#### **Описание проекта**

Проект URL Shortener - это веб-сервис, который предоставляет возможность создавать сокращенные URL и перенаправлять на исходные длинные URL. Сервис включает два основных API-эндпоинта:

* **POST /shorten**: принимает JSON-запрос с длинным URL и возвращает сокращенный URL.
* **GET /{shortURL}**: перенаправляет на исходный длинный URL.

#### **Структура проекта**

url-shortener/

├── main.go

├── main\_test.go

├── go.mod

└── go.sum

#### **Основные компоненты**

1. **main.go**: основной файл с логикой приложения.
2. **main\_test.go**: файл с тестами для проверки корректности работы сервиса.
3. **go.mod** и **go.sum**: файлы, управляющие зависимостями Go.

#### Установка и запуск

##### **Требования**

* Go 1.18 или выше
* MariaDB/MySQL сервер

##### **Настройка базы данных**

Создайте базу данных urlshortener и таблицу urls:

CREATE DATABASE urlshortener;

USE urlshortener;

CREATE TABLE urls (

id INT AUTO\_INCREMENT PRIMARY KEY,

original\_url TEXT NOT NULL,

short\_url VARCHAR(255) NOT NULL UNIQUE

);

short\_url VARCHAR(255) NOT NULL UNIQUE

);

**Настройка и запуск приложения**

1. Склонируйте репозиторий

git clone https://github.com/yourusername/url-shortener.git

cd url-shortener

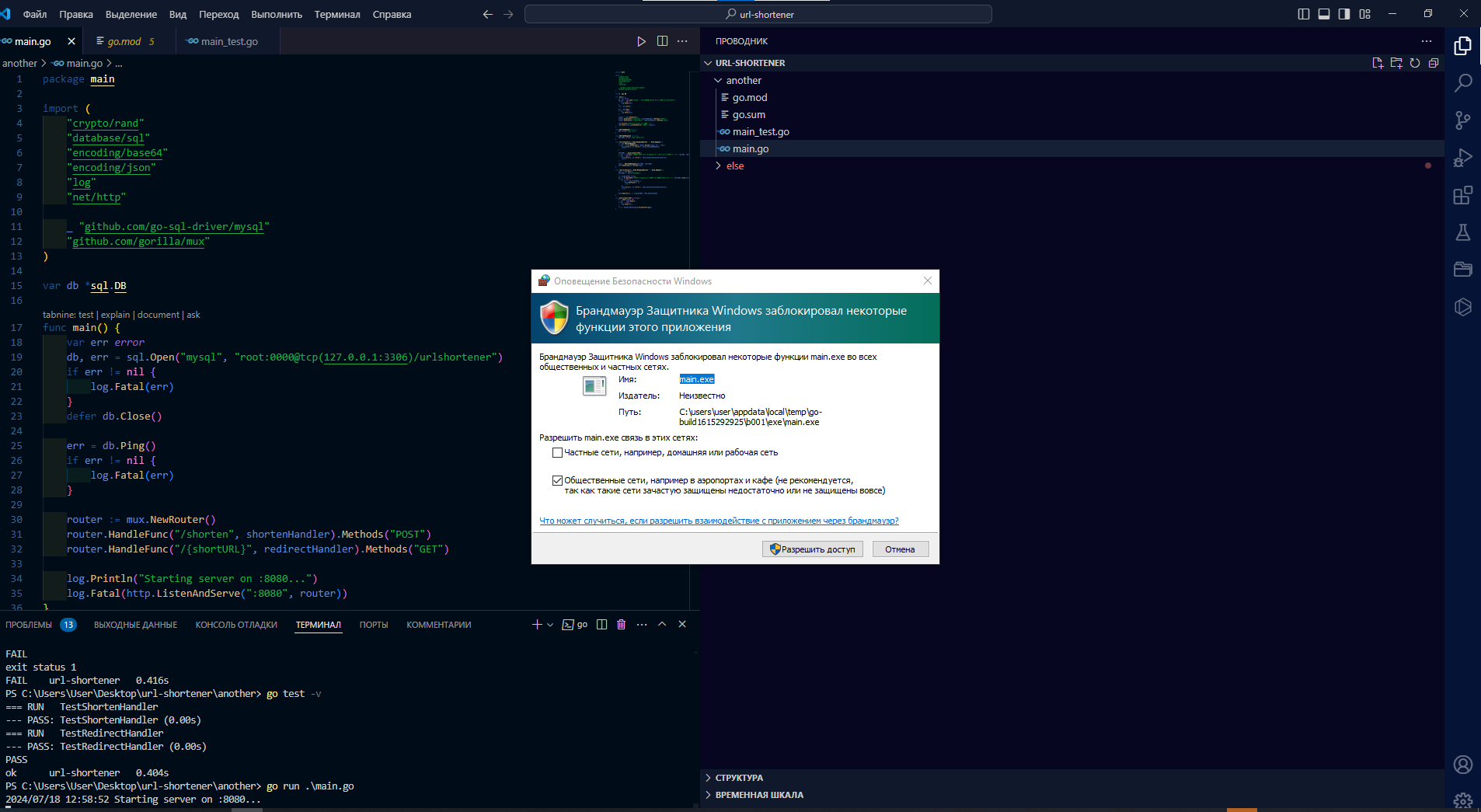
1. Установите зависимости:

go mod tidy

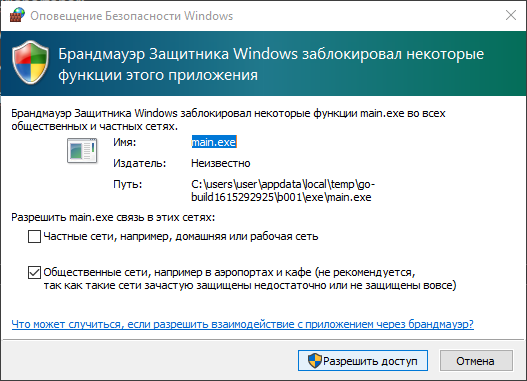
1. Запустите приложение:

go run main.go

Запускаем main.go с помощью команды go run .\main.go



Запускаем процесс



Отображение рабочего процесса



##### **API-эндпоинты**

**POST /shorten**

**Описание**: Принимает JSON-запрос с длинным URL и возвращает сокращенный URL.

**Пример запроса**:

{

"url": "http://example.com"

}

**Пример ответа**:

{

"short\_url": "randomShortURL"

}

#### **Пример использования с Postman**

**POST /shorten**

В Postman создайте новый запрос типа POST.

Установите URL на <http://localhost:8080/shorten>.

Перейдите на вкладку Body, выберите raw и установите формат JSON.

Введите JSON:

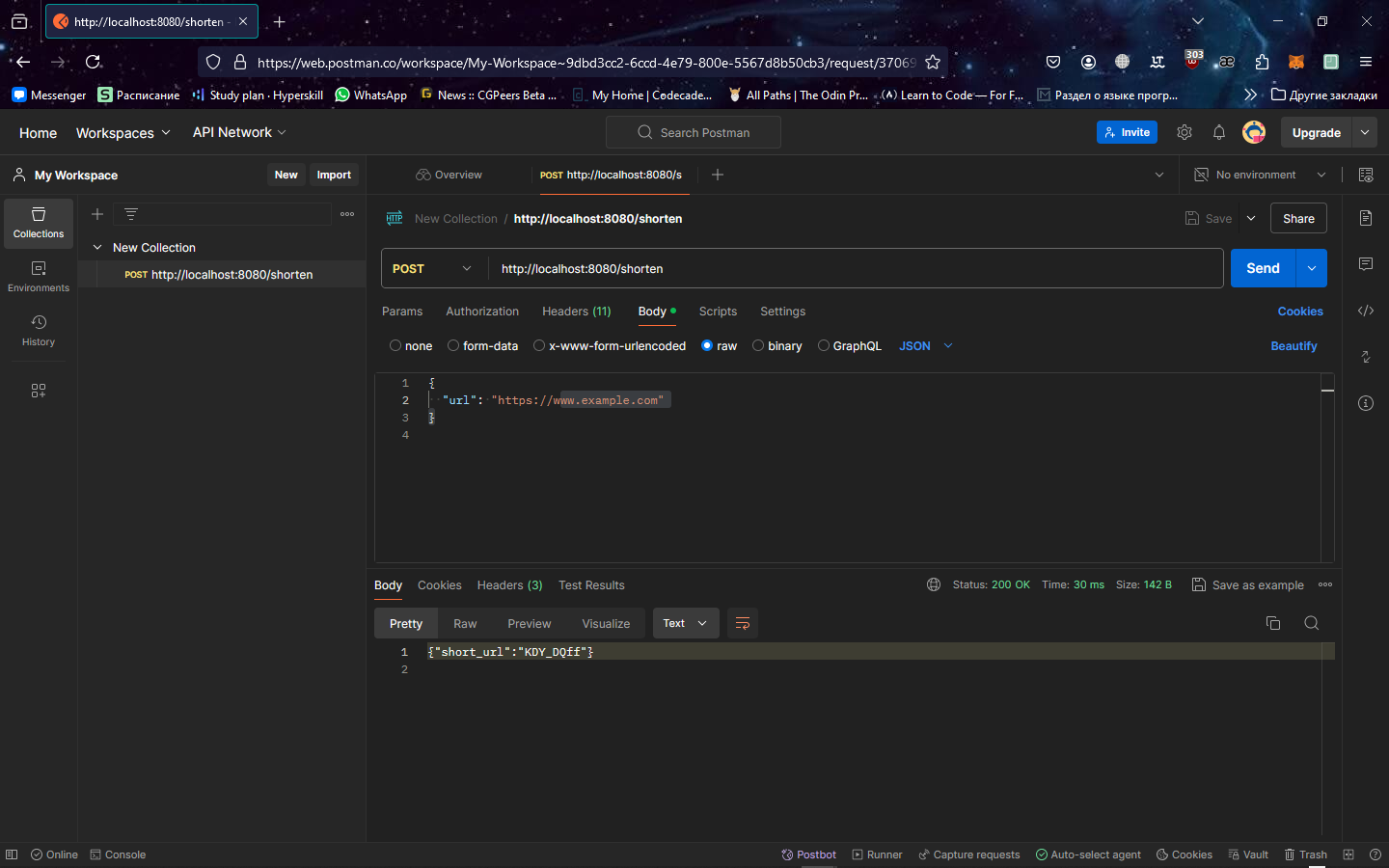
Делаем POST запрос и получаем KDY\_DQff случайно сгенерированный с