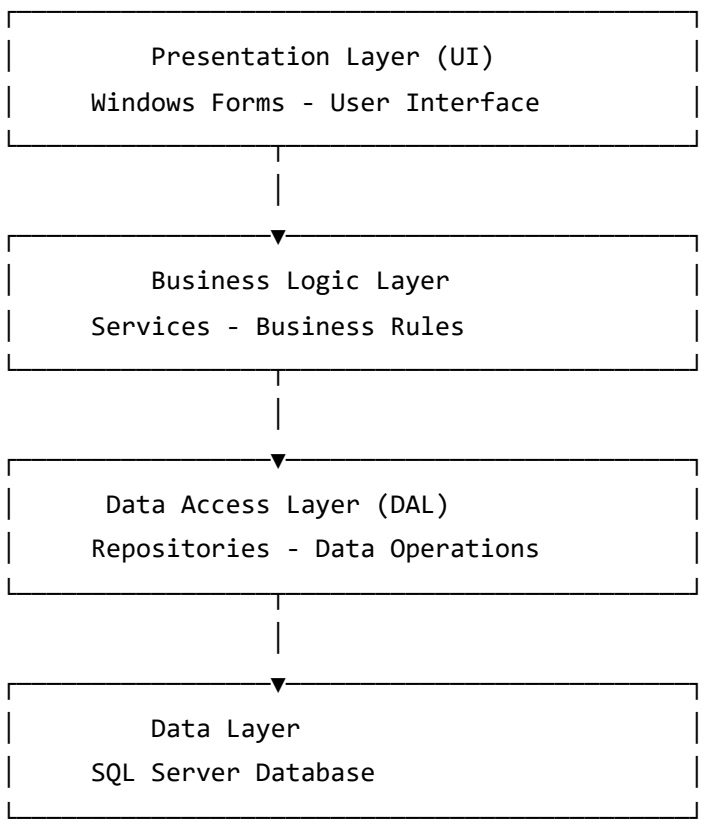
CeyPASS Cihaz Paneli - Teknik Dokümantasyon

İçindekiler

Sistem Mimarisi

Genel Bakış

CeyPASS Cihaz Paneli, **N-Tier (Katmanlı) Mimari** prensiplerine göre tasarlanmış bir Windows Forms uygulamasıdır. Sistem, 4 ana katmandan oluşur:



Mimari Prensipler

1. **Separation of Concerns**: Her katman kendi sorumluluğuna odaklanır
2. **Dependency Inversion**: Üst katmanlar soyutlamalara (interface) bağımlıdır
3. **Single Responsibility**: Her sınıf tek bir sorumluluğa sahiptir
4. **Open/Closed Principle**: Genişletmeye açık, değişikliğe kapalı

Katman Detayları

1. Presentation Layer (UI)

Namespace: `CeyPASSCihazPanel.UI`

Formlar

loginForm.cs

- **Amaç**: Kullanıcı kimlik doğrulama
- **Bağımlılıklar**: `IAuthService` , `IAdminLookupService`

- **Özellikler**:
  - Kullanıcı adı/şifre doğrulama
  - Firma seçimi
  - Oturum başlatma

anaForm.cs

- **Amaç**: Ana uygulama arayüzü
- **Bağımlılıklar**: `IAdminLookupService`, `IDeviceService`
- **Sekmeler**:
  i. **Personel/Kart Tanımlama**: Tekil personel/kart yetkilendirme
  ii. **Toplu Tanımlama**: Çoklu personel tanımlama
  iii. **Toplu Silme**: Çoklu personel silme
  iv. **Yetki Yönetimi**: Cihaz bazlı yetki yönetimi
  v. **Offline Veri**: Cihazlardan veri çekme
  vi. **Cihaz Yönetimi**: Cihaz kontrol ve bilgileri

Önemli Metodlar

```
// Personel tanımlama
private async void btnTanimla_Click(object sender, EventArgs e)
{
    // 1. Seçili personeli al
    // 2. Veritabanından yetkili cihazları kontrol et
    // 3. Cihazlara bağlan
    // 4. Kullanıcıyı cihaza ekle
    // 5. Veritabanını güncelle
}

// Cihaz bağlantısı kontrolü
private bool CihazaEklenmisMi(CihazBaglantisi baglanti, string personelId)
{
    // ZKTeco SDK kullanarak cihazda kullanıcı kontrolü
}
```

