РУКОВОДСТВО АДМИНИСТРАТОРА

Обзор системы

Task Management System — серверное приложение для управления задачами с REST API, аутентификацией через Telegram, кэшированием и поддержкой WebSocket.

Основные возможности:

- Управление задачами через REST API
- Аутентификация через Telegram
- Ролевая модель доступа
- Экспорт данных в CSV
- AI-анализ задач (LLM)
- WebSocket для real-time уведомлений
- Планировщик задач (Cron)
- Логирование и мониторинг

Установка и настройка

Требования

- Python 3.8+
- Доступ к Telegram Bot API (для аутентификации)
- OpenAI API ключ (для LLM функций, опционально)

Шаги установки

1. **Клонирование и настройка окружения:**

```
git clone <repository-url>
cd task-management-system
python -m venv venv
source venv/bin/activate  # Linux/Mac
# или venv\Scripts\activate  # Windows
pip install -r requirements.txt
```

2. **Настройка переменных окружения:**

Создайте файл .env в корневой директории:

```
FLASK_SECRET_KEY=your-secure-secret-key-here
TELEGRAM_BOT_TOKEN=your-telegram-bot-token
OPENAI_API_KEY=your-openai-api-key  # опционально
```

3. **Настройка конфигурации:**

Отредактируйте config.yaml согласно требованиям вашего окружения.

4. **Создание необходимых директорий:**

```
mkdir -p data logs
```

5. **Запуск сервера:**

```
python server.py
```

Конфигурация

Основные параметры конфигурации

Безопасность (security:)

```
security:
  enabled: false  # Включение/отключение системы безопасности
  validation_method: telegram_username  # Метод валидации пользователей
  session_timeout_hours: 1  # Таймаут сессии в часах
  refresh_token_days: 7  # Срок действия refresh токена в днях
  admin_only_endpoints:  # Эндпоинты только для администраторов
    - /api/users
    - /api/system/*
  rate_limiting:  # Ограничение запросов
    requests_per_minute: 100  # Запросов в минуту
    llm_requests_per_day: 50  # LLM запросов в день
```

Сервер (server:)

```
server:
  host: 0.0.0.0  # Хост для прослушивания
  port: 5000  # Порт сервера
```

```
  ssl_enabled: false  # Включение SSL/TLS
  max_connections: 1000  # Максимальное количество подключений
  request_timeout: 30  # Таймаут запроса в секундах
  cors_origins:  # Разрешенные CORS origins
   - https://kanban.example.com
   - http://193.233.171.205:3000
  debug: false  # Режим отладки
```

Производительность (performance:)

```
performance:
  cache_enabled: true  # Включение кэширования
  cache_ttl_seconds: 300  # Время жизни кэша (5 минут)
  csv_read_batch_size: 1000  # Размер пакета чтения CSV
  worker_processes: 4  # Количество рабочих процессов
  max_file_size_mb: 10  # Максимальный размер файла в MB
```

LLM настройки (llm:)

```
llm:
  enabled: true  # Включение LLM функций
  provider: openai  # Провайдер LLM
  api_key: ${OPENAI_API_KEY}  # API ключ из переменных окружения
  model: gpt-4-turbo-preview  # Модель LLM
  max_tokens: 2000  # Максимальное количество токенов
  cache_minutes: 60  # Кэширование ответов на 60 минут
  timeout_seconds: 60  # Таймаут запроса к LLM
```

Логирование (logging:)

```
logging:
  level: INFO  # Уровень логирования (DEBUG, INFO, WARNING, ERROR, CRITICAL)
  file_path: ./logs/task_system.log  # Путь к файлу логов
```

```
  max_size_mb: 100  # Максимальный размер файла лога
  format: '%(asctime)s - %(name)s - %(levelname)s - %(message)s'  # Формат логов
  retention_days: 30  # Хранение логов в днях
```

