Архитектурные диаграммы LMS ЦУМ

📊 Диаграммы системы

1. Общая архитектура (C4 Context)

```
graph TB
    User[Сотрудник ЦУМ]
    Admin[Администратор]

    subgraph "LMS System"
        iOS[iOS App]
        Backend[Backend Services]
        DB[(Database)]
    end

    subgraph "External Systems"
        AD[Microsoft AD]
        Mail[Email Service]
        Storage[File Storage]
    end

    User --> iOS
    Admin --> iOS
    iOS --> Backend
    Backend --> DB
    Backend --> AD
    Backend --> Mail
    Backend --> Storage
```

2. Микросервисная архитектура

```
graph LR
    subgraph "Client Layer"
        iOS[iOS App]
    end

    subgraph "API Gateway"
        Kong[Kong Gateway]
    end

    subgraph "Service Layer"
        Auth[Auth Service]
        User[User Service]
        Course[Course Service]
        Comp[Competency Service]
        Notif[Notification Service]
        Org[OrgStructure Service]
    end

    subgraph "Data Layer"
        PG[(PostgreSQL)]
        Redis[(Redis Cache)]
        RabbitMQ[RabbitMQ]
    end

    iOS --> Kong
    Kong --> Auth
    Kong --> User
    Kong --> Course
    Kong --> Comp
    Kong --> Notif
    Kong --> Org

    Auth --> PG
    User --> PG
    Course --> PG
    Comp --> PG

    Auth --> Redis
    Course --> Redis

    Notif --> RabbitMQ
```

3. iOS Clean Architecture

```
graph TD
    subgraph "Presentation Layer"
        View[SwiftUI Views]
        VM[ViewModels]
        Coord[Coordinators]
    end

    subgraph "Domain Layer"
        UC[Use Cases]
        Entity[Entities]
        Repo[Repository Protocols]
    end

    subgraph "Data Layer"
        Impl[Repository Implementations]
        API[API Client]
        Cache[Local Cache]
    end

    View --> VM
    VM --> UC
    UC --> Repo
    Repo --> Impl
    Impl --> API
    Impl --> Cache
    Coord --> View
```

4. Поток аутентификации

```
sequenceDiagram
    participant User
    participant iOS
    participant Gateway
    participant AuthService
    participant AD
    participant DB

    User->>iOS: Ввод логина/пароля
    iOS->>Gateway: POST /auth/login
    Gateway->>AuthService: Validate credentials
    AuthService->>AD: Check AD credentials
    AD-->>AuthService: User info
    AuthService->>DB: Create/Update user
    AuthService->>AuthService: Generate JWT
    AuthService-->>Gateway: JWT token
    Gateway-->>iOS: Auth response
    iOS->>iOS: Save token
    iOS-->>User: Показать главный экран
```

5. Структура базы данных

```
erDiagram
    User ||--o{ UserRole : has
    User ||--o{ Enrollment : has
    User ||--o{ CompetencyLevel : has

    Course ||--o{ Module : contains
    Module ||--o{ Lesson : contains
    Course ||--o{ CourseCompetency : requires

    Competency ||--o{ CourseCompetency : used_in
    Competency ||--o{ CompetencyLevel : measured_by

    User {
        uuid id PK
        string email
        string firstName
        string lastName
        string position
        datetime createdAt
    }
    Course {
        uuid id PK
        string title
        text description
        string status
        datetime publishedAt
    }

    Enrollment {
        uuid id PK
        uuid userId FK
        uuid courseId FK
        float progress
        datetime startedAt
        datetime completedAt
    }
```

6. Процесс развертывания

```
graph LR
    subgraph "Development"
        Dev[Developer]
        Local[Local Env]
    end

    subgraph "CI/CD"
        GH[GitHub]
        Actions[GitHub Actions]
        Tests[Automated Tests]
    end

    subgraph "Staging"
        Stage[Staging Server]
        TestFlight[TestFlight]
    end

    subgraph "Production"
        Railway[Railway.app]
        AppStore[App Store]
    end

    Dev --> Local
    Local --> GH
    GH --> Actions
    Actions --> Tests
    Tests --> Stage
    Tests --> TestFlight
    Stage --> Railway
    TestFlight --> AppStore
```

