Система охраны труда

Техническая документация

Команда разработки

Содержание

1. Система охраны труда

Добро пожаловать в техническую документацию **Системы охраны труда** - комплексной веб-платформы для управления безопасностью на рабочих местах.

1.1 Скачать документацию

Скачать полную документацию в PDF

1.2 О системе

Система охраны труда представляет собой современное решение для автоматизации процессов управления безопасностью, включающее:

- **Управление документами** и формами по охране труда
- **Оценку профессиональных рисков** с использованием матричного метода
- **Учет средств индивидуальной защиты (СИЗ)**
- **Медицинские осмотры** и обучение сотрудников
- **Аналитику и отчетность** по безопасности
- **Систему поддержки** и уведомлений

1.3 Архитектура

Система построена на современном технологическом стеке:

- **Backend**: Python FastAPI + SQLAlchemy + PostgreSQL
- **Frontend**: React + TypeScript + Vite + shadcn/ui
- **База данных**: PostgreSQL с 33 таблицами
- **Файловое хранилище**: SFTP + локальное хранилище
- **Аутентификация**: JWT токены

1.4 Пользователи системы

1.4.1 Администраторы системы

Полный контроль над системой, управление пользователями, аналитика и мониторинг.

1.4.2 Организации

Управление сотрудниками, документами, оценка рисков, учет СИЗ и медицинских осмотров.

1.4.3 Сотрудники

Заполнение форм, доступ к документам, просмотр уведомлений и личной информации.

1.5 Быстрый старт

1. **Установка и настройка** - настройка системы
2. **Руководство администратора** - управление системой
3. **API документация** - интеграция с системой
4. **База данных** - схема базы данных

1.6 Структура документации

```
graph TD
    A[Главная] --> B[Обзор системы]
    A --> C[Установка и настройка]
    A --> D[Руководство администратора]
    A --> E[Руководство организации]
    A --> F[Управление документами]
    A --> G[Безопасность и риски]
    A --> H[API документация]
    A --> I[База данных]
    A --> J[Разработка]
    A --> K[Развертывание]
```

1.7 Полезные ссылки

- **API документация** - полное описание всех эндпоинтов
- **Схема базы данных** - структура данных
- **FAQ** - часто задаваемые вопросы
- **Changelog** - история изменений

1.8 Поддержка

Если у вас возникли вопросы или проблемы:

- Обратитесь к **FAQ**
- Изучите **API документацию**
- Проверьте **схему базы данных**

Документация актуальна и регулярно обновляется. Последнее обновление: 5 февраля 2026 г.

5 февраля 2026 г.

5 февраля 2026 г.

2. Обзор системы

2.1 Обзор системы

2.1.1 Назначение

Система охраны труда - это комплексная веб-платформа, предназначенная для автоматизации процессов управления безопасностью на рабочих местах. Система обеспечивает полный цикл управления охраной труда от оценки рисков до ведения документации и отчетности.

2.1.2 Основные цели

- **Автоматизация** процессов управления охраной труда
- **Централизация** документооборота по безопасности
- **Стандартизация** процедур оценки рисков
- **Контроль** соблюдения требований безопасности
- **Аналитика** и отчетность по безопасности

2.1.3 Целевая аудитория

Администраторы системы

- Полный контроль над системой
- Управление пользователями и организациями
- Мониторинг использования системы
- Аналитика и отчетность

Руководители организаций

- Управление сотрудниками и процессами
- Контроль соблюдения требований безопасности
- Аналитика по безопасности организации
- Принятие управленческих решений

Специалисты по охране труда

- Ведение документации по безопасности
- Проведение оценки рисков
- Учет СИЗ и медицинских осмотров
- Подготовка отчетов

Сотрудники организаций

- Заполнение форм и документов
- Доступ к инструкциям по безопасности
- Получение уведомлений
- Просмотр личной информации

2.1.4 Ключевые возможности

Документооборот

- Загрузка и управление документами
- Создание шаблонов для автозаполнения
- Электронные формы с валидацией
- Контроль версий документов

⚠ Оценка рисков

- Матричный метод оценки (5x5)
- Классификатор опасностей
- Планирование мероприятий по снижению рисков
- Автоматический расчет коэффициентов

Управление СИЗ

- Справочники средств защиты
- Нормы выдачи по должностям
- Учет выдачи и сроков действия
- Контроль обеспеченности

Медицинские осмотры

- Учет различных типов осмотров
- Контроль сроков прохождения
- Результаты и ограничения
- Планирование следующих осмотров

Аналитика и отчетность

- Дашборды с KPI показателями
- Графики и диаграммы
- Автоматические отчеты
- Экспорт данных

2.1.5 Бизнес-процессы

1. Регистрация организации

```
graph LR
    A[Регистрация] --> B[Импорт из ФНС]
    B --> C[Настройка участников]
    C --> D[Создание рабочих зон]
    D --> E[Готово к работе]
```

2. Оценка рисков

```
graph LR
    A[Выбор должности] --> B[Идентификация опасностей]
    B --> C[Оценка рисков]
    C --> D[Планирование мероприятий]
    D --> E[Контроль выполнения]
```

3. Управление СИЗ

```
graph LR
    A[Определение норм] --> B[Заказ СИЗ]
    B --> C[Выдача сотрудникам]
    C --> D[Контроль сроков]
    D --> E[Замена/обновление]
```

2.1.6 Преимущества системы

✅ **Для организаций**

- **Соответствие требованиям** законодательства
- **Снижение рисков** несчастных случаев
- **Автоматизация** рутинных процессов
- **Централизованное** управление безопасностью
- **Аналитика** для принятия решений

✅ **Для сотрудников**

- **Упрощенный** доступ к документации
- **Автоматические** напоминания
- **Прозрачность** процессов безопасности
- **Мобильный** доступ к информации

✅ **Для администраторов**

- **Масштабируемость** системы
- **Гибкая** настройка под потребности
- **Интеграция** с внешними системами
- **Мониторинг** использования

2.1.7 Планы развития

Краткосрочные (3-6 месяцев)

- Мобильное приложение
- Расширенная аналитика
- Интеграция с 1С
- Автоматические уведомления

Среднесрочные (6-12 месяцев)

- ИИ для анализа рисков
- Интеграция с системами видеонаблюдения
- Расширенная отчетность
- Многоязычность

Долгосрочные (1-2 года)

- Платформа для обучения
- Интеграция с IoT устройствами

- Предиктивная аналитика
- Экосистема партнеров

2.1.8 Метрики успеха

Количественные показатели

- **Снижение** количества инцидентов на 30%
- **Ускорение** процессов документооборота в 3 раза
- **Покрытие** 100% сотрудников системой учета
- **Сокращение** времени на подготовку отчетов на 80%

Качественные показатели

- **Повышение** осведомленности сотрудников о безопасности
- **Улучшение** культуры безопасности в организации
- **Стандартизация** процессов управления рисками
- **Повышение** эффективности работы специалистов по ОТ

Система постоянно развивается и совершенствуется на основе обратной связи пользователей и изменений в законодательстве.

5 февраля 2026 г.

5 февраля 2026 г.

2.2 Архитектура системы

2.2.1 Общая архитектура

Система охраны труда построена по принципу **многоуровневой архитектуры** с четким разделением ответственности между компонентами.

```
graph TB
    subgraph "Frontend Layer"
        A[React Frontend]
        B[Mobile App]
    end

    subgraph "API Gateway"
        C[FastAPI Backend]
        D[Authentication]
        E[Rate Limiting]
    end

    subgraph "Business Logic"
        F[Document Management]
        G[Risk Assessment]
        H[PPE Management]
        I[User Management]
    end

    subgraph "Data Layer"
        J[PostgreSQL Database]
        K[File Storage]
        L[Cache Layer]
    end

    subgraph "External Services"
        M[FNS API]
        N[SMTP Server]
        O[SFTP Storage]
    end

    A --> C
    B --> C
    C --> F
    C --> G
    C --> H
    C --> I
    F --> J
    G --> J
    H --> J
    I --> J
    F --> K
    C --> L
    C --> M
    C --> N
    C --> O
```

2.2.2 Принципы архитектуры

1. Разделение ответственности (Separation of Concerns)

- Каждый компонент отвечает за свою область функциональности
- Четкие границы между слоями
- Независимость компонентов

2. Масштабируемость (Scalability)

- Горизонтальное масштабирование
- Асинхронная обработка
- Кэширование данных

3. Безопасность (Security)

- Многоуровневая аутентификация

- Авторизация на уровне ресурсов
- Шифрование данных

4. Надежность (Reliability)

- Обработка ошибок
- Резервное копирование
- Мониторинг системы

2.2.3 Frontend Layer

React Frontend

- **Технологии**: React 18, TypeScript, Vite
- **UI Framework**: shadcn/ui + Tailwind CSS
- **State Management**: TanStack Query
- **Routing**: Wouter

```
// Пример структуры компонента
interface DocumentFormProps {
  documentId: number;
  onSubmit: (data: FormData) => void;
}

const DocumentForm: React.FC<DocumentFormProps> = ({ documentId, onSubmit }) => {
  const { data: document } = useQuery(['document', documentId],
    () => fetchDocument(documentId)
  );

  return (
    <form onSubmit={handleSubmit}>
      {/* Форма документа */}
    </form>
  );
};
```

Мобильное приложение

- **Технологии**: React Native
- **Функциональность**: просмотр документов, уведомления, заполнение форм
- **Синхронизация**: офлайн-режим с последующей синхронизацией

2.2.4 API Gateway

FastAPI Backend

- **Технологии**: Python 3.11+, FastAPI, Pydantic
- **Аутентификация**: JWT токены
- **Документация**: автоматическая генерация OpenAPI/Swagger

```
# Пример API эндпоинта
@router.post("/documents/upload", response_model=DocumentResponse)
async def upload_document(
    file: UploadFile = File(...),
    name: str = Form(...),
    current_user: User = Depends(get_current_active_user),
    db: Session = Depends(get_db)
):
    """Загрузка нового документа"""
    # Логика загрузки документа
    pass
```

Middleware

- **CORS**: настройка для фронтенда
- **Rate Limiting**: ограничение запросов
- **Logging**: логирование всех запросов
- **Error Handling**: централизованная обработка ошибок

2.2.5 Business Logic Layer

Модули системы

1. DOCUMENT MANAGEMENT

```python
class DocumentService:
    def upload_document(self, file: UploadFile, metadata: dict) -> Document:
        """Загрузка и обработка документа"""
        pass

    def extract_template_fields(self, file_path: str) -> dict:
        """Извлечение полей шаблона"""
        pass

    def generate_filled_document(self, template_id: int, data: dict) -> str:
        """Генерация заполненного документа"""
        pass
```

2. RISK ASSESSMENT

```python
class RiskAssessmentService:
    def calculate_risk_score(self, severity: int, probability: int,
                             coefficients: dict) -> RiskResult:
        """Расчет балла риска"""
        pass

    def get_risk_recommendations(self, risk_level: str) -> list:
        """Получение рекомендаций по снижению риска"""
        pass
```

3. PPE MANAGEMENT

```python
class PPEService:
    def get_ppe_norms(self, position_id: int) -> list:
        """Получение норм выдачи СИЗ для должности"""
        pass

    def track_ppe_issuance(self, employee_id: int, ppe_items: list) -> None:
        """Учет выдачи СИЗ"""
        pass
```

2.2.6 Data Layer

PostgreSQL Database

- **Версия**: PostgreSQL 14+
- **ORM**: SQLAlchemy 2.0
- **Миграции**: Alembic
- **Индексы**: оптимизированы для частых запросов

```sql
-- Пример структуры таблицы
CREATE TABLE safety_users (
    id SERIAL PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    full_name VARCHAR(200) NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    role VARCHAR(20) DEFAULT 'user',
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_users_username ON safety_users(username);
```

```
CREATE INDEX idx_users_email ON safety_users(email);
CREATE INDEX idx_users_role ON safety_users(role);
```

File Storage

- **Локальное хранилище**: для разработки и тестирования
- **SFTP**: для продакшн среды
- **Структура**: организована по типам документов и организациям

```
uploads/
├── documents/
│   ├── templates/
│   ├── filled/
│   └── public/
├── organizations/
│   ├── {org_id}/
│   │   ├── documents/
│   │   ├── forms/
│   │   └── reports/
└── temp/
```

Cache Layer

- **Redis**: для кэширования сессий и часто запрашиваемых данных
- **TTL**: настраиваемое время жизни кэша
- **Invalidation**: автоматическая инвалидация при изменениях

2.2.7 External Services

FNS API Integration

```
class FNSService:
    def get_organization_data(self, inn: str) -> OrganizationData:
        """Получение данных организации из ФНС"""
        pass

    def validate_inn(self, inn: str) -> bool:
        """Валидация ИНН"""
        pass
```

Email Service

```
class EmailService:
    def send_notification(self, to: str, subject: str, body: str) -> bool:
        """Отправка email уведомления"""
        pass

    def send_bulk_notifications(self, recipients: list, message: dict) -> int:
        """Массовая отправка уведомлений"""
        pass
```

2.2.8 Security Architecture

Аутентификация

```
sequenceDiagram
    participant U as User
    participant F as Frontend
    participant A as API
    participant D as Database

    U->>F: Login credentials
    F->>A: POST /auth/login
    A->>D: Validate credentials
    D-->>A: User data
    A-->>F: JWT token
    F-->>U: Success + token
```

Авторизация

- **RBAC**: Role-Based Access Control
- **Уровни доступа**: system, organization, user
- **Проверка прав**: на каждом эндпоинте

```
def require_organization_membership(organization_id: int):
    def decorator(func):
        @wraps(func)
        async def wrapper(*args, **kwargs):
            current_user = kwargs.get('current_user')
            if not check_membership(current_user.id, organization_id):
                raise HTTPException(403, "Access denied")
            return await func(*args, **kwargs)
        return wrapper
    return decorator
```

