

Contents

A Extra Information

- A.1 List of third party libraries and tools
- A.2 Example of a model definition

B User Guide

- B.1 Instructions

C Source code

- C.1 ./frontend/src/actions/actionTypes.js
- C.2 ./frontend/src/actions/auth/authUser.js
- C.3 ./frontend/src/actions/auth/authUserResult.js
- C.4 ./frontend/src/actions/auth/getServiceList.js
- C.5 ./frontend/src/actions/auth/getUser.js
- C.6 ./frontend/src/actions/auth/index.js
- C.7 ./frontend/src/actions/auth/logoutUser.js
- C.8 ./frontend/src/actions/auth/receiveService.js
- C.9 ./frontend/src/actions/auth/receiveServiceList.js
- C.10 ./frontend/src/actions/auth/updateUser.js
- C.11 ./frontend/src/actions/dashboard/changeDashboardPage.js
- C.12 ./frontend/src/actions/dashboard/changeSelectedModel.js
- C.13 ./frontend/src/actions/dashboard/changeSidebarItem.js
- C.14 ./frontend/src/actions/dashboard/createAttribute.js
- C.15 ./frontend/src/actions/dashboard/createEntry.js
- C.16 ./frontend/src/actions/dashboard/createModel.js
- C.17 ./frontend/src/actions/dashboard/deleteAttribute.js
- C.18 ./frontend/src/actions/dashboard/deleteAttributeLocally.js
- C.19 ./frontend/src/actions/dashboard/deleteEntry.js
- C.20 ./frontend/src/actions/dashboard/deleteEntryLocally.js
- C.21 ./frontend/src/actions/dashboard/deleteModel.js
- C.22 ./frontend/src/actions/dashboard/deleteModelLocally.js

C.23	./frontend/src/actions/dashboard/receiveAttribute.js	...
C.24	./frontend/src/actions/dashboard/receiveEntry.js	...
C.25	./frontend/src/actions/dashboard/receiveModel.js	...
C.26	./frontend/src/actions/dashboard/selectAttribute.js	...
C.27	./frontend/src/actions/dashboard/updateAttribute.js	...
C.28	./frontend/src/actions/dashboard/updateAttributeLocally.js	...
C.29	./frontend/src/actions/dashboard/updateModel.js	...
C.30	./frontend/src/actions/dashboard/updateModelLocally.js	...
C.31	./frontend/src/actions/dashboard/updateService.js	...
C.32	./frontend/src/actions/dashboard/updateServiceLocally.js	...
C.33	./frontend/src/actions/dashboard/updateValue.js	...
C.34	./frontend/src/actions/dashboard/updateValueLocally.js	...
C.35	./frontend/src/actions/other/showError.js	...
C.36	./frontend/src/actions/setup/analyseNaturalText.js	...
C.37	./frontend/src/actions/setup/analyseSpreadsheet.js	...
C.38	./frontend/src/actions/setup/createService.js	...
C.39	./frontend/src/actions/setup/index.js	...
C.40	./frontend/src/actions/setup/newService.js	...
C.41	./frontend/src/actions/setup/nextScreen.js	...
C.42	./frontend/src/actions/setup/receiveService.js	...
C.43	./frontend/src/actions/setup/selectService.js	...
C.44	./frontend/src/actions/setup/setServiceCreateMethod.js	...
C.45	./frontend/src/actions/setup/setServiceName.js	...
C.46	./frontend/src/actions/setup/updateModelPreview.js	...
C.47	./frontend/src/actions/setup/updateNaturalText.js	...
C.48	./frontend/src/components/AuthForm.jsx	...
C.49	./frontend/src/components/Button.jsx	...
C.50	./frontend/src/components/dashboard/about/About.jsx	...
C.51	./frontend/src/components/dashboard/Dashboard.jsx	...
C.52	./frontend/src/components/dashboard/entries/Column.jsx	...
C.53	./frontend/src/components/dashboard/entries/Entries.jsx	...
C.54	./frontend/src/components/dashboard/entries/Row.jsx	...
C.55	./frontend/src/components/dashboard/entries/RowHeader.jsx	...
C.56	./frontend/src/components/dashboard/entries/rowStyle.js	...
C.57	./frontend/src/components/dashboard/entries/Tabs.jsx	...
C.58	./frontend/src/components/dashboard/pages/Pages.jsx	...
C.59	./frontend/src/components/dashboard/Sidebar.jsx	...
C.60	./frontend/src/components/dashboard/SidebarItem.jsx	...

C.61	./frontend/src/components/dashboard/structure/Attribute.jsx
C.62	./frontend/src/components/dashboard/structure/DialogBox.jsx
C.63	./frontend/src/components/dashboard/structure/Model.jsx
C.64	./frontend/src/components/dashboard/structure/Structure.jsx
C.65	./frontend/src/components/dashboard/TopBar.jsx
C.66	./frontend/src/components/Frame.jsx
C.67	./frontend/src/components/HomePage.jsx
C.68	./frontend/src/components/Logo.jsx
C.69	./frontend/src/components/MethodButton.jsx
C.70	./frontend/src/components/RoundButton.jsx
C.71	./frontend/src/components/ServiceList.jsx
C.72	./frontend/src/components/ServiceListItem.jsx
C.73	./frontend/src/components/setup/Setup.jsx
C.74	./frontend/src/components/setup/SetupMethod.jsx
C.75	./frontend/src/components/setup/SetupName.jsx
C.76	./frontend/src/components/setup/SetupNatural.jsx
C.77	./frontend/src/components/setup/SetupSpreadsheet.jsx
C.78	./frontend/src/components/StyleConstant.js
C.79	./frontend/src/components/TextInput.jsx
C.80	./frontend/src/containers/AuthFormContainer.js
C.81	./frontend/src/containers/dashboard/AboutContainer.js
C.82	./frontend/src/containers/dashboard/EntriesContainer.js
C.83	./frontend/src/containers/dashboard/PagesContainer.js
C.84	./frontend/src/containers/dashboard/SidebarContainer.js
C.85	./frontend/src/containers/dashboard/StructureContainer.js
C.86	./frontend/src/containers/HomePageContainer.js
C.87	./frontend/src/containers/ServiceListContainer.js
C.88	./frontend/src/containers/setup/SetupContainer.js
C.89	./frontend/src/containers/setup/SetupMethodContainer.js
C.90	./frontend/src/containers/setup/SetupNameContainer.js
C.91	./frontend/src/containers/setup/SetupNaturalContainer.js
C.92	./frontend/src/containers/setup/SetupSpreadsheetContainer.js
C.93	./frontend/src/index.js
C.94	./frontend/src/reducers/index.js
C.95	./frontend/src/utils/API.js
C.96	./frontend/src/utils/Auth.js
C.97	./frontend/src/utils/capitalizeString.js
C.98	./frontend/src/utils/createMethods.js

C.99	./frontend/src/utils/normalizr.js
C.100	./frontend/src/utils/setupScreens.js
C.101	./frontend/src/index.css
C.102	./frontend/package.json
C.103	./backend/package.json
C.104	./backend/src/components/natural.js
C.105	./backend/src/components/parse.js
C.106	./backend/src/components/service.js
C.107	./backend/src/components/utils.js
C.108	./backend/src/config/bootstrap.js
C.109	./backend/src/config/connections.js
C.110	./backend/src/config/passport.js
C.111	./backend/src/nlp/index.py
C.112	./backend/src/nlp/spacyparse.py
C.113	./backend/src/index.js
C.114	./backend/src/middleware/authentication.js
C.115	./backend/src/models/attribute.js
C.116	./backend/src/models/entry.js
C.117	./backend/src/models/index.js
C.118	./backend/src/models/model.js
C.119	./backend/src/models/service.js
C.120	./backend/src/models/user.js
C.121	./backend/src/models/value.js
C.122	./backend/src/routes/api.js
C.123	./backend/src/routes/attribute.js
C.124	./backend/src/routes/auth.js
C.125	./backend/src/routes/entry.js
C.126	./backend/src/routes/model.js
C.127	./backend/src/routes/service.js
C.128	./backend/src/routes/value.js
C.129	./backend/test/natural_test.js
C.130	./backend/test/parse_test.js

Appendix A

Extra Information

A.1 List of third party libraries and tools

- **ESLint**: A linting utility for Javascript
- **eslint-plugin-async-await**: ESLint plugin which enables async/await functionality
- **eslint-plugin-react**: React linting rules for ESLint
- **neutrino**: A companion tool for setting up the development environment
- **bcrypt**: A library for hashing passwords
- **body-parser**: A Node.js body parsing middle for Express.js
- **chai**: A test assertion library for Node.js
- **compromise**: A Natural Language Processing module
- **express**: Minimalistic web framework for Node.js
- **immutable**: A library which provides immutable data structures
- **isomorphic-fetch**: A Javascript polyfill for fetch
- **jsonwebtoken**: Support for JSON Web Tokens
- **multer**: Node.js middleware for multipart uploads
- **natural**: Natural language facilities for node
- **passport**: Authentication middleware for Node.js
- **passport-local**: A username and password strategy for passport
- **pg**: PostgreSQL client for Node.js
- **radium**: A toolchain for React styling

- **react**: A Javascript for building user interfaces
- **react-dom**: React bindings for the browser
- **react-redux**: React bindings for Redux
- **redux**: State container for Javascript applications
- **redux-immutable**: Support for Immutable in Redux
- **request-promise**: A HTTP request client with Promise support
- **sbd**: Sentence boundary detection
- **sequelize**: A promise-based ORM
- **sequelize-cli**: A Sequelize command line interface
- **spacy**: NLP processing library with dependency parsing
- **xlsx**: Spreadsheet parser and writer in Javascript
- **normalizr**: Library for normalizing nested JSON according to a schema
- **prop-types**: Runtime type checking for React
- **react-dropzone**: A HTML5 drag-drop zone for React
- **react-router**: Declarative routing for React
- **react-router-redux**: Bindings for react-router and Redux
- **react-scripts**: A development environment setup tool
- **underscore**: A library with utility functions
- **mocha**: Test framework for Javascript
- **redux-thunk**: Thunk middleware for Redux
- **spacy-nlp**: Exposes main Spacy functionality

A.2 Example of a model definition

```

{
  [
    {
      "name": "pet",
      "properties": [
        {

```

```

        "type": "string",
        "name": "name",
        "required": false ,
        "multiple": false
    },
    {
        "type": "string",
        "name": "breed",
        "required": false ,
        "multiple": false
    },
    {
        "type": "Owner",
        "name": "owner",
        "required": false ,
        "multiple": false
    },
    {
        "type": "Toy",
        "name": "likes_toy",
        "required": false ,
        "multiple": false
    }
]
},
{
    "name": "owner",
    "properties": [
        {
            "type": "string",
            "name": "name",
            "required": false ,
            "multiple": false
        },
        {
            "type": "Pet",
            "name": "owns_pet",

```

```
        "required": false ,
        "multiple": false
    }
]
},
{
    "name": "toy",
    "properties": [
        {
            "type": "string",
            "name": "name",
            "required": false ,
            "multiple": false
        }
    ]
}
];
```


Appendix B

User Guide

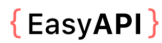
B.1 Instructions

The following guide provides instructions on how to use EasyAPI to create an API.

B.1.1 Setting up the environment

In order to run this project on your machine, follow these instructions:

1. Ensure you have Python 2.6+/3.3+ installed
2. If you haven't got pip installed, follow the instructions on the official pip installation page
3. Run the command "pip install --ignore-installed -U spacy" in Terminal/Command Line to install Spacy
4. Run the command "python -m spacy download en" to download the English corpus in Spacy. This may take over 30 minutes depending on the internet connection
5. If you have any issues following the two above instructions, please refer to the official instructions on the Spacy website
6. Install PostgreSQL by following the steps on the official website
7. Install Node.js from the download links in the official website (this will also install NPM)
8. Install Flask with the command "pip install flask"
9. Install the Node.js environment runners with the command "npm install -g neutrino react-scripts"
10. Run the PostgreSQL server and note the connection details
11. In the file "backend/src/config/connections.js", enter your connection details
12. Install the dependencies for the back-end with the command "npm install" while in the "./backend" directory
13. Install the dependencies for the front-end with the command "npm install" while in the "./frontend" directory



Welcome to EasyAPI!

If you are a new user, please create a new account by filling in the following fields.

If you are already a user, please enter your username and password.

A screenshot of the authentication form. It consists of two text input fields stacked vertically. The first field contains the text "martin". The second field contains seven dots, representing a masked password. Below these fields is a green rectangular button with the word "Next" in white text.

Figure B.1: The authentication screen

14. Run the python server with the command "python backend/src/nlp/index.py"
15. In a separate Terminal/Command Line process, run the back-end server with the command "npm start" while in the "./backend" directory
16. In a separate Terminal/Command Line process, run the front-end server with the command "npm start" while in the "./frontend" directory
17. Open "http://localhost:3000/" in a web browser and the application will appear

If you have issues running any of the commands above, try prefixing them with the "sudo" command. This will run them in an administrative mode.

B.1.2 Creating an account

When you first open the EasyAPI web application, you will be presented with an authentication screen. Here you will be asked to either create a new account or login with an existing one.

Since this is the first time, enter a username and a secret password into the text fields. Afterwards, click on the "Next" below.

If you receive any errors, simply adjust either your username or password to match the criteria.

B.1.3 Logging in with an account

If you are returning to the application with an existing account, you can use the authentication screen to login. Simply enter the email and password you used previously into the text-fields. The system will detect that these details were used before and will authenticate you.



Which API would you like to work on?



Figure B.2: List of APIs screen



What is the name of your API?

Next

Figure B.3: Name selection screen

B.1.4 Creating a new API

After successful authentication you should be transferred to the API List screen. Here you can view and modify existing APIs and also create new ones.

To create a new API, click on the green plus button. This will transfer you to the setup screen where you can continue creating your API.

B.1.5 Creating a new API with a domain description

Once you are on the setup screen, you can follow the instructions to create your API.

We will now build an API from a domain description. In the first screen, you are asked to enter a name for the API. Think of a 1-3 word title which effectively sums up your domain. For example, if we want to create an API for a Pet store, we can use the name “Pet Store”. After entering the name, click on the “Next” button.

How do you want to create your API?

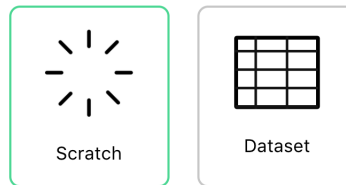


Figure B.4: Method selection screen

In the second screen, you will be asked to select how you want to create your API. Here we have two options: from a domain description or spreadsheet. In this case we want to create it from a domain description, therefore select the option by clicking on the “From scratch” option. Once it is selected, you can click on the next button.

Finally, we will be presented with a textbox where we can enter our domain description. In order for EasyAPI to extract the correct definition, it is best to describe the domain directly. One-by-one describe all the various entities that are in your domain, with their respective attributes and relationships.

