Отчет об анализе проекта MedNAIS

Дата анализа: 11 декабря 2025
Репозиторий: https://github.com/N1Mee/MedNAIS
Основной проект: `/mednais-analysis/mednais-sop-marketplace-main/`

📋 Обзор проекта

Используемые технологии

Frontend:
- Next.js 14.2.28 (App Router)
- React 18.2.0
- TypeScript 5.2.2
- Tailwind CSS 3.3.3
- Radix UI (компоненты)
- React Hook Form + Zod (формы и валидация)

Backend:
- Next.js API Routes (основной бэкенд)
- FastAPI (Python proxy на порту 8001)
- Prisma ORM 6.7.0 (JavaScript + Python клиенты)

Базы данных:
- PostgreSQL (основная БД)
- Qdrant (векторная БД для поиска)

Внешние сервисы:
- Stripe (платежи)
- AWS S3 / LocalStack (хранилище файлов)
- Firebase Admin (аутентификация)
- OpenAI (AI генерация SOP)

Инфраструктура:
- Docker Compose (локальная разработка)
- Yarn 1.22.22 (менеджер пакетов)

🚨 КРИТИЧЕСКИЕ ПРОБЛЕМЫ

1. ❌ Отсутствие файла `.env` и `.env.example`

Серьезность: КРИТИЧЕСКАЯ

Проблема:
- В репозитории полностью отсутствуют файлы `.env` и `.env.example`
- README.md упоминает необходимость копирования `.env.example`, но файл не существует
- Невозможно запустить проект без знания всех необходимых переменных окружения

Найденные переменные окружения:

```
# Базы данных
DATABASE_URL="postgresql://postgres:postgres@localhost:5432/mednais_sops?
schema=public"

# JWT и аутентификация
JWT_SECRET="your-secret-key"
NEXTAUTH_URL="http://localhost:3000"

# Firebase
FIREBASE_SERVICE_ACCOUNT_KEY="json-string"
NEXT_PUBLIC_FIREBASE_PROJECT_ID="project-id"

# AWS S3
AWS_S3_ENDPOINT="http://localhost:4566"
AWS_S3_REGION="us-east-1"
AWS_S3_ACCESS_KEY_ID="123"
AWS_S3_SECRET_ACCESS_KEY="123"
AWS_S3_BUCKET_NAME="mednais-uploads"
AWS_S3_FOLDER_PREFIX=""

# Stripe
STRIPE_API_KEY="sk_test_..."

# Email (SMTP)
SMTP_HOST="smtp.gmail.com"
SMTP_PORT="587"
SMTP_USER="your-email@gmail.com"
SMTP_PASSWORD="your-app-password"
FROM_EMAIL="noreply@yourdomain.com"

# OAuth
GOOGLE_CLIENT_ID="your-google-client-id"
APPLE_CLIENT_ID="your-apple-client-id"

# Другие
PYTHON_PATH="/usr/bin/python3"
```

Решение:
Создать файл `.env.example` со всеми необходимыми переменными и добавить подробную документацию.

---

2. ❌ Отсутствие Dockerfile для продакшн деплоя

Серьезность: КРИТИЧЕСКАЯ

Проблема:
- В проекте есть `docker-compose.yml`, но только для локальных сервисов (PostgreSQL, LocalStack, Qdrant)
- Отсутствует Dockerfile для сборки основного Next.js приложения
- Невозможно задеплоить приложение в контейнеризированном окружении

Решение:
Создать `Dockerfile` и обновить `docker-compose.yml` для включения основного приложения.

---

3. ⚠️ Незавершенная JWT аутентификация в Stripe

Серьезность: ВЫСОКАЯ

Файл: `backend/stripe_routes.py:52-54`

```
# TODO: Implement proper JWT verification here
# For now, we'll assume the user is authenticated
# This is a simplified version - in production, verify JWT properly
```

Проблема:
- Stripe endpoints не проверяют JWT токены должным образом
- Потенциальная уязвимость безопасности
- Hardcoded значение `"buyerId": "GUEST"` в строке 162

Решение:
Реализовать полную JWT верификацию для Stripe endpoints.

