📁 Structure AgroDeep Platform - Microservices Architecture

```
agrodeep-platform/
├── 📚 docs/
│   ├── architecture/
│   │   ├── ADR-001-microservices-migration.md
│   │   ├── service-contracts.yaml
│   │   └── data-flow-diagrams/
│   ├── api/
│   │   └── openapi/
│   │       ├── auth-service.yaml
│   │       ├── product-service.yaml
│   │       └── order-service.yaml
│   └── runbooks/
│       ├── deployment.md
│       └── incident-response.md
│
├── 🌐 gateway/                # Kong API Gateway
│   ├── kong.yml
│   ├── plugins/
│   │   ├── rate-limiting.lua
│   │   ├── jwt-auth.lua
│   │   └── cors.lua
│   └── docker-compose.yml
│
├── 🔐 service-mesh/           # Istio Configuration
│   ├── istio-config.yaml
│   ├── virtual-services/
│   │   ├── auth-vs.yaml
│   │   ├── product-vs.yaml
│   │   └── order-vs.yaml
│   ├── destination-rules/
│   └── gateway.yaml
│
├── 🎯 services/
│   │
│   ├── 🔑 auth-service/         # Authentication & Authorization
│   │   ├── src/
│   │   │   ├── domain/
│   │   │   │   ├── entities/
│   │   │   │   │   ├── User.ts
│   │   │   │   │   └── Role.ts
│   │   │   │   ├── repositories/
│   │   │   │   │   └── IUserRepository.ts
│   │   │   │   └── services/
│   │   │   │       ├── AuthService.ts
│   │   │   │       └── TokenService.ts
│   │   │   │
│   │   │   ├── application/
│   │   │   │   ├── usecases/
│   │   │   │   │   ├── LoginUseCase.ts
│   │   │   │   │   ├── RegisterUseCase.ts
│   │   │   │   │   └── RefreshTokenUseCase.ts
│   │   │   │   └── dto/
│   │   │   ├── infrastructure/
│   │   │   │   ├── database/
│   │   │   │   │   ├── PostgresUserRepository.ts
│   │   │   │   │   └── migrations/
│   │   │   │   ├── messaging/
│   │   │   │   │   └── RabbitMQPublisher.ts
│   │   │   │   └── cache/
│   │   │   │       └── RedisCache.ts
│   │   │   └── interfaces/
│   │   │       ├── http/
│   │   │       │   ├── routes/
│   │   │       │   │   └── auth.routes.ts
│   │   │       │   ├── controllers/
│   │   │       │   │   └── AuthController.ts
│   │   │       │   └── middlewares/
│   │   │       └── grpc/
│   │   │           └── auth.proto
│   │   ├── tests/
│   │   │   ├── unit/
│   │   │   ├── integration/
│   │   │   └── e2e/
│   │   ├── Dockerfile
│   │   ├── package.json
│   │   └── tsconfig.json
│   │
│   ├── 📦 product-service/       # Product Catalog & Search
│   │   ├── src/
│   │   │   ├── domain/
│   │   │   │   ├── entities/
│   │   │   │   │   ├── Product.ts
│   │   │   │   │   ├── Category.ts
│   │   │   │   │   └── Tag.ts
│   │   │   │   ├── repositories/
│   │   │   │   │   └── IProductRepository.ts
│   │   │   │   └── services/
│   │   │   │       ├── ProductService.ts
│   │   │   │       └── SearchService.ts
│   │   │   ├── application/
│   │   │   │   ├── usecases/
│   │   │   │   │   ├── CreateProductUseCase.ts
│   │   │   │   │   ├── SearchProductsUseCase.ts
│   │   │   │   │   └── GetProductByIdUseCase.ts
│   │   │   │   │
│   │   │   │   └── dto/
│   │   │   ├── infrastructure/
│   │   │   │   ├── database/
│   │   │   │   │   ├── PostgresProductRepository.ts
│   │   │   │   │   └── ElasticsearchProductRepository.ts
│   │   │   │   └── storage/
│   │   │   │       └── MinIOStorage.ts
│   │   │   └── interfaces/
│   │   │       ├── http/
