

Contents

1 Source Code

- 1.1 index.html
- 1.2 actionTypes.js
- 1.3 authUser.js
- 1.4 authUserResult.js
- 1.5 getServiceList.js
- 1.6 index.js
- 1.7 logoutUser.js
- 1.8 receiveService.js
- 1.9 receiveServiceList.js
- 1.10 updateUser.js
- 1.11 changeDashboardPage.js
- 1.12 changeSelectedModel.js
- 1.13 changeSidebarItem.js
- 1.14 createAttribute.js
- 1.15 createEntry.js
- 1.16 createModel.js
- 1.17 deleteAttribute.js
- 1.18 deleteAttributeLocally.js
- 1.19 deleteEntry.js
- 1.20 deleteEntryLocally.js
- 1.21 deleteModel.js
- 1.22 deleteModelLocally.js
- 1.23 receiveAttribute.js
- 1.24 receiveEntry.js
- 1.25 receiveModel.js
- 1.26 selectAttribute.js
- 1.27 updateAttribute.js
- 1.28 updateAttributeLocally.js
- 1.29 updateModel.js

1.30	updateModelLocally.js	
1.31	updateService.js	
1.32	updateServiceLocally.js	
1.33	updateValue.js	
1.34	updateValueLocally.js	
1.35	showError.js	
1.36	analyseNaturalText.js	
1.37	analyseSpreadsheet.js	
1.38	createService.js	
1.39	generateWebhookURL.js	
1.40	index.js	
1.41	newService.js	
1.42	nextScreen.js	
1.43	receiveService.js	
1.44	receiveWebhookURL.js	
1.45	selectDevice.js	
1.46	selectService.js	
1.47	setDeviceFlowDirection.js	
1.48	setServiceCreateMethod.js	
1.49	setServiceName.js	
1.50	setupDeviceQuery.js	
1.51	updateModelPreview.js	
1.52	updateNaturalText.js	
1.53	AuthForm.jsx	
1.54	test.js	
1.55	Button.jsx	
1.56	test.js	
1.57	About.jsx	
1.58	test.js	
1.59	Dashboard.jsx	
1.60	test.js	
1.61	Column.jsx	
1.62	test.js	
1.63	Entries.jsx	
1.64	test.js	
1.65	Row.jsx	
1.66	test.js	
1.67	RowHeader.jsx	
1.68	test.js	
1.69	rowStyle.js	
1.70	Tabs.jsx	
1.71	test.js	

1.72	Pages.jsx
1.73	test.js
1.74	Sidebar.jsx
1.75	test.js
1.76	SidebarItem.jsx
1.77	test.js
1.78	Attribute.jsx
1.79	test.js
1.80	DialogBox.jsx
1.81	test.js
1.82	Model.jsx
1.83	test.js
1.84	Structure.jsx
1.85	test.js
1.86	TopBar.jsx
1.87	test.js
1.88	Frame.jsx
1.89	test.js
1.90	HomePage.jsx
1.91	Logo.jsx
1.92	test.js
1.93	MethodButton.jsx
1.94	test.js
1.95	RoundButton.jsx
1.96	test.js
1.97	ServiceList.jsx
1.98	test.js
1.99	ServiceListItem.jsx
1.100	test.js
1.101	Setup.jsx
1.102	SetupMethod.jsx
1.103	test.js
1.104	SetupName.jsx
1.105	test.js
1.106	SetupNatural.jsx
1.107	test.js
1.108	SetupSpreadsheet.jsx
1.109	test.js
1.110	StyleConstant.js
1.111	TextInput.jsx
1.112	test.js
1.113	AuthFormContainer.js

1.114AboutContainer.js
1.115EntriesContainer.js
1.116PagesContainer.js
1.117SidebarContainer.js
1.118StructureContainer.js
1.119HomePageContainer.js
1.120NameInput.js
1.121ServiceListContainer.js
1.122SetupContainer.js
1.123SetupMethodContainer.js
1.124SetupNameContainer.js
1.125SetupNaturalContainer.js
1.126SetupSpreadsheetContainer.js
1.127index.css
1.128index.js
1.129index.js
1.130test.js
1.131annotateText.js
1.132API.js
1.133Auth.js
1.134capitalizeString.js
1.135createMethods.js
1.136formatSentences.js
1.137normalizr.js
1.138setupScreens.js
1.139natural.js
1.140parse.js
1.141service.js
1.142utils.js
1.143bootstrap.js
1.144connections.js
1.145passport.js
1.146index.js
1.147authentication.js
1.148attribute.js
1.149entry.js
1.150index.js
1.151model.js
1.152service.js
1.153user.js
1.154value.js
1.155index.py

1.156	spacyparse.py
1.157	api.js
1.158	attribute.js
1.159	auth.js
1.160	entry.js
1.161	index.js
1.162	model.js
1.163	service.js
1.164	value.js
1.165	natural_test.js
1.166	parse_test.js
1.167	service_test.js

Chapter 1

Source Code

1.1 index.html

```
1 <!doctype html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1">
6     <link rel="shortcut icon" href="%PUBLIC_URL%/favicon.ico">
7     <!--
8       Notice the use of %PUBLIC_URL% in the tag above.
9       It will be replaced with the URL of the 'public' folder during the build.
10      Only files inside the 'public' folder can be referenced from the HTML.
11
12      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
13      work correctly both with client-side routing and a non-root public URL.
14      Learn how to configure a non-root public URL by running 'npm run build'.
15    -->
16    <title>React App</title>
17  </head>
18  <body>
19    <div id="root"></div>
20    <!--
21      This HTML file is a template.
22      If you open it directly in the browser, you will see an empty page.
23
24      You can add webfonts, meta tags, or analytics to this file.
25      The build step will place the bundled scripts into the <body> tag.
26
27      To begin the development, run 'npm start'.
28      To create a production bundle, use 'npm run build'.
29    -->
30  </body>
31 </html>
```

1.2 actionTypes.js

```
1 export { ANALYSE_NATURAL_TEXT } from './setup/analyseNaturalText.js';
2 export { UPDATE_MODEL_PREVIEW } from './setup/updateModelPreview.js';
3 export { GENERATE_WEBHOOK_URL } from './setup/generateWebhookURL.js';
4 export { NEW_SERVICE } from './setup/newService.js';
5 export { RECEIVE_WEBHOOK_URL } from './setup/receiveWebhookURL.js';
6 export { SELECT_DEVICE } from './setup/selectDevice.js';
7 export { SET_DEVICE_FLOW_DIRECTION } from './setup/setDeviceFlowDirection.js';
8 export { SET_SERVICE_CREATE_METHOD } from './setup/setServiceCreateMethod.js';
9 export { SET_SERVICE_NAME } from './setup/setServiceName.js';
10 export { SETUP_DEVICE_QUERY } from './setup/setupDeviceQuery.js';
11 export { NEXT_SCREEN } from './setup/nextScreen.js';
12 export { UPDATE_NATURAL_TEXT } from './setup/updateNaturalText.js';
13 export { AUTH_USER } from './auth/authUser.js';
14 export { UPDATE_USER } from './auth/updateUser.js';
15 export { AUTH_USER_RESULT } from './auth/authUserResult.js';
16 export { LOGOUT_USER } from './auth/logoutUser.js';
17 export { CHANGE_SIDEBAR_ITEM } from './dashboard/changeSidebarItem.js';
18 export { RECEIVE_SERVICE_LIST } from './auth/receiveServiceList.js';
19 export { SELECT_SERVICE } from './setup/selectService.js';
20 export { RECEIVE_SERVICE } from './setup/receiveService.js';
21 export { CHANGE_SELECTED_MODEL } from './dashboard/changeSelectedModel.js';
22 export { RECEIVE_ENTRY } from './dashboard/receiveEntry.js';
23 export { RECEIVE_MODEL } from './dashboard/receiveModel.js';
24 export { RECEIVE_ATTRIBUTE } from './dashboard/receiveAttribute.js';
25 export { DELETE_ENTRY_LOCALLY } from './dashboard/deleteEntryLocally.js';
26 export { UPDATE_VALUE_LOCALLY } from './dashboard/updateValueLocally.js';
27 export { UPDATE_SERVICE_LOCALLY } from './dashboard/updateServiceLocally.js';
28 export { UPDATE_MODEL_LOCALLY } from './dashboard/updateModelLocally.js';
29 export { UPDATE_ATTRIBUTE_LOCALLY } from './dashboard/updateAttributeLocally.js';
30 export { SELECT_ATTRIBUTE } from './dashboard/selectAttribute.js';
31 export { DELETE_MODEL_LOCALLY } from './dashboard/deleteModelLocally.js';
32 export { DELETE_ATTRIBUTE_LOCALLY } from './dashboard/deleteAttributeLocally.js';
```


1.3 authUser.js

```
1 import { authUserResult } from '../authUserResult';
2 import { authenticateUser } from '../../utils/API';
3 import { saveToken } from '../../utils/Auth';
4 import { showError } from '../other/showError';
5
6 export function authUser(username, password) {
7   return function (dispatch) {
8     authenticateUser(username, password)
9       .then(result => dispatch(authUserResult(result)))
10      .then((result) => {
11        if (result.success) {
12          saveToken(result.token);
13        }
14      })
15      .catch(e => showError(e.message));
16   };
17 }
```

1.4 authUserResult.js

```
1 export const AUTH_USER_RESULT = 'AUTH_USER_RESULT';
2
3 export function authUserResult(result) {
4   return {
5     type: AUTH_USER_RESULT,
6     success: result.success,
7     errors: result.errors,
8     token: result.token,
9   };
10 }
```

1.5 getServiceList.js

```
1 import { receiveServiceList } from './receiveServiceList';
2 import { showError } from '../other/showError';
3 import * as API from '../../utils/API';
4
5 export function getServiceList() {
6   return function (dispatch) {
7     API.getServiceList()
8       .then((result) => {
9         dispatch(receiveServiceList(result));
10      })
11     .catch(e => dispatch(showError(e.message)));
12   };
13 }
```

1.6 index.js

```
1 export { authUser } from './authUser.js';
2 export { updateUser } from './updateUser.js';
3 export { logoutUser } from './logoutUser.js';
```

1.7 logoutUser.js

```
1 import { removeToken } from '../utils/Auth';
2
3 export const LOGOUT_USER = 'LOGOUT_USER';
4
5 export function logoutUser() {
6   removeToken();
7   return {
8     type: LOGOUT_USER,
9   };
10 }
```

1.8 receiveService.js

```
1 export const RECEIVE_SERVICE = 'RECEIVE_SERVICE';
2
3 export default function receiveService(data) {
4   return {
5     type: RECEIVE_SERVICE,
6     data,
7   };
8 }
```

1.9 receiveServiceList.js

```
1 export const RECEIVE_SERVICE_LIST = 'RECEIVE_SERVICE_LIST';
2
3 export function receiveServiceList(data) {
4   return {
5     type: RECEIVE_SERVICE_LIST,
6     services: data.services,
7   };
8 }
```

1.10 updateUser.js

```
1 export const UPDATE_USER = 'UPDATE_USER';
2
3 export function updateUser(username, password) {
4   return {
5     type: UPDATE_USER,
6     username,
7     password,
8   };
9 }
```


1.11 changeDashboardPage.js

```
1 import { push } from 'react-router-redux';
2 import { changeSidebarItem } from './changeSidebarItem';
3
4 export const CHANGE_DASHBOARD_PAGE = 'CHANGE_DASHBOARD_PAGE';
5
6 export function changeDashboardPage(index, item) {
7   return function (dispatch, getState) {
8     dispatch(changeSidebarItem(index));
9     dispatch(push(item.path));
10   };
11 }
```

1.12 changeSelectedModel.js

```
1 export const CHANGE_SELECTED_MODEL = 'CHANGE_SELECTED_MODEL';
2
3 export const changeSelectedModel = id => ({
4   type: CHANGE_SELECTED_MODEL,
5   id,
6 });
```

1.13 changeSidebarItem.js

```
1 export const CHANGE_SIDEBAR_ITEM = 'CHANGE_SIDEBAR_ITEM';
2
3 export const changeSidebarItem = index => ({
4   type: CHANGE_SIDEBAR_ITEM,
5   index,
6 });
```

1.14 createAttribute.js

```
1 import { push } from 'react-router-redux';
2 import { postAttribute } from '../utils/API';
3 import { showError } from '../other/showError';
4 import { receiveAttribute } from './receiveAttribute';
5
6
7 export function createAttribute(model) {
8   return function (dispatch, getState) {
9     postAttribute({
10       model,
11       name: '',
12       type: 'string',
13       required: false,
14       multiple: false,
15     })
16     .then((result) => {
17       if (result.success) {
18         dispatch(receiveAttribute(result.attribute));
19       } else {
20         showError(result.error);
21       }
22     })
23     .catch(e =>
24       showError(e));
25   };
26 }
```

1.15 createEntry.js

```
1 import { push } from 'react-router-redux';
2 import { changeSidebarItem } from './changeSidebarItem';
3 import { postEntry } from '../../utils/API';
4 import { showError } from '../other/showError';
5 import { receiveEntry } from './receiveEntry';
6
7
8 export function createEntry(index, item) {
9   return function (dispatch, getState) {
10     const state = getState().toJS();
11
12     const model = state.dashboard.selectedModel ||
13       state.serviceById[state.user.currentServiceId].Models[0];
14     postEntry(model)
15       .then((result) => {
16         if (result.success) {
17           dispatch(receiveEntry(result.entry));
18         } else {
19           showError(result.error);
20         }
21       })
22       .catch(e =>
23         showError(e));
24   };
25 }
```

1.16 createModel.js

```
1 import { push } from 'react-router-redux';
2 import { postModel } from '../utils/API';
3 import { showError } from '../other/showError';
4 import { receiveModel } from './receiveModel';
5
6
7 export function createModel() {
8   return function (dispatch, getState) {
9     const state = getState().toJS();
10
11     postModel({
12       service: state.user.currentServiceId,
13       name: '',
14     })
15     .then((result) => {
16       if (result.success) {
17         dispatch(receiveModel(result.model));
18       } else {
19         showError(result.error);
20       }
21     })
22     .catch(e =>
23       showError(e));
24   };
25 }
```

1.17 deleteAttribute.js

```
1 import * as API from '../utils/API';
2 import { showError } from '../other/showError';
3 import { deleteAttributeLocally } from './deleteAttributeLocally';
4
5
6 export function deleteAttribute(id) {
7   return function (dispatch) {
8     dispatch(deleteAttributeLocally(id));
9     API.deleteAttribute({
10       id,
11     })
12     .then((result) => {
13       if (!result.success) {
14         showError(result.error);
15       }
16     })
17     .catch(e =>
18       showError(e));
19   };
20 }
```

1.18 deleteAttributeLocally.js

```
1 export const DELETE_ATTRIBUTE_LOCALLY = 'DELETE_ATTRIBUTE_LOCALLY';
2
3 export const deleteAttributeLocally = id => ({
4   type: DELETE_ATTRIBUTE_LOCALLY,
5   id,
6 });
```


1.19 deleteEntry.js

```
1 import { push } from 'react-router-redux';
2 import { changeSidebarItem } from '../changeSidebarItem';
3 import * as API from '../../utils/API';
4 import { showError } from '../other/showError';
5 import { deleteEntryLocally } from '../deleteEntryLocally';
6
7
8 export function deleteEntry(id) {
9   return function (dispatch, getState) {
10     const entry = getState().get('entryById').toJS()[id];
11     console.log('this is ', getState().get('entryById').toJS(), entry, id);
12     API.deleteEntry(id)
13       .then((result) => {
14         if (result.success) {
15           dispatch(deleteEntryLocally(entry));
16         } else {
17           showError(result.error);
18         }
19       })
20       .catch(e =>
21         showError(e));
22   };
23 }
```

1.20 deleteEntryLocally.js

```
1 export const DELETE_ENTRY_LOCALLY = 'DELETE_ENTRY_LOCALLY';
2
3 export const deleteEntryLocally = entry => ({
4   type: DELETE_ENTRY_LOCALLY,
5   entry,
6 });
```

1.21 deleteModel.js

```
1 import * as API from '../utils/API';
2 import { showError } from '../other/showError';
3 import { deleteModelLocally } from './deleteModelLocally';
4
5
6 export function deleteModel(id) {
7   return function (dispatch) {
8     dispatch(deleteModelLocally(id));
9     API.deleteModel({
10       id,
11     })
12     .then((result) => {
13       if (!result.success) {
14         showError(result.error);
15       }
16     })
17     .catch(e =>
18       showError(e));
19   };
20 }
```

1.22 deleteModelLocally.js

```
1 export const DELETE_MODEL_LOCALLY = 'DELETE_MODEL_LOCALLY';
2
3 export const deleteModelLocally = id => ({
4   type: DELETE_MODEL_LOCALLY,
5   id,
6 });
```

1.23 receiveAttribute.js

```
1 export const RECEIVE_ATTRIBUTE = 'RECEIVE_ATTRIBUTE';
2
3 export const receiveAttribute = attribute => ({
4   type: RECEIVE_ATTRIBUTE,
5   attribute,
6 });
```

1.24 receiveEntry.js

```
1 export const RECEIVE_ENTRY = 'RECEIVE_ENTRY';
2
3 export const receiveEntry = entry => ({
4   type: RECEIVE_ENTRY,
5   entry,
6 });
```

1.25 receiveModel.js

```
1 export const RECEIVE_MODEL = 'RECEIVE_MODEL';
2
3 export const receiveModel = model => ({
4   type: RECEIVE_MODEL,
5   model,
6 });
```

1.26 selectAttribute.js

```
1 export const SELECT_ATTRIBUTE = 'SELECT_ATTRIBUTE';
2
3 export const selectAttribute = id => ({
4   type: SELECT_ATTRIBUTE,
5   id,
6 });
```


1.27 updateAttribute.js

```
1 import * as API from '../utils/API';
2 import { showError } from '../other/showError';
3 import { updateAttributeLocally } from './updateAttributeLocally';
4
5 export function updateAttribute(id, changes) {
6   return function (dispatch) {
7     dispatch(updateAttributeLocally(id, changes));
8     API.patchAttribute({ id, ...changes })
9       .then((result) => {
10       if (!result.success) {
11         showError(result.error);
12       }
13     })
14     .catch(e => showError(e));
15   };
16 }
```

1.28 updateAttributeLocally.js

```
1 export const UPDATE_ATTRIBUTE_LOCALLY = 'UPDATE_ATTRIBUTE_LOCALLY';
2
3 export const updateAttributeLocally = (id, changes) => ({
4   type: UPDATE_ATTRIBUTE_LOCALLY,
5   id,
6   changes,
7 });
```

1.29 updateModel.js

```
1 import * as API from '../utils/API';
2 import { showError } from '../other/showError';
3 import { updateModelLocally } from './updateModelLocally';
4
5 export function updateModel(id, name) {
6   return function (dispatch) {
7     dispatch(updateModelLocally(id, name));
8     API.patchModel({ id, name })
9       .then((result) => {
10       if (!result.success) {
11         showError(result.error);
12       }
13     })
14     .catch(e => showError(e));
15   };
16 }
```

1.30 updateModelLocally.js

```
1 export const UPDATE_MODEL_LOCALLY = 'UPDATE_MODEL_LOCALLY';
2
3 export const updateModelLocally = (id, name) => ({
4   type: UPDATE_MODEL_LOCALLY,
5   id,
6   name,
7 });
```

1.31 updateService.js

```
1 import { push } from 'react-router-redux';
2 import { changeSidebarItem } from './changeSidebarItem';
3 import * as API from '../utils/API';
4 import { showError } from '../other/showError';
5 import { receiveEntry } from './receiveEntry';
6
7
8 export function updateService(changes) {
9   return function (dispatch, getStore) {
10     const id = getStore().toJS().user.currentServiceId;
11     API.updateService(id, changes)
12       .then((result) => {
13         if (!result.success) {
14           showError(result.error);
15         }
16       })
17       .catch(e =>
18         showError(e));
19   };
20 }
```

1.32 updateServiceLocally.js

```
1 export const UPDATE_SERVICE_LOCALLY = 'UPDATE_SERVICE_LOCALLY';
2
3 export const updateServiceLocally = changes => ({
4   type: UPDATE_SERVICE_LOCALLY,
5   changes,
6 });
```

1.33 updateValue.js

```
1 import { push } from 'react-router-redux';
2 import { changeSidebarItem } from '../changeSidebarItem';
3 import * as API from '../../utils/API';
4 import { showError } from '../other/showError';
5 import { receiveEntry } from '../receiveEntry';
6
7
8 export function updateValue(entryId, attributeId, value) {
9   return function (dispatch) {
10     API.updateValue(entryId, attributeId, value)
11       .then((result) => {
12         if (!result.success) {
13           showError(result.error);
14         }
15       })
16       .catch(e =>
17         showError(e));
18   };
19 }
```

1.34 updateValueLocally.js

```
1 export const UPDATE_VALUE_LOCALLY = 'UPDATE_VALUE_LOCALLY';
2
3 export const updateValueLocally = (entry, id, value) => ({
4   type: UPDATE_VALUE_LOCALLY,
5   entry,
6   id,
7   value,
8 });
```


1.35 showError.js

```
1 export const SHOW_ERROR = 'SHOW_ERROR';
2
3 export function showError(message) {
4   console.error(`${message}`);
5   return {
6     type: SHOW_ERROR,
7     message,
8   };
9 }
```

1.36 analyseNaturalText.js

```
1 import { updateModelPreview } from './updateModelPreview';
2 import { updateNaturalText } from './updateNaturalText';
3 import { extractModelFromText } from '../utils/API';
4
5 export const ANALYSE_NATURAL_TEXT = 'ANALYSE_NATURAL_TEXT';
6
7
8 export function analyseNaturalText(text) {
9   return function (dispatch) {
10     dispatch(updateNaturalText(text));
11
12     return extractModelFromText(text)
13       .then(result => dispatch(updateModelPreview(result)))
14       .catch(console.log);
15   };
16 }
```

1.37 analyseSpreadsheet.js

```
1 // @flow
2
3 import { updateModelPreview } from './updateModelPreview';
4 import { showError } from '../other/showError';
5 import { postAnalyzeSpreadsheet } from '../../utils/API';
6
7 export function analyseSpreadsheet(file) {
8   return function (dispatch) {
9     console.log(postAnalyzeSpreadsheet);
10    return postAnalyzeSpreadsheet(file)
11      .then(result => dispatch(updateModelPreview(result)))
12      .catch(showError);
13   };
14 }
```

1.38 createService.js

```
1 import { push } from 'react-router-redux';
2 import { postService } from '../utils/API';
3 import { nextScreen } from './nextScreen';
4 import { showError } from '../other/showError';
5 import { receiveService } from './receiveService';
6
7 export const CREATE_SERVICE = 'CREATE_SERVICE';
8
9 export function createService() {
10   return function (dispatch, getState) {
11     const state = getState();
12
13     const setup = state.get('setup');
14     return postService(setup.get('name'), setup.get('modelDefinitionPreview'))
15       .then((result) => {
16         if (result.success) {
17           dispatch(receiveService(result.service));
18           dispatch(push('/service/dashboard'));
19         } else {
20           dispatch(showError(result.error));
21         }
22       })
23       .catch(e =>
24         showError(e));
25   };
26 }
```

1.39 generateWebhookURL.js

```
1 // @flow
2
3 export const GENERATE_WEBHOOK_URL = 'GENERATE_WEBHOOK_URL';
4
5 export function generateWebhookURL() {
6   return {
7     type: GENERATE_WEBHOOK_URL,
8   };
9 }
```

1.40 index.js

```
1 export { analyseNaturalText } from './analyseNaturalText.js';
2 export { updateModelPreview } from './updateModelPreview.js';
3 export { generateWebhookURL } from './generateWebhookURL.js';
4 export { setServiceName } from './setServiceName.js';
5 export { setServiceCreateMethod } from './setServiceCreateMethod.js';
6 export { nextScreen } from './nextScreen.js';
7 export { newService } from './newService.js';
8 export { createService } from './createService.js';
9 export { selectService } from './selectService.js';
```

1.41 newService.js

```
1 import { push } from 'react-router-redux';
2
3 export const NEW_SERVICE = 'NEW_SERVICE';
4
5 export function newService() {
6   return (dispatch) => {
7     dispatch(push('/service/setup'));
8   };
9 }
```

1.42 nextScreen.js

```
1 // @flow
2
3 export const NEXT_SCREEN = 'NEXT_SCREEN';
4
5 export function nextScreen() {
6   return {
7     type: NEXT_SCREEN,
8   };
9 }
```


