نظام إدارة المخزون المُبسّط

Table of Contents

ASP.NET Core MVC مع ApplicationUser في نفس Database

المحتويات

1. المعمارية المُبسّطة (Database واحد فقط)
2. ApplicationUser المُحسّن
3. الموديلات والعلاقات
4. إدارة المستخدمين والصلاحيات
5. الخدمات والـ Controllers
6. الشامل والمنظم ERD

الجزء الأول: المعمارية المُبسّطة

الفرق مع النسخة السابقة:

العنصر	النسخة السابقة	النسخة الجديدة
عدد قواعد البيانات	2 (InventoryDB + IdentityDB)	1 فقط (InventoryDB)
جداول الموظفين	Employee + Department	محذوفة تماماً
إدارة المستخدمين	AspNetUsers منفصلة	AspNetUsers في نفس DB

العنصر	النسخة السابقة	النسخة الجديدة
الربط	AspNetUserId في Employee	UserId في العمليات مباشرة
الصلاحيات	Roles + Claims	Roles من الهوية

الفوائد:

✅ قاعدة بيانات واحدة = بساطة أكثر
✅ لا حاجة لـ Dependency Injection معقد
✅ أداء أفضل (لا Joins بين DBs)
✅ إدارة أسهل للبيانات
✅ موحّد وآمن Transactions

الجزء الثاني: ApplicationUser المُحسّن

```
using Microsoft.AspNetCore.Identity;

public class ApplicationUser : IdentityUser
{
    // معلومات الملف الشخصي
    [Required(ErrorMessage = "الاسم الأول مطلوب")]
    [StringLength(100)]
    public string FirstName { get; set; }

    [Required(ErrorMessage = "الاسم الأخير مطلوب")]
    [StringLength(100)]
    public string LastName { get; set; }

    // الاسم الكامل (خاصية computed)
    [NotMapped]
    public string FullName =&gt; $"{FirstName} {LastName}";

    // معلومات إضافية
    public DateTime? HireDate { get; set; }
    public DateTime? BirthDate { get; set; }
    public string Address { get; set; }
    public string ProfileImageUrl { get; set; }

    // حالة المستخدم
    public bool IsActive { get; set; } = true;
    public DateTime CreatedAt { get; set; } = DateTime.Now;
    public DateTime? ModifiedAt { get; set; }
    public DateTime? LastLogin { get; set; }

    // العلاقات مع العمليات
    public ICollection&lt;SalesOrder&gt; SalesOrders { get; set; }
    public ICollection&lt;PurchaseOrder&gt; PurchaseOrders { get; set; }
    public ICollection&lt;StockMovement&gt; StockMovements { get; set; }
}
```

الجزء الثالث: جدول المنتجات والتصنيفات

```
public class Category
{
    public int Id { get; set; }

    [Required(ErrorMessage = "اسم التصنيف مطلوب")]
    [StringLength(100)]
```

```
    public string Name { get; set; }

    [StringLength(500)]
    public string Description { get; set; }

    public DateTime CreatedAt { get; set; } = DateTime.Now;
    public DateTime? ModifiedAt { get; set; }

    // العلاقة مع المنتجات
    public ICollection&lt;Product&gt; Products { get; set; } = new List&lt;Product&gt;();
}

public class Product
{
    public int Id { get; set; }

    [Required(ErrorMessage = "اسم المنتج مطلوب")]
    [StringLength(200)]
    public string Name { get; set; }

    [StringLength(1000)]
    public string Description { get; set; }

    [Required(ErrorMessage = "السعر مطلوب")]
    [Range(0.01, double.MaxValue)]
    public decimal Price { get; set; }

    [Required(ErrorMessage = "الكمية المتاحة مطلوبة")]
    [Range(0, int.MaxValue)]
    public int QuantityInStock { get; set; }

    [Required(ErrorMessage = "الحد الأدنى للمخزون مطلوب")]
    [Range(1, int.MaxValue)]
    public int MinimumStockLevel { get; set; }

    // الرمز الشريطي
    [Required(ErrorMessage = "رمز المنتج مطلوب")]
    [StringLength(128)]
    public string Barcode { get; set; }

    public string BarcodeImage { get; set; }

    // الربط مع التصنيف
    [Required]
    public int CategoryId { get; set; }
    public Category Category { get; set; }

    public DateTime CreatedAt { get; set; } = DateTime.Now;
    public DateTime? ModifiedAt { get; set; }

    // العلاقات
    public ICollection&lt;SalesOrderDetail&gt; SalesOrderDetails { get; set; }
    public ICollection&lt;PurchaseOrderDetail&gt; PurchaseOrderDetails { get; set; }
    public ICollection&lt;StockMovement&gt; StockMovements { get; set; }
}
```

