

МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА

**ПРОФЕСИОНАЛНА ГИМНАЗИЯ “ГЕН. ВЛАДИМИР ЗАИМОВ” гр. СОПОТ**

4330 гр. Сопот, ул. ”Иван Вазов” №1, тел./факс: /03134/ 83-31, 83-32, e-mail: [pgzaimov@yahoo.com](mailto:pgzaimov@yahoo.com)

**ПРОЕКТ**

**Заглавие на проекта:**

**Хранителен магазин GastroSoft**

**Задача № 1**

*Ученик: Тихомир Георгиев Петков*

***Професия:*** *„Системен програмист“*

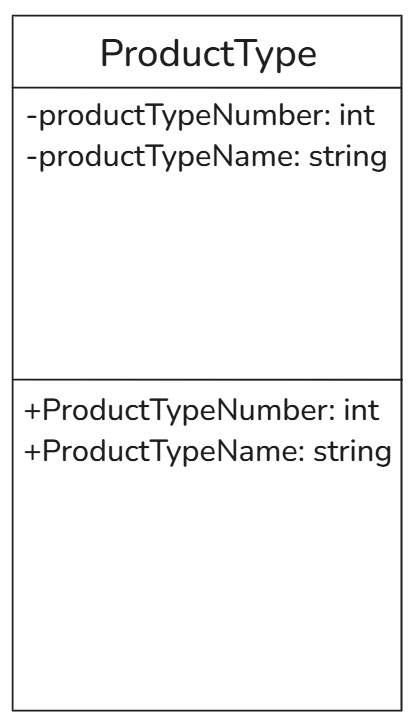
***Специалност:*** *„Системно програмиране“*

Сопот, 2025 г.

**ТОЧКА 1.**

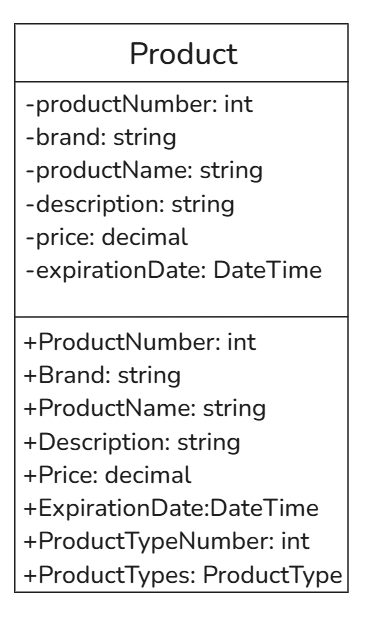
Базата данни се състои от 4 таблици – Products – за продукти, - ProductTypes – за типове продукти, Users – за потребители и Sells – за продажби/покупки.

Спазен е основният принцип на oбектно-ориентираното програмиране - инкапсулация.



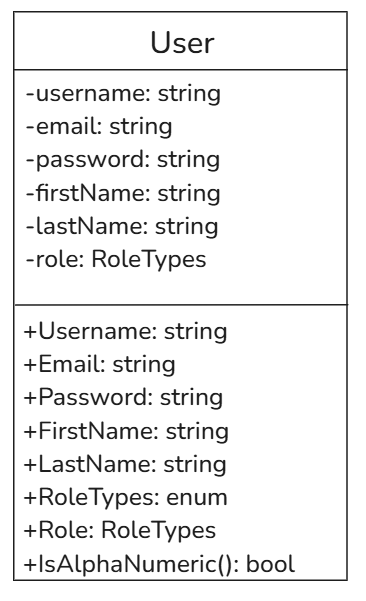
**Фиг. 1 UML диаграма на класа ProductType.**

Класът се състои от полета и свойства за съхранение на данни – productTypeNumber играе роля на първичен ключ в релацията, ProductTypeName – име на типа продукт.

****

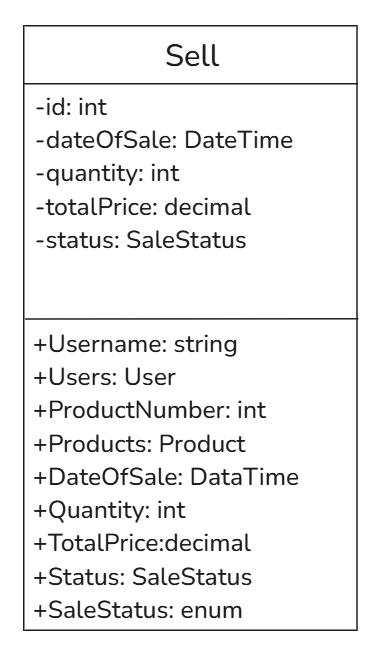
**Фиг. 2 UML диаграма на класа Product.**

Класът се състои от полета и свойства за съхранение на данни – ProductNumber играе роля на първичен ключ в релацията, Brand – марка на дадения продукт, Description – кратко описание за продукта, Price – цена на продукта, ExpirationDate – за срок на годност, ProductTypeNumber – номер на типа продукт – играе роля на външен ключ.



