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

ПРОЕКТ

Заглавие на проекта: Куриерска фирма
Задача № 7

Ученик: Мартен Добройков

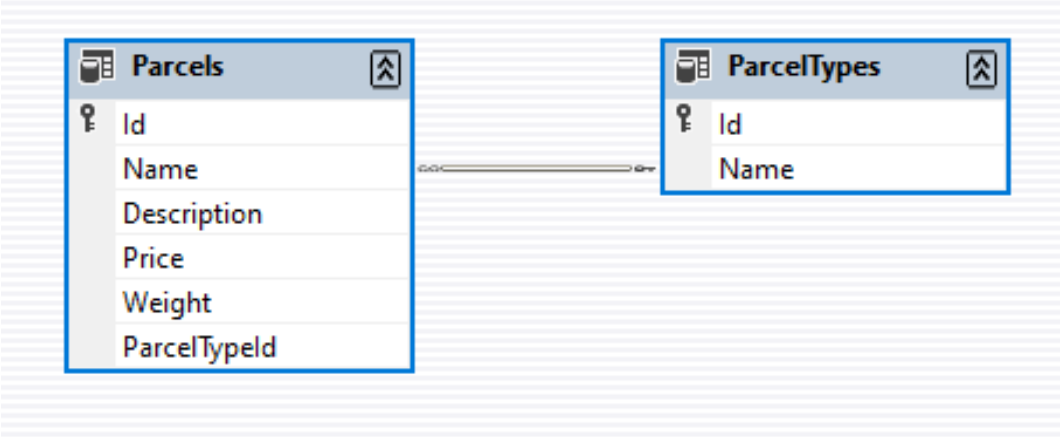
Професия: *„Системен програмист“*
Специалност: *„Системно програмиране“*

Сопот, 2025 г.

ТОЧКА 1.

Описание на БД

Диаграма на БД:



Използвана е технологията First-code за създаване на база данни.

1. Файл ParcelType.cs

```
namespace ProjectMarten.Model
{
    public class ParcelType
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public ICollection<Parcel> Parcels { get; set; }
    }
}
```

Този клас представлява таблица ***"ParcelTypes"*** и описва различните типове пратки, които системата поддържа. Колекцията ***Parcels*** съдържа всички пратки, които са от този конкретен тип, което улеснява извличането на свързани данни.

2. Файл Parcel.cs

```
namespace ProjectMarten.Model
{
    public class Parcel
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public decimal Price { get; set; }
        public decimal Weight { get; set; }

        public int ParcelTypeId { get; set; }
        public ParcelType ParcelTypes { get; set; }

    }
}
```

Този клас представлява таблица ***"Parcels"*** в базата данни. Всеки обект от този клас съответства на една пратка в куриерската система. Свойството ***ParcelTypes*** позволява лесен достъп до свързания тип пратка без необходимост от допълнителни заявки.