الجزء الرابع: جداول الطلبات والفواتير

حالة الطلب

```
public enum OrderStatus
{
    Pending,       // قيد الانتظار
    Confirmed,     // مؤكد
    Shipped,       // مرسل
    Delivered,     // تم التسليم
    Cancelled,     // ملغي
    Completed      // مكتمل
}

public enum PaymentStatus
{
    PaidInFull,       // تم دفع المبلغ كامل
    PartiallyPaid,    // تم دفع جزء وباقي جزء
    Unpaid            // لم يتم الدفع
}
```

طلب البيع (Sales Order)

```
public class SalesOrder
{
    public int Id { get; set; }

    // الربط مع العميل
    [Required]
    public int CustomerId { get; set; }
    public Customer Customer { get; set; }

    // الربط مع المستخدم (من يقوم بإدخال الطلب)
    [Required]
    public string UserId { get; set; }
    public ApplicationUser User { get; set; }

    // بيانات الطلب
    [Required]
    [StringLength(50)]
    public string InvoiceNumber { get; set; }

    [Required]
    public DateTime OrderDate { get; set; } = DateTime.Now;

    [StringLength(500)]
    public string Notes { get; set; }

    // حالات الطلب والدفع
    [Required]
    public OrderStatus Status { get; set; } = OrderStatus.Pending;

    [Required]
    public PaymentStatus PaymentStatus { get; set; } = PaymentStatus.Unpaid;

    // الحسابات المالية
    public decimal TotalAmount { get; set; }
    public decimal DiscountAmount { get; set; } = 0;
    public decimal TaxPercentage { get; set; } = 0;
    public decimal TaxAmount { get; set; }
    public decimal TotalWithTax { get; set; }
    public decimal PaidAmount { get; set; } = 0;
    public decimal RemainingAmount { get; set; }

    // تواريخ إضافية
```

```
    public DateTime? ShippedDate { get; set; }
    public DateTime? DeliveredDate { get; set; }
    public DateTime CreatedAt { get; set; } = DateTime.Now;
    public DateTime? ModifiedAt { get; set; }

    // العلاقة مع التفاصيل
    [Required]
    public ICollection&lt;SalesOrderDetail&gt; SalesOrderDetails { get; set; } = new List&
}

public class SalesOrderDetail
{
    public int Id { get; set; }

    [Required]
    public int SalesOrderId { get; set; }
    public SalesOrder SalesOrder { get; set; }

    [Required]
    public int ProductId { get; set; }
    public Product Product { get; set; }

    [Required]
    [Range(1, int.MaxValue)]
    public int Quantity { get; set; }

    [Required]
    [Range(0.01, double.MaxValue)]
    public decimal UnitPrice { get; set; }

    [Range(0, 100)]
    public decimal DiscountPercentage { get; set; } = 0;

    public decimal Subtotal =&gt; Quantity * UnitPrice * (1 - DiscountPercentage / 100);

    [StringLength(300)]
    public string Notes { get; set; }

    public DateTime AddedAt { get; set; } = DateTime.Now;
}
```