For our pet store example, we may want to model a Pet store. We want to store information about our customers, employees, and our pets. This would be an appropriate domain description for those entities:

“A customer has a name. A customer owns a pet. A customer also has a phone number. An employee has a name, salary and an age. Pets have a type, a name and an customer.”

When you type, the system will preview the model definition that was extracted from your description. If you think that something has been incorrectly parsed, try to rephrase it in simpler, clearer terms. If it still fails to correctly infer your domain description, in the next screen you will be able to modify it manually.

B.1.6 Creating a new API with a spreadsheet

If you already have a spreadsheet containing data which you would like to convert to the API, you can do so directly in the setup.

After creating a new API in the service list screen and typing the name of the API, select the spreadsheet from the creation methods. Afterwards, click on the “Next” button, which will transfer you to the spreadsheet setup screen.

In this screen you will see an area where you can upload your spreadsheet. Simply open File Explorer on Windows or Finder on MacOS and find the file. Once ready, simply drag and drop it onto the upload area on the screen.

It will take a few seconds to upload the file. It will attempt to extract the model definition from the spreadsheet by analysing the various pages, headings, and values. Once finished, EasyAPI will present a preview of the model

Please describe the various things and entities, along with their properties and relationships

A customer has a name. A customer owns a pet. A customer also has a phone number. An employee has a name, salary and an age. Pets have a type, a name and an customer.

Customer	Employee	Pet
name (string)	name (string)	type (string)
owns pet (Pet)	salary (string)	name (string)
phone number (integer)	age (string)	customer (Customer)

Next

Figure B.5: Setup from scratch screen

definition.

If you find that the preview doesn't match your spreadsheet content, you can try one of these steps to help the parser:

- Make sure each spreadsheet page has its first row used for column headings.
- For each column, make sure the values are of the same type (string, integer, etc.)

If these steps did not help, you will be able to modify the model definition in the next screen.

B.1.7 Dashboard screen

Once you have finished creating your API, you will be transferred to the dashboard screen. Through this screen you will be able to modify various aspects of your API.

On the right of the screen there is a sidebar through which you can navigate to other dashboard screens. In the centre-right of the screen you will see the content of the page you have currently selected.

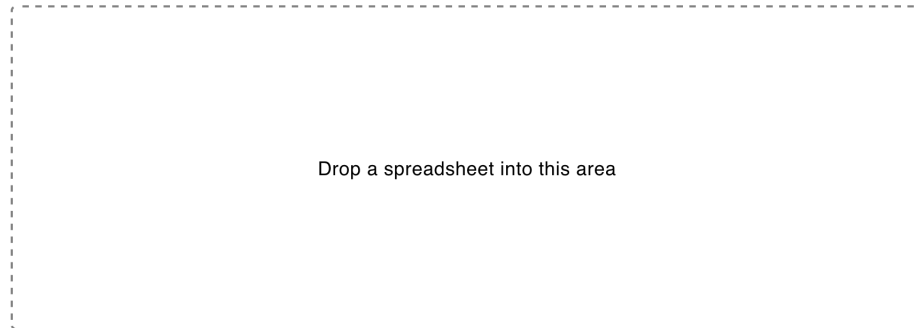
B.1.8 Modifying the database structure

We will first look at how we can change our database structure. Here you can modify the underlying model definition, which defines how the database is structured.

Initially you will see boxes representing the various existing entities in your database. Each box will have a list of attributes for that entity.

If you decide to create a new entity to store in the database, click on the green plus button on the top of the screen.

{ EasyAPI }



Next

Figure B.6: Spreadsheet upload screen

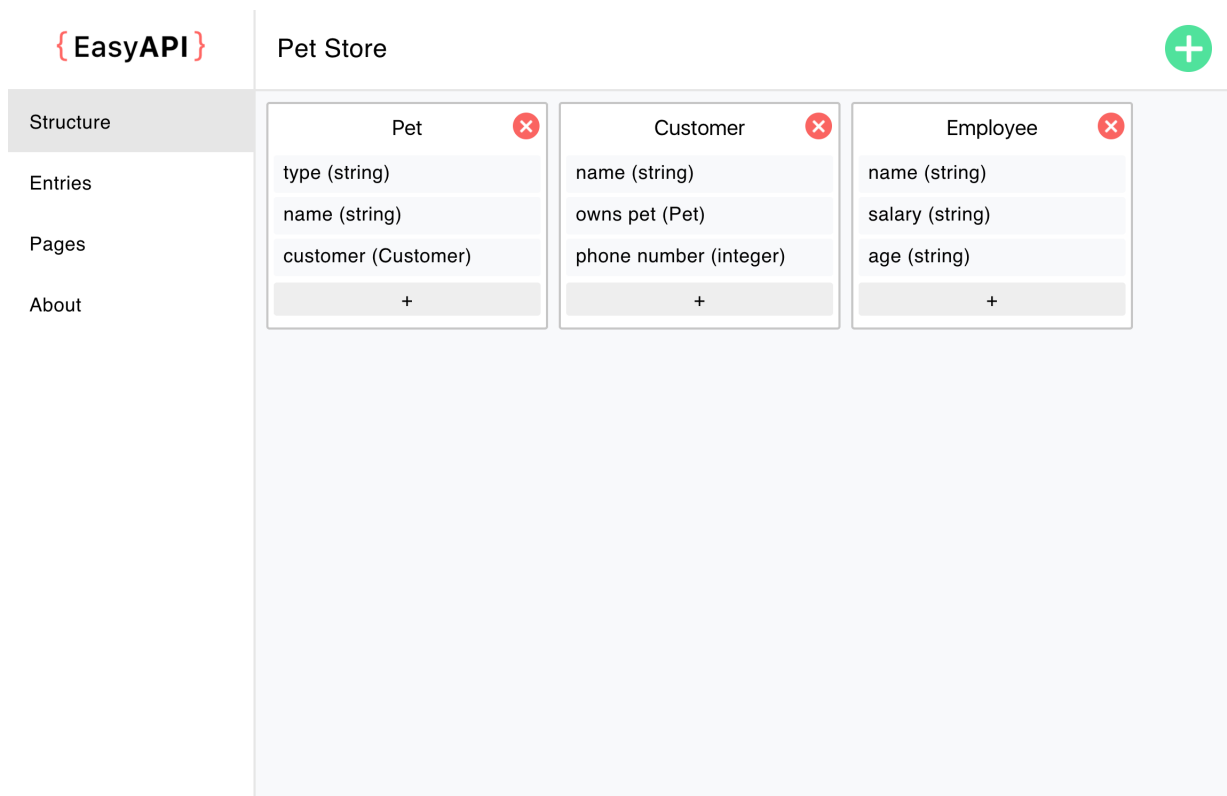


Figure B.7: Dashboard structure screen

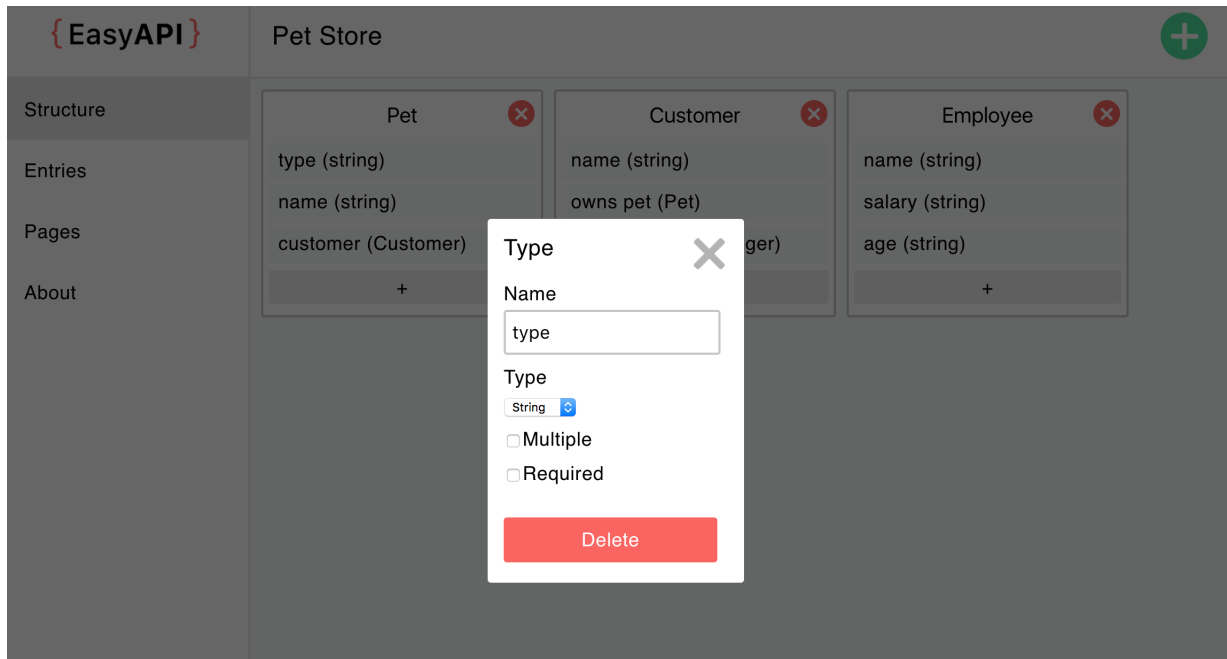


Figure B.8: Modify attribute dialog box

To modify the name of an entity, select the title of an entity-box and type in the new entity name.

To modify an attribute of an entity, click on the specific attribute you want to edit. This will open a dialogue box which will present a number of fields for that attribute and let you modify it.

To delete an entity, click on the red button in a model box.

B.1.9 Modifying the database entries

The dashboard screen also allows you to manually modify the data in your database. Start by clicking on the "Entities" list item in the sidebar.

Your database is presented through a spreadsheet-like format. On the top of the screen you will see tabs for your corresponding entities. By clicking on each tab, the corresponding table will be shown.

To edit a value, simply click on it and type the new value. Be aware that it needs to conform to the defined type.

To delete an entry, click on red button which appears on the right to delete the row.

To create a new entry, click on the green button on the top of the screen and fill in the values in the row.

B.1.10 Modifying your public pages

The "Pages" screen in the dashboard screen gives you control over which actions will be available to the public. Upon opening this page, you will see a list of your entities, each with a list of actions with check-boxes.

There are five optional actions for each entity:

- **Find One:** This action is used to retrieve data of one resource on the server. It corresponds to the HTTP method GET.
- **Find:** This action is similar to the one above, except it is used to retrieve a list of resources.




{ EasyAPI }	Pet Store				
Structure	Pet	Customer	Employee		
Entries	ID	Type	Name	Customer	
Pages	1	cat	Tom	2	
About	2	dog	Rex	1	

Figure B.9: Entries screen

- **Create:** This action is used to add a new entry to the database. It corresponds to the “POST” HTTP method.
- **Update:** This action allows users to modify existing resources and it corresponds to the PATCH HTTP action. Note that it does not correspond to the PUT HTTP action, as the request only requires the attributes to be updated.
- **Delete:** This action is used to delete an entry from the database.

To make a page available to the public, simply tick the checkbox next to an action. The URL of the endpoint will appear below.

B.1.11 Modifying metadata and publishing

Finally, you can also edit the metadata of the API and publish it.

You will find this screen by clicking on “About” in the sidebar.

To modify the name of the API, select the text field with the “Name” label and type your new name.

To modify the URL of the API, select the text field with the “URL” label and type in the new identifier for the API.

To publish your API, tick the checkbox next to the label “Public”. You can unpublish it by un-ticking this checkbox.

B.1.12 Accessing your API

After you publish your API, the first thing you might want to do is to test it out. You can do so by returning to the “Pages” screen, where you will see URLs under every enabled action. These are in the form of URL templates, where variables in curly brackets should be replaced by a value. For instance if a URL is the following:

`http://localhost:9001/api/api/petstore/cats/id`

You should replace “id” with an entry id. For certain actions where a request body is required, you will need to use a tool such as Postman to make a request.