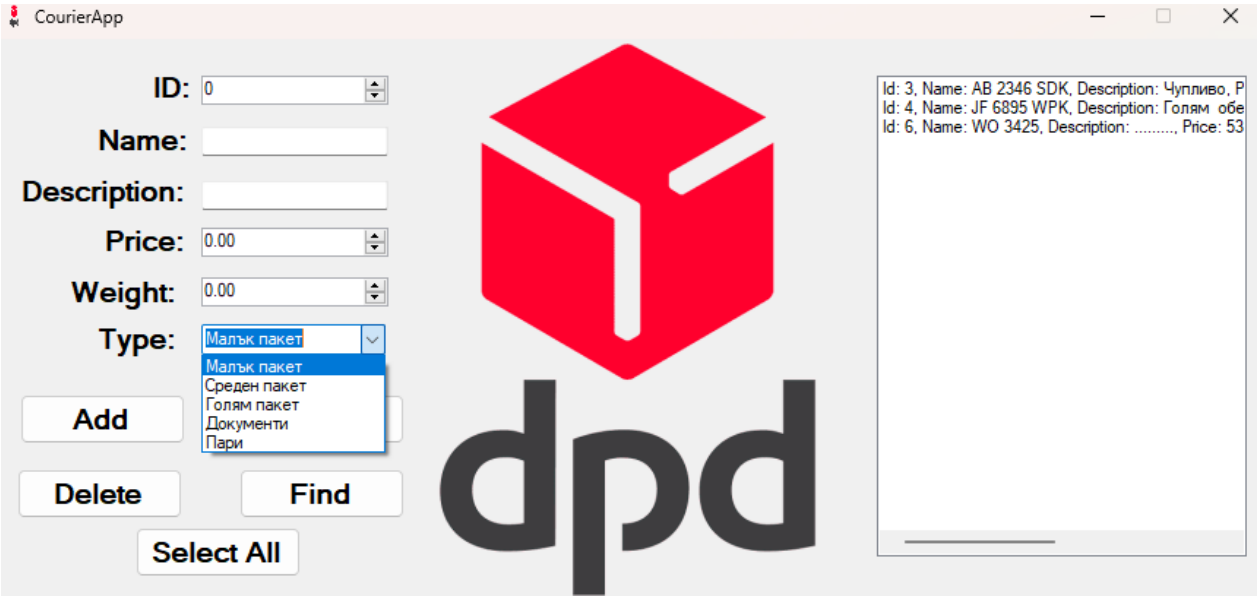
3. Файл CourierContext.cs

```
namespace ProjectMarten.Model
{
    public class CourierContext : DbContext
    {
        public CourierContext() : base("CourierContext")
        {
        }
        public DbSet<Parcel> Parcels { get; set; }
        public DbSet<ParcelType> ParcelTypes { get; set; }
    }
}
```

Този клас наследява ***DbContext*** на ***Entity Framework*** и служи като основна точка за взаимодействие с базата данни. Дефинира кои класове трябва да бъдат мапнати към таблици в базата. Конструкторът указва името на connection string-а от конфигурационния файл.

ГЛАВА 2.

Функционално описание: Отделните екрани с кратко описание за работа с приложението



При стартиране на приложението се отваря форма с два основни раздела: лявата страна съдържа полета за въвеждане на данни за пратки (ID, име, описание, цена, тегло и тип), а дясната показва списък с всички съществуващи пратки. Над полетата има пет бутона:

- ***Добави*** - създава нова пратка
- ***Редактирай*** - променя съществуваща пратка
- ***Изтрий*** - премахва избрана пратка
- ***Търси*** - намира пратка по ID
- ***Покажи всички*** - обновява списъка с пратки

Бутоните позволяват пълно управление на пратките в системата - създаване, преглед, редактиране и изтриване на записи.

ГЛАВА 3.

Анализ на задачата

1. Файл Form1.cs

1.1 Полета

```
ParcelController parcelController = new ParcelController();
ParcelTypeController parcelTypeController = new ParcelTypeController();
```

Тези две полета създават инстанции на контролерите, които управляват операциите с пратки (***ParcelController***) и типовете пратки ***(ParcelTypeController***) в базата данни.

1.2 Метод Form1_Load

```
private void Form1_Load(object sender, EventArgs e)
{
    List<Parcel> allParcels = parcelController.GetAll();
    List<ParcelType> allParcelTypes =
parcelTypeController.GetParcelTypes();
    cmbType.DataSource = allParcelTypes;
    cmbType.DisplayMember = "Name";
    cmbType.ValueMember = "Id";
    foreach (var item in allParcels)
    {
        listBox1.Items.Add($"Id: {item.Id}, Name:
{item.Name}, Description: {item.Description}," +
            $" Price: {item.Price}, Weight:
{item.Weight}, Type: {item.ParcelTypes.Name}");
    }

}
```

Методът ***Form1_Load*** инициализира формата при зареждане:

- Зарежда всички пратки и типове пратки от базата данни
- Конфигурира ***ComboBox*** за избор на тип пратка:
- Задава източник на данни (***DataSource***)
- Поле за показване (***DisplayMember***)
- Стойност (***ValueMember***)
- Попълва ***ListBox*** с информация за всички пратки

