Документация

Съдържание

[1.Използвани технологии за реализация 2](#_Toc62587678)

[2.Компоненти 2](#_Toc62587679)

[2.1. View („Изглед“) слоя 2](#_Toc62587680)

[2.2. Model(„Модел“) слоя 3](#_Toc62587681)

[2.3 Controller (“Контролер”) 6](#_Toc62587682)

[3. Връзка между различните компоненти при потребителско взаимодействие 7](#_Toc62587683)

[3.1.AccountingForm 7](#_Toc62587684)

[3.2.AddSupplier 10](#_Toc62587685)

[3.3.AddUser 11](#_Toc62587686)

[3.4.Deliveries 14](#_Toc62587687)

[3.5.EmployeesHome 16](#_Toc62587688)

[3.6.EmployeesMenu 16](#_Toc62587689)

[3.7.EmployeesProfile 17](#_Toc62587690)

[3.8.Login 17](#_Toc62587691)

[3.9.ManagerHome 17](#_Toc62587692)

[3.10.ManagerMenu 18](#_Toc62587693)

[3.11.ManagerProfile 21](#_Toc62587694)

[3.12.ManagerTables 21](#_Toc62587695)

[3.13.ProductsInStockForm 21](#_Toc62587696)

[4. Допълнителни функционалности и становища 23](#_Toc62587697)

Приложението ”RestaurantSystem” е разработено върху основата на MVC (Model-View-Controller) архитектура. Този вид архитектурен шаблон е подходящ за разработка, понеже разделя приложението на три основни компонента и позволява на разработчиците да работят паралелно по различни компоненти, без да оказват влияние или да си пречат един на друг.

Компонентите на MVC шаблона са три: Model, View и Controller

## 1.Използвани технологии за реализация

За създаването на проекта ”RestaurantSystem” е използвана средата за разработка VisualStudio и езика C#. Приложението е WindowsForm Desktop Application. Употребен е Entity Framework и релационна база данни.

## 2.Компоненти

Във всеки слой на приложението се намират съответните му компоненти.

## 2.1. View („Изглед“) слоя

Това е частта от приложението, която представлява представянето на данните.

Изгледите се създават от данните, събрани от данните на модела.View изисква от модела да даде информация, така че да подаде отново изходната презентация на потребителя.

Изгледът също така представлява данните от чатове, диаграми и таблици. Например всеки клиентски изглед ще включва всички компоненти на потребителския интерфейс, като текстови полета, падащи менюта и т.н.

Елементите на View в проекта са:

* AccountingForm – форма за разглеждане на отчетите
* AddUser – форма за добавяне на потребител в системата
* AddSupplier – форма за добавяне на доставчик
* Deliveries – форма за преглед и одобрение на доставките
* EmployeesHome – форма за началното меню на приложението за служител
* EmployeesMenu – форма за менюто на заведените за служител
* EmployeesProfile – форма за профил на служител
* EmployeesTables – форма за масите на заведението за служител
* Login – форма за вписване
* ManagerHome – форма за началното меню на приложението за мениджър
* ManagerMenu – форма за менюто на заведените за мениджър
* ManagerProfile – форма за профил на мениджър
* ManagerTables – форма за масите на заведението на мениджър
* ProductsInStockForm – форма за наличните продукти в заведението

## 2.2. Model(„Модел“) слоя

Модела съхранява данни и свързаната с тях логика. Той представлява данни, които се прехвърлят между компонентите на контролера или друга свързана бизнес логика.

Чрез Entity Framework Code First подход се създава релационна база данни на приложението. Code First ще създаде базата данни и таблици въз основа на дефинираните класове и техните свойства. Така че базата данни се генерира от кода. Когато се стартира код, базата данни ще бъде създадена. EF core предоставя както допълнително ниво на абстракция, така и лесен начин за обработка на данните от базата.

В приложението EF е вписан в Model. В модела се намират класовете DayAccountings, MonthAccountings, YearAccountings, Delivery, DeliveryProducts2, Dish, DishProducts2, Employee, Employer, EmployerReport, Expenses, Product, Role, Supplier, Table и User. EF позволява използването на LINQ заявки за извличане на информация от базата и следи промените настъпили в обектите, които трябва да бъдат изпратени в DB.Таблиците използвани в приложението са:

DayAccountings

* Id – индекс на отчет за деня(първичен ключ)
* DayExpense – разход за деня
* DayIncome – приход за деня
* DayProfit – печалба за деня
* Date – дата на създаване на отчет
* MonthAccointgId – индекс на отчет за месеца(външен ключ)

MonthAccountings

* Id – индекс на отчет за месец(първичен ключ)
* MonthExpense – разход за месец
* MonthIncome – приход за месец
* MonthProfit – печалба за месец
* Date – дата на създаване на отчет
* YearAccountingId – индекс на отчет за година(външен ключ)

YearAccountings

* Id – индекс на отчет за година(първичен ключ)
* YearExpense – разход за година
* YearIncome – приход за година
* YearProfit – печалба за година
* Date – дата на създаване на отчет

Delivery

* Id – индекс на доставка(първичен ключ)
* DeliveryPrice – цена на доставка
* SupplierId – индекс на доставчик(външен ключ)
* DeliveryDate – дата на доставка
* DeliveryQuantity – количество на доставка
* Approved – одобрение на доставка(true или false)

DeliveryProducts

* DeliveryId – индекс на доставка(външен ключ)
* ProductId – индекс на продукт(външен ключ)

Dishes

* Id – индекс на ястие(първичен ключ)
* DishName – име на ястие
* DishPrice – цена на ястие
* DishWeight – тегло на ястие

DishProducts

* DishId – индекс на ястие(външен ключ)
* ProductId – индекс на продукт(външен ключ)

Employee

* EmpId – индекс на служител(първичен ключ)
* FirstName – име на служител
* LastName – фамилия на служител
* Email – е-мейл на служител
* JobPosition – работа на служителя
* Salary - заплата
* ImageUrl – снимка

Employer

* Id – индекс на работодател(първичен ключ)
* FirstName – име на работодател
* LastName – фамилия на работодател
* Salary - заплата

EmployerReport

* Id – индекс на отчет на работодател(първичен ключ)
* EmployerId - индекс на работодател(външен ключ)
* Bill – цена на отчет
* ReportDate – дата на отчет
* DayAccountingId – индекс на отчет за деня(външен ключ)

DishEmployerReport

* Dish\_Id –индекс на ястие(външен ключ)
* EmployerReport\_Id –индекс на работодател(външен ключ)

Expenses

* Id – индекс на разход(първичен ключ)
* Name – име на разход
* Value – стойност на разход
* ExpenseDate – дата на разход

Product

* Id – индекс на продукт(първичен ключ)
* Name – име на продукт
* Quantity – количество на продукт
* Price – цена на продукт
* DeliveryPrice – цена на продукт в доставка

