# Тестовое задание 1

* При разработке использовать фреймворк Yii2 с шаблоном Advanced.
* Готовый код разместить на GitHub
* В коде максимально использовать встроенные классы Yii2 (хелперы, контроллеры, модели, экшены и т.д.)
* Приветствуется качественное оформление кода: комментарии, соблюдение PSR, разделение на классы и т.д.

## 1. Создать таблицу и модель данных

* Использовать драйвер SQLite (база данных в файле). Расположение и имя файла выбрать самостоятельно.
* Использовать стандарный файл миграций Yii2. Предусмотреть в дальнейшем возможность перевода базы на другие типы (MySQL, MSSql и т.д.)
* На продуктовой среде запретить откат данной миграции.
* Структура БД ниже:
* create table url\_status  
  (  
   hash\_string varchar(32) not null,  
   created\_at datetime not null,  
   updated\_at datetime not null,  
   url varchar(255) not null,  
   status\_code int null,  
   query\_count int null,  
   constraint url\_status\_hash\_string\_uindex  
   unique (hash\_string)  
  );  
    
  alter table url\_status  
   add primary key (hash\_string);
  + hash\_string - md5-хеш от url
  + created\_at - дата и время добавления записи
  + updated\_at - дата и время обновления записи
  + url - адрес тестируемой страницы, например https://example.com/page/test
  + status\_code - HTTP-код ответа страницы
  + query\_count - количество запросов для данного адреса

## 2. Разработать REST-сервис

* Эндпоинт сервиса /CheckStatus. Количество элементов в блоке url[] не ограничено.
* **Пример запроса:** ```http request POST /CheckStatus Content-Type: application/json
* { “url”: [ “https://somesite\_1.com”, “https://somesite\_2.com”, “https://somesite\_3.com”, “https://somesite\_4.com” ] } ```
* Для каждого url из входного запроса:
  + При наличии данных по данному URL в таблице (проверить по хешу):
    - если с момента последнего обновления прошло более 10 минут:
      * обновить status\_code, (ожидание ответа - 5 секунд, при таймауте записывать код 0)
      * увеличить счетчик просмотров query\_count
      * обновить время updated\_at
    - если меньше 10 минут
      * получить status\_code из таблицы
      * увеличить счетчик просмотров query\_count
  + При отсутвии такого URL в таблице:
    - добавить новую запись:
      * hash\_string вычислить как md5(url)
      * created\_at, updated\_at - текущее время
      * status\_code - HTTP-код ответа страницы (ожидание ответа - 5 секунд, при таймауте записывать код 0)
      * url - обрабатываемый URL
      * query\_count = 1
* Сформировать ответ, в котором каджому URL сопоставлен полученный HTTP-код.
* **Пример ответа:** json { "codes": [ { "url": "https://somesite\_1.com", "code": 200 }, { "url": "https://somesite\_2.com", "code": 404 }, { "url": "https://somesite\_3.com", "code": 503 }, { "url": "https://somesite\_4.com", "code": 200 } ] }

## 3. Разработать консольную команду

Вызов команды yii check-status/statistics должен выводить в STDOUT информацию по всем запросам за последние 24 часа (по полю updated\_at), у которых статус ответа не равен 200. В выводе должны присутствовать поля url и status\_code. Формат произвольный.

## 4. Дополнительные требования

Выполнение дополнительных требований не обязательно, но желательно. Если какое-то из них не получается выполнить, просьба объяснить причину. Например: большие трудозатраты, никогда не сталкивался с такой задачей и т.д.

Если требования не выполняются по причине больших трудозатрат, просьба как минимум обозначить ход решения задачи и оценку времени на выполнение.

### 4.1 Оформить код в виде composer-библиотеки

* Весь код приложения должен быть в отдельном модуле и подключаться к Yii2-приложению командой composer require \_module-name\_
* После подключения не должно производиться дополнительных манипуляций с кодом для работы с сервисом
* Загружать модуль на packagist.com не обязательно

### 4.2 Вынести параметры модуля в конфигурацию

* Самостоятельно определить настройки, которые можно вынести в конфигурацию
* Вынести эти настройки в конфигурацию модуля/приложения

### 4.3 Оптимизировать скорость опроса хостов

* Необходимо реализовать параллельный опрос статусов хостов в п.2 (REST сервис)

### 4.4 Реализовать SOAP-сервис с аналогичным функционалом

* Добавить эндпоинт, на котором будет работать SOAP-сервис. Функционал повторяет п.3
* Сервис должен рабоать по WSDL-схеме, которую необходимо разработать самостоятельно
* По запросу GET /soap-endpoint?wsdl должна возвращаться WSDL-схема
* По запросу POST /soap-endpoint должен работать сам сервис