Лист утверждения

|  |
| --- |
| Федеральное государственное бюджетное образовательное учреждение  Вышего образования  Российский экономический университет иимени Г.В.Плеханова  Московский приборостроительный техникум  Специальность 09.02.05 Прикладная информатика (по отраслям)  Профессиональный модуль: ПМ 02 Разработка, внедрение и адаптация программного обеспечения отраслевой направленности  Учебная практика: УП 02.01 Разработка и эксплуатация информационных систем |

|  |  |  |
| --- | --- | --- |
| Заказчик:  И.М. Щаников \_\_\_\_\_\_\_\_  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2018 г. | Консультант:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ К.А. Минаев  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. |  |

Тема: «Разработка информационной системы: "Больница", реализация подсистем: Складские

|  |  |
| --- | --- |
| Подп и дата |  |
| Инв № дубл |  |
| Взам. инв.№ | Разработчики  студенты группы П-319  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Д.Э.Васильев  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/М.П. Макаров  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/В.А. Позднев |
| Подп и дата |  |
| Инд № подп |  |

помещения для медикаментов, Терапевтическое отделение, Карты

лечения, Запись на приём»

Текст программы

МПТ И-3-16 УП 02.01 РЭИС 08 18

Листов 61

|  |  |  |
| --- | --- | --- |
| Разработчики:  \_\_\_\_\_\_\_\_\_\_Д.А. Голованов  \_\_\_\_\_\_\_\_\_А.С. Гаврилочев  \_\_\_\_\_\_\_\_\_\_\_А.А. Игнатьев  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2018 г. | | |
|  | 2018 | ТП |

АННОТАЦИЯ

В данном документе будет приведено описание и текст готовых модулей базы данных, а также готовой программы для работы с БД

СОДЕРЖАНИЕ

[1. СТРУКТУРА КОДА БАЗЫ ДАННЫХ 4](#_Toc10450386)

[1.1. Описание объектов баз данных 4](#_Toc10450387)

[1.2. Скрипт объектов баз данных 12](#_Toc10450388)

[2. СТРУКТУРА КОДА ПРОГРАММЫ 47](#_Toc10450389)

[2.1. Описание модулей программы 47](#_Toc10450390)

[2.2. Код программных модулей 47](#_Toc10450391)

# 1. СТРУКТУРА КОДА БАЗЫ ДАННЫХ

## 1.1. Описание объектов баз данных

Таблица 1.1 – Описание объектов базы данных

|  |  |
| --- | --- |
| Название | Описание |
| 1 | 2 |
| Таблица LogChange | Таблица, используемая для создания логов изменений таблиц БД |
| Таблица Manufacturer | Таблица, содержащая информацию о производителе медикаментов |
| Таблица CategoryMedicament | Таблица, содержащая информацию о категории медикаментов |
| Таблица TypeWrite | Таблица, содержащая информацию о типе записи на приём |
| Таблица DataCitizen | Таблица, содержащая информацию о гражданине, проходящем лечение |
| Таблица TypeUse | Таблица, содержащая информацию о способе использования назначенных медикаментов |
| Таблица DirectoryDisease | Таблица, содержащая информацию о всех возможных болезнях, по которым ставится диагноз |
| Таблица SpecialityPersonal | Таблица, содержащая информацию о специальностях персонала |
| Таблица WorkShedule | Таблица, содержащая информацию о режимах работы персонала, такие как 2/2 и т.д. |
| Таблица Role | Таблица, содержащая информацию о ролях для ПП, у каждой роли свои права в использовании приложения |
| Таблица TypeDischarge | Таблица, содержащая информацию о типах выписки, такие как рабочая или в учебное заведение |
| Таблица Regulation | Таблица, содержащая информацию о правилах для определённой специальности |
| Таблица Medicament | Таблица, содержащая информацию о медикаментах |
| Таблица Personal | Таблица, содержащая информацию о сотрудниках больницы, их персональных данных |
| Таблица Storage | Таблица, содержащая информацию о складских ячейках и наличии в них определённых медикаментов |
| Таблица DeliveryMedicament | Таблица, содержащая информацию о поставках медикаментов |
| Таблица CategoriesDisease | Таблица, содержащая информацию о категориях болезней |
| Таблица FormWrite | Таблица, содержащая информацию о формах записи |
| Таблица WriteAppointment | Таблица, содержащая информацию о записи на приём, её времени и статусе приёма |
| 1 | 2 |
| Таблица TherapeuticDepartament | Таблица, содержащая информацию о палатах в терапевтическом отделении, палатах и занятых палатах |
| Таблица Diagnosis | Таблица, содержащая информацию о диагнозах, болезни и назначенных лекарствах |
| Таблица CardTreatments | Таблица, содержащая информацию о картах лечениях, назначенном диагнозе |
| Таблица Discharge | Таблица, содержащая информацию о выписке, её типе |
| Представление DischargeOnSchool | Виртуальная таблица, содержит информацию для создания документа по выписке в учебное заведение |
| Представление DischargeOnSchool | Виртуальная таблица, содержит информацию для создания документа по выписке на работу |
| Представление RoomInfo | Виртуальная таблица, содержит информацию для создания отчётного документа по палатам в терапевтическом отделении |
| Представление StorageStatus | Виртуальная таблица, содержит информацию для создания небольшой выписки о состоянии мест на складе и находящихся там медикаментов |
| Представление Talon | Виртуальная таблица, содержит информацию для создания печатного документа типа Талон, где содержится информация о времени приёма и пациенте |
| Представление DocumentForDeliveryMEdicament | Виртуальная таблица, содержит информацию для создания документа о поставках медикаментов в больницу и зачислению их на склад |
| Триггер DeleteOnPalateNewCitizen | Триггер для вычитания выписанного человека из палаты, где он проходил лечение |
| Триггер InsertOnPalateNewCitizen | Триггер, созданный для добавления человека в палату и вычитания одного места из общего количества мест в палате |
| Триггер NaznashenieMedicamentov | Триггер для списания медикаментов со склада при назначении диагноза в карту лечения |
| Триггер Tovar\_Prihod | Триггер, добавляющий в таблицу склада поставленный товар и его количество из таблицы поставки медикаментов |
| Триггер логирования dbo.changes\_CardTreatments\_trigger | Триггер для логирования изменений в таблице CardTreatments |
| Триггер логирования  dbo.changes\_CategoriesDisease\_trigger | Триггер для логирования изменений в таблице CategoriesDisease |
| Триггер логирования  dbo.changes\_CategoryOfMedicament\_trigger | Триггер для логирования изменений в таблице CategoryOfMedicament |
| Триггер логирования  dbo.changes\_DataCitizen\_trigger | Триггер для логирования изменений в таблице DataCitizen |
| Триггер логирования  dbo.changes\_DeliveryMedicament\_trigger | Триггер для логирования изменений в таблице DeliveryMedicament |
| Триггер логирования  dbo.changes\_Diagnosis\_trigger | Триггер для логирования изменений в таблице Diagnosis |
| Триггер логирования  dbo.changes\_DirectoryDisease\_trigger | Триггер для логирования изменений в таблице DirectoryDisease |
| Триггер логирования  dbo.changes\_Discharge\_trigger | Триггер для логирования изменений в таблице Discharge |
| 1 | 2 |
| Триггер логирования  dbo.changes\_FormWrite\_trigger | Триггер для логирования изменений в таблице FormWrite |
| Триггер логирования  dbo.changes\_Manufacturer\_trigger | Триггер для логирования изменений в таблице Manufacturer |
| Триггер логирования  dbo.changes\_Medicament\_trigger | Триггер для логирования изменений в таблице Medicament |
| Триггер логирования  dbo.changes\_Personal\_trigger | Триггер для логирования изменений в таблице Personal |
| Триггер логирования  dbo.changes\_Regulation\_trigger | Триггер для логирования изменений в таблице Regulation |
| Триггер логирования  dbo.changes\_Role\_trigger | Триггер для логирования изменений в таблице Role |
| Триггер логирования  dbo.changes\_SpeciatlityPersonal\_trigger | Триггер для логирования изменений в таблице SpeciatlityPersonal |
| Триггер логирования  dbo.changes\_Storage\_trigger | Триггер для логирования изменений в таблице Storage |
| Триггер логирования dbo.changes\_TherapeuticDepartament\_trigger | Триггер для логирования изменений в таблице TherapeuticDepartament |
| Триггер логирования  dbo.changes\_TypeDischarge\_trigger | Триггер для логирования изменений в таблице TypeDischarge |
| Триггер логирования  dbo.changes\_TypeUse\_trigger | Триггер для логирования изменений в таблице TypeUse |
| Триггер логирования  dbo.changes\_ TypeWrite \_trigger | Триггер для логирования изменений в таблице TypeWrite |
| Триггер логирования  dbo.changes\_ Workshedule \_trigger | Триггер для логирования изменений в таблице Workshedule |
| Триггер логирования  dbo.changes\_ WriteAppointment \_trigger | Триггер для логирования изменений в таблице WriteAppointment |
| Процедура Add\_Personal | Процедура для добавления нового сотрудника в таблицу Personal |
| Процедура UPD\_Personal | Процедура для обновления значений в таблице Personal |
| Процедура Add\_WorkShedule | Процедура для добавления нового рабочего графика |
| Процедура Add\_SpecialityPersonal | Процедура для добавления новой специализации персонала |
| Процедура Add\_Role | Процедура для добавления новой роли |
| Процедура UPD\_ WorkShedule | Процедура для изменения рабочего графика |
| Процедура UPD\_ SpecialityPersonal | Процедура для изменения специальности персонала |
| Процедура UPD\_ Role | Процедура для изменения роли |
| Процедура UPD\_ Personal\_For\_Admin | Процедура для изменения персонала в панели администрирования |
| Процедура logdel\_Workshedule | Процедура для логического удаления графика персонала |
| 1 | 2 |
| Процедура logdel\_Storage | Процедура для логического удаления складской ячейки |
| Процедура logdel\_SpeciatlityPersonal | Процедура для логического удаления специальности персонала |
| Процедура logdel\_Role | Процедура для логического удаления роли |
| Процедура logdel\_Personal | Процедура для логического удаления персонала |
| Процедура для поиска Search\_Role\_SpecialityPersonal | Процедура для поиска ролей во вкладке администрирования |
| Процедура для поиска Search\_Personal\_ForAdmin | Процедура для поиска персонала на вкладке администрирования |
| Процедура Select\_WorkShedule | Процедура для выборки данных из таблицы WorkShedule |
| Процедура Select\_SpecialityPersonal | Процедура для выборки данных из таблицы SpecialityPersonal |
| Процедура Select\_Role | Процедура для выборки данных из таблицы Role |
| Процедура Select\_Personal | Процедура для выборки данных из таблицы Personal |
| Функция Answer\_ZeroAmount\_CardTreatments | Функция для проверки карты лечения, есть ли невылеченные диагнозы |
| Процедура Select\_Diagnos\_For\_Name\_Disease | Процедура, выбирающая имена болезней в диагнозах |
| Процедура Select\_CardTreatments | Процедура, выбирающая карты лечения |
| Процедура logdel\_WriteAppointment | Процедура, осуществляющая логическое удаление записи на приём |
| Процедура logdel\_TypeUse | Процедура, осуществляющая логическое удаление TypeUse |
| Процедура logdel\_TypeDischarge | Процедура, осуществляющая логическое удаление TypeDischarge |
| Процедура logdel\_Discharge | Процедура, осуществляющая логическое удаление Discharge |
| Процедура logdel\_DirectoryDisease | Процедура, осуществляющая логическое удаление DirectoryDisease |
| 1 | 2 |
| Процедура logdel\_Diagnosis | Процедура, осуществляющая логическое удаление Diagnosis |
| Процедура logdel\_CardTreatments | Процедура, осуществляющая логическое удаление CardTreatments |
| Процедура UPD\_TypeUse | Процедура, осуществляющая изменение в TypeUse |
| Процедура UPD\_TypeDischarge | Процедура, осуществляющая изменение в TypeDischarge |
| Процедура UPD\_DirectoryDisease | Процедура, осуществляющая изменение в DirectoryDisease |
| Процедура UPD\_Diagnosis | Процедура, осуществляющая изменение в Diagnosis |
| Процедура UPD\_CardTreatments | Процедура, осуществляющая изменение в CardTreatments |
| Процедура Add\_TypeUse | Процедура, осуществляющая добавление в TypeUse |
| Процедура Add\_TypeDischarge | Процедура, осуществляющая добавление в TypeDischarge |
| Процедура Add\_Discharge | Процедура, осуществляющая добавление в Discharge |
| Процедура Add\_DirectoryDisease | Процедура, осуществляющая добавление в DirectoryDisease |
| Процедура Add\_Diagnosis | Процедура, осуществляющая добавление в Diagnosis |
| Процедура Add\_CardTreatments | Процедура, осуществляющая добавление в CardTreatments |
| Процедура Select\_TypeUse | Процедура, осуществляющая выборку в TypeUse |
| Процедура Select\_TypeDischarge\_Name\_NameDischarge | Процедура, осуществляющая выборку названий выписок в TypeDischarge |
| Процедура Select\_Name\_Disease | Процедура, осуществляющая выборку в DirectoryDisease |
| Процедура Select\_Medicament | Процедура, осуществляющая выборку в Medicament |
| 1 | 2 |
| Процедура Select\_Id\_Departament | Процедура, осуществляющая выборку в TherapeuticDepartament |
| Процедура Select\_Discharge | Процедура, осуществляющая выборку в Discharge |
| Процедура Select\_All\_Diagnos | Процедура, осуществляющая выборку всех диагнозов в Diagnosis |
| Процедура CardTreatments\_Diagnoz\_Select | Процедура, осуществляющая выборку диагнозов в CardTreatments |
| Процедура Select\_CardTreatments | Процедура, осуществляющая выборку в CardTreatments |
| Процедура Sum\_Diagnosis\_For\_WriteAppointment | Процедура для суммирования всех диагнозов для человека в WriteAppointment |
| Процедура Cured\_All\_CardTreatments | Процедура для постановки во всех картах лечения статуса лечения |
| Процедура Select\_Type\_Write | Процедура, осуществляющая выборку в TypeWrite |
| Процедура Select\_DataCitizen\_ONLI\_FIO | Процедура, осуществляющая выборку ФИО в таблице DataCitizen |
| Процедура Select\_DataCitizen | Процедура, осуществляющая выборку в DataCitizen |
| Процедура Select\_Date\_Appointment | Процедура, осуществляющая выборку дат в WriteAppointment |
| Процедура Select\_WriteAppointment | Процедура, осуществляющая выборку в WriteAppointment |
| Процедура Add\_DataCitizen | Процедура, осуществляющая добавление в DataCitizen |
| Процедура Add\_FormWrite | Процедура, осуществляющая добавление в FormWrite |
| Процедура Add\_TypeWrite | Процедура, осуществляющая добавление в TypeWrite |
| Процедура Add\_WriteAppointment | Процедура, осуществляющая добавление в WriteAppointment |
| Процедура UPD\_DataCitizen | Процедура, осуществляющая изменение в DataCitizen |
| 1 | 2 |
| Процедура UPD\_TypeWrite | Процедура, осуществляющая изменение в TypeWrite |
| Процедура UPD\_WriteAppointment | Процедура, осуществляющая изменение в WriteAppointment |
| Процедура logdel\_DataCitizen | Процедура, осуществляющая логическое удаление |
| Процедура logdel\_TypeWrite | Процедура, осуществляющая логическое удаление в TypeWrite |
| Процедура Search\_DataCitizen | Процедура, осуществляющая поиск в DataCitizen |
| Процедура Search\_WriteAppointment | Процедура, осуществляющая поиск в WriteAppointment |
| Процедура SELECT\_Regulation | Процедура, осуществляющая выборку правил в Regulation |
| Процедура SELECT\_Profile\_Personal | Процедура, осуществляющая выборку профилей персонала |
| Процедура Select\_Role\_Percon | Процедура, осуществляющая выборку Ролей персонала |
| Процедура UPD\_Personal | Процедура, осуществляющая изменение в таблице Personal |
| Процедура Select\_CategoryOfMedicament | Процедура, осуществляющая выборку в CategoryOfMedicament |
| Процедура Select\_DeliveryMedicament | Процедура, осуществляющая выборку поставок в DeliveryMedicament |
| Процедура Select\_Manufacturer | Процедура, осуществляющая выборку в Manufacturer |
| Процедура Select\_Medicament\_FOR\_Storage | Процедура, осуществляющая выборку медикаментов на складе |
| Процедура Select\_Medicament | Процедура, осуществляющая выборку медимакетов |
| Процедура Select\_Storage\_Id | Процедура, осуществляющая выборку ячеек склада в Storage |
| Процедура Add\_DeliveryMedicament | Процедура, осуществляющая добавление поставок в DeliveryMedicament |
| 1 | 2 |
| Процедура Add\_Manufacturer | Процедура, осуществляющая добавление в Manufacturer |
| Процедура Add\_Medicament | Процедура, осуществляющая добавление в Medicament |
| Процедура Add\_Storage | Процедура, осуществляющая добавление в Storage |
| Процедура UPD\_DeliveryMedicament | Процедура, осуществляющая изменение в DeliveryMedicament |
| Процедура UPD\_Manufacturer | Процедура, осуществляющая изменение в Manufacturer |
| Процедура UPD\_Medicament | Процедура, осуществляющая изменение в Medicament |
| Процедура UPD\_Storage | Процедура, осуществляющая изменение в Storage |
| Процедура logdel\_DeliveryMedicament | Процедура, осуществляющая логическое удаление в DeliveryMedicament |
| Процедура logdel\_Manufacturer | Процедура, осуществляющая логическое удаление в Manufacturer |
| Процедура logdel\_Medicament | Процедура, осуществляющая логическое удаление в Medicament |
| Процедура logdel\_Storage | Процедура, осуществляющая логическое удаление ячеек склада в Storage |
| Процедура Search\_DeliveryMedicamen | Процедура, осуществляющая поиск DeliveryMedicamen |
| Процедура Search\_Medicament | Процедура, осуществляющая поиск Medicament |
| Процедура Select\_CategoriesDisease | Процедура, осуществляющая выборку в CategoriesDisease |
| Процедура Select\_Personal\_Like\_Category | Процедура, осуществляющая выборку в Personal по категориям |
| Процедура Select\_TherapeuticDepartament | Процедура, осуществляющая выборку в TherapeuticDepartament |
| Процедура Add\_CategoriesDisease | Процедура, осуществляющая добавление в CategoriesDisease |
| 1 | 2 |
| Процедура Add\_TherapeuticDepartament | Процедура, осуществляющая добавление в TherapeuticDepartament |
| Процедура UPD\_CategoriesDisease | Процедура, осуществляющая изменение в CategoriesDisease |
| Процедура UPD\_TherapeuticDepartament | Процедура, осуществляющая изменение в TherapeuticDepartament |
| Функция Answer\_UPD\_TherapeuticDepartament\_Amount | Функция, осуществляющая обновление палат в терапии |
| Функция Answer\_UPD\_Unique | Функция, обновляющая уникальные значения |
| Процедура logdel\_CategoriesDisease | Процедура, осуществляющая логическое удаление в CategoriesDisease |
| Процедура logdel\_TherapeuticDepartament | Процедура, осуществляющая логическое удаление в TherapeuticDepartament |

## 1.2. Скрипт объектов баз данных

Код базы данных:

Create DataBase [Life\_of\_Bionic]

go

set quoted\_identifier on

go

set ansi\_nulls on

go

set ansi\_padding on

go

use [Life\_of\_Bionic]

go

Create table [dbo].[LogChange]

(

[ID\_Change] [int] not null identity(1,1),

[Change\_Date] [datetime] not null default getdate(),

[Change\_Type] [varchar] (15) not null,

[Change\_Info] [varchar] (max)

Constraint [PK\_ChangeLog] primary key clustered

([ID\_Change] ASC ) on [PRIMARY]

)

--Производитель

Create table [dbo].[Manufacturer]

(

[id\_Manufacturer] [int] not null identity(1,1),

[Name\_Manufacturer] [varchar] (100) not null,

[Adress] [varchar] (100) null,

[Mail] [varchar] (50) null,

[Manufacturer\_Logical\_Delete] [bit] not null default (0)

Constraint [PK\_Manufacturer] primary key clustered

([id\_Manufacturer] ASC ) on [PRIMARY],

CONSTRAINT uc\_Manufacturer UNIQUE (Name\_Manufacturer),

CONSTRAINT uc\_ManufacturerMail UNIQUE (Mail),

Constraint [CH\_check\_Mail\_Manufacturer] check

([Mail] like '%@%.%' and [Mail] not in ('!','#','$','%','^','&','\*','(',')','\_','-','=','+',

'"','№',';',':','?','{','}','[',']','<','>','/','\'))

)

--Категории медикаментов

Create table [dbo].[CategoryOfMedicament]

(

[id\_CategoryOfMedicament] [int] not null identity(1,1),

[Name\_MedCategory] [varchar] (1) not null,

[CategoryOfMedicament\_Logical\_Delete] [bit] not null default (0)

Constraint [PK\_CategoryOfMedicament] primary key clustered

([id\_CategoryOfMedicament] ASC ) on [PRIMARY],

CONSTRAINT uc\_CategoryOfMedicament UNIQUE (Name\_MedCategory),

Constraint [CH\_check\_NameMedCategory] check

([Name\_MedCategory] like '[A-Z]' or [Name\_MedCategory] like '[a-z]' )

)

--Тип записи на приём

Create table [dbo].[TypeWrite]

(

[id\_TypeWrite] [int] not null identity(1,1),

[Name\_Write] [varchar] (50) not null,

[TypeWrite\_Logical\_Delete] [bit] not null default (0)

Constraint [PK\_TypeWrite] primary key clustered

([id\_TypeWrite] ASC ) on [PRIMARY],

CONSTRAINT uc\_TypeWriteName UNIQUE (Name\_Write),

Constraint [CH\_check\_Name\_Write] check

([Name\_Write] not in ('!','@','#','$','%','^','&','\*','(',')','\_','-','=','+',

'"','№',';',':','?','{','}','[',']','<','>','/','\','1','2','3','4','5','6','7','8','9','0'))

)

--Данные граждан

Create table [dbo].[DataCitizen]

(

[id\_Citizen] [int] not null identity(1,1),

[NameCit] [varchar] (50) not null,

[SurnameCit] [varchar] (50) not null,

[PatronymicCit] [varchar] (50) null,

[Snils] [varchar] (11) not null,

[SeriesPassportCit] [int] not null,

[NumberPassportCit] [int] not null,

[DateBirthCit] [date] not null,

[DataCitizen\_Logical\_Delete] [bit] not null default (0)

Constraint [PK\_DataCitizen] primary key clustered

([id\_Citizen] ASC ) on [PRIMARY],

Constraint [CH\_check\_NameCitizen] check

(([NameCit] like '%[А-Я]%' or [NameCit] like '%[а-я]%') and [NameCit] not in ('!','@','#','$','%','^','&','\*','(',')','\_','-','=','+',

'"','№',';',':','?','{','}','[',']','<','>','/','\','1','2','3','4','5','6','7','8','9','0')),

Constraint [CH\_check\_SurnameCitizen] check

(([PatronymicCit] like '%[А-Я]%' or [PatronymicCit] like '%[а-я]%') and [surnameCit] not in ('!','@','#','$','%','^','&','\*','(',')','\_','-','=','+',

'"','№',';',':','?','{','}','[',']','<','>','/','\','1','2','3','4','5','6','7','8','9','0')),

Constraint [CH\_check\_PatronymicCitizen] check

(([PatronymicCit] like '%[А-Я]%' or [PatronymicCit] like '%[а-я]%') and [PatronymicCit] not in ('!','@','#','$','%','^','&','\*','(',')','\_','-','=','+',

'"','№',';',':','?','{','}','[',']','<','>','/','\','1','2','3','4','5','6','7','8','9','0')),

Constraint [CH\_check\_SnilsCitizen] check

([Snils] like '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),

Constraint [CH\_check\_SeriesPassportCitizen] check

([SeriesPassportCit] like '[0-9][0-9][0-9][0-9]'),

Constraint [CH\_check\_NumberPassportCitizen] check

([NumberPassportCit] like '[0-9][0-9][0-9][0-9][0-9][0-9]'),

CONSTRAINT uc\_PassportCitizen UNIQUE (SeriesPassportCit,NumberPassportCit),

CONSTRAINT uc\_SnilsCit UNIQUE (Snils),

)

--Тип употребления

Create table [dbo].[TypeUse]

(

[id\_TypeUse] [int] not null identity(1,1),

[Name\_Use] [varchar] (50) not null,

[TypeUse\_Logical\_Delete] [bit] not null default (0)

Constraint [PK\_TypeUse] primary key clustered

([id\_TypeUse] ASC ) on [PRIMARY],

Constraint [CH\_check\_Name\_Use] check

(([Name\_Use] like '%[А-Я]%' or [Name\_Use] like '%[а-я]%')

and [Name\_Use] not in ('!','@','#','$','%','^','&','\*','(',')','\_','-','=','+',

'"','№',';',':','?','{','}','[',']','<','>','\','1','2','3','4','5','6','7','8','9','0')),

CONSTRAINT uc\_TypeUse UNIQUE (Name\_Use)

)

--Справочник болезней

Create table [dbo].[DirectoryDisease]

(

[id\_Disease] [int] not null identity(1,1),

[Name\_Disease] [varchar] (50) not null,

[DirectoryDisease\_Logical\_Delete] [bit] not null default (0)

Constraint [PK\_Disease] primary key clustered

([id\_Disease] ASC ) on [PRIMARY],

CONSTRAINT uc\_DirectoryDisease UNIQUE (Name\_Disease),

Constraint [CH\_check\_Name\_Disease] check

(([Name\_Disease] like '%[А-Я]%' or [Name\_Disease] like '%[а-я]%')

and [Name\_Disease] not in ('!','@','#','$','%','^','&','\*','(',')','\_','-','=','+',

'"','№',';',':','?','{','}','[',']','<','>','/','\','1','2','3','4','5','6','7','8','9','0'))

)

--Специальность персонала

Create table [dbo].[SpecialityPersonal]

(

[id\_SpecialityPersonal] [int] not null identity(1,1),

[Name\_SpecialityPersonal] [varchar] (150) not null,

[SpecialityPersonal\_Logical\_Delete] [bit] not null default (0)

Constraint [PK\_SpecialityPersonal] primary key clustered

([id\_SpecialityPersonal] ASC ) on [PRIMARY],

CONSTRAINT uc\_NameSpecialityPersonal UNIQUE (Name\_SpecialityPersonal),

Constraint [CH\_check\_Name\_SpecialityPersonal] check

(([Name\_SpecialityPersonal] like '%[А-Я]%' or [Name\_SpecialityPersonal] like '%[а-я]%')

and [Name\_SpecialityPersonal] not in ('!','@','#','$','%','^','&','\*','(',')','\_','-','=','+',

'"','№',';',':','?','{','}','[',']','<','>','/','\','1','2','3','4','5','6','7','8','9','0'))

)

--Рабочий график персонала

Create table [dbo].[WorkSchedule]

(

[id\_WorkSchedule] [int] not null identity(1,1),

[weekdays] [VARCHAR] (16) not null,

[WorkSchedule\_Logical\_Delete] [bit] not null default (0)

constraint [PK\_Student] primary key clustered

([id\_WorkSchedule] ASC) on [PRIMARY],

CONSTRAINT uc\_WorkSchedule UNIQUE (weekdays),

Constraint [CH\_check\_WorkSchedule] check

([weekdays] like '[0-7]/[0-7]' )

)

--Роли

Create table [dbo].[Role]

(

[id\_Role] [int] not null identity(1,1),

[Write] [bit] not null,

[CardDesign] [bit] not null,

[AcceptanceMedication] [bit] not null,

[ResolutionStatement] [bit] not null,

[AdmissionPatient] [bit] not null,

[id\_SpecialityPersonal] [int] not null, --выступает в качестве наименования роли

[Role\_Logical\_Delete] [bit] not null default (0)

Constraint [PK\_Role] primary key clustered

([id\_Role] ASC ) on [PRIMARY],

constraint [FK\_SpecialityPersonal] foreign key ([id\_SpecialityPersonal])

references [dbo].[SpecialityPersonal] ([id\_SpecialityPersonal])

)

-- Тип выписки

Create table [dbo].[TypeDischarge]

(

[id\_TypeDischarge] [int] not null identity(1,1),

[NameDischarge] [varchar] (50) not null,

[TypeDischarge\_Logical\_Delete] [bit] not null default (0)

Constraint [PK\_TypeDischarge] primary key clustered

([id\_TypeDischarge] ASC ) on [PRIMARY],

Constraint [CH\_check\_NameDischarge] check

(([NameDischarge] like '%[А-Я]%' or [NameDischarge] like '%[а-я]%')

and [NameDischarge] not in ('!','@','#','$','%','^','&','\*','(',')','\_','-','=','+',

'"','№',';',':','?','{','}','[',']','<','>','/','\','1','2','3','4','5','6','7','8','9','0')),

CONSTRAINT uc\_TypeDischarge UNIQUE (NameDischarge)

)

--Правила оказания услуг

Create table [dbo].[Regulation]

(

[id\_Regulation] [int] not null identity(1,1),

[TextRegulation] [text] not null,

[id\_SpecialityPersonal] [int] not null,

[Regulation\_Logical\_Delete] [bit] not null default (0)

Constraint [PK\_Regulation] primary key clustered

([id\_Regulation] ASC ) on [PRIMARY],

constraint [FK\_Speciality] foreign key ([id\_SpecialityPersonal])

references [dbo].[SpecialityPersonal] ([id\_SpecialityPersonal])

)

--Медикаменты

Create table [dbo].[Medicament]

(

[id\_Medicament] [int] not null identity(1,1),

[Name\_Medicament] [varchar] (100) not null,

[id\_Manufacturer] [int] not null,

[id\_CategoryOfMedicament] [int] not null,

[Medicament\_Logical\_Delete] [bit] not null default (0)

Constraint [PK\_Medicament] primary key clustered

([id\_Medicament] ASC ) on [PRIMARY],

constraint [FK\_CategoryOfMedicament] foreign key ([id\_CategoryOfMedicament])

references [dbo].[CategoryOfMedicament] ([id\_CategoryOfMedicament]),

constraint [FK\_Manufacturer] foreign key ([id\_Manufacturer])

references [dbo].[Manufacturer] ([id\_Manufacturer]),

Constraint [CH\_check\_NameMedicament] check

([Name\_Medicament] not in ('!','@','#','$','%','^','&','\*','(',')','\_','-','=','+',

'"',';',':','?','{','}','[',']','<','>','/','\')),

CONSTRAINT uc\_NameMedicament UNIQUE (Name\_Medicament),

)

--Персонал

Create table [dbo].[Personal]

(

[id\_worker] [int] not null identity(1,1),

[NamePers] [varchar] (50) not null,

[SurnamePers] [varchar] (50)not null,

[PatronymicPers] [varchar] (50) null,

[SeriesPassportPers] [int] not null,

[NumberPassportPers] [int] not null,

[User\_Nick] [VARCHAR] (50) not null,

[User\_Pass] [VARBINARY] (max) not null,

[id\_WorkSchedule] [int] null,

[id\_Role] [int] null,

[Personal\_Logical\_Delete] [bit] not null default (0)

Constraint [PK\_Personal] primary key clustered

([id\_worker] ASC ) on [PRIMARY],

constraint [FK\_WorkSchedule] foreign key ([id\_WorkSchedule])

references [dbo].[WorkSchedule] ([id\_WorkSchedule]),

constraint [FK\_RolePersonal] foreign key ([id\_Role])

references [dbo].[Role] ([id\_Role]),

Constraint [CH\_check\_NamePersonal] check

(([NamePers] like '%[А-Я]%' or [NamePers] like '%[а-я]%')

and [NamePers] not in ('!','@','#','$','%','^','&','\*','(',')','\_','-','=','+',

'"','№',';',':','?','{','}','[',']','<','>','/','\','1','2','3','4','5','6','7','8','9','0')),

Constraint [CH\_check\_SurnamePersonal] check

(([SurnamePers] like '%[А-Я]%' or [SurnamePers] like '%[а-я]%')

and [SurnamePers] not in ('!','@','#','$','%','^','&','\*','(',')','\_','-','=','+',

'"','№',';',':','?','{','}','[',']','<','>','/','\','1','2','3','4','5','6','7','8','9','0')),

Constraint [CH\_check\_PatronymicPersonal] check

(([PatronymicPers] like '%[А-Я]%' or [PatronymicPers] like '%[а-я]%')

and [PatronymicPers] not in ('!','@','#','$','%','^','&','\*','(',')','\_','-','=','+',

'"','№',';',':','?','{','}','[',']','<','>','/','\','1','2','3','4','5','6','7','8','9','0')),

Constraint [CH\_check\_SeriesPassportPersonal] check

([SeriesPassportPers] like '[0-9][0-9][0-9][0-9]'),

Constraint [CH\_check\_NumberPassportPersonal] check

([NumberPassportPers] like '[0-9][0-9][0-9][0-9][0-9][0-9]'),

Constraint [CH\_check\_UserNick] check

([User\_Nick] not in ('!','@','#','$','%','^','&','\*','(',')','\_','-','=','+',

'"','№',';',':','?','{','}','[',']','<','>','/','\')),

CONSTRAINT uc\_UserNick UNIQUE (User\_Nick),

CONSTRAINT uc\_PassportPersonal UNIQUE (SeriesPassportPers,NumberPassportPers)

)

--Склад

Create table [dbo].[Storage]

(

[id\_spot] [int] not null identity(1,1),

[Amount] [int] not null,

[occupiedSpace] [int] not null default (0), --занятое место на складе

[Storage\_Logical\_Delete] [bit] not null default (0)

Constraint [PK\_Storage] primary key clustered

([id\_spot] ASC ) on [PRIMARY],

Constraint [CH\_check\_AmountOnSpotStorage] check

(Amount>=occupiedSpace),

)

--Доставка медикаментов

Create table [dbo].[DeliveryMedicament]

(

[id\_DeliveryMedicament] [int] not null identity(1,1),

[Amount] [int] not null,

[DateOfDelivery] [date] not null,

[id\_Medicament] [int] not null,

[id\_worker] [int] not null,

[id\_spot] [int] not null,

[DeliveryMedicament\_Logical\_Delete] [bit] not null default (0)

Constraint [PK\_DeliveryMedicament] primary key clustered

([id\_DeliveryMedicament] ASC ) on [PRIMARY],

constraint [FK\_MedicamentDelivery] foreign key ([id\_Medicament])

references [dbo].[Medicament] ([id\_Medicament]),

constraint [FK\_PersonalDelivery] foreign key ([id\_worker])

references [dbo].[Personal] ([id\_worker]),

constraint [FK\_Storage] foreign key ([id\_spot])

references [dbo].[Storage] ([id\_spot]),

Constraint [CH\_Check\_DateOfDelivery] check (DATEDIFF (DAY,DateOfDelivery,getDate())=0)

)

-- Категория болезни

Create table [dbo].[CategoriesDisease]

(

[id\_CategoriesDisease] [int] not null identity(1,1),

[Name\_CategoriesDisease] [varchar] (50) not null,

[CategoriesDisease\_Logical\_Delete] [bit] not null default (0)

Constraint [PK\_CategoriesDisease] primary key clustered

([id\_CategoriesDisease] ASC ) on [PRIMARY],

Constraint [CH\_check\_Name\_CategoriesDisease] check

(([Name\_CategoriesDisease] like '%[А-Я]%' or [Name\_CategoriesDisease] like '%[а-я]%')

and [Name\_CategoriesDisease] not in ('!','@','#','$','%','^','&','\*','(',')','\_','=','+',

'"','№',';',':','?','{','}','[',']','<','>','\')),

CONSTRAINT uc\_CategoriesDisease UNIQUE (Name\_CategoriesDisease)

)

--Форма записи

Create table [dbo].[FormWrite]

(

[id\_FormWrite] [INT] not null identity(1,1),

[id\_worker] [int] not null,

[id\_TypeWrite] [int] not null,

[mail] [varchar] (50) null,

[PhoneNumber] [varchar] (16) null,

[Sites] [varchar] (100) null,

[Adress] [varchar] (100) null,

[FormWrite\_Logical\_Delete] [bit] not null default (0)

Constraint [PK\_FormWrite] primary key clustered

([id\_FormWrite] ASC ) on [PRIMARY],

constraint [FK\_PersonalWrite] foreign key ([id\_worker])

references [dbo].[Personal] ([id\_worker]),

constraint [FK\_TypeWrite] foreign key ([id\_TypeWrite])

references [dbo].[TypeWrite] ([id\_TypeWrite]),

Constraint [CH\_check\_MailFormWrite] check

([Mail] like '%@%.%' and [Mail] not in ('!','#','$','%','^','&','\*','(',')','\_','-','=','+',

'"','№',';',':','?','{','}','[',']','<','>','/','\')),

Constraint [CH\_check\_PhoneNumber] check

([PhoneNumber] like '+[0-9]([0-9][0-9][0-9])[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9]') -- +X(XXX)XXX-XX-XX

)

--Запись на приём

Create table [dbo].[WriteAppointment]

(

[id\_WriteAppointment] [int] not null identity(1,1),

[times] [datetime] not null,

[visit] [bit] not null default (0),

[id\_Citizen] [int] not null,

[id\_FormWrite] [int] not null,

[MedicalDepartament] [int] not null,

[Categories\_Disease] [varchar] (50) null,

[SentToTreatment] [bit] not null default (0),

[WriteAppointment\_Logical\_Delete] [bit] not null default (0)

Constraint [PK\_WriteAppointment] primary key clustered

([id\_WriteAppointment] ASC ) on [PRIMARY],

constraint [FK\_Citizen] foreign key ([id\_Citizen])

references [dbo].[DataCitizen] ([id\_Citizen]),

constraint [FK\_FormWrite] foreign key ([id\_FormWrite])

references [dbo].[FormWrite] ([id\_FormWrite])

)

--Терапевтическое отделение

Create table [dbo].[TherapeuticDepartament]

(

[id\_Room] [int] not null identity(1,1),

[amountRooms] [int] not null,

[BusyRoom] [int] not null, --занятно мест

[id\_worker] [int] not null,

[id\_CategoriesDisease] [int] not null,

[TherapeuticDepartament\_Logical\_Delete] [bit] not null default (0)

Constraint [PK\_TherapeuticDepartament] primary key clustered

([id\_Room] ASC ) on [PRIMARY],

constraint [FK\_PersonalTherapeutic] foreign key ([id\_worker])

references [dbo].[Personal] ([id\_worker]),

constraint [FK\_CategoriesDisease] foreign key ([id\_CategoriesDisease])

references [dbo].[CategoriesDisease] ([id\_CategoriesDisease]),

Constraint [CH\_check\_RoomsInTherapy] check

(amountRooms>=BusyRoom)

)

--Диагноз

Create table [dbo].[Diagnosis]

(

[id\_Diagnoz] [int] not null identity(1,1),

[TimeDisease] [int] not null,

[Amount] [int] not null,

[ID\_Room] [int] not null,

[id\_TypeUse] [int] not null,

[id\_Medicament] [int] not null,

[id\_Disease] [int] not null,

[Diagnosis\_Logical\_Delete] [bit] not null default (0)

Constraint [PK\_Diagnoz] primary key clustered

([id\_Diagnoz] ASC ) on [PRIMARY],

constraint [FK\_Room] foreign key ([id\_Room])

references [dbo].[TherapeuticDepartament] ([id\_Room]),

constraint [FK\_TypeUse] foreign key ([id\_TypeUse])

references [dbo].[TypeUse] ([id\_TypeUse]),

constraint [FK\_MedicamentDiagnosis] foreign key ([id\_Medicament])

references [dbo].[Medicament] ([id\_Medicament]),

constraint [FK\_DiseaseDiagnosis] foreign key ([id\_Disease])

references [dbo].[DirectoryDisease] ([id\_Disease])

)

--Карта лечения

Create table [dbo].[CardTreatments]

(

[id\_card] [int] not null identity(1,1),

[id\_WriteAppointment] [int] not null,

[id\_Diagnoz] [int] not null,

[Cured] [bit] not null default (0),

[CardTreatments\_Logical\_Delete] [bit] not null default (0)

Constraint [PK\_Card] primary key clustered

([id\_card] ASC ) on [PRIMARY],

constraint [FK\_Disease] foreign key ([id\_WriteAppointment])

references [dbo].[WriteAppointment] ([id\_WriteAppointment]),

constraint [FK\_Diaznoz] foreign key ([id\_Diagnoz])

references [dbo].[Diagnosis] ([id\_Diagnoz])

)

--Выписка

Create table [dbo].[Discharge]

(

[id\_Discharge] [int] not null identity(1,1),

[DateDischarge] [date] not null,

[id\_WriteAppointment] [int] not null,

[id\_TypeDischarge] [int] not null,

[Discharge\_Logical\_Delete] [bit] not null default (0)

Constraint [PK\_Discharge] primary key clustered

([id\_Discharge] ASC ) on [PRIMARY],

constraint [FK\_Card] foreign key ([id\_WriteAppointment])

references [dbo].[WriteAppointment] ([id\_WriteAppointment]),

constraint [FK\_TypeDischarge] foreign key ([id\_TypeDischarge])

references [dbo].[TypeDischarge] ([id\_TypeDischarge]),

)

Код триггеров:

Триггер DeleteOnPalateNewCitizen

create TRIGGER DeleteOnPalateNewCitizen ON [dbo].[CardTreatments]

FOR delete

as

declare @Diagnoz [int] = (select id\_Diagnoz from deleted)

declare @NumberRoom [int] = (SELECT id\_Room FROM dbo.Diagnosis

where dbo.Diagnosis.id\_Diagnoz = @Diagnoz)

declare @Palata [int] = (SELECT dbo.TherapeuticDepartament.id\_Room FROM dbo.TherapeuticDepartament

where dbo.TherapeuticDepartament.id\_Room = @NumberRoom)

declare @AmountInRoom [int] = (SELECT dbo.TherapeuticDepartament.amountRooms FROM dbo.TherapeuticDepartament

where dbo.TherapeuticDepartament.id\_Room = @NumberRoom)

Declare @YzeZapolnInRoom[int] = (SELECT dbo.TherapeuticDepartament.BusyRoom FROM dbo.TherapeuticDepartament

where dbo.TherapeuticDepartament.id\_Room = @NumberRoom)

if @YzeZapolnInRoom < @AmountInRoom

Begin

update dbo.TherapeuticDepartament set

AmountRooms = AmountRooms,

BusyRoom = @YzeZapolnInRoom - 1

where id\_Room = @NumberRoom

end

else

if @YzeZapolnInRoom = @AmountInRoom

RAISERROR('Палата Свободна',16,0) WITH NOWAIT;

Триггер InsertOnPalateNewCitizen:

create

TRIGGER InsertOnPalateNewCitizen ON [dbo].[CardTreatments]

FOR insert, update

as

declare @Diagnoz [int] = (select id\_Diagnoz from inserted)

declare @NumberRoom [int] = (SELECT id\_Room FROM dbo.Diagnosis

where dbo.Diagnosis.id\_Diagnoz = @Diagnoz)

declare @Palata [int] = (SELECT dbo.TherapeuticDepartament.id\_Room FROM dbo.TherapeuticDepartament

where dbo.TherapeuticDepartament.id\_Room = @NumberRoom)

declare @AmountInRoom [int] = (SELECT dbo.TherapeuticDepartament.amountRooms FROM dbo.TherapeuticDepartament

where dbo.TherapeuticDepartament.id\_Room = @NumberRoom)

Declare @YzeZapolnInRoom[int] = (SELECT dbo.TherapeuticDepartament.BusyRoom FROM dbo.TherapeuticDepartament

where dbo.TherapeuticDepartament.id\_Room = @NumberRoom)

if @YzeZapolnInRoom < @AmountInRoom

Begin

update dbo.TherapeuticDepartament set

AmountRooms = AmountRooms,

BusyRoom = @YzeZapolnInRoom + 1

where id\_Room = @NumberRoom

end

else RAISERROR('Палата переполнен

а',16,0) WITH NOWAIT;

Триггер NaznashenieMedicamentov:

create TRIGGER NaznashenieMedicamentov ON [dbo].[CardTreatments]

FOR insert, update

as

declare @Diagnoz [int] = (select id\_Diagnoz from inserted)

declare @MedInDiagnoz [int] = (select id\_Medicament from dbo.Diagnosis

where dbo.Diagnosis.id\_Diagnoz = @Diagnoz)

declare @Medicament [int] = (select id\_medicament from dbo.Medicament

where dbo.Medicament.id\_Medicament = @MedInDiagnoz)

declare @PostavMed [int] = (select id\_Spot from dbo.DeliveryMedicament

where dbo.DeliveryMedicament.id\_Medicament = @Medicament)

declare @AmountOnStorage [int] = (select occupiedSpace from dbo.Storage

where dbo.Storage.id\_spot = @PostavMed)

declare @AmountNaznach [int] = (SELECT Amount FROM dbo.Diagnosis

where dbo.Diagnosis.id\_Diagnoz = @Diagnoz)

if @AmountNaznach <= @AmountOnStorage

Begin

update dbo.Storage set

Amount = Amount,

occupiedSpace = @AmountOnStorage - @AmountNaznach

where id\_spot = @MedInDiagnoz

end

else RAISERROR('Недостаток лекарств',16,0) WITH NOWAIT;

Триггер Tovar\_Prihod:

Create TRIGGER Tovar\_Prihod ON dbo.DeliveryMedicament

FOR INSERT, UPDATE

AS

Declare @CountPrih[int] = (Select Amount from inserted)

Declare @WhatSklad[int] = (Select id\_spot from inserted)

Declare @MaxOnSklad[int] = (Select dbo.Storage.Amount from dbo.Storage where dbo.Storage.id\_spot = @WhatSklad)

Declare @YzeZapoln[int] = (Select dbo.Storage.occupiedSpace from dbo.Storage where dbo.Storage.id\_spot = @WhatSklad)

Declare @DobavlyamCount[int] = (Select dbo.Storage.occupiedSpace + @CountPrih from dbo.Storage where dbo.Storage.id\_spot = @WhatSklad)

if @DobavlyamCount < @MaxOnSklad

begin

update dbo.Storage set

Amount = Amount,

occupiedSpace = @YzeZapoln + @CountPrih

where id\_spot = @WhatSklad

end

else RAISERROR('Склад переполнен',16,0) WITH NOWAIT;

Триггер dbo.changes\_CardTreatments\_trigger

--- ТРИГГЕР ОТСЛЕЖИВАЮЩИЙ ИЗМЕНЕНИЯ КАРТ ЛЕЧЕНИЯ

create trigger dbo.changes\_CardTreatments\_trigger on dbo.CardTreatments

FOR INSERT, UPDATE, DELETE

as

set NOCOUNT ON;

declare @change\_type as varchar(15)

declare @count as int

-- определеяем тип произошедших изменений INSERT,UPDATE, или DELETE

set @change\_type = 'inserted'

select @count = COUNT(\*) FROM DELETED

if @count > 0

begin

set @change\_type = 'deleted'

select @count = COUNT(\*) from INSERTED

if @Count > 0

set @change\_type = 'updated'

end

-- обработка удаления

if @change\_type = 'deleted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Удалено из ',

'Карта №'+ Convert(varchar (4),id\_card) from deleted

end

else

begin

if @change\_type = 'inserted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Добавлено в ',

'Карта №'+ Convert(varchar (4),id\_card) from inserted

end

-- обработка обновления

else

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Изменено в ',

'Карта №'+ Convert(varchar (4),id\_card) from inserted

end

end

Триггер dbo.changes\_CategoriesDisease\_trigger

--- ТРИГГЕР ОТСЛЕЖИВАЮЩИЙ ИЗМЕНЕНИЯ КАТЕГОРИИ БОЛЕЗНЕЙ

create trigger dbo.changes\_CategoriesDisease\_trigger on dbo.CategoriesDisease

FOR INSERT, UPDATE, DELETE

as

set NOCOUNT ON;

declare @change\_type as varchar(15)

declare @count as int

-- определеяем тип произошедших изменений INSERT,UPDATE, или DELETE

set @change\_type = 'inserted'

select @count = COUNT(\*) FROM DELETED

if @count > 0

begin

set @change\_type = 'deleted'

select @count = COUNT(\*) from INSERTED

if @Count > 0

set @change\_type = 'updated'

end

-- обработка удаления

if @change\_type = 'deleted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Удалено из ',

'Категория болезней: '+Name\_CategoriesDisease from deleted

end

else

begin

if @change\_type = 'inserted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Добавлено в ',

'Категория болезней: '+Name\_CategoriesDisease from inserted

end

-- обработка обновления

else

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Изменено в ',

'Категория болезней: '+Name\_CategoriesDisease from inserted

end

end

Триггер dbo.changes\_CategoryOfMedicament\_trigger

--- ТРИГГЕР ОТСЛЕЖИВАЮЩИЙ ИЗМЕНЕНИЯ КАТЕГОРИИ МЕДИКАМЕНТОВ

create trigger dbo.changes\_CategoryOfMedicament\_trigger on dbo.CategoryOfMedicament

FOR INSERT, UPDATE, DELETE

as

set NOCOUNT ON;

declare @change\_type as varchar(15)

declare @count as int

-- определеяем тип произошедших изменений INSERT,UPDATE, или DELETE

set @change\_type = 'inserted'

select @count = COUNT(\*) FROM DELETED

if @count > 0

begin

set @change\_type = 'deleted'

select @count = COUNT(\*) from INSERTED

if @Count > 0

set @change\_type = 'updated'

end

-- обработка удаления

if @change\_type = 'deleted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Удалено из ',

'Категория медикаментов: '+Name\_MedCategory from deleted

end

else

begin

if @change\_type = 'inserted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Добавлено в ',

'Категория медикаментов: '+Name\_MedCategory from inserted

end

-- обработка обновления

else

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Изменено в ',

'Категория медикаментов: '+Name\_MedCategory from inserted

end

end

Триггер dbo.changes\_DataCitizen\_trigger

--- ТРИГГЕР ОТСЛЕЖИВАЮЩИЙ ИЗМЕНЕНИЯ ГРАЖДАННИА

create trigger dbo.changes\_DataCitizen\_trigger on dbo.DataCitizen

FOR INSERT, UPDATE, DELETE

as

set NOCOUNT ON;

declare @change\_type as varchar(15)

declare @count as int

-- определеяем тип произошедших изменений INSERT,UPDATE, или DELETE

set @change\_type = 'inserted'

select @count = COUNT(\*) FROM DELETED

if @count > 0

begin

set @change\_type = 'deleted'

select @count = COUNT(\*) from INSERTED

if @Count > 0

set @change\_type = 'updated'

end

-- обработка удаления

if @change\_type = 'deleted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Удалено из ',

'Гражданин: '+NameCit+' '+SurnameCit+' '+PatronymicCit from deleted

end

else

begin

if @change\_type = 'inserted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Добавлено в ',

'Гражданин: '+NameCit+' '+SurnameCit+' '+PatronymicCit from inserted

end

-- обработка обновления

else

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Изменено в ',

'Гражданин: '+NameCit+' '+SurnameCit+' '+PatronymicCit from inserted

end

end

Триггер dbo.changes\_DeliveryMedicament\_trigger

--- ТРИГГЕР ОТСЛЕЖИВАЮЩИЙ ИЗМЕНЕНИЯ ПОСТАВКИ МЕДИКАМЕНТОВ

create trigger dbo.changes\_DeliveryMedicament\_trigger on dbo.DeliveryMedicament

FOR INSERT, UPDATE, DELETE

as

set NOCOUNT ON;

declare @change\_type as varchar(15)

declare @count as int

-- определеяем тип произошедших изменений INSERT,UPDATE, или DELETE

set @change\_type = 'inserted'

select @count = COUNT(\*) FROM DELETED

if @count > 0

begin

set @change\_type = 'deleted'

select @count = COUNT(\*) from INSERTED

if @Count > 0

set @change\_type = 'updated'

end

-- обработка удаления

if @change\_type = 'deleted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Удалено из ',

'Доставка №'+ Convert(varchar (4),id\_DeliveryMedicament) from deleted

end

else

begin

if @change\_type = 'inserted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Добавлено в ',

'Доставка №'+ Convert(varchar (4),id\_DeliveryMedicament) from inserted

end

-- обработка обновления

else

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Изменено в ',

'Доставка №'+ Convert(varchar (4),id\_DeliveryMedicament) from inserted

end

end

Триггер dbo.changes\_Diagnosis\_trigger

--- ТРИГГЕР ОТСЛЕЖИВАЮЩИЙ ИЗМЕНЕНИЯ ДИАГНОЗА

create trigger dbo.changes\_Diagnosis\_trigger on dbo.Diagnosis

FOR INSERT, UPDATE, DELETE

as

set NOCOUNT ON;

declare @change\_type as varchar(15)

declare @count as int

-- определеяем тип произошедших изменений INSERT,UPDATE, или DELETE

set @change\_type = 'inserted'

select @count = COUNT(\*) FROM DELETED

if @count > 0

begin

set @change\_type = 'deleted'

select @count = COUNT(\*) from INSERTED

if @Count > 0

set @change\_type = 'updated'

end

-- обработка удаления

if @change\_type = 'deleted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Удалено из ',

'Диагноз №'+ Convert(varchar (4),id\_Diagnoz) from deleted

end

else

begin

if @change\_type = 'inserted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Добавлено в ',

'Диагноз №'+ Convert(varchar (4),id\_Diagnoz) from inserted

end

-- обработка обновления

else

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Изменено в ',

'Диагноз №'+ Convert(varchar (4),id\_Diagnoz) from inserted

end

end

Триггер dbo.changes\_DirectoryDisease\_trigger

--- ТРИГГЕР ОТСЛЕЖИВАЮЩИЙ ИЗМЕНЕНИЯ СПРАВОЧНИКА БОЛЕЗНЕЙ

create trigger dbo.changes\_DirectoryDisease\_trigger on dbo.DirectoryDisease

FOR INSERT, UPDATE, DELETE

as

set NOCOUNT ON;

declare @change\_type as varchar(15)

declare @count as int

-- определеяем тип произошедших изменений INSERT,UPDATE, или DELETE

set @change\_type = 'inserted'

select @count = COUNT(\*) FROM DELETED

if @count > 0

begin

set @change\_type = 'deleted'

select @count = COUNT(\*) from INSERTED

if @Count > 0

set @change\_type = 'updated'

end

-- обработка удаления

if @change\_type = 'deleted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Удалено из ',

'Название болезни: '+ Name\_Disease from deleted

end

else

begin

if @change\_type = 'inserted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Добавлено в ',

'Название болезни: '+ Name\_Disease from inserted

end

-- обработка обновления

else

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Изменено в ',

'Название болезни: '+ Name\_Disease from inserted

end

end

Триггер dbo.changes\_Discharge\_trigger

--- ТРИГГЕР ОТСЛЕЖИВАЮЩИЙ ИЗМЕНЕНИЯ ВЫПИСКИ

create trigger dbo.changes\_Discharge\_trigger on dbo.Discharge

FOR INSERT, UPDATE, DELETE

as

set NOCOUNT ON;

declare @change\_type as varchar(15)

declare @count as int

-- определеяем тип произошедших изменений INSERT,UPDATE, или DELETE

set @change\_type = 'inserted'

select @count = COUNT(\*) FROM DELETED

if @count > 0

begin

set @change\_type = 'deleted'

select @count = COUNT(\*) from INSERTED

if @Count > 0

set @change\_type = 'updated'

end

-- обработка удаления

if @change\_type = 'deleted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Удалено из ',

'Выписка №'+ Convert(varchar (4),[id\_Discharge]) from deleted

end

else

begin

if @change\_type = 'inserted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Добавлено в ',

'Выписка №'+ Convert(varchar (4),[id\_Discharge]) from inserted

end

-- обработка обновления

else

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Изменено в ',

'Выписка №'+ Convert(varchar (4),[id\_Discharge]) from inserted

end

end

Триггер dbo.changes\_FormWrite\_trigger

--- ТРИГГЕР ОТСЛЕЖИВАЮЩИЙ ИЗМЕНЕНИЯ ФОРМЫ ЗАПИСИ

create trigger dbo.changes\_FormWrite\_trigger on dbo.FormWrite

FOR INSERT, UPDATE, DELETE

as

set NOCOUNT ON;

declare @change\_type as varchar(15)

declare @count as int

-- определеяем тип произошедших изменений INSERT,UPDATE, или DELETE

set @change\_type = 'inserted'

select @count = COUNT(\*) FROM DELETED

if @count > 0

begin

set @change\_type = 'deleted'

select @count = COUNT(\*) from INSERTED

if @Count > 0

set @change\_type = 'updated'

end

-- обработка удаления

if @change\_type = 'deleted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Удалено из ',

'Форма на запись №'+ Convert(varchar (4),id\_FormWrite) from deleted

end

else

begin

if @change\_type = 'inserted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Добавлено в ',

'Форма на запись №'+ Convert(varchar (4),id\_FormWrite) from inserted

end

-- обработка обновления

else

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Изменено в ',

'зФорма на запись №'+ Convert(varchar (4),id\_FormWrite) from inserted

end

end

Триггер dbo.changes\_manufacturer\_trigger

--- ТРИГГЕР ОТСЛЕЖИВАЮЩИЙ ИЗМЕНЕНИЯ ПРОИЗВОДИТЕЛЯ

create trigger dbo.changes\_manufacturer\_trigger on dbo.Manufacturer

FOR INSERT, UPDATE, DELETE

as

set NOCOUNT ON;

declare @change\_type as varchar(15)

declare @count as int

-- определеяем тип произошедших изменений INSERT,UPDATE, или DELETE

set @change\_type = 'inserted'

select @count = COUNT(\*) FROM DELETED

if @count > 0

begin

set @change\_type = 'deleted'

select @count = COUNT(\*) from INSERTED

if @Count > 0

set @change\_type = 'updated'

end

-- обработка удаления

if @change\_type = 'deleted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Удалено из ',

'Производитель: '+Name\_Manufacturer from deleted

end

else

begin

if @change\_type = 'inserted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Добавлено в ',

'Производитель: '+Name\_Manufacturer from inserted

end

-- обработка обновления

else

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Изменено в ',

'Производитель: '+Name\_Manufacturer from inserted

end

end

Триггер dbo.changes\_Medicament\_trigger

--- ТРИГГЕР ОТСЛЕЖИВАЮЩИЙ ИЗМЕНЕНИЯ МЕДИКАМЕНТОВ

create trigger dbo.changes\_Medicament\_trigger on dbo.Medicament

FOR INSERT, UPDATE, DELETE

as

set NOCOUNT ON;

declare @change\_type as varchar(15)

declare @count as int

-- определеяем тип произошедших изменений INSERT,UPDATE, или DELETE

set @change\_type = 'inserted'

select @count = COUNT(\*) FROM DELETED

if @count > 0

begin

set @change\_type = 'deleted'

select @count = COUNT(\*) from INSERTED

if @Count > 0

set @change\_type = 'updated'

end

-- обработка удаления

if @change\_type = 'deleted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Удалено из ',

'Медикамент: '+ Name\_Medicament from deleted

end

else

begin

if @change\_type = 'inserted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Добавлено в ',

'Медикамент: '+ Name\_Medicament from inserted

end

-- обработка обновления

else

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Изменено в ',

'Медикамент: '+ Name\_Medicament from inserted

end

end

Триггер dbo.changes\_Personal\_trigger

--- ТРИГГЕР ОТСЛЕЖИВАЮЩИЙ ИЗМЕНЕНИЯ ПЕРСОНАЛА

create trigger dbo.changes\_Personal\_trigger on dbo.Personal

FOR INSERT, UPDATE, DELETE

as

set NOCOUNT ON;

declare @change\_type as varchar(15)

declare @count as int

-- определеяем тип произошедших изменений INSERT,UPDATE, или DELETE

set @change\_type = 'inserted'

select @count = COUNT(\*) FROM DELETED

if @count > 0

begin

set @change\_type = 'deleted'

select @count = COUNT(\*) from INSERTED

if @Count > 0

set @change\_type = 'updated'

end

-- обработка удаления

if @change\_type = 'deleted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Удалено из ',

'Сотрудник: '+ SurnamePers+' '+NamePers+' '+PatronymicPers from deleted

end

else

begin

if @change\_type = 'inserted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Добавлено в ',

'Сотрудник: '+ SurnamePers+' '+NamePers+' '+PatronymicPers from inserted

end

-- обработка обновления

else

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Изменено в ',

'Сотрудник: '+ SurnamePers+' '+NamePers+' '+PatronymicPers from inserted

end

end

Триггер dbo.changes\_Regulation\_trigger

--- ТРИГГЕР ОТСЛЕЖИВАЮЩИЙ ИЗМЕНЕНИЯ ПРАВИЛ

create trigger dbo.changes\_Regulation\_trigger on dbo.Regulation

FOR INSERT, UPDATE, DELETE

as

set NOCOUNT ON;

declare @change\_type as varchar(15)

declare @count as int

-- определеяем тип произошедших изменений INSERT,UPDATE, или DELETE

set @change\_type = 'inserted'

select @count = COUNT(\*) FROM DELETED

if @count > 0

begin

set @change\_type = 'deleted'

select @count = COUNT(\*) from INSERTED

if @Count > 0

set @change\_type = 'updated'

end

-- обработка удаления

if @change\_type = 'deleted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Удалено из ',

'Правило №'+ CONVERT(varchar (4),id\_Regulation) from deleted

end

else

begin

if @change\_type = 'inserted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Добавлено в ',

'Правило №'+ CONVERT(varchar (4),id\_Regulation) from inserted

end

-- обработка обновления

else

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Изменено в ',

'Правило №'+ CONVERT(varchar (4),id\_Regulation) from inserted

end

end

Триггер dbo.changes\_Role\_trigger

--- ТРИГГЕР ОТСЛЕЖИВАЮЩИЙ ИЗМЕНЕНИЯ РОЛЕЙ

create trigger dbo.changes\_Role\_trigger on dbo.[Role]

FOR INSERT, UPDATE, DELETE

as

set NOCOUNT ON;

declare @change\_type as varchar(15)

declare @count as int

-- определеяем тип произошедших изменений INSERT,UPDATE, или DELETE

set @change\_type = 'inserted'

select @count = COUNT(\*) FROM DELETED

if @count > 0

begin

set @change\_type = 'deleted'

select @count = COUNT(\*) from INSERTED

if @Count > 0

set @change\_type = 'updated'

end

-- обработка удаления

if @change\_type = 'deleted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Удалено из ',

'Роли №'+ CONVERT(varchar (4),id\_Role) from deleted

end

else

begin

if @change\_type = 'inserted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Добавлено в ',

'Роли №'+ CONVERT(varchar (4),id\_Role) from inserted

end

-- обработка обновления

else

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Изменено в ',

'Роли №'+ CONVERT(varchar (4),id\_Role) from inserted

end

end

Триггер dbo.changes\_SpecialityPersonal\_trigger

--- ТРИГГЕР ОТСЛЕЖИВАЮЩИЙ ИЗМЕНЕНИЯ СПЕЦИАЛЬНОСТЕЙ ПЕРСОНАЛА

create trigger dbo.changes\_SpecialityPersonal\_trigger on dbo.SpecialityPersonal

FOR INSERT, UPDATE, DELETE

as

set NOCOUNT ON;

declare @change\_type as varchar(15)

declare @count as int

-- определеяем тип произошедших изменений INSERT,UPDATE, или DELETE

set @change\_type = 'inserted'

select @count = COUNT(\*) FROM DELETED

if @count > 0

begin

set @change\_type = 'deleted'

select @count = COUNT(\*) from INSERTED

if @Count > 0

set @change\_type = 'updated'

end

-- обработка удаления

if @change\_type = 'deleted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Удалено из ',

'Название специальности: '+ Name\_SpecialityPersonal from deleted

end

else

begin

if @change\_type = 'inserted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Добавлено в ',

'Название специальности: '+ Name\_SpecialityPersonal from inserted

end

-- обработка обновления

else

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Изменено в ',

'Название специальности: '+ Name\_SpecialityPersonal from inserted

end

end

Триггер dbo.changes\_Storage\_trigger

--- ТРИГГЕР ОТСЛЕЖИВАЮЩИЙ ИЗМЕНЕНИЯ СКЛАДА

create trigger dbo.changes\_Storage\_trigger on dbo.Storage

FOR INSERT, UPDATE, DELETE

as

set NOCOUNT ON;

declare @change\_type as varchar(15)

declare @count as int

-- определеяем тип произошедших изменений INSERT,UPDATE, или DELETE

set @change\_type = 'inserted'

select @count = COUNT(\*) FROM DELETED

if @count > 0

begin

set @change\_type = 'deleted'

select @count = COUNT(\*) from INSERTED

if @Count > 0

set @change\_type = 'updated'

end

-- обработка удаления

if @change\_type = 'deleted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Удалено из ',

'Ячейка склаад №'+ CONVERT(varchar (4),id\_spot) from deleted

end

else

begin

if @change\_type = 'inserted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Добавлено в ',

'Ячейка склаад №'+ CONVERT(varchar (4),id\_spot) from inserted

end

-- обработка обновления

else

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Изменено в ',

'Ячейка склаад №'+ CONVERT(varchar (4),id\_spot) from inserted

end

end

Триггер dbo.changes\_TherapeuticDepartament\_trigger

--- ТРИГГЕР ОТСЛЕЖИВАЮЩИЙ ИЗМЕНЕНИЯ ТЕРАПЕВТИЧЕСКОГО ОТДЕЛЕНИЯ

create trigger dbo.changes\_TherapeuticDepartament\_trigger on dbo.TherapeuticDepartament

FOR INSERT, UPDATE, DELETE

as

set NOCOUNT ON;

declare @change\_type as varchar(15)

declare @count as int

-- определеяем тип произошедших изменений INSERT,UPDATE, или DELETE

set @change\_type = 'inserted'

select @count = COUNT(\*) FROM DELETED

if @count > 0

begin

set @change\_type = 'deleted'

select @count = COUNT(\*) from INSERTED

if @Count > 0

set @change\_type = 'updated'

end

-- обработка удаления

if @change\_type = 'deleted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Удалено из ',

'Палата №'+ Convert(varchar (4),id\_Room) from deleted

end

else

begin

if @change\_type = 'inserted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Добавлено в ',

'Палата №'+ Convert(varchar (4),id\_Room) from inserted

end

-- обработка обновления

else

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Изменено в ',

'Палата №'+ Convert(varchar (4),id\_Room) from inserted

end

end

Триггер dbo.changes\_TypeDischarge\_trigger

--- ТРИГГЕР ОТСЛЕЖИВАЮЩИЙ ИЗМЕНЕНИЯ ТИПА ВЫПИСКИ

create trigger dbo.changes\_TypeDischarge\_trigger on dbo.TypeDischarge

FOR INSERT, UPDATE, DELETE

as

set NOCOUNT ON;

declare @change\_type as varchar(15)

declare @count as int

-- определеяем тип произошедших изменений INSERT,UPDATE, или DELETE

set @change\_type = 'inserted'

select @count = COUNT(\*) FROM DELETED

if @count > 0

begin

set @change\_type = 'deleted'

select @count = COUNT(\*) from INSERTED

if @Count > 0

set @change\_type = 'updated'

end

-- обработка удаления

if @change\_type = 'deleted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Удалено из ',

'Тип выписки: '+ NameDischarge from deleted

end

else

begin

if @change\_type = 'inserted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Добавлено в ',

'Тип выписки: '+ NameDischarge from inserted

end

-- обработка обновления

else

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Изменено в ',

'Тип выписки: '+ NameDischarge from inserted

end

end

Триггер dbo.changes\_TypeUse\_trigger

--- ТРИГГЕР ОТСЛЕЖИВАЮЩИЙ ИЗМЕНЕНИЯ ТИПА УПОТРЕБЛЕНИЯ

create trigger dbo.changes\_TypeUse\_trigger on dbo.TypeUse

FOR INSERT, UPDATE, DELETE

as

set NOCOUNT ON;

declare @change\_type as varchar(15)

declare @count as int

-- определеяем тип произошедших изменений INSERT,UPDATE, или DELETE

set @change\_type = 'inserted'

select @count = COUNT(\*) FROM DELETED

if @count > 0

begin

set @change\_type = 'deleted'

select @count = COUNT(\*) from INSERTED

if @Count > 0

set @change\_type = 'updated'

end

-- обработка удаления

if @change\_type = 'deleted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Удалено из ',

'Типе исопльзования: '+ Name\_Use from deleted

end

else

begin

if @change\_type = 'inserted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Добавлено в ',

'Типе исопльзования: '+ Name\_Use from inserted

end

-- обработка обновления

else

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Изменено в ',

'Типе исопльзования: '+ Name\_Use from inserted

end

end

Триггер dbo.changes\_TypeWrite\_trigger

--- ТРИГГЕР ОТСЛЕЖИВАЮЩИЙ ИЗМЕНЕНИЯ ТИПА ЗАПИСИ

create trigger dbo.changes\_TypeWrite\_trigger on dbo.TypeWrite

FOR INSERT, UPDATE, DELETE

as

set NOCOUNT ON;

declare @change\_type as varchar(15)

declare @count as int

-- определеяем тип произошедших изменений INSERT,UPDATE, или DELETE

set @change\_type = 'inserted'

select @count = COUNT(\*) FROM DELETED

if @count > 0

begin

set @change\_type = 'deleted'

select @count = COUNT(\*) from INSERTED

if @Count > 0

set @change\_type = 'updated'

end

-- обработка удаления

if @change\_type = 'deleted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Удалено из ',

'Типе записи: '+Name\_Write from deleted

end

else

begin

if @change\_type = 'inserted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Добавлено в ',

'Типе записи: '+Name\_Write from inserted

end

-- обработка обновления

else

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Изменено в ',

'Типе записи: '+Name\_Write from inserted

end

end

Триггер dbo.changes\_WorkSchedulel\_trigger

--- ТРИГГЕР ОТСЛЕЖИВАЮЩИЙ ИЗМЕНЕНИЯ ГРАФИКА ПЕРСОНАЛА

create trigger dbo.changes\_WorkSchedulel\_trigger on dbo.WorkSchedule

FOR INSERT, UPDATE, DELETE

as

set NOCOUNT ON;

declare @change\_type as varchar(15)

declare @count as int

-- определеяем тип произошедших изменений INSERT,UPDATE, или DELETE

set @change\_type = 'inserted'

select @count = COUNT(\*) FROM DELETED

if @count > 0

begin

set @change\_type = 'deleted'

select @count = COUNT(\*) from INSERTED

if @Count > 0

set @change\_type = 'updated'

end

-- обработка удаления

if @change\_type = 'deleted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Удалено из ',

'График: '+ weekdays from deleted

end

else

begin

if @change\_type = 'inserted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Добавлено в ',

'График: '+ weekdays from inserted

end

-- обработка обновления

else

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Изменено в ',

'График: '+ weekdays from inserted

end

end

Триггер dbo.changes\_WriteAppointment\_trigger

--- ТРИГГЕР ОТСЛЕЖИВАЮЩИЙ ИЗМЕНЕНИЯ ЗАПИСИ НА ПРИЁМ

create trigger dbo.changes\_WriteAppointment\_trigger on dbo.WriteAppointment

FOR INSERT, UPDATE, DELETE

as

set NOCOUNT ON;

declare @change\_type as varchar(15)

declare @count as int

-- определеяем тип произошедших изменений INSERT,UPDATE, или DELETE

set @change\_type = 'inserted'

select @count = COUNT(\*) FROM DELETED

if @count > 0

begin

set @change\_type = 'deleted'

select @count = COUNT(\*) from INSERTED

if @Count > 0

set @change\_type = 'updated'

end

-- обработка удаления

if @change\_type = 'deleted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Удалено из ',

'Запись №'+ Convert(varchar (4),id\_WriteAppointment) from deleted

end

else

begin

if @change\_type = 'inserted'

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Добавлено в ',

'Запись №'+ Convert(varchar (4),id\_WriteAppointment) from inserted

end

-- обработка обновления

else

begin

insert into [dbo].[LogChange] (CHANGE\_TYPE, Change\_Info) select 'Изменено в ',

'Запись №'+ Convert(varchar (4),id\_WriteAppointment) from inserted

end

end

Представление DischargeOnSchool

create view DischargeOnSchool as

SELECT

[dbo].TypeDischarge.NameDischarge as 'Наименование выписки',

[dbo].DataCitizen.SurnameCit +' ' + dbo.DataCitizen.NameCit + ' ' +dbo.DataCitizen.PatronymicCit as 'ФИО пациента',

[dbo].DataCitizen.DateBirthCit as 'Дата рождения',

[dbo].DirectoryDisease.Name\_Disease as 'Болезнь',

[dbo].Diagnosis.TimeDisease as 'Срок лечения',

[dbo].CardTreatments.Cured as 'Статус лечения',

[dbo].WriteAppointment.times as 'Дата поступления',

[dbo].Discharge.DateDischarge as 'Дата выписки'

FROM

dbo.DataCitizen INNER JOIN

dbo.TypeDischarge INNER JOIN

dbo.Discharge ON dbo.TypeDischarge.id\_TypeDischarge = dbo.Discharge.id\_TypeDischarge INNER JOIN

dbo.DirectoryDisease INNER JOIN

dbo.Diagnosis ON dbo.DirectoryDisease.id\_Disease = dbo.Diagnosis.id\_Disease INNER JOIN

dbo.CardTreatments ON dbo.Diagnosis.id\_Diagnoz = dbo.CardTreatments.id\_Diagnoz INNER JOIN

dbo.WriteAppointment ON dbo.CardTreatments.id\_WriteAppointment = dbo.WriteAppointment.id\_WriteAppointment ON dbo.Discharge.id\_WriteAppointment = dbo.WriteAppointment.id\_WriteAppointment ON

dbo.DataCitizen.id\_Citizen = dbo.WriteAppointment.id\_Citizen INNER JOIN

dbo.TherapeuticDepartament ON dbo.Diagnosis.ID\_Room = dbo.TherapeuticDepartament.id\_Room INNER JOIN

dbo.Personal ON dbo.TherapeuticDepartament.id\_worker = dbo.Personal.id\_worker

where

NameDischarge = 'В учебное заведение'

Представление DischargeOnWork

create view DischargeOnWork as

SELECT

dbo.TypeDischarge.NameDischarge as 'Наименование выписки',

dbo.DataCitizen.SurnameCit +' ' + dbo.DataCitizen.NameCit + ' ' +dbo.DataCitizen.PatronymicCit as 'ФИО пациента',

convert(varchar,[dbo].DataCitizen.SeriesPassportCit) + ' ' + convert(varchar,[dbo].DataCitizen.NumberPassportCit) as 'Серия,номер паспорта',

dbo.DataCitizen.DateBirthCit as 'Дата рождения',

dbo.DirectoryDisease.Name\_Disease as 'Название заболевания',

dbo.WriteAppointment.times as 'Дата поступления',

dbo.Diagnosis.TimeDisease as 'Срок лечения',

dbo.CardTreatments.Cured as 'Статус лечения',

dbo.Discharge.DateDischarge as 'Дата выписки',

dbo.Personal.SurnamePers +' '+ dbo.Personal.namePers +' '+ dbo.Personal.PatronymicPers as 'ФИО врача'

FROM

dbo.DataCitizen INNER JOIN

dbo.TypeDischarge INNER JOIN

dbo.Discharge ON dbo.TypeDischarge.id\_TypeDischarge = dbo.Discharge.id\_TypeDischarge INNER JOIN

dbo.DirectoryDisease INNER JOIN

dbo.Diagnosis ON dbo.DirectoryDisease.id\_Disease = dbo.Diagnosis.id\_Disease INNER JOIN

dbo.CardTreatments ON dbo.Diagnosis.id\_Diagnoz = dbo.CardTreatments.id\_Diagnoz INNER JOIN

dbo.WriteAppointment ON dbo.CardTreatments.id\_WriteAppointment = dbo.WriteAppointment.id\_WriteAppointment ON dbo.Discharge.id\_WriteAppointment = dbo.WriteAppointment.id\_WriteAppointment ON

dbo.DataCitizen.id\_Citizen = dbo.WriteAppointment.id\_Citizen INNER JOIN

dbo.TherapeuticDepartament ON dbo.Diagnosis.ID\_Room = dbo.TherapeuticDepartament.id\_Room INNER JOIN

dbo.Personal ON dbo.TherapeuticDepartament.id\_worker = dbo.Personal.id\_worker

where

NameDischarge = 'Рабочая выписка'

Представление RoomInfo

create view RoomInfo as

SELECT

dbo.TherapeuticDepartament.id\_Room as 'Номер палаты',

dbo.TherapeuticDepartament.amountRooms as 'Количество мест',

dbo.TherapeuticDepartament.BusyRoom as 'Мест занято',

dbo.CategoriesDisease.Name\_CategoriesDisease as 'Категория заболеваний',

dbo.Personal.SurnamePers + ' ' + dbo.Personal.NamePers + ' ' + dbo.Personal.PatronymicPers

+ ', ' + dbo.SpecialityPersonal.Name\_SpecialityPersonal as 'ФИО, специальность врача'

FROM

dbo.CategoriesDisease INNER JOIN

dbo.TherapeuticDepartament ON dbo.CategoriesDisease.id\_CategoriesDisease = dbo.TherapeuticDepartament.id\_CategoriesDisease INNER JOIN

dbo.Personal ON dbo.TherapeuticDepartament.id\_worker = dbo.Personal.id\_worker INNER JOIN

dbo.Role ON dbo.Personal.id\_Role = dbo.Role.id\_Role INNER JOIN

dbo.SpecialityPersonal ON dbo.Role.id\_SpecialityPersonal = dbo.SpecialityPersonal.id\_SpecialityPersonal

Представление StorageStatus

create view StorageStatus as

SELECT

dbo.Storage.id\_spot as 'Номер ячейки склада',

dbo.Medicament.Name\_Medicament as 'Название медикамента',

dbo.Storage.Amount as 'Общее количество места',

dbo.Storage.occupiedSpace as 'Занято места'

FROM

dbo.DeliveryMedicament INNER JOIN

dbo.Medicament ON dbo.DeliveryMedicament.id\_Medicament = dbo.Medicament.id\_Medicament INNER JOIN

dbo.Storage ON dbo.DeliveryMedicament.id\_spot = dbo.Storage.id\_spot

Представление Talon

create view Talon as

SELECT

dbo.WriteAppointment.id\_WriteAppointment as 'Номер талона',

dbo.DataCitizen.SurnameCit + dbo.DataCitizen.NameCit + dbo.DataCitizen.PatronymicCit as 'ФИО пациента',

dbo.DataCitizen.Snils as 'Снилс',

dbo.FormWrite.Adress as 'Адрес',

dbo.FormWrite.Sites as 'Сайт',

dbo.FormWrite.PhoneNumber as 'Номер телефона',

dbo.FormWrite.mail as 'Почтовый ящик',

dbo.WriteAppointment.visit as 'Было ли посещение',

dbo.WriteAppointment.times as 'Время записи'

FROM

dbo.DataCitizen INNER JOIN

dbo.WriteAppointment ON dbo.DataCitizen.id\_Citizen = dbo.WriteAppointment.id\_Citizen INNER JOIN

dbo.FormWrite ON dbo.WriteAppointment.id\_FormWrite = dbo.FormWrite.id\_FormWrite

Представление DocumentForDeliveryMEdicament

create view DocumentForDeliveryMEdicament as

SELECT

dbo.Manufacturer.Name\_Manufacturer as 'Производитель',

dbo.Medicament.Name\_Medicament as 'Наименование лекарства',

dbo.CategoryOfMedicament.Name\_MedCategory as 'Категория лекарства',

dbo.DeliveryMedicament.Amount as 'Поступившее количество',

dbo.DeliveryMedicament.DateOfDelivery as 'Дата поставки',

dbo.Personal.SurnamePers + ' ' + dbo.Personal.NamePers + ' ' + dbo.Personal.PatronymicPers as 'ФИО сотрудника',

dbo.Storage.id\_spot as 'Ячейка склада',

dbo.Storage.Amount as 'Места в ячейке',

dbo.Storage.occupiedSpace as 'Занятно места'

FROM

dbo.CategoryOfMedicament INNER JOIN

dbo.Medicament ON dbo.CategoryOfMedicament.id\_CategoryOfMedicament = dbo.Medicament.id\_CategoryOfMedicament INNER JOIN

dbo.DeliveryMedicament ON dbo.Medicament.id\_Medicament = dbo.DeliveryMedicament.id\_Medicament INNER JOIN

dbo.Manufacturer ON dbo.Medicament.id\_Manufacturer = dbo.Manufacturer.id\_Manufacturer INNER JOIN

dbo.Personal ON dbo.DeliveryMedicament.id\_worker = dbo.Personal.id\_worker INNER JOIN

dbo.Storage ON dbo.DeliveryMedicament.id\_spot = dbo.Storage.id\_spot

Процедура Add\_Personal

create procedure [Add\_Personal]

@NamePers varchar (50),

@SurnamePers varchar (50),

@PatronymicPers varchar (50),

@SeriesPassportPers int,

@NumberPassportPers int,

@User\_Nick varchar (50),

@User\_Pass varchar (20)

as

IF (NOT EXISTS(SELECT Personal.id\_worker

FROM dbo.Personal

WHERE User\_Nick = @User\_Nick))

begin

OPEN SYMMETRIC KEY SSN\_Key\_01

DECRYPTION BY CERTIFICATE cert1;

insert into [dbo].[Personal]

(NamePers,SurnamePers,PatronymicPers,SeriesPassportPers,NumberPassportPers,User\_Nick,User\_Pass)

values

(@NamePers,@SurnamePers,@PatronymicPers,@SeriesPassportPers,@NumberPassportPers,@User\_Nick,EncryptByKey(Key\_GUID('SSN\_Key\_01'),convert(varbinary (max),@User\_Pass)))

end

else

THROW 50250, 'Данный логин уже существует', 1

Процедура UPD\_Personal

Create procedure [UPD\_Personal]

@SeriesPassportPers int,

@NumberPassportPers int,

@User\_Pass varchar (18)

as

OPEN SYMMETRIC KEY SSN\_Key\_01

DECRYPTION BY CERTIFICATE cert1;

if ((EncryptByKey(Key\_GUID('SSN\_Key\_01'),convert(varbinary (max),(Select Personal.User\_Pass

FROM Personal WHERE Personal.SeriesPassportPers = @SeriesPassportPers and Personal.NumberPassportPers = @NumberPassportPers and Personal\_Logical\_Delete = 0)))) != @User\_Pass)

THROW 50562, 'Старый пароль не верный', 1

Update [dbo].[Personal]

set

User\_Pass = EncryptByKey(Key\_GUID('SSN\_Key\_01'),convert(varbinary (max),@User\_Pass))

where

id\_worker = (SELECT Personal.id\_worker FROM Personal WHERE Personal.SeriesPassportPers = @SeriesPassportPers and Personal.NumberPassportPers = @NumberPassportPers and Personal\_Logical\_Delete = 0)

Процедура Add\_WorkSchedule

create procedure [Add\_WorkSchedule]

@weekdays varchar (16)

as

if (Exists(SELECT WorkSchedule.id\_WorkSchedule FROM WorkSchedule WHERE WorkSchedule.weekdays = @weekdays and WorkSchedule.WorkSchedule\_Logical\_Delete = 0))

THROW 50666, 'Данный график уже создан', 1

else if (Exists(SELECT WorkSchedule.id\_WorkSchedule FROM WorkSchedule WHERE WorkSchedule.weekdays = @weekdays and WorkSchedule.WorkSchedule\_Logical\_Delete = 1))

UPDATE [WorkSchedule]

set

WorkSchedule\_Logical\_Delete = 0

where

WorkSchedule.id\_WorkSchedule = (SELECT WorkSchedule.id\_WorkSchedule FROM WorkSchedule WHERE WorkSchedule.weekdays = @weekdays and WorkSchedule.WorkSchedule\_Logical\_Delete = 1)

else

insert into [dbo].[WorkSchedule] (weekdays)

values (@weekdays)

Процедура Add\_SpecialityPersonal

create procedure [Add\_SpecialityPersonal]

@Name\_SpecialityPersonal varchar (50)

as

if (Exists(SELECT SpecialityPersonal.id\_SpecialityPersonal FROM SpecialityPersonal WHERE SpecialityPersonal.Name\_SpecialityPersonal = @Name\_SpecialityPersonal and SpecialityPersonal\_Logical\_Delete = 0))

THROW 50665, 'Данная специализация уже создана', 1

else if (Exists(SELECT SpecialityPersonal.id\_SpecialityPersonal FROM SpecialityPersonal WHERE SpecialityPersonal.Name\_SpecialityPersonal = @Name\_SpecialityPersonal and SpecialityPersonal\_Logical\_Delete = 1))

UPDATE [SpecialityPersonal]

set

SpecialityPersonal\_Logical\_Delete = 0

where

id\_SpecialityPersonal = (SELECT SpecialityPersonal.id\_SpecialityPersonal FROM SpecialityPersonal WHERE SpecialityPersonal.Name\_SpecialityPersonal = @Name\_SpecialityPersonal and SpecialityPersonal\_Logical\_Delete = 1)

else

insert into [dbo].[SpecialityPersonal] (Name\_SpecialityPersonal)

values (@Name\_SpecialityPersonal)

Процедура Add\_Role

create procedure [Add\_Role]

@Write bit,

@CardDesign bit,

@AcceptanceMedication bit,

@ResolutionStatement bit,

@id\_SpecialityPersonal int

as

if (Exists(SELECT dbo.[Role].id\_Role FROM [Role] WHERE Write = @Write and CardDesign = @CardDesign and AcceptanceMedication = @AcceptanceMedication

and ResolutionStatement = @ResolutionStatement and id\_SpecialityPersonal = @id\_SpecialityPersonal and Role\_Logical\_Delete = 0))

THROW 50664, 'Данная роль уже существует', 1

else if (Exists(SELECT dbo.[Role].id\_Role FROM [Role] WHERE Write = @Write and CardDesign = @CardDesign and AcceptanceMedication = @AcceptanceMedication

and ResolutionStatement = @ResolutionStatement and id\_SpecialityPersonal = @id\_SpecialityPersonal and Role\_Logical\_Delete = 1))

UPDATE [Role]

set

Role\_Logical\_Delete = 0

where

id\_Role = (SELECT dbo.[Role].id\_Role FROM [Role] WHERE Write = @Write and CardDesign = @CardDesign and AcceptanceMedication = @AcceptanceMedication

and ResolutionStatement = @ResolutionStatement and id\_SpecialityPersonal = @id\_SpecialityPersonal and Role\_Logical\_Delete = 1)

else

insert into [dbo].[Role] (Write,CardDesign,AcceptanceMedication,ResolutionStatement,id\_SpecialityPersonal)

values (@Write,@CardDesign,@AcceptanceMedication,@ResolutionStatement,@id\_SpecialityPersonal)

Процедура UPD\_WorkSchedule

create procedure [UPD\_WorkSchedule]

@id\_WorkSchedule int,

@weekdays varchar (16)

as

if (@id\_WorkSchedule != (SELECT WorkSchedule.id\_WorkSchedule FROM WorkSchedule WHERE weekdays = @weekdays and WorkSchedule\_Logical\_Delete = 0))

THROW 50664, 'Данная роль уже существует', 1

else if (@id\_WorkSchedule != (SELECT WorkSchedule.id\_WorkSchedule FROM WorkSchedule WHERE weekdays = @weekdays and WorkSchedule\_Logical\_Delete = 1))

begin

Update [dbo].[WorkSchedule]

set

WorkSchedule\_Logical\_Delete = 1

where

id\_WorkSchedule = @id\_WorkSchedule

Update [dbo].[WorkSchedule]

set

WorkSchedule\_Logical\_Delete = 0

where

id\_WorkSchedule = (SELECT WorkSchedule.id\_WorkSchedule FROM WorkSchedule WHERE weekdays = @weekdays and WorkSchedule\_Logical\_Delete = 1)

end

else

Update [dbo].[WorkSchedule]

set

weekdays = @weekdays

where

id\_WorkSchedule = @id\_WorkSchedule

Процедура UPD\_SpecialityPersonal

create procedure [UPD\_SpecialityPersonal]

@id\_SpecialityPersonal int,

@Name\_SpecialityPersonal varchar (50)

as

if (@id\_SpecialityPersonal != (SELECT SpecialityPersonal.id\_SpecialityPersonal FROM SpecialityPersonal

WHERE SpecialityPersonal.Name\_SpecialityPersonal = @Name\_SpecialityPersonal and SpecialityPersonal\_Logical\_Delete = 0))

THROW 50665, 'Данная специализация уже создана', 1

else if (@id\_SpecialityPersonal != (SELECT SpecialityPersonal.id\_SpecialityPersonal FROM SpecialityPersonal

WHERE SpecialityPersonal.Name\_SpecialityPersonal = @Name\_SpecialityPersonal and SpecialityPersonal\_Logical\_Delete = 1))

begin

Update [dbo].[SpecialityPersonal]

set

SpecialityPersonal\_Logical\_Delete = 0

where

id\_SpecialityPersonal = (SELECT SpecialityPersonal.id\_SpecialityPersonal FROM SpecialityPersonal

WHERE SpecialityPersonal.Name\_SpecialityPersonal = @Name\_SpecialityPersonal and SpecialityPersonal\_Logical\_Delete = 1)

Update [dbo].[SpecialityPersonal]

set

SpecialityPersonal\_Logical\_Delete = 1

where

id\_SpecialityPersonal = @id\_SpecialityPersonal

end

else

Update [dbo].[SpecialityPersonal]

set

Name\_SpecialityPersonal = @Name\_SpecialityPersonal

where

id\_SpecialityPersonal = @id\_SpecialityPersonal

Процедура UPD\_Role

create procedure [UPD\_Role]

@id\_Role int,

@Write bit,

@CardDesign bit,

@AcceptanceMedication bit,

@ResolutionStatement bit,

@id\_SpecialityPersonal int

as

if (@id\_Role != (SELECT dbo.[Role].id\_Role FROM [Role] WHERE Write = @Write and CardDesign = @CardDesign and AcceptanceMedication = @AcceptanceMedication

and ResolutionStatement = @ResolutionStatement and id\_SpecialityPersonal = @id\_SpecialityPersonal and Role\_Logical\_Delete = 0))