Role

* Id – индекс на роля(първичен ключ)
* Name – име на роля

Supplier

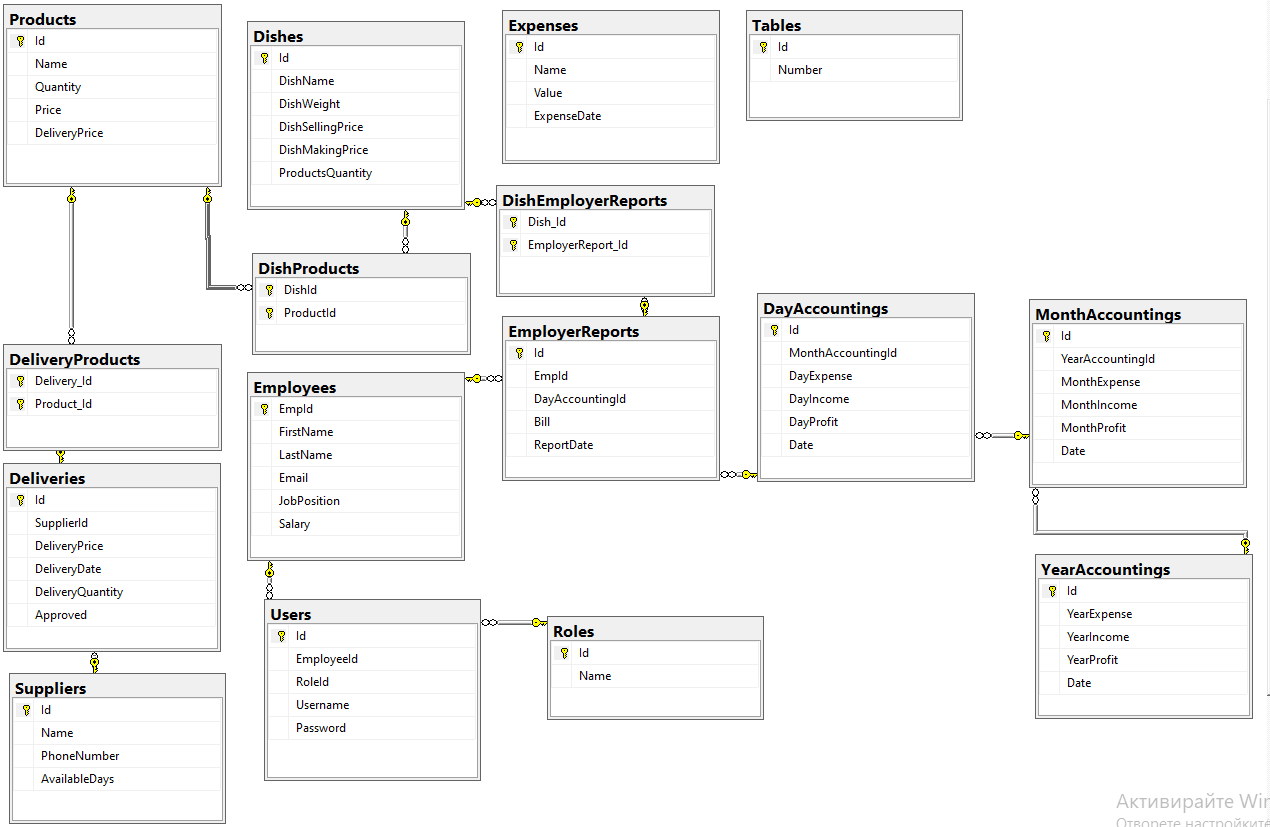
* Id – индекс на доставчик(първичен ключ)
* Name – име на доставчик
* PhoneNumber – телефон на доставчик
* AvailableDays – дни на работа на доставчик

Table

* Id – индекс на маса(първичен ключ)
* Number – номер на маса

User

* Id – индекс на потребител(първичен ключ)
* EmployerId – индекс на работодател(външен ключ)
* RoleId – индекс на роля(външен ключ)
* Username – потребителско има
* Password – парола на потребител



## 2.3 Controller (“Контролер”)

Това е трета част от архитектурата на приложението. Той приема потребителския вход (т.е. данните, които потребителя въвежда, неговите заявки и т.н. и ги преобразува в команди към модела или изгледа.

Контролерът в програмата се състой от:

* DBController – контролер за работа с базата данни, които съдържа методите за обработка на данни
* FormToDBController – контролер за предаване данните от форма към DBController
* LabelController – контролер за текст на Label елемнти във формите
* ListBoxController – контролер за ListBox елементи във формите
* Registration Controller
* TextBoxController – контролер за TextBox елементи във формите

## 3. Връзка между различните компоненти при потребителско взаимодействие

## 3.1.AccountingForm

Формата е разделена на две основни части. Едната е разделена три подчасти(счетоводство за ден,месец и година) като всяка една от тях има по три текстови кутии и по един бутон. Текстът на кутиите се генерира автоматично при натискането на бутона. Дизайнът на другата основна част е изграден чрез ComboBox, в който се избира месеца от годината, а под него в TextBox се задава година и още три текстови кутии.

Методите, които се изпълняват в тази форма от DBController са:

* **public void AddDayAccounting(string dateString, double expenses, double incomes, double profit, List<EmployerReport> dayReports)** – метод, който проверява съществуването на създаден отчет за дена, ако не съществува в базата данни, следва допълнителна проверка за съществуването на отчет за месеца. Ако в базата данни няма отчет за месеца се създава нов отчет за деня и нов отчет за месеца. При съществуващ отчет за месеца се създава отчет за деня и се добавя в базата данни.
* **public bool AddElectricityExpense(string dateString, double elValue)** – метод, който проверява съществуването на разход с име „Ток“ и дата в базата данни. Ако съществува такъв разход връща стройност false. При липса на такъв разход се създава и връща стойност true.
* **public bool AddWaterExpense(string dateString, double wValue)**- метод, който проверява съществуването на разход с име „Вода“ и дата в базата данни. Ако съществува такъв разход връща стройност false. При липса на такъв разход се създава такъв и връща стойност true.
* **public bool AddInternetExpense(string dateString, double iValue)**- метод, който проверява съществуването на разход с име „Ток“ и дата в базата данни. Ако съществува такъв разход връща стройност false. При липса на такъв разход се създава такъв и връща стойност true.
* **public bool EditExpenses(string dateString, double eValue, double wValue, double iValue)**- метод, който променя стойностите на разходите за ток, вода и интернет като създава List<> , който се пълни с разходите отговарящи на датата в dateString. Има и допълнителна променлива check от тип bool. Във while цикъл докато в списъка има по-малко от три елемента се добавя разход с име “невалиден”. Списъкът се обхожда чрез foreach цикъл и се правят проверки за името на разхода и стойностите на eValue, Value и iValue .Ако името на разхода в итерацията отговаря на „Ток“ и стойността на eValue е различна от 0, стойността на разхода с име „Ток“ се променя с тази на eValue и check приема стойност false.

