacturerFormUnit(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TManufacturerFormUnit::ManufacturerGridExit(

TObject \*Sender)

{

if(DM->ManufacturerTBL->State==dsInsert || DM->ManufacturerTBL->State==dsEdit)

DM->ManufacturerTBL->Post();

}

//---------------------------------------------------------------------------

void \_\_fastcall TManufacturerFormUnit::ManufacturerButtonClick(

TObject \*Sender)

{

ManufacturerFormUnit->Close();

}

//---------------------------------------------------------------------------

**MesureForm.h**

#ifndef MesureFormH

#define MesureFormH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <DBCtrls.hpp>

#include <DBGrids.hpp>

#include <ExtCtrls.hpp>

#include <Grids.hpp>

//---------------------------------------------------------------------------

class TMesureFormUnit : public TForm

{

\_\_published: // IDE-managed Components

TDBNavigator \*MesureDBNavigator;

TDBGrid \*MesureGrid;

TPanel \*MesurePanel;

TButton \*MesureButton;

void \_\_fastcall MesureGridExit(TObject \*Sender);

void \_\_fastcall MesureButtonClick(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TMesureFormUnit(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TMesureFormUnit \*MesureFormUnit;

//---------------------------------------------------------------------------

#endif

**MesureForm.cpp**

#include <vcl.h>

#pragma hdrstop

#include "MesureForm.h"

#include "DataModuleUnit.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TMesureFormUnit \*MesureFormUnit;

//---------------------------------------------------------------------------

\_\_fastcall TMesureFormUnit::TMesureFormUnit(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TMesureFormUnit::MesureGridExit(TObject \*Sender)

{

if(DM->MesureTBL->State==dsInsert || DM->MesureTBL->State==dsEdit)

DM->MesureTBL->Post();

}

//---------------------------------------------------------------------------

void \_\_fastcall TMesureFormUnit::MesureButtonClick(TObject \*Sender)

{

MesureFormUnit->Close();

}

//---------------------------------------------------------------------------

**OwnerForm.h**

#include <vcl.h>

#pragma hdrstop

#include "OwnerForm.h"

#include "DataModuleUnit.h"

#include "EditVehicleForm.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TOwnerFormUnit \*OwnerFormUnit;

//---------------------------------------------------------------------------

\_\_fastcall TOwnerFormUnit::TOwnerFormUnit(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TOwnerFormUnit::OwnerSortCBSelect(TObject \*Sender)

{

switch(OwnerSortCB->ItemIndex)

{

case 0: DM->OwnerTBL->IndexName ="RDB$PRIMARY1";break;

case 1: DM->OwnerTBL->IndexName ="OWNENAMERNDX";break;

case 2: DM->OwnerTBL->IndexName ="OWNERBENDX";break;

case 3: DM->OwnerTBL->IndexName ="OWNERMOLNDX";break;

case 4: DM->OwnerTBL->IndexName ="OWNERPNNDX";break;

case 5: DM->OwnerTBL->IndexName ="OWNEREMNDX";break;

case 6: DM->OwnerTBL->IndexName ="OWNERADDNDX";break;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TOwnerFormUnit::OwnerEditChange(TObject \*Sender)

{

AnsiString FieldName ="NAME";

TLocateOptions Opts;

Opts.Clear();

Opts << loPartialKey;

Opts << loCaseInsensitive;

switch(OwnerSearchCB->ItemIndex)

{

case 0: FieldName ="NAME";break;

case 1: FieldName ="BUL\_EGN";break;

case 2: FieldName ="MOL";break;

case 3: FieldName ="PHONE\_NUMBER";break;

case 4: FieldName ="E\_MAIL";break;

case 5: FieldName ="ADDRESS";break;

}

DM->OwnerTBL->Locate(FieldName, OwnerEdit->Text, Opts);

}

//---------------------------------------------------------------------------

void \_\_fastcall TOwnerFormUnit::DBGrid1Exit(TObject \*Sender)

{

if(DM->OwnerTBL->State==dsInsert || DM->OwnerTBL->State==dsEdit)

DM->OwnerTBL->Post();

}

//---------------------------------------------------------------------------

void \_\_fastcall TOwnerFormUnit::OwnerButtonSelectClick(TObject \*Sender)

{

EditVehicleFormUnit->EVFUNameEdit->Text =

DM->OwnerTBL->FieldByName("NAME")->AsString;

EditVehicleFormUnit->EVFUEgnBulEdit->Text =

DM->OwnerTBL->FieldByName("BUL\_EGN")->AsString;