THROW 50664, 'Данная роль уже существует', 1

else if (@id\_Role != (SELECT dbo.[Role].id\_Role FROM [Role] WHERE Write = @Write and CardDesign = @CardDesign and AcceptanceMedication = @AcceptanceMedication

and ResolutionStatement = @ResolutionStatement and id\_SpecialityPersonal = @id\_SpecialityPersonal and Role\_Logical\_Delete = 1))

begin

UPDATE [Role]

set

Role\_Logical\_Delete = 0

where

id\_Role = (SELECT dbo.[Role].id\_Role FROM [Role] WHERE Write = @Write and CardDesign = @CardDesign and AcceptanceMedication = @AcceptanceMedication

and ResolutionStatement = @ResolutionStatement and id\_SpecialityPersonal = @id\_SpecialityPersonal and Role\_Logical\_Delete = 1)

UPDATE [Role]

set

Role\_Logical\_Delete = 1

where

id\_Role = @id\_Role

end

else

Update [dbo].[Role]

set

Write = @Write,

CardDesign = @CardDesign,

AcceptanceMedication = @AcceptanceMedication,

ResolutionStatement = @ResolutionStatement,

id\_SpecialityPersonal = @id\_SpecialityPersonal

where

id\_Role = @id\_Role

Процедура UPD\_Personal\_For\_Admin

create procedure [UPD\_Personal\_For\_Admin]

@id\_worker int,

@NamePers varchar (50),

@SurnamePers varchar (50),

@PatronymicPers varchar (50),

@SeriesPassportPers int,

@NumberPassportPers int,

@User\_Nick varchar (50),

@ID\_Role int,

@id\_WorkSchedule int

as

IF (@id\_worker != (SELECT Personal.id\_worker FROM Personal

WHERE (SeriesPassportPers = @SeriesPassportPers and NumberPassportPers = @NumberPassportPers)

or User\_Nick = User\_Nick and Personal\_Logical\_Delete = 0))

THROW 50667, 'Данный пользователь уже существует',1

else IF (@id\_worker != (SELECT Personal.id\_worker FROM Personal

WHERE (SeriesPassportPers = @SeriesPassportPers and NumberPassportPers = @NumberPassportPers)

or User\_Nick = User\_Nick and Personal\_Logical\_Delete = 1))

begin

Update [dbo].[Personal]

set

NamePers = @NamePers,

SurnamePers = @SurnamePers,

PatronymicPers = @PatronymicPers,

Personal\_Logical\_Delete = 0,

id\_role = @id\_Role,

id\_WorkSchedule = @id\_WorkSchedule

where

id\_worker = (SELECT Personal.id\_worker FROM Personal

WHERE (SeriesPassportPers = @SeriesPassportPers and NumberPassportPers = @NumberPassportPers)

or User\_Nick = User\_Nick and Personal\_Logical\_Delete = 1)

Update [dbo].[Personal]

set

Personal\_Logical\_Delete = 1

where

id\_worker = @id\_worker

end

else

Update [dbo].[Personal]

set

NamePers = @NamePers,

SurnamePers = @SurnamePers,

PatronymicPers = @PatronymicPers,

SeriesPassportPers = @SeriesPassportPers,

NumberPassportPers = @NumberPassportPers,

User\_Nick = @User\_Nick,

id\_role = @id\_Role,

id\_WorkSchedule = @id\_WorkSchedule

where

id\_worker = @id\_worker

Процедура logdel\_WorkSchedule

create procedure [logdel\_WorkSchedule]

@id\_WorkSchedule int

as

Update [dbo].[WorkSchedule]

set

WorkSchedule\_Logical\_Delete = 1

where

id\_WorkSchedule = @id\_WorkSchedule

Update [dbo].[Personal]

set

id\_WorkSchedule = null

where

id\_WorkSchedule = @id\_WorkSchedule

Процедура logdel\_Storage

Create procedure [logdel\_Storage]

@id\_Spot\_OLD int,

@id\_Spot\_New int

as

if (EXISTS(Select DeliveryMedicament.id\_DeliveryMedicament From DeliveryMedicament WHERE DeliveryMedicament.id\_spot = @id\_Spot\_New and DeliveryMedicament.DeliveryMedicament\_Logical\_Delete = 0))

THROW 50564, 'Данныя чейка склада уже используется', 1

Update [dbo].[Storage]

set

Storage\_Logical\_Delete = 1

where

id\_Spot = @id\_Spot\_OLD

Update [dbo].[DeliveryMedicament]

set

id\_spot = @id\_Spot\_New

where

id\_Spot = @id\_Spot\_OLD

Процедура logdel\_SpecialityPersonal

create procedure [logdel\_SpecialityPersonal]

@id\_SpecialityPersonal int

as

Update [dbo].[SpecialityPersonal]

set

SpecialityPersonal\_Logical\_Delete = 1

where

id\_SpecialityPersonal = @id\_SpecialityPersonal

Update [dbo].[Role]

set

id\_SpecialityPersonal = (Select SpecialityPersonal.id\_SpecialityPersonal FROM SpecialityPersonal WHERE Name\_SpecialityPersonal = 'Неопределенный поьзователь')

where

id\_SpecialityPersonal = @id\_SpecialityPersonal

Update [dbo].[TherapeuticDepartament]

set

id\_worker = null

where

TherapeuticDepartament.id\_worker IN (SELECT dbo.Personal.id\_worker

FROM dbo.Role INNER JOIN

dbo.Personal ON dbo.Role.id\_Role = dbo.Personal.id\_Role INNER JOIN

dbo.TherapeuticDepartament ON dbo.Personal.id\_worker = dbo.TherapeuticDepartament.id\_worker

WHERE Role.id\_SpecialityPersonal = (Select SpecialityPersonal.id\_SpecialityPersonal FROM SpecialityPersonal WHERE Name\_SpecialityPersonal = 'Неопределенный пользователь'))

Процедура logdel\_Role

create procedure [logdel\_Role]

@id\_Role int

as

Update [dbo].[Role]

set

Role\_Logical\_Delete = 1

where

id\_Role = @id\_Role

Update [dbo].[Personal]

set

id\_Role = null

where

id\_Role = @id\_Role

Update [dbo].[TherapeuticDepartament]

set

id\_worker = null

where

TherapeuticDepartament.id\_worker IN (SELECT dbo.Personal.id\_worker

FROM dbo.Personal INNER JOIN

dbo.TherapeuticDepartament ON dbo.Personal.id\_worker = dbo.TherapeuticDepartament.id\_worker

where Personal.id\_Role = NULL and Personal\_Logical\_Delete = 0)

Процедура logdel\_Personal

create procedure [logdel\_Personal]

@id\_worker int

as

Update [dbo].[Personal]

set

Personal\_Logical\_Delete = 1

where

id\_worker = @id\_worker

Update [dbo].[TherapeuticDepartament]

set

id\_worker = null

where

id\_worker = @id\_worker

Процедура Search\_Role\_SpecialityPesonal

create procedure [Search\_Role\_SpecialityPesonal]

@Name\_SpecialityPersonal varchar (50)

as

SELECT dbo.Role.id\_Role as 'Номер роли', dbo.Role.Write as 'Запись', dbo.Role.CardDesign as 'Оформление карт',

dbo.Role.AcceptanceMedication as 'Приём лекарств', dbo.Role.ResolutionStatement as 'Выписка пациентов', dbo.Role.AdmissionPatient as 'Приём пациентов',

dbo.SpecialityPersonal.Name\_SpecialityPersonal as 'Специализация'

FROM dbo.Role INNER JOIN

dbo.SpecialityPersonal ON dbo.Role.id\_SpecialityPersonal = dbo.SpecialityPersonal.id\_SpecialityPersonal

WHERE Role\_Logical\_Delete = 0 and SpecialityPersonal.Name\_SpecialityPersonal like '%' + @Name\_SpecialityPersonal + '%'

Процедура Search\_\_Personal\_For\_Admin

create procedure Search\_\_Personal\_For\_Admin

@Name\_Personal varchar (50)

as

SELECT Role.id\_Role, Personal.id\_worker, dbo.Personal.SurnamePers + ' ' + dbo.Personal.NamePers + ' ' + dbo.Personal.PatronymicPers as 'ФИО',

Convert(varchar,Personal.SeriesPassportPers) + ' ' + Convert(varchar,Personal.NumberPassportPers) as 'Серия и номер', dbo.Role.id\_Role as 'Номер роли', dbo.SpecialityPersonal.Name\_SpecialityPersonal as 'Специализация'

FROM dbo.Role INNER JOIN

dbo.Personal ON dbo.Role.id\_Role = dbo.Personal.id\_Role INNER JOIN

dbo.SpecialityPersonal ON dbo.Role.id\_SpecialityPersonal = dbo.SpecialityPersonal.id\_SpecialityPersonal

WHERE Personal.Personal\_Logical\_Delete = 0 and dbo.Personal.NamePers+ ' ' +

dbo.Personal.SurnamePers + ' ' + dbo.Personal.PatronymicPers LIKE '%' + @Name\_Personal + '%'

Процедура Select\_WorkSchedule

create procedure [Select\_WorkSchedule]

as

SELECT WorkSchedule.id\_WorkSchedule, WorkSchedule.weekdays as 'График'

FROM WorkSchedule

WHERE WorkSchedule\_Logical\_Delete = 0

Процедура Select\_SpecialityPersonal

create procedure [Select\_SpecialityPersonal]

as

SELECT SpecialityPersonal.id\_SpecialityPersonal, SpecialityPersonal.Name\_SpecialityPersonal AS 'Специализация'

From SpecialityPersonal

WHERE SpecialityPersonal\_Logical\_Delete = 0

Процедура Select\_Role

create procedure [Select\_Role]

as

SELECT dbo.Role.id\_Role as 'Номер роли', dbo.Role.Write as 'Запись', dbo.Role.CardDesign as 'Оформление карт',

dbo.Role.AcceptanceMedication as 'Приём лекарств', dbo.Role.ResolutionStatement as 'Выписка пациентов', dbo.Role.AdmissionPatient as 'Приём пациентов',

dbo.SpecialityPersonal.Name\_SpecialityPersonal as 'Специализация'

FROM dbo.Role INNER JOIN

dbo.SpecialityPersonal ON dbo.Role.id\_SpecialityPersonal = dbo.SpecialityPersonal.id\_SpecialityPersonal

WHERE Role\_Logical\_Delete = 0

Процедура Select\_Personal

create procedure [Select\_Personal]

as

SELECT Role.id\_Role, Personal.id\_worker, dbo.Personal.SurnamePers + ' ' + dbo.Personal.NamePers + ' ' + dbo.Personal.PatronymicPers as 'ФИО',

Convert(varchar,Personal.SeriesPassportPers) + ' ' + Convert(varchar,Personal.NumberPassportPers) as 'Серия и номер', dbo.Role.id\_Role as 'Номер роли', dbo.SpecialityPersonal.Name\_SpecialityPersonal as 'Специализация'

FROM dbo.Role INNER JOIN

dbo.Personal ON dbo.Role.id\_Role = dbo.Personal.id\_Role INNER JOIN

dbo.SpecialityPersonal ON dbo.Role.id\_SpecialityPersonal = dbo.SpecialityPersonal.id\_SpecialityPersonal

WHERE Personal.Personal\_Logical\_Delete = 0

Функция Answer\_ZeroAmount\_CardTreatments

Create FUNCTION Answer\_ZeroAmount\_CardTreatments

(@id\_WriteAppointment int)

RETURNS bit

AS

BEGIN

DECLARE @Amswer\_Amount int

if (EXISTS(SELECT \*

FROM dbo.CardTreatments INNER JOIN

dbo.WriteAppointment ON dbo.CardTreatments.id\_WriteAppointment = dbo.WriteAppointment.id\_WriteAppointment

WHERE WriteAppointment.id\_WriteAppointment = @id\_WriteAppointment and CardTreatments.Cured = 0 and CardTreatments.CardTreatments\_Logical\_Delete = 0

and WriteAppointment.WriteAppointment\_Logical\_Delete = 0))

Return (1)

Return (0)

END

Процедура Select\_Diagnos\_For\_Name\_Disease

create procedure Select\_Diagnos\_For\_Name\_Disease

@Name\_Disease varchar (50)

as

Select dbo.DirectoryDisease.Name\_Disease as 'Название болезни', dbo.Diagnosis.TimeDisease as 'Срок лечения', dbo.Medicament.Name\_Medicament as 'Назначаемое лекарство', dbo.Diagnosis.Amount as 'Количество', dbo.TypeUse.Name\_Use as 'Способ применения', dbo.Diagnosis.ID\_Room as 'Номер палаты лечения'

FROM dbo.Diagnosis INNER JOIN

dbo.DirectoryDisease ON dbo.Diagnosis.id\_Disease = dbo.DirectoryDisease.id\_Disease INNER JOIN

dbo.Medicament ON dbo.Diagnosis.id\_Medicament = dbo.Medicament.id\_Medicament INNER JOIN

dbo.TypeUse ON dbo.Diagnosis.id\_TypeUse = dbo.TypeUse.id\_TypeUse

WHERE dbo.DirectoryDisease.Name\_Disease like '%' + @Name\_Disease + '%' and Diagnosis.Diagnosis\_Logical\_Delete = 0

Процедура Search\_CardTreatments

create procedure [dbo].[Search\_CardTreatments]

@Snils varchar(11)

as

SELECT dbo.CardTreatments.id\_WriteAppointment, dbo.DataCitizen.NameCit + ' ' + dbo.DataCitizen.SurnameCit+ ' ' + dbo.DataCitizen.PatronymicCit as 'ФИО', dbo.DataCitizen.Snils AS 'СНИЛС', COUNT(CASE

WHEN dbo.CardTreatments.CardTreatments\_Logical\_Delete = 0 THEN 1

ELSE NULL

END) as 'Количество диагнозов',

ISNULL(dbo.WriteAppointment.MedicalDepartament, 0) AS 'Палата лечения', ISNULL(WriteAppointment.СategoriesDisease, 'Пациенту не назначен активный диагноз') as 'Категория болезни'

FROM dbo.CardTreatments INNER JOIN

dbo.Diagnosis ON dbo.CardTreatments.id\_Diagnoz = dbo.Diagnosis.id\_Diagnoz INNER JOIN

dbo.WriteAppointment ON dbo.CardTreatments.id\_WriteAppointment = dbo.WriteAppointment.id\_WriteAppointment INNER JOIN

dbo.DataCitizen ON dbo.WriteAppointment.id\_Citizen = dbo.DataCitizen.id\_Citizen

WHERE dbo.WriteAppointment.visit = 1 and dbo.WriteAppointment.SentToTreatment = 1 and dbo.DataCitizen.Snils like '%' + @Snils + '%' and WriteAppointment.WriteAppointment\_Logical\_Delete = 0

GROUP BY dbo.CardTreatments.id\_WriteAppointment,dbo.DataCitizen.NameCit, dbo.DataCitizen.SurnameCit, dbo.DataCitizen.PatronymicCit, dbo.DataCitizen.Snils,

dbo.WriteAppointment.MedicalDepartament, WriteAppointment.СategoriesDisease

Процедура logdel\_WriteAppointment

create procedure [logdel\_WriteAppointment]

@id\_WriteAppointment int

as

Update [dbo].[WriteAppointment]

set

WriteAppointment\_Logical\_Delete = 1

where

id\_WriteAppointment = @id\_WriteAppointment

if (EXISTS(Select WriteAppointment.id\_WriteAppointment FROM WriteAppointment WHERE WriteAppointment.id\_WriteAppointment = @id\_WriteAppointment and WriteAppointment.visit = 1 and WriteAppointment.SentToTreatment = 1))

begin

Update [dbo].[CardTreatments]

set

CardTreatments\_Logical\_Delete = 1

where

id\_WriteAppointment = @id\_WriteAppointment

UPDATE Discharge

set

Discharge\_Logical\_Delete = 1

where Discharge.id\_card IN (Select id\_card FROM CardTreatments WHERE CardTreatments.id\_WriteAppointment = @id\_WriteAppointment)

end

Процедура logdel\_TypeUse

create procedure [logdel\_TypeUse]

@id\_TypeUse int

as

Update [dbo].[TypeUse]

set

TypeUse\_Logical\_Delete = 1

where

id\_TypeUse = @id\_TypeUse

Update [dbo].[Diagnosis]

set

id\_TypeUse = null

where

id\_TypeUse = @id\_TypeUse

Процедура logdel\_TypeDischarge

create procedure [logdel\_TypeDischarge]

@id\_TypeDischarge int

as

Update [dbo].[TypeDischarge]

set

TypeDischarge\_Logical\_Delete = 1

where

id\_TypeDischarge = @id\_TypeDischarge

Процедура logdel\_Discharge

create procedure [logdel\_Discharge]

@id\_Discharge int

as

Update [dbo].[Discharge]

set

Discharge\_Logical\_Delete = 1

where

id\_Discharge = @id\_Discharge

Процедура logdel\_DirectoryDisease

create procedure [logdel\_DirectoryDisease]

@id\_Disease int

as

Update [dbo].[DirectoryDisease]

set

DirectoryDisease\_Logical\_Delete = 1

where

id\_Disease = @id\_Disease

update Diagnosis

set

Diagnosis\_Logical\_Delete = 1

where

Diagnosis.id\_Disease = @id\_Disease

Процедура logdel\_Diagnosis

create procedure [logdel\_Diagnosis]

@id\_Diagnoz int

as

Update [dbo].[Diagnosis]

set

Diagnosis\_Logical\_Delete = 1

where

id\_Diagnoz = @id\_Diagnoz

Процедура logdel\_CardTreatments

create procedure [logdel\_CardTreatments]

@id\_card int,

@Amount\_Use int

as

declare @id\_WriteAppointment INT

SET @id\_WriteAppointment = (Select CardTreatments.id\_WriteAppointment FROM CardTreatments WHERE CardTreatments.id\_card = @id\_card and CardTreatments.CardTreatments\_Logical\_Delete = 0 and CardTreatments.Cured = 0)

if (EXISTS(Select id\_card FROM CardTreatments WHERE CardTreatments.id\_card = @id\_card and CardTreatments.CardTreatments\_Logical\_Delete = 0 and CardTreatments.Cured = 0))

begin

if (@Amount\_Use > (SELECT Diagnosis.Amount FROM dbo.CardTreatments INNER JOIN

dbo.Diagnosis ON dbo.CardTreatments.id\_Diagnoz = dbo.Diagnosis.id\_Diagnoz

WHERE CardTreatments.id\_card = @id\_card))

THROW 50563, 'Использовано больше лекарств, чем положено', 1

Update [dbo].[Storage]

set

occupiedSpace += ((Select Diagnosis.Amount FROM CardTreatments JOIN Diagnosis ON CardTreatments.id\_Diagnoz = Diagnosis.id\_Diagnoz WHERE CardTreatments.id\_card = @id\_card) - @Amount\_Use)

where

id\_spot = (SELECT DISTINCT Storage.id\_spot

FROM dbo.CardTreatments INNER JOIN

dbo.Diagnosis ON dbo.CardTreatments.id\_Diagnoz = dbo.Diagnosis.id\_Diagnoz INNER JOIN

dbo.Medicament ON dbo.Diagnosis.id\_Medicament = dbo.Medicament.id\_Medicament INNER JOIN

dbo.DeliveryMedicament ON dbo.Medicament.id\_Medicament = dbo.DeliveryMedicament.id\_Medicament INNER JOIN

dbo.Storage ON dbo.DeliveryMedicament.id\_spot = dbo.Storage.id\_spot

WHERE CardTreatments.id\_card = @id\_card)

end

Update [dbo].[CardTreatments]

set

CardTreatments\_Logical\_Delete = 1

where

id\_card = @id\_card

if (EXISTS(Select CardTreatments.id\_card FROM CardTreatments WHERE CardTreatments.id\_WriteAppointment = @id\_WriteAppointment and CardTreatments.Cured = 0 and CardTreatments.CardTreatments\_Logical\_Delete = 0))

Update [dbo].[WriteAppointment]

set

MedicalDepartament = NULL,

СategoriesDisease = NULL

where

id\_WriteAppointment = @id\_WriteAppointment

Процедура UPD\_TypeUse

Create procedure [UPD\_TypeUse]

@id\_TypeUse int,

@NameUse varchar (50)

as

if (@id\_TypeUse != (Select TypeUse.id\_TypeUse FROM TypeUse WHERE TypeUse.Name\_Use = @NameUse and TypeUse.TypeUse\_Logical\_Delete = 0))

THROW 50252, 'Данный тип приёма медикаментов уже существует',1

else if (@id\_TypeUse != (Select TypeUse.id\_TypeUse FROM TypeUse WHERE TypeUse.Name\_Use = @NameUse and TypeUse.TypeUse\_Logical\_Delete = 1))

begin

Update [dbo].[TypeUse]

set

TypeUse\_Logical\_Delete = 0

where

id\_TypeUse = (Select TypeUse.id\_TypeUse FROM TypeUse WHERE TypeUse.Name\_Use = @NameUse and TypeUse.TypeUse\_Logical\_Delete = 1)

Update [dbo].[TypeUse]

set

TypeUse\_Logical\_Delete = 1

where

id\_TypeUse = @id\_TypeUse

end

else

Update [dbo].[TypeUse]

set

Name\_Use = @NameUse

where

id\_TypeUse = @id\_TypeUse

Процедура UPD\_TypeDischarge

create procedure [UPD\_TypeDischarge]

@id\_TypeDischarge int,

@NameDischarge varchar (50),

@TypeDischarge\_Logical\_Delete bit

as

if (@id\_TypeDischarge != (Select TypeDischarge.NameDischarge FROM TypeDischarge WHERE TypeDischarge.NameDischarge = @NameDischarge and TypeDischarge.TypeDischarge\_Logical\_Delete = 0))

THROW 50251, 'Данный тип выписки уже существует',1

else if (@id\_TypeDischarge != (Select TypeDischarge.NameDischarge FROM TypeDischarge WHERE TypeDischarge.NameDischarge = @NameDischarge and TypeDischarge.TypeDischarge\_Logical\_Delete = 1))

begin

Update [dbo].[TypeDischarge]

set

TypeDischarge\_Logical\_Delete = 0

where

id\_TypeDischarge = (Select TypeDischarge.NameDischarge FROM TypeDischarge WHERE TypeDischarge.NameDischarge = @NameDischarge and TypeDischarge.TypeDischarge\_Logical\_Delete = 1)

Update [dbo].[TypeDischarge]

set

TypeDischarge\_Logical\_Delete = 1

where

id\_TypeDischarge = @id\_TypeDischarge

end

else

Update [dbo].[TypeDischarge]

set

NameDischarge = @NameDischarge

where

id\_TypeDischarge = @id\_TypeDischarge

Процедура UPD\_DirectoryDisease

create procedure [UPD\_DirectoryDisease]

@id\_Disease int,

@Name\_Disease varchar (50)

as

if (@id\_Disease != (Select DirectoryDisease.id\_Disease FROM DirectoryDisease WHERE DirectoryDisease.Name\_Disease = @Name\_Disease and DirectoryDisease.DirectoryDisease\_Logical\_Delete = 0))

THROW 50254, 'Данное название болезни уже существует',1

else if (@id\_Disease != (Select DirectoryDisease.id\_Disease FROM DirectoryDisease WHERE DirectoryDisease.Name\_Disease = @Name\_Disease and DirectoryDisease.DirectoryDisease\_Logical\_Delete = 1))

begin

Update [dbo].[DirectoryDisease]

set

DirectoryDisease\_Logical\_Delete = 0

where

id\_Disease = (Select DirectoryDisease.id\_Disease FROM DirectoryDisease WHERE DirectoryDisease.Name\_Disease = @Name\_Disease and DirectoryDisease.DirectoryDisease\_Logical\_Delete = 1)

Update [dbo].[DirectoryDisease]

set

DirectoryDisease\_Logical\_Delete = 1

where

id\_Disease = @id\_Disease

end

else

Update [dbo].[DirectoryDisease]

set

Name\_Disease = @Name\_Disease

where

id\_Disease = @id\_Disease

Процедура UPD\_Diagnosis

create procedure [UPD\_Diagnosis]

@id\_Diagnoz int,

@TimeDisease int,

@Amount int,

@ID\_Room int,

@id\_TypeUse int,

@id\_Medicament int,

@id\_Disease int,

@Diagnosis\_Logical\_Deletebit bit

as

if (@id\_Diagnoz != (Select Diagnosis.id\_Diagnoz FROM [dbo].[Diagnosis] WHERE TimeDisease = @TimeDisease and Amount = @Amount

and ID\_Room = @ID\_Room and id\_TypeUse = @id\_TypeUse and id\_Medicament = @id\_Medicament and id\_Disease = @id\_Disease and Diagnosis\_Logical\_Delete = 0))

THROW 50255, 'Данный диагноз уже существует',1

else if (@id\_Diagnoz != (Select Diagnosis.id\_Diagnoz FROM [dbo].[Diagnosis] WHERE TimeDisease = @TimeDisease and Amount = @Amount

and ID\_Room = @ID\_Room and id\_TypeUse = @id\_TypeUse and id\_Medicament = @id\_Medicament and id\_Disease = @id\_Disease and Diagnosis\_Logical\_Delete = 1))

begin

Update [dbo].[Diagnosis]

set

Diagnosis\_Logical\_Delete = 0

where

id\_Diagnoz = (Select Diagnosis.id\_Diagnoz FROM [dbo].[Diagnosis] WHERE TimeDisease = @TimeDisease and Amount = @Amount

and ID\_Room = @ID\_Room and id\_TypeUse = @id\_TypeUse and id\_Medicament = @id\_Medicament and id\_Disease = @id\_Disease and Diagnosis\_Logical\_Delete = 1)

Update [dbo].[Diagnosis]

set

Diagnosis\_Logical\_Delete = 1

where

id\_Diagnoz = @id\_Diagnoz

end

else

Update [dbo].[Diagnosis]

set

TimeDisease= @TimeDisease,

Amount = @Amount,

id\_Medicament = @id\_Medicament,

ID\_Room = @ID\_Room,

id\_TypeUse = @id\_TypeUse,

id\_Disease = @id\_Disease,

Diagnosis\_Logical\_Delete = @Diagnosis\_Logical\_Deletebit

where

id\_Diagnoz = @id\_Diagnoz

Процедура UPD\_CardTreatments

create procedure [dbo].[UPD\_CardTreatments]

@id\_WriteAppointment int,

@id\_Diagnoz int,

@Cured bit

as

DECLARE @id\_card int

SET @id\_card = (Select CardTreatments.id\_card FROM CardTreatments WHERE CardTreatments.id\_Diagnoz = @id\_Diagnoz and

CardTreatments.id\_WriteAppointment = @id\_WriteAppointment and CardTreatments.Cured = 0 and CardTreatments\_Logical\_Delete = 0)

Update [dbo].[CardTreatments]

set

Cured = @Cured

where

ID\_card = @ID\_card

if (EXISTS(Select CardTreatments.id\_card FROM CardTreatments WHERE CardTreatments.id\_WriteAppointment = @id\_WriteAppointment and CardTreatments.Cured = 0 and CardTreatments.CardTreatments\_Logical\_Delete = 0))

Update [dbo].[WriteAppointment]

set

MedicalDepartament = NULL,

СategoriesDisease = NULL

where

id\_WriteAppointment = @id\_WriteAppointment

Процедура Add\_TypeUse

create procedure [Add\_TypeUse]

@NameUse varchar (50)

as

DECLARE @id\_TypeUse int

if (EXISTS(Select TypeUse.id\_TypeUse FROM TypeUse WHERE TypeUse.Name\_Use = @NameUse and TypeUse.TypeUse\_Logical\_Delete = 1))

begin

set @id\_TypeUse = (Select TypeUse.id\_TypeUse FROM TypeUse WHERE TypeUse.Name\_Use = @NameUse)

UPDATE TypeUse

SET

TypeUse\_Logical\_Delete = 0

where

id\_TypeUse = id\_TypeUse

end

else if (EXISTS(Select TypeUse.id\_TypeUse FROM TypeUse WHERE TypeUse.Name\_Use = @NameUse and TypeUse.TypeUse\_Logical\_Delete = 0))

THROW 50252, 'Данный тип приёма медикаментов уже существует',1

else

insert into [dbo].[TypeUse] (Name\_Use)

values (@NameUse)

Процедура Add\_TypeDischarge

create procedure [Add\_TypeDischarge]

@NameDischarge varchar (50)

as

DECLARE @id\_TypeDischarge int

if (EXISTS(Select TypeDischarge.NameDischarge FROM TypeDischarge WHERE TypeDischarge.NameDischarge = @NameDischarge and TypeDischarge.TypeDischarge\_Logical\_Delete = 1))

begin

set @id\_TypeDischarge = (Select TypeDischarge.id\_TypeDischarge FROM TypeDischarge WHERE TypeDischarge.NameDischarge = @NameDischarge)

exec UPD\_TypeDischarge @id\_TypeDischarge, @NameDischarge, 0

end

else if (EXISTS(Select TypeDischarge.NameDischarge FROM TypeDischarge WHERE TypeDischarge.NameDischarge = @NameDischarge and TypeDischarge.TypeDischarge\_Logical\_Delete = 0))

THROW 50251, 'Данный тип выписки уже существует',1

else

insert into [dbo].[TypeDischarge] (NameDischarge)

values (@NameDischarge)

Процедура Add\_Discharge

create procedure [dbo].[Add\_Discharge]

@DateDischarge varchar (10),

@id\_card int,

@id\_TypeDischarge int

as

DECLARE @ID\_WriteAppointment int

SET @ID\_WriteAppointment = (SELECT CardTreatments.id\_WriteAppointment FROM CardTreatments WHERE CardTreatments.id\_card = @id\_card and CardTreatments.CardTreatments\_Logical\_Delete = 0)

if (@id\_TypeDischarge = (SELECT TypeDischarge.id\_TypeDischarge FROM TypeDischarge WHERE TypeDischarge.NameDischarge = 'Полная медицинская выписка')

and EXISTS(SELECT dbo.Discharge.id\_Discharge FROM Discharge JOIN TypeDischarge ON Discharge.id\_Discharge = TypeDischarge.id\_TypeDischarge WHERE

TypeDischarge.id\_TypeDischarge = @id\_TypeDischarge and Discharge.id\_card = @id\_card and Discharge.Discharge\_Logical\_Delete = 1))

UPDATE Discharge

set

Discharge\_Logical\_Delete = 0

else if ('Полная медицинская выписка' IN (SELECT TypeDischarge.NameDischarge

FROM dbo.CardTreatments INNER JOIN

dbo.Discharge ON dbo.CardTreatments.id\_card = dbo.Discharge.id\_card INNER JOIN

dbo.TypeDischarge ON dbo.Discharge.id\_TypeDischarge = dbo.TypeDischarge.id\_TypeDischarge INNER JOIN

dbo.WriteAppointment ON dbo.CardTreatments.id\_WriteAppointment = dbo.WriteAppointment.id\_WriteAppointment

WHERE WriteAppointment.id\_WriteAppointment = 1 and CardTreatments.CardTreatments\_Logical\_Delete = 0))

THROW 50253, 'Данный пациент уже выписан',1

else

insert into [dbo].[Discharge] (DateDischarge,id\_card,id\_TypeDischarge)

values (@DateDischarge,@id\_card,@id\_TypeDischarge)

Процедура Add\_DirectoryDisease

create procedure [Add\_DirectoryDisease]

@Name\_Disease varchar (50)

as

DECLARE @id\_DirectoryDisease int

if (EXISTS(Select DirectoryDisease.id\_Disease FROM DirectoryDisease WHERE DirectoryDisease.Name\_Disease = @Name\_Disease and DirectoryDisease.DirectoryDisease\_Logical\_Delete = 1))

begin

set @id\_DirectoryDisease = (Select DirectoryDisease.id\_Disease FROM DirectoryDisease WHERE DirectoryDisease.Name\_Disease = @Name\_Disease and DirectoryDisease\_Logical\_Delete = 1)

Update DirectoryDisease

set

DirectoryDisease\_Logical\_Delete = 0

where

DirectoryDisease.Name\_Disease = @Name\_Disease

end

else if (EXISTS(Select DirectoryDisease.id\_Disease FROM DirectoryDisease WHERE DirectoryDisease.Name\_Disease = @Name\_Disease and DirectoryDisease.DirectoryDisease\_Logical\_Delete = 0))

THROW 50254, 'Данное название болезни уже существует',1

else

insert into [dbo].[DirectoryDisease] (Name\_Disease)

values (@Name\_Disease)

Процедура Add\_Diagnosis

Create procedure [dbo].[Add\_Diagnosis]

@TimeDisease int,

@Amount int,

@ID\_Room int,

@id\_TypeUse int,

@id\_Medicament int,

@id\_Disease int

as

DECLARE @id\_Diagnosis int

if (EXISTS(Select Diagnosis.id\_Diagnoz FROM [dbo].[Diagnosis] WHERE TimeDisease = @TimeDisease and Amount = @Amount

and ID\_Room = @ID\_Room and id\_TypeUse = @id\_TypeUse and id\_Medicament = @id\_Medicament and id\_Disease = @id\_Disease and Diagnosis\_Logical\_Delete = 1))

begin

set @id\_Diagnosis = (Select Diagnosis.id\_Diagnoz FROM [dbo].[Diagnosis] WHERE TimeDisease = @TimeDisease and Amount = @Amount

and ID\_Room = @ID\_Room and id\_TypeUse = @id\_TypeUse and id\_Medicament = @id\_Medicament and id\_Disease = @id\_Disease)

exec UPD\_Diagnosis @id\_Diagnosis, @TimeDisease,@Amount,@ID\_Room,@id\_TypeUse,@id\_Medicament,@id\_Disease, 0

end

else if (EXISTS(Select Diagnosis.id\_Diagnoz FROM [dbo].[Diagnosis] WHERE TimeDisease = @TimeDisease and Amount = @Amount

and ID\_Room = @ID\_Room and id\_TypeUse = @id\_TypeUse and id\_Medicament = @id\_Medicament and id\_Disease = @id\_Disease and Diagnosis\_Logical\_Delete = 0))

THROW 50255, 'Данный диагноз уже существует',1

else

insert into [dbo].[Diagnosis] (TimeDisease,Amount,ID\_Room,id\_TypeUse,id\_Medicament,id\_Disease)

values (@TimeDisease,@Amount,@ID\_Room,@id\_TypeUse,@id\_Medicament,@id\_Disease)

Процедура Add\_CardTreatments

Create procedure [dbo].[Add\_CardTreatments]

@id\_WriteAppointment int,

@id\_Diagnoz int,

@Name\_SpecialityPersonal varchar(50)

as

DECLARE @id\_CardTreatments int

DECLARE @id\_WriteAppointment\_Old int

if (@Name\_SpecialityPersonal != (SELECT dbo.SpecialityPersonal.Name\_SpecialityPersonal

FROM dbo.Role INNER JOIN

dbo.Personal ON dbo.Role.id\_Role = dbo.Personal.id\_Role INNER JOIN

dbo.SpecialityPersonal ON dbo.Role.id\_SpecialityPersonal = dbo.SpecialityPersonal.id\_SpecialityPersonal INNER JOIN

dbo.TherapeuticDepartament ON dbo.Personal.id\_worker = dbo.TherapeuticDepartament.id\_worker INNER JOIN

dbo.Diagnosis ON dbo.TherapeuticDepartament.id\_Room = dbo.Diagnosis.ID\_Room

where Diagnosis.id\_Diagnoz = @id\_Diagnoz) and @Name\_SpecialityPersonal != 'Главный врач')

THROW 50660, 'Данный диагноз находится вне вашей компитенции или на отделение не назначен заведующий', 1

else if (EXISTS(Select CardTreatments.id\_card FROM CardTreatments WHERE CardTreatments.id\_Diagnoz = @id\_Diagnoz and

CardTreatments.id\_WriteAppointment = @id\_WriteAppointment and CardTreatments.Cured = 0 and CardTreatments\_Logical\_Delete = 0))

THROW 50256, 'Данные диагноз уже назначен и в процессе лечения',1

else if ((Select Diagnosis.ID\_Room FROM Diagnosis WHERE Diagnosis.id\_Diagnoz = @id\_Diagnoz ) !=

(Select Diagnosis.ID\_Room FROM CardTreatments JOIN dbo.Diagnosis ON dbo.CardTreatments.id\_Diagnoz = dbo.Diagnosis.id\_Diagnoz

WHERE CardTreatments.Cured = 0 and CardTreatments.id\_WriteAppointment = @id\_WriteAppointment and CardTreatments\_Logical\_Delete = 0) )

THROW 50257, 'На данный момент пациент лечится в другой палате',1

else if (EXISTS(Select CardTreatments.id\_card FROM CardTreatments WHERE CardTreatments.id\_Diagnoz = @id\_Diagnoz and

CardTreatments.id\_WriteAppointment = @id\_WriteAppointment and CardTreatments.Cured = 0 and CardTreatments\_Logical\_Delete = 1))

begin

SET @id\_CardTreatments = (Select CardTreatments.id\_card FROM CardTreatments WHERE CardTreatments.id\_Diagnoz = @id\_Diagnoz and

CardTreatments.id\_WriteAppointment = @id\_WriteAppointment and CardTreatments.Cured = 0 and CardTreatments\_Logical\_Delete = 1)

Update [dbo].[CardTreatments]

set

CardTreatments\_Logical\_Delete = 0

where

ID\_card = @id\_CardTreatments

end

else

insert into [dbo].[CardTreatments] (id\_WriteAppointment,Cured,id\_Diagnoz)

values (@id\_WriteAppointment,0,@id\_Diagnoz)

Update [dbo].[WriteAppointment]

set

MedicalDepartament = (Select Diagnosis.ID\_Room FROM Diagnosis WHERE Diagnosis.id\_Diagnoz = @id\_Diagnoz),

СategoriesDisease = (Select dbo.CategoriesDisease.Name\_CategoriesDisease FROM Diagnosis JOIN TherapeuticDepartament ON Diagnosis.ID\_Room = TherapeuticDepartament.id\_Room

JOIN CategoriesDisease ON TherapeuticDepartament.id\_Room = CategoriesDisease.id\_CategoriesDisease

Where Diagnosis.id\_Diagnoz = @id\_Diagnoz)

where

id\_WriteAppointment = @id\_WriteAppointment

Процедура Select\_TypeUse

create procedure [dbo].[Select\_TypeUse]

as

SELECT TypeUse.id\_TypeUse, TypeUse.Name\_Use

FROM TypeUse

WHERE TypeUse.TypeUse\_Logical\_Delete = 0

Процедура Select\_TypeDischarge\_Name\_NameDischarge

create procedure [dbo].[Select\_TypeDischarge\_Name\_NameDischarge]

as

SELECT TypeDischarge.id\_TypeDischarge, dbo.TypeDischarge.NameDischarge

FROM dbo.TypeDischarge

WHERE dbo.TypeDischarge.TypeDischarge\_Logical\_Delete = 0

Процедура Select\_Name\_Disease

create procedure Select\_Name\_Disease

as

SELECT DirectoryDisease.id\_Disease, dbo.DirectoryDisease.Name\_Disease

FROM dbo.DirectoryDisease

WHERE dbo.DirectoryDisease.DirectoryDisease\_Logical\_Delete = 0

Процедура Select\_Medicament

create procedure Select\_Medicament

as

SELECT dbo.Medicament.id\_Medicament, dbo.Medicament.Name\_Medicament + ' на складе: ' + CONVERT(varchar, Storage.occupiedSpace) As 'Лекарство'

FROM dbo.DeliveryMedicament

JOIN dbo.Medicament ON dbo.DeliveryMedicament.id\_DeliveryMedicament = dbo.Medicament.id\_Medicament

JOIN Storage ON dbo.DeliveryMedicament.id\_DeliveryMedicament = Storage.id\_spot

WHERE dbo.Medicament.Medicament\_Logical\_Delete = 0

exec Select\_Medicament

Процедура Select\_Id\_Departament

create procedure [dbo].[Select\_Id\_Departament]

as

SELECT dbo.TherapeuticDepartament.id\_Room, '№'+CONVERT(varchar, dbo.TherapeuticDepartament.id\_Room) + ' свободных мест: ' + CONVERT(varchar,dbo.TherapeuticDepartament.BusyRoom) as 'Терапевтическое отделение'

FROM dbo.TherapeuticDepartament

WHERE dbo.TherapeuticDepartament.amountRooms != dbo.TherapeuticDepartament.BusyRoom and dbo.TherapeuticDepartament.TherapeuticDepartament\_Logical\_Delete = 0

Процедура Select\_Discharge

create procedure [dbo].[Select\_Discharge]

@ID\_WriteAppointment int

as

SELECT Discharge.id\_Discharge, dbo.DataCitizen.NameCit + ' ' + dbo.DataCitizen.SurnameCit + ' ' + dbo.DataCitizen.PatronymicCit as 'ФИО' ,

dbo.DataCitizen.Snils as 'Снилс', dbo.TypeDischarge.NameDischarge as 'Тип выписки', dbo.Discharge.DateDischarge as 'Дата выписки'

FROM dbo.Discharge INNER JOIN

dbo.TypeDischarge ON dbo.Discharge.id\_TypeDischarge = dbo.TypeDischarge.id\_TypeDischarge INNER JOIN

dbo.WriteAppointment ON dbo.Discharge.id\_WriteAppointment = dbo.WriteAppointment.id\_WriteAppointment INNER JOIN

dbo.DataCitizen ON dbo.WriteAppointment.id\_Citizen = dbo.DataCitizen.id\_Citizen

WHERE dbo.WriteAppointment.id\_WriteAppointment = @ID\_WriteAppointment and Discharge.Discharge\_Logical\_Delete = 0

Процедура Select\_CardTreatments

create procedure [dbo].[Select\_CardTreatments]

as

SELECT dbo.CardTreatments.id\_WriteAppointment, dbo.DataCitizen.NameCit + ' ' + dbo.DataCitizen.SurnameCit+ ' ' + dbo.DataCitizen.PatronymicCit as 'ФИО', dbo.DataCitizen.Snils AS 'СНИЛС',

COUNT(CASE

WHEN dbo.CardTreatments.CardTreatments\_Logical\_Delete = 0 THEN 1

ELSE NULL

END) as 'Количество диагнозов',

ISNULL(dbo.WriteAppointment.MedicalDepartament, 0) AS 'Палата лечения', ISNULL(WriteAppointment.СategoriesDisease, 'Пациенту не назначен активный диагноз') as 'Категория болезни'

FROM dbo.CardTreatments INNER JOIN

dbo.Diagnosis ON dbo.CardTreatments.id\_Diagnoz = dbo.Diagnosis.id\_Diagnoz INNER JOIN

dbo.WriteAppointment ON dbo.CardTreatments.id\_WriteAppointment = dbo.WriteAppointment.id\_WriteAppointment INNER JOIN

dbo.DataCitizen ON dbo.WriteAppointment.id\_Citizen = dbo.DataCitizen.id\_Citizen

WHERE dbo.WriteAppointment.visit = 1 and dbo.WriteAppointment.SentToTreatment = 1 and WriteAppointment\_Logical\_Delete = 0

GROUP BY dbo.CardTreatments.id\_WriteAppointment,dbo.DataCitizen.NameCit, dbo.DataCitizen.SurnameCit, dbo.DataCitizen.PatronymicCit, dbo.DataCitizen.Snils,

dbo.WriteAppointment.MedicalDepartament, WriteAppointment.СategoriesDisease

Процедура Select\_All\_Diagnos

create procedure Select\_All\_Diagnos

as

SELECT Diagnosis.id\_Diagnoz, dbo.DirectoryDisease.Name\_Disease as 'Название болезни',

dbo.Diagnosis.TimeDisease 'Период лечения (в днях)' , dbo.Medicament.Name\_Medicament as 'Название лекарства',

dbo.Diagnosis.Amount as 'Количество медикаментов' , dbo.TypeUse.Name\_Use 'Применение' , dbo.TherapeuticDepartament.id\_Room 'Номер палаты',

dbo.CategoriesDisease.Name\_CategoriesDisease

FROM dbo.Diagnosis INNER JOIN

dbo.DirectoryDisease ON dbo.Diagnosis.id\_Disease = dbo.DirectoryDisease.id\_Disease INNER JOIN

dbo.Medicament ON dbo.Diagnosis.id\_Medicament = dbo.Medicament.id\_Medicament INNER JOIN

dbo.TherapeuticDepartament ON dbo.Diagnosis.ID\_Room = dbo.TherapeuticDepartament.id\_Room INNER JOIN

dbo.CategoriesDisease ON dbo.TherapeuticDepartament.id\_CategoriesDisease = dbo.CategoriesDisease.id\_CategoriesDisease INNER JOIN

dbo.TypeUse ON dbo.Diagnosis.id\_TypeUse = dbo.TypeUse.id\_TypeUse

WHERE dbo.Diagnosis.Diagnosis\_Logical\_Delete = 0

Процедура CardTreatments\_Diagnoz\_Select

create procedure [CardTreatments\_Diagnoz\_Select]

@id\_WriteAppointment int

as

SELECT dbo.CardTreatments.id\_card, dbo.CardTreatments.id\_WriteAppointment, dbo.CardTreatments.id\_Diagnoz, dbo.DirectoryDisease.Name\_Disease as 'Название болезни', dbo.Diagnosis.TimeDisease as 'Срок лечения', dbo.Medicament.Name\_Medicament as 'Назначаемое лекарство',

dbo.Diagnosis.Amount as 'Количество', dbo.TypeUse.Name\_Use as 'Способ применения', dbo.Diagnosis.ID\_Room as 'Номер палаты лечения', dbo.CardTreatments.Cured as 'Состояние'