Ако името е „Ток“ и стойността на eValue е 0 започва следваща итерация на цикъла. Ако името на разхода в итерацията отговаря на „Ток“ и стойността на eValue е различна от 0, стойността на разхода с име „Ток“ се променя с тази на eValue и check приема стойност false. Ако името е „Ток“ и стойността на eValue е 0 започва следваща итерация на цикъла.

Ако името на разхода в итерацията отговаря на „Вода“ и стойността на wValue е различна от 0, стойността на разхода с име „Вода“ се променя с тази на wValue и check приема стойност false. Ако името е „Вода“ и стойността на wValue е 0 започва следваща итерация на цикъла.

Ако името на разхода в итерацията отговаря на „Интернет“ и стойността на iValue е различна от 0, стойността на разхода с име „Интернет“ се променя с тази на iValue и check приема стойност false. Ако името е „Интернет“ и стойността на iValue е 0 започва следваща итерация на цикъла.

Ако разходът е с име „невалиден“ и check има стойност true цикъла прекратява.

* **public double GetDayReportIncomes(String dateString)** – метод, който приема стройността на общия приход от отчетите на служителите, чрез списък
* с всички отчети за деня. Създадена е променлива за разхода за деня. Обхожда се списъка и се добавят приходите на отчетите в една променлива.
* **public double GetDayReportExpenses(String dateString)** – метод, който приема стройността на общия разход от отчетите на служителите, чрез списък с всички отчети за деня. Създадена е променлива за разхода за деня. Обхожда се списъка и се добавят разходите на отчетите в една променлива.
* **private double CalculateProfit(double incomes, double expenses)**- приема стойността на разликата между прихода и разхода
* **public List<Expenses> GetExpenses(string dateString)** – метод, който връща списък със допълнителните разходи за ток, вода и интернет

За бутона „Изчисли“ във счетоводство за деня се генерират автоматично данните във текстовите кутии dayIncomes, dayExpenses и dayProfit. Като при натискането му се изпълнява addAccountingForDay\_Click, който:

* Създава променливи от тип string за деня, месеца и годината в момента
* Ако деня и месеца са string-ове с дължина по-малка от нула се добавя „0“ пред тях
* Създава се един string с пълната дата за и се добавя string за часа
* Създават се променливи за приход и разход
* Приход приема стойността на метода GetDayReportIncomes(dateString)
* Разхода приема стойността на метода GetDayReportIncomes(dateString)
* Печалбата се изчислява като се извика метода CalculateProfit(incomes, expenses)

За бутона „Изчисли“ във счетоводство за месеца се генерират автоматично данните във текстовите кутии monthIncomes, monthExpenses и monthProfit. Като при натискането му се изпълнява addAccountingForDay\_Click, който:

* Създава променливи от тип string за месеца и годината в момента
* Ако деня и месеца е с дължина по-малка от нула се добавя „0“ пред тях
* Създава се един string за датата с месец и година и се добавя string за часа
* Създават се променливи за приход и разход.
* Печалбата се изчислява като се извика метода CalculateProfit(incomes, expenses)

В частта „Добавяне на допълнителни разходи“ включва ComboBox за месеците от годината, текстови кутии за годината, ток, вода и интернет и три бутона – „Добави разход“, „Прегледай разходите“ и „Промени разход“. При потребителско взаймодействие:

* Проверяват се елементите за правилно попълнени данни
* Ако годината е различна от четири-цифрено число и не избран месец се появява съобщението "Моля въведете валидни данни за година/месец."
* Ако текстовите кутии са празни се появява съобщението "Моля въведете поне един разход."
* При други грешно въведени данни се появява съобщението "Моля въведете валидни данни."

При бутона „Добави разход“ се изпълнява private void addExpense\_Click(object sender, EventArgs e), който :

* Проверява ако съществува разходите за ток, вода и интернет съществуват и ако съществуват се появява съобщение, че съответния разход е вече добавен. Ако не съществува се викат съответно методите AddElectricityExpense(string dateString, double elValue), AddWaterExpense(string dateString, double wValue), public bool AddInternetExpense(string dateString, double iValue)
* При успешно добавяне се появява съобщението "Разходите за периода (датата) са успешно добавени."

При бутона „Промени разход“ се изпълнява private void editExpenses\_Click(object sender, EventArgs e), който :

* Проверява кои разходи са попълнени, за да се променят
* При проверка със метода public bool EditExpenses() при стойност false за разход със зададения период се появява съобщение "Някой от разходите, които се опитвате да редактирате за периода, не съществуват."
* Ако разходите съществуват се и public bool EditExpenses() е със стойност true се появява съобщение "Разходите са успешно редактирани."

При бутона „Прегледай разходите“ се изпълнява private void seeExpenses\_Click(object sender, EventArgs e), който :

* Проверява за избран месец и правилно попълнена година. При грешни месец и дата се появява съобщението "Моля въведете валидни данни за година/месец."
* Създава се списък на разходите чрез GetExpenses(dateString) като се използват избрания месец и година
* Ако списъкът е празен се връща съобщението "Няма добавени разходи за периода "+датата
* Проверява се ако името на разхода е „Ток“ и текстовата кутия за ток присвоява стойността на разхода за ток
* Проверява се ако името на разхода е „Вода“ и текстовата кутия за вода присвоява стойността на разхода за вода
* Проверява се ако името на разхода е „Интернет“ и текстовата кутия за интернет присвоява стойността на разхода за интернет

## 3.2.AddSupplier

Формата се състой от ListBox, три текстови кутии за име, телефон и свободни дни на доставчик и три бутона за добавяне, редактиране и премахване на доставчик. Методите, които се използват във формата от контролера са:

* **public ICollection<Supplier> LoadSuppliers()** – метод, който връща колекция със всички доставчици във базата данни.
* **public void AddSupplier(string supplierName , string supplierPhone , stringsupplierAvailableDays)** – метод, който приема името, три променливи от тип string, за да създаде доставчик със име,телефон и свободни дни във базата данни
* **public void EditSupplier(string supplierName, string supplierPhone, string supplierAvailableDays)**-метод, който избира доставчик по име и му променя телефона и свободните дни съответно със supplierName и supplierPhone и се запазва във базата данни
* **public void DeleteSupplier(string supplierName)** – метод, който приема името на доставчик и го изтрива от базата данни
* **public Supplier SelectSupplierByName(string name)** - метод, който приема променлива от тип string, за да върне доставчик от базата със съответното име

При стартирането на формата ListBox-а се запълва със всички доставчици чрез метода LoadSupplier(). В него се използва метода LoadSuppliers(). За всеки един доставчик се създава нов елемент в ListBox-а и се изписват неговите данни. При натискането на бутоните се изпълняват следните събития:

