CRM SİSTEMİ

Customer Relationship Management

Nesne Yönelimli Programlama 2 (NTP-2)

Dönem Projesi

Ahmet Baran Bozkurt
11210000045
Matematik / 4. Sınıf

04.05.2026

Sunum Akışı

1. **Projenin Amacı**
   CRM nedir, neyi çözer
2. **Sistem Gereksinimleri**
   .NET, SQLite, kütüphaneler
3. **Mimari Yapı**
   3-Layer Architecture
4. **OOP İlkelerinin Uygulanması**
   Inheritance, Encapsulation, Generics, Enum
5. **Veri İşleme Yöntemi**
   SQLite + ADO.NET + Repository Pattern
6. **Uygulama Ekranları**
   Dashboard, müşteri ve aktivite modülleri
7. **Sonuç ve Kazanımlar**
   Öğrendiklerim ve gelecek geliştirmeler

Projenin Amacı

Problem

İşletmeler, müşteri bilgilerini ve onlarla yapılan etkileşimleri (aramalar, toplantılar, e-postalar) takip etmekte zorlanıyor.

Veriler farklı sistemlerde dağınık halde duruyor.

Çözüm

Tek bir merkezi sistemde:

- Müşteri ve şirket yönetimi
- Aktivite takibi (Call, Meeting, Email, Note)
- İstatistiksel raporlama
- Dashboard görünümü

OOP Açısından

Real-world entity'ler arasındaki 1-N ilişkileri ve kalıtım, encapsulation, polymorphism gibi OOP ilkelerini doğal olarak sergileyen ideal bir konu.

Sistem Gereksinimleri

Geliştirme Ortamı

- Visual Studio 2022 (Community)
- C# 7.3+
- .NET Framework 4.7.2 / 4.8

Çalışma Platformu

- Windows 10 / 11
- Windows Forms Application
- Min. 4 GB RAM

Veritabanı

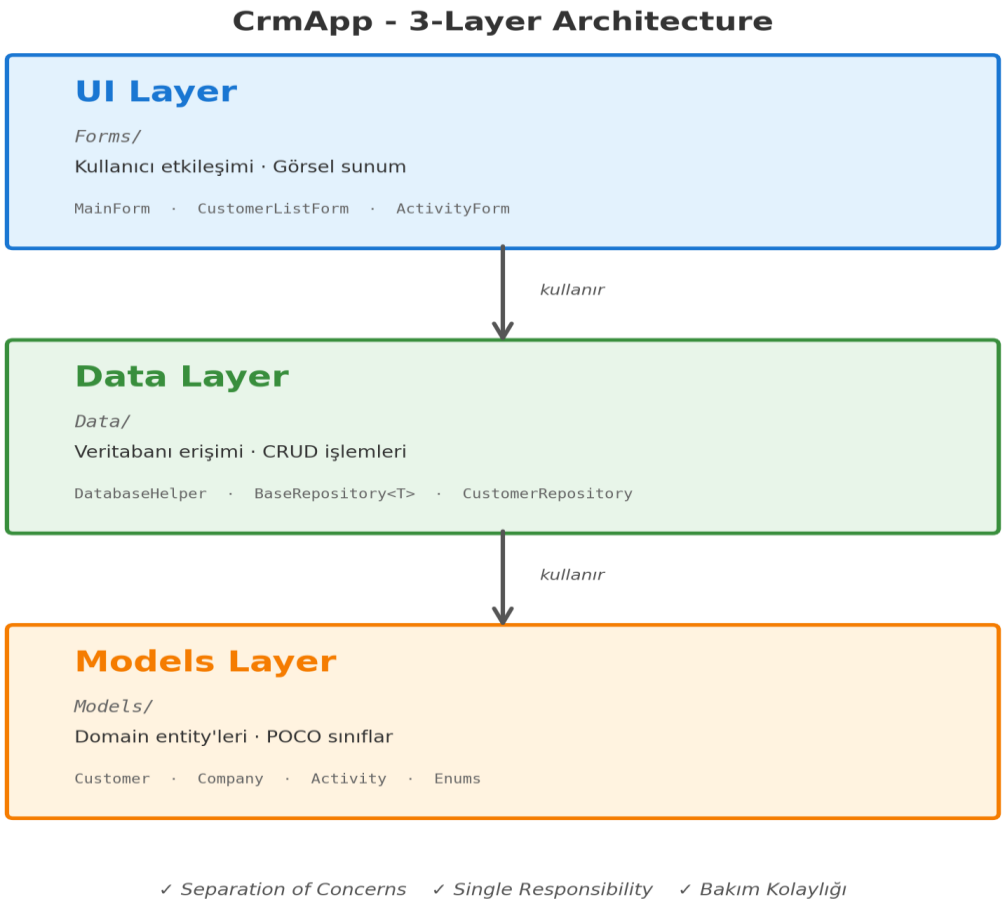
- SQLite (file-based, embedded)
- System.Data.SQLite.Core
- Tek .db dosyası, taşınabilir

