ASI-2
React.js
Step By Step

2018

Ce Tp a pour objectif de concevoir une application cross technologie permettant d’appréhender la fonction et l’usage de chaque technologie.
Cette première partie vous permettra de mettre en œuvre l’application Web basée sur un socle React.js

Part I :
React.js

Table des matières

1. Introduction

Ce Tp se déroule en 6 étapes comme suit :

- **Step 0** : Création du socle d'application
- **Step 1** : Mise en place des premiers composants MainPanel, BrowseContentPanel
- **Step 2 :** Mise en place de composants de composants BrowsePresentationPanel, EditSlidPanel
- **Step 3** : Manipulation des actions et des reducers : selectedReducer,updateModelReducer
- **Step 4** : Création d'un Service de communication : service, reducer
- **Step 5** : Création d'une section d'ajout de contenu et drag and drop des contents
- **Step 6** : Création des commandes de navigation, connexion aux reducers et services, création du Watcher

CHAQUE ETAPE et SOUS ETAPE DEVRA ETRE VALIDEE PAR UN ENSEIGNANT AVANT DE PASSER A LA SUIVANTE.

L'évaluation prendra en compte votre compréhension de l'étape, la mise en œuvre et les bonnes pratiques de programmation.

2. Présentation générale

2.1. Présentation du projet dans sa globalité

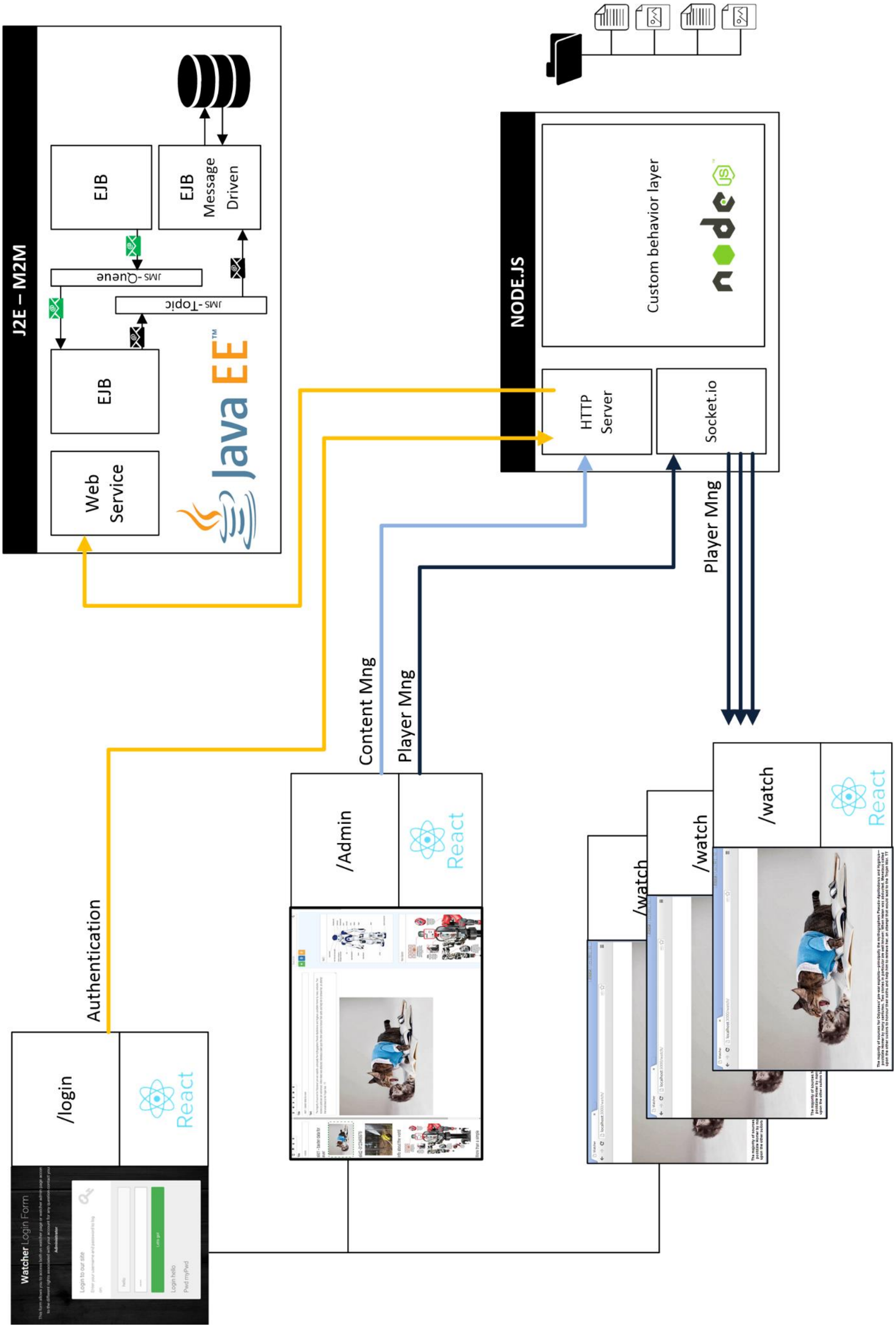


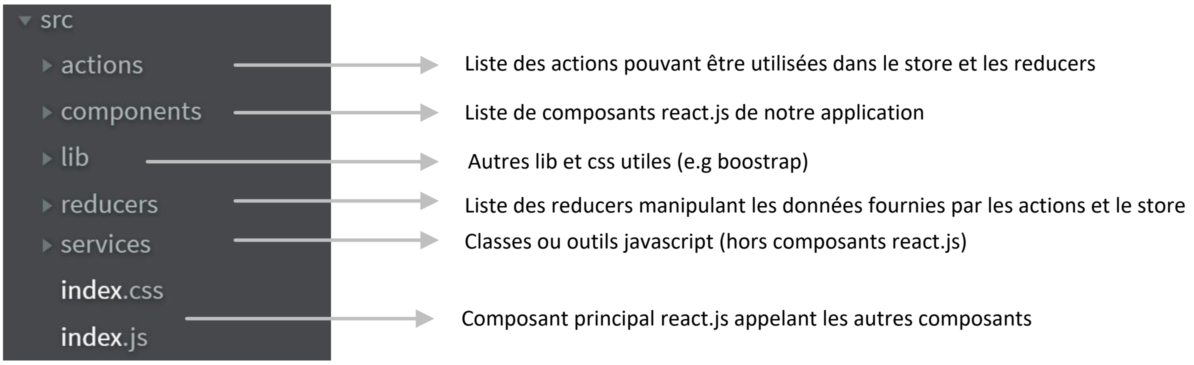
Figure 1: schema d'architecture

3. STEP 0 : Création du socle de l'application

Objectif : Prise en main bases de concepts de React.js, des bonnes pratiques, de l'usage des Composants, Découpage de l'application