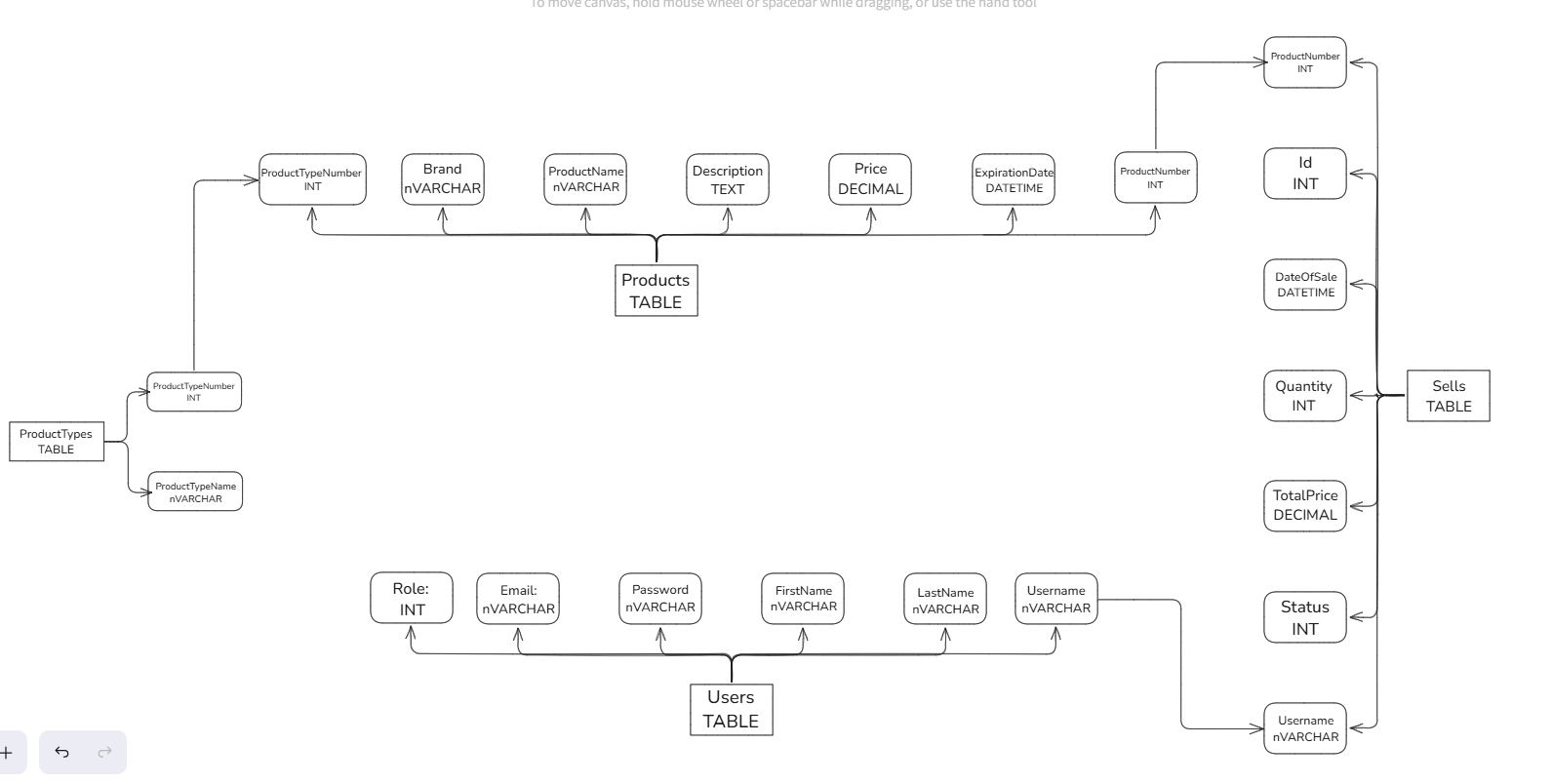
**Фиг. 3 UML диаграма на класа User.**

Класът се състои от полета и свойства за съхранение на данни – Username – потребителско име - играе роля на първичен ключ в релацията, FirstName – първо име на потребителя, LastName – фамилия на потребителя, Email – имейл на потребителя, Password – парола на потребителя, Role – роля на потребителя от тип RoleTypes, RoleTypes – enum тип и помощен булев метод IsAlphaNumeric, който е използван за валидация на част от свойствата.



**Фиг. 4 UML диаграма на класа Sell.**

Класът се състои от свойства за съхранение на данни – Id – играе роля на първичен ключ, Username - потребителско име – играе роля на външен ключ, Users от типа на класа User и Products от типа Product , за да може Entity Framework да направи връзките между ключовите колони.

****

**Фиг. 5 ER диаграма на БД**

ER диаграмата описва връзките както следва:

Първичен ключ от ProductType отива към таблицата Products с ключ ProductTypeNumber.

Първичен ключ от Users отива към Sells с ключ Username.

Първичен ключ oт Products отива към Sells с ключ ProductNumber.

Връзки: два броя 1xM и една МxN.

Следният код описва контекста. Класът ProjectDBContext наследява DbContext, за да може Entity Framework да има контекст за таблиците и техните връзки. Презаписваме конструктора с параметър с истинското име на базата данни – в случая FoodShop. DbSet-oвете от типа на класовете (базите) описват кои са таблиците. Нямат нужда от капсулация, тъй като имаме капсулация в модела.