Bağımlılıklar

- Sadece 1 NuGet paketi
- Üçüncü parti minimum
- Kurulum gerektirmeyen yapı

Katmanlı Mimari (3-Layer Architecture)

Form'dan veritabanına net bir akış · Sorumluluk ayrımı



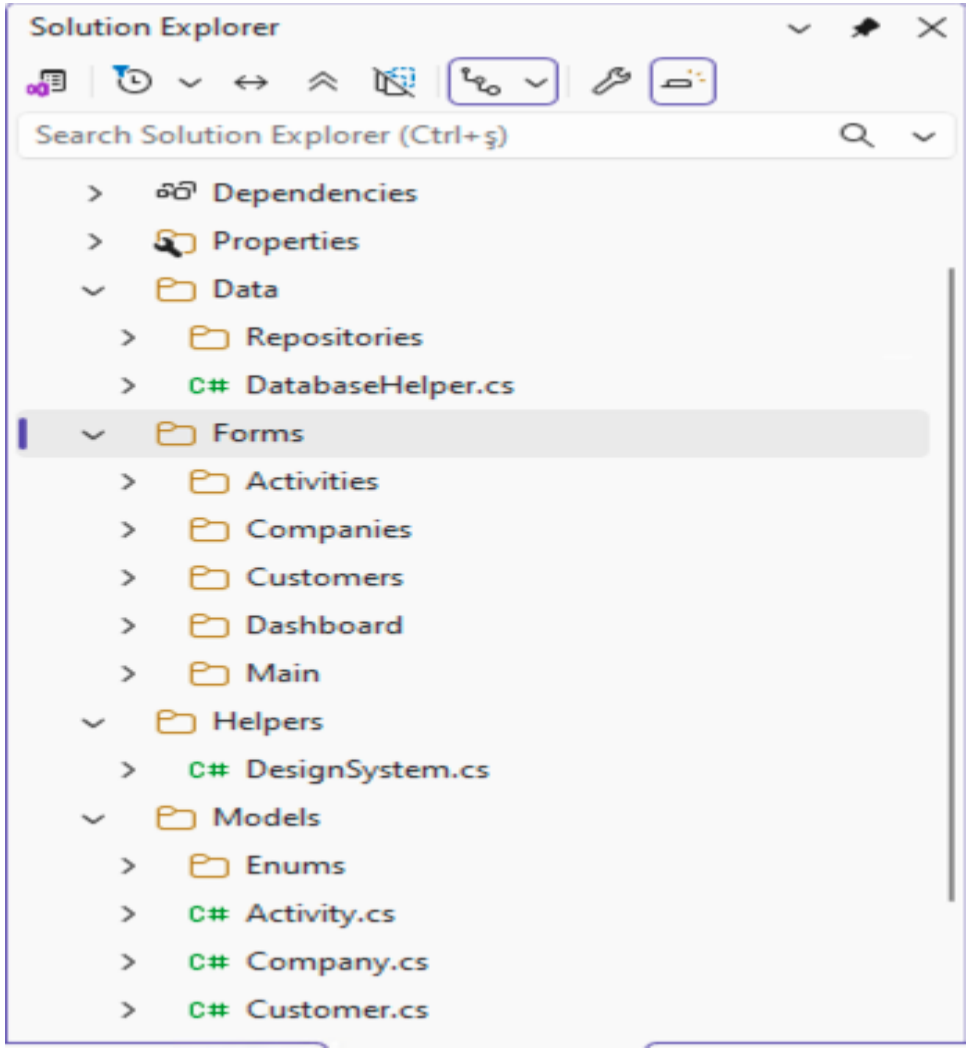
Neden Katmanlı?

- UI Layer sadece kullanıcıyla konuşur
- Data Layer SQL'i tek yerde toplar
- Models Layer veriyi temsil eder
- Bir katmandaki değişiklik diğerini etkilemez
- DRY (Don't Repeat Yourself) prensibi
- SRP (Single Responsibility Principle)

Bu yaklaşım profesyonel yazılım dünyasında "Clean Architecture"ın temel taşıdır.