2.2.9 Monitoring & Logging

Логирование

- **Структурированные логи**: JSON формат
- **Уровни**: DEBUG, INFO, WARNING, ERROR, CRITICAL
- **Ротация**: по размеру и времени
- **Централизованный сбор**: ELK Stack

Мониторинг

- **Метрики**: производительность, использование ресурсов
- **Алерты**: автоматические уведомления о проблемах
- **Дашборды**: визуализация состояния системы

2.2.10 Deployment Architecture

Контейнеризация

```
# Dockerfile для backend
FROM python:3.11-slim

WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt

COPY . .
EXPOSE 8000

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

Orchestration

- **Docker Compose**: для разработки
- **Kubernetes**: для продакшн
- **Load Balancer**: распределение нагрузки
- **Auto-scaling**: автоматическое масштабирование

2.2.11 Data Flow

Типичный запрос

```
sequenceDiagram
    participant U as User
    participant F as Frontend
    participant A as API
```

```
    participant S as Service
    participant D as Database
    participant C as Cache

    U->>F: User action
    F->>A: API request + JWT
    A->>A: Validate token
    A->>C: Check cache
    alt Cache hit
        C-->>A: Cached data
    else Cache miss
        A->>S: Business logic
        S->>D: Database query
        D-->>S: Data
        S-->>A: Processed data
        A->>C: Store in cache
    end
    A-->>F: Response
    F-->>U: UI update
```

2.2.12 Performance Considerations

Оптимизация базы данных

- **Индексы**: на часто используемых полях
- **Партиционирование**: для больших таблиц
- **Connection pooling**: переиспользование соединений
- **Query optimization**: оптимизированные запросы

Кэширование

- **Application cache**: часто используемые данные
- **Database cache**: результаты запросов
- **CDN**: статические ресурсы
- **Browser cache**: клиентское кэширование

Асинхронность

- **Background tasks**: тяжелые операции
- **Message queues**: обработка событий
- **WebSockets**: real-time уведомления
- **Streaming**: большие файлы

Архитектура системы спроектирована с учетом требований масштабируемости, безопасности и производительности.

5 февраля 2026 г.

5 февраля 2026 г.

2.3 ✨ Функциональные возможности

2.3.1 Обзор функций

Система охраны труда предоставляет широкий спектр функциональных возможностей, охватывающих все аспекты управления безопасностью на рабочих местах.

2.3.2 Функции для администраторов системы

Управление системой

- **Аутентификация администратора**: безопасный вход в систему
- **Управление пользователями**: создание, редактирование, деактивация
- **Управление организациями**: регистрация, настройка, мониторинг
- **Системные настройки**: конфигурация параметров системы

Аналитика и мониторинг

- **Дашборд администратора**: общая статистика системы
- **Метрики использования**: активность пользователей и организаций
- **Отчеты по системе**: детальная аналитика работы
- **Мониторинг производительности**: состояние системы в реальном времени

Администрирование

- **Очистка данных**: удаление устаревших записей
- **Резервное копирование**: автоматическое создание бэкапов
- **Обновления системы**: управление версиями
- **Логирование**: просмотр системных логов

2.3.3 Функции для организаций

Управление организацией

- **Регистрация организации**: создание профиля организации
- **Импорт из ФНС**: автоматическое получение данных из налоговой
- **Настройка участников**: управление ролями и правами доступа
- **Рабочие зоны**: создание и управление зонами работы

Управление сотрудниками

- **Реестр сотрудников**: полный список сотрудников организации
- **Роли и права**: назначение ролей и управление доступом
- **Профили сотрудников**: детальная информация о каждом сотруднике
- **Статистика**: аналитика по сотрудникам и отделам

Организационная структура

- **Должности**: управление должностями в организации
- **Подразделения**: структурирование организации

- **Назначения**: привязка сотрудников к рабочим зонам
- **Иерархия**: построение организационной структуры

2.3.4 Управление документами

Документооборот

- **Загрузка документов**: поддержка Excel, Word, PDF
- **Категоризация**: организация документов по типам и категориям
- **Версионирование**: контроль версий документов
- **Поиск**: быстрый поиск по документам

Шаблоны и формы

- **Создание шаблонов**: разработка шаблонов для автозаполнения
- **Извлечение полей**: автоматическое определение полей в шаблонах
- **Электронные формы**: создание интерактивных форм
- **Валидация**: проверка корректности заполнения

Контроль доступа

- **Уровни доступа**: public, system, organization
- **Права пользователей**: read, write, admin
- **Временные ограничения**: настройка сроков доступа
- **Аудит доступа**: логирование всех обращений к документам

2.3.5 Безопасность и риски

⚠ Оценка профессиональных рисков

- **Матричный метод**: оценка по шкале 5x5
- **Классификатор опасностей**: база данных опасностей
- **Расчет коэффициентов**: автоматический расчет рисков
- **Рекомендации**: предложения по снижению рисков

Управление рисками

- **Планирование мероприятий**: разработка планов снижения рисков
- **Контроль выполнения**: отслеживание выполнения мероприятий
- **Ответственные**: назначение ответственных лиц
- **Сроки**: контроль временных рамок

Аналитика рисков

- **Статистика рисков**: анализ по организации
- **Тренды**: отслеживание изменений рисков
- **Сравнение**: сопоставление с нормативами
- **Отчеты**: детальная отчетность по рискам

2.3.6 Средства индивидуальной защиты (СИЗ)

Справочники СИЗ

- **Типы защиты**: классификация по видам защиты
- **Наименования СИЗ**: полный каталог средств защиты
- **Нормы выдачи**: стандарты выдачи по должностям
- **Спецификации**: технические характеристики СИЗ

Учет выдачи СИЗ

- **Журнал выдачи**: учет всех выдач СИЗ
- **Сроки действия**: контроль сроков использования
- **Подписи**: электронные подписи получателей
- **Инвентаризация**: учет наличия СИЗ

Контроль обеспеченности

- **Статистика выдачи**: аналитика по СИЗ
- **Покрытие**: процент обеспеченности сотрудников
- **Планирование**: прогнозирование потребностей
- **Отчеты**: детальная отчетность по СИЗ

2.3.7 Медицинские осмотры

Типы осмотров

- **Предварительные**: осмотры при приеме на работу
- **Периодические**: регулярные медицинские осмотры
- **Внеочередные**: осмотры по показаниям
- **Целевые**: осмотры для конкретных работ

Учет результатов

- **Результаты осмотров**: годен, негоден, годен с ограничениями
- **Ограничения**: медицинские ограничения к работе
- **Рекомендации**: рекомендации врачей
- **Справки**: учет медицинских справок

Контроль сроков

- **Планирование**: график медицинских осмотров
- **Напоминания**: автоматические уведомления
- **Просрочки**: контроль просроченных осмотров
- **Статистика**: аналитика по медосмотрам

2.3.8 Аналитика и отчетность

Дашборды

- **KPI показатели**: ключевые метрики безопасности

- **Графики**: визуализация данных
- **Тренды**: анализ изменений во времени
- **Сравнения**: сопоставление показателей

Отчеты

- **Автоматические отчеты**: регулярная генерация отчетов
- **Настраиваемые отчеты**: создание пользовательских отчетов
- **Экспорт данных**: выгрузка в различных форматах
- **Расписание**: автоматическая отправка отчетов

Карты ОПР

- **Генерация карт**: автоматическое создание карт оценки рисков
- **Шаблоны**: различные форматы карт
- **Заполнение**: автоматическое заполнение данными
- **Печать**: подготовка к печати

2.3.9 Система уведомлений

Email уведомления

- **SMTP настройка**: конфигурация почтового сервера
- **Шаблоны**: настраиваемые шаблоны писем
- **Расписание**: автоматическая отправка по расписанию
- **Статистика**: отчеты по отправленным уведомлениям

Внутренние уведомления

- **Типы уведомлений**: напоминания, предупреждения, информация
- **Приоритеты**: настройка важности уведомлений
- **Группировка**: объединение связанных уведомлений
- **История**: архив всех уведомлений

2.3.10 Система поддержки

Обращения в поддержку

- **Создание обращений**: форма для создания обращений
- **Категоризация**: классификация проблем
- **Приоритеты**: настройка важности обращений
- **Статусы**: отслеживание статуса обработки

Коммуникация

- **Сообщения**: обмен сообщениями с поддержкой
- **Вложения**: прикрепление файлов к обращениям
- **История**: полная история переписки
- **Уведомления**: информирование о статусе обращения

2.3.11 Файловое хранилище

Управление файлами

- **Загрузка**: загрузка файлов в систему
- **Организация**: структурированное хранение
- **Поиск**: быстрый поиск файлов
- **Метаданные**: информация о файлах

Интеграции

- **SFTP**: интеграция с SFTP серверами
- **Локальное хранилище**: хранение на локальном сервере
- **Синхронизация**: синхронизация между хранилищами
- **Резервирование**: автоматическое резервное копирование

2.3.12 Безопасность

Аутентификация

- **JWT токены**: безопасная аутентификация
- **Роли и права**: система ролей и разрешений
- **Сессии**: управление пользовательскими сессиями
- **Пароли**: требования к сложности паролей

Авторизация

- **Контроль доступа**: проверка прав на ресурсы
- **Организационная изоляция**: разделение данных организаций
- **Аудит**: логирование всех действий пользователей
- **Шифрование**: защита конфиденциальных данных

2.3.13 Мобильность

Мобильное приложение

- **Кроссплатформенность**: iOS и Android
- **Офлайн режим**: работа без интернета
- **Синхронизация**: автоматическая синхронизация данных
- **Push уведомления**: мгновенные уведомления

Веб-интерфейс

- **Адаптивность**: оптимизация для мобильных устройств
- **PWA**: прогрессивное веб-приложение
- **Офлайн поддержка**: работа в офлайн режиме
- **Быстрая загрузка**: оптимизация производительности

2.3.14 Интеграции

Государственные сервисы

- **ФНС API**: получение данных организаций
- **Валидация ИНН**: проверка корректности ИНН
- **Автоматическое заполнение**: заполнение данных организации
- **Обновление данных**: синхронизация с государственными реестрами

Внешние системы

- **1С**: интеграция с системами учета
- **HR системы**: синхронизация данных сотрудников
- **Почтовые сервисы**: интеграция с email провайдерами
- **API**: RESTful API для интеграции

Все функции системы разработаны с учетом требований российского законодательства и лучших практик в области охраны труда.

5 февраля 2026 г.

5 февраля 2026 г.

2.4 Технологический стек

2.4.1 Обзор технологий

Система охраны труда построена на современном технологическом стеке, обеспечивающем высокую производительность, масштабируемость и безопасность.

2.4.2 Frontend

React Ecosystem

- **React 18.2+**: современная библиотека для создания пользовательских интерфейсов
- **TypeScript 5.0+**: типизированный JavaScript для повышения надежности кода
- **Vite 4.0+**: быстрый инструмент сборки и разработки

```
{
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "typescript": "^5.0.0",
    "vite": "^4.0.0"
  }
}
```

UI Framework

- **shadcn/ui**: современная библиотека компонентов
- **Tailwind CSS 3.0+**: utility-first CSS фреймворк
- **Radix UI**: низкоуровневые компоненты для доступности

```
// Пример использования shadcn/ui компонента
import { Button } from "@/components/ui/button"
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card"

const DocumentCard = ({ document }: { document: Document }) => (
  <Card>
    <CardHeader>
      <CardTitle>{document.name}</CardTitle>
    </CardHeader>
    <CardContent>
      <Button variant="outline">Скачать</Button>
    </CardContent>
  </Card>
)
```

State Management

- **TanStack Query**: управление серверным состоянием
- **Zustand**: легковесное управление клиентским состоянием
- **React Hook Form**: управление формами с валидацией

```
// Пример использования TanStack Query
import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query'

const useDocuments = () => {
  return useQuery({
    queryKey: ['documents'],
    queryFn: fetchDocuments,
    staleTime: 5 * 60 * 1000, // 5 минут
  })
}

const useCreateDocument = () => {
  const queryClient = useQueryClient()

  return useMutation({
    mutationFn: createDocument,
    onSuccess: () => {
      queryClient.invalidateQueries({ queryKey: ['documents'] })
```

```
    },
  })
}
```

Routing & Navigation

- **Wouter**: легковесный роутер для React
- **React Router**: альтернативный роутер (опционально)

```
import { Route, useLocation } from 'wouter'

const App = () => (
  <div>
    <Route path="/" component={Dashboard} />
    <Route path="/documents" component={Documents} />
    <Route path="/documents/:id" component={DocumentDetail} />
  </div>
)
```

2.4.3 Backend

Python Framework

- **Python 3.11+**: современная версия Python с улучшенной производительностью
- **FastAPI 0.100+**: высокопроизводительный веб-фреймворк
- **Pydantic 2.0+**: валидация данных и сериализация

```
# Пример FastAPI эндпоинта
from fastapi import FastAPI, Depends, HTTPException
from pydantic import BaseModel
from typing import List

app = FastAPI(title="Safety API", version="1.0.0")

class DocumentCreate(BaseModel):
    name: str
    description: str | None = None
    category: str

@app.post("/documents/", response_model=DocumentResponse)
async def create_document(
    document: DocumentCreate,
    current_user: User = Depends(get_current_active_user),
    db: Session = Depends(get_db)
):
    """Создание нового документа"""
    db_document = Document(**document.dict(), uploaded_by=current_user.id)
    db.add(db_document)
    db.commit()
    db.refresh(db_document)
    return db_document
```

Database & ORM

- **PostgreSQL 14+**: надежная реляционная база данных
- **SQLAlchemy 2.0+**: современный Python ORM
- **Alembic**: система миграций базы данных

```
# Пример модели SQLAlchemy
from sqlalchemy import Column, Integer, String, DateTime, Boolean, Text
from sqlalchemy.ext.declarative import declarative_base
from datetime import datetime