EditVehicleFormUnit->EVFUMolEdit->Text =

DM->OwnerTBL->FieldByName("MOL")->AsString;

EditVehicleFormUnit->EVFUTelNumEdit->Text =

DM->OwnerTBL->FieldByName("PHONE\_NUMBER")->AsString;

EditVehicleFormUnit->EVFUEmailEdit->Text =

DM->OwnerTBL->FieldByName("E\_MAIL")->AsString;

EditVehicleFormUnit->EVFUAddressEdit->Text =

DM->OwnerTBL->FieldByName("ADDRESS")->AsString;

OwnerFormUnit->Close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TOwnerFormUnit::FormClose(TObject \*Sender,

TCloseAction &Action)

{

if(OwnerButtonSelect->Visible == true)

OwnerButtonSelect->Visible = false;

}

//---------------------------------------------------------------------------

**PartForm.h**

#ifndef PartFormH

#define PartFormH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <ComCtrls.hpp>

#include <DBGrids.hpp>

#include <ExtCtrls.hpp>

#include <Grids.hpp>

#include <DBCtrls.hpp>

//---------------------------------------------------------------------------

class TPartFormUnit : public TForm

{

\_\_published: // IDE-managed Components

TPanel \*PartPanel;

TLabel \*PartLabelSort;

TLabel \*PartSearch;

TEdit \*PartEdit;

TComboBox \*PartSortCB;

TDBNavigator \*PartDBNavigator;

TDBGrid \*DBGrid1;

void \_\_fastcall PartGridExit(TObject \*Sender);

void \_\_fastcall PartSortCBSelect(TObject \*Sender);

void \_\_fastcall PartEditChange(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TPartFormUnit(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TPartFormUnit \*PartFormUnit;

//---------------------------------------------------------------------------

#endif

**PartForm.cpp**

#include <vcl.h>

#pragma hdrstop

#include "PartForm.h"

#include "DataModuleUnit.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TPartFormUnit \*PartFormUnit;

//---------------------------------------------------------------------------

\_\_fastcall TPartFormUnit::TPartFormUnit(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TPartFormUnit::PartGridExit(TObject \*Sender)

{

if(DM->ItemNomTBL->State==dsInsert || DM->ItemNomTBL->State==dsEdit)

DM->ItemNomTBL->Post();

}

//---------------------------------------------------------------------------

void \_\_fastcall TPartFormUnit::PartSortCBSelect(TObject \*Sender)

{

switch(PartSortCB->ItemIndex)

{

case 0: DM->ItemNomTBL->IndexName ="RDB$PRIMARY40";break;

case 1: DM->ItemNomTBL->IndexName ="ITEMNOMTBLNAMENDX";break;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TPartFormUnit::PartEditChange(TObject \*Sender)

{

TLocateOptions Opts;

Opts.Clear();

Opts << loPartialKey;

Opts << loCaseInsensitive;

DM->ItemNomTBL->Locate("NAME", PartEdit->Text, Opts);

}

//---------------------------------------------------------------------------

**PossitionForm.h**

#ifndef PossitionFormH

#define PossitionFormH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <DBCtrls.hpp>

#include <DBGrids.hpp>

#include <ExtCtrls.hpp>

#include <Grids.hpp>

//---------------------------------------------------------------------------

class TPossitionFormUnit : public TForm

{

\_\_published: // IDE-managed Components

TDBNavigator \*PossitionDBNavigator;

TDBGrid \*PossitionGrid;

TPanel \*PossitionPanel;

TButton \*PossitionButton;

void \_\_fastcall PossitionGridExit(TObject \*Sender);

void \_\_fastcall PossitionButtonClick(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TPossitionFormUnit(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TPossitionFormUnit \*PossitionFormUnit;

//---------------------------------------------------------------------------

#endif

**PossitionForm.cpp**

#include <vcl.h>

#pragma hdrstop

#include "PossitionForm.h"

#include "DataModuleUnit.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TPossitionFormUnit \*PossitionFormUnit;

//---------------------------------------------------------------------------

\_\_fastcall TPossitionFormUnit::TPossitionFormUnit(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TPossitionFormUnit::PossitionGridExit(TObject \*Sender)

{

if(DM->PossitionTBL->State==dsInsert || DM->PossitionTBL->State==dsEdit)

DM->PossitionTBL->Post();

}

//---------------------------------------------------------------------------

void \_\_fastcall TPossitionFormUnit::PossitionButtonClick(TObject \*Sender)

{

PossitionFormUnit->Close();