7. Модульная структура iOS

```
graph TD
    subgraph "Feature Modules"
        Auth[Auth Module]
        Feed[Feed Module]
        Course[Courses Module]
        Comp[Competencies]
        Org[OrgStructure]
        CMI5[CMI5 Content]
        SCORM[SCORM Content]
    end

    subgraph "Core Services"
        Logger[ComprehensiveLogger]
        Network[NetworkService]
        Storage[StorageService]
        Analytics[AnalyticsService]
    end

    subgraph "Common"
        UI[UI Components]
        Ext[Extensions]
        Utils[Utilities]
    end

    Auth --> Logger
    Feed --> Logger
    Course --> Network
    CMI5 --> Storage

    Auth --> UI
    Feed --> UI
    Course --> UI
```

8. Жизненный цикл курса

```
stateDiagram-v2
    [*] --> Draft: Создан
    Draft --> Review: Отправить на проверку
    Review --> Draft: Вернуть на доработку
    Review --> Published: Утвердить
    Published --> Active: Активировать
    Active --> Archived: Архивировать
    Archived --> Active: Восстановить
    Active --> Updated: Обновить
    Updated --> Active: Применить
```

9. Процесс тестирования (TDD)

```
graph LR
    subgraph "TDD Cycle"
        Red[Write Test
RED]
        Green[Write Code
GREEN]
        Refactor[Refactor
REFACTOR]
        Log[Add Logging
LOG]
    end

    Red --> Green
    Green --> Refactor
    Refactor --> Log
    Log --> Red

    subgraph "Test Types"
        Unit[Unit Tests
90%+]
        Integration[Integration Tests
80%+]
        UI[UI Tests
70%+]
        E2E[E2E Tests
Critical paths]
    end
```

10. Система логирования

```
graph TD
    subgraph "Log Sources"
        UI[UI Events]
        Net[Network Calls]
        Data[Data Changes]
        Nav[Navigation]
        Error[Errors]
    end

    subgraph "ComprehensiveLogger"
        Logger[Logger Core]
        Queue[Log Queue]
        Upload[Log Uploader]
    end

    subgraph "Log Storage"
        Local[Local SQLite]
        Server[Log Server]
        Cloud[Cloud Storage]
    end
    UI --> Logger
    Net --> Logger
    Data --> Logger
    Nav --> Logger
    Error --> Logger

    Logger --> Queue
    Queue --> Local
    Queue --> Upload
    Upload --> Server
    Server --> Cloud
```

📈 Метрики и мониторинг

Dashboard структура

```
┌─────────────────────────────────────────────────────┐
│                  LMS Dashboard                        │
├─────────────────┬───────────────┬───────────────────┤
│ Active Users    │ API Latency   │ Error Rate        │
│ 1,234           │ 45ms avg      │ 0.02%             │
├─────────────────┼───────────────┼───────────────────┤
│ Course Progress │ Test Coverage │ Build Status      │
│ 67% avg         │ 88.5%         │ ✅ Passing        │
├─────────────────┴───────────────┴───────────────────┤
│              Performance Graphs                       │
│  [CPU Usage Graph]     [Memory Graph]               │
│  [Network I/O Graph]   [Disk Usage Graph]           │
└─────────────────────────────────────────────────────┘
```

🔄 Процесс интеграции

Feature Registry Flow

```
graph TD
    A[Новый модуль] --> B{Регистрация в
FeatureRegistry}
    B --> C[Добавить enum case]
    B --> D[Определить иконку]
    B --> E[Создать view]
    B --> F[Добавить описание]

    C --> G[Включить feature flag]
    D --> G
    E --> G
    F --> G

    G --> H[Интеграционный тест]
    H --> I{Тест прошел?}
    I -->|Да| J[Deploy to TestFlight]
    I -->|Нет| K[Исправить]
    K --> H
```

Эти диаграммы помогут новым участникам быстрее понять архитектуру системы и основные процессы разработки.