طلب الشراء (Purchase Order)

```
public class PurchaseOrder
{
    public int Id { get; set; }

    // الربط مع الموردين
    [Required]
    public int SupplierId { get; set; }
    public Supplier Supplier { get; set; }

    // الربط مع المستخدم
    [Required]
    public string UserId { get; set; }
    public ApplicationUser User { get; set; }

    // بيانات الطلب
    [Required]
    [StringLength(50)]
    public string InvoiceNumber { get; set; }

    [Required]
    public DateTime OrderDate { get; set; } = DateTime.Now;
```

```
    [StringLength(500)]
    public string Notes { get; set; }

    // حالات الطلب والدفع
    [Required]
    public OrderStatus Status { get; set; } = OrderStatus.Pending;

    [Required]
    public PaymentStatus PaymentStatus { get; set; } = PaymentStatus.Unpaid;

    // الحسابات المالية
    public decimal TotalAmount { get; set; }
    public decimal DiscountAmount { get; set; } = 0;
    public decimal TaxPercentage { get; set; } = 0;
    public decimal TaxAmount { get; set; }
    public decimal TotalWithTax { get; set; }
    public decimal PaidAmount { get; set; } = 0;
    public decimal RemainingAmount { get; set; }

    // تواريخ إضافية
    public DateTime? ReceivedDate { get; set; }
    public DateTime CreatedAt { get; set; } = DateTime.Now;
    public DateTime? ModifiedAt { get; set; }

    [Required]
    public ICollection&lt;PurchaseOrderDetail&gt; PurchaseOrderDetails { get; set; } = new
}

public class PurchaseOrderDetail
{
    public int Id { get; set; }

    [Required]
    public int PurchaseOrderId { get; set; }
    public PurchaseOrder PurchaseOrder { get; set; }

    [Required]
    public int ProductId { get; set; }
    public Product Product { get; set; }

    [Required]
    [Range(1, int.MaxValue)]
    public int Quantity { get; set; }

    [Required]
    [Range(0.01, double.MaxValue)]
    public decimal UnitPrice { get; set; }

    [Range(0, 100)]
    public decimal DiscountPercentage { get; set; } = 0;

    public decimal Subtotal =&gt; Quantity * UnitPrice * (1 - DiscountPercentage / 100);

    [StringLength(300)]
    public string Notes { get; set; }

    public DateTime AddedAt { get; set; } = DateTime.Now;
}
```

الجزء الخامس: جداول الموردين والعملاء

```
public class Supplier
{
    public int Id { get; set; }

    [Required(ErrorMessage = "اسم الموردين مطلوب")]
    [StringLength(200)]
    public string Name { get; set; }

    [StringLength(500)]
    public string Description { get; set; }

    [Required(ErrorMessage = "البريد الإلكتروني مطلوب")]
    [EmailAddress]
    public string Email { get; set; }

    [Phone]
    public string Phone { get; set; }

    [StringLength(300)]
    public string Address { get; set; }

    [StringLength(100)]
    public string City { get; set; }

    public DateTime CreatedAt { get; set; } = DateTime.Now;
    public DateTime? ModifiedAt { get; set; }

    public ICollection&lt;PurchaseOrder&gt; PurchaseOrders { get; set; }
}

public class Customer
{
    public int Id { get; set; }

    [Required(ErrorMessage = "اسم العميل مطلوب")]
    [StringLength(200)]
    public string Name { get; set; }

    [StringLength(500)]
    public string Description { get; set; }

    [Required(ErrorMessage = "البريد الإلكتروني مطلوب")]
    [EmailAddress]
    public string Email { get; set; }

    [Phone]
    public string Phone { get; set; }

    [StringLength(300)]
    public string Address { get; set; }

    [StringLength(100)]
    public string City { get; set; }

    public DateTime CreatedAt { get; set; } = DateTime.Now;
    public DateTime? ModifiedAt { get; set; }

    public ICollection&lt;SalesOrder&gt; SalesOrders { get; set; }
}
```

الجزء السادس: تتبع حركات المخزون

```
public class StockMovement
{
    public int Id { get; set; }

    [Required]
    public int ProductId { get; set; }
    public Product Product { get; set; }

    [Required]
    [StringLength(50)]
    public string Type { get; set; } // "Purchase", "Sale", "Adjustment", "Return"

    [Required]
    [Range(1, int.MaxValue)]
    public int Quantity { get; set; }

    // الربط مع المستخدم الذي قام بالعملية
    [Required]
    public string UserId { get; set; }
    public ApplicationUser User { get; set; }

    // رقم المرجع (OrderId أو غيره)
    public int? ReferenceOrderId { get; set; }

    [StringLength(300)]
    public string Notes { get; set; }

    public DateTime Date { get; set; } = DateTime.Now;
}
```

