# ДОКУМЕНТАЦИЯ

## Система интеллектуального анализа документов "Искусственный интеллект НИКА"

### Версия 1.0

## 1. ОБЩЕЕ ОПИСАНИЕ СИСТЕМЫ

### 1.1. Назначение

Система "Искусственный интеллект НИКА" предназначена для интеллектуального анализа электронных документов различных форматов с применением технологий искусственного интеллекта, машинного обучения и обработки естественного языка. Система включает веб-интерфейс чат-бота на Vue.js для интерактивного взаимодействия с пользователями.

### 1.2. Основные компоненты

* **Backend**: Python-сервисы для обработки документов и ИИ-анализа
* **Frontend**: Vue.js приложение с интерфейсом чат-бота
* **База данных**: PostgreSQL для хранения данных
* **ML-сервисы**: Модули машинного обучения на TensorFlow/PyTorch
* **NLP-модули**: Обработка естественного языка (BERT, GPT-подобные модели)

## 2. АРХИТЕКТУРА СИСТЕМЫ

### 2.1. Общая архитектура

┌─────────────────────────────────────────────────┐

│ Vue.js Frontend (Чат-бот) │

├─────────────────────────────────────────────────┤

│ REST API (FastAPI) │

├─────────────────────────────────────────────────┤

│ Микросервисная архитектура │

├──────────┬──────────┬──────────┬────────────────┤

│ Загрузка │ Анализ │ ML │ Рекомендации │

│документов│ текста │ модули │ генератор │

├──────────┴──────────┴──────────┴────────────────┤

│ PostgreSQL / Redis │

└─────────────────────────────────────────────────┘

### 2.2. Frontend архитектура (Vue.js)

#### Структура компонентов

src/

├── components/

│ ├── ChatBot/

│ │ ├── ChatWindow.vue # Основное окно чата

│ │ ├── MessageList.vue # Список сообщений

│ │ ├── MessageItem.vue # Компонент сообщения

│ │ ├── InputArea.vue # Поле ввода

│ │ ├── FileUploader.vue # Загрузчик файлов

│ │ └── TypingIndicator.vue # Индикатор набора

│ ├── DocumentAnalysis/

│ │ ├── DocumentViewer.vue # Просмотр документов

│ │ ├── AnalysisResults.vue # Результаты анализа

│ │ └── RecommendationCard.vue # Карточка рекомендации

│ └── Common/

│ ├── Header.vue # Заголовок

│ ├── Sidebar.vue # Боковая панель

│ └── LoadingSpinner.vue # Индикатор загрузки

├── views/

│ ├── Chat.vue # Страница чата

│ ├── Dashboard.vue # Дашборд

│ └── Documents.vue # Управление документами

├── store/ # Vuex хранилище

│ ├── modules/

│ │ ├── chat.js # Состояние чата

│ │ ├── documents.js # Состояние документов

│ │ └── user.js # Данные пользователя

│ └── index.js

├── services/ # API сервисы

│ ├── api.js # Базовый API клиент

│ ├── chatService.js # Сервис чата

│ └── documentService.js # Сервис документов

└── router/ # Vue Router

└── index.js

## 3. ФУНКЦИОНАЛЬНЫЕ МОДУЛИ

### 3.1. Модуль чат-бота (Vue.js Frontend)

#### Основные функции:

* **Интерактивный диалог**: Обмен сообщениями в реальном времени
* **Загрузка документов**: Drag & Drop интерфейс для загрузки файлов
* **Визуализация результатов**: Графики, таблицы, карточки с рекомендациями
* **История диалогов**: Сохранение и навигация по предыдущим сессиям
* **Контекстные подсказки**: Автодополнение и предложения запросов

#### Технические характеристики:

* **Framework**: Vue.js 3.x с Composition API
* **State Management**: Vuex 4.x
* **UI Framework**: Vuetify 3.x / Element Plus
* **HTTP Client**: Axios
* **WebSocket**: Socket.io-client для real-time обновлений
* **Графики**: Chart.js / Apache ECharts

### 3.2. Backend модули

#### 3.2.1. Модуль загрузки и конвертации документов

* Поддержка форматов: XLSX, XLS, DOCX, DOC, PDF
* Извлечение текста, таблиц, метаданных
* Унификация данных для дальнейшей обработки

#### 3.2.2. Модуль интеллектуального анализа

* Семантический анализ на основе BERT
* Извлечение именованных сущностей
* Определение тематики документа
* Анализ числовых данных

#### 3.2.3. Модуль машинного обучения

* Классификация документов
* Кластеризация
* Прогнозирование
* Обнаружение аномалий

#### 3.2.4. Модуль генерации рекомендаций

* Формирование правил на основе анализа
* Генерация текстовых рекомендаций (GPT-подобная архитектура)
* Ранжирование по релевантности

## 4. API ДОКУМЕНТАЦИЯ

### 4.1. REST API Endpoints

#### Аутентификация

POST /api/auth/login

POST /api/auth/logout

POST /api/auth/refresh

#### Чат

POST /api/chat/message # Отправить сообщение

GET /api/chat/history # История чата

GET /api/chat/session/{id} # Получить сессию

DELETE /api/chat/session/{id} # Удалить сессию

#### Документы

POST /api/documents/upload # Загрузить документ

GET /api/documents/list # Список документов

GET /api/documents/{id} # Получить документ

POST /api/documents/{id}/analyze # Анализировать документ

GET /api/documents/{id}/results # Результаты анализа

#### WebSocket Events

// Client -> Server

socket.emit('message', { text, sessionId })

socket.emit('typing', { sessionId })

socket.emit('file\_upload', { file, sessionId })

