Техническое задание для бэкенд-разработчика астрономического приложения

6 мая 2025 г.

Фокус

Реализация вычислительного ядра для анализа данных комет

1 Введение

Разработать серверную часть приложения на Python, которая:

- Выполняет расчёты из 4 заданий хакатона (сублимация, графики, масса, размер ядра)
- Обрабатывает данные из `.txt`/`.fits`-файлов (по шаблону `C_2023_A3.xlsx`)
- Предоставляет API для фронтенда (оконного приложения)

Требования к ОС: Windows 10/11 (совместимость с серверным развёртыванием)

2 Функциональные требования

2.1 Общие для всех модулей

- **Формат входных данных:**
  - `.txt` (обязательно) с колонками: `Дни до перигелия`, `Звёздная величина`, `Afrho`
  - `.fits` (опционально, через `astropy.io.fits`)
- **Валидация данных:** Проверка на корректность числовых значений и структуры файлов
- **Логирование:** Запись ошибок в `error.log` с timestamp

2.2 Специфика по заданиям

Модуль	Функции	Формулы/Методы
Сублимация (Задание 1)	• Расчёт температуры сублимации • Определение состава кометы	$T_{sub} = \frac{H}{\ln(P_0) - \ln(P) + \ln(\mu)}$
Графики (Задание 2)	• Генерация данных для графиков • Интерполяция недостающих точек	Кубические сплайны, линейная интерполяция

Масса кометы (Задание 3)	• Расчёт массы по фотометрическим данным • Учёт параметров наблюдения	$N = 10^{-0.4(m_к - m_{лк})} \cdot \Delta^2 \cdot r^2/(1.37 \times 10^{-38} \cdot f(C_2))$
Размер ядра (Задание 4)	• Оценка диаметра ядра • Учёт альбедо	$D = \sqrt{\frac{4A}{\pi p H}}$

2.3 API-эндпоинты (REST)

- `POST /api/calculate` - вход: файл/параметры, выход: JSON с результатами
  - Параметры: `file`, `calculation_type`
  - Ответ: `{result: float, units: string, error: string|null}`
- `GET /api/plot_data` - возврат данных для построения графиков
  - Параметры: `dataset_id`
  - Ответ: `{x: array, y: array, labels: object}`

3 Технические требования

- **Язык:** Python 3.10+
- **Библиотеки:**
  - Обязательные: `astropy`, `numpy`, `fastapi`, `pydantic`, `uvicorn`
  - Опциональные: `pandas`, `scipy`, `redis`
- **Производительность:**
  - Время обработки файла $\leq$ 300 мс (95 персентиль)
  - Поддержка 50 RPS (тест через `locust`)
- **Безопасность:**
  - Валидация MIME-типов файлов
  - Ограничение размера файлов ($\leq$ 10 MB)

4 Архитектура

```
backend/
 app/
    api/
       endpoints.py          # FastAPI роуты
       schemas.py           # Pydantic модели
    core/
       calculations/        # Модули расчётов
          sublimation.py
          plots.py
          mass.py
          nucleus.py
    services/
       file_parser.py       # Парсеры файлов
       logger.py           # Логирование
    config.py               # Конфигурация
```

```
tests/
   unit/                    # Юнит-тесты
   integration/             # Интеграционные тесты
requirements.txt
main.py                      # Точка входа
```

5 Критерии приёмки

1. **Корректность расчётов:**
   - Совпадение результатов с эталонными данными (погрешность $< 1\%$)
   - Обработка edge cases (пустые файлы, NaN значения)
2. **Производительность:**
   - Нагрузочное тестирование: 1000 запросов без ошибок
   - Задержка $<$ 500ms при 50 RPS
3. **Документация:**
   - Автоматическая Swagger-документация API
   - README с примерами запросов/ответов

6 План разработки

День	Задачи
1	Настройка FastAPI, базовые эндпоинты, парсинг TXT
2	Реализация модуля массы кометы (Задание 3) + тесты
3	Модуль сублимации (Задание 1) с параметрами μ, H, P_0
4	Генерация данных для графиков (Задание 2)
5	Реализация расчёта размера ядра (Задание 4)
6	Оптимизация, кеширование, документация

7 Дополнительно

- **Тестирование:**
  - Покрытие кода $\geq 80\%$ (pytest-cov)
  - Моки для внешних зависимостей
- **Развёртывание:**
  - Docker-образ с Python 3.10
  - Helm-чарт для Kubernetes (опционально)
- **Мониторинг:**
  - Prometheus-метрики
  - Health-check эндпоинт