3.1.Création d'un socle

3.1.1. L'objectif est de créer une structure de programmation comme suit :

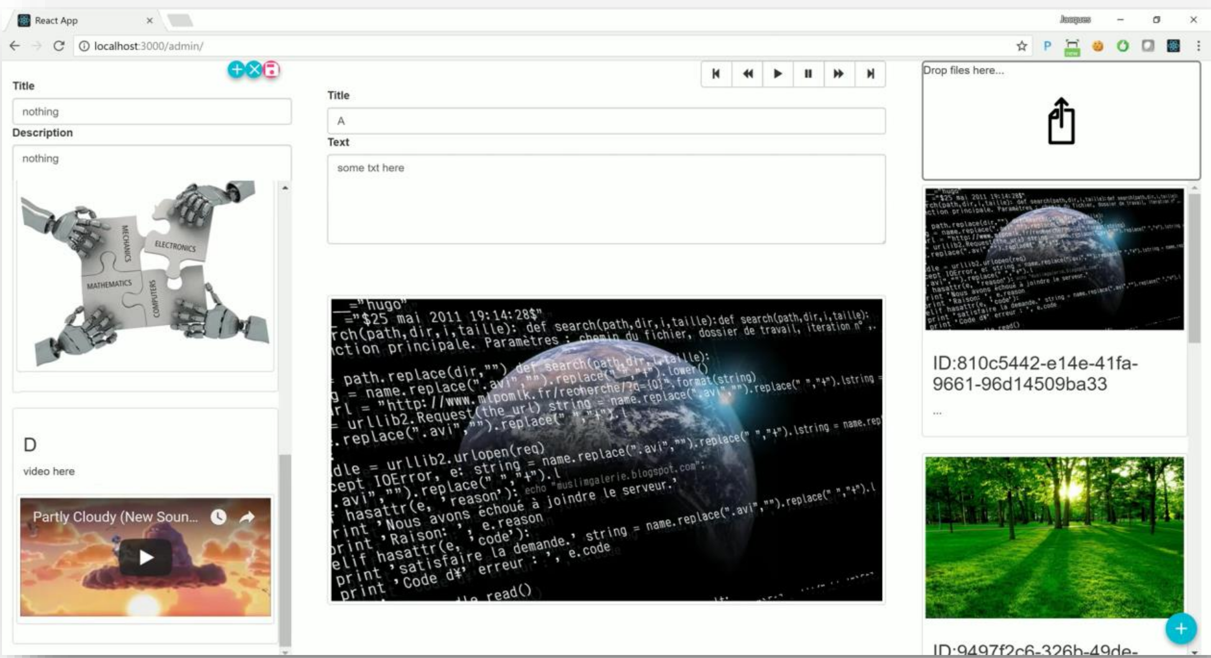


3.1.2.A quoi sert l'organisation une telle organisation dans un projet front ?

3.1.3.Lors de la création d'un composant où positionneriez-vous les css spécifiques à ce composant ?

3.2. Architecture du projet global : découpage en composants

Soit le projet suivant https://youtu.be/fwc7-z-qxxQ:

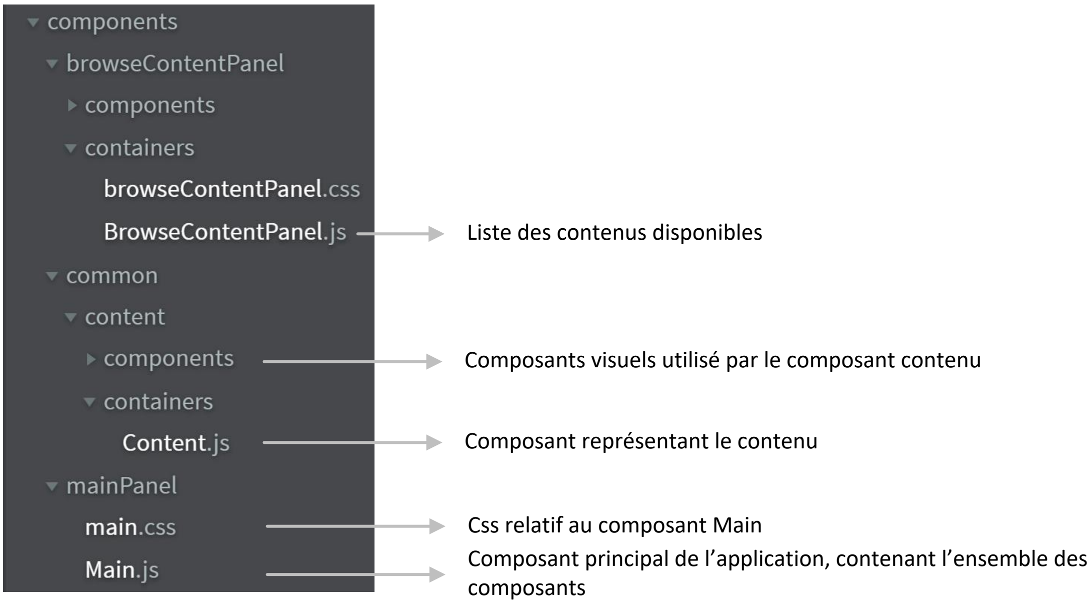


3.2.1.Proposer un découpage en composants de ce projet

4. STEP 1 : Création des premiers composants : MainPanel, BrowseContentPanel

4.1. Structure de l'application

Créer la structure suivante :



4.2. Composant Main

```
import React from 'react';
import './main.css';
import '../../lib/bootstrap-3.3.7-dist/css/bootstrap.min.css';

import * as contentMapTmp from '../../source/contentMap.json';


export default class Main extends React.Component{
    constructor(props) {
        super(props);

         this.state = {
             contentMap:contentMapTmp,
        }
    }

    render() {
        return (

            <div className='container-fluid height-100'>
                <div className="row height-100">
                    <div className='col-md-3 col-lg-3 height-100 vertical-scroll'>
                    </div>
                    <div className='col-md-6 col-lg-6 height-100'>
                    </div>
                    <div className='col-md-3 col-lg-3 height-100'>
                    </div>
                </div>
            </div>
        );
    }
}
```

4.2.1. A quoi sert la `extends React.Component` ?

4.2.2. Que représente `props` ? à quoi sert-il ?

4.2.3. Que représente `state` ? à quoi sert-il ?

4.2.4. Quelle est la fonction principale de `render()` ?
4.2.5. Modifier votre application react.js afin de prendre en compte ce composant

FAIRE VALIDER L'ETAT D'AVANCEMENT

4.3.Composant Content:

4.3.1.Créer un composant Content dans components/common/content/container possédant les propriétés suivantes :

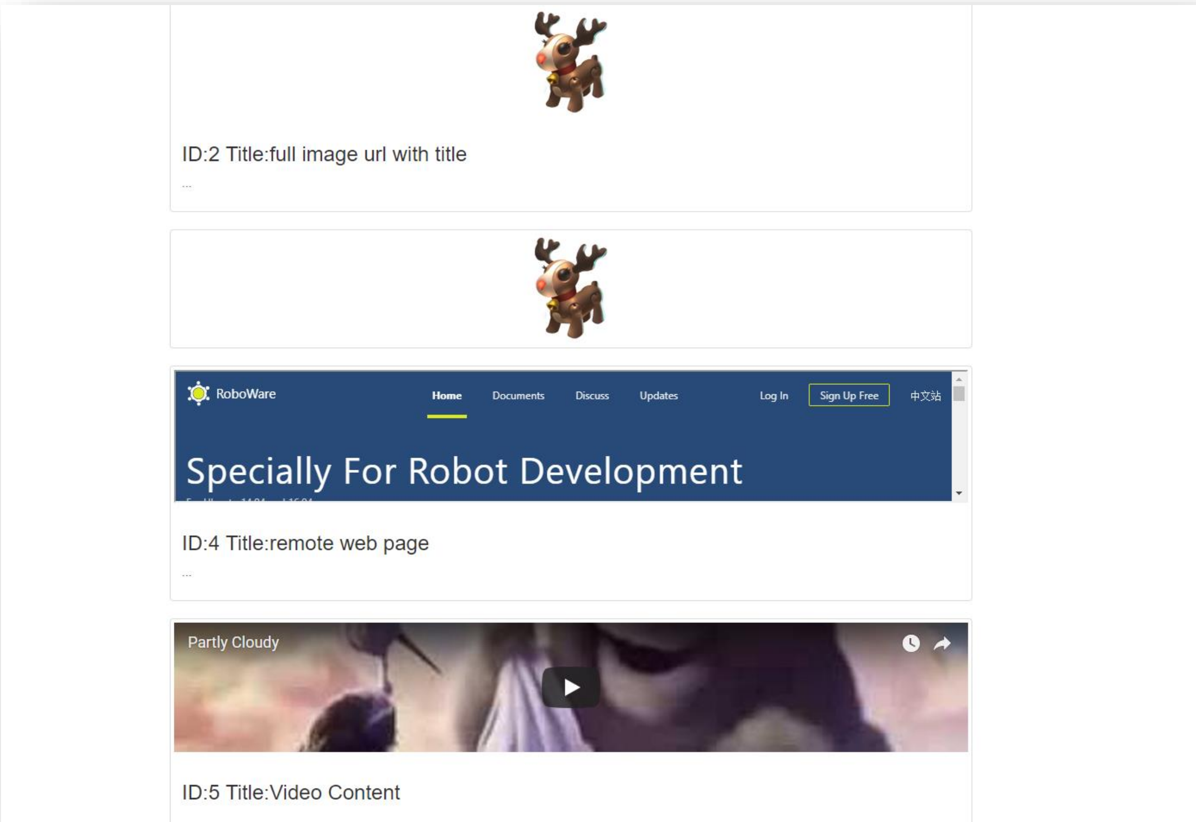
Propriétés d'entrée :