Base = declarative_base()

class User(Base):
    __tablename__ = "safety_users"

    id = Column(Integer, primary_key=True, index=True)
    username = Column(String(50), unique=True, index=True, nullable=False)
    email = Column(String(100), unique=True, index=True, nullable=False)
    full_name = Column(String(200), nullable=False)
    password_hash = Column(String(255), nullable=False)
    role = Column(String(20), default="user")
    is_active = Column(Boolean, default=True)
```

```
    created_at = Column(DateTime, default=datetime.utcnow)
    updated_at = Column(DateTime, default=datetime.utcnow, onupdate=datetime.utcnow)
```

Authentication & Security

- **JWT**: JSON Web Tokens для аутентификации
- **bcrypt**: хеширование паролей
- **python-jose**: работа с JWT токенами
- **passlib**: библиотека для работы с паролями

```python
# Пример аутентификации
from jose import JWTError, jwt
from passlib.context import CryptContext
from datetime import datetime, timedelta

pwd_context = CryptContext(schemes=["bcrypt"], deprecated="auto")

def verify_password(plain_password: str, hashed_password: str) -> bool:
    return pwd_context.verify(plain_password, hashed_password)

def create_access_token(data: dict, expires_delta: timedelta | None = None):
    to_encode = data.copy()
    if expires_delta:
        expire = datetime.utcnow() + expires_delta
    else:
        expire = datetime.utcnow() + timedelta(minutes=15)
    to_encode.update({"exp": expire})
    encoded_jwt = jwt.encode(to_encode, SECRET_KEY, algorithm=ALGORITHM)
    return encoded_jwt
```

2.4.4 Database

PostgreSQL Features

- **JSON/JSONB**: хранение неструктурированных данных
- **Full-text search**: полнотекстовый поиск
- **Indexing**: оптимизированные индексы
- **Partitioning**: партиционирование больших таблиц

```sql
-- Пример использования JSONB
CREATE TABLE organizations (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    contacts JSONB,
    additional_data JSONB
);

-- Создание индекса для JSONB
CREATE INDEX idx_organizations_contacts ON organizations USING GIN (contacts);

-- Запрос с использованием JSONB
SELECT * FROM organizations
WHERE contacts->>'email' = 'admin@example.com';
```

Database Optimization

- **Connection pooling**: переиспользование соединений
- **Query optimization**: оптимизация запросов
- **Caching**: кэширование результатов
- **Monitoring**: мониторинг производительности

2.4.5 File Processing

Document Processing

- **openpyxl**: работа с Excel файлами

- **python-docx**: работа с Word документами
- **PyPDF2**: обработка PDF файлов
- **python-magic**: определение типов файлов

```python
# Пример обработки Excel файла
import openpyxl
from openpyxl import load_workbook

def extract_excel_fields(file_path: str) -> dict:
    """Извлечение полей из Excel шаблона"""
    workbook = load_workbook(file_path)
    fields = {}

    for sheet_name in workbook.sheetnames:
        sheet = workbook[sheet_name]
        sheet_fields = []

        for row in sheet.iter_rows():
            for cell in row:
                if cell.value and isinstance(cell.value, str):
                    if '{' in cell.value and '}' in cell.value:
                        # Извлекаем поля в фигурных скобках
                        import re
                        matches = re.findall(r'\{([^}]+)\}', cell.value)
                        sheet_fields.extend(matches)

        fields[sheet_name] = list(set(sheet_fields))

    return fields
```

File Storage

- **SFTP**: безопасная передача файлов
- **Local storage**: локальное хранение
- **File validation**: проверка типов файлов
- **Virus scanning**: сканирование на вирусы

2.4.6 External Integrations

API Integrations

- **httpx**: асинхронные HTTP запросы
- **requests**: синхронные HTTP запросы
- **FNS API**: интеграция с налоговой службой
- **SMTP**: отправка email уведомлений

```python
# Пример интеграции с FNS API
import httpx
from typing import Optional

class FNSService:
    def __init__(self, api_key: str):
        self.api_key = api_key
        self.base_url = "https://api-fns.ru/api"

    async def get_organization_data(self, inn: str) -> Optional[dict]:
        """Получение данных организации из ФНС"""
        async with httpx.AsyncClient() as client:
            response = await client.get(
                f"{self.base_url}/egr",
                params={"req": inn, "key": self.api_key}
            )
            if response.status_code == 200:
                return response.json()
        return None
```

Email Service

- **aiosmtplib**: асинхронная отправка email
- **email-validator**: валидация email адресов

- **Jinja2**: шаблонизация email сообщений

```
# Пример отправки email
import aiosmtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

async def send_email(to: str, subject: str, body: str):
    """Отправка email уведомления"""
    message = MIMEMultipart()
    message["From"] = SMTP_USER
    message["To"] = to
    message["Subject"] = subject
    message.attach(MIMEText(body, "html"))

    await aiosmtplib.send(
        message,
        hostname=SMTP_HOST,
        port=SMTP_PORT,
        username=SMTP_USER,
        password=SMTP_PASSWORD,
        use_tls=True
    )
```

2.4.7 DevOps & Deployment

Containerization

- **Docker**: контейнеризация приложений
- **Docker Compose**: оркестрация для разработки
- **Multi-stage builds**: оптимизация образов

```
# Multi-stage Dockerfile
FROM node:18-alpine AS frontend-build
WORKDIR /app
COPY package*.json ./
RUN npm ci --only=production
COPY . .
RUN npm run build

FROM python:3.11-slim AS backend-build
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY . .

FROM python:3.11-slim AS production
WORKDIR /app
COPY --from=backend-build /app .
COPY --from=frontend-build /app/dist ./static
EXPOSE 8000
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

Monitoring & Logging

- **Prometheus**: сбор метрик
- **Grafana**: визуализация метрик
- **ELK Stack**: централизованное логирование
- **Sentry**: отслеживание ошибок

```
# Пример настройки логирования
import logging
import structlog
from pythonjsonlogger import jsonlogger

# Настройка структурированного логирования
structlog.configure(
    processors=[
        structlog.stdlib.filter_by_level,
        structlog.stdlib.add_logger_name,
        structlog.stdlib.add_log_level,
        structlog.stdlib.PositionalArgumentsFormatter(),
        structlog.processors.TimeStamper(fmt="iso"),
        structlog.processors.StackInfoRenderer(),
        structlog.processors.format_exc_info,
        structlog.processors.UnicodeDecoder(),
        structlog.processors.JSONRenderer()
    ],
```

```
    context_class=dict,
    logger_factory=structlog.stdlib.LoggerFactory(),
    wrapper_class=structlog.stdlib.BoundLogger,
    cache_logger_on_first_use=True,
)
```

CI/CD

- **GitHub Actions**: автоматизация сборки и развертывания
- **Testing**: pytest для backend, Jest для frontend
- **Code Quality**: black, flake8, mypy для Python
- **Security**: bandit для проверки безопасности

```yaml
# GitHub Actions workflow
name: CI/CD Pipeline

on:
  push:
    branches: [main, develop]
  pull_request:
    branches: [main]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Set up Python
        uses: actions/setup-python@v4
        with:
          python-version: '3.11'
      - name: Install dependencies
        run: |
          pip install -r requirements.txt
          pip install -r requirements-dev.txt
      - name: Run tests
        run: pytest tests/ --cov=app --cov-report=xml
      - name: Code quality
        run: |
          black --check app/
          flake8 app/
          mypy app/
```

2.4.8 Development Tools

Code Quality

- **Black**: форматирование Python кода
- **Flake8**: линтинг Python кода
- **MyPy**: статическая типизация
- **Pre-commit**: хуки для Git

```yaml
# .pre-commit-config.yaml
repos:
  - repo: https://github.com/psf/black
    rev: 23.3.0
    hooks:
      - id: black
        language_version: python3.11

  - repo: https://github.com/pycqa/flake8
    rev: 6.0.0
    hooks:
      - id: flake8

  - repo: https://github.com/pre-commit/mirrors-mypy
    rev: v1.3.0
    hooks:
      - id: mypy
        additional_dependencies: [types-all]
```

Testing

- **pytest**: тестирование Python кода

- **pytest-asyncio**: тестирование асинхронного кода
- **pytest-cov**: покрытие кода тестами
- **factory-boy**: создание тестовых данных

```
# Пример теста
import pytest
from fastapi.testclient import TestClient
from app.main import app
from app.database import get_db
from tests.factories import UserFactory

client = TestClient(app)

@pytest.fixture
def test_user():
    return UserFactory()

def test_create_document(test_user):
    response = client.post(
        "/documents/",
        json={"name": "Test Document", "category": "safety"},
        headers={"Authorization": f"Bearer {test_user.token}"}
    )
    assert response.status_code == 201
    assert response.json()["name"] == "Test Document"
```

2.4.9 Performance & Optimization

Caching

- **Redis**: кэширование данных
- **Memcached**: альтернативное кэширование
- **Application cache**: кэширование на уровне приложения
- **CDN**: кэширование статических ресурсов

Database Optimization

- **Connection pooling**: пул соединений с БД
- **Query optimization**: оптимизация SQL запросов
- **Indexing strategy**: стратегия индексирования
- **Partitioning**: партиционирование таблиц

Async Processing

- **Celery**: асинхронная обработка задач
- **Redis**: брокер сообщений
- **Background tasks**: фоновые задачи
- **Queue management**: управление очередями

Технологический стек выбран с учетом требований производительности, масштабируемости и современности разработки.

5 февраля 2026 г.

5 февраля 2026 г.

3. Установка и настройка

3.1 Установка и настройка

3.1.1 Обзор

Данное руководство поможет вам установить и настроить систему охраны труда на вашем сервере. Система поддерживает развертывание как в Docker контейнерах, так и в нативной среде.

3.1.2 Требования к системе

Минимальные требования

- **CPU**: 2 ядра
- **RAM**: 4 GB
- **Диск**: 50 GB свободного места
- **ОС**: Ubuntu 20.04+ / CentOS 8+ / Windows Server 2019+

Рекомендуемые требования

- **CPU**: 4+ ядра
- **RAM**: 8+ GB
- **Диск**: 100+ GB SSD
- **ОС**: Ubuntu 22.04 LTS

Программное обеспечение

- **Python**: 3.11+
- **Node.js**: 18+
- **PostgreSQL**: 14+
- **Redis**: 6+ (опционально)
- **Nginx**: 1.18+ (для продакшн)

3.1.3 Установка через Docker (Рекомендуется)

Предварительные требования

```
# Установка Docker
curl -fsSL https://get.docker.com -o get-docker.sh
sh get-docker.sh

# Установка Docker Compose
sudo curl -L "https://github.com/docker/compose/releases/download/v2.20.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

Клонирование репозитория

```
git clone https://github.com/safety-system/back_opp.git
cd back_opp
```

Настройка переменных окружения

```
# Копирование файла конфигурации
cp .env.example .env
```

```
# Редактирование конфигурации
nano .env
```

Пример .env файла:

```
# База данных
DATABASE_URL=postgresql://safety_user:safety_password@db:5432/safety_db

# Redis
REDIS_URL=redis://redis:6379

# JWT
SECRET_KEY=your-secret-key-here
ACCESS_TOKEN_EXPIRE_MINUTES=15

# CORS
CORS_ORIGINS=["http://localhost:3000", "https://yourdomain.com"]

# Email
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_USER=your-email@gmail.com
SMTP_PASSWORD=your-app-password

# Файловое хранилище
UPLOAD_DIR=uploads
SFTP_HOST=your-sftp-server.com
SFTP_USER=safety_user
SFTP_PASSWORD=safety_password
```

Запуск системы

```
# Запуск всех сервисов
docker-compose up -d

# Проверка статуса
docker-compose ps

# Просмотр логов
docker-compose logs -f
```

Инициализация базы данных

```
# Выполнение миграций
docker-compose exec backend alembic upgrade head

# Создание администратора
docker-compose exec backend python -c "
from app.database import SessionLocal
from app.models.models import User
from app.utils.auth import get_password_hash

db = SessionLocal()
admin = User(
    username='admin',
    email='admin@example.com',
    full_name='Администратор Системы',
    password_hash=get_password_hash('admin123'),
    role='admin',
    is_active=True,
    password_set=True,
    first_login_completed=True
)
db.add(admin)
db.commit()
db.close()
print('Администратор создан: admin / admin123')
"
```

3.1.4 Нативная установка

Установка зависимостей

UBUNTU/DEBIAN

```
# Обновление системы
sudo apt update && sudo apt upgrade -y

# Установка Python 3.11
sudo apt install software-properties-common -y
sudo add-apt-repository ppa:deadsnakes/ppa -y
```

```
sudo apt update
sudo apt install python3.11 python3.11-venv python3.11-dev -y

# Установка PostgreSQL
sudo apt install postgresql postgresql-contrib -y

# Установка Node.js
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt install nodejs -y

# Установка системных зависимостей
sudo apt install build-essential libpq-dev libffi-dev -y
```

CENTOS/RHEL

```
# Установка EPEL репозитория
sudo yum install epel-release -y

# Установка Python 3.11
sudo yum install python311 python311-pip python311-devel -y

# Установка PostgreSQL
sudo yum install postgresql-server postgresql-contrib -y
sudo postgresql-setup initdb
sudo systemctl enable postgresql
sudo systemctl start postgresql

# Установка Node.js
curl -fsSL https://rpm.nodesource.com/setup_18.x | sudo bash -
sudo yum install nodejs -y

# Установка системных зависимостей
sudo yum groupinstall "Development Tools" -y
sudo yum install postgresql-devel libffi-devel -y
```

Настройка базы данных

```
# Переключение на пользователя postgres
sudo -u postgres psql

# Создание базы данных и пользователя
CREATE DATABASE safety_db;
CREATE USER safety_user WITH PASSWORD 'safety_password';
GRANT ALL PRIVILEGES ON DATABASE safety_db TO safety_user;
\q
```

Установка Backend