**public class ProjectDBContext:DbContext**

**{**

**public ProjectDBContext() :base("FoodShop")**

**{**

**}**

**public DbSet<Product> Products**

**{**

**get;**

**set;**

**}**

**public DbSet<ProductType> ProductTypes**

**{**

**get;**

**set;**

**}**

**public DbSet<User> Users**

**{**

**get;**

**set;**

**}**

**public DbSet<Sell> Sells**

**{**

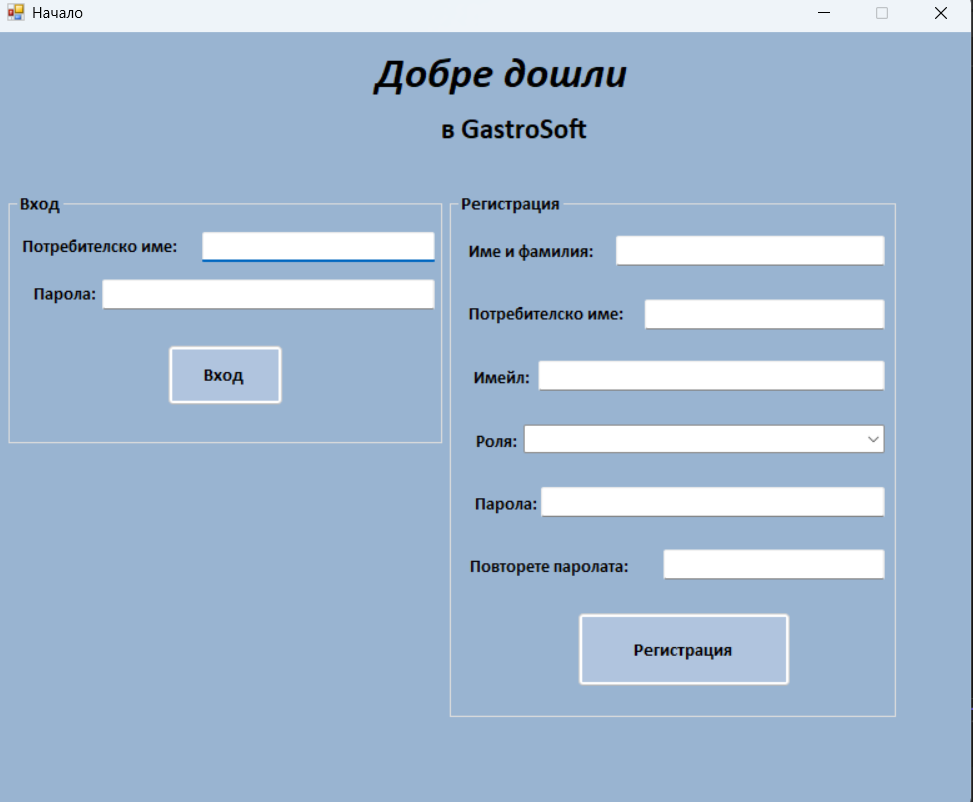
**get;**

**set;**

**}**

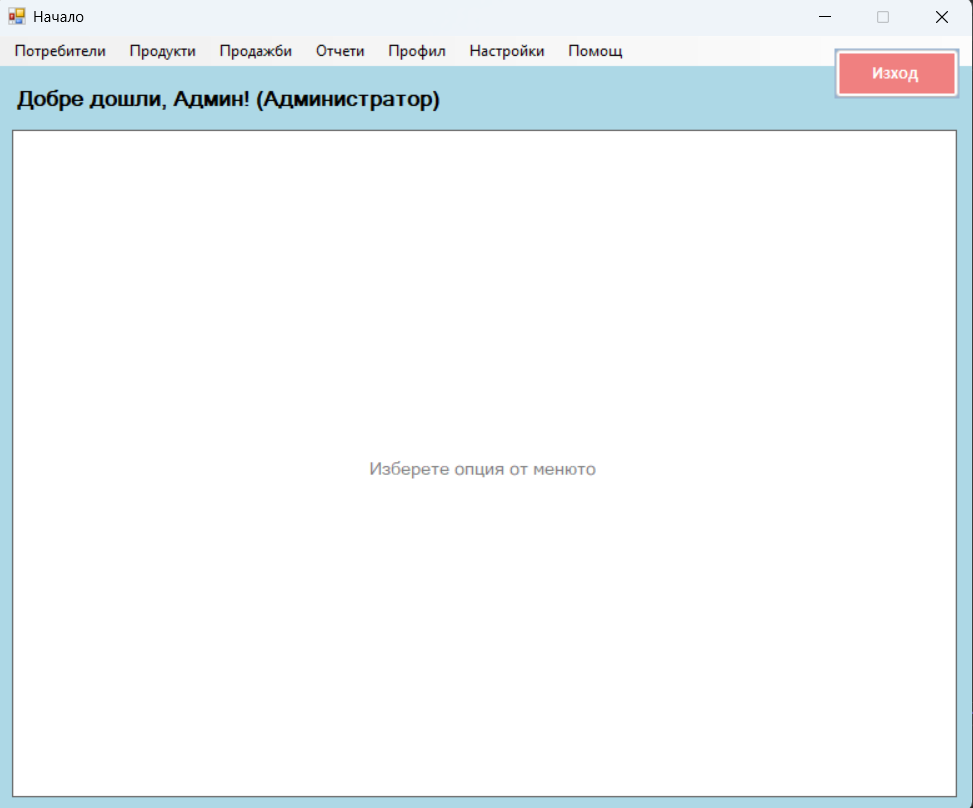
**}**

**ГЛАВА 2.**

****

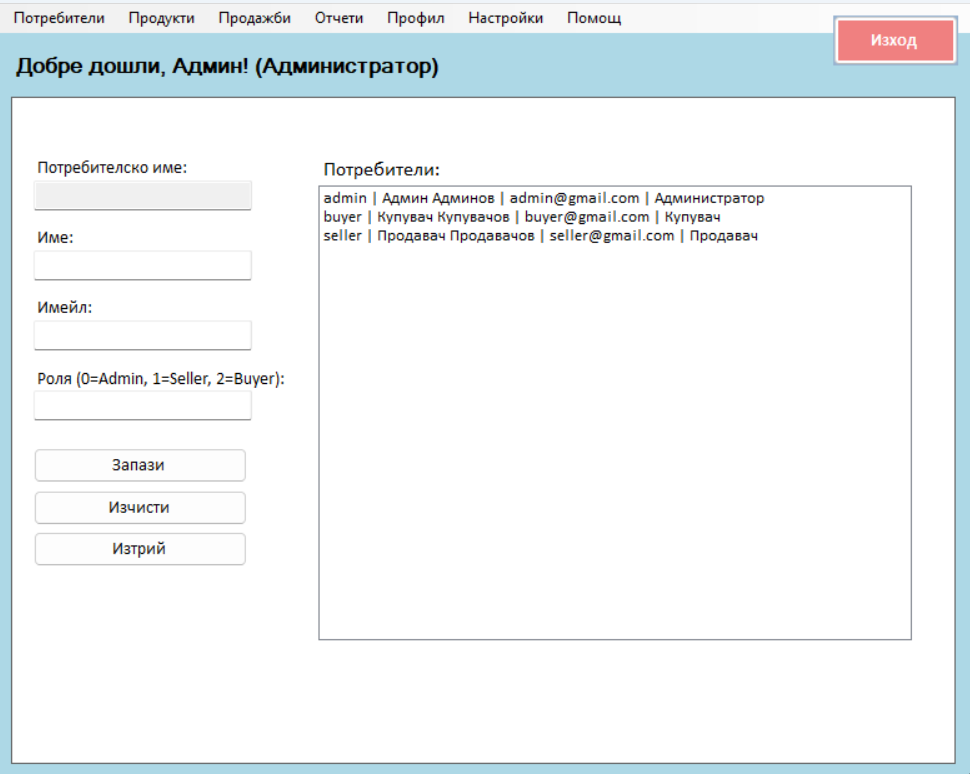
**Фиг. 6 index.cs**

Началната форма за вход и регистрация на GastroSoft.