1.43 receiveService.js

```
1 // @flow
2
3 export const RECEIVE_SERVICE = 'RECEIVE_SERVICE';
4
5 export function receiveService(service) {
6   return {
7     type: RECEIVE_SERVICE,
8     service,
9   };
10 }
```

1.44 receiveWebhookURL.js

```
1 // @flow
2
3 export const RECEIVE_WEBHOOK_URL = 'RECEIVE_WEBHOOK_URL';
4
5 export function receiveWebhookURL(url) {
6   return {
7     type: RECEIVE_WEBHOOK_URL,
8     url,
9   };
10 }
```

1.45 selectDevice.js

```
1 // @flow
2
3 export const SELECT_DEVICE = 'SELECT_DEVICE';
4
5 export function selectDevice(device: number) {
6   return {
7     type: SELECT_DEVICE,
8     device,
9   };
10 }
```

1.46 selectService.js

```
1 import { push } from 'react-router-redux';
2 export const SELECT_SERVICE = 'SELECT_SERVICE';
3
4
5 export function selectService(id) {
6   return (dispatch) => {
7     dispatch({
8       type: SELECT_SERVICE,
9       id,
10    });
11    dispatch(push('/service/dashboard'));
12  };
13 }
```

1.47 setDeviceFlowDirection.js

```
1 // @flow
2
3 export const SET_DEVICE_FLOW_DIRECTION = 'SET_DEVICE_FLOW_DIRECTION';
4
5 type deviceFlowDirection = 'query' | 'webhook';
6
7 export function setDeviceFlowDirection(direction: deviceFlowDirection) {
8   return {
9     type: SET_DEVICE_FLOW_DIRECTION,
10    direction,
11  };
12 }
```

1.48 setServiceCreateMethod.js

```
1 // @flow
2
3 export const SET_SERVICE_CREATE_METHOD = 'SET_SERVICE_CREATE_METHOD';
4
5 type methodString = 'strach' | 'spreadsheet' | 'device';
6
7 export function setServiceCreateMethod(method: methodString) {
8   return {
9     type: SET_SERVICE_CREATE_METHOD,
10    method,
11  };
12 }
```

1.49 setServiceName.js

```
1 // @flow
2
3 export const SET_SERVICE_NAME = 'SET_SERVICE_NAME';
4
5 export function setServiceName(name) {
6   return {
7     type: SET_SERVICE_NAME,
8     name,
9   };
10 }
```

1.50 setupDeviceQuery.js

```
1 // @flow
2
3 export const SETUP_DEVICE_QUERY = 'SETUP_DEVICE_QUERY';
4
5 export function setupDeviceQuery(
6   url: string,
7   method: string,
8   attributes: [{ string: string }],
9   interval: number,
10 ) {
11   return {
12     type: SETUP_DEVICE_QUERY,
13     url,
14     method,
15     attributes,
16     interval,
17   };
18 }
```


1.51 updateModelPreview.js

```
1
2 export const UPDATE_MODEL_PREVIEW = 'UPDATE_MODEL_PREVIEW';
3
4 export function updateModelPreview(preview) {
5   return {
6     type: UPDATE_MODEL_PREVIEW,
7     preview,
8   };
9 }
```

1.52 updateNaturalText.js

```
1
2 export const UPDATE_NATURAL_TEXT = 'UPDATE_NATURAL_TEXT';
3
4 export function updateNaturalText(text) {
5   return {
6     type: UPDATE_NATURAL_TEXT,
7     text,
8   };
9 }
```

1.53 AuthForm.jsx

```
1  import React from 'react';
2  import PropTypes from 'prop-types';
3  import TextInput from './TextInput';
4  import Button from './Button';
5  import { Color } from './StyleConstant';
6
7  const style = {
8    width: 140,
9    field: {
10     marginBottom: 5,
11   },
12   error: {
13     margin: 4,
14     fontSize: 14,
15     color: Color.red,
16   }
17 }
18
19 const AuthForm = ({
20   onSubmit,
21   onChange,
22   errors = {},
23   username,
24   password,
25 }) => (
26   <div>
27     <h1>Login</h1>
28     <form action="/" onSubmit={e => {
29       e.preventDefault();
30       onSubmit({username, password});
31     }} method="post">
32       <div style={style.field}>
33         <TextInput
34           name="username"
35           placeholder="Username"
36           onChange={username => onChange({username})}
37           text={username}
38         />
39         <p style={style.error}>{errors.username}</p>
40       </div>
41       <div style={style.field}>
42         <TextInput
43           name="password"
44           placeholder="Password"
45           onChange={password => onChange({password})}
46           text={password}
47         />
48         <p style={style.error}>{errors.password}</p>
49       </div>
50     </div>
```

```
51         <Button type="submit" text="Log In"/>
52     </div>
53 </form>
54 </div>
55 )
56
57 AuthForm.propTypes = {
58     onSubmit: PropTypes.func.isRequired,
59     onChange: PropTypes.func.isRequired,
60     errors: PropTypes.array,
61     username: PropTypes.string.isRequired,
62     password: PropTypes.string.isRequired
63 };
64
65 export default AuthForm;
```

1.54 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import AuthForm from './AuthForm';
8
9 describe('<AuthForm />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<AuthForm
12       errors={}
13       username="username"
14       password="password"
15     />);
16   });
17 });
```

1.55 Button.jsx

```
1  import React from 'react';
2  import PropTypes from 'prop-types';
3  import Radium from 'radium';
4  import { Color, Dimensions } from './StyleConstant';
5
6  const activeStyle = {
7    backgroundColor: Color.greenDark,
8    border: 'none',
9    outline: 'none',
10 };
11
12 const style = {
13   base: {
14     backgroundColor: Color.green,
15     minWidth: Dimensions.fieldWidth,
16     height: Dimensions.fieldHeight,
17     border: 'none',
18     borderRadius: Dimensions.borderRadius,
19     cursor: 'pointer',
20     transition: `${Dimensions.transitionTime.normal} background-color`,
21     fontSize: Dimensions.fontSize.normal,
22     color: Color.whiteText,
23     ':hover': {
24       backgroundColor: Color.greenLight,
25     },
26     ':active': activeStyle,
27     ':focus': activeStyle,
28   },
29   isDisabled: {
30     pointerEvents: 'none',
31     backgroundColor: Color.grey,
32   },
33 };
34
35 const Button = ({ text, onClick, isDisabled, type }) => (
36   <button type={type} onClick={onClick} style={[style.base, isDisabled ? style.isDisabled : {}]}>
37     {text}
38   </button>
39 );
40
41 Button.propTypes = {
42   text: PropTypes.string,
43   onClick: PropTypes.func,
44   isDisabled: PropTypes.bool,
45   type: PropTypes.string,
46 };
47
48
49 export default Radium(Button);
```

1.56 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import Button from './Button';
8
9 describe('<Button />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<Button
12       text="Next"
13       isDisabled={false}
14     />);
15   });
16 });
```

1.57 About.jsx

```
1 import React from 'react';
2 import PropTypes from 'prop-types';
3 import TopBar from '../TopBar';
4 import { Color } from '../StyleConstant';
5 import TextInput from '../TextInput';
6 import Button from '../Button';
7
8 const style = {
9   base: {
10     height: '100vh',
11     overflowY: 'auto',
12     padding: 30,
13   },
14   h3: {
15     padding: 0,
16     margin: 0,
17   },
18   label: {
19     marginTop: 10,
20   },
21   field: {
22     marginBottom: 10,
23     marginTop: 4,
24   },
25 };
26
27 const metaExample = {
28   name: 'Pets',
29   url: 'pets',
30   author: 'Martin Hartt',
31   isPublic: false,
32 };
33
34 const About = ({ name, meta = metaExample, onChange = () => {} }) => <div>
35   <TopBar name={name} />
36   <div style={style.base}>
37     <h3 style={style.h3}>About</h3>
38     {
39       Object.keys(meta).map(key =>
40         (typeof (meta[key].value) === 'boolean') ?
41         <div>
42           <input id={key} type="checkbox" checked={meta[key].value === true} onChange={e => onChange({ [key]: !!e.target.checked })} />
43           <label htmlFor={key} style={style.label}>{key}</label>
44         </div>
45         :
46         <div>
47           <label style={style.label} htmlFor={key}>{meta[key].label}</label>
48           <div style={style.field}>
49             <TextInput id={key} text={meta[key].value} onChange={value => onChange({ [key]: value })} />
50           </div>
```



```
51         </div>,
52
53     )
54 }
55
56 </div>
57 </div>;
58
59 About.propTypes = {
60   name: PropTypes.string,
61   meta: PropTypes.shape({
62     name: PropTypes.string,
63     url: PropTypes.string,
64     author: PropTypes.string,
65     public: PropTypes.bool,
66   }),
67   onChange: PropTypes.func,
68 };
69
70 export default About;
```

1.58 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import About from './About';
8
9 describe('<About />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<About
12       name="Example"
13       meta={{
14         name: 'Example',
15         url: 'example',
16         author: 'Martin Hartt',
17         public: true,
18       }}
19     />);
20   });
21 });
```

1.59 Dashboard.jsx

```
1 import React from 'react';
2 import PropTypes from 'prop-types';
3 import Logo from '../Logo';
4 import SidebarContainer from '../../containers/dashboard/SidebarContainer';
5 import { Color, lightBorder } from '../StyleConstant';
6
7
8 const style = {
9   base: {
10     display: 'flex',
11   },
12   sidebar: {
13     width: 230,
14     borderRight: lightBorder,
15     height: '100vh',
16   },
17   main: {
18     flex: 1,
19   },
20   logo: {
21     textAlign: 'center',
22     padding: '20px 0',
23     borderBottom: lightBorder,
24   },
25 };
26
27 const Dashboard = ({ children }) => <div style={style.base}>
28   <div style={style.sidebar}>
29     <div style={style.logo}>
30       <Logo />
31     </div>
32     <SidebarContainer />
33   </div>
34   <div style={style.main}>
35     {children}
36   </div>
37 </div>;
38
39 Dashboard.propTypes = {
40   children: PropTypes.node,
41 };
42
43 export default Dashboard;
```

1.60 test.js

```
1
2
3 import React from 'react';
4 import { expect } from 'chai';
5 import { shallow } from 'enzyme';
6 import sinon from 'sinon';
7
8 import Dashboard from './Dashboard';
9
10 describe('<Dashboard />', () => {
11   it('renders the component correctly', () => {
12     const wrapper = shallow(<Dashboard>
13       <p>Hello</p>
14     </Dashboard>);
15   });
16 });
```

1.61 Column.jsx

```
1 import React from 'react';
2 import PropTypes from 'prop-types';
3 import Radium from 'radium';
4
5 const style = {
6   base: {
7     minWidth: 70,
8     // height: '80%',
9     marginLeft: 0,
10    textAlign: 'center',
11    marginRight: 5,
12    width: 190,
13  },
14  item: {
15    borderRadius: 3,
16    cursor: 'pointer',
17    ':hover': {
18      backgroundColor: '#EEE',
19    },
20    border: 'none',
21    height: 45,
22    fontSize: 18,
23    ':focus': {
24      outline: 0,
25      border: 0,
26    },
27  },
28  first: {
29    width: 80,
30  },
31 };
32
33 const Column = ({ value, isItem, onChange, first = false }) =>
34   isItem ?
35     <input style={[style.base, style.item, first && style.first]} value={value} onChange={onChange} /> :
36     <div style={[style.base, first && style.first]}>{value}</div>;
37
38
39 Column.propTypes = {
40   value: PropTypes.string.isRequired,
41   isItem: PropTypes.bool.isRequired,
42   onChange: PropTypes.func,
43 };
44
45 export default Radium(Column);
```

1.62 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import Column from './Column';
8
9 describe('<Column />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<Column
12       value="Hello"
13       isItem
14     />);
15   });
16 });
```

1.63 Entries.jsx

```
1  import React from 'react';
2  import PropTypes from 'prop-types';
3  import TopBar from '../TopBar';
4  import Tabs from '../Tabs';
5  import RowHeader from '../RowHeader';
6  import Column from './Column';
7  import Row from './Row';
8  import capitalizeString from '../../utils/capitalizeString';
9  import { Color } from '../../StyleConstant';
10
11  const style = {
12    base: {
13      backgroundColor: Color.lighterGrey,
14      overflowX: 'auto',
15    },
16    main: {
17      height: 'calc(100vh - 77px)',
18      overflowY: 'auto',
19    },
20  };
21
22
23  function decode(string, type) {
24    switch (type) {
25      case 'integer':
26        return parseInt(string, 10);
27      case 'float':
28        return parseFloat(string);
29      default:
30        return string;
31    }
32  }
33
34  const Entries = ({ name, entries = [], attributes = [], headers = [], onSelected, onDelete, onCreate, onUpdate }) =>
35    <div style={style.base}>
36      <TopBar name={name} onNew={() => onCreate()} />
37      <div style={style.main}>
38        <Tabs headers={headers} onSelected={onSelected} />
39        <RowHeader>
40          <Column key="headerid" value="ID" first />
41          {attributes.map(attr => <Column value={capitalizeString(attr.name)} key={attr.id} />)}
42        </RowHeader>
43        {entries.map(entry =>
44          <Row key={entry.id} onDelete={() => onDelete(entry.realId)}>
45            <Column key={`_${entry.id}.id`} value={entry.id} first />
46
47            {attributes.map(attr =>
48              <Column
49                key={`_${entry.realId}.${attr.id}`}
50                value={entry[attr.name] ? decode(entry[attr.name].value, attr.type) : ''}
```

```
51         isItem
52         onChange={e => onUpdate(entry.realId, attr.id, e.target.value, entry[attr.name].id)}
53         />)}
54     </Row>,
55     )}
56 </div>
57 </div>;
58
59 Entries.propTypes = {
60   name: PropTypes.string.isRequired,
61   entries: PropTypes.array.isRequired,
62   attributes: PropTypes.array.isRequired,
63   headers: PropTypes.array.isRequired,
64   onSelect: PropTypes.func,
65   onDelete: PropTypes.func,
66   onCreate: PropTypes.func,
67   onUpdate: PropTypes.func,
68 };
69
70 export default Entries;
```


1.64 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import Entries from './Entries';
8
9 describe('<Entries />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<Entries
12       name="Example"
13       entries=[]
14       attributes={}
15       headers=[]
16       onSelect={false}
17     />);
18   });
19 });
```

1.65 Row.jsx

```
1  import React from 'react';
2  import PropTypes from 'prop-types';
3  import rowStyle from './rowStyle';
4  import RoundButton from '../RoundButton';
5  import { Color } from '../StyleConstant';
6
7
8  const style = {
9    base: rowStyle,
10 };
11
12 const Row = ({ children, onDelete }) => <div style={style.base}>
13   {children}
14   <RoundButton onClick={onDelete} text="remove" color={Color.red} />
15
16 </div>;
17
18 Row.propTypes = {
19   children: PropTypes.node,
20   onDelete: PropTypes.func,
21 };
22
23 export default Row;
```

1.66 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import Row from './Row';
8
9 describe('<Row />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<Row>
12       <p>Hello</p>
13     </Row>);
14   });
15 });
```

1.67 RowHeader.jsx

```
1 import React from 'react';
2 import PropTypes from 'prop-types';
3 import Radium from 'radium';
4 import rowStyle from './rowStyle';
5 import { Color } from '../../StyleConstant';
6
7 const style = {
8   base: [
9     rowStyle,
10    {
11      backgroundColor: Color.lighterGrey,
12    },
13  ],
14 };
15
16 const RowHeader = ({ children }) => <div style={style.base}>{children}</div>;
17
18 RowHeader.propTypes = {
19   children: PropTypes.node,
20 };
21
22 export default Radium(RowHeader);
```

1.68 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import RowHeader from './RowHeader';
8
9 describe('<RowHeader />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<RowHeader>
12       <p>Test</p>
13     </RowHeader>);
14   });
15 });
```

1.69 rowStyle.js

```
1  import { Color } from '../StyleConstant';
2
3  const rowStyle = {
4    display: 'flex',
5    flexDirection: 'row',
6    height: 57,
7    alignItems: 'center',
8    backgroundColor: Color.whiteText,
9    borderBottom: '2px solid ${Color.lightGrey}'
10 };
11
12 export default rowStyle;
```

1.70 Tabs.jsx

```
1 import React from 'react';
2 import PropTypes from 'prop-types';
3 import Radium from 'radium';
4 import { Color } from '../StyleConstant';
5
6 const style = {
7   base: {
8     display: 'flex',
9     flexDirection: 'row',
10    backgroundColor: Color.whiteText,
11    borderBottom: '2px solid ${Color.lightGrey}',
12  },
13  tab: {
14    display: 'inline-block',
15    cursor: 'pointer',
16    minWidth: 100,
17    height: 56,
18    display: 'flex',
19    justifyContent: 'center',
20    alignItems: 'center',
21    borderTop: '3px solid transparent',
22  },
23  selected: {
24    backgroundColor: Color.lighterGrey,
25    borderTop: '3px solid ${Color.green}',
26  },
27 };
28
29 const Tabs = ({ headers, onSelected }) => <div style={style.base}>
30   {headers.map(header =>
31     <div
32       key={header.text}
33       style={[style.tab, header.selected && style.selected]}
34       onClick={() => onSelected(header.id)}
35     >
36       {header.text}
37     </div>)}
38 </div>;
39
40 Tabs.propTypes = {
41   headers: PropTypes.array,
42 };
43
44 export default Radium(Tabs);
```

1.71 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import Tabs from './Tabs';
8
9 describe('<Tabs />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<Tabs
12       headers=[]
13     />);
14   });
15 });
```


1.72 Pages.jsx

```
1 import React from 'react';
2 import PropTypes from 'prop-types';
3 import TopBar from '../TopBar';
4 import { Color } from '../../../StyleConstant';
5 import TextInput from '../../../TextInput';
6 import Button from '../../../Button';
7
8 const style = {
9   base: {
10     backgroundColor: Color.lighterGrey,
11     height: '100vh',
12     overflowY: 'auto',
13   },
14   page: {
15     backgroundColor: Color.whiteText,
16     margin: 20,
17     padding: 15,
18     borderRadius: 5,
19     border: '2px solid ${Color.lightGrey}',
20   },
21   title: {
22     margin: 0,
23     padding: 0,
24     fontWeight: 600,
25   },
26   method: {
27     color: Color.green,
28   },
29   description: {
30     padding: 0,
31     marginBottom: 0,
32   },
33   label: {
34   },
35   field: {
36     marginBottom: 10,
37     marginTop: 4,
38   },
39 };
40
41
42 const pagesExamples = [
43   {
44     method: 'GET',
45     path: '/owners',
46     operation: 'find',
47     model: 'owners',
48   },
49   {
50     method: 'GET',
```

```

51     path: '/pets/{id}',
52     operation: 'findById',
53     model: 'pets',
54   },
55 ];
56
57 function bind(model, action, onChange) {
58   const name = `${model.name}`;
59   const prop = `${action.prop}`;
60   console.log('bind', name, prop);
61   return console.log.bind(console, name, prop); // onChange(name, { [prop]: !!e.target.checked });
62 }
63
64 const Pages = ({ name, models = [], actions = [], onChange, urlPrefix }) => <div style={style.base}>
65   <TopBar name={name} />
66   {models.map((model, modelIndex) => <div style={style.page}>
67     <h3 style={style.title}>{model.name}</h3>
68     {actions.map(action => (
69       <div>
70         <input
71           id={action}
72           type="checkbox"
73           checked={model[action.prop].value === true}
74           onChange={bind(model, action)}
75         />
76         <label
77           htmlFor={action}
78           style={style.label}
79         >
80           {action.label} ({action.method} {urlPrefix}{model.name})
81         </label>
82       </div>
83     ))}
84   </div>)}
85 </div>;
86
87 Pages.propTypes = {
88   name: PropTypes.string,
89   models: PropTypes.array,
90   actions: PropTypes.array,
91   onChange: PropTypes.func,
92   urlPrefix: PropTypes.string,
93 };
94
95 export default Pages;

```

1.73 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import Pages from './Pages';
8
9 describe('<Pages />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<Pages
12       name="Example"
13       models={}
14       actions={}
15       urlPrefix="example"
16     />);
17   });
18 });
```

1.74 Sidebar.jsx

```
1 import React from 'react';
2 import PropTypes from 'prop-types';
3 import SidebarItem from './SidebarItem';
4
5 const itemsExample = [
6   { name: 'Structure', path: '/service/X/structure', selected: true },
7   { name: 'Entries', path: '/service/X/entries' },
8   { name: 'Pages', path: '/service/X/pages' },
9   { name: 'About', path: '/service/X/about' },
10  { name: 'Publish', path: '/service/X/publish' },
11 ];
12
13 const Sidebar = ({ items = itemsExample, onSelect }) => <div>
14   {items.map(
15     (item, i) => <SidebarItem item={item} key={item.name} onClick={() => onSelect(i, items[i])} />,
16   )}
17 </div>;
18
19 Sidebar.propTypes = {
20   items: PropTypes.arrayOf(PropTypes.shape({
21     name: PropTypes.string,
22     path: PropTypes.string,
23     selected: PropTypes.bool,
24   })),
25   onSelect: PropTypes.func,
26 };
27
28 export default Sidebar;
```

1.75 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import Sidebar from './Sidebar';
8
9 describe('<Sidebar />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<Sidebar
12       items={}[]>
13     />);
14   });
15 });
```

1.76 SidebarItem.jsx

```
1 import React from 'react';
2 import PropTypes from 'prop-types';
3 import Radium from 'radium';
4 import { Link } from 'react-router';
5 import { Color } from '../StyleConstant';
6
7 const style = {
8   base: {
9     height: 57,
10    display: 'flex',
11    alignItems: 'center',
12    paddingLeft: 20,
13    textDecoration: 'none',
14    color: Color.black,
15    transition: '0.3s all',
16    ':hover': {
17      background: Color.lighterGrey,
18    },
19  },
20  selected: {
21    background: Color.lightGrey,
22    ':hover': {
23      background: Color.lightGrey,
24    },
25  },
26 };
27
28 const SidebarItem = ({ item, onClick }) =>
29   <div
30     style={{ textDecoration: 'none', cursor: 'pointer' }}
31     onClick={onClick}
32   >
33     <div style={[style.base, item.selected && style.selected]}>
34       {item.name}
35     </div>
36   </div>;
37
38 SidebarItem.propTypes = {
39   item: PropTypes.shape({
40     name: PropTypes.string,
41     path: PropTypes.string,
42     selected: PropTypes.bool,
43   }),
44   onClick: PropTypes.func,
45 };
46
47 export default Radium(SidebarItem);
```

1.77 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import SidebarItem from './SidebarItem';
8
9 describe('<SidebarItem />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<SidebarItem
12       item={{
13         name: 'Hello',
14         path: 'hello',
15         selected: true,
16       }}
17     />);
18   });
19 });
```

1.78 Attribute.jsx

```
1 import React from 'react';
2 import PropTypes from 'prop-types';
3 import Radium from 'radium';
4 import capitalizeString from '../.../utils/capitalizeString';
5 import { Color } from '../.../StyleConstant';
6
7 const style = {
8   base: {
9     backgroundColor: Color.lighterGrey,
10    marginBottom: 4,
11    borderRadius: 3,
12    padding: '5px 9px',
13    cursor: 'pointer',
14    minHeight: 25,
15    transition: '0.3s all',
16    ':hover': {
17      backgroundColor: Color.lightGrey,
18    },
19  },
20  noInteraction: {
21    cursor: 'default',
22  },
23 };
24
25 const prettify = string => string && string.replace(/_/g, ' ');
26
27 function formatAttribute(attribute) {
28   console.log(attribute);
29   const leftPar = attribute.multiple ? '[' : '';
30   const rightPar = attribute.multiple ? ']' : '';
31   return `${prettify(attribute.name)} (${leftPar}${attribute.type}${rightPar})`;
32 }
33
34 const Attribute = ({ attribute, onClick, enableInteractions }) => <div onClick={onClick} style={[style.base, !enableInteractions && style.
   noInteraction]}>
35   {formatAttribute(attribute)}
36 </div>;
37
38 Attribute.propTypes = {
39   attribute: PropTypes.shape({
40     name: PropTypes.string,
41     multiple: PropTypes.bool,
42   }),
43   enableInteractions: PropTypes.bool,
44 };
45
46 export default Radium(Attribute);
```


1.79 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import Attribute from './Attribute';
8
9 describe('<Attribute />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<Attribute
12       attribute={{
13         name: 'Attribute',
14         multiple: true,
15       }}
16       enableInteractions
17     />);
18   });
19 });
```