```
# Клонирование репозитория
git clone https://github.com/safety-system/back_opp.git
cd back_opp

# Создание виртуального окружения
python3.11 -m venv venv
source venv/bin/activate

# Установка зависимостей
pip install -r requirements.txt

# Настройка переменных окружения
cp .env.example .env
nano .env
```

Установка Frontend

```
# Переход в директорию фронтенда
cd frontend

# Установка зависимостей
npm install

# Сборка для продакшн
npm run build
```

Запуск системы

```
# Запуск backend
cd ../python_backend
source venv/bin/activate
uvicorn main:app --host 0.0.0.0 --port 8000
```

```
# Запуск frontend (в отдельном терминале)
cd frontend
npm run preview
```

3.1.5 ⚙ Конфигурация

Настройка Nginx (Продакшн)

```
server {
    listen 80;
    server_name yourdomain.com;

    # Редирект на HTTPS
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    server_name yourdomain.com;

    # SSL сертификаты
    ssl_certificate /path/to/certificate.crt;
    ssl_certificate_key /path/to/private.key;

    # Frontend
    location / {
        root /path/to/frontend/dist;
        try_files $uri $uri/ /index.html;
    }

    # API
    location /api/ {
        proxy_pass http://localhost:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    # WebSocket (если используется)
    location /ws/ {
        proxy_pass http://localhost:8000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}
```

Настройка systemd сервисов

```
# /etc/systemd/system/safety-backend.service
[Unit]
Description=Safety System Backend
After=network.target

[Service]
Type=simple
User=safety
WorkingDirectory=/opt/safety-system/python_backend
Environment=PATH=/opt/safety-system/venv/bin
ExecStart=/opt/safety-system/venv/bin/uvicorn main:app --host 0.0.0.0 --port 8000
Restart=always

[Install]
WantedBy=multi-user.target
```

```
# Активация сервиса
sudo systemctl daemon-reload
sudo systemctl enable safety-backend
sudo systemctl start safety-backend
```

3.1.6 Настройка окружения

Переменные окружения

ОБЯЗАТЕЛЬНЫЕ ПЕРЕМЕННЫЕ

```
# База данных
DATABASE_URL=postgresql://user:password@localhost:5432/database

# Безопасность
```

```
SECRET_KEY=your-secret-key-here
ACCESS_TOKEN_EXPIRE_MINUTES=15

# CORS
CORS_ORIGINS=["http://localhost:3000"]
```

ОПЦИОНАЛЬНЫЕ ПЕРЕМЕННЫЕ

```
# Redis (для кэширования)
REDIS_URL=redis://localhost:6379

# Email
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_USER=your-email@gmail.com
SMTP_PASSWORD=your-app-password

# Файловое хранилище
UPLOAD_DIR=uploads
SFTP_HOST=your-sftp-server.com
SFTP_USER=safety_user
SFTP_PASSWORD=safety_password

# Логирование
LOG_LEVEL=INFO
LOG_FILE=/var/log/safety-system.log

# Мониторинг
SENTRY_DSN=your-sentry-dsn
```

Настройка SSL сертификатов

```
# Использование Let's Encrypt
sudo apt install certbot python3-certbot-nginx -y
sudo certbot --nginx -d yourdomain.com

# Автоматическое обновление
sudo crontab -e
# Добавить строку:
0 12 * * * /usr/bin/certbot renew --quiet
```

3.1.7 Настройка базы данных

Миграции

```
# Создание новой миграции
alembic revision --autogenerate -m "Description of changes"

# Применение миграций
alembic upgrade head

# Откат миграции
alembic downgrade -1
```

Резервное копирование

```
# Создание бэкапа
pg_dump -h localhost -U safety_user -d safety_db > backup_$(date +%Y%m%d_%H%M%S).sql

# Восстановление из бэкапа
psql -h localhost -U safety_user -d safety_db < backup_20240115_120000.sql
```

Мониторинг базы данных

```
-- Проверка размера базы данных
SELECT pg_size_pretty(pg_database_size('safety_db'));

-- Проверка активных соединений
SELECT count(*) FROM pg_stat_activity WHERE datname = 'safety_db';

-- Проверка медленных запросов
SELECT query, mean_time, calls
FROM pg_stat_statements
ORDER BY mean_time DESC
LIMIT 10;
```

3.1.8 Проверка установки

Проверка Backend

```
# Проверка доступности API
curl http://localhost:8000/api/health

# Проверка документации
curl http://localhost:8000/api/docs
```

Проверка Frontend

```
# Проверка доступности фронтенда
curl http://localhost:3000
```

Проверка базы данных

```
# Подключение к базе данных
psql -h localhost -U safety_user -d safety_db

# Проверка таблиц
\dt

# Проверка пользователей
SELECT username, email, role FROM safety_users;
```

3.1.9 Устранение неполадок

Частые проблемы

ПРОБЛЕМА: ОШИБКА ПОДКЛЮЧЕНИЯ К БАЗЕ ДАННЫХ

Решение: 1. Проверить статус PostgreSQL: `sudo systemctl status postgresql` 2. Проверить настройки в .env файле 3. Проверить права пользователя базы данных 4. Проверить сетевые настройки

ПРОБЛЕМА: ОШИБКА 500 ПРИ ЗАГРУЗКЕ ФАЙЛОВ

Решение: 1. Проверить права на директорию uploads 2. Проверить свободное место на диске 3. Проверить настройки Nginx (client_max_body_size) 4. Проверить логи приложения

ПРОБЛЕМА: МЕДЛЕННАЯ РАБОТА СИСТЕМЫ

Решение: 1. Проверить использование ресурсов сервера 2. Оптимизировать запросы к базе данных 3. Настроить кэширование Redis 4. Проверить индексы в базе данных

Логи и мониторинг

```
# Просмотр логов backend
tail -f /var/log/safety-system/backend.log

# Просмотр логов Nginx
tail -f /var/log/nginx/access.log
tail -f /var/log/nginx/error.log

# Мониторинг ресурсов
htop
iostat -x 1
```

3.1.10 Поддержка

При возникновении проблем с установкой: - **Email**: support@safety-system.com - **Документация**: https://docs.safety-system.com - **GitHub Issues**: https://github.com/safety-system/back_opp/issues - **Telegram**: @safety_system_support

---

Руководство по установке регулярно обновляется. Для получения актуальной информации проверяйте последнюю версию документации.

5 февраля 2026 г.

5 февраля 2026 г.

4. Руководство администратора

4.1 Руководство администратора

4.1.1 Обзор

Руководство администратора системы охраны труда содержит подробные инструкции по управлению системой, пользователями, организациями и мониторингу работы платформы.

4.1.2 Аутентификация администратора

Вход в систему

Администраторы системы имеют специальные права доступа ко всем функциям платформы.

URL: `/api/admin/auth/login`

Метод: `POST`

Параметры:

```
{
  "username": "admin",
  "password": "your_password"
}
```

Ответ:

```
{
  "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...",
  "token_type": "bearer"
}
```

Использование токена

Все последующие запросы должны содержать заголовок авторизации:

```
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...
```

4.1.3 Административная панель

Дашборд администратора

Главная страница администратора предоставляет обзор состояния системы.

URL: `/api/admin/dashboard`

Метод: `GET`

Ответ:

```
{
  "user_stats": {
    "total": 1250,
    "active": 1180,
    "new_this_month": 45,
    "inactive": 70
  },
  "document_stats": {
    "total": 3400,
    "templates": 150,
    "active": 3200,
    "new_this_month": 120
  },
  "form_stats": {
    "total": 5600,
```

```
    "pending": 45,
    "approved": 5200,
    "drafts": 355
  },
  "recent_activities": [
    {
      "type": "form",
      "action": "submitted",
      "user": "Иванов И.И.",
      "item": "Форма #1234",
      "timestamp": "2024-01-15T10:30:00Z"
    }
  ]
}
```

Общая статистика системы

Детальная статистика по всем аспектам работы системы.

URL: `/api/admin/statistics`

Метод: `GET`

Ответ:

```
{
  "total_users": 1250,
  "total_documents": 3400,
  "total_forms": 5600,
  "pending_forms": 45,
  "expired_trainings": 12,
  "total_organizations": 85,
  "total_organization_members": 1250,
  "total_document_downloads": 15600,
  "total_support_tickets": 234
}
```

4.1.4 Управление пользователями

Просмотр пользователей

Получение списка всех пользователей системы с возможностью фильтрации.

URL: `/api/users/`

Метод: `GET`

Параметры запроса: - `skip` (int): количество записей для пропуска (по умолчанию: 0) - `limit` (int): максимальное количество записей (по умолчанию: 100) - `department` (string): фильтр по отделу - `role` (string): фильтр по роли - `is_active` (boolean): фильтр по активности

Пример запроса:

```
GET /api/users/?role=admin&is_active=true&limit=50
```

Ответ:

```
[
  {
    "id": 1,
    "username": "admin",
    "email": "admin@example.com",
    "full_name": "Администратор Системы",
    "role": "admin",
    "department": "IT",
    "is_active": true,
    "created_at": "2024-01-01T00:00:00Z"
  }
]
```

Получение информации о пользователе

Детальная информация о конкретном пользователе.

URL: `/api/users/{user_id}`

Метод: `GET`

Ответ:

```
{
  "id": 1,
  "username": "admin",
  "email": "admin@example.com",
  "full_name": "Администратор Системы",
  "employee_number": "EMP001",
  "position": "Системный администратор",
  "department": "IT",
  "role": "admin",
  "is_active": true,
  "hire_date": "2024-01-01T00:00:00Z",
  "password_set": true,
  "first_login_completed": true,
  "created_at": "2024-01-01T00:00:00Z",
  "updated_at": "2024-01-15T10:30:00Z"
}
```

4.1.5 Управление организациями

Просмотр организаций

Получение списка всех зарегистрированных организаций.

URL: `/api/organizations/`

Метод: `GET`

Ответ:

```
[
  {
    "id": 1,
    "type": "ЮЛ",
    "inn": "7736207543",
    "ogrn": "1027739850962",
    "full_name": "ООО \"Компания\"",
    "short_name": "Компания",
    "status": "Действующее",
    "registration_date": "2020-01-15T00:00:00Z",
    "full_address": "г. Москва, ул. Примерная, д. 1",
    "main_activity_text": "Разработка ПО",
    "created_at": "2024-01-01T00:00:00Z"
  }
]
```

Импорт организации из ФНС

Автоматическое получение и обновление данных организации из API ФНС.

URL: `/api/organizations/fns/import/{inn}`

Метод: `POST`

Параметры: - `inn` (string): ИНН организации (10 цифр для ЮЛ, 12 для ИП) - `force_update` (boolean): принудительное обновление существующих данных

Пример запроса:

```
POST /api/organizations/fns/import/7736207543?force_update=true
```

Ответ:

```
{
  "id": 1,
  "type": "ЮЛ",
  "inn": "7736207543",
  "ogrn": "1027739850962",
  "full_name": "ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ \"КОМПАНИЯ\"",
  "status": "Действующее",
```

```
  "registration_date": "2002-12-26T00:00:00Z",
  "full_address": "г. Москва, ул. Примерная, д. 1",
  "main_activity_text": "Разработка компьютерного программного обеспечения",
  "created_at": "2024-01-01T00:00:00Z",
  "updated_at": "2024-01-15T10:30:00Z"
}
```


4.1.6 Отчеты и аналитика

Отчет об активности пользователей

Анализ активности пользователей за указанный период.

URL: `/api/admin/reports/user-activity`

Метод: `GET`

Параметры: - `days` (int): период в днях (по умолчанию: 30)

Ответ:


```
{
  "period_days": 30,
  "user_activity": [
    {
      "user_id": 1,
      "full_name": "Иванов И.И.",
      "department": "Безопасность",
      "forms_created": 15
    },
    {
      "user_id": 2,
      "full_name": "Петров П.П.",
      "department": "HR",
      "forms_created": 8
    }
  ]
}
```


Отчет об использовании документов

Статистика использования документов в системе.

URL: `/api/admin/reports/document-usage`

Метод: `GET`

Ответ:


```
{
  "document_usage": [
    {
      "document_id": 1,
      "name": "Инструкция по охране труда",
      "type": "instruction",
      "category": "general_safety",
      "usage_count": 156
    },
    {
      "document_id": 2,
      "name": "Форма допуска к работе",
      "type": "form",
      "category": "work_permit",
      "usage_count": 89
    }
  ]
}
```


Статистика по отделам

Анализ активности по отделам организации.

URL: `/api/admin/reports/department-stats`

Метод: `GET`

Ответ:

```
{
  "department_stats": [
    {
      "department": "Безопасность",
      "user_count": 15,
      "form_count": 234
    },
    {
      "department": "HR",
      "user_count": 8,
      "form_count": 156
    }
  ]
}
```

4.1.7 Системное администрирование

Очистка старых черновиков

Удаление устаревших черновиков форм для освобождения места.

URL: `/api/admin/cleanup/old-drafts`

Метод: `POST`

Параметры: - `days_old` (int): возраст черновиков в днях (по умолчанию: 30)

Ответ:

```
{
  "message": "Удалено 45 старых черновиков",
  "deleted_count": 45
}
```

Массовая отправка уведомлений

Отправка уведомлений группе пользователей или всем пользователям.

URL: `/api/admin/notifications/send-bulk`

Метод: `POST`

Параметры: - `title` (string): заголовок уведомления - `message` (string): текст уведомления - `user_ids` (array): список ID пользователей (опционально) - `department` (string): отдел для отправки (опционально)

Пример запроса:

```
{
  "title": "Обновление системы",
  "message": "Система будет обновлена завтра с 02:00 до 04:00",
  "department": "IT"
}
```

Ответ:

```
{
  "message": "Отправлено 25 уведомлений",
  "sent_count": 25
}
```

Информация о системе

Получение информации о состоянии системы и ресурсах.

URL: `/api/admin/system-info`

Метод: `GET`

Ответ:


```
{
  "disk": {
    "total": 1073741824000,
    "used": 536870912000,
    "free": 536870912000,
    "percent": 50.0
  },
  "memory": {
    "total": 8589934592,
    "used": 4294967296,
    "free": 4294967296,
    "percent": 50.0
  },
  "uploads_size": 1048576000,
  "database_tables": {
    "users": 1250,
    "documents": 3400,
    "forms": 5600,
    "access_permissions": 8900
  }
}
```


4.1.8 Управление доступом

Система ролей

Система использует иерархическую модель ролей:

1. **admin** - администратор системы (полный доступ)
2. **owner** - владелец организации (полный доступ к организации)
3. **manager** - менеджер организации (управление процессами)
4. **specialist** - специалист (базовый доступ)

Права доступа

Каждая роль имеет определенный набор прав:

```
graph TD
    A[admin] --> B[Все права системы]
    C[owner] --> D[Управление организацией]
    C --> E[Управление пользователями]
    C --> F[Управление документами]
    G[manager] --> H[Управление процессами]
    G --> I[Просмотр отчетов]
    J[specialist] --> K[Заполнение форм]
    J --> L[Просмотр документов]
```

4.1.9 Мониторинг и логирование

Логи системы

Все действия администратора логируются для аудита:

- Вход в систему
- Управление пользователями
- Изменение настроек
- Массовые операции

Метрики производительности

Система отслеживает ключевые метрики:

- Время отклика API
- Использование ресурсов
- Количество активных пользователей
- Ошибки и исключения

4.1.10 Устранение неполадок

Частые проблемы

ПРОБЛЕМА: ПОЛЬЗОВАТЕЛЬ НЕ МОЖЕТ ВОЙТИ В СИСТЕМУ

Решение: 1. Проверить статус пользователя (активен/неактивен) 2. Проверить правильность пароля 3. Проверить срок действия токена 4. Проверить логи системы на наличие ошибок

ПРОБЛЕМА: ОРГАНИЗАЦИЯ НЕ ИМПОРТИРУЕТСЯ ИЗ ФНС

Решение: 1. Проверить корректность ИНН 2. Проверить доступность API ФНС 3. Проверить настройки интеграции 4. Проверить логи API запросов

ПРОБЛЕМА: МЕДЛЕННАЯ РАБОТА СИСТЕМЫ

Решение: 1. Проверить использование ресурсов сервера 2. Проверить производительность базы данных 3. Проверить сетевые соединения 4. Оптимизировать запросы к БД

Контакты поддержки

При возникновении критических проблем: - Email: admin-support@example.com - Телефон: +7 (495) 123-45-67 - Система поддержки: `/api/support/`

Руководство администратора регулярно обновляется в соответствии с изменениями в системе.

5 февраля 2026 г.

5 февраля 2026 г.

5. API документация

5.1 API документация

5.1.1 Обзор API

Система охраны труда предоставляет RESTful API для интеграции с внешними системами и автоматизации процессов. API построен на основе OpenAPI 3.0 спецификации и поддерживает JSON формат данных.

5.1.2 Базовые URL

- **Production**: `https://api.safety-system.com/api`
- **Staging**: `https://staging-api.safety-system.com/api`
- **Development**: `http://localhost:8000/api`

5.1.3 Аутентификация

JWT токены

Все API запросы требуют аутентификации через JWT токены.

Получение токена:

```
POST /api/auth/login
Content-Type: application/json

{
  "username": "your_username",
  "password": "your_password"
}
```

Ответ:

```
{
  "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...",
  "token_type": "bearer"
}
```

Использование токена:

```
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...
```

Время жизни токена

- **Access token**: 15 минут
- **Refresh token**: 7 дней (если поддерживается)

5.1.4 Общие принципы

HTTP методы

- `GET` - получение данных
- `POST` - создание новых ресурсов
- `PUT` - полное обновление ресурса
- `PATCH` - частичное обновление ресурса
- `DELETE` - удаление ресурса

Коды ответов

- `200` - OK (успешный запрос)
- `201` - Created (ресурс создан)
- `400` - Bad Request (неверный запрос)
- `401` - Unauthorized (не авторизован)
- `403` - Forbidden (нет прав доступа)
- `404` - Not Found (ресурс не найден)
- `422` - Unprocessable Entity (ошибка валидации)
- `500` - Internal Server Error (внутренняя ошибка сервера)

Пагинация

Для эндпоинтов, возвращающих списки, используется пагинация:

Параметры: - `skip` (int): количество записей для пропуска (по умолчанию: 0) - `limit` (int): максимальное количество записей (по умолчанию: 100, максимум: 1000)

Пример:

```
GET /api/documents/?skip=20&limit=50
```

Ответ:

```
{
  "items": [...],
  "total": 1000,
  "skip": 20,
  "limit": 50,
  "has_next": true,
  "has_prev": true
}
```

Фильтрация и поиск

Многие эндпоинты поддерживают фильтрацию:

```
GET /api/users/?role=admin&is_active=true&department=IT
```

Сортировка

Сортировка через параметр `sort`:

```
GET /api/documents/?sort=created_at:desc
```

5.1.5 Основные эндпоинты

Аутентификация

ВХОД В СИСТЕМУ

```
POST /api/auth/login
```

ПОЛУЧЕНИЕ ИНФОРМАЦИИ О ТЕКУЩЕМ ПОЛЬЗОВАТЕЛЕ

```
GET /api/auth/me
```

УСТАНОВКА ПАРОЛЯ (ПЕРВЫЙ ВХОД)

```
POST /api/auth/setup-password
```

Административные функции

ДАШБОРД АДМИНИСТРАТОРА

```
GET /api/admin/dashboard
```

ОБЩАЯ СТАТИСТИКА

```
GET /api/admin/statistics
```

ОТЧЕТЫ

```
GET /api/admin/reports/user-activity?days=30
GET /api/admin/reports/document-usage
GET /api/admin/reports/department-stats
```

СИСТЕМНОЕ АДМИНИСТРИРОВАНИЕ

```
POST /api/admin/cleanup/old-drafts?days_old=30
POST /api/admin/notifications/send-bulk
GET /api/admin/system-info
```

Организации

СПИСОК ОРГАНИЗАЦИЙ

```
GET /api/organizations/
```

СОЗДАНИЕ ОРГАНИЗАЦИИ

```
POST /api/organizations/
```

ИНФОРМАЦИЯ ОБ ОРГАНИЗАЦИИ

```
GET /api/organizations/{organization_id}
```

ИМПОРТ ИЗ ФНС

```
POST /api/organizations/fns/import/{inn}?force_update=true
```

Сотрудники

СПИСОК СОТРУДНИКОВ ОРГАНИЗАЦИИ

```
GET /api/organizations/{organization_id}/employees
```

ИНФОРМАЦИЯ О СОТРУДНИКЕ

```
GET /api/organizations/{organization_id}/employees/{employee_id}
```

СТАТИСТИКА СОТРУДНИКОВ

```
GET /api/organizations/{organization_id}/employees/stats/summary
```

ДОСТУПНЫЕ РОЛИ

```
GET /api/organizations/{organization_id}/employees/roles/available
```

Документы

СПИСОК ДОКУМЕНТОВ

```
GET /api/documents/
```

ЗАГРУЗКА ДОКУМЕНТА

```
POST /api/documents/upload
Content-Type: multipart/form-data

file: [binary]
name: "Название документа"
description: "Описание"
document_type: "instruction"
category: "safety"
is_template: true
```

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

```
GET /api/documents/{document_id}
```

ОБНОВЛЕНИЕ ДОКУМЕНТА

```
PUT /api/documents/{document_id}
```

УДАЛЕНИЕ ДОКУМЕНТА

```
DELETE /api/documents/{document_id}
```

ГЕНЕРАЦИЯ ДОКУМЕНТОВ СИЗ

```
POST /api/documents/generate/ppe-issuance/{organization_id}
```

Формы

СПИСОК ФОРМ

```
GET /api/forms/
```

СОЗДАНИЕ ФОРМЫ

```
POST /api/forms/
```

ИНФОРМАЦИЯ О ФОРМЕ

```
GET /api/forms/{form_id}
```

ОБНОВЛЕНИЕ ФОРМЫ

```
PUT /api/forms/{form_id}
```

ГЕНЕРАЦИЯ ЗАПОЛНЕННОГО ДОКУМЕНТА

```
POST /api/forms/{form_id}/generate-document
```

СКАЧИВАНИЕ ЗАПОЛНЕННОГО ДОКУМЕНТА

```
GET /api/forms/{form_id}/download
```

СИЗ (Средства индивидуальной защиты)

ДОЛЖНОСТИ

```
GET /api/ppe/positions?q=инженер&category=engineering
```

ТИПЫ ЗАЩИТЫ

```
GET /api/ppe/types
```

НАИМЕНОВАНИЯ СИЗ

```
GET /api/ppe/items?type_id=1&position_id=5
```

НОРМЫ ВЫДАЧИ

```
GET /api/ppe/norms?position_id=1&type_id=2
```

УЧЕТ ВЫДАЧИ СИЗ

```
GET /api/ppe/organizations/{organization_id}/ppe-issuance/
POST /api/ppe/organizations/{organization_id}/ppe-issuance/
PUT /api/ppe/organizations/{organization_id}/ppe-issuance/{issuance_id}
DELETE /api/ppe/organizations/{organization_id}/ppe-issuance/{issuance_id}
```

УЧЕТ ОБЕСПЕЧЕННОСТИ СИЗ

```
GET /api/ppe/organizations/{organization_id}/ppe-issue-accounts/
POST /api/ppe/organizations/{organization_id}/ppe-issue-accounts/
PUT /api/ppe/organizations/{organization_id}/ppe-issue-accounts/{account_id}
DELETE /api/ppe/organizations/{organization_id}/ppe-issue-accounts/{account_id}
```

⚠ Оценка рисков

СОЗДАНИЕ ОЦЕНКИ РИСКА

```
POST /api/risk-assessment/
```

СПИСОК ОЦЕНОК РИСКОВ

```
GET /api/risk-assessment/
```

ИНФОРМАЦИЯ ОБ ОЦЕНКЕ РИСКА

```
GET /api/risk-assessment/{assessment_id}
```

ОБНОВЛЕНИЕ ОЦЕНКИ РИСКА

```
PUT /api/risk-assessment/{assessment_id}
```

РУКОВОДЯЩИЕ ПРИНЦИПЫ

```
GET /api/risk-assessment/guidelines
```

Медицинские осмотры

СОЗДАНИЕ МЕДОСМОТРА

```
POST /api/medical-examinations/
```

СПИСОК МЕДОСМОТРОВ

```
GET /api/medical-examinations/?organization_id=1&user_id=5
```

ИНФОРМАЦИЯ О МЕДОСМОТРЕ

```
GET /api/medical-examinations/{examination_id}
```

ОБНОВЛЕНИЕ МЕДОСМОТРА

```
PUT /api/medical-examinations/{examination_id}
```

Дашборд

ДАННЫЕ ДАШБОРДА

```
GET /api/dashboard/
```

СРОЧНЫЕ УВЕДОМЛЕНИЯ

```
GET /api/dashboard/notifications
```

KPI ПОКАЗАТЕЛИ

```
GET /api/dashboard/kpi
```

ДАННЫЕ ДЛЯ ГРАФИКОВ

```
GET /api/dashboard/chart-data
```

СОБЫТИЯ

```
GET /api/dashboard/events?limit=10
```

Поддержка

СОЗДАНИЕ ОБРАЩЕНИЯ

```
POST /api/support/tickets/
```

СПИСОК ОБРАЩЕНИЙ

```
GET /api/support/tickets/
```

ИНФОРМАЦИЯ ОБ ОБРАЩЕНИИ

```
GET /api/support/tickets/{ticket_id}
```

ОБНОВЛЕНИЕ ОБРАЩЕНИЯ

```
PUT /api/support/tickets/{ticket_id}
```

ДОБАВЛЕНИЕ СООБЩЕНИЯ

```
POST /api/support/tickets/{ticket_id}/messages
```

ЗАГРУЗКА ВЛОЖЕНИЯ

```
POST /api/support/tickets/{ticket_id}/attachments
```

Файловое хранилище

ЗАГРУЗКА ФАЙЛА

```
POST /api/storage/upload?subdirectory=documents
```

ПОЛУЧЕНИЕ ФАЙЛА

```
GET /api/storage/files/{file_path}
```

ДЕРЕВО ДИРЕКТОРИЙ

```
GET /api/storage/directory-tree
```

СТРУКТУРА ДИРЕКТОРИИ

```
GET /api/storage/directory-structure?path=/documents
```

5.1.6 Примеры использования

Создание документа с шаблоном

```python
import requests

# Аутентификация
auth_response = requests.post('https://api.safety-system.com/api/auth/login', json={
    'username': 'admin',
    'password': 'password'
})
```

```
token = auth_response.json()['access_token']
headers = {'Authorization': f'Bearer {token}'}

# Загрузка документа
with open('template.xlsx', 'rb') as f:
    files = {'file': f}
    data = {
        'name': 'Инструкция по охране труда',
        'description': 'Общая инструкция по безопасности',
        'document_type': 'instruction',
        'category': 'general_safety',
        'is_template': True
    }

    response = requests.post(
        'https://api.safety-system.com/api/documents/upload',
        files=files,
        data=data,
        headers=headers
    )

document = response.json()
print(f"Документ создан с ID: {document['id']}")
```

Создание формы на основе шаблона