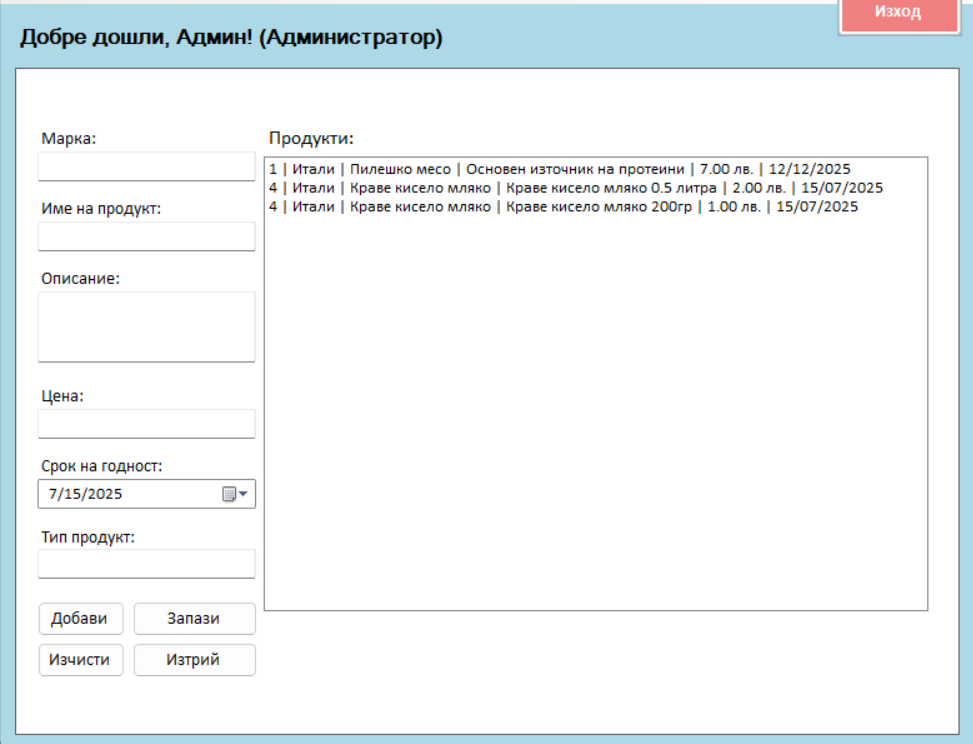


**Фиг. 7 AdminControl.cs**

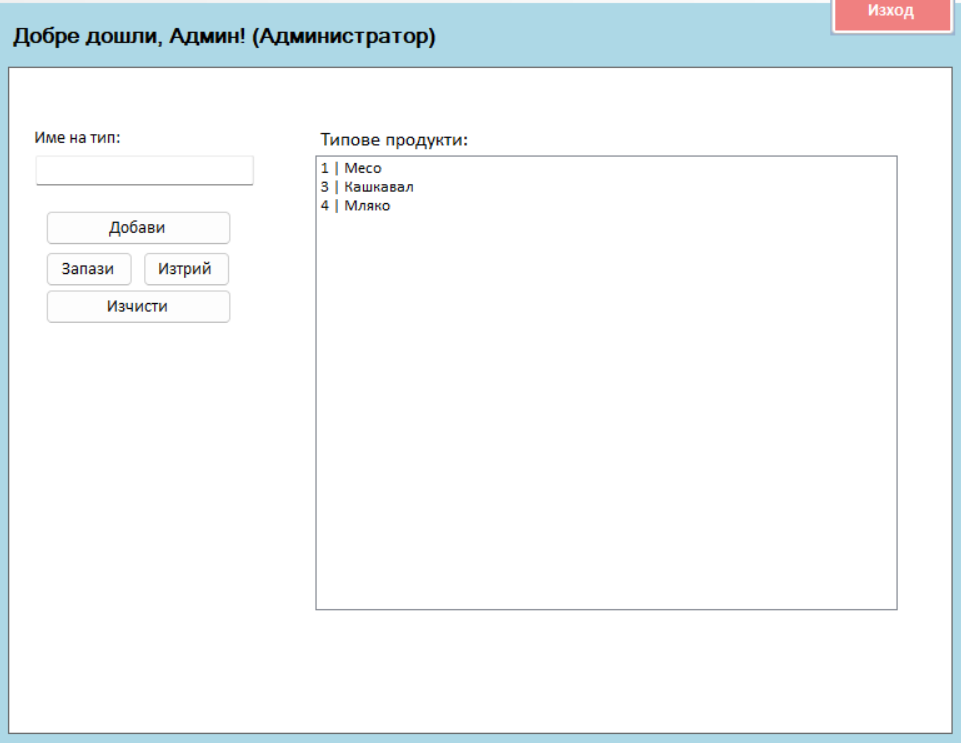
Начален екран на регистриран администратор.



