# Тестовое задание для junior веб-разработчика (Laravel)

Тестовое задание состоит из двух частей: основной задачи и нескольких дополнительных. Основное задание сделайте обязательно. Дополнительные вы можете сделать по своему усмотрению и не обязательно делать все, можете сделать любые из 4-х. Но чем больше вы сделаете, тем будет лучше.

Ограничений по времени выполнения задания нет, присылайте, как сделаете. Кто первый пришлет тестовое задание, которое пройдет по нашим параметрам, того мы пригласим на стажировку. Выполненное задание отправьте, пожалуйста, в телеграм [@titaeva](http://t.me/titaeva) или на email [hr@make.st](mailto:hr@make.st). Укажите какой объем работы был выполнен, и что для этого использовалось.

# Основное задание

Создайте сервис для хранения постов пользователей, состоящий из API на Laravel и базы данных. СУБД рекомендуется использовать PostgreSQL или MySQL, из этих двух — на ваш выбор. Таблицы для базы данных должны создаваться и изменяться посредством миграций. API должна будет принимать данные в формате JSON, сохранять их в базу данных, выбирать ранее сохраненные и отдавать обратно клиенту, все так же в формате JSON. В рамках основной задачи никакой верстки делать не требуется.

В сервисе должны быть следующие сущности:

1. Пользователи, у которых есть уникальный идентификатор (далее ID) имя, email и зашифрованный пароль. В основном задании вход и регистрацию делать не требуется.
2. Посты, состоящие из уникального идентификатора (далее ID), заголовка длиной не более 100 символов, краткого описания длиной не более 255 символов, текста, не ограниченного по количеству символов и даты создания. Каждый пост должен иметь одного автора (то есть, принадлежать пользователю).

Для тестирования можете создавать пользователей и посты любым удобным вам способом — через интерфейс базы, через tinker или как угодно еще.

В API должны быть реализованы следующие методы:

1. GET /api/users — просмотр списка существующих пользователей. Разделение на страницы в этом методе не нужно. В ответ должен прийти массив всех пользователей в формате JSON. У каждого пользователя должны прийти ID, имя и email. Пароля в ответе быть не должно!
2. GET /api/posts — просмотр списка всех постов. В списке должно быть разделение на страницы по 15 постов. Номер страницы должен передаваться в get-параметре. То есть, на запрос /posts сервер должен возвращать первые 15 постов, на запрос /posts?page=2 — вторые 15 постов, и так далее. В ответ должен прийти массив постов в формате JSON. В каждом посте в массиве должны быть ID этого поста, заголовок, краткое описание и **имя автора.** Полный текст в этом запросе передавать **не нужно**!
3. GET /api/posts/{id} — просмотр конкретного поста. Вместо {id} должен передаваться ID поста. То есть, чтобы посмотреть пост с ID 10, нужно сделать запрос /posts/10. В ответ на запрос должен приходить один пост. В методе должны передаваться заголовок, краткое описание, **полный текст,** а также **имя и email автора**!
4. POST /api/posts — создание поста. В запросе должны передаваться ID пользователя-автора, заголовок, краткое описание и полный текст поста. Сервер должен из полей запроса создать в базе данных пост. После создания пост должен быть доступен для просмотра и отображаться в общем списке. В методе должна быть следующая **валидация**: заголовок не должен быть длиннее 100 символов и краткое описание не должно быть длиннее 255 символов!

Для тестирования API рекомендуется использовать Postman, Insomnia или встроенный HTTP-клиент. Но можете использовать и Curl или любой другой удобный вам клиент.

Код вашего приложения должен быть сохранен на Github или в любом другом git-репозитории. Желательно, чтобы в репозитории было несколько коммитов, а не один со всем кодом сразу.

# Дополнительные задания

## Простые

1. Создайте сиды для вашей базы данных, которые создадут несколько пользователей и постов, заполненных случайными тестовыми данными. Не забывайте, что у каждого поста должен быть пользователь-автор.
2. Добавьте в метод GET /api/posts параметр order, чтобы в запросе /api/posts?order=newer посты были отсортированы от более новых к более старым, а в запросе /api/posts?order=oldest — от старых к новым. Если параметра в запросе нет, то по-умолчанию посты должны сортироваться от новых к старым.

## Посложнее

1. Создайте систему оповещений пользователей. Пусть при создании поста всем пользователям отправляется уведомление о том, что создан новый пост. Создайте еще один метод GET /notifications?user\_id={id}, который будет получать все уведомления пользователя, id которого вы передали. Достаточно будет чтобы оповещения можно было посмотреть по этому методу, но будет плюсом, если сможете также сделать, чтобы они рассылались на электронную почту.
2. Создайте простую html-страницу по адресу /admin/users, которая будет представлять собой панель управления пользователями. На этой странице должны быть два блока: таблица со всеми пользователями (без паролей) и форма создания пользователя. В форме должны быть поля для имени, имейла и пароля, а также кнопка «Сохранить». При сохранении формы в базе должен создаться новый пользователь и произойти редирект на эту же страницу. Не забывайте, что пароль в базе вы должны будете сохранить в зашифрованном виде!

# На что будем смотреть в первую очередь

1. **Владение базовыми инструментами Larave**l: Eloquent ORM, реквестами, ресурсами, разделением маршрутов приложения на web-маршруты и api-маршруты.
2. **Соблюдение принципов KISS, DRY и YAGNI.**
3. **Умение пользоваться git.**

Будет плюсом читаемый и лаконичный код, соблюдение PSR, комментарии в сложных местах, отсутствие мусора в виде неиспользуемых методов, отладочного или закомментированного кода, владение продвинутыми инструментами Laravel такими, как разграничение доступа или каналы уведомлений.