Экспорт (export:)

```
export:
  allow_all: true  # Разрешить экспорт всем пользователям
  csv_export_enabled: true  # Включить экспорт в CSV
  max_export_records: 10000  # Максимальное количество записей для экспорта
```

Управление данными

Структура данных

Система использует CSV файлы для хранения данных:

1. data/users.csv — Пользователи системы
2. data/tasks.csv — Задачи
3. data/events.csv — События системы
4. data/docs.csv — Документы

Форматы файлов

Пользователи (users.csv):

```
id,username,telegram_id,role,created_at,updated_at,is_active
```

Задачи (tasks.csv):

```
id,title,description,status,priority,assigned_to,created_by,due_date,created_at,updated_at
```

Ручное управление данными

Для ручного управления данными можно:

1. Остановить сервер
2. Отредактировать CSV файлы напрямую
3. Запустить сервер снова

Внимание: Прямое редактирование CSV файлов может привести к повреждению данных. Рекомендуется использовать API.

Безопасность

Настройки безопасности

1. **Включение безопасности:**
    - Установите security.enabled: true в конфигурации
    - Настройте FLASK_SECRET_KEY в переменных окружения
2. **Аутентификация:**
    - По умолчанию используется Telegram аутентификация
    - Настройте Telegram Bot Token в .env файле
3. **Авторизация:**
    - Ролевая модель доступа
    - Администраторы имеют доступ ко всем эндпоинтам
    - Обычные пользователи ограничены в правах
4. **CORS:**
    - Настройте разрешенные origins в server.cors_origins
    - Для продакшена укажите конкретные домены

Рекомендации для продакшена

1. **SSL/TLS:**
    - Установите server.ssl_enabled: true
    - Настройте SSL сертификаты
2. **Секретные ключи:**
    - Никогда не используйте дефолтные значения
    - Храните секреты в переменных окружения
    - Регулярно обновляйте ключи
3. **Ограничение доступа:**
    - Настройте брандмауэр
    - Ограничьте доступ к админ-панели по IP
    - Используйте VPN для администрирования

Мониторинг и логирование

Health Check

Endpoint: GET /api/health

Возвращает:

- Статус системы
- Количество записей в базе данных
- Состояние компонентов системы
- Конфигурационные параметры

Логирование

Локация логов: ./logs/task_system.log

Ротация логов:

- Максимальный размер: 100 MB
- Хранение: 30 дней
- Автоматическая ротация при достижении лимита

Уровни логирования:

- DEBUG — Детальная информация для отладки
- INFO — Основная информация о работе системы
- WARNING — Предупреждения о потенциальных проблемах
- ERROR — Критические ошибки
- CRITICAL — Критические системные ошибки

Мониторинг производительности

1. **Кэш:**
    - Статус: включен/выключен
    - Эффективность: можно отслеживать по логам
2. **Память:**
    - Максимальный размер файлов: 10 MB
    - Автоматическая очистка устаревших данных
3. **Сеть:**
    - Таймаут запросов: 30 секунд

- Максимальное количество подключений: 1000

Планировщик задач (Cron)

Включение планировщика

Установите cron.enabled: true в конфигурации.

Настройка задач

```
cron:
 enabled: true
 jobs:
  deadline_notifications:
   schedule: "0 9 * * *"  # Каждый день в 9:00
   enabled: true
   task: "check_deadlines"
   description: "Проверка сроков и отправка уведомлений"

  daily_report:
   schedule: "0 18 * * 1-5"  # Пн-Пт в 18:00
   enabled: true
   task: "generate_daily_report"
   description: "Генерация ежедневного отчета"
```

Формат расписания

Примеры cron выражений:

- "* * * * *" — Каждую минуту
- "0 * * * *" — Каждый час
- "0 9 * * *" — Каждый день в 9:00
- "0 18 * * 1-5" — Понедельник-Пятница в 18:00
- "*/15 * * * *" — Каждые 15 минут