}

//---------------------------------------------------------------------------

**RepairForm.h**

#ifndef RepairFormH

#define RepairFormH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <ComCtrls.hpp>

#include <DBGrids.hpp>

#include <ExtCtrls.hpp>

#include <Grids.hpp>

#include <DBCtrls.hpp>

//---------------------------------------------------------------------------

class TRepairShopFormUnit : public TForm

{

\_\_published: // IDE-managed Components

TPanel \*RShopPanel;

TLabel \*RShopLabelSort;

TLabel \*RShopLabelSearch;

TEdit \*RShopEdit;

TComboBox \*RShopSortCB;

TButton \*RShopButtonSelect;

TDBNavigator \*RShopDBNavigator;

TDBGrid \*RShopGrid;

void \_\_fastcall RShopButtonSelectClick(TObject \*Sender);

void \_\_fastcall RShopEditChange(TObject \*Sender);

void \_\_fastcall RShopSortCBSelect(TObject \*Sender);

void \_\_fastcall RShopGridColExit(TObject \*Sender);

void \_\_fastcall FormClose(TObject \*Sender, TCloseAction &Action);

private: // User declarations

public: // User declarations

\_\_fastcall TRepairShopFormUnit(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TRepairShopFormUnit \*RepairShopFormUnit;

//---------------------------------------------------------------------------

#endif

**RepairForm.cpp**

#include <vcl.h>

#pragma hdrstop

#include "RepairForm.h"

#include "DataModuleUnit.h"

#include "EditPartForm.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TRepairShopFormUnit \*RepairShopFormUnit;

//---------------------------------------------------------------------------

\_\_fastcall TRepairShopFormUnit::TRepairShopFormUnit(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TRepairShopFormUnit::RShopButtonSelectClick(

TObject \*Sender)

{

RepairEditFormUnit->REFUShopNameEdit->Text =

DM->RShopTBL->FieldByName("NAME")->AsString;

RepairEditFormUnit->REFUShopAddressEdit->Text =

DM->RShopTBL->FieldByName("ADDRESS")->AsString;

RepairShopFormUnit->Close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TRepairShopFormUnit::RShopEditChange(TObject \*Sender)

{

TLocateOptions Opts;

Opts.Clear();

Opts << loPartialKey;

Opts << loCaseInsensitive;

DM->ServiceTBL->Locate("NAME", RShopEdit->Text, Opts);

}

//---------------------------------------------------------------------------

void \_\_fastcall TRepairShopFormUnit::RShopSortCBSelect(TObject \*Sender)

{

switch(RShopSortCB->ItemIndex)

{

case 0: DM->RShopTBL->IndexName ="RDB$PRIMARY14";break;

case 1: DM->RShopTBL->IndexName ="RSHOPNAMENDX";break;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TRepairShopFormUnit::RShopGridColExit(TObject \*Sender)

{

if(DM->RShopTBL->State==dsInsert || DM->RShopTBL->State==dsEdit)

DM->RShopTBL->Post();

}

//---------------------------------------------------------------------------

void \_\_fastcall TRepairShopFormUnit::FormClose(TObject \*Sender,

TCloseAction &Action)

{

if(RShopButtonSelect->Visible == true)

RShopButtonSelect->Visible = false;

}

//---------------------------------------------------------------------------

**ServiceForm.h**

#ifndef ServiceFormH

#define ServiceFormH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <ComCtrls.hpp>

#include <DBGrids.hpp>

#include <ExtCtrls.hpp>

#include <Grids.hpp>

#include <DBCtrls.hpp>

//---------------------------------------------------------------------------

class TServiceFormUnit : public TForm

{

\_\_published: // IDE-managed Components

TPanel \*ServicePanel;

TLabel \*ServiceLabelSort;

TLabel \*ServiceLabelSearch;

TEdit \*ServiceEdit;

TComboBox \*ServiceSortCB;

TDBNavigator \*ServiceDBNavigator;

TDBGrid \*ServiceGrid;

TButton \*ServiceButtonSelect;

void \_\_fastcall ServiceSortCBSelect(TObject \*Sender);

void \_\_fastcall ServiceGridExit(TObject \*Sender);

void \_\_fastcall ServiceEditChange(TObject \*Sender);

void \_\_fastcall ServiceButtonSelectClick(TObject \*Sender);

void \_\_fastcall FormClose(TObject \*Sender, TCloseAction &Action);

private: // User declarations

public: // User declarations

\_\_fastcall TServiceFormUnit(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TServiceFormUnit \*ServiceFormUnit;

//---------------------------------------------------------------------------

#endif

**ServiceForm.cpp**

#include <vcl.h>

#pragma hdrstop

#include "ServiceForm.h"