- id : id du contenu
- src : url ou base_64 du contenu
- type : type de contenu (img, img_url, video, web)
- title : titre du contenu
- onlyContent : boolean indiquant si il faut afficher toutes les propriétés du contenu ou simplement le visuel

Fonctionnalités :

- 2 affichages possibles simplement le visuel ou toutes les propriétés
- Utiliser plusieurs composants visuels (à placer dans components) permettant de traiter les différents cas d'affichages

4.3.2. Ajouter des composants `Contenu` au composant `Main` pour le visualiser



4.3.3. Pourquoi est-il intéressant de proposer des composants « visuels» ?

4.3.4. Que représente les différents répertoires de componants/common/Container (containers, components) ?

4.3.5. Pourquoi le composant `Container` a-t-il été placé dans un répertoire common ?

FAIRE VALIDER L'ETAT D'AVANCEMENT

4.4. Composant BrowseContentPanel

4.4.1. Créer un composant BrowseContentPanel comme précisé dans la section 4.1 possédant les propriétés suivantes :

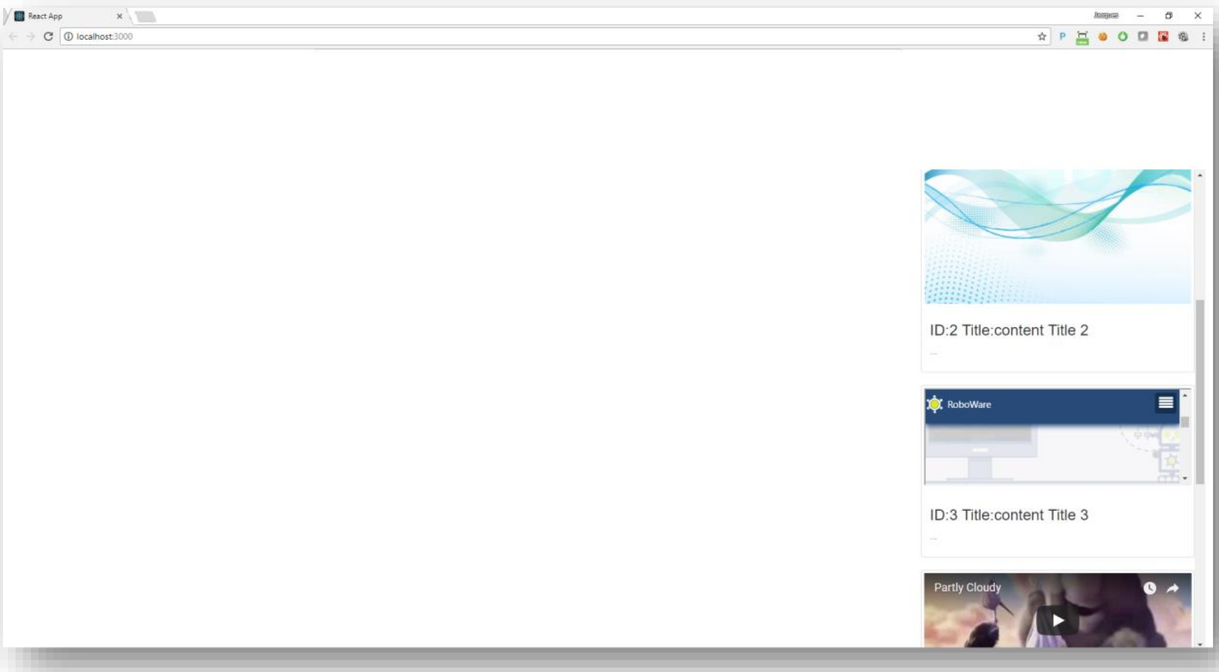
Propriétés d'entrée :

- contentMap: Dictionnaire des contenus disponible issu du fichier json contentMap.json

Fonctionnalités :