---

4. ⚠️ Hardcoded значения в критических местах

Серьезность: ВЫСОКАЯ

Проблемы:

1. **Stripe buyerId** (`backend/stripe_routes.py:162`):
   `python`
   `"buyerId": "GUEST",  # TODO: Get from user context`
   - Все покупки записываются с buyerId “GUEST”
   - Нет связи с реальным пользователем

2. **LocalStack credentials** (используются в примерах):
   - ACCESS_KEY_ID: “123”
   - SECRET_ACCESS_KEY: “123”
   - Необходимо заменить на реальные AWS credentials для продакшна

Решение:
- Получать реального пользователя из JWT токена
- Добавить валидацию окружения (dev/prod)

⚠️ ВАЖНЫЕ ПРОБЛЕМЫ

5. ⚠️ Отсутствующие зависимости

Серьезность: СРЕДНЯЯ

Проблема:
- `node_modules/` не установлены (ожидаемо для репозитория)
- Python зависимости не установлены
- Нет инструкций по установке Python зависимостей для backend

Файлы с зависимостями:
- `package.json` - Node.js зависимости
- `backend/requirements.txt` - FastAPI backend
- `scripts/requirements.txt` - AI processing scripts

Решение:
Добавить в README.md секцию по установке всех зависимостей:

```
# Node.js dependencies
yarn install

# Python backend dependencies
pip install -r backend/requirements.txt

# Python scripts dependencies
pip install -r scripts/requirements.txt

# Prisma
npx prisma generate
```

---

6. ⚠️ Отсутствие миграций базы данных в продакшне

Серьезность: СРЕДНЯЯ

Проблема:
- `init_services.sh` использует `npx prisma migrate deploy`, но это не подходит для продакшн окружения
- Нет документации по настройке PostgreSQL в продакшне
- `docker-compose.yml` настроен только для локальной разработки (127.0.0.1)

Решение:
- Создать отдельную конфигурацию для продакшна
- Добавить документацию по работе с миграциями
- Настроить proper PostgreSQL hosting

---

7. ⚠️ Неполная реализация OAuth

Серьезность: СРЕДНЯЯ

Файл: `app/auth/page.tsx`

```
// TODO: Implement Google OAuth flow
// TODO: Implement Apple Sign In flow
```

Проблема:
- Google и Apple OAuth не реализованы полностью
- Кнопки есть, но функциональность отсутствует
- В DEPLOYMENT_NOTES.md помечено как “Needs configuration ⏳”

Решение:
Реализовать полные OAuth flows или удалить неработающие кнопки.

---

8. ⚠️ Проблемы с категориями и админ-панелью

Серьезность: СРЕДНЯЯ

Файлы:
- `app/api/categories/route.ts:25`
- `app/api/categories/suggestions/route.ts:45`

```
// TODO: In production, add admin check here
// TODO: Add admin check in production
```

Проблема:
- Любой пользователь может создавать категории
- Нет системы ролей и прав доступа
- Потенциальная проблема со спамом и злоупотреблениями

Решение:
Реализовать систему ролей (admin, user) и проверку прав доступа.

---

9. ⚠️ Незавершенная функциональность групп

Серьезность: СРЕДНЯЯ

Файл: `app/groups/[id]/group-detail-tabs.tsx`

```
// TODO: Implement approve functionality
// TODO: Implement reject functionality
```

Проблема:
- Функции одобрения/отклонения участников группы не реализованы
- База данных поддерживает статусы (PENDING, APPROVED, REJECTED), но нет UI

Решение:
Реализовать полный функционал управления участниками групп.

📝 ЗАМЕЧАНИЯ И УЛУЧШЕНИЯ

10. 📌 Структура проекта

Проблема:
- Основной код находится в `mednais-analysis/mednais-sop-marketplace-main/`
- В корне есть папки `Uploads/` и `sop-marketplace-analysis/` которые не относятся к коду
- Неясная структура репозитория

Рекомендация:
Переместить основной код в корень репозитория для упрощения навигации.

