**Тестовое задание на должность «Backend-стажёр в юнит**

**Авито.Недвижимость».**

Язык программирования: PHP

База данных: MYSQL

В качестве ОС для сервера была выбрана Ubuntu 18.04 LTS

В качестве веб-сервера выбран Apache

**Шаг 1**: На виртуальной машине Virtual Box установлено Ubuntu 18.04 + Apache + MYSQL + PHPMYADMIN + Open SSH.

**Шаг 2**: в качестве среды разработки выбрана PHP Storm. Среда настроена на автоматическую выгрузку всех файлов проекта на сервер.

**Шаг 3**: создана база данных со следующими полями:



id – уникальный идентификатор объявления. Генерируется приложением

header – заголовок объявления

text – описание объявления

price – цена

date – дата добавления объявления (записывается автоматически средствами MYSQL)

**Шаг 4**: Разработано 3 страницы для каждого из запросов.

Add.php – получает от пользователя параметры методом POST. Добавляет объявление в базу. Генерирует случайный, но ранее не использовавшийся ID для объявления.

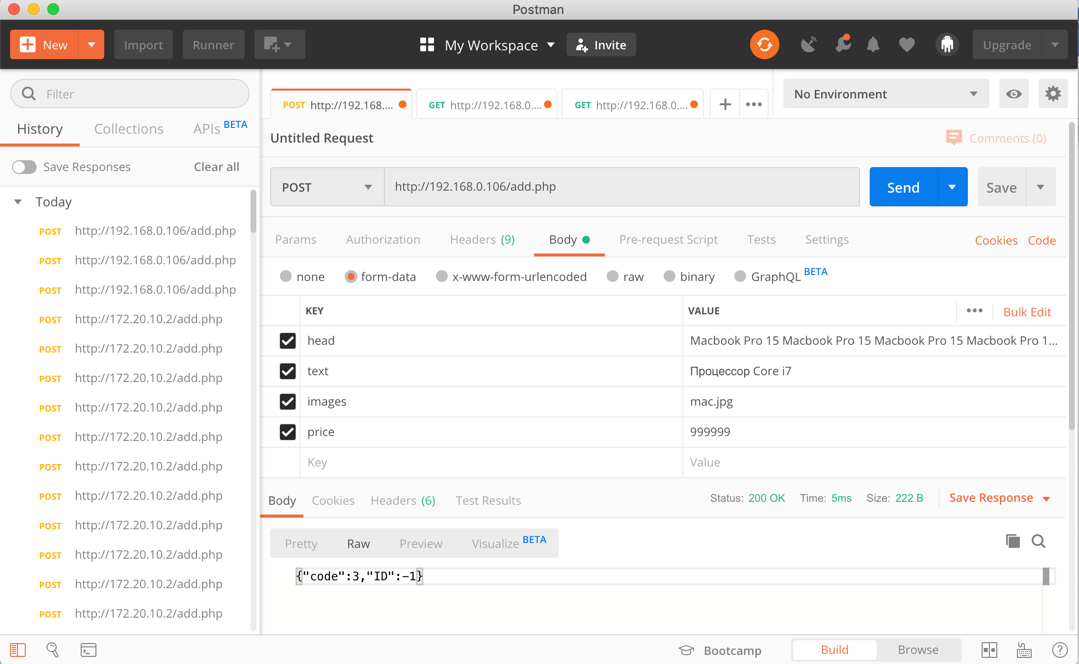
Getads.php – Получает от пользователя параметры методом GET. Возвращает постраничный список всех объявлений

Getadbyid.php – получает от пользователя параметры методом GET. Возвращает конкретное объявление по его уникальному идентификатору.