#include "DataModuleUnit.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TServiceFormUnit \*ServiceFormUnit;

//---------------------------------------------------------------------------

\_\_fastcall TServiceFormUnit::TServiceFormUnit(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TServiceFormUnit::ServiceSortCBSelect(TObject \*Sender)

{

switch(ServiceSortCB->ItemIndex)

{

case 0: DM->AllServiceTBL->IndexName ="RDB$PRIMARY22";break;

case 1: DM->AllServiceTBL->IndexName ="SERVICETBLNAMENDX";break;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TServiceFormUnit::ServiceGridExit(TObject \*Sender)

{

if(DM->AllServiceTBL->State==dsInsert || DM->AllServiceTBL->State==dsEdit)

DM->AllServiceTBL->Post();

}

//---------------------------------------------------------------------------

void \_\_fastcall TServiceFormUnit::ServiceEditChange(TObject \*Sender)

{

TLocateOptions Opts;

Opts.Clear();

Opts << loPartialKey;

Opts << loCaseInsensitive;

DM->AllServiceTBL->Locate("NAME", ServiceEdit->Text, Opts);

}

//---------------------------------------------------------------------------

void \_\_fastcall TServiceFormUnit::ServiceButtonSelectClick(TObject \*Sender)

{

DM->RepairServiceTBL->Append();

DM->RepairServiceTBL->FieldByName("SERVICE\_ID")->AsInteger=

DM->AllServiceTBL->FieldByName("ID")->AsInteger;

DM->RepairServiceTBL->FieldByName("REPAIR\_ID")->AsInteger=

DM->RepairTBL->FieldByName("ID")->AsInteger;

DM->RepairServiceTBL->FieldByName("TOTAL\_PRICE")->AsInteger= 0;

DM->RepairServiceTBL->Post();

DM->ItemTBL->Refresh();

ServiceFormUnit->Close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TServiceFormUnit::FormClose(TObject \*Sender,

TCloseAction &Action)

{

if(ServiceButtonSelect->Visible == true)

ServiceButtonSelect->Visible = false;

}

//---------------------------------------------------------------------------

**TypeForm.h**

#ifndef TypeFormH

#define TypeFormH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <DBCtrls.hpp>

#include <DBGrids.hpp>

#include <ExtCtrls.hpp>

#include <Grids.hpp>

//---------------------------------------------------------------------------

class TTypeFormUnit : public TForm

{

\_\_published: // IDE-managed Components

TDBNavigator \*TypeDBNavigator;

TDBGrid \*TypeGrid;

TPanel \*TypePanel;

TButton \*TypeButton;

void \_\_fastcall TypeButtonClick(TObject \*Sender);

void \_\_fastcall TypeGridExit(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TTypeFormUnit(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TTypeFormUnit \*TypeFormUnit;

//---------------------------------------------------------------------------

#endif

**TypeForm.cpp**

#include <vcl.h>

#pragma hdrstop

#include "TypeForm.h"

#include "DataModuleUnit.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TTypeFormUnit \*TypeFormUnit;

//---------------------------------------------------------------------------

\_\_fastcall TTypeFormUnit::TTypeFormUnit(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TTypeFormUnit::TypeButtonClick(TObject \*Sender)

{

TypeFormUnit->Close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TTypeFormUnit::TypeGridExit(TObject \*Sender)

{

if(DM->VehicleTypeTBL->State==dsInsert || DM->VehicleTypeTBL->State==dsEdit)

DM->VehicleTypeTBL->Post();

}

//---------------------------------------------------------------------------

**VehicleForm.h**

#ifndef VehicleFormH

#define VehicleFormH

//---------------------------------------------------------------------------

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <ComCtrls.hpp>

#include <DBGrids.hpp>

#include <ExtCtrls.hpp>

#include <Grids.hpp>

#include <DBCtrls.hpp>

//---------------------------------------------------------------------------

class TVehicleNomFormUnit : public TForm

{

\_\_published: // IDE-managed Components

TPanel \*VehicleNomPanel;

TDBGrid \*VehicleNomGrid;

TDBNavigator \*VehicleNomDBNavigator;

TLabel \*VehicleNomLabelSort;

TComboBox \*VehicleNomSortCB;

TLabel \*VehicleNomrLabelSearch;

TComboBox \*VehicleNomSearchCB;

TEdit \*VehicleNomEdit;