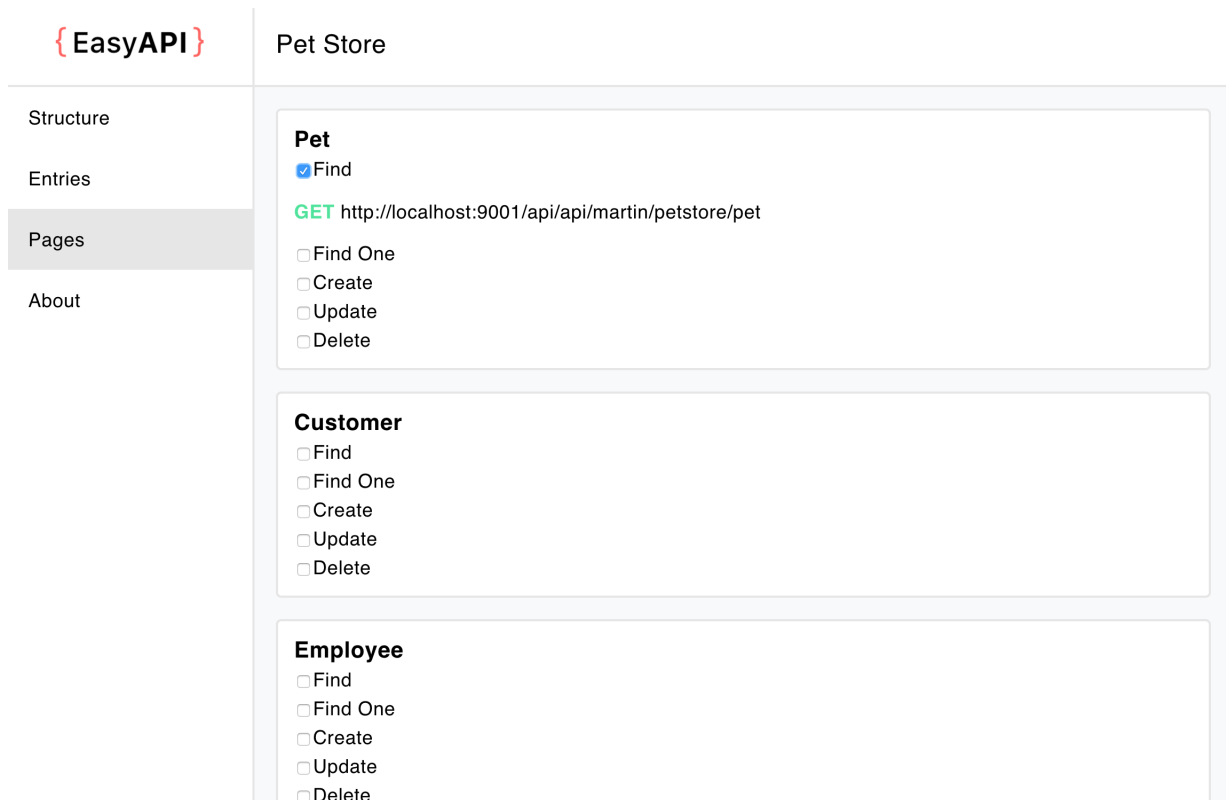


Figure B.10: Pages screen

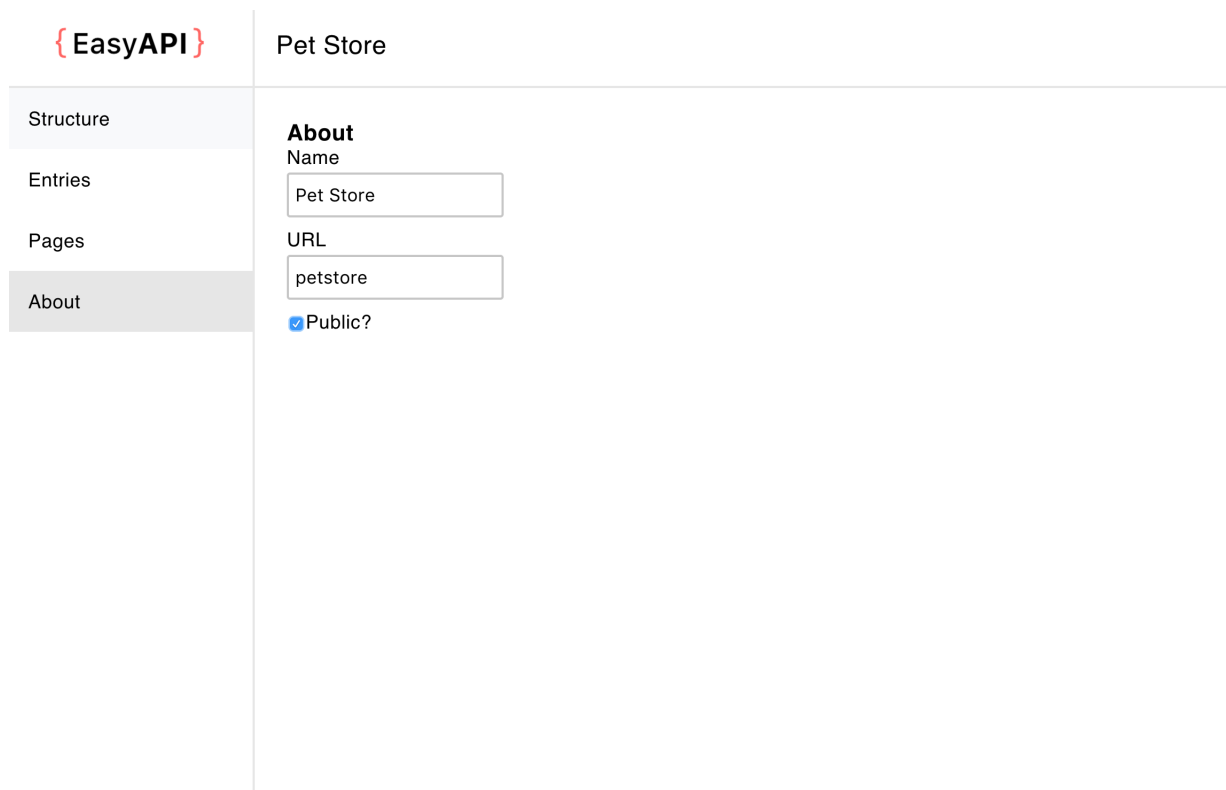


Figure B.11: About screen

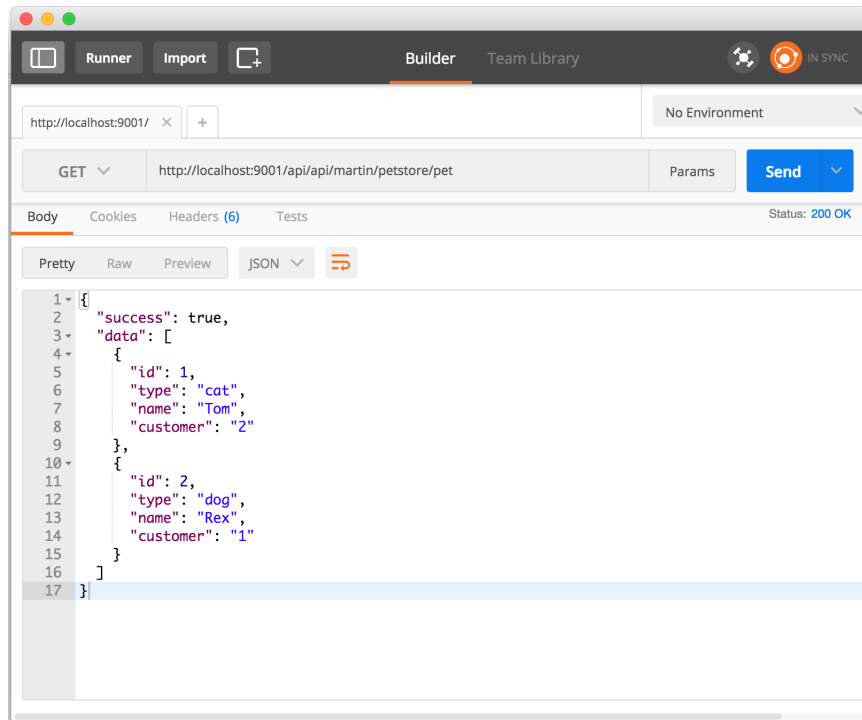


Figure B.12: Accessing API through Postman

By making a request to the specified URL with the correct HTTP action and body, you should see the expected results. For the POST and PATCH actions, the body should be specified in JSON format.

Appendix C

Source code

NOTE: Import statements from absolute paths (e.g. `import redux from 'redux'`) indicate the use of an external library. A list of all libraries used is in the section A.1. This does not apply to relative paths (e.g. `import getUser from './getUser'`).

C.1 ./frontend/src/actions/actionTypes.js

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

C.2 `./frontend/src/actions/auth/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) => {
10         dispatch(authUserResult(result));
11         if (result.success) {
12           saveToken(result.token);
13         }
14       })
15       .catch(e => showError(e.message));
16   };
17 }
```

C.3 `./frontend/src/actions/auth/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 }
```

C.4 `./frontend/src/actions/auth/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 }
```

C.5 `./frontend/src/actions/auth/getUser.js`

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


C.6 `./frontend/src/actions/auth/index.js`

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

C.7 `./frontend/src/actions/auth/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 }
```

C.8 `./frontend/src/actions/auth/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  }
```

C.9 `./frontend/src/actions/auth/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 }
```

C.10 `./frontend/src/actions/auth/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  }
```

C.11 `./frontend/src/actions/dashboard/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) {
8      dispatch(changeSidebarItem(index));
9      dispatch(push(item.path));
10   };
11 }
```

C.12 `./frontend/src/actions/dashboard/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  });
```

C.13 `./frontend/src/actions/dashboard/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  });
```


C.14 ./frontend/src/actions/dashboard/createAttribute.js

```
1  import { postAttribute } from '../../../utils/API';
2  import { showError } from '../../../other/showError';
3  import { receiveAttribute } from './receiveAttribute';
4  import { receiveEntry } from './receiveEntry';
5
6
7  export function createAttribute(model) {
8    return function (dispatch, getState) {
9      const state = getState().toJS();
10     const newId = (state.modelById[model].Attributes ? (state.modelById[model].Attributes.
        length + 1) : 1);
11     postAttribute({
12       model,
13       name: `attribute ${newId}`,
14       type: 'string',
15       required: false,
16       multiple: false,
17     })
18     .then((result) => {
19       if (result.success) {
20         result.entries.map(e => dispatch(receiveEntry(e)));
21         dispatch(receiveAttribute(result.attribute));
22       } else {
23         showError(result.error);
24       }
25     })
26     .catch(e =>
27       showError(e));
28   };
29 }
```

C.15 `./frontend/src/actions/dashboard/createEntry.js`

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

C.16 `./frontend/src/actions/dashboard/createModel.js`

```
1  import { postModel } from '../../utils/API';
2  import { showError } from '../../other/showError';
3  import { receiveModel } from './receiveModel';
4
5
6  export function createModel() {
7    return function (dispatch, getState) {
8      const state = getState().toJS();
9      const newId = state.serviceById[state.user.currentServiceId].Models &&
10        (state.serviceById[state.user.currentServiceId].Models.length + 1);
11
12      postModel({
13        service: state.user.currentServiceId,
14        name: `Model ${newId}`,
15      })
16        .then((result) => {
17          if (result.success) {
18            dispatch(receiveModel(result.model));
19          } else {
20            showError(result.error);
21          }
22        })
23        .catch(e =>
24          showError(e));
25    };
26  }
```

C.17 `./frontend/src/actions/dashboard/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  }
```

C.18 `./frontend/src/actions/dashboard/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  });
```

C.19 `./frontend/src/actions/dashboard/deleteEntry.js`

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

C.20 `./frontend/src/actions/dashboard/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  });
```

C.21 `./frontend/src/actions/dashboard/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  }
```


C.22 `./frontend/src/actions/dashboard/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  });
```

C.23 `./frontend/src/actions/dashboard/receiveAttribute.js`

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

C.24 `./frontend/src/actions/dashboard/receiveEntry.js`

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

C.25 `./frontend/src/actions/dashboard/receiveModel.js`

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

C.26 `./frontend/src/actions/dashboard/selectAttribute.js`

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

C.27 `./frontend/src/actions/dashboard/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 }
```

C.28 `./frontend/src/actions/dashboard/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  });
```

C.29 `./frontend/src/actions/dashboard/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, changes) {
6    return function (dispatch) {
7      dispatch(updateModelLocally(id, changes));
8      API.patchModel(id, changes)
9        .then((result) => {
10         if (!result.success) {
11           showError(result.error);
12         }
13       })
14        .catch(e => showError(e));
15    };
16  }
```


C.30 `./frontend/src/actions/dashboard/updateModelLocally.js`

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

C.31 `./frontend/src/actions/dashboard/updateService.js`

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

C.32 `./frontend/src/actions/dashboard/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 });
```

C.33 `./frontend/src/actions/dashboard/updateValue.js`

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

C.34 ./frontend/src/actions/dashboard/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  });
```

C.35 `./frontend/src/actions/other/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  }
```

C.36 ./frontend/src/actions/setup/analyseNaturalText.js

```
1 import { updateModelPreview } from './updateModelPreview';
2 import { updateNaturalText } from './updateNaturalText';
3 import { extractModelFromText } from '../../utils/API';
4 import { showError } from '../../other/showError';
5
6 export const ANALYSE_NATURAL_TEXT = 'ANALYSE_NATURAL_TEXT';
7
8
9 export function analyseNaturalText(text) {
10   return function (dispatch) {
11     dispatch(updateNaturalText(text));
12
13     return extractModelFromText(text)
14       .then(result => dispatch(updateModelPreview(result)))
15       .catch(e => dispatch(showError(e.message)));
16   };
17 }
```

C.37 `./frontend/src/actions/setup/analyseSpreadsheet.js`

```
1
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     return postAnalyzeSpreadsheet(file)
10       .then(result => dispatch(updateModelPreview(result)))
11       .catch(showError);
12   };
13 }
```


C.38 ./frontend/src/actions/setup/createService.js

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

C.39 ./frontend/src/actions/setup/index.js

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

C.40 `./frontend/src/actions/setup/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  }
```

C.41 `./frontend/src/actions/setup/nextScreen.js`

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

C.42 `./frontend/src/actions/setup/receiveService.js`

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

C.43 `./frontend/src/actions/setup/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 }
```

C.44 `./frontend/src/actions/setup/setServiceCreateMethod.js`

```
1
2
3  export const SET_SERVICE_CREATE_METHOD = 'SET_SERVICE_CREATE_METHOD';
4
5  export function setServiceCreateMethod(method) {
6    return {
7      type: SET_SERVICE_CREATE_METHOD,
8      method,
9    };
10 }
```

C.45 `./frontend/src/actions/setup/setServiceName.js`

```
1
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 }
```


C.46 `./frontend/src/actions/setup/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 }
```

C.47 ./frontend/src/actions/setup/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  }
```

C.48 ./frontend/src/components/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>Welcome to EasyAPI!</h1>
28      <p>If you are a new user, please create a new account by filling in the following fields.</p>
29      <p><p> If you are already a user, please enter your username and password.</p>
30      <form
31        action="/" onSubmit={(e) => {
32          e.preventDefault();
33          onSubmit({ username, password });
34        }} method="post"
35      >
36        <div style={style.field}>
37          <TextInput
38            name="username"
39            placeholder="Username"
40            onChange={username => onChange({ username })}
41            text={username}
42          />
43          <p style={style.error}>{errors.username}</p>
44        </div>
45        <div style={style.field}>
46          <TextInput
```

```
46         name="password"
47         placeholder="Password"
48         type="password"
49         onChange={password => onChange({ password })}
50         text={password}
51     />
52     <p style={style.error}>{errors.password}</p>
53 </div>
54 <div>
55     <Button type="submit" text="Next" />
56 </div>
57 </form>
58 </div>
59 );
60
61 AuthForm.propTypes = {
62     onSubmit: PropTypes.func.isRequired,
63     onChange: PropTypes.func.isRequired,
64     errors: PropTypes.array,
65     username: PropTypes.string.isRequired,
66     password: PropTypes.string.isRequired,
67 };
68
69 export default AuthForm;
```

C.49 ./frontend/src/components/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     {}]}>
38     {text}
39   </button>
40 );
41
42 Button.propTypes = {
43   text: PropTypes.string,
44   onClick: PropTypes.func,
45   isDisabled: PropTypes.bool,
46   type: PropTypes.string,
```

```
46  };  
47  
48  
49  export default Radium(Button);
```

C.50 ./frontend/src/components/dashboard/about/About.jsx

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

```

```
45     }
46
47     </div>
48 </div>;
49
50 About.propTypes = {
51   name: PropTypes.string,
52   meta: PropTypes.shape({
53     name: PropTypes.string,
54     url: PropTypes.string,
55     author: PropTypes.string,
56     public: PropTypes.bool,
57   }),
58   onChange: PropTypes.func,
59 };
60
61 export default About;
```


C.51 ./frontend/src/components/dashboard/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 { lightBorder } from '../StyleConstant';
6
7
8  const style = {
9    base: {
10      display: 'flex',
11    },
12    sidebar: {
13      minWidth: 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;
```

C.52 ./frontend/src/components/dashboard/entries/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      marginLeft: 0,
9      textAlign: 'center',
10     marginRight: 5,
11     width: 190,
12   },
13   item: {
14     borderRadius: 3,
15     cursor: 'pointer',
16     ':hover': {
17       backgroundColor: '#EEE',
18     },
19     border: 'none',
20     height: 45,
21     fontSize: 18,
22     ':focus': {
23       outline: 0,
24       border: 0,
25     },
26   },
27   first: {
28     width: 80,
29   },
30 };
31
32 const Column = ({ value, type, isItem, onChange, first = false }) =>
33   isItem ?
34     <input style={[style.base, style.item, first && style.first]} type={(type === 'string' ? 'text' : 'number')} value={value || ''} onChange={onChange} /> :
35     <div style={[style.base, first && style.first]}>{value}</div>;
36
37
38 Column.propTypes = {
39   value: PropTypes.any,
40   type: PropTypes.string,
41   isItem: PropTypes.bool,
42   first: PropTypes.bool,
43   onChange: PropTypes.func,
44 };
45
```

```
46 export default Radium(Column);
```