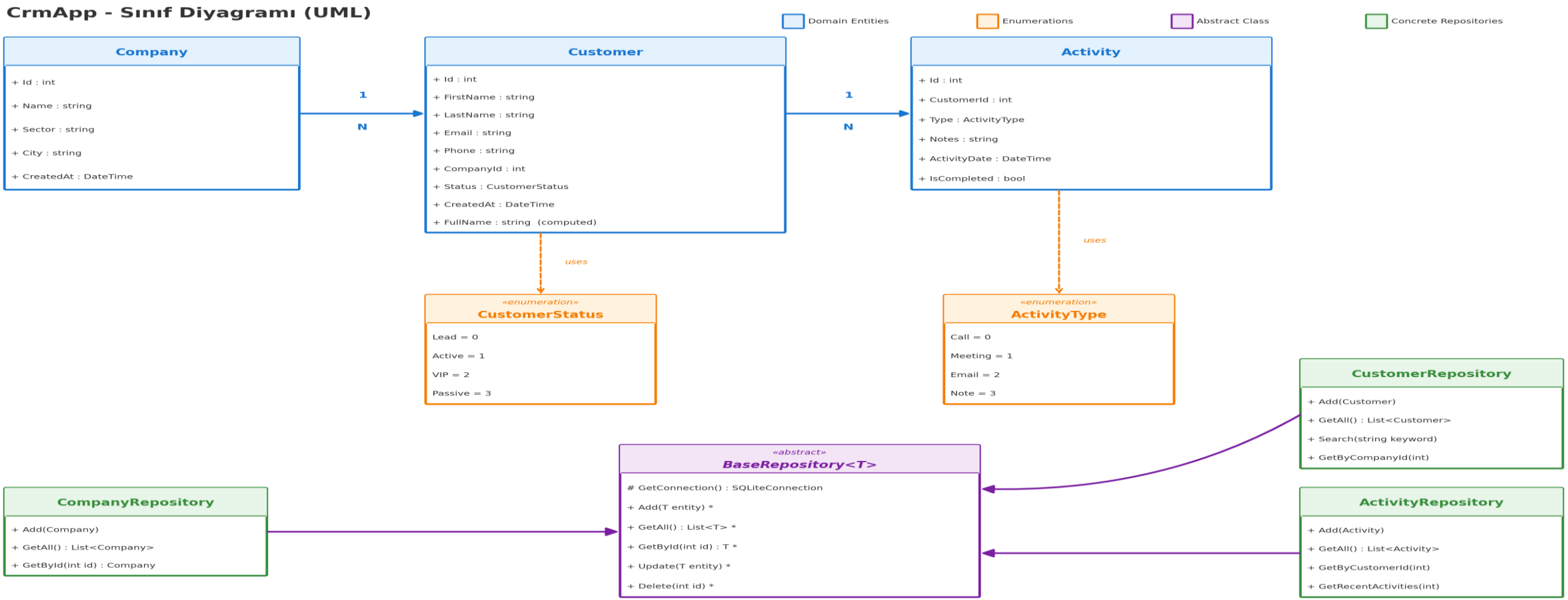
Proje Klasör Yapısı

```
CrmApp/
├── Models/              → Domain entity'leri
│   ├── Customer.cs
│   ├── Company.cs
│   ├── Activity.cs
│   └── Enums/
│       ├── CustomerStatus.cs
│       └── ActivityType.cs
├── Data/                → Veri katmanı
│   ├── DatabaseHelper.cs
│   └── Repositories/
│       ├── BaseRepository.cs
│       ├── CustomerRepository.cs
│       ├── CompanyRepository.cs
│       └── ActivityRepository.cs
├── Forms/               → UI katmanı
│   ├── MainForm.cs
│   ├── CustomerListForm.cs
│   ├── CustomerEditForm.cs
│   └── ActivityForm.cs
├── Helpers/             → Yardımcı sınıflar
│   └── DesignSystem.cs
├── crm.db               → SQLite DB
├── seed.sql             → İlk kurulum verisi
└── Program.cs
```



Sınıf Diyagramı (UML)

Domain entity'leri · Repository hiyerarşisi · İlişkiler



1-N İlişkiler: Company → Customer · Customer → Activity **Inheritance:** 3 Repository → BaseRepository<T>

OOP: Inheritance + Generics

BaseRepository<T> — Üç kavramın bir arada kullanımı

```
public abstract class BaseRepository<T> where T : class
{
    protected SQLiteConnection GetConnection()
        => DatabaseHelper.GetConnection();

    public abstract void Add(T entity);
    public abstract List<T> GetAll();
    public abstract T GetById(int id);
    public abstract void Update(T entity);
    public abstract void Delete(int id);
}
```

```
public class CustomerRepository
    : BaseRepository<Customer>
{
    public override void Add(Customer customer)
    {
        // INSERT INTO Customers ...
    }

    public override List<Customer> GetAll()
    {
        // SELECT * FROM Customers ...
    }
}
```

🎯 abstract class

Doğrudan instantiate edilemez. Şablon görevi görür.

generic <T>

Tip-güvenli kod. T herhangi bir reference type olabilir.

where T : class

Constraint: T sadece reference type. Value type yasak.

polymorphism

Aynı interface, farklı davranışlar. Inheritance'ın gücü.