│   │   │       │   └── routes/
│   │   │       │       ├── products.routes.ts
│   │   │       │       └── categories.routes.ts
│   │   │       └── events/
│   │   │           └── ProductEventHandler.ts
│   │   ├── tests/
│   │   ├── Dockerfile
│   │   └── package.json
│   │
│   ├── 🛒 order-service/          # Order Management
│   │   ├── src/
│   │   │   ├── domain/
│   │   │   │   ├── entities/
│   │   │   │   │   ├── Order.ts
│   │   │   │   │   ├── OrderItem.ts
│   │   │   │   │   └── OrderStatus.ts
│   │   │   │   ├── aggregates/
│   │   │   │   │   └── OrderAggregate.ts
│   │   │   │   └── services/
│   │   │   │       └── OrderService.ts
│   │   │   ├── application/
│   │   │   │   ├── usecases/
│   │   │   │   │   ├── CreateOrderUseCase.ts
│   │   │   │   │   ├── UpdateOrderStatusUseCase.ts
│   │   │   │   │   └── GetOrderHistoryUseCase.ts
│   │   │   │   ├── sagas/
│   │   │   │   │   └── OrderSaga.ts
│   │   │   │   └── dto/
│   │   │   ├── infrastructure/
│   │   │   │   ├── database/
│   │   │   │   │   └── PostgresOrderRepository.ts
│   │   │   │   ├── messaging/
│   │   │   │   │   ├── OrderEventPublisher.ts
│   │   │   │   │   └── PaymentEventConsumer.ts
│   │   │   │   └── cache/
│   │   │   └── interfaces/
│   │   │       ├── http/
│   │   │       └── events/
│   │   ├── tests/
│   │   ├── Dockerfile
│   │   └── package.json
│   │
│   ├── 💳 payment-service/       # Payment Integration
│   │   ├── src/
│   │   │   ├── domain/
│   │   │   │   ├── entities/
│   │   │   │   │   ├── Payment.ts
│   │   │   │   │   └── PaymentMethod.ts
│   │   │   │   └── services/
│   │   │   │       ├── StripePaymentService.ts
│   │   │   │       └── PayPalPaymentService.ts
│   │   │   ├── application/
│   │   │   │   ├── usecases/
│   │   │   │   │   ├── ProcessPaymentUseCase.ts
│   │   │   │   │   └── RefundPaymentUseCase.ts
│   │   │   │   └── dto/
│   │   │   ├── infrastructure/
│   │   │   │   ├── database/
│   │   │   │   ├── external/
│   │   │   │   │   ├── StripeClient.ts
│   │   │   │   │   └── PayPalClient.ts
│   │   │   │   └── messaging/
│   │   │   └── interfaces/
│   │   ├── tests/
│   │   ├── Dockerfile
│   │   └── package.json
│   │
│   ├── 🚚 delivery-service/      # GPS Tracking & Route Optimization
│   │   ├── src/
│   │   │   ├── domain/
│   │   │   │   ├── entities/
│   │   │   │   │   ├── Delivery.ts
│   │   │   │   │   └── Vehicle.ts
│   │   │   │   └── services/
│   │   │   │       ├── TrackingService.ts
│   │   │   │       └── RouteOptimizationService.ts
│   │   │   ├── application/
│   │   │   │   ├── usecases/
│   │   │   │   │   ├── CreateDeliveryUseCase.ts
│   │   │   │   │   └── UpdateLocationUseCase.ts
│   │   │   │   └── dto/
│   │   │   ├── infrastructure/
│   │   │   │   ├── database/
│   │   │   │   │   ├── PostgresDeliveryRepository.ts
│   │   │   │   │   └── QdrantVectorDB.ts
│   │   │   │   ├── messaging/
│   │   │   │   └── websocket/
│   │   │   │       └── LocationBroadcaster.ts
│   │   │   └── interfaces/
│   │   ├── tests/
│   │   ├── Dockerfile
│   │   └── package.json
│   │
│   ├── 🤖 ai-service/           # ML Recommendations & NLP
│   │   ├── src/
│   │   │   ├── domain/
│   │   │   │   ├── entities/
│   │   │   │   │   └── Recommendation.ts