- Affiche l'ensemble des contenus disponibles dans contentMap

4.4.2. Ajouter un composant `BrowseContentPanel` au composant `Main` pour le visualiser



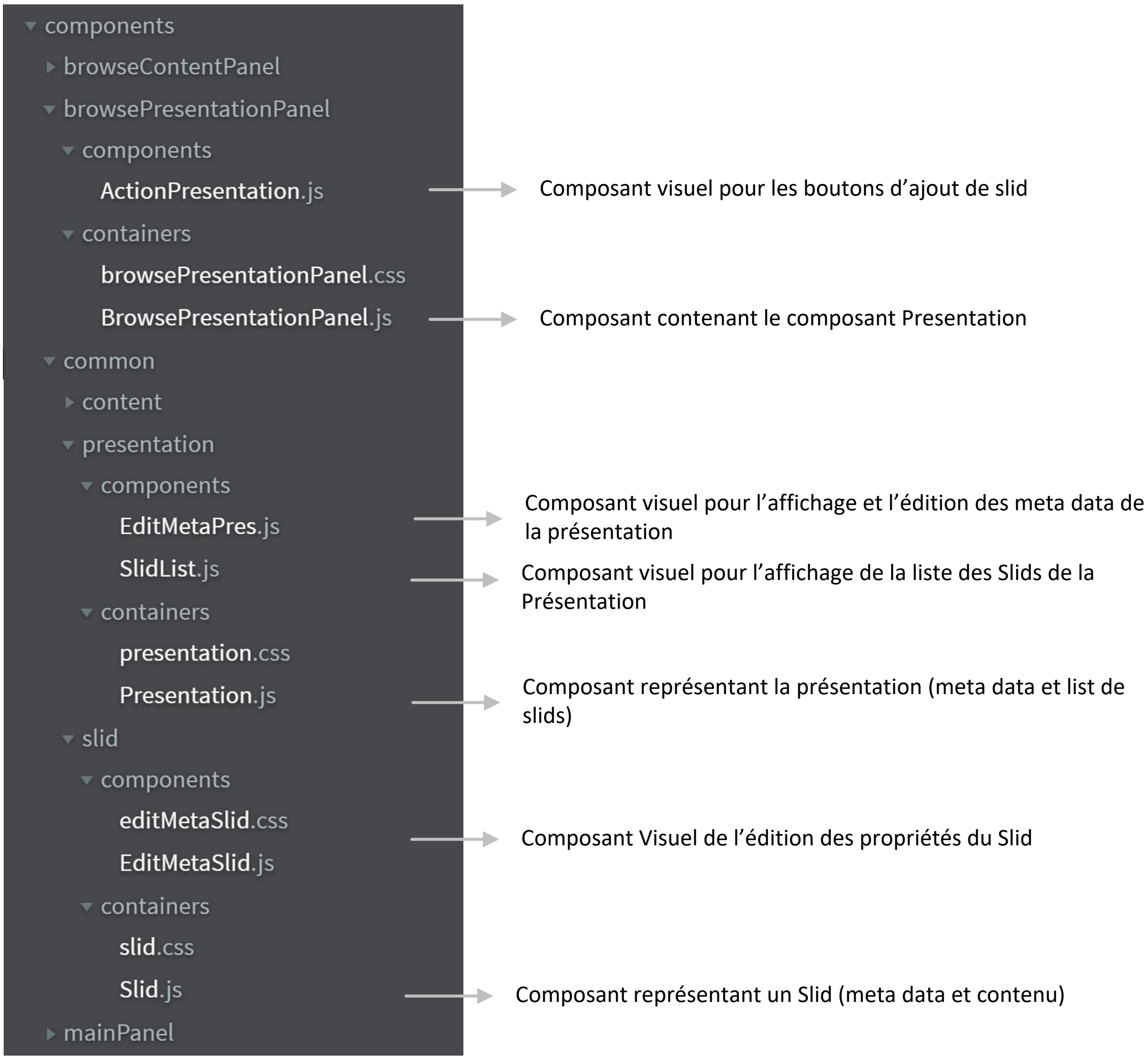
FAIRE VALIDER L'ETAT D'AVANCEMENT

5. STEP 2 : Création de composants de composants : BrowsePresentationPanel, EditSlidPanel

Objectif : Mise en place de l'ensemble de composants de composants, mise en place de binding simple pour des champs de saisies

5.1.Structure de l'application

Créer la structure suivante :



5.2.Composant Slid:

5.2.1. Créer un composant Slid dans components/common/slid/container possédant les propriétés suivantes :

Propriétés d'entrée :

- id : id du slid
- title : titre du slid
- txt : texte de description du slid
- content_id : **ID** du `Contenu` du slid
- contentMap : Dictionnaire des contenus disponible issu du fichier json contentMap.json

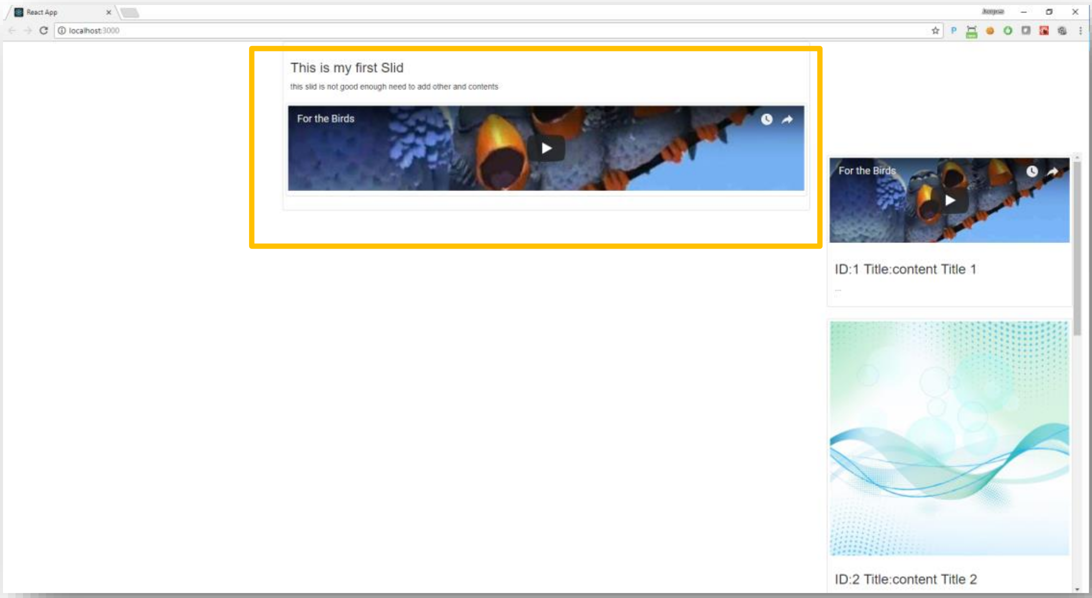
- displayMode : String : 'SHORT' ou 'FULL_MNG'

Fonctionnalités, 2 modes d'affichage :

- SHORT: Affiche les propriétés du slid et le composant Content associé
- FULL_MNG : Affiche un formulaire d'édition des propriétés du Slid et le composant Content associé

(note cet affichage sera réalisé dans la partie suivante, effectuer uniquement l'affiche SHORT ici)

Modifier le Composant Main pour afficher un slid



5.3.Composant EditMetaSlid:

```
import React from 'react';
import './editMetaSlid.css'
 export default class EditMetaSlid extends React.Component{
    constructor(props) {
        super(props);
    }
         render(){
        return (
                <div className="form-group">
                    <label htmlFor="currentSlideTitle">Title </label>
                    <input
                        type="text"
                        className="form-control"
                        id="currentSlideTitle"
                        onChange={this.props.handleChangeTitle}
                        value={this.props.title}
                    />
                    <label htmlFor="currentSlideText">Text</label>
                    <textarea
                        rows="5"
                        type="text"
                        className="form-control"
                        id="currentSlideText"
                        onChange={this.props.handleChangeTxt}
                        value={this.props.txt}>
                    </textarea>
                </div>
        );     }}
```

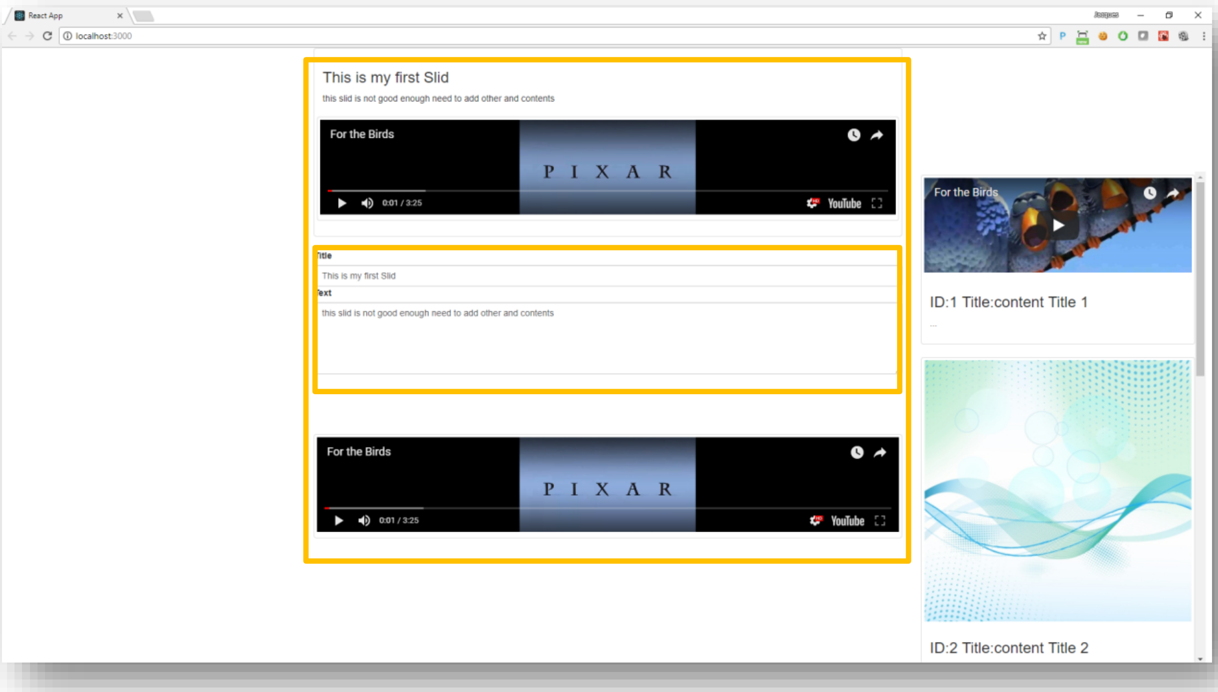
5.3.1. Pourquoi ce composant est-il considéré comme un composant visuel?