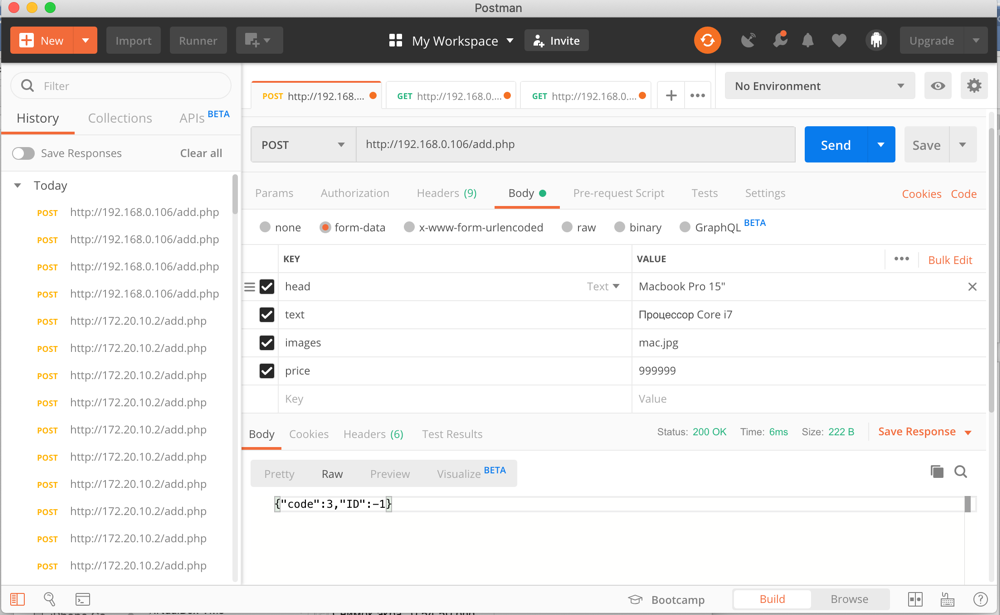
**Шаг 5**: Тестирование с помощью POSTMAN.

**add.php:**

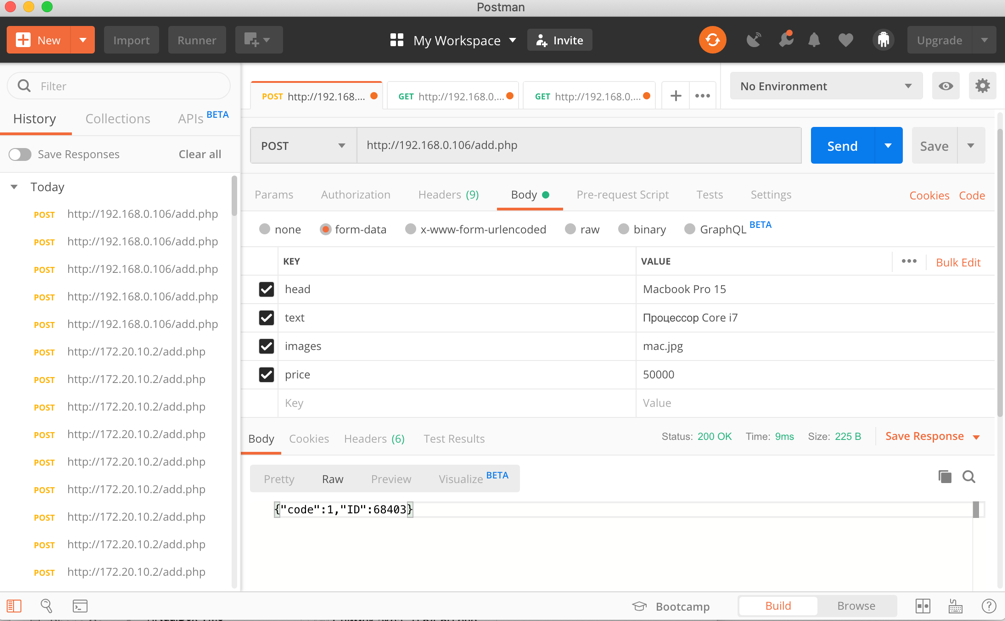
Пробуем добавить объявление со слишком длинным заголовком – получаем код ошибки 3 (не прошел валидацию)



Пробуем добавить объявление с заголовком, содержащим кавычку (используется для SQL инъекций) получаем тот же самый код ошибки 3