C.53 ./frontend/src/components/dashboard/entries/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  const Entries = ({ name, entries = [], attributes = [], headers = [], onSelected, onDelete,
    onCreate, onUpdate }) =>
23    <div style={style.base}>
24      <TopBar name={name} enableNew onNew={onCreate} />
25      <div style={style.main}>
26        <Tabs headers={headers} onSelected={onSelected} />
27        <RowHeader>
28          <Column key="headerid" value="ID" first />
29          {attributes.map(attr => <Column value={capitalizeString(attr.name)} key={`_${attr.id}
            rowheader`} />)}
30        </RowHeader>
31
32        {entries.map(entry =>
33          <Row key={`_${entry.id}row` } onDelete={() => onDelete(entry.realId)}>
34            <Column key={`_${entry.id}column` } value={entry.id} first />
35
36            {attributes.map(attr =>
37              <Column
38                key={`column_${entry.realId}x_${attr.id}`}
39                type={attr.type}
40                value={entry[attr.name] ? entry[attr.name].value : ''}
41                isItem
42                onChange={e => onUpdate(entry.realId, attr.id, e.target.value, entry[attr.name]
                  && entry[attr.name].id)}
43              />)}
44            </Column>
45          </Row>
46        )}
```

```
44         </Row>,
45     )}
46 </div>
47 </div>;
48
49 Entries.propTypes = {
50     name: PropTypes.string.isRequired,
51     entries: PropTypes.array.isRequired,
52     attributes: PropTypes.array.isRequired,
53     headers: PropTypes.array.isRequired,
54     onSelected: PropTypes.func,
55     onDelete: PropTypes.func,
56     onCreate: PropTypes.func,
57     onUpdate: PropTypes.func,
58 };
59
60 export default Entries;
```

C.54 ./frontend/src/components/dashboard/entries/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;
```

C.55 ./frontend/src/components/dashboard/entries/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);
```

C.56 `./frontend/src/components/dashboard/entries/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;
```


C.57 ./frontend/src/components/dashboard/entries/Tabs.jsx

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

C.58 ./frontend/src/components/dashboard/pages/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  import capitalizeString from '../../utils/capitalizeString';
8
9  const style = {
10    base: {
11      backgroundColor: Color.lighterGrey,
12      height: '100vh',
13      overflowY: 'auto',
14    },
15    page: {
16      backgroundColor: Color.whiteText,
17      margin: 20,
18      padding: 15,
19      borderRadius: 5,
20      border: '2px solid ${Color.lightGrey}',
21    },
22    title: {
23      margin: 0,
24      padding: 0,
25      fontWeight: 600,
26    },
27    method: {
28      color: Color.green,
29    },
30    description: {
31      padding: 0,
32      marginBottom: 0,
33    },
34    label: {
35    },
36    field: {
37      marginBottom: 10,
38      marginTop: 4,
39    },
40    action: {
41      marginTop: 5,
42    },
43  };
44
45
46  const Pages = ({ name, models = [], actions = [], onChange, urlPrefix }) => <div style={style.
```

```

    base}>
47   <TopBar name={name} />
48   {models.map((model, modelIndex) => <div style={style.page} key={`_${modelIndex}model`} >
49     <h3 style={style.title}>{capitalizeString(model.name)}</h3>
50     {actions.map(action => (
51       <div key={`_${action.label} ${model.id}`} style={style.action}>
52         <input
53           id={`_${action.label}-${model.id}`}
54           type="checkbox"
55           checked={model[action.prop]}
56           onChange={e => onChange(model.id, { [action.prop]: e.target.checked })}
57         />
58         <label
59           htmlFor={`_${action.label}-${model.id}`}
60           style={style.label}
61         >
62           {action.label}
63         </label>
64         {model[action.prop] && <p><b style={style.method}>{action.method}</b> {urlPrefix}{model
          .shortName}{action.suffix}</p>}}
65       </div>
66     )]}
67   </div>)}
68 </div>;
69
70 Pages.propTypes = {
71   name: PropTypes.string,
72   models: PropTypes.array,
73   actions: PropTypes.array,
74   onChange: PropTypes.func,
75   urlPrefix: PropTypes.string,
76 };
77
78 export default Pages;

```

C.59 ./frontend/src/components/dashboard/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   )}
18 </div>;
19
20 Sidebar.propTypes = {
21   items: PropTypes.arrayOf(PropTypes.shape({
22     name: PropTypes.string,
23     path: PropTypes.string,
24     selected: PropTypes.bool,
25   })),
26   onSelect: PropTypes.func,
27 };
28 export default Sidebar;
```

C.60 ./frontend/src/components/dashboard/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);
```

C.61 ./frontend/src/components/dashboard/structure/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   const leftPar = attribute.multiple ? '[' : '';
29   const rightPar = attribute.multiple ? ']' : '';
30   return `${prettify(attribute.name)}${attribute.required ? '*' : ''} (${leftPar}${attribute.type}${rightPar})`;
31 }
32
33 const Attribute = ({ attribute, onClick, enableInteractions }) => <div onClick={onClick} style
    =[{style.base, !enableInteractions} && style.noInteraction]>
34   {formatAttribute(attribute)}
35 </div>;
36
37 Attribute.propTypes = {
38   attribute: PropTypes.shape({
39     name: PropTypes.string,
40     multiple: PropTypes.bool,
41     required: PropTypes.bool,
42   }),
43   enableInteractions: PropTypes.bool,
44 };
```

45

46 export default Radium(Attribute);

C.62 ./frontend/src/components/dashboard/structure/DialogBox.jsx

```
1  import React from 'react';
2  import PropTypes from 'prop-types';
3  import { Color } from '../../../StyleConstant';
4  import TextInput from '../../../TextInput';
5  import capitalizeString from '../../../utils/capitalizeString';
6
7  const style = {
8    base: {
9      width: 210,
10     height: 310,
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     backgroundSize: '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     marginRight: 5,
41   },
42   cover: {
43     position: 'fixed',
44     width: '100%',
45     height: '100%',
46     top: 0,
```



```

91         <option value={option}>{capitalizeString(option)}</option>,
92     )}
93 </select>
94 </div>
95 );
96 }
97 case 'boolean': {
98     return (
99         <div key={attr.value} style={style.field}>
100             <label style={style.label} htmlFor={attr.value}><input type="checkbox" checked={
101                 object[attr.value]} onChange={e => onChange({ [attr.value]: e.target.checked })}
102                 />{attr.label}</label>
103             </div>
104         );
105     }
106 }
107
108 const DialogBox = ({ name, object, attributes, onChange, onClose, onDelete }) =>
109     <div>
110         <div style={style.cover} onClick={onClose} />
111         <div style={style.base}>
112             <div style={style.close} onClick={onClose} />
113             <h3 style={style.title}>{name && capitalizeString(name)}</h3>
114             {attributes.map(attr => field(attr, object, onChange))}
115             <div onClick={onDelete} style={style.delete}>Delete</div>
116         </div>
117     </div>;
118
119 DialogBox.propTypes = {
120     name: PropTypes.string,
121     object: PropTypes.object,
122     attributes: PropTypes.array,
123     onChange: PropTypes.func,
124     onClose: PropTypes.func,
125     onDelete: PropTypes.func,
126 };
127
128 export default DialogBox;

```

C.63 ./frontend/src/components/dashboard/structure/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      marginRight: 10,
21    },
22    title: {
23      margin: 0,
24      padding: '5px 0px',
25      textAlign: 'center',
26      borderRadius: 3,
27      ':hover': {
28        backgroundColor: '#EEE',
29      },
30      border: 'none',
31      ':focus': {
32        outline: 0,
33        border: 0,
34      },
35      fontSize: 20,
36      width: '100%',
37    },
38    close: {
39      position: 'absolute',
40      top: 8,
41      right: 6,
42    },
43    attributes: {
44      marginTop: 10,
45    },
46    newAttribute: {
```

```

47     textAlign: 'center',
48     backgroundColor: '#EEE',
49     margin: '6px 0',
50     borderRadius: 3,
51     padding: '5px 9px',
52     cursor: 'pointer',
53     color: 'black',
54     transition: '0.5s all',
55     fontSize: 18,
56     ':hover': {
57         backgroundColor: '#DDD',
58     },
59 },
60 };
61
62 const Model = ({ model, onClickAttribute, onDelete, onChange, onAttributeCreate,
    enableInteractions = true }) => <div style={style.base}>
63     <input disabled={!enableInteractions} style={style.title} value={capitalizeString(model.
        name)} onChange={e => onChange(model.id, e.target.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 </div>;
78
79 Model.propTypes = {
80     model: PropTypes.shape({
81         name: PropTypes.string,
82         id: PropTypes.number,
83         attributes: PropTypes.array,
84     }).isRequired,
85     onClickAttribute: PropTypes.func,
86     onDelete: PropTypes.func,
87     onChange: PropTypes.func,
88     onAttributeCreate: PropTypes.func,
89     enableInteractions: PropTypes.bool,

```

```
90  };  
91  
92  export default Radium(Model);
```

C.64 ./frontend/src/components/dashboard/structure/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      display: 'flex',
16      flexFlow: 'row wrap',
17      alignItems: 'flex-start',
18    },
19    model: {
20
21    },
22    newModel: {
23      textAlign: 'center',
24      backgroundColor: 'white',
25      border: '2px solid rgba(198, 198, 198, 0.34)',
26      borderRadius: 3,
27      width: 250,
28      padding: 5,
29      paddingBottom: 10,
30      fontSize: 27,
31      cursor: 'pointer',
32      color: 'black',
33      transition: '0.5s all',
34      ':hover': {
35        border: '2px solid rgba(198, 198, 198, 0.8)',
36      },
37    },
38  };
39
40  const attributes = [
41    {
42      value: 'name',
43      label: 'Name',
44      type: 'string',
45    },
46    {
```

```

47     value: 'type',
48     label: 'Type',
49     type: 'enum',
50     options: ['string', 'integer', 'float'],
51   },
52   {
53     value: 'multiple',
54     label: 'Multiple',
55     type: 'boolean',
56   },
57   {
58     value: 'required',
59     label: 'Required',
60     type: 'boolean',
61   },
62 ];
63
64 const Structure = ({
65   name,
66   models = [],
67   selectedAttribute,
68   onSelectAttribute,
69   onModelCreate,
70   onModelDelete,
71   onModelChange,
72   onAttributeCreate,
73   onAttributeDelete,
74   onAttributeChange,
75 }) => <div style={style.base}>
76   <TopBar name={name} onNew={() => onModelCreate()} enableNew />
77   {selectedAttribute && <DialogBox
78     name={selectedAttribute.name}
79     object={selectedAttribute}
80     attributes={attributes}
81     onChange={changes => onAttributeChange(selectedAttribute.id, changes)}
82     onDelete={() => onAttributeDelete(selectedAttribute.id)}
83     onClose={() => onSelectAttribute(undefined)}
84   />}
85   <div style={style.main}>
86     {models.map(model =>
87       <Model key={`model-${model.id}`} onChange={onModelChange} onDelete={onModelDelete}
88         onAttributeCreate={onAttributeCreate} onClickAttribute={onSelectAttribute} model={
89           model} />,
90     )}
91   </div>
92 </div>;

```



```
92  Structure.propTypes = {
93    name: PropTypes.string,
94    models: PropTypes.array,
95    selectedAttribute: PropTypes.object,
96    onSelectAttribute: PropTypes.func,
97    onModelCreate: PropTypes.func,
98    onModelDelete: PropTypes.func,
99    onModelChange: PropTypes.func,
100    onAttributeCreate: PropTypes.func,
101    onAttributeDelete: PropTypes.func,
102    onAttributeChange: PropTypes.func,
103  };
104
105  export default Radium(Structure);
```

C.65 ./frontend/src/components/dashboard/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, enableNew }) =>
25    <div style={style.base}>
26      <h2 style={style.h2}>{name}</h2>
27      {enableNew && <RoundButton text="add" onClick={onNew} />}
28    </div>;
29
30  TopBar.propTypes = {
31    name: PropTypes.string,
32    enableNew: PropTypes.bool,
33    onNew: PropTypes.func,
34  };
35
36  export default TopBar;
```

C.66 ./frontend/src/components/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;
```

C.67 ./frontend/src/components/HomePage.jsx

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

C.68 `./frontend/src/components/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;
```

C.69 ./frontend/src/components/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 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={`/${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);
```


C.70 ./frontend/src/components/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: `${Dimensions.transitionTime.normal} opacity`,
21     fontSize: 30,
22     backgroundSize: 'contain',
23     color: Color.whiteText,
24     ':hover': {
25       opacity: 0.8,
26     },
27     ':active': activeStyle,
28     ':focus': activeStyle,
29   },
30   isDisabled: {
31     pointerEvents: 'none',
32     backgroundColor: Color.grey,
33   },
34 };
35
36 const RoundButton = ({ text, onClick, isDisabled, color = Color.green, small = false }) => (
37   <button onClick={onClick} style={[style.base, isDisabled && style.isDisabled, {
38     backgroundColor: color, backgroundImage: 'url('/img/${text}.png')' }, small && { width:
39     25, height: 25 }]} />
40 );
41
42 RoundButton.propTypes = {
43   text: PropTypes.string,
44   onClick: PropTypes.func,
45   isDisabled: PropTypes.bool,
46   color: PropTypes.string,
```

```
45     small: PropTypes.bool,
46   };
47
48
49   export default Radium(RoundButton);
```

C.71 ./frontend/src/components/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
29              =`() => this.props.onSelect(service.id)` 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);
```

C.72 ./frontend/src/components/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);
```

C.73 ./frontend/src/components/setup/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);
```

C.74 ./frontend/src/components/setup/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);
```


C.75 ./frontend/src/components/setup/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);
```