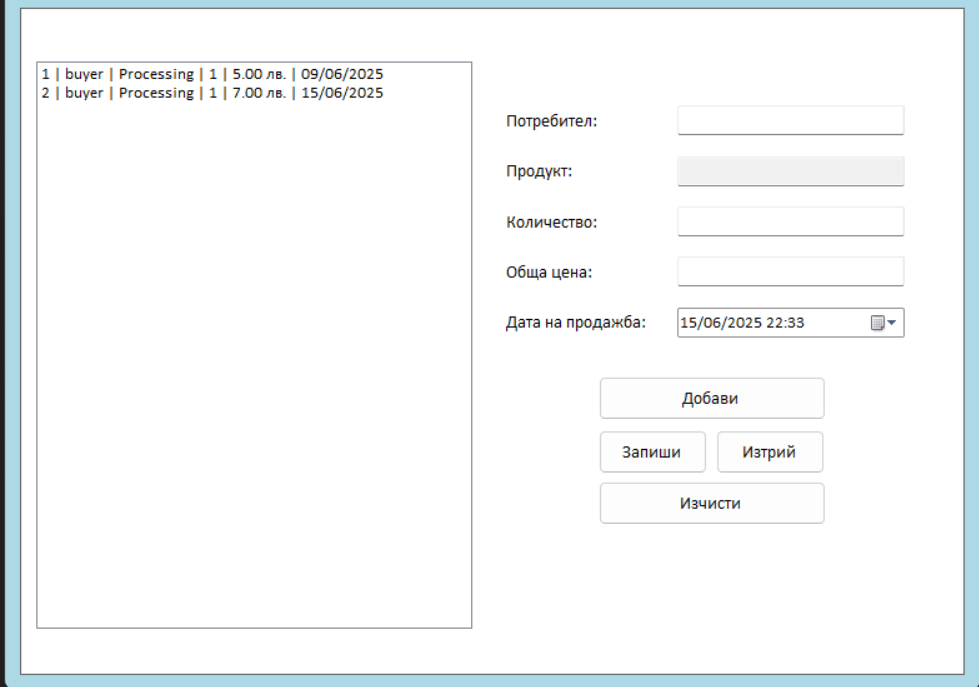
**Фиг. 8 Управление на потребители**

****

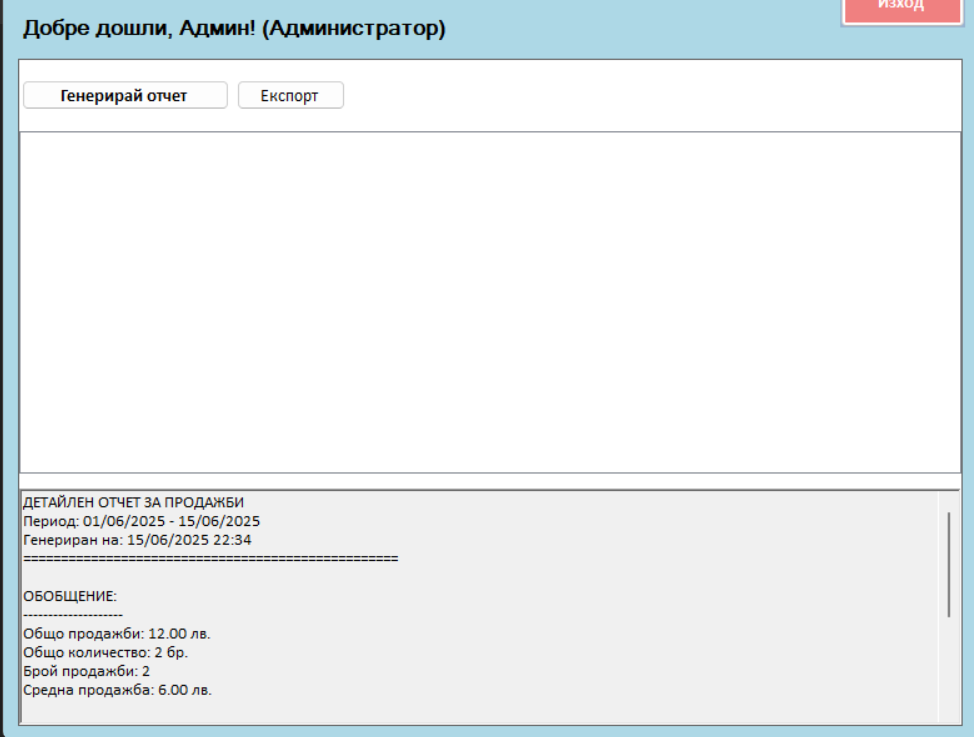
**Фиг. 9 Управление на продукти**

****

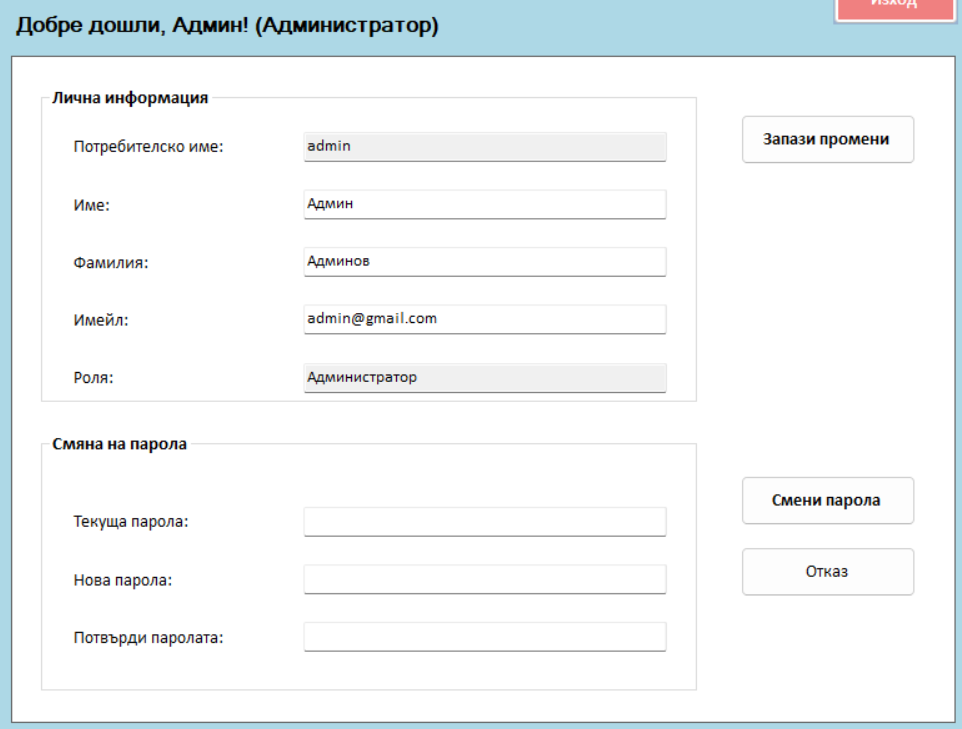
**Фиг. 10 Управление на тип продукти**

****

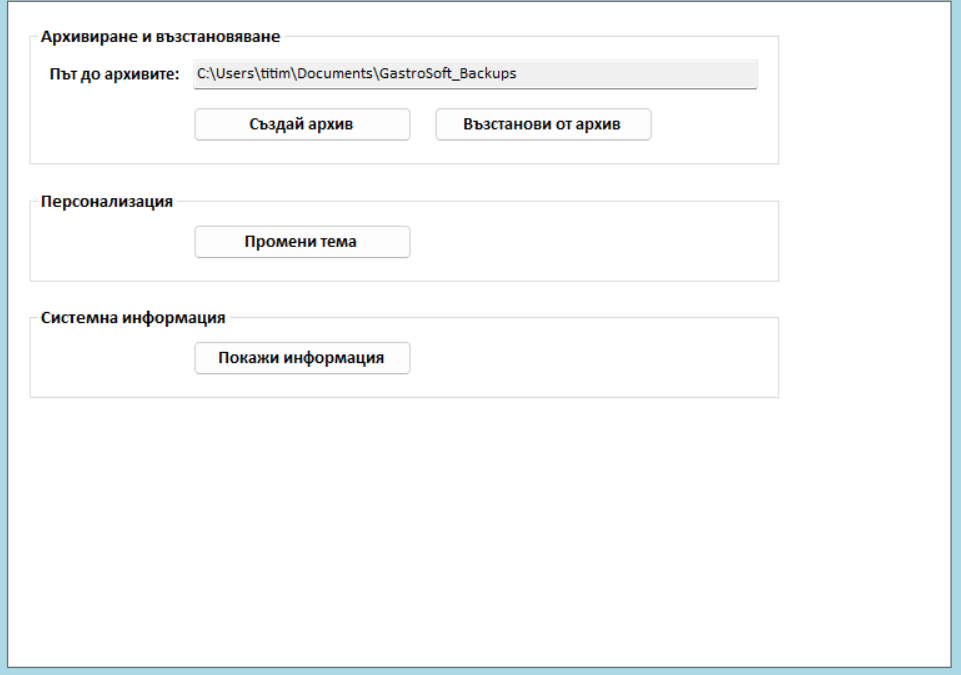
**Фиг. 11 Управление на покупки/продажби**