За „Добави доставчик“ е private void addSupplierButton\_Click(object sender, EventArgs e):

* Прави се валидация на текстовата кутия за име,телефон и свободни дни
* Ако името е празно се визуализира съобщението "Моля въведете валидно на доставчик."
* За телефон се проверява дали дължината на въведените данни в текстовата кутия за телефон са с дължина девет или десет. Ако е различно от девет или десет излиза съобщението "Моля въведете валиден телефон"
* За свободните дни се създава масив от тип string с всичките дни от седмицата. Ако попълнените данни във текстовата кутия са с дължина по-малки от 5 или не съвпадат с дните от седмицата излиза съобщението "Моля въведете валидни дни от седмицата"
* Декларирани са променливи spName, spPhone, spAvailableDays, който взимат текста въведен съответно във supplierName, supplierPhone и supplierAvailableDays
* Създава се доставчик и се взима от базата чрез SelecetSupprlierByName(spName)
* Ако доставчика не е празен елемент се изпълнява метода public void AddSupplier(string supplierName , string supplierPhone , stringsupplierAvailableDays)
* Ако съществува такъв доставчик се визуализира съобщението "Този доставчик вече е добавен."

За „Редактирай доставчик“ е private void editSupplier\_Click(object sender, EventArgs e):

* Отново се правят същите валидации за текстовите кутии
* Декларирани са променливи spName, spPhone, spAvailableDays, който взимат текста въведен съответно във supplierName, supplierPhone и supplierAvailableDays
* Ако името на доставчика конфигурира в ListBox-а се показва съобщението "Този доставчик вече е добавен."
* Ако името на доставчика липса в ListBox-а, чрез метода public void EditSupplier(string supplierName, string supplierPhone, string supplierAvailableDays) се променя доставчика и списъка с доставчици се променя с настоящите редакции

За „Премахни доставчик“ е private void deleteSupplier\_Click(object sender, EventArgs e):

* Избира се елемент от ListBox-а и чрез метода Split (',', '-') се разделя на отделни string-ове.
* Декларира се променлива за името на доставчик като тя приема първия елемент от масива със string-ове.
* Прави се проверка за ибран елемент от ListBox-а.
* Ако не е избран се показва съобщението "Изберете доствчик за премахване.
* Ако е наличен избран елемент се изпълнява метода public void DeleteSupplier(string supplierName)

## 3.3.AddUser

Дизайнът на AddUser се състой в един ComboBox, който се съдържат служителите в базата данни, три текстови полета за длъжност, потребителско име и парола, и един бутон за запазване на въведените данни. Методи използвани от контролера във формата са:

* **public Employee GetEmployeeByFirstAndLastName(string firstName, string lastName)** – метод, който приема две променливи от тип string за име и фамилия на служител. Чрез променливите връща служителя с името и фамилията.
* **public String GetEmployerJobPosition(string firstName, string lastName)** – метод, който приема две променливи от тип string за име и фамилия на служител и връща string отговарящ на позицията на служителя. Служителят се намира чрез Employee GetEmployeeByFirstAndLastName().
* **public String GetEmployerUserName(string firstName, string lastName)** – този метод приема две променливи от тип string за име и фамилия на служител. Създава се променлива от тип Emploeey , чрез името и фамилията се търси в базата данни , и променлива от тип User. Чрез индекса на служителя се проверява за създаден вече такъв User. Прави се проверка, ако променлива от тип User е празна. При липса на такъв потребител метода връща "".При наличието на такъв служител се връща string за потребителското име на съответния служител.
* **public String GetEmployerPassword(string firstName, string lastName)** - този метод приема две променливи от тип string за име и фамилия на служител. Създава се променлива от тип Emploeey , чрез името и фамилията се търси в базата данни , и променлива от тип User. Чрез индекса на служителя се проверява за създаден вече такъв User. Прави се проверка, ако променлива от тип User е празна. При липса на такъв потребител метода връща "".При наличието на такъв служител се връща string за паролата име на съответния потребител.
* **public Role GetRoleByName(string name)** – метод, който приема една променлива и чрез нея връща от базата данни роля със съответното име.
* **public void AddUserToDataBase(string username, string password, string roleName, string firstName, string lastName)** –в този метод се приемат пет променливи от тип string за потребителско име, парола, име на роля, име и фамилия. Чрез името и фамилията и проверка в базата данни се създава променлива от тип Employee. Създава се променлива от тип Role и се взима от базата данни чрез GetRoleByName(string name). Декларирана е и променлива за проверка от от тип User. Прави се проверка дали променливата е празна, ако не е се връща ролята като се избере от базата данни чрез името. При празна променлива се създава нов User със данните, които приема метода.

Като се стартира формата, методът public void LoadEmployees() добавя за всеки един служител в базата данни в ComboBox-а по един елемент със неговото име и фамилия. При избор на служител от ComboBox-а се изпълнява метода public void LoadEmployeeProperties(). В него се декларират променливите string employeeName(приема текста на избрания елемент) string employeeFirstName(чрез метода Split(' ') приема стойността на първия елемент в employeeName) и string employeeLastName(чрез метода Split(' ') приема стойността на втория елемент в employeeName). Използват се методите public String GetEmployerJobPosition(string firstName, string lastName), public String GetEmployerUserName(string firstName, string lastName), public String GetEmployerPassword(string firstName, string lastName) и се променя текста в текстовите кутии съответно на JobPosition, usernameTextbox, passwordTextbox.

Методът, който се изпълнява при натискането на бутона „Запази“ е private void Save\_Click(object sender, EventArgs e):

* Той започва съ проверка от тип if. Изпълняват се булевите методи ValidUsername() и trueValidPass().
* ValidUsername()- метод за проверка на валидно потребителско име. В него се декларира string със специални елементи за потребителско име и започват серия проверки.
* Ако текста в usernameTextbox е дължина по-малка или равна на три се визуализира съобщението "Моля въведете име от поне 4 символа.".
* Ако текста в usernameTextbox е дължина по-голяма от 20 се появява съобщението "Въведеното име е твърде дълго.".
* При наличето на някой от специалните символи, съобщението е "Името може да се състои само от букви и цифри.".
* trueValidPass() – метод за проверка на валидна парола. В него се декларират променливи от тип bool – passLength, hasDigit, hasUpper, hasLower, hasSpecialChar, които първоначално са false и string със специални елементи за потребителско име и започват if проверки.
* При въведен текст с дължина по-голяма или равна на 6 passLength = true. Ако дължината е по-малка от 6 на формата излиза Label с тектс "Моля въведете парола с поне 6 символа."
* Във foreach цикъл се обхождат символите на въведения текст за парола.
* Ако има символ, който е цифра, hasDigit = true.
* Ако има символ, който е главна буква, hasUpper = true.
* Ако има символ, който е малка буква, hasLower = true.
* Във foreach се обхождат специалните символи и вътре се прави проверка ако паролата има някой от тях. При изпълнено условие hasSpecialChar = true.
* В нова проверка се проверява ако, hasSpecialChar е със стойност false. Ако е така се на формата излиза съобщението "Моля въведете парола, която да съдържа поне един специален символ, различен от буква или цифра."
* В друга проверка се проверяват всички други булеви променливи. Ако всички са true на метода се връща стойност false.
* Ако някоя от променливите е със стойност false се показва на формата текста "Моля въведете парола, в която има поне една цифра, малка и голяма буква."
* При успешни валидации за потребителско име и парола се изпълнява метода public void AddUserToDataBase(string username, string password, string roleName, string firstName, string lastName) и излиза съобшението "Потребителят е успешно създаден!"