C.76 ./frontend/src/components/setup/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 Model from '../dashboard/structure/Model';
7
8  const styles = {
9    nextButton: {
10      marginTop: 100,
11      float: 'right',
12    },
13    field: {
14      width: 850,
15      marginLeft: 'auto',
16      marginRight: 'auto',
17      textAlign: 'center',
18      marginTop: 100,
19    },
20    preview: {
21      display: 'flex',
22      flexFlow: 'row wrap',
23      alignItems: 'flex-start',
24      marginTop: 10,
25    },
26  };
27
28
29  const SetupNatural = ({ text, onChange, onDone, preview, nextEnabled }) => (
30    <div>
31      <div style={styles.field}>
32        <p>Please describe the various things and entities, <br />along with their properties and
33          relationships</p>
34        <div>
35          <TextInput
36            text={text}
37            onChange={onChange}
38            long
39          />
40        </div>
41        <div style={styles.preview}>
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 SetupNatural.propTypes = {
51   text: PropTypes.string,
52   onChange: PropTypes.func,
53   onDone: PropTypes.func,
54   preview: PropTypes.array,
55   nextEnabled: PropTypes.bool,
56 };
57
58 /* eslint-disable new-cap */
59 export default Radium(SetupNatural);
```

C.77 ./frontend/src/components/setup/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: 850,
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   preview: {
33     display: 'flex',
34     flexFlow: 'row wrap',
35     alignItems: 'flex-start',
36     marginTop: 10,
37   },
38 };
39
40 const SetupSpreadsheet = ({ onChange, onDone, preview, nextEnabled }) => (
41   <div>
42     <div style={styles.field}>
43       <div>
44         <Dropzone onDrop={onChange} style={styles.sheet}>
45           Drop a spreadsheet into this area
46         </Dropzone>
```

```

47     </div>
48     <div style={styles.preview}>
49         {preview && preview.map(a => <Model enableInteractions={false} model={a} />)}
50     </div>
51 </div>
52 <div style={styles.nextButton} >
53     <Button isDisabled={!nextEnabled} onClick={onDone} text="Next" />
54 </div>
55 </div>
56 );
57
58 SetupSpreadsheet.propTypes = {
59     onChange: PropTypes.func,
60     onDone: PropTypes.func,
61     preview: PropTypes.array,
62     nextEnabled: PropTypes.bool,
63 };
64
65 /* eslint-disable new-cap */
66 export default Radium(SetupSpreadsheet);

```

C.78 ./frontend/src/components/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}';
```

C.79 ./frontend/src/components/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);
```


C.80 `./frontend/src/containers/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 => ({
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;
```

C.81 ./frontend/src/components/dashboard/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;
```

C.82 ./frontend/src/containers/dashboard/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     entries.sort((a, b) => a.realId - b.realId);
58     attributes.sort((a, b) => a.id - b.id);
59
60     return {
61         name: service.name,
62         headers,
63         attributes,
64         entries,
65     };
66 };
67
68 const mapDispatchToProps = (dispatch) => {
69     const update = debounce((id, attr, value) => dispatch(updateValue(id, attr, value)), 1000);
70
71     return {
72         onSelected: id => dispatch(changeSelectedModel(id)),
73         onCreate: () => dispatch(createEntry()),
74         onDelete: id => dispatch(deleteEntry(id)),
75         onUpdate: (id, attr, value, valueId) => {
76             valueId && dispatch(updateValueLocally(id, valueId, value));
77             update(id, attr, value);
78         },
79     };
80 };
81
82 const EntriesContainer = connect(
83     mapStateToProps,
84     mapDispatchToProps,
85 )(Entries);
86
87 export default EntriesContainer;

```

C.83 ./frontend/src/containers/dashboard/PagesContainer.js

```
1  import { connect } from 'react-redux';
2  import { updateModel } from '../../../actions/dashboard/updateModel';
3  import Pages from '../../../components/dashboard/pages/Pages';
4
5
6  const mapStateToProps = (immutableState) => {
7    const state = immutableState.toJS();
8
9    const service = state.serviceById[state.user.currentServiceId];
10   const models = service.Models.map(id => state.modelById[id]);
11
12   const actions = [
13     {
14       label: 'Find',
15       prop: 'isFindEnabled',
16       method: 'GET',
17       suffix: '',
18     },
19     {
20       label: 'Find One',
21       prop: 'isFindOneEnabled',
22       method: 'GET',
23       suffix: '/:id',
24     },
25     {
26       label: 'Create',
27       prop: 'isCreateEnabled',
28       method: 'POST',
29       suffix: '',
30     },
31     {
32       label: 'Update',
33       prop: 'isUpdateEnabled',
34       method: 'PATCH',
35       suffix: '/:id',
36     },
37     {
38       label: 'Delete',
39       prop: 'isDeleteEnabled',
40       method: 'DELETE',
41       suffix: '/:id',
42     },
43   ];
44
45   const urlPrefix = 'http://localhost:9001/api/api/${state.user.username}/${service.shortName}
    /';
```

```
46
47   return {
48     name: service.name,
49     actions,
50     models,
51     urlPrefix,
52   };
53 };
54
55 const mapDispatchToProps = dispatch => ({
56   onChange: (id, changes) => {
57     dispatch(updateModel(id, changes));
58   },
59 });
60
61 const PagesContainer = connect(
62   mapStateToProps,
63   mapDispatchToProps,
64 )(Pages);
65
66 export default PagesContainer;
```

C.84 ./frontend/src/containers/dashboard/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;
```

C.85 ./frontend/src/containers/dashboard/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;
```

C.86 `./frontend/src/containers/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;
```

C.87 ./frontend/src/containers/ServiceListContainer.js

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

C.88 `./frontend/src/containers/setup/SetupContainer.js`

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

C.89 `./frontend/src/containers/setup/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;
```

C.90 ./frontend/src/containers/setup/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;
```

C.91 ./frontend/src/containers/setup/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    return {
12      text: state.getIn(['setup', 'naturalText']),
13      preview,
14      nextEnabled: preview && !!preview.length,
15    };
16  };
17
18  const mapDispatchToProps = dispatch => ({
19    onDone: () => dispatch(createService()),
20    onChange: text => dispatch(analyseNaturalText(text)),
21  });
22
23  const SetupNaturalContainer = connect(
24    mapStateToProps,
25    mapDispatchToProps,
26  )(SetupNatural);
27
28  export default SetupNaturalContainer;
```

C.92 ./frontend/src/containers/setup/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,
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;
```


C.93 ./frontend/src/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   const isStuck = !store.getState().getIn(['user', 'currentServiceId']) && (nextState.location.
34     pathname !== '/service/setup');
35
36   if (!isAuthenticated() || isStuck) {
37     return replace({
38       pathname: '/',
39     });
40   }
41
42   const r = () => render(
43     <Provider store={store}>
44       <Router history={history}>
45         <Route
```

```

46         path="/"
47         component={HomePageContainer}
48     />
49     <Route
50         path="/services"
51         component={ServiceListContainer}
52     />
53     <Route
54         path="/service/setup"
55         component={SetupContainer}
56         onEnter={requireAuth}
57     />
58     <Route
59         path="/service/dashboard"
60         component={Dashboard}
61         onEnter={requireAuth}
62     >
63         <Route
64             path="structure"
65             component={StructureContainer}
66         />
67         <Route
68             path="entries"
69             component={EntriesContainer}
70         />
71         <Route
72             path="pages"
73             component={PagesContainer}
74         />
75         <Route
76             path="about"
77             component={AboutContainer}
78         />
79         <IndexRedirect to="structure" />
80     </Route>
81
82 </Router>
83 </Provider>,
84 document.getElementById('root'),
85 );
86
87 r();
88 store.subscribe(r);

```

C.94 ./frontend/src/reducers/index.js

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

```

46 } from '../utils/setupScreens';
47
48
49 const {
50     naturalLanguage,
51     spreadsheet,
52     device,
53 } = createMethods;
54
55
56 const NEW_ID = '-1';
57
58
59 const defaultState = fromJS({
60     routing: {
61         locationBeforeTransitions: null,
62     },
63     user: {
64         currentServiceId: null,
65         username: '',
66         password: '',
67         authenticated: isAuthenticated(),
68         services: [],
69         token: getToken(),
70     },
71     dashboard: {
72         items: [
73             {
74                 name: 'Structure',
75                 path: '/service/dashboard/structure',
76                 selected: true,
77             },
78             {
79                 name: 'Entries',
80                 path: '/service/dashboard/entries',
81             },
82             {
83                 name: 'Pages',
84                 path: '/service/dashboard/pages',
85             },
86             {
87                 name: 'About',
88                 path: '/service/dashboard/about',
89             },
90         ],
91         selectedAttribute: null,
92         selectedModel: null,

```

```

93     },
94     setup: {
95         name: '',
96         screen: 'SERVICE_SETUP_SCREEN_NAME',
97         method: 'CREATE_METHOD_NATURAL_LANGUAGE',
98     },
99     serviceById: {
100     },
101     modelById: {
102     },
103     attributeById: {
104     },
105     entryById: {},
106     valueById: {},
107     endpointById: {},
108 });
109
110 function easyAPI(state = defaultState, action) {
111     switch (action.type) {
112         case LOCATION_CHANGE: {
113             return state.setIn(['routing', 'locationBeforeTransitions'], action.payload);
114         }
115         case NEW_SERVICE: {
116             return state
117                 .setIn(['user', 'currentServiceId'], NEW_ID)
118                 .setIn(['serviceById', NEW_ID], Map({
119                     id: NEW_ID,
120                     name: null,
121                     author: state.getIn(['user', 'name']),
122                     models: List(),
123                     endpoints: List(),
124                 })));
125         }
126         case NEXT_SCREEN: {
127             switch (state.getIn(['setup', 'screen'])) {
128                 case SERVICE_SETUP_SCREEN_NAME:
129                     return state.setIn(['setup', 'screen'], SERVICE_SETUP_SCREEN_METHOD);
130                 case SERVICE_SETUP_SCREEN_METHOD:
131                     const method = state.getIn(['setup', 'method']);
132                     switch (method) {
133                         case naturalLanguage:
134                             return state.setIn(['setup', 'screen'], SERVICE_SETUP_SCREEN_NATURAL);
135                         case spreadsheet:
136                             return state.setIn(['setup', 'screen'], SERVICE_SETUP_SCREEN_SPREADSHEET);
137                         case device:
138                             return state.setIn(['setup', 'screen'], SERVICE_SETUP_SCREEN_NAME);
139                         default:

```

```

140         return state;
141     }
142 }
143
144     return state;
145 }
146 case SET_SERVICE_NAME: {
147     return state
148         .setIn([
149             'setup',
150             'name',
151             ], capitalizeString(action.name));
152 }
153 case SET_SERVICE_CREATE_METHOD: {
154     const newState = state
155         .setIn([
156             'setup',
157             'method',
158             ], action.method);
159     switch (action.method) {
160         case naturalLanguage: {
161             return newState
162                 .setIn(['setup', 'naturalText'], '');
163         }
164         case spreadsheet: {
165             return newState
166                 .setIn(['setup', 'spreadsheet'], '');
167         }
168         case device: {
169             return newState
170                 .setIn(['setup', 'device'], Map({
171                     selectedDevice: '',
172                     flowDirection: null,
173                 }));
174         }
175         default: {
176             return newState;
177         }
178     }
179 }
180 case UPDATE_NATURAL_TEXT: {
181     return state.setIn(['setup', 'naturalText'], action.text);
182 }
183 case UPDATE_MODEL_PREVIEW: {
184     return state
185         .setIn(['setup', 'modelDefinitionPreview'], action.preview);
186 }

```

```
187     case SELECT_DEVICE: {
188         return state
189         .setIn(['setup', 'selectedDevice'], state.device);
190     }
191     case SET_DEVICE_FLOW_DIRECTION: {
192         return state
193         .setIn(['setup', 'flowDirection'], state.direction);
194     }
195     case RECEIVE_WEBHOOK_URL: {
196         return state
197         .setIn(['setup', 'webhookURL'], state.url);
198     }
199     case SETUP_DEVICE_QUERY: {
200         return state
201         .setIn(['setup', 'query'], Map({
202             url: action.url,
203             method: action.method,
204             attributes: action.attributes,
205             interval: action.interval,
206         }));
207     }
208     case UPDATE_USER: {
209         const { username, password } = action;
210
211         if (username && password) {
212             return state
213             .setIn(['user', 'username'], username)
214             .setIn(['user', 'password'], password);
215         } else if (username) {
216             return state
217             .setIn(['user', 'username'], username);
218         }
219         return state
220         .setIn(['user', 'password'], password);
221     }
222     case AUTH_USER_RESULT: {
223         return state
224         .setIn(['user', 'authenticated'], action.success)
225         .setIn(['user', 'errors'], action.errors)
226         .setIn(['user', 'token'], action.token);
227     }
228     case LOGOUT_USER: {
229         return state
230         .setIn(['user', 'authenticated'], false)
231         .setIn(['user', 'errors'], null)
232         .setIn(['user', 'token'], null);
233     }
```

```

234     case CHANGE_SIDEBAR_ITEM: {
235         return state
236             .setIn(
237                 ['dashboard', 'items'],
238                 state.getIn(['dashboard', 'items']).map((item, i) => item.set('selected', i ===
                    action.index)),
239             );
240     }
241     case RECEIVE_SERVICE_LIST: {
242         const services = action.services;
243
244         const entities = normalizeServices({ services }).entities;
245
246         const serviceIds = entities.services.defined.services;
247
248         const serviceById = entities.service || {};
249         const modelById = entities.model || {};
250         const attributeById = entities.attribute || {};
251         const entryById = entities.entry || {};
252         const valueById = entities.value || {};
253
254         return state
255             .setIn(['user', 'services'], fromJS(serviceIds))
256             .set('serviceById', fromJS(serviceById).merge(state.get('serviceById')))
257             .set('modelById', fromJS(modelById).merge(state.get('modelById')))
258             .set('attributeById', fromJS(attributeById).merge(state.get('attributeById')))
259             .set('entryById', fromJS(entryById).merge(state.get('entryById')))
260             .set('valueById', fromJS(valueById).merge(state.get('valueById')));
261     }
262     case RECEIVE_SERVICE: {
263         const entities = normalizeService(action.service).entities;
264
265         const serviceById = entities.service || {};
266         const modelById = entities.model || {};
267         const attributeById = entities.attribute || {};
268         const entryById = entities.entry || {};
269         const valueById = entities.value || {};
270
271         return state
272             .setIn(['user', 'currentServiceId'], action.service.id)
273             .set('serviceById', fromJS(serviceById).merge(state.get('serviceById')))
274             .set('modelById', fromJS(modelById).merge(state.get('modelById')))
275             .set('attributeById', fromJS(attributeById).merge(state.get('attributeById')))
276             .set('entryById', fromJS(entryById).merge(state.get('entryById')))
277             .set('valueById', fromJS(valueById).merge(state.get('valueById')));
278     }
279     case RECEIVE_ENTRY: {

```