5.3.2. A quoi correspond l'attribut `onChange` ? que se passe-t-il si cet attribut n'est pas setté ?

5.3.3. A quoi servent les fonctions `this.props.handleChangeTxt` et `this.props.handleChangeTitle` ? qui définit ces fonctions ?

5.3.4. Compléter le composant `Slid` afin d'afficher le composant `EditMetaSlid` si `displayMode=FULL_MNG`

5.3.5. Modifier le composant `Main` afin d'affiche un `Slid` en mode `SHORT` et un `Slid` en mode FULL_`MNG`



FAIRE VALIDER L'ETAT D'AVANCEMENT

5.4.Composant Presentation:

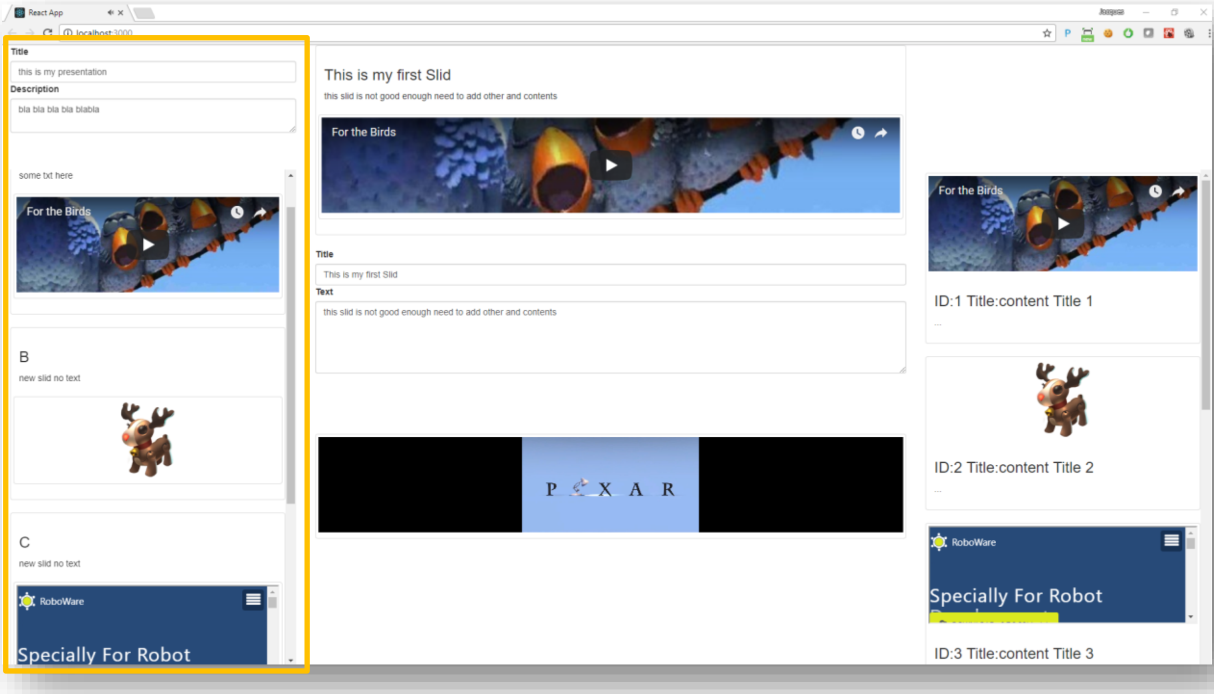
5.4.1. Créer un composant Presentation dans components/common/presentation/container possédant les propriétés suivantes :

Propriétés d'entrée :

- id : id de la présentation
- title : titre du slid
- description : texte décrivant la présentation
- slidArray :tableau de slids
- contentMap : Dictionnaire des contenus disponible issu du fichier json contentMap.json

Fonctionnalités :

- Affiche un formulaire d'édition des propriétés de la présentation
- Affiche la liste des slids de la présentation (displayMode : SHORT)

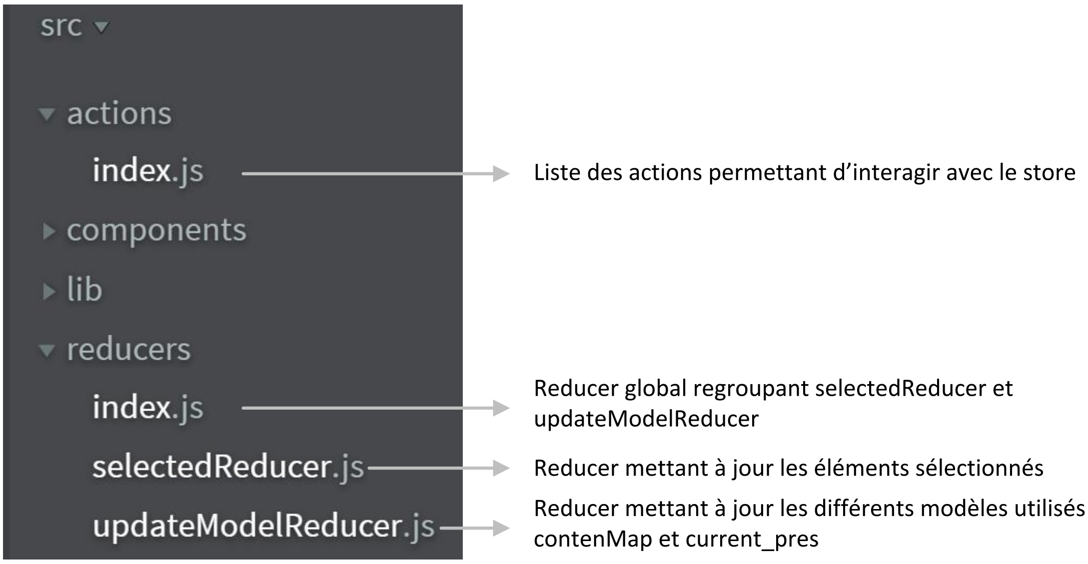


6. STEP 3 : Manipulation des actions et des reducers

Objectif : Comprendre l'intérêt de redux et sa chaine de communication. Manipuler les objets actions et reducers

6.1.Structure de l'application

Créer la structure suivante :