1.80 DialogBox.jsx

```
1 import React from 'react';
2 import PropTypes from 'prop-types';
3 import { Color, lightBorder } from '../StyleConstant';
4 import TextInput from '../TextInput';
5 import capitalizeString from '../utils/capitalizeString';
6
7 const style = {
8   base: {
9     width: 400,
10    height: 400,
11    position: 'absolute',
12    backgroundColor: Color.whiteText,
13    top: '50%',
14    left: '50%',
15    transform: 'translate(-50%, -50%)',
16    borderRadius: 3,
17    padding: 15,
18    zIndex: 50,
19  },
20  close: {
21    width: 35,
22    height: 35,
23    backgroundImage: 'url("/img/cross.png")',
24    backgroundColor: 'cover',
25    opacity: 0.3,
26    position: 'absolute',
27    right: 15,
28    top: 15,
29    cursor: 'pointer',
30  },
31  title: {
32    fontWeight: 400,
33    margin: 0,
34    marginBottom: 20,
35  },
36  label: {
37    display: 'block',
38    marginTop: 10,
39    marginBottom: 5,
40  },
41  cover: {
42    position: 'fixed',
43    width: '100%',
44    height: '100%',
45    top: 0,
46    left: 0,
47    background: 'rgba(0, 0, 0, 0.6)',
48    zIndex: 5,
49  },
50  delete: {
```

```

51     width: 200,
52     height: 42,
53     backgroundColor: Color.red,
54     borderRadius: 3,
55     color: 'white',
56     textAlign: 'center',
57     lineHeight: '$ {42}px',
58     marginTop: 30,
59     cursor: 'pointer',
60 },
61 };
62
63 function field(attr, object, onChange) {
64     switch (attr.type) {
65         case 'string': {
66             return (
67                 <div key={attr.value}>
68                     <label style={style.label} htmlFor={attr.value}>{attr.label}</label>
69                     <TextInput id={attr.value} text={object[attr.value]} onChange={val => onChange({ [attr.value]: val })} />
70                 </div>
71             );
72         }
73         case 'integer': {
74             return (
75                 <div key={attr.value}>
76                     <label style={style.label} htmlFor={attr.value}>{attr.label}</label>
77                     <TextInput id={attr.value} type="number" text={object[attr.value]} onChange={val => onChange({ [attr.value]: val })} />
78                 </div>
79             );
80         }
81         case 'enum': {
82             return (
83                 <div key={attr.value}>
84                     <label style={style.label} htmlFor={attr.value}>{attr.label}</label>
85                     <select value={object[attr.value]} onChange={e => onChange({ [attr.value]: e.target.value })}>
86                         {attr.options.map(option =>
87                             <option value={option}>{capitalizeString(option)}</option>,
88                         )}
89                     </select>
90                 </div>
91             );
92         }
93         case 'boolean': {
94             return (
95                 <div key={attr.value}>
96                     <label style={style.label} htmlFor={attr.value}><input type="checkbox" checked={object[attr.value]} onChange={e => onChange({ [
97                         attr.value: e.target.checked })} />{attr.label}</label>
98                 </div>
99             );
100         }
101     }
102 }

```

```
103 const DialogBox = ({ name, object, attributes, onChange, onClose, onDelete }) =>
104   <div>
105     <div style={style.cover} onClick={onClose} />
106     <div style={style.base}>
107       <div style={style.close} onClick={onClose} />
108       <h3 style={style.title}>{name && capitalizeString(name)}</h3>
109       {attributes.map(attr => field(attr, object, onChange))}
110       <div onClick={onDelete} style={style.delete}>Delete</div>
111     </div>
112   </div>;
113
114 DialogBox.propTypes = {
115   name: PropTypes.string,
116   object: PropTypes.object,
117   attributes: PropTypes.array,
118   onChange: PropTypes.func,
119   onClose: PropTypes.func,
120   onDelete: PropTypes.func,
121 };
122
123 export default DialogBox;
```

1.81 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import DialogBox from './DialogBox';
8
9 describe('<DialogBox />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<DialogBox
12       name="Example"
13       object={}
14       attributes={}
15     />);
16   });
17 });
```

1.82 Model.jsx

```
1 import React from 'react';
2 import PropTypes from 'prop-types';
3 import Radium from 'radium';
4 import Attribute from './Attribute';
5 import { Color } from '../../StyleConstant';
6 import capitalizeString from '../../utils/capitalizeString';
7 import RoundButton from '../../RoundButton';
8
9 const style = {
10   base: {
11     backgroundColor: Color.white,
12     background: '#FFFFFF',
13     border: '2px solid ${Color.grey}',
14     borderRadius: 3,
15     marginBottom: 10,
16     width: 250,
17     padding: 5,
18     position: 'relative',
19     zIndex: 0,
20   },
21   title: {
22     margin: 0,
23     padding: '5px 0px',
24     textAlign: 'center',
25     borderRadius: 3,
26     ':hover': {
27       backgroundColor: '#EEE',
28     },
29     border: 'none',
30     ':focus': {
31       outline: 0,
32       border: 0,
33     },
34     fontSize: 20,
35     width: '100%',
36   },
37   close: {
38     position: 'absolute',
39     top: 8,
40     right: 6,
41   },
42   attributes: {
43     marginTop: 10,
44   },
45   newAttribute: {
46     textAlign: 'center',
47     backgroundColor: '#EEE',
48     margin: '6px 0',
49     borderRadius: 3,
50     padding: '5px 9px',
```

```

51     cursor: 'pointer',
52     color: 'black',
53     transition: '0.5s all',
54     fontSize: 18,
55     ':hover': {
56         backgroundColor: '#DDD',
57     },
58 },
59 };
60
61 const Model = ({ model, onClickAttribute, onDelete, onChange, onAttributeCreate, enableInteractions = true }) => <div style={style.base}>
62     <input disabled={!enableInteractions} style={[[style.title]] value={capitalizeString(model.name)} onChange={e => onChange(model.id, e.target
63         .value)} />
64     <div style={style.close}>
65         {enableInteractions && <RoundButton text="remove" onClick={() => onDelete(model.id)} color={Color.red} small />}
66     </div>
67     <div style={style.attributes}>
68         {model.attributes && model.attributes.map(attribute =>
69             <Attribute
70                 key={`attr-${attribute.name}-${attribute.id}`}
71                 onClick={() => enableInteractions && onClickAttribute(attribute.id)}
72                 attribute={attribute}
73                 enableInteractions={enableInteractions}
74             />)}
75         {enableInteractions && <div key="newAttribute" style={style.newAttribute} onClick={() => onAttributeCreate(model.id)}></div>}
76     </div>
77
78 Model.propTypes = {
79     model: PropTypes.shape({
80         name: PropTypes.string,
81         id: PropTypes.number,
82         attributes: PropTypes.array,
83     }).isRequired,
84     onClickAttribute: PropTypes.func,
85     onDelete: PropTypes.func,
86     onChange: PropTypes.func,
87     onAttributeCreate: PropTypes.func,
88     enableInteractions: PropTypes.bool,
89 };
90
91 export default Radium(Model);

```

1.83 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import Model from './Model';
8
9 describe('<Model />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<Model
12       model={{
13         name: 'Test',
14         id: 3,
15         attributes: [],
16       }}
17     />);
18   });
19 });
```


1.84 Structure.jsx

```
1 import React from 'react';
2 import PropTypes from 'prop-types';
3 import Radium from 'radium';
4 import Model from './Model';
5 import TopBar from '../TopBar';
6 import DialogBox from './DialogBox';
7 import { lightBorder, Color } from '../../StyleConstant';
8
9 const style = {
10   main: {
11     backgroundColor: Color.lighterGrey,
12     height: 'calc(100vh - 97px)',
13     padding: 10,
14     overflowY: 'auto',
15   },
16   model: {
17
18   },
19   newModel: {
20     textAlign: 'center',
21     backgroundColor: 'white',
22     border: '2px solid rgba(198, 198, 198, 0.34)',
23     borderRadius: 3,
24     width: 250,
25     padding: 5,
26     paddingBottom: 10,
27     fontSize: 27,
28     cursor: 'pointer',
29     color: 'black',
30     transition: '0.5s all',
31     ':hover': {
32       border: '2px solid rgba(198, 198, 198, 0.8)',
33     },
34   },
35 };
36
37 const attributes = [
38   {
39     value: 'name',
40     label: 'Name',
41     type: 'string',
42   },
43   {
44     value: 'type',
45     label: 'Type',
46     type: 'enum',
47     options: ['string', 'integer', 'float'],
48   },
49   {
50     value: 'multiple',
```

```

51     label: 'Multiple',
52     type: 'boolean',
53   },
54   {
55     value: 'required',
56     label: 'Required',
57     type: 'boolean',
58   },
59 ];
60
61 const Structure = ({
62   name,
63   models = [],
64   selectedAttribute,
65   onSelectAttribute,
66   onModelCreate,
67   onModelDelete,
68   onModelChange,
69   onAttributeCreate,
70   onAttributeDelete,
71   onAttributeChange,
72 }) => <div style={style.base}>
73   <TopBar name={name} onNew={() => onModelCreate()} />
74   {selectedAttribute && <DialogBox
75     name={selectedAttribute.name}
76     object={selectedAttribute}
77     attributes={attributes}
78     onChange={changes => onAttributeChange(selectedAttribute.id, changes)}
79     onDelete={() => onAttributeDelete(selectedAttribute.id)}
80     onClose={() => onSelectAttribute(undefined)}
81   />}
82   <div style={style.main}>
83     {models.map(model =>
84       <Model key={`model-${model.id}`} onChange={onModelChange} onDelete={onModelDelete} onAttributeCreate={onAttributeCreate}
85         onClickAttribute={onSelectAttribute} model={model} />,
86     )}
87   </div>
88 </div>;
89
90 Structure.propTypes = {
91   name: PropTypes.string,
92   models: PropTypes.array,
93   selectedAttribute: PropTypes.object,
94   onSelectAttribute: PropTypes.func,
95   onModelCreate: PropTypes.func,
96   onModelDelete: PropTypes.func,
97   onModelChange: PropTypes.func,
98   onAttributeCreate: PropTypes.func,
99   onAttributeDelete: PropTypes.func,
100   onAttributeChange: PropTypes.func,
101 };
102 export default Radium(Structure);

```

1.85 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import Structure from './Structure';
8
9 describe('<Structure />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<Structure
12       name="Example"
13       models={} />);
14   });
15 });
16 });
```

1.86 TopBar.jsx

```
1 import React from 'react';
2 import PropTypes from 'prop-types';
3 import { lightBorder, Color } from '../StyleConstant';
4 import RoundButton from '../RoundButton';
5
6 const style = {
7   base: {
8     height: 74.5,
9     borderBottom: lightBorder,
10    display: 'flex',
11    alignItems: 'center',
12    justifyContent: 'space-between',
13    padding: '0 20px',
14    backgroundColor: Color.whiteText,
15  },
16  h2: {
17    margin: 0,
18    padding: 0,
19    fontWeight: 500,
20    fontSize: 24,
21  },
22 };
23
24 const TopBar = ({ name, onNew }) =>
25   <div style={style.base}>
26     <h2 style={style.h2}>{name}</h2>
27     <RoundButton text="add" onClick={onNew} />
28   </div>;
29
30 TopBar.propTypes = {
31   name: PropTypes.string,
32   onNew: PropTypes.func,
33 };
34
35 export default TopBar;
```

1.87 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import TopBar from './TopBar';
8
9 describe('<TopBar />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<TopBar
12       name="Example"
13     />);
14   });
15 });
```

1.88 Frame.jsx

```
1  import React from 'react';
2  import PropTypes from 'prop-types';
3  import Logo from '../Logo';
4
5  const style = {
6    width: '90%',
7    maxWidth: 960,
8    marginLeft: 'auto',
9    marginRight: 'auto',
10   marginTop: 40,
11 };
12
13 const Frame = ({ children }) => <div style={style}>
14   <Logo />
15   {children}
16 </div>;
17
18 Frame.propTypes = {
19   children: PropTypes.node,
20 };
21
22 export default Frame;
```

1.89 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import Frame from './Frame';
8
9 describe('<Frame />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<Frame>
12       <p>Hello</p>
13     </Frame>);
14   });
15 });
```

1.90 HomePage.jsx

```
1 import React from 'react';
2 import PropTypes from 'prop-types';
3 import Radium from 'radium';
4 import { Router, Route, browserHistory } from 'react-router';
5 import AuthFormContainer from '../containers/AuthFormContainer';
6 import ServiceListContainer from '../containers/ServiceListContainer';
7 import Frame from './Frame';
8
9 const HomePage = ({ authenticated }) => (
10   <Frame>
11     {authenticated ?
12       <ServiceListContainer />
13       :
14       <AuthFormContainer />
15     }
16   </Frame>
17 );
18
19 HomePage.propTypes = {
20   authenticated: PropTypes.bool,
21 };
22
23 /* eslint-disable new-cap */
24 export default Radium(HomePage);
```


1.91 Logo.jsx

```
1 import React from 'react';
2
3 const style = {
4   width: 140,
5 };
6
7 const Logo = () => (
8   <a href="/">
9     
10  </a>
11 );
12
13 export default Logo;
```

1.92 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import Logo from './Logo';
8
9 describe('<Logo />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<Logo />);
12
13   });
14 });
```

1.93 MethodButton.jsx

```
1 import React from 'react';
2 import PropTypes from 'prop-types';
3 import Radium from 'radium';
4 import { Color, Dimensions } from './StyleConstant';
5 import createMethods from '../utils/createMethods';
6
7 const {
8   naturalLanguage,
9   spreadsheet,
10  device,
11 } = createMethods;
12
13 const activeStyle = {
14   outline: 'none',
15 };
16
17 const style = {
18   base: {
19     minWidth: 150,
20     height: 170,
21     border: `${Dimensions.borderWidth}px solid ${Color.grey}`,
22     borderRadius: 10,
23     backgroundColor: Color.whiteText,
24     cursor: 'pointer',
25     margin: '0 13px',
26     transition: `${Dimensions.transitionTime.normal} all`,
27     fontSize: Dimensions.fontSize.normal,
28     color: Color.black,
29     ':hover': {
30       border: `${Dimensions.borderWidth}px solid ${Color.greenLight}`,
31     },
32     ':active': activeStyle,
33     ':focus': activeStyle,
34   },
35   selected: {
36     border: `${Dimensions.borderWidth}px solid ${Color.greenDark}`,
37     ':hover': {
38       border: `${Dimensions.borderWidth}px solid ${Color.green}`,
39     },
40   },
41   image: {
42     scratch: {
43       width: 83,
44     },
45     spreadsheet: {
46       width: 81,
47     },
48     device: {
49       width: 80,
50     },
51   },
52 }
```

```

51   },
52   inner: {
53     textAlign: 'center',
54     marginBottom: 20,
55   },
56 };
57
58 const MethodButton = ({ method, onClick, isSelected }) => {
59   let text;
60   let image;
61
62   switch (method) {
63     case naturalLanguage:
64       text = 'Scratch';
65       image = 'scratch';
66       break;
67     case spreadsheet:
68       text = 'Dataset';
69       image = 'spreadsheet';
70       break;
71     case device:
72       text = 'Device';
73       image = 'device';
74       break;
75   }
76
77   return (
78     <button
79       onClick={onClick} style={[
80         style.base,
81         isSelected ? style.selected : {},
82       ]}
83     >
84       <div style={style.inner}>
85         <img src={`/img/${image}.png`} style={style.image[image]} alt={text} />
86       </div>
87       {text}
88     </button>
89   );
90 };
91
92 MethodButton.propTypes = {
93   method: PropTypes.string,
94   onClick: PropTypes.func,
95   isSelected: PropTypes.bool,
96 };
97
98 export default Radium(MethodButton);

```

1.94 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6 import createMethods from '../utils/createMethods';
7
8 const {
9   naturalLanguage,
10 } = createMethods;
11
12 import MethodButton from './MethodButton';
13
14 describe('<MethodButton />', () => {
15   it('renders the component correctly', () => {
16     const wrapper = shallow(<MethodButton
17       method={naturalLanguage}
18       isSelected
19     />);
20   });
21 });
```

1.95 RoundButton.jsx

```
1  import React from 'react';
2  import PropTypes from 'prop-types';
3  import Radium from 'radium';
4  import { Color, Dimensions } from './StyleConstant';
5
6  const activeStyle = {
7    opacity: 1.0,
8    border: 'none',
9    outline: 'none',
10 };
11
12 const style = {
13   base: {
14     backgroundColor: Color.green,
15     width: Dimensions.fieldHeight,
16     height: Dimensions.fieldHeight,
17     border: 'none',
18     borderRadius: '50%',
19     cursor: 'pointer',
20     transition: `${Dimensions.transitionTime.normal} opacity`,
21     fontSize: 30,
22     backgroundSize: 'contain',
23     color: Color.whiteText,
24     ':hover': {
25       opacity: 0.8,
26     },
27     ':active': activeStyle,
28     ':focus': activeStyle,
29   },
30   isDisabled: {
31     pointerEvents: 'none',
32     backgroundColor: Color.grey,
33   },
34 };
35
36 const RoundButton = ({ text, onClick, isDisabled, color = Color.green, small = false }) => (
37   <button onClick={onClick} style={[style.base, isDisabled && style.isDisabled, { backgroundColor: color, backgroundImage: 'url(/img/${text}.png)'} ], small && { width: 25, height: 25 }} />
38 );
39
40 RoundButton.propTypes = {
41   text: PropTypes.string,
42   onClick: PropTypes.func,
43   isDisabled: PropTypes.bool,
44   color: PropTypes.string,
45   small: PropTypes.bool,
46 };
47
48
49 export default Radium(RoundButton);
```

1.96 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import RoundButton from './RoundButton';
8
9 describe('<RoundButton />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<RoundButton
12       text="add"
13       color="red"
14       small
15     />);
16   });
17 });
```

1.97 ServiceList.jsx

```
1  import React from 'react';
2  import PropTypes from 'prop-types';
3  import Radium from 'radium';
4  import RoundButton from './RoundButton';
5  import ServiceListItem from './ServiceListItem';
6
7  const style = {
8    p: {
9      textAlign: 'center',
10    },
11    list: {
12      display: 'flex',
13      flexDirection: 'column',
14      alignItems: 'center',
15    },
16  };
17
18  class ServiceList extends React.Component {
19    componentDidMount() {
20      this.props.onReady();
21    }
22
23    render() {
24      return (
25        <div>
26          <p style={style.p}>Which API would you like to work on?</p>
27          <div style={style.list}>
28            {this.props.services.map(service => <ServiceListItem key={`si-${service.id}`} onClick={() => this.props.onSelect(service.id)}
29              service={service} />)}
30            <RoundButton text="add" onClick={this.props.onCreate} />
31          </div>
32        </div>
33      );
34    }
35
36    ServiceList.propTypes = {
37      services: PropTypes.array,
38      onSelect: PropTypes.func,
39      onCreate: PropTypes.func,
40      onReady: PropTypes.func,
41    };
42
43    /* eslint-disable new-cap */
44    export default Radium(ServiceList);
```


1.98 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import ServiceList from './ServiceList';
8
9 describe('<ServiceList />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<ServiceList
12       services={} />);
13   });
14 });
15 });
```

1.99 ServiceListItem.jsx

```
1 import React from 'react';
2 import PropTypes from 'prop-types';
3 import Radium from 'radium';
4 import { Color, Dimensions } from './StyleConstant';
5
6 const style = {
7   marginBottom: 20,
8   border: `${Dimensions.borderWidth}px solid ${Color.grey}`,
9   borderRadius: 4,
10  width: 270,
11  height: 60,
12  cursor: 'pointer',
13  display: 'flex',
14  alignItems: 'center',
15  justifyContent: 'center',
16  transition: `all ${Dimensions.transitionTime.normal}`,
17  ':hover': {
18    border: `${Dimensions.borderWidth}px solid ${Color.green}`,
19  },
20 };
21
22 const ServiceListItem = ({ service, onClick }) => <div style={style} onClick={onClick}>
23   {service && service.name}
24 </div>;
25
26 ServiceListItem.propTypes = {
27   service: PropTypes.shape({
28     name: PropTypes.string,
29   }),
30   onClick: PropTypes.func,
31 };
32
33 export default Radium(ServiceListItem);
```

1.100 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import ServiceListItem from './ServiceListItem';
8
9 describe('<ServiceListItem />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<ServiceListItem
12       service={{
13         name: 'Example',
14       }}
15     />);
16   });
17 });
```

1.101 Setup.jsx

```
1  import React from 'react';
2  import PropTypes from 'prop-types';
3  import Radium from 'radium';
4  import Frame from '../Frame';
5  import SetupNameContainer from '../../containers/setup/SetupNameContainer';
6  import SetupMethodContainer from '../../containers/setup/SetupMethodContainer';
7  import SetupNaturalContainer from '../../containers/setup/SetupNaturalContainer';
8  import SetupSpreadsheetContainer from '../../containers/setup/SetupSpreadsheetContainer';
9  import {
10     SERVICE_SETUP_SCREEN_METHOD,
11     SERVICE_SETUP_SCREEN_NAME,
12     SERVICE_SETUP_SCREEN_NATURAL,
13     SERVICE_SETUP_SCREEN_SPREADSHEET,
14 } from '../../utils/setupScreens';
15
16 const Setup = ({ screen }) => {
17     let inner;
18
19     switch (screen) {
20         case SERVICE_SETUP_SCREEN_NAME:
21             inner = (<SetupNameContainer />);
22             break;
23         case SERVICE_SETUP_SCREEN_METHOD:
24             inner = (<SetupMethodContainer />);
25             break;
26         case SERVICE_SETUP_SCREEN_NATURAL:
27             inner = (<SetupNaturalContainer />);
28             break;
29         case SERVICE_SETUP_SCREEN_SPREADSHEET:
30             inner = (<SetupSpreadsheetContainer />);
31             break;
32         default:
33             inner = (<p>{'404 Setup screen not found'}</p>);
34     }
35
36     return (
37         <Frame>
38             {inner}
39         </Frame>
40     );
41 };
42
43 Setup.propTypes = {
44     screen: PropTypes.string,
45 };
46
47 /* eslint-disable new-cap */
48 export default Radium(Setup);
```

1.102 SetupMethod.jsx

```
1  import React from 'react';
2  import PropTypes from 'prop-types';
3  import Radium from 'radium';
4  import MethodButton from '../MethodButton';
5  import Button from '../Button';
6  import createMethods from '../../utils/createMethods';
7
8  const {
9    naturalLanguage,
10   spreadsheet,
11 } = createMethods;
12
13
14  const styles = {
15    nextButton: {
16      marginTop: 100,
17      float: 'right',
18    },
19    field: {
20      width: 700,
21      marginLeft: 'auto',
22      marginRight: 'auto',
23      textAlign: 'center',
24      marginTop: 100,
25    },
26    methods: {
27      display: 'flex',
28      justifyContent: 'center',
29    },
30  };
31
32  const SetupMethod = ({ method, onChange, onDone }) => (
33    <div>
34      <div style={styles.field}>
35        <p>How do you want to create your API?</p>
36        <div style={styles.methods}>
37          <MethodButton
38            method={naturalLanguage}
39            isSelected={method === naturalLanguage}
40            onClick={() => onChange(naturalLanguage)}
41          />
42          <MethodButton
43            method={spreadsheet}
44            isSelected={method === spreadsheet}
45            onClick={() => onChange(spreadsheet)}
46          />
47        </div>
48      </div>
49      <div style={styles.nextButton} >
50        <Button isDisabled={!method} onClick={onDone} text="Next" />
```

```
51     </div>
52   </div>
53 );
54
55 SetupMethod.propTypes = {
56   method: PropTypes.string,
57   onChange: PropTypes.func,
58   onDone: PropTypes.func,
59 };
60
61 /* eslint-disable new-cap */
62 export default Radium(SetupMethod);
```

1.103 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6 import createMethods from '../utils/createMethods';
7
8 const {
9   naturalLanguage,
10  spreadsheet,
11 } = createMethods;
12
13 import SetupMethod from './SetupMethod';
14
15 describe('<SetupMethod />', () => {
16   it('renders the component correctly', () => {
17     const wrapper = shallow(<SetupMethod
18       method={naturalLanguage}
19     />);
20   });
21 });
```

1.104 SetupName.jsx