2. Business Logic Layer

Namespace: `CeyPASSCihazPanel.Business`

Servisler

IDeviceService / DeviceService

Sorumluluk: Cihaz bağlantı ve işlem yönetimi

Temel Metodlar:

```
public interface IDeviceService
{
    void Start(IList<Terminal> terminals);
    bool TryGetConnection(string ip, out CihazBaglantisi connection);
    Dictionary<string, CihazBaglantisi> GetAllConnections();
    Task<List<OfflineLog>> GetOfflineData(List<string> deviceIps);
    Task<bool> ClearOfflineData(List<string> deviceIps);
    Task<bool> SynchronizeTime(List<string> deviceIps);
    Task<CihazBilgi> GetDeviceInfo(string ip);
    Task<bool> RestartDevice(List<string> deviceIps);
    Task<bool> PowerOffDevice(List<string> deviceIps);
}
```

Önemli Özellikler:

- Cihaz bağlantı havuzu yönetimi (`Dictionary<string, CihazBaglantisi>`)
- Otomatik yeniden bağlanma mekanizması
- Thread-safe operasyonlar
- Asenkron cihaz işlemleri

Bağlantı Yönetimi:

```
private void BaglantilariKontrolEt(object state)
{
    foreach (var kvp in _connections)
    {
        if (!kvp.Value.Axe.GetConnectState())
        {
            YenidenBaglan(kvp.Value);
        }
    }
}
```

IAdminLookupService / AdminLookUpService

Sorumluluk: Veri sorgulama ve yetkilendirme işlemleri

Temel Metodlar:

```
public interface IAdminLookupService
{
    IList<Terminal> GetAktifCihazlar(int? firmaId);
    IList<Personel> GetAktifPersoneller(int? firmaId);
    IList<PuantajsizKart> GetAktifPuantajsizKartlar(int? firmaId);
    IList<PersonelCihazDurum> GetPersonelCihazDurumlari(int personelId, int? firmaId);
    bool PersonelYetkiKaydet(int personelId, List<int> cihazIdler, int? firmaId);
    bool PersonelYetkiSil(int personelId, int cihazId, int? firmaId);
}
```

IAuthService / AuthService

Sorumluluk: Kullanıcı kimlik doğrulama

```
public interface IAuthService
{
    LoginResult Login(string username, string password, int? firmaId);
}
```

3. Data Access Layer (DAL)

Namespace: `CeyPASSCihazPanel.DAL`

Repository Pattern Implementasyonu

Her entity için ayrı repository:

ICihazRepository / SqlCihazRepository

```
public interface ICihazRepository
{
    IList<Terminal> GetAktifCihazlar(int? firmaId);
    IList<CihazListeItem> GetCihazListeItems(int? firmaId);
    int? GetCihazIdByIp(string ip);
}
```

Örnek Implementasyon:

```
public IList<Terminal> GetAktifCihazlar(int? firmaId)
{
    var list = new List<Terminal>();
    using (var conn = new SqlConnection(_connStr))
    {
        conn.Open();
        string sql = @"
            SELECT CihazId, CihazAdi, IP, Port
            FROM Cihazlar
            WHERE Aktif = 1
            AND (@FirmaId IS NULL OR FirmaId = @FirmaId)
            ORDER BY CihazAdi";

        using (var cmd = new SqlCommand(sql, conn))
        {
            cmd.Parameters.AddWithValue("@FirmaId",
                (object)firmaId ?? DBNull.Value);
            // ... reader implementation
        }
    }
    return list;
}
```

IPersonelRepository / SqlPersonelRepository

- `GetAktifPersoneller(int? firmaId)` : Aktif personel listesi
- `GetById(int personelId)` : Personel detayı

IKisiCihazYetkiRepository / SqlKisiCihazYetkiRepository

- `GetYetkiliCihazlar(int personelId)` : Personelin yetkili olduğu cihazlar
- `GetPersonelCihazDurumlari(int personelId, int? firmaId)` : Tüm cihazlardaki durum

- `YetkiKaydet(int personelId, List<int> cihazIdler, int? firmaId)` : Yetki ekleme
- `YetkiSil(int personelId, int cihazId, int? firmaId)` : Yetki silme

4. Entities Layer

Namespace: `CeyPASSCihazPanel.Entities.Models`

Ana Model Sınıfları

Personel

```
public class Personel
{
    public string PersonelId { get; set; }
    public string Ad { get; set; }
    public string Soyad { get; set; }
    public int? FirmaId { get; set; }
    public int? KartNo { get; set; }
    public DateTime? IstenCikisTarihi { get; set; }
}
```