помощью функции generateShortURL

{

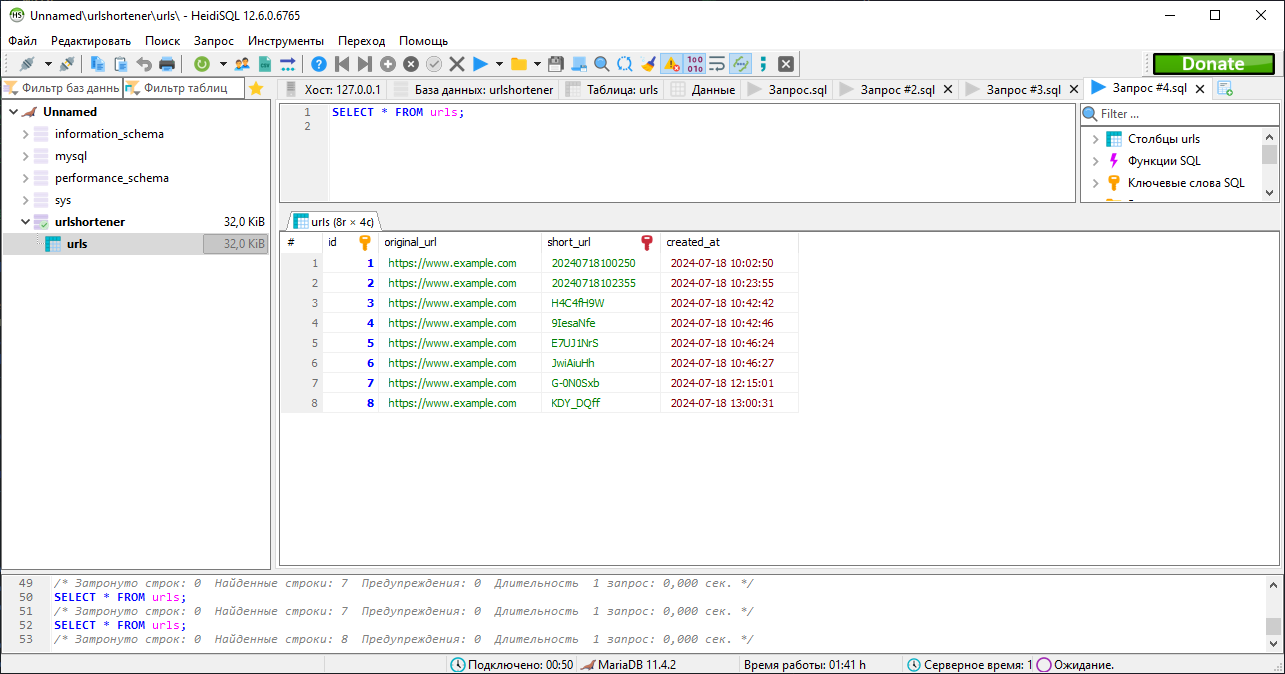
"url": "http://example.com"