FROM dbo.CardTreatments INNER JOIN

dbo.Diagnosis ON dbo.CardTreatments.id\_Diagnoz = dbo.Diagnosis.id\_Diagnoz INNER JOIN

dbo.WriteAppointment ON dbo.CardTreatments.id\_WriteAppointment = dbo.WriteAppointment.id\_WriteAppointment INNER JOIN

dbo.TypeUse ON dbo.Diagnosis.id\_TypeUse = dbo.TypeUse.id\_TypeUse INNER JOIN

dbo.DirectoryDisease ON dbo.Diagnosis.id\_Disease = dbo.DirectoryDisease.id\_Disease INNER JOIN

dbo.Medicament ON dbo.Diagnosis.id\_Medicament = dbo.Medicament.id\_Medicament

WHERE CardTreatments.id\_WriteAppointment = @id\_WriteAppointment and CardTreatments.CardTreatments\_Logical\_Delete = 0

Процедура Sum\_Diagnosis\_For\_WriteAppointment

create procedure Sum\_Diagnosis\_For\_WriteAppointment

@id\_WriteAppointment int

as

SELECT Diagnosis.id\_Diagnoz, SUM(Diagnosis.Amount)

FROM dbo.CardTreatments INNER JOIN

dbo.Diagnosis ON dbo.CardTreatments.id\_Diagnoz = dbo.Diagnosis.id\_Diagnoz INNER JOIN

dbo.WriteAppointment ON dbo.CardTreatments.id\_WriteAppointment = dbo.WriteAppointment.id\_WriteAppointment

WHERE WriteAppointment.id\_WriteAppointment = @id\_WriteAppointment and CardTreatments.CardTreatments\_Logical\_Delete = 0

GROUP BY Diagnosis.id\_Diagnoz

Процедура Cured\_All\_CardTreatmens

create procedure Cured\_All\_CardTreatmens

@id\_WriteAppointment int

as

Update [dbo].[CardTreatments]

set

Cured = 1

where

CardTreatments.id\_WriteAppointment = @id\_WriteAppointment

Процедура Celect\_TypeWrite

create procedure Celect\_TypeWrite

as

SELECT TypeWrite.id\_TypeWrite, 'Тип записи: ' + dbo.TypeWrite.Name\_Write as 'Тип записи'

FROM dbo.TypeWrite

Процедура Select\_DataCitizen\_ONLI\_FIO

create procedure Select\_DataCitizen\_ONLI\_FIO

as

SELECT id\_Citizen, SurnameCit + ' ' + NameCit + ' ' + PatronymicCit as 'ФИО'

FROM dbo.DataCitizen

Процедура Select\_DataCitizen

create procedure Select\_DataCitizen

as

SELECT id\_Citizen, SurnameCit + ' ' + NameCit + ' ' + PatronymicCit as 'ФИО', Snils as 'Снилс',

CONVERT(VARCHAR, SeriesPassportCit) + ' ' + CONVERT(VARCHAR, NumberPassportCit) AS 'Серия и номер', DateBirthCit as 'День рождения'

FROM dbo.DataCitizen

Процедура Select\_Date\_Appointment

Create procedure Select\_Date\_Appointment

@Date date

as

Select dbo.Day\_of\_the\_week.id\_Day\_of\_the\_week , dbo.Day\_of\_the\_week.Record\_Time

FROM dbo.Day\_of\_the\_week

WHERE dbo.Day\_of\_the\_week.Record\_Time not in (SELECT dbo.Day\_of\_the\_week.Record\_Time

FROM dbo.Day\_of\_the\_week INNER JOIN

dbo.WriteAppointment ON dbo.Day\_of\_the\_week.id\_Day\_of\_the\_week = dbo.WriteAppointment.id\_Day\_of\_the\_week

WHERE WriteAppointment.times = @Date and WriteAppointment\_Logical\_Delete = 0)

Процедура Select\_WriteAppointment

create procedure Select\_WriteAppointment

as

SELECT dbo.DataCitizen.id\_Citizen, dbo.Personal.id\_worker, dbo.WriteAppointment.id\_WriteAppointment, dbo.FormWrite.id\_FormWrite, dbo.Day\_of\_the\_week.id\_Day\_of\_the\_week , dbo.DataCitizen.SurnameCit + ' ' + dbo.DataCitizen.NameCit + ' ' + dbo.DataCitizen.PatronymicCit AS 'ФИО',

convert(varchar,dbo.WriteAppointment.times)+ ' ' + dbo.Day\_of\_the\_week.Record\_Time AS 'Время записи', dbo.WriteAppointment.visit AS 'Посетил', dbo.WriteAppointment.SentToTreatment AS 'Положили',

dbo.TypeWrite.Name\_Write AS 'Тип записи', dbo.Personal.SurnamePers + ' ' + dbo.Personal.NamePers + ' ' + dbo.Personal.PatronymicPers AS 'Регистратор'

FROM dbo.Day\_of\_the\_week INNER JOIN

dbo.DataCitizen INNER JOIN

dbo.TypeWrite INNER JOIN

dbo.FormWrite ON dbo.TypeWrite.id\_TypeWrite = dbo.FormWrite.id\_TypeWrite INNER JOIN

dbo.WriteAppointment ON dbo.FormWrite.id\_FormWrite = dbo.WriteAppointment.id\_FormWrite ON dbo.DataCitizen.id\_Citizen = dbo.WriteAppointment.id\_Citizen ON

dbo.Day\_of\_the\_week.id\_Day\_of\_the\_week = dbo.WriteAppointment.id\_Day\_of\_the\_week INNER JOIN

dbo.Personal ON dbo.FormWrite.id\_worker = dbo.Personal.id\_worker

Процедура Add\_DataCitizen

create procedure [Add\_DataCitizen]

@NameCit varchar (50),

@SurnameCit varchar (50),

@PatronymicCit varchar (50),

@Snils varchar (11),

@SeriesPassportCit int,

@NumberPassportCit int,

@DateBirthCit date

as

if ((SELECT

CASE

WHEN MONTH(CONVERT(date,@DateBirthCit)) > MONTH(getDate()) THEN DATEDIFF(YEAR,CONVERT(date,@DateBirthCit),getDate())-1

WHEN MONTH(CONVERT(date,@DateBirthCit)) < MONTH(getDate()) THEN DATEDIFF(YEAR,CONVERT(date,@DateBirthCit),getDate())

WHEN MONTH(CONVERT(date,@DateBirthCit)) = MONTH(getDate()) THEN

CASE

WHEN DAY(CONVERT(date,@DateBirthCit)) > DAY(getDate()) THEN DATEDIFF(YEAR,CONVERT(date,@DateBirthCit),getDate())-1

ELSE DATEDIFF(YEAR,CONVERT(date,@DateBirthCit),getDate())

END

END) < 18 )

THROW 50288, 'Возраст добавляемого гражданина меньше 18 лет!', 1

else if (EXISTS(SELECT dbo.DataCitizen.id\_Citizen FROM DataCitizen Where DataCitizen.Snils = @Snils and DataCitizen\_Logical\_Delete = 0)

and EXISTS(SELECT dbo.DataCitizen.id\_Citizen FROM DataCitizen Where DataCitizen.NumberPassportCit = @NumberPassportCit

AND DataCitizen.SeriesPassportCit = @SeriesPassportCit and DataCitizen\_Logical\_Delete = 0))

THROW 50668, 'Данный гражданин уже зарегистрирован',1

ELSE if (EXISTS(SELECT dbo.DataCitizen.id\_Citizen FROM DataCitizen Where DataCitizen.Snils = @Snils and DataCitizen\_Logical\_Delete = 1)

and EXISTS(SELECT dbo.DataCitizen.id\_Citizen FROM DataCitizen Where DataCitizen.NumberPassportCit = @NumberPassportCit

AND DataCitizen.SeriesPassportCit = @SeriesPassportCit and DataCitizen\_Logical\_Delete = 1))

UPDATE DataCitizen

set

DataCitizen\_Logical\_Delete = 0,

NameCit = @NameCit ,

SurnameCit = @SurnameCit ,

PatronymicCit = @PatronymicCit,

DateBirthCit = @DateBirthCit

where

Snils = @Snils

else

insert into [dbo].[DataCitizen] (NameCit,SurnameCit,PatronymicCit,Snils,SeriesPassportCit,NumberPassportCit,DateBirthCit)

values (@NameCit,@SurnameCit,@PatronymicCit,@Snils,@SeriesPassportCit,@NumberPassportCit,@DateBirthCit)

Процедура Add\_FormWrite

create procedure [Add\_FormWrite]

@id\_worker int,

@id\_TypeWrite int,

@mail varchar (50),

@PhoneNumber varchar (12),

@Sites varchar (100),

@Adress varchar (100)

as

if (EXISTS(Select FormWrite.id\_FormWrite FROM FormWrite WHERE id\_worker = @id\_worker and id\_TypeWrite = @id\_TypeWrite

and mail = @mail and PhoneNumber = @PhoneNumber and Sites = @Sites and @Adress = Adress))

return

else

insert into [dbo].[FormWrite] (id\_worker,id\_TypeWrite,mail,PhoneNumber,Sites,Adress)

values (@id\_worker,@id\_TypeWrite,@mail,@PhoneNumber,@Sites,@Adress)

Процедура Add\_TypeWrite

create procedure [Add\_TypeWrite]

@Name\_TypeWrite varchar (50)

as

if (EXISTS(Select TypeWrite.id\_TypeWrite FROM TypeWrite WHERE Name\_Write = @Name\_TypeWrite and TypeWrite\_Logical\_Delete = 0))

THROW 50669, 'Данный тип записи уже создан', 1

else if (EXISTS(Select TypeWrite.id\_TypeWrite FROM TypeWrite WHERE Name\_Write = @Name\_TypeWrite and TypeWrite\_Logical\_Delete = 1))

Update TypeWrite

set

TypeWrite\_Logical\_Delete = 0

where

Name\_Write = @Name\_TypeWrite

else

insert into [dbo].[TypeWrite] (Name\_Write)

values (@Name\_TypeWrite)

Процедура Add\_WriteAppointment

Create procedure [Add\_WriteAppointment]

@times date,

@id\_Day\_of\_the\_week int,

@id\_Citizen int,

@id\_FormWrite int

as

if (@times < getDate() and convert(varchar,getDate(),108) > (Select Record\_Time

FROM dbo.Day\_of\_the\_week WHERE id\_Day\_of\_the\_week = @id\_Day\_of\_the\_week))

THROW 50671, 'Невозможно создать запись на приём задним числом',1

else if (EXISTS(Select WriteAppointment.id\_WriteAppointment FROM WriteAppointment WHERE WriteAppointment\_Logical\_Delete = 0 and times = @times

and id\_Day\_of\_the\_week = @id\_Day\_of\_the\_week and id\_Citizen = @id\_Citizen and id\_FormWrite = @id\_FormWrite))

THROW 50672, 'Данный гражданин уже записан', 1

else if (EXISTS(Select WriteAppointment.id\_WriteAppointment FROM WriteAppointment WHERE WriteAppointment\_Logical\_Delete = 1 and times = @times

and id\_Day\_of\_the\_week = @id\_Day\_of\_the\_week and id\_Citizen = @id\_Citizen and id\_FormWrite = @id\_FormWrite))

Update [dbo].[WriteAppointment]

set

WriteAppointment\_Logical\_Delete = 0

where

id\_WriteAppointment = (Select WriteAppointment.id\_WriteAppointment FROM WriteAppointment WHERE WriteAppointment\_Logical\_Delete = 1 and times = @times

and id\_Day\_of\_the\_week = @id\_Day\_of\_the\_week and id\_Citizen = @id\_Citizen and id\_FormWrite = @id\_FormWrite)

else

insert into [dbo].[WriteAppointment] (times,id\_Citizen,id\_FormWrite, id\_Day\_of\_the\_week)

values (@times,@id\_Citizen,@id\_FormWrite, @id\_Day\_of\_the\_week)

Процедура UPD\_DataCitizen

create procedure [UPD\_DataCitizen]

@id\_Citizen int,

@NameCit varchar (50),

@SurnameCit varchar (50),

@PatronymicCit varchar (50),

@Snils varchar (11),

@SeriesPassportCit int,

@NumberPassportCit int,

@DateBirthCit varchar (10),

@DataCitizen\_Logical\_Delete bit

as

if ((SELECT

CASE

WHEN MONTH(CONVERT(date,@DateBirthCit)) > MONTH(getDate()) THEN DATEDIFF(YEAR,CONVERT(date,@DateBirthCit),getDate())-1

WHEN MONTH(CONVERT(date,@DateBirthCit)) < MONTH(getDate()) THEN DATEDIFF(YEAR,CONVERT(date,@DateBirthCit),getDate())

WHEN MONTH(CONVERT(date,@DateBirthCit)) = MONTH(getDate()) THEN

CASE

WHEN DAY(CONVERT(date,@DateBirthCit)) > DAY(getDate()) THEN DATEDIFF(YEAR,CONVERT(date,@DateBirthCit),getDate())-1

ELSE DATEDIFF(YEAR,CONVERT(date,@DateBirthCit),getDate())

END

END) < 18 )

THROW 50288, 'Возраст добавляемого гражданина меньше 18 лет!', 1

else if (@id\_Citizen != (SELECT dbo.DataCitizen.id\_Citizen FROM DataCitizen Where DataCitizen.Snils = @Snils and DataCitizen\_Logical\_Delete = 0)

and EXISTS(SELECT dbo.DataCitizen.id\_Citizen FROM DataCitizen Where DataCitizen.NumberPassportCit = @NumberPassportCit

AND DataCitizen.SeriesPassportCit = @SeriesPassportCit and DataCitizen\_Logical\_Delete = 0))

THROW 50668, 'Данный гражданин уже зарегистрирован',1

ELSE if (@id\_Citizen != (SELECT dbo.DataCitizen.id\_Citizen FROM DataCitizen Where DataCitizen.Snils = @Snils and DataCitizen\_Logical\_Delete = 1)

and EXISTS(SELECT dbo.DataCitizen.id\_Citizen FROM DataCitizen Where DataCitizen.NumberPassportCit = @NumberPassportCit

AND DataCitizen.SeriesPassportCit = @SeriesPassportCit and DataCitizen\_Logical\_Delete = 1))

begin

UPDATE DataCitizen

set

DataCitizen\_Logical\_Delete = 0,

NameCit = @NameCit ,

SurnameCit = @SurnameCit ,

PatronymicCit = @PatronymicCit,

DateBirthCit = @DateBirthCit

where

id\_Citizen = (SELECT dbo.DataCitizen.id\_Citizen FROM DataCitizen Where DataCitizen.Snils = @Snils and DataCitizen\_Logical\_Delete = 1)

and EXISTS(SELECT dbo.DataCitizen.id\_Citizen FROM DataCitizen Where DataCitizen.NumberPassportCit = @NumberPassportCit

AND DataCitizen.SeriesPassportCit = @SeriesPassportCit and DataCitizen\_Logical\_Delete = 1)

UPDATE DataCitizen

set

DataCitizen\_Logical\_Delete = 1

where

id\_Citizen = @id\_Citizen

end

else

Update [dbo].[DataCitizen]

set

NameCit = @NameCit,

SurnameCit = @SurnameCit,

PatronymicCit = @PatronymicCit,

Snils = @Snils,

SeriesPassportCit = @SeriesPassportCit,

NumberPassportCit = @NumberPassportCit,

DateBirthCit = @DateBirthCit,

DataCitizen\_Logical\_Delete = @DataCitizen\_Logical\_Delete

where

id\_Citizen = @id\_Citizen

Процедура UPD\_TypeWrite

create procedure [UPD\_TypeWrite]

@id\_TypeWrite int,

@Name\_TypeWrite varchar (50),

@TypeWrite\_Logical\_Delete bit

as

if (EXISTS(Select TypeWrite.id\_TypeWrite FROM TypeWrite WHERE Name\_Write = @Name\_TypeWrite and TypeWrite\_Logical\_Delete = 0))

THROW 50669, 'Данный тип записи уже создан', 1

else if (@id\_TypeWrite != (Select TypeWrite.id\_TypeWrite FROM TypeWrite WHERE Name\_Write = @Name\_TypeWrite and TypeWrite\_Logical\_Delete = 1))

begin

Update TypeWrite

set

TypeWrite\_Logical\_Delete = 0

where

Name\_Write = (Select TypeWrite.id\_TypeWrite FROM TypeWrite WHERE Name\_Write = @Name\_TypeWrite and TypeWrite\_Logical\_Delete = 1)

Update TypeWrite

set

TypeWrite\_Logical\_Delete = 1

where

Name\_Write = @Name\_TypeWrite

end

else

Update [dbo].[TypeWrite]

set

Name\_Write = @Name\_TypeWrite

where

id\_TypeWrite = @id\_TypeWrite

Процедура UPD\_WriteAppointment

create procedure [UPD\_WriteAppointment]

@id\_WriteAppointment int,

@id\_Day\_of\_the\_week int,

@visit bit,

@id\_Citizen int,

@SentToTreatment bit,

@times datetime,

@id\_FormWrite int

as

if (@times < getDate() and convert(varchar,getDate(),108) > (Select Record\_Time

FROM dbo.Day\_of\_the\_week WHERE id\_Day\_of\_the\_week = @id\_Day\_of\_the\_week))

THROW 50671, 'Невозможно создать запись на приём задним числом',1

else if (@id\_WriteAppointment != (Select WriteAppointment.id\_WriteAppointment FROM WriteAppointment WHERE WriteAppointment\_Logical\_Delete = 0 and times = @times

and id\_Day\_of\_the\_week = @id\_Day\_of\_the\_week and id\_Citizen = @id\_Citizen and id\_FormWrite = @id\_FormWrite))

THROW 50672, 'Данный гражданин уже записан', 1

else if (@id\_WriteAppointment != (Select WriteAppointment.id\_WriteAppointment FROM WriteAppointment WHERE WriteAppointment\_Logical\_Delete = 1 and times = @times

and id\_Day\_of\_the\_week = @id\_Day\_of\_the\_week and id\_Citizen = @id\_Citizen and id\_FormWrite = @id\_FormWrite))

begin

Update [dbo].[WriteAppointment]

set

WriteAppointment\_Logical\_Delete = 0

where

id\_WriteAppointment = (Select WriteAppointment.id\_WriteAppointment FROM WriteAppointment WHERE WriteAppointment\_Logical\_Delete = 1 and times = @times

and id\_Day\_of\_the\_week = @id\_Day\_of\_the\_week and id\_Citizen = @id\_Citizen and id\_FormWrite = @id\_FormWrite)

Update [dbo].[WriteAppointment]

set

WriteAppointment\_Logical\_Delete = 1

where

id\_WriteAppointment = @id\_WriteAppointment

end

else

Update [dbo].[WriteAppointment]

set

id\_Day\_of\_the\_week = @id\_Day\_of\_the\_week,

visit = @visit,

id\_Citizen = @id\_Citizen,

id\_FormWrite = @id\_FormWrite,

SentToTreatment = @SentToTreatment

where

id\_WriteAppointment = @id\_WriteAppointment

Процедура logdel\_DataCitizen

create procedure [logdel\_DataCitizen]

@id\_Citizen int

as

Update [dbo].[DataCitizen]

set

DataCitizen\_Logical\_Delete = 1

where

id\_Citizen = @id\_Citizen

Update [dbo].[WriteAppointment]

set

WriteAppointment\_Logical\_Delete = 1

where

id\_Citizen = @id\_Citizen

if (EXISTS(SELECT id\_WriteAppointment

FROM dbo.DataCitizen INNER JOIN

dbo.WriteAppointment ON dbo.DataCitizen.id\_Citizen = dbo.WriteAppointment.id\_Citizen

where DataCitizen.id\_Citizen = @id\_Citizen and WriteAppointment.visit = 1 and WriteAppointment.SentToTreatment = 1 and WriteAppointment\_Logical\_Delete = 0))

begin

Update [dbo].[CardTreatments]

set

CardTreatments\_Logical\_Delete = 1

where

id\_WriteAppointment in (SELECT id\_WriteAppointment

FROM dbo.DataCitizen INNER JOIN

dbo.WriteAppointment ON dbo.DataCitizen.id\_Citizen = dbo.WriteAppointment.id\_Citizen

where DataCitizen.id\_Citizen = @id\_Citizen and WriteAppointment\_Logical\_Delete = 0)

UPDATE Discharge

set

Discharge\_Logical\_Delete = 1

where Discharge.id\_WriteAppointment IN (SELECT Discharge.id\_Discharge

FROM dbo.DataCitizen INNER JOIN

dbo.WriteAppointment ON dbo.DataCitizen.id\_Citizen = dbo.WriteAppointment.id\_Citizen INNER JOIN

dbo.Discharge ON dbo.WriteAppointment.id\_WriteAppointment = dbo.Discharge.id\_WriteAppointment

WriteAppointment\_Logical\_Delete = 0 and DataCitizen.id\_Citizen = @id\_Citizen)

end

Процедура logdel\_TypeWrite

create procedure [logdel\_TypeWrite]

@id\_TypeWrite int

as

Update [dbo].[TypeWrite]

set

TypeWrite\_Logical\_Delete = 1

where

id\_TypeWrite = @id\_TypeWrite

Процедура Search\_DataCitizen

create procedure Search\_DataCitizen

@Name\_DateCitizen varchar (100)

AS

SELECT id\_Citizen, SurnameCit + ' ' + NameCit + ' ' + PatronymicCit as 'ФИО', Snils as 'Снилс',

CONVERT(VARCHAR, SeriesPassportCit) + ' ' + CONVERT(VARCHAR, NumberPassportCit) AS 'Серия и номер', DateBirthCit as 'День рождения'

FROM dbo.DataCitizen

WHERE SurnameCit + ' ' + NameCit + ' ' + PatronymicCit like '%' + @Name\_DateCitizen +'%'

Процедура Search\_WriteAppointment

create procedure Search\_WriteAppointment

@Name\_DateCitizen varchar (100)

as

SELECT dbo.DataCitizen.id\_Citizen, dbo.Personal.id\_worker, dbo.WriteAppointment.id\_WriteAppointment, dbo.FormWrite.id\_FormWrite, dbo.DataCitizen.SurnameCit + ' ' + dbo.DataCitizen.NameCit + ' ' + dbo.DataCitizen.PatronymicCit AS 'ФИО',

dbo.WriteAppointment.times AS 'Время записи', dbo.WriteAppointment.visit AS 'Посетил', dbo.WriteAppointment.SentToTreatment AS 'Положили',

dbo.TypeWrite.Name\_Write AS 'Тип записи', dbo.Personal.SurnamePers + ' ' + dbo.Personal.NamePers + ' ' + dbo.Personal.PatronymicPers AS 'Регистратор'

FROM dbo.TypeWrite INNER JOIN

dbo.FormWrite ON dbo.TypeWrite.id\_TypeWrite = dbo.FormWrite.id\_TypeWrite INNER JOIN

dbo.WriteAppointment ON dbo.FormWrite.id\_FormWrite = dbo.WriteAppointment.id\_FormWrite INNER JOIN

dbo.DataCitizen ON dbo.WriteAppointment.id\_Citizen = dbo.DataCitizen.id\_Citizen INNER JOIN

dbo.Personal ON dbo.FormWrite.id\_worker = dbo.Personal.id\_worker

where SurnameCit + ' ' + NameCit + ' ' + PatronymicCit like '%' + @Name\_DateCitizen +'%'

Процедура Select\_Date\_Appointment

Create procedure Select\_Date\_Appointment

@Date date

as

Select dbo.Day\_of\_the\_week.id\_Day\_of\_the\_week , dbo.Day\_of\_the\_week.Record\_Time

FROM dbo.Day\_of\_the\_week

WHERE dbo.Day\_of\_the\_week.Record\_Time not in (SELECT dbo.Day\_of\_the\_week.Record\_Time

FROM dbo.Day\_of\_the\_week INNER JOIN

dbo.WriteAppointment ON dbo.Day\_of\_the\_week.id\_Day\_of\_the\_week = dbo.WriteAppointment.id\_Day\_of\_the\_week

WHERE WriteAppointment.times = @Date and WriteAppointment\_Logical\_Delete = 0)

Процедура SELECT\_Profile\_Personal

Create procedure [dbo].[SELECT\_Profile\_Personal]

@User\_Nick varchar (50)

as

SELECT dbo.Personal.NamePers, dbo.Personal.SurnamePers, dbo.Personal.PatronymicPers, dbo.Personal.SeriesPassportPers, dbo.Personal.NumberPassportPers, dbo.Personal.id\_worker, dbo.Personal.User\_Nick,

dbo.WorkSchedule.weekdays, dbo.SpecialityPersonal.Name\_SpecialityPersonal

FROM dbo.Personal INNER JOIN

dbo.Role ON dbo.Personal.id\_Role = dbo.Role.id\_Role INNER JOIN

dbo.SpecialityPersonal ON dbo.Role.id\_SpecialityPersonal = dbo.SpecialityPersonal.id\_SpecialityPersonal INNER JOIN

dbo.WorkSchedule ON dbo.Personal.id\_WorkSchedule = dbo.WorkSchedule.id\_WorkSchedule

WHERE Personal.User\_Nick = @User\_Nick

Процедура UPD\_Personal

Create procedure [UPD\_Personal]

@SeriesPassportPers int,

@NumberPassportPers int,

@User\_Pass varchar (18),

@New\_User\_Pass varchar (18)

as

OPEN SYMMETRIC KEY SSN\_Key\_01

DECRYPTION BY CERTIFICATE cert1;

if ( (convert(char,DecryptByKey((Select Personal.User\_Pass

FROM Personal WHERE Personal.SeriesPassportPers = @SeriesPassportPers and Personal.NumberPassportPers = @NumberPassportPers and Personal\_Logical\_Delete = 0)))) != @User\_Pass)

THROW 50562, 'Старый пароль не верный', 1

else

Update [dbo].[Personal]

set

User\_Pass = EncryptByKey(Key\_GUID('SSN\_Key\_01'),convert(varbinary (max),@New\_User\_Pass))

where

id\_worker = (SELECT Personal.id\_worker FROM Personal WHERE Personal.SeriesPassportPers = @SeriesPassportPers and Personal.NumberPassportPers = @NumberPassportPers and Personal\_Logical\_Delete = 0)

Процедура Select\_CategoryOfMedicament

create procedure [dbo].[Select\_CategoryOfMedicament]

as

SELECT CategoryOfMedicament.id\_CategoryOfMedicament, CategoryOfMedicament.Name\_MedCategory FROM CategoryOfMedicament

WHERE CategoryOfMedicament.CategoryOfMedicament\_Logical\_Delete = 0

create procedure [dbo].[Select\_DeliveryMedicament]

as

SELECT dbo.DeliveryMedicament.id\_DeliveryMedicament, Medicament.id\_Medicament, dbo.Medicament.Name\_Medicament as 'Лекарство', dbo.DeliveryMedicament.Amount as 'Количество',

dbo.DeliveryMedicament.DateOfDelivery 'Дата поставки', dbo.Personal.NamePers + ' ' + dbo.Personal.SurnamePers + ' ' + dbo.Personal.PatronymicPers as 'Принял поставку'

FROM dbo.DeliveryMedicament INNER JOIN

dbo.Medicament ON dbo.DeliveryMedicament.id\_Medicament = dbo.Medicament.id\_Medicament INNER JOIN

dbo.Personal ON dbo.DeliveryMedicament.id\_worker = dbo.Personal.id\_worker

where DeliveryMedicament.DeliveryMedicament\_Logical\_Delete = 0

Процедура Select\_DeliveryMedicament

create procedure [dbo].[Select\_DeliveryMedicament]

as

SELECT dbo.DeliveryMedicament.id\_DeliveryMedicament, Medicament.id\_Medicament, dbo.Medicament.Name\_Medicament as 'Лекарство', dbo.DeliveryMedicament.Amount as 'Количество',

dbo.DeliveryMedicament.DateOfDelivery 'Дата поставки', dbo.Personal.NamePers + ' ' + dbo.Personal.SurnamePers + ' ' + dbo.Personal.PatronymicPers as 'Принял поставку'

FROM dbo.DeliveryMedicament INNER JOIN

dbo.Medicament ON dbo.DeliveryMedicament.id\_Medicament = dbo.Medicament.id\_Medicament INNER JOIN

dbo.Personal ON dbo.DeliveryMedicament.id\_worker = dbo.Personal.id\_worker

where DeliveryMedicament.DeliveryMedicament\_Logical\_Delete = 0

Процедура Select\_Manufacturer

create procedure [dbo].[Select\_Manufacturer]

as

SELECT Manufacturer.id\_Manufacturer, Manufacturer.Name\_Manufacturer FROM Manufacturer

WHERE Manufacturer.Manufacturer\_Logical\_Delete = 0

Процедура Select\_Medicament\_FOR\_Storage

create procedure Select\_Medicament\_FOR\_Storage

as

SELECT DISTINCT dbo.Medicament.id\_Medicament, Storage.id\_spot as 'Ячейка склада' , dbo.Medicament.Name\_Medicament as 'Название', dbo.CategoryOfMedicament.Name\_MedCategory as 'Категория'

, dbo.Storage.occupiedSpace as 'Количество в наличии', ISNULL(dbo.Manufacturer.Name\_Manufacturer, 'Производитель не назначен') as 'Производитель', dbo.Manufacturer.Adress as 'Адрес отправки', dbo.Manufacturer.Mail as 'Mail компании'

FROM dbo.CategoryOfMedicament INNER JOIN

dbo.Medicament ON dbo.CategoryOfMedicament.id\_CategoryOfMedicament = dbo.Medicament.id\_CategoryOfMedicament INNER JOIN

dbo.DeliveryMedicament ON dbo.Medicament.id\_Medicament = dbo.DeliveryMedicament.id\_Medicament INNER JOIN

dbo.Manufacturer ON dbo.Medicament.id\_Manufacturer = dbo.Manufacturer.id\_Manufacturer INNER JOIN

dbo.Storage ON dbo.DeliveryMedicament.id\_spot = dbo.Storage.id\_spot

WHERE dbo.Medicament.Medicament\_Logical\_Delete = 0

Процедура Select\_Medicament

Create procedure [dbo].[Select\_Medicament]

as

SELECT dbo.Medicament.id\_Medicament, dbo.Medicament.Name\_Medicament + ' на складе: ' + CONVERT(varchar, Storage.occupiedSpace) as 'Лекарство'

FROM dbo.DeliveryMedicament

JOIN dbo.Medicament ON dbo.DeliveryMedicament.id\_DeliveryMedicament = dbo.Medicament.id\_Medicament

JOIN Storage ON dbo.DeliveryMedicament.id\_DeliveryMedicament = Storage.id\_spot

WHERE dbo.Medicament.Medicament\_Logical\_Delete = 0

Процедура Select\_Storage\_Id

create procedure [dbo].[Select\_Storage\_Id]

as

SELECT Storage.id\_spot, '№' + convert(varchar,Storage.id\_spot) + ' мест: ' + CONVERT(VARCHAR,(Storage.Amount - Storage.occupiedSpace)) as 'Ячейка склада' From Storage

WHERE Storage.Storage\_Logical\_Delete = 0

Процедура Add\_Manufacturer

create procedure [Add\_Manufacturer]

@Name\_Manufacturer varchar (100),

@Adress varchar (100),

@Mail varchar (50)

as

if (EXISTS(Select Manufacturer.id\_Manufacturer FROM Manufacturer WHERE Name\_Manufacturer = @Name\_Manufacturer and Adress = @Adress and Mail = @Mail

and Manufacturer\_Logical\_Delete = 0 ))

THROW 50259, 'Данный производитель уже существует',1

else if (EXISTS(Select Manufacturer.id\_Manufacturer FROM Manufacturer WHERE Name\_Manufacturer = @Name\_Manufacturer and Adress = @Adress and Mail = @Mail

and Manufacturer\_Logical\_Delete = 1 ))

Update [dbo].[Manufacturer]

set

Manufacturer\_Logical\_Delete = 0

where

id\_Manufacturer = (Select Manufacturer.id\_Manufacturer FROM Manufacturer WHERE Name\_Manufacturer = @Name\_Manufacturer and Adress = @Adress and Mail = @Mail

and Manufacturer\_Logical\_Delete = 1 )

else

insert into [dbo].[Manufacturer] (Name\_Manufacturer,Adress,Mail)

values (@Name\_Manufacturer,@Adress,@Mail)

Процедура Add\_DeliveryMedicament

Create procedure [dbo].[Add\_DeliveryMedicament]

@Amount int,

@DateOfDelivery varchar (10),

@id\_Medicament int,

@id\_worker int,

@id\_spot int

as

if (EXISTS(SELECT DISTINCT DeliveryMedicament.id\_spot

FROM dbo.DeliveryMedicament INNER JOIN

dbo.Medicament ON dbo.DeliveryMedicament.id\_Medicament = dbo.Medicament.id\_Medicament

WHERE Medicament.Medicament\_Logical\_Delete = 0 and DeliveryMedicament.id\_spot = @id\_spot)

and @id\_Medicament NOT IN (SELECT DISTINCT DeliveryMedicament.id\_Medicament

FROM dbo.DeliveryMedicament INNER JOIN

dbo.Medicament ON dbo.DeliveryMedicament.id\_Medicament = dbo.Medicament.id\_Medicament

WHERE Medicament.Medicament\_Logical\_Delete = 0 and DeliveryMedicament.id\_spot = @id\_spot))

THROW 50260, 'Данная ячейка склада занята другим лекарством', 1

else if (DATEDIFF (DAY,@DateOfDelivery,getDate())!=0)

THROW 50261, 'Дата поставки не совпадает с сегоднешней ', 1

else if ((SELECT dbo.Storage.Amount

FROM dbo.DeliveryMedicament INNER JOIN

dbo.Medicament ON dbo.DeliveryMedicament.id\_Medicament = dbo.Medicament.id\_Medicament INNER JOIN

dbo.Storage ON dbo.DeliveryMedicament.id\_spot = dbo.Storage.id\_spot where Storage.id\_spot = @id\_spot) - (SELECT dbo.Storage.occupiedSpace

FROM dbo.DeliveryMedicament INNER JOIN

dbo.Medicament ON dbo.DeliveryMedicament.id\_Medicament = dbo.Medicament.id\_Medicament INNER JOIN

dbo.Storage ON dbo.DeliveryMedicament.id\_spot = dbo.Storage.id\_spot where Storage.id\_spot = @id\_spot) < @Amount)

throw 50565 , 'Количество медикаментов превышает допустимый размер ячейки',1

else

insert into [dbo].[DeliveryMedicament] (Amount,DateOfDelivery,id\_Medicament,id\_worker,id\_spot)

values (@Amount,@DateOfDelivery,@id\_Medicament,@id\_worker,@id\_spot)

Процедура Add\_Medicament

create procedure [Add\_Medicament]

@Name\_Medicament varchar (100),

@id\_Manufacturer int,

@id\_CategoryOfMedicament int

as

if (EXISTS(Select Medicament.id\_Medicament FROM Medicament WHERE Medicament.Name\_Medicament = @Name\_Medicament and Medicament.id\_Manufacturer = @id\_Manufacturer

and Medicament.id\_CategoryOfMedicament = @id\_CategoryOfMedicament and Medicament.Medicament\_Logical\_Delete = 0 ))

THROW 50258, 'Данное лекарство уже существует',1

else if (EXISTS(Select Medicament.id\_Medicament FROM Medicament WHERE Medicament.Name\_Medicament = @Name\_Medicament and Medicament.id\_Manufacturer = @id\_Manufacturer

and Medicament.id\_CategoryOfMedicament = @id\_CategoryOfMedicament and Medicament.Medicament\_Logical\_Delete = 1 ))

Update [dbo].[Medicament]

set

Medicament\_Logical\_Delete = 0

where

id\_Medicament = (Select Medicament.id\_Medicament FROM Medicament WHERE Medicament.Name\_Medicament = @Name\_Medicament and Medicament.id\_Manufacturer = @id\_Manufacturer

and Medicament.id\_CategoryOfMedicament = @id\_CategoryOfMedicament and Medicament.Medicament\_Logical\_Delete = 1 )

else

insert into [dbo].[Medicament] (Name\_Medicament,id\_Manufacturer,id\_CategoryOfMedicament)

values (@Name\_Medicament,@id\_Manufacturer,@id\_CategoryOfMedicament)

Процедура Add\_Storage

create procedure [Add\_Storage]

@Amount int

as

insert into [dbo].[Storage] (Amount)

values (@Amount)

Процедура UPD\_DeliveryMedicament

Create procedure [UPD\_DeliveryMedicament]

@id\_DeliveryMedicament int,

@Amount int,

@DateOfDelivery varchar (10),

@id\_Medicament int,

@id\_worker int,

@id\_spot int

as

if (DATEDIFF (DAY,@DateOfDelivery,getDate()) != 0)

THROW 50659, 'Невозможно изменить поставку передним или задним числом',1

else if (EXISTS(SELECT DISTINCT DeliveryMedicament.id\_spot

FROM dbo.DeliveryMedicament INNER JOIN

dbo.Medicament ON dbo.DeliveryMedicament.id\_Medicament = dbo.Medicament.id\_Medicament

WHERE Medicament.Medicament\_Logical\_Delete = 0 and DeliveryMedicament.id\_Medicament = @id\_Medicament)

and @id\_spot NOT IN (SELECT DISTINCT DeliveryMedicament.id\_spot

FROM dbo.DeliveryMedicament INNER JOIN

dbo.Medicament ON dbo.DeliveryMedicament.id\_Medicament = dbo.Medicament.id\_Medicament

WHERE Medicament.Medicament\_Logical\_Delete = 0 and DeliveryMedicament.id\_Medicament = @id\_Medicament))

THROW 50260, 'Данная ячейка склада занята другим лекарством', 1

else if ((SELECT dbo.Storage.Amount

FROM dbo.DeliveryMedicament INNER JOIN

dbo.Medicament ON dbo.DeliveryMedicament.id\_Medicament = dbo.Medicament.id\_Medicament INNER JOIN

dbo.Storage ON dbo.DeliveryMedicament.id\_spot = dbo.Storage.id\_spot where Storage.id\_spot = @id\_spot) - (SELECT dbo.Storage.occupiedSpace

FROM dbo.DeliveryMedicament INNER JOIN

dbo.Medicament ON dbo.DeliveryMedicament.id\_Medicament = dbo.Medicament.id\_Medicament INNER JOIN

dbo.Storage ON dbo.DeliveryMedicament.id\_spot = dbo.Storage.id\_spot where Storage.id\_spot = @id\_spot) < @Amount)

throw 50565 , 'Количество медикаментов превышает допустимый размер ячейки',1

else

Update [dbo].[DeliveryMedicament]

set

Amount = @Amount,

DateOfDelivery = @DateOfDelivery,

id\_Medicament = @id\_Medicament,

id\_worker =@id\_worker,

id\_spot = @id\_spot

where

id\_DeliveryMedicament = @id\_DeliveryMedicament

Процедура UPD\_Manufacturer

create procedure [UPD\_Manufacturer]

@id\_Manufacturer int,

@Name\_Manufacturer varchar (100),

@Adress varchar (100),

@Mail varchar (50)

as

IF( @id\_Manufacturer != (Select Manufacturer.id\_Manufacturer From Manufacturer WHERE Name\_Manufacturer = @Name\_Manufacturer and Adress =@Adress and Mail = @Mail and Manufacturer\_Logical\_Delete = 0))

THROW 50661, 'Данный производитель уже существует', 1

else if ( @id\_Manufacturer != (Select Manufacturer.id\_Manufacturer From Manufacturer WHERE Name\_Manufacturer = @Name\_Manufacturer and Adress =@Adress and Mail = @Mail and Manufacturer\_Logical\_Delete = 1))

begin

Update [dbo].[Manufacturer]

set

Manufacturer\_Logical\_Delete = 0

where

id\_Manufacturer = (Select Manufacturer.id\_Manufacturer From Manufacturer WHERE Name\_Manufacturer = @Name\_Manufacturer and Adress =@Adress and Mail = @Mail and Manufacturer\_Logical\_Delete = 1)

Update [dbo].[Manufacturer]

set

Manufacturer\_Logical\_Delete = 1

where

id\_Manufacturer = @id\_Manufacturer

end

else

Update [dbo].[Manufacturer]

set

Name\_Manufacturer = @Name\_Manufacturer,

Adress = @Adress,

mail = @mail

where

id\_Manufacturer = @id\_Manufacturer

Процедура UPD\_Medicament

create procedure [UPD\_Medicament]

@id\_Medicament int,

@Name\_Medicament varchar (100),

@id\_Manufacturer int,

@id\_CategoryOfMedicament int

as

if (@id\_Medicament != (Select Medicament.id\_Medicament FROM Medicament WHERE Medicament.Name\_Medicament = @Name\_Medicament and Medicament.id\_Manufacturer = @id\_Manufacturer

and Medicament.id\_CategoryOfMedicament = @id\_CategoryOfMedicament and Medicament.Medicament\_Logical\_Delete = 0))

THROW 50657, 'Данный медикамент уже создан',1

else if (@id\_Medicament != (Select Medicament.id\_Medicament FROM Medicament WHERE Medicament.Name\_Medicament = @Name\_Medicament and Medicament.id\_Manufacturer = @id\_Manufacturer

and Medicament.id\_CategoryOfMedicament = @id\_CategoryOfMedicament and Medicament.Medicament\_Logical\_Delete = 1))

begin

Update [dbo].[Medicament]

set

Medicament\_Logical\_Delete = 0

where

id\_Medicament = (Select Medicament.id\_Medicament FROM Medicament WHERE Medicament.Name\_Medicament = @Name\_Medicament and Medicament.id\_Manufacturer = @id\_Manufacturer

and Medicament.id\_CategoryOfMedicament = @id\_CategoryOfMedicament and Medicament.Medicament\_Logical\_Delete = 1)

Update [dbo].[Medicament]

set

Medicament\_Logical\_Delete = 1

where

id\_Medicament = @id\_Medicament

end

else

Update [dbo].[Medicament]

set

Name\_Medicament = @Name\_Medicament,

id\_Manufacturer = @id\_Manufacturer,

id\_CategoryOfMedicament = @id\_CategoryOfMedicament

where

id\_Medicament = @id\_Medicament

Процедура UPD\_Storage

Create procedure [UPD\_Storage]

@id\_Spot int,

@Amount int

as

if ( @Amount < (Select Storage.occupiedSpace FROM Storage WHERE Storage.Storage\_Logical\_Delete = 0 and Storage.id\_spot = @id\_Spot))

THROW 50565, 'Количество медикаментов превышает допустимый размер ячейки', 1

Update [dbo].[Storage]

set

Amount = @Amount

where

id\_Spot = @id\_Spot

Процедура logdel\_DeliveryMedicament

Create procedure [logdel\_DeliveryMedicament]

@id\_DeliveryMedicament int,

@id\_Medicament int,

@Amount\_Use int

as

if ( @Amount\_Use > (

SELECT DISTINCT dbo.Storage.occupiedSpace

FROM dbo.DeliveryMedicament INNER JOIN

dbo.Medicament ON dbo.DeliveryMedicament.id\_Medicament = dbo.Medicament.id\_Medicament INNER JOIN

dbo.Storage ON dbo.DeliveryMedicament.id\_spot = dbo.Storage.id\_spot

WHERE DeliveryMedicament.id\_Medicament = @id\_Medicament and Medicament.Medicament\_Logical\_Delete = 0))

THROW 50656,' На данный момент на складе отсутствует данное количество лекарственных средств',1

Update [dbo].[Storage]

set

occupiedSpace -= @Amount\_Use

where

id\_spot = (SELECT DISTINCT dbo.Storage.id\_spot

FROM dbo.DeliveryMedicament INNER JOIN

dbo.Medicament ON dbo.DeliveryMedicament.id\_Medicament = dbo.Medicament.id\_Medicament INNER JOIN

dbo.Storage ON dbo.DeliveryMedicament.id\_spot = dbo.Storage.id\_spot

WHERE DeliveryMedicament.id\_Medicament = @id\_Medicament and Medicament.Medicament\_Logical\_Delete = 0)

Update [dbo].[DeliveryMedicament]

set

DeliveryMedicament\_Logical\_Delete = 1

where

id\_DeliveryMedicament = @id\_DeliveryMedicament

Процедура logdel\_Manufacturer

create procedure [logdel\_Manufacturer]

@id\_Manufacturer int

as

Update [dbo].[Manufacturer]

set

Manufacturer\_Logical\_Delete = 1

where

id\_Manufacturer = @id\_Manufacturer

Update [dbo].[Medicament]

set

id\_Manufacturer = null

where

id\_Manufacturer = @id\_Manufacturer

Процедура logdel\_Medicament

Create procedure [logdel\_Medicament]

@id\_Medicament int

as

Update [dbo].[Storage]

set

occupiedSpace = 0

where

id\_Spot = (SELECT DISTINCT DeliveryMedicament.id\_spot

FROM dbo.DeliveryMedicament INNER JOIN

dbo.Medicament ON dbo.DeliveryMedicament.id\_Medicament = dbo.Medicament.id\_Medicament

WHERE Medicament.id\_Medicament = @id\_Medicament and Medicament.Medicament\_Logical\_Delete = 0 and DeliveryMedicament.DeliveryMedicament\_Logical\_Delete = 0

)

Update [dbo].[Medicament]

set

Medicament\_Logical\_Delete = 1

where

id\_Medicament = @id\_Medicament

Процедура logdel\_Storage

Create procedure [logdel\_Storage]

@id\_Spot\_OLD int,

@id\_Spot\_New int

as

if (EXISTS(Select DeliveryMedicament.id\_DeliveryMedicament From DeliveryMedicament WHERE DeliveryMedicament.id\_spot = @id\_Spot\_New and DeliveryMedicament.DeliveryMedicament\_Logical\_Delete = 0))