Terminal

```
public class Terminal
{
    public int CihazId { get; set; }
    public string CihazAdi { get; set; }
    public string IP { get; set; }
    public int Port { get; set; }
}
```

CihazBilgi

```
public class CihazBilgi
{
    public string CihazAdi { get; set; }
    public string IPAdres { get; set; }
    public string Model { get; set; }
    public string SeriNo { get; set; }
    public string FirmwareVersion { get; set; }
    public int KullaniciKapasitesi { get; set; }
    public int MevcutKullaniciSayisi { get; set; }
    public int LogKapasitesi { get; set; }
    public int MevcutLogSayisi { get; set; }
    public DateTime? CihazSaati { get; set; }
    public bool BaglantiDurumu { get; set; }
}
```

OfflineLog

```
public class OfflineLog
{
    public string CihazAdi { get; set; }
    public string PersonelId { get; set; }
    public string AdSoyad { get; set; }
    public DateTime Tarih { get; set; }
    public string DogrulamaTipi { get; set; }
    public string GirisCikis { get; set; }
}
```

Veritabanı Yapısı

Tablo Şemaları

Kisiler (Personel)

```
CREATE TABLE Kisiler (
    PersonelId INT PRIMARY KEY IDENTITY,
    Ad NVARCHAR(100),
    Soyad NVARCHAR(100),
    FirmaId INT,
    KartNo INT,
    IstenCikisTarihi DATETIME NULL
)
```

Cihazlar (Terminals)

```
CREATE TABLE Cihazlar (
    CihazId INT PRIMARY KEY IDENTITY,
    CihazAdi NVARCHAR(100),
    IP NVARCHAR(15),
    Port INT,
    FirmaId INT,
    Aktif BIT DEFAULT 1
)
```

KisiCihazYetki (Authorization)

```
CREATE TABLE KisiCihazYetki (
    Id INT PRIMARY KEY IDENTITY,
    PersonelId INT,
    CihazId INT,
    FirmaId INT,
    EklenmeTarihi DATETIME DEFAULT GETDATE(),
    FOREIGN KEY (PersonelId) REFERENCES Kisiler(PersonelId),
    FOREIGN KEY (CihazId) REFERENCES Cihazlar(CihazId)
)
```

Kullanicilar (System Users)

```
CREATE TABLE Kullanicilar (
    KullaniciId INT PRIMARY KEY IDENTITY,
    KullaniciAdi NVARCHAR(50) UNIQUE,
    Sifre NVARCHAR(100),
    FirmaId INT NULL,
    Aktif BIT DEFAULT 1
)
```

Cihaz İletişim Protokolü

ZKTeco SDK Kullanımı

Bağlantı Kurma

```
CZKEMClass axe = new CZKEMClass();
bool connected = axe.Connect_Net(ip, port);
```

Kullanıcı Ekleme

```
bool success = axe.SSR_SetUserInfo(
    machineNumber: 1,
    enrollNumber: personelId,
    name: adSoyad,
    password: "",
    privilege: 0,  // Normal user
    enabled: true
);
```

Kullanıcı Silme

```
bool success = axe.SSR_DeleteEnrollData(
    machineNumber: 1,
    enrollNumber: personelId,
    backupNumber: 12  // Tüm veriler
);
```

Offline Veri Okuma

```
axe.ReadGeneralLogData(machineNumber: 1);
while (axe.SSR_GetGeneralLogData(
    machineNumber: 1,
    out string enrollNumber,
    out int verifyMode,
    out int inOutMode,
    out int year, out int month, out int day,
    out int hour, out int minute, out int second,
    out int workCode))
{
    // Log kaydını işle
}
```

Cihaz Bilgisi Alma

```
axe.GetDeviceInfo(machineNumber, 1, out int userCount);
axe.GetDeviceInfo(machineNumber, 2, out int fpCount);
axe.GetDeviceInfo(machineNumber, 8, out int logCount);
axe.GetSerialNumber(machineNumber, out string serialNumber);
axe.GetFirmwareVersion(machineNumber, out string firmwareVersion);
```

Doğrulama Tipleri (Verify Mode)

- `0` : Şifre
- `1` : Parmak izi
- `3` : Şifre + Parmak izi
- `4` : Yüz tanıma
- `15` : Diğer