│   │   │   │   └── services/
│   │   │   │       ├── RecommendationEngine.ts
│   │   │   │       └── SentimentAnalysisService.ts
│   │   │   ├── application/
│   │   │   │   ├── usecases/
│   │   │   │   │   ├── GetRecommendationsUseCase.ts
│   │   │   │   │   └── AnalyzeReviewsUseCase.ts
│   │   │   │   └── dto/
│   │   │   ├── infrastructure/
│   │   │   │   ├── database/
│   │   │   │   │   └── QdrantVectorRepository.ts
│   │   │   │   ├── ml/
│   │   │   │   │   ├── models/
│   │   │   │   │   │   ├── collaborative_filtering.py
│   │   │   │   │   │   └── bert_sentiment.py
│   │   │   │   │   └── training/
│   │   │   │   └── cache/
│   │   │   └── interfaces/
│   │   ├── models/              # ML Models
│   │   │   └── checkpoints/
│   │   ├── tests/
│   │   ├── Dockerfile
│   │   └── requirements.txt
│   │
│   ├── 📊 analytics-service/    # ClickHouse Analytics
│   │   ├── src/
│   │   │   ├── domain/
│   │   │   │   ├── entities/
│   │   │   │   │   └── Metric.ts
│   │   │   │   └── services/
│   │   │   │       └── AnalyticsService.ts
│   │   │   ├── application/
│   │   │   │   ├── usecases/
│   │   │   │   │   ├── GenerateReportUseCase.ts
│   │   │   │   │   └── GetKPIsUseCase.ts
│   │   │   │   └── dto/
│   │   │   ├── infrastructure/
│   │   │   │   ├── database/
│   │   │   │   │   └── ClickHouseRepository.ts
│   │   │   │   ├── streaming/
│   │   │   │   │   └── KafkaConsumer.ts
│   │   │   │   └── reporting/
│   │   │   │       └── PDFGenerator.ts
│   │   │   └── interfaces/
│   │   ├── tests/
│   │   ├── Dockerfile
│   │   └── package.json
│   │
│   ├── ⛓️ blockchain-service/     # Hyperledger Fabric Traceability
│   │   ├── chaincode/
│   │   │   ├── product-trace/
│   │   │   │   ├── lib/
│   │   │   │   │   └── ProductTrace.ts
│   │   │   │   ├── index.ts
│   │   │   │   └── package.json
│   │   │   └── transaction-history/
│   │   ├── src/
│   │   │   ├── domain/
│   │   │   │   └── services/
│   │   │   │       └── BlockchainService.ts
│   │   │   ├── application/
│   │   │   │   └── usecases/
│   │   │   │       ├── RecordTransactionUseCase.ts
│   │   │   │       └── VerifyTraceabilityUseCase.ts
│   │   │   ├── infrastructure/
│   │   │   │   ├── fabric/
│   │   │   │   │   ├── FabricClient.ts
│   │   │   │   │   └── network-config.yaml
│   │   │   │   └── database/
│   │   │   └── interfaces/
│   │   ├── network/               # Fabric Network Config
│   │   │   ├── docker-compose.yaml
│   │   │   ├── crypto-config/
│   │   │   └── channel-artifacts/
│   │   ├── tests/
│   │   ├── Dockerfile
│   │   └── package.json
│   │
│   └── 🔔 notification-service/   # Push, Email, SMS
│       ├── src/
│       │   ├── domain/
│       │   │   ├── entities/
│       │   │   │   └── Notification.ts
│       │   │   └── services/
│       │   │       ├── PushService.ts
│       │   │       ├── EmailService.ts
│       │   │       └── SMSService.ts
│       │   ├── application/
│       │   │   ├── usecases/
│       │   │   │   ├── SendNotificationUseCase.ts
│       │   │   │   └── SendBulkNotificationsUseCase.ts
│       │   │   └── dto/
│       │   ├── infrastructure/
│       │   │   ├── database/
│       │   │   │   └── RedisNotificationRepository.ts
