Программный продукт «TeamClub» Инструкция по установке

Локальное развертывание backend (developer guide)

Этот гайд нацелен на быстрый запуск dev-среды: MongoDB (реплика-сет), Redis, MinIO и сам backend.

Предпосылки
- Docker + Docker Compose
- Node.js >= 22, Yarn >= 4.9.2
- Порты: 3000 (backend), 27017-27019 (Mongo), 6379 (Redis), 9000/9090 (MinIO)

Шаг 1. Инфраструктура через Docker
```bash
docker compose -f docker-compose.dev.yml up -d
```
Поднимутся:
- MongoDB реплика-сет: `mongodb1:27017`, `mongodb2:27018`, `mongodb3:27019`
- Redis: `redis:6379` (Bull Board на http://localhost:3010)
- MinIO: API `http://localhost:9000`, консоль `http://localhost:9090` (логин/пароль: `minioadmin/minioadmin`), пользователь `teamclub/teamclub`, бакет `teamclub-store`.

Шаг 2. Конфигурация окружения
Создайте `.env` в корне на основе примера ниже.

```dotenv
# App
APP_NAME=Teamclub Backend
APP_ENV=development
APP_TIMEZONE=UTC
HTTP_HOST=0.0.0.0
HTTP_PORT=3000
URL_VERSIONING_ENABLE=true
URL_VERSION=1

# Database (см. src/configs/database.config.ts)

DATABASE_URL=mongodb://localhost:27017,localhost:27018,localhost:27019
DATABASE_DEBUG=false
```

```
# Redis (см. src/configs/redis.config.ts)
REDIS_HOST=localhost
REDIS_PORT=6379
REDIS_USERNAME=
REDIS_PASSWORD=

# Auth (см. src/configs/auth.config.ts)
AUTH_JWT_ACCESS_TOKEN_SECRET_KEY=dev-access-secret
AUTH_JWT_ACCESS_TOKEN_EXPIRED=15m
AUTH_JWT_REFRESH_TOKEN_SECRET_KEY=dev-refresh-secret
AUTH_JWT_REFRESH_TOKEN_EXPIRED=7d
AUTH_JWT_SUBJECT=teamclub
AUTH_JWT_AUDIENCE=teamclub
AUTH_JWT_ISSUER=teamclub

# AWS S3 compatible (MinIO) (см. src/configs/aws.config.ts)
AWS_S3_HOST=http://localhost:9000
AWS_S3_PUBLIC_BUCKET=teamclub-store
AWS_S3_PUBLIC_REGION=us-east-1
AWS_S3_PUBLIC_CDN=
AWS_S3_PUBLIC_CREDENTIAL_KEY=teamclub
AWS_S3_PUBLIC_CREDENTIAL_SECRET=teamclub
AWS_S3_PRIVATE_BUCKET=teamclub-store
AWS_S3_PRIVATE_REGION=us-east-1
AWS_S3_PRIVATE_CREDENTIAL_KEY=teamclub
AWS_S3_PRIVATE_CREDENTIAL_SECRET=teamclub

# SES (локально не используется)
AWS_SES_REGION=eu-central-1
AWS_SES_CREDENTIAL_KEY=
AWS_SES_CREDENTIAL_SECRET=
```

Шаг 3. Установка и запуск backend

```bash
yarn install
yarn build
yarn start:dev
```

Backend будет доступен на `http://localhost:3000`. Swagger — `http://localhost:3000/docs`.

Авторизация и доступ к системе

После запуска backend и выполнения сидирования базы данных, для авторизации доступны следующие учетные записи:

Тестовые пользователи

Для авторизации в системе используйте следующие учетные данные:

Основные пользователи:
- `superadmin@mail.com` / `password`
- `admin@mail.com` / `password`
- `member@mail.com` / `password`
- `user@mail.com` / `password`

Дополнительные пользователи:
- `user4@mail.com` до `user55@mail.com` / `password`

Примечание: В процессе сидирования создается 52 дополнительных пользователя (user4-user55) с ролью "user" и случайными именами для тестирования.

API авторизация

Для API-запросов доступны следующие варианты:

1. JWT токены (для веб/мобильных приложений):
- Эндпоинт авторизации: `POST /api/v1/auth/login/credential`
- Используйте email и password из учетных записей выше
- В ответе получите access и refresh токены
- Используйте Bearer токен в заголовке: `Authorization: Bearer <access_token>`

2. API ключи (для server-to-server интеграции):
- API Key: `v8VB0yY887lMpTA2VJMV`
- API Key Secret: `zeZbtGTugBTn3Qd5UXtSZBwt7gn3bg`
- Передавайте в заголовках: `x-api-key` и `x-api-key-secret`

Дополнительные сервисы

```
**MinIO Storage:**
- Консоль: `http://localhost:9090`
- API: `http://localhost:9000`
- Root пользователь: `minioadmin` / `minioadmin`
- App пользователь: `teamclub` / `teamclub`

**BullMQ Board (мониторинг очередей):**
- URL: `http://localhost:3010`
- Логин: `admin` / `admin123`

### Шаг 4. Начальные данные (опционально)
Есть команды сидирования:
```bash
yarn migrate:seed
```
или в прод-сценарии (после build):
```bash
yarn migrate:seed:prod
```

Частые проблемы
 - Mongo replica set не инициализировался: перезапустите сервисы, проверьте healthcheck `mongodb1` (docker logs).
  - MinIO недоступен: убедитесь, что порты 9000/9090 свободны.
- Redis конфликт портов: измените порт в compose и `.env`.

Остановка
```bash
docker compose -f docker-compose.dev.yml down -v
```
```

````
# TeamClub Frontend

 Фронтенд приложение TeamClub, построенное на React 19, Vite и Chakra UI.

 ## Требования

 - **Node.js**: версия 20 или выше
 - **Yarn**: версия 4.9.2 (управляется через Corepack)

 ## Пошаговый план запуска проекта

 ### Шаг 1: Включение Corepack

 Включаем Corepack для использования Yarn 4:

 ```bash
 corepack enable
 ```

 ### Шаг 2: Установка зависимостей

 Устанавливаем все необходимые зависимости проекта:

 ```bash
 yarn install
 ```

 ### Шаг 3: Генерация API типов по Swagger

 > **Важно**: Убедитесь, что у вас есть доступ к Swagger спецификации API или она уже находится в проекте.

 Генерируем TypeScript типы из OpenAPI/Swagger схемы:

 ```bash
 yarn codegen:api
 ```

 ### Шаг 4: Генерация типов темы Chakra UI

 Генерируем типы для темы оформления:
````

```bash
yarn theme:build
```

Или можно выполнить шаги 3 и 4 одной командой:

```bash
yarn gen
```

Шаг 5: Запуск приложения в режиме разработки

Запускаем dev сервер:

```bash
yarn dev
```

✅ Приложение будет доступно по адресу: **http://localhost:5173**

> **Примечание**: Команда `yarn dev` автоматически запускает ESLint в режиме отслеживания изменений и Vite dev server.

Шаг 6: Настройка бэкенда (опционально)

В режиме разработки все запросы к `/api` проксируются на `http://localhost:3000/api`.

Варианты настройки:

- Убедитесь, что ваш бэкенд запущен на порту 3000
- Или измените URL прокси в `vite.config.ts`
- Или настройте переменную окружения `VITE_API_BASE_URL` в файле `.env.local`

Переменные окружения

Создайте файл `.env.local` в корне проекта (опционально):

```env
# URL для API (по умолчанию "/")
```

```
VITE_API_BASE_URL=/
```

> **Примечание**: В режиме разработки можно не указывать эту переменную, так как Vite использует прокси.

Доступные команды

Разработка

```bash
# Запуск dev сервера с ESLint watcher
yarn dev

# Только Vite dev server (без ESLint)
vite dev
```

Сборка

```bash
# Production сборка
yarn build

# Предпросмотр production сборки
yarn preview
```

Линтинг и форматирование

```bash
# Проверка ESLint
yarn lint

# Автоматическое исправление ESLint ошибок
yarn lint:fix

# ESLint в режиме отслеживания
yarn lint:watch

# Форматирование кода через Prettier
yarn format
```

```
# Проверка форматирования
yarn format:check
```

Генерация типов

```bash
# Генерация API типов из OpenAPI схемы
yarn codegen:api

# Генерация типов темы Chakra UI
yarn theme:build

# Генерация всех типов (тема + API)
yarn gen
```

Архитектура (Feature-Sliced Design)

```bash
# Проверка FSD архитектуры
yarn fsd

# Автоматическое исправление FSD нарушений
yarn fsd:fix

# FSD в режиме отслеживания
yarn fsd:watch
```

Storybook

```bash
# Запуск Storybook
yarn sb

# Сборка Storybook
yarn build:sb
```

Тестирование

```bash
```

```
# Запуск unit тестов
yarn test:node

# Запуск E2E тестов
yarn test:e2e
```

Запуск через Docker

Сборка Docker образа

```bash
docker build -t teamclub-frontend .
```

Запуск контейнера

```bash
docker run -p 8080:80 -e VITE_API_URL=/api teamclub-frontend
```

Приложение будет доступно по адресу: http://localhost:8080

Подробнее о Docker деплое смотрите в [README.docker.md](./README.docker.md).

Структура проекта

Проект организован по методологии **Feature-Sliced Design (FSD)**:

```
src/
├── app/          # Инициализация приложения, роутинг, провайдеры
├── pages/        # Страницы приложения
├── widgets/      # Крупные композитные блоки
├── features/     # Функциональные возможности пользователя
├── entities/     # Бизнес-сущности
└── shared/     # Переиспользуемые модули
```

```
## Технологический стек

- **React 19** - UI библиотека
- **React Router 7** - Роутинг
- **Vite** - Сборщик и dev сервер
- **TypeScript** - Типизация
- **Chakra UI 3** - Компонентная библиотека
- **TanStack Query** - Управление серверным состоянием
- **React Hook Form** - Работа с формами
- **Zod** - Валидация схем
- **Zustand** - Управление клиентским состоянием
- **Day.js** - Работа с датами
- **Axios** - HTTP клиент

## Решение проблем

### Yarn не найден

Убедитесь, что Corepack включен:

```bash
corepack enable
```

Ошибки ESLint при запуске

Попробуйте автоматически исправить ошибки:

```bash
yarn lint:fix
```

Проблемы с зависимостями

Очистите кэш и переустановите:

```bash
# Удалите node_modules и кэш Yarn
rm -rf node_modules .yarn/cache

# Переустановите зависимости
yarn install
```
```

```
### API запросы не работают

Убедитесь, что:

1. Бэкенд запущен на порту 3000
2. Проверьте настройки прокси в `vite.config.ts`
3. Проверьте переменную окружения `VITE_API_BASE_URL`

## Дополнительная информация

- **Port**: По умолчанию Vite использует порт 5173
- **HMR**: Поддерживается Hot Module Replacement для быстрой разработки
- **Browser Support**: Современные браузеры (ES2015+)

## Полезные ссылки

- [Vite Documentation](https://vitejs.dev/)
- [React Documentation](https://react.dev/)
- [Chakra UI Documentation](https://www.chakra-ui.com/)
- [Feature-Sliced Design](https://feature-sliced.design/)
```