## 3.4.Deliveries

Оформлението на Deliveries формата се състой във два ListBox-а за чакащи одобрение доставки и одобрени доставки; TextBox за продукт, количество за добавяне и име на доставчкик; три бутона за правене, редакция и одоборение на доставка. Логиката на формата се изпълнява от следните методи от „Контролера“:

* **public ICollection<Delivery> LoadDeliveries()** –метод, който връща спъсък със всички доставки запазени във базата данни
* **public void AddDelivery(string productName, int productQuantity, string supplierName)** – метод, който приема три променливи за името на продукта, количеството на продукта и името на доставчика. Декларира се колекция с доставките като се използва LoadDeliveries(). От базата се взима продукта със зададеното име и доставчика със зададеното име. Прави се празна колекция на за продукти и се добавя продукта, извлечен по-рано от базата, добавя се и количеството зададено в productQuantity към неговото.

Създава се нова доставка, която е неодобрена.

* **public ICollection<Product> GetDeliveryProducts(int id)** – метод, който прима индекса на доставка и връща продуктите, който са свързани с нея, в колекция
* **public void EditDelivery(string productName,int productQuantity,string supplierName)** – метод, който приема три променливи за името на продукта, количеството на продукта и името на доставчика. Декларира се колекция с доставките като се използва LoadDeliveries(). От базата се взима продукта със зададеното име и доставчика със зададеното име. Чрез foreach цикъл се обхожда колекцията с доставки. Декларира се колекция, която се приема стойността на метода public ICollection<Product> GetDeliveryProducts(int id) и се правят няколко проверки. Ако доставчика съвпада със доставката в текущата итерация на цикъла се влиза в нова проверка, която проверява дали продукта е част от продуктите на доставката. При изпълнено условие, доставката в итерацията се променя като се изменят количеството, цената като се добави количеството зададено в метода. Промяна има и при количеството на продукта. Ако условието не е изпълнено продукта, деклариран по-рано, се добавя в доставката
* **public void ApproveDelivery(string supplierName,double dPrice)** – метод, който приема променливи за името на доставчик и цената на доставка. Чрез тях се избира доставката с тези параметри от базата данни и нейната стойност Approved се променя на true
* **public void AddProductsToDelivery(Delivery delivery, ICollection<Product> productsInDelivery)** – метод, който добавя продуктите в доставката
* **public Product SelectProductByName(string name)** – метод, който приема стринг с името на продукта и връща продукт от базата данни със същото име
* **public Supplier SelectSupplierByName(string name)** - метод, който приема стринг с името на доставчика и връща доставчик от базата данни със същото име

При стартирането на формата се изпълнява метода LoadDeliveries() деклариран в кода на формата. Този метод визуализира доставките в двата ListBox-a. Бутоните във формата са „Направи доставка“, „Промени доставка“ и „Одобри доставка“. При натискането на всеки един бутон се изпълняват различни събития. При натискането на „Направи доставка“ отговаря private void addDelivery\_Click(object sender, EventArgs e), който:

* Започва се проверка за правилно въведени данни
* Ако количеството зададен в полето ProductQuantity е по-малко или рано на 5 се визуализира съобщение за грешка "Моля въведете количество по - голямо 5"
* Ако името на продукта е с дължина по-малка от 2 или името на доставчика е с дължина по-малка от 3 се изписва "Моля въведете валидно име на продукт / доставчик"
* При въвеждане на други невалидни данни се появява съобщението "Моля въведете валидни данни."
* При липса на грешни данни се декларират променливи, които приемат стойностите на productName.Text, productQuantity.Text, deliverySupplier.Text , и продукт, който приема стойността метода SelectProductByName(string name) и доставчик, които по същия начин пеиема public Supplier SelectSupplierByName(string name)
* Започва проверка за това дали променливата за продукт е празна. Ако е празна се появява съобщението "Този продукт не съществува." Ако има стойност променливата се почва нова проверка за стойността на доставчика
* Ако и тя е празна се показва "Този доставчик не съществува."
* При наличие на стойности в двете променливи се вика метода public void AddDelivery(string productName, int productQuantity, string supplierName) и се актуализира списъка със доставки

За бутона „Промени доставка“ отговаря събитието private void editDelivery\_Click(object sender, EventArgs e), който:

* Започва със същата проверка за валидация на попълнената информация в текстовите полета
* При изпълнена валидация се декларира масив от тип стринг, който приема елементите в избрания елемент от списъка за чакащи доставки като се използва метода Split().
* Създават се променливи, които приемат стойностите на productName.Text, productQuantity.Text, deliverySupplier.Text , и продукт, който приема стойността метода SelectProductByName(string name).
* Проверява се дали е избран елемент от waitingDeliveries. Ако не е се връща съобщението "Изберете поръчка от неодобрените"
* Проверява се дали първия елемент от стринг масива отговаря на deliverySupplier.Text. Съобщението, което излиза е "Този доставчик не е за избраната поръчка."
* Допълнителна проверка се прави ако променлива от тип Product е със стойност null. Ако е така се показва текста "Този продукт не съществува"
* Ако доставчика отговаря и продукта има стойност се вика метода EditDelivery(string productName,int productQuantity,string supplierName) и се зареждат на ново доставките със LoadDeliveries()

За бутона „Одобри доставка“ отговаря събитието private void approveDelivery\_Click(object sender, EventArgs e), който:

* В него се декларират масив от тип string, който се запълва с избрания елемент от чакащите доставки, използвайки метода Split() по символите за запетая, тире и две точки и променлива за текста изписан във текстовата кутия deliverySupplier.
* Прави се проверка за това, дали е избран елемент от чакащи доставки. Ако не е, на формата излиза "Изберете поръчка от неодобрените."
* Ако първия елемент от масива не отговаря на текста на deliverySupplier се показва съобщение за грешка "Грешен достачик."
* Ако името на доставчика конфигурира в масива се вика метода за промяната на булевото пропърти на доставка, public void ApproveDelivery(string productName,int productQuantity,string supplierName)

