Test Assignment

Разработка REST API

1. Задание

Задание делится на обязательное и часть по выбору.

1.1. Обязательная часть

Необходимо разработать REST API. REST API должен поддерживать следующие запросы:

1. Авторизация пользователя, начало сессии пользователя. Для авторизации 2 обязательных поля - email, password.
2. (если нужно) Окончание сессии пользователя.
3. Просмотр списка пользователей (только для авторизованных пользователей).
4. Просмотр пользователя по идентификатору (только для авторизованных пользователей).
5. Создание пользователя (только для авторизованных пользователей).
6. Удаление пользователя (только для авторизованных пользователей).
7. (будет плюсом) Обновление пользователя (только для авторизованных пользователей).

В системе должен быть начальный, заранее созданный пользователь.

Ответы необходимо отправлять в Content-Type: application/json.

Объект пользователя (TypeScript interface), можно добавить идентификатор (ID) или использовать для этого почту:

```
interface User {
    /**
     * A valid unique email.
     */
    email: string;
    /**
     * A string that has minimum length of 8, at least one upper-case, one lowercase English letter, one digit, one symbol from the list '~!@#$%^&*()_+{}[]'
     */
```

```
    password: string;
    /**
     * A non-empty string.
     */
    name: string;
    /**
     * A non-negative integer.
     */
    age: number;
}
```

1.2. Первый вариант - расширить REST API, без фронтенда.

Добавить API для управления постами.

1. Просмотр постов для пользователя (можно неавторизированным).
2. Создание поста для пользователя (пользователь может создать только пост от своего имени).
3. Удаление поста для пользователя (пользователь может удалить только свои посты).
4. (будет плюсом) Редактирование поста пользователя (пользователь может редактировать только свои посты).

Каждый пост обязан иметь не пустое строковое свойство text и дату последнего обновления:

```
interface Post {
    /**
     * Either `userId` or `userEmail` is required.
     * If you use ID for users, `userId` can have any type, but
justification is required.
     */
    userId?: unknown;
    /**
     * Either `userId` or `userEmail` is required.
     * If you don't use ID for users, a valid user email must be
present here.
     */
    userEmail?: string;

    /**
     * A non-empty string.
     */
    text: string;
    /**
```

```
     * An ISO 8601 date.
     */
    updatedAt: string;
}
```

1.3. Второй вариант - сделать фронтенд часть.

Необходимо разработать веб интерфейс на API. В интерфейс должны входить 2 страницы:

1. Страница логина (Sign In).
2. Страница списка пользователей (User List).

2. Требования к выполнению

- Необходимо создать **private git repository** в GitHub, BitBucket или GitLab на котором сохранить **source code** и пошарить к нему доступ для проверки.
- Выбор фреймворка остается за разработчиком.
- REST API должно быть безопасным и не допускать несанкционированный доступ к данным. HTTPS настраивать не обязательно.
- Выбор среды разработки остается за разработчиком.
- Проводить валидацию всех запросов.
- REST API должно быть задокументировано через Swagger/OpenAPI 3 или Postman Collection.
- Для авторизации использовать JWT токен
- В качестве базы данных использовать MongoDB.
- Код должен быть написан в стиле https://github.com/airbnb/javascript. Для проверки стиля кода рекомендуется использовать https://eslint.org/, однако можно использовать и другие инструменты. Вместо 100 длину строки кода можно выбрать между 80 и 120.
- В **git repository** должен быть файл README.md с документацией на английском языке.
- **Результат работы необходимо разместить в сети**, с публичным IP адресом или доменом. Выбор провайдера на выбор разработчика (например, Heroku, DigitalCloud, Google Cloud Platform, AWS, Azure). *Совет: базу данных удобно будет разместить в MongoDB Atlas.*
- Задание должно быть **production ready**, хорошо протестировано и не содержать каких-либо ошибок и **bugs**.

2.1. Дополнительные требования к фронтенду

- **Web application (frontend)** должен работать как минимум в Google Chrome браузере.
- Страница логин должна быть доступна без авторизации и в случае выхода из системы, пользователь должен быть перенаправлен на страницу логина.
- Страница пользователей должна быть доступна только для авторизованных пользователей. Также, пользователь должен уметь создавать и удалять пользователей либо на этой странице, либо на отдельной странице.
- Для реализации можно использовать любые средства разработки и технологии. Дизайн по выбору разработчика.
- Приложение может быть не адаптивным (non-responsive).
- Выбор способа выдавать фронтенд (serve website) часть остается за разработчиком (e.g. separate Node.JS process, Azure Blob Storage, Google Container Service).
- Код можно разместить либо в репозитории с серверной частью, либо в отдельном репозитории.

3. Приветствуется (не будет минусом)

- Использование HTTPS.
- Защита начального пользователя от удаления.

- Документирование исходного кода **на английском** через JSDoc (с описанием типов) либо использование **TypeScript**.
- Автоматические тесты (unit, integration, API или functional).
- Наличие возможности получения списка ошибок стиля кода в консоль или же их автоматическое исправление в проекте 1 - 2 командами из консоли.

3.1. Приветствуется в бекенде (не будет минусом)

- Архитектура приложения с несколькими слоями (multitier, n-tier architecture), использование шаблонов проектирования. **Но не стоит сильно усложнять :)**
- Разумное логированние приложения. Выбор подхода к логгированию остается за разработчиком, например, https://www.npmjs.com/package/log4js
- Возможность генерации документации REST API из кода, а не отдельным файлом, который меняется вручную.
- Возможность объединения валидации и документации.

3.2. Приветствуется во фронтенде (не будет минусом)

- Использование одного из распространенных фреймворков - React.JS, Angular 2+, Vue.JS.
- Использование одного из популярных библиотек стилей (например, Bootstrap, Material) или же использование своих стилей, объединенных единой идеей и руководством (guideline), которое нужно приложить.
- Использование SASS, SCSS или Less.