│       │   │   ├── external/
│       │   │   │   ├── FCMClient.ts
│       │   │   │   ├── SendGridClient.ts
│       │   │   │   └── TwilioClient.ts
│       │   │   └── messaging/
│       │   │       └── NotificationEventConsumer.ts
│       │   └── interfaces/
│       ├── tests/
│       ├── Dockerfile
│       └── package.json
│
├── 🌐 clients/
│   │
│   ├── web-app/                # Next.js PWA
│   │   ├── src/
│   │   │   ├── app/
│   │   │   │   ├── (auth)/
│   │   │   │   ├── (admin)/
│   │   │   │   ├── (market)/
│   │   │   │   ├── (customer)/
│   │   │   │   └── landing/
│   │   │   ├── components/
│   │   │   ├── hooks/
│   │   │   ├── stores/
│   │   │   ├── lib/
│   │   │   │   └── api-client.ts   # SDK for microservices
│   │   │   └── types/
│   │   ├── public/
│   │   ├── next.config.js
│   │   └── package.json
│   │
│   ├── mobile-app/             # Flutter App
│   │   ├── lib/
│   │   │   ├── features/
│   │   │   ├── core/
│   │   │   └── shared/
│   │   ├── pubspec.yaml
│   │   └── README.md
│   │
│   └── admin-dashboard/          # React Admin Panel
│       ├── src/
│       │   ├── pages/
│       │   ├── components/
│       │   └── services/
│       ├── package.json
│       └── README.md
│
├── 🗄️ infrastructure/
│   │
│   ├── kubernetes/
│   │   ├── namespaces/
│   │   ├── deployments/
│   │   │   ├── auth-deployment.yaml
│   │   │   ├── product-deployment.yaml
│   │   │   └── order-deployment.yaml
│   │   ├── services/
│   │   ├── ingress/
│   │   ├── configmaps/
│   │   ├── secrets/
│   │   └── helm/
│   │       ├── agrodeep/
│   │       │   ├── Chart.yaml
│   │       │   ├── values.yaml
│   │       │   └── templates/
│   │       └── databases/
│   │
│   ├── terraform/
│   │   ├── modules/
│   │   │   ├── vpc/
│   │   │   ├── eks/
│   │   │   ├── rds/
│   │   │   └── elasticache/
│   │   ├── environments/
│   │   │   ├── dev/
│   │   │   ├── staging/
│   │   │   └── production/
│   │   └── main.tf
│   │
│   ├── databases/
│   │   ├── postgres/
│   │   │   ├── docker-compose.yml
│   │   │   └── init-scripts/
│   │   ├── redis/
│   │   │   └── redis.conf
│   │   ├── clickhouse/
│   │   │   └── config.xml
│   │   │
│   │   └── qdrant/
│   │       └── config.yaml
│   │
│   └── monitoring/
│       ├── prometheus/
│       │   └── prometheus.yml
│       ├── grafana/
│       │   ├── dashboards/
│       │   └── datasources/
│       ├── loki/
│       │   └── loki-config.yaml
│       └── jaeger/
│           └── jaeger-config.yaml
│
├── 🔧 shared/
│   ├── proto/                  # gRPC Protobuf
│   │   ├── auth.proto
│   │   ├── product.proto
│   │   └── order.proto
│   │
│   ├── events/                 # Event Schemas
│   │   ├── UserCreatedEvent.ts
│   │   ├── OrderPlacedEvent.ts
│   │   └── PaymentProcessedEvent.ts
│   │
│   ├── types/                  # Shared Types
│   │   └── common.ts
│   │
│   └── utils/
│       ├── logger.ts
│       └── error-handler.ts
│
├── 🧪 tests/
│   ├── integration/
│   ├── e2e/
│   └── performance/
│
├── 📜 scripts/
│   ├── setup-dev-env.sh
│   ├── deploy.sh
│   ├── seed-databases.sh
│   └── backup.sh
│
├── docker-compose.yml          # Local Development
├── docker-compose.prod.yml     # Production
├── .env.example
├── .gitignore
├── README.md
```