```
1  import React from 'react';
2  import PropTypes from 'prop-types';
3  import Radium from 'radium';
4  import TextInput from '../TextInput';
5  import Button from '../Button';
6
7  const styles = {
8    nextButton: {
9      marginTop: 100,
10     float: 'right',
11   },
12   field: {
13     width: 500,
14     margin: 'auto',
15     textAlign: 'center',
16     marginTop: 100,
17   },
18 };
19
20 const SetupMethod = ({ name, onChange, onDone }) => (
21   <div>
22     <div style={styles.field}>
23       <p>What is the name of your API?</p>
24       <TextInput placeholder={'Name'} text={name} onChange={onChange} />
25     </div>
26     <div style={styles.nextButton}>
27       <Button onClick={onDone} text="Next" isDisabled={!name || !name.length} />
28     </div>
29   </div>
30 );
31
32 SetupMethod.propTypes = {
33   name: PropTypes.string,
34   onChange: PropTypes.func,
35   onDone: PropTypes.func,
36 };
37
38 /* eslint-disable new-cap */
39 export default Radium(SetupMethod);
```


1.105 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import SetupName from './SetupName';
8
9 describe('<SetupName />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<SetupName
12       name="Example"
13     />);
14   });
15 });
```

1.106 SetupNatural.jsx

```
1  import React from 'react';
2  import PropTypes from 'prop-types';
3  import Radium from 'radium';
4  import Button from '../Button';
5  import TextInput from '../TextInput';
6  import capitalizeString from '../../utils/capitalizeString';
7  import Model from '../../dashboard/structure/Model';
8
9  const styles = {
10    nextButton: {
11      marginTop: 100,
12      float: 'right',
13    },
14    field: {
15      width: 700,
16      marginLeft: 'auto',
17      marginRight: 'auto',
18      textAlign: 'center',
19      marginTop: 100,
20    },
21  };
22
23
24  const SetupNatural = ({ text, onChange, onDone, preview, nextEnabled }) => (
25    <div>
26      <div style={styles.field}>
27        <p>Please describe the various things and entities, <br />along with their properties and relationships</p>
28        <div>
29          <TextInput
30            text={text}
31            onChange={onChange}
32            long
33          />
34        </div>
35        {preview && preview.map(a => <Model enableInteractions={false} model={a} />)}
36      </div>
37      <div style={styles.nextButton}>
38        <Button disabled={!nextEnabled} onClick={onDone} text="Next" />
39      </div>
40    </div>
41  );
42
43  SetupNatural.propTypes = {
44    text: PropTypes.string,
45    onChange: PropTypes.func,
46    onDone: PropTypes.func,
47    preview: PropTypes.array,
48    nextEnabled: PropTypes.bool,
49  };
50
```

```
51  /* eslint-disable new-cap */  
52  export default Radium(SetupNatural);
```

1.107 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import SetupNatural from './SetupNatural';
8
9 describe('<SetupNatural />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<SetupNatural
12       text="A dog has a bone."
13       preview=[]
14       nextEnabled={false}
15     />);
16   });
17 });
```

1.108 SetupSpreadsheet.jsx

```
1  import React from 'react';
2  import PropTypes from 'prop-types';
3  import Radium from 'radium';
4  import Button from '../Button';
5  import TextInput from '../TextInput';
6  import capitalizeString from '../../utils/capitalizeString';
7  import Dropzone from 'react-dropzone';
8  import Model from '../../dashboard/structure/Model';
9
10 const styles = {
11   nextButton: {
12     marginTop: 100,
13     float: 'right',
14   },
15   field: {
16     width: 700,
17     marginLeft: 'auto',
18     marginRight: 'auto',
19     textAlign: 'center',
20     marginTop: 100,
21   },
22   sheet: {
23     width: '100%',
24     height: 300,
25     border: '2px solid gray',
26     borderRadius: 5,
27     borderStyle: 'dashed',
28     alignItems: 'center',
29     justifyContent: 'center',
30     display: 'flex',
31   },
32 };
33
34 const SetupSpreadsheet = ({ onChange, onDone, preview, nextEnabled }) => (
35   <div>
36     <div style={styles.field}>
37       <div>
38         <Dropzone onDrop={onChange} style={styles.sheet}>
39           Drop a spreadsheet file into this area
40         </Dropzone>
41       </div>
42       {preview && preview.map(a => <Model enableInteractions={false} model={a} />)}
43     </div>
44     <div style={styles.nextButton} >
45       <Button isDisabled={!nextEnabled} onClick={onDone} text="Next" />
46     </div>
47   </div>
48 );
49
50 SetupSpreadsheet.propTypes = {
```

```
51     onChange: PropTypes.func,
52     onDone: PropTypes.func,
53     preview: PropTypes.array,
54     nextEnabled: PropTypes.bool,
55   };
56
57   /* eslint-disable new-cap */
58   export default Radium(SetupSpreadsheet);
```

1.109 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import SetupSpreadsheet from './SetupSpreadsheet';
8
9 describe('<SetupSpreadsheet />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<SetupSpreadsheet
12       preview={[]}
13       nextEnabled={false}
14     />);
15   });
16 });
```

1.110 StyleConstant.js

```
1  export const Color = {
2    green: '#50E39C',
3    greenLight: '#54F0A5',
4    greenDark: '#4BD793',
5    red: '#FA6461',
6    redLight: '#FA706E',
7    redDark: '#EE5F5C',
8    whiteText: '#FFFFFF',
9    black: '#000000',
10   grey: '#C6C6C6',
11   lightGrey: '#E6E6E6',
12   lighterGrey: '#F8F9FB',
13 };
14
15 export const Dimensions = {
16   fieldHeight: 44,
17   fieldWidth: 125,
18   borderRadius: 3,
19   borderWidth: 2.9,
20   padding: 6,
21   fontSize: {
22     normal: 17,
23   },
24   transitionTime: {
25     normal: '0.25s',
26   }
27 }
28
29 export const lightBorder = '2px solid ${Color.lightGrey}';
```


1.111 TextInput.jsx

```
1  import React from 'react';
2  import PropTypes from 'prop-types';
3  import Radium from 'radium';
4  import { Color, Dimensions } from './StyleConstant';
5
6  const activeStyle = {
7    outline: 'none',
8    border: `${Dimensions.borderWidth}px solid ${Color.green}`,
9  };
10
11  const styles = {
12    base: {
13      border: `${Dimensions.borderWidth}px solid ${Color.grey}`,
14      minWidth: Dimensions.fieldWidth,
15      height: Dimensions.fieldHeight - Dimensions.borderWidth * 2,
16      borderRadius: Dimensions.borderRadius,
17      fontSize: Dimensions.fontSize.normal,
18      padding: `0 ${Dimensions.padding}px`,
19      transition: `${Dimensions.transitionTime.normal} all`,
20      ':active': activeStyle,
21      ':focus': activeStyle,
22    },
23    long: {
24      width: 500,
25      height: 130,
26      padding: Dimensions.padding,
27    },
28  };
29
30  const TextInput = ({ text, placeholder, onChange, long = false, name, type = 'text', id }) => (
31    long ? (
32      <textarea
33        value={text}
34        placeholder={placeholder}
35        onChange={e => onChange(e.target.value)}
36        style={[styles.base, styles.long]}
37        name={name}
38        id={id}
39      />
40    ) : (
41      <input
42        value={text}
43        name={name}
44        type={type}
45        placeholder={placeholder}
46        onChange={e => onChange(e.target.value)}
47        style={styles.base}
48        id={id}
49      />
50    )
  )
```

```
51
52 );
53
54 TextInput.propTypes = {
55   text: PropTypes.string.isRequired,
56   placeholder: PropTypes.string,
57   onChange: PropTypes.func,
58   long: PropTypes.bool,
59   name: PropTypes.string,
60   type: PropTypes.string,
61   id: PropTypes.any,
62 };
63
64 /* eslint-disable new-cap */
65 export default Radium(TextInput);
```

1.112 test.js

```
1
2 import React from 'react';
3 import { expect } from 'chai';
4 import { shallow } from 'enzyme';
5 import sinon from 'sinon';
6
7 import TextInput from './TextInput';
8
9 describe('<TextInput />', () => {
10   it('renders the component correctly', () => {
11     const wrapper = shallow(<TextInput
12       text="Hello"
13       placeholder="Example"
14       long
15       name="ok"
16       type="text"
17       id={4}
18     />);
19   });
20 });
```

1.113 AuthFormContainer.js

```
1 import { connect } from 'react-redux';
2 import {
3   updateUser,
4   authUser,
5 } from '../actions/auth';
6 import AuthForm from '../components/AuthForm';
7
8 const mapStateToProps = (state, ownProps) => ({
9   username: state
10     .getIn(['user', 'username']),
11   password: state
12     .getIn(['user', 'password']),
13   errors: state
14     .getIn(['user', 'errors']),
15 });
16
17 const mapDispatchToProps = (dispatch, ownProps) => ({
18   onSubmit: ({ username, password }) => dispatch(authUser(username, password)),
19   onChange: ({ username, password }) => dispatch(updateUser(username, password)),
20 });
21
22 const AuthFormContainer = connect(
23   mapStateToProps,
24   mapDispatchToProps,
25 )(AuthForm);
26
27 export default AuthFormContainer;
```

1.114 AboutContainer.js

```
1 import { connect } from 'react-redux';
2 import { updateService } from '../actions/dashboard/updateService';
3 import { updateServiceLocally } from '../actions/dashboard/updateServiceLocally';
4 import About from '../components/dashboard/about/About';
5
6
7 const mapStateToProps = (immutableState) => {
8   const state = immutableState.toJS();
9
10  const service = state.serviceById[state.user.currentServiceId];
11
12  return {
13    name: service.name,
14    meta: {
15      name: {
16        value: service.name,
17        label: 'Name',
18      },
19      shortName: {
20        value: service.shortName,
21        label: 'URL',
22      },
23      isPublic: {
24        value: service.isPublic,
25        label: 'Public?',
26      },
27    },
28  };
29 };
30
31 const mapDispatchToProps = dispatch => ({
32   onChange: (changes) => {
33     dispatch(updateServiceLocally(changes));
34     dispatch(updateService(changes));
35   },
36 });
37
38 const AboutContainer = connect(
39   mapStateToProps,
40   mapDispatchToProps,
41 )(About);
42
43 export default AboutContainer;
```

1.115 EntriesContainer.js

```
1 import { connect } from 'react-redux';
2 import { debounce, difference } from 'underscore';
3 import { changeSelectedModel } from '../../actions/dashboard/changeSelectedModel';
4 import { createEntry } from '../../actions/dashboard/createEntry';
5 import { deleteEntry } from '../../actions/dashboard/deleteEntry';
6 import { updateValue } from '../../actions/dashboard/updateValue';
7 import { updateValueLocally } from '../../actions/dashboard/updateValueLocally';
8 import Entries from '../../components/dashboard/entries/Entries';
9
10
11 const mapStateToProps = (immutableState) => {
12   const state = immutableState.toJS();
13
14   const service = state.serviceById[state.user.currentServiceId];
15
16   const selectedModel = state.dashboard.selectedModel || service.Models[0];
17
18   if (!service) {
19     return {};
20   }
21
22   const model = state.modelById[selectedModel];
23
24   if (!model) return {};
25
26   model.attributes = model.Attributes ? model.Attributes.map(i => state.attributeById[i]) : [];
27   model.entries = model.Entries ? model.Entries.map(i => state.entryById[i]) : [];
28
29   const headers = service.Models
30     .map(i => state.modelById[i])
31     .map(m => ({ id: m.id, text: m.name, selected: m.id === selectedModel }));
32   const attributes = model.attributes;
33
34   const entries = [];
35   for (const entry of model.entries) {
36     const obj = { id: entry.index, realId: entry.id };
37
38     const values = entry.Values ? entry.Values.map(i => state.valueById[i]) : [];
39
40     const missing = difference(
41       attributes.map(a => a.id),
42       values.map(v => v.Attribute || v.AttributeId),
43     ).map(id => state.attributeById[id]);
44
45     for (const valueObj of values) {
46       const value = valueObj.value;
47
48       const attr = state.attributeById[valueObj.Attribute || valueObj.AttributeId];
49
50       if (!attr) continue;
```

```
51     obj[attr.name] = { value, id: valueObj.id };
52   }
53
54   entries.push(obj);
55 }
56
57 return {
58   name: service.name,
59   headers,
60   attributes,
61   entries,
62 };
63 };
64
65 const mapDispatchToProps = (dispatch) => {
66   const update = debounce((id, attr, value) => dispatch(updateValue(id, attr, value)), 1000);
67
68   return {
69     onSelected: id => dispatch(changeSelectedModel(id)),
70     onCreate: () => dispatch(createEntry()),
71     onDelete: id => dispatch(deleteEntry(id)),
72     onUpdate: (id, attr, value, valueId) => {
73       dispatch(updateValueLocally(id, valueId, value));
74       update(id, attr, value);
75     },
76   };
77 };
78
79 const EntriesContainer = connect(
80   mapStateToProps,
81   mapDispatchToProps,
82 )(Entries);
83
84 export default EntriesContainer;
```

1.116 PagesContainer.js

```
1 import { connect } from 'react-redux';
2 import { updateModel } from '../../actions/dashboard/updateModel';
3 import { updateModelLocally } from '../../actions/dashboard/updateModelLocally';
4 import Pages from '../../components/dashboard/pages/Pages';
5
6
7 const mapStateToProps = (immutableState) => {
8   const state = immutableState.toJS();
9
10  const service = state.serviceById[state.user.currentServiceId];
11  const models = service.Models.map(id => state.modelById[id]);
12
13  const actions = [
14    {
15      label: 'Find',
16      prop: 'isFindEnabled',
17      method: 'GET',
18    },
19    {
20      label: 'Find One',
21      prop: 'isFindOneEnabled',
22      method: 'GET',
23    },
24    {
25      label: 'Create',
26      prop: 'isCreateEnabled',
27      method: 'POST',
28    },
29    {
30      label: 'Update',
31      prop: 'isUpdateEnabled',
32      method: 'PATCH',
33    },
34    {
35      label: 'Delete',
36      prop: 'isDeleteEnabled',
37      method: 'DELETE',
38    },
39  ];
40
41  const urlPrefix = `http://localhost:8000/api/${state.user.username}/${service.shortName}/`;
42
43  return {
44    name: service.name,
45    actions,
46    models,
47    urlPrefix,
48  };
49 };
50
```



```
51 const mapDispatchToProps = dispatch => ({
52   onChange: (changes) => {
53     dispatch(updateModelLocally(changes));
54     dispatch(updateModel(changes));
55   },
56 });
57
58 const PagesContainer = connect(
59   mapStateToProps,
60   mapDispatchToProps,
61 )(Pages);
62
63 export default PagesContainer;
```

1.117 SidebarContainer.js

```
1  import { connect } from 'react-redux';
2  import {
3    changeDashboardPage,
4  } from '../../actions/dashboard/changeDashboardPage';
5  import Sidebar from '../../components/dashboard/Sidebar';
6  import 'immutable';
7
8  const mapStateToProps = state => ({
9    items: state.getIn(['dashboard', 'items']).toJS(),
10 });
11
12 const mapDispatchToProps = dispatch => ({
13   onSelect: (index, item) => dispatch(changeDashboardPage(index, item)),
14 });
15
16
17 const SidebarContainer = connect(
18   mapStateToProps,
19   mapDispatchToProps,
20 )(Sidebar);
21
22 export default SidebarContainer;
```

1.118 StructureContainer.js

```
1 import { connect } from 'react-redux';
2 import {
3   } from '../actions/dashboard/changeSidebarItem';
4 import Structure from '../components/dashboard/structure/Structure';
5 import { selectAttribute } from '../actions/dashboard/selectAttribute';
6 import { createModel } from '../actions/dashboard/createModel';
7 import { createAttribute } from '../actions/dashboard/createAttribute';
8 import { deleteModel } from '../actions/dashboard/deleteModel';
9 import { deleteAttribute } from '../actions/dashboard/deleteAttribute';
10 import { updateModel } from '../actions/dashboard/updateModel';
11 import { updateAttribute } from '../actions/dashboard/updateAttribute';
12
13 const mapStateToProps = (immutableState) => {
14   const state = immutableState.toJS();
15
16   const service = state.serviceById[state.user.currentServiceId];
17
18   if (!service) {
19     return {};
20   }
21
22   const models = service.Models
23     .map(i => state.modelById[i])
24     .map(model => Object.assign(model, {
25       attributes: model.Attributes && model.Attributes.map(i => state.attributeById[i]),
26     }));
27
28   const selectedAttribute = state.dashboard.selectedAttribute &&
29     state.attributeById[state.dashboard.selectedAttribute];
30
31   return {
32     name: service.name,
33     models,
34     selectedAttribute,
35   };
36 };
37
38 const mapDispatchToProps = dispatch => ({
39   onSelectAttribute: id => dispatch(selectAttribute(id)),
40   onModelCreate: () => dispatch(createModel()),
41   onAttributeCreate: attribute => dispatch(createAttribute(attribute)),
42   onModelDelete: id => dispatch(deleteModel(id)),
43   onAttributeDelete: id => dispatch(deleteAttribute(id)),
44   onModelChange: (id, name) => dispatch(updateModel(id, name)),
45   onAttributeChange: (id, changes) => dispatch(updateAttribute(id, changes)),
46 });
47
48
49 const StructureContainer = connect(
50   mapStateToProps,
```

```
51     mapDispatchToProps ,  
52   )(Structure);  
53  
54   export default StructureContainer;
```

1.119 HomePageContainer.js

```
1 import { connect } from 'react-redux';
2 import HomePage from '../components/HomePage';
3
4 const mapStateToProps = state => ({
5   authenticated: state.getIn(['user', 'authenticated']),
6 });
7
8
9 const HomePageContainer = connect(
10   mapStateToProps,
11 )(HomePage);
12
13 export default HomePageContainer;
```


1.121 ServiceListContainer.js

```
1  import ServiceList from '../components/ServiceList';
2  import 'immutable';
3  import { connect } from 'react-redux';
4  import { push } from 'react-router-redux';
5  import {
6    selectService,
7    newService,
8  } from '../actions/setup';
9  import { getServiceList } from '../actions/auth/getServiceList';
10
11  const mapStateToProps = (state) => {
12    const services = state.getIn(['user', 'services'])
13      .map(id => state.getIn(['serviceById', `${id}`]))
14      .filter(e => !!e)
15      .toJS();
16
17    return {
18      services,
19    };
20  };
21
22  const mapDispatchToProps = dispatch => ({
23    onReady: () => dispatch(getServiceList()),
24    onSelect: id =>
25      dispatch(selectService(id)),
26    onCreate: () => dispatch(newService()),
27  });
28
29
30  const ServiceListContainer = connect(
31    mapStateToProps,
32    mapDispatchToProps,
33  )(ServiceList);
34
35  export default ServiceListContainer;
```

1.122 SetupContainer.js

```
1 import { connect } from 'react-redux';
2 import Setup from '../../components/setup/Setup';
3 import 'immutable';
4
5 const mapStateToProps = (state, ownProps) => ({
6   screen: state.getIn(['setup', 'screen']),
7 });
8
9 const SetupContainer = connect(
10   mapStateToProps,
11 )(Setup);
12
13 export default SetupContainer;
```


1.123 SetupMethodContainer.js

```
1 import { connect } from 'react-redux';
2 import {
3   setServiceCreateMethod,
4   nextScreen,
5 } from '../../actions/setup';
6 import SetupMethod from '../../components/setup/SetupMethod';
7
8 const mapStateToProps = state => ({
9   method: state
10     .get('setup')
11     .get('method'),
12 });
13
14 const mapDispatchToProps = dispatch => ({
15   onDone: () => dispatch(nextScreen()),
16   onChange: method => dispatch(setServiceCreateMethod(method)),
17 });
18
19 const SetupMethodContainer = connect(
20   mapStateToProps,
21   mapDispatchToProps,
22 )(SetupMethod);
23
24 export default SetupMethodContainer;
```

1.124 SetupNameContainer.js

```
1  import { connect } from 'react-redux';
2  import 'immutable';
3  import {
4    setServiceName,
5    nextScreen,
6  } from '../actions/setup';
7  import ServiceSetupName from '../components/setup/SetupName';
8
9  const mapStateToProps = state => ({
10    name: state
11      .get('setup')
12      .get('name'),
13  });
14
15  const mapDispatchToProps = dispatch => ({
16    onDone: () => dispatch(nextScreen()),
17    onChange: name => dispatch(setServiceName(name)),
18  });
19
20
21  const SetupNameContainer = connect(
22    mapStateToProps,
23    mapDispatchToProps,
24  )(ServiceSetupName);
25
26  export default SetupNameContainer;
```

1.125 SetupNaturalContainer.js

```
1  import { connect } from 'react-redux';
2  import 'immutable';
3  import {
4    analyseNaturalText,
5    createService,
6  } from '../actions/setup';
7  import SetupNatural from '../components/setup/SetupNatural';
8
9  const mapStateToProps = (state) => {
10    const preview = state.getIn(['setup', 'modelDefinitionPreview']);
11    console.log(preview, preview && preview.size);
12    return {
13      text: state.getIn(['setup', 'naturalText']),
14      preview: preview,
15      nextEnabled: preview && !!preview.length,
16    };
17  };
18
19  const mapDispatchToProps = dispatch => ({
20    onDone: () => dispatch(createService()),
21    onChange: text => dispatch(analyseNaturalText(text)),
22  });
23
24  const SetupNaturalContainer = connect(
25    mapStateToProps,
26    mapDispatchToProps,
27  )(SetupNatural);
28
29  export default SetupNaturalContainer;
```

1.126 SetupSpreadsheetContainer.js

```
1  import { connect } from 'react-redux';
2  import 'immutable';
3  import {
4    createService,
5  } from '../../actions/setup';
6  import { analyseSpreadsheet } from '../../actions/setup/analyseSpreadsheet';
7  import SetupSpreadsheet from '../../components/setup/SetupSpreadsheet';
8
9  const mapStateToProps = (state) => {
10    const preview = state.getIn(['setup', 'modelDefinitionPreview']);
11    return {
12      file: state.getIn(['setup', 'file']),
13      preview: preview,
14      nextEnabled: preview && !!preview.length,
15    };
16  };
17
18
19  const mapDispatchToProps = dispatch => ({
20    onDone: () => dispatch(createService()),
21    onChange: ([file]) => dispatch(analyseSpreadsheet(file)),
22  });
23
24
25  const SetupSpreadsheetContainer = connect(
26    mapStateToProps,
27    mapDispatchToProps,
28  )(SetupSpreadsheet);
29
30  export default SetupSpreadsheetContainer;
```

1.127 index.css

```
1
2
3 /** Ultra Light */
4 @font-face {
5     font-family: "San Francisco";
6     font-weight: 100;
7     src: url("https://applesocial.s3.amazonaws.com/assets/styles/fonts/sanfrancisco/sanfranciscodisplay-ultralight-webfont.woff");
8 }
9
10 /** Thin */
11 @font-face {
12     font-family: "San Francisco";
13     font-weight: 200;
14     src: url("https://applesocial.s3.amazonaws.com/assets/styles/fonts/sanfrancisco/sanfranciscodisplay-thin-webfont.woff");
15 }
16
17 /** Regular */
18 @font-face {
19     font-family: "San Francisco";
20     font-weight: 400;
21     src: url("https://applesocial.s3.amazonaws.com/assets/styles/fonts/sanfrancisco/sanfranciscodisplay-regular-webfont.woff");
22 }
23
24 /** Medium */
25 @font-face {
26     font-family: "San Francisco";
27     font-weight: 500;
28     src: url("https://applesocial.s3.amazonaws.com/assets/styles/fonts/sanfrancisco/sanfranciscodisplay-medium-webfont.woff");
29 }
30
31 /** Semi Bold */
32 @font-face {
33     font-family: "San Francisco";
34     font-weight: 600;
35     src: url("https://applesocial.s3.amazonaws.com/assets/styles/fonts/sanfrancisco/sanfranciscodisplay-semibold-webfont.woff");
36 }
37
38 /** Bold */
39 @font-face {
40     font-family: "San Francisco";
41     font-weight: 700;
42     src: url("https://applesocial.s3.amazonaws.com/assets/styles/fonts/sanfrancisco/sanfranciscodisplay-bold-webfont.woff");
43 }
44
45 body {
46     /*font-family: "San Francisco", sans-serif;*/
47     font-family: sans-serif;
48     font-weight: 500;
49     letter-spacing: 0.4px;
50     font-size: 18px;
```

```
51     margin: 0;
52
53     -webkit-font-smoothing: antialiased;
54 }
```