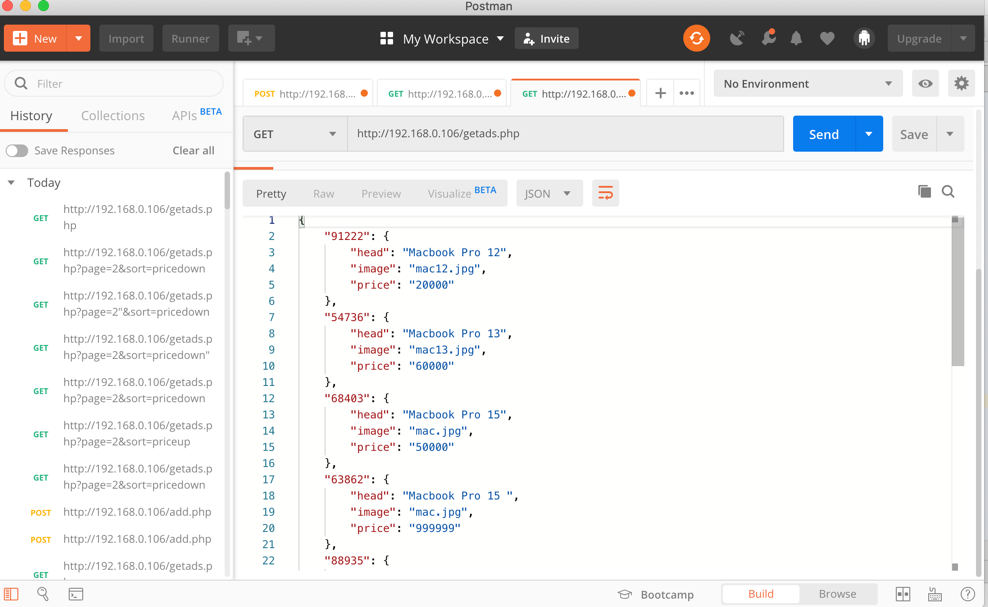


Пробуем добавить объявление с нормальными полями – получаем код ошибки 1, что значит что все в порядке, а также получаем автоматически сгенерированный приложением ID объявления.

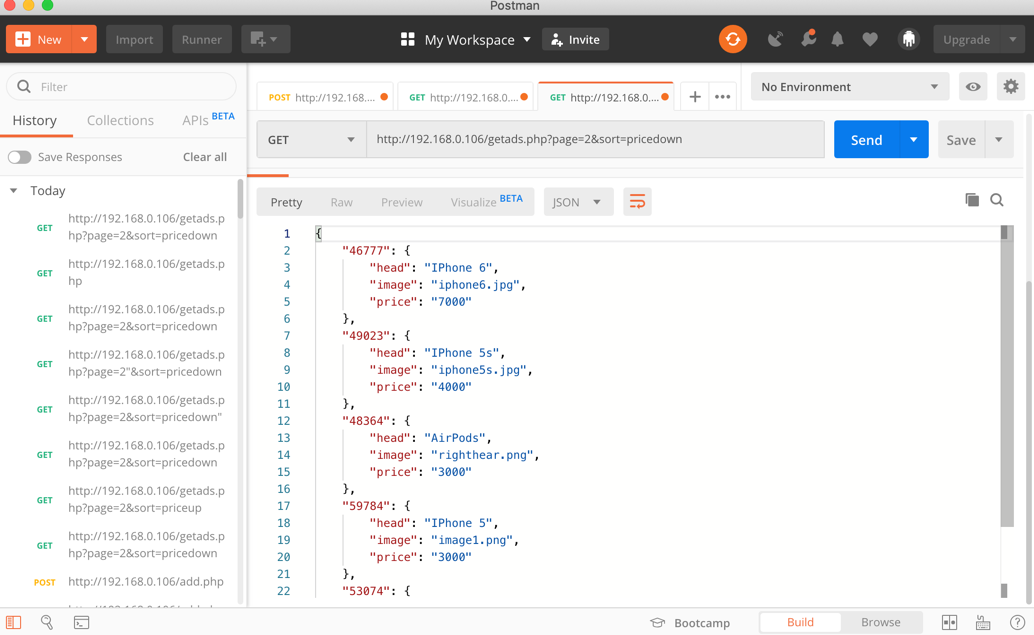


**getads.php:**

Отправляем запрос без параметров, получаем первую страницу объявлений, отсортированную по умолчанию – по дате публикации