```
# Создание формы
form_data = {
    'document_id': document['id'],
    'form_data': {
        'employee_name': 'Иванов И.И.',
        'position': 'Инженер',
        'department': 'Производство',
        'date': '2024-01-15'
    },
    'status': 'draft'
}

response = requests.post(
    'https://api.safety-system.com/api/forms/',
    json=form_data,
    headers=headers
)

form = response.json()
print(f"Форма создана с ID: {form['id']}")
```

Генерация заполненного документа

```
# Генерация заполненного документа
response = requests.post(
    f'https://api.safety-system.com/api/forms/{form["id"]}/generate-document',
    headers=headers
)

result = response.json()
print(f"Документ сгенерирован: {result['filled_document_path']}")
```

5.1.7 Безопасность

Rate Limiting

API имеет ограничения на количество запросов: - **Аутентифицированные пользователи**: 1000 запросов в час - **Неаутентифицированные**: 100 запросов в час

Валидация данных

Все входные данные проходят валидацию: - **Pydantic модели** для структурированных данных - **Проверка типов** и форматов - **Санитизация** пользовательского ввода

CORS

Настроены правила CORS для безопасного взаимодействия с фронтендом: - **Разрешенные домены**: настраиваются в конфигурации - **Методы**: GET, POST, PUT, DELETE, OPTIONS - **Заголовки**: Authorization, Content-Type, X-Requested-With

5.1.8 Мониторинг и логирование

Логирование запросов

Все API запросы логируются: - **Время запроса** - **IP адрес клиента** - **Метод и URL** - **Статус ответа** - **Время выполнения**

Метрики

Система собирает метрики: - **Количество запросов** по эндпоинтам - **Время отклика** API - **Ошибки** и исключения - **Использование ресурсов**

5.1.9 SDK и библиотеки

Python SDK

```python
from safety_system_sdk import SafetySystemClient

client = SafetySystemClient(
    base_url='https://api.safety-system.com',
    username='your_username',
    password='your_password'
)

# Создание документа
document = client.documents.create(
    name='Test Document',
    file_path='./document.xlsx'
)

# Создание формы
form = client.forms.create(
    document_id=document.id,
    form_data={'field1': 'value1'}
)
```

JavaScript SDK

```javascript
import { SafetySystemClient } from '@safety-system/sdk';

const client = new SafetySystemClient({
  baseUrl: 'https://api.safety-system.com',
  username: 'your_username',
  password: 'your_password'
});

// Создание документа
const document = await client.documents.create({
  name: 'Test Document',
  file: fileInput.files[0]
});

// Создание формы
const form = await client.forms.create({
  documentId: document.id,
  formData: { field1: 'value1' }
});
```

5.1.10 Дополнительные ресурсы

- **OpenAPI спецификация**: `https://api.vrm-sot.ru/api/docs#/` (Swagger UI)
- **ReDoc документация**: `https://api.vrm-sot.ru/api/redoc`
- **Postman коллекция**: может быть импортирована напрямую из Swagger UI
  В Postman выберите **File → Import → Link** и укажите ссылку на OpenAPI JSON:
  `https://api.vrm-sot.ru/api/openapi.json`
- **Примеры кода**: в репозитории проекта

API документация обновляется автоматически при изменении кода. Для получения актуальной информации используйте Swagger UI.

5 февраля 2026 г.

5 февраля 2026 г.

6. База данных

6.1 Схема базы данных

6.1.1 Обзор

База данных системы охраны труда построена на PostgreSQL и содержит 33 таблицы, обеспечивающие полный функционал управления безопасностью на рабочих местах.

6.1.2 Архитектура базы данных

```
erDiagram
    safety_users ||--o{ organization_members : "участвует в"
    organizations ||--o{ organization_members : "содержит"
    organizations ||--o{ work_zones : "имеет"
    organizations ||--o{ organization_positions : "содержит"
    organizations ||--o{ ppe_issuance : "ведет учет"
    organizations ||--o{ medical_examinations : "проводит"

    safety_users ||--o{ safety_documents : "загружает"
    safety_users ||--o{ safety_filled_forms : "создает"
    safety_users ||--o{ medical_examinations : "проходит"
    safety_users ||--o{ user_work_zones : "назначен на"

    safety_documents ||--o{ safety_filled_forms : "используется в"
    safety_documents ||--o{ safety_document_access : "имеет доступ"

    positions ||--o{ issue_norms : "имеет нормы"
    protection_types ||--o{ issue_norms : "определяет тип"
    ppe_items ||--o{ issue_norms : "входит в норму"

    hazard_classifiers ||--o{ risk_measures : "требует мер"
    positions ||--o{ risk_measures : "связан с мерами"
```

6.1.3 Пользователи и аутентификация

safety_users

Основная таблица пользователей системы.

```sql
CREATE TABLE safety_users (
    id SERIAL PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    full_name VARCHAR(200) NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    employee_number VARCHAR(20) UNIQUE,
    position VARCHAR(100),
    department VARCHAR(100),
    role VARCHAR(20) DEFAULT 'user',
    is_active BOOLEAN DEFAULT TRUE,
    hire_date TIMESTAMP,
    password_set BOOLEAN DEFAULT FALSE,
    first_login_completed BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Индексы
CREATE INDEX idx_users_username ON safety_users(username);
CREATE INDEX idx_users_email ON safety_users(email);
CREATE INDEX idx_users_employee_number ON safety_users(employee_number);
CREATE INDEX idx_users_role ON safety_users(role);
```

Описание полей: - `id` - уникальный идентификатор пользователя - `username` - имя пользователя для входа - `email` - электронная почта - `full_name` - полное имя пользователя - `password_hash` - хеш пароля - `employee_number` - табельный номер сотрудника - `position` - должность - `department` - отдел - `role` - роль в системе (admin, manager, user) - `is_active` - активность пользователя - `hire_date` - дата приема на работу - `password_set` - установлен ли пароль - `first_login_completed` - завершен ли первый вход

6.1.4 Организации и участники

organizations

Таблица организаций с полной информацией из ФНС.

```
CREATE TABLE organizations (
    id SERIAL PRIMARY KEY,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    -- Основные данные
    type VARCHAR NOT NULL, -- ИП или ЮЛ
    inn VARCHAR(12) UNIQUE NOT NULL, -- ИНН (10 для ЮЛ, 12 для ИП)
    ogrn VARCHAR(15), -- ОГРН/ОГРНИП
    registration_date TIMESTAMP,
    status VARCHAR,

    -- Данные для ИП
    full_name VARCHAR,
    short_name VARCHAR,
    gender VARCHAR,
    citizenship VARCHAR,

    -- Адрес
    postal_code VARCHAR,
    region_code VARCHAR,
    full_address TEXT,
    address_date TIMESTAMP,

    -- Контактные данные
    email VARCHAR,
    contacts JSONB,

    -- Коды статистики
    okpo VARCHAR,
    oktmo VARCHAR,
    okfs VARCHAR,
    okogu VARCHAR,

    -- Налоговая информация
    tax_registration_date TIMESTAMP,
    tax_office_reg VARCHAR,
    tax_office_reg_date TIMESTAMP,
    tax_office VARCHAR,
    tax_office_date TIMESTAMP,

    -- Информация ПФР
    pf_registration_number VARCHAR,
    pf_registration_date TIMESTAMP,
    pf_code VARCHAR,

    -- Виды деятельности
    main_activity_code VARCHAR,
    main_activity_text TEXT,
    main_activity_date TIMESTAMP,
    additional_activities JSONB,

    -- История изменений и события
    history JSONB,
    events JSONB,

    -- Система налогообложения
    tax_system VARCHAR,
    tax_system_date TIMESTAMP
);

-- Индексы
CREATE INDEX idx_organizations_inn ON organizations(inn);
CREATE INDEX idx_organizations_type ON organizations(type);
```

organization_members

Связь пользователей с организациями и их ролями.

```
CREATE TABLE organization_members (
    id SERIAL PRIMARY KEY,
    user_id INTEGER NOT NULL REFERENCES safety_users(id),
    organization_id INTEGER NOT NULL REFERENCES organizations(id),
    role VARCHAR(20) NOT NULL, -- owner, admin, manager, specialist
    is_active BOOLEAN DEFAULT TRUE,
    invited_by INTEGER REFERENCES safety_users(id),
    invited_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    accepted_at TIMESTAMP,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```
-- Индексы
CREATE INDEX idx_org_members_user_id ON organization_members(user_id);
CREATE INDEX idx_org_members_org_id ON organization_members(organization_id);
CREATE INDEX idx_org_members_role ON organization_members(role);
```

organization_invitations

Приглашения пользователей в организации.

```
CREATE TABLE organization_invitations (
    id SERIAL PRIMARY KEY,
    organization_id INTEGER NOT NULL REFERENCES organizations(id),
    email VARCHAR(100) NOT NULL,
    role VARCHAR(20) NOT NULL,
    invited_by INTEGER NOT NULL REFERENCES safety_users(id),
    token VARCHAR(255) UNIQUE NOT NULL,
    expires_at TIMESTAMP NOT NULL,
    is_accepted BOOLEAN DEFAULT FALSE,
    accepted_at TIMESTAMP,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

6.1.5 Документооборот

safety_documents

Документы по охране труда.

```
CREATE TABLE safety_documents (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    description TEXT,
    file_path VARCHAR(500) NOT NULL,
    file_type VARCHAR(10) NOT NULL, -- xlsx, docx, pdf
    file_size INTEGER,
    original_filename VARCHAR(255),
    document_type VARCHAR(50), -- instruction, form, certificate
    category VARCHAR(100), -- fire_safety, electrical_safety
    is_template BOOLEAN DEFAULT FALSE,
    template_fields JSONB,
    status VARCHAR(20) DEFAULT 'active', -- active, archived, draft
    uploaded_by INTEGER REFERENCES safety_users(id),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Индексы
CREATE INDEX idx_documents_uploaded_by ON safety_documents(uploaded_by);
CREATE INDEX idx_documents_type ON safety_documents(document_type);
CREATE INDEX idx_documents_category ON safety_documents(category);
CREATE INDEX idx_documents_status ON safety_documents(status);
```

safety_filled_forms

Заполненные формы на основе шаблонов.

```
CREATE TABLE safety_filled_forms (
    id SERIAL PRIMARY KEY,
    user_id INTEGER NOT NULL REFERENCES safety_users(id),
    document_id INTEGER NOT NULL REFERENCES safety_documents(id),
    form_data JSONB NOT NULL,
    status VARCHAR(20) DEFAULT 'draft', -- draft, submitted, approved, rejected
    submitted_at TIMESTAMP,
    reviewed_by INTEGER REFERENCES safety_users(id),
    reviewed_at TIMESTAMP,
    review_notes TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Индексы
CREATE INDEX idx_forms_user_id ON safety_filled_forms(user_id);
CREATE INDEX idx_forms_document_id ON safety_filled_forms(document_id);
CREATE INDEX idx_forms_status ON safety_filled_forms(status);
```

safety_document_access_control

Контроль доступа к документам.

```
CREATE TABLE safety_document_access_control (
    id SERIAL PRIMARY KEY,
    document_id INTEGER NOT NULL REFERENCES safety_documents(id),
    access_level VARCHAR(20) NOT NULL DEFAULT 'organization', -- public, system, organization
    organization_id INTEGER REFERENCES organizations(id),
    system_only BOOLEAN DEFAULT FALSE,
    allow_read BOOLEAN DEFAULT TRUE,
    allow_download BOOLEAN DEFAULT TRUE,
    allow_preview BOOLEAN DEFAULT TRUE,
    created_by INTEGER NOT NULL REFERENCES safety_users(id),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_by INTEGER REFERENCES safety_users(id)
);
```

6.1.6 СИЗ (Средства индивидуальной защиты)

positions

Справочник должностей.

```
CREATE TABLE positions (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    category VARCHAR(100) NOT NULL,
    activity_area VARCHAR(255),
    danger TEXT,
    risk_management TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

protection_types

Типы средств защиты.

```
CREATE TABLE protection_types (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255) UNIQUE NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

ppe_items

Наименования СИЗ.

```
CREATE TABLE ppe_items (
    id SERIAL PRIMARY KEY,
    name TEXT UNIQUE NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

issue_norms

Нормы выдачи СИЗ.

```
CREATE TABLE issue_norms (
    id SERIAL PRIMARY KEY,
    position_id INTEGER NOT NULL REFERENCES positions(id),
    protection_type_id INTEGER NOT NULL REFERENCES protection_types(id),
    ppe_item_id INTEGER NOT NULL REFERENCES ppe_items(id),
    norm_text VARCHAR(255),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

ppe_issuance

Журнал выдачи СИЗ.

```
CREATE TABLE ppe_issuance (
    id SERIAL PRIMARY KEY,
    organization_id INTEGER NOT NULL REFERENCES organizations(id),
    employee_name VARCHAR(255),
    position_name VARCHAR(255),
    protection_type VARCHAR(255),
    ppe_item TEXT,
    ppe_item_id INTEGER REFERENCES ppe_items(id),
    norm_text VARCHAR(255),
    quantity INTEGER DEFAULT 1,
    unit VARCHAR(50),
    employee_signature TEXT,
    issuer_name VARCHAR(255),
    issuer_signature TEXT,
    issue_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    expiry_date TIMESTAMP,
    recipient_user_id INTEGER REFERENCES safety_users(id),
    comment TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Индексы
CREATE INDEX idx_ppe_issuance_org_id ON ppe_issuance(organization_id);
CREATE INDEX idx_ppe_issuance_recipient ON ppe_issuance(recipient_user_id);
CREATE INDEX idx_ppe_issuance_expiry ON ppe_issuance(expiry_date);
```

6.1.7 ⚠ Опасности и риски

hazard_classifiers

Классификатор опасностей.

```
CREATE TABLE hazard_classifiers (
    id SERIAL PRIMARY KEY,
    number VARCHAR(20),
    code VARCHAR(20),
    name TEXT,
    event TEXT,
    profession TEXT,
    work_types TEXT,
    equipment TEXT,
    materials TEXT,
    source_sheet VARCHAR(50),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

risk_measures

Мероприятия по снижению рисков.