1.128 index.js

```
1  import React from 'react';
2  import { render } from 'react-dom';
3  import { Provider } from 'react-redux';
4  import { createStore, applyMiddleware } from 'redux';
5  import { Router, IndexRedirect, Route, browserHistory } from 'react-router';
6  import thunk from 'redux-thunk';
7  import { syncHistoryWithStore, routerMiddleware } from 'react-router-redux';
8  import easyAPI from './reducers';
9  import './index.css';
10 import SetupContainer from './containers/setup/SetupContainer';
11 import Dashboard from './components/dashboard/Dashboard';
12 import StructureContainer from './containers/dashboard/StructureContainer';
13 import EntriesContainer from './containers/dashboard/EntriesContainer';
14 import AboutContainer from './containers/dashboard/AboutContainer';
15 import PagesContainer from './containers/dashboard/PagesContainer';
16 import ServiceListContainer from './containers/ServiceListContainer';
17 import HomePageContainer from './containers/HomePageContainer';
18 import { isAuthenticated } from './utils/Auth';
19
20 const middleware = routerMiddleware(browserHistory);
21 const store = createStore(easyAPI,
22   window.__REDUX_DEVTOOLS_EXTENSION__ && window.__REDUX_DEVTOOLS_EXTENSION__(),
23   applyMiddleware(thunk, middleware),
24 );
25
26 const history = syncHistoryWithStore(browserHistory, store, {
27   selectLocationState(state) {
28     return state.get('routing').toJS();
29   },
30 });
31
32 function requireAuth(nextState, replace) {
33   console.log(nextState);
34   const isStuck = !store.getState().getIn(['user', 'currentServiceId']) && (nextState.location.pathname !== '/service/setup');
35
36   if (!isAuthenticated() || isStuck) {
37     return replace({
38       pathname: '/',
39     });
40   }
41 }
42
43 const r = () => render(
44   <Provider store={store}>
45     <Router history={history}>
46       <Route
47         path="/"
48         component={HomePageContainer}
49       />
50       <Route
```

```
51     path="/services"
52     component={ServiceListContainer}
53   />
54   <Route
55     path="/service/setup"
56     component={SetupContainer}
57     onEnter={requireAuth}
58   />
59   <Route
60     path="/service/dashboard"
61     component={Dashboard}
62     onEnter={requireAuth}
63   >
64     <Route
65       path="structure"
66       component={StructureContainer}
67     />
68     <Route
69       path="entries"
70       component={EntriesContainer}
71     />
72     <Route
73       path="pages"
74       component={PagesContainer}
75     />
76     <Route
77       path="about"
78       component={AboutContainer}
79     />
80     <IndexRedirect to="structure" />
81   </Route>
82
83   </Router>
84 </Provider>,
85 document.getElementById('root'),
86 );
87
88 r();
89 store.subscribe(r);
```


1.129 index.js

```
1 // @flow
2 import { List, Map, fromJS } from 'immutable';
3 import {
4   UPDATE_MODEL_PREVIEW,
5   NEW_SERVICE,
6   RECEIVE_WEBHOOK_URL,
7   SELECT_DEVICE,
8   SET_DEVICE_FLOW_DIRECTION,
9   SET_SERVICE_CREATE_METHOD,
10  SET_SERVICE_NAME,
11  SETUP_DEVICE_QUERY,
12  NEXT_SCREEN,
13  UPDATE_NATURAL_TEXT,
14  UPDATE_USER,
15  AUTH_USER_RESULT,
16  LOGOUT_USER,
17  CHANGE_SIDEBAR_ITEM,
18  RECEIVE_SERVICE_LIST,
19  SELECT_SERVICE,
20  RECEIVE_SERVICE,
21  CHANGE_SELECTED_MODEL,
22  RECEIVE_ENTRY,
23  DELETE_ENTRY_LOCALLY,
24  UPDATE_VALUE_LOCALLY,
25  UPDATE_SERVICE_LOCALLY,
26  SELECT_ATTRIBUTE,
27  RECEIVE_MODEL,
28  RECEIVE_ATTRIBUTE,
29  DELETE_MODEL_LOCALLY,
30  DELETE_ATTRIBUTE_LOCALLY,
31  UPDATE_ATTRIBUTE_LOCALLY,
32  UPDATE_MODEL_LOCALLY,
33 } from '../actions/actionTypes';
34 import capitalizeString from '../utils/capitalizeString';
35 import formatSentences from '../utils/formatSentences';
36 import createMethods from '../utils/createMethods';
37 import { isAuthenticated, getToken } from '../utils/Auth';
38 import { normalizeServices, normalizeService, normalizeEntry, normalizeModel, normalizeAttribute } from '../utils/normalizr';
39 import {
40   LOCATION_CHANGE,
41 } from 'react-router-redux';
42 import {
43   SERVICE_SETUP_SCREEN_METHOD,
44   SERVICE_SETUP_SCREEN_NAME,
45   SERVICE_SETUP_SCREEN_NATURAL,
46   SERVICE_SETUP_SCREEN_SPREADSHEET,
47   SERVICE_SETUP_SCREEN_DEVICE,
48 } from '../utils/setupScreens';
49
50
```

```
51  const {
52    naturalLanguage,
53    spreadsheet,
54    device,
55  } = createMethods;
56
57
58  const NEW_ID = '-1';
59
60
61  const defaultState = fromJS({
62    routing: {
63      locationBeforeTransitions: null,
64    },
65    user: {
66      currentServiceId: null,
67      username: '',
68      password: '',
69      authenticated: isAuthenticated(),
70      services: [],
71      token: getToken(),
72    },
73    dashboard: {
74      items: [
75        {
76          name: 'Structure',
77          path: '/service/dashboard/structure',
78          selected: true,
79        },
80        {
81          name: 'Entries',
82          path: '/service/dashboard/entries',
83        },
84        {
85          name: 'Pages',
86          path: '/service/dashboard/pages',
87        },
88        {
89          name: 'About',
90          path: '/service/dashboard/about',
91        },
92      ],
93      selectedAttribute: null,
94      selectedModel: null,
95    },
96    setup: {
97      name: '',
98      screen: 'SERVICE_SETUP_SCREEN_NAME',
99      method: 'CREATE_METHOD_NATURAL_LANGUAGE',
100    },
101    serviceById: {
102    },
103    modelById: {
```

```

104     },
105     attributeById: {
106     },
107     entryById: {},
108     valueById: {},
109     endpointById: {},
110 });
111
112 function easyAPI(state = defaultState, action) {
113     switch (action.type) {
114         case LOCATION_CHANGE: {
115             return state.setIn(['routing', 'locationBeforeTransitions'], action.payload);
116         }
117         case NEW_SERVICE: {
118             return state
119                 .setIn(['user', 'currentServiceId'], NEW_ID)
120                 .setIn(['serviceById', NEW_ID], Map({
121                     id: NEW_ID,
122                     name: null,
123                     author: state.getIn(['user', 'name']),
124                     models: List(),
125                     endpoints: List(),
126                 }));
127         }
128         case NEXT_SCREEN: {
129             switch (state.getIn(['setup', 'screen'])) {
130                 case SERVICE_SETUP_SCREEN_NAME:
131                     return state.setIn(['setup', 'screen'], SERVICE_SETUP_SCREEN_METHOD);
132                 break;
133                 case SERVICE_SETUP_SCREEN_METHOD:
134                     const method = state.getIn(['setup', 'method']);
135                     switch (method) {
136                         case naturalLanguage:
137                             return state.setIn(['setup', 'screen'], SERVICE_SETUP_SCREEN_NATURAL);
138                         break;
139                         case spreadsheet:
140                             return state.setIn(['setup', 'screen'], SERVICE_SETUP_SCREEN_SPREADSHEET);
141                         break;
142                         case device:
143                             return state.setIn(['setup', 'screen'], SERVICE_SETUP_SCREEN_NAME);
144                         break;
145                         default:
146                             return state;
147                     }
148                 break;
149             }
150
151             return state;
152         }
153         case SET_SERVICE_NAME: {
154             console.log('capitalizeString', capitalizeString('ok'));
155             return state
156                 .setIn([

```

```
157         'setup',
158         'name',
159         ], capitalizeString(action.name));
160     }
161     case SET_SERVICE_CREATE_METHOD: {
162         const newState = state
163         .setIn([
164             'setup',
165             'method',
166             ], action.method);
167         switch (action.method) {
168             case naturalLanguage: {
169                 return newState
170                 .setIn(['setup', 'naturalText'], '');
171             }
172             case spreadsheet: {
173                 return newState
174                 .setIn(['setup', 'spreadsheet'], '');
175             }
176             case device: {
177                 return newState
178                 .setIn(['setup', 'device'], Map({
179                     selectedDevice: '',
180                     flowDirection: null,
181                 }));
182             }
183             default: {
184                 return newState;
185             }
186         }
187     }
188     case UPDATE_NATURAL_TEXT: {
189         return state.setIn(['setup', 'naturalText'], formatSentences(action.text));
190     }
191     case UPDATE_MODEL_PREVIEW: {
192         return state
193         .setIn(['setup', 'modelDefinitionPreview'], action.preview);
194     }
195     case SELECT_DEVICE: {
196         return state
197         .setIn(['setup', 'selectedDevice'], state.device);
198     }
199     case SET_DEVICE_FLOW_DIRECTION: {
200         return state
201         .setIn(['setup', 'flowDirection'], state.direction);
202     }
203     case RECEIVE_WEBHOOK_URL: {
204         return state
205         .setIn(['setup', 'webhookURL'], state.url);
206     }
207     case SETUP_DEVICE_QUERY: {
208         return state
209         .setIn(['setup', 'query'], Map({
```

```

210         url: action.url,
211         method: action.method,
212         attributes: action.attributes,
213         interval: action.interval,
214     }));
215 }
216 case UPDATE_USER: {
217     const { username, password } = action;
218
219     if (username && password) {
220         return state
221             .setIn(['user', 'username'], username)
222             .setIn(['user', 'password'], password);
223     } else if (username) {
224         return state
225             .setIn(['user', 'username'], username);
226     }
227     return state
228         .setIn(['user', 'password'], password);
229 }
230 case AUTH_USER_RESULT: {
231     return state
232         .setIn(['user', 'authenticated'], action.success)
233         .setIn(['user', 'errors'], action.errors)
234         .setIn(['user', 'token'], action.token);
235 }
236 case LOGOUT_USER: {
237     return state
238         .setIn(['user', 'authenticated'], false)
239         .setIn(['user', 'errors'], null)
240         .setIn(['user', 'token'], null);
241 }
242 case CHANGE_SIDEBAR_ITEM: {
243     // console.log('CHANGE_SIDEBAR_ITEM', state.getIn(['dashboard', 'items']).map((item, i) => item.set('selected', i === action.index)).
244         toJS());
245     return state
246         .setIn(
247             ['dashboard', 'items'],
248             state.getIn(['dashboard', 'items']).map((item, i) => item.set('selected', i === action.index)),
249         );
250 }
251 case RECEIVE_SERVICE_LIST: {
252     const services = action.services;
253
254     const entities = normalizeServices({ services }).entities;
255
256     const serviceIds = entities.services.undefined.services;
257
258     const serviceById = entities.service || {};
259     const modelById = entities.model || {};
260     const attributeById = entities.attribute || {};
261     const entryById = entities.entry || {};
262     const valueById = entities.value || {};

```

```

262
263     return state
264     .setIn(['user', 'services'], fromJS(serviceIds))
265     .set('serviceById', fromJS(serviceById).merge(state.get('serviceById')))
266     .set('modelById', fromJS(modelById).merge(state.get('modelById')))
267     .set('attributeById', fromJS(attributeById).merge(state.get('attributeById')))
268     .set('entryById', fromJS(entryById).merge(state.get('entryById')))
269     .set('valueById', fromJS(valueById).merge(state.get('valueById')));
270 }
271 case RECEIVE_SERVICE: {
272     // TODO
273     console.log(JSON.stringify(action.service));
274
275     const entities = normalizeService(action.service).entities;
276
277     console.log(entities);
278     const serviceById = entities.service || {};
279     const modelById = entities.model || {};
280     const attributeById = entities.attribute || {};
281     const entryById = entities.entry || {};
282     const valueById = entities.value || {};
283
284     return state
285     .setIn(['user', 'currentServiceId'], action.service.id)
286     .set('serviceById', fromJS(serviceById).merge(state.get('serviceById')))
287     .set('modelById', fromJS(modelById).merge(state.get('modelById')))
288     .set('attributeById', fromJS(attributeById).merge(state.get('attributeById')))
289     .set('entryById', fromJS(entryById).merge(state.get('entryById')))
290     .set('valueById', fromJS(valueById).merge(state.get('valueById')));
291 }
292 case RECEIVE_ENTRY: {
293     const entities = normalizeEntry(action.entry).entities;
294
295     const model = action.entry.ModelId;
296
297     const valueById = entities.value || {};
298     const entryById = entities.entry || {};
299
300     const entryIdsPath = ['modelById', `${model}`, 'Entries'];
301
302     return state
303     .setIn(entryIdsPath, state.getIn(entryIdsPath).push(action.entry.id))
304     .set('entryById', fromJS(entryById).merge(state.get('entryById')))
305     .set('valueById', fromJS(valueById).merge(state.get('valueById')));
306 }
307 case RECEIVE_MODEL: {
308     const entities = normalizeModel(action.model).entities;
309
310     const modelById = entities.model || {};
311
312     const modelIdsPath = ['serviceById', `${state.getIn(['user', 'currentServiceId'])}`, 'Models'];
313
314     return state

```

```

315     .setIn(modelIdsPath, state.getIn(modelIdsPath).push(action.model.id))
316     .set('modelById', fromJS(modelById).merge(state.get('modelById')));
317 }
318 case RECEIVE_ATTRIBUTE: {
319     const entities = normalizeAttribute(action.attribute).entities;
320
321     const attributeById = entities.attribute || {};
322
323     const attributeIdsPath = ['modelById', `${action.attribute.ModelId}`, 'Attributes'];
324
325     return state
326         .setIn(attributeIdsPath, (state.getIn(attributeIdsPath) || fromJS([])).push(action.attribute.id))
327         .set('attributeById', fromJS(attributeById).merge(state.get('attributeById')));
328 }
329 case SELECT_SERVICE: {
330     return state.setIn(
331         [
332             'user', 'currentServiceId',
333         ],
334         action.id,
335     );
336 }
337 case CHANGE_SELECTED_MODEL: {
338     return state.setIn(
339         [
340             'dashboard', 'selectedModel',
341         ],
342         action.id,
343     );
344 }
345 case DELETE_ENTRY_LOCALLY: {
346     const entries = ['modelById', `${action.entry.ModelId}`, 'Entries'];
347     return state
348         .deleteIn(['entryById', `${action.entry.id}`])
349         .setIn(entries, state.getIn(entries).filter(i => i !== action.entry.id));
350 }
351 case DELETE_MODEL_LOCALLY: {
352     const models = ['serviceById', `${state.getIn(['user', 'currentServiceId'])}`, 'Models'];
353     return state
354         .deleteIn(['modelById', `${action.id}`])
355         .setIn(models, state.getIn(models).filter(i => i !== action.id));
356 }
357 case DELETE_ATTRIBUTE_LOCALLY: {
358     const modelId = state.getIn(['attributeById', `${action.id}`, 'ModelId']);
359     const attributes = ['modelById', `${modelId}`, 'Attributes'];
360     return state
361         .deleteIn(['attributeById', `${action.id}`])
362         .setIn(attributes, state.getIn(attributes).filter(i => i !== action.id));
363 }
364 case UPDATE_VALUE_LOCALLY: {
365     const valueIdsPath = ['valueById', `${action.value}`, 'Entries'];
366
367     return state

```

```
368     .setIn(['valueById', `${action.id}`, 'value'], action.value);
369   }
370   case UPDATE_SERVICE_LOCALLY: {
371     const servicePath = ['serviceById', `${state.getIn(['user', 'currentServiceId'])}`];
372
373     return state
374       .setIn(servicePath, state.getIn(servicePath).merge(fromJS(action.changes)));
375   }
376   case SELECT_ATTRIBUTE: {
377     return state
378       .setIn(['dashboard', 'selectedAttribute'], action.id);
379   }
380   case UPDATE_MODEL_LOCALLY: {
381     return state
382       .setIn(['modelById', `${action.id}`, 'name'], action.name);
383   }
384   case UPDATE_ATTRIBUTE_LOCALLY: {
385     const path = ['attributeById', `${action.id}`];
386     return state
387       .setIn(path, state.getIn(path).merge(fromJS(action.changes)));
388   }
389   default:
390     return state;
391 }
392 }
393
394 export default easyAPI;
```


1.130 test.js

```
1 import reducer from './index';
2 import { List, Map, fromJS } from 'immutable';
3 import * as types from '../actions/actionTypes';
4
5 const defaultState = fromJS({
6   routing: {
7     locationBeforeTransitions: null,
8   },
9   user: {
10    currentServiceId: '1',
11    username: '',
12    password: '',
13    authenticated: undefined,
14    services: [],
15    token: undefined,
16  },
17  dashboard: {
18    items: [
19      {
20        name: 'Structure',
21        path: '/service/dashboard/structure',
22        selected: true,
23      },
24      {
25        name: 'Entries',
26        path: '/service/dashboard/entries',
27      },
28      {
29        name: 'Pages',
30        path: '/service/dashboard/pages',
31      },
32      {
33        name: 'About',
34        path: '/service/dashboard/about',
35      },
36    ],
37    selectedAttribute: null,
38    selectedModel: null,
39  },
40  setup: {
41    name: '',
42    screen: 'SERVICE_SETUP_SCREEN_NAME',
43    method: 'CREATE_METHOD_NATURAL_LANGUAGE',
44  },
45  serviceById: {
46  },
47  modelById: {
48  },
49  attributeById: {
50  },
```

```
51     entryById: {},
52     valueById: {},
53     endpointById: {},
54   });
55
56   describe('easyAPI reducer', () => {
57     it('should return the initial state', () => {
58       expect(
59         reducer(undefined, {}),
60       ).toEqual(defaultState);
61     });
62   });
```

1.131 annotateText.js

```
1 export default function annotateText(text = '', preview = []) {
2   let html = text;
3   let adj = 0;
4   preview.forEach(a => {
5     html = html.substr(0, a.start + adj) + '<b>' + html.substr(a.start + adj);
6     adj += 3;
7     html = html.substr(0, a.end + adj) + '</b>' + html.substr(a.end + adj);
8     adj += 4;
9   });
10  return html;
11 }
```

1.132 API.js

```
1 import { getToken } from './Auth';
2
3 function curryReq(path, useToken = true, method = 'POST') {
4   return async (params) => {
5     const headers = {
6       'Content-Type': 'application/json',
7     };
8
9     if (useToken) {
10       headers.Authorization = `bearer ${getToken()}`;
11     }
12
13     const response = await fetch(`/api${path}`, {
14       method,
15       headers,
16       body: JSON.stringify(params),
17     });
18
19     const json = await response.json();
20     console.log(`API response ${method} ${path}`, params, json);
21     return json;
22   };
23 }
24
25 export const req = (path, params) => curryReq(path)(params);
26
27 export const extractModelFromText = text => curryReq('/service/parseText')({ text });
28
29 export const authenticateUser = (username, password) => curryReq('/auth/login', false)({ username, password });
30
31 export const getService = id => curryReq(`/service/${id}`, true, 'GET')({});
32
33 export const getServiceList = () => curryReq('/service', true, 'GET')();
34
35 export const postService = (name, models) => curryReq('/service', true, 'POST')({ name, models });
36
37 export const postEntry = model => curryReq('/entry', true, 'POST')({ model });
38 export const deleteEntry = id => curryReq('/entry', true, 'DELETE')({ id });
39 export const updateValue = (entry, attribute, value) => curryReq('/value', true, 'PATCH')({ entry, attribute, value });
40 export const updateService = (id, changes) => curryReq(`/service/${id}`, true, 'PATCH')(changes);
41
42 export const postModel = curryReq('/model', true, 'POST');
43 export const deleteModel = curryReq('/model', true, 'DELETE');
44 export const patchModel = obj => curryReq(`/model/${obj.id}`, true, 'PATCH')(obj);
45
46 export const postAttribute = curryReq('/attribute', true, 'POST');
47 export const patchAttribute = obj => curryReq(`/attribute/${obj.id}`, true, 'PATCH')(obj);
48 export const deleteAttribute = curryReq('/attribute', true, 'DELETE');
49
50
```

```
51 export async function postAnalyzeSpreadsheet(file) {
52   const formData = new FormData();
53   formData.append('spreadsheet', file);
54
55   const headers = {
56   };
57
58   headers.Authorization = `bearer ${getToken()}`;
59
60   const response = await fetch('/api/service/parseSpreadsheet', {
61     method: 'POST',
62     headers,
63     body: formData,
64   });
65
66   const json = await response.json();
67   console.log('FILE API response', json);
68   return json;
69 }
```

1.133 Auth.js

```
1 // let savedToken;
2
3 const localStorage = window.localStorage || null;
4
5
6 export function saveToken(token) {
7   // savedToken = token;
8   if (!localStorage) return;
9   localStorage.setItem('token', token);
10 }
11 export function isAuthenticated() {
12   // return savedToken != null;
13   if (!localStorage) return;
14   return localStorage.getItem('token') != null;
15 }
16 export function removeToken() {
17   // savedToken = undefined;
18   if (!localStorage) return;
19   localStorage.removeItem('token');
20 }
21 export function getToken() {
22   // return savedToken;
23   if (!localStorage) return;
24   return localStorage.getItem('token');
25 }
```

1.134 capitalizeString.js

```
1  const capitalizeWord = (str) => str.charAt(0).toUpperCase() + str.slice(1);
2
3  function capitalizeString(str) {
4    return str.split(/\s+/).map(capitalizeWord).join(' ');
5  }
6
7  export default capitalizeString;
```

1.135 createMethods.js

```
1  const naturalLanguage = 'CREATE_METHOD_NATURAL_LANGUAGE';
2  const spreadsheet = 'CREATE_METHOD_SPREADSHEET';
3  const device = 'CREATE_METHOD_DEVICE';
4
5  const createMethods = {
6    naturalLanguage,
7    spreadsheet,
8    device,
9  }
10
11  export default createMethods;
```


1.136 formatSentences.js

```
1  const capitalize = (str) => str.charAt(0).toUpperCase() + str.slice(1);
2
3  function formatSentences(str) {
4    return str.split(/\.\s+/)
5      .map(capitalize)
6      .join('. ')
7      // .replace(/\s+/g, ' ')
8      // .replace(/\s+\./, '.')
9      // .replace(/([,\.])[\.]+/, '$1');
10   // TODO Improve sentence splitting
11
12 }
13
14 export default formatSentences;
```

1.137 normalizr.js

```
1  import { normalize, schema } from 'normalizr';
2
3  const attribute = new schema.Entity('attribute');
4
5  const value = new schema.Entity('value', {
6    Attribute: attribute,
7  });
8
9  const entry = new schema.Entity('entry', {
10    Values: [value],
11  });
12
13  const model = new schema.Entity('model', {
14    Attributes: [attribute],
15    Entries: [entry],
16  });
17
18  const endpoint = new schema.Entity('endpoint');
19
20  const service = new schema.Entity('service', {
21    Endpoints: [endpoint],
22    Models: [model],
23  });
24
25  const services = new schema.Entity('services', {
26    services: [service],
27  });
28
29  export const normalizeServices = data => normalize(data, services);
30  export const normalizeService = data => normalize(data, service);
31  export const normalizeEntry = data => normalize(data, entry);
32  export const normalizeModel = data => normalize(data, model);
33  export const normalizeAttribute = data => normalize(data, attribute);
```

1.138 setupScreens.js

```
1 export const SERVICE_SETUP_SCREEN_NAME = 'SERVICE_SETUP_SCREEN_NAME';
2 export const SERVICE_SETUP_SCREEN_METHOD = 'SERVICE_SETUP_SCREEN_METHOD';
3 export const SERVICE_SETUP_SCREEN_NATURAL = 'SERVICE_SETUP_SCREEN_NATURAL';
4 export const SERVICE_SETUP_SCREEN_SPREADSHEET = 'SERVICE_SETUP_SCREEN_SPREADSHEET';
5 export const SERVICE_SETUP_SCREEN_DEVICE = 'SERVICE_SETUP_SCREEN_DEVICE';
```