```

280     const entities = normalizeEntry(action.entry).entities;
281
282     const model = action.entry.ModelId;
283
284     const valueById = entities.value || {};
285     const entryById = entities.entry || {};
286
287     const entryIdsPath = ['modelById', `${model}`, 'Entries'];
288
289     const existingEntries = state
290       .getIn(entryIdsPath);
291     const newEntries = existingEntries ?
292       existingEntries.toSet()
293       .union(fromJS([action.entry.id]).toSet())
294       .toList() :
295       fromJS([action.entry.id]);
296
297     return state
298       .setIn(entryIdsPath, newEntries)
299       .set('entryById', fromJS(state.get('entryById')).merge(entryById))
300       .set('valueById', fromJS(state.get('valueById')).merge(valueById));
301   }
302   case RECEIVE_MODEL: {
303     const entities = normalizeModel(action.model).entities;
304
305     const modelById = entities.model || {};
306
307     const modelIdsPath = ['serviceById', `${state.getIn(['user', 'currentServiceId'])}`, 'Models'];
308
309     return state
310       .setIn(modelIdsPath, state.getIn(modelIdsPath).push(action.model.id))
311       .set('modelById', fromJS(modelById).merge(state.get('modelById')));
312   }
313   case RECEIVE_ATTRIBUTE: {
314     const entities = normalizeAttribute(action.attribute).entities;
315
316     const attributeById = entities.attribute || {};
317
318     const attributeIdsPath = ['modelById', `${action.attribute.ModelId}`, 'Attributes'];
319
320     return state
321       .setIn(attributeIdsPath, (state.getIn(attributeIdsPath) || fromJS([])).push(action.attribute.id))
322       .set('attributeById', fromJS(attributeById).merge(state.get('attributeById')));
323   }
324   case SELECT_SERVICE: {

```

```

325     return state.setIn(
326       [
327         'user', 'currentServiceId',
328       ],
329       action.id,
330     );
331   }
332   case CHANGE_SELECTED_MODEL: {
333     return state.setIn(
334       [
335         'dashboard', 'selectedModel',
336       ],
337       action.id,
338     );
339   }
340   case DELETE_ENTRY_LOCALLY: {
341     const entries = ['modelById', `${action.entry.ModelId}`, 'Entries'];
342     return state
343       .deleteIn(['entryById', `${action.entry.id}`])
344       .setIn(entries, state.getIn(entries).filter(i => i !== action.entry.id));
345   }
346   case DELETE_MODEL_LOCALLY: {
347     const models = ['serviceById', `${state.getIn(['user', 'currentServiceId'])}`, 'Models'];
348     return state
349       .deleteIn(['modelById', `${action.id}`])
350       .setIn(models, state.getIn(models).filter(i => i !== action.id));
351   }
352   case DELETE_ATTRIBUTE_LOCALLY: {
353     const modelId = state.getIn(['attributeById', `${action.id}`, 'ModelId']);
354     const attributes = ['modelById', `${modelId}`, 'Attributes'];
355     return state
356       .deleteIn(['attributeById', `${action.id}`])
357       .setIn(attributes, state.getIn(attributes).filter(i => i !== action.id));
358   }
359   case UPDATE_VALUE_LOCALLY: {
360     return state
361       .setIn(['valueById', `${action.id}`, 'value'], action.value);
362   }
363   case UPDATE_SERVICE_LOCALLY: {
364     const servicePath = ['serviceById', `${state.getIn(['user', 'currentServiceId'])}`];
365
366     return state
367       .setIn(servicePath, state.getIn(servicePath).merge(fromJS(action.changes)));
368   }
369   case SELECT_ATTRIBUTE: {
370     return state
371       .setIn(['dashboard', 'selectedAttribute'], action.id);

```

```
372     }
373     case UPDATE_MODEL_LOCALLY: {
374         const modelPath = ['modelById', `${action.id}`];
375         return state
376             .setIn(modelPath, state.getIn(modelPath).merge(fromJS(action.changes)));
377     }
378     case UPDATE_ATTRIBUTE_LOCALLY: {
379         const path = ['attributeById', `${action.id}`];
380         return state
381             .setIn(path, state.getIn(path).merge(fromJS(action.changes)));
382     }
383     default:
384         return state;
385 }
386 }
387
388 export default easyAPI;
```

C.95 ./frontend/src/utils/API.js

```
1  import { getToken, saveToken } 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
21      if (json.token) {
22        saveToken(json.token);
23      }
24
25      return json;
26    };
27  }
28
29  export const req = (path, params) => curryReq(path)(params);
30
31  export const extractModelFromText = text => curryReq('/service/parseText')({ text });
32
33  export const authenticateUser = (username, password) => curryReq('/auth/login', false)({
34    username, password });
35
36  export const getUserInfo = () => curryReq('/auth/profile', true)();
37
38  export const getService = id => curryReq(`/service/${id}`, true, 'GET')({});
39
40  export const getServiceList = () => curryReq('/service', true, 'GET')();
41
42  export const postService = (name, models) => curryReq('/service', true, 'POST')({ name, models
43    });
44
45  export const postEntry = model => curryReq('/entry', true, 'POST')({ model });
46  export const deleteEntry = id => curryReq('/entry', true, 'DELETE')({ id });
47  export const updateValue = (entry, attribute, value) => curryReq('/value', true, 'PATCH')({
```

```

    entry, attribute, value });
45 export const updateService = (id, changes) => curryReq('/service/${id}', true, 'PATCH')(changes
    );
46
47 export const postModel = curryReq('/model', true, 'POST');
48 export const deleteModel = curryReq('/model', true, 'DELETE');
49 export const patchModel = (id, obj) => curryReq('/model/${id}', true, 'PATCH')(obj);
50
51 export const postAttribute = curryReq('/attribute', true, 'POST');
52 export const patchAttribute = obj => curryReq('/attribute/${obj.id}', true, 'PATCH')(obj);
53 export const deleteAttribute = curryReq('/attribute', true, 'DELETE');
54
55
56 export async function postAnalyzeSpreadsheet(file) {
57   const formData = new FormData();
58   formData.append('spreadsheet', file);
59
60   const headers = {
61   };
62
63   headers.Authorization = `bearer ${getToken()}`;
64
65   const response = await fetch('/api/service/parseSpreadsheet', {
66     method: 'POST',
67     headers,
68     body: formData,
69   });
70
71   const json = await response.json();
72
73   return json;
74 }

```

C.96 ./frontend/src/utils/Auth.js

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

C.97 `./frontend/src/utils/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;
```

C.98 ./frontend/src/utils/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;
```


C.99 ./frontend/src/utils/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);
```

C.100 `./frontend/src/utils/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';
```

C.101 `./frontend/src/index.css`

```
1
2 body {
3   font-family: sans-serif;
4   font-weight: 500;
5   letter-spacing: 0.4px;
6   font-size: 18px;
7   margin: 0;
8 }
```

C.102 ./frontend/package.json

```
1  {
2    "name": "easyapi",
3    "version": "0.1.0",
4    "private": true,
5    "devDependencies": {
6      "eslint": "^3.18.0",
7      "eslint-config-airbnb": "^14.1.0",
8      "eslint-plugin-import": "^2.2.0",
9      "eslint-plugin-jsx-a11y": "^4.0.0",
10     "eslint-plugin-react": "^6.10.3",
11     "react-scripts": "0.8.5"
12   },
13   "dependencies": {
14     "chai": "^3.5.0",
15     "enzyme": "^2.8.1",
16     "immutable": "^3.8.1",
17     "normalizr": "^3.2.2",
18     "prop-types": "^15.5.8",
19     "radium": "^0.18.1",
20     "react": "^15.4.2",
21     "react-addons-test-utils": "^15.5.1",
22     "react-dom": "^15.4.2",
23     "react-dropzone": "^3.12.3",
24     "react-redux": "^5.0.2",
25     "react-router": "^3.0.2",
26     "react-router-redux": "^4.0.8",
27     "redux": "^3.6.0",
28     "redux-immutable": "^3.1.0",
29     "redux-thunk": "^2.2.0",
30     "underscore": "^1.8.3"
31   },
32   "scripts": {
33     "start": "react-scripts start",
34     "build": "react-scripts build",
35     "test": "react-scripts test --env=jsdom",
36     "eject": "react-scripts eject"
37   },
38   "proxy": "http://localhost:9001"
39 }
```

C.103 ./backend/package.json

```
1  {
2    "devDependencies": {
3      "eslint": "^3.18.0",
4      "eslint-config-airbnb": "^14.1.0",
5      "eslint-plugin-import": "^2.2.0",
6      "eslint-plugin-jsx-a11y": "^4.0.0",
7      "eslint-plugin-react": "^6.10.3",
8      "neutrino": "^4.2.0",
9      "neutrino-preset-airbnb-base": "^4.2.0",
10     "neutrino-preset-mocha": "^4.2.0",
11     "neutrino-preset-node": "^4.0.1"
12   },
13   "dependencies": {
14     "bcrypt": "^1.0.2",
15     "body-parser": "^1.16.0",
16     "chai": "^3.5.0",
17     "compromise": "^7.0.28",
18     "eslint-plugin-async-await": "^0.0.0",
19     "express": "^4.14.0",
20     "immutable": "^3.8.1",
21     "isomorphic-fetch": "^2.2.1",
22     "jsonwebtoken": "^7.3.0",
23     "multer": "^1.3.0",
24     "natural": "^0.4.0",
25     "nlp_compromise": "^6.5.3",
26     "passport": "^0.3.2",
27     "passport-local": "^1.0.0",
28     "pg": "^6.1.2",
29     "radium": "^0.18.1",
30     "react": "^15.4.2",
31     "react-dom": "^15.4.2",
32     "react-redux": "^5.0.2",
33     "redux": "^3.6.0",
34     "redux-immutable": "^3.0.10",
35     "request-promise": "^4.1.1",
36     "sbd": "^1.0.12",
37     "sequelize": "^3.30.2",
38     "sequelize-cli": "^2.5.1",
39     "spacy-nlp": "^1.0.7",
40     "xlsx": "^0.9.10",
41     "underscore": "^1.8.3",
42     "mocha": "^3.2.0"
43   },
44   "scripts": {
45     "start": "neutrino start --presets neutrino-preset-node && node build/index.js",
46     "build": "neutrino build --presets neutrino-preset-node",
```

```
47     "test": "neutrino test --presets neutrino-preset-node neutrino-preset-mocha"
48   },
49   "config": {}
50 }
```

C.104 `./backend/src/components/natural.js`

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

```

47   for (const value of values) {
48       sum += Number(value);
49   }
50   return sum / values.length >= 0.5;
51 }
52
53 // Finds the existence of property. Returns string of 'required', 'optional', 'unknown'
54 function findIfPropertyIsRequired(prop, context) {
55     // https://en.wikipedia.org/wiki/Auxiliary_verb
56     const optionalKeywords = ['may', 'might', 'could', 'should', 'maybe', 'possible', 'possibly',
57                               'optionally', 'optional', 'ought'];
58
59     const requiredKeywords = ['must', 'needs', 'need', 'shall', 'will'];
60
61     const allRequiredInformation = [];
62
63     // Find if the relationship has monads attached
64     if (!context.modifiers || !context.modifiers.length) return false;
65     const monads = context.modifiers.filter(o => o.arc === 'aux');
66
67     for (const monad of monads) {
68         if (optionalKeywords.find(k => k === monad.lemma)) {
69             allRequiredInformation.push(false);
70         } else if (requiredKeywords.find(k => k === monad.lemma)) {
71             allRequiredInformation.push(true);
72         }
73     }
74
75     return decide(allRequiredInformation) || false;
76 }
77
78 // Finds if a property has multiple instances
79 function findIfPropertyHasMultiple(prop) {
80     const determiners = prop.modifiers ? prop.modifiers.filter(o => o.arc === 'det') : [];
81     const adjModifiers = prop.modifiers ? prop.modifiers.filter(o => o.arc === 'amod') : [];
82     const numModifiers = prop.modifiers ? prop.modifiers.filter(o => o.arc === 'nummod') : [];
83
84     const combined = determiners.concat(adjModifiers).concat(numModifiers);
85
86     // If the noun is plural then it will be multiple
87     if (prop.POS_fine === 'NNS') {
88         return true;
89     }
90
91     if (combined.length === 0) return false;
92
93     // Find all information related to upper bound
94     const allCardinalityInfo = [];

```



```

93   for (const modifier of combined) {
94     const singleKeywords = ['a', 'single', 'one'];
95     const multipleKeywords = ['many', 'multiple', 'several'];
96
97     if (modifier.arc === 'nummod') {
98       // Parse value of number
99       allCardinalityInfo.push(compromise.value(modifier.lemma).number > 1);
100    }
101
102    if (singleKeywords.find(k => k === modifier.lemma)) {
103      allCardinalityInfo.push(false);
104    } else if (multipleKeywords.find(k => k === modifier.lemma)) {
105      allCardinalityInfo.push(true);
106    }
107  }
108
109  return decide(allCardinalityInfo) || false;
110 }
111
112 function isContainment(relationship) {
113   const containmentWords = [
114     'have',
115     'include',
116     'incorporate',
117     'consist',
118     'comprise',
119     'contain',
120   ];
121
122   return containmentWords.find(w => w == relationship.lemma);
123 }
124
125 function buildPhrase(tree, transform = w => w, space = ' ') {
126   const othersInPhrase = tree.othersInPhrase;
127
128   if (othersInPhrase.length) {
129     return [tree, ...othersInPhrase].sort((a, b) => a.start - b.start).map(o => o.word).map(
130       transform).join(space);
131   }
132   return tree.word;
133 }
134
135 function propertyName(prop, relationship) {
136   let entity = '';
137
138   entity = buildPhrase(prop);

```

```
139     if (isContainment(relationship)) {
140         return entity;
141     }
142
143     const presentVerb = compromise.verb(relationship.word).to_present();
144
145     return `${presentVerb} ${entity}`;
146 }
147
148 const capitalizeWord = str => str.charAt(0).toUpperCase() + str.slice(1);
149
150 function propertyType(prop, entities = []) {
151     for (const entity of entities) {
152         if (entity.raw === prop.raw ||
153             entity.lemma === prop.lemma) {
154             return capitalizeWord(entity.lemma);
155         }
156     }
157
158     if (!prop || !prop.name) {
159         return 'string';
160     }
161
162     const propWords = prop.name.split(' ');
163
164     // Check criteria for number.
165     const integerKeywords = [
166         'number',
167         'integer',
168         'digit',
169         'digits',
170         'numbers',
171         'integers',
172     ];
173
174     const floatKeywords = [
175         'float',
176         'double',
177         'floats',
178         'doubles',
179         'decimal',
180         'decimals',
181         'amount',
182     ];
183
184     if (intersection(propWords, integerKeywords).length > 0) {
185         return 'integer';
```