THROW 50564, 'Данныя чейка склада уже используется', 1

Update [dbo].[Storage]

set

Storage\_Logical\_Delete = 1

where

id\_Spot = @id\_Spot\_OLD

Update [dbo].[DeliveryMedicament]

set

id\_spot = @id\_Spot\_New

where

id\_Spot = @id\_Spot\_OLD

Процедура Search\_DeliveryMedicamen

create Procedure Search\_DeliveryMedicamen

@Date\_DeliveryMedicament varchar(10)

as

SELECT dbo.DeliveryMedicament.id\_DeliveryMedicament, Medicament.id\_Medicament, dbo.Medicament.Name\_Medicament as 'Лекарство', dbo.DeliveryMedicament.Amount as 'Количество',

dbo.DeliveryMedicament.DateOfDelivery 'Дата поставки', dbo.Personal.NamePers + ' ' + dbo.Personal.SurnamePers + ' ' + dbo.Personal.PatronymicPers as 'Принял поставку'

FROM dbo.DeliveryMedicament INNER JOIN

dbo.Medicament ON dbo.DeliveryMedicament.id\_Medicament = dbo.Medicament.id\_Medicament INNER JOIN

dbo.Personal ON dbo.DeliveryMedicament.id\_worker = dbo.Personal.id\_worker

where DeliveryMedicament.DeliveryMedicament\_Logical\_Delete = 0 and DeliveryMedicament.DateOfDelivery like '%' + @Date\_DeliveryMedicament + '%'

Процедура Search\_Medicament

create Procedure Search\_Medicament

@Name\_Medicament varchar(50)

as

SELECT DISTINCT dbo.Medicament.id\_Medicament, Storage.id\_spot as 'Ячейка склада' , dbo.Medicament.Name\_Medicament as 'Название', dbo.CategoryOfMedicament.Name\_MedCategory as 'Категория'

, dbo.Storage.occupiedSpace as 'Количество в наличии', ISNULL(dbo.Manufacturer.Name\_Manufacturer, 'Производитель не назначен') as 'Производитель', dbo.Manufacturer.Adress as 'Адрес отправки', dbo.Manufacturer.Mail as 'Mail компании'

FROM dbo.CategoryOfMedicament INNER JOIN

dbo.Medicament ON dbo.CategoryOfMedicament.id\_CategoryOfMedicament = dbo.Medicament.id\_CategoryOfMedicament INNER JOIN

dbo.DeliveryMedicament ON dbo.Medicament.id\_Medicament = dbo.DeliveryMedicament.id\_Medicament INNER JOIN

dbo.Manufacturer ON dbo.Medicament.id\_Manufacturer = dbo.Manufacturer.id\_Manufacturer INNER JOIN

dbo.Storage ON dbo.DeliveryMedicament.id\_spot = dbo.Storage.id\_spot

WHERE dbo.Medicament.Medicament\_Logical\_Delete = 0 and Medicament.Name\_Medicament like '%' + @Name\_Medicament + '%'

Процедура Select\_CategoriesDisease

create procedure Select\_CategoriesDisease

as

SELECT CategoriesDisease.id\_CategoriesDisease, Name\_CategoriesDisease

FROM dbo.CategoriesDisease

WHERE CategoriesDisease.CategoriesDisease\_Logical\_Delete = 0

Процедура Select\_Personal\_Like\_Category

create procedure Select\_Personal\_Like\_Category

as

SELECT dbo.Personal.id\_worker, dbo.Personal.NamePers+ ' ' +

dbo.Personal.SurnamePers + ' ' + dbo.Personal.PatronymicPers as 'Сотрудник', dbo.SpecialityPersonal.Name\_SpecialityPersonal as 'Специализация'

FROM dbo.Personal INNER JOIN

dbo.Role ON dbo.Personal.id\_Role = dbo.Role.id\_Role INNER JOIN

dbo.SpecialityPersonal ON dbo.Role.id\_SpecialityPersonal = dbo.SpecialityPersonal.id\_SpecialityPersonal

WHERE SpecialityPersonal.Name\_SpecialityPersonal not LIKE 'Админ'

and SpecialityPersonal.Name\_SpecialityPersonal not LIKE 'Подсобный рабочий'

and SpecialityPersonal.Name\_SpecialityPersonal not LIKE 'Регистратор'

and SpecialityPersonal.Name\_SpecialityPersonal not LIKE 'Неопределенный пользователь' and

Personal\_Logical\_Delete = 0

Процедура Select\_TherapeuticDepartament

create procedure Select\_TherapeuticDepartament

as

SELECT dbo.TherapeuticDepartament.id\_Room as 'Номер отделения', dbo.TherapeuticDepartament.amountRooms as 'Всего мест',

dbo.TherapeuticDepartament.BusyRoom as 'Занято мест', dbo.CategoriesDisease.Name\_CategoriesDisease as 'Категория болезни', dbo.Personal.NamePers+ ' ' +

dbo.Personal.SurnamePers + ' ' + dbo.Personal.PatronymicPers as 'Заведующий', dbo.SpecialityPersonal.Name\_SpecialityPersonal as 'Специализация'

FROM dbo.Role INNER JOIN

dbo.Personal ON dbo.Role.id\_Role = dbo.Personal.id\_Role INNER JOIN

dbo.SpecialityPersonal ON dbo.Role.id\_SpecialityPersonal = dbo.SpecialityPersonal.id\_SpecialityPersonal INNER JOIN

dbo.TherapeuticDepartament ON dbo.Personal.id\_worker = dbo.TherapeuticDepartament.id\_worker INNER JOIN

dbo.CategoriesDisease ON dbo.TherapeuticDepartament.id\_CategoriesDisease = dbo.CategoriesDisease.id\_CategoriesDisease

WHERE TherapeuticDepartament.TherapeuticDepartament\_Logical\_Delete = 0

Процедура Add\_CategoriesDisease

create procedure [Add\_CategoriesDisease]

@Name\_CategoriesDisease varchar (50)

as

if (EXISTS(SELECT CategoriesDisease.id\_CategoriesDisease FROM CategoriesDisease WHERE Name\_CategoriesDisease = @Name\_CategoriesDisease and CategoriesDisease\_Logical\_Delete = 0))

THROW 50662, 'Данная категория уже существует', 1

else if (EXISTS(SELECT CategoriesDisease.id\_CategoriesDisease FROM CategoriesDisease WHERE Name\_CategoriesDisease = @Name\_CategoriesDisease and CategoriesDisease\_Logical\_Delete = 1))

UPDATE CategoriesDisease

set

CategoriesDisease\_Logical\_Delete = 0

where

CategoriesDisease.Name\_CategoriesDisease = @Name\_CategoriesDisease

else

insert into [dbo].[CategoriesDisease] (Name\_CategoriesDisease)

values (@Name\_CategoriesDisease)

Процедура Add\_TherapeuticDepartament

create procedure [Add\_TherapeuticDepartament]

@amountRooms int,

@id\_worker int,

@id\_CategoriesDisease int

as

insert into [dbo].[TherapeuticDepartament] (amountRooms,BusyRoom,id\_worker,id\_CategoriesDisease)

values (@amountRooms,@id\_worker,@id\_CategoriesDisease)

Процедура UPD\_CategoriesDisease

create procedure [UPD\_CategoriesDisease]

@id\_CategoriesDisease int,

@Name\_CategoriesDisease varchar (50)

as

DECLARE @Answer int

SET @Answer = (Select dbo.Answer\_UPD\_Unique(@id\_CategoriesDisease,@Name\_CategoriesDisease))

if ( @Answer = 0)

THROW 50662, 'Данная категория уже существует', 1

else if (@Answer = 1)

begin

Update [dbo].[CategoriesDisease]

set

CategoriesDisease\_Logical\_Delete = 0

where

ID\_CategoriesDisease = (SELECT CategoriesDisease.id\_CategoriesDisease FROM CategoriesDisease WHERE CategoriesDisease.Name\_CategoriesDisease = @Name\_CategoriesDisease and CategoriesDisease\_Logical\_Delete = 1)

Update [dbo].[CategoriesDisease]

set

CategoriesDisease\_Logical\_Delete = 1

where

ID\_CategoriesDisease = @ID\_CategoriesDisease

end

else

Update [dbo].[CategoriesDisease]

set

Name\_CategoriesDisease = @Name\_CategoriesDisease

where

ID\_CategoriesDisease = @ID\_CategoriesDisease

Процедура UPD\_TherapeuticDepartament

create procedure [UPD\_TherapeuticDepartament]

@id\_Room int,

@amountRooms int,

@id\_worker int,

@id\_CategoriesDisease int

as

DECLARE @Answer\_UPD\_TherapeuticDepartament\_Amount bit

SET @Answer\_UPD\_TherapeuticDepartament\_Amount = (select dbo.Answer\_UPD\_TherapeuticDepartament\_Amount (@id\_Room,@amountRooms))

if (@Answer\_UPD\_TherapeuticDepartament\_Amount = 0)

THROW 50663, 'Количество мест не может быть меньше количества занятых', 1

else

Update [dbo].[TherapeuticDepartament]

set

amountRooms = @amountRooms,

id\_worker = @id\_worker,

id\_CategoriesDisease = @id\_CategoriesDisease

where

id\_Room = @id\_Room

Функция Answer\_UPD\_TherapeuticDepartament\_Amount

create FUNCTION Answer\_UPD\_TherapeuticDepartament\_Amount(@id\_Room int, @amountRooms int)

RETURNS bit

AS

BEGIN

IF (@amountRooms < (Select TherapeuticDepartament.BusyRoom FROM TherapeuticDepartament Where id\_Room = @id\_Room))

Return (0)

Return (1)

END

Функция Answer\_UPD\_Unique

create FUNCTION Answer\_UPD\_Unique

(@id\_CategoriesDisease int, @Name\_CategoriesDisease varchar(50))

RETURNS int

AS

BEGIN

IF (EXISTS(SELECT CategoriesDisease.id\_CategoriesDisease FROM CategoriesDisease WHERE CategoriesDisease.Name\_CategoriesDisease = @Name\_CategoriesDisease and CategoriesDisease\_Logical\_Delete = 0)

and @id\_CategoriesDisease != (SELECT CategoriesDisease.id\_CategoriesDisease FROM CategoriesDisease WHERE CategoriesDisease.Name\_CategoriesDisease = @Name\_CategoriesDisease and CategoriesDisease\_Logical\_Delete = 0))

Return (0)

else IF (EXISTS(SELECT CategoriesDisease.id\_CategoriesDisease FROM CategoriesDisease WHERE CategoriesDisease.Name\_CategoriesDisease = @Name\_CategoriesDisease and CategoriesDisease\_Logical\_Delete = 1)

and @id\_CategoriesDisease != (SELECT CategoriesDisease.id\_CategoriesDisease FROM CategoriesDisease WHERE CategoriesDisease.Name\_CategoriesDisease = @Name\_CategoriesDisease and CategoriesDisease\_Logical\_Delete = 1))

Return (1)

Return (2)

END

Процедура logdel\_CategoriesDisease

create procedure [logdel\_CategoriesDisease]

@id\_CategoriesDisease int

as

Update [dbo].[CategoriesDisease]

set

CategoriesDisease\_Logical\_Delete = 1

where

id\_CategoriesDisease = @id\_CategoriesDisease

Процедура logdel\_TherapeuticDepartament

create procedure [logdel\_TherapeuticDepartament]

@id\_Room int

as

Update [dbo].[TherapeuticDepartament]

set

TherapeuticDepartament\_Logical\_Delete = 1

where

id\_Room = @id\_Room

Update [dbo].[Diagnosis]

set

Diagnosis\_Logical\_Delete = 1

where

id\_Room = @id\_Room

Процедура Search\_TherapeuticDepartament

create procedure Search\_TherapeuticDepartament

@id\_Room int

as

SELECT dbo.TherapeuticDepartament.id\_Room as 'Номер отделения', dbo.TherapeuticDepartament.amountRooms as 'Всего мест',

dbo.TherapeuticDepartament.BusyRoom as 'Занято мест', dbo.CategoriesDisease.Name\_CategoriesDisease as 'Категория болезни', dbo.Personal.NamePers+ ' ' +

dbo.Personal.SurnamePers + ' ' + dbo.Personal.PatronymicPers as 'Заведующий', dbo.SpecialityPersonal.Name\_SpecialityPersonal as 'Специализация'

FROM dbo.Role INNER JOIN

dbo.Personal ON dbo.Role.id\_Role = dbo.Personal.id\_Role INNER JOIN

dbo.SpecialityPersonal ON dbo.Role.id\_SpecialityPersonal = dbo.SpecialityPersonal.id\_SpecialityPersonal INNER JOIN

dbo.TherapeuticDepartament ON dbo.Personal.id\_worker = dbo.TherapeuticDepartament.id\_worker INNER JOIN

dbo.CategoriesDisease ON dbo.TherapeuticDepartament.id\_CategoriesDisease = dbo.CategoriesDisease.id\_CategoriesDisease

WHERE TherapeuticDepartament.TherapeuticDepartament\_Logical\_Delete = 0 and TherapeuticDepartament.id\_Room = @id\_Room

Процедура Select\_Personal\_Like\_Category

create procedure Select\_Personal\_Like\_Category

@Name\_Personal varchar (50)

as

SELECT dbo.Personal.id\_worker, dbo.Personal.NamePers+ ' ' +

dbo.Personal.SurnamePers + ' ' + dbo.Personal.PatronymicPers as 'Сотрудник', dbo.SpecialityPersonal.Name\_SpecialityPersonal as 'Специализация'

FROM dbo.Personal INNER JOIN

dbo.Role ON dbo.Personal.id\_Role = dbo.Role.id\_Role INNER JOIN

dbo.SpecialityPersonal ON dbo.Role.id\_SpecialityPersonal = dbo.SpecialityPersonal.id\_SpecialityPersonal

WHERE SpecialityPersonal.Name\_SpecialityPersonal not LIKE 'Админ'

and SpecialityPersonal.Name\_SpecialityPersonal not LIKE 'Подсобный рабочий'

and SpecialityPersonal.Name\_SpecialityPersonal not LIKE 'Регистратор'

and SpecialityPersonal.Name\_SpecialityPersonal not LIKE 'Неопределенный поьзователь' and

Personal\_Logical\_Delete = 0 and dbo.Personal.NamePers+ ' ' +

dbo.Personal.SurnamePers + ' ' + dbo.Personal.PatronymicPers LIKE '%' + @Name\_Personal + '%'

# 2. СТРУКТУРА КОДА ПРОГРАММЫ

## 2.1. Описание модулей программы

Таблица 2 – Описание модулей программы

|  |  |
| --- | --- |
| Название модуля | Описание модуля |
| Autoriz.cs | Модуль для осуществления подключения к базе и авторизации аккуанта для манимулирования в программе |
| ConnectionForm.cs | Модуль для осуществления подключения к серверам и выбору конкретной БД для работы приложения |
| CreateWriteForm.cs | Модуль для манипулирования таблицей формы записи |
| DataBaseConfiguration.cs | Модуль для |
| DB.cs | Модуль для подключения и выгрузке таблиц из БД |
| DeliMediForm.cs | Модуль для манипулирования таблицами по поставке медикаментов |
| DiagForm.cs | Модуль манипулирования таблицами для создания диагнозов |
| DischargeForm.cs | Модуль манипулирования таблицами для создания выписок |
| DocumentOutPut.cs | Модуль для создания шаблонов выходных документов в различных форматах |
| EditDelForm.cs | Модуль, осуществляющий вызов диалогового окна пользователю |
| MedicamentForm.cs | Модуль манипулирования таблицами для создания новых медикаментов |
| ProfileForm.cs | Модуль для вывода информации о авторизированном персонале, возможность смены пароля |
| RegForm.cs | Модуль для создания нового аккунта персонала |

## 2.2. Код программных модулей

Autoriz.cs

using System;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Windows.Forms;

namespace WindowsFormsApp9

{

class AutorizForm

{

public static int id\_Worker = 0;

static string Nick = "";

public static string Spec = "";

public static bool[] role = new bool[5];

//0 - запись

//1 - оформление карт

//2 - прием медикаментов

//3 - разрешение на выписку

//4 - прием пациентов

private Form FAutoriz = new Form();

private Label lab = new Label();

private TextBox Tb = new TextBox();

private MaskedTextBox mtb = new MaskedTextBox();

private Button but = new Button();

private Font DefFont = new Font("Microsoft Sans Serif", 10);

private Color DefBackColor = Color.FromName("Control");

private Color DefTextColor = Color.FromName("ControlText");

//Позиции //позиции

private Point locAutLabel = new Point(20, 10); //label логина

private Point locAutTB = new Point(20, 33); //TextBox логина

private Point locPassLabel = new Point(20, 63); //label пароля

private Point locPassTB = new Point(20, 86); //TextBox пароля

private Point locEntBut = new Point(20, 116); //кнопка вход

private Point locCanBut = new Point(90, 116); //кнопка отмены

//размеры //размеры

private Size sizeAutTB = new Size(150, 20); //TextBox логина

private Size sizePassTB = new Size(150, 20); //TextBox пароля

private Size sizeEntBut = new Size(70, 30); //кнопка вход

private Size sizeCanBut = new Size(80, 30); //кнопка отмены

public void NewAutorizForm()

{

CreateLoginLabel();

CreateLoginTB();

CreatePassLabel();

CreatePassTB();

CreateEnterButton();

CreateCancelButton();

FAutoriz.Text = "Вход";

FAutoriz.Size = new Size(205,195);

FAutoriz.FormBorderStyle = FormBorderStyle.FixedSingle;

FAutoriz.MaximizeBox = false;

FAutoriz.StartPosition = FormStartPosition.CenterScreen;

(FAutoriz.Controls["LoginTextBox"] as TextBox).Text = MainForm.shifLogin;

if((FAutoriz.Controls["LoginTextBox"] as TextBox).Text != "")

{

FAutoriz.ActiveControl = (FAutoriz.Controls["PassTextBox"] as MaskedTextBox);

}

FAutoriz.Controls["EnterButton"].Click += new EventHandler(ButtonEnterClick);

FAutoriz.Controls["CancelButton"].Click += new EventHandler(ButtonCancelClick);

FAutoriz.ShowDialog();

}

//кнопка вход

private void ButtonEnterClick(object sender, EventArgs e)

{

DataTable DT = new DataTable();

DataRow DR;

DB.LogIn = false;

//Выгружаем список разрешений пользователя

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@User\_Nick", (FAutoriz.Controls["LoginTextBox"]as TextBox).Text),

new SqlParameter("@User\_Pass", (FAutoriz.Controls["PassTextBox"] as MaskedTextBox).Text),

};

DT = DB.GetData("Select\_Role\_Percon", SP);

//если нет ролей

if (DB.LastGet)

if (DT.Rows.Count == 0)

{

MessageBox.Show("У вас нет прав на просмотр и изменение каких либо данных.\nПожалуйста обратитесь к администратору для получения прав.");

DB.LogIn = false;

return;

}

else

{

Nick = (FAutoriz.Controls["LoginTextBox"] as TextBox).Text;

//запоминаем роли

if (DT.Rows.Count != 0)

{

DR = DT.Rows[0];

for (int i = 0; i < role.Length; i++)

{

role[i] = (bool)DR[i + 1];

}

DB.LogIn = true;

}

try

{

SP = new SqlParameter[]

{

new SqlParameter("@User\_Nick", Nick),

};

DT = DB.GetData("SELECT\_Profile\_Personal", SP);

id\_Worker = Convert.ToInt32(DT.Rows[0]["id\_worker"]);

}

catch

{

MessageBox.Show("Что то пошло не так.\nПопробуйте снова.", "Ошибка");

return;

}

//выгружаем специальность

SP = new SqlParameter[1]

{

new SqlParameter("@User\_Nick", (FAutoriz.Controls["LoginTextBox"]as TextBox).Text),

};

DT = DB.GetData("Select\_Spesialization\_For\_Personal", SP);

//запоминаем специальность

if (DT.Rows.Count != 0)

{

DR = DT.Rows[0];

Spec = DR[0].ToString();

}

}

if (DB.LogIn)

{

string Login = Shifrovanie.Shifrovanie.Encrypt(Nick, "qwerty555");

Regedit.Reestr.Write("LifeOfBionic", "login", Login);

FAutoriz.Close();

}

}

//кнопка отмена

private void ButtonCancelClick(object sender, EventArgs e)

{

FAutoriz.Close();

}

private void CreateLoginLabel()

{

lab = new Label()

{

Name = "LoginLabel",

Text = "Логин: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locAutLabel,

AutoSize = true,

};

FAutoriz.Controls.Add(lab);

}

private void CreateLoginTB()

{

Tb = new TextBox()

{

Name = "LoginTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locAutTB,

Size = sizeAutTB,

};

FAutoriz.Controls.Add(Tb);

}

private void CreatePassLabel()

{

lab = new Label()

{

Name = "PassLabel",

Text = "Пароль: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locPassLabel,

AutoSize = true,

};

FAutoriz.Controls.Add(lab);

}

private void CreatePassTB()

{

mtb = new MaskedTextBox()

{

Name = "PassTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locPassTB,

Size = sizePassTB,

};

mtb.UseSystemPasswordChar = true;

FAutoriz.Controls.Add(mtb);

}

private void CreateEnterButton()

{

but = new Button()

{

Name = "EnterButton",

Text = "Вход",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locEntBut,

Size = sizeEntBut,

};

FAutoriz.Controls.Add(but);

}

private void CreateCancelButton()

{

but = new Button()

{

Name = "CancelButton",

Text = "Отмена",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locCanBut,

Size = sizeCanBut,

};

FAutoriz.Controls.Add(but);

}

}

}

ConnectionForm.cs

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Sql;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.ComboBox;

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

namespace WindowsFormsApp9

{

class ConnectionForm

{

private static int cdt, cbd;

private static String[] CDT;

private static String[] CBD;

private Form FCon = new Form();

private Label lab = new Label();

private TextBox Tb = new TextBox();

private MaskedTextBox mtb = new MaskedTextBox();

private ComboBox cb = new ComboBox();

private Button but = new Button();

private Font DefFont = new Font("Microsoft Sans Serif", 8);

private Color DefBackColor = Color.FromName("Control");

private Color DefTextColor = Color.FromName("ControlText");

//Позиции //позиции

private Point locServLabel = new Point(20, 10); //label сервера

private Point locServCB = new Point(20, 33); //ComboBox сервера

private Point locLoginLabel = new Point(20, 63); //label Логина

private Point locLoginTB = new Point(20, 86); //TextBox Логина

private Point locPassLabel = new Point(20, 116); //label пароля

private Point locPassTB = new Point(20, 139); //TextBox пароля

private Point locBDLabel = new Point(20, 169); //label источника данных

private Point locBDCB = new Point(20, 192); //ComboBox источника данных

private Point locProvBut = new Point(20, 257); //кнопка Проверки

private Point locConnectBut = new Point(105, 257);//кнопка подключения

private Point locCanBut = new Point(205, 257); //кнопка отмены

private Point locRefBut = new Point(170, 8); //копка обновления

private Point locCleBut = new Point(20, 222); //копка обновления

//размеры //размеры

private Size sizeServCB = new Size(250, 20); //ComboBox сервера

private Size sizeLoginTB = new Size(250, 20); //TextBox Логина

private Size sizePassTB = new Size(250, 20); //TextBox пароля

private Size sizeBDCB = new Size(250, 20); //ComboBox источника

private Size sizeProvBut = new Size(75, 30); //кнопка Проверки

private Size sizeConnectBut = new Size(95, 30); //кнопка подключения

private Size sizeCanBut = new Size(65, 30); //кнопка отмены

private Size sizeRefBut = new Size(100, 20); //кнопка обновления

private Size sizeCleBut = new Size(250, 30); //кнопка обновления

public ConnectionForm()

{

}

public void NewConForm()

{

CreateServLabel();

CreateServCB();

CreateLoginLabel();

CreateLoginTB();

CreatePassLabel();

CreatePassTB();

CreateBDLabel();

CreateBDCB();

CreateConnectButton();

CreateProvButton();

CreateCancelButton();

CreateRefreshButton();

CreateClearButton();

FCon.FormClosed += new FormClosedEventHandler(FormClosing);

(FCon.Controls["RefreshButton"] as Button).Click += new EventHandler(ButtonRefreshClick);

(FCon.Controls["ProvButton"] as Button).Click += new EventHandler(ProvLogin);

(FCon.Controls["CancelButton"] as Button).Click += new EventHandler(ButtonCancelClick);

(FCon.Controls["ConnectButton"] as Button).Click += new EventHandler(ButtonConnectClick);

(FCon.Controls["ServCB"] as ComboBox).KeyPress += new KeyPressEventHandler(KeyPressProv);

(FCon.Controls["LoginTextBox"] as TextBox).KeyPress += new KeyPressEventHandler(KeyPressProv);

(FCon.Controls["PassTextBox"] as MaskedTextBox).KeyPress += new KeyPressEventHandler(KeyPressProv);

(FCon.Controls["ClearButton"] as Button).Click += new EventHandler(ClearButton);

Enab();

if (DataBaseConfiguration.connect)

{

(FCon.Controls["LoginTextBox"] as TextBox).Text = DataBaseConfiguration.cui;

(FCon.Controls["PassTextBox"] as MaskedTextBox).Text = DataBaseConfiguration.cpw;

try

{

for (int i = 0; i < CDT.Length; i++)

(FCon.Controls["ServCB"] as ComboBox).Items.Add(CDT[i]);

}

catch { }

(FCon.Controls["ServCB"] as ComboBox).Text = DataBaseConfiguration.cds;

try

{

for (int i = 0; i < CBD.Length; i++)

(FCon.Controls["BDCB"] as ComboBox).Items.Add(CBD[i]);

}

catch { }

(FCon.Controls["BDCB"] as ComboBox).Text = DataBaseConfiguration.baseName;

}

else { ButtonRefreshClick(new Button(),new EventArgs()); }

FCon.Text = "Подключение";

FCon.Size = new Size(305, 335);

FCon.FormBorderStyle = FormBorderStyle.FixedSingle;

FCon.MaximizeBox = false;

FCon.StartPosition = FormStartPosition.CenterScreen;

FCon.Show();

}

//включение выключение компонентов

private void Enab()

{

(FCon.Controls["RefreshButton"] as Button).Enabled = true;

(FCon.Controls["ServCB"] as ComboBox).Enabled = true;

(FCon.Controls["LoginTextBox"] as TextBox).Enabled = true;

(FCon.Controls["PassTextBox"] as MaskedTextBox).Enabled = true;

(FCon.Controls["BDCB"] as ComboBox).Enabled = true;

(FCon.Controls["ProvButton"] as Button).Enabled = true;

(FCon.Controls["ConnectButton"] as Button).Enabled = true;

if (DataBaseConfiguration.connect)

{

(FCon.Controls["RefreshButton"] as Button).Enabled = false;

(FCon.Controls["ServCB"] as ComboBox).Enabled = false;

(FCon.Controls["LoginTextBox"] as TextBox).Enabled = false;

(FCon.Controls["PassTextBox"] as MaskedTextBox).Enabled = false;

(FCon.Controls["ProvButton"] as Button).Enabled = false;

}

else

{

(FCon.Controls["BDCB"] as ComboBox).Text = "";

(FCon.Controls["BDCB"] as ComboBox).Enabled = false;

(FCon.Controls["ConnectButton"] as Button).Enabled = false;

}

if ((FCon.Controls["ServCB"] as ComboBox).Text == "Поиск серверов...")

{

(FCon.Controls["ServCB"] as ComboBox).Enabled = false;

(FCon.Controls["BDCB"] as ComboBox).Enabled = false;

(FCon.Controls["ConnectButton"] as Button).Enabled = false;

}

if ((FCon.Controls["BDCB"] as ComboBox).Text == "Поиск...")

{

(FCon.Controls["BDCB"] as ComboBox).Enabled = false;

}

}

//закрытие формы

private void FormClosing(object sender, EventArgs e)

{

Application.OpenForms[0].Enabled = true;

AutorizForm AF = new AutorizForm();

AF.NewAutorizForm();

}

//кнопка сброса реестра

private void ClearButton(object sender, EventArgs e)

{

try

{

Regedit.Reestr.DelKey("LifeOfBionic");

}

catch { }

}

//кнопка проверки

private void ProvLogin(object sender, EventArgs e)

{

DataBaseConfiguration.connect = true;

Enab();

(FCon.Controls["ConnectButton"] as Button).Enabled = false;

ComboBox CB = (FCon.Controls["BDCB"] as ComboBox);

string cds = (FCon.Controls["ServCB"] as ComboBox).Text;

string cui = (FCon.Controls["LoginTextBox"] as TextBox).Text;

string cpw = (FCon.Controls["PassTextBox"] as MaskedTextBox).Text;

CB.Enabled = false;

CB.Text = "Поиск...";

Thread t = new Thread(() => DataBaseConfiguration.SearchBases(FCon, CB, (FCon.Controls["ConnectButton"] as Button), cds, cui, cpw));

t.Start();

}

//обновление списка серверов

private void ButtonRefreshClick(object sender, EventArgs e)

{

(FCon.Controls["ProvButton"] as Button).Enabled = false;

ComboBox CB = (FCon.Controls["ServCB"] as ComboBox);

CB.Text = "Поиск серверов...";

CB.Enabled = false;

Thread t = new Thread(() => DataBaseConfiguration.searchServers(FCon, CB, (FCon.Controls["ProvButton"] as Button)));

t.Start();

}

//кнопка отмены

private void ButtonCancelClick(object sender, EventArgs e)

{

if (!DataBaseConfiguration.connect)

{

FCon.Close();

}

else

{

DataBaseConfiguration.connect = false;

Enab();

}

}

//кнопка подключиться

private void ButtonConnectClick(object sender, EventArgs e)

{

(FCon.Controls["BDCB"] as ComboBox).Enabled = false;

DataBaseConfiguration.cds = (FCon.Controls["ServCB"] as ComboBox).Text;

DataBaseConfiguration.cui = (FCon.Controls["LoginTextBox"] as TextBox).Text;

DataBaseConfiguration.cpw = (FCon.Controls["PassTextBox"] as MaskedTextBox).Text;

DataBaseConfiguration.baseName = (FCon.Controls["BDCB"] as ComboBox).Text;

CDT = new string[(FCon.Controls["ServCB"] as ComboBox).Items.Count];

CBD = new string[(FCon.Controls["BDCB"] as ComboBox).Items.Count];

for(int i = 0; i< (FCon.Controls["ServCB"] as ComboBox).Items.Count;i++)

CDT[i] = (FCon.Controls["ServCB"] as ComboBox).Items[i].ToString();

cdt = (FCon.Controls["ServCB"] as ComboBox).SelectedIndex;

for (int i = 0; i < (FCon.Controls["BDCB"] as ComboBox).Items.Count; i++)

CBD[i] = (FCon.Controls["BDCB"] as ComboBox).Items[i].ToString();

cbd = (FCon.Controls["BDCB"] as ComboBox).SelectedIndex;

MainForm.Shifr();

FCon.Close();

}

//Esc и Enter

private void KeyPressProv(object sender, KeyPressEventArgs e)

{

switch (e.KeyChar)

{

case (char)13:

{

if ((FCon.Controls["ProvButton"] as Button).Enabled)

ProvLogin(new Button(), new EventArgs());

break;

}

case (char)27:

{

if ((FCon.Controls["CancelButton"] as Button).Enabled)

ButtonCancelClick(new Button(), new EventArgs());

break;

}

}

}

private void CreateServLabel()

{

lab = new Label()

{

Name = "ServLabel",

Text = "Сервер: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locServLabel,

AutoSize = true,

};

FCon.Controls.Add(lab);

}

private void CreateServCB()

{

cb = new ComboBox()

{

Name = "ServCB",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locServCB,

Size = sizeServCB,

};

FCon.Controls.Add(cb);

}

private void CreateLoginLabel()

{

lab = new Label()

{

Name = "LoginLabel",

Text = "Логин сервера: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locLoginLabel,

AutoSize = true,

};

FCon.Controls.Add(lab);

}

private void CreateLoginTB()

{

Tb = new TextBox()

{

Name = "LoginTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locLoginTB,

Size = sizeLoginTB,

};

FCon.Controls.Add(Tb);

}

private void CreatePassLabel()

{

lab = new Label()

{

Name = "PassLabel",

Text = "Пароль сервера: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locPassLabel,

AutoSize = true,

};

FCon.Controls.Add(lab);

}

private void CreatePassTB()

{

mtb = new MaskedTextBox()

{

Name = "PassTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locPassTB,

Size = sizePassTB,

};

mtb.UseSystemPasswordChar = true;

FCon.Controls.Add(mtb);

}

private void CreateBDLabel()

{

lab = new Label()

{

Name = "BDLabel",

Text = "Источник данных: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locBDLabel,

AutoSize = true,

};

FCon.Controls.Add(lab);

}

private void CreateBDCB()

{

cb = new ComboBox()

{

Name = "BDCB",

Text = "",

Font = DefFont,

Visible = true,

Enabled = false,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locBDCB,

Size = sizeBDCB,

AutoCompleteMode = AutoCompleteMode.SuggestAppend,

AutoCompleteSource = AutoCompleteSource.ListItems,

};

FCon.Controls.Add(cb);

}

private void CreateConnectButton()

{

but = new Button()

{

Name = "ConnectButton",

Text = "Подключиться",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locConnectBut,

Size = sizeConnectBut,

};

FCon.Controls.Add(but);

}

private void CreateProvButton()

{

but = new Button()

{

Name = "ProvButton",

Text = "Проверка",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locProvBut,

Size = sizeProvBut,

Enabled = false,

};

FCon.Controls.Add(but);

}

private void CreateCancelButton()

{

but = new Button()

{

Name = "CancelButton",

Text = "Отмена",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locCanBut,

Size = sizeCanBut,

};

FCon.Controls.Add(but);

}

private void CreateRefreshButton()

{

but = new Button()

{

Name = "RefreshButton",

Text = "Обновить",

Font = new Font("Microsoft Sans Serif", 7),

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locRefBut,

Size = sizeRefBut,

};

FCon.Controls.Add(but);

}

private void CreateClearButton()

{

but = new Button()

{

Name = "ClearButton",

Text = "Забыть данные о подключении",

Font = new Font("Microsoft Sans Serif", 9),

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locCleBut,

Size = sizeCleBut,

};

FCon.Controls.Add(but);

}

}

}

CreateWriteForm.cs

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp9

{

class CreateWriteForm

{

public bool FormEnab = true;

public bool citEnab = true;

public string id\_cit = "";

public string NameSit = "";

public string SurSit = "";

public string PatSit = "";

public string snils = "";

public string SeriesPass = "";

public string NumbPass = "";

public DateTime DateB = DateTime.Now;

private DataTable DateRecord = new DataTable();

private DataTable DataCit = new DataTable();

private DataTable Type = new DataTable();

private Form FNewWrite = new Form();

private Label lab = new Label();

private TextBox Tb = new TextBox();

private MaskedTextBox mtb = new MaskedTextBox();

private Button but = new Button();

private ComboBox cb = new ComboBox();

private DateTimePicker DTP = new DateTimePicker();

private Font DefFont = new Font("Microsoft Sans Serif", 10);

private Color DefBackColor = Color.FromName("Control");

private Color DefTextColor = Color.FromName("ControlText");

//Позиции //позиции

private Point locCitizenLabel = new Point(100, 10);//label граждана

private Point locCitizenCB = new Point(100, 33); //ComboBox граждана

private Point locSurLabel = new Point(20, 63); //label фамилии

private Point locSurTB = new Point(20, 86); //textBox фамилии

private Point locNameLabel = new Point(190, 63); //label имени

private Point locNameTB = new Point(190, 86); //textBox имени

private Point locPatLabel = new Point(20, 119); //label отчества

private Point locPatTB = new Point(20, 149); //textBox отчества

private Point locDateBLabel = new Point(190, 119); //label даты рождения

private Point locDateBTB = new Point(190, 149); //textBox даты рождения

private Point locSNILSLabel = new Point(20, 179); //label снилс

private Point locSNILSTB = new Point(20, 202); //textBox снилс

private Point locSeriesLabel = new Point(190, 179);//label серии пасспорта

private Point locSeriesTB = new Point(190, 202); //textBox серии пасспорта

private Point locNumberLabel = new Point(20, 232); //label номера пасспорта

private Point locNumberTB = new Point(20, 255); //textBox номера пасспорта

private Point locAddBut = new Point(20, 300); //кнопка добавить

private Point locExtBut = new Point(200, 300); //кнопка выход

private Point locFormLabel = new Point(485, 20); //label существующих форм

private Point locFormCB = new Point(485, 43); //comboBox существующих форм

private Point locTypeLabel = new Point(400, 73); //label типа записи

private Point locTypeCB = new Point(400, 96); //comboBox типа записи

private Point locMailLabel = new Point(570, 126); //label почты

private Point locMailTB = new Point(570, 149); //comboBox почты

private Point locPNumberLabel = new Point(400, 126);//label Телефона

private Point locPNumberTB = new Point(400, 149); //comboBox Телефона

private Point locSiteLabel = new Point(570, 179); //label Сайта

private Point locSiteTB = new Point(570, 202); //comboBox Сайта

private Point locAddressLabel = new Point(400, 179);//label Адреса

private Point locAddressTB = new Point(400, 202); //comboBox Адреса

private Point locDateLabel = new Point(400, 232); //label Время записи

private Point locDateTB = new Point(400, 255); //DateTimePicker Время записи

private Point locDateCB = new Point(510, 255); //comboBox Время записи

private Point locFormBut = new Point(485, 250); //кнопка добавить форму

//размеры //размеры

private Size sizeCitizenCB = new Size(150, 20); //comboBox существующих форм

private Size sizeSurTB = new Size(150, 20); //textBox фамилии

private Size sizeNameTB = new Size(150, 20); //textBox имени

private Size sizePatTB = new Size(150, 20); //textBox отчества

private Size sizeDateBTB = new Size(150, 20); //textBox даты рождения

private Size sizeSNILSTB = new Size(150, 20); //textBox снилс

private Size sizeSeriesTB = new Size(150, 20); //textBox серии паспорта

private Size sizeNumberTB = new Size(150, 20); //textBox номера пасспорта

private Size sizeAddBut = new Size(150, 30); //кнопка добавить

private Size sizeExtBut = new Size(100, 30); //кнопка выход

private Size sizeFormCB = new Size(150, 20); //comboBox существующих форм

private Size sizeTypeCB = new Size(150, 20); //comboBox типа записи

private Size sizeMailTB = new Size(150, 20); //textBox почты

private Size sizePNumberTB = new Size(150, 20); //textBox Телефона

private Size sizeSiteTB = new Size(150, 20); //textBox Сайта

private Size sizeAddressTB = new Size(150, 20); //textBox Адреса

private Size sizeDateTB = new Size(100, 20); //DateTimePicker времени записи

private Size sizeDateCB = new Size(60, 20); //ComboBox времени записи

private Size sizeFormBut = new Size(150, 30); //кнопка добавить форму

public CreateWriteForm()

{

}

public void NewWrite()

{

CreateCitizenLabel();

CreateSurLabel();

CreateSurTB();

CreateNameLabel();

CreateNameTB();

CreatePatLabel();

CreatePatTB();

CreateDateBLabel();

CreateDateBTB();

CreateSNILSLabel();

CreateSNILSBTB();

CreateSeriesLabel();

CreateSeriesTB();

CreateNumLabel();

CreateNumTB();

CreateAddButton();

CreateExtButton();

//CreateFormLabel();

//CreateFormCB();

CreateTypeLabel();

CreateMailLabel();

CreateMailTB();

CreatePNumberLabel();

CreatePNumberTB();

CreateSiteLabel();

CreateSiteTB();

CreateAddressLabel();

CreateAddressTB();

CreateDateLabel();

CreateDateTB();

CreateDateCB();

//CreateFormButton();

CreateCitizenCB();

CreateTypeCB();

if (FormEnab)

{

FNewWrite.Text = "Добавление записи";

FNewWrite.Controls["AddButton"].Text = "Добавить";

FNewWrite.Controls["AddButton"].Click += AddButtonClick;

}

else

{

FNewWrite.Text = "Редактирование записи";

FNewWrite.Controls["AddButton"].Text = "Редактировать";

FNewWrite.Controls["AddButton"].Click += UpdButtonClick;

FNewWrite.Controls["AddButton"].Enabled = true;

}

FNewWrite.Size = new Size(760, 400);

FNewWrite.FormBorderStyle = FormBorderStyle.FixedSingle;

FNewWrite.MaximizeBox = false;

FNewWrite.StartPosition = FormStartPosition.CenterScreen;

FNewWrite.ShowDialog();

}

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

{

if (FormEnab)

switch ((sender as ComboBox).SelectedItem)

{

case "По телефону":

{

(FNewWrite.Controls["PNumberTextBox"] as MaskedTextBox).Enabled = true;

(FNewWrite.Controls["AddressTextBox"] as TextBox).Enabled = false;

(FNewWrite.Controls["SiteTextBox"] as TextBox).Enabled = false;

(FNewWrite.Controls["MailTextBox"] as MaskedTextBox).Enabled = false;

break;

}

case "Через интернет":

{

(FNewWrite.Controls["SiteTextBox"] as TextBox).Enabled = true;

(FNewWrite.Controls["AddressTextBox"] as TextBox).Enabled = false;

(FNewWrite.Controls["PNumberTextBox"] as MaskedTextBox).Enabled = false;

(FNewWrite.Controls["MailTextBox"] as MaskedTextBox).Enabled = false;

break;

}

case "По почте":

{

(FNewWrite.Controls["MailTextBox"] as MaskedTextBox).Enabled = true;

(FNewWrite.Controls["AddressTextBox"] as TextBox).Enabled = false;

(FNewWrite.Controls["PNumberTextBox"] as MaskedTextBox).Enabled = false;

(FNewWrite.Controls["SiteTextBox"] as TextBox).Enabled = false;

break;

}

case "Через больницу":

{

(FNewWrite.Controls["AddressTextBox"] as TextBox).Enabled = true;

(FNewWrite.Controls["PNumberTextBox"] as MaskedTextBox).Enabled = false;

(FNewWrite.Controls["SiteTextBox"] as TextBox).Enabled = false;

(FNewWrite.Controls["MailTextBox"] as MaskedTextBox).Enabled = false;

break;

}

}

}

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

{

if ((sender as ComboBox).SelectedItem != "Добавить нового")

{

if (FormEnab)

{

(FNewWrite.Controls["NameTextBox"] as TextBox).Enabled = false;

(FNewWrite.Controls["SurTextBox"] as TextBox).Enabled = false;

(FNewWrite.Controls["PatTextBox"] as TextBox).Enabled = false;

(FNewWrite.Controls["SNILSTextBox"] as TextBox).Enabled = false;

(FNewWrite.Controls["SeriesTextBox"] as TextBox).Enabled = false;

(FNewWrite.Controls["NumberTextBox"] as TextBox).Enabled = false;

(FNewWrite.Controls["DateBPicker"] as DateTimePicker).Enabled = false;

}

DataRow[] DRCit = DataCit.Select("[ФИО] = '" + (sender as ComboBox).Text + "'");

SqlParameter[] SP = new SqlParameter[0];

DataTable DT = DB.GetData("Select\_DataCitizen", SP);

int id = Convert.ToInt32(DRCit[0]["id\_Citizen"]);

DataRow[] DRAllCit = DT.Select("[id\_Citizen] = '" + id.ToString() + "'");

string[] FIOCit = DRAllCit[0]["ФИО"].ToString().Split(new char[] { ' ' });

string[] Pass = DRAllCit[0]["Серия и номер"].ToString().Split(new char[] { ' ', 'и' });

(FNewWrite.Controls["SurTextBox"] as TextBox).Text = FIOCit[0];

(FNewWrite.Controls["NameTextBox"] as TextBox).Text = FIOCit[1];

(FNewWrite.Controls["PatTextBox"] as TextBox).Text = FIOCit[2];

(FNewWrite.Controls["SNILSTextBox"] as TextBox).Text = DRAllCit[0]["Снилс"].ToString();

(FNewWrite.Controls["SeriesTextBox"] as TextBox).Text = Pass[0];

(FNewWrite.Controls["NumberTextBox"] as TextBox).Text = Pass[1];

CultureInfo provider = CultureInfo.InvariantCulture;

(FNewWrite.Controls["DateBPicker"] as DateTimePicker).Value = DateTime.ParseExact(DRAllCit[0]["День рождения"].ToString().Substring(0, 10), "dd.MM.yyyy", provider);

}

else

{

(FNewWrite.Controls["NameTextBox"] as TextBox).Enabled = true;

(FNewWrite.Controls["SurTextBox"] as TextBox).Enabled = true;

(FNewWrite.Controls["PatTextBox"] as TextBox).Enabled = true;

(FNewWrite.Controls["SNILSTextBox"] as TextBox).Enabled = true;

(FNewWrite.Controls["SeriesTextBox"] as TextBox).Enabled = true;

(FNewWrite.Controls["NumberTextBox"] as TextBox).Enabled = true;

(FNewWrite.Controls["DateBPicker"] as DateTimePicker).Enabled = true;

}

}

//конпка выхода

private void ExitButtonClick(object sender, EventArgs e)

{

FNewWrite.Close();

}

//добавить запись

private void AddButtonClick(object sender, EventArgs e)

{

//добавляем форму записи

int id\_FormWrite = 0;

DataRow[] DRType = Type.Select("[Тип записи] = '" + (FNewWrite.Controls["FormCB"] as ComboBox).Text + "'");

