*Приложение за обработка на информация за лекарства*

**I. Кратко описание на програмата**

Приложението е разработено на:

* WPF ( .Net C# )

Използвана база данни:

* MS SQL Server

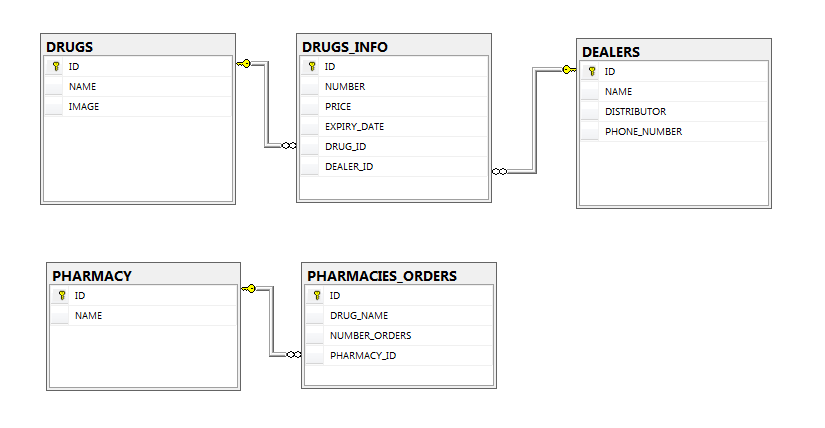
**Предназначение**

Програмата може да обработва данни за лекарства. Има възможност за добавяне на данни за лекарство: име, годност, брой и цена, данни за доставчик: име, дистрибутор, телефонен номер, данни за аптеки: име и поръчки. Има възможност за сортиране на лекарствата по цена или брой, намиране на фирма доставчик с най – ниска цена на лекарството, проверка на срок на годност на лекарства.

**II. Обща структура на програмата**

**Проектиране**

* Модел на БД

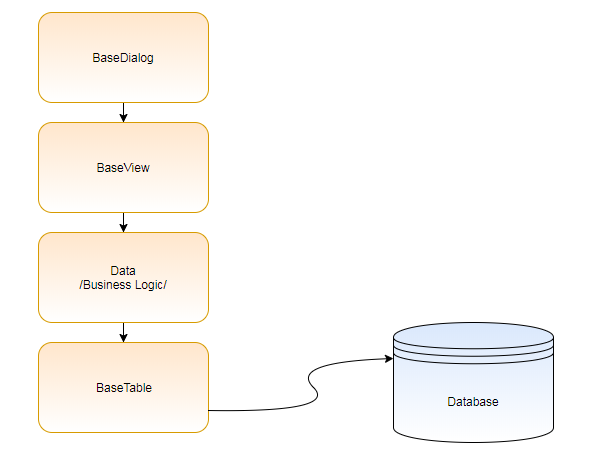


* Архитектура на приложение

Архитектурата се основава на разделни слоеве, като всеки слой използва функциите и услугите само на слоя под него.

Приложението е разделено на класове, които следват йерархична структура.

За всяка таблица, съдържаща се в базата данни е дефиниран отделен клас, който се явява наследник на класа „BaseRecord“. От своя страна наследниците на този клас се използват в класа „BaseTable”, с помощта на който се извършват основните операции по обръщение към базата данни. Следват класовете за бизнес логика, които се представят с наставка „Data”. Следват в йерархията класовете наследници на „BaseTabView“, който служи за изобразаване на данните. На най – високо ниво се намират класовете наследници на „BaseDialog“, който служи за обработка на отделните записи.



**Програмиране**

**Описание на класовете**

**Клас DataBaseConnection – за връзка към БД, по метода на Singleton**

***Член променливи:***

public SqlConnection sqlConnection = null; - инстанция на базата данни

private static DataBaseConnection databaseInstance = null; - инстанция на singleton класа

public static string ConnectionString

{ get { return "Data Source=[SERVER NAME]\\SQLEXPRESS;Initial Catalog=Pharmacy;Integrated Security=True"; } } – променлива, нужна за връзка към БД

public event EventHandler OnConnectingStart = null; - събитие за отчитане на свързване към БД

public event EventHandler OnConnectingEnd = null; - събитие за отчитане край на връзката към БД

public event EventHandler<EventArgs<Exception>> OnConnectingError = null; - събитие за отчитане на грешка при връзка към БД

***Конструктор:***

private DataBaseConnection() – инициализира нов обект от тип sqlConnection(), използвайки член – променливата ConnectionString;

***Методи:***

public static DataBaseConnection Instance – статичен метод, който връща единствен обект на класа DataBaseConnection(), по модела на Singleton. Връща един и същ обект в цялото приложение за осъществяване на връзка към БД.

public static async void OpenConnection() – асинхронен статичен метод за отваряне на връзка към БД. В него се предизвиква събити за обновяване на потребителският интерфейс при стартиране на свързване към БД, при приключване на свързването към БД или при грешка по време на свързването.

public static void CloseConnection() – статичен метод за затваряне на връзка към БД

**public class EventArgs<T> : EventArgs –** клас за предаване на грешка (exception) към събитие „OnConnectingError“

„Базови класове“

**public class BaseRecord :** **InotifyPropertyChanged –** базов клас за запис от БД.

***Член променливи:***

public event PropertyChangedEventHandler PropertyChanged; - събитие за обновяване на полетата на наследника в клас имплементиращ **PropertyChanged**

public Type DerivedType – тип на наследника

public int StartFromProperty – начална граница на променливата за обновяване

public int EndToProperty – крайна граница на променливата за обновяване

***Конструктор:***

public BaseRecord(Type type, int endToProperty = -1, int startFromProperty = 0)

***Методи:***

public virtual List<string> GetFields() – връща полетата на всеки един клас, като използва endToProperty и startFromProperty. Изполва се рефлекция за достъпване на променливите от DerivedType (наследника)

public void UpdateAll() – актуализира всички полета

private void NotifySinglePropertyChanged([CallerMemberName]string name = null) –Предизвиква събитие **PropertyChanged,** за известяване на конкретна променлива по име.

**public class BaseRecordObservableCollection : ObservableCollection<BaseRecord> -** базов клас за колекция

***Конструктор:***

public BaseRecordObservableCollection() : base()

***Методи:***

protected override void OnCollectionChanged(NotifyCollectionChangedEventArgs e) – проверяваме събитието за промяна на елемент в колекцията. Извикаме UpdateAll на елемента (наследник на BaseRecord) за предизвикване на събитието за обновяване на всичките му променливи. Съответно списъка е свързан към това събитие и се обновява графичния интерфейс.

