**Техническая документация: Система управления лизинговыми данными с Telegram-ботом**

**1. Введение**

Система предназначена для автоматического обновления лизинговых данных из XLSX-файла и предоставления доступа к данным через Telegram-бота. Основные функции:

* Ежедневное обновление данных из внешнего источника
* Хранение данных в MySQL
* Управление через Telegram-бота
* REST API для интеграций

**2. Архитектура системы**

**2.1. Компоненты системы**

1. **Внешний источник данных**: XLSX-файл по HTTP
2. **Django-приложение**: Обработка данных и бизнес-логика
3. **MySQL Database**: Хранение лизинговых данных
4. **Telegram Bot**: Пользовательский интерфейс
5. **Webhook Handler**: Обработчик запросов от Telegram

**2.2. Технологический стек**

* **Backend**: Django 4.x
* **Database**: MySQL 8.x
* **Bot Framework**: python-telegram-bot 20.x
* **Data Processing**: Pandas 2.x, Openpyxl 3.x
* **HTTP Client**: Requests 2.x

**3. Установка и настройка**

**3.1. Требования**

* Python 3.10+
* MySQL Server 8+
* Сертификат SSL (для продакшена)

**3.2. Конфигурация окружения**

bash

Copy

# Установка зависимостей

pip install -r requirements.txt

# Файл .env

TELEGRAM\_BOT\_TOKEN=your\_bot\_token

DATABASE\_URL=mysql://user:password@host/dbname

WEBHOOK\_URL=https://your-domain.com/bot/webhook/

**3.3. Миграции базы данных**

bash

Copy

python manage.py makemigrations

python manage.py migrate

**4. Детали реализации**

**4.1. Модель данных**

python

Copy

class LeasingRecord(models.Model):

contract\_number = models.CharField(max\_length=100, unique=True)

client\_name = models.CharField(max\_length=255)

amount = models.DecimalField(max\_digits=15, decimal\_places=2)

start\_date = models.DateField()

end\_date = models.DateField()

status = models.CharField(max\_length=50)

**4.2. Команда обновления данных**

**Команда**: python manage.py update\_data

Алгоритм работы:

1. Загрузка XLSX-файла по HTTP
2. Парсинг данных с помощью Pandas
3. Преобразование структуры данных
4. Пакетное обновление записей в БД

mermaid

Copy

sequenceDiagram

participant User

participant Bot

participant Django

participant DB

User->>Bot: /update

Bot->>Django: call\_command('update\_data')

Django->>External: GET XLSX

External->>Django: XLSX Response

Django->>Django: Parse Data

Django->>DB: Bulk Update

DB->>Django: Update Result

Django->>Bot: Success/Fail

Bot->>User: Result

**4.3. Telegram Bot API**

**Доступные команды:**

| **Команда** | **Описание** | **Пример** |
| --- | --- | --- |
| /start | Начало работы | /start |
| /update | Обновить данные | /update |
| /stats | Статистика данных | /stats |

**5. Настройка Webhook**

**5.1. Регистрация вебхука**

bash

Copy

python manage.py set\_webhook

**5.2. Схема запросов**

http

Copy

POST /bot/webhook/

Content-Type: application/json

{

"update\_id": 123456,

"message": {

"message\_id": 123,

"from": {"id": 654321},

"chat": {"id": 654321},

"text": "/update"

}

}

**6. Обработка ошибок**

**6.1. Типовые ошибки**

| **Код ошибки** | **Причина** | **Способ решения** |
| --- | --- | --- |
| 502 | Неверный URL вебхука | Проверить WEBHOOK\_URL |
| 504 | Таймаут подключения | Увеличить timeout |
| 1001 | Ошибка парсинга XLSX | Проверить формат файла |
| 1002 | Ошибка подключения к БД | Проверить credentials |

**6.2. Логирование**

Файлы логов:

* logs/bot\_errors.log - Ошибки бота
* logs/data\_processing.log - Ошибки обработки данных

**7. Мониторинг и администрирование**

**7.1. Статистические метрики**

sql

Copy

-- Активные договоры

SELECT COUNT(\*)

FROM leasing\_records

WHERE status = 'active'

AND end\_date > CURDATE();

**7.2. Планировщик задач**

Пример cron-задания для ежедневного обновления:

bash

Copy

0 3 \* \* \* /usr/bin/python3 /app/manage.py update\_data

**8. Безопасность**

**8.1. Меры защиты**

1. Валидация входящих запросов от Telegram
2. HTTPS для всех внешних соединений
3. Изоляция окружений (dev/stage/prod)
4. Регулярное резервное копирование БД

**8.2. Рекомендации по аудиту**

1. Проверка прав доступа к файлам конфигурации
2. Мониторинг подозрительной активности в логах
3. Регулярное обновление зависимостей

**9. Разработка и тестирование**

**9.1. Запуск в dev-режиме**

bash

Copy

python manage.py runserver 0.0.0.0:8000

**9.2. Тестовые данные**

python

Copy

LeasingRecord.objects.create(

contract\_number='TEST-001',

client\_name='Test Client',

amount=100000.00,

start\_date='2024-01-01',

end\_date='2025-01-01',

status='active'

)

**10. Дорожная карта развития**

**10.1. Планируемые улучшения**

* Добавление REST API
* Интеграция с системами аналитики
* Реализация ролевой модели доступа
* Поддержка многоязычности

**10.2. Контакты разработчиков**

Поддержка системы: [dev-team@example.com](https://mailto:dev-team@example.com/)

Данная документация охватывает основные аспекты системы. Для получения дополнительной информации обратитесь к исходному коду или свяжитесь с командой поддержки.