OOP: Encapsulation

Customer.cs — Property'ler ve computed values

```
public class Customer
{
    public int Id { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Email { get; set; }
    public string Phone { get; set; }
    public int CompanyId { get; set; }
    public CustomerStatus Status { get; set; }
    public DateTime CreatedAt { get; set; }

    // Computed property — encapsulation örneği
    public string FullName => $"{FirstName} {LastName}";
}
```

Önemli Kavramlar

- Property syntax (get/set) — alan erişimini kontrol altına alır
- Computed property — runtime'da hesaplanır, DB'de tutulmaz
- Expression-bodied member (=>) — kısa yazım
- POCO sınıf — sadece veri taşır, davranış yok
- String interpolation ($"...") — değişkenleri string'e gömer

💡 **Tasarım Kararı:** Customer sınıfı bilinçli olarak *anemic* — DB veya UI'ı bilmez. Bu, layered architecture'da iş mantığının service/repository katmanına ait olmasının doğal sonucudur.

OOP: Enum Kullanımı

Type safety + Self-documenting code

```
public enum CustomerStatus
{
    Lead = 0,       // Yeni potansiyel
    Active = 1,     // Aktif müşteri
    VIP = 2,        // Stratejik
    Passive = 3     // Pasif
}
```

```
public enum ActivityType
{
    Call = 0,       // Telefon araması
    Meeting = 1,    // Toplantı
    Email = 2,      // E-posta
    Note = 3        // Sistem notu
}
```

Neden Enum?

```
// ✓ Type-safe (compiler kontrol eder)
customer.Status = CustomerStatus.VIP;

// ✗ Hatalı – compiler hatası verir
customer.Status = "VIP";

// DB'de int olarak saklanır
INSERT INTO Customers (Status) VALUES (2);
```

- Type safety — typo'lar compile-time'da yakalanır
- Self-documenting — "VIP" 2'den daha okunabilir
- Performans — int comparison string'den çok hızlı
- DB optimizasyonu — int kolonlar index'lenebilir

Veri İşleme: SQLite Tercihi

Neden file-based bir veritabanı seçtim?

SQLite vs SQL Server LocalDB

Özellik	SQL Server LocalDB	SQLite ✓
Yapı	Ayrı Windows servisi	Process içinde, embedded
Kurulum	LocalDB kurulumu gerek	Hiç kurulum gerekmez
DB Dosyası	.mdf + servis kaydı	Tek .db dosyası
Connection String	Karmaşık (LocalDB)\MSSQL...	Data Source=crm.db;Version=3;
Taşıma	Hedef makinede LocalDB lazım	Dosya kopyala, biter
Demo Riski	Versiyon uyumsuzluğu olabilir	Sıfır risk
ADO.NET API'ı	Aynı (SqlConnection)	Aynı (SQLiteConnection)

Üretim Notu: SQLite bugün iOS, Android, macOS dahil milyarlarca cihazda kullanılan production-grade bir veritabanıdır.

DatabaseHelper.cs

DB yaşam döngüsü yöneticisi · Static class

```
public static class DatabaseHelper
{
    public static string DatabasePath => Path.Combine(
        AppDomain.CurrentDomain.BaseDirectory, "crm.db");

    public static string ConnectionString =>
        $"Data Source={DatabasePath};Version=3;";

    public static SQLiteConnection GetConnection()
    {
        var conn = new SQLiteConnection(ConnectionString);
        conn.Open();
        return conn;
    }

    public static void InitializeDatabase()
    {
        bool isFirstRun = !File.Exists(DatabasePath);
        if (isFirstRun)
            SQLiteConnection.CreateFile(DatabasePath);

        using (var conn = GetConnection())
        {
            CreateSchema(conn);              // CREATE TABLE IF NOT EXISTS
            if (isFirstRun) SeedData(conn);  // İlk veri yüklemesi
        }
    }
}
```

Static class

Tek bir DB → Singleton-lite. State tutmaz, sadece utility.

Idempotent

İlk çalıştırmada DB yarat, sonrasında dokunma.

ACID seed

Transaction içinde toplu INSERT. Yarım kalmaz.

Computed paths

DB her zaman EXE'nin yanında. Taşınabilir.