**public abstract class BaseTable<T> - темплейтен базов клас за всяка една таблица от базата данни, който имплементира обръщенията към нея.**

***Член променливи:***

protected T Record; - променлива за съхранение на обект от тип Т

protected string ColumnID - променлива за съхранение на името на колоната на ID (може да не се казва ID в някои таблици)

protected string TableName – променлива за името на таблицата, която ще се обработва в даден момент

protected bool UseUpdateCounter – променлива, която отразява дали се изполва полето UpdateCounter, налично във всяка таблица. Полето е нужно за поддържане на актуалността на достъпвания запис. Възможно е в даден момент няколко човека да достъпват и променят един и същи запис и всяка промяна увеличава стойността на UpdateCounter с единица. Ако записа, който ние променяме не е със същата стойност на UpdateCounter в базата данни, то направените промени ще се отменят.

protected SqlTransaction externalTransaction – променлива за използване външна транзакция

protected SqlTransaction sqlTransaction – променлива за вътрешна транзакция

public bool UseExternalTransaction - променлива за отчитане дали се използва външна транзакция за заключване на запис

***Конструктор:***

public BaseTable(string tableName, string columnID, bool useUpdateCounter, bool useExternalTransaction = false, SqlTransaction ExternalTransaction = null) – приема всички гореизброени, като ако няма външна транзакция полетата, свързани с нея приемат подразбиращи се стойности

***Методи:***

protected abstract int GetIDValue(T record); - връща стойността на ID от типа „Т“, задължително се имплементира от наследниците.

protected abstract void SetIDValue(int ID, T record); - установява стойността на ID от типа „Т“, задължително се имплементира от наследниците.

protected virtual int GetUpdateCounterValue(T record); - връща стойността на UpdateCounter, не е задължително да се имплементира от наследниците, защото може някои таблици да не го поддържат.

protected virtual void IncreaseUpdateCounterValue(T record) – увеличава стойността на UpdateCounter, не е задължително да се имплементира от наследниците, защото може някои таблици да не го поддържат.

protected abstract T DataToRecord(SqlDataReader reader) - Четене на запис от БД. Копира данните от reader към съответният клас от тип Т, за да може да се обработват полетата му и връща обекта. Задължително се имплементира от наследниците.

protected abstract void RecordToData(SqlCommand command, T record) - Копира данните от структурата към reader, за да може да се представят във вид за вмъкване в БД. Задължително се имплементира от наследниците.

protected abstract List<string> GetColumns() - връща списък от всички колони налични в БД за съответната таблица.

protected abstract T DefaultRecordConstructor() – връща обект от тип „Т“ по подразбиращ се конструтор

public bool SelectAllRecords(List<T> recordsArray, string whereClause = "") – метод за селектиране на всички записи от БД, свързани със съответната таблица. Връща списък с всички селектирани данни като параметър по референция. Възможност за добавяне на условие, по което да се извлекат данните. Образува се заявка и след това се използва връзката към БД за четене. Използва се метода DataToRecord за преобразуване на обекта и след като се изчете, се запълва списъка. Възможност за прихващане на грешки. Връща истина, ако не е възникнала грешка, в противен случай връща лъжа.

public bool SelectRecord(int ID, out T record) – метод, който селектира само един запис по зададено ID. Връща записа като параметър (параметъра, който е подаден е неинициализиран). Отново се конструира заявка като се преизползва метода SelectAllRecords, но със условие за селектиране, което ще е по ID. Проверка, ако има повече от един върнат запис, се връща лъжа и се спира изпълнението. Не бива да има запис с един и същ primary key. Ако не е намерен запис, отново се връща лъжа и се спира изпълението. Ако всички проверки са минали се връща истина и програмата продължата своето изпълнение.

public bool InsertRecord(T record) – метод за вмъкване на запис в БД. Вземат се всички колони, съдържащи се в БД, чрез метода GetColumnsStringForInsert(), който използва GetColumns(). GetColumns() връща всички колони за съответната таблица. Метода използва и GetParametersStringForInsert(). Създава се заявка спрямо името на таблицата, колоните и желаните параметри за вмъкване. Използва се метода RecordToData() за подготвяне на записа за вмъкване в БД. Вмъква се записа и се взема новото ID, генерирано от БД, за да се установи в записа. При успешно вмъкване се връща истина, в противен случай се връща лъжа.

public bool EditRecord(int ID, T record) – метод за актуализиране на запис в БД. Приема като параметри ID, по което да се актуализира записа, подаден като параметър. Прави се проверка за използване на външна транзкация. Проверява се дали записа съществува. Проверява се дали се изпозлва UpdateCounter. Ако да се проверява стойностите на прочетения запис и записа, който ще се добавя. При успешна проверка се увеличава стойността на UpdateCounter-a на записа, който ще се добавя. При успешно обновяване се изпълнява „commit” на транзакцията (външна или вътрешна в зависимост от параметъра UseExternalTransaction). Има блок за обработка на грешки.

public bool DeleteRecord(int ID) – метод за изтриване на запис от БД по ID. Извлича се избраният запис чрез метода SelectRecord(). Ако не е намерен търсеният запис, се връща лъжа. В противен случай се изтрива записа от БД и се връща истина.

public bool SyncRecords(List<T> newRecords, string whereClause = "") – метод за синхронизация на записите. Модифицира всички записи от таблицата, които имат разлики с подадения списък със записи. Използва се вградения метод на List – where за определяне на съответните разлики (по ID, чрез функцията GetIDValue) за добавяне и изтриване към БД. Използва се вградения метод на List – intersect за определяне на записите с различно съдържание. Извиква се виртуалната функция GetEqualityComparerByContent за използване на компаратор по съдържание. Трябва да се имплементира от наследника.

private string GetColumnsStringForInsert() – метод, който връща стринг от колоните на текущият запис. Този стринг в последствие се използва за конструиране на заявките.

private string GetParametersStringForInsert() – метод, който връща стринг от параметрите на текущият запис. Този стринг в последствие се използва за конструиране на заявките.

private string GetColumnsAndParametersStringForEdit() – връща стринг със съответните колони и съответните им параметри

protected SqlCommand CreateSqlCommand(string query) - връща обект от SqlCommand, по подаден параметър за заявка.

private void DisposeTransaction() – метод, който занулява транзакцията

private bool RollbackTransaction(SqlTransaction sqlTransaction) - метод, който отхвърля направените промени в транзакцията

**public abstract class BaseTabView : ListView– клас за визуализиране на данните в контрола, който наследява ListView**

***Член променливи:***

public enum ContextMenuTypes – показва вида контекстно меню. При някои view е възможно потребителя да няма достъп до всички операции и съответно полетата за операции в контекстоно меню може да са други. По подразбиране контекстното меню е Default.

public BaseRecordObservableCollection itemsSource – променлива, която играе ролята на източник на данните, т.е. те се визуализират на базата на тази колекция

private enum ContextMenuItems – изброим тим за наличните операции при кликане на десен бутон върху view. По подразбиране са преглед, добавяне, редактиране и изтриване.

private ContextMenu contextMenu = null; - обект от тип ContextMenu за създаване на контекстно меню.

***Конструктор:***

public BaseTabView(ContextMenuTypes \_contextMenuType = ContextMenuTypes.Default) : base() – по подразбиране контекстното меню е зададено като Default.