****

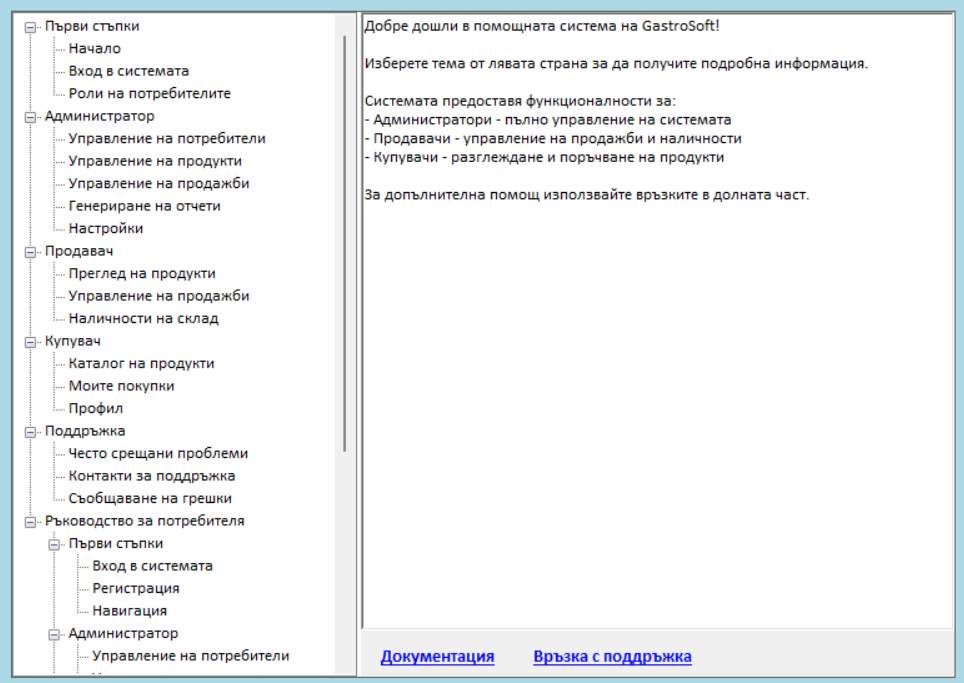
**Фиг. 12 Генериране на отчет спрямо продажбите.**