## 3.5.EmployeesHome

Тази форма е предназначена за служителите на заведението без мениджъра и администратора. Оформлението на тази форма се от три части. В първата се виждат три бутона- „Профил“, „Маси“ и „Меню“ и при натискането на всеки един бутон се изпълнява събитие:

* За „Профил“ това е private void ProfileBtn\_Click(object sender, EventArgs e) – отваря формата EmployeesProfile
* За „Маси“ това е private void Table\_Click(object sender, EventArgs e) – отваря формата EmployeesTables
* За „Меню“ това е private void Menu\_Click(object sender, EventArgs e) – отваря формата EmployeesMenu

Тези форми се отварят във другата част на формата, която се състой от панела panelDesktopPanel. В горната част на се намират бутоните за минимизация, разширяване на цял екран и затваряне на приложението. При натискането им се случват следните събития:

* private void BtnMinimize\_Click\_1(object sender, EventArgs e) – минимизира формата
* private void BtnMaximize\_Click\_1(object sender, EventArgs e) – прави формата на цял екран
* private void BtnCloseChildForm\_Click\_1(object sender, EventArgs e) – затваря всички отворени форми

## 3.6.EmployeesMenu

Формата EmployeesMenu предназначена за служителите на заведението(сервитьори и бармани). Дизайнът представлява един DataGridView, в който излизат ястията в менюто на ресторанта .Той разделен на пет колони, които представляват пропъртита на ястието: Id, DishName, DishSellingPrice, DishWeigth и DishProducts.

## 3.7.EmployeesProfile

EmployeesProfile формата е предназначена за преглед на профила на служителя, който в момента използва приложението. Съдържа пет текстови полета, които се попълват автоматично от базата данни. Първият TextBox показва името на служителя, втория- неговата фамилия, третия- неговия е-мейл, четвърия – длъжността му, а в петия се изписва заплатата му.

## 3.8.Login

## 3.9.ManagerHome

ManagerHome е форма специална за администратор и мениджър на заведението. Тя има достъп до всички функционалности на приложението. Дизайнът и е съставен в три отделни части – panelDesktopPanel, PanelMenu и PanelTittleBar. Като в PanelMenu се намират бутоните, които отварят другите форми в другата част на формата panelDesktopPanel. В PanelTittleBar се намират бутони за минимизация, разгъване на цял екран и затваряне на приложението.

Бутоните в PanelMenu са „Добави нов потребите“, „Служители“, „Маси“, „Меню“, „Наличност“, „Доставки“, „Добави доставчик“ и „Счетоводство“. Като се натисне всеки един бутон се изпълнява специално за него събитие:

* За бутона „Добави нов потребите“ е private void AddUser\_Click(object sender, EventArgs e) – отваря се формата AddUser в панела panelDesktopPanel
* За бутона „Служители“ е private void Employees\_Click(object sender, EventArgs e) - отваря се формата ManagerProfile в панела panelDesktopPanel
* За бутона „Маси“ е private void Table\_Click(object sender, EventArgs e) - отваря се формата ManagerTables в панела panelDesktopPanel
* За бутона „Меню“ е private void Menu\_Click(object sender, EventArgs e) - отваря се формата ManagerMenu в панела panelDesktopPanel
* За бутона „Наличност“ е private void Stock\_Click(object sender, EventArgs e) - отваря се формата ProductsInStockForm в панела panelDesktopPanel
* За бутона „Доставки“ е private void Delivery\_Click(object sender, EventArgs e) - отваря се формата Deliveries в панела panelDesktopPanel
* За бутона „Добави доставчик“ - отваря се формата AddSupplier в панела panelDesktopPanel
* За бутона „Счетоводство“ е private void Accounting\_Click(object sender, EventArgs e) - отваря се формата AccountingForm в панела panelDesktopPanel

Бутоните в PanelTittleBar са три като при натискането се изпълняват следните събития:

* За бутона btnMinimize отгваря метода private void BtnMinimize\_Click(object sender, EventArgs e) – минимизара се формата ManagerHome
* За бутона btnMaximize отговаря метода private void BtnMaximize\_Click(object sender, EventArgs e) – разгъва се формата ManagerHome на цял екран
* За бутона btnClose отговаря метода private void BtnClose\_Click(object sender, EventArgs e) – затваря ManagerHome

## 3.10.ManagerMenu

Дизайнът на формата е разделен на части. В едната се съдържат два ListBox-а, които отговарят за името на ястието и параметрите в него. Другата част е съставена от няколко текстови кутии, отговарящи за името на ястие, цената му, продуктите в него, количество на продуктите в него и теглото му. Има и три бутона до текстовите полета – „Добави ново ястие“, „Редактирай ястие“, „Изтрий ястие“. За логиката на формата се викат следните методи от контролера на приложението:

* **public ICollection<Dish> LoadDishes()** – метод, който връща колекция с всички ястия в базата данни
* **public void EditDish(string dName, double dPrice, double dWeight, ICollection<Product> productsInDish, string productsQuantities)** – метод, който приема стойностите на променливите string dName, double dPrice, double dWeight, string productsQuantities и колекция от елементи от вида Product. В него се декларира Dish dish, която чрез проверка за наличие в базата данни по стойността на dName се извлича ястие с това име и dishProductsIds, която съдържа индексите на продуктите в ястието. Прави се проверка за dish. Ако има записано ястие в променливата, то за това ястие се променят неговите цена за продан и тегло с dPrice и dWeight. Премахват се старите продукти за приготвяне и се добавят новите. Обхожда се колекцията productsInDish като за всеки продукт се взима цената му. Има и допълнителна променлива, която взима количеството на продуктите за приготвяне. В цикъла се изчислява цената за приготвяне на ястието
* **public void AddProductsToDish(Dish dish, ICollection<Product> products)** – метод, който добавя продуктите в ястиети
* **public bool RemoveDish(string dName)** – булев метод, в който се създава променлива от тип dish и чрез стойността на dName се търси в базата данни. Ако има такъв продукт, на функцията се връща истинна стойност, ако няма се връща стойност false
* **public List<DishProducts> SelectAllDishProducts(int id)** – метод, който приема индекса на ястие връща в списък всички индекси на продуктите свързани с него
* **public Dish SelectDishById(int id)** – метод, който прием индекса на ястие и връща ястието, което отговаря на индекса в басата данни
* **public Dish SelectDishByName(string name)** – метод, който приема името на ястие и връща ястието, което отговаря на това име
* **public void AddDish(string dName, double dPrice, double dWeight, ICollection<int> productsInDish, string productsQuantities)** – метод, който приема string dName, double dPrice, double dWeight, ICollection<int> productsInDish, string productsQuantities, с които създава ястие с тези параметри в базата данни. В него се създава и връзката между ястието и продуктите, необходими за приготвяне

Елементите в menuItems и menuItemsParameters се запълват чрез метода във формата LoadDishes(). Метода създава по един елемент в menuItems за всяко едно ястие в базата данни. В menuItemsParameters за всяко едно ястие се добавят неговите параметри – цена за продан, тегло, продукти и необходимото количество за приготвяне.

