Тестовое задание CMS-разработчик

Данное тестовое задание, в первую очередь, направлено на оценку того как хорошо вы знакомы *с архитектурным стилем взаимодействия RESTfull API* для *стандартных операций CRUD в рамках языка PHP,* а также ваши навыки написания простейшего *frontend для web приложений*. Кроме того оценивается базовый уровень взаимодействия с платформой контейнеризации *Docker и его CLI инструмента Docker Compose*

За основу возьмем Open Source решение InstantCMS от компании InstantSoft. Умение поддерживать и дорабатывать данное решение, очень актуально для нас, так как часть наших web ресурсов уже сейчас работает на нём.

Ссылка на макет для тестового задания:
https://www.figma.com/design/sRGBz5XJaAlD0iX8bFHFlM/%D0%A2%D0%B5%D1%81%D1%82%D0%BE%D0%B2%D0%BE%D0%B5-%D0%B7%D0%B0%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5?node-id=0-1&node-type=canvas&t=5imZICv00gIJT1BT-0

Разворачиваем окружение.

Готовое и настроенное docker окружение под CMS, можно найти по этой ссылке. Там же находится необходимая пошаговая инструкция для данного процесса.

Ключевым моментом является файл **init.sh** - bash скрипт, в котором описаны все инструкции по работе с переменными окружения контейнеров (**файл .env**) Там же вы можете указать необходимую вам версию CMS. Данный bash скрипт реализован в качестве “своеобразного” CLI для удобной работы с окружением.

Установка компонента InstantCMS JSON API

Исходный код компонента и находится по адресу. Краткая инструкция компонента и несколько примеров, вы можете найти по адресу. Установка компонента осуществляется стандартно из маркетплейса административной панели CMS, при условии наличия правильных прав на файлы системы управления контентом у веб сервера из окружения docker. В случае успешной установки компонента вы увидите форму настроек логирования и форму создания ключа доступа API. Ключ доступа необходимо сгенерировать и выдать на него необходимые методы, с которыми планирует работать.

Задачи

- **Backend:**

  Необходимо разработать RESTfull API для операций CRUD в соответствии с

документацией

Смотри раздел “**Разработчикам методов**”. В парадигме instantCMS реализация данной задачи выражена в написание бизнес-логики нескольких методов операций CRUD (*create, read, update, delete*). Сохранение сущностей в базе данных CRM не *обязательно, но желательно*. Достаточно просто описать бизнес-логику метода.

- **Frontend:**

В рамках самостоятельно написанного *RESTfull CRUD* реализовать вывод информации на страницах CMS (достаточно одного метода работающего, например, со списком - ссылка на макет указана в описании). Вывод данной информации можно оформить используя *возможности instantCMS или просто используя стандартный frontend stack (html, css, javascript)*. Крайне желательно использование фреймворка *css bootstrap* (выбор версии на усмотрение разработчика). Использование *препроцессинга ccs, jquery и сборщиков фронтенда* по типу *gulp* или *webpack* - на усмотрение разработчика.

- **Тестирование API**

Для проверки корректности работы разработанного RESTful API, необходимо реализовать базовые тесты с использованием PHPUnit. Тесты должны охватывать ключевые CRUD-операции и проверять следующие сценарии:
Проверка получения списка элементов.
Проверка получения конкретного элемента по его ID.
Проверка, что при запросе несуществующего элемента API возвращает корректную ошибку (например 200, 204, 404, 500).
Проверка успешного создания нового элемента
Проверка успешного удаления элемента.

Результатом тестовой работы служит архивы созданных файлов:

- Отдельно для **backend**, отдельно для **frontend** *с сохранением их структуры* размещения в системе управления контентом *InstantCMS* или ссылка на **публичный git репозиторий** с историей разработки
- Файл базы данных .sql (с логином/паролем)
- Документация для проведения запросов по API к веб сайту
    - Эндпоинты для тестирования (тесты должны быть написаны с использованием PHPUnit):

- `/api/items` - для получения списка элементов (GET)
- `/api/items/{id}` - для получения элемента по ID (GET)
- `/api/items` - для создания нового элемента (POST)
- `/api/items/{id}` - для обновления элемента по ID (PUT)
- `/api/items/{id}` - для удаления элемента по ID (DELETE)