Please use '!option handwritten true' to enable handwritten

Design Patterns in OllamaNet

Architectural Patterns
CQRS Pattern
Unit of Work Pattern
Microservices Architecture
Circuit Breaker Pattern
Event-Driven Architecture
Repository Pattern
API Gateway Pattern
Uses for commands
Uses for queries
Manages
Often employs
Uses
Form of
Implements
Uses
Uses
Form of
Core architectural pattern for entire OllamaNet platform
Used for resilient communication with external services and RabbitMQ
Used across all services for data access abstraction
Implemented with Ocelot in the API Gateway service
Data Patterns
DTO Pattern
ORM
Specification Pattern
Cache-Aside Pattern
Behavioral Patterns
Strategy Pattern
Template Method
Observer Pattern
Chain of Responsibility
Mediator Pattern
Creational Patterns
Singleton
Builder Pattern
Factory Method
Dependency Injection
Structural Patterns
Proxy Pattern
Facade Pattern
Decorator Pattern
Adapter Pattern
Used for service discovery via RabbitMQ pub/sub
Used throughout all .NET services via built-in DI container
Implementation Examples
Observer Pattern - RabbitMQ for service discovery - Event-based configuration updates - Message subscribers
Dependency Injection - .NET built-in DI container - Service registration - Lifetime management
Repository Pattern - Entity Framework repositories - Service-specific data access - Query encapsulation
Gateway Pattern - Ocelot API Gateway - Request routing - Cross-cutting concerns
Circuit Breaker - Polly integration - RabbitMQ resilience - HTTP client resilience
InferenceService (publisher) AdminService, ConversationService (subscribers)
All .NET services
All services with data access
Primary: API Gateway
ConversationService, AdminService

Key design patterns used throughout the OllamaNet platform, organized by category with implementation examples.

Each pattern addresses specific architectural concerns:
- Microservices: Service boundaries and independence
- Gateway: Centralized access and cross-cutting concerns
- Repository: Data access abstraction
- Circuit Breaker: Resilience and fault tolerance
- Observer: Event-driven communication
- Dependency Injection: Loose coupling and testability

Lines indicate relationships between patterns.
Notes indicate primary services implementing each pattern.