Формат: минута час день месяц день_недели

Настройки планировщика

```
scheduler:
```

```
timezone: "Europe/Moscow"  # Часовой пояс
jobstore: "memory"  # Хранилище задач (memory или sqlalchemy)
thread_pool_size: 5  # Размер пула потоков
misfire_grace_time: 600  # Допустимая задержка выполнения (10 минут)
coalesce: true  # Объединение пропущенных задач
```

Мониторинг задач

Проверьте логи для мониторинга выполнения задач:

```
grep "cron" ./logs/task_system.log
```

Интеграции

Telegram интеграция

Настройка:

1. Создайте бота через @BotFather
2. Получите API токен
3. Добавьте токен в .env файл:

```
TELEGRAM_BOT_TOKEN=your-bot-token-here
```

Функции:

- Аутентификация пользователей
- Уведомления о задачах
- Синхронизация статусов

PostgreSQL (опционально)

Для включения PostgreSQL:

```
db_postgresql:
enabled: true
# Добавьте настройки подключения
```

LLM интеграция

Поддерживаемые провайдеры:

- OpenAI (по умолчанию)
- Другие провайдеры через расширение системы

Настройка:

1. Получите API ключ от провайдера
2. Добавьте в .env файл:

```
OPENAI_API_KEY=your-api-key-here
```

3. Настройте параметры в конфигурации

Устранение неисправностей

Общие проблемы

1. Сервер не запускается

Проверьте:

- Наличие всех зависимостей: pip install -r requirements.txt
- Правильность настроек в config.yaml
- Доступность порта (не занят ли порт 5000)
- Логи: cat ./logs/task_system.log

2. Проблемы с аутентификацией

Проверьте:

- Наличие TELEGRAM_BOT_TOKEN в .env
- Правильность FLASK_SECRET_KEY
- Настройки CORS в конфигурации

3. Ошибки базы данных

Проверьте:

- Существование директории ./data
- Права доступа к CSV файлам
- Целостность данных в CSV файлах

4. Проблемы с LLM

Проверьте:

- Наличие OPENAI_API_KEY в .env
- Доступность API провайдера
- Настройки таймаутов и лимитов

Диагностические команды

1. **Проверка health системы:**

```
curl http://localhost:5000/api/health
```

2. **Просмотр логов в реальном времени:**

```
tail -f ./logs/task_system.log
```

3. **Проверка зависимостей:**

```
pip list | grep -E "flask|pydantic|socket"
```

Резервное копирование

Автоматическое резервное копирование

Добавьте задачу в планировщик:

```
backup_job:
 schedule: "0 2 * * *"  # Каждый день в 2:00
 enabled: true
 task: "backup_data"
 description: "Резервное копирование данных"
```

Ручное резервное копирование

1. **Остановите сервер:**

```
# Найдите PID процесса
ps aux | grep server.py
# Остановите процесс
kill <pid>
```

2. **Создайте резервную копию:**

```
# Создайте архив с данными
tar -czf backup_$(date +%Y%m%d_%H%M%S).tar.gz data/ logs/ config.yaml
.env
```

3. **Храните резервные копии:**

- Минимум 7 ежедневных бэкапов
- 4 еженедельных бэкапа
- 12 ежемесячных бэкапов

Восстановление из резервной копии

1. **Остановите сервер**
2. **Восстановите данные:**

```
tar -xzf backup_20240101_120000.tar.gz
```

3. **Проверьте целостность:**

```
python -c "import csv; csv.reader(open('data/users.csv'))"
```

4. **Запустите сервер**

Контакты для поддержки

- **Логи ошибок:** ./logs/task_system.log
- **Health endpoint:** http://<server-address>:5000/api/health
- **Документация API:** http://<server-address>:5000/

Рекомендации:

1. Всегда проверяйте логи перед обращением в поддержку
2. Предоставляйте версию системы из health endpoint
3. Указывайте время возникновения проблемы