***Методи:***

protected override void OnInitialized(EventArgs e) – инициализира view-то. Извиква се метода Initialize()

public void Update() – вика метода LoadData(), който е абстрактен метод за инициализиране на данните.

private void Initialize() – инициализира нов обект от тип BaseRecordObservableCollection(). Добавя се метод за прихващане на събитие за натиснат десен бутон на мишката. Извикват се методите OnViewIntialized(),OnInitColumns(), LoadData(),CreateContextMenu(), описани по – долу.

protected abstract void OnViewIntialized(); - инициализира член – променливите в наследника.

protected abstract void OnInitColumns() - инициализира колоните на контролата в наследника, използвана за визуализиране на данните.

protected abstract void LoadData() - попълва данните за визуализиране в наследника

public virtual void Close() – затваря контролата за визуализация на данните

private void OnMouseDoubleClick(object sender, System.Windows.Input.MouseButtonEventArgs e) – метод за прихващане на ляв двоен клик на мишката върху view-to. Вика се метода OnPreviewClick(), който визуализира избраният запис в диалог.

private void CreateContextMenu() – проверява за дали има налично контекстно меню и ако няма се продължава изпълнението на програмата. Ако има налично меню, то се създава чрез обект от ContextMenu() и се добавя метод OnContextMenuOpened(), който обработва събитието създаване на контекстно меню.

private void OnContextMenuOpened(object sender, RoutedEventArgs e) – метод, който инициализира контекстното меню. Проверява се дали контекстното меню е по подразбиране чрез променливата ContextMenuTypes. Ако е така се инициализират полетата в него. Добавят се полетата „Преглед“, „Добави“, „Редактирай“ и „Изтрий“, за които за всяко отделно кликване върху тях се добавят методи за обработка на събитията при кликването им.

Ако контекстното меню не е по подразбиране, то се вика метода FillContextMenu(), който е виртуален и връща списък от желаните полета в контекстното меню.

protected virtual void OnPreviewClick(object sender, RoutedEventArgs e) – метод , който обработва натискането на „Преглед“ от контекстното меню. Проверява се дали има селектиран елемент. Ако има селектиран елемент, той се взема, чрез SelectedItem, в противен случай програмата продължава своето изпълнение. Селектираният запис се подава на съответния обработващ диалог и той се визуализира.

protected virtual void OnAddClick(object sender, RoutedEventArgs e) - метод , който обработва натискането на „Добави“ от контекстното меню. Отваря се диалог за вмъкване на нов запис. Ако потребителят е натиснал „ОК“ в диалога, записа се добавя и в базата данни, както и в колекцията от тип BaseRecordObservableCollection, за да може да се актуализира записа и в самото view. Ако потребителят е натиснал „Отказ“ в диалога, то никакви действия не се предприемат.

protected virtual void OnEditClick(object sender, RoutedEventArgs e) - метод , който обработва натискането на „Редактирай“ от контекстното меню. Проверява се дали има селектиран елемент. Ако има селектиран елемент, той се взема, чрез SelectedItem, в противен случай програмата продължава своето изпълнение. Селектираният запис се подава на съответния обработващ диалог за редактиране на данните. Ако потребителят е потвърдил чрез натискането на бутона „ОК“ в диалога, то промените се отразяват базата данни, както и в колекцията, която се използва като източник за визуализация на елементите във view-to. Ако потребителят е натиснал бутона „Отказ“ в диалога, то никакви действия не се предприемат.

protected virtual void OnDeleteClick(object sender, RoutedEventArgs e) - метод , който обработва натискането на „Изтрий“ от контекстното меню. Проверява се дали има селектиран елемент. Ако има селектиран елемент, той се взема, чрез SelectedItem, в противен случай програмата продължава своето изпълнение. Извежда се съобщение за потвърждение за изтриване на записа. Ако потребителят е потвърдил чрез натискането на бутона „ОК“ в диалога, то записа се изтрива от базата данни, както и от колекцията, която се използва като източник за визуализация на елементите във view-to. Ако потребителят е натиснал бутона „Отказ“ в, то никакви действия не се предприемат.

**public class BaseTabItem : TabItem –** клас, който описва характеристиките на таб контрола. View-тата се закачат към таб контролата, където се визуализират данните.

***Член променливи:***

private BaseTabView listview = null; - обект за съхранение на текущото view.

protected event EventHandler Close = null; - обект събите.

***Конструктор:***

public BaseTabItem(string headerTitle) – приема като параметър заглавието на view-то

Инициализира обект от тип Grid, където ще бъде добавено view-то и задава заглавието.

***Методи:***

public void SetView(BaseTabView view) – задава view, като подаденото view в grid обекта.

protected void SetHeader(UIElement header) – създава dockPanel, добавя заглавие, добавя CloseButton след заглавието, прихваща се събитието на CloseButton за натискане. В него се извиква Close на базовия ListView. Взима се референция към родителя на този обект (tabControl) и съответно се премахва текущия таб. Предизвиква се събитието Close.

**public class BaseDialog : Window –** клас за диалогови прозорци. При кликане от контекстното меню „Преглед“, „Редактирай“ или „Добави“ в кое да е view се показва диалогов прозорец в различни режими, съответно – само за преглеждане на данните, режим на редактиране или в режим на добавяне, при който всички полета са празни и се очаква да се въведат данни от потребителя.

***Член променливи:***

private bool isUsingOK; - булева променлива за указване, че се използва бутона „ОК“

private bool isUsingCancel; - булева променлива за указване, че се използва бутона „Cancel“

protected int TabOrder – променлива, която съхранява реда на табулиране на контролите

public enum DialogModes – декларация на тип, който указва различните режими на диалоговия прозорец

protected DialogModes DialogMode – променлива от тип DialogModes

***Конструктор:***

public BaseDialog(DialogModes dialogMode, DependencyObject parent = null, bool \_isUsingOK = true, bool \_isUsingCancel = true) – приема като параметри – режима, в който се отваря диалога, по подразбиращ се параметър за указване на родителя на диалога (понеже може да бъде модален), променливи указващи дали се ползват бутоните „OK” и „Cancel”. Всички променливи се инициализират.

***Методи:***

private void BaseDialog\_Loaded(object sender, RoutedEventArgs e) – метод, който обработва събитие при зареждане на диалога. Копират се данните от селектираният елемент в съответните контроли. Зарежда се по подразбираща се икона от ресурсите.

private void BaseDialog\_Initialized(object sender, EventArgs e) – викат се методите LoadData() и OnInitControls(), описани по – долу.

protected virtual void OnInitControls() – взимат се референции към реалните обекти „ОК“ бутон и „Cancel” бутон в базовия. Към тях се прихващат събития при клик на мишката. В наследниците се инициализират заглавието на диалога, бутоните на диалога в зависимост от режима, в който е отворен. Задава се и последователността на табулация.

protected void OnOkClick(object sender, RoutedEventArgs e) - метод, който обработва събитието при клик върху бутона „ОК“. Вика се метода ValidateData(), който е имплементиран във всеки наследник за валидация на въведените данни. Ако валидацията е успешна се вика метода CopyControlsToData(), който копира данните обратно от контролите във view-то и диалога се затваря.

private void OnCancelClick(object sender, RoutedEventArgs e) - затваря се диалоговия прозорец, без да се предприемат никакви действия.

protected virtual void GetOKButton(ref Button okButton) – задава се бутона „ОК“ в наследниците

protected virtual void GetCancelButton(ref Button cancelButton) – задава се бутона „Cancel” в наследниците

protected virtual void CopyDataToControls() – имплементира се от наследниците, взимат се данните от view-то и се изобразяват в контролите на диалога.

protected override void CopyControlsToData() - имплементира се от наследниците, вземат се поле по поле данните и се съхраняват стойностите им в променлива в класа наследник, за да се изобразят във view-то.

protected virtual bool ValidateData() – валидира данните

protected virtual void LoadData() – зарежда данните

protected int AddTabIndex() – увеличава стойността, отнасяща се до табулацията

private void BaseDialog\_SourceInitialized(object sender, EventArgs e) – премахва бутоните за максимизиране и минимизиране на прозореца.