****

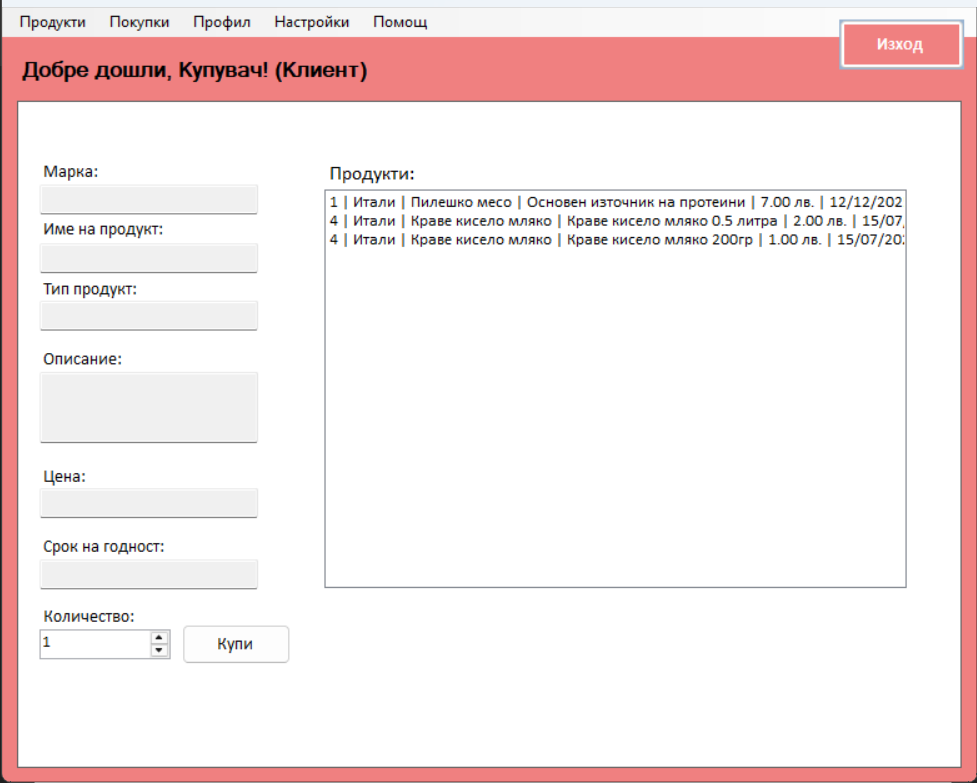
**Фиг. 13 Управление на профил.**

****

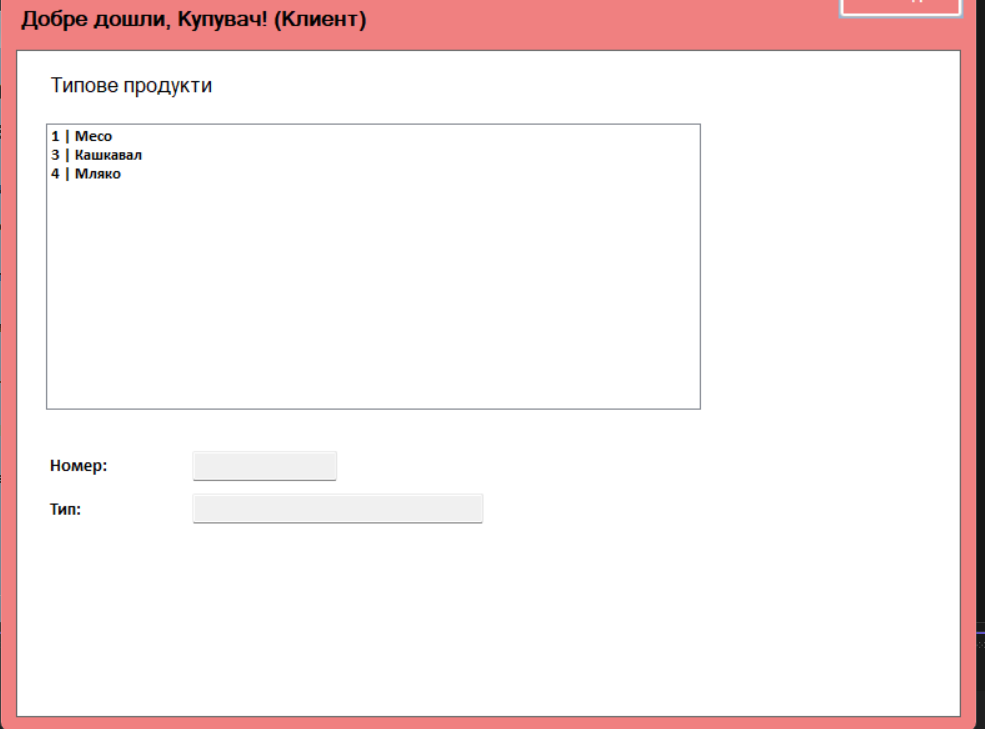
**Фиг. 14 Настройки**

****

**Фиг. 15 Помощ**

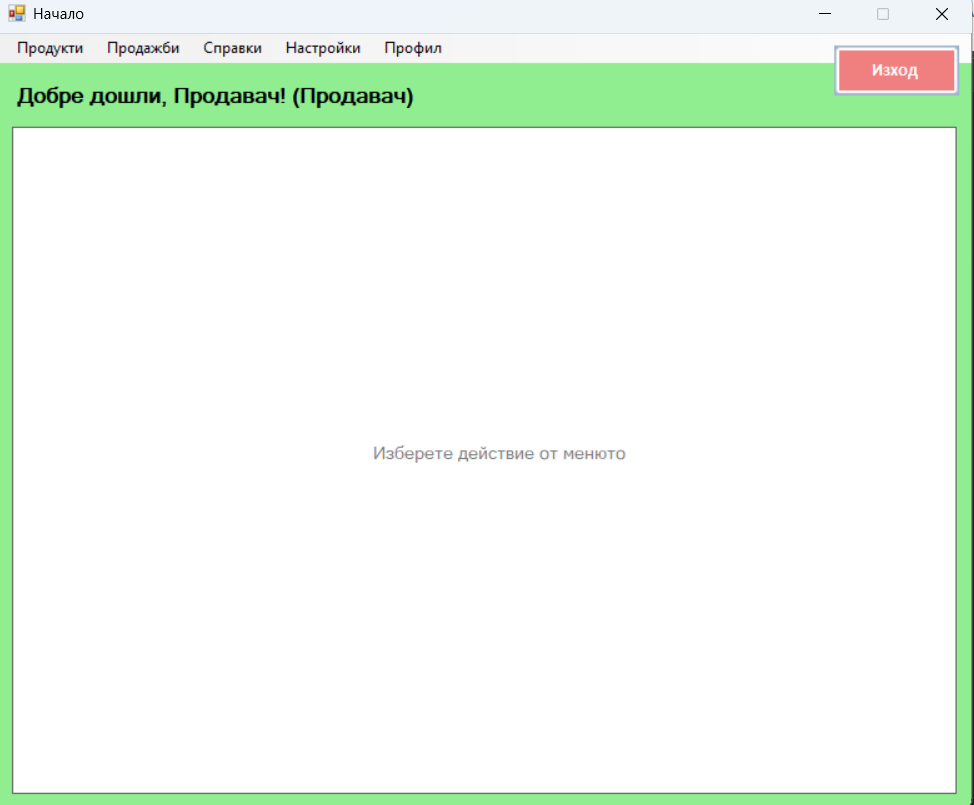
****

**Фиг. 16 Начален екран на купувач и каталог с продукти.**

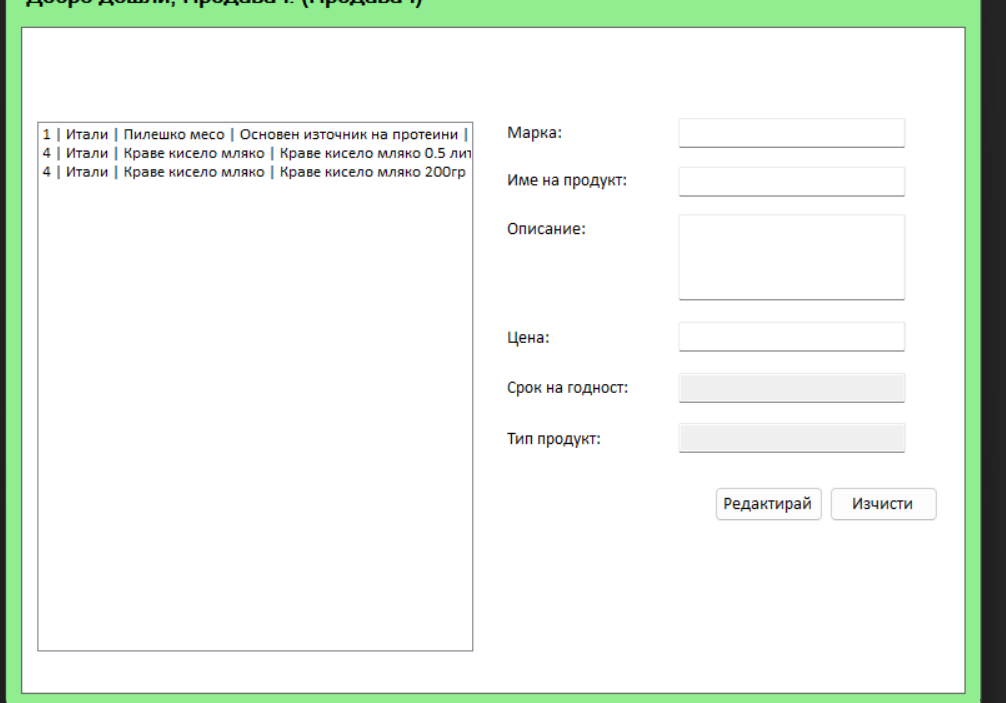
****

**Фиг. 17 Каталог с тип продукти за купувач**

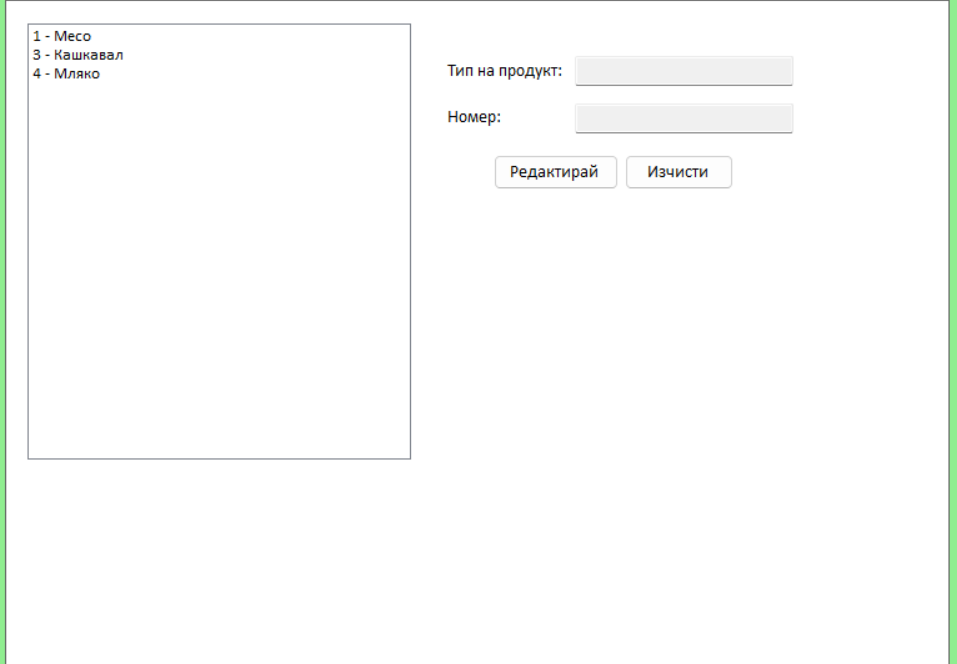
Разликата е, че купувачът само вижда и няма правомощия да променя. Профилът, настройките и помощ са аналогични.

****

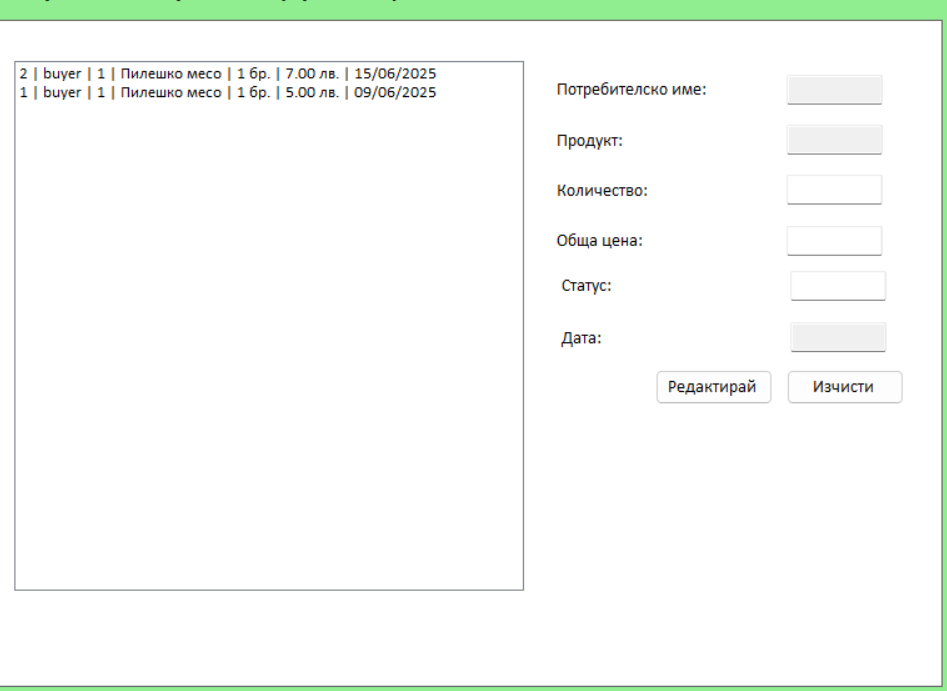
**Фиг. 18 Начален екран на продавач.**

****

**Фиг. 19 Екран за управление на продукти.**

****

**Фиг. 20 Екран каталог за типове продукти.**

****

**Фиг. 21 Екран за контрол над продажбите.**

Справките (отчетите), настройките и помощ са аналогични.

**ГЛАВА 3.**

За технология с БД е използван Entity Framework, който прави базата данни изцяло от C# код. Entity Framework сам генерира MSSQL код, който създава базата данни.

Описание на CRUD операциите- особености на отделните методи

Методът Get() намира обекта (записа) с id в параметър само, ако не е null.

public Product Get(int productId)

{

Product findedProduct = dbContext.Products.Find(productId);

if (findedProduct != null)

dbContext.Entry(findedProduct).Reference(x => x.ProductTypes).Load();

return findedProduct;

}