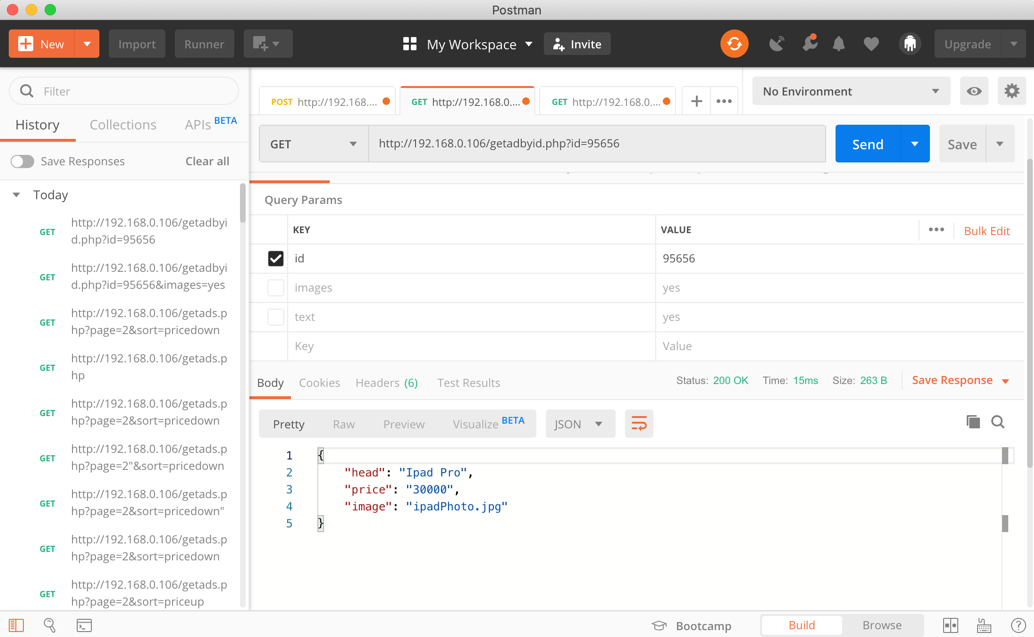


Теперь отправляем запрос с двумя параметрами: page = 2 и sort = pricedown, получаем вторую страницу объявлений отсортированную по убыванию цены