}

Нажмите Send. В ответе вы получите сокращенный URL.

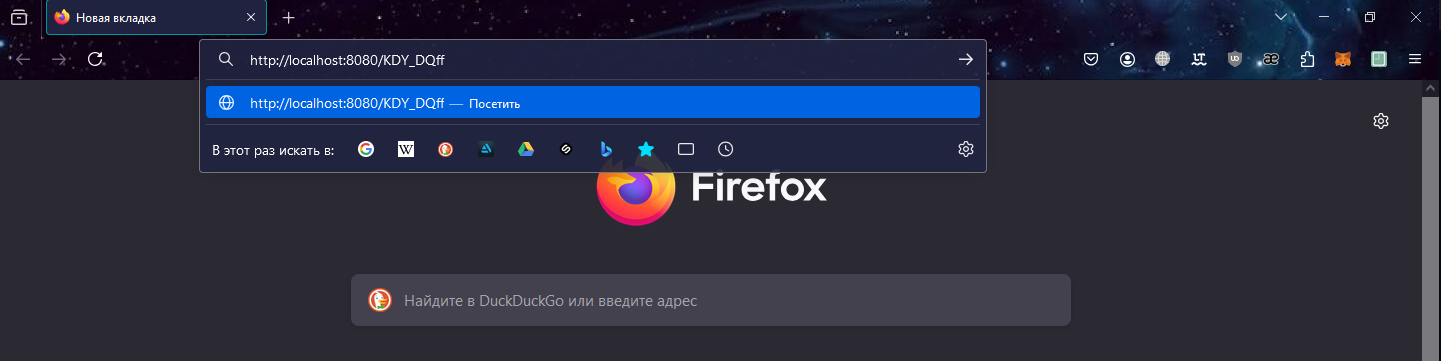


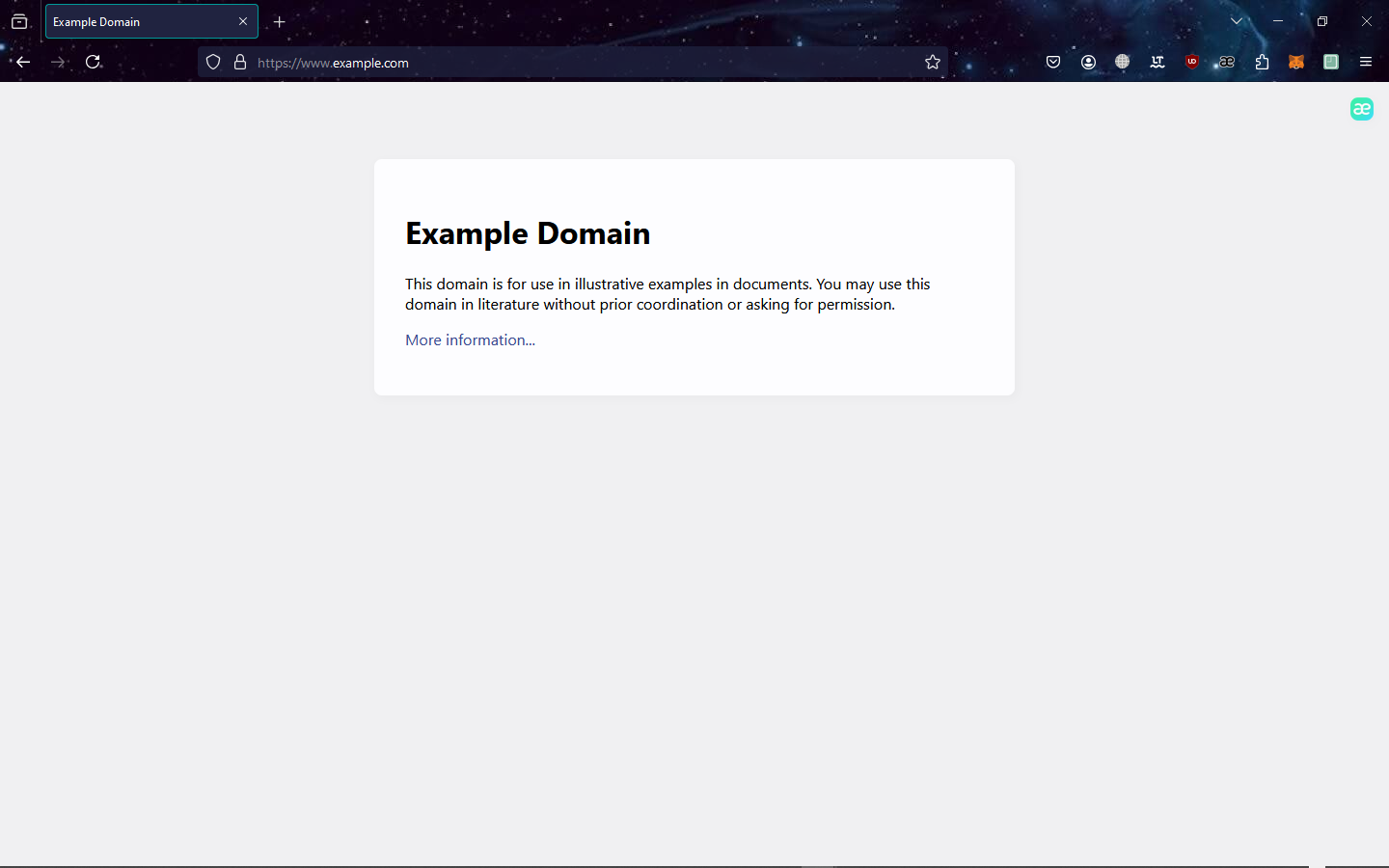
Обновляем базу данных и видим отображение в ней



