## Описание

Предлагаем вам решить тестовое задание - реализовать компонент, который будет возможно внедрять в систему. Компонент оперирует сущностью Сотрудник. Для показа компонента предлагается реализовать демо-приложение состоящее из данного компонента. Приложение должно состоять из одной страницы которая отображает: список сотрудников в виде таблицы; модальное окно для создания сотрудника; модальное окно для редактирования сотрудника; модальное окно/tooltip при удалении сотрудника.

## Общие требования

* Предоставить **ссылку на репозиторий** в любом хостинг сервисе(github, gitlab). Будет проверяться наличие и содержание **коммитов** (больше одного). В README.md предоставить **инструкцию по запуску** приложения.
* При разработке использовать **Angular 2+/**React.
* Для эмуляции работы сервера воспользоваться **json-server** (<https://github.com/typicode/json-server>).
* Верстка должна быть **адаптирована**. Использовать любой фреймворк для верстки. Приложение должно корректно отображаться на максимальных и минимальных размерах экрана(десктоп/мобилка). Для проверки можно использовать DevTools браузера.
* Приложение должно работать на **IE11**, Chrome.
* Программный код должен быть оформлен по одному из общепринятых соглашений по оформлению. Например, Google JavaScript Style Guide (<https://google.github.io/styleguide/jsguide.html>), Angular Style Guide <https://angular.io/guide/styleguide>. При отправке укажите какой Style Guide использовали. **Будет проверяться командой lint**.
* Система должна корректно обрабатывать статусы ответов с ошибками и выводить понятные пользователю сообщения. Сообщения выводить с помощью любой **библиотеки нотификаций**. Либо написать самому (будет плюсом). Пример работы (<https://www.cssscript.com/demo/animated-queued-toast-messages-pure-javascript-siiimple-toast/>)

## Дополнительные требования

* Вместо фрейморка для верстки выполнить верстку самостоятельно через css.
* Вместо библиотеки нотификаций написать собственный toast notification.

## Описание сущности

Person (Сотрудник):

{

id: number,

firstName: string,

lastName: string,

}

API взаимодействия

GET api/v1/persons

Возвращает массив объектов типа Person:

[Person, Person, …]

GET api/v1/person/[id]

Где id – уникальный идентификатор сотрудника. Возвращает объект типа Person.

PUT api/v1/person/[id]

Где id – уникальный идентификатор сотрудника. В теле запроса передавать объект Person. Id должен быть null или undefined. Обновляет данные сотрудника согласно значениям, указанным в объекте Person в теле. Обновляет навыки сотрудника согласно указанному набору.

POST api/v1/person

В теле запроса передавать объект Person. Создаёт нового сотрудника в системе с указанными навыками.

DELETE api/v1/person/[id]

Где id – уникальный идентификатор сотрудника. Удаляет с указанным id сотрудника из системы.

## Статусы ответов

200 – успешное выполнение запроса. 400 – неверный запрос. 404 – сущность не найдена в системе. 500 – серверная ошибка (например, при обработки данных).

## Макеты

Ниже представлены примеры макетов для демо приложения