1.139 natural.js

```
1  import request from 'request-promise';
2  import compromise from 'nlp_compromise';
3
4  /**
5   * Natural Service: A service for extracting information from natural speech.
6   */
7  import { sentences as seperateSentences } from 'sbd';
8
9  // Uses spacy to deconstruct text into a dependancy parse tree
10 function parse(text) {
11   return request.post('http://localhost:5000/parse', {
12     form: {
13       text: seperateSentences(text).join('<#SENT_SEPERATOR#>'),
14     },
15   })
16   .then(res => JSON.parse(res));
17 }
18
19 // In the dependency parse tree it finds first object which satisfies the condition
20 function find(object, condition) {
21   if (condition(object)) return object;
22
23   if (!object || !object.modifiers || object.modifiers.length === 0) return null;
24   for (const child of object.modifiers) {
25     const result = find(child, condition);
26     if (result) return result;
27   }
28   return null;
29 }
30
31 // In the dependency parse tree it finds all objects which satisfy the condition
32 function findAll(object, condition) {
33   let found = [];
34   if (condition(object)) found.push(object);
35
36   if (!object || !object.modifiers || object.modifiers.length === 0) return found;
37
38   for (const child of object.modifiers) {
39     const result = findAll(child, condition);
40     if (result.length) found = [...result, ...found];
41   }
42   return found;
43 }
44
45 // From an array of booleans decide the final value
46 function decide(values) {
47   if (values.length === 0) return null;
48
49   let sum = 0;
50   for (const value of values) {
```

```

51     sum += Number(value);
52 }
53 return sum / values.length >= 0.5;
54 }
55
56 // Finds the existence of property. Returns string of 'required', 'optional', 'unknown'
57 function findIfPropertyIsRequired(prop, context) {
58     // https://en.wikipedia.org/wiki/Auxiliary_verb
59     const optionalKeywords = ['may', 'might', 'could', 'should', 'maybe', 'possible', 'possibly', 'optionally', 'optional', 'ought'];
60     const requiredKeywords = ['must', 'needs', 'need', 'shall', 'will'];
61
62     const allRequiredInformation = [];
63
64     // Find if the relationship has monads attached
65     if (!context.modifiers || !context.modifiers.length) return false;
66     const monads = context.modifiers.filter(o => o.arc === 'aux');
67
68     for (const monad of monads) {
69         if (optionalKeywords.find(k => k === monad.lemma)) {
70             allRequiredInformation.push(false);
71         } else if (requiredKeywords.find(k => k === monad.lemma)) {
72             allRequiredInformation.push(true);
73         }
74     }
75
76     return decide(allRequiredInformation) || false;
77 }
78
79 // Finds if a property has multiple instances
80 function findIfPropertyHasMultiple(prop) {
81     const determiners = prop.modifiers ? prop.modifiers.filter(o => o.arc === 'det') : [];
82     const adjModifiers = prop.modifiers ? prop.modifiers.filter(o => o.arc === 'amod') : [];
83     const numModifiers = prop.modifiers ? prop.modifiers.filter(o => o.arc === 'nummod') : [];
84
85     const combined = determiners.concat(adjModifiers).concat(numModifiers);
86     // console.log(prop.lemma, ' findupper ', combined);
87
88     // If the noun is plural then it will be multiple
89     if (prop.POS_fine === 'NNS') {
90         return true;
91     }
92
93     if (combined.length === 0) return false;
94
95     // Find all information related to upper bound
96     const allCardinalityInfo = [];
97     for (const modifier of combined) {
98         const singleKeywords = ['a', 'single', 'one'];
99         const multipleKeywords = ['many', 'multiple', 'several'];
100         // const singleNumbers = ['one', 'zero'];
101
102         if (modifier.arc === 'nummod') {
103             // Parse value of number

```

```

104     allCardinalityInfo.push(compromise.value(modifier.lemma).number > 1);
105 }
106
107 if (singleKeywords.find(k => k === modifier.lemma)) {
108     allCardinalityInfo.push(false);
109 } else if (multipleKeywords.find(k => k === modifier.lemma)) {
110     allCardinalityInfo.push(true);
111 }
112 }
113
114 return decide(allCardinalityInfo) || false;
115 }
116
117 function isContainment(relationship) {
118     const containmentWords = [
119         'have',
120         'include',
121         'incorporate',
122         'consist',
123         'comprise',
124         'contain',
125     ];
126
127     return containmentWords.find(w => w == relationship.lemma);
128 }
129
130 function buildPhrase(tree, transform = w => w, space = '_') {
131     const othersInPhrase = tree.othersInPhrase;
132
133     if (othersInPhrase.length) {
134         return [tree, ...othersInPhrase].sort((a, b) => a.start - b.start).map(o => o.word).map(transform).join(space);
135     }
136     return tree.word;
137 }
138
139 function propertyName(prop, relationship, multiple) {
140     let entity = '';
141
142     const correctedNoun = multiple ?
143         compromise.noun(prop.lemma).pluralize() :
144         compromise.noun(prop.lemma).singularize();
145
146
147     const othersInPhrase = prop.othersInPhrase;
148
149     entity = buildPhrase(prop);
150
151     if (isContainment(relationship)) {
152         return entity;
153     }
154
155     const presentVerb = compromise.verb(relationship.word).to_present();
156

```

```

157     return `${presentVerb}_${entity}`;
158 }
159
160 const capitalizeWord = str => str.charAt(0).toUpperCase() + str.slice(1);
161
162 function propertyType(prop, entities = []) {
163     for (const entity of entities) {
164         if (entity.raw === prop.raw ||
165             entity.lemma === prop.lemma) {
166             return capitalizeWord(entity.lemma);
167         }
168     }
169
170     // Check criteria for date.
171     const dateKeywords = [
172         'date',
173         'day', // TODO Add more keywords
174     ];
175
176     // if prop.raw.toLowerCase.includes()
177
178     // Check criteria for number.
179     const numberKeywords = [
180         'number',
181         'integer',
182         'float',
183         'double',
184     ];
185
186     // Check for integer or float
187
188     // TODO
189
190     return 'string';
191 }
192
193 function categoriseProp(prop, context, relationship, entities) {
194     const multiple = findIfPropertyHasMultiple(prop);
195
196     const type = propertyType(prop, entities);
197     const name = propertyName(prop, relationship, multiple);
198     const required = findIfPropertyIsRequired(prop, context);
199
200     return {
201         type,
202         name,
203         raw: prop.word,
204         lemma: prop.lemma,
205         required,
206         multiple,
207     };
208 }
209

```

```

210 function getConjunctions(object) {
211   return followModifiers(object, o => o.arc === 'conj');
212 }
213
214 function followModifiers(tree, condition) {
215   if (!tree || !tree.modifiers || tree.modifiers.length === 0) return [];
216
217   const [modifier] = tree.modifiers.filter(condition);
218   const deeperConjunctions = followModifiers(modifier, condition);
219
220   if (deeperConjunctions.length) {
221     return [
222       modifier,
223       ...deeperConjunctions,
224     ];
225   }
226   if (modifier) {
227     return [modifier];
228   }
229   return [];
230 }
231
232 function postprocess(modelStructure, entities) {
233   for (const models of modelStructure) {
234     for (const prop of models.attributes) {
235       prop.type = propertyType(prop, entities);
236     }
237   }
238 }
239
240 function flatMap(array, lambda) {
241   if (!array) return [];
242   return Array.prototype.concat.apply([], array.map(lambda));
243 }
244
245 function flatten(array) {
246   if (!array) return [];
247   return Array.prototype.concat.apply([], array);
248 }
249
250 function filterTree(tree, condition, depth = 0) {
251   if (!tree) return;
252   if (depth === 0) tree = JSON.parse(JSON.stringify(tree)); // Clone the tree
253
254   const modifiers = flatMap(
255     tree.modifiers,
256     m => filterTree(m, e => condition(e, depth, tree), depth + 1),
257   );
258
259   if (condition(tree)) {
260     if (modifiers.length < 1) {
261       // delete tree.modifiers;
262       return Object.assign(tree, {

```



```

263     modifiers: undefined,
264   });
265 }
266 return Object.assign(tree, {
267   modifiers,
268 });
269 }
270 return modifiers;
271 }
272
273 function assignNounPhrase(p) {
274   const preps = findAll(p, o => o.arc === 'prep');
275   const prepPhrases = preps.map(
276     o => [o, ...(o.modifiers.filter(m => m.arc === 'pobj'))],
277   );
278
279   const tags = ['compound', 'amod'];
280   const more = findAll(p, m => tags.includes(m.arc));
281
282   p.othersInPhrase = [...flatten(prepPhrases), ...more].sort((a, b) => a.start - b.start);
283
284   return p;
285 }
286
287 async function generateModelStructure(text) {
288   // Annotate raw text with POS and get dependency structure
289   const parseResult = await parse(text);
290   const modelStructure = [];
291   let allEntities = [];
292
293   // Useful transformations
294   // Remove oxford comma!
295
296   for (const sentenceResult of parseResult.data) {
297     // Find potential entities
298     // const potentialEntities = sentenceResult.parse_list
299     //   .filter(word => word.POS_coarse === 'NOUN');
300
301     // Find relationships
302     const potentialRelationships = sentenceResult.parse_list
303       .filter(word => word.POS_fine.startsWith('V'));
304
305     // Build up tree of words to their place in parse tree
306     const tokens = sentenceResult.parse_list;
307     const cleanTree = filterTree(sentenceResult.parse_tree[0], m => m.POS_fine.startsWith('V') || m.POS_fine.startsWith('N') || m.POS_fine
308       === 'PRP');
309
310     const treeIndex = {};
311     const cleanTreeIndex = {};
312     tokens.forEach((token) => {
313       treeIndex[token.id] = find(sentenceResult.parse_tree[0], obj => obj.id === token.id);
314       cleanTreeIndex[token.id] = find(cleanTree, obj => obj.id === token.id);
315     });

```

```

315
316
317 // console.log(cleanTree, cleanTreeIndex)
318
319 for (const relationship of potentialRelationships) {
320   // First containment
321   let inTree = cleanTreeIndex[relationship.id];
322
323   const nounTree = filterTree(inTree, m => m.POS_fine.startsWith('N') || m.POS_fine === 'PRP');
324   const compareDepth = (a, b) => a.depth - b.depth;
325
326   if (!nounTree || nounTree.length < 1) continue;
327   // Find subject and object
328   // console.log('\n\n\nOK ', inTree, nounTree);
329   const [subject] = nounTree.filter(o => o.arc.includes('subj')).sort(compareDepth);
330   const [object] = nounTree.filter(o => o.arc.includes('obj')).sort(compareDepth);
331
332   let attributes = [];
333   if (object) {
334     // This is the attributes
335     const fullObject = treeIndex[object.id];
336     attributes = [fullObject, ...getConjunctions(fullObject)];
337
338     attributes = attributes.map(assignNounPhrase);
339   }
340   let entities = [];
341   if (subject) {
342     // This is entities
343     const fullSubject = treeIndex[subject.id];
344     entities = [fullSubject, ...getConjunctions(fullSubject)];
345     allEntities = [...allEntities, ...entities];
346     allEntities = allEntities.map(assignNounPhrase);
347   }
348
349
350   inTree = treeIndex[relationship.id];
351
352   const attributesWithTypes = [];
353   for (const property of attributes) {
354     attributesWithTypes.push(categoriseProp(property, inTree, relationship, entities));
355   }
356
357   for (const entity of entities) {
358     const existingEntity = modelStructure.find(s => s.name === entity.lemma);
359
360     if (existingEntity) {
361       existingEntity.attributes = existingEntity.attributes.concat(attributesWithTypes);
362     } else {
363       modelStructure.push({
364         name: entity.lemma, // buildPhrase(entity, w => capitalizeWord(w), ' '),
365         raw: entity.word,
366         attributes: attributesWithTypes,
367       });

```

```
368     }
369   }
370 }
371 }
372
373   postprocess(modelStructure, allEntities);
374   return modelStructure;
375 }
376
377
378 const Natural = {
379   _find: find,
380   _findAll: findAll,
381   _findIfPropertyIsRequired: findIfPropertyIsRequired,
382   _findIfPropertyHasMultiple: findIfPropertyHasMultiple,
383   _filterTree: filterTree,
384   seperateSentences,
385   generateModelStructure,
386   parse,
387 };
388
389 export default Natural;
```

1.140 parse.js

```
1 import XLSX from 'xlsx';
2 import Natural from '../components/natural';
3 import { object } from 'underscore';
4
5 export function parseSpreadsheet(file) {
6   // Assume spreadsheet is array of csv's
7   const workbook = XLSX.readFile(file.path);
8
9   const sheetNames = workbook.SheetNames;
10
11   const csvs = sheetNames
12     .map(name => workbook.Sheets[name])
13     .map(sheet => XLSX.utils.sheet_to_csv(sheet));
14
15   const sheetByName = object(sheetNames, csvs);
16
17   const allModelDefinitions = [];
18
19   for (const name of sheetNames) {
20     const csv = sheetByName[name];
21     const modelDefinition = {};
22     const [headingLine, ...rowLines] = csv.split('\n');
23     const headings = headingLine.split(',');
24     const rows = rowLines
25       .map(r => r.split(','))
26       .filter(r => r.join('').trim().length > 0);
27
28     console.log(rows);
29
30     modelDefinition.name = name;
31
32     const attributes = [];
33     const entries = [];
34
35     for (let i = 0; i < headings.length; i++) {
36       const headingName = headings[i].toLowerCase();
37
38       // Get first 20 rows for sample data
39       const types = determineType(new Set(rows.slice(0, 20).map(row => findType(row[i]))));
40       attributes.push(Object.assign({ name: headingName }, types));
41     }
42
43     rows.forEach((row) => {
44       const entry = {};
45       attributes.forEach((attribute, i) => {
46         entry[attribute.name] = row[i];
47       });
48       entries.push(entry);
49     });
50
51   }
```

```

51     modelDefinition.entries = entries;
52     modelDefinition.attributes = attributes;
53     allModelDefinitions.push(modelDefinition);
54 }
55 return Promise.resolve(allModelDefinitions);
56 }
57
58 // Given a array of type information, determines the type which encompasses all values
59 export function determineType(information) {
60     let type;
61     let multiple = false;
62     let required = true;
63
64     for (const value of information) {
65         if (value === null || value === undefined) {
66             required = false;
67             continue;
68         }
69
70         if (value.type === 'string') {
71             type = 'string';
72         } else if (value.type === 'float') {
73             if (type !== 'string') {
74                 type = 'float';
75             }
76         } else if (value.type === 'integer') {
77             if (type !== 'float' || type !== 'string') {
78                 type = 'integer';
79             }
80         }
81
82         if (value.multiple == true) {
83             multiple = true;
84         }
85     }
86
87     return {
88         type,
89         multiple,
90         required,
91     };
92 }
93
94 // Given a string, finds the most likely type
95 export function findType(raw) {
96     // If there is no value assume null
97     if ((raw === null) || (raw === undefined)) return null;
98
99     const string = raw.trim();
100     if (string.length === 0) return null;
101
102     const object = safeJSONParse(string);
103     const multiple = Array.isArray(object);

```

```
104     let type;
105
106     if (multiple) {
107         type = 'string';
108         type = determineType(object.map(findType)).type;
109         console.log(type);
110         if (type.multiple) {
111             throw new Error('Multidimensional arrays are not supported!');
112         }
113     } else {
114         // Check for floats
115         if (/^-?((\d+\.\d*)|(\d+\.\d*))$/.test(string)) {
116             type = 'float';
117         } else if (/^-?(\d+)$/.test(string)) {
118             type = 'integer';
119         } else {
120             type = 'string';
121         }
122     }
123
124     console.log(string, type);
125     return {
126         type,
127         multiple,
128         example: string,
129     };
130 }
131
132 function safeJSONParse(string) {
133     try {
134         return JSON.parse(string);
135     } catch (e) {
136         return null;
137     }
138 }
139
140 export function parseNaturalLanguage(text) {
141     return Natural.generateModelStructure(text);
142 }
```

1.141 service.js

```
1 import databaseModels from '../models';
2 import { stringToShortName } from './utils';
3
4 const { Service, Model, Attribute, Entry, Value } = databaseModels;
5
6 /* Model definition format
7
8 {
9   name: string,
10  modelDefinitions: [
11    {
12      name: string,
13      attributes: [
14        {
15          name: string,
16          type: string,
17          required: boolean,
18          multiple: boolean,
19        }
20      ],
21      entries: [
22        {
23          [key]: value,
24        }
25      ]
26    }
27  ]
28 }
29 */
30
31 export async function createService(name, modelDefinitions, userId) {
32   let service = await Service.create({
33     name,
34     isPublic: false,
35     shortName: stringToShortName(name),
36     UserId: userId,
37   });
38
39   service = service.toJSON();
40
41   await Model.bulkCreate(modelDefinitions.map(def => ({
42     name: def.name,
43     ServiceId: service.id,
44     shortName: stringToShortName(def.name),
45   })));
46
47   let models = await Model.findAll({
48     where: {
49       ServiceId: service.id,
50     },
```

```

51 });
52
53 const attributesToCreate = [];
54 const entriesToCreate = [];
55 const entryByIndexByModel = {};
56
57 let i = 0;
58 for (const modelDefinition of modelDefinitions) {
59   const model = models[i];
60   i++;
61   // Create attributes
62   for (const attributeDefinition of modelDefinition.attributes) {
63     attributesToCreate.push({
64       name: attributeDefinition.name,
65       type: attributeDefinition.type,
66       required: attributeDefinition.required,
67       multiple: attributeDefinition.multiple,
68       ModelId: model.id,
69     });
70   }
71
72   if (!modelDefinition.entries || modelDefinition.entries.length === 0) {
73     continue;
74   }
75
76   const entryByIndex = {};
77   // Create entries
78   let index = 1;
79   for (const entriesDefinition of modelDefinition.entries) {
80     entriesToCreate.push({
81       index,
82       ModelId: model.id,
83     });
84     index++;
85     entryByIndex[index] = entriesDefinition;
86   }
87   entryByIndexByModel[modelDefinition.name] = entryByIndex;
88 }
89
90 await Attribute.bulkCreate(attributesToCreate);
91 await Entry.bulkCreate(entriesToCreate);
92
93 models = await Model.findAll({
94   where: {
95     ServiceId: service.id,
96   },
97   include: [{ all: true, nested: true }],
98 });
99
100 const valuesToCreate = [];
101
102 // Index: model > entry > attribute > value
103

```



```
104     console.log(entryByIndexByModel);
105
106     for (const model of models) {
107         for (const entry of model.Entries) {
108             for (const attribute of model.Attributes) {
109                 console.log(model.name, entry.index, attribute.name);
110                 const entryDefinition = entryByIndexByModel[model.name][entry.index];
111                 valuesToCreate.push({
112                     AttributeId: attribute.id,
113                     EntryId: entry.id,
114                     value: entryDefinition && entryDefinition[attribute.name],
115                 });
116             }
117         }
118     }
119
120     await Value.bulkCreate(valuesToCreate);
121
122     service = await Service.findOne({
123         where: {
124             id: service.id,
125         },
126         include: [{ all: true, nested: true }],
127     });
128
129     return service;
130 }
```

1.142 utils.js

```
1
2
3 export function stringToShortName(string) {
4   return string.toLowerCase().replace(/\W/g, '');
5 }
6
7 export function encode(value, type) {
8   return `${value}`;
9 }
10
11 export function decode(string, type) {
12   switch (type) {
13     case 'integer':
14       return parseInt(string, 10);
15     case 'float':
16       return parseFloat(string);
17     default:
18       return string;
19   }
20 }
```

1.143 bootstrap.js

```
1  /**
2   * Bootstrap: All scripts that should be executed before server starts running
3   */
4
5  export default function bootstrap() {
6    return Promise.resolve();
7  }
```

1.144 connections.js

```
1  const connections = {
2    development: {
3      username: 'martinkubat',
4      password: '',
5      database: 'martinkubat',
6      host: 'localhost',
7      dialect: 'postgres',
8    },
9    test: {
10     username: 'root',
11     password: null,
12     database: 'database_test',
13     host: '127.0.0.1',
14     dialect: 'mysql',
15   },
16   production: {
17     username: 'root',
18     password: null,
19     database: 'database_production',
20     host: '127.0.0.1',
21     dialect: 'mysql',
22   },
23 };
24
25 export default connections;
```

1.145 passport.js

```
1 import passport from 'passport';
2 import { Strategy as LocalStrategy } from 'passport-local';
3 import models from '../models';
4 import jwt from 'jsonwebtoken';
5
6 const { User } = models;
7
8 passport.use(new LocalStrategy({
9   usernameField: 'username',
10  passwordField: 'password',
11  session: false,
12  passReqToCallback: true,
13 }, (req, username, password, done) => User.findOne({
14   where: {
15     username,
16   },
17 })))
18   .then(async (foundUser) => {
19     let user;
20     if (foundUser) {
21       // User exists
22       if (!(await foundUser.validatePassword(password))) {
23         console.log('Invalid password');
24         return done(null, false, {
25           message: 'Incorrect password.',
26         });
27       }
28       user = foundUser;
29     } else {
30       // New user
31       user = await User.create({
32         username,
33         passwordHash: User.generateHash(password),
34       });
35     }
36
37     const payload = {
38       user: user.id,
39     };
40
41     const token = jwt.sign(payload, 'secret');
42
43     return done(null, {
44       user: {
45         username: user.username,
46       },
47       token,
48     });
49   })
50   .catch(err => done(err)),
```

```
51  });
52
53  passport.serializeUser((user, done) => {
54    done(null, user.id);
55  });
56
57  passport.deserializeUser((id, done) => {
58    User.find({
59      where: { id },
60    }, (err, [user]) => {
61      done(err, user);
62    });
63  });
64
65  export default passport;
```

1.146 index.js

```
1  import Express from 'express';
2  import bodyParser from 'body-parser';
3  import passport from './config/passport';
4  import index from './routes/index';
5  import auth from './routes/auth';
6  import service from './routes/service';
7  import model from './routes/model';
8  import entry from './routes/entry';
9  import attribute from './routes/attribute';
10 import value from './routes/value';
11 import api from './routes/api';
12 import bootstrap from './config/bootstrap';
13 import models from './models';
14 import authentication from './middleware/authentication';
15
16
17 bootstrap().then(async () => {
18   /* eslint-disable new-cap */
19   const app = Express();
20   const port = 9001;
21
22   await models.sequelize.sync();
23
24   app.use(bodyParser.json());
25
26   app.use(passport.initialize());
27   app.use(authentication);
28
29   app.use('/api', index);
30   app.use('/api/service', service);
31   app.use('/api/auth', auth);
32   app.use('/api/model', model);
33   app.use('/api/attribute', attribute);
34   app.use('/api/entry', entry);
35   app.use('/api/value', value);
36   app.use('/api/api/', api);
37
38   // catch 404 and forward to error handler
39   app.use((req, res, next) => {
40     const err = new Error('Not Found!');
41     err.status = 404;
42     next(err);
43   });
44
45   // error handler
46   app.use((err, req, res) => {
47     // set locals, only providing error in development
48     /* eslint-disable no-param-reassign */
49     res.locals.message = err.message;
50     res.locals.error = req.app.get('env') === 'development' ? err : {};
```

```
51
52     // render the error page
53     res.status(err.status || 500);
54     res.render('error');
55   });
56
57   app.listen(port);
58   console.log('Server is running on port ${port}');
59 }
60 .catch(err => console.error(err));
```


1.147 authentication.js

```
1  import jwt from 'jsonwebtoken';
2  import models from '../models';
3
4  const { User } = models;
5
6  export default function(req, res, next) {
7    if (req.originalUrl.startsWith('/api/auth/')) {
8      return next();
9    }
10
11    if (!req.headers.authorization) {
12      return res.status(401).end();
13    }
14
15    const token = req.headers.authorization.split(' ')[1];
16    return jwt.verify(token, 'secret', (err, decoded) => {
17      if (err) return res.status(401).end();
18
19      const userId = decoded.user;
20
21      return User.findById(userId)
22        .then(user => {
23          if (user) {
24            req.user = user;
25            return next();
26          }
27          return res.status(401).end();
28        })
29        .catch(err => res.status(401).end());
30    });
31  };
```

1.148 attribute.js

```
1 export default function (sequelize, DataTypes) {
2   const Attribute = sequelize.define('Attribute', {
3     name: DataTypes.STRING,
4     type: DataTypes.STRING,
5     multiple: DataTypes.BOOLEAN,
6     required: DataTypes.BOOLEAN,
7   }, {
8     classMethods: {
9       associate(models) {
10         Attribute.belongsTo(models.Model, {
11           onDelete: 'CASCADE',
12           foreignKey: {
13             allowNull: false,
14           },
15         });
16       },
17     },
18   });
19   return Attribute;
20 }
21 }
```

1.149 entry.js

```
1 export default function (sequelize, DataTypes) {
2   const Entry = sequelize.define('Entry', {
3     index: DataTypes.INTEGER,
4   }, {
5     classMethods: {
6       associate(models) {
7         Entry.belongsTo(models.Model, {
8           onDelete: 'CASCADE',
9           foreignKey: {
10             allowNull: false,
11           },
12         });
13         Entry.hasMany(models.Value);
14       },
15     },
16   });
17   return Entry;
18 }
19 }
```