---

11. 📌 FastAPI Proxy

Замечание:
- FastAPI backend используется только как proxy для Next.js
- Согласно DEPLOYMENT_NOTES.md: “workaround for Kubernetes ingress routing”
- Добавляет дополнительную сложность

Рекомендация:
- В продакшне использовать прямой роутинг к Next.js
- Или переместить Stripe логику в Next.js API routes

---

12. 📌 Версии зависимостей

Потенциальные проблемы:

1. **Next.js 14.2.28** - не последняя версия (актуальная 15.x)
2. **ESLint конфликт:**
   `json`
   `"eslint": "9.24.0",`
   `"eslint-config-next": "15.3.0"`
   - Версия ESLint (9.x) может конфликтовать с `eslint-config-next` (15.3.0)
3. **Zod 4.1.12** - очень новая версия, может иметь breaking changes
4. **UUID 13.0.0** - нестабильная версия (обычно используется ^9.0.0)

Рекомендация:
Проверить совместимость версий и обновить до стабильных.

---

13. 📌 TypeScript конфигурация

Замечание:

```
"typescript": {
  "ignoreBuildErrors": false
}
```

Проблема:
- В `next.config.js` установлено `ignoreBuildErrors: false`
- Но в `tsconfig.json` используется `"strict": true`
- Это может приводить к проблемам при сборке

Рекомендация:
Убедиться, что весь код соответствует строгим правилам TypeScript.

---

14. 📌 Безопасность и секреты

Проблемы:

1. **Hardcoded credentials в docker-compose.yml:**
   ```yaml
   POSTGRES_PASSWORD: postgres
   ACCESS_KEY_ID: "123"
   SECRET_ACCESS_KEY: "123"
   ```

2. **Отсутствие валидации входных данных:**
   - Некоторые API endpoints не используют Zod schemas
   - Потенциальные SQL injection риски

3. **JWT секрет:**
   - Нет проверки наличия `JWT_SECRET`
   - Приложение может запуститься с небезопасным секретом

Рекомендация:
- Использовать environment-specific конфигурации
- Добавить валидацию всех критических env variables при старте
- Использовать Docker secrets в продакшне

---

15. 📌 Документация

Проблемы:

1. **README.md** содержит неверную информацию:
   - Упоминает несуществующий `.env.example`
   - Не описывает Python backend setup
   - Отсутствует информация о FastAPI proxy

2. **API документация:**
   - Нет Swagger/OpenAPI документации для FastAPI
   - Нет подробного описания Next.js API routes

3. **Deployment:**
   - `DEPLOYMENT_NOTES.md` содержит полезную информацию, но устарела
   - Нет инструкций по деплою на реальный домен

Рекомендация:
Обновить всю документацию и добавить API docs.

---

16. 📌 Тестирование

Проблема:
- Полное отсутствие тестов
- Нет `jest.config.js`, `vitest.config.ts` или других testing frameworks
- Нет CI/CD pipeline

Рекомендация:
Добавить unit и integration тесты для критических частей приложения.

---

17. 📌 Prisma Schema

Потенциальные проблемы:

1. **Двойной generator:**
   ```prisma
   generator client {
   provider = “prisma-client-js”
   }

generator db {
provider = “prisma-client-py”
}
```
   - Используются оба клиента (JS и Python)
   - Необходимо запускать prisma generate` для обоих

1. **Cascade deletions:**
   - Многие связи используют `onDelete: Cascade`
   - Может привести к непреднамеренному удалению данных

Рекомендация:
Пересмотреть стратегию удаления и добавить soft deletes где необходимо.

---

18. 📌 Firebase конфигурация

Замечание:

Файл: `lib/firebase.ts`

```
// Firebase is not needed for this demo app
// Authentication is handled through test endpoints
export const auth = null;
export const googleProvider = null;
export const appleProvider = null;
```

Проблема:
- Firebase помечен как неиспользуемый, но `firebase-admin` используется
- Неясно, требуется ли Firebase для продакшн версии

Рекомендация:
Уточнить требования к Firebase и обновить документацию.