6.2.Mise à jour du Slid sélectionné

6.2.1. Création d'un composant d'affichage du Slid sélectionné : crée le répertoire /components/editSlidPanel/containers

6.2.2. Créer le composant `EditSlidPanel`

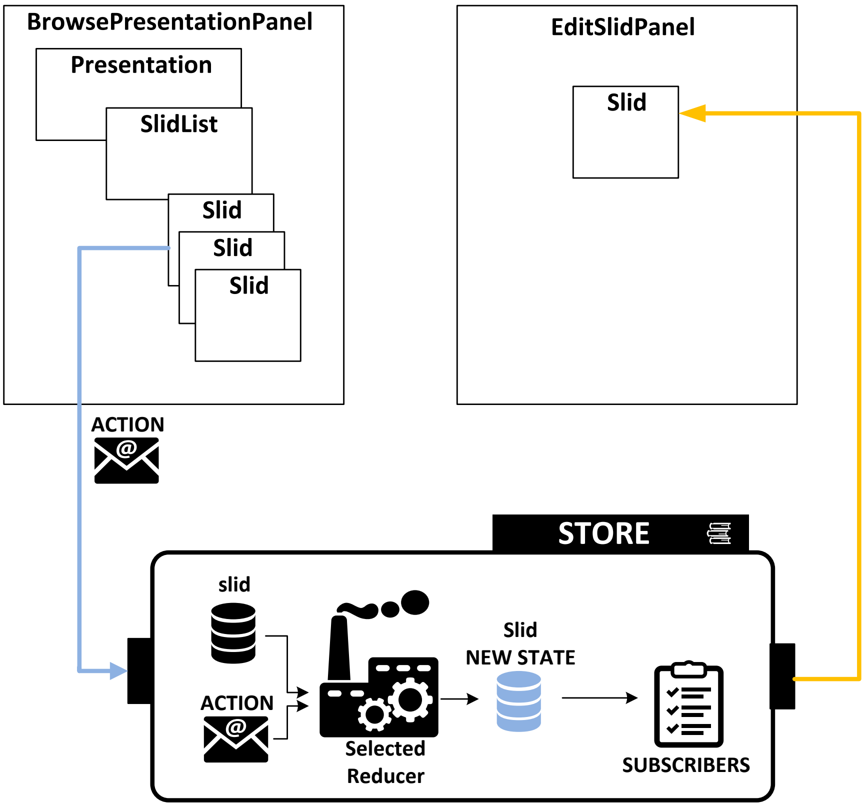
Propriétés d'entrée :

- selected_slid : slid à afficher
- contentMap : Dictionnaire des contenus disponible issu du fichier json contentMap.json

Fonctionnalités :

- Affiche le slid sélectionné en mode édition (`FULL_MNG`)

6.3. Création des actions et des reducers : selection



6.3.1. Créer une action permettant de mettre à jour le `Slid` sélectionné

```
export const setSelectedSlid=(slid_obj)=>{
    return {
        type: 'UPDATE_SELECTED_SLID',
        obj:slid_obj
    };
}
```

6.3.2. Créer le reducer selectedReducer permettant de mettre à jour le store avec le `Slid` sélectionné

```
const selectedReducer= (state={slid:{}},action) => {
    console.log(action);
    switch (action.type) {
        case 'UPDATE_SELECTED_SLID':
            const newState1={slid:action.obj};
            return newState1;
    default:
      return state;
    }
}
export default selectedReducer;
```

6.3.3. Créer un store en lui associant un reducer global contenant le `selectedReducer` dans le composant `Main`.

6.3.4. Utiliser l'outil `Provider` (react-redux) afin de fournir le store à tous les composant du `Main`

6.4. Mise à jour du composant Slid

6.4.1. Modifier le composant `Slid` afin de déclencher l'action créé lors d'un clic sur la div du slid

```
import React from 'react';
import './slid.css';
```

```
import EditMetaSlid from '../components/EditMetaSlid';
import Content from '../../content/containers/Content';


import { connect } from 'react-redux';
import {setSelectedSlid } from '../../../../actions'

 class Slid extends React.Component{

    constructor(props) {
                 ...
        this.updateSelectedSlid=this.updateSelectedSlid.bind(this);
    }
                 ...

    updateSelectedSlid(){
        const tmpSlid={id:this.props.id,
                       title:this.props.title,
                       txt:this.props.txt,
                       content_id:this.props.content_id};
       this.props.dispatch(setSelectedSlid(tmpSlid));
    }
                 ...
}

export default connect()(Slid);
```

6.4.2. A quoi sert la ligne `connect()(Slid)` ? Pourquoi n'y a-t-il pas d'argument à `connect()` ?

6.4.3. Que fait la ligne `this.props.dispatch(setSelectedSlid(tmpSlid))` ?

6.4.4. A quoi correspond `setSelectedSlid` ?

6.4.5. Que se passerait-il si un nouvel objet n'était pas créé (`tmpSlid`) ?

6.5.Mise à jour du composant EditSlidPanel

6.5.1. Modifier le composant `EditSlidPanel` afin de mettre à jour le slid sélectionné via le store

```
       ...

import {connect } from 'react-redux';

class EditSlidPanel extends React.Component{
       ...
const mapStateToProps = (state, ownProps) => {
  return {
    selected_slid: state.selectedReducer.slid,
  } };

export default connect(mapStateToProps)(EditSlidPanel);
```

6.5.2. Que permet de réaliser l'argument `mapStateToProps` ?

FAIRE VALIDER L'ETAT D'AVANCEMENT

6.6.Diffusion de ContentMap

6.6.1. Créer un nouveau reducer `updateModelReducer` permettant de manipuler la présentation courante et le contentMap.

```
var Tools = require('../services/Tools.js');

const updateModelReducer= (state={presentation:{},content_map:{}},action) => {
    console.log(action);
    switch (action.type) {
        case 'UPDATE_PRESENTATION':
```

```
            return ;              //TO DO
        case 'UPDATE_PRESENTATION_SLIDS':
            return;               //TO DO
       case 'UPDATE_CONTENT_MAP':
            return ;              //TO DO
        case 'ADD_CONTENT':
            return;               //TO DO
    default:
      return state;
    }
}


      export default updateModelReducer;
```

6.6.2. Ajouter `updateModelReducer` au `globalReducer`

6.6.3. Créer une nouvelle action `updateContentMap` permettant de mettre à jour le contentMap

6.6.4. Modifier le composant `Main` de façon à mettre à jour le `contentMap` du store

```
...
import * as contentMapTmp from '../../source/contentMap.json';
import * as presTmp from '../../source/pres.json';
...
import { createStore } from 'redux';
import globalReducer from '../../reducers';
import {updateContentMap,updatePresentation} from '../../actions';
import { Provider } from 'react-redux';
...

const store = createStore(globalReducer);


export default class Main extends React.Component{
    constructor(props) {
        super(props);

             ...

        store.dispatch(updateContentMap(contentMapTmp));
    }
```

6.6.5. Modifier tous les composants afin qu'ils mettent à jour le `contentMap` automatiquement par l'intermédiaire du store.

6.7.Mise à jour des modifications des Slids

A cette étape d'avancement les modifications des slids sont prises en compte uniquement localement (`EditSlidPanel`). Afin de propager les modifications à l'ensemble de l'application les différentes étapes suivantes doivent être réalisées :

6.7.1. Comme demandé précédemment pour le `contentMap` , diffuser la présentation courante à l'ensemble des composants en ayant besoin (essentiellement `Presentation`) :

- 6.7.1.1. Créer une action `updatePresentation` permettant de diffuser une mise à jour de la présentation
- 6.7.1.2. Compléter le `updateModelReducer` (`action.type=UPDATE_PRESENTATION`) afin de remplacer la présentation courante du store et de diffuser la nouvelle
- 6.7.1.3. Modifier le composant `Main` afin qu'il diffuse la mise à jour de la présentation courante (comme réalisé pour le `contentMap`) .