1.150 index.js

```
1  import fs from 'fs';
2  import path from 'path';
3  import Sequelize from 'sequelize';
4  import connections from '../config/connections';
5
6  const basename = path.basename(__filename);
7  const env = process.env.NODE_ENV || 'development';
8  const db = {};
9
10 const config = connections[env];
11
12 let sequelize;
13 if (config.use_env_variable) {
14   sequelize = new Sequelize(process.env[config.use_env_variable]);
15 } else {
16   sequelize = new Sequelize(config.database, config.username, config.password, config);
17 }
18
19 import attribute from './attribute';
20 import entry from './entry';
21 import model from './model';
22 import service from './service';
23 import user from './user';
24 import value from './value';
25
26 const models = {
27   attribute,
28   entry,
29   model,
30   service,
31   user,
32   value,
33 };
34
35 const capitalizeString = str => str.charAt(0).toUpperCase() + str.slice(1);
36
37 for (const modelName in models) {
38   if (!models.hasOwnProperty(modelName)) continue;
39
40   db[capitalizeString(modelName)] = models[modelName](sequelize, Sequelize);
41 }
42
43 console.log(db);
44
45 Object.keys(db).forEach((modelName) => {
46   if (db[modelName].associate) {
47     db[modelName].associate(db);
48   }
49 });
50
```

```
51 db.sequelize = sequelize;
52 db.Sequelize = Sequelize;
53
54 export default db;
```

1.151 model.js

```
1  export default function (sequelize, DataTypes) {
2    const Model = sequelize.define('Model', {
3      name: DataTypes.STRING,
4      shortName: DataTypes.STRING,
5      isFindEnabled: {
6        type: DataTypes.BOOLEAN,
7        defaultValue: false,
8      },
9      isFindOneEnabled: {
10       type: DataTypes.BOOLEAN,
11       defaultValue: false,
12     },
13     isCreateEnabled: {
14       type: DataTypes.BOOLEAN,
15       defaultValue: false,
16     },
17     isUpdateEnabled: {
18       type: DataTypes.BOOLEAN,
19       defaultValue: false,
20     },
21     isDeleteEnabled: {
22       type: DataTypes.BOOLEAN,
23       defaultValue: false,
24     },
25   }, {
26     classMethods: {
27       associate(models) {
28         Model.belongsTo(models.Service, {
29           onDelete: 'CASCADE',
30           foreignKey: {
31             allowNull: false,
32           },
33         });
34         Model.hasMany(models.Attribute);
35         Model.hasMany(models.Entry);
36       },
37     },
38   });
39
40   return Model;
41 }
```

1.152 service.js

```
1 export default function (sequelize, DataTypes) {
2   const Service = sequelize.define('Service', {
3     name: DataTypes.STRING,
4     shortName: DataTypes.STRING,
5     isPublic: DataTypes.BOOLEAN,
6   }, {
7     classMethods: {
8       associate(models) {
9         Service.belongsTo(models.User, {
10           onDelete: 'CASCADE',
11           foreignKey: {
12             allowNull: false,
13           },
14         });
15         Service.hasMany(models.Model);
16       },
17     },
18   });
19   return Service;
20 }
21 }
```

1.153 user.js

```
1  import bcrypt from 'bcrypt';
2
3  export default function (sequelize, DataTypes) {
4    const User = sequelize.define('User', {
5      username: {
6        type: DataTypes.STRING,
7        unique: true,
8      },
9      passwordHash: DataTypes.STRING,
10   }, {
11     classMethods: {
12       associate(models) {
13         User.hasMany(models.Service);
14       },
15       generateHash: password => bcrypt.hashSync(password, bcrypt.genSaltSync(8), null),
16     },
17     instanceMethods: {
18       generateHash: password => bcrypt.hashSync(password, bcrypt.genSaltSync(8), null),
19       validPassword(password) {
20         console.log(password, this.passwordHash);
21         return bcrypt.compare(password, this.passwordHash);
22       },
23       toJSON() {
24         const response = this.get();
25         response.passwordHash = undefined;
26         return response;
27       },
28     },
29   });
30
31   return User;
32 }
```


1.154 value.js

```
1 export default function (sequelize, DataTypes) {
2   const Value = sequelize.define('Value', {
3     value: DataTypes.STRING,
4   }, {
5     classMethods: {
6       associate(models) {
7         Value.belongsTo(models.Entry, {
8           onDelete: 'CASCADE',
9           foreignKey: {
10             allowNull: false,
11           },
12         });
13         Value.belongsTo(models.Attribute);
14       },
15     },
16   });
17
18   return Value;
19 }
```

1.155 index.py

```
1 from flask import Flask, request, jsonify
2 app = Flask(__name__)
3 from spacyparse import parse
4
5 @app.route("/")
6 def index():
7     return "Hello World!"
8
9 @app.route("/parse", methods=['POST'])
10 def dependency():
11     text = request.form.get('text')
12
13     print(text)
14     result = parse(text)
15
16     return jsonify(data=result)
17
18 if __name__ == "__main__":
19     app.run()
```

1.156 spacyparse.py

```
1 # Credit: https://github.com/kengz/spacy-nlp/blob/master/src/py/nlp.py
2
3 # MIT License
4 #
5 # Copyright (c) 2016 Wah Loon Keng
6 #
7 # Permission is hereby granted, free of charge, to any person obtaining a copy
8 # of this software and associated documentation files (the "Software"), to deal
9 # in the Software without restriction, including without limitation the rights
10 # to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
11 # copies of the Software, and to permit persons to whom the Software is
12 # furnished to do so, subject to the following conditions:
13 #
14 # The above copyright notice and this permission notice shall be included in all
15 # copies or substantial portions of the Software.
16 #
17 # THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
18 # IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
19 # FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
20 # AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
21 # LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
22 # OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
23 # SOFTWARE.
24
25 from collections import OrderedDict
26 from spacy.en import English
27 nlp = English()
28
29 # Helper methods
30 #####
31
32 def merge_ents(doc):
33     '''Helper: merge adjacent entities into single tokens; modifies the doc.'''
34     for ent in doc.ents:
35         ent.merge(ent.root.tag_, ent.text, ent.label_)
36     return doc
37
38
39 def format_POS(token, light=False, flat=False, depth=0):
40     '''helper: form the POS output for a token'''
41     subtree = OrderedDict([
42         ("word", token.text),
43         ("lemma", token.lemma_), # trigger
44         ("NE", token.ent_type_), # trigger
45         ("POS_fine", token.tag_),
46         ("POS_coarse", token.pos_),
47         ("arc", token.dep_),
48         ("id", token.i),
49         ("start", token.idx),
50         ("depth", depth),
```

```

51         ("modifiers", [])
52     ])
53     if light:
54         subtree.pop("lemma")
55         subtree.pop("NE")
56     if flat:
57         subtree.pop("arc")
58         subtree.pop("modifiers")
59     return subtree
60
61
62 def POS_tree_(root, light=False, depth=0):
63     '''
64     Helper: generate a POS tree for a root token.
65     The doc must have merge_ents(doc) ran on it.
66     '''
67     subtree = format_POS(root, light=light, depth=depth)
68     for c in root.children:
69         subtree["modifiers"].append(POS_tree_(c, light=False, depth=depth+1))
70     return subtree
71
72
73 def parse_tree(doc, light=False):
74     '''generate the POS tree for all sentences in a doc'''
75     merge_ents(doc) # merge the entities into single tokens first
76     return [POS_tree_(sent.root, light=light) for sent in doc.sents]
77
78
79 def parse_list(doc, light=False):
80     '''tag the doc first by NER (merged as tokens) then
81     POS. Can be seen as the flat version of parse_tree'''
82     merge_ents(doc) # merge the entities into single tokens first
83     return [format_POS(token, light=light, flat=True) for token in doc]
84
85 # s = "find me flights from New York to London next month"
86 # doc = nlp(s)
87 # parse_list(doc)
88
89
90 # Primary methods
91 #####
92
93 def parse_sentence(sentence):
94     '''
95     Main method: parse an input sentence and return the nlp properties.
96     '''
97
98     doc = nlp(sentence)
99     reply = OrderedDict([
100         ("text", doc.text),
101         ("len", len(doc)),
102         ("tokens", [token.text for token in doc]),
103         ("noun_phrases", [token.text for token in doc.noun_chunks]),

```

```
104         ("parse_tree", parse_tree(doc)),
105         ("parse_list", parse_list(doc))
106     ])
107     return reply
108
109 # res = parse_sentence("find me flights from New York to London next month.")
110
111
112 def parse(input):
113     '''
114     parse for multi-sentences; split and apply parse in a list.
115     '''
116     return [parse_sentence(sent) for sent in input.split("<#SENT_SEPERATOR#>")]
```

1.157 api.js

```
1  import { Router } from 'express';
2  import { object } from 'underscore';
3  import databaseModels from '../models';
4  import { decode } from '../components/utils';
5
6  const { Service, Model, Attribute, Entry, Value, User } = databaseModels;
7
8
9  /* eslint-disable new-cap */
10 const router = Router();
11
12 router.all('/:user/:service/:model/:id?', async (req, res) => {
13   const username = req.param('user');
14   const serviceShortName = req.param('service');
15   const modelShortName = req.param('model');
16   const id = req.param('id');
17   const method = req.method;
18   const input = req.body;
19
20   let data;
21
22   try {
23     const user = await User.findOne({
24       where: {
25         username,
26       },
27     });
28
29     const service = await Service.findOne({
30       where: {
31         shortName: serviceShortName,
32         UserId: user.id,
33       },
34     });
35
36     if (!service.isPublic) {
37       return res.status(403).send({ success: false });
38     }
39
40     const model = await Model.findOne({
41       where: {
42         shortName: modelShortName,
43       },
44     });
45
46     const attributes = await Attribute.findAll({
47       where: {
48         ModelId: model.id,
49       },
50     });
```

```

51
52 data = { user, service, model };
53
54 switch (method) {
55   case 'GET': {
56     if (id) {
57       // Find One
58       if (!model.isFindOneEnabled) {
59         return res.status(403).send({ success: false });
60       }
61
62       const entry = await Entry.findOne({
63         where: {
64           index: id,
65           ModelId: model.id,
66         },
67       });
68       const values = await Value.findAll({
69         where: {
70           EntryId: entry.id,
71         },
72       });
73
74       const valueByAttributeId = object(
75         values.map(v => v.AttributeId), values.map(v => v.value),
76       );
77       const obj = {};
78       obj.id = entry.index;
79       for (const attribute of attributes) {
80         obj[attribute.name] = decode(valueByAttributeId[attribute.id], attribute.type);
81       }
82
83       data = obj;
84     } else {
85       // Find All
86       if (!model.isFindEnabled) {
87         return res.status(403).send({ success: false });
88       }
89
90       const entries = await Entry.findAll({
91         where: {
92           ModelId: model.id,
93         },
94       });
95       const values = await Value.findAll({
96         where: {
97           EntryId: entries.map(a => a.id),
98         },
99       });
100      data = { values, attributes, entries };
101
102      const objects = [];
103      for (const entry of entries) {

```

```

104         const obj = {};
105
106         const localValues = values.filter(v => v.EntryId === entry.id);
107         const valueByAttributeId = object(
108             localValues.map(v => v.AttributeId), localValues.map(v => v.value),
109         );
110         obj.id = entry.index;
111         for (const attribute of attributes) {
112             obj[attribute.name] = decode(valueByAttributeId[attribute.id], attribute.type);
113         }
114
115         objects.push(obj);
116     }
117
118     data = objects;
119 }
120 break;
121 }
122 case 'POST': {
123     // Create
124     if (!model.isCreateEnabled) {
125         return res.status(403).send({ success: false });
126     }
127     const newestEntry = await Entry.findOne({
128         where: {
129             ModelId: model.id,
130         },
131         order: 'index DESC',
132     });
133
134     const index = (newestEntry ? newestEntry.index : 0) + 1;
135
136     const entry = await Entry.create({
137         index,
138         ModelId: model.id,
139     });
140
141     const obj = {};
142     obj.id = entry.index;
143
144     const valuePromises = [];
145     for (const attribute of attributes) {
146         valuePromises.push(
147             Value.create({
148                 EntryId: entry.id,
149                 AttributeId: attribute.id,
150                 value: input[attribute.name],
151             }),
152         );
153         obj[attribute.name] = decode(input[attribute.name], attribute.type) || null;
154     }
155     await Promise.all(valuePromises);
156     data = obj;

```



```

157     break;
158 }
159 case 'PATCH': {
160     // Update
161     if (!model.isUpdateEnabled) {
162         return res.status(403).send({ success: false });
163     }
164
165     const entry = await Entry.findOne({
166         where: {
167             index: id,
168             ModelId: model.id,
169         },
170     });
171     const values = await Value.findAll({
172         where: {
173             EntryId: entry.id,
174         },
175     });
176
177     const valuePromises = [];
178     const valueByAttributeId = object(values.map(v => v.AttributeId), values.map(v => v));
179
180
181     const obj = {};
182     obj.id = entry.id;
183     for (const attribute of attributes) {
184         const newValue = input[attribute.name];
185         if (newValue) {
186             const oldValue = valueByAttributeId[attribute.id];
187             if (newValue !== oldValue.value) {
188                 if (oldValue) {
189                     // Update
190                     valuePromises.push(
191                         Value.update(
192                             { value: newValue },
193                             { where: { id: oldValue.id } },
194                         ),
195                     );
196                 } else {
197                     // Create
198                     valuePromises.push(
199                         Value.create({
200                             EntryId: entry.id,
201                             AttributeId: attribute.id,
202                             value: newValue,
203                         }),
204                     );
205                 }
206             }
207             obj[attribute.name] = newValue;
208         } else {
209             obj[attribute.name] = valueByAttributeId[attribute.id].value;

```

```
210     }
211   }
212
213   await Promise.all(valuePromises);
214   data = obj;
215
216   break;
217 }
218 case 'DELETE': {
219   // Delete
220   if (!model.isDeleteEnabled) {
221     return res.status(403).send({ success: false });
222   }
223
224   const entry = await Entry.findOne({
225     where: {
226       index: id,
227       ModelId: model.id,
228     },
229   });
230
231   await Value.destroy({
232     where: {
233       EntryId: entry.id,
234     },
235   });
236
237   const result = await Entry.destroy({
238     where: {
239       index: id,
240       ModelId: model.id,
241     },
242   });
243   data = Boolean(result);
244
245   break;
246 }
247 default: {
248   return res.status(400).send({ success: false });
249 }
250 }
251 } catch (e) {
252   return res.status(500).send({ success: false, error: e });
253 }
254
255 res.send({ success: true, data });
256 });
257
258
259 export default router;
```

1.158 attribute.js

```
1  import { Router } from 'express';
2  import databaseModels from '../models';
3
4  const { Service, Model, Attribute, Entry, Value } = databaseModels;
5
6  /* eslint-disable new-cap */
7  const router = Router();
8
9  /* POST scratch. */
10 router.post('/', async (req, res) => {
11   const modelId = req.param('model');
12   const name = req.param('name');
13   const type = req.param('type');
14   const required = req.param('required');
15   const multiple = req.param('multiple');
16
17   try {
18     const attribute = await Attribute.create({
19       name,
20       type,
21       required,
22       multiple,
23       ModelId: modelId,
24     });
25
26     const response = {
27       attribute,
28       success: true,
29     };
30     return res.json(response);
31   } catch (e) {
32     return res.status(501).json({
33       error: e,
34       success: false,
35     });
36   }
37 });
38
39 router.patch('/:id', async (req, res) => {
40   const attributeId = req.param('id');
41
42   const toUpdate = {};
43
44   if (req.param('name')) {
45     toUpdate.name = req.param('name');
46   }
47   if (req.param('type')) {
48     toUpdate.type = req.param('type');
49   }
50 }
```

```
51   try {
52     const attribute = await Attribute.update(
53       toUpdate,
54       { where: { id: attributeId } },
55     );
56
57     return res.json({
58       attribute,
59       success: true,
60     });
61   } catch (e) {
62     return res.status(501).json({
63       error: e,
64       success: false,
65     });
66   }
67 });
68
69 router.get('/', async (req, res) => {
70   try {
71     const modelId = req.param('model');
72     const attributes = await Attribute.findAll({
73       where: {
74         ModelId: modelId,
75       },
76       include: [{ all: true }],
77     });
78     return res.json({
79       attributes,
80       success: true,
81     });
82   } catch (e) {
83     return res.status(501).json({
84       error: e,
85       success: false,
86     });
87   }
88 });
89
90 router.delete('/', async (req, res) => {
91   try {
92     const id = req.param('id');
93     const result = await Attribute.destroy({
94       where: {
95         id,
96       },
97     });
98     return res.json({
99       result,
100       success: true,
101     });
102   } catch (e) {
103     return res.status(501).json({
```

```
104         error: e,  
105         success: false,  
106     });  
107 }  
108 });  
109  
110 export default router;
```

1.159 auth.js

```
1 import { Router } from 'express';
2 import passport from '../config/passport';
3
4 /* eslint-disable new-cap */
5 const router = Router();
6
7 function validate(form) {
8   const errors = {};
9   let success = true;
10
11   if (!form || !form.username || form.username.length < 5) {
12     success = false;
13     errors.username = 'This is not a valid username.';
14   }
15
16   if (!form || !form.password || form.password.length < 5) {
17     success = false;
18     errors.password = 'This password is too short.';
19   }
20
21   return {
22     success,
23     errors,
24   };
25 }
26
27 /* GET index. */
28 router.post('/login', (req, res, next) => {
29   const validation = validate({
30     username: req.param('username'),
31     password: req.param('password'),
32   });
33
34   if (!validation.success) {
35     return res.status(400).json({
36       success: false,
37       errors: validation.errors,
38     });
39   }
40
41   return passport.authenticate('local', (err, user) => {
42     console.log(err, user);
43     if (err || !user) {
44       return res.status(400).json({
45         success: false,
46         message: 'Incorrect details',
47       });
48     }
49
50     return res.status(200).json(Object.assign({
```

```
51         success: true,
52         errors: {},
53         }, user));
54     })(req, res, next);
55 });
56
57
58 export default router;
```

1.160 entry.js

```
1 import { Router } from 'express';
2 import databaseModels from '../models';
3
4 const { Service, Model, Attribute, Entry, Value } = databaseModels;
5
6 /* eslint-disable new-cap */
7 const router = Router();
8
9 /* POST scratch. */
10 router.post('/', async (req, res) => {
11   const modelId = req.param('model');
12
13   try {
14     const newestEntry = await Entry.findOne({
15       where: {
16         ModelId: modelId,
17       },
18       order: 'index DESC',
19     });
20
21     const index = (newestEntry ? newestEntry.index : 0) + 1;
22
23     const attributes = await Attribute.findAll({
24       where: {
25         ModelId: modelId,
26       },
27     });
28
29     let entry = await Entry.create({
30       index,
31       ModelId: modelId,
32     });
33
34     const valuePromises = [];
35     for (const attribute of attributes) {
36       valuePromises.push(
37         Value.create({
38           EntryId: entry.id,
39           AttributeId: attribute.id,
40           value: '',
41         }),
42       );
43     }
44     await Promise.all(valuePromises);
45
46     entry = await Entry.findOne({
47       where: {
48         id: entry.id,
49       },
50       include: [{ all: true }],
```



```

51     });
52
53     const response = {
54         entry,
55         success: true,
56     };
57     return res.json(response);
58 } catch (e) {
59     return res.status(501).json({
60         error: e,
61         success: false,
62     });
63 }
64 });
65
66 router.get('/', async (req, res) => {
67     try {
68         const modelId = req.param('model');
69         const entries = await Entry.findAll({
70             where: {
71                 ModelId: modelId,
72             },
73             include: [{ all: true }],
74         });
75         return res.json({
76             entries,
77             success: true,
78         });
79     } catch (e) {
80         return res.status(501).json({
81             error: e,
82             success: false,
83         });
84     }
85 });
86
87 router.delete('/', async (req, res) => {
88     try {
89         const id = req.param('id');
90
91         await Value.destroy({
92             where: {
93                 EntryId: id,
94             },
95         });
96
97         await Entry.destroy({
98             where: {
99                 id,
100             },
101         });
102
103         return res.json({

```

```
104         success: true,
105     });
106     } catch (e) {
107         return res.status(501).json({
108             error: e,
109             success: false,
110         });
111     }
112 });
113
114 export default router;
```

1.161 index.js

```
1 import { Router } from 'express';
2 import { User, Service } from '../models';
3
4 /* eslint-disable new-cap */
5 const router = Router();
6
7 router.get('/models', (req, res) => {
8   User.findAll({
9     include: [Service],
10   }).then((users) => {
11     res.send(users);
12   });
13 });
14
15 export default router;
```

1.162 model.js

```
1  import { Router } from 'express';
2  import databaseModels from '../models';
3  import { stringToShortName } from '../components/utils';
4
5  const { Service, Model, Attribute, Entry, Value } = databaseModels;
6
7  /* eslint-disable new-cap */
8  const router = Router();
9
10 /* POST scratch. */
11 router.post('/', async (req, res) => {
12   const serviceId = req.param('service');
13   const name = req.param('name');
14
15   try {
16     const model = await Model.create({
17       name,
18       shortName: stringToShortName(name),
19       ServiceId: serviceId,
20     });
21
22     const response = {
23       model,
24       success: true,
25     };
26     return res.json(response);
27   } catch (e) {
28     return res.status(501).json({
29       error: e,
30       success: false,
31     });
32   }
33 });
34
35 router.patch('/:id', async (req, res) => {
36   const newName = req.param('name');
37   const modelId = req.param('id');
38
39   try {
40     const model = await Model.update(
41       {
42         name: newName,
43         shortName: stringToShortName(newName),
44       },
45       { where: { id: modelId } },
46     );
47
48     return res.json({
49       model,
50       success: true,
```

```
51     });
52   } catch (e) {
53     return res.status(501).json({
54       error: e,
55       success: false,
56     });
57   }
58 });
59
60 router.get('/', async (req, res) => {
61   try {
62     const serviceId = req.param('service');
63     const model = await Model.findAll({
64       where: {
65         ServiceId: serviceId,
66       },
67       include: [{ all: true }],
68     });
69     return res.json({
70       model,
71       success: true,
72     });
73   } catch (e) {
74     return res.status(501).json({
75       error: e,
76       success: false,
77     });
78   }
79 });
80
81 router.delete('/', async (req, res) => {
82   try {
83     const id = req.param('id');
84     const result = await Model.destroy({
85       where: {
86         id,
87       },
88     });
89     return res.json({
90       result,
91       success: true,
92     });
93   } catch (e) {
94     return res.status(501).json({
95       error: e,
96       success: false,
97     });
98   }
99 });
100
101
102 export default router;
```

1.163 service.js

```
1 import { Router } from 'express';
2 import multer from 'multer';
3 import { parseSpreadsheet, parseNaturalLanguage } from '../components/parse';
4 import { createService, findServices } from '../components/service';
5 import databaseModels from '../models';
6
7 const { Service, Model, Attribute, Entry, Value } = databaseModels;
8
9 const upload = multer({ dest: 'upload/' });
10
11 /* eslint-disable new-cap */
12 const router = Router();
13
14 router.post('/parseText', (req, res) => {
15   const text = req.param('text');
16   return parseNaturalLanguage(text)
17     .then(result => res.send(result));
18 });
19
20 router.post('/parseSpreadsheet', upload.single('spreadsheet'), (req, res) => {
21   console.log(req.file);
22   return parseSpreadsheet(req.file)
23     .then(result => res.send(result));
24 });
25
26 /* POST scratch. */
27 router.post('/', async (req, res) => {
28   const name = req.param('name');
29   const modelDefinitions = req.param('models');
30
31   try {
32     const service = await createService(
33       name,
34       modelDefinitions,
35       req.user.id,
36     );
37
38     const response = {
39       service,
40       success: true,
41     };
42     return res.json(response);
43   } catch (e) {
44     console.error(e);
45     return res.status(501).json({
46       error: e,
47       success: false,
48     });
49   }
50 });
```

```
51
52 router.get('/', async (req, res) => {
53   try {
54     const services = await Service.findAll({
55       where: {
56         UserId: req.user.id,
57       },
58       include: [{ all: true, nested: true }],
59     });
60     return res.json({
61       services,
62       success: true,
63     });
64   } catch (e) {
65     return res.status(501).json({
66       error: e,
67       success: false,
68     });
69   }
70 });
71
72
73 router.get('/:id', async (req, res) => {
74   try {
75     const serviceId = req.param('id');
76     const service = await Service.findOne({
77       where: {
78         id: serviceId,
79         UserId: req.user.id,
80       },
81       include: [{ all: true, nested: true }],
82     });
83     return res.json({
84       service,
85       success: true,
86     });
87   } catch (e) {
88     return res.status(501).json({
89       error: e,
90       success: false,
91     });
92   }
93 });
94
95 router.patch('/:id', async (req, res) => {
96   try {
97     const serviceId = req.param('id');
98     const toUpdate = {};
99
100     if (req.param('name')) {
101       toUpdate.name = req.param('name');
102     }
103     if (req.body.isPublic !== undefined) {
```

```
104     toUpdate.isPublic = req.body.isPublic;
105   }
106   if (req.param('shortName')) {
107     toUpdate.shortName = req.param('shortName');
108   }
109
110   const service = await Service.update(
111     toUpdate,
112     { where: { id: serviceId } },
113   );
114   return res.json({
115     service,
116     success: true,
117   });
118 } catch (e) {
119   return res.status(501).json({
120     error: e,
121     success: false,
122   });
123 }
124 });
125
126
127 export default router;
```


