# Front-End Challenge

Привет, давайте создадим простой блог.

Начнем с простого MVP. Он будет иметь функции сообщений и комментариев.

## Функциональные требования

- В блоге должна быть страница последних сообщений (`/` route)

- В блоге должна быть страница поста (`/ posts /: postId` route)

- В блоге должна быть страница создания поста (маршрут `/ posts / new`)

- Пользователь должен иметь возможность создать новый пост

## Технические требования

- Код должен быть написан с TypeScript

- [ESLint и Prettier] (https://dev.to/robertcoopercode/using-eslint-and-prettier-in-a-typescript-project-53jb) настройка для кодирования текста

- Мы хотим поддерживать SSR в нашем приложении. В качестве наиболее простого решения используйте [Next.js] (https://nextjs.org/), потому что он поддерживает SSR из коробки

- Пользовательский интерфейс должен быть построен с помощью [styled-components] (https://www.styled-components.com/)

- Убедитесь, что везде используются крючки React вместо компонентов на основе классов

- Redux как слой данных с любым промежуточным ПО, которое вы предпочитаете

- Проект должен иметь четкий README с шагами для его запуска

- Сетевой запрос обрабатывается с помощью [axios] (https://github.com/axios/axios)

- Используйте функциональное программирование, если вы ноу-хау. [Ramda.js] (http://ramdajs.com/) или [lodash / fp] (https://github.com/lodash/lodash/wiki/FP-Guide) - это плюс к вашему счету

- Разверните приложение на любом бесплатном хостинге для просмотра. Этот пункт не является обязательным, но приятно иметь :)

## API

Чтобы сохранить данные и работать с реальным API, мы будем использовать простой Node.js API, созданный за несколько минут. API размещен в облаке, и вы можете запросить его из любого места. Документация конечных точек API: https://documenter.getpostman.com/view/1917440/RzteTChV

## Дизайн

Дизайн зависит от вас. Просто, минималистично и чисто. В качестве общего примера проверьте [стандартный интерфейс Ghost] (https://blog.ghost.org/).

## Условия

- Код должен быть чистым, проходящим проверку ESLint и простым для понимания

- Задача обычно занимает \*\* от 4 до 6 часов \*\*. Если вам нужно больше времени, вы можете взять его, и это приветствуется, но результаты должны быть отправлены \*\* не позднее, чем через 48 часов после старта \*\*

- Умение писать чистую бизнес-логику оценивается выше, чем хорошая разметка

- Код вызова должен быть помещен в \*\* GitHub \*\* хранилище. Отправьте нам ссылку на хранилище сразу после этого. Благодарность!

Если у вас есть какие-либо вопросы о деталях конкурса, просите подробности, это будет оценено.

Удачи и с нетерпением ждем работы с вами!