„Клас за съобщения“

**public static class MessageBoxes** – клас за извеждане на съобщения

***Член променливи:***

public static readonly string AddErrorMessage = "Грешка при добавяне на запис.";

public static readonly string PreviewErrorMessage = "Грешка при преглед на запис.";

public static readonly string EditErrorMessage = "Грешка при редакция на запис.";

public static readonly string DeleteErrorMessage = "Грешка при изтриване на запис.";

public static readonly string DeleteMessage = "Сигурни ли сте, че искате да изтриете ибрания запис?";

public static readonly string LoadDataErrorMessage = "Грешка при зареждане на данни.";

public static readonly string NoRecordsFoundMessage = "Няма намерени резултати";

***Методи:***

public static MessageBoxResult ShowError(string text) – извежда съобщение за грешка и връща резултата от избора на потребителя

public static MessageBoxResult ShowWarning(string text) – извежда съобщение за предупреждение и връща резултата от избора на потребителя

public static MessageBoxResult ShowInfo(string text) - извежда съобщение за инфомация и връща резултата от избора на потребителя

public static MessageBoxResult MessageBoxShowDeleteMessage() – извежда съобщение за потвърждение за изтриване на елемент и връща резултата от избора на потребителя

„Клас за водене на отчет“

**public class Log** - Клас за водене на отчет в дебъг конзолата и във файл

***Член променливи:***

private static Log logInstance = null; - инстанция на обекта

public static string LogFilePath – променлива за съхранение на директорията на файловете

public string LogFileName – променлива за съхранение името на файла

public enum LogType – дефинира се вида „log” – INFO, WARNING или ERROR

***Конструктор:***

private Log() – инициализира името на файла с „Pharmacy“

***Методи:***

public void InitializeLog() – инициализира текуща дата и време, която ще се добави към името на файла. Взема се текущата директория на проекта и се задава в променливата LogFilePath. Създава директория ‚ logs‘ в директорията на програмата. Изтрива лог файлове по – стари от месец.

public static void LogInfo(string message, params object[] args) – инициализира обект от тип StackFrame. Той взима на информация за името на класа, викащата функция, файла, реда, колонота едно ниво нагоре. Взима се съобщението, подадено като параметър и се форматира чрез подадените параметри - args. Вика се функцията LogMessage(), описана по – долу. Тя извежда съобщението на конзола и в „log” файла.

public static void LogWarning(string message, params object[] args) – инициализира обект от тип StackFrame и се вика функцията LogMessage().

public static void LogError(string message, params object[] args) - инициализира обект от тип StackFrame и се вика функцията LogError() с два параметъра подадени и преподадоени args.

public static void LogError(string message, StackFrame stackFrame = null, params object[] args) – форматира съобщението и го извежда в конзолата и във файл.

public static void LogException(Exception exception) – инициализира обект от тип StackFrame и вика функцията LogError(), която извежда съобщение за грешка в конзолата и във файл.

private static StackFrame GetStackFrame() – връща обект от тип StackFrame, като взима класа n нива нагоре. Това се прави, ако имаме извикване на функция, която вика друга функция която пък вика Log.

private static void LogMessage (string message, LogType logType, string className, string method, string filePath, int lineNumber, int columnNumber) – приема като параметри съобщение, вид на съобщението, име на фунцкия, име на файл, номер на ред, номер на колона. Ако сме в дебъг режим извежда съобщението на конзолата и извежда съобщението в указаният файл.

**public partial class MainWindow : Window -**

***Член променливи:***

private DataBaseConnection dataBaseConnection = null; - променлива за връзка към БД

***Конструктор:***

public MainWindow() – инициализира комонентите и връзката към БД

***Методи:***

private void OnExitButtonClicked(object sender, RoutedEventArgs e) – при кликане от менюто бутон “Exit” излиза от програмата

private void OnDrugsMenuItemClicked(object sender, RoutedEventArgs e) – при кликане от менюто „Лекарства“ – отваря се view с всички налични лекарства, където с десен бутон могат да се преглеждат, добавят, редактират и изтриват лекарства

private void OnDealersMenuItemClicked(object sender, RoutedEventArgs e) – при кликане от менюто „Доставчици“ – отваря се view с всички налични доставчици, където с десен бутон могат да се преглеждат, добавят, редактират и изтриват

private void OnPharmaciesMenuItemClicked(object sender, RoutedEventArgs e) – при кликане от менюто „Аптеки“ – отваря се view с всички налични аптеки, където с десен бутон могат да се преглеждат, добавят, редактират и изтриват

private void UpdateUIOnDatabaseConnection(bool isEnable) – обновява потребителският интерфейс при връзка към БД

public void InitializeDatabaseInstance() – инициализира член променливата dataBaseConnection и закача събития към нея – при стартиране на връзката, при край на връзката и при грешка при свързване

private void DataBaseConnection\_OnConnectingError(object sender, EventArgs<Exception> e) – метод, който обработва събитието при грешка при свързване към БД се извежда съобщение

private void DataBaseConnection\_OnConnectingEnd(object sender, EventArgs e) - метод, който обработва събитието при успешна връзка към БД. Извежда се в статус бар съобщение „Готово“, както и в log-a. Вика се функцията UpdateUIOnDatabaseConnection() и се обновяват прозорците, ако има отворени такива.

private void DataBaseConnection\_OnConnectingStart(object sender, EventArgs e) - метод, който обработва събитието при успешна връзка към БД. Обновява се статус бара със съобщение „Свързване към:“ – името на базата данни и се вика функцията UpdateUIOnDatabaseConnection()

private void OnConnectButtonClicked(object sender, RoutedEventArgs e) – при кликане от менюто „Свързване“ се затваря(ако е отворена) и отваря наново връзката към БД