الجزء السابع: DbContext الموحّد

```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

public class InventoryDbContext : IdentityDbContext&lt;ApplicationUser&gt;
{
    public InventoryDbContext(DbContextOptions&lt;InventoryDbContext&gt; options)
        : base(options)
    {
    }

    // جداول المنتجات والمخزون
    public DbSet&lt;Category&gt; Categories { get; set; }
    public DbSet&lt;Product&gt; Products { get; set; }
    public DbSet&lt;StockMovement&gt; StockMovements { get; set; }

    // جداول الطلبات والفواتير
    public DbSet&lt;SalesOrder&gt; SalesOrders { get; set; }
    public DbSet&lt;SalesOrderDetail&gt; SalesOrderDetails { get; set; }
    public DbSet&lt;PurchaseOrder&gt; PurchaseOrders { get; set; }
    public DbSet&lt;PurchaseOrderDetail&gt; PurchaseOrderDetails { get; set; }

    // جداول الموردين والعملاء
    public DbSet&lt;Supplier&gt; Suppliers { get; set; }
    public DbSet&lt;Customer&gt; Customers { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);
```

```
        // تكوين العلاقات
        // Product -&gt; Category (One-to-Many)
        modelBuilder.Entity&lt;Product&gt;()
            .HasOne(p =&gt; p.Category)
            .WithMany(c =&gt; c.Products)
            .HasForeignKey(p =&gt; p.CategoryId)
            .OnDelete(DeleteBehavior.Restrict);

        // SalesOrder -&gt; ApplicationUser (Many-to-One)
        modelBuilder.Entity&lt;SalesOrder&gt;()
            .HasOne(o =&gt; o.User)
            .WithMany(u =&gt; u.SalesOrders)
            .HasForeignKey(o =&gt; o.UserId)
            .OnDelete(DeleteBehavior.Restrict);

        // SalesOrder -&gt; Customer (Many-to-One)
        modelBuilder.Entity&lt;SalesOrder&gt;()
            .HasOne(o =&gt; o.Customer)
            .WithMany(c =&gt; c.SalesOrders)
            .HasForeignKey(o =&gt; o.CustomerId)
            .OnDelete(DeleteBehavior.Restrict);

        // SalesOrder -&gt; SalesOrderDetails (One-to-Many)
        modelBuilder.Entity&lt;SalesOrder&gt;()
            .HasMany(o =&gt; o.SalesOrderDetails)
            .WithOne(d =&gt; d.SalesOrder)
            .HasForeignKey(d =&gt; d.SalesOrderId)
            .OnDelete(DeleteBehavior.Cascade);

        // SalesOrderDetail -&gt; Product (Many-to-One)
        modelBuilder.Entity&lt;SalesOrderDetail&gt;()
            .HasOne(d =&gt; d.Product)
            .WithMany(p =&gt; p.SalesOrderDetails)
            .HasForeignKey(d =&gt; d.ProductId)
            .OnDelete(DeleteBehavior.Restrict);

        // PurchaseOrder -&gt; ApplicationUser (Many-to-One)
        modelBuilder.Entity&lt;PurchaseOrder&gt;()
            .HasOne(o =&gt; o.User)
            .WithMany(u =&gt; u.PurchaseOrders)
            .HasForeignKey(o =&gt; o.UserId)
            .OnDelete(DeleteBehavior.Restrict);

        // PurchaseOrder -&gt; Supplier (Many-to-One)
        modelBuilder.Entity&lt;PurchaseOrder&gt;()
            .HasOne(o =&gt; o.Supplier)
            .WithMany(s =&gt; s.PurchaseOrders)
            .HasForeignKey(o =&gt; o.SupplierId)
            .OnDelete(DeleteBehavior.Restrict);

        // PurchaseOrder -&gt; PurchaseOrderDetails (One-to-Many)
        modelBuilder.Entity&lt;PurchaseOrder&gt;()
            .HasMany(o =&gt; o.PurchaseOrderDetails)
            .WithOne(d =&gt; d.PurchaseOrder)
            .HasForeignKey(d =&gt; d.PurchaseOrderId)
            .OnDelete(DeleteBehavior.Cascade);

        // PurchaseOrderDetail -&gt; Product (Many-to-One)
        modelBuilder.Entity&lt;PurchaseOrderDetail&gt;()
            .HasOne(d =&gt; d.Product)
            .WithMany(p =&gt; p.PurchaseOrderDetails)
            .HasForeignKey(d =&gt; d.ProductId)
            .OnDelete(DeleteBehavior.Restrict);

        // StockMovement -&gt; Product (Many-to-One)
```

```
        modelBuilder.Entity&lt;StockMovement&gt;()
            .HasOne(m =&gt; m.Product)
            .WithMany(p =&gt; p.StockMovements)
            .HasForeignKey(m =&gt; m.ProductId)
            .OnDelete(DeleteBehavior.Restrict);

        // StockMovement -&gt; ApplicationUser (Many-to-One)
        modelBuilder.Entity&lt;StockMovement&gt;()
            .HasOne(m =&gt; m.User)
            .WithMany(u =&gt; u.StockMovements)
            .HasForeignKey(m =&gt; m.UserId)
            .OnDelete(DeleteBehavior.Restrict);

        // Indexes للبحث السريع
        modelBuilder.Entity&lt;Product&gt;()
            .HasIndex(p =&gt; p.Barcode)
            .IsUnique();

        modelBuilder.Entity&lt;SalesOrder&gt;()
            .HasIndex(o =&gt; o.InvoiceNumber)
            .IsUnique();

        modelBuilder.Entity&lt;PurchaseOrder&gt;()
            .HasIndex(o =&gt; o.InvoiceNumber)
            .IsUnique();

        modelBuilder.Entity&lt;ApplicationUser&gt;()
            .HasIndex(u =&gt; u.Email)
            .IsUnique();
    }
}
```