```
CREATE TABLE risk_measures (
    id SERIAL PRIMARY KEY,
    hazard_id INTEGER REFERENCES hazard_classifiers(id),
    position_id INTEGER REFERENCES positions(id),
    title VARCHAR(500) NOT NULL,
    description TEXT,
    responsible VARCHAR(255),
    due_date TIMESTAMP,
    status VARCHAR(50) DEFAULT 'planned', -- planned, in_progress, done, canceled
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

risk_assessments

Оценки профессиональных рисков.

```
CREATE TABLE risk_assessments (
    id SERIAL PRIMARY KEY,
    organization_id INTEGER NOT NULL REFERENCES organizations(id),
    position_id INTEGER NOT NULL REFERENCES organization_positions(id),
    hazard_id INTEGER NOT NULL REFERENCES hazard_classifiers(id),
    severity INTEGER NOT NULL,
    probability INTEGER NOT NULL,
    k1_coefficient FLOAT DEFAULT 1.0,
    k2_coefficient FLOAT DEFAULT 1.0,
    k3_coefficient FLOAT DEFAULT 1.0,
    base_risk_score INTEGER,
    final_risk_score FLOAT,
    risk_level VARCHAR(50),
```

```
    risk_zone VARCHAR(20),
    control_measures TEXT,
    responsible_person VARCHAR(255),
    due_date TIMESTAMP,
    status VARCHAR(50) DEFAULT 'planned',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Индексы
CREATE INDEX idx_risk_assessments_org_id ON risk_assessments(organization_id);
CREATE INDEX idx_risk_assessments_position_id ON risk_assessments(position_id);
CREATE INDEX idx_risk_assessments_hazard_id ON risk_assessments(hazard_id);
```

6.1.8 Медицинские осмотры

medical_examinations

Медицинские осмотры сотрудников.

```
CREATE TABLE medical_examinations (
    id SERIAL PRIMARY KEY,
    user_id INTEGER NOT NULL REFERENCES safety_users(id),
    organization_id INTEGER NOT NULL REFERENCES organizations(id),
    examination_type VARCHAR(50) NOT NULL, -- preliminary, periodic, extraordinary
    examination_date TIMESTAMP NOT NULL,
    next_examination_date TIMESTAMP,
    result VARCHAR(50) NOT NULL, -- fit, unfit, conditionally_fit
    restrictions TEXT,
    recommendations TEXT,
    medical_organization VARCHAR(255),
    doctor_name VARCHAR(200),
    certificate_number VARCHAR(100),
    notes TEXT,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Индексы
CREATE INDEX idx_med_exams_user_id ON medical_examinations(user_id);
CREATE INDEX idx_med_exams_org_id ON medical_examinations(organization_id);
CREATE INDEX idx_med_exams_type ON medical_examinations(examination_type);
```

6.1.9 Рабочие зоны

work_zones

Рабочие зоны организации.

```
CREATE TABLE work_zones (
    id SERIAL PRIMARY KEY,
    organization_id INTEGER NOT NULL REFERENCES organizations(id),
    name VARCHAR(200) NOT NULL,
    description TEXT,
    features TEXT,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Индексы
CREATE INDEX idx_work_zones_org_id ON work_zones(organization_id);
```

user_work_zones

Связь пользователей с рабочими зонами.

```
CREATE TABLE user_work_zones (
    id SERIAL PRIMARY KEY,
    user_id INTEGER NOT NULL REFERENCES safety_users(id),
    work_zone_id INTEGER NOT NULL REFERENCES work_zones(id),
    organization_id INTEGER NOT NULL REFERENCES organizations(id),
    start_date TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    end_date TIMESTAMP,
    role_in_zone VARCHAR(200),
    is_primary BOOLEAN DEFAULT FALSE,
    notes TEXT,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
```

```
);

-- Индексы
CREATE INDEX idx_user_work_zones_user_id ON user_work_zones(user_id);
CREATE INDEX idx_user_work_zones_zone_id ON user_work_zones(work_zone_id);
CREATE INDEX idx_user_work_zones_org_id ON user_work_zones(organization_id);
```

6.1.10 Система поддержки

support_tickets

Обращения в поддержку.

```
CREATE TABLE support_tickets (
    id SERIAL PRIMARY KEY,
    organization_id INTEGER NOT NULL REFERENCES organizations(id),
    created_by INTEGER NOT NULL REFERENCES safety_users(id),
    title VARCHAR(255) NOT NULL,
    description TEXT NOT NULL,
    status VARCHAR(50) DEFAULT 'new', -- new, in_progress, resolved, closed
    contact_name VARCHAR(200) NOT NULL,
    contact_email VARCHAR(100) NOT NULL,
    contact_phone VARCHAR(20),
    organization_name VARCHAR(255) NOT NULL,
    tariff_plan VARCHAR(100),
    priority VARCHAR(20) DEFAULT 'medium', -- low, medium, high, urgent
    category VARCHAR(100),
    tags JSONB,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    resolved_at TIMESTAMP,
    closed_at TIMESTAMP
);
```

6.1.11 Уведомления

safety_notifications

Уведомления пользователей.

```
CREATE TABLE safety_notifications (
    id SERIAL PRIMARY KEY,
    user_id INTEGER NOT NULL REFERENCES safety_users(id),
    title VARCHAR(255) NOT NULL,
    message TEXT NOT NULL,
    notification_type VARCHAR(50), -- reminder, alert, info
    is_read BOOLEAN DEFAULT FALSE,
    related_document_id INTEGER REFERENCES safety_documents(id),
    related_form_id INTEGER REFERENCES safety_filled_forms(id),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

6.1.12 Индексы и оптимизация

Основные индексы

```
-- Составные индексы для частых запросов
CREATE INDEX idx_org_members_user_org ON organization_members(user_id, organization_id);
CREATE INDEX idx_forms_user_status ON safety_filled_forms(user_id, status);
CREATE INDEX idx_ppe_issuance_org_date ON ppe_issuance(organization_id, issue_date);
CREATE INDEX idx_med_exams_user_type ON medical_examinations(user_id, examination_type);

-- Индексы для поиска
CREATE INDEX idx_users_full_name_gin ON safety_users USING GIN (to_tsvector('russian', full_name));
CREATE INDEX idx_organizations_full_name_gin ON organizations USING GIN (to_tsvector('russian', full_name));
CREATE INDEX idx_documents_name_gin ON safety_documents USING GIN (to_tsvector('russian', name));
```

JSONB индексы

```
-- Индексы для JSONB полей
CREATE INDEX idx_organizations_contacts_gin ON organizations USING GIN (contacts);
CREATE INDEX idx_forms_data_gin ON safety_filled_forms USING GIN (form_data);
CREATE INDEX idx_documents_template_fields_gin ON safety_documents USING GIN (template_fields);
```

6.1.13 Миграции

Alembic

Система использует Alembic для управления миграциями базы данных.

```
# Пример миграции
"""Add employee number to users

Revision ID: abc123def456
Revises: previous_revision
Create Date: 2024-01-15 10:30:00.000000

"""
from alembic import op
import sqlalchemy as sa

def upgrade():
    op.add_column('safety_users', sa.Column('employee_number', sa.String(20), nullable=True))
    op.create_index('idx_users_employee_number', 'safety_users', ['employee_number'], unique=True)

def downgrade():
    op.drop_index('idx_users_employee_number', table_name='safety_users')
    op.drop_column('safety_users', 'employee_number')
```

6.1.14 Производительность

Партиционирование

Для больших таблиц используется партиционирование:

```
-- Партиционирование по дате для логов
CREATE TABLE safety_audit_logs (
    id BIGSERIAL,
    user_id INTEGER,
    action VARCHAR(100),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
) PARTITION BY RANGE (created_at);

-- Создание партиций по месяцам
CREATE TABLE safety_audit_logs_2024_01 PARTITION OF safety_audit_logs
    FOR VALUES FROM ('2024-01-01') TO ('2024-02-01');
```

Материализованные представления

Для сложных запросов используются материализованные представления:

```
-- Статистика по организациям
CREATE MATERIALIZED VIEW organization_stats AS
SELECT
    o.id,
    o.name,
    COUNT(om.user_id) as user_count,
    COUNT(d.id) as document_count,
    COUNT(f.id) as form_count
FROM organizations o
LEFT JOIN organization_members om ON o.id = om.organization_id
LEFT JOIN safety_documents d ON o.id = d.organization_id
LEFT JOIN safety_filled_forms f ON o.id = f.organization_id
GROUP BY o.id, o.name;

-- Обновление материализованного представления
REFRESH MATERIALIZED VIEW organization_stats;
```

6.1.15 Безопасность

Row Level Security (RLS)

Для обеспечения изоляции данных организаций:

```
-- Включение RLS для таблицы документов
ALTER TABLE safety_documents ENABLE ROW LEVEL SECURITY;

-- Политика доступа для пользователей организации
CREATE POLICY org_documents_policy ON safety_documents
    FOR ALL TO authenticated_users
    USING (
        EXISTS (
```

```
            SELECT 1 FROM organization_members om
            WHERE om.user_id = current_user_id()
            AND om.organization_id = (
                SELECT organization_id FROM safety_documents
                WHERE id = safety_documents.id
            )
        )
    );
```

Шифрование

Чувствительные данные шифруются:

```
-- Функция для шифрования
CREATE OR REPLACE FUNCTION encrypt_sensitive_data(data TEXT)
RETURNS TEXT AS $$
BEGIN
    RETURN pgp_sym_encrypt(data, current_setting('app.encryption_key'));
END;
$$ LANGUAGE plpgsql;

