Тестовое задание

Общие положения:

- Данное тестовое задание не направлено на полноценную реализацию задачи, сколько на возможность продемонстрировать уровень понимая в таких вопросах, как: PHP, REST API, работа с СУБД, основы безопасности. **Помните**, у задания не существует единственно правильного решения, но каждому вашему решению должно находиться объяснение
- Постарайтесь показать свои лучшие практики написания кода
- В процессе выполнения можно использовать любые библиотеки и фреймворки, если они позволяют реализовать задачу качественно и быстрее, но в случае использования библиотек будьте готовы объяснить, почему используется конкретная библиотека, если появятся таковые вопросы
- Тем не менее, предпочтительна реализация на Laravel (8+), Yii2 (последняя стабильная версия), Symfony 5+. В качестве СУБД можно использовать MySQL 5.7 или PostgreSQL актуальной стабильной версии
- Предпочтительно выложить сайт в GIt репозиторий. По возможности, ведите историю коммитов, чтобы мы могли отследить Ваш процесс разработки\
- По возможности и при наличии времени, очищайте неиспользуемый шаблонный код

Общее описание:

Необходимо реализовать систему принятия и обработки заявок пользователей с сайта. Любой пользователь может отправить данные по публичному API, реализованному нами, оставив заявку с каким-то текстом,. Затем заявка рассматривается ответственным лицом и ей устанавливается статус Завершено. Чтобы установить этот статус, ответственное лицо должно оставить комментарий. Пользователь должен получить свой ответ по email.

При этом, ответственное лицо должно иметь возможность получить список заявок, отфильтровать их по статусу и по дате, а также иметь возможность ответить задающему вопрос через email.

Сущности:

Заявка	
id	Уникальный идентификатор
name	Имя пользователя - строка, обязательная
email	Email пользователя - строка,

	обязательная
status	Статус - enum(“Active”, “Resolved”)
message	Сообщение пользователя - текст, обязательный
comment	Ответ ответственного лица - текст, обязательный, если статус Resolved
created_at	Время создания заявки - timestamp или datetime
updated_at	Время ответа на заявку

Endpointы API:

Методы API должны быть документированы каким-нибудь средством документации на ваш выбор. Предпочтительно, с наличием песочницы.

- **GET /requests/** - получение заявок ответственным лицом, с фильтрацией по статусу
- **PUT /requests/{id}/** - ответ на конкретную задачу ответственным лицом
- **POST /requests/** - отправка заявки пользователями системы

Дополнения:

- Вы можете дополнять задачу отдельными методами и расширять объем входящих параметров, если посчитаете нужным
- Вы можете сделать авторизацию как и для публичного пользователя, так и для ответственного лица так, как вы посчитаете нужным
- Вы можете сами рассмотреть особенности безопасности входящих запросов, чтобы избежать кроссдоменных запросов или же наоборот, разрешить их безопасно
- Вам необязательно делать web интерфейс для отправки заявок и ответа на них
- Вам разрешено делать дополнительные улучшения, дополнительные фильтрации и методы API, но будьте готовы их прокомментировать
- Вам также разрешено переименовать поля сущностей и добавить новые, если вы считаете, что они приведут к большему пониманию того, что происходит в предметной области задачи или расширят ее (Например, endpoint удаления задачи, прикрепление за заявкой ответственного лица и д.р.)
- Дополнительно будет преимуществом, если вы представите себе, что заявки отправляются очень часто и храниться их может огромное количество
- Для отправки email можете воспользоваться NullObject реализацией какого-нибудь стандартного интерфейса, либо же сохранять email в виде plain файлов в директории временных файлов вашего фреймворка или по вашему выбору
- Unit тесты также приветствуются