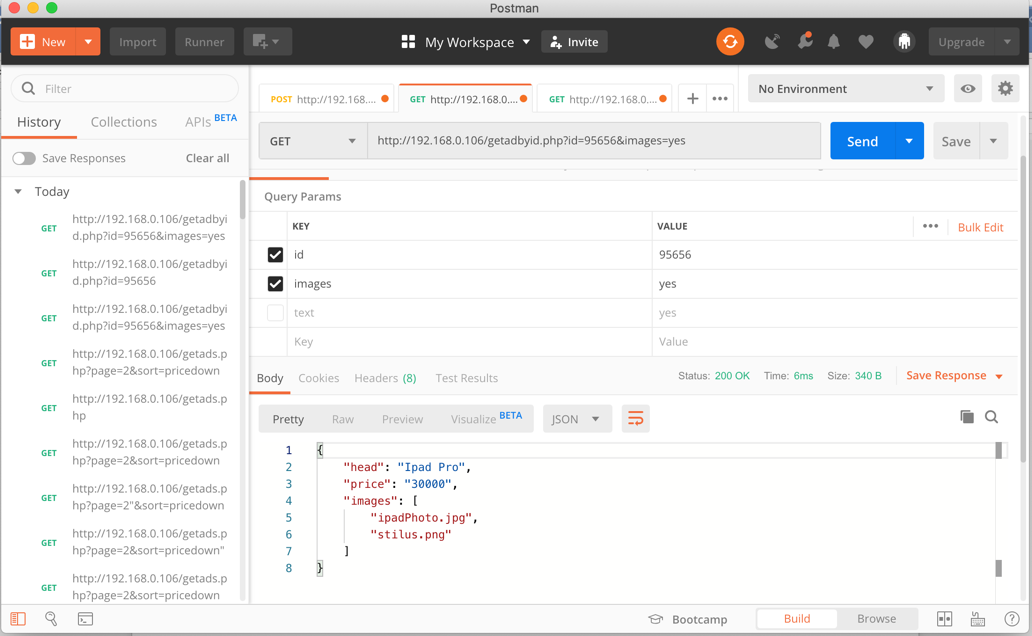


**getadbyid.php:**

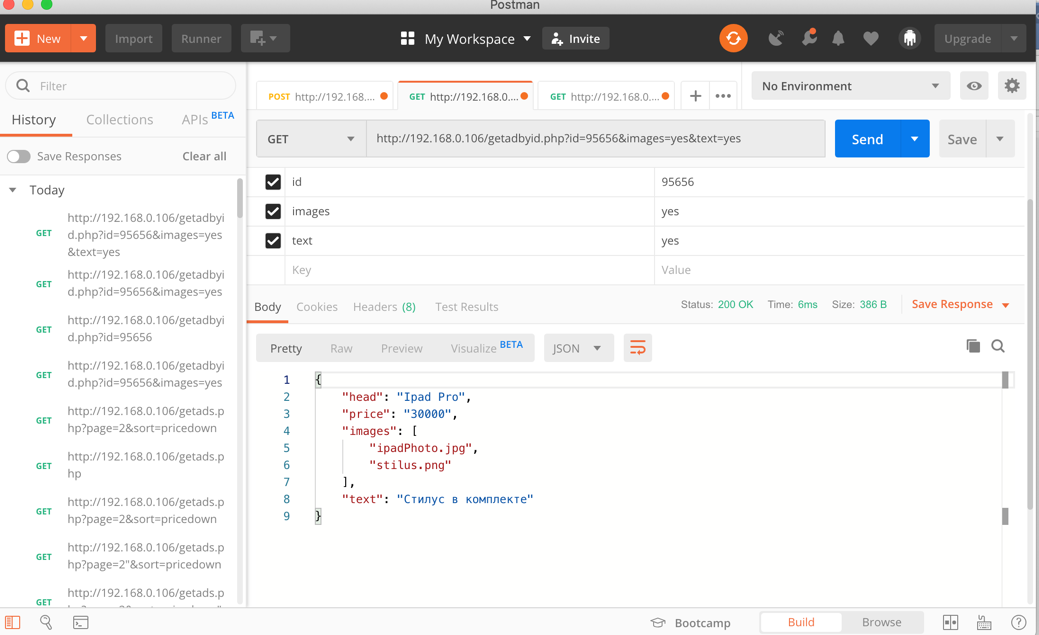
Отправляем запрос с одним параметром – id объявления, получаем заголовок, ссылку на первое фото и цену



Отправляем запрос с двумя параметрами – id и images = yes (получить все фото) – получаем заголовок, цену и массив со всеми ссылками на фото

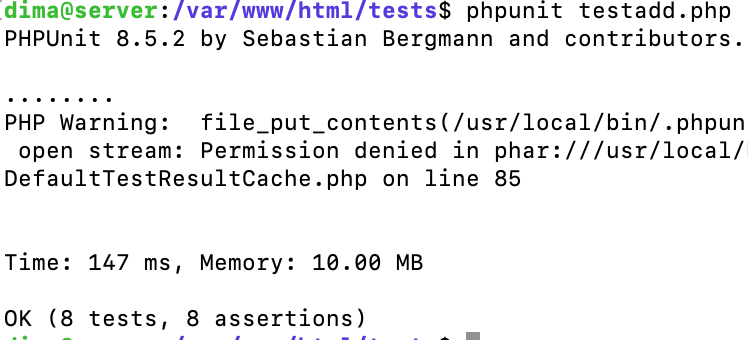


Добавляем третий параметр text = yes, также в ответе получаем описание объявления



**Шаг 6**: Написание Unit тестов

**/tests/testadd.php**

****

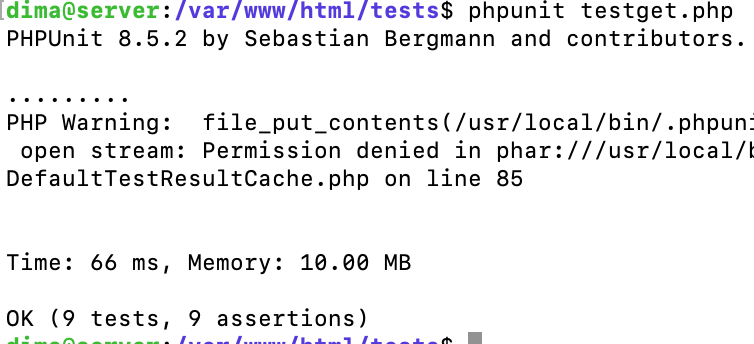
**testValidateFields()** – тестируем функцию validateFields, задача которой проверить все ли поля, пришедшие от пользователя отвечают требованиям приложения. Содержит 3 теста. Метод теста – сравнение ожидания и ответа функции.

**testAntiInjection()** – тестируем функцию antiSQLInjectionValidate, задача которой проверить все пришедшие поля на наличие кавычек, которые могут использоваться для SQL инъекций. Содержит 3 теста. Метод теста – сравнение ожидания и ответа функции.

**testGetUniqueID()** – тестируем функцию getUniqueID, задача которой вернуть уникальный ID объявления, сгенерированный случайно, но не содержащийся в базе данных. Создаем мок объекта mysqli, который в функции query вернет нам объект типа resultStub Метод которого fetch\_assoc вернет нам false (это значит что такого ID В базе данных не обнаружено, а соответственно сгенерированное значение Уникально) Функция getUniqueID должна вернуть любое число. Метод теста – проверка типа данных, который вернет функция.