**public partial class App : Application -** Клас който обработва стартиране и изход на приложението

***Член променливи:***

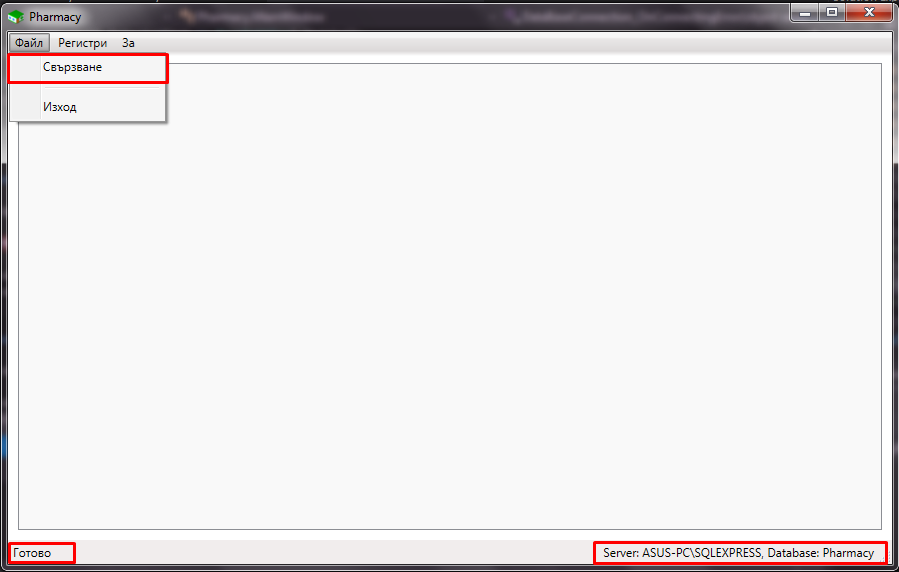
public static readonly string AppName = "Pharmacy" - име на приложението

***Методи:***

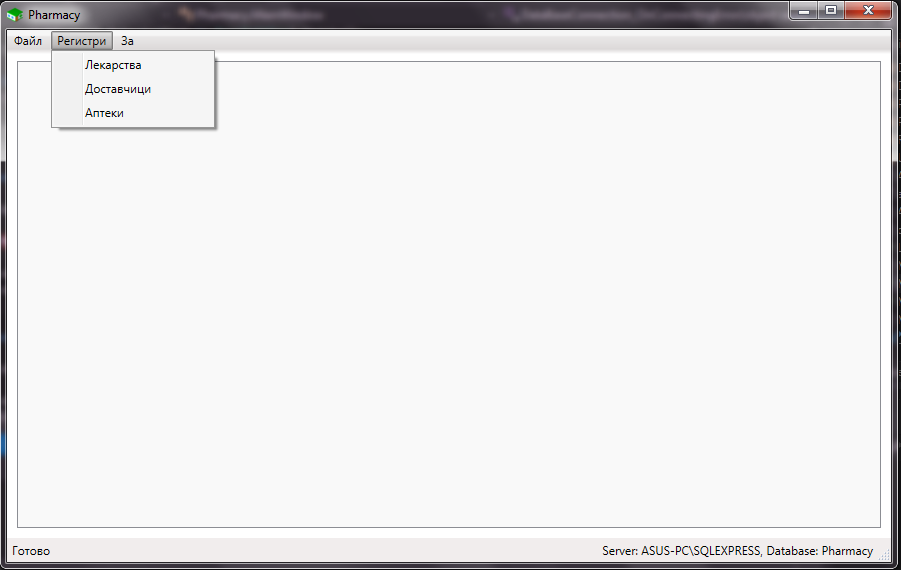
protected override void OnStartup(StartupEventArgs e) – метод, който обработва стартиране на приложението. Инициализира се лога и се записва в него инфомацията при стартиране, и се показва главният прозорец на програмата

protected override void OnExit(ExitEventArgs e) - метод, който обработва изход на приложението. Затваря се връзката към БД и се излиза от приложението

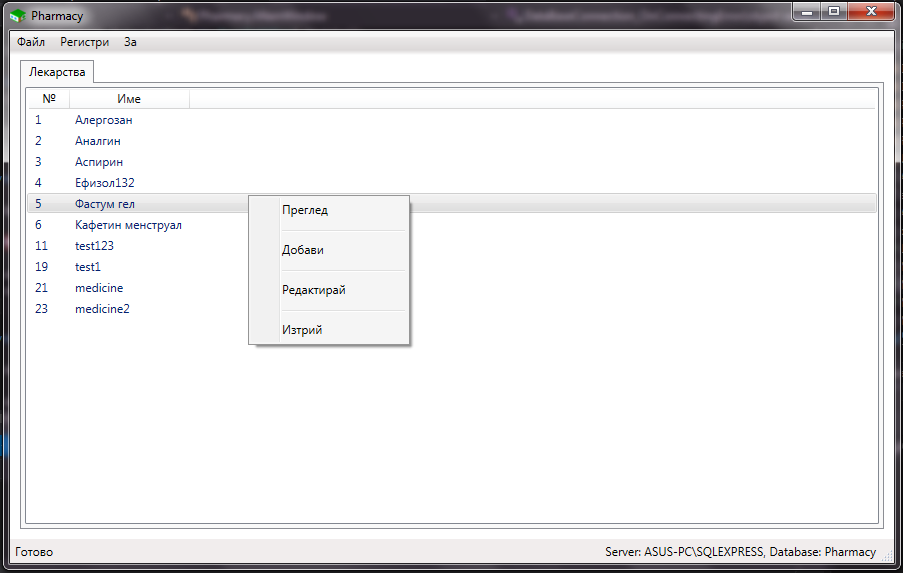
**III. Ръководство за потребителя**



В долният ляв ъгъл се показва състоянието на връзката към БД. В долният десен ъгъл се показва името на сървъра, както и името на БД. Ако връзката към БД прекъсне, потребителят отново може да направи опит за свързване, чрез кликане на „Свързване“ от менюто.

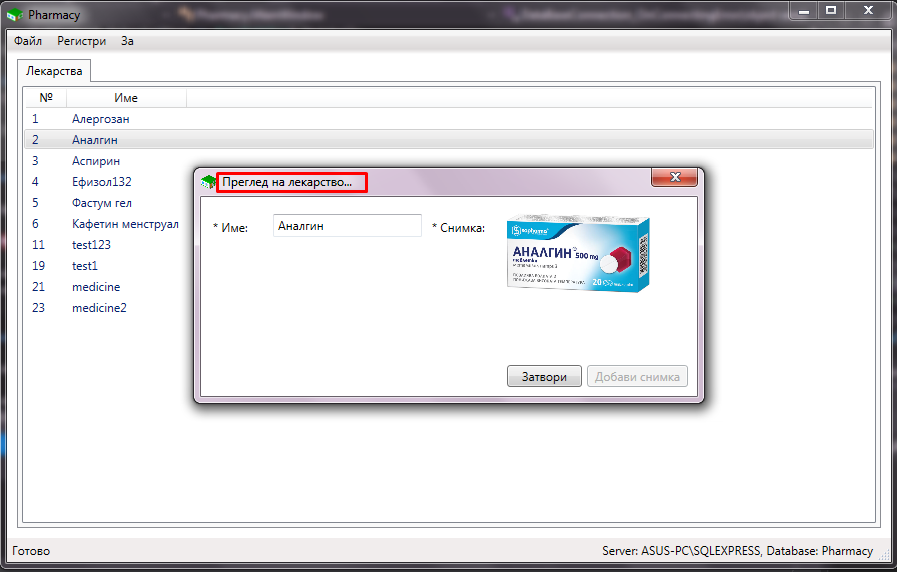


В менюто „Регистри“ се поместват всички данни, налични в БД. Могат да се обработват данни за „Лекарства“, „Доставчици“ и „Аптеки“.

