# Структура проекта

## Описание директорий

**assets** - контент, ресурсы используемые в проекте (иконки, изображения);

**docs** - документация проекта, предоставленная заказчиком;

**js** - исходный код Javascript - язык управления элементами на сайте;

**libs** - исходный код Javascript используемых библиотек;

**php** - исходный код PHP;

**styles** - исходный код CSS - описание внешнего вида веб-страницы;

**templates** - HTML разметка, которая используется для структурирования и отображения веб-страницы и её контента;

**tools** - дополнительные конфигурационные файлы, утилиты и прочее.

## Структура директории **js**

### Файл **api.js**

*Модуль, формирования обращений к API*

#### request(url, requestData)

Шаблон запроса для get и post методов по api

Параметры:

url : string - Url для обращения

requestData : array\_like, optional - Данные для отправки по post

#### *get(url)*

*Get-запрос по api*

Параметры:

url : string - Url для обращения

#### *post(url,* d*ata)*

*Post-запрос по api*

Параметры:

url : string - Url для обращения

data : array\_like - Данные для отправки по post

#### *cache(type, page, url)*

*Запрос на получение данных из кэша*

Параметры:

type: string - Тип текущего отображения(tree, table)

page: integer, optional - номер “страницы” в кэше для подгрузки в таблицу

url: string, optional - Url, который является ключом в кэше, значение - ответ по

данному url

#### 

#### getAll(page)

*Получить значение из кэша по странице*

Параметры:

page: integer, string - Страница данных, необходимая для подгрузки. Может принимать значение ‘all’ для получения всех данных из кэша.

#### 

#### postObject(typeObject)

*Формирование данных о предприятии и отправка их по post методу*

Параметры:

typeObject: string - Тип объекта для формирования и отправки данных

### Файл **change\_view.js**

*Модуль смены вида отображения данных*

#### *removeChilds(element)*

*Используется для очистки блока с таблицей/деревом. Также подходит для удаления любых дочерних элементов*

Параметры:

element: DOM-object - Элемент с детьми, которых необходимо удалить

#### *changeView()*

*Переключение табличного и древовидного представлений*

### Файл **contextmenu.js**

*Модуль формирование контекстного меню для работы с таблицей. Удаление/Добавление столбцов, сохранение текущего отображения*

#### add\_delete\_column()

*Формирование контекстного меню и его функционала*

#### deleteColumn(title)

*Удаление столбца из таблицы*

Параметры:

title: string - Имя заголовка

#### addColumn(title)

*Добавление столбца в таблицу*

Параметры:

title: string - Имя заголовка

#### saveView()

*Сохранение текущего отображения в БД*

### Файл **createNew.js**

*Модуль создания нового объекта*

#### *createNewObject()*

*Создание нового подразделения/предприятия*

#### *createSelectList(selectlist)*

*Создание выпадающего списка*

Параметры:

selectlist: array\_like - Массив данных для заполнения выпадающего списка

#### *postNew()*

*Вызов необходимых функций для отправки post запроса с данными