Bootstrap UI & React.js – Articles app

# Preface

Since the scope of the task is to be only Front-end oriented but also pretends to cover several tricky aspects of the UI related development, it has to step on features present in the newer browsers like “required” attribute, LocalStorage, History Management etc.

The Test app represents 2 aspects of a common web solution:

* Backend
  + The place where the user has the option to list and manage its data
  + Performing data validations
  + Storing it into the browser **Local Storage**
* Front-end
  + Simple UI slicing
  + Browser compatibility
  + Single page app behavior

The tools and libraries that should be used are

* Bootstrap UI – for all form elements + data grid in the “backend”
* ReactJS – for all the rest
* Any other needed for the developer (like a wrapper for the Local Storage through React)
* A useful reference is this documentation – <https://react-bootstrap.github.io/>

# The task:

## Backend part

It should be reachable on address like http://someapp.dev/admin/{locale}/articles

To be developed UI for listing and managing article items in 3 languages: **English**, **German** and **Bulgarian**

### Listing page

To be displayed a **grid** with the article items as follow (the used UI is personal choice☺):

|  |  |  |
| --- | --- | --- |
| + Add new | | |
| Date | **Article name** | **Actions** |
| *11.09.2015* | *Article 1* | *Edit Delete* |
| *11.09.2015* | *Article 2* | *Edit Delete* |
| *11.09.2015* | *Article 3* | *Edit Delete* |
| *11.09.2015* | *Article 4* | *Edit Delete* |

**NOTE: the grid data has to be read from the browser Local Storage!**

Actions:

* **Add new** – a button that refers to the “Add new page” screen
* **Edit** – such action is shown for **each record** in the listing and it opens the page for record editing
* **Delete** – such action is shown for **each record** and it allows a record deletion upon confirmation.

 Delete confirmation:

Are you sure you want to delete record: **RECORD\_NAME** ?

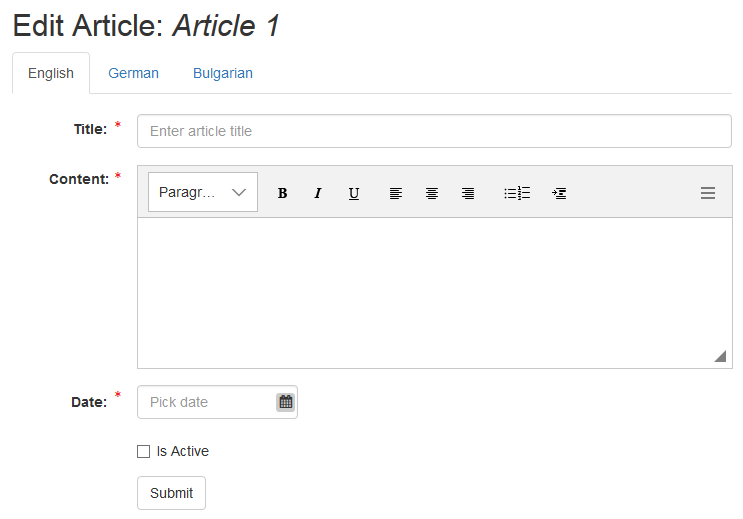
**YES NO**

### Add/Edit article item

It should be reachable on address like http://someapp.dev/admin/{locale}/articles/add and respectively http://someapp.dev/admin/{locale}/articles/edit/{id}

The required components are:

* Tabs
* TextBox
* Editor (an example one is this: <https://www.npmjs.com/package/react-bootstrap-editor>)
* DatePicker
* CheckBox
* Button



**NOTE: the items data has to be read and stored inside the browser Local Storage!**

On this page is shown a form with fields in **2 groups**:

1. **Locale based** – means that the data has to be entered for all system available languages
2. **Common** – properties that are not language based

Locale based fields

All these fields are placed in tab holders – according to the language:

* **Title** – a text field used to be set up the desired name of the page
* **Content** – a simple textarea – used to be defined the article content

Common fields

These fields are placed below the locale based such and are as follow:

* **Date** – a calendar picker
* **Is active** – a check box that sets if the current article will be active or no (respectively displayed or no in the “public” part)

Validations

* + Title, Content and Date are required fields

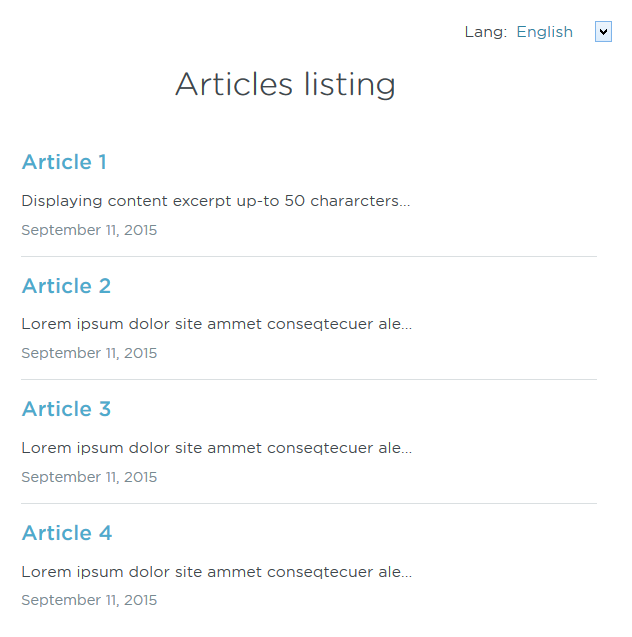
## Front-end part

It should be built using React templates. The Whole front-end side should act as a “Single Page Application” where the routing has to be WITHOUT using “#”

The locale of the page is changeable via drop-down placed at the top right corner

### Listing page

Here are shown all active article items that have to look like:



The displayed data is:

* **Article Title** – as link to the article details page that refers to ‘http://someapp.dev/{locale}/articles/{article-id}’
* **Content** – stripped up-to the 50th character
* **Article Date** – formatted as in the image above

### **Article details page**

A page that displays the full content of an article and that looks like



The displayed data is:

* **Article Title** – as link to the article details page that refers to ‘*..someapp.dev/{locale}/articles/{article-id}’*
* **Content** – the full content
* **Article Date** – formatted as in the image above

Actions:

* Previous – a link (again based to slug) to the previous article (if any)
* Next – a link (again based to slug) to the next article (if any)
* Back to the listing – a link to the articles listing page