1.3 Метод LoadRecord

```
private void LoadRecord(Parcel parcel)
{
    numId.Text = parcel.Id.ToString();
    txtName.Text = parcel.Name;
    txtDescription.Text = parcel.Description.ToString();
    numPrice.Text = parcel.Price.ToString();
    numWeight.Text = parcel.Weight.ToString();
    cmbType.Text = parcel.ParcelTypes.Name;
}
```

Методът LoadRecord зарежда данните от подаден обект Parcel в съответните контроли на формата:

- *ID* в NumericUpDown полето numId
- *Име* в TextBox полето txtName
- *Описание* в TextBox полето txtDescription
- *Цена* в NumericUpDown полето numPrice
- *Тегло* в NumericUpDown полето numWeight
- *Име* на тип пратка в ComboBox контролата cmbType

1.4 Метод ClearScreen

```
private void ClearScreen()
{
    txtDescription.Clear();
    txtName.Clear();
    numId.Value = 0;
    numPrice.Value = 0;
```

```
    numWeight.Value = 0;
    cmbType.Text = "";
}
```

Методът *ClearScreen* изчиства всички входни полета на формата, връщайки ги в първоначално състояние. Нулира *NumericUpDown* контролите и изчиства TextBox полетата.

1.5 Метод CheckIfNull

```
private void CheckIfNull()
{
    if (numId.Value == 0)
    {
        MessageBox.Show("Не сте въвели Id", "ГРЕШКА!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        numId.Focus();
        return;
    }
    if (txtName.Text == "")
    {
        MessageBox.Show("Не сте въвели име", "ГРЕШКА!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        txtName.Focus();
        return;
    }

    if (txtDescription.Text == "")
    {
        MessageBox.Show("Не сте въвели описание", "ГРЕШКА!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        txtDescription.Focus();
        return;
    }
    if (numPrice.Value == 0)
    {
        MessageBox.Show("Не сте въвели цена", "ГРЕШКА!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        numPrice.Focus();
        return;
    }
    if (numWeight.Value == 0)
    {
```

```
        MessageBox.Show("Не сте въвели тегло", "ГРЕШКА!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        numWeight.Focus();
        return;
    }
    if (cmbType.SelectedIndex == -1)
    {
        MessageBox.Show("Не сте избрали тип", "ГРЕШКА!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        cmbType.Focus();
        return;
    }
}
```

Методът *CheckIfNull* извършва валидация на входните данни:

- Проверява дали всички *задължителни полета* са попълнени
- При *липса* на данни показва съобщение за грешка
- *Фокусира* съответното поле за корекция

1.6 Бутон Add

```
private void btnAdd_Click(object sender, EventArgs e)
{
    CheckIfNull();
    Parcel newParcel = new Parcel
    {
        Id = (int)numId.Value,
        Name = txtName.Text,
        Description = txtDescription.Text,
        Price = numPrice.Value,
        Weight = numWeight.Value,
        ParcelTypeId = (int)cmbType.SelectedValue
    };
    parcelController.Create(newParcel);
    MessageBox.Show("Успешно добавихте пратка", "ГОТОВО!",
MessageBoxButtons.OK, MessageBoxIcon.Information);
    ClearScreen();

}
```

Бутонът Add обработва събитието за добавяне на нова пратка:

- Проверява за попълнени всички задължителни полета чрез *CheckIfNull()*
- Създава нов обект *Parcel* с данни от формата

- Извиква *Create* метода на *parcelController* за запис в базата данни
- Показва съобщение за *успех* и изчиства формата

1.7 Бутон Update

```
private void btnUpdate_Click(object sender, EventArgs e)
{
    int findId = 0;
    if (numId.Value == 0)
    {
        MessageBox.Show("Не сте въвели Id", "ГРЕШКА!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        numId.Focus();
        return;
    }
    else
    {
        findId = (int)numId.Value;
    }
    Parcel findedParcel = parcelController.Get(findId);
    if (findedParcel == null)
    {
        MessageBox.Show("Не е намерена пратка с такова Id",
"ГРЕШКА!", MessageBoxButtons.OK, MessageBoxIcon.Error);
        numId.Focus();
        return;
    }
    else
    {
        LoadRecord(findedParcel);
        Parcel updatedParcel = new Parcel
        {
            Id = (int)numId.Value,
            Name = txtName.Text,
            Description = txtDescription.Text,
            Price = numPrice.Value,
            Weight = numWeight.Value,
            ParcelTypeId = (int)cmbType.SelectedValue
```

```
        };
        parcelController.Update(findId, updatedParcel);
        MessageBox.Show($"Успешно обновихте пратка номер
{findId}", "ГОТОВО!", MessageBoxButtons.OK,
MessageBoxIcon.Information);
    }
    ClearScreen();
    btnSelectAll_Click(sender, e);
}
```