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@id\_worker", AutorizForm.id\_Worker),

new SqlParameter("@id\_TypeWrite ", DRType[0]["id\_TypeWrite"]),

new SqlParameter("@mail", (FNewWrite.Controls["MailTextBox"] as MaskedTextBox).Text),

new SqlParameter("@PhoneNumber", (FNewWrite.Controls["PNumberTextBox"] as MaskedTextBox).Text),

new SqlParameter("@Sites", (FNewWrite.Controls["SiteTextBox"] as TextBox).Text),

new SqlParameter("@Adress", (FNewWrite.Controls["AddressTextBox"] as TextBox).Text),

};

switch (DRType[0]["Тип записи"])

{

case "По почте":

{

SP[3].Value = DBNull.Value;

SP[4].Value = DBNull.Value;

SP[5].Value = DBNull.Value;

break;

}

case "По телефону":

{

SP[2].Value = DBNull.Value;

SP[4].Value = DBNull.Value;

SP[5].Value = DBNull.Value;

break;

}

case "Через больницу":

{

SP[2].Value = DBNull.Value;

SP[3].Value = DBNull.Value;

SP[4].Value = DBNull.Value;

break;

}

case "Через интернет":

{

SP[2].Value = DBNull.Value;

SP[3].Value = DBNull.Value;

SP[5].Value = DBNull.Value;

break;

}

}

id\_FormWrite = DB.WriteData("[Add\_FormWrite]", SP);

//добавляем граждана

int id\_Citizen = 0;

if (DB.LastWrite)

if (((FNewWrite.Controls["CitizenCB"] as ComboBox).SelectedItem == "Добавить нового") | ((FNewWrite.Controls["CitizenCB"] as ComboBox).SelectedItem == ""))

{

DateTime res = (FNewWrite.Controls["DateBPicker"] as DateTimePicker).Value;

SP = new SqlParameter[7]

{

new SqlParameter("@NameCit", (FNewWrite.Controls["NameTextBox"] as TextBox).Text),

new SqlParameter("@SurnameCit", (FNewWrite.Controls["SurTextBox"] as TextBox).Text),

new SqlParameter("@PatronymicCit", (FNewWrite.Controls["PatTextBox"] as TextBox).Text),

new SqlParameter("@Snils", (FNewWrite.Controls["SNILSTextBox"] as TextBox).Text),

new SqlParameter("@SeriesPassportCit", (FNewWrite.Controls["SeriesTextBox"] as TextBox).Text),

new SqlParameter("@NumberPassportCit", (FNewWrite.Controls["NumberTextBox"] as TextBox).Text),

new SqlParameter("@DateBirthCit", res),

};

id\_Citizen = DB.WriteData("Add\_DataCitizen", SP);

UpdBox(1, (FNewWrite.Controls["CitizenCB"] as ComboBox));

string ss = (FNewWrite.Controls["SurTextBox"] as TextBox).Text + " " + (FNewWrite.Controls["NameTextBox"] as TextBox).Text + " " + (FNewWrite.Controls["PatTextBox"] as TextBox).Text;

//MessageBox.Show(ss);

(FNewWrite.Controls["CitizenCB"] as ComboBox).SelectedItem = ss;

}

else

{

DataRow[] DRCit = DataCit.Select("[ФИО] = '" + (FNewWrite.Controls["CitizenCB"] as ComboBox).Text + "'");

id\_Citizen = Convert.ToInt32(DRCit[0]["id\_Citizen"]);

}

//добавляем запись

DataRow[] DRDR = DateRecord.Select("[Record\_Time] = '" + (FNewWrite.Controls["DateCB"] as ComboBox).Text + "'");

SP = new SqlParameter[]

{

new SqlParameter("@times", (FNewWrite.Controls["DatePicker"] as DateTimePicker).Value.Date),

new SqlParameter("@id\_Day\_of\_the\_week", DRDR[0]["id\_Day\_of\_the\_week"]),

new SqlParameter("@id\_Citizen", id\_Citizen),

new SqlParameter("@id\_FormWrite", id\_FormWrite),

//new SqlParameter("@id\_TypeWrite", ),

//new SqlParameter("@mail", ),

//new SqlParameter("@PhoneNumber", ),

//new SqlParameter("@Sites", ),

//new SqlParameter("@Adress", ),

};

if (DB.LastWrite)

DB.WriteData("[Add\_WriteAppointment]", SP);

if (DB.LastWrite)

FNewWrite.Close();

}

//Изменить запись

private void UpdButtonClick(object sender, EventArgs e)

{

DateTime res = (FNewWrite.Controls["DateBPicker"] as DateTimePicker).Value;

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@id\_Citizen", id\_cit),

new SqlParameter("@NameCit", (FNewWrite.Controls["NameTextBox"] as TextBox).Text),

new SqlParameter("@SurnameCit", (FNewWrite.Controls["SurTextBox"] as TextBox).Text),

new SqlParameter("@PatronymicCit", (FNewWrite.Controls["PatTextBox"] as TextBox).Text),

new SqlParameter("@Snils", (FNewWrite.Controls["SNILSTextBox"] as TextBox).Text),

new SqlParameter("@SeriesPassportCit", (FNewWrite.Controls["SeriesTextBox"] as TextBox).Text),

new SqlParameter("@NumberPassportCit", (FNewWrite.Controls["NumberTextBox"] as TextBox).Text),

new SqlParameter("@DateBirthCit", res),

};

DB.WriteData("[UPD\_DataCitizen]", SP);

if (DB.LastUpd)

FNewWrite.Close();

}

private void DatePickerValueChanged(object sender, EventArgs e)

{

UpdBox(2,FNewWrite.Controls["DateCB"] as ComboBox);

}

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

{

try

{

if ((sender as TextBox).Text != "")

{

(FNewWrite.Controls["AddButton"] as Button).Enabled = true;

}

else

{

(FNewWrite.Controls["AddButton"] as Button).Enabled = false;

}

}

catch { }

try

{

if ((sender as MaskedTextBox).Text != "")

{

(FNewWrite.Controls["AddButton"] as Button).Enabled = true;

}

else

{

(FNewWrite.Controls["AddButton"] as Button).Enabled = false;

}

}

catch { }

}

private void UpdBox(int n, ComboBox cb)

{

switch (n)

{

case 1:

{

SqlParameter[] SP = new SqlParameter[0];

DataCit = DB.GetData("Select\_DataCitizen\_ONLI\_FIO", SP);

DataRow RW;

cb.Items.Clear();

cb.Text = "";

cb.Items.Add("Добавить нового");

for (int i = 0; i < DataCit.Rows.Count; i++)

{

RW = DataCit.Rows[i];

cb.Items.Add(RW["ФИО"]);

}

cb.SelectedIndex = 0;

break;

}

case 2:

{

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@Date", (FNewWrite.Controls["DatePicker"] as DateTimePicker).Value.Date),

};

DateRecord = DB.GetData("Select\_Date\_Appointment", SP);

DataRow RW;

for (int i = 0; i < DateRecord.Rows.Count; i++)

{

DateRecord.Rows[i]["Record\_Time"] = DateRecord.Rows[i]["Record\_Time"].ToString().Substring(0, 5);

}

cb.Items.Clear();

cb.Text = "";

for (int i = 0; i < DateRecord.Rows.Count; i++)

{

RW = DateRecord.Rows[i];

cb.Items.Add(RW["Record\_Time"]);

}

cb.SelectedIndex = 0;

break;

}

}

}

private void CreateCitizenLabel()

{

lab = new Label()

{

Name = "CitizenLabel",

Text = "Граждан: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locCitizenLabel,

AutoSize = true,

Enabled = citEnab,

};

FNewWrite.Controls.Add(lab);

}

private void CreateCitizenCB()

{

cb = new ComboBox()

{

Name = "CitizenCB",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locCitizenCB,

Size = sizeCitizenCB,

AutoCompleteMode = AutoCompleteMode.SuggestAppend,

AutoCompleteSource = AutoCompleteSource.ListItems,

Enabled = FormEnab,

};

cb.SelectedIndexChanged += Citizen\_SelectedIndexChanged;

UpdBox(1, cb);

if (NameSit != "")

{

DataRow[] DRSit = DataCit.Select("[ФИО] LIKE '%" + SurSit + " " + NameSit + " " + PatSit + "%'");

cb.SelectedItem = DRSit[0]["ФИО"].ToString();

}

FNewWrite.Controls.Add(cb);

}

private void CreateSurLabel()

{

lab = new Label()

{

Name = "SurLabel",

Text = "Фамилия: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locSurLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreateSurTB()

{

Tb = new TextBox()

{

Name = "SurTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locSurTB,

Size = sizeSurTB,

MaxLength = 50,

};

FNewWrite.Controls.Add(Tb);

}

private void CreateNameLabel()

{

lab = new Label()

{

Name = "NameLabel",

Text = "Имя: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locNameLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreateNameTB()

{

Tb = new TextBox()

{

Name = "NameTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locNameTB,

Size = sizeNameTB,

MaxLength = 50,

};

FNewWrite.Controls.Add(Tb);

}

private void CreatePatLabel()

{

lab = new Label()

{

Name = "PatLabel",

Text = "Отчество: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locPatLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreatePatTB()

{

Tb = new TextBox()

{

Name = "PatTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locPatTB,

Size = sizePatTB,

MaxLength = 50,

};

FNewWrite.Controls.Add(Tb);

}

private void CreateDateBLabel()

{

lab = new Label()

{

Name = "DateBLabel",

Text = "Дата рождения: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locDateBLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreateDateBTB()

{

DTP = new DateTimePicker()

{

Name = "DateBPicker",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locDateBTB,

Size = sizeDateBTB,

Format = DateTimePickerFormat.Custom,

CustomFormat = "dd.MM.yyyy",

Value = new DateTime(2000, 1, 1),

};

FNewWrite.Controls.Add(DTP);

}

private void CreateSNILSLabel()

{

lab = new Label()

{

Name = "SNILSLabel",

Text = "Снилс: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locSNILSLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreateSNILSBTB()

{

Tb = new TextBox()

{

Name = "SNILSTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locSNILSTB,

Size = sizeSNILSTB,

MaxLength = 11,

};

FNewWrite.Controls.Add(Tb);

}

private void CreateSeriesLabel()

{

lab = new Label()

{

Name = "SeriesLabel",

Text = "Серия пасспорта: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locSeriesLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreateSeriesTB()

{

Tb = new TextBox()

{

Name = "SeriesTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locSeriesTB,

Size = sizeSeriesTB,

MaxLength = 4,

};

FNewWrite.Controls.Add(Tb);

}

private void CreateNumLabel()

{

lab = new Label()

{

Name = "NumberLabel",

Text = "Номер пасспорта: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locNumberLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreateNumTB()

{

Tb = new TextBox()

{

Name = "NumberTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locNumberTB,

Size = sizeNumberTB,

MaxLength = 6,

};

FNewWrite.Controls.Add(Tb);

}

private void CreateAddButton()

{

but = new Button()

{

Name = "AddButton",

Text = "Добавить",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locAddBut,

Size = sizeAddBut,

Enabled = false,

};

FNewWrite.Controls.Add(but);

}

private void CreateExtButton()

{

but = new Button()

{

Name = "ExtButton",

Text = "Выйти",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locExtBut,

Size = sizeExtBut,

};

but.Click += ExitButtonClick;

FNewWrite.Controls.Add(but);

}

private void CreateTypeLabel()

{

lab = new Label()

{

Name = "TypeLabel",

Text = "Тип записи",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locTypeLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreateTypeCB()

{

cb = new ComboBox()

{

Name = "FormCB",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locTypeCB,

Size = sizeTypeCB,

AutoCompleteMode = AutoCompleteMode.SuggestAppend,

AutoCompleteSource = AutoCompleteSource.ListItems,

Enabled = FormEnab,

};

cb.SelectedIndexChanged += TypeCB\_SelectedIndexChanged;

SqlParameter[] SP = new SqlParameter[0];

Type = DB.GetData("Celect\_TypeWrite", SP);

DataRow RW;

cb.Items.Clear();

cb.Text = "";

for (int i = 0; i < Type.Rows.Count; i++)

{

RW = Type.Rows[i];

cb.Items.Add(RW["Тип записи"]);

}

cb.SelectedIndex = 0;

FNewWrite.Controls.Add(cb);

}

private void CreateMailLabel()

{

lab = new Label()

{

Name = "MailLabel",

Text = "Почта: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locMailLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreateMailTB()

{

mtb = new MaskedTextBox()

{

Name = "MailTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locMailTB,

Size = sizeMailTB,

MaxLength = 50,

Enabled = false,

};

mtb.TextChanged += Form\_TextChanged;

FNewWrite.Controls.Add(mtb);

}

private void CreatePNumberLabel()

{

lab = new Label()

{

Name = "PNumberLabel",

Text = "Номер телефона: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locPNumberLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreatePNumberTB()

{

mtb = new MaskedTextBox()

{

Name = "PNumberTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locPNumberTB,

Size = sizePNumberTB,

Enabled = false,

};

FNewWrite.Controls.Add(mtb);

}

private void CreateSiteLabel()

{

lab = new Label()

{

Name = "SiteLabel",

Text = "Сайт: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locSiteLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreateSiteTB()

{

Tb = new TextBox()

{

Name = "SiteTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locSiteTB,

Size = sizeSiteTB,

MaxLength = 100,

Enabled = false,

};

FNewWrite.Controls.Add(Tb);

}

private void CreateAddressLabel()

{

lab = new Label()

{

Name = "AddressLabel",

Text = "Адрес: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locAddressLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreateAddressTB()

{

Tb = new TextBox()

{

Name = "AddressTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locAddressTB,

Size = sizeAddressTB,

MaxLength = 100,

Enabled = false,

};

FNewWrite.Controls.Add(Tb);

}

private void CreateDateLabel()

{

lab = new Label()

{

Name = "DateLabel",

Text = "Время записи: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locDateLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreateDateTB()

{

DTP = new DateTimePicker()

{

Name = "DatePicker",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locDateTB,

Size = sizeDateTB,

Format = DateTimePickerFormat.Custom,

CustomFormat = "dd.MM.yyyy",

Enabled = FormEnab,

};

DTP.ValueChanged += new EventHandler(DatePickerValueChanged);

FNewWrite.Controls.Add(DTP);

}

private void CreateDateCB()

{

cb = new ComboBox()

{

Name = "DateCB",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locDateCB,

Size = sizeDateCB,

AutoCompleteMode = AutoCompleteMode.SuggestAppend,

AutoCompleteSource = AutoCompleteSource.ListItems,

DropDownStyle = ComboBoxStyle.DropDownList,

Enabled = FormEnab,

};

UpdBox(2,cb);

FNewWrite.Controls.Add(cb);

}

}

}

DataBaseConfiguration.cs

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp9

{

class CreateWriteForm

{

public bool FormEnab = true;

public bool citEnab = true;

public string id\_cit = "";

public string NameSit = "";

public string SurSit = "";

public string PatSit = "";

public string snils = "";

public string SeriesPass = "";

public string NumbPass = "";

public DateTime DateB = DateTime.Now;

private DataTable DateRecord = new DataTable();

private DataTable DataCit = new DataTable();

private DataTable Type = new DataTable();

private Form FNewWrite = new Form();

private Label lab = new Label();

private TextBox Tb = new TextBox();

private MaskedTextBox mtb = new MaskedTextBox();

private Button but = new Button();

private ComboBox cb = new ComboBox();

private DateTimePicker DTP = new DateTimePicker();

private Font DefFont = new Font("Microsoft Sans Serif", 10);

private Color DefBackColor = Color.FromName("Control");

private Color DefTextColor = Color.FromName("ControlText");

//Позиции //позиции

private Point locCitizenLabel = new Point(100, 10);//label граждана

private Point locCitizenCB = new Point(100, 33); //ComboBox граждана

private Point locSurLabel = new Point(20, 63); //label фамилии

private Point locSurTB = new Point(20, 86); //textBox фамилии

private Point locNameLabel = new Point(190, 63); //label имени

private Point locNameTB = new Point(190, 86); //textBox имени

private Point locPatLabel = new Point(20, 119); //label отчества

private Point locPatTB = new Point(20, 149); //textBox отчества

private Point locDateBLabel = new Point(190, 119); //label даты рождения

private Point locDateBTB = new Point(190, 149); //textBox даты рождения

private Point locSNILSLabel = new Point(20, 179); //label снилс

private Point locSNILSTB = new Point(20, 202); //textBox снилс

private Point locSeriesLabel = new Point(190, 179);//label серии пасспорта

private Point locSeriesTB = new Point(190, 202); //textBox серии пасспорта

private Point locNumberLabel = new Point(20, 232); //label номера пасспорта

private Point locNumberTB = new Point(20, 255); //textBox номера пасспорта

private Point locAddBut = new Point(20, 300); //кнопка добавить

private Point locExtBut = new Point(200, 300); //кнопка выход

private Point locFormLabel = new Point(485, 20); //label существующих форм

private Point locFormCB = new Point(485, 43); //comboBox существующих форм

private Point locTypeLabel = new Point(400, 73); //label типа записи

private Point locTypeCB = new Point(400, 96); //comboBox типа записи

private Point locMailLabel = new Point(570, 126); //label почты

private Point locMailTB = new Point(570, 149); //comboBox почты

private Point locPNumberLabel = new Point(400, 126);//label Телефона

private Point locPNumberTB = new Point(400, 149); //comboBox Телефона

private Point locSiteLabel = new Point(570, 179); //label Сайта

private Point locSiteTB = new Point(570, 202); //comboBox Сайта

private Point locAddressLabel = new Point(400, 179);//label Адреса

private Point locAddressTB = new Point(400, 202); //comboBox Адреса

private Point locDateLabel = new Point(400, 232); //label Время записи

private Point locDateTB = new Point(400, 255); //DateTimePicker Время записи

private Point locDateCB = new Point(510, 255); //comboBox Время записи

private Point locFormBut = new Point(485, 250); //кнопка добавить форму

//размеры //размеры

private Size sizeCitizenCB = new Size(150, 20); //comboBox существующих форм

private Size sizeSurTB = new Size(150, 20); //textBox фамилии

private Size sizeNameTB = new Size(150, 20); //textBox имени

private Size sizePatTB = new Size(150, 20); //textBox отчества

private Size sizeDateBTB = new Size(150, 20); //textBox даты рождения

private Size sizeSNILSTB = new Size(150, 20); //textBox снилс

private Size sizeSeriesTB = new Size(150, 20); //textBox серии паспорта

private Size sizeNumberTB = new Size(150, 20); //textBox номера пасспорта

private Size sizeAddBut = new Size(150, 30); //кнопка добавить

private Size sizeExtBut = new Size(100, 30); //кнопка выход

private Size sizeFormCB = new Size(150, 20); //comboBox существующих форм

private Size sizeTypeCB = new Size(150, 20); //comboBox типа записи

private Size sizeMailTB = new Size(150, 20); //textBox почты

private Size sizePNumberTB = new Size(150, 20); //textBox Телефона

private Size sizeSiteTB = new Size(150, 20); //textBox Сайта

private Size sizeAddressTB = new Size(150, 20); //textBox Адреса

private Size sizeDateTB = new Size(100, 20); //DateTimePicker времени записи

private Size sizeDateCB = new Size(60, 20); //ComboBox времени записи

private Size sizeFormBut = new Size(150, 30); //кнопка добавить форму

public CreateWriteForm()

{

}

public void NewWrite()

{

CreateCitizenLabel();

CreateSurLabel();

CreateSurTB();

CreateNameLabel();

CreateNameTB();

CreatePatLabel();

CreatePatTB();

CreateDateBLabel();

CreateDateBTB();

CreateSNILSLabel();

CreateSNILSBTB();

CreateSeriesLabel();

CreateSeriesTB();

CreateNumLabel();

CreateNumTB();

CreateAddButton();

CreateExtButton();

//CreateFormLabel();

//CreateFormCB();

CreateTypeLabel();

CreateMailLabel();

CreateMailTB();

CreatePNumberLabel();

CreatePNumberTB();

CreateSiteLabel();

CreateSiteTB();

CreateAddressLabel();

CreateAddressTB();

CreateDateLabel();

CreateDateTB();

CreateDateCB();

//CreateFormButton();

CreateCitizenCB();

CreateTypeCB();

if (FormEnab)

{

FNewWrite.Text = "Добавление записи";

FNewWrite.Controls["AddButton"].Text = "Добавить";

FNewWrite.Controls["AddButton"].Click += AddButtonClick;

}

else

{

FNewWrite.Text = "Редактирование записи";

FNewWrite.Controls["AddButton"].Text = "Редактировать";

FNewWrite.Controls["AddButton"].Click += UpdButtonClick;

FNewWrite.Controls["AddButton"].Enabled = true;

}

FNewWrite.Size = new Size(760, 400);

FNewWrite.FormBorderStyle = FormBorderStyle.FixedSingle;

FNewWrite.MaximizeBox = false;

FNewWrite.StartPosition = FormStartPosition.CenterScreen;

FNewWrite.ShowDialog();

}

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

{

if (FormEnab)

switch ((sender as ComboBox).SelectedItem)

{

case "По телефону":

{

(FNewWrite.Controls["PNumberTextBox"] as MaskedTextBox).Enabled = true;

(FNewWrite.Controls["AddressTextBox"] as TextBox).Enabled = false;

(FNewWrite.Controls["SiteTextBox"] as TextBox).Enabled = false;

(FNewWrite.Controls["MailTextBox"] as MaskedTextBox).Enabled = false;

break;

}

case "Через интернет":

{

(FNewWrite.Controls["SiteTextBox"] as TextBox).Enabled = true;

(FNewWrite.Controls["AddressTextBox"] as TextBox).Enabled = false;

(FNewWrite.Controls["PNumberTextBox"] as MaskedTextBox).Enabled = false;

(FNewWrite.Controls["MailTextBox"] as MaskedTextBox).Enabled = false;

break;

}

case "По почте":

{

(FNewWrite.Controls["MailTextBox"] as MaskedTextBox).Enabled = true;

(FNewWrite.Controls["AddressTextBox"] as TextBox).Enabled = false;

(FNewWrite.Controls["PNumberTextBox"] as MaskedTextBox).Enabled = false;

(FNewWrite.Controls["SiteTextBox"] as TextBox).Enabled = false;

break;

}

case "Через больницу":

{

(FNewWrite.Controls["AddressTextBox"] as TextBox).Enabled = true;

(FNewWrite.Controls["PNumberTextBox"] as MaskedTextBox).Enabled = false;

(FNewWrite.Controls["SiteTextBox"] as TextBox).Enabled = false;

(FNewWrite.Controls["MailTextBox"] as MaskedTextBox).Enabled = false;

break;

}

}

}

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

{

if ((sender as ComboBox).SelectedItem != "Добавить нового")

{

if (FormEnab)

{

(FNewWrite.Controls["NameTextBox"] as TextBox).Enabled = false;

(FNewWrite.Controls["SurTextBox"] as TextBox).Enabled = false;

(FNewWrite.Controls["PatTextBox"] as TextBox).Enabled = false;

(FNewWrite.Controls["SNILSTextBox"] as TextBox).Enabled = false;

(FNewWrite.Controls["SeriesTextBox"] as TextBox).Enabled = false;

(FNewWrite.Controls["NumberTextBox"] as TextBox).Enabled = false;

(FNewWrite.Controls["DateBPicker"] as DateTimePicker).Enabled = false;

}

DataRow[] DRCit = DataCit.Select("[ФИО] = '" + (sender as ComboBox).Text + "'");

SqlParameter[] SP = new SqlParameter[0];

DataTable DT = DB.GetData("Select\_DataCitizen", SP);

int id = Convert.ToInt32(DRCit[0]["id\_Citizen"]);

DataRow[] DRAllCit = DT.Select("[id\_Citizen] = '" + id.ToString() + "'");

string[] FIOCit = DRAllCit[0]["ФИО"].ToString().Split(new char[] { ' ' });

string[] Pass = DRAllCit[0]["Серия и номер"].ToString().Split(new char[] { ' ', 'и' });

(FNewWrite.Controls["SurTextBox"] as TextBox).Text = FIOCit[0];

(FNewWrite.Controls["NameTextBox"] as TextBox).Text = FIOCit[1];

(FNewWrite.Controls["PatTextBox"] as TextBox).Text = FIOCit[2];

(FNewWrite.Controls["SNILSTextBox"] as TextBox).Text = DRAllCit[0]["Снилс"].ToString();

(FNewWrite.Controls["SeriesTextBox"] as TextBox).Text = Pass[0];

(FNewWrite.Controls["NumberTextBox"] as TextBox).Text = Pass[1];

CultureInfo provider = CultureInfo.InvariantCulture;

(FNewWrite.Controls["DateBPicker"] as DateTimePicker).Value = DateTime.ParseExact(DRAllCit[0]["День рождения"].ToString().Substring(0, 10), "dd.MM.yyyy", provider);

}

else

{

(FNewWrite.Controls["NameTextBox"] as TextBox).Enabled = true;

(FNewWrite.Controls["SurTextBox"] as TextBox).Enabled = true;

(FNewWrite.Controls["PatTextBox"] as TextBox).Enabled = true;

(FNewWrite.Controls["SNILSTextBox"] as TextBox).Enabled = true;

(FNewWrite.Controls["SeriesTextBox"] as TextBox).Enabled = true;

(FNewWrite.Controls["NumberTextBox"] as TextBox).Enabled = true;

(FNewWrite.Controls["DateBPicker"] as DateTimePicker).Enabled = true;

}

}

//конпка выхода

private void ExitButtonClick(object sender, EventArgs e)

{

FNewWrite.Close();

}

//добавить запись

private void AddButtonClick(object sender, EventArgs e)

{

//добавляем форму записи

int id\_FormWrite = 0;

DataRow[] DRType = Type.Select("[Тип записи] = '" + (FNewWrite.Controls["FormCB"] as ComboBox).Text + "'");

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@id\_worker", AutorizForm.id\_Worker),

new SqlParameter("@id\_TypeWrite ", DRType[0]["id\_TypeWrite"]),

new SqlParameter("@mail", (FNewWrite.Controls["MailTextBox"] as MaskedTextBox).Text),

new SqlParameter("@PhoneNumber", (FNewWrite.Controls["PNumberTextBox"] as MaskedTextBox).Text),

new SqlParameter("@Sites", (FNewWrite.Controls["SiteTextBox"] as TextBox).Text),

new SqlParameter("@Adress", (FNewWrite.Controls["AddressTextBox"] as TextBox).Text),

};

switch (DRType[0]["Тип записи"])

{

case "По почте":

{

SP[3].Value = DBNull.Value;

SP[4].Value = DBNull.Value;

SP[5].Value = DBNull.Value;

break;

}

case "По телефону":

{

SP[2].Value = DBNull.Value;

SP[4].Value = DBNull.Value;

SP[5].Value = DBNull.Value;

break;

}

case "Через больницу":

{

SP[2].Value = DBNull.Value;

SP[3].Value = DBNull.Value;

SP[4].Value = DBNull.Value;

break;

}

case "Через интернет":

{

SP[2].Value = DBNull.Value;

SP[3].Value = DBNull.Value;

SP[5].Value = DBNull.Value;

break;

}

}

id\_FormWrite = DB.WriteData("[Add\_FormWrite]", SP);

//добавляем граждана

int id\_Citizen = 0;

if (DB.LastWrite)

if (((FNewWrite.Controls["CitizenCB"] as ComboBox).SelectedItem == "Добавить нового") | ((FNewWrite.Controls["CitizenCB"] as ComboBox).SelectedItem == ""))

{

DateTime res = (FNewWrite.Controls["DateBPicker"] as DateTimePicker).Value;

SP = new SqlParameter[7]

{

new SqlParameter("@NameCit", (FNewWrite.Controls["NameTextBox"] as TextBox).Text),

new SqlParameter("@SurnameCit", (FNewWrite.Controls["SurTextBox"] as TextBox).Text),

new SqlParameter("@PatronymicCit", (FNewWrite.Controls["PatTextBox"] as TextBox).Text),

new SqlParameter("@Snils", (FNewWrite.Controls["SNILSTextBox"] as TextBox).Text),

new SqlParameter("@SeriesPassportCit", (FNewWrite.Controls["SeriesTextBox"] as TextBox).Text),

new SqlParameter("@NumberPassportCit", (FNewWrite.Controls["NumberTextBox"] as TextBox).Text),

new SqlParameter("@DateBirthCit", res),

};

id\_Citizen = DB.WriteData("Add\_DataCitizen", SP);

UpdBox(1, (FNewWrite.Controls["CitizenCB"] as ComboBox));

string ss = (FNewWrite.Controls["SurTextBox"] as TextBox).Text + " " + (FNewWrite.Controls["NameTextBox"] as TextBox).Text + " " + (FNewWrite.Controls["PatTextBox"] as TextBox).Text;

//MessageBox.Show(ss);

(FNewWrite.Controls["CitizenCB"] as ComboBox).SelectedItem = ss;

}

else

{

DataRow[] DRCit = DataCit.Select("[ФИО] = '" + (FNewWrite.Controls["CitizenCB"] as ComboBox).Text + "'");

id\_Citizen = Convert.ToInt32(DRCit[0]["id\_Citizen"]);

}

//добавляем запись

DataRow[] DRDR = DateRecord.Select("[Record\_Time] = '" + (FNewWrite.Controls["DateCB"] as ComboBox).Text + "'");

SP = new SqlParameter[]

{

new SqlParameter("@times", (FNewWrite.Controls["DatePicker"] as DateTimePicker).Value.Date),

new SqlParameter("@id\_Day\_of\_the\_week", DRDR[0]["id\_Day\_of\_the\_week"]),

new SqlParameter("@id\_Citizen", id\_Citizen),

new SqlParameter("@id\_FormWrite", id\_FormWrite),

//new SqlParameter("@id\_TypeWrite", ),

//new SqlParameter("@mail", ),

//new SqlParameter("@PhoneNumber", ),

//new SqlParameter("@Sites", ),

//new SqlParameter("@Adress", ),

};

if (DB.LastWrite)

DB.WriteData("[Add\_WriteAppointment]", SP);

if (DB.LastWrite)

FNewWrite.Close();

}

//Изменить запись

private void UpdButtonClick(object sender, EventArgs e)

{

DateTime res = (FNewWrite.Controls["DateBPicker"] as DateTimePicker).Value;

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@id\_Citizen", id\_cit),

new SqlParameter("@NameCit", (FNewWrite.Controls["NameTextBox"] as TextBox).Text),

new SqlParameter("@SurnameCit", (FNewWrite.Controls["SurTextBox"] as TextBox).Text),

new SqlParameter("@PatronymicCit", (FNewWrite.Controls["PatTextBox"] as TextBox).Text),

new SqlParameter("@Snils", (FNewWrite.Controls["SNILSTextBox"] as TextBox).Text),

new SqlParameter("@SeriesPassportCit", (FNewWrite.Controls["SeriesTextBox"] as TextBox).Text),

new SqlParameter("@NumberPassportCit", (FNewWrite.Controls["NumberTextBox"] as TextBox).Text),

new SqlParameter("@DateBirthCit", res),

};

DB.WriteData("[UPD\_DataCitizen]", SP);

if (DB.LastUpd)

FNewWrite.Close();

}

private void DatePickerValueChanged(object sender, EventArgs e)

{

UpdBox(2,FNewWrite.Controls["DateCB"] as ComboBox);

}

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

{

try

{

if ((sender as TextBox).Text != "")

{

(FNewWrite.Controls["AddButton"] as Button).Enabled = true;

}

else

{

(FNewWrite.Controls["AddButton"] as Button).Enabled = false;

}

}

catch { }

try

{

if ((sender as MaskedTextBox).Text != "")

{

(FNewWrite.Controls["AddButton"] as Button).Enabled = true;

}

else

{

(FNewWrite.Controls["AddButton"] as Button).Enabled = false;

}

}

catch { }

}

private void UpdBox(int n, ComboBox cb)

{

switch (n)

{

case 1:

{

SqlParameter[] SP = new SqlParameter[0];

DataCit = DB.GetData("Select\_DataCitizen\_ONLI\_FIO", SP);

DataRow RW;

cb.Items.Clear();

cb.Text = "";

cb.Items.Add("Добавить нового");

for (int i = 0; i < DataCit.Rows.Count; i++)

{

RW = DataCit.Rows[i];

cb.Items.Add(RW["ФИО"]);

}

cb.SelectedIndex = 0;

break;

}

case 2:

{

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@Date", (FNewWrite.Controls["DatePicker"] as DateTimePicker).Value.Date),

};

DateRecord = DB.GetData("Select\_Date\_Appointment", SP);

DataRow RW;

for (int i = 0; i < DateRecord.Rows.Count; i++)

{

DateRecord.Rows[i]["Record\_Time"] = DateRecord.Rows[i]["Record\_Time"].ToString().Substring(0, 5);

}

cb.Items.Clear();

cb.Text = "";

for (int i = 0; i < DateRecord.Rows.Count; i++)

{

RW = DateRecord.Rows[i];

cb.Items.Add(RW["Record\_Time"]);

}

cb.SelectedIndex = 0;

break;

}

}

}

private void CreateCitizenLabel()

{

lab = new Label()

{

Name = "CitizenLabel",

Text = "Граждан: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locCitizenLabel,

AutoSize = true,

Enabled = citEnab,

};

FNewWrite.Controls.Add(lab);

}

private void CreateCitizenCB()

{

cb = new ComboBox()

{

Name = "CitizenCB",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locCitizenCB,

Size = sizeCitizenCB,

AutoCompleteMode = AutoCompleteMode.SuggestAppend,

AutoCompleteSource = AutoCompleteSource.ListItems,

Enabled = FormEnab,

};

cb.SelectedIndexChanged += Citizen\_SelectedIndexChanged;

UpdBox(1, cb);

if (NameSit != "")

{

DataRow[] DRSit = DataCit.Select("[ФИО] LIKE '%" + SurSit + " " + NameSit + " " + PatSit + "%'");

cb.SelectedItem = DRSit[0]["ФИО"].ToString();

}

FNewWrite.Controls.Add(cb);

}

private void CreateSurLabel()

{

lab = new Label()

{

Name = "SurLabel",

Text = "Фамилия: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locSurLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreateSurTB()

{

Tb = new TextBox()

{

Name = "SurTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locSurTB,

Size = sizeSurTB,

MaxLength = 50,

};

FNewWrite.Controls.Add(Tb);

}

private void CreateNameLabel()

{

lab = new Label()

{

Name = "NameLabel",

Text = "Имя: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locNameLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreateNameTB()

{

Tb = new TextBox()

{

Name = "NameTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locNameTB,

Size = sizeNameTB,

MaxLength = 50,

};

FNewWrite.Controls.Add(Tb);

}

private void CreatePatLabel()

{

lab = new Label()

{

Name = "PatLabel",

Text = "Отчество: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locPatLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreatePatTB()

{

Tb = new TextBox()

{

Name = "PatTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locPatTB,

Size = sizePatTB,

MaxLength = 50,

};

FNewWrite.Controls.Add(Tb);

}

private void CreateDateBLabel()

{

lab = new Label()

{

Name = "DateBLabel",

Text = "Дата рождения: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locDateBLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreateDateBTB()

{

DTP = new DateTimePicker()

{

Name = "DateBPicker",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locDateBTB,

Size = sizeDateBTB,

Format = DateTimePickerFormat.Custom,

CustomFormat = "dd.MM.yyyy",

Value = new DateTime(2000, 1, 1),

};

FNewWrite.Controls.Add(DTP);

}

private void CreateSNILSLabel()

{

lab = new Label()

{

Name = "SNILSLabel",

Text = "Снилс: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locSNILSLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreateSNILSBTB()

{

Tb = new TextBox()

{

Name = "SNILSTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locSNILSTB,

Size = sizeSNILSTB,

MaxLength = 11,

};

FNewWrite.Controls.Add(Tb);

}

private void CreateSeriesLabel()

{

lab = new Label()

{

Name = "SeriesLabel",

Text = "Серия пасспорта: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locSeriesLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreateSeriesTB()

{

Tb = new TextBox()

{

Name = "SeriesTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locSeriesTB,

Size = sizeSeriesTB,

MaxLength = 4,

};

FNewWrite.Controls.Add(Tb);

}

private void CreateNumLabel()

{

lab = new Label()

{

Name = "NumberLabel",

Text = "Номер пасспорта: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locNumberLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreateNumTB()

{

Tb = new TextBox()

{

Name = "NumberTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locNumberTB,

Size = sizeNumberTB,

MaxLength = 6,

};

FNewWrite.Controls.Add(Tb);

}

private void CreateAddButton()

{

but = new Button()

{

Name = "AddButton",

Text = "Добавить",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locAddBut,

Size = sizeAddBut,

Enabled = false,

};

FNewWrite.Controls.Add(but);

}

private void CreateExtButton()

{

but = new Button()

{

Name = "ExtButton",

Text = "Выйти",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locExtBut,

Size = sizeExtBut,

};

but.Click += ExitButtonClick;

FNewWrite.Controls.Add(but);

}

private void CreateTypeLabel()

{

lab = new Label()

{

Name = "TypeLabel",

Text = "Тип записи",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locTypeLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreateTypeCB()

{

cb = new ComboBox()

{

Name = "FormCB",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locTypeCB,

Size = sizeTypeCB,

AutoCompleteMode = AutoCompleteMode.SuggestAppend,

AutoCompleteSource = AutoCompleteSource.ListItems,

Enabled = FormEnab,

};

cb.SelectedIndexChanged += TypeCB\_SelectedIndexChanged;

SqlParameter[] SP = new SqlParameter[0];

Type = DB.GetData("Celect\_TypeWrite", SP);

DataRow RW;

cb.Items.Clear();

cb.Text = "";

for (int i = 0; i < Type.Rows.Count; i++)

{

RW = Type.Rows[i];

cb.Items.Add(RW["Тип записи"]);

}

cb.SelectedIndex = 0;

FNewWrite.Controls.Add(cb);

}

private void CreateMailLabel()

{

lab = new Label()

{

Name = "MailLabel",

Text = "Почта: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locMailLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreateMailTB()

{

mtb = new MaskedTextBox()

{

Name = "MailTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locMailTB,

Size = sizeMailTB,

MaxLength = 50,

Enabled = false,

};

mtb.TextChanged += Form\_TextChanged;

FNewWrite.Controls.Add(mtb);

}

private void CreatePNumberLabel()

{

lab = new Label()

{

Name = "PNumberLabel",

Text = "Номер телефона: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locPNumberLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreatePNumberTB()

{

mtb = new MaskedTextBox()

{

Name = "PNumberTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locPNumberTB,

Size = sizePNumberTB,

Enabled = false,

};

FNewWrite.Controls.Add(mtb);

}

private void CreateSiteLabel()

{

lab = new Label()

{

Name = "SiteLabel",

Text = "Сайт: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locSiteLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreateSiteTB()

{

Tb = new TextBox()

{

Name = "SiteTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locSiteTB,

Size = sizeSiteTB,

MaxLength = 100,

Enabled = false,

};

FNewWrite.Controls.Add(Tb);

}

private void CreateAddressLabel()

{

lab = new Label()

{

Name = "AddressLabel",

Text = "Адрес: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locAddressLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreateAddressTB()

{

Tb = new TextBox()

{

Name = "AddressTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locAddressTB,

Size = sizeAddressTB,

MaxLength = 100,

Enabled = false,

};

FNewWrite.Controls.Add(Tb);

}

private void CreateDateLabel()

{

lab = new Label()

{

Name = "DateLabel",

Text = "Время записи: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locDateLabel,

AutoSize = true,

};

FNewWrite.Controls.Add(lab);

}

private void CreateDateTB()

{

DTP = new DateTimePicker()

{

Name = "DatePicker",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locDateTB,

Size = sizeDateTB,

Format = DateTimePickerFormat.Custom,

CustomFormat = "dd.MM.yyyy",

Enabled = FormEnab,

};

DTP.ValueChanged += new EventHandler(DatePickerValueChanged);

FNewWrite.Controls.Add(DTP);

}

private void CreateDateCB()

{

cb = new ComboBox()

{

Name = "DateCB",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locDateCB,

Size = sizeDateCB,

AutoCompleteMode = AutoCompleteMode.SuggestAppend,

AutoCompleteSource = AutoCompleteSource.ListItems,

DropDownStyle = ComboBoxStyle.DropDownList,

Enabled = FormEnab,

};

UpdBox(2,cb);

FNewWrite.Controls.Add(cb);

}

}

}

DB.cs

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp9

{

static class DB

{

public static bool LogIn = false;

public static bool LastWrite = false;

public static bool LastLogDel = false;

public static bool LastGet = false;

public static bool LastUpd = false;

public static bool LastFunk = false;

public static DataTable GetData(string ProcName, SqlParameter[] SP)

{

try

{

SqlConnection sql = new SqlConnection(DataBaseConfiguration.connectString);

sql.Open();

SqlCommand GetData = new SqlCommand(ProcName, sql);

GetData.CommandType = CommandType.StoredProcedure;

for (int i = 0; i < SP.Length; i++)

{

GetData.Parameters.Add(SP[i]);

}

DataTable DT = new DataTable();

SqlDataReader DR = GetData.ExecuteReader();

DT.Load(DR);

sql.Close();

LastGet = true;

return DT;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

LastGet = false;

return null;

}

}

public static int WriteData(string ProcName, SqlParameter[] SP)

{

try

{

SqlConnection sql = new SqlConnection(DataBaseConfiguration.connectString);

sql.Open();

SqlCommand AddCitizen = new SqlCommand(ProcName, sql);

AddCitizen.CommandType = CommandType.StoredProcedure;

for (int i = 0; i < SP.Length; i++)

{

AddCitizen.Parameters.Add(SP[i]);

}

var id = AddCitizen.ExecuteNonQuery();

sql.Close();

LastWrite = true;

return (int)id;

}

catch(Exception ex)

{

MessageBox.Show(ex.Message.ToString());

LastWrite = false;

return -10;

}

}

//public static void DelData(int id)

//{

//}

public static void LogDelData(string ProcName, SqlParameter[] SP)

{

try

{

SqlConnection sql = new SqlConnection(DataBaseConfiguration.connectString);

sql.Open();

SqlCommand UpdData = new SqlCommand(ProcName, sql);

UpdData.CommandType = CommandType.StoredProcedure;

for (int i = 0; i < SP.Length; i++)

{

UpdData.Parameters.Add(SP[i]);

}

UpdData.ExecuteNonQuery();

sql.Close();

LastLogDel = true;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

LastLogDel = false;

}

}

public static void UpdData(string ProcName, SqlParameter[] SP)

{

try

{

SqlConnection sql = new SqlConnection(DataBaseConfiguration.connectString);

sql.Open();

SqlCommand UpdData = new SqlCommand(ProcName, sql);

UpdData.CommandType = CommandType.StoredProcedure;

for (int i = 0; i < SP.Length; i++)

{

UpdData.Parameters.Add(SP[i]);

}

UpdData.ExecuteNonQuery();

sql.Close();

LastUpd = true;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

LastUpd = false;

}

}

public static DataTable Funk(string FunkName, object[] vs)

{

DataTable DT = new DataTable();

SqlConnection sql = new SqlConnection(DataBaseConfiguration.connectString);

string query = string.Format("SELECT dbo." + FunkName, vs[0]);

SqlCommand cmd = new SqlCommand(query, sql);

try

{

sql.Open();

SqlDataReader DR = cmd.ExecuteReader();

DT.Load(DR);

sql.Close();

LastFunk = true;

return DT;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

LastFunk = false;

return null;

}

}

}

}

DeliMediForm.cs

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp9

{

static class DB

{

public static bool LogIn = false;

public static bool LastWrite = false;

public static bool LastLogDel = false;

public static bool LastGet = false;

public static bool LastUpd = false;

public static bool LastFunk = false;

public static DataTable GetData(string ProcName, SqlParameter[] SP)

{

try

{

SqlConnection sql = new SqlConnection(DataBaseConfiguration.connectString);

sql.Open();

SqlCommand GetData = new SqlCommand(ProcName, sql);

GetData.CommandType = CommandType.StoredProcedure;

for (int i = 0; i < SP.Length; i++)

{

GetData.Parameters.Add(SP[i]);

}

DataTable DT = new DataTable();

SqlDataReader DR = GetData.ExecuteReader();

DT.Load(DR);

sql.Close();

LastGet = true;

return DT;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

LastGet = false;

return null;

}

}

public static int WriteData(string ProcName, SqlParameter[] SP)

{

try

{

SqlConnection sql = new SqlConnection(DataBaseConfiguration.connectString);

sql.Open();

SqlCommand AddCitizen = new SqlCommand(ProcName, sql);

AddCitizen.CommandType = CommandType.StoredProcedure;

for (int i = 0; i < SP.Length; i++)

{

AddCitizen.Parameters.Add(SP[i]);

}

var id = AddCitizen.ExecuteNonQuery();

sql.Close();

LastWrite = true;

return (int)id;

}

catch(Exception ex)

{

MessageBox.Show(ex.Message.ToString());

LastWrite = false;

return -10;

}

}

//public static void DelData(int id)

//{

//}

public static void LogDelData(string ProcName, SqlParameter[] SP)

{

try

{

SqlConnection sql = new SqlConnection(DataBaseConfiguration.connectString);

sql.Open();

SqlCommand UpdData = new SqlCommand(ProcName, sql);

UpdData.CommandType = CommandType.StoredProcedure;

for (int i = 0; i < SP.Length; i++)

{

UpdData.Parameters.Add(SP[i]);