-- Функция для расшифровки
CREATE OR REPLACE FUNCTION decrypt_sensitive_data(encrypted_data TEXT)
RETURNS TEXT AS $$
BEGIN
    RETURN pgp_sym_decrypt(encrypted_data, current_setting('app.encryption_key'));
END;
$$ LANGUAGE plpgsql;
```

Схема базы данных оптимизирована для производительности, безопасности и масштабируемости.

5 февраля 2026 г.

5 февраля 2026 г.

7. ? Часто задаваемые вопросы (FAQ)

7.1 Аутентификация и доступ

7.1.1 Как войти в систему?

1. Перейдите на страницу входа
2. Введите ваше имя пользователя и пароль
3. Нажмите кнопку "Войти"

Если у вас нет учетной записи, обратитесь к администратору системы.

7.1.2 Что делать, если забыл пароль?

Обратитесь к администратору системы для сброса пароля. Администратор может: - Сбросить пароль и отправить временный пароль - Создать новую учетную запись - Проверить статус вашей учетной записи

7.1.3 Как изменить пароль?

1. Войдите в систему
2. Перейдите в раздел "Профиль" или "Настройки"
3. Выберите "Изменить пароль"
4. Введите текущий пароль и новый пароль
5. Подтвердите изменения

7.1.4 Что означают роли в системе?

- **admin** - администратор системы (полный доступ ко всем функциям)
- **owner** - владелец организации (полный доступ к организации)
- **manager** - менеджер организации (управление процессами и отчетами)
- **specialist** - специалист (базовый доступ к функциям)

7.2 Организации

7.2.1 Как зарегистрировать организацию?

1. Обратитесь к администратору системы
2. Предоставьте ИНН организации
3. Администратор импортирует данные из ФНС
4. Вам будет предоставлен доступ к организации

7.2.2 Как добавить сотрудников в организацию?

1. Войдите в систему как владелец или администратор организации
2. Перейдите в раздел "Сотрудники"
3. Нажмите "Добавить сотрудника"
4. Заполните данные сотрудника
5. Назначьте роль и отправьте приглашение

7.2.3 Как изменить роль сотрудника?

1. Перейдите в раздел "Сотрудники"
2. Найдите нужного сотрудника
3. Нажмите "Изменить роль"
4. Выберите новую роль
5. Подтвердите изменения

7.2.4 Что такое рабочие зоны?

Рабочие зоны - это физические или логические области в организации, где выполняются определенные виды работ. Они используются для: - Оценки рисков по зонам - Назначения сотрудников на конкретные участки - Учета СИЗ по рабочим местам - Планирования мероприятий по безопасности

7.3 Документы и формы

7.3.1 Какие форматы файлов поддерживаются?

Система поддерживает следующие форматы: - **Excel**: .xlsx, .xls - **Word**: .docx, .doc - **PDF**: .pdf

7.3.2 Как создать шаблон документа?

1. Подготовьте документ с полями для заполнения в фигурных скобках: `{поле_имя}`
2. Загрузите документ через раздел "Документы"
3. Отметьте "Это шаблон"
4. Система автоматически извлечет поля для заполнения

7.3.3 Как заполнить форму на основе шаблона?

1. Перейдите в раздел "Формы"
2. Нажмите "Создать форму"
3. Выберите шаблон документа
4. Заполните все обязательные поля
5. Сохраните как черновик или отправьте на рассмотрение

7.3.4 Как скачать заполненный документ?

1. Найдите нужную форму в разделе "Формы"
2. Нажмите "Сгенерировать документ"
3. Дождитесь обработки
4. Скачайте готовый документ

7.3.5 Как настроить доступ к документам?

1. Откройте документ
2. Перейдите в "Настройки доступа"
3. Выберите уровень доступа:
4. **Публичный** - доступен всем организациям
5. **Системный** - только для администраторов

6. **Организационный** - только для вашей организации
7. Настройте права (чтение, скачивание, предварительный просмотр)

7.4 СИЗ и безопасность

7.4.1 Как добавить нормы выдачи СИЗ?

1. Перейдите в раздел "СИЗ" → "Нормы выдачи"
2. Нажмите "Добавить норму"
3. Выберите должность
4. Выберите тип защиты и конкретное СИЗ
5. Укажите норму выдачи
6. Сохраните

7.4.2 Как вести учет выдачи СИЗ?

1. Перейдите в раздел "СИЗ" → "Учет выдачи"
2. Нажмите "Новая выдача"
3. Выберите сотрудника
4. Выберите СИЗ из норм или добавьте вручную
5. Укажите количество и срок действия
6. Добавьте подписи получателя и выдающего

7.4.3 Как настроить напоминания о замене СИЗ?

1. Перейдите в настройки уведомлений
2. Включите уведомления о СИЗ
3. Укажите период предупреждения (например, за 30 дней)
4. Выберите способ уведомления (email, в системе)

7.4.4 Как провести оценку рисков?

1. Перейдите в раздел "Оценка рисков"
2. Нажмите "Новая оценка"
3. Выберите должность и рабочую зону
4. Выберите опасности из классификатора
5. Оцените тяжесть последствий (1-5)
6. Оцените вероятность возникновения (1-5)
7. Система автоматически рассчитает уровень риска
8. Добавьте мероприятия по снижению риска

7.4.5 Что означают уровни риска?

- **Низкий** (1-6 баллов) - приемлемый риск
- **Средний** (7-12 баллов) - требует внимания
- **Высокий** (13-20 баллов) - требует немедленных мер
- **Критический** (21-25 баллов) - недопустимый риск

7.5 Медицинские осмотры

7.5.1 Как добавить медицинский осмотр?

1. Перейдите в раздел "Медицинские осмотры"
2. Нажмите "Добавить осмотр"
3. Выберите сотрудника
4. Укажите тип осмотра (предварительный, периодический, внеочередной)
5. Заполните результаты осмотра
6. Укажите дату следующего осмотра

7.5.2 Как настроить напоминания о медосмотрах?

1. Перейдите в настройки уведомлений
2. Включите уведомления о медосмотрах
3. Укажите период предупреждения
4. Выберите ответственных за получение уведомлений

7.5.3 Какие типы медосмотров поддерживаются?

- **Предварительный** - при приеме на работу
- **Периодический** - регулярные осмотры
- **Внеочередной** - по показаниям

7.6 Отчеты и аналитика

7.6.1 Как создать отчет?

1. Перейдите в раздел "Отчеты"
2. Выберите тип отчета
3. Настройте параметры (период, фильтры)
4. Нажмите "Создать отчет"
5. Скачайте или отправьте по email

7.6.2 Как настроить автоматические отчеты?

1. Перейдите в "Настройки" → "Автоматические отчеты"
2. Выберите тип отчета
3. Настройте расписание (еженедельно, ежемесячно)
4. Укажите получателей
5. Сохраните настройки

7.6.3 Что показывают KPI на дашборде?

- **Инциденты** - количество и тренд инцидентов
- **Проверки** - процент выполненных проверок
- **Травматизм** - коэффициент травматизма
- **Обучение** - процент обученных сотрудников

- **СИЗ** - процент обеспеченности СИЗ

7.7 Поддержка

7.7.1 Как создать обращение в поддержку?

1. Перейдите в раздел "Поддержка"
2. Нажмите "Создать обращение"
3. Заполните форму:
4. Тема обращения
5. Описание проблемы
6. Приоритет
7. Категория
8. При необходимости прикрепите файлы
9. Отправьте обращение

7.7.2 Как отследить статус обращения?

1. Перейдите в раздел "Поддержка"
2. Найдите ваше обращение в списке
3. Статус отображается в колонке "Статус"
4. Для получения подробной информации откройте обращение

7.7.3 Какие категории обращений доступны?

- Технические проблемы
- Вопросы по функциональности
- Запрос новых функций
- Обучение пользователей
- Другие вопросы

7.8 Технические вопросы

7.8.1 Какие браузеры поддерживаются?

Система поддерживает современные браузеры: - Chrome 90+ - Firefox 88+ - Safari 14+ - Edge 90+

7.8.2 Можно ли использовать систему на мобильных устройствах?

Да, система адаптирована для мобильных устройств и имеет мобильное приложение для iOS и Android.

7.8.3 Как часто обновляется система?

Система обновляется регулярно. О плановых обновлениях пользователи уведомляются заранее через систему уведомлений.

7.8.4 Что делать, если система работает медленно?

1. Проверьте скорость интернет-соединения

2. Очистите кэш браузера
3. Попробуйте другой браузер
4. Обратитесь в поддержку, если проблема сохраняется

7.8.5 Как обеспечить безопасность данных?

Система использует современные методы защиты: - Шифрование данных при передаче (HTTPS) - Хеширование паролей - Контроль доступа на уровне ролей - Регулярное резервное копирование - Аудит всех действий пользователей

7.9 Мобильное приложение

7.9.1 Как скачать мобильное приложение?

Мобильное приложение доступно в: - App Store (iOS) - Google Play (Android)

7.9.2 Какие функции доступны в мобильном приложении?

- Просмотр документов
- Заполнение форм
- Получение уведомлений
- Просмотр личной информации
- Работа в офлайн режиме

7.9.3 Как синхронизировать данные в мобильном приложении?

Данные синхронизируются автоматически при подключении к интернету. Для принудительной синхронизации нажмите кнопку "Синхронизировать" в меню приложения.

7.10 Интеграции

7.10.1 Можно ли интегрировать систему с 1С?

Да, система предоставляет API для интеграции с 1С и другими учетными системами. Обратитесь в поддержку для получения технической документации по интеграции.

7.10.2 Как импортировать данные из Excel?

1. Подготовьте Excel файл в соответствии с шаблоном
2. Перейдите в соответствующий раздел системы
3. Нажмите "Импорт из Excel"
4. Выберите файл и настройте параметры импорта
5. Проверьте предварительный просмотр
6. Подтвердите импорт

7.10.3 Можно ли экспортировать данные из системы?

Да, большинство разделов системы поддерживают экспорт данных в различных форматах (Excel, PDF, CSV).

Если вы не нашли ответ на свой вопрос, обратитесь в службу поддержки через систему или по email: support@safety-system.com

5 февраля 2026 г.

5 февраля 2026 г.

8. История изменений

8.1 [1.2.0] - 2024-01-15

8.1.1 ✨ Новые функции

- **Медицинские осмотры**: добавлен полный функционал учета медицинских осмотров сотрудников
- **Назначения на рабочие зоны**: возможность назначения сотрудников на конкретные рабочие зоны
- **Расширенная аналитика**: новые KPI показатели и графики на дашборде
- **Автоматические уведомления**: система уведомлений о просроченных медосмотрах и СИЗ
- **Мобильное приложение**: выпущена первая версия мобильного приложения

8.1.2 Улучшения

- **Производительность**: оптимизированы запросы к базе данных, улучшена скорость работы
- **Пользовательский интерфейс**: обновлен дизайн дашборда и форм
- **Безопасность**: усилена защита данных, добавлено логирование действий
- **API**: расширена функциональность API, добавлены новые эндпоинты

8.1.3 Исправления

- Исправлена ошибка с загрузкой больших файлов
- Устранена проблема с отображением русских символов в отчетах
- Исправлена ошибка валидации форм с длинными текстами
- Устранена проблема с синхронизацией данных между организациями

8.1.4 Изменения в базе данных

- Добавлена таблица `medical_examinations`
- Добавлена таблица `user_work_zones`
- Добавлено поле `employee_number` в таблицу `safety_users`
- Добавлено поле `features` в таблицу `work_zones`

8.2 [1.1.0] - 2023-12-01

8.2.1 ✨ Новые функции

- **Система поддержки**: полнофункциональная система обращений в поддержку
- **Email уведомления**: настройка и отправка email уведомлений
- **Файловое хранилище**: интеграция с SFTP для хранения файлов
- **Расширенная отчетность**: новые типы отчетов и экспорт данных
- **Контроль доступа**: детальная настройка прав доступа к документам

8.2.2 Улучшения

- **Импорт из ФНС**: улучшена стабильность импорта данных организаций
- **Оценка рисков**: добавлены коэффициенты K1, K2, K3 для точного расчета

- **СИЗ**: расширен функционал учета средств индивидуальной защиты
- **Поиск**: улучшен полнотекстовый поиск по документам

8.2.3 Исправления

- Исправлена ошибка с отображением дат в разных часовых поясах
- Устранена проблема с кэшированием данных пользователей
- Исправлена ошибка валидации ИНН для ИП
- Устранена проблема с загрузкой шаблонов Word

8.2.4 Изменения в базе данных

- Добавлена таблица `support_tickets`
- Добавлена таблица `support_ticket_messages`
- Добавлена таблица `support_ticket_attachments`
- Добавлена таблица `safety_document_access_control`
- Добавлена таблица `ppe_issue_accounts`

8.3 [1.0.0] - 2023-10-01

8.3.1 Первый релиз

- **Базовая функциональность**: управление пользователями, организациями, документами
- **Документооборот**: загрузка, создание шаблонов, заполнение форм
- **Оценка рисков**: матричный метод 5x5 с классификатором опасностей
- **СИЗ**: справочники, нормы выдачи, учет выдачи
- **Аналитика**: базовые отчеты и статистика
- **API**: RESTful API с документацией Swagger

8.3.2 Архитектура

- **Backend**: Python FastAPI + SQLAlchemy + PostgreSQL
- **Frontend**: React + TypeScript + Vite
- **Аутентификация**: JWT токены
- **Файловое хранилище**: локальное хранилище

8.3.3 База данных

- 25 основных таблиц
- Система ролей и разрешений
- Аудит действий пользователей
- Индексы для оптимизации производительности

8.4 [0.9.0] - 2023-09-15

8.4.1 Бета-версия

- **Тестирование**: закрытое тестирование с ограниченным кругом пользователей

- **Обратная связь**: сбор отзывов и предложений пользователей
- **Исправления**: устранение критических ошибок
- **Оптимизация**: улучшение производительности

8.4.2 Основные исправления

- Исправлена ошибка с аутентификацией пользователей
- Устранена проблема с загрузкой файлов
- Исправлена ошибка расчета рисков
- Улучшена стабильность работы с большими объемами данных

8.5 [0.8.0] - 2023-08-01

8.5.1 Альфа-версия

- **Разработка**: активная разработка основных функций
- **Тестирование**: внутреннее тестирование команды разработки
- **Интеграция**: подключение внешних сервисов (ФНС API)
- **Документация**: создание технической документации

8.5.2 ✨ Реализованные функции

- Базовая аутентификация и авторизация
- Управление пользователями и организациями
- Загрузка и обработка документов
- Создание и заполнение форм
- Базовые отчеты и статистика

8.6 [0.7.0] - 2023-07-01

8.6.1 Разработка MVP

- **Архитектура**: проектирование архитектуры системы
- **База данных**: создание схемы базы данных
- **API**: разработка основных эндпоинтов
- **Frontend**: создание базового пользовательского интерфейса

8.6.2 Планирование

- Определение требований к системе
- Анализ бизнес-процессов
- Выбор технологического стека
- Создание технического задания

8.7 Планы на будущее

8.7.1 [1.3.0] - Планируется на Q2 2024

- **ИИ для анализа рисков**: автоматическая оценка рисков с использованием машинного обучения
- **Интеграция с IoT**: подключение датчиков и устройств мониторинга
- **Расширенная аналитика**: предиктивная аналитика и прогнозирование
- **Многоязычность**: поддержка английского языка

8.7.2 [1.4.0] - Планируется на Q3 2024

- **Платформа обучения**: встроенная система обучения сотрудников
- **Интеграция с видеонаблюдением**: анализ видео для выявления нарушений
- **Мобильные уведомления**: push-уведомления в мобильном приложении
- **Расширенная интеграция**: подключение к системам HR и ERP

8.7.3 [2.0.0] - Планируется на Q4 2024

- **Микросервисная архитектура**: переход на микросервисы
- **Облачная платформа**: SaaS версия системы
- **API Marketplace**: экосистема интеграций с партнерами
- **Глобальная локализация**: поддержка международных стандартов

8.8 Процесс обновлений

8.8.1 Плановые обновления

- **Критические исправления**: в течение 24 часов
- **Минорные обновления**: еженедельно
- **Мажорные обновления**: ежемесячно
- **Уведомления**: пользователи уведомляются за 48 часов до обновления

8.8.2 Уведомления об обновлениях

- Email уведомления администраторам
- Уведомления в системе
- Обновления в мобильном приложении
- Публикация в блоге и социальных сетях

8.8.3 Обратная совместимость

- API версионирование для обеспечения совместимости
- Миграции базы данных без потери данных
- Поддержка старых версий клиентов в течение 6 месяцев
- Документирование всех изменений

8.9 Статистика разработки

8.9.1 Команда разработки

- **Backend разработчики**: 3 человека
- **Frontend разработчики**: 2 человека
- **DevOps инженеры**: 1 человек
- **QA тестировщики**: 2 человека
- **Аналитики**: 1 человек

8.9.2 Метрики качества

- **Покрытие тестами**: 85%
- **Время отклика API**: < 200ms
- **Доступность системы**: 99.9%
- **Количество багов**: < 5 критических в месяц

8.9.3 Пользователи

- **Активные организации**: 85
- **Активные пользователи**: 1,250
- **Загруженные документы**: 3,400
- **Созданные формы**: 5,600

История изменений ведется с момента начала разработки системы. Для получения подробной информации об изменениях обращайтесь к команде разработки.

5 февраля 2026 г.

5 февраля 2026 г.

9. Скачать документацию в PDF

Документация доступна для скачивания в формате PDF.

9.1 Прямая ссылка

 Скачать полную документацию (PDF)

9.2 Содержание PDF

PDF версия включает: - Полную структуру документации - Все разделы и подразделы - Код примеры и диаграммы - Навигацию по разделам

9.3 Примечание

PDF версия генерируется автоматически при каждой сборке документации и всегда содержит актуальную версию.

5 февраля 2026 г.

5 февраля 2026 г.