📝 Architect.yaml Configuration Guide

5 dakikada microservice mimarinizi tanımlayın!

🎯 Temel Yapı

```yaml
project_name: "my-project"
version: "1.0.0"

services:
  service-name:
    language: dotnet|java|nodejs
    database: postgresql|mongodb
    architecture: cqrs|event-driven|n-tier
    entities:
      EntityName:
        field1: type
        field2: type
    events:
      - EventName1
      - EventName2
```

🔧 Service Konfigürasyonu

Language Options

- **dotnet** → ASP.NET Core 9 + Entity Framework
- **java** → Spring Boot 3.2 + JPA + Maven
- **nodejs** → Express.js + Mongoose/Sequelize

Database Options

- **postgresql** → Relational DB (önerilen)
- **mongodb** → NoSQL (Node.js ile ideal)

Architecture Options

- **cqrs** → Command/Query separation + MediatR
- **event-driven** → Event publishing + Kafka
- **n-tier** → Classic layered architecture

📊 Entity Field Types

Type	Description	Example
`string`	Text field	`name: string`
`int`	Integer	`age: int`
`float`	Decimal	`price: float`
`bool`	Boolean	`isActive: bool`
`datetime`	Timestamp	`createdAt: datetime`
`hashed-string`	Auto-hashed password	`password: hashed-string`

📋 Hızlı Template'ler

🛒 E-commerce Platform

yaml

```yaml
project_name: "ecommerce"

services:
  user:
    language: dotnet
    database: postgresql
    architecture: cqrs
    entities:
      User:
        username: string
        email: string
        password: hashed-string
        firstName: string
        isActive: bool
    events:
      - UserRegistered
      - UserUpdated

  product:
    language: java
    database: postgresql
    architecture: cqrs
    entities:
      Product:
        name: string
        price: float
        stock: int
        category: string
      Category:
        name: string
        description: string
    events:
      - ProductCreated
      - StockUpdated

  order:
    language: nodejs
    database: mongodb
    architecture: event-driven
    entities:
      Order:
        userId: string
        status: string
        totalAmount: float
        orderDate: datetime
    events:
```

```
    - OrderCreated
    - OrderCompleted
```

📝 Blog Platform

```yaml
project_name: "blog-platform"

services:
  user:
    language: dotnet
    database: postgresql
    architecture: cqrs
    entities:
      User:
        username: string
        email: string
        password: hashed-string
        role: string
    events:
      - UserRegistered

  blog:
    language: nodejs
    database: mongodb
    architecture: event-driven
    entities:
      Post:
        title: string
        content: string
        authorId: string
        publishedAt: datetime
        isPublished: bool
      Comment:
        postId: string
        userId: string
        content: string
    events:
      - PostPublished
      - CommentAdded
```

🏦 Banking System

```yaml
```

```yaml
project_name: "banking-system"

services:
  account:
    language: java
    database: postgresql
    architecture: cqrs
    entities:
      Account:
        accountNumber: string
        userId: string
        balance: float
        accountType: string
        isActive: bool
    events:
      - AccountCreated
      - BalanceUpdated

  transaction:
    language: dotnet
    database: postgresql
    architecture: event-driven
    entities:
      Transaction:
        fromAccount: string
        toAccount: string
        amount: float
        transactionType: string
        timestamp: datetime
        status: string
    events:
      - TransactionCreated
      - TransactionCompleted
```

✅ Best Practices

🎯 Naming Conventions

- **Services**: kebab-case (`user-management`, `order-service`)
- **Entities**: PascalCase (`User`, `OrderItem`)
- **Fields**: camelCase (`firstName`, `createdAt`)
- **Events**: PascalCase (`UserRegistered`, `OrderCreated`)

🏗️ Architecture Selection

- **CQRS**: Karmaşık business logic + read/write ayrımı
- **Event-driven**: Loosely coupled + async operations
- **N-tier**: Basit CRUD operasyonları

🗄️ Database Selection

- **PostgreSQL**: Relational data + ACID transactions
- **MongoDB**: Document-based + flexible schema

🔐 Security Fields

```yaml
entities:
  User:
    username: string
    email: string
    password: hashed-string  # Otomatik bcrypt hash
    salt: string             # Otomatik salt
    lastLogin: datetime
```

🚦 Validation Rules

✅ Required Fields

- `project_name` (string)
- `services` (object)
- `services.*.language` (dotnet|java|nodejs)
- `services.*.entities` (object)

⚠️ Optional Fields

- `database` (default: postgresql)
- `architecture` (default: cqrs)
- `events` (default: [])
- `version` (default: 1.0.0)

🎉 Quick Start

1. **Dosya oluştur**: `architect.yaml`
2. **Template seç**: Yukarıdaki örneklerden birini kopyala
3. **Özelleştir**: Service, entity ve field'ları düzenle
4. **Çalıştır**: `microfactory`

5. **Deploy**: `docker-compose up -d`

💡 Pro Tips

- **Entity başına 5-10 field** ideal
- **Password field'ları** mutlaka `hashed-string` kullanın
- **Event isimleri** Past Tense kullanın (`Created`, `Updated`)
- **Complex relationships** için separate entity oluşturun
- **Service başına 2-5 entity** ideal mikroservis boyutu

⏰ **5 dakikada microservice mimariniz hazır!** 🚀