الجزء الثامن: تكوين Program.cs

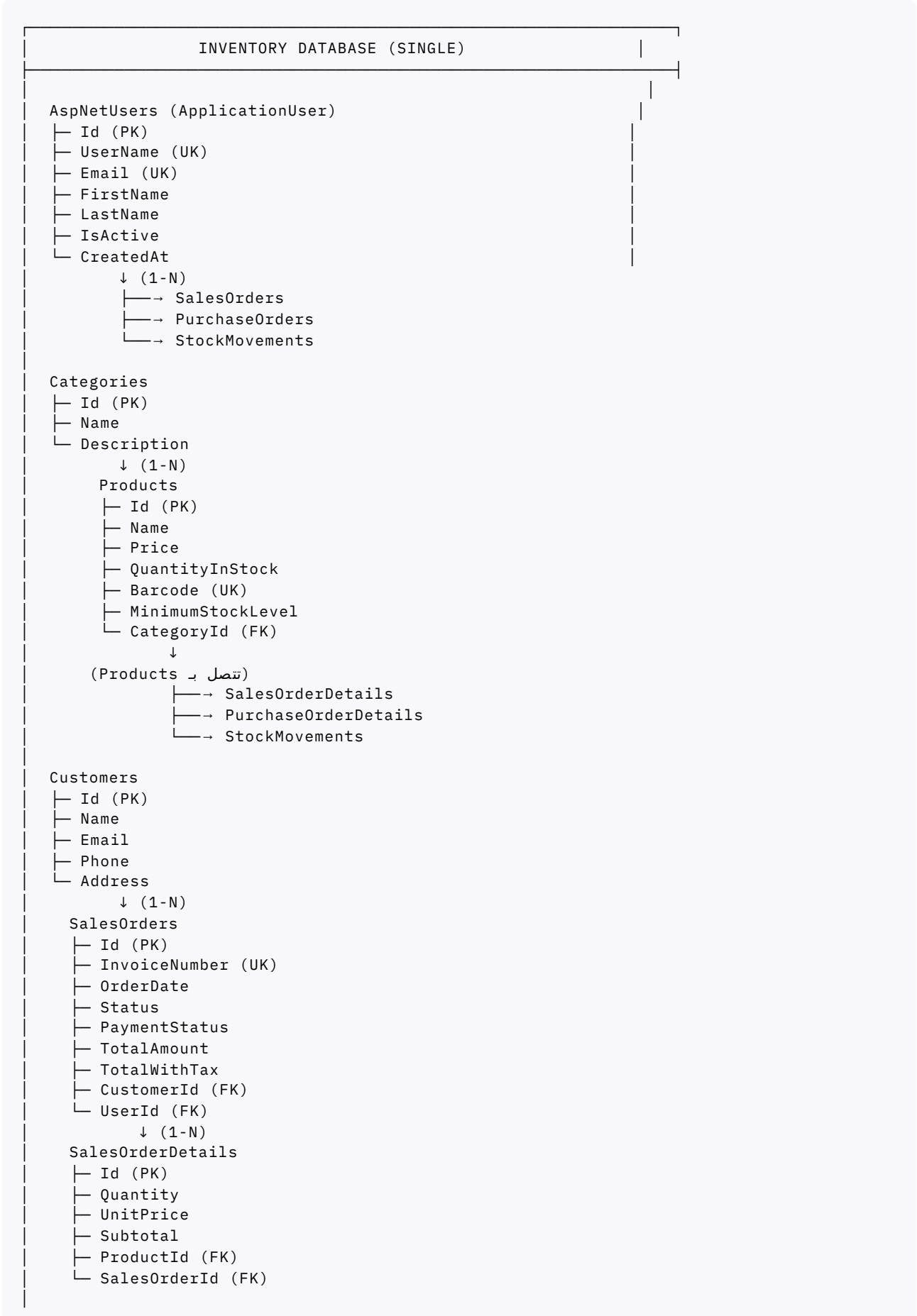
```
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;

var builder = WebApplicationBuilder.CreateBuilder(args);

// إضافة DbContext
builder.Services.AddDbContext&lt;InventoryDbContext&gt;(options =&gt;
    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

// إضافة Identity
builder.Services.AddIdentity&lt;ApplicationUser, IdentityRole&gt;()
    .AddEntityFrameworkStores&lt;InventoryDbContext&gt;()
    .AddDefaultTokenProviders();

// تكوين سياسات كلمات المرور
builder.Services.Configure&lt;IdentityOptions&gt;(options =&gt;
{
    options.Password.RequiredLength = 8;
    options.Password.RequireDigit = true;
    options.Password.RequireNonAlphanumeric = true;
    options.Password.RequireUppercase = true;
    options.Password.RequireLowercase = true;
});

// إضافة الخدمات
builder.Services.AddScoped&lt;IUnitOfWork, UnitOfWork&gt;();
builder.Services.AddScoped&lt;ISalesOrderService, SalesOrderService&gt;();
builder.Services.AddScoped&lt;IPurchaseOrderService, PurchaseOrderService&gt;();
builder.Services.AddScoped&lt;IUserService, UserService&gt;();
```

```
builder.Services.AddScoped&lt;IBarcodeService, BarcodeService&gt;();

// إضافة AutoMapper
builder.Services.AddAutoMapper(AppDomain.CurrentDomain.GetAssemblies());

// إضافة Controllers و Views
builder.Services.AddControllersWithViews();

var app = builder.Build();

// تهيئة قاعدة البيانات والأدوار
using (var scope = app.Services.CreateScope())
{
    var services = scope.ServiceProvider;
    var roleManager = services.GetRequiredService&lt;RoleManager&lt;IdentityRole&gt;&gt;()
    await SeedRoles(roleManager);
}

// Middleware
app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();

app.UseAuthentication();
app.UseAuthorization();

app.MapControllerRoute(
    name: "areas",
    pattern: "{area:exists}/{controller=Home}/{action=Index}/{id?}");

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();

// تهيئة الأدوار
async Task SeedRoles(RoleManager&lt;IdentityRole&gt; roleManager)
{
    string[] roles = { "Admin", "Manager", "Employee" };

    foreach (var role in roles)
    {
        if (!await roleManager.RoleExistsAsync(role))
        {
            await roleManager.CreateAsync(new IdentityRole(role));
        }
    }
}
```

الجزء التاسع: appsettings.json

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=.;Database=InventoryManagementDB;Integrated Security=true
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information"
    }
  }
}
```

الجزء العاشر: ERD الموحّد