Giriş/Çıkış Modları (InOut Mode)

- `0` : Giriş
- `1` : Çıkış
- `2` : Ara çıkış
- `3` : Ara giriş
- `4` : Mesai başlangıcı
- `5` : Mesai bitişi

Güvenlik ve Yetkilendirme

Kimlik Doğrulama Akışı

1. Kullanıcı login formunda kimlik bilgilerini girer
2. `AuthService.Login()` metodu çağrılır
3. Veritabanında kullanıcı sorgulanır
4. Şifre kontrolü yapılır (plain text - güvenlik iyileştirmesi gerekli)
5. `LoginResult` döndürülür
6. Başarılı ise `anaForm` açılır ve kullanıcı bilgileri aktarılır

Firma Bazlı Veri İzolasyonu

Tüm sorgularda `@FirmaId` parametresi kullanılır:

```
WHERE (@FirmaId IS NULL OR FirmaId = @FirmaId)
```

- `FirmaId = NULL` : Tüm firmalar (admin)
- `FirmaId = X` : Sadece X firmasının verileri

SQL Injection Koruması

Tüm sorgularda parametreli komutlar kullanılır:

```
cmd.Parameters.AddWithValue("@PersonelId", personelId);
```

Hata Yönetimi

Exception Handling Stratejisi

UI Katmanı

```
try
{
    // İşlem
}
catch (Exception ex)
{
    MessageBox.Show($"Hata: {ex.Message}", "Hata",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    LogYaz($"HATA: {ex.Message}");
}
```

Business Katmanı

```
try
{
    // Cihaz işlemi
}
catch (Exception ex)
{
    // Log ve false döndür
    return false;
}
```

Cihaz Bağlantı Hataları

- Otomatik yeniden bağlanma mekanizması
- Timer ile periyodik bağlantı kontrolü (30 saniye)
- Bağlantı durumu göstergesi

Performans ve Optimizasyon

Asenkron İşlemler

Cihaz işlemleri için `async/await` kullanımı:

```
private async void btnOfflineVeriCek_Click(object sender, EventArgs e)
{
    var logs = await _deviceService.GetOfflineData(deviceIps);
    // UI güncelleme
}
```

Connection Pooling

SQL Server bağlantıları otomatik olarak pool'lanır:

```
using (var conn = new SqlConnection(_connStr))
{
    // Bağlantı kullanımdan sonra pool'a döner
}
```

DataGridView Optimizasyonu

- Virtual mode kullanımı (büyük veri setleri için)
- Lazy loading
- Sayfalama desteği

ComboBox Optimizasyonu

```
private void AdjustComboDropDownWidth(ComboBox combo)
{
    int maxWidth = combo.Width;
    using (Graphics g = combo.CreateGraphics())
    {
        foreach (var item in combo.Items)
        {
            int itemWidth = (int)g.MeasureString(
                item.ToString(), combo.Font).Width;
            maxWidth = Math.Max(maxWidth, itemWidth);
        }
    }
    combo.DropDownWidth = maxWidth + 20;
}
```

Deployment ve Konfigürasyon

Konfigürasyon Dosyası (App.config)

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0"
            sku=".NETFramework,Version=v4.7.2" />
    </startup>
    <connectionStrings>
        <add name="CeyPASS"
             connectionString="Server=SERVER\INSTANCE;
                               Database=CeyPASS;
                               User Id=USERNAME;
                               Password=PASSWORD;" />
    </connectionStrings>
</configuration>
```

Otomatik Güncelleme Konfigürasyonu

update.xml (Web sunucusunda):

```
<?xml version="1.0" encoding="UTF-8"?>
<item>
    <version>1.0.0.1</version>
    <url>http://server/updates/CeyPASSCihazPanel.zip</url>
    <changelog>
        <![CDATA[
        - Yeni özellikler
        - Hata düzeltmeleri
        ]]>
    </changelog>
    <mandatory>true</mandatory>
</item>
```

Program.cs güncelleme ayarları:

```
AutoUpdater.Mandatory = true;
AutoUpdater.UpdateMode = Mode.ForcedDownload;
AutoUpdater.ShowSkipButton = false;
AutoUpdater.Start(@"http://192.168.0.23/CeyPASS-CihazPanel-Updates/update.xml");
```

Dependency Injection Konfigürasyonu