Repository Pattern

CustomerRepository.Add() — Parametrized SQL örneği

```
public override void Add(Customer customer)
{
    using (var conn = GetConnection())
    using (var cmd = new SQLiteCommand(
        @"INSERT INTO Customers
          (FirstName, LastName, Email, Phone,
           CompanyId, Status, CreatedAt)
          VALUES (@fn, @ln, @email, @phone,
                  @cid, @status, @createdAt)", conn))
    {
        cmd.Parameters.AddWithValue("@fn", customer.FirstName);
        cmd.Parameters.AddWithValue("@ln", customer.LastName);
        cmd.Parameters.AddWithValue("@email", customer.Email);
        cmd.Parameters.AddWithValue("@phone", customer.Phone);
        cmd.Parameters.AddWithValue("@cid", customer.CompanyId);
        cmd.Parameters.AddWithValue("@status", (int)customer.Status);
        cmd.Parameters.AddWithValue("@createdAt",
                                    DateTime.Now.ToString("o"));
        cmd.ExecuteNonQuery();
    }
}
```

Parametrized Query

@fn placeholder'ları SQL injection'a karşı koruma sağlar.

Double using

Connection + Command — IDisposable pattern, kaynak sızıntısı yok.

Enum → int cast

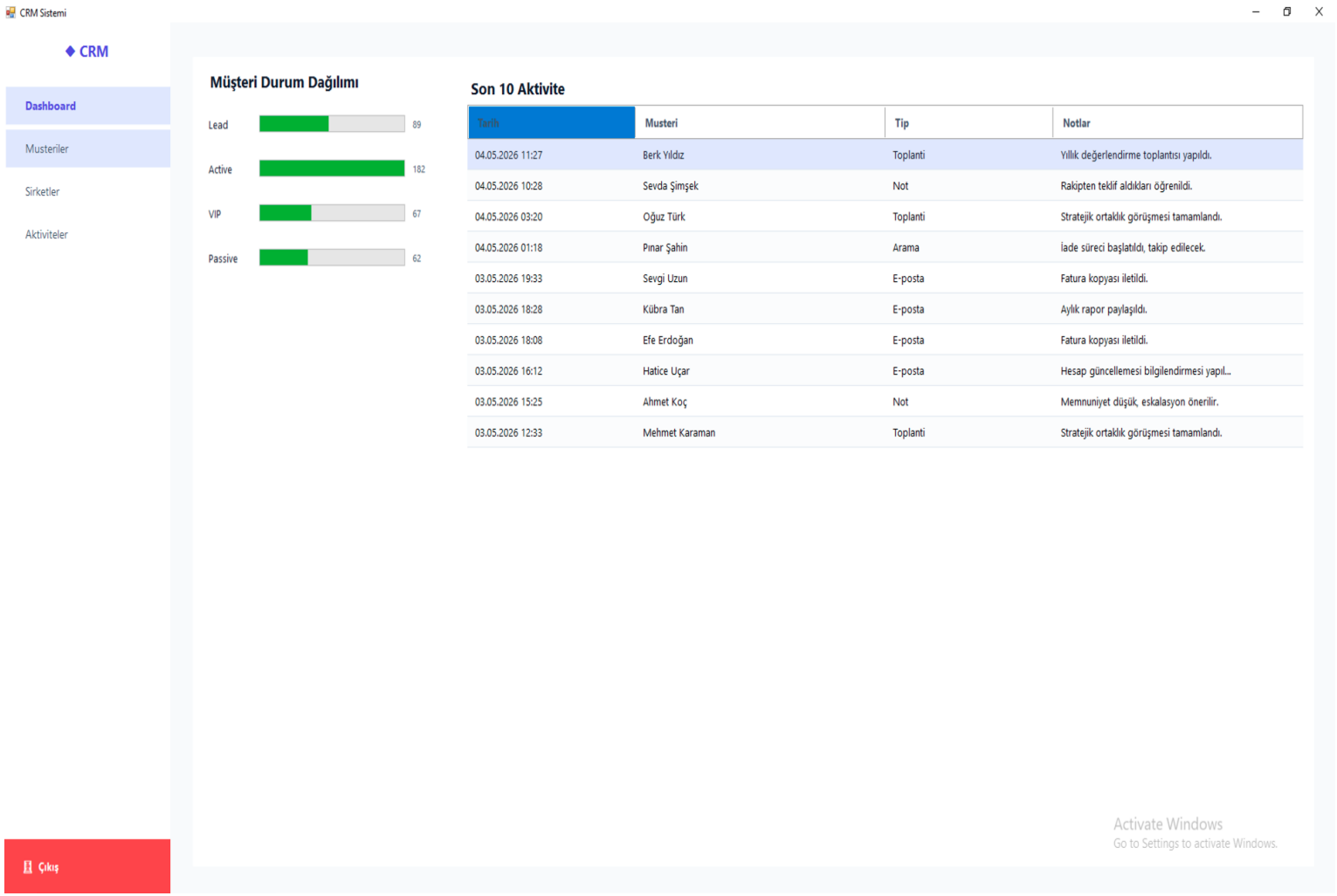
(int)customer.Status — DB integer, kod enum.