За трите бутона на формата(„Добави ново ястие“, „Редактирай ястие“, „Изтрий ястие“), при натискането им, се случват събитията:

За бутона „Добави ново ястие“ е private void addMenuItem\_Click(object sender, EventArgs e):

* Първо се прави проверка за въведени грешни данни в текстовите кутии
* Ако теглото на ястието е по-малко или равно на нула, на формата излиза съобщението "Моля въведете тегло по - голямо от нула"
* Ако цената на ястието е по-малка или равна на нула, на формата излиза съобщението "Моля въведете цена на ястие по - голяма от нула"
* Ако в текстовите кутии за име на ястие и имена на продукти в ястието са с дължина по-малка или равна на три, съобщението, което ще се появи е "Моля въведете валидно име на ястие / продукти в ястието"
* Ако броят на въведените име на продукти не отговарят на броя на количествата написани в productQuantities ще се появи съобщението "Моля въведете съответните количества за всяко едно ястие"
* При въвеждане на други грешни данни, текста, който ще се визуализира на формата е "Моля въведете валидни данни."
* В метода се декларират – колекция за индекси, масив от стрингове за въведените продукти, цената на ястието, теглото на ястието, името на ястието, масив от стрингове от количествата на продуктите за приготвяне на ястието и една променлива от тип int за броене
* Във foreach цикъл се обхода масива за имена на продуктите
* Създава се продукт с името в текущата итерация, като се използва метода public Product SelectProductByName(string name), който връща продукт от базата с това име
* Прави се проверка ако продукта е празен
* Ако има такъв продукт се добавя към индекса му към масива с индекси за продукти и се прави нова проверка
* Ако количеството на продукта в базата е по-малко от зададено в количеството за необходим брой на продукти се изписва съобщението "Количеството на продукта " + името на продукта + " не достига." и се продължава към следващата итерация на цикъла
* Ако няма такъв продукт в базата данни излиза съобщението "Въведеният продукт " + името на продукта + " не е наличен." и се продължава към следващата итерация на цикъла
* Преди да свърши итерацията на цикъла променливата за броене се увеличава с едно
* Накрая в метода се вика от контролера public void AddDish(string dName, double dPrice, double dWeight, ICollection<int> productsInDish, string productsQuantities) и LoadDishes()

За бутона „Редактирай ястие“ е private void private void editMenuItem\_Click(object sender, EventArgs e):

* Първо се прави проверка за въведени грешни данни в текстовите кутии
* Ако теглото на ястието е по-малко или равно на нула, на формата излиза съобщението "Моля въведете тегло по - голямо от нула"
* Ако цената на ястието е по-малка или равна на нула, на формата излиза съобщението "Моля въведете цена на ястие по - голяма от нула"
* Ако в текстовите кутии за име на ястие и имена на продукти в ястието са с дължина по-малка или равна на три, съобщението, което ще се появи е "Моля въведете валидно име на ястие / продукти в ястието"
* Ако броят на въведените име на продукти не отговарят на броя на количествата написани в productQuantities ще се появи съобщението "Моля въведете съответните количества за всяко едно ястие"
* При въвеждане на други грешни данни, текста, който ще се визуализира на формата е "Моля въведете валидни данни."
* В метода се декларират – колекция за продуктите на ястието, масив от стрингове за въведените продукти, цената на ястието, теглото на ястието, името на ястието, масив от стрингове от количествата на продуктите за приготвяне на ястието и една променлива от тип int за броене
* Прави се проверка ако ListBox-a с имена на ястия съдържа въведено в текстовата кутия име на ястие
* Ако не е сред елементите на ListBox-а се изписва съобщението "Ястието, което се опитвате да редактирате, не съществува."
* Ако е сред тях във foreach цикъл се обхода масива за имена на продуктите
* Създава се продукт с името в текущата итерация, като се използва метода public Product SelectProductByName(string name), който връща продукт от базата с това име
* Прави се проверка ако продукта е празен
* Ако има такъв продукт се добавя към колекцията с продукти и се прави нова проверка
* Ако количеството на продукта в базата е по-малко от зададено в количеството за необходим брой на продукти се изписва съобщението "Количеството на продукта " + името на продукта + " не достига." и се продължава към следващата итерация на цикъла
* Ако няма такъв продукт в базата данни излиза съобщението "Въведеният продукт " + името на продукта + " не е наличен." и се продължава към следващата итерация на цикъла
* Преди да свърши итерацията на цикъла променливата за броене се увеличава с едно
* Накрая в метода се вика от контролера public void public void EditDish(string dName, double dPrice, double dWeight, ICollection<Product> productsInDish, string productsQuantities)

За бутона „Изтрий ястие“ е private void private void deleteMenuItem\_Click(object sender, EventArgs e):

* Декларира се променлива string dName = itemName.Text
* Прави се проверка за дължината на текста въведен в полето за име на ястия. Ако е по-малка от 3 се показва съобщението "Моля въведете валидно име на ястие"
* Следва проверка с метода RemoveDish(dName)
* Ако метода има стойност true се трие ястието с въведеното име
* Ако има стойност false се визуализира на формата съобщението "Въведеното ястие " + dName + " не съществува."

## 3.11.ManagerProfile

Основната функционалност, която се изпълнява във формата ManagerProfile е регистрация на служител, чрез вече вписан в системата мениджър. Формата се състои от dataGridView, panel и бутони. В dataGridView се съдържа списък с вече регистрираните служители, като колоните Employee id, First name, Last name, Email, Job position и Salary отговарят за съответните данни на даден потребителски профил. В panel се попълват данните за всеки служител в полетата означени със съответния етикет, които след успешна регистрация се зареждат в dataGridView. Бутоните под panel служат за изпълняване на функционалностите регистрация, редакция, изтриване, запазване и отказ от промени. Методите, които се изпълняват в тази форма са следните:

* private void ManagerProfile\_Load(object sender, EventArgs e) – метод, който зарежда формата, стиловете и дизайна й и данните в dataGridView
* private void addBtn\_Click(object sender, EventArgs e) – метод, който добавя създадения профил в dataGridView, като го поставя на последно място в списъка
* private void editBtn\_Click(object sender, EventArgs e) – метод, който подготвя даден профил за редакция, който преди това е избран, чрез кликване върху него в dataGridView
* private void cancelBtn\_Click(object sender, EventArgs e) – метод, който отменя направените промени
* private void deleteBtn\_Click(object sender, EventArgs e) – метод, който изтрива маркирания профил на служител, като го изтрива и от базата данни
* private void saveBtn\_Click(object sender, EventArgs e) – метод, който потвърждава регистрацията и редакцията на профил. Използва валидация с регулярни изрази за проверка на попълнените полета с данни. При успешна регистрация или редакция данните се визуализират в dataGridView