Переходим на локально запущенный сервер по адресу 8080

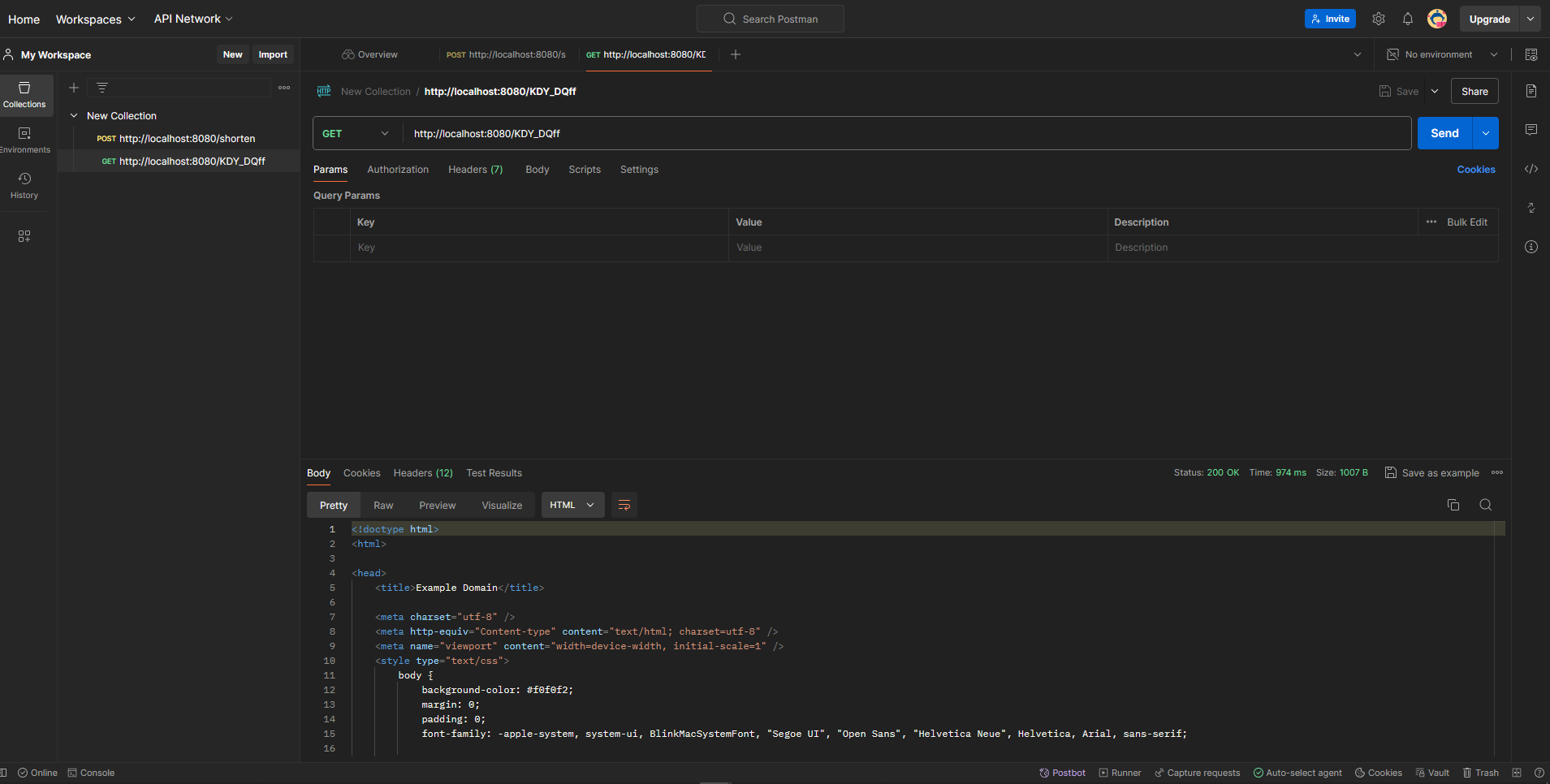
http://localhost:8080/shorten/KDY\_DQff



Получаем в ответ страничку о использовании заглушки exemple 

**GET /{shortURL}**

* В Postman создайте новый запрос типа GET.
* Установите URL на <http://localhost:8080/{shortURL>}, где {shortURL} замените на сокращенный URL, полученный ранее к примеру KDY\_DQff
* Нажмите Send. Вы будете перенаправлены на исходный длинный URL

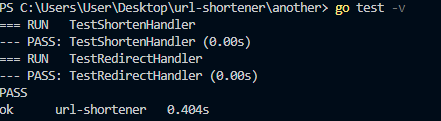


**Тестирование**

Для запуска тестов используйте команду:

go test -v

Выполняем тесты



Тестирование функции TestShortenHandler.

Эта функция обрабатывает POST-запрос к конечной точке /shorten, которая принимает JSON-полезную нагрузку с URL-адресом и возвращает сокращенную версию этого URL-адреса.

Тест использует пакет sqlmock для создания моковой базы данных и устанавливает ожидания для операций с базой данных.

В этом случае ожидается, что будет выполнено инструкция INSERT с указанным URL-адресом и сгенерированным коротким URL-адресом.

Тест создает POST-запрос с JSON-полезной нагрузкой, содержащей URL-адрес для сокращения. Затем он настраивает запись ответа для захвата HTTP-ответа. Функция http.HandlerFunc(shortenHandler) вызывается для обработки запроса и записи ответа в rr.

После этого выполняются проверки с помощью пакета assert: ожидается, что код ответа будет http.StatusOK (200), и возвращенный короткий URL-адрес не должен быть пустым. Наконец, тест проверяет, были ли выполнены все ожидаемые ожидания с помощью mock.ExpectationsWereMet().

Если есть невыполненные ожидания, тест завершается с ошибкой.

Тестовая функция TestRedirectHandler для обработчика HTTP-запросов redirectHandler.

Эта функция отвечает за перенаправление пользователей на исходный URL-адрес на основе короткого URL-адреса, указанного в пути запроса.

В тесте используется пакет sqlmock для создания моковой базы данных и ожиданий для SQL-запросов, которые будут выполняться во время теста.

Объект mockDB используется для замены фактического подключения к базе данных в коде приложения. Ожидается, что при выполнении запроса GET к маршруту /shortURL123 обработчик redirectHandler вернет ответ с кодом состояния 302 (Found) и заголовком Location со значением "[http://example.com](http://example.com/)", которое указывает на исходный URL-адрес.

После выполнения запроса и получения ответа тест проверяет, были ли выполнены все ожидаемые ожидания с помощью mock.ExpectationsWereMet().

Если есть невыполненные ожидания, тест выдает ошибку.