```
private static void ConfigureServices(IServiceCollection services)
{
    // UI
    services.AddTransient<loginForm>();
    services.AddTransient<anaForm>();

    // DataAccess
    services.AddScoped<IUserRepository, SqlUserRepository>();
    services.AddScoped<ICihazRepository, SqlCihazRepository>();
    services.AddScoped<IPersonelRepository, SqlPersonelRepository>();
    services.AddScoped<IPuantajsizKartRepository, SqlPuantajsizKartRepository>();
    services.AddScoped<IKisiCihazYetkiRepository, SqlKisiCihazYetkiRepository>();
    services.AddScoped<IPuantajsizKartCihazYetkiRepository,
        SqlPuantajsizKartCihazYetkiRepository>();

    // Business
    services.AddScoped<IAuthService, AuthService>();
    services.AddScoped<IAdminLookupService, AdminLookUpService>();
    services.AddScoped<IDeviceService, DeviceService>();
}
```

Build ve Release

Debug Build

```
Configuration: Debug
Platform: Any CPU
Output: bin\Debug\CeyPASSCihazPanel.exe
```

Release Build

```
Configuration: Release
Platform: Any CPU
Output: bin\Release\CeyPASSCihazPanel.exe
Optimizations: Enabled
```

Setup Projesi

`CeyPASSCihazPanel.Setup` projesi ile MSI installer oluşturulur:

- Gerekli DLL'ler
- .NET Framework 4.7.2 prerequisite
- zkemkeeper COM bileşeni
- Kısayol oluşturma

Geliştirme Best Practices

Kod Standartları

1. **Naming Conventions**:
   - PascalCase: Class, Method, Property
   - camelCase: Local variables, parameters
   - _camelCase: Private fields
2. **Async Naming**:
   - Async metodlar `Async` suffix alır: `GetDataAsync()`
3. **Interface Naming**:
   - `I` prefix: `IDeviceService`

Testing Stratejisi

- Unit testler için xUnit veya NUnit kullanımı önerilir
- Repository'ler için mock data kullanımı
- Integration testler için test veritabanı

Logging

UI üzerinde log görüntüleme:

```
private void LogYaz(string mesaj)
{
    txtLog.AppendText($"[{DateTime.Now:HH:mm:ss}] {mesaj}\r\n");
}
```

Gelecek iyileştirme: Dosya bazlı logging (NLog, Serilog)

Sorun Giderme

Yaygın Hatalar

1. Cihaz Bağlantı Hatası

Hata: "Cihaza bağlanılamadı"
Çözüm:

- IP adresini kontrol edin
- Port numarasını kontrol edin (varsayılan: 4370)
- Ağ bağlantısını kontrol edin
- Firewall ayarlarını kontrol edin

2. Veritabanı Bağlantı Hatası

Hata: "SQL Server'a bağlanılamadı"
Çözüm:

- Connection string'i kontrol edin
- SQL Server servisinin çalıştığından emin olun
- Kullanıcı adı/şifre doğruluğunu kontrol edin
- Veritabanı erişim izinlerini kontrol edin

3. COM Bileşeni Hatası

Hata: "zkemkeeper COM bileşeni bulunamadı"
Çözüm:

- ZKTeco SDK'sını yükleyin
- COM bileşenini kaydedin: `regsvr32 zkemkeeper.dll`

Versiyon Geçmişi

v1.0 (Mevcut)

- İlk stabil sürüm
- Temel personel/kart yönetimi
- Cihaz bağlantı ve kontrol
- Offline veri toplama
- Otomatik güncelleme

Gelecek Teknik İyileştirmeler

Güvenlik

- [ ] Şifre hash'leme (BCrypt, PBKDF2)
- [ ] JWT token bazlı authentication
- [ ] Role-based access control (RBAC)
- [ ] Audit logging

Performans

- [ ] Redis cache entegrasyonu
- [ ] Asenkron veritabanı işlemleri (Dapper)
- [ ] Bulk insert/update operasyonları

Mimari

- [ ] CQRS pattern implementasyonu
- [ ] Event-driven architecture
- [ ] Microservices dönüşümü
- [ ] REST API katmanı

Monitoring

- [ ] Application Insights entegrasyonu
- [ ] Health check endpoints

- [ ] Performance metrics
- [ ] Error tracking (Sentry, Raygun)

Doküman Versiyonu: 1.0
Son Güncelleme: 26 Aralık 2025
Hazırlayan: Tahir Koca