```
│  Suppliers
│  ├─ Id (PK)
│  ├─ Name
│  ├─ Email
│  ├─ Phone
│  └─ Address
│        ↓ (1-N)
│  PurchaseOrders
│  ├─ Id (PK)
│  ├─ InvoiceNumber (UK)
│  ├─ OrderDate
│  ├─ Status
│  ├─ PaymentStatus
│  ├─ TotalAmount
│  ├─ TotalWithTax
│  ├─ SupplierId (FK)
│  └─ UserId (FK)
│        ↓ (1-N)
│  PurchaseOrderDetails
│  ├─ Id (PK)
│  ├─ Quantity
│  ├─ UnitPrice
│  ├─ Subtotal
│  ├─ ProductId (FK)
│  └─ PurchaseOrderId (FK)
│
│  StockMovements
│  ├─ Id (PK)
│  ├─ Type (Purchase/Sale/Adjustment)
│  ├─ Quantity
│  ├─ Date
│  ├─ ProductId (FK)
│  └─ UserId (FK)
│
└──────────────────────────────────────────────────────────────────────────┘
```

الجزء الحادي عشر: خدمة إدارة المستخدمين

```
public interface IUserService
{
    Task&lt;ApplicationUser&gt; GetUserByIdAsync(string id);
    Task&lt;List&lt;ApplicationUser&gt;&gt; GetAllUsersAsync();
    Task&lt;IdentityResult&gt; CreateUserAsync(ApplicationUser user, string password, stri
    Task&lt;IdentityResult&gt; UpdateUserAsync(ApplicationUser user);
    Task&lt;IdentityResult&gt; DeleteUserAsync(string id);
    Task&lt;IdentityResult&gt; ChangePasswordAsync(string userId, string oldPassword, stri
    Task&lt;bool&gt; IsUserActiveAsync(string userId);
}

public class UserService : IUserService
{
    private readonly UserManager&lt;ApplicationUser&gt; _userManager;
    private readonly RoleManager&lt;IdentityRole&gt; _roleManager;
    private readonly ILogger&lt;UserService&gt; _logger;

    public UserService(
        UserManager&lt;ApplicationUser&gt; userManager,
        RoleManager&lt;IdentityRole&gt; roleManager,
        ILogger&lt;UserService&gt; logger)
    {
        _userManager = userManager;
        _roleManager = roleManager;
        _logger = logger;
```

```
    }

    public async Task&lt;ApplicationUser&gt; GetUserByIdAsync(string id)
    {
        return await _userManager.FindByIdAsync(id);
    }

    public async Task&lt;List&lt;ApplicationUser&gt;&gt; GetAllUsersAsync()
    {
        return _userManager.Users.Where(u =&gt; u.IsActive).ToList();
    }

    public async Task&lt;IdentityResult&gt; CreateUserAsync(ApplicationUser user, string pa
    {
        try
        {
            var result = await _userManager.CreateAsync(user, password);

            if (result.Succeeded &amp;&amp; !string.IsNullOrEmpty(role))
            {
                await _userManager.AddToRoleAsync(user, role);
                _logger.LogInformation($"User {user.UserName} created with role {role}");
            }

            return result;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Error creating user");
            throw;
        }
    }

    public async Task&lt;IdentityResult&gt; UpdateUserAsync(ApplicationUser user)
    {
        return await _userManager.UpdateAsync(user);
    }

    public async Task&lt;IdentityResult&gt; DeleteUserAsync(string id)
    {
        var user = await GetUserByIdAsync(id);
        if (user == null)
            throw new KeyNotFoundException($"User not found");

        user.IsActive = false;
        return await UpdateUserAsync(user);
    }

    public async Task&lt;IdentityResult&gt; ChangePasswordAsync(string userId, string oldPa
    {
        var user = await GetUserByIdAsync(userId);
        return await _userManager.ChangePasswordAsync(user, oldPassword, newPassword);
    }

    public async Task&lt;bool&gt; IsUserActiveAsync(string userId)
    {
        var user = await GetUserByIdAsync(userId);
        return user?.IsActive ?? false;
    }
}
```

الجزء الثاني عشر: Migration الأول

```
# في Package Manager Console<a></a>
Add-Migration InitialCreate -Context InventoryDbContext -Project Inventory.Data
Update-Database -Context InventoryDbContext
```

ملخص التغييرات الرئيسية:

الميزة	الوصف
Database واحد فقط	كل شيء في نفس الـ InventoryDB
ApplicationUser مركزي	جميع العمليات تربط بـ UserId
بدون Employee/Department	تم حذفهم تماماً
Roles من Identity	Admin, Manager, Employee
IdentityDbContext	يرث من IdentityDbContext بدلاً من DbContext
أداء أفضل	لا Joins بين قاعدات بيانات
تطوير أسرع	كود أبسط وأقل تعقيداً