1.164 value.js

```
1 import { Router } from 'express';
2 import databaseModels from '../models';
3
4 const { Service, Model, Attribute, Entry, Value } = databaseModels;
5
6 /* eslint-disable new-cap */
7 const router = Router();
8
9 router.patch('/', async (req, res) => {
10   const entryId = req.param('entry');
11   const attributeId = req.param('attribute');
12   const newValue = req.param('value');
13
14   try {
15     const [foundValue] = await Value.findOrCreate({
16       where: {
17         EntryId: entryId,
18         AttributeId: attributeId,
19       },
20       include: [{ all: true }],
21     });
22
23     // TODO Validate new value
24
25     const [value] = await Value.update(
26       { value: newValue },
27       { where: { id: foundValue.id } },
28     );
29
30     const response = {
31       value,
32       success: true,
33     };
34     return res.json(response);
35   } catch (e) {
36     return res.status(501).json({
37       error: e,
38       success: false,
39     });
40   }
41 });
42
43
44 export default router;
```

1.165 natural_test.js

```
1 import { expect } from 'chai';
2 import { describe, it } from 'mocha';
3 import Natural from '../src/components/natural';
4
5 describe('Natural Service', () => {
6   it('should exist', () => {
7     /* eslint-disable no-unused-expressions */
8     expect(Natural).to.exist;
9   });
10
11 describe('seperateSentences', () => {
12   it('should correctly seperate a string into different sentences', () => {
13     const text = 'On Jan. 20, former Sen. Barack Obama became the 44th
14       President of the U.S. Millions attended the Inauguration.';
15
16     const expected = [
17       'On Jan. 20, former Sen. Barack Obama became the 44th \n President of the U.S.',
18       'Millions attended the Inauguration.',
19     ];
20
21     expect(Natural.seperateSentences(text)).to.deep.equal(expected);
22   });
23 });
24
25 describe('parse', () => {
26   it('should deconstruct a sentence and annotate recognisable entities.', async () => {
27     const text = 'Bob brought the pizza to Alice.';
28
29     const result = await Natural.parse(text);
30
31     expect(result).to.exist;
32     expect(result.data[0].parse_list.length).to.equal(7);
33     expect(result.data[0].noun_phrases.length).to.equal(3);
34     expect(result.data[0].text).to.equal('Bob brought the pizza to Alice.');
```

```

51         modifiers: [],
52     },
53 ],
54 },
55 ],
56 };
57
58 const expected = {
59     lemma: 'yellow',
60     pos: 'ADJ',
61     modifiers: [],
62 };
63
64 const result = Natural._find(tree, o => o.lemma === 'yellow');
65 expect(result).to.deep.equal(expected);
66 });
67 });
68
69 describe('filterTree', () => {
70     it('should remove nodes which don\'t match a condition', () => {
71         const tree = {
72             pos: 'VBZ',
73             modifiers: [
74                 {
75                     pos: 'JJ',
76                 },
77                 {
78                     pos: 'NN',
79                 },
80             ],
81         };
82
83         const result = Natural._filterTree(tree, o => o.pos !== 'JJ');
84         console.log(result);
85         const expected = {
86             pos: 'VBZ',
87             modifiers: [
88                 {
89                     modifiers: undefined,
90                     pos: 'NN',
91                 },
92             ],
93         };
94         expect(result).to.deep.equal(expected);
95     });
96
97     it('should keep child nodes which match the condition', () => {
98         const tree = {
99             pos: 'VBZ',
100             word: 'store',
101             modifiers: [
102                 {
103                     pos: 'IN',

```

```

104         word: 'about',
105         modifiers: [
106             {
107                 pos: 'NN',
108                 word: 'movies',
109             },
110         ],
111     },
112     {
113         pos: 'NN',
114         word: 'information',
115     },
116 ],
117 };
118
119 const result = Natural._filterTree(tree, o => o.pos !== 'IN');
120 const expected = {
121     pos: 'VBZ',
122     word: 'store',
123     modifiers: [
124         {
125             pos: 'NN',
126             word: 'movies',
127             modifiers: undefined,
128         },
129         {
130             pos: 'NN',
131             word: 'information',
132             modifiers: undefined,
133         },
134     ],
135 };
136 expect(result).to.deep.equal(expected);
137 });
138
139 it('should not alter the original tree', () => {
140     const tree = {
141         pos: 'VBZ',
142         word: 'store',
143         modifiers: [
144             {
145                 pos: 'IN',
146                 word: 'about',
147                 modifiers: [
148                     {
149                         pos: 'NN',
150                         word: 'movies',
151                     },
152                 ],
153             },
154             {
155                 pos: 'NN',
156                 word: 'information',

```

```

157     },
158   ],
159 };
160
161   const copy = JSON.parse(JSON.stringify(tree));
162   const result = Natural._filterTree(tree, o => o.pos !== 'IN');
163   expect(tree).to.deep.equal(copy);
164 });
165 });
166
167 describe('findAll', () => {
168   it('should find all modifiers in tree which satisfy a condition', () => {
169     const tree = {
170       lemma: 'runs',
171       pos: 'VERB',
172       modifiers: [
173         {
174           lemma: 'duck',
175           pos: 'NOUN',
176           modifiers: [
177             {
178               lemma: 'yellow',
179               pos: 'ADJ',
180               modifiers: [],
181             },
182             {
183               lemma: 'happy',
184               pos: 'ADJ',
185               modifiers: [],
186             },
187           ],
188         },
189       ],
190     };
191
192     const expected = [
193       {
194         lemma: 'happy',
195         pos: 'ADJ',
196         modifiers: [],
197       },
198       {
199         lemma: 'yellow',
200         pos: 'ADJ',
201         modifiers: [],
202       },
203     ];
204
205     const result = Natural._findAll(tree, o => o.pos === 'ADJ');
206     expect(result).to.deep.equal(expected);
207   });
208 });
209

```

```

210 describe('findIfPropertyIsRequired', () => {
211   it('should deduce a property is not required when no information is given', () => {
212     const prop = {
213       lemma: 'cat',
214       modifiers: [],
215     };
216
217     const context = {
218       lemma: 'play',
219       modifiers: [],
220     };
221
222     const result = Natural._findIfPropertyIsRequired(prop, context);
223     expect(result).to.equal(false);
224   });
225
226   it('should deduce a property is required when there is only required keywords', () => {
227     const prop = {
228       lemma: 'cat',
229       modifiers: [],
230     };
231
232     const context = {
233       lemma: 'play',
234       modifiers: [
235         { lemma: 'must', arc: 'aux' },
236       ],
237     };
238
239     const result = Natural._findIfPropertyIsRequired(prop, context);
240     expect(result).to.equal(true);
241   });
242
243   it('should deduce a property is not required when there are only optional keywords', () => {
244     const prop = {
245       lemma: 'cat',
246       modifiers: [],
247     };
248
249     const context = {
250       lemma: 'play',
251       modifiers: [
252         { lemma: 'might', arc: 'aux' },
253       ],
254     };
255
256     const result = Natural._findIfPropertyIsRequired(prop, context);
257     expect(result).to.equal(false);
258   });
259
260   it('should deduce a property is required when there are more required keywords than optional keywords', () => {
261     const prop = {
262       lemma: 'cat',

```

```

263     modifiers: [],
264 };
265
266     const context = {
267       lemma: 'play',
268       modifiers: [
269         { lemma: 'might', arc: 'aux' },
270         { lemma: 'needs', arc: 'aux' },
271         { lemma: 'must', arc: 'aux' },
272       ],
273     };
274
275     const result = Natural._findIfPropertyIsRequired(prop, context);
276     expect(result).to.equal(true);
277   });
278
279   it('should deduce a property is not required when there are more optional keywords than required keywords', () => {
280     const prop = {
281       lemma: 'cat',
282       modifiers: [],
283     };
284
285     const context = {
286       lemma: 'play',
287       modifiers: [
288         { lemma: 'might', arc: 'aux' },
289         { lemma: 'may', arc: 'aux' },
290         { lemma: 'could', arc: 'aux' },
291         { lemma: 'needs', arc: 'aux' },
292         { lemma: 'must', arc: 'aux' },
293       ],
294     };
295
296     const result = Natural._findIfPropertyIsRequired(prop, context);
297     expect(result).to.equal(false);
298   });
299 });
300
301 describe('findIfPropertyHasMultiple', () => {
302   it('should determine its singular if no information is given', () => {
303     const prop = {
304       lemma: 'cat',
305       modifiers: [],
306     };
307
308     const result = Natural._findIfPropertyHasMultiple(prop);
309     expect(result).to.equal(false);
310   });
311
312   it('should determine its multiple if word is plural', () => {
313     const prop = {
314       lemma: 'cats',
315       POS_fine: 'NNS',

```

```

316     modifiers: [],
317   };
318
319   const result = Natural._findIfPropertyHasMultiple(prop);
320   expect(result).to.equal(true);
321 });
322
323 it('should determine its multiple if prop has modifiers with plural keywords', () => {
324   ['det', 'amod'].forEach((arc) => {
325     const prop = {
326       lemma: 'cats',
327       POS_fine: 'NN',
328       modifiers: [
329         { arc, lemma: 'many' },
330       ],
331     };
332
333     const result = Natural._findIfPropertyHasMultiple(prop);
334     expect(result).to.equal(true);
335   });
336 });
337
338 it('should determine its singular if prop has modifiers with singular keywords', () => {
339   ['det', 'amod'].forEach((arc) => {
340     const prop = {
341       lemma: 'cats',
342       POS_fine: 'NN',
343       modifiers: [
344         { arc, lemma: 'single' },
345       ],
346     };
347
348     const result = Natural._findIfPropertyHasMultiple(prop);
349     expect(result).to.equal(false);
350   });
351 });
352
353 it('should determine its singular if prop has modifiers with singular keywords', () => {
354   ['det', 'amod'].forEach((arc) => {
355     const prop = {
356       lemma: 'cats',
357       POS_fine: 'NN',
358       modifiers: [
359         { arc, lemma: 'single' },
360       ],
361     };
362
363     const result = Natural._findIfPropertyHasMultiple(prop);
364     expect(result).to.equal(false);
365   });
366 });
367
368 it('should determine its singular if prop has singular number', () => {

```



```

369     ['zero', 'one'].forEach((lemma) => {
370         const prop = {
371             lemma: 'cats',
372             POS_fine: 'NN',
373             modifiers: [
374                 { arc: 'nummod', lemma },
375             ],
376         };
377
378         const result = Natural._findIfPropertyHasMultiple(prop);
379         expect(result).toEqual(false);
380     });
381 });
382
383 it('should determine its singular if prop has singular number', () => {
384     ['twenty two', 'nine', 'fifty', 'ten thousand'].forEach((lemma) => {
385         const prop = {
386             lemma: 'cats',
387             POS_fine: 'NN',
388             modifiers: [
389                 { arc: 'nummod', lemma },
390             ],
391         };
392
393         const result = Natural._findIfPropertyHasMultiple(prop);
394         expect(result).toEqual(true);
395     });
396 });
397 });
398
399 describe('generateModelStructure', () => {
400     it('should correctly analyse basic Pet model structure', async () => {
401         const text = 'A pet has a name, breed and owner. The Owner has a name. The owner owns a pet. Toy has a name. Pet likes a toy.';
402
403         const modelStructure = await Natural.generateModelStructure(text);
404
405         const expected = [
406             {
407                 name: 'pet',
408                 raw: 'pet',
409                 properties: [
410                     {
411                         type: 'string',
412                         name: 'name',
413                         raw: 'name',
414                         lemma: 'name',
415                         required: false,
416                         multiple: false,
417                     },
418                     {
419                         type: 'string',
420                         name: 'breed',
421                         raw: 'breed',

```

```
422         lemma: 'breed',
423         required: false,
424         multiple: false,
425     },
426     {
427         type: 'Owner',
428         name: 'owner',
429         raw: 'owner',
430         lemma: 'owner',
431         required: false,
432         multiple: false,
433     },
434     {
435         type: 'Toy',
436         name: 'likes_toy',
437         raw: 'toy',
438         lemma: 'toy',
439         required: false,
440         multiple: false,
441     },
442 ],
443 },
444 {
445     name: 'owner',
446     raw: 'Owner',
447     properties: [
448         {
449             type: 'string',
450             name: 'name',
451             raw: 'name',
452             lemma: 'name',
453             required: false,
454             multiple: false,
455         },
456         {
457             type: 'Pet',
458             name: 'owns_pet',
459             raw: 'pet',
460             lemma: 'pet',
461             required: false,
462             multiple: false,
463         },
464     ],
465 },
466 {
467     name: 'toy',
468     raw: 'Toy',
469     properties: [
470         {
471             type: 'string',
472             name: 'name',
473             raw: 'name',
474             lemma: 'name',
```

```
475         required: false,
476         multiple: false,
477     },
478 ],
479 },
480 ];
481
482     expect(modelStructure).to.deep.equal(expected);
483 });
484 });
485 });
```

1.166 parse_test.js

```
1 import { expect } from 'chai';
2 import { describe, it } from 'mocha';
3 import { parseSpreadsheet, findType, determineType } from '../src/components/parse';
4
5 describe('Parse Service', () => {
6   it('should exist', () => {
7     expect(parseSpreadsheet).to.exist;
8   });
9
10  describe('parseSpreadsheet', () => {
11    const input = 'policyID,statecode,county,eq_site_limit,hu_site_limit,fl_site_limit,fr_site_limit,tiv_2011,tiv_2012,eq_site_deductible,
12      hu_site_deductible,fl_site_deductible,fr_site_deductible,point_latitude,point_longitude,line,construction,point_granularity
13  119736,FL,CLAY COUNTY,498960,498960,498960,498960,498960,792148.9,0,9979.2,0,0,30.102261,-81.711777,Residential,Masonry,1
14  448094,FL,CLAY COUNTY,1322376.3,1322376.3,1322376.3,1322376.3,1322376.3,1438163.57,0,0,0,0,30.063936,-81.707664,Residential,Masonry,3
15  206893,FL,CLAY COUNTY,190724.4,190724.4,190724.4,190724.4,190724.4,192476.78,0,0,0,0,30.089579,-81.700455,Residential,Wood,1
16  333743,FL,CLAY COUNTY,0,79520.76,0,0,79520.76,86854.48,0,0,0,0,30.063236,-81.707703,Residential,Wood,3
17  172534,FL,CLAY COUNTY,0,254281.5,0,254281.5,254281.5,246144.49,0,0,0,0,30.060614,-81.702675,Residential,Wood,1
18  785275,FL,CLAY COUNTY,0,515035.62,0,0,515035.62,884419.17,0,0,0,0,30.063236,-81.707703,Residential,Masonry,3
19  995932,FL,CLAY COUNTY,0,19260000,0,0,19260000,20610000,0,0,0,0,30.102226,-81.713882,Commercial,Reinforced Concrete,1
20  223488,FL,CLAY COUNTY,328500,328500,328500,328500,328500,348374.25,0,16425,0,0,30.102217,-81.707146,Residential,Wood,1
21  433512,FL,CLAY COUNTY,315000,315000,315000,315000,315000,265821.57,0,15750,0,0,30.118774,-81.704613,Residential,Wood,1
22  142071,FL,CLAY COUNTY,705600,705600,705600,705600,705600,1010842.56,14112,35280,0,0,30.100628,-81.703751,Residential,Masonry,1
23  253816,FL,CLAY COUNTY,831498.3,831498.3,831498.3,831498.3,831498.3,1117791.48,0,0,0,0,30.10216,-81.719444,Residential,Masonry,1
24  894922,FL,CLAY COUNTY,0,24059.09,0,0,24059.09,33952.19,0,0,0,0,30.095957,-81.695099,Residential,Wood,1
25  422834,FL,CLAY COUNTY,0,48115.94,0,0,48115.94,66755.39,0,0,0,0,30.100073,-81.739822,Residential,Wood,1
26  582721,FL,CLAY COUNTY,0,28869.12,0,0,28869.12,42826.99,0,0,0,0,30.09248,-81.725167,Residential,Wood,1
27  842700,FL,CLAY COUNTY,0,56135.64,0,0,56135.64,50656.8,0,0,0,0,30.101356,-81.726248,Residential,Wood,1
28  874333,FL,CLAY COUNTY,0,48115.94,0,0,48115.94,67905.07,0,0,0,0,30.113743,-81.727463,Residential,Wood,1
29  580146,FL,CLAY COUNTY,0,48115.94,0,0,48115.94,66938.9,0,0,0,0,30.121655,-81.732391,Residential,Wood,3
30  456149,FL,CLAY COUNTY,0,80192.49,0,0,80192.49,86421.04,0,0,0,0,30.109537,-81.741661,Residential,Wood,1
31  767862,FL,CLAY COUNTY,0,48115.94,0,0,48115.94,73798.5,0,0,0,0,30.11824,-81.745335,Residential,Wood,3
32  353022,FL,CLAY COUNTY,0,60946.79,0,0,60946.79,62467.29,0,0,0,0,30.065799,-81.717416,Residential,Wood,1
33  367814,FL,CLAY COUNTY,0,28869.12,0,0,28869.12,42727.74,0,0,0,0,30.082993,-81.710581,Residential,Wood,1
34  671392,FL,CLAY COUNTY,0,13410000,0,0,13410000,11700000,0,0,0,0,30.091921,-81.711929,Commercial,Reinforced Concrete,3
35  772887,FL,CLAY COUNTY,0,1669113.93,0,0,1669113.93,2099127.76,0,0,0,0,30.117352,-81.711884,Residential,Masonry,1
36  983122,FL,CLAY COUNTY,0,179562.23,0,0,179562.23,211372.57,0,0,0,0,30.095783,-81.713181,Residential,Wood,3
37  934215,FL,CLAY COUNTY,0,177744.16,0,0,177744.16,157171.16,0,0,0,0,30.110518,-81.727478,Residential,Wood,1
38  385951,FL,CLAY COUNTY,0,17757.58,0,0,17757.58,16948.72,0,0,0,0,30.10288,-81.705719,Residential,Wood,1
39  716332,FL,CLAY COUNTY,0,130129.87,0,0,130129.87,101758.43,0,0,0,0,30.068468,-81.71624,Residential,Wood,1
40  751262,FL,CLAY COUNTY,0,42854.77,0,0,42854.77,63592.88,0,0,0,0,30.068468,-81.71624,Residential,Wood,1
41  633663,FL,CLAY COUNTY,0,785.58,0,0,785.58,662.18,0,0,0,0,30.068468,-81.71624,Residential,Wood,1
42  105851,FL,CLAY COUNTY,0,170361.91,0,0,170361.91,177176.38,0,0,0,0,30.068468,-81.71624,Residential,Wood,1
43 '
44   it('should find the correct model definition', () => {
45     // const result = parseSpreadsheet([input]);
46     // expect(result).to.equal([
47     //   {
48     //     "name": "",
49     //     "raw": "",
50     //     "properties": [
```

```

50      //      {
51      //          "type": "integer",
52      //          "name": "policyID",
53      //          "raw": "policyID",
54      //          "lemma": "policyID",
55      //          "required": true,
56      //          "multiple": false
57      //      },
58      //      {
59      //          "type": "string",
60      //          "name": "breed",
61      //          "raw": "breed",
62      //          "lemma": "breed",
63      //          "required": false,
64      //          "multiple": false
65      //      },
66      //      {
67      //          "type": "Owner",
68      //          "name": "owner",
69      //          "raw": "owner",
70      //          "lemma": "owner",
71      //          "required": false,
72      //          "multiple": false
73      //      },
74      //      {
75      //          "type": "Toy",
76      //          "name": "likes_toy",
77      //          "raw": "toy",
78      //          "lemma": "toy",
79      //          "required": false,
80      //          "multiple": false
81      //      }
82      //  ]
83      // },
84      // {
85      //     "name": "owner",
86      //     "raw": "Owner",
87      //     "properties": [
88      //         {
89      //             "type": "string",
90      //             "name": "name",
91      //             "raw": "name",
92      //             "lemma": "name",
93      //             "required": false,
94      //             "multiple": false
95      //         },
96      //         {
97      //             "type": "Pet",
98      //             "name": "owns_pet",
99      //             "raw": "pet",
100            //             "lemma": "pet",
101            //             "required": false,
102            //             "multiple": false

```

```

103     //     }
104     //   ]
105     // },
106     // {
107     //   "name": "toy",
108     //   "raw": "Toy",
109     //   "properties": [
110     //     {
111     //       "type": "string",
112     //       "name": "name",
113     //       "raw": "name",
114     //       "lemma": "name",
115     //       "required": false,
116     //       "multiple": false
117     //     }
118     //   ]
119     // }
120   // ]);
121 });
122 });
123
124 describe('findType', () => {
125   it('should return null if no value is supplied', () => {
126     const result = findType();
127     expect(result).to.equal(null);
128   });
129
130   it('should return float if string contains one dot', () => {
131     const result = findType('5.3');
132     expect(result.type).to.equal('float');
133   });
134
135   it('should return string if string contains more than one dot', () => {
136     const result = findType('5.3.3');
137     expect(result.type).to.equal('string');
138   });
139
140   it('should return integer if string is only digits', () => {
141     const result = findType('432');
142     expect(result.type).to.equal('integer');
143   });
144
145   it('should return string otherwise', () => {
146     const result = findType('This is a sentence. ');
147     expect(result.type).to.equal('string');
148   });
149
150   it('should detect arrays and find the type of elements', () => {
151     const result = findType('[5.5,3.2,2.3] ');
152
153     expect(result.multiple).to.equal(true);
154     expect(result.type).to.equal('float');
155   });

```

```

156
157     it('should detect arrays but throw if they are multidimensional', () => {
158         const result = findType('[5.5],[3.2],[2.3]');
159
160         // expect(findType).toThrow(Error); TODO Check for throwing error
161     });
162 });
163
164 describe('determineType', () => {
165     return;
166     it('should return string if one of the types is string', () => {
167         const result = determineType([
168             {
169                 type: 'string',
170                 multiple: 'false',
171             },
172             {
173                 type: 'float',
174                 multiple: 'false',
175             },
176             {
177                 type: 'integer',
178                 multiple: 'false',
179             },
180         ]);
181
182         expect(result.type).toEqual('string');
183         expect(result.required).toEqual(true);
184     });
185
186     it('should return float if one of the types is float and there is no string', () => {
187         const result = determineType([
188             {
189                 type: 'float',
190                 multiple: 'false',
191             },
192             {
193                 type: 'integer',
194                 multiple: 'false',
195             },
196         ]);
197
198         expect(result.type).toEqual('float');
199         expect(result.required).toEqual(true);
200     });
201
202     it('should return integer if one of the types is float and there is no string', () => {
203         const result = determineType([
204             {
205                 type: 'integer',
206                 multiple: 'false',
207             },
208             {

```

```
209         type: 'integer',
210         multiple: 'false',
211     },
212 ];
213
214     expect(result.type).to.equal('float');
215     expect(result.required).to.equal(true);
216 });
217
218 it('should not be required if one of the types is not required', () => {
219     const result = determineType([
220         {
221             type: 'string',
222             multiple: 'false',
223         },
224         null,
225     ]);
226
227     expect(result.type).to.equal('string');
228     expect(result.required).to.equal(false);
229 });
230 });
231 });
```


1.167 service_test.js

```
1 import { describe, it } from 'mocha';
2 // import { createService } from '../src/components/service';
3
4 describe('createService', () => {
5   it('should correctly create a service', () => {
6     // createService(
7     //   'Cats',
8     //   [
9     //     {
10      //       name: 'guy',
11      //       attributes: [
12      //         {
13      //           type: 'string',
14      //           name: 'name',
15      //         },
16      //         {
17      //           type: 'integer',
18      //           name: 'age',
19      //         },
20      //       ],
21      //       entries: [
22      //         {
23      //           name: 'Tom',
24      //           age: 50,
25      //         },
26      //         {
27      //           name: 'Jack',
28      //           age: 20,
29      //         },
30      //       ],
31      //     },
32      //   ],
33      //   1,
34      // );
35   });
36 });
```