6.7.2. Mise à jour la présentation de l'application lors de modification de Slid

- 6.7.2.1. Créer une nouvelle action `updateSlid` permettant de déclencher la mise à jour d'un `Slid`
- 6.7.2.2. Dans le composant `EditSlidPanel`, créer une fonction `updateCurrentSlid(id,title,txt,content_id)` permettant de lancer l'action `updateSlid` lorsque le Slid courant est modifié (titre, text, contenu)

FAIRE VALIDER L'ETAT D'AVANCEMENT

7. STEP 4 : Création d'un Service de communication : service, reducer

Objectif : Utiliser un service de communications web services, utilisation du store dans un service autre qu'un composant React.js

7.1.Création d'un Service de communication

7.1.1. Afin de communiquer avec le serveur utiliser le fichier Comm.js fourni à positionner dans le répertoire /services

7.1.2. Que permet de réaliser l'exemple de fonction suivante :

```
loadPres(presId,callback,callbackErr){
 axios.get('/loadPres')
   .then(function (data) {
     var size = Object.keys(data.data).length;
     let loadedPres=""
     if(size >0){
         loadedPres=data.data[Object.keys(data.data)[0]];
     }
       callback(loadedPres);
    })
    .catch(function (error) {
       callbackErr(error);
    });
}
```

7.1.3. Que sont respectivement `callback` et `callbackErr` ?

7.1.4. Modifier le composant Main afin de pouvoir mettre à jour la présentation courante et le contentMap depuis le serveur (`store.dispatch`)

FAIRE VALIDER L'ETAT D'AVANCEMENT

8. STEP 5 : Création d'une section d'ajout de contenu et drag and drop

Objectif : Utiliser les évènements drag and drop de javascript, mise à jour du contenu des Slids

8.1. Mise en place des évènements drag and drop

8.1.1. Modifier le composant `Content` afin qu'il autorise le drag du composant et qu'il déclenche la fonction `drag(ev)`

8.1.2. Modifier le composant `Slid` afin qu'il autorise les drop et déclenche une fonction `drop(ev)` .

Note : vous trouverez des informations sur les évènements de drag and drop au lien ci-dessous :

https://www.w3schools.com/html/html5_draganddrop.asp

8.2. Mise à jour du Slid modifié

8.2.1. Créer une nouvelle action `updateDraggedElt` permettant de désigner un contenu dragged (id sera suffisant).

8.2.2. Modifier le reducer `selectedReducer` permettant de mettre à jour le composant courant dragged

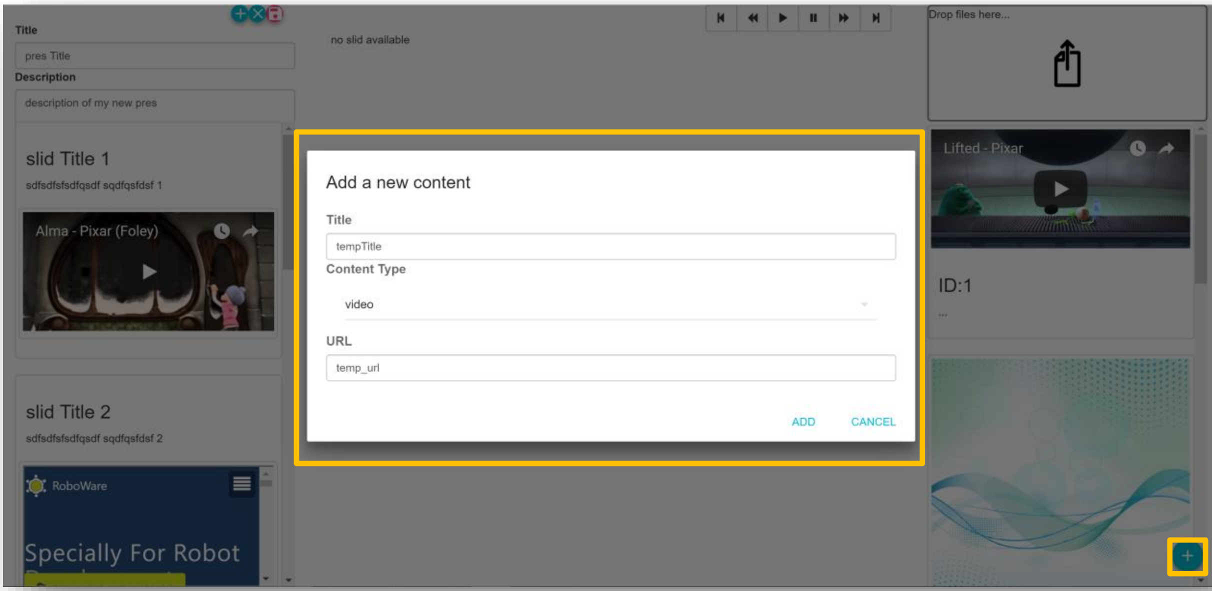
8.2.3. Modifier le composant `Content` afin qu'il déclenche l'action `updateDraggedElt` lorsqu'un contenu est dragged.

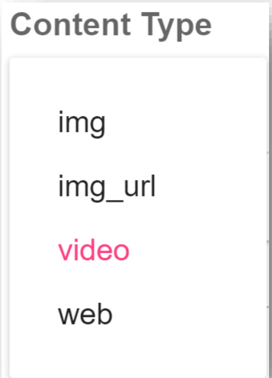
8.2.4. Modifier le composant `Slid` afin qu'il écoute et enregistre les modifications du contenu courant dragged

8.2.5. Modifier la fonction `drop(ev)` du composant `Slid` afin qu'elle déclenche une mise à jour du slid sélectionné avec le nouveau contenu.

8.3. Section d'ajout

A l'aide de material-ui (Dialog http://www.material-ui.com/#/components/dialog) créer une zone d'ajout (nouveau composant appelé `AddContentPanel`) de contenu comme précisé dans les visuels ci-dessous :





8.3.1. Ajouter un bouton d'ajout dans le composant `browseContentPanel` permettant d'ouvrir la fenêtre Dialogue présenté ci-dessus.

8.3.2. Créer le composant `AddContentPanel` permettant de définir le titre du contenu, le type de contenu (img, img_url, video, web) et la source du contenu.

8.3.3. Créer une nouvelle action `addContent` permettant d'ajouter un `Content` dans le `contentMap`.

8.3.4. Modifier le reducer `updateModelReducer` afin d'ajouter un nouveau content dans le `contentMap` (génération automatique de uuid à l'aide de Tools).

8.3.5. Ajouter un bouton Add au composant `AddContentPanel` permettant de déclancher l'action `addContent` .

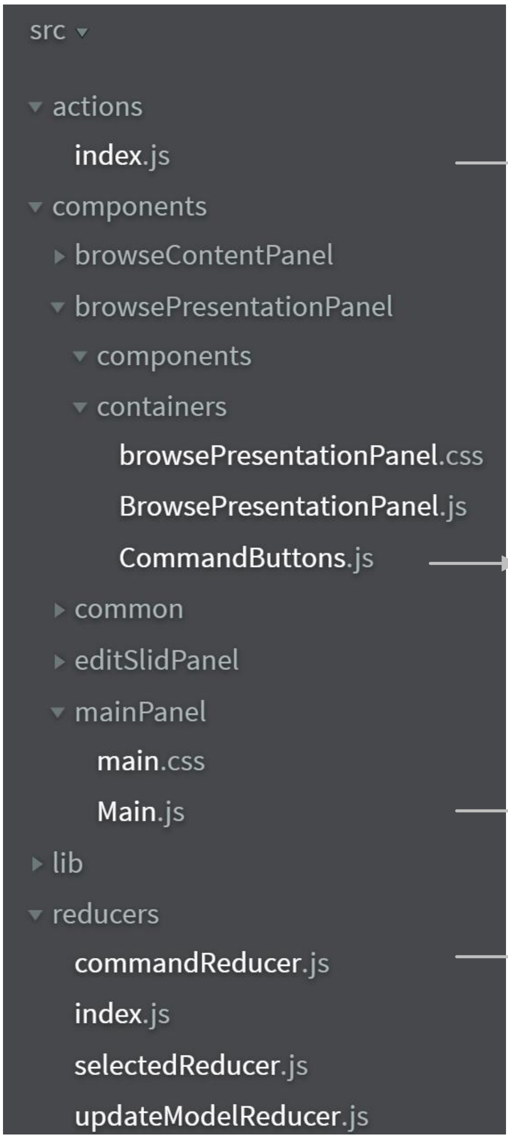
FAIRE VALIDER L'ETAT D'AVANCEMENT

9. STEP 6 : Slid Navigation, presentation Add Slid et Save action

Objectif : Modifier la présentation général en ajoutant des Slids, utiliser des boutons déclenchant des évènement réseaux (socket.io)