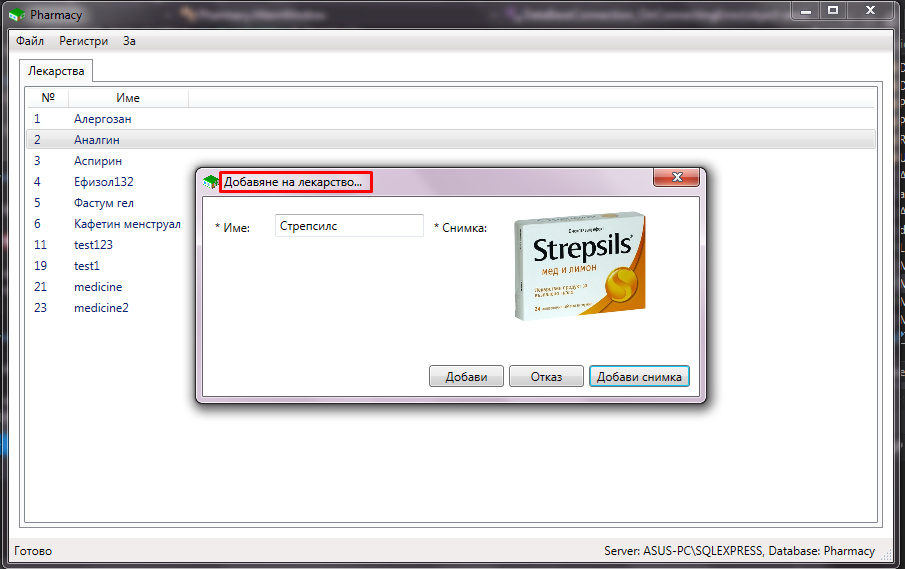


При селектиране от менюто „Лекарства“ , примерно, се извеждат всички налични такива. За всеки един запис са налични опците „Преглед“, „Добави“, „Редактирай“ и „Изтрий“, като съответно се изкарва диалогов прозорец в различни режими. Това се отнася за всички регистри.

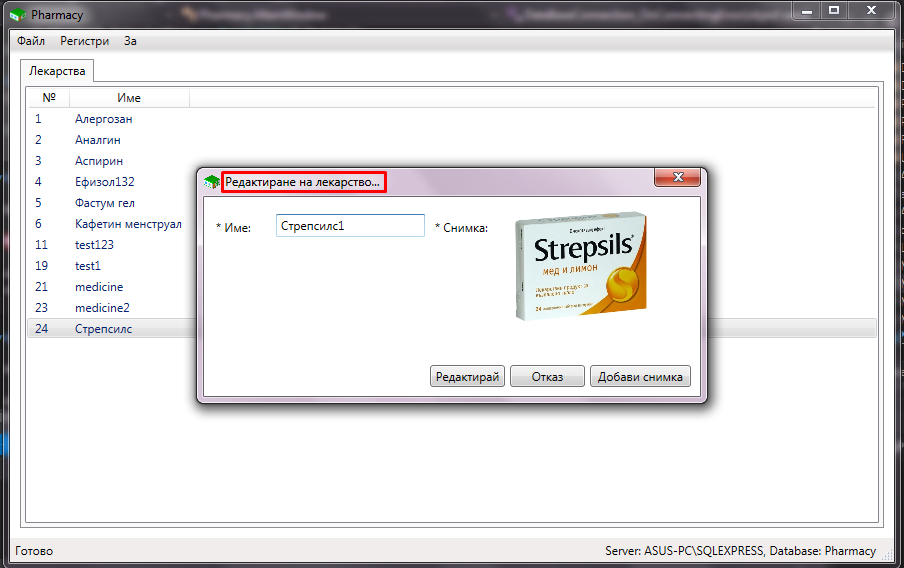
Режим „Преглед“



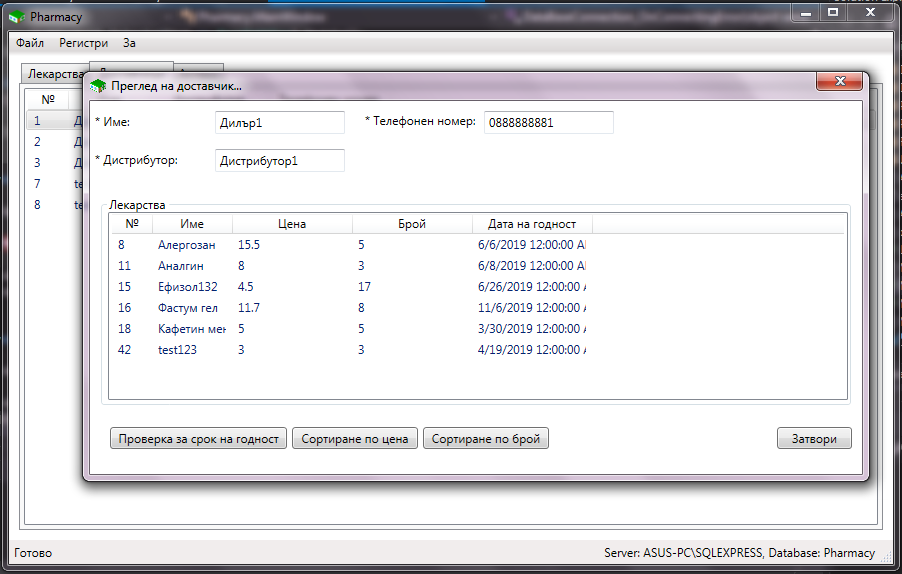
Режим „Добавяне“



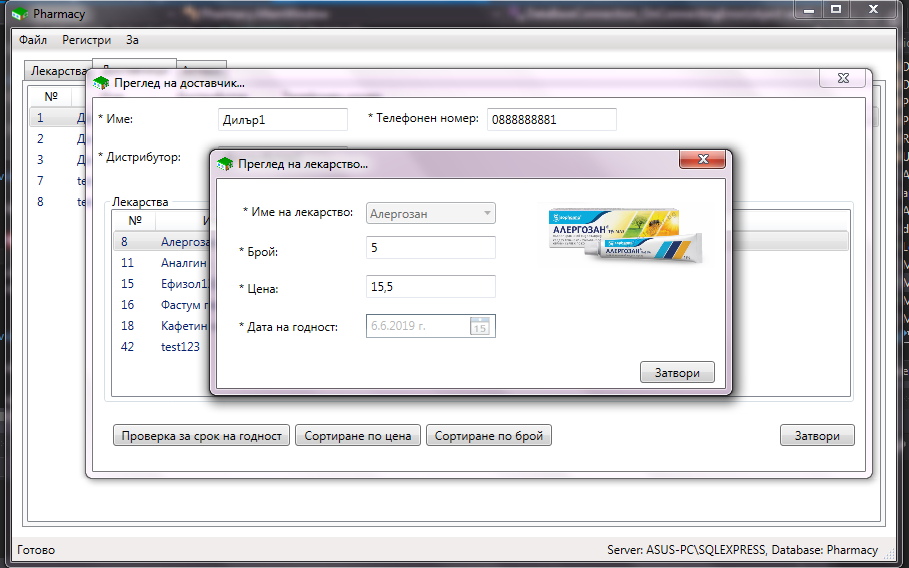
Режим „Редактиране“



При режим „Преглед“ на доставчик можем да видим лекарствата, които може да предлага на съответните аптеки.