}

UpdData.ExecuteNonQuery();

sql.Close();

LastLogDel = true;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

LastLogDel = false;

}

}

public static void UpdData(string ProcName, SqlParameter[] SP)

{

try

{

SqlConnection sql = new SqlConnection(DataBaseConfiguration.connectString);

sql.Open();

SqlCommand UpdData = new SqlCommand(ProcName, sql);

UpdData.CommandType = CommandType.StoredProcedure;

for (int i = 0; i < SP.Length; i++)

{

UpdData.Parameters.Add(SP[i]);

}

UpdData.ExecuteNonQuery();

sql.Close();

LastUpd = true;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

LastUpd = false;

}

}

public static DataTable Funk(string FunkName, object[] vs)

{

DataTable DT = new DataTable();

SqlConnection sql = new SqlConnection(DataBaseConfiguration.connectString);

string query = string.Format("SELECT dbo." + FunkName, vs[0]);

SqlCommand cmd = new SqlCommand(query, sql);

try

{

sql.Open();

SqlDataReader DR = cmd.ExecuteReader();

DT.Load(DR);

sql.Close();

LastFunk = true;

return DT;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

LastFunk = false;

return null;

}

}

}

}

DiagForm.cs

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp9

{

class DiagForm

{

private DataTable Medicaments;

private DataTable Disease;

private DataTable Deportaments;

private DataTable TypeUse;

public string id\_Diag = "";

public string NameMeca = "";

public string NameDise = "";

public string TimeHeal = "";

public string AmounMed = "";

public string NumbDeprot = "";

public string TypeDise = "";

public string CaptionButton = "Создать";

private Form FDiag = new Form();

private Label lab = new Label();

private TextBox Tb = new TextBox();

private MaskedTextBox mtb = new MaskedTextBox();

private ComboBox cb = new ComboBox();

private Button but = new Button();

private Font DefFont = new Font("Microsoft Sans Serif", 10);

private Color DefBackColor = Color.FromName("Control");

private Color DefTextColor = Color.FromName("ControlText");

//Позиции //позиции

private Point locMedicamentLabel = new Point(20, 10); //label лекарств

private Point locMedicamentCB = new Point(20, 33); //ComboBox лекарств

private Point locDiseaseLabel = new Point(20, 63); //label болезней

private Point locDiseaseCB = new Point(20, 86); //ComboBox болезней

private Point locDiseaseBut = new Point(180, 86); //кнопка болезней

private Point locTimeLabel = new Point(20, 116); //label срок лечения

private Point locTimeTB = new Point(20, 139); //TextBox срок лечения

private Point locAmountLabel = new Point(20, 169); //label кол-во лекарств

private Point locAmountTB = new Point(20, 192); //textBox кол-во лекарств

private Point locRoomLabel = new Point(20, 222); //label выбор палаты

private Point locRoomCB = new Point(20, 245); //ComboBox выбор палаты

private Point locUseLabel = new Point(20, 275); //label тип потребления

private Point locUseCB = new Point(20, 298); //ComboBox тип потребления

private Point locUseBut = new Point(180, 298); //кнопка типов потребления

private Point locApplyBut = new Point(20, 338); //кнопка Создать

private Point locCanBut = new Point(150, 338); //кнопка Отмена

//размеры //размеры

private Size sizeMedicamentCB = new Size(200, 20); //ComboBox лекарств

private Size sizeDiseaseCB = new Size(150, 20); //ComboBox болезней

private Size sizeDiseaseBut = new Size(75, 25); //кнопка болезней

private Size sizeTimeTB = new Size(100, 20); //TextBox срок лечения

private Size sizeAmountTB = new Size(50, 20); //textBox кол-во лекарств

private Size sizeRoomCB = new Size(200, 20); //ComboBox выбор палаты

private Size sizeUseCB = new Size(150, 20); //ComboBox тип потребления

private Size sizeUseBut = new Size(75, 25); //кнопка типов потребления

private Size sizeApplyBut = new Size(100, 30); //кнопка Создать

private Size sizeCanBut = new Size(100, 30); //кнопка Отмена

public void NewDiagForm()

{

CreateMedicamentLabel();

CreateMedicamentCB();

CreateDiseaseLabel();

CreateDiseaseCB();

CreateDiseaseButton();

CreateTimeLabel();

CreateTimeTB();

CreateAmountLabel();

CreateAmountTB();

CreateRoomLabel();

CreateRoomCB();

CreateUseLabel();

CreateUseCB();

CreateUseBut();

CreateApplyButton();

CreateCanButton();

FDiag.Text = "Создание диагноза";

FDiag.Size = new Size(280, 425);

FDiag.FormBorderStyle = FormBorderStyle.FixedSingle;

FDiag.MaximizeBox = false;

FDiag.StartPosition = FormStartPosition.CenterScreen;

FDiag.FormClosing += new FormClosingEventHandler(FormClosing);

FDiag.Controls["CanButton"].Click += new EventHandler(CancelButtonClick);

if ((FDiag.Controls["ApplyButton"] as Button).Text == "Создать")

FDiag.Controls["ApplyButton"].Click += new EventHandler(ApplyButtonClick);

else

FDiag.Controls["ApplyButton"].Click += new EventHandler(DiagnozButtonClick);

FDiag.Controls["DiseaseCB"].TextChanged += DiseaseTextChange;

FDiag.Controls["DiseaseButton"].Click += DiseasiButton;

FDiag.Controls["UseCB"].TextChanged += TypeUseTextChange;

FDiag.Controls["UseButton"].Click += TypeButtonClick;

FDiag.ShowDialog();

}

//закрытие формы

private void FormClosing(object sender, EventArgs e)

{

(Application.OpenForms[0] as MainForm).UpdGrids(2);

}

//отмена

private void CancelButtonClick(object sender, EventArgs e)

{

DialogResult DR = MessageBox.Show("Вы уверены что хотите отменить создание?\n" +

"Все веденные данные будут потеряны.", "Закрыть?", MessageBoxButtons.YesNo);

if (DR == DialogResult.Yes)

FDiag.Close();

}

//кнопка создать

private void ApplyButtonClick(object sender, EventArgs e)

{

DataRow[] DRDerp = Deportaments.Select("[Терапевтическое отделение] = '"+ (FDiag.Controls["RoomCB"] as ComboBox).Text + "'");

DataRow[] DRType = TypeUse.Select("[Name\_Use] = '" + (FDiag.Controls["UseCB"] as ComboBox).Text + "'");

DataRow[] DRMedi = Medicaments.Select("[Name\_Medicament] = '" + (FDiag.Controls["MedicamentCB"] as ComboBox).Text + "'");

DataRow[] DRDise = Disease.Select("[Name\_Disease] = '" + (FDiag.Controls["DiseaseCB"] as ComboBox).Text + "'");

try

{

SqlParameter[] SP = new SqlParameter[6]

{

new SqlParameter ("@TimeDisease", (FDiag.Controls["TimeTextBox"] as TextBox).Text),

new SqlParameter ("@Amount", (FDiag.Controls["AmountTextBox"] as TextBox).Text),

new SqlParameter ("@ID\_Room", DRDerp[0]["id\_Room"]),

new SqlParameter ("@id\_TypeUse", DRType[0]["id\_TypeUse"]),

new SqlParameter ("@id\_Medicament", DRMedi[0]["id\_Medicament"]),

new SqlParameter ("@id\_Disease", DRDise[0]["id\_Disease"]),

};

DB.WriteData("Add\_Diagnosis", SP);

}

catch

{

MessageBox.Show("Неудалось добавить введенные данные.\nПожалуйста проверьте правильность введенных данных.");

}

if (DB.LastWrite)

FDiag.Close();

}

//кнопка изменить Диагноз

private void DiagnozButtonClick(object sender, EventArgs e)

{

DataRow[] DRDerp = Deportaments.Select("[Терапевтическое отделение] = '" + (FDiag.Controls["RoomCB"] as ComboBox).Text + "'");

DataRow[] DRType = TypeUse.Select("[Name\_Use] = '" + (FDiag.Controls["UseCB"] as ComboBox).Text + "'");

DataRow[] DRMedi = Medicaments.Select("[Name\_Medicament] = '" + (FDiag.Controls["MedicamentCB"] as ComboBox).Text + "'");

DataRow[] DRDise = Disease.Select("[Name\_Disease] = '" + (FDiag.Controls["DiseaseCB"] as ComboBox).Text + "'");

try

{

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@id\_Diagnoz", id\_Diag),

new SqlParameter("@TimeDisease", (FDiag.Controls["TimeTextBox"] as TextBox).Text),

new SqlParameter("@Amount", (FDiag.Controls["AmountTextBox"] as TextBox).Text),

new SqlParameter("@ID\_Room", DRDerp[0]["id\_Room"]),

new SqlParameter ("@id\_TypeUse", DRType[0]["id\_TypeUse"]),

new SqlParameter ("@id\_Medicament", DRMedi[0]["id\_Medicament"]),

new SqlParameter ("@id\_Disease", DRDise[0]["id\_Disease"]),

new SqlParameter("@Diagnosis\_Logical\_Deletebit", false),

};

DB.UpdData("UPD\_Diagnosis", SP);

}

catch

{

MessageBox.Show("Неудалось добавить введенные данные.\nПожалуйста проверьте правильность введенных данных.");

}

if (DB.LastWrite)

FDiag.Close();

}

//кнопка изменения болзени

private void DiseasiButton(object sender, EventArgs e)

{

switch ((sender as Button).Text)

{

case "Изменить":

{

DataRow[] DRDise = Disease.Select("[Name\_Disease] = '" + (FDiag.Controls["DiseaseCB"] as ComboBox).Text + "'");

EditDelForm.EditData = (FDiag.Controls["DiseaseCB"] as ComboBox).Text;

EditDelForm.NewMess("Изменить", "Удалить", 2);

if (EditDelForm.LastResult == "Изменить")

{

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@id\_Disease", DRDise[0]["id\_Disease"]),

new SqlParameter("@Name\_Disease", EditDelForm.EditData)

};

DB.UpdData("[UPD\_DirectoryDisease]", SP);

}

if (EditDelForm.LastResult == "Удалить")

{

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@id\_Disease", DRDise[0]["id\_Disease"]),

};

DB.LogDelData("[logdel\_DirectoryDisease]", SP);

}

break;

}

case "Создать":

{

string Text = (FDiag.Controls["DiseaseCB"] as ComboBox).Text;

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@Name\_Disease", Text),

};

DB.WriteData("[Add\_DirectoryDisease]", SP);

break;

}

}

UpdDisease((FDiag.Controls["DiseaseCB"] as ComboBox));

}

//кнопка создания типов потребления

private void TypeButtonClick(object sender, EventArgs e)

{

switch ((sender as Button).Text)

{

case "Изменить":

{

DataRow[] DRType = TypeUse.Select("[Name\_Use] = '" + (FDiag.Controls["UseCB"] as ComboBox).Text + "'");

EditDelForm.EditData = (FDiag.Controls["UseCB"] as ComboBox).Text;

EditDelForm.NewMess("Изменить", "Удалить", 2);

if (EditDelForm.LastResult == "Изменить")

{

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@id\_TypeUse", DRType[0]["id\_TypeUse"]),

new SqlParameter("@NameUse", EditDelForm.EditData)

};

DB.UpdData("[UPD\_TypeUse]", SP);

}

if (EditDelForm.LastResult == "Удалить")

{

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@id\_TypeUse", DRType[0]["id\_TypeUse"]),

};

DB.LogDelData("[logdel\_TypeUse]", SP);

}

break;

}

case "Создать":

{

string Text = (FDiag.Controls["UseCB"] as ComboBox).Text;

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@NameUse", Text),

};

DB.WriteData("[Add\_TypeUse]", SP);

break;

}

}

UpdType((FDiag.Controls["UseCB"] as ComboBox));

}

//Выбор болезни

private void DiseaseTextChange(object sender, EventArgs e)

{

string SText = (sender as ComboBox).Text;

FDiag.Controls["DiseaseButton"].Enabled = (SText != "");

if ((sender as ComboBox).SelectedIndex == -1)

{

FDiag.Controls["DiseaseButton"].Text = "Создать";

}

else

{

FDiag.Controls["DiseaseButton"].Text = "Изменить";

}

}

//выбор типов потреблениия

private void TypeUseTextChange(object sender, EventArgs e)

{

string SText = (sender as ComboBox).Text;

FDiag.Controls["UseButton"].Enabled = (SText != "");

if ((sender as ComboBox).SelectedIndex == -1)

{

FDiag.Controls["UseButton"].Text = "Создать";

}

else

{

FDiag.Controls["UseButton"].Text = "Изменить";

}

}

//обновление Болезней

private void UpdDisease(ComboBox cb)

{

SqlParameter[] SP = new SqlParameter[0];

Disease = DB.GetData("Select\_Name\_Disease", SP);

DataRow RW;

cb.Items.Clear();

cb.Text = "";

for (int i = 0; i < Disease.Rows.Count; i++)

{

RW = Disease.Rows[i];

cb.Items.Add(RW["Name\_Disease"]);

}

}

//обновление типов потребления

private void UpdType(ComboBox cb)

{

SqlParameter[] SP = new SqlParameter[0];

TypeUse = DB.GetData("Select\_TypeUse", SP);

DataRow RW;

cb.Items.Clear();

cb.Text = "";

for (int i = 0; i < TypeUse.Rows.Count; i++)

{

RW = TypeUse.Rows[i];

cb.Items.Add(RW["Name\_Use"]);

}

}

private void CreateMedicamentLabel()

{

lab = new Label()

{

Name = "MedicamentLabel",

Text = "Лекарство: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locMedicamentLabel,

AutoSize = true,

};

FDiag.Controls.Add(lab);

}

private void CreateMedicamentCB()

{

cb = new ComboBox()

{

Name = "MedicamentCB",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locMedicamentCB,

Size = sizeMedicamentCB,

AutoCompleteMode = AutoCompleteMode.SuggestAppend,

AutoCompleteSource = AutoCompleteSource.ListItems,

};

SqlParameter[] SP = new SqlParameter[0];

Medicaments = DB.GetData("Select\_Medicament", SP);

DataRow RW;

for (int i = 0; i < Medicaments.Rows.Count; i++)

{

RW = Medicaments.Rows[i];

cb.Items.Add(RW[1]);

}

if (NameMeca != "")

{

DataRow[] DRMedi = Medicaments.Select("[Name\_Medicament] LIKE '%" + NameMeca + "%'");

string s = DRMedi[0]["Name\_Medicament"].ToString();

cb.Text = s;

}

FDiag.Controls.Add(cb);

}//id колонки

private void CreateDiseaseLabel()

{

lab = new Label()

{

Name = "DiseaseLabel",

Text = "Название болезни: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locDiseaseLabel,

AutoSize = true,

};

FDiag.Controls.Add(lab);

}

private void CreateDiseaseCB()

{

cb = new ComboBox()

{

Name = "DiseaseCB",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locDiseaseCB,

Size = sizeDiseaseCB,

AutoCompleteMode = AutoCompleteMode.SuggestAppend,

AutoCompleteSource = AutoCompleteSource.ListItems,

};

UpdDisease(cb);

if (NameDise != "")

{

DataRow[] DRDise = Disease.Select("[Name\_Disease] LIKE '%" + NameDise + "%'");

string s = DRDise[0]["Name\_Disease"].ToString();

cb.Text = s;

}

FDiag.Controls.Add(cb);

}

private void CreateDiseaseButton()

{

but = new Button()

{

Name = "DiseaseButton",

Text = "Изменить",

Font = new Font("Microsoft Sans Serif", 9),

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locDiseaseBut,

Size = sizeDiseaseBut,

Enabled = false,

};

FDiag.Controls.Add(but);

}

private void CreateTimeLabel()

{

lab = new Label()

{

Name = "TimeLabel",

Text = "Срок лечения: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locTimeLabel,

AutoSize = true,

};

FDiag.Controls.Add(lab);

}

private void CreateTimeTB()

{

Tb = new TextBox()

{

Name = "TimeTextBox",

Text = TimeHeal,

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locTimeTB,

Size = sizeTimeTB,

MaxLength = 3,

};

FDiag.Controls.Add(Tb);

}

private void CreateAmountLabel()

{

lab = new Label()

{

Name = "AmountLabel",

Text = "Кол-во лекарств: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locAmountLabel,

AutoSize = true,

};

FDiag.Controls.Add(lab);

}

private void CreateAmountTB()

{

Tb = new TextBox()

{

Name = "AmountTextBox",

Text = AmounMed,

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locAmountTB,

Size = sizeAmountTB,

MaxLength = 3,

};

FDiag.Controls.Add(Tb);

}

private void CreateRoomLabel()

{

lab = new Label()

{

Name = "RoomLabel",

Text = "Выбор палаты: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locRoomLabel,

AutoSize = true,

};

FDiag.Controls.Add(lab);

}

private void CreateRoomCB()

{

cb = new ComboBox()

{

Name = "RoomCB",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locRoomCB,

Size = sizeRoomCB,

AutoCompleteMode = AutoCompleteMode.SuggestAppend,

AutoCompleteSource = AutoCompleteSource.ListItems,

};

SqlParameter[] SP = new SqlParameter[0];

Deportaments = DB.GetData("Select\_Id\_Departament", SP);

DataRow RW;

for (int i = 0; i < Deportaments.Rows.Count; i++)

{

RW = Deportaments.Rows[i];

cb.Items.Add(RW["Терапевтическое отделение"]);

}

if (NumbDeprot != "")

{

DataRow[] DRDerp = Deportaments.Select("[Терапевтическое отделение] LIKE '%" + NumbDeprot + "%'");

string s = DRDerp[0]["Терапевтическое отделение"].ToString();

cb.Text = s;

}

FDiag.Controls.Add(cb);

}

private void CreateUseLabel()

{

lab = new Label()

{

Name = "UseLabel",

Text = "Тип потребления: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locUseLabel,

AutoSize = true,

};

FDiag.Controls.Add(lab);

}

private void CreateUseCB()

{

cb = new ComboBox()

{

Name = "UseCB",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locUseCB,

Size = sizeUseCB,

};

UpdType(cb);

if (TypeDise != "")

{

DataRow[] DRType = TypeUse.Select("[Name\_Use] LIKE '%" + TypeDise + "%'");

string s = DRType[0]["Name\_Use"].ToString();

cb.Text = s;

}

FDiag.Controls.Add(cb);

}

private void CreateUseBut()

{

but = new Button()

{

Name = "UseButton",

Text = "Создать",

Font = new Font("Microsoft Sans Serif", 8),

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locUseBut,

Size = sizeUseBut,

Enabled = false,

};

FDiag.Controls.Add(but);

}

private void CreateApplyButton()

{

but = new Button()

{

Name = "ApplyButton",

Text = CaptionButton,

Font = new Font("Microsoft Sans Serif", 9),

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locApplyBut,

Size = sizeApplyBut,

};

FDiag.Controls.Add(but);

}

private void CreateCanButton()

{

but = new Button()

{

Name = "CanButton",

Text = "Отмена",

Font = new Font("Microsoft Sans Serif", 9),

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locCanBut,

Size = sizeCanBut,

};

FDiag.Controls.Add(but);

}

}

}

DischargeForm.cs

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp9

{

class DiagForm

{

private DataTable Medicaments;

private DataTable Disease;

private DataTable Deportaments;

private DataTable TypeUse;

public string id\_Diag = "";

public string NameMeca = "";

public string NameDise = "";

public string TimeHeal = "";

public string AmounMed = "";

public string NumbDeprot = "";

public string TypeDise = "";

public string CaptionButton = "Создать";

private Form FDiag = new Form();

private Label lab = new Label();

private TextBox Tb = new TextBox();

private MaskedTextBox mtb = new MaskedTextBox();

private ComboBox cb = new ComboBox();

private Button but = new Button();

private Font DefFont = new Font("Microsoft Sans Serif", 10);

private Color DefBackColor = Color.FromName("Control");

private Color DefTextColor = Color.FromName("ControlText");

//Позиции //позиции

private Point locMedicamentLabel = new Point(20, 10); //label лекарств

private Point locMedicamentCB = new Point(20, 33); //ComboBox лекарств

private Point locDiseaseLabel = new Point(20, 63); //label болезней

private Point locDiseaseCB = new Point(20, 86); //ComboBox болезней

private Point locDiseaseBut = new Point(180, 86); //кнопка болезней

private Point locTimeLabel = new Point(20, 116); //label срок лечения

private Point locTimeTB = new Point(20, 139); //TextBox срок лечения

private Point locAmountLabel = new Point(20, 169); //label кол-во лекарств

private Point locAmountTB = new Point(20, 192); //textBox кол-во лекарств

private Point locRoomLabel = new Point(20, 222); //label выбор палаты

private Point locRoomCB = new Point(20, 245); //ComboBox выбор палаты

private Point locUseLabel = new Point(20, 275); //label тип потребления

private Point locUseCB = new Point(20, 298); //ComboBox тип потребления

private Point locUseBut = new Point(180, 298); //кнопка типов потребления

private Point locApplyBut = new Point(20, 338); //кнопка Создать

private Point locCanBut = new Point(150, 338); //кнопка Отмена

//размеры //размеры

private Size sizeMedicamentCB = new Size(200, 20); //ComboBox лекарств

private Size sizeDiseaseCB = new Size(150, 20); //ComboBox болезней

private Size sizeDiseaseBut = new Size(75, 25); //кнопка болезней

private Size sizeTimeTB = new Size(100, 20); //TextBox срок лечения

private Size sizeAmountTB = new Size(50, 20); //textBox кол-во лекарств

private Size sizeRoomCB = new Size(200, 20); //ComboBox выбор палаты

private Size sizeUseCB = new Size(150, 20); //ComboBox тип потребления

private Size sizeUseBut = new Size(75, 25); //кнопка типов потребления

private Size sizeApplyBut = new Size(100, 30); //кнопка Создать

private Size sizeCanBut = new Size(100, 30); //кнопка Отмена

public void NewDiagForm()

{

CreateMedicamentLabel();

CreateMedicamentCB();

CreateDiseaseLabel();

CreateDiseaseCB();

CreateDiseaseButton();

CreateTimeLabel();

CreateTimeTB();

CreateAmountLabel();

CreateAmountTB();

CreateRoomLabel();

CreateRoomCB();

CreateUseLabel();

CreateUseCB();

CreateUseBut();

CreateApplyButton();

CreateCanButton();

FDiag.Text = "Создание диагноза";

FDiag.Size = new Size(280, 425);

FDiag.FormBorderStyle = FormBorderStyle.FixedSingle;

FDiag.MaximizeBox = false;

FDiag.StartPosition = FormStartPosition.CenterScreen;

FDiag.FormClosing += new FormClosingEventHandler(FormClosing);

FDiag.Controls["CanButton"].Click += new EventHandler(CancelButtonClick);

if ((FDiag.Controls["ApplyButton"] as Button).Text == "Создать")

FDiag.Controls["ApplyButton"].Click += new EventHandler(ApplyButtonClick);

else

FDiag.Controls["ApplyButton"].Click += new EventHandler(DiagnozButtonClick);

FDiag.Controls["DiseaseCB"].TextChanged += DiseaseTextChange;

FDiag.Controls["DiseaseButton"].Click += DiseasiButton;

FDiag.Controls["UseCB"].TextChanged += TypeUseTextChange;

FDiag.Controls["UseButton"].Click += TypeButtonClick;

FDiag.ShowDialog();

}

//закрытие формы

private void FormClosing(object sender, EventArgs e)

{

(Application.OpenForms[0] as MainForm).UpdGrids(2);

}

//отмена

private void CancelButtonClick(object sender, EventArgs e)

{

DialogResult DR = MessageBox.Show("Вы уверены что хотите отменить создание?\n" +

"Все веденные данные будут потеряны.", "Закрыть?", MessageBoxButtons.YesNo);

if (DR == DialogResult.Yes)

FDiag.Close();

}

//кнопка создать

private void ApplyButtonClick(object sender, EventArgs e)

{

DataRow[] DRDerp = Deportaments.Select("[Терапевтическое отделение] = '"+ (FDiag.Controls["RoomCB"] as ComboBox).Text + "'");

DataRow[] DRType = TypeUse.Select("[Name\_Use] = '" + (FDiag.Controls["UseCB"] as ComboBox).Text + "'");

DataRow[] DRMedi = Medicaments.Select("[Name\_Medicament] = '" + (FDiag.Controls["MedicamentCB"] as ComboBox).Text + "'");

DataRow[] DRDise = Disease.Select("[Name\_Disease] = '" + (FDiag.Controls["DiseaseCB"] as ComboBox).Text + "'");

try

{

SqlParameter[] SP = new SqlParameter[6]

{

new SqlParameter ("@TimeDisease", (FDiag.Controls["TimeTextBox"] as TextBox).Text),

new SqlParameter ("@Amount", (FDiag.Controls["AmountTextBox"] as TextBox).Text),

new SqlParameter ("@ID\_Room", DRDerp[0]["id\_Room"]),

new SqlParameter ("@id\_TypeUse", DRType[0]["id\_TypeUse"]),

new SqlParameter ("@id\_Medicament", DRMedi[0]["id\_Medicament"]),

new SqlParameter ("@id\_Disease", DRDise[0]["id\_Disease"]),

};

DB.WriteData("Add\_Diagnosis", SP);

}

catch

{

MessageBox.Show("Неудалось добавить введенные данные.\nПожалуйста проверьте правильность введенных данных.");

}

if (DB.LastWrite)

FDiag.Close();

}

//кнопка изменить Диагноз

private void DiagnozButtonClick(object sender, EventArgs e)

{

DataRow[] DRDerp = Deportaments.Select("[Терапевтическое отделение] = '" + (FDiag.Controls["RoomCB"] as ComboBox).Text + "'");

DataRow[] DRType = TypeUse.Select("[Name\_Use] = '" + (FDiag.Controls["UseCB"] as ComboBox).Text + "'");

DataRow[] DRMedi = Medicaments.Select("[Name\_Medicament] = '" + (FDiag.Controls["MedicamentCB"] as ComboBox).Text + "'");

DataRow[] DRDise = Disease.Select("[Name\_Disease] = '" + (FDiag.Controls["DiseaseCB"] as ComboBox).Text + "'");

try

{

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@id\_Diagnoz", id\_Diag),

new SqlParameter("@TimeDisease", (FDiag.Controls["TimeTextBox"] as TextBox).Text),

new SqlParameter("@Amount", (FDiag.Controls["AmountTextBox"] as TextBox).Text),

new SqlParameter("@ID\_Room", DRDerp[0]["id\_Room"]),

new SqlParameter ("@id\_TypeUse", DRType[0]["id\_TypeUse"]),

new SqlParameter ("@id\_Medicament", DRMedi[0]["id\_Medicament"]),

new SqlParameter ("@id\_Disease", DRDise[0]["id\_Disease"]),

new SqlParameter("@Diagnosis\_Logical\_Deletebit", false),

};

DB.UpdData("UPD\_Diagnosis", SP);

}

catch

{

MessageBox.Show("Неудалось добавить введенные данные.\nПожалуйста проверьте правильность введенных данных.");

}

if (DB.LastWrite)

FDiag.Close();

}

//кнопка изменения болзени

private void DiseasiButton(object sender, EventArgs e)

{

switch ((sender as Button).Text)

{

case "Изменить":

{

DataRow[] DRDise = Disease.Select("[Name\_Disease] = '" + (FDiag.Controls["DiseaseCB"] as ComboBox).Text + "'");

EditDelForm.EditData = (FDiag.Controls["DiseaseCB"] as ComboBox).Text;

EditDelForm.NewMess("Изменить", "Удалить", 2);

if (EditDelForm.LastResult == "Изменить")

{

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@id\_Disease", DRDise[0]["id\_Disease"]),

new SqlParameter("@Name\_Disease", EditDelForm.EditData)

};

DB.UpdData("[UPD\_DirectoryDisease]", SP);

}

if (EditDelForm.LastResult == "Удалить")

{

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@id\_Disease", DRDise[0]["id\_Disease"]),

};

DB.LogDelData("[logdel\_DirectoryDisease]", SP);

}

break;

}

case "Создать":

{

string Text = (FDiag.Controls["DiseaseCB"] as ComboBox).Text;

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@Name\_Disease", Text),

};

DB.WriteData("[Add\_DirectoryDisease]", SP);

break;

}

}

UpdDisease((FDiag.Controls["DiseaseCB"] as ComboBox));

}

//кнопка создания типов потребления

private void TypeButtonClick(object sender, EventArgs e)

{

switch ((sender as Button).Text)

{

case "Изменить":

{

DataRow[] DRType = TypeUse.Select("[Name\_Use] = '" + (FDiag.Controls["UseCB"] as ComboBox).Text + "'");

EditDelForm.EditData = (FDiag.Controls["UseCB"] as ComboBox).Text;

EditDelForm.NewMess("Изменить", "Удалить", 2);

if (EditDelForm.LastResult == "Изменить")

{

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@id\_TypeUse", DRType[0]["id\_TypeUse"]),

new SqlParameter("@NameUse", EditDelForm.EditData)

};

DB.UpdData("[UPD\_TypeUse]", SP);

}

if (EditDelForm.LastResult == "Удалить")

{

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@id\_TypeUse", DRType[0]["id\_TypeUse"]),

};

DB.LogDelData("[logdel\_TypeUse]", SP);

}

break;

}

case "Создать":

{

string Text = (FDiag.Controls["UseCB"] as ComboBox).Text;

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@NameUse", Text),

};

DB.WriteData("[Add\_TypeUse]", SP);

break;

}

}

UpdType((FDiag.Controls["UseCB"] as ComboBox));

}

//Выбор болезни

private void DiseaseTextChange(object sender, EventArgs e)

{

string SText = (sender as ComboBox).Text;

FDiag.Controls["DiseaseButton"].Enabled = (SText != "");

if ((sender as ComboBox).SelectedIndex == -1)

{

FDiag.Controls["DiseaseButton"].Text = "Создать";

}

else

{

FDiag.Controls["DiseaseButton"].Text = "Изменить";

}

}

//выбор типов потреблениия

private void TypeUseTextChange(object sender, EventArgs e)

{

string SText = (sender as ComboBox).Text;

FDiag.Controls["UseButton"].Enabled = (SText != "");

if ((sender as ComboBox).SelectedIndex == -1)

{

FDiag.Controls["UseButton"].Text = "Создать";

}

else

{

FDiag.Controls["UseButton"].Text = "Изменить";

}

}

//обновление Болезней

private void UpdDisease(ComboBox cb)

{

SqlParameter[] SP = new SqlParameter[0];

Disease = DB.GetData("Select\_Name\_Disease", SP);

DataRow RW;

cb.Items.Clear();

cb.Text = "";

for (int i = 0; i < Disease.Rows.Count; i++)

{

RW = Disease.Rows[i];

cb.Items.Add(RW["Name\_Disease"]);

}

}

//обновление типов потребления

private void UpdType(ComboBox cb)

{

SqlParameter[] SP = new SqlParameter[0];

TypeUse = DB.GetData("Select\_TypeUse", SP);

DataRow RW;

cb.Items.Clear();

cb.Text = "";

for (int i = 0; i < TypeUse.Rows.Count; i++)

{

RW = TypeUse.Rows[i];

cb.Items.Add(RW["Name\_Use"]);

}

}

private void CreateMedicamentLabel()

{

lab = new Label()

{

Name = "MedicamentLabel",

Text = "Лекарство: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locMedicamentLabel,

AutoSize = true,

};

FDiag.Controls.Add(lab);

}

private void CreateMedicamentCB()

{

cb = new ComboBox()

{

Name = "MedicamentCB",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locMedicamentCB,

Size = sizeMedicamentCB,

AutoCompleteMode = AutoCompleteMode.SuggestAppend,

AutoCompleteSource = AutoCompleteSource.ListItems,

};

SqlParameter[] SP = new SqlParameter[0];

Medicaments = DB.GetData("Select\_Medicament", SP);

DataRow RW;

for (int i = 0; i < Medicaments.Rows.Count; i++)

{

RW = Medicaments.Rows[i];

cb.Items.Add(RW[1]);

}

if (NameMeca != "")

{

DataRow[] DRMedi = Medicaments.Select("[Name\_Medicament] LIKE '%" + NameMeca + "%'");

string s = DRMedi[0]["Name\_Medicament"].ToString();

cb.Text = s;

}

FDiag.Controls.Add(cb);

}//id колонки

private void CreateDiseaseLabel()

{

lab = new Label()

{

Name = "DiseaseLabel",

Text = "Название болезни: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locDiseaseLabel,

AutoSize = true,

};

FDiag.Controls.Add(lab);

}

private void CreateDiseaseCB()

{

cb = new ComboBox()

{

Name = "DiseaseCB",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locDiseaseCB,

Size = sizeDiseaseCB,

AutoCompleteMode = AutoCompleteMode.SuggestAppend,

AutoCompleteSource = AutoCompleteSource.ListItems,

};

UpdDisease(cb);

if (NameDise != "")

{

DataRow[] DRDise = Disease.Select("[Name\_Disease] LIKE '%" + NameDise + "%'");

string s = DRDise[0]["Name\_Disease"].ToString();

cb.Text = s;

}

FDiag.Controls.Add(cb);

}

private void CreateDiseaseButton()

{

but = new Button()

{

Name = "DiseaseButton",

Text = "Изменить",

Font = new Font("Microsoft Sans Serif", 9),

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locDiseaseBut,

Size = sizeDiseaseBut,

Enabled = false,

};

FDiag.Controls.Add(but);

}

private void CreateTimeLabel()

{

lab = new Label()

{

Name = "TimeLabel",

Text = "Срок лечения: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locTimeLabel,

AutoSize = true,

};

FDiag.Controls.Add(lab);

}

private void CreateTimeTB()

{

Tb = new TextBox()

{

Name = "TimeTextBox",

Text = TimeHeal,

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locTimeTB,

Size = sizeTimeTB,

MaxLength = 3,

};

FDiag.Controls.Add(Tb);

}

private void CreateAmountLabel()

{

lab = new Label()

{

Name = "AmountLabel",

Text = "Кол-во лекарств: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locAmountLabel,

AutoSize = true,

};

FDiag.Controls.Add(lab);

}

private void CreateAmountTB()

{

Tb = new TextBox()

{

Name = "AmountTextBox",

Text = AmounMed,

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locAmountTB,

Size = sizeAmountTB,

MaxLength = 3,

};

FDiag.Controls.Add(Tb);

}

private void CreateRoomLabel()

{

lab = new Label()

{

Name = "RoomLabel",

Text = "Выбор палаты: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locRoomLabel,

AutoSize = true,

};

FDiag.Controls.Add(lab);

}

private void CreateRoomCB()

{

cb = new ComboBox()

{

Name = "RoomCB",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locRoomCB,

Size = sizeRoomCB,

AutoCompleteMode = AutoCompleteMode.SuggestAppend,

AutoCompleteSource = AutoCompleteSource.ListItems,

};

SqlParameter[] SP = new SqlParameter[0];

Deportaments = DB.GetData("Select\_Id\_Departament", SP);

DataRow RW;

for (int i = 0; i < Deportaments.Rows.Count; i++)

{

RW = Deportaments.Rows[i];

cb.Items.Add(RW["Терапевтическое отделение"]);

}

if (NumbDeprot != "")

{

DataRow[] DRDerp = Deportaments.Select("[Терапевтическое отделение] LIKE '%" + NumbDeprot + "%'");

string s = DRDerp[0]["Терапевтическое отделение"].ToString();

cb.Text = s;

}

FDiag.Controls.Add(cb);

}

private void CreateUseLabel()

{

lab = new Label()

{

Name = "UseLabel",

Text = "Тип потребления: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locUseLabel,

AutoSize = true,

};

FDiag.Controls.Add(lab);

}

private void CreateUseCB()

{

cb = new ComboBox()

{

Name = "UseCB",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locUseCB,

Size = sizeUseCB,

};

UpdType(cb);

if (TypeDise != "")

{

DataRow[] DRType = TypeUse.Select("[Name\_Use] LIKE '%" + TypeDise + "%'");

string s = DRType[0]["Name\_Use"].ToString();

cb.Text = s;

}

FDiag.Controls.Add(cb);

}

private void CreateUseBut()

{

but = new Button()

{

Name = "UseButton",

Text = "Создать",

Font = new Font("Microsoft Sans Serif", 8),

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locUseBut,

Size = sizeUseBut,

Enabled = false,

};

FDiag.Controls.Add(but);

}

private void CreateApplyButton()

{

but = new Button()

{

Name = "ApplyButton",

Text = CaptionButton,

Font = new Font("Microsoft Sans Serif", 9),

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locApplyBut,

Size = sizeApplyBut,

};

FDiag.Controls.Add(but);

}

private void CreateCanButton()

{

but = new Button()

{

Name = "CanButton",

Text = "Отмена",

Font = new Font("Microsoft Sans Serif", 9),

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locCanBut,

Size = sizeCanBut,

};

FDiag.Controls.Add(but);

}

}

}

DocumentOutPutMedicament.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Word = Microsoft.Office.Interop.Word;

using Excel = Microsoft.Office.Interop.Excel;

using System.Windows.Forms;

namespace WindowsFormsApp9

{

class DocumentOutPut

{

private static Word.Application winword = new Word.Application(); //СОздаём COM-объект Word

private static object missing = System.Reflection.Missing.Value;

//талон записи на прием

public static void PrintWriteAppointment(int NumTalon, string FIO, string TypeWrite, string DateDischarge)

{

winword.Visible = false;

winword.Documents.Application.Caption = "Талон"; //заголовок документа

Word.Document wordDoc = winword.Documents.Add(ref missing, ref missing, ref missing, ref missing);

wordDoc.Paragraphs.Add();

Word.Paragraph Talon = wordDoc.Paragraphs.Add();

Talon.Range.Font.Name = "Times New Roman";

Talon.Range.Font.Size = 26;

Talon.Range.Font.Bold = 1;

Talon.Range.Text = "Талон";

Talon.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphCenter;

wordDoc.Paragraphs.Add();

Word.Paragraph NumTal = wordDoc.Paragraphs.Add();

NumTal.Range.Font.Name = "Times New Roman";

NumTal.Range.Font.Size = 14;

NumTal.Range.Text = "#" + NumTalon;

NumTal.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphCenter;

wordDoc.Paragraphs.Add();

Word.Paragraph Surname = wordDoc.Paragraphs.Add();

Surname.Range.Font.Name = "Times New Roman";

Surname.Range.Font.Size = 15;

Surname.Range.Text = "ФИО пациента: " + FIO;

Surname.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphLeft;

wordDoc.Paragraphs.Add();

Word.Paragraph Zapis = wordDoc.Paragraphs.Add();

Zapis.Range.Font.Name = "Times New Roman";

Zapis.Range.Font.Size = 12;

Zapis.Range.Text = "Записан с помощью:" + TypeWrite;

Zapis.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphRight;

wordDoc.Paragraphs.Add();

Word.Paragraph DateZapis = wordDoc.Paragraphs.Add();

Zapis.Range.Font.Name = "Times New Roman";

Zapis.Range.Font.Size = 10;

Zapis.Range.Text = "Дата записи: " + DateDischarge;

Zapis.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphRight;

winword.Visible = true;

}

//выписка на работу

public static void PrintWorkDischarge(string FIO, int numVip, string Naprav, string DateRojden,

string PassportInfo, string DiseaseInfo, string DatePostup, int SrokLech, string StatusLech,

string Vrach, string DateVIpiski)

{

winword.Visible = false;

winword.Documents.Application.Caption = "Рабочая выписка"; //заголовок документа

Word.Document wordDocVipiska = winword.Documents.Add(ref missing, ref missing, ref missing, ref missing);

wordDocVipiska.Paragraphs.Add();

Word.Paragraph Vipiska = wordDocVipiska.Paragraphs.Add();

Vipiska.Range.Font.Name = "Times New Roman";

Vipiska.Range.Font.Size = 25;

Vipiska.Range.Font.Bold = 1;

Vipiska.Range.Text = "Выписка # " + numVip;

Vipiska.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphCenter;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph VipiskaType = wordDocVipiska.Paragraphs.Add();

VipiskaType.Range.Font.Name = "Times New Roman";

VipiskaType.Range.Font.Size = 20;

VipiskaType.Range.Font.Bold = 1;

VipiskaType.Range.Text = "Направление: " + Naprav;

VipiskaType.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphCenter;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph Surname = wordDocVipiska.Paragraphs.Add();

Surname.Range.Font.Name = "Times New Roman";

Surname.Range.Font.Size = 14;

Surname.Range.Text = "ФИО пациента: " + FIO;

Surname.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphLeft;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph DateRojd = wordDocVipiska.Paragraphs.Add();

DateRojd.Range.Font.Name = "Times New Roman";

DateRojd.Range.Font.Size = 14;

DateRojd.Range.Text = "Дата рождения: " + DateRojden + " (" + "28" + " лет)";

DateRojd.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphLeft;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph Passport = wordDocVipiska.Paragraphs.Add();

Passport.Range.Font.Name = "Times New Roman";

Passport.Range.Font.Size = 14;

Passport.Range.Text = "Паспортные данные: " + PassportInfo;

Passport.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphLeft;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph Disease = wordDocVipiska.Paragraphs.Add();

Disease.Range.Font.Name = "Times New Roman";

Disease.Range.Font.Size = 16;

Disease.Range.Text = "Заболевание: " + DiseaseInfo;

Disease.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphCenter;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph InfoDisease = wordDocVipiska.Paragraphs.Add();

InfoDisease.Range.Font.Name = "Times New Roman";

InfoDisease.Range.Font.Size = 14;

InfoDisease.Range.Text = "Дата поступления: " + DatePostup + ", Срок лечения в стационаре: "

+ SrokLech + " суток," + " Статус лечения: " + StatusLech;

InfoDisease.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphLeft;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph Doctor = wordDocVipiska.Paragraphs.Add();

Doctor.Range.Font.Name = "Times New Roman";

Doctor.Range.Font.Size = 12;

Doctor.Range.Text = "Лечащий врач: " + Vrach;

Doctor.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphLeft;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph DateOutDisease = wordDocVipiska.Paragraphs.Add();

DateOutDisease.Range.Font.Name = "Times New Roman";

DateOutDisease.Range.Font.Size = 12;

DateOutDisease.Range.Text = "Подпись лечащего врача \_\_\_\_\_\_\_\_\_\_\_ " + " "

+ "Дата выписки: " + DateVIpiski;

DateOutDisease.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphRight;

winword.Visible = true;

}

//Выписка в учебное заведение

public static void PrintSchoolDischarge(string FIO, int numVip, string Naprav, string DateRojden,

string DiseaseInfo, string DatePostup, int SrokLech, string StatusLech, string DateVIpiski)

{

winword.Visible = false;

winword.Documents.Application.Caption = "Выписка в учебное заведение"; //заголовок документа

Word.Document wordDocVipiska = winword.Documents.Add(ref missing, ref missing, ref missing, ref missing);

wordDocVipiska.Paragraphs.Add();

Word.Paragraph Vipiska = wordDocVipiska.Paragraphs.Add();

Vipiska.Range.Font.Name = "Times New Roman";

Vipiska.Range.Font.Size = 25;

Vipiska.Range.Font.Bold = 1;

Vipiska.Range.Text = "Выписка # " + numVip;

Vipiska.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphCenter;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph VipiskaType = wordDocVipiska.Paragraphs.Add();

VipiskaType.Range.Font.Name = "Times New Roman";

VipiskaType.Range.Font.Size = 20;

VipiskaType.Range.Font.Bold = 1;

VipiskaType.Range.Text = "Направление: " + Naprav;

VipiskaType.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphCenter;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph Surname = wordDocVipiska.Paragraphs.Add();

Surname.Range.Font.Name = "Times New Roman";

Surname.Range.Font.Size = 14;

Surname.Range.Text = "ФИО пациента: " + FIO;

Surname.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphLeft;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph DateRojd = wordDocVipiska.Paragraphs.Add();

DateRojd.Range.Font.Name = "Times New Roman";

DateRojd.Range.Font.Size = 14;

DateRojd.Range.Text = "Дата рождения: " + DateRojden + " (" + "28" + " лет)";

DateRojd.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphLeft;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph Disease = wordDocVipiska.Paragraphs.Add();

Disease.Range.Font.Name = "Times New Roman";

Disease.Range.Font.Size = 16;

Disease.Range.Text = "Заболевание: " + DiseaseInfo;

Disease.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphCenter;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph InfoDisease = wordDocVipiska.Paragraphs.Add();

InfoDisease.Range.Font.Name = "Times New Roman";

InfoDisease.Range.Font.Size = 14;

InfoDisease.Range.Text = "Дата поступления: " + DatePostup + ", Срок лечения в стационаре: "

+ SrokLech + " суток," + " Статус лечения: " + StatusLech;

InfoDisease.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphLeft;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph DateOutDisease = wordDocVipiska.Paragraphs.Add();

DateOutDisease.Range.Font.Name = "Times New Roman";

DateOutDisease.Range.Font.Size = 12;

DateOutDisease.Range.Text = "Дата выписки: " + DateVIpiski;

DateOutDisease.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphRight;

winword.Visible = true;

}

//печать приёмки товара

public static void PrintDelivery(string[,] Data)

{

string[,] SM = new string[20, 9];

//заполнение массива цифрами

int k = 0;

for (int i = 0; i < SM.GetLength(0); i++)

for (int j = 0; j < SM.GetLength(1); j++)

{

k++;

SM[i, j] = k.ToString();

}

//Объявляем приложение

Excel.Application excelApp = new Excel.Application();

