

أفضل مشروع نعمله مع بعض

مشروع إدارة النقل والشحن (Transportation Management System - TMS)

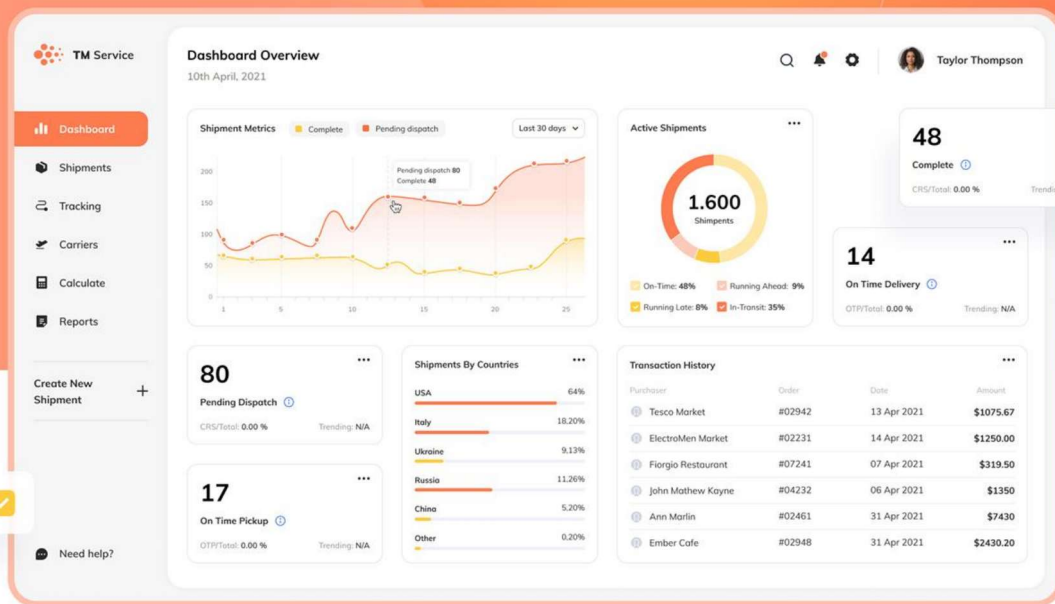
لأنه فيه Backend المشروع نفسه مناسب جدًا لتعلم

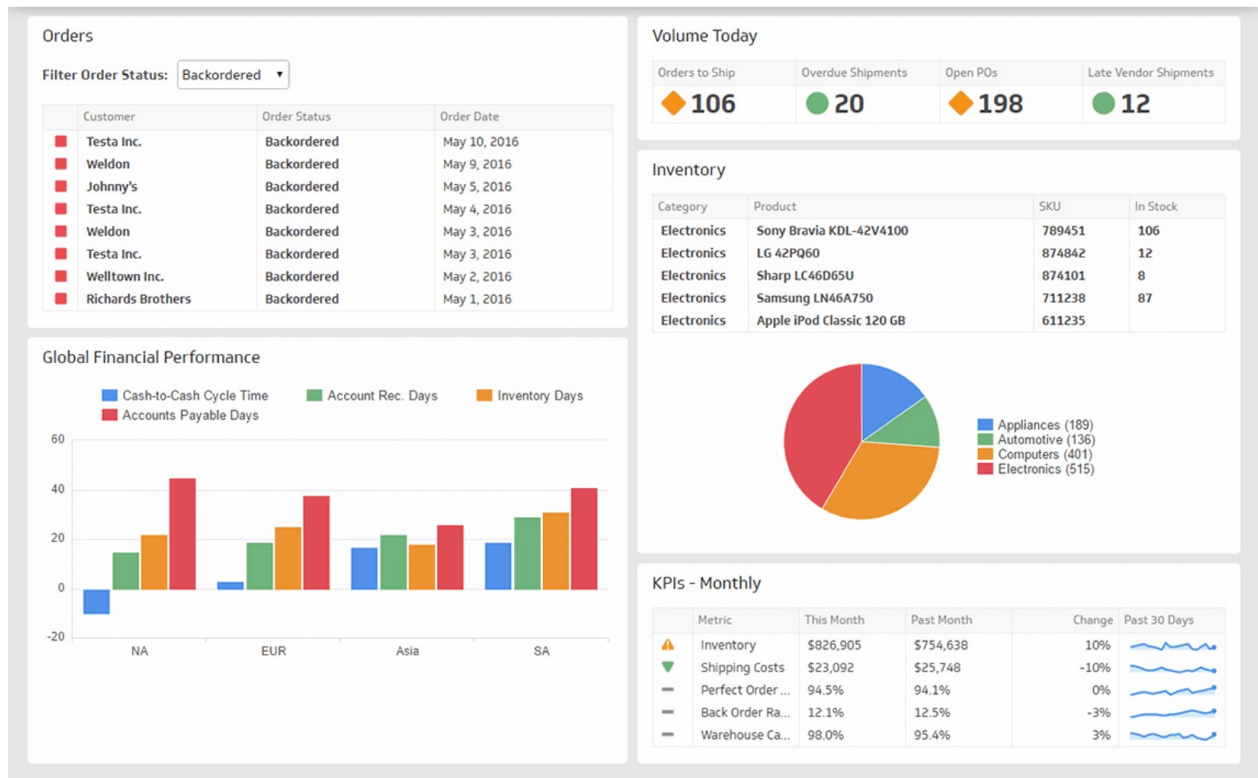
- Users + Roles
 - Vehicles
 - Drivers
 - Trips (الرحلات)
- Loads / Shipments
 - Tracking
 - Reports
- Dashboard