Бутонът *Update* обработва актуализацията на съществуваща пратка:

- Намира пратка по *ID*
- Създава нов обект с актуализираните данни
- Извиква *Update* метода на контролера
- Показва съобщение за успех и обновява интерфейса

1.8 Бутон Delete

```
private void btnDelete_Click(object sender, EventArgs e)
{
    int findId = 0;
    if (numId.Value == 0)
    {
        MessageBox.Show("Не сте въвели Id", "ГРЕШКА!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        numId.Focus();
        return;
    }
    else
    {
        findId = (int)numId.Value;
    }
    Parcel findedParcel = parcelController.Get(findId);
    if (findedParcel == null)
    {
        MessageBox.Show("Не е намерена пратка с такова Id",
"ГРЕШКА!", MessageBoxButtons.OK, MessageBoxIcon.Error);
        numId.Focus();
        return;
    }
    else
    {
```

```
        DialogResult dialogResult = MessageBox.Show($"Сигурни ли сте,
че искате да изтриете пратка номер {findId}?", "Потвърждение",
MessageBoxButtons.YesNo, MessageBoxIcon.Question);
        if (dialogResult == DialogResult.Yes)
        {
            parcelController.Delete(findId);
            MessageBox.Show($"Успешно изтрихте пратка номер
{findId}", "ГОТОВО!", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            ClearScreen();
        }
    }
    btnSelectAll_Click(sender, e);
}
```

Бутонът *Delete* обработва изтриването на пратка:

- Намира пратка по *ID*
- Изисква потвърждение от потребителя
- При потвърждение извиква *Delete* метода на контролера
- Показва съобщение за успех и обновява интерфейса

1.9 Бутон Find

```
private void btnFind_Click(object sender, EventArgs e)
{
    int findId = 0;
    if (numId.Value == 0)
    {
        MessageBox.Show("Не сте въвели Id", "ГРЕШКА!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        numId.Focus();
        return;
    }
    else
    {
        findId = (int)numId.Value;
    }
    Parcel findedParcel = parcelController.Get(findId);
    if (findedParcel == null)
    {
        MessageBox.Show("Не е намерена пратка с такова Id",
"ГРЕШКА!", MessageBoxButtons.OK, MessageBoxIcon.Error);
        numId.Focus();
        return;
    }
    else
    {
```

```
        LoadRecord(findedParcel);
        MessageBox.Show($"Успешно намерихте пратка номер
{findId}", "ГОТОВО!", MessageBoxButtons.OK,
MessageBoxIcon.Information);
    }
}
```

Бутонът *Find* обработва търсенето на пратка по *ID*:

- Намира *пратка* в базата данни
- Зарежда данните във формата чрез *LoadRecord*
- Показва съобщение за успешно намиране

1.10 Бутон Select All

```
private void btnSelectAll_Click(object sender, EventArgs e)
{
    List<Parcel> allParcels = parcelController.GetAll();
    listBox1.Items.Clear();
    foreach(var item in allParcels)
    {
        listBox1.Items.Add($"Id: {item.Id}, Name: {item.Name},
Description: {item.Description}," +
            $" Price: {item.Price}, Weight: {item.Weight}, Type:
{item.ParcelTypes.Name}");
    }
}
```

Бутонът *SelectAll* обновява списъка с пратки:

- Извлича *всички пратки* от базата данни
- Изчиства текущото съдържание на *ListBox*
- Попълва списъка с форматирани данни за *всяка пратка*