**testAddAd()** – тестируем функцию addAd, задача которой записать в базу данных новое объявление с полями, пришедшими от пользователя. Создаем мок объекта mysqli, который в функции query вернет нам true Что означает что объявление успешно добавлено в базу данных Если функция addAd вернет true, значит объявление успешно добавлено в базу. Метод теста – сравнение ожидания и ответа функции.

**/tests/testget.php**

****

**testValidateID() –** тестируем функцию validateID, задача которой проверить валидность ID, полученного от пользователя. Метод теста – сравнение ожидания и ответа функции.

**testGetAd()** – тестируем функцию getAdByID, задача которой вернуть ассоциативный массив со всеми параметрами объявления. Создаем мок объекта mysqli, его метод query возвращает объект типа resultStubForGetAdFromDataBase, метод fetch\_assoc которого возвращает ассоциативный массив со всеми полями объявления (header - название столбца в базе head - название ключа для поля заголовок в структуре программы) проверяем наличие всех полей в ответе функции getAdByID. Содержит 5 тестов. Метод теста – проверка наличия ключа в ассоциативном массиве, которые возвращает функция.

**testGetIDS()** – тестируем функцию getIDS, задача которой вернуть ID 10-ти объявлений, в зависимости от того, какая страница запрошена пользователем, а также отсортированные в порядке, выбранным пользователем. Создаем мок объекта mysqli, его метод query возвращает объект типа resultStubForGetIDSFromDataBase метод fetch\_assoc которого возвращает ассоциативный массив только с одним ключом - id ровно 10 раз Так как функция getIDS возвраащет 10 ID объявлений. Метод теста – проверка количества возвращенных результатов.

**Шаг 7:** Внедрение кэширования при помощи Redis

Для работы используется Redis + PHP Redis

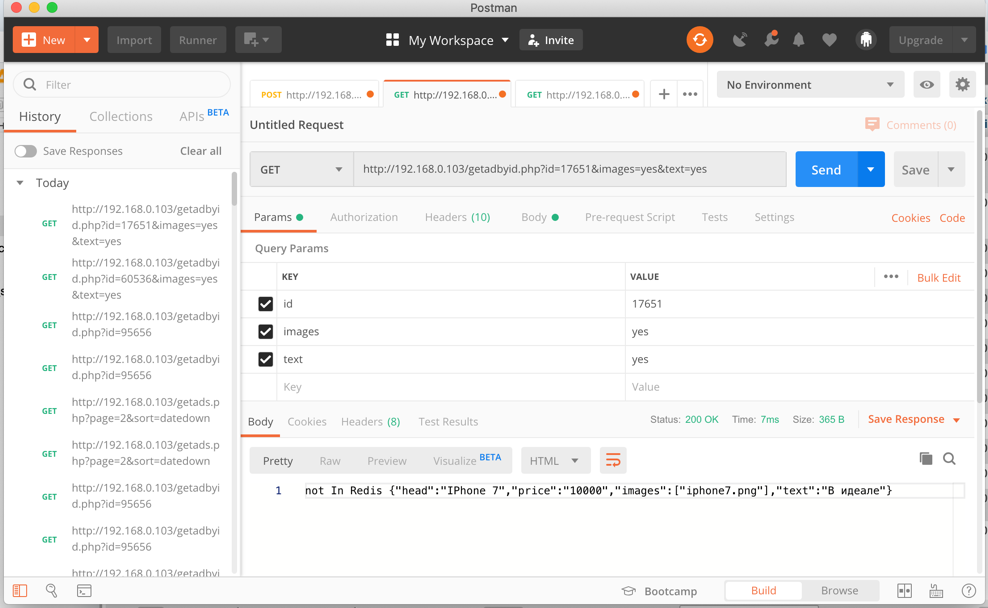
Была переработана функция getAdByID(), когда пользователь хочет получить то, или иное объявление, функция getAdByID() ищет данные запрошенного объявление с Redis, и, если не находит, обращается по старинке к MYSQL и заносит в Redis, таким образом при следующем обращении пользователь получит данные из кэша.

Если что-то пойдет не так, в коде написано какие строчки надо закоментить и раскоментить чтобы убрать Redis из этой функции.

Тестируем:

Временно для тестирования выводится строку «Not in Redis», если объявление запрашивается в первый раз и еще не было добавлено в кэш, и «In Redis» если объявление взято из кэша.

Просим вернуть нам объявление с ID 17651 – видим что данное объявление взято из базы



Просим вернуть это же объявление еще раз – видим, что данное объявление уже взято из кэша