Transport Management System





4

أجزاء المشروع التي نتعلمها

Domain تصميم ال-1)

1. Vehicle
2. Driver
3. Trip
4. Shipment
5. FuelRecord
6. Maintenance

8.NET بـ API إنشاء 2)

- Minimal APIs أو Controllers (اختبارك)
- DTOs

- Mapping
- Validation

3) Database + EF Core

- Migrations
- Seeding Data
- Relations (One-to-Many / Many-to-Many)

4) Authentication

تسجيل دخول

JWT Token

Authorization (Roles + Policies)

5) CQRS

Commands

Queries

MediatR

6) Services

Trip assignment

Tracking updates

7) Dashboard API

Daily loads count

Driver performance

Vehicle status

فكرة المشروع

Transport & Finance Management System

يشمل:

- إدارة:
 - Companies / Customers
 - Vehicles
 - Drivers
 - Trips (رحلات)
 - Shipments (الشحنات)
 - الجزء المالي:
 - Invoices (فواتير)
 - Payments (مدفوعات)
 - Expenses (مصاريف تشغيل: وقود، صيانة)
 - حساب الربح / الخسارة لكل رحلة أو فترة

المعمارية التي هنتغل بيها (Clean Architecture)

فيها 4 طبقات رئيسية Solution هنتعمل

7. Domain

- a. Entities
- b. Value Objects
- c. Enums
- d. Domain Events (لاحقاً)

8. Application

- a. DTOs / ViewModels
- b. Interfaces (Repositories, Services)
- c. Use Cases (Commands / Queries – CQRS)
- d. Business Rules

9. Infrastructure

- a. EF Core DbContext
- b. Repositories (Repository + Unit Of Work)
- c. Implementations للتعامل مع SQL Server

10.API (Presentation)

- a. Controllers
- b. Endpoints
- c. Dependency Injection

اللي هنستخدمهم Design Patterns

خلال الشغل هنطبق

- Repository Pattern
 - Unit of Work
 - CQRS (Commands / Queries)
 - Specification Pattern (لفلترة البيانات)
- Strategy Pattern (مثلاً لحساب تكلفة الرحلة أو الضريبة بأكثر من طريقة)
 - Factory Pattern (لإنشاء أنواع مختلفة من الـ Payment)
- Domain Events (عشان نصدّر فاتورة Event مثلاً لما رحلة تخلص نطلق)

Junior: المرحلة القادمة هي اللي هتفرّقك عن

- CQRS
- MediatR
- Validation
- Authentication
- Caching
- Logging
- SOLID
- Domain Events
- Specifications Pattern
- DDD Concepts

فعلاً بإذن الله Senior وهنمشي فيهم واحدة واحدة لحد ما توصل 🙏🔥

JWT TOKEN وعاوز اضيف عليهم تص

(خريطة المشروع (من البداية للنهاية

✓ تم إنجازه

- **Domain Layer**
 - Entities (Vehicle, Driver, Trip, Invoice)
 - قواعد الدومين + العلاقات
- **Infrastructure Layer**
 - TransportDbContext
 - Repositories
 - Unit of Work
 - مثال عملي للتجربة

→ **CQRS المطلوب إضافته قبل** SOON

👉 أعلى Layer هذه الأشياء لازم ننهيها قبل ما نطلع لأي

■ **1** Aggregate لكل Repositories توحيد الـ

- IVehicleRepository
- IDriverRepository
- ITripRepository
- IInvoiceRepository

مختلفة لكل واحد Queries لأنك هتحتاج

■ **2** Configuration إعداد Fluent API كامل للعلاقات بـ

- One-to-One: Vehicle ↔ Driver ✓ بدأناه
- One-to-Many: Vehicle → Trips ✗ نكملة

- في الجداول Fields خصائص الـ
 - String طول
 - Required
 - Default values

مثال:

```
builder.Entity<Vehicle>(b =>
{
    b.Property(v => v.PlateNumber)
      .IsRequired()
      .HasMaxLength(50);
});
```

3 (اختياري) Seed Data

تضيف بيانات اختبار للمركبات والسائقين
APIs عشان نختبر

4 Transactions تفعيل الـ UoW العمليات الكبيرة داخل

لو عايز نعمل

- BeginTransaction
- Commit
- Rollback

5 Repositories تسجيل الـ DI في

في Program.cs:

```
services.AddScoped<IVehicleRepository, VehicleRepository>();
services.AddScoped<IDriverRepository, DriverRepository>();
```



```
services.AddScoped<ITripRepository, TripRepository>();  
services.AddScoped<IInvoiceRepository, InvoiceRepository>();
```

الملفات Namespaces تنظيم 6