//Количество листов в рабочей книге

excelApp.SheetsInNewWorkbook = 1;

//Добавить рабочую книгу

Excel.Workbook workBook = excelApp.Workbooks.Add(Type.Missing);

//Отключить отображение окон с сообщениями

excelApp.DisplayAlerts = false;

//Получаем первый лист документа (счет начинается с 1)

Excel.Worksheet sheet = (Excel.Worksheet)excelApp.Worksheets.get\_Item(1);

//Название листа (вкладки снизу)

sheet.Name = "Приёмка медикаментов";

//Ячейка с названием

Excel.Range range1 = sheet.get\_Range("A1:I1");

range1.Merge(Type.Missing);

range1.Cells.Font.Name = "Tahoma";

range1.Cells.HorizontalAlignment = Excel.Constants.xlCenter;

range1.Cells.Font.Size = 18;

range1.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

//Ячейка производитель

Excel.Range range2 = sheet.get\_Range("A2:A2");

range2.Merge(Type.Missing);

range2.Columns.ColumnWidth = 30;

range2.Cells.Font.Name = "Tahoma";

range2.Cells.HorizontalAlignment = Excel.Constants.xlCenter;

range2.Cells.Font.Size = 12;

range2.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

//Ячейка лекарства

Excel.Range range3 = sheet.get\_Range("B2:B2");

range3.Merge(Type.Missing);

range3.Columns.ColumnWidth = 30;

range3.Cells.Font.Name = "Tahoma";

range3.Cells.HorizontalAlignment = Excel.Constants.xlCenter;

range3.Cells.Font.Size = 12;

range3.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

//Ячейка категории

Excel.Range range4 = sheet.get\_Range("C2:C2");

range4.Merge(Type.Missing);

range4.Columns.ColumnWidth = 15;

range4.Cells.Font.Name = "Tahoma";

range4.Cells.HorizontalAlignment = Excel.Constants.xlCenter;

range4.Cells.Font.Size = 12;

range4.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

//Ячейка поступившего количества

Excel.Range range5 = sheet.get\_Range("D2:D2");

range5.Merge(Type.Missing);

range5.Columns.ColumnWidth = 15;

range5.Cells.Font.Name = "Tahoma";

range5.Cells.HorizontalAlignment = Excel.Constants.xlCenter;

range5.Cells.Font.Size = 12;

range5.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

//Ячейка дата поставки

Excel.Range range6 = sheet.get\_Range("E2:E2");

range6.Merge(Type.Missing);

range6.Columns.ColumnWidth = 18;

range6.Cells.Font.Name = "Tahoma";

range6.Cells.HorizontalAlignment = Excel.Constants.xlCenter;

range6.Cells.Font.Size = 12;

range6.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

//Ячейка сотрудника

Excel.Range range7 = sheet.get\_Range("F2:F2");

range7.Merge(Type.Missing);

range7.Columns.ColumnWidth = 15;

range7.Cells.Font.Name = "Tahoma";

range7.Cells.HorizontalAlignment = Excel.Constants.xlCenter;

range7.Cells.Font.Size = 12;

range7.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

//Ячейка склада

Excel.Range range8 = sheet.get\_Range("G2:G2");

range8.Merge(Type.Missing);

range8.Columns.ColumnWidth = 18;

range8.Cells.Font.Name = "Tahoma";

range8.Cells.HorizontalAlignment = Excel.Constants.xlCenter;

range8.Cells.Font.Size = 12;

range8.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

//Ячейка общего места на складе

Excel.Range range9 = sheet.get\_Range("H2:H2");

range9.Merge(Type.Missing);

range9.Columns.ColumnWidth = 15;

range9.Cells.Font.Name = "Tahoma";

range9.Cells.HorizontalAlignment = Excel.Constants.xlCenter;

range9.Cells.Font.Size = 12;

range9.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

//Ячейка общего места на складе

Excel.Range range10 = sheet.get\_Range("I2:I2");

range10.Merge(Type.Missing);

range10.Columns.ColumnWidth = 15;

range10.Cells.Font.Name = "Tahoma";

range10.Cells.HorizontalAlignment = Excel.Constants.xlCenter;

range10.Cells.Font.Size = 12;

range10.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

//Заполнение ячеек

range1.Cells[1, 1] = String.Format("Поставки медикаментов");

range2.Cells[1, 1] = String.Format("Производитель");

range3.Cells[1, 1] = String.Format("Медикамент");

range4.Cells[1, 1] = String.Format("Категория");

range5.Cells[1, 1] = String.Format("Поступило");

range6.Cells[1, 1] = String.Format("Дата поставки");

range7.Cells[1, 1] = String.Format("Сотрудник");

range8.Cells[1, 1] = String.Format("Ячейка склада");

range9.Cells[1, 1] = String.Format("Места всего");

range10.Cells[1, 1] = String.Format("Занято места");

//Ячейка заполнения производителей

Excel.Range range11 = sheet.get\_Range("A3:I999");

range11.Columns.ColumnWidth = 15;

range11.Cells.Font.Name = "Tahoma";

range11.Cells.HorizontalAlignment = Excel.Constants.xlCenter;

range11.Cells.Font.Size = 10;

range11.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

for (int i = 0; i < SM.GetLength(0); i++)

{

for (int j = 0; j < SM.GetLength(1); j++)

{

range11.Cells[i+1, j+1] = SM[i,j];

}

}

excelApp.Visible = true;

excelApp.UserControl = true;

}

}

}

EditDelForm.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Word = Microsoft.Office.Interop.Word;

using Excel = Microsoft.Office.Interop.Excel;

using System.Windows.Forms;

namespace WindowsFormsApp9

{

class DocumentOutPut

{

private static Word.Application winword = new Word.Application(); //СОздаём COM-объект Word

private static object missing = System.Reflection.Missing.Value;

//талон записи на прием

public static void PrintWriteAppointment(int NumTalon, string FIO, string TypeWrite, string DateDischarge)

{

winword.Visible = false;

winword.Documents.Application.Caption = "Талон"; //заголовок документа

Word.Document wordDoc = winword.Documents.Add(ref missing, ref missing, ref missing, ref missing);

wordDoc.Paragraphs.Add();

Word.Paragraph Talon = wordDoc.Paragraphs.Add();

Talon.Range.Font.Name = "Times New Roman";

Talon.Range.Font.Size = 26;

Talon.Range.Font.Bold = 1;

Talon.Range.Text = "Талон";

Talon.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphCenter;

wordDoc.Paragraphs.Add();

Word.Paragraph NumTal = wordDoc.Paragraphs.Add();

NumTal.Range.Font.Name = "Times New Roman";

NumTal.Range.Font.Size = 14;

NumTal.Range.Text = "#" + NumTalon;

NumTal.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphCenter;

wordDoc.Paragraphs.Add();

Word.Paragraph Surname = wordDoc.Paragraphs.Add();

Surname.Range.Font.Name = "Times New Roman";

Surname.Range.Font.Size = 15;

Surname.Range.Text = "ФИО пациента: " + FIO;

Surname.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphLeft;

wordDoc.Paragraphs.Add();

Word.Paragraph Zapis = wordDoc.Paragraphs.Add();

Zapis.Range.Font.Name = "Times New Roman";

Zapis.Range.Font.Size = 12;

Zapis.Range.Text = "Записан с помощью:" + TypeWrite;

Zapis.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphRight;

wordDoc.Paragraphs.Add();

Word.Paragraph DateZapis = wordDoc.Paragraphs.Add();

Zapis.Range.Font.Name = "Times New Roman";

Zapis.Range.Font.Size = 10;

Zapis.Range.Text = "Дата записи: " + DateDischarge;

Zapis.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphRight;

winword.Visible = true;

}

//выписка на работу

public static void PrintWorkDischarge(string FIO, int numVip, string Naprav, string DateRojden,

string PassportInfo, string DiseaseInfo, string DatePostup, int SrokLech, string StatusLech,

string Vrach, string DateVIpiski)

{

winword.Visible = false;

winword.Documents.Application.Caption = "Рабочая выписка"; //заголовок документа

Word.Document wordDocVipiska = winword.Documents.Add(ref missing, ref missing, ref missing, ref missing);

wordDocVipiska.Paragraphs.Add();

Word.Paragraph Vipiska = wordDocVipiska.Paragraphs.Add();

Vipiska.Range.Font.Name = "Times New Roman";

Vipiska.Range.Font.Size = 25;

Vipiska.Range.Font.Bold = 1;

Vipiska.Range.Text = "Выписка # " + numVip;

Vipiska.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphCenter;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph VipiskaType = wordDocVipiska.Paragraphs.Add();

VipiskaType.Range.Font.Name = "Times New Roman";

VipiskaType.Range.Font.Size = 20;

VipiskaType.Range.Font.Bold = 1;

VipiskaType.Range.Text = "Направление: " + Naprav;

VipiskaType.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphCenter;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph Surname = wordDocVipiska.Paragraphs.Add();

Surname.Range.Font.Name = "Times New Roman";

Surname.Range.Font.Size = 14;

Surname.Range.Text = "ФИО пациента: " + FIO;

Surname.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphLeft;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph DateRojd = wordDocVipiska.Paragraphs.Add();

DateRojd.Range.Font.Name = "Times New Roman";

DateRojd.Range.Font.Size = 14;

DateRojd.Range.Text = "Дата рождения: " + DateRojden + " (" + "28" + " лет)";

DateRojd.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphLeft;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph Passport = wordDocVipiska.Paragraphs.Add();

Passport.Range.Font.Name = "Times New Roman";

Passport.Range.Font.Size = 14;

Passport.Range.Text = "Паспортные данные: " + PassportInfo;

Passport.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphLeft;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph Disease = wordDocVipiska.Paragraphs.Add();

Disease.Range.Font.Name = "Times New Roman";

Disease.Range.Font.Size = 16;

Disease.Range.Text = "Заболевание: " + DiseaseInfo;

Disease.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphCenter;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph InfoDisease = wordDocVipiska.Paragraphs.Add();

InfoDisease.Range.Font.Name = "Times New Roman";

InfoDisease.Range.Font.Size = 14;

InfoDisease.Range.Text = "Дата поступления: " + DatePostup + ", Срок лечения в стационаре: "

+ SrokLech + " суток," + " Статус лечения: " + StatusLech;

InfoDisease.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphLeft;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph Doctor = wordDocVipiska.Paragraphs.Add();

Doctor.Range.Font.Name = "Times New Roman";

Doctor.Range.Font.Size = 12;

Doctor.Range.Text = "Лечащий врач: " + Vrach;

Doctor.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphLeft;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph DateOutDisease = wordDocVipiska.Paragraphs.Add();

DateOutDisease.Range.Font.Name = "Times New Roman";

DateOutDisease.Range.Font.Size = 12;

DateOutDisease.Range.Text = "Подпись лечащего врача \_\_\_\_\_\_\_\_\_\_\_ " + " "

+ "Дата выписки: " + DateVIpiski;

DateOutDisease.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphRight;

winword.Visible = true;

}

//Выписка в учебное заведение

public static void PrintSchoolDischarge(string FIO, int numVip, string Naprav, string DateRojden,

string DiseaseInfo, string DatePostup, int SrokLech, string StatusLech, string DateVIpiski)

{

winword.Visible = false;

winword.Documents.Application.Caption = "Выписка в учебное заведение"; //заголовок документа

Word.Document wordDocVipiska = winword.Documents.Add(ref missing, ref missing, ref missing, ref missing);

wordDocVipiska.Paragraphs.Add();

Word.Paragraph Vipiska = wordDocVipiska.Paragraphs.Add();

Vipiska.Range.Font.Name = "Times New Roman";

Vipiska.Range.Font.Size = 25;

Vipiska.Range.Font.Bold = 1;

Vipiska.Range.Text = "Выписка # " + numVip;

Vipiska.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphCenter;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph VipiskaType = wordDocVipiska.Paragraphs.Add();

VipiskaType.Range.Font.Name = "Times New Roman";

VipiskaType.Range.Font.Size = 20;

VipiskaType.Range.Font.Bold = 1;

VipiskaType.Range.Text = "Направление: " + Naprav;

VipiskaType.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphCenter;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph Surname = wordDocVipiska.Paragraphs.Add();

Surname.Range.Font.Name = "Times New Roman";

Surname.Range.Font.Size = 14;

Surname.Range.Text = "ФИО пациента: " + FIO;

Surname.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphLeft;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph DateRojd = wordDocVipiska.Paragraphs.Add();

DateRojd.Range.Font.Name = "Times New Roman";

DateRojd.Range.Font.Size = 14;

DateRojd.Range.Text = "Дата рождения: " + DateRojden + " (" + "28" + " лет)";

DateRojd.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphLeft;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph Disease = wordDocVipiska.Paragraphs.Add();

Disease.Range.Font.Name = "Times New Roman";

Disease.Range.Font.Size = 16;

Disease.Range.Text = "Заболевание: " + DiseaseInfo;

Disease.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphCenter;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph InfoDisease = wordDocVipiska.Paragraphs.Add();

InfoDisease.Range.Font.Name = "Times New Roman";

InfoDisease.Range.Font.Size = 14;

InfoDisease.Range.Text = "Дата поступления: " + DatePostup + ", Срок лечения в стационаре: "

+ SrokLech + " суток," + " Статус лечения: " + StatusLech;

InfoDisease.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphLeft;

wordDocVipiska.Paragraphs.Add();

Word.Paragraph DateOutDisease = wordDocVipiska.Paragraphs.Add();

DateOutDisease.Range.Font.Name = "Times New Roman";

DateOutDisease.Range.Font.Size = 12;

DateOutDisease.Range.Text = "Дата выписки: " + DateVIpiski;

DateOutDisease.Range.Paragraphs.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphRight;

winword.Visible = true;

}

//печать приёмки товара

public static void PrintDelivery(string[,] Data)

{

string[,] SM = new string[20, 9];

//заполнение массива цифрами

int k = 0;

for (int i = 0; i < SM.GetLength(0); i++)

for (int j = 0; j < SM.GetLength(1); j++)

{

k++;

SM[i, j] = k.ToString();

}

//Объявляем приложение

Excel.Application excelApp = new Excel.Application();

//Количество листов в рабочей книге

excelApp.SheetsInNewWorkbook = 1;

//Добавить рабочую книгу

Excel.Workbook workBook = excelApp.Workbooks.Add(Type.Missing);

//Отключить отображение окон с сообщениями

excelApp.DisplayAlerts = false;

//Получаем первый лист документа (счет начинается с 1)

Excel.Worksheet sheet = (Excel.Worksheet)excelApp.Worksheets.get\_Item(1);

//Название листа (вкладки снизу)

sheet.Name = "Приёмка медикаментов";

//Ячейка с названием

Excel.Range range1 = sheet.get\_Range("A1:I1");

range1.Merge(Type.Missing);

range1.Cells.Font.Name = "Tahoma";

range1.Cells.HorizontalAlignment = Excel.Constants.xlCenter;

range1.Cells.Font.Size = 18;

range1.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

//Ячейка производитель

Excel.Range range2 = sheet.get\_Range("A2:A2");

range2.Merge(Type.Missing);

range2.Columns.ColumnWidth = 30;

range2.Cells.Font.Name = "Tahoma";

range2.Cells.HorizontalAlignment = Excel.Constants.xlCenter;

range2.Cells.Font.Size = 12;

range2.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

//Ячейка лекарства

Excel.Range range3 = sheet.get\_Range("B2:B2");

range3.Merge(Type.Missing);

range3.Columns.ColumnWidth = 30;

range3.Cells.Font.Name = "Tahoma";

range3.Cells.HorizontalAlignment = Excel.Constants.xlCenter;

range3.Cells.Font.Size = 12;

range3.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

//Ячейка категории

Excel.Range range4 = sheet.get\_Range("C2:C2");

range4.Merge(Type.Missing);

range4.Columns.ColumnWidth = 15;

range4.Cells.Font.Name = "Tahoma";

range4.Cells.HorizontalAlignment = Excel.Constants.xlCenter;

range4.Cells.Font.Size = 12;

range4.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

//Ячейка поступившего количества

Excel.Range range5 = sheet.get\_Range("D2:D2");

range5.Merge(Type.Missing);

range5.Columns.ColumnWidth = 15;

range5.Cells.Font.Name = "Tahoma";

range5.Cells.HorizontalAlignment = Excel.Constants.xlCenter;

range5.Cells.Font.Size = 12;

range5.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

//Ячейка дата поставки

Excel.Range range6 = sheet.get\_Range("E2:E2");

range6.Merge(Type.Missing);

range6.Columns.ColumnWidth = 18;

range6.Cells.Font.Name = "Tahoma";

range6.Cells.HorizontalAlignment = Excel.Constants.xlCenter;

range6.Cells.Font.Size = 12;

range6.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

//Ячейка сотрудника

Excel.Range range7 = sheet.get\_Range("F2:F2");

range7.Merge(Type.Missing);

range7.Columns.ColumnWidth = 15;

range7.Cells.Font.Name = "Tahoma";

range7.Cells.HorizontalAlignment = Excel.Constants.xlCenter;

range7.Cells.Font.Size = 12;

range7.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

//Ячейка склада

Excel.Range range8 = sheet.get\_Range("G2:G2");

range8.Merge(Type.Missing);

range8.Columns.ColumnWidth = 18;

range8.Cells.Font.Name = "Tahoma";

range8.Cells.HorizontalAlignment = Excel.Constants.xlCenter;

range8.Cells.Font.Size = 12;

range8.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

//Ячейка общего места на складе

Excel.Range range9 = sheet.get\_Range("H2:H2");

range9.Merge(Type.Missing);

range9.Columns.ColumnWidth = 15;

range9.Cells.Font.Name = "Tahoma";

range9.Cells.HorizontalAlignment = Excel.Constants.xlCenter;

range9.Cells.Font.Size = 12;

range9.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

//Ячейка общего места на складе

Excel.Range range10 = sheet.get\_Range("I2:I2");

range10.Merge(Type.Missing);

range10.Columns.ColumnWidth = 15;

range10.Cells.Font.Name = "Tahoma";

range10.Cells.HorizontalAlignment = Excel.Constants.xlCenter;

range10.Cells.Font.Size = 12;

range10.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

//Заполнение ячеек

range1.Cells[1, 1] = String.Format("Поставки медикаментов");

range2.Cells[1, 1] = String.Format("Производитель");

range3.Cells[1, 1] = String.Format("Медикамент");

range4.Cells[1, 1] = String.Format("Категория");

range5.Cells[1, 1] = String.Format("Поступило");

range6.Cells[1, 1] = String.Format("Дата поставки");

range7.Cells[1, 1] = String.Format("Сотрудник");

range8.Cells[1, 1] = String.Format("Ячейка склада");

range9.Cells[1, 1] = String.Format("Места всего");

range10.Cells[1, 1] = String.Format("Занято места");

//Ячейка заполнения производителей

Excel.Range range11 = sheet.get\_Range("A3:I999");

range11.Columns.ColumnWidth = 15;

range11.Cells.Font.Name = "Tahoma";

range11.Cells.HorizontalAlignment = Excel.Constants.xlCenter;

range11.Cells.Font.Size = 10;

range11.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

for (int i = 0; i < SM.GetLength(0); i++)

{

for (int j = 0; j < SM.GetLength(1); j++)

{

range11.Cells[i+1, j+1] = SM[i,j];

}

}

excelApp.Visible = true;

excelApp.UserControl = true;

}

}

}

MedicamentForm.cs

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp9

{

class MedicamentsForm

{

private Form FMedic = new Form();

private Label lab = new Label();

private TextBox Tb = new TextBox();

private MaskedTextBox mtb = new MaskedTextBox();

private Button but = new Button();

private ComboBox cb = new ComboBox();

private Font DefFont = new Font("Microsoft Sans Serif", 10);

private Color DefBackColor = Color.FromName("Control");

private Color DefTextColor = Color.FromName("ControlText");

//Позиции //позиции

private Point locNameLabel = new Point(20, 10); //label название лекарства

private Point locNameTB = new Point(20, 33); //textBox название лекарства

private Point locCategoryLabel = new Point(220, 10); //label категории лекарств

private Point locCategoryCB = new Point(220, 33); //ComboBox категории лекарств

private Point locManufLabel = new Point(130, 63); //label выбора производителя

private Point locManufCB = new Point(130, 86); //ComboBox выбора производителя

private Point locNewManufLabel = new Point(20, 116); //label создания производителя

private Point locNewManufTB = new Point(20, 149); //textBox создания производителя

private Point locMailLabel = new Point(220, 116); //label Почта

private Point locMailTB = new Point(220, 149); //textBox Почта

private Point locAddressLabel = new Point(20, 179); //label Адрес

private Point locAddressTB = new Point(20, 202); //textBox Адрес

private Point locNewMaufBut = new Point(220, 200); //кнопка добавления производителя

private Point locMedicBut = new Point(60, 240); //кнопка добавления медикаментов

private Point locCanBut = new Point(230, 240); //кнопка отмены

//размеры //размеры

private Size sizeNameTB = new Size(150, 20); //textBox название лекарства

private Size sizeCategoryCB = new Size(150, 20); //ComboBox категории лекарств

private Size sizeManufCB = new Size(150, 20); //ComboBox выбора производителя

private Size sizeNewManufTB = new Size(150, 20); //textBox создания производителя

private Size sizeMailTB = new Size(150, 20); //textBox Почта

private Size sizeAddressTB = new Size(150, 20); //textBox Адрес

private Size sizeNewMaufBut = new Size(100, 28); //кнопка добавления производителя

private Size sizeMedicBut = new Size(100, 30); //кнопка добавления медикаментов

private Size sizeCanBut = new Size(100, 30); //кнопка отмены

public MedicamentsForm()

{

}

public void NewMedicamentsForm()

{

CreateNameLabel();

CreateNameTB();

CreateCategoryLabel();

CreateCategoryCB();

CreateManufLabel();

CreateManufCB();

CreateNewManufLabel();

CreateNewManufTB();

CreateMailLabel();

CreateMailTB();

CreateAddressLabel();

CreateAddressTB();

CreateNewMaufButton();

CreateMedicButton();

CreateCancelButton();

FMedic.Text = "Добавление медикамента";

FMedic.Size = new Size(405, 325);

FMedic.FormBorderStyle = FormBorderStyle.FixedSingle;

FMedic.MaximizeBox = false;

FMedic.StartPosition = FormStartPosition.CenterScreen;

FMedic.FormClosing += new FormClosingEventHandler(FormClosing);

FMedic.Controls["CancelButton"].Click += new EventHandler(CancelButtonClick);

FMedic.Controls["NewMaufButton"].Click += new EventHandler(NewManufButtonClick);

FMedic.Controls["MedicButton"].Click += new EventHandler(ApplayButtonClick);

FMedic.Show();

}

//закрытие формы

private void FormClosing(object sender, EventArgs e)

{

Application.OpenForms[0].Enabled = true;

}

//кнопка отмены

private void CancelButtonClick(object sender, EventArgs e)

{

DialogResult DR = MessageBox.Show("Вы уверены что хотите отменить создание?\n" +

"Все веденные данные будут потеряны.", "Закрыть?", MessageBoxButtons.YesNo);

if (DR == DialogResult.Yes)

FMedic.Close();

}

//кнопка добавить производителя

private void NewManufButtonClick(object sender, EventArgs e)

{

SqlConnection sql = new SqlConnection(DataBaseConfiguration.connectString);

sql.Open();

SqlCommand AddManuf = new SqlCommand("Add\_Manufacturer", sql);

AddManuf.CommandType = CommandType.StoredProcedure;

AddManuf.Parameters.AddWithValue("@N\_Manufacturer", (FMedic.Controls["NewManufTextBox"] as TextBox).Text);

AddManuf.Parameters.AddWithValue("@Adress", (FMedic.Controls["AddressTextBox"] as TextBox).Text);

AddManuf.Parameters.AddWithValue("@Mail", (FMedic.Controls["MailTextBox"] as TextBox).Text);

AddManuf.Parameters.AddWithValue("@Manufacturer\_Logical\_Delete", false);

AddManuf.ExecuteNonQuery();

sql.Close();

}

//кнопка создать лекарство

private void ApplayButtonClick(object sender, EventArgs e)

{

SqlConnection sql = new SqlConnection(DataBaseConfiguration.connectString);

sql.Open();

SqlCommand AddMedi = new SqlCommand("Add\_Medicamen]", sql);

AddMedi.CommandType = CommandType.StoredProcedure;

AddMedi.Parameters.AddWithValue("@Name\_Medicament", (FMedic.Controls["NameTextBox"] as TextBox).Text);

//AddMedi.Parameters.AddWithValue("@id\_Manufacturer", );

//AddMedi.Parameters.AddWithValue("@id\_CategoryOfMedicament", );

AddMedi.Parameters.AddWithValue("@Medicament\_Logical\_Delete", false);

AddMedi.ExecuteNonQuery();

sql.Close();

}

private void CreateNameLabel()

{

lab = new Label()

{

Name = "NameLabel",

Text = "Название лекарства: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locNameLabel,

AutoSize = true,

};

FMedic.Controls.Add(lab);

}

private void CreateNameTB()

{

Tb = new TextBox()

{

Name = "NameTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locNameTB,

Size = sizeNameTB,

};

FMedic.Controls.Add(Tb);

}

private void CreateCategoryLabel()

{

lab = new Label()

{

Name = "CategoryLabel",

Text = "Категория: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locCategoryLabel,

AutoSize = true,

};

FMedic.Controls.Add(lab);

}

private void CreateCategoryCB()

{

cb = new ComboBox()

{

Name = "CategoryCB",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locCategoryCB,

Size = sizeCategoryCB,

AutoCompleteMode = AutoCompleteMode.SuggestAppend,

AutoCompleteSource = AutoCompleteSource.ListItems,

};

FMedic.Controls.Add(cb);

}

private void CreateManufLabel()

{

lab = new Label()

{

Name = "ManufLabel",

Text = "Производитель: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locManufLabel,

AutoSize = true,

};

FMedic.Controls.Add(lab);

}

private void CreateManufCB()

{

cb = new ComboBox()

{

Name = "ManufCB",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locManufCB,

Size = sizeManufCB,

AutoCompleteMode = AutoCompleteMode.SuggestAppend,

AutoCompleteSource = AutoCompleteSource.ListItems,

};

FMedic.Controls.Add(cb);

}

private void CreateNewManufLabel()

{

lab = new Label()

{

Name = "NewManufLabel",

Text = "Название производителя: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locNewManufLabel,

AutoSize = true,

};

FMedic.Controls.Add(lab);

}

private void CreateNewManufTB()

{

Tb = new TextBox()

{

Name = "NewManufTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locNewManufTB,

Size = sizeNewManufTB,

};

FMedic.Controls.Add(Tb);

}

private void CreateMailLabel()

{

lab = new Label()

{

Name = "MailLabel",

Text = "Почта: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locMailLabel,

AutoSize = true,

};

FMedic.Controls.Add(lab);

}

private void CreateMailTB()

{

Tb = new TextBox()

{

Name = "MailTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locMailTB,

Size = sizeMailTB,

};

FMedic.Controls.Add(Tb);

}

private void CreateAddressLabel()

{

lab = new Label()

{

Name = "AddressLabel",

Text = "Адрес: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locAddressLabel,

AutoSize = true,

};

FMedic.Controls.Add(lab);

}

private void CreateAddressTB()

{

Tb = new TextBox()

{

Name = "AddressTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locAddressTB,

Size = sizeAddressTB,

};

FMedic.Controls.Add(Tb);

}

private void CreateNewMaufButton()

{

but = new Button()

{

Name = "NewMaufButton",

Text = "Создать",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locNewMaufBut,

Size = sizeNewMaufBut,

};

FMedic.Controls.Add(but);

}

private void CreateMedicButton()

{

but = new Button()

{

Name = "MedicButton",

Text = "Добавить лекарство",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locMedicBut,

Size = sizeMedicBut,

};

FMedic.Controls.Add(but);

}

private void CreateCancelButton()

{

but = new Button()

{

Name = "CancelButton",

Text = "Отмена",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locCanBut,

Size = sizeCanBut,

};

FMedic.Controls.Add(but);

}

}

}

ProfileForm.cs

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp9

{

class ProfileForm

{

private static Form FProf = new Form();

private static Panel FPanel = new Panel()

{

Dock = DockStyle.Fill

};

private static string FIO = "";

private static string Spec = "Неопределена";

private static string Shedul = "Неопределен";

private static string Prav = "Сдесь должны быть правила оказания услуг";

private static string SeriesPass = "";

private static string NumberPass = "";

public static void NewProfileForm()

{

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@User\_Nick", AutorizForm.Nick),

};

DataTable DT = DB.GetData("[SELECT\_Profile\_Personal]", SP);

try

{

DataRow DR = DT.Rows[0];

FIO = DR["SurnamePers"] + " " + DR["NamePers"] + " " + DR["PatronymicPers"];

Spec = DR["Name\_SpecialityPersonal"].ToString();

Shedul = DR["weekdays"].ToString();

SeriesPass = DR["SeriesPassportPers"].ToString();

NumberPass = DR["NumberPassportPers"].ToString();

}

catch

{

return;

}

SP = new SqlParameter[]

{

new SqlParameter("@User\_Nick", AutorizForm.Nick),

};

DT = DB.GetData("SELECT\_Regulation", SP);

try

{

Prav = DT.Rows[0]["TextRegulation"].ToString();

}

catch { }

FProf = new Form();

FProf.Controls.Add(FPanel);

FProf.Text = "Профиль " + AutorizForm.Nick;

FProf.FormBorderStyle = FormBorderStyle.FixedSingle;

FProf.MaximizeBox = false;

FProf.StartPosition = FormStartPosition.CenterScreen;

MainForm.CreateLabel(FPanel, "LogInLabel", "Пользователь: "+AutorizForm.Nick, 20, 20, 16);

MainForm.CreateLabel(FPanel, "FIOLabel", "ФИО: "+FIO, 20, 60, 10);

MainForm.CreateLabel(FPanel, "SpecLabel", "Специальность: "+Spec, 20, 90, 10);

MainForm.CreateLabel(FPanel, "ShedulLabel", "График: " + Shedul, 20, 120, 10);

MainForm.CreateButton(FPanel, "ExitButton", "Выход из системы", 20, 480 - 95, 175, 30, 10);

FPanel.Controls["ExitButton"].Click += ExitButton;

int maxWidth = 0;

for (int i = 0; i < FPanel.Controls.Count; i++)

{

if (maxWidth < FPanel.Controls[i].Width)

maxWidth = FPanel.Controls[i].Width;

}

FProf.Width = maxWidth + 50;

FProf.Height = 460;

CreateTabControl();

FProf.ShowDialog();

}

private static void TextChange(object sender, EventArgs e)

{

bool Redy1 = ((((FPanel.Controls["TabC"] as TabControl).Controls["PassPage"] as TabPage).Controls["OldPassTB"] as MaskedTextBox).Text != "");

bool Redy2 = ((((FPanel.Controls["TabC"] as TabControl).Controls["PassPage"] as TabPage).Controls["NewPassTB"] as MaskedTextBox).Text != "");

bool Redy3 = ((((FPanel.Controls["TabC"] as TabControl).Controls["PassPage"] as TabPage).Controls["ApplyNewPassTB"] as MaskedTextBox).Text != "");

((((FPanel.Controls["TabC"] as TabControl).Controls["PassPage"] as TabPage).Controls["PassBut"] as Button).Enabled) = ((Redy1) & (Redy2) & (Redy3));

}

private static void PassButton(object sender, EventArgs e)

{

string OldPass = (((FPanel.Controls["TabC"] as TabControl).Controls["PassPage"] as TabPage).Controls["OldPassTB"] as MaskedTextBox).Text;

string NewPass = (((FPanel.Controls["TabC"] as TabControl).Controls["PassPage"] as TabPage).Controls["NewPassTB"] as MaskedTextBox).Text;

string AcceptNewPass = (((FPanel.Controls["TabC"] as TabControl).Controls["PassPage"] as TabPage).Controls["ApplyNewPassTB"] as MaskedTextBox).Text;

if (NewPass == AcceptNewPass)

{

SqlParameter[] SP = new SqlParameter[]

{

new SqlParameter("@SeriesPassportPers", SeriesPass),

new SqlParameter("@NumberPassportPers", NumberPass),

new SqlParameter("@User\_Pass", OldPass),

new SqlParameter("@New\_User\_Pass", NewPass),

};

DB.UpdData("[UPD\_Personal]", SP);

(((FPanel.Controls["TabC"] as TabControl).Controls["PassPage"] as TabPage).Controls["OldPassTB"] as MaskedTextBox).Text = "";

(((FPanel.Controls["TabC"] as TabControl).Controls["PassPage"] as TabPage).Controls["NewPassTB"] as MaskedTextBox).Text = "";

(((FPanel.Controls["TabC"] as TabControl).Controls["PassPage"] as TabPage).Controls["ApplyNewPassTB"] as MaskedTextBox).Text = "";

MessageBox.Show("Пароль успешно изменен");

}

else

{

MessageBox.Show("Новые пароли должны совпадать!", "Ошибка ввода!");

}

}

private static void ExitButton(object sender, EventArgs e)

{

Form F = Application.OpenForms[0];

(F as MainForm).LastLabel = new LinkLabel();

(F as MainForm).DelControls();

(F as MainForm).SearchPanel.Visible = false;

AutorizForm.Spec = "";

DB.LogIn = false;

(F as MainForm).Form1\_Activated(F, new EventArgs());

FProf.Close();

}

private static void CreateTabControl()

{

TabControl tab = new TabControl()

{

Name = "TabC",

Location = new Point(20, 150),

Width = FProf.Width - 50,

Height = FProf.Height - 230,

};

tab.TabPages.Add("Правила оказания услуг");

TextBox TB = new TextBox()

{

Name = "tabTB",

Text = Prav,

Multiline = true,

Dock = DockStyle.Fill,

ReadOnly = true,

ScrollBars = ScrollBars.Both,

};

tab.TabPages[0].Controls.Add(TB);

TabPage Tp = new TabPage()

{

Name = "PassPage",

Text = "Смена пароля",

};

tab.TabPages.Add(Tp);

Label Lb = new Label()

{

Name = "OldPassLabel",

Text = "Введите предыдущий пароль",

Location = new Point(20, 10),

AutoSize = true,

};

MaskedTextBox MTB = new MaskedTextBox()

{

Name = "OldPassTB",

Location = new Point(20, 33),

Size = new Size(150, 20),

};

MTB.TextChanged += TextChange;

MTB.UseSystemPasswordChar = true;

tab.TabPages[1].Controls.Add(Lb);

tab.TabPages[1].Controls.Add(MTB);

Lb = new Label()

{

Name = "NewPassLabel",

Text = "Введите новый пароль",

Location = new Point(20, 33+30),

AutoSize = true,

};

MTB = new MaskedTextBox()

{

Name = "NewPassTB",

Location = new Point(20, 63+23),

Size = new Size(150, 20),

};

MTB.TextChanged += TextChange;

MTB.UseSystemPasswordChar = true;

tab.TabPages[1].Controls.Add(Lb);

tab.TabPages[1].Controls.Add(MTB);

Lb = new Label()

{

Name = "ApplyNewPassLabel",

Text = "Подтвердите новый пароль",

Location = new Point(20, 63+23+30),

AutoSize = true,

};

MTB = new MaskedTextBox()

{

Name = "ApplyNewPassTB",

Location = new Point(20, 63 + 23 + 30 + 23),

Size = new Size(150, 20),

};

MTB.TextChanged += TextChange;

MTB.UseSystemPasswordChar = true;

tab.TabPages[1].Controls.Add(Lb);

tab.TabPages[1].Controls.Add(MTB);

Button B = new Button()

{

Name = "PassBut",

Text = "Сменить пароль",

Location = new Point(20, 63 + 23 + 30 + 23+30),

Size = new Size(150,30),

Enabled = false,

};

tab.TabPages[1].Controls.Add(B);

B.Click += PassButton;

FPanel.Controls.Add(tab);

}

}

}

RegForm.cs

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp9

{

class RegForm

{

private Form FReg = new Form();

private Label lab = new Label();

private TextBox Tb = new TextBox();

private MaskedTextBox mtb = new MaskedTextBox();

private ComboBox cb = new ComboBox();

private Button but = new Button();

private Font DefFont = new Font("Microsoft Sans Serif", 10);

private Color DefBackColor = Color.FromName("Control");

private Color DefTextColor = Color.FromName("ControlText");

//Позиции //позиции

private Point locSurLabel = new Point(20, 10); //label фамилии

private Point locSurTB = new Point(20, 33); //textBox фамилии

private Point locNameLabel = new Point(200, 10); //label имени

private Point locNameTB = new Point(200, 33); //textBox имени

private Point locPatLabel = new Point(20, 63); //label отчества

private Point locPatTB = new Point(20, 86); //textBox отчества

private Point locAutLabel = new Point(200, 63); //label логина

private Point locAutTB = new Point(200, 86); //TextBox логина

private Point locPassLabel = new Point(20, 116); //label пароля

private Point locPassTB = new Point(20, 139); //TextBox пароля

private Point locAgPassLabel = new Point(200, 116); //label повтора пароля

private Point locAgPassTB = new Point(200, 139); //TextBox повтора пароля

private Point locSeriesLabel = new Point(20, 169); //label серии пасспорта

private Point locSeriesTB = new Point(20, 192); //textBox серии пасспорта

private Point locNumberLabel = new Point(200, 169); //label номера пасспорта

private Point locNumberTB = new Point(200, 192); //textBox номера пасспорта

private Point locRegBut = new Point(20, 230); //кнопка вход

private Point locCanBut = new Point(200, 230); //кнопка отмены

//размеры //размеры

private Size sizeSurTB = new Size(150, 20); //textBox фамилии

private Size sizeNameTB = new Size(150, 20); //textBox имени

private Size sizePatTB = new Size(150, 20); //textBox отчества

private Size sizeAutTB = new Size(150, 20); //TextBox логина

private Size sizePassTB = new Size(150, 20); //TextBox пароля

private Size sizeAgPassTB = new Size(150, 20); //TextBox повтора пароля

private Size sizeSeriesTB = new Size(150, 20); //textBox серии паспорта

private Size sizeNumberTB = new Size(150, 20); //textBox номера пасспорта

private Size sizeRegBut = new Size(150, 30); //кнопка вход

private Size sizeCanBut = new Size(150, 30); //кнопка отмены

public RegForm()

{

}

public void NewRegForm()

{

CreateSurLabel();

CreateSurTB();

CreateNameLabel();

CreateNameTB();

CreatePatLabel();

CreatePatTB();

CreateRegButton();

CreateCancelButton();

CreateLoginLabel();

CreateLoginTB();

CreatePassLabel();

CreatePassTB();

CreateAgainPassTB();

CreateAgainPassLabel();

CreateSeriesTB();

CreateSeriesLabel();

CreateNumLabel();

CreateNumTB();

FReg.Text = "Регистрация";

FReg.Size = new Size(385, 310);

FReg.FormBorderStyle = FormBorderStyle.FixedSingle;

FReg.MaximizeBox = false;

FReg.StartPosition = FormStartPosition.CenterScreen;

FReg.FormClosing += new FormClosingEventHandler(FormClosing);

FReg.Controls["RegButton"].Click += new EventHandler(RegButtonClick);

FReg.Controls["CancelButton"].Click += new EventHandler(CancelButtonClick);

FReg.Controls["PassTextBox"].TextChanged += new EventHandler(AggPassTextChange);

FReg.Controls["PassAgainTextBox"].TextChanged += new EventHandler(AggPassTextChange);

FReg.ShowDialog();

}

//закрытие формы

private void FormClosing(object sender, EventArgs e)

{

}

//кнопка зарегистрироваться

private void RegButtonClick(object sender, EventArgs e)

{

if (aggPass)

{

SqlParameter[] SP = new SqlParameter[7]

{

new SqlParameter("@SurnamePers",(FReg.Controls["SurTextBox"] as TextBox).Text),

new SqlParameter("@NamePers",(FReg.Controls["NameTextBox"] as TextBox).Text),

new SqlParameter("@PatronymicPers",(FReg.Controls["PatTextBox"] as TextBox).Text),

new SqlParameter("@SeriesPassportPers",(FReg.Controls["SeriesTextBox"] as TextBox).Text),

new SqlParameter("@NumberPassportPers",(FReg.Controls["NumberTextBox"] as TextBox).Text),

new SqlParameter("@User\_Nick",(FReg.Controls["LoginTextBox"] as TextBox).Text),

new SqlParameter("@User\_Pass",(FReg.Controls["PassTextBox"] as MaskedTextBox).Text),

};

int id = DB.WriteData("Add\_Personal", SP);

if (id != -10)

{

FReg.Close();

}

}

else

{

MessageBox.Show("Указанные пароли должны совпадать!","Ошибка ввода!");

}

}

//кнопка отмены

private void CancelButtonClick(object sender, EventArgs e)

{

FReg.Close();

}

//при измнении текста

bool aggPass = false;

private void AggPassTextChange(object sender, EventArgs e)

{

MaskedTextBox Pass = (FReg.Controls["PassTextBox"] as MaskedTextBox);

MaskedTextBox AggPass = (FReg.Controls["PassAgainTextBox"] as MaskedTextBox);

if (Pass.Text != AggPass.Text)

{

Pass.BackColor = Color.Red;

AggPass.BackColor = Color.Red;

aggPass = false;

}

else

{

Pass.BackColor = Color.FromName("Control");

AggPass.BackColor = Color.FromName("Control");

aggPass = true;

}

}

private void CreateLoginLabel()

{

lab = new Label()

{

Name = "LoginLabel",

Text = "Логин: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locAutLabel,

AutoSize = true,

};

FReg.Controls.Add(lab);

}

private void CreateLoginTB()

{

Tb = new TextBox()

{

Name = "LoginTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locAutTB,

Size = sizeAutTB,

MaxLength = 50,

};

FReg.Controls.Add(Tb);

}

private void CreatePassLabel()

{

lab = new Label()

{

Name = "PassLabel",

Text = "Пароль: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locPassLabel,

AutoSize = true,

};

FReg.Controls.Add(lab);

}

private void CreatePassTB()

{

mtb = new MaskedTextBox()

{

Name = "PassTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locPassTB,

Size = sizePassTB,

MaxLength = 30,

};

mtb.UseSystemPasswordChar = true;

FReg.Controls.Add(mtb);

}

private void CreateAgainPassLabel()

{

lab = new Label()

{

Name = "PassAgainLabel",

Text = "Повторите пароль: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locAgPassLabel,

AutoSize = true,

};

FReg.Controls.Add(lab);

}

private void CreateAgainPassTB()

{

mtb = new MaskedTextBox()

{

Name = "PassAgainTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locAgPassTB,

Size = sizeAgPassTB,

MaxLength = 30,

};

mtb.UseSystemPasswordChar = true;

FReg.Controls.Add(mtb);

}

private void CreateSurLabel()

{

lab = new Label()

{

Name = "SurLabel",

Text = "Фамилия: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locSurLabel,

AutoSize = true,

};

FReg.Controls.Add(lab);

}

private void CreateSurTB()

{

Tb = new TextBox()

{

Name = "SurTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locSurTB,

Size = sizeSurTB,

MaxLength = 50,

};

FReg.Controls.Add(Tb);

}

private void CreateNameLabel()

{

lab = new Label()

{

Name = "NameLabel",

Text = "Имя: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locNameLabel,

AutoSize = true,

};

FReg.Controls.Add(lab);

}

private void CreateNameTB()

{

Tb = new TextBox()

{

Name = "NameTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locNameTB,

Size = sizeNameTB,

MaxLength = 50,

};

FReg.Controls.Add(Tb);

}

private void CreatePatLabel()

{

lab = new Label()

{

Name = "PatLabel",

Text = "Отчество: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locPatLabel,

AutoSize = true,

};

FReg.Controls.Add(lab);

}

private void CreatePatTB()

{

Tb = new TextBox()

{

Name = "PatTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locPatTB,

Size = sizePatTB,

MaxLength = 50,

};

FReg.Controls.Add(Tb);

}

private void CreateRegButton()

{

but = new Button()

{

Name = "RegButton",

Text = "Зарегистрироваться",

Font = new Font("Microsoft Sans Serif", 9),

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locRegBut,

Size = sizeRegBut,

};

FReg.Controls.Add(but);

}

private void CreateCancelButton()

{

but = new Button()

{

Name = "CancelButton",

Text = "Отмена",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locCanBut,

Size = sizeCanBut,

};

FReg.Controls.Add(but);

}

private void CreateSeriesLabel()

{

lab = new Label()

{

Name = "SeriesLabel",

Text = "Серия пасспорта: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locSeriesLabel,

AutoSize = true,

};

FReg.Controls.Add(lab);

}

private void CreateSeriesTB()

{

Tb = new TextBox()

{

Name = "SeriesTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locSeriesTB,

Size = sizeSeriesTB,

MaxLength = 4,

};

FReg.Controls.Add(Tb);

}

private void CreateNumLabel()

{

lab = new Label()

{

Name = "NumberLabel",

Text = "Номер пасспорта: ",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locNumberLabel,

AutoSize = true,

};

FReg.Controls.Add(lab);

}

private void CreateNumTB()

{

Tb = new TextBox()

{

Name = "NumberTextBox",

Text = "",

Font = DefFont,

Visible = true,

BackColor = DefBackColor,

ForeColor = DefTextColor,

Location = locNumberTB,

Size = sizeNumberTB,

MaxLength = 6,

};

FReg.Controls.Add(Tb);

}

}

}