9.1.Composant de modification de la présentation Add, Remove, Save

9.1.1.Créer la structure suivante :



Ajouter une action pour la sauvegarde de la présentation

Composant permettant l'ajout, la suppression de Slid et la sauvegarde de la présentation courante

Ajoute de listener directement sur le store pour mettre à jour la présentation courante

Reducer en charge de générer les ordre de sauvegarde de la présentation ainsi que les ordres de navigations

9.1.2. Créer le composant `CommandButton` contenant 3 boutons `Add`, `Remove` et `Save`

9.1.3. Créer des handler pour `Add` et `Remove` déclenchant l'action `updateSlid` utilisant le reducer `updateModelReducer` pour ajouter ou supprimer un slid

9.1.4. Créer le reducer `commandReducer` se déclenchant sur l'action `COMMAND_PRESENTATION` et retournant l'action à effectuer

9.1.5. Créer un handler pour Save déclenchant l'action `sendNavCmd` utiliser dans le reducer `commandReducer`

9.1.6. Modifier le composant Main de façon à ce qu'il intercepte le retour du reducer `commandReducer` et lance l'action de sauvegarde sur le service `Comm.js`

```
...
var Comm = require('../../services/Comm.js');

const store = createStore(globalReducer);

export default class Main extends React.Component{
    constructor(props) {
        super(props);
         this.comm=new Comm();
         this.state = {
             contentMap:contentMapTmp,
             current_pres:presTmp,
        }
        //send action to the store for update the current contentMap
        store.dispatch(updateContentMap(contentMapTmp));
        //send action to the store for update the current presentation
        store.dispatch(updatePresentation(presTmp));
        store.subscribe(() => {
             this.setState(
             {presentation:store.getState().updateModelReducer.presentation});
             this.setState(
                    {contentMap:store.getState().updateModelReducer.content_map});
             if(store.getState().commandReducer.cmdPres == 'SAVE_CMD'){
                    this.comm.savPres(
                        store.getState().updateModelReducer.presentation,
                        this.callbackErr
                    );
                }
        });
         // Bind local function to the current object
        this.loadContentUpdate=this.loadContentUpdate.bind(this);
        this.loadPresUpdate=this.loadPresUpdate.bind(this);
        this.callbackErr =this.callbackErr.bind(this);


        //FIRST ACTIONS
       // try to load the contentMap from the server
        this.comm.loadContent(this.loadContentUpdate,this.callbackErr);
       // try to load the presentation from the server
        this.comm.loadPres(0,this.loadPresUpdate,this.callbackErr);
       // create the sokect connection between the server and the web browser
        this.comm.socketConnection(this.state.uuid);
    }

    loadContentUpdate(data){
        //send action to the store for update the current contentMap
        store.dispatch(updateContentMap(data));
    }

    loadPresUpdate(data){
        //send action to the store for update the current presentation
        store.dispatch(updatePresentation(data));
    }

    callbackErr(msg){
        console.error('Network Failure ?');
        console.error(msg);

    }
...
```

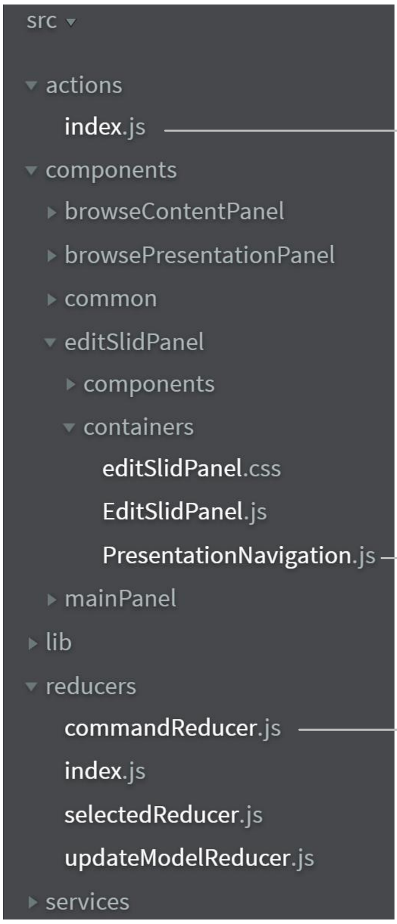
9.1.7. Pourquoi le composant `CommandButton` est il positionner dans les containers au même niveau que `BrowsePresentationPanel` ?

9.1.8. Que veut dire la fonction `store.subscribe(() => {…}) ;` ?

FAIRE VALIDER L'ETAT D'AVANCEMENT

9.2.Composant de Navigation de la présentation

9.2.1.Créer la structure suivante :

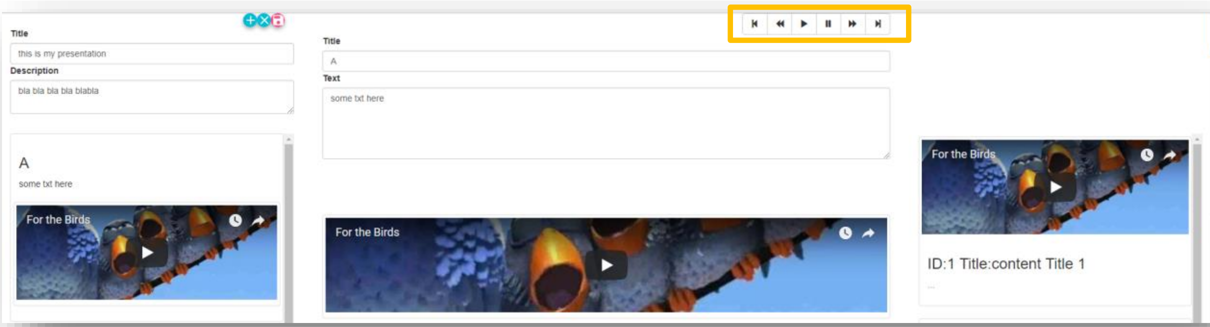


Ajouter une action permettant d'envoyer une information de navigation

Composant permettant d'envoyer les ordres de navigations aux watcher

Reducer traitant les actions liées à la navigation au sein de la présentation

9.2.2.Créer un nouveau composant `PresentationNavigation` composer des boutons : `Start, End, Previous, Next, Last, First` ?



9.2.3. Créer une action permettant de déclencher les évènements de navigation

9.2.4. Créer **UN** handler se déclenchant sur l'usage des boutons précédemment créés

9.2.5. Modifier Commande reducer afin qu'il réponde aux évènements de navigation

9.2.6. Modifier le Main.js afin qu'il puisse déclencher le services Comm et les évènement socket.io associés

9.2.7. Qu'est ce que socket.io ? en quoi ce services est-il intéressant ?

9.2.8. Utiliser votre application Node.js et le watcher fourni pour tester votre application

FAIRE VALIDER L'ETAT D'AVANCEMENT