TButton \*VehicleNomButtonSelect;

void \_\_fastcall VehicleNomSortCBSelect(TObject \*Sender);

void \_\_fastcall VehicleNomEditChange(TObject \*Sender);

void \_\_fastcall VehicleNomGridExit(TObject \*Sender);

void \_\_fastcall VehicleNomButtonSelectClick(TObject \*Sender);

void \_\_fastcall FormClose(TObject \*Sender, TCloseAction &Action);

private: // User declarations

public: // User declarations

\_\_fastcall TVehicleNomFormUnit(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TVehicleNomFormUnit \*VehicleNomFormUnit;

//---------------------------------------------------------------------------

#endif

**VehicleForm.cpp**

#include <vcl.h>

#pragma hdrstop

#include "VehicleForm.h"

#include "DataModuleUnit.h"

#include "EditVehicleForm.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TVehicleNomFormUnit \*VehicleNomFormUnit;

//---------------------------------------------------------------------------

\_\_fastcall TVehicleNomFormUnit::TVehicleNomFormUnit(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TVehicleNomFormUnit::VehicleNomSortCBSelect(TObject \*Sender)

{

switch(VehicleNomSortCB->ItemIndex)

{

case 0: DM->VehicleNomTBL->IndexName ="RDB$PRIMARY6";break;

case 1: DM->VehicleNomTBL->IndexName ="VEHNOMMODELNDX";break;

case 2: DM->VehicleNomTBL->IndexName ="VEHNOMENGINEVOLNDX";break;

case 3: DM->VehicleNomTBL->IndexName ="VEHNOMLOADNDX";break;

case 4: DM->VehicleNomTBL->IndexName ="VEHNOMAXLESNDX";break;

case 5: DM->VehicleNomTBL->IndexName ="VEHNOMSEATSNDX";break;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TVehicleNomFormUnit::VehicleNomEditChange(TObject \*Sender)

{

AnsiString FieldName ="MODEL";

TLocateOptions Opts;

Opts.Clear();

Opts << loPartialKey;

Opts << loCaseInsensitive;

switch(VehicleNomSearchCB->ItemIndex)

{

case 0: FieldName ="MODEL";break;

case 1: FieldName ="ENGINE\_TYPE";break;

case 2: FieldName ="LOADABILITY";break;

case 3: FieldName ="AXLES\_NUMBER";break;

case 4: FieldName ="SEATS\_NUMBER";break;

}

DM->VehicleNomTBL->Locate(FieldName, VehicleNomEdit->Text, Opts);

}

//---------------------------------------------------------------------------

void \_\_fastcall TVehicleNomFormUnit::VehicleNomGridExit(TObject \*Sender)

{

if(DM->VehicleNomTBL->State==dsInsert || DM->VehicleNomTBL->State==dsEdit)

DM->VehicleNomTBL->Post();

}

//---------------------------------------------------------------------------

void \_\_fastcall TVehicleNomFormUnit::VehicleNomButtonSelectClick(

TObject \*Sender)

{

EditVehicleFormUnit->EVFUBrandEdit->Text =

DM->VehicleNomTBL->FieldByName("BRAND\_NAME")->AsString;

EditVehicleFormUnit->EVFUModelEdit->Text =

DM->VehicleNomTBL->FieldByName("MODEL")->AsString;

EditVehicleFormUnit->EVFUCategoryEdit->Text =

DM->VehicleNomTBL->FieldByName("CATEGORY\_NAME")->AsString;

EditVehicleFormUnit->EVFUEngineTypeEdit->Text =

DM->VehicleNomTBL->FieldByName("ENGINE\_NAME")->AsString;

EditVehicleFormUnit->EVFUTypeEdit->Text =

DM->VehicleNomTBL->FieldByName("TYPE\_NAME")->AsString;

EditVehicleFormUnit->EVFUEngineVolumeEdit->Text =

DM->VehicleNomTBL->FieldByName("ENGINE\_VOLUME")->AsString;

EditVehicleFormUnit->EVFULoadabilityEdit->Text =

DM->VehicleNomTBL->FieldByName("LOADABILITY")->AsString;

EditVehicleFormUnit->EVFUAxlesEdit->Text =

DM->VehicleNomTBL->FieldByName("AXLES\_NUMBER")->AsString;

EditVehicleFormUnit->EVFUSeatsEdit->Text =

DM->VehicleNomTBL->FieldByName("SEATS\_NUMBER")->AsString;

VehicleNomFormUnit->Close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TVehicleNomFormUnit::FormClose(TObject \*Sender,

TCloseAction &Action)

{

if(VehicleNomButtonSelect->Visible == true)

VehicleNomButtonSelect->Visible = false;

}