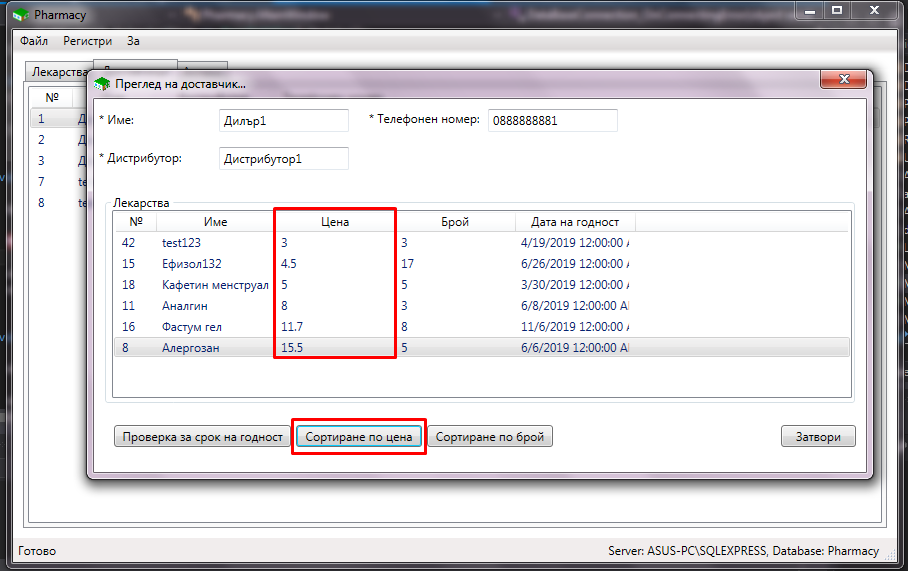


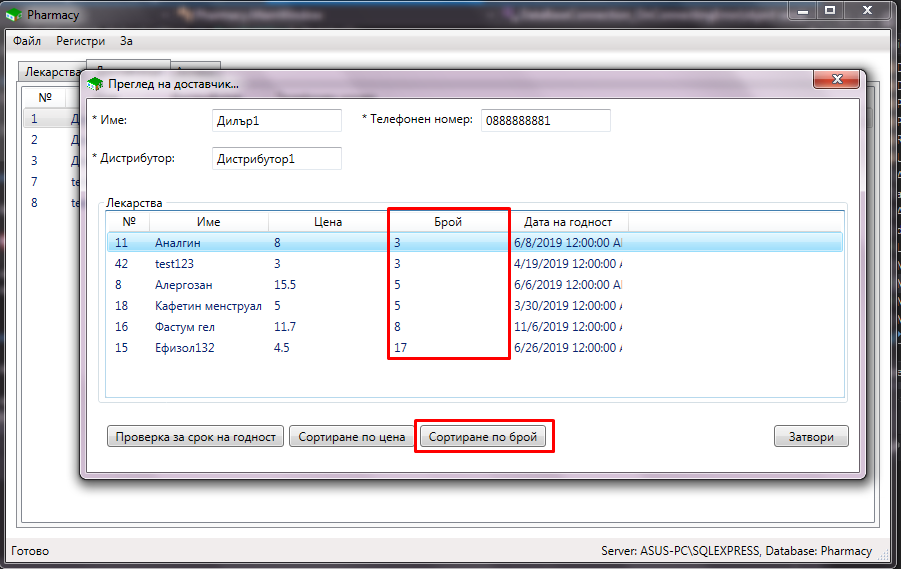
При натискане на преглед на лекарство от контекста на доставчик можем да видим наличната бройка, точната цена, датата на годност, както и изображение на дадено лекарство. Всеки дистрибутор може да има едни и същи лекарства на различни цени. Налични са и другите режими на работа на диалога – редактиране, добавяне и изтриване.



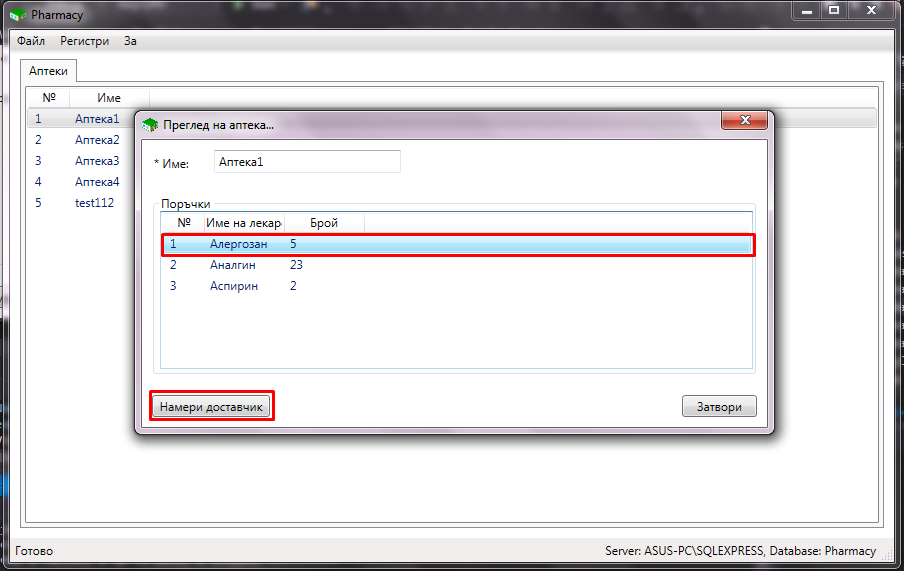
Налични са и функции за проверка на годност на лекарството – ако дадено лекарство е негодно, то броят му се занулява и то се премахва от списъка с предлаганите лекарства. Ако остава един месец до изтичане на годността му, то тогава цената му се намалява с 50%.

Предоставят се функции и за сортиране по цена и по брой на лекарства.