🔍 ДОПОЛНИТЕЛЬНЫЕ НАХОДКИ

19. 📌 Неиспользуемые файлы и папки

Найдено:
- `app/page-old.tsx` - старая версия главной страницы
- `test2/page.tsx` - тестовая страница

- `lib/i18n/translations-old.ts` - старые переводы
- `Uploads/` в корне репозитория - должны быть в gitignore

Рекомендация:
Очистить репозиторий от устаревшего кода.

---

20. 📌 Отсутствие .gitignore записей

Проблема:
Необходимо проверить `.gitignore` на наличие:
- `.env` и `.env.local`
- `node_modules/`
- `.next/`
- `dist/`
- Database files
- IDE specific files

---

21. 📌 Проблемы с i18n

Замечание:
- Проект использует `next-intl` для интернационализации
- Есть файлы с переводами, но неясно, насколько полная реализация
- В `TRANSLATION_GUIDE.md` может быть больше информации

Рекомендация:
Проверить полноту переводов и документировать процесс добавления новых языков.

📊 СТАТИСТИКА ПРОЕКТА

Размер проекта

```
Общее количество файлов: 416+
TypeScript/TSX файлы: ~150+
Python файлы: ~5
Конфигурационные файлы: ~15
```

Зависимости

```
Node.js пакетов: 100+
Python пакетов: 6-10
```

Структура

```
app/              - Next.js страницы и API routes
components/       - React компоненты
lib/              - Утилиты и конфигурации
prisma/           - Схема БД и миграции
backend/          - FastAPI proxy
scripts/          - Python скрипты для AI
public/           - Статические файлы
```

✅ РЕКОМЕНДАЦИИ ПО ПРИОРИТЕТАМ

🔴 Критично (необходимо исправить перед деплоем):

1. ✅ Создать `.env.example` с полным списком переменных
2. ✅ Создать Dockerfile для продакшн деплоя
3. ✅ Реализовать JWT верификацию в Stripe endpoints
4. ✅ Заменить hardcoded “GUEST” buyerId на реального пользователя
5. ✅ Настроить правильную конфигурацию для продакшн БД

🟡 Важно (рекомендуется исправить в ближайшее время):

1. ✅ Реализовать или удалить OAuth кнопки (Google, Apple)
2. ✅ Добавить систему ролей и админ-проверки
3. ✅ Завершить функциональность групп (approve/reject)
4. ✅ Добавить полную документацию по установке
5. ✅ Проверить и обновить версии зависимостей

🟢 Желательно (улучшения качества кода):

1. ✅ Добавить тесты
2. ✅ Очистить репозиторий от устаревшего кода
3. ✅ Настроить CI/CD
4. ✅ Добавить API документацию (Swagger)
5. ✅ Улучшить структуру проекта

🎯 ЗАКЛЮЧЕНИЕ

Проект **MedNAIS** представляет собой довольно сложное и хорошо структурированное Next.js приложение с продвинутыми функциями:

✅ Сильные стороны:

- Современный tech stack (Next.js 14, TypeScript, Prisma)
- Хорошая структура компонентов
- Использование best practices (Zod, React Hook Form)
- Присутствует базовая документация
- Поддержка Docker для локальной разработки

⚠️ Основные проблемы:

- **Отсутствие .env конфигурации** - блокирует запуск
- **Незавершенная аутентификация** - проблемы безопасности
- **Отсутствие Dockerfile** - невозможен деплой
- **Неполная реализация функций** - OAuth, админка, группы

📋 Следующие шаги:

1. Создать `.env.example` и настроить окружение
2. Исправить критические проблемы безопасности
3. Создать Dockerfile и настроить деплой
4. Завершить незавершенные фичи
5. Добавить тесты и CI/CD

Проект готов к разработке на ~70-80%. Основная функциональность реализована, но требуется значительная работа по конфигурации, безопасности и завершению некоторых фич перед продакшн деплоем.

Создано: Автоматизированный анализ проекта
Следующий этап: Исправление ошибок и настройка для деплоя