Создание **API** (**Rest**) для управления списком задач (**To-Do List**), включающий эндпоинты для CRUD-операций (Create, Read, Update, Delete) с использованием методов `GET`, `POST`, `PUT/PATCH`, `DELETE`, работающих с ресурсами типа `task` (например, `/tasks`, `/tasks/{id}`), с описанием структуры запросов/ответов, кодов состояния и данных в формате JSON.

CRUD API для управления задачами (To-Do List API)
1. Цель:
Предоставить набор эндпоинтов для управления задачами (создание, получение, обновление, удаление).

2. Ресурсы:

- Задача (Task): Содержит ID, название (title), описание (description), статус (status - "todo", "in_progress", "done").

3. Эндпоинты (Endpoints) и Методы:

- `GET /tasks` - Получить список всех задач.
- `POST /tasks` - Создать новую задачу.
- `GET /tasks/{id}` - Получить одну задачу по ID.
- `PUT /tasks/{id}` - Полностью обновить задачу по ID.
- `DELETE /tasks/{id}` - Удалить задачу по ID.

4. Описание операций (Примеры):

- Создание задачи (POST /tasks):
    - Запрос: json

```
{
  "title": "Купить молоко",
  "description": "Обязательно 3.2% жирности",
  "status": "todo"
}
```

    - Ответ (201 Created): json

```
{
  "id": 1,
  "title": "Купить молоко",
  "description": "Обязательно 3.2% жирности",
  "status": "todo"
}
```

- Получение списка (GET /tasks):
    - Ответ (200 OK): json

```
[
  { "id": 1, "title": "Купить молоко", "status": "todo" },
  { "id": 2, "title": "Запустить API", "status": "in_progress" }
]
```

- Обновление статуса (PATCH /tasks/{id}):
    - Запрос (PATCH /tasks/2):json

```
{
  "status": "done"
}
```

    - Ответ (200 OK): json

```
{
  "id": 2,
```

```
  "title": "Запустить API",
  "description": "...",
  "status": "done"
}
```

5. Ожидаемые результаты:

- Реализована логика CRUD.
- Используются стандартные HTTP-методы.
- Применяются корректные HTTP- коды (200, 201, 404).
- Формат обмена данными — JSON.

Пример реализации: Можно использовать Node.js + Express, Python + Flask/Django, Java + Spring Boot, или Go, Rust, C++, C …

Программа минимум: 0) REST API без БД

Что было бы полезно добавить:

1) Работа с БД - например SQLite, MongoDB и другие;
2) Добавление максимально функционального стандартного API Gateway (балансировщик нагрузки, “ограничитель” трафика, авторизация и аутентификация);
3) Добавление кеширования;
4) Добавление очереди сообщений;
5) Добавление инструментов логирования и сбора метрик;
6) Добавление отказоустойчивости на всех уровнях (БД, Кеш, Gateway и так далее);
7) Автоматическое развертывание получившегося “монстра”;