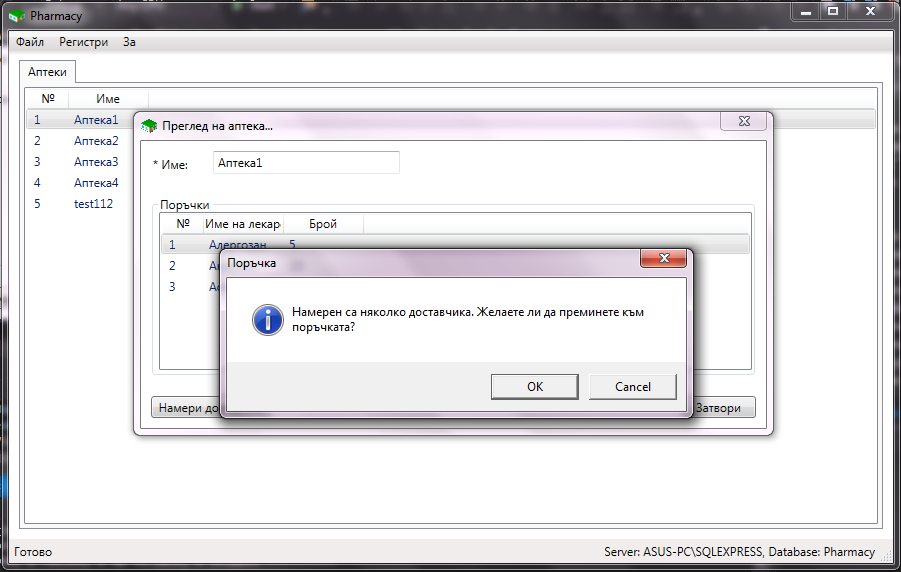




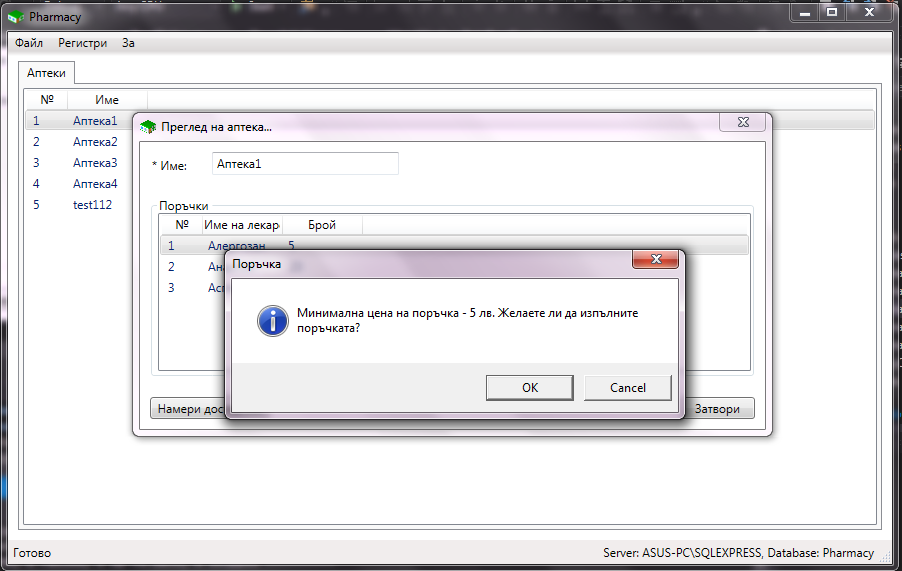
При преглед на аптеки се извеждат всички поръчки на дадена аптека. Налична е функция за намиране на доставчик. При натискане на бутона трябва да имаме селектирана поръчка, в противен случай извежда грешка. Възможно е да има един доставчик, който да има наличната бройка, няколко доставчика или нито един. При наличие на няколко доставчика се избира този с най – ниска цена за даденото лекарство.



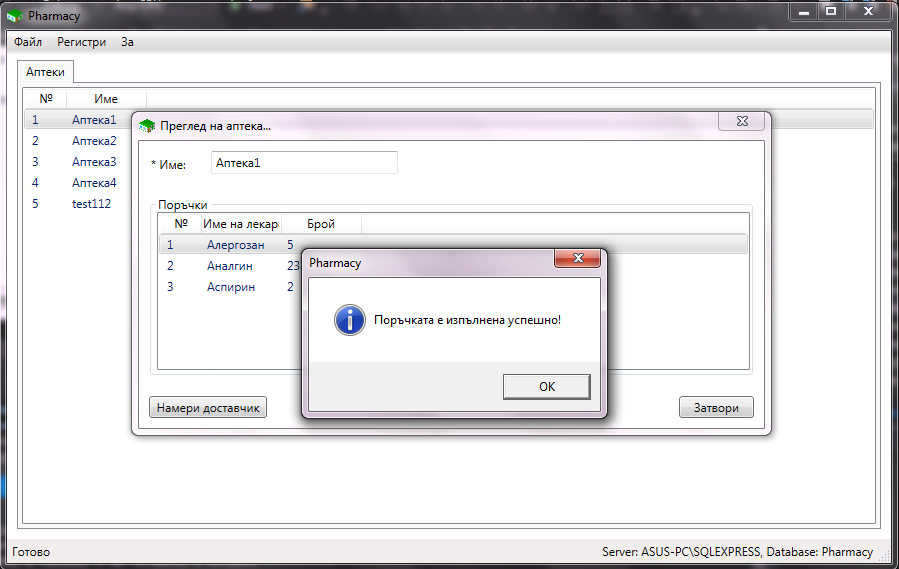
Извежда се съобщение, че са намерени няколко доставчика и се отправя запитване за преминаване към поръчка.



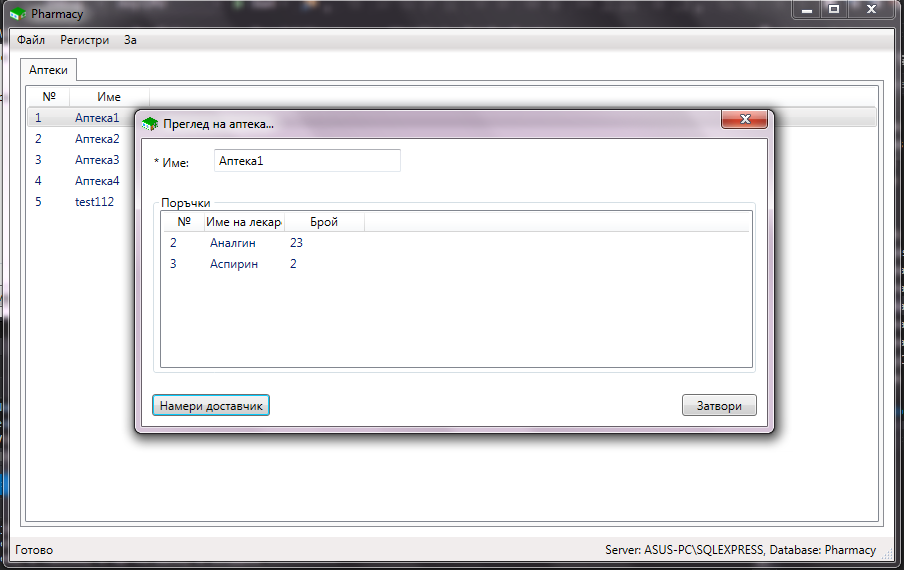
При съгласие се извежда цената на поръчката и се прави последно потвърждение.



При натискане на бутона „ОК“ се извежда съобщение за изпълнена поръчка.



При изпълнена поръчка следва изтриване от списъка за поръчки на аптеката.



**IV. Бъдещо развитие на приложението**

* Използване на клиент – сървър архитектура
* Добавяне на възможност за експорт на данни в Excel
* Добавяне на възможност за импорт на данни от Excel
* Добавяне на възможност за сортиране по всички колони
* Добавяне на възможност за изпълнение на няколко поръчки едновременно
* Добавяне на unit тестове
* Възможност за автоматизиране на някои от функционалностите