# Block 1 – Project Setup Technical Specification v1

Настоящий документ описывает архитектуру и реализацию первого блока программы проекта «Мозаика». Блок 1 представляет собой создание и настройку базовой архитектуры приложения, включая микросервисную структуру, окружение Docker, подключение MongoDB и организацию тестовой среды.

## 1. Цели блока

Основная цель Блока 1 – создание архитектуры проекта и среды для дальнейшей разработки всех модулей. Блок обеспечивает:  
• Подъём микросервисов в контейнерах через Docker Compose.  
• Возможность локального тестирования каждого микросервиса.  
• Центральное хранилище данных (MongoDB).  
• Единый формат обмена данными между сервисами (JSON).  
• Базовые точки здоровья (/health) и тестовые эндпоинты (/echo).  
• Структуру папок для тестирования и ведения логов.

## 2. Архитектура проекта

Архитектура построена на микросервисной модели. Каждый сервис является отдельным контейнером с изолированной логикой и API. Взаимодействие осуществляется по HTTP (REST) с передачей данных в JSON-формате.

### Структура проекта

/project-root/  
├── docker-compose.yml  
├── services/  
│ ├── image-loader/  
│ ├── scaler/  
│ ├── color-cluster/  
│ ├── cell-mapper/  
│ └── cnc-exporter/  
├── tests/  
│ ├── image-loader/  
│ ├── scaler/  
│ ├── color-cluster/  
│ ├── cell-mapper/  
│ └── cnc-exporter/  
└── README.md

## 3. Docker Compose

Docker Compose-файл описывает все контейнеры и их параметры. Каждый микросервис имеет собственный порт и набор переменных окружения. MongoDB используется как общий сервис для хранения данных.  
  
Пример:  
- image-loader (порт 3001)  
- scaler (порт 3002)  
- color-cluster (порт 3003)  
- cell-mapper (порт 3004)  
- cnc-exporter (порт 3005)  
- mongo (порт 27017, volume mongo\_data)  
  
Сервисы автоматически поднимаются командой:  
docker compose up --build

## 4. Логика микросервисов

### image-loader

Загрузка изображений, конвертация в base64 и подготовка к дальнейшей обработке.

### scaler

Масштабирование изображения под размеры полотна мозаики.

### color-cluster

Кластеризация по цветам, выделение цветовых пятен.

### cell-mapper

Построение сетки ячеек, расчёт линий разреза и мест крепления.

### cnc-exporter

Формирование CNC-данных (DXF/JSON → G-code).

## 5. Формат взаимодействия между микросервисами

Передача данных между микросервисами осуществляется в формате JSON. Общая структура сообщения:  
  
{  
 "input": "JPEG",  
 "output": "JSON/DXF",  
 "next\_service": "color-cluster"  
}  
  
Каждый сервис принимает данные через POST-запрос, обрабатывает их и передаёт результат следующему модулю.

## 6. Тестирование

Для всех микросервисов создана единая структура каталога /tests, включающая подпапки:  
- INPUTS.md — описание входных данных;  
- LOG.md — записи логов и параметров тестов;  
- RESULTS.md — результаты тестов (JSON, скриншоты и т.п.).  
  
Каждый сервис имеет эндпоинт /health для проверки работоспособности и /echo для быстрой диагностики передачи данных.

## 7. Настройки окружения

Проект поддерживает два режима:  
• Локальный (на ноутбуке разработчика).  
• Серверный (на хостинге fornex.com).  
  
Подключение MongoDB задаётся через переменную MONGO\_URL (например, mongodb://mongo:27017/mosaic).

## 8. Итог

Блок 1 завершён успешно. Все микросервисы работают корректно, архитектура проверена. Docker Compose обеспечивает стабильный запуск и взаимодействие между компонентами. Проект готов к переходу к следующему этапу — разработке блока 2 (загрузка и отображение исходных изображений).