└── package.json
~~~

📊 **Database Distribution**

Service → Database Mapping

Service	Primary DB	Secondary	Cache
Auth Service	PostgreSQL	-	Redis
Product Service	PostgreSQL	Elasticsearch	Redis
Order Service	PostgreSQL	-	Redis
Payment Service	PostgreSQL	-	Redis
Delivery Service	PostgreSQL	QdrantDB	Redis
AI Service	QdrantDB	ClickHouse	Redis
Analytics Service	ClickHouse	-	Redis
Blockchain Service	Hyperledger	PostgreSQL	-
Notification Service	Redis	PostgreSQL	-

🔄 **Event-Driven Communication**

Event Types

```typescript
// User Events
UserCreatedEvent
UserUpdatedEvent
UserDeletedEvent

// Product Events
ProductCreatedEvent
ProductUpdatedEvent
ProductDeletedEvent
ProductViewedEvent

// Order Events
OrderPlacedEvent
OrderConfirmedEvent
OrderShippedEvent
OrderDeliveredEvent
OrderCancelledEvent

// Payment Events
PaymentInitiatedEvent
~~~

PaymentProcessedEvent
PaymentFailedEvent
PaymentRefundedEvent

// Delivery Events
DeliveryAssignedEvent
DeliveryStartedEvent
LocationUpdatedEvent
DeliveryCompletedEvent
```

🔐 **Security Architecture**

Authentication Flow

```
Client → Kong Gateway → JWT Validation → Service Mesh (mTLS) → Microservice
```

### **Service-to-Service Communication**

```
Service A → Istio Sidecar → mTLS → Istio Sidecar → Service B
~~~

---

## 📈 **Scalability Patterns**

### **Horizontal Scaling Strategy**

```yaml
# Kubernetes HPA Configuration
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: product-service-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: product-service
  minReplicas: 2
  maxReplicas: 10
  metrics:
  - type: Resource
~~~

```
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 70
```

## 🎯 **Migration Strategy**

### **Phase 1: Infrastructure Setup** (Week 1-2)
- Setup Kubernetes cluster
- Deploy Kong Gateway
- Configure Istio Service Mesh
- Setup databases (PostgreSQL, Redis, ClickHouse, QdrantDB)

### **Phase 2: Core Services** (Week 3-4)
- Migrate Auth Service
- Migrate Product Service
- Migrate Order Service

### **Phase 3: Advanced Services** (Week 5-6)
- Implement Payment Service
- Implement Delivery Service
- Implement AI Service

### **Phase 4: Analytics & Blockchain** (Week 7-8)
- Setup Analytics Service
- Implement Blockchain Service
- Implement Notification Service

### **Phase 5: Testing & Optimization** (Week 9-10)
- Integration testing
- Performance testing
- Security audit
- Production deployment

## 📊 **Monitoring Stack**

```
Logs: Loki → Grafana
Metrics: Prometheus → Grafana
Traces: Jaeger
APM: OpenTelemetry
Alerts: Alertmanager → Slack/PagerDuty
```

```
```