# **Go Proxy API**

Это Go API, которое служит прокси для внешнего API. Оно принимает данные в GET-параметрах URL, преобразует их в JSON-формат и пересылает POST-запросом во внешний API, а затем возвращает ответ внешнего API клиенту.

## **Запуск**

1. Перед запуском приложения необходимо установить переменную окружения PREDICT\_API\_BEARER\_TOKEN.  
   Windows (CMD):  
   set PREDICT\_API\_BEARER\_TOKEN="0l62<EJi/zJx]a?"
2. Запустите приложение:  
   go run main.go
3. Сервер будет запущен на порту 8080. В консоли появится сообщение типа: 2025/07/04 17:36:10 Server started on http://localhost:8080/predict

## **Тестирование**

API доступно по адресу http://localhost:8080/predict и ожидает GET-запрос со следующими параметрами в URL:

* age (int)
* gender (int)
* rdw (float64)
* wbc (float64)
* rbc (float64)
* hgb (float64)
* hct (float64)
* mcv (float64)
* mch (float64)
* mchc (float64)
* plt (float64)
* neu (float64)
* eos (float64)
* bas (float64)
* lym (float64)
* mon (float64)
* soe (float64)
* chol (float64)
* glu (float64)

Значение uid для внешнего API фиксировано как "web-client".

### **Пример запроса с использованием Postman / cURL**

#### **Пример URL-запроса:**

* http://localhost:8080/predict?age=30&gender=1&rdw=12.5&wbc=7.2&rbc=4.5&hgb=14.0&hct=42.0&mcv=90.0&mch=30.0&mchc=33.0&plt=250.0&neu=60.0&eos=2.0&bas=0.5&lym=30.0&mon=7.0&soe=10.0&chol=200.0&glu=90.0

#### **Использование cURL:**

* curl -X GET "http://localhost:8080/predict?age=30&gender=1&rdw=12.5&wbc=7.2&rbc=4.5&hgb=14.0&hct=42.0&mcv=90.0&mch=30.0&mchc=33.0&plt=250.0&neu=60.0&eos=2.0&bas=0.5&lym=30.0&mon=7.0&soe=10.0&chol=200.0&glu=90.0"

## **Реализация**

API реализовано на языке Go с использованием стандартной библиотеки net/http для создания веб-сервера и обработки HTTP-запросов.

### **Основные компоненты:**

1. Структура PredictRequest:
   * Определяет формат JSON-тела, которое будет отправлено во внешний POST-API.
   * Использует теги json:"fieldName" для корректного сопоставления полей структуры Go с ключами в JSON.
2. Функция predictHandler:
   * Это основной HTTP-хендлер, который обрабатывает все входящие GET-запросы на маршрут /predict.
   * Валидация метода: В начале функции проверяется, что запрос является методом GET. Если это не так, возвращается ошибка 405 Method Not Allowed.
   * Получение токена авторизации: Токен Bearer извлекается из переменной окружения PREDICT\_API\_BEARER\_TOKEN для обеспечения безопасности и гибкости конфигурации. Если переменная не установлена, возвращается ошибка 500 Internal Server Error.
   * Парсинг параметров URL: Извлекает все необходимые параметры (такие как age, gender, rdw и т.д.) из URL-запроса.
   * Валидация и преобразование данных: Используются вспомогательные функции parseFloat и parseInt для безопасного преобразования строковых параметров из URL в числовые типы (float64 и int). В случае отсутствия параметра или ошибки парсинга возвращается ошибка 400 Bad Request с указанием проблемы.
   * Формирование тела запроса для внешнего API: Извлеченные и валидированные данные используются для создания экземпляра структуры PredictRequest. Значение uid фиксировано как "web-client".
   * Сериализация в JSON: Созданный объект PredictRequest сериализуется в JSON-формат с помощью json.Marshal.
   * Создание и отправка POST-запроса к внешнему API:
     + Создается новый HTTP-запрос методом POST к https://apiml.labhub.online/api/v1/predict/hba1c.
     + Устанавливаются необходимые заголовки: Authorization (с полученным токеном), Content-Type: application/json и Accept: application/json; charset=utf-8.
     + Используется http.Client с настроенным таймаутом в 10 секунд для предотвращения длительного ожидания ответа от внешнего сервиса.
   * Обработка ответа от внешнего API:
     + Читается тело ответа от внешнего API.
     + Статус-код и заголовки Content-Type из ответа внешнего API устанавливаются для ответа прокси-API.
     + Тело ответа внешнего API передается напрямую клиенту, который вызвал прокси-API.
3. Функция main:
   * Регистрирует predictHandler для обработки запросов по маршруту /predict с помощью http.HandleFunc.
   * Запускает HTTP-сервер на порту 8080 с помощью http.ListenAndServe. В случае ошибки запуска сервера (например, если порт уже занят), приложение завершает работу с фатальной ошибкой.