// Server -> Client

socket.on('response', { text, recommendations })

socket.on('analysis\_progress', { progress, status })

socket.on('analysis\_complete', { results })

## 5. МОДЕЛИ ДАННЫХ

### 5.1. Структура сообщения чата

interface ChatMessage {

id: string;

sessionId: string;

sender: 'user' | 'bot';

text: string;

timestamp: Date;

attachments?: Attachment[];

recommendations?: Recommendation[];

metadata?: {

processingTime: number;

confidence: number;

entities: Entity[];

};

}

### 5.2. Структура документа

interface Document {

id: string;

name: string;

type: 'excel' | 'word' | 'pdf';

size: number;

uploadDate: Date;

status: 'uploaded' | 'processing' | 'analyzed' | 'error';

analysisResults?: AnalysisResult;

extractedData?: {

text: string;

tables: Table[];

metadata: DocumentMetadata;

};

}

### 5.3. Структура рекомендации

interface Recommendation {

id: string;

type: 'action' | 'insight' | 'warning';

title: string;

description: string;

priority: 'high' | 'medium' | 'low';

confidence: number;

relatedEntities: string[];

suggestedActions?: Action[];

}

## 6. ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС

### 6.1. Основные экраны

#### 6.1.1. Экран чата

* **Область сообщений**: Отображение диалога с поддержкой markdown
* **Поле ввода**: Многострочный ввод с поддержкой команд
* **Панель документов**: Drag & Drop зона для загрузки файлов
* **Панель рекомендаций**: Карточки с рекомендациями

#### 6.1.2. Дашборд

* **Статистика использования**: Графики и метрики
* **Последние документы**: Список недавно загруженных файлов
* **Активные сессии**: Текущие диалоги
* **Системные уведомления**: Важные события

#### 6.1.3. Управление документами

* **Библиотека документов**: Таблица с фильтрацией и поиском
* **Предпросмотр**: Быстрый просмотр содержимого
* **Результаты анализа**: Детальная информация по каждому документу

### 6.2. UI/UX принципы

* **Адаптивный дизайн**: Поддержка desktop/tablet/mobile
* **Темная/светлая тема**: Переключаемые цветовые схемы
* **Accessibility**: WCAG 2.1 Level AA compliance
* **Интернационализация**: i18n поддержка (русский/английский)

## 7. БЕЗОПАСНОСТЬ

### 7.1. Frontend безопасность

* **XSS защита**: Sanitization пользовательского ввода
* **CSRF токены**: Защита от межсайтовой подделки запросов
* **Content Security Policy**: Ограничение источников контента
* **HTTPS only**: Принудительное использование SSL

### 7.2. Аутентификация и авторизация

* **JWT токены**: Access и Refresh токены
* **Role-Based Access Control**: Ролевая модель доступа
* **Multi-Factor Authentication**: Опциональная 2FA
* **Session management**: Контроль сессий пользователя

## 8. ПРОИЗВОДИТЕЛЬНОСТЬ

### 8.1. Оптимизация Frontend

* **Code splitting**: Ленивая загрузка модулей
* **Tree shaking**: Удаление неиспользуемого кода
* **Compression**: Gzip/Brotli сжатие
* **CDN**: Статика через Content Delivery Network
* **Service Workers**: Офлайн поддержка и кеширование

### 8.2. Метрики производительности

* **First Contentful Paint**: < 1.5s
* **Time to Interactive**: < 3.5s
* **Largest Contentful Paint**: < 2.5s
* **Cumulative Layout Shift**: < 0.1

## 9. МОНИТОРИНГ И ЛОГИРОВАНИЕ

### 9.1. Frontend мониторинг

* **Error tracking**: Sentry интеграция
* **Performance monitoring**: Web Vitals метрики
* **User analytics**: Отслеживание действий пользователя
* **A/B testing**: Эксперименты с интерфейсом

### 9.2. Backend мониторинг

* **Application Performance Monitoring**: Prometheus + Grafana
* **Log aggregation**: ELK Stack (Elasticsearch, Logstash, Kibana)
* **Health checks**: Endpoints для проверки состояния
* **Alerts**: Уведомления о критических событиях

## 10. ТЕСТИРОВАНИЕ

### 10.1. Frontend тестирование

* **Unit tests**: Jest + Vue Test Utils
* **Integration tests**: Cypress
* **E2E tests**: Playwright/Selenium
* **Visual regression**: Percy/Chromatic

### 10.2. Backend тестирование

* **Unit tests**: pytest
* **API tests**: Postman/Newman
* **Load testing**: Locust/K6
* **Security testing**: OWASP ZAP

## 11. DEPLOYMENT

### 11.1. Контейнеризация

# Frontend Dockerfile

FROM node:18-alpine AS builder

WORKDIR /app

COPY package\*.json ./

RUN npm ci

COPY . .

RUN npm run build

FROM nginx:alpine

COPY --from=builder /app/dist /usr/share/nginx/html

COPY nginx.conf /etc/nginx/nginx.conf

### 11.2. Orchestration

* **Kubernetes**: Deployment манифесты
* **Docker Compose**: Локальная разработка
* **CI/CD**: GitLab CI / GitHub Actions
* **Environment**: dev/staging/production

## 12. ПОДДЕРЖКА И ОБСЛУЖИВАНИЕ

### 12.1. Версионирование

* **Semantic Versioning**: MAJOR.MINOR.PATCH
* **Changelog**: Документирование изменений
* **Migration guides**: Руководства по обновлению
* **Deprecation policy**: Политика устаревания

### 12.2. Документация

* **API Documentation**: OpenAPI/Swagger
* **Component Library**: Storybook
* **User Guide**: Руководство пользователя
* **Developer Guide**: Руководство разработчика