```

186     }
187
188     if (intersection(propWords, floatKeywords).length > 0) {
189         return 'float';
190     }
191
192     return 'string';
193 }
194
195 function categoriseProp(prop, context, relationship, entities) {
196     const multiple = findIfPropertyHasMultiple(prop);
197
198     const type = propertyType(prop, entities);
199     const name = propertyName(prop, relationship, multiple);
200     const required = findIfPropertyIsRequired(prop, context);
201
202     return {
203         type,
204         name,
205         raw: prop.word,
206         lemma: prop.lemma,
207         required,
208         multiple,
209     };
210 }
211
212 function followModifiers(tree, condition) {
213     if (!tree || !tree.modifiers || tree.modifiers.length === 0) return [];
214
215     const [modifier] = tree.modifiers.filter(condition);
216     const deeperConjunctions = followModifiers(modifier, condition);
217
218     if (deeperConjunctions.length) {
219         return [
220             modifier,
221             ...deeperConjunctions,
222         ];
223     }
224     if (modifier) {
225         return [modifier];
226     }
227     return [];
228 }
229
230 function getConjunctions(object) {
231     return followModifiers(object, o => o.arc === 'conj');
232 }

```

```

233
234 function postprocess(modelStructure, entities) {
235   for (const models of modelStructure) {
236     for (const prop of models.attributes) {
237       prop.type = propertyType(prop, entities);
238     }
239   }
240 }
241
242 function flatMap(array, lambda) {
243   if (!array) return [];
244   return Array.prototype.concat.apply([], array.map(lambda));
245 }
246
247 function flatten(array) {
248   if (!array) return [];
249   return Array.prototype.concat.apply([], array);
250 }
251
252 function filterTree(tree, condition, depth = 0) {
253   if (!tree) return;
254   if (depth === 0) tree = JSON.parse(JSON.stringify(tree)); // Clone the tree
255
256   const modifiers = flatMap(
257     tree.modifiers,
258     m => filterTree(m, e => condition(e, depth, tree), depth + 1),
259   );
260
261   if (condition(tree)) {
262     if (modifiers.length < 1) {
263       // delete tree.modifiers;
264       return Object.assign(tree, {
265         modifiers: undefined,
266       });
267     }
268     return Object.assign(tree, {
269       modifiers,
270     });
271   }
272   return modifiers;
273 }
274
275 function assignNounPhrase(p) {
276   const preps = findAll(p, o => o.arc === 'prep');
277   const prepPhrases = preps.map(
278     o => [o, ...(o.modifiers.filter(m => m.arc === 'pobj'))],
279   );

```

```

280
281   const tags = ['compound', 'amod'];
282   const more = findAll(p, m => tags.includes(m.arc));
283
284   p.othersInPhrase = [...flatten(prephrases), ...more].sort((a, b) => a.start - b.start);
285
286   return p;
287 }
288
289 async function generateModelStructure(text) {
290   // Annotate raw text with POS and get dependency structure
291   const parseResult = await parse(text);
292   const modelStructure = [];
293   let allEntities = [];
294
295   for (const sentenceResult of parseResult.data) {
296     // Find relationships
297     const potentialRelationships = sentenceResult.parse_list
298       .filter(word => word.POS_fine.startsWith('V'));
299
300     // Build up tree of words to their place in parse tree
301     const tokens = sentenceResult.parse_list;
302     const cleanTree = filterTree(sentenceResult.parse_tree[0], m => m.POS_fine.startsWith('V')
303       || m.POS_fine.startsWith('N') || m.POS_fine === 'PRP');
304
305     const treeIndex = {};
306     const cleanTreeIndex = {};
307     tokens.forEach((token) => {
308       treeIndex[token.id] = find(sentenceResult.parse_tree[0], obj => obj.id === token.id);
309       cleanTreeIndex[token.id] = find(cleanTree, obj => obj.id === token.id);
310     });
311
312     for (const relationship of potentialRelationships) {
313       // First containment
314       let inTree = cleanTreeIndex[relationship.id];
315
316       const nounTree = filterTree(inTree, m => m.POS_fine.startsWith('N') || m.POS_fine === '
317         PRP');
318
319       const compareDepth = (a, b) => a.depth - b.depth;
320
321       if (!nounTree || nounTree.length < 1) continue;
322
323       // Find subject and object
324       const [subject] = nounTree.filter(o => o.arc.includes('subj')).sort(compareDepth);
325       const [object] = nounTree.filter(o => o.arc.includes('obj')).sort(compareDepth);
326
327       let attributes = [];
328       if (object) {

```

```

325         // This is the attributes
326         const fullObject = treeIndex[object.id];
327         attributes = [fullObject, ...getConjunctions(fullObject)];
328
329         attributes = attributes.map(assignNounPhrase);
330     }
331     let entities = [];
332     if (subject) {
333         // This is entities
334         const fullSubject = treeIndex[subject.id];
335         entities = [fullSubject, ...getConjunctions(fullSubject)];
336         allEntities = [...allEntities, ...entities];
337         allEntities = allEntities.map(assignNounPhrase);
338     }
339
340
341     inTree = treeIndex[relationship.id];
342
343     const attributesWithTypes = [];
344     for (const property of attributes) {
345         attributesWithTypes.push(categoriseProp(property, inTree, relationship, entities));
346     }
347
348     for (const entity of entities) {
349         const existingEntity = modelStructure.find(s => s.name === entity.lemma);
350
351         if (existingEntity) {
352             existingEntity.attributes = existingEntity.attributes.concat(attributesWithTypes);
353         } else {
354             modelStructure.push({
355                 name: entity.lemma,
356                 raw: entity.word,
357                 attributes: attributesWithTypes,
358             });
359         }
360     }
361 }
362 }
363
364 postprocess(modelStructure, allEntities);
365 return modelStructure;
366 }
367
368
369 const Natural = {
370     _find: find,
371     _findAll: findAll,

```

```
372   _findIfPropertyIsRequired: findIfPropertyIsRequired,
373   _findIfPropertyHasMultiple: findIfPropertyHasMultiple,
374   _filterTree: filterTree,
375   seperateSentences,
376   generateModelStructure,
377   parse,
378 };
379
380 export default Natural;
```

C.105 `./backend/src/components/parse.js`

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



```

47
48     modelDefinition.entries = entries;
49     modelDefinition.attributes = attributes;
50     allModelDefinitions.push(modelDefinition);
51 }
52 return Promise.resolve(allModelDefinitions);
53 }
54
55 // Given a array of type information, determines the type which encompasses all values
56 export function determineType(information) {
57     let type;
58     let multiple = false;
59     let required = true;
60
61     for (const value of information) {
62         if (value === null || value === undefined) {
63             required = false;
64             continue;
65         }
66
67         if (value.type === 'string') {
68             type = 'string';
69         } else if (value.type === 'float') {
70             if (type !== 'string') {
71                 type = 'float';
72             }
73         } else if (value.type === 'integer') {
74             if (type !== 'float' && type !== 'string') {
75                 type = 'integer';
76             }
77         }
78
79         if (value.multiple === true) {
80             multiple = true;
81         }
82     }
83
84     return {
85         type,
86         multiple,
87         required,
88     };
89 }
90
91 // Given a string, finds the most likely type
92 export function findType(raw) {
93     // If there is no value assume null

```

```

94     if ((raw === null) || (raw === undefined)) return null;
95
96     const string = raw.trim();
97     if (string.length === 0) return null;
98
99     const object = safeJSONParse(string);
100    const multiple = Array.isArray(object);
101    let type;
102
103    if (multiple) {
104        type = 'string';
105        type = determineType(object.map(findType)).type;
106
107        if (type.multiple) {
108            throw new Error('Multidimensional arrays are not supported!');
109        }
110    } else {
111        // Check for floats
112        if (/^-?((\d+\.\d*)|(\d+\.\d*))$/.test(string)) {
113            type = 'float';
114        } else if (/^-?(\d+)$/.test(string)) {
115            type = 'integer';
116        } else {
117            type = 'string';
118        }
119    }
120
121    return {
122        type,
123        multiple,
124        example: string,
125    };
126 }
127
128 function safeJSONParse(string) {
129     try {
130         return JSON.parse(string);
131     } catch (e) {
132         return null;
133     }
134 }
135
136 export function parseNaturalLanguage(text) {
137     return Natural.generateModelStructure(text);
138 }

```

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

C.107 `./backend/src/components/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 }
```

C.108 `./backend/src/config/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  }
```

C.109 `./backend/src/config/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;
```


C.110 `./backend/src/config/passport.js`

```
1  import passport from 'passport';
2  import { Strategy as LocalStrategy } from 'passport-local';
3  import jwt from 'jsonwebtoken';
4  import models from '../models';
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.validPassword(password))) {
23         return done(null, false, {
24           message: 'Incorrect password.',
25         });
26       }
27       user = foundUser;
28     } else {
29       // New user
30       user = await User.create({
31         username,
32         passwordHash: User.generateHash(password),
33       });
34     }
35
36     const payload = {
37       user: user.id,
38     };
39
40     // Change secret in production
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;
```

C.111 `./backend/src/nlp/index.py`

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

C.112 ./backend/src/nlp/spacyparse.py

NOTE: This file is copied from <https://github.com/kengz/spacy-nlp/blob/master/src/py/nlp.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
86 # Primary methods
87 #####
88
89 def parse_sentence(sentence):
90     '''
91     Main method: parse an input sentence and return the nlp properties.

```

```

92     '''
93
94     doc = nlp(sentence)
95     reply = OrderedDict([
96         ("text", doc.text),
97         ("len", len(doc)),
98         ("tokens", [token.text for token in doc]),
99         ("noun_phrases", [token.text for token in doc.noun_chunks]),
100        ("parse_tree", parse_tree(doc)),
101        ("parse_list", parse_list(doc))
102    ])
103    return reply
104
105
106 def parse(input):
107     '''
108     parse for multi-sentences; split and apply parse in a list.
109     '''
110     return [parse_sentence(sent) for sent in input.split("<#SENT_SEPERATOR#>")]

```

C.113 `./backend/src/index.js`

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

```
47     // render the error page
48     res.status(err.status || 500);
49     res.render('error');
50   });
51
52   app.listen(port);
53
54   console.log('Server is now running on port ${port}');
55 })
56 .catch(err => console.error(err));
```


C.114 ./backend/src/middleware/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/login')) {
8      return next();
9    }
10   if (req.originalUrl.startsWith('/api/api')) {
11     return next();
12   }
13
14   if (!req.headers.authorization) {
15     return res.status(401).end();
16   }
17
18   const token = req.headers.authorization.split(' ')[1];
19   return jwt.verify(token, 'secret', (err, decoded) => {
20     if (err) return res.status(401).end();
21
22     const userId = decoded.user;
23
24     return User.findById(userId)
25       .then((user) => {
26         if (user) {
27           req.user = user;
28           return next();
29         }
30         return res.status(401).end();
31       })
32       .catch(() => res.status(401).end());
33   });
34 }
```

C.115 `./backend/src/models/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
20    return Attribute;
21  }
```

C.116 `./backend/src/models/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
18    return Entry;
19 }
```

C.117 `./backend/src/models/index.js`

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

```
47 db.Sequelize = Sequelize;
48
49 export default db;
```

C.118 `./backend/src/models/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 }
```

C.119 `./backend/src/models/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
20    return Service;
21 }
```

C.120 `./backend/src/models/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         return bcrypt.compare(password, this.passwordHash);
21       },
22     },
23   });
24
25   return User;
26 }
```


C.121 `./backend/src/models/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 }
```

C.122 ./backend/src/routes/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     if (!user) {
30       return res.status(404).send({ success: false, message: 'This user (${username}) was not found!' });
31     }
32
33     const service = await Service.findOne({
34       where: {
35         shortName: serviceShortName,
36         UserId: user.id,
37       },
38     });
39
40     if (!service) {
41       return res.status(404).send({ success: false, message: 'This service (${serviceShortName}) was not found!' });
42     }
43
44     if (!service.isPublic) {
45       return res.status(403).send({ success: false, message: 'This service is not public!' });
46     }
47   }
48 }
```

```

45     }
46
47     const model = await Model.findOne({
48         where: {
49             shortName: modelShortName,
50             ServiceId: service.id,
51         },
52     });
53
54     if (!model) {
55         return res.status(404).send({ success: false, message: 'This model (${modelShortName})
56             was not found!' });
57     }
58
59     const attributes = await Attribute.findAll({
60         where: {
61             ModelId: model.id,
62         },
63     });
64
65     data = { user, service, model };
66
67     const forbiddenResponse = { success: false, message: 'This action is not public!' };
68
69     switch (method) {
70         case 'GET': {
71             if (id) {
72                 // Find One
73                 if (!model.isFindOneEnabled) {
74                     return res.status(403).send(forbiddenResponse);
75                 }
76
77                 const entry = await Entry.findOne({
78                     where: {
79                         index: id,
80                         ModelId: model.id,
81                     },
82                 });
83
84                 if (!entry) {
85                     return res.status(404).send({ success: false, message: 'This resource doesn\'t
86                         exist!' });
87                 }
88
89                 const values = await Value.findAll({
90                     where: {
91                         EntryId: entry.id,

```

```

90         },
91     });
92
93     const valueByAttributeId = object(
94         values.map(v => v.AttributeId), values.map(v => v.value),
95     );
96     const obj = {};
97     obj.id = entry.index;
98     for (const attribute of attributes) {
99         obj[attribute.name] = decode(valueByAttributeId[attribute.id], attribute.type);
100     }
101
102     data = obj;
103 } else {
104     // Find All
105     if (!model.isFindEnabled) {
106         return res.status(403).send(forbiddenResponse);
107     }
108
109     const entries = await Entry.findAll({
110         where: {
111             ModelId: model.id,
112         },
113     });
114     const values = await Value.findAll({
115         where: {
116             EntryId: entries.map(a => a.id),
117         },
118     });
119     data = { values, attributes, entries };
120
121     const objects = [];
122     for (const entry of entries) {
123         const obj = {};
124
125         const localValues = values.filter(v => v.EntryId === entry.id);
126         const valueByAttributeId = object(
127             localValues.map(v => v.AttributeId), localValues.map(v => v.value),
128         );
129         obj.id = entry.index;
130         for (const attribute of attributes) {
131             obj[attribute.name] = decode(valueByAttributeId[attribute.id], attribute.type);
132         }
133
134         objects.push(obj);
135     }
136

```

```
137         data = objects;
138     }
139     break;
140 }
141 case 'POST': {
142     // Create
143     if (!model.isCreateEnabled) {
144         return res.status(403).send(forbiddenResponse);
145     }
146     const newestEntry = await Entry.findOne({
147         where: {
148             ModelId: model.id,
149         },
150         order: 'index DESC',
151     });
152
153     const index = (newestEntry ? newestEntry.index : 0) + 1;
154
155     const entry = await Entry.create({
156         index,
157         ModelId: model.id,
158     });
159
160     const obj = {};
161     obj.id = entry.index;
162
163     const valuePromises = [];
164     for (const attribute of attributes) {
165         valuePromises.push(
166             Value.create({
167                 EntryId: entry.id,
168                 AttributeId: attribute.id,
169                 value: input[attribute.name],
170             }),
171         );
172         obj[attribute.name] = decode(input[attribute.name], attribute.type) || null;
173     }
174     await Promise.all(valuePromises);
175     data = obj;
176     break;
177 }
178 case 'PATCH': {
179     // Update
180     if (!model.isUpdateEnabled) {
181         return res.status(403).send(forbiddenResponse);
182     }
183 }
```

