Средствами **Angular** и **Angular CLI** реализуйте небольшое **SPA** приложение для блога с использованием **CRUD** операций и **REST API**. Для хранения, получения данных можно использовать **json-server** или ему подобные инструменты, либо реализация бекенда node.js своими силами, если это возможно, которые позволят создать описанный ниже функционал.

Должен быть реализован следующий список страниц:

1. Авторизация **/auth/sign-in**
2. Регистрация **/auth/sign-up**
3. Страница со списком записей блога **/posts**
4. Главная страница с точно такой же логикой и что и страница постов
5. Страница с детальной записью в блоге **/posts/:id**
6. Страница с созданием поста **/posts/create**
7. Страница с редактированием поста **/posts/edit**

**Регистрация:**

Переход на страницу регистрации не возможен авторизованному пользователю.

Форма регистрации должна состоять из:

* Имя
* Фамилия
* Email
* Роль (Администратор(**writer**), обычный юзер(**reader**))
* Пароль
* Подтверждение пароля

Требования к форме:

1. Поле имя должно быть не менее 3х символов.
2. Поле фамилия должно быть не менее 3х символов.
3. Поле email должно соответствовать формату email адреса.
4. Пароль и подтверждение пароля должны иметь валидацию на совпадение пароля
5. Минимальное кол-во символов пароля 6

**Авторизация:**

Переход на страницу авторизации не возможен авторизованному пользователю.

Форма авторизации состоит из:

* Email
* Пароль

Требования к форме:

1. Валидация на обязательное заполнение полей
2. При отсутствии пользователя с таким логином или паролем, выводить ошибку

**Страница постов:**

Страница по умолчанию должна состоять из 10 постов. Так же, должна быть возможность выбора кол-ва выводимых постов на страницу, например, в виде select с опциями 10, 20, 30.

Параметр с кол-вом постов должен сохранятся в виде **get** параметра в адресной строке и при обновлении страницы должно запрашиваться указанное кол-во постов в адресной строке.

Страница с постами должна иметь возможность переключения по страницам.

Переключение по страницам должно сохраняться в виде **get** параметра в адресной строке и при обновлении страницы должны запрашиваться посты указанного номера страницы.

Если, при заходе на страницу без дополнительных параметров, то номер страницы и лимит не должны отображаться в адресной строке.

Пример параметров: **mysite.com/?page=1&limit=20**

**Блок поста на странице постов:**

Блок поста должен содержать в себе:

* Заголовок
* Маленький текст
* Дату создания
* Дату изменения, если оно было.
* Автора поста (Полное имя)
* Кнопку удаления поста (Только для администратора)

Детальная страница поста:

Детальная страница поста содержать в себе:

* Заголовок
* Большой текст
* Последнюю дату изменения или дату создания
* Автора поста (Полное имя)

**Если текущий пользователь Администратор, должны отображаться кнопки для редактирования и удаления поста.**

Страница 404

Редирект на страницу 404 должен происходить, если какая либо страница по указанному адресу не найдена.

Страница редактирования и создания поста:

Переход на страницу редактирования и создания поста не возможен авторизованному пользователю.

Форма редактирования состоит из:

* Заголовок
* Маленький текст
* Большой текст
* Дата обновления поста (можно сделать скрытым полем)

Форма создания состоит из:

* Заголовок
* Маленький текст
* Большой текст
* Дата создания поста (можно сделать скрытым полем)

При редактировании поста, заголовок, маленький текст и большой текст должны подставляться в форму автоматически.

Основной макет:

Макет должен состоять из:

* **Шапки сайта**  
  В шапке сайта должно быть меню, главная, посты. При переходе на соответствующую страницу, ссылка должна подсвечиваться активным цветом.
* **Основного контента страницы**

Требования к функционалу:

* Все действия на сайте должны происходить без перезагрузки страницы.
* Подтверждение действий (удалить пост, редактирование поста) выполняется с помощью диалогового окна.
* После перезагрузки страницы состояние списка постов и его кол-во должно сохраняться.
* У пользователя должна быть возможность для выхода из профиля путём нажатия кнопки. Размещение её на свой вкус.
* Авторизация должна сохраняться при обновлении страницы
* Администратор может **удалять**, **редактировать**, **создавать** посты.
* Обычный авторизованный пользователь может **создавать и редактировать** только **свои** посты. **Удалять свои посты он не может.**
* Неавторизованный пользователь может только **читать** посты.

Технические требования:

* **Использовать json-server или ему подобные инструменты для работы с данными обязательно.**
* Диалоговые окна должны быть реализованы без использования "alert", "prompt" и "confirm".
* Верстка может быть выполнена с использованием UI библиотек (Bootstrap, Bulma и т.п.)
* Адаптивность.
* Логика приложения должна быть разбита на разумное количество самодостаточных компонентов.
* Json-server должен иметь возможность запуска командой, находящейся в файле package.json, в разделе scripts.
* Фронт должен иметь возможность запуска командой, находящейся в файле package.json, в разделе scripts.

**Формат данных в итоге должен быть таким:**

****

**Примечания:**

При создании поста, к нему необходимо привязать пользователя, который его создал.

Оформление тестового задания:

* Ссылка на публичный репозиторий (GitHub, GitLab) с исходным кодом.
* Код должен быть написан понятно и аккуратно, с соблюдением, без лишних элементов и функций, не имеющих отношения к функционалу тестового задания, снабжен понятными комментариями, если вы считаете, что они там нужны.
* Чистота и оформление кода — не менее важный фактор. Код должен быть написан в едином стиле (желательно в рекомендуемом для конкретного языка). Также к чистоте относятся отсутствие копипаста и дублирования логики.
* В README.MD проекта должно быть описано, как запустить проект и всё, что для этого понадобится.