## 3.12.ManagerTables

Във формата ManagerTables се изпълнява функционалността, свързана с добавяне и обработка на поръчки. Състои се от три groupBox-a. В първия от тях се избира маса от падащ списък, за която ще се изпълнява поръчката. Има и възможност за добавяне на маса с пореден номер, която се записва в списъка. Във втория groupBox се добавят продукти от падащ списък, свързани с формата за меню, избира се количество и автоматично се зарежда съответната цена. При кликване на бутона "Добави", данните за ястието се добавят в третия groupBox форматирани по предварително зададен начин. Също така чрез този бутон се намалява количеството на продуктите, участващи в дадено ястие, във формата за наличност, както и в базата данни. Освен данни за ястието, в третия groupBox автоматично се зарежда настоящата дата и избраната от списъка маса и се изчислява крайната сума за цялата поръчка. При кликване на бутона "ОК", данните се изчистват. Функционалността се изпълнява чрез следните методи:

* private void ManagerTables\_Load(object sender, EventArgs e) - зареждане на датата
* public void FillCategoryList() - зареждане на ястията от менюто в падащия списък
* private void categoryList\_SelectedIndexChanged(object sender, EventArgs e) - зареждане на цената на избраното ястие
* private void addItem\_Click(object sender, EventArgs e) - добавяне на ястие към поръчката чрез валидация, изчисляване на цената по количеството, намаляване на количеството в наличността, добавяне в отчетите за деня
* private void tablesAddBtn\_Click(object sender, EventArgs e) - добавяне на маса в падащия списък
* public void FillTablesList() - зареждане на масите от базата данни

## 3.13.ProductsInStockForm

Формата ProductsInStockForm е разделена на два GroupBox-a. В едни се намират два ListBox-a – един за наличните продукти в базата, а другия е за количество/цена за брой/цена за доствка. Вдругия се намират четири текстови кутии за име на продукт, налично количество, цена за брой и цена за доставка и два бутона за добавяне на продукт и редакция на продукт. Логиката, която с реализира във формата, използва следните методи от контролера:

* **public ICollection<Product> LoadProduct()** – метод, който връща колекция със всички налични продукти от базата данни
* **public void AddProduct(string name, int quantity, double price, double dlprice)** - метод, който приема променливите string name, int quantity, double price, double dlprice. Чрез тези променливи създава продукт с име name, количество quantity, цена за брой price и цена за доставка dlprice. Добавя се в базата данни и се запзват промените
* **public void EditProduct(string name, int quantity, double price, double dlprice)** –метод, който приема променливите string name, int quantity, double price, double dlprice. Създава се Product product, който се извлича от базата данни по името записано в name
* **public Product SelectProductByName(string name)** – метод, който приема стринг с името на продукта и връща продукт от базата данни със същото име

В формата има създаден метод LoadProducts(), който се изпълнява при стартирането на формата. Той е отговорен за запълването на ListBox-овете s имената и праметрите на продуктите.

Бутоните във формата са „Добави продукт“и „Редактирай продукт“. При натискането на бутоните се изпълняват различни събития. Всеки един бутон започва със проверка с булев метод за валидация деклариран във формата public bool ProductValidation(string productQuantity, string productPrice, string deliveryPrice, string productName):

Декларират се променливи за

* int quantity, която приема текста като число от тип int в productQuantity
* double prPrice, която приема текста като число от тип double в productPrice
* double dlPrice, която приема текста като число от тип double в deliveryPrice
* string prName, която приема текста в productName

Започва се проверка на променливите :

* Ако quantity е по-малко или равно на 0 се показва съобщението "Моля въведете количество по-голямо от нула." и се връща стойност на метода false
* Ако prPrice е по-малко или равно на 0 показва съобщението "Моля въведете цена на продукта по - голяма или равна на нула." и се връща стойност на метода false
* Ако Ако plPrice е по-малко или равно на 0 показва съобщението "Моля въведете доставна цена на продукта по - голяма или равна на нула." и се връща стойност на метода false
* Ако prName е с дължина по-малка или равна на 2 се показва съобщението "Моля въведете валидно име на продукта." и се връща стойност на метода false
* При въведени други грешни данни или не са въведени такива се визуализира съобщението "Моля въведете валидни данни." и се връща стойност на метода false
* При въведени правилно въведени данни и се връща стойност на метода true

За бутона „Добави продукт“ е събитието public void button1\_Click(object sender, EventArgs e), който:

* Този метод проверява стойността на ProductValidation(productQuantity.Text, productPrice.Text, deliveryPrice.Text, productName.Text)
* Декларират се променливите int quantity = int.Parse(productQuantity.Text); double prPrice = double.Parse(productPrice.Text); double dlPrice = double.Parse(deliveryPrice.Text); string prName = productName.Text;
* Проверява се ако ListBox-a с имена продукти съдържа елемент с текст съпадащ с този на prName
* Ако има такъв елемент се показва съобщение за грешка "Този продукт вече е добавен. Ако искате да го редактирате моля натиснете бутона редактирай."
* Ако няма такъв елемент се вика методът public void AddProduct(string name, int quantity, double price, double dlprice) и се създава нов продукт, който се добавя в ListBox-овете.

За бутона „Редактирай продукт“ е събитието public void button2\_Click(object sender, EventArgs e), който:

* Този метод проверява стойността на ProductValidation(productQuantity.Text, productPrice.Text, deliveryPrice.Text, productName.Text)
* Декларират се променливите int quantity = int.Parse(productQuantity.Text); double prPrice = double.Parse(productPrice.Text); double dlPrice = double.Parse(deliveryPrice.Text); string prName = productName.Text;
* Проверява се ако ListBox-a с имена продукти съдържа елемент с текст съпадащ с този на prName
* Ако има такъв елемент се показва съобщение за грешка "Продуктът, който се опитвате да редактирате, не съществува."
* Ако няма такъв елемент се вика методът public void EditProduct(string name, int quantity, double price, double dlprice) и се променя продукта със съответнте му въведени параметри и се актуализират ListBox-овете.

## 4. Допълнителни функционалности

При стартирането на формата се изпълнява методите от контролера public void AddDeliveryBasedOnProductQuantity() и public void RemoveWaitingDeliveries().

RemoveWaitingDeliveries() прави проверка за всички доставки , които не са били одобрени и ги премахва от базата.

AddDeliveryBasedOnProductQuantity() взима първия доставчик със свободни дни, които съдържат и днешния ден. Проверяват се всички продукти в базата и ако някой има количество по-малко или равно на 5. За всеки продукт, които има количество по-малко или равно на 5 се създава една доставка със всички тези продукти.