```

184     const entry = await Entry.findOne({
185       where: {
186         index: id,
187         ModelId: model.id,
188       },
189     });
190
191
192     if (!entry) {
193       return res.status(404).send({ success: false, message: 'This resource doesn\'t exist
194         !' });
195     }
196
197     const values = await Value.findAll({
198       where: {
199         EntryId: entry.id,
200       },
201     });
202
203     const valuePromises = [];
204     const valueByAttributeId = object(values.map(v => v.AttributeId), values.map(v => v));
205
206     const obj = {};
207     obj.id = entry.id;
208     for (const attribute of attributes) {
209       const newValue = input[attribute.name];
210       if (newValue) {
211         const oldValue = valueByAttributeId[attribute.id];
212         if (newValue !== oldValue.value) {
213           if (oldValue) {
214             // Update
215             valuePromises.push(
216               Value.update(
217                 { value: newValue },
218                 { where: { id: oldValue.id } },
219               ),
220             );
221           } else {
222             // Create
223             valuePromises.push(
224               Value.create({
225                 EntryId: entry.id,
226                 AttributeId: attribute.id,
227                 value: newValue,
228               }),
229             );

```

```
230         }
231     }
232     obj[attribute.name] = newValue;
233 } else {
234     obj[attribute.name] = valueByAttributeId[attribute.id].value;
235 }
236 }
237
238 await Promise.all(valuePromises);
239 data = obj;
240
241 break;
242 }
243 case 'DELETE': {
244     // Delete
245     if (!model.isDeleteEnabled) {
246         return res.status(403).send(forbiddenResponse);
247     }
248
249     const entry = await Entry.findOne({
250         where: {
251             index: id,
252             ModelId: model.id,
253         },
254     });
255
256
257     if (!entry) {
258         return res.status(404).send({ success: false, message: 'This resource doesn\'t exist'
259             !' });
260     }
261
262     await Value.destroy({
263         where: {
264             EntryId: entry.id,
265         },
266     });
267
268     const result = await Entry.destroy({
269         where: {
270             index: id,
271             ModelId: model.id,
272         },
273     });
274     data = Boolean(result);
275
276     break;
```

```
276     }
277     default: {
278         return res.status(400).send({ success: false, message: 'This action is not supported by
           EasyAPI!' });
279     }
280 }
281 } catch (e) {
282     return res.status(500).send({ success: false, error: e });
283 }
284
285 res.send({ success: true, data });
286 });
287
288
289 export default router;
```


C.123 `./backend/src/routes/attribute.js`

```
1  import { Router } from 'express';
2  import databaseModels from '../models';
3
4  const { 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 entries = await Entry.findAll({
27       where: {
28         ModelId: modelId,
29       },
30     });
31
32     await Value.bulkCreate(entries.map(e => ({
33       EntryId: e.id,
34       AttributeId: attribute.id,
35     })));
36
37     const newEntries = await Entry.findAll({
38       where: {
39         ModelId: modelId,
40       },
41       include: [{ all: true }],
42     });
43
44     const response = {
45       attribute,
46       entries: newEntries,
```

```

47         success: true,
48     };
49     return res.json(response);
50 } catch (e) {
51     return res.status(501).json({
52         error: e,
53         success: false,
54     });
55 }
56 });
57
58 router.patch('/:id', async (req, res) => {
59     const attributeId = req.param('id');
60
61     const toUpdate = {};
62
63     if (req.param('name')) {
64         toUpdate.name = req.param('name');
65     }
66     if (req.param('type')) {
67         toUpdate.type = req.param('type');
68     }
69
70     try {
71         const attribute = await Attribute.update(
72             toUpdate,
73             { where: { id: attributeId } },
74         );
75
76         return res.json({
77             attribute,
78             success: true,
79         });
80     } catch (e) {
81         return res.status(501).json({
82             error: e,
83             success: false,
84         });
85     }
86 });
87
88 router.get('/', async (req, res) => {
89     try {
90         const modelId = req.param('model');
91         const attributes = await Attribute.findAll({
92             where: {
93                 ModelId: modelId,

```

```
94     },
95     include: [{ all: true }],
96   });
97   return res.json({
98     attributes,
99     success: true,
100   });
101 } catch (e) {
102   return res.status(501).json({
103     error: e,
104     success: false,
105   });
106 }
107 });
108
109 router.delete('/', async (req, res) => {
110   try {
111     const id = req.param('id');
112     const result = await Attribute.destroy({
113       where: {
114         id,
115       },
116     });
117     return res.json({
118       result,
119       success: true,
120     });
121   } catch (e) {
122     return res.status(501).json({
123       error: e,
124       success: false,
125     });
126   }
127 });
128
129 export default router;
```

C.124 ./backend/src/routes/auth.js

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

```
47     return res.status(200).json(Object.assign({
48         success: true,
49         errors: {},
50     }, user));
51 })(req, res, next);
52 });
53
54 router.post('/profile', (req, res) => {
55     res.status(200).json(Object.assign({
56         success: true,
57         errors: {},
58         username: req.user.username,
59     }));
60 });
61
62
63 export default router;
```

C.125 ./backend/src/routes/entry.js

```
1  import { Router } from 'express';
2  import databaseModels from '../models';
3
4  const { 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;
```


C.126 ./backend/src/routes/model.js

```
1  import { Router } from 'express';
2  import databaseModels from '../models';
3  import { stringToShortName } from '../components/utils';
4
5  const { Model } = 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 modelId = req.param('id');
37
38   const toUpdate = {};
39
40   if (req.param('name')) {
41     toUpdate.name = req.param('name');
42     toUpdate.shortName = stringToShortName(toUpdate.name);
43   }
44
45   if (req.param('isFindEnabled')) {
46     toUpdate.isFindEnabled = req.param('isFindEnabled');
```

```

47   }
48
49   if (req.param('isFindOneEnabled')) {
50     toUpdate.isFindOneEnabled = req.param('isFindOneEnabled');
51   }
52
53   if (req.param('isCreateEnabled')) {
54     toUpdate.isCreateEnabled = req.param('isCreateEnabled');
55   }
56
57   if (req.param('isUpdateEnabled')) {
58     toUpdate.isUpdateEnabled = req.param('isUpdateEnabled');
59   }
60
61   if (req.param('isDeleteEnabled')) {
62     toUpdate.isDeleteEnabled = req.param('isDeleteEnabled');
63   }
64
65   try {
66     const model = await Model.update(
67       toUpdate,
68       { where: { id: modelId } },
69     );
70
71     return res.json({
72       model,
73       success: true,
74     });
75   } catch (e) {
76     return res.status(501).json({
77       error: e,
78       success: false,
79     });
80   }
81 });
82
83 router.get('/', async (req, res) => {
84   try {
85     const serviceId = req.param('service');
86     const model = await Model.findAll({
87       where: {
88         ServiceId: serviceId,
89       },
90       include: [{ all: true }],
91     });
92     return res.json({
93       model,

```

```
94         success: true,
95     });
96 } catch (e) {
97     return res.status(501).json({
98         error: e,
99         success: false,
100     });
101 }
102 });
103
104 router.delete('/', async (req, res) => {
105     try {
106         const id = req.param('id');
107         const result = await Model.destroy({
108             where: {
109                 id,
110             },
111         });
112         return res.json({
113             result,
114             success: true,
115         });
116     } catch (e) {
117         return res.status(501).json({
118             error: e,
119             success: false,
120         });
121     }
122 });
123
124
125 export default router;
```

C.127 ./backend/src/routes/service.js

```
1  import { Router } from 'express';
2  import multer from 'multer';
3  import { parseSpreadsheet, parseNaturalLanguage } from '../components/parse';
4  import { createService } from '../components/service';
5  import databaseModels from '../models';
6
7  const { Service } = 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     .catch(e => console.log(e));
19 });
20
21 router.post('/parseSpreadsheet', upload.single('spreadsheet'), (req, res) => parseSpreadsheet(
22   req.file)
23   .then(result => res.send(result)));
24
25 /* POST scratch. */
26 router.post('/', async (req, res) => {
27   const name = req.param('name');
28   const modelDefinitions = req.param('models');
29
30   try {
31     const service = await createService(
32       name,
33       modelDefinitions,
34       req.user.id,
35     );
36
37     const response = {
38       service,
39       success: true,
40     };
41     return res.json(response);
42   } catch (e) {
43     return res.status(501).json({
44       error: e,
45       success: false,
46     });
47   }
48 });
```

```
46     }
47   });
48
49   router.get('/', async (req, res) => {
50     try {
51       const services = await Service.findAll({
52         where: {
53           UserId: req.user.id,
54         },
55         include: [{ all: true, nested: true }],
56       });
57       return res.json({
58         services,
59         success: true,
60       });
61     } catch (e) {
62       return res.status(501).json({
63         error: e,
64         success: false,
65       });
66     }
67   });
68
69
70   router.get('/:id', async (req, res) => {
71     try {
72       const serviceId = req.param('id');
73       const service = await Service.findOne({
74         where: {
75           id: serviceId,
76           UserId: req.user.id,
77         },
78         include: [{ all: true, nested: true }],
79       });
80       return res.json({
81         service,
82         success: true,
83       });
84     } catch (e) {
85       return res.status(501).json({
86         error: e,
87         success: false,
88       });
89     }
90   });
91
92   router.patch('/:id', async (req, res) => {
```

```
93     try {
94         const serviceId = req.param('id');
95         const toUpdate = {};
96
97         if (req.param('name')) {
98             toUpdate.name = req.param('name');
99         }
100         if (req.body.isPublic !== undefined) {
101             toUpdate.isPublic = req.body.isPublic;
102         }
103         if (req.param('shortName')) {
104             toUpdate.shortName = req.param('shortName');
105         }
106
107         const service = await Service.update(
108             toUpdate,
109             { where: { id: serviceId } },
110         );
111         return res.json({
112             service,
113             success: true,
114         });
115     } catch (e) {
116         return res.status(501).json({
117             error: e,
118             success: false,
119         });
120     }
121 });
122
123
124 export default router;
```

C.128 ./backend/src/routes/value.js

```
1  import { Router } from 'express';
2  import databaseModels from '../models';
3
4  const { 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;
```

C.129 `./backend/test/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.');
```

```
35     });
36   });
37
38   describe('find', () => {
39     it('should find first modifier in tree which satisfies condition', () => {
40       const tree = {
41         lemma: 'runs',
42         pos: 'VERB',
43         modifiers: [
44           {
45             lemma: 'duck',
46             pos: 'NOUN',
```



```

47         modifiers: [
48             {
49                 lemma: 'yellow',
50                 pos: 'ADJ',
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
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 find = (tree, condition) => {
193       const result = [];
194       if (condition(tree)) {
195         result.push(tree);
196       }
197       if (tree.modifiers) {
198         tree.modifiers.forEach(modifier => {
199           result.push(...find(modifier, condition));
200         });
201       }
202       return result;
203     };
204
205     expect(find(tree, t => t.pos === 'VERB')).toHaveLength(1);
206     expect(find(tree, t => t.pos === 'NOUN')).toHaveLength(1);
207     expect(find(tree, t => t.pos === 'ADJ')).toHaveLength(2);
208   });
209 });

```

```

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     {
245         const prop = {
246             lemma: 'cat',
247             modifiers: [],
248         };
249
250         const context = {
251             lemma: 'play',
252             modifiers: [
253                 { lemma: 'might', arc: 'aux' },
254             ],
255         };
256
257         const result = Natural._findIfPropertyIsRequired(prop, context);
258         expect(result).to.equal(false);
259     });
260
261 it('should deduce a property is required when there are more required keywords than
262     optional keywords', () => {
263     const prop = {
264         lemma: 'cat',
265         modifiers: [],
266     };
267
268     const context = {
269         lemma: 'play',
270         modifiers: [
271             { lemma: 'might', arc: 'aux' },
272             { lemma: 'needs', arc: 'aux' },
273             { lemma: 'must', arc: 'aux' },
274         ],
275     };
276
277     const result = Natural._findIfPropertyIsRequired(prop, context);
278     expect(result).to.equal(true);
279 });
280
281 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).to.equal(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).to.equal(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
402             pet. Toy has a name. Pet likes a toy.';
403
404         const modelStructure = await Natural.generateModelStructure(text);
405
406         const expected = [
407             {
408                 name: 'pet',
409                 raw: 'pet',
410                 attributes: [
411                     {
412                         type: 'string',
413                         name: 'name',
414                         raw: 'name',
415                         lemma: 'name',
416                         required: false,
417                         multiple: false,
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     attributes: [
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          attributes: [
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  });
```

C.130 `./backend/test/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
11   describe('findType', () => {
12     it('should return null if no value is supplied', () => {
13       const result = findType();
14       expect(result).to.equal(null);
15     });
16
17     it('should return float if string contains one dot', () => {
18       const result = findType('5.3');
19       expect(result.type).to.equal('float');
20     });
21
22     it('should return string if string contains more than one dot', () => {
23       const result = findType('5.3.3');
24       expect(result.type).to.equal('string');
25     });
26
27     it('should return integer if string is only digits', () => {
28       const result = findType('432');
29       expect(result.type).to.equal('integer');
30     });
31
32     it('should return string otherwise', () => {
33       const result = findType('This is a sentence. ');
34       expect(result.type).to.equal('string');
35     });
36   });
37
38   describe('determineType', () => {
39     it('should return string if one of the types is string', () => {
40       const result = determineType([
41         {
42           type: 'string',
43           multiple: 'false',
44         },
45         {
46           type: 'float',
```

```

47         multiple: 'false',
48     },
49     {
50         type: 'integer',
51         multiple: 'false',
52     },
53     ]);
54
55     expect(result.type).to.equal('string');
56     expect(result.required).to.equal(true);
57 });
58
59 it('should return float if one of the types is float and there is no string', () => {
60     const result = determineType([
61         {
62             type: 'float',
63             multiple: 'false',
64         },
65         {
66             type: 'integer',
67             multiple: 'false',
68         },
69     ]);
70
71     expect(result.type).to.equal('float');
72     expect(result.required).to.equal(true);
73 });
74
75 it('should return integer if one of the types is integer and there is no string', () => {
76     const result = determineType([
77         {
78             type: 'integer',
79             multiple: 'false',
80         },
81         {
82             type: 'integer',
83             multiple: 'false',
84         },
85     ]);
86
87     expect(result.type).to.equal('integer');
88     expect(result.required).to.equal(true);
89 });
90
91 it('should not be required if one of the types is not required', () => {
92     const result = determineType([
93         {

```

```
94         type: 'string',
95         multiple: 'false',
96     },
97     null,
98 ]);
99
100     expect(result.type).to.equal('string');
101     expect(result.required).to.equal(false);
102 });
103 });
104 });
```