Методът GetAll() връща списък с всички обекти (записи) от БД.

public List<Product> GetAll()

{

return dbContext.Products.Include("ProductTypes").ToList();

}

Методът Create() създава запис. Използва параметър обект, който става запис в БД.

public void Create(Product product) {

dbContext.Products.Add(product);

dbContext.SaveChanges();

}

Методът Update() променя вече съществуващ запис. Това става като за параметри имаме id и обект, който ще замести записа. Първо намираме записа, а после – изреждаме всички свойства (колони), които ще замести.

public void Update(int productId, Product product)

{

Product findedProduct = dbContext.Products.Find(productId);

if (findedProduct != null)

dbContext.Entry(findedProduct).Reference(x=> x.ProductTypes).Load();

findedProduct.ProductTypeNumber = product.ProductTypeNumber;

findedProduct.ExpirationDate=product.ExpirationDate;

findedProduct.Price = product.Price;

findedProduct.Brand = product.Brand;

findedProduct.ProductName = product.ProductName;

findedProduct.Description = product.Description;

dbContext.SaveChanges();

}

Методът Delete() трие запис, по посочен id в параметър.

public void Delete(int productId)

{

Product findedProduct = dbContext.Products.Find(productId);

if (findedProduct != null)

dbContext.Entry(findedProduct).Reference(x => x.ProductTypes).Load();

dbContext.Products.Remove(findedProduct);

dbContext.SaveChanges();

}