ISO 8601 tarih

ToString("o") ile alfabetik sıralama = kronolojik sıralama.

Uygulama Ekranı: Dashboard

MainForm — Ana ekran ve özet kartlar



Bileşenler

Sol Sidebar (240px)

Logo + 4 modül + çıkış butonu

4 Stat Kartı

Müşteri, Aktif, Şirket, Bu Ay Aktivite

Status Dağılımı

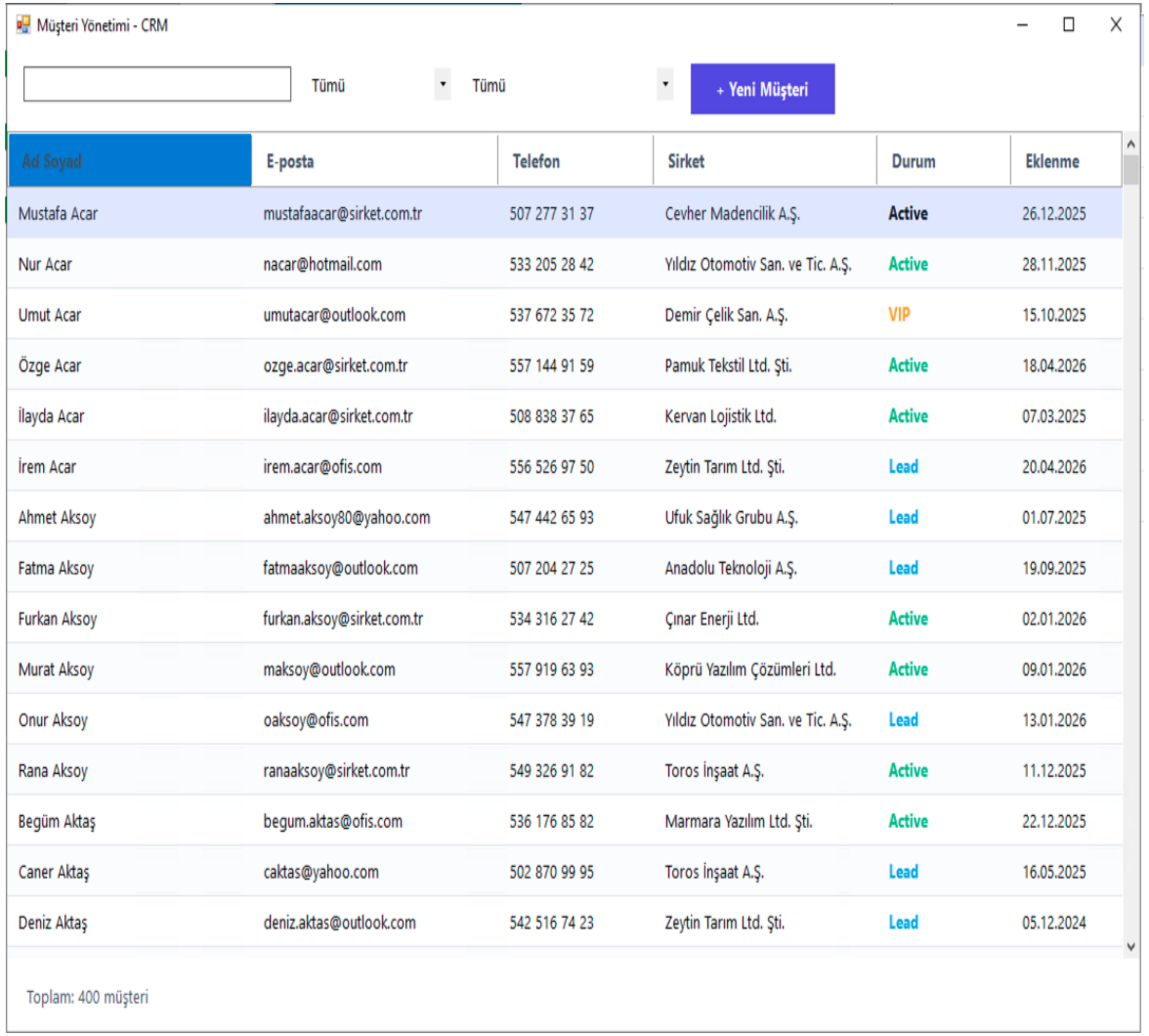
ProgressBar ile Lead/Active/VIP/Passive

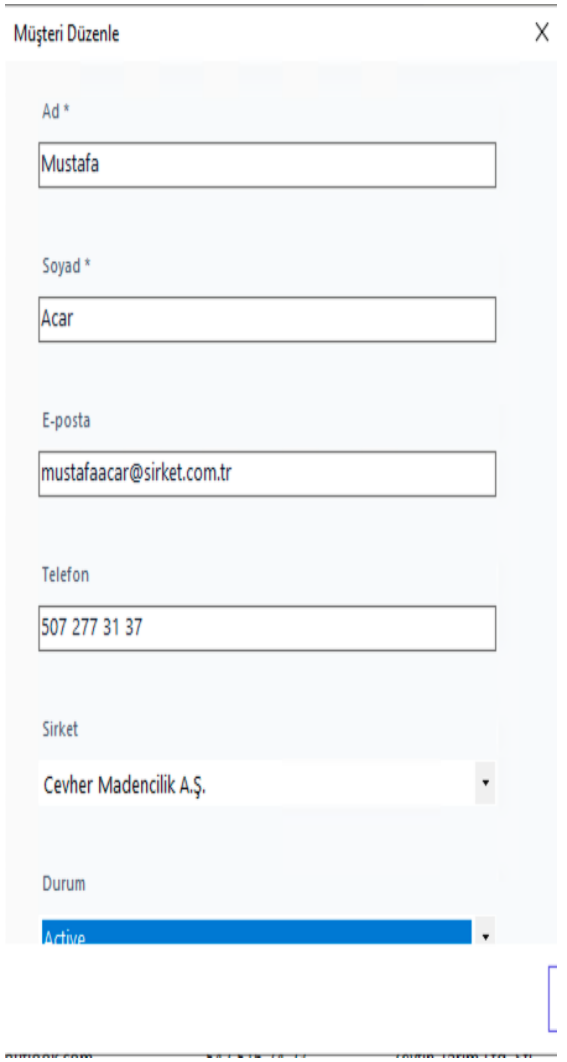
Son 10 Aktivite

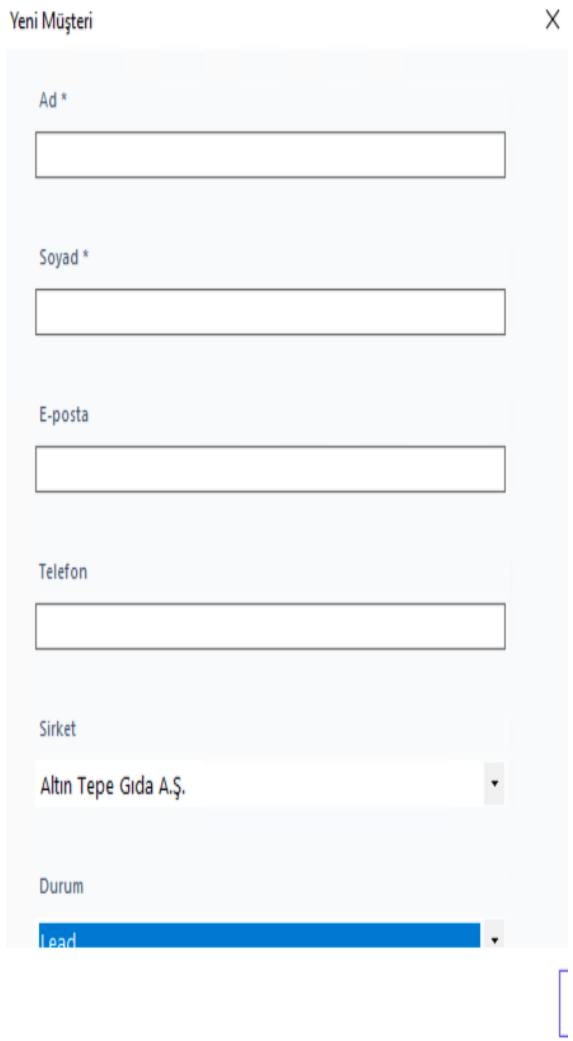
DataGridView, kronolojik

Uygulama Ekranı: Müşteri Modülü

CustomerListForm + CustomerEditForm

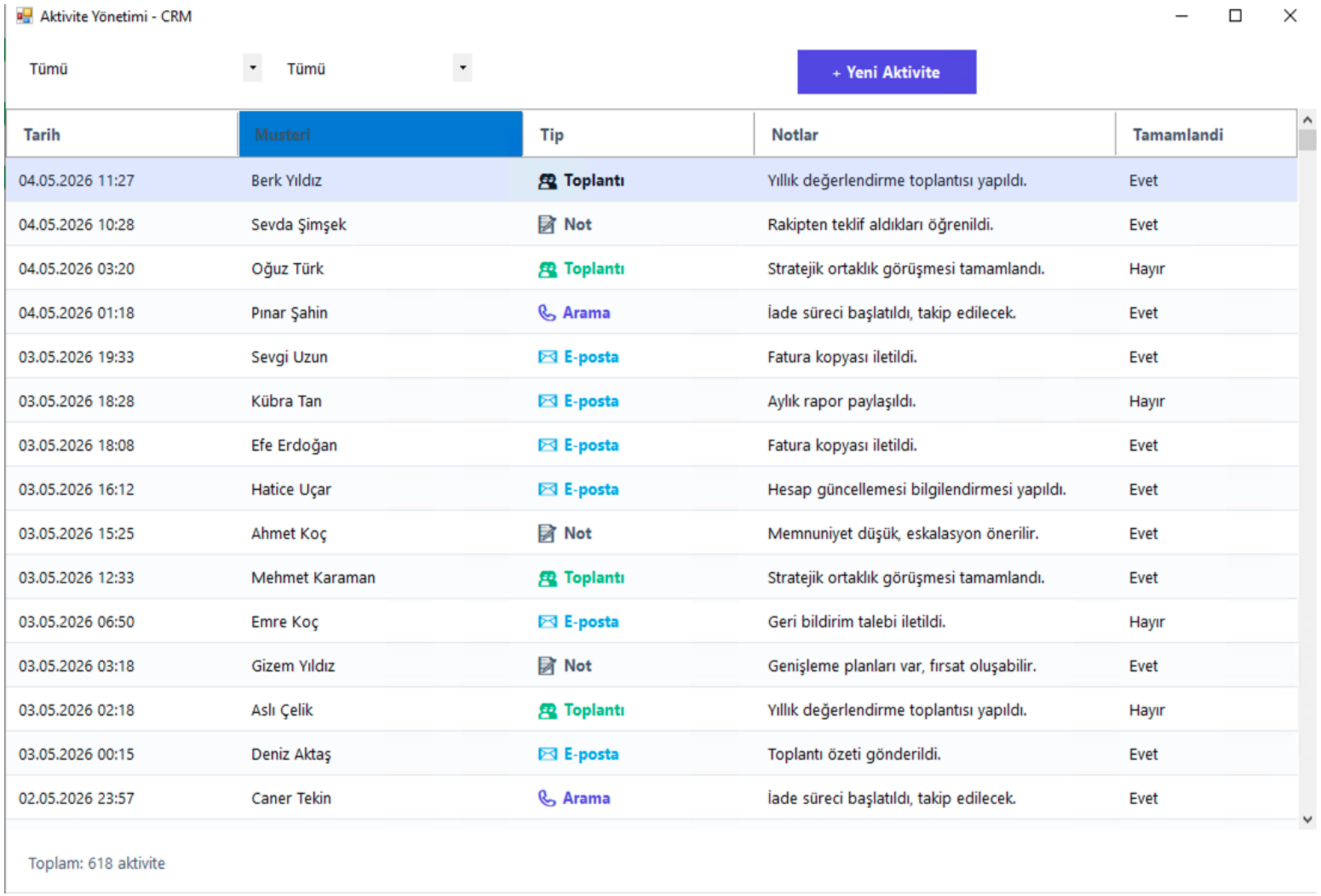






Uygulama Ekranı: Aktivite Modülü

Customer-Activity ilişkisinin pratik uygulaması



Veri İlişkisi

```
Customer.Id (1)
        │
        │ FK: Activity.CustomerId
        ▼
Activity (N)
```

- 1-N ilişki: bir müşterinin N aktivitesi
- Foreign Key bütünlüğü DB seviyesinde
- Kronolojik sıralama (en yeni → eski)
- 4 aktivite tipi: Call, Meeting, Email, Note

Sonuç ve Kazanımlar

🎯 Teknik Kazanımlar

- C# OOP'un gerçek bir projede uygulanması
- Layered architecture pratiği
- Repository Pattern ile soyutlama
- Generic abstract class tasarımı
- Parametrized SQL ve ADO.NET

Kavramsal Kazanımlar

- Sınıflar arası ilişki modellemesi
- Encapsulation, Inheritance, Polymorphism
- Separation of Concerns prensibi
- Single Responsibility Principle
- Type safety ve self-documenting code

🚀 İleride Geliştirme

- Authentication ve yetki yönetimi
- Reporting modülü PDF export
- Cloud sync ve REST API
- Mobile companion (Xamarin/MAUI)
- Logging altyapısı (ILogger)

Teşekkürler! 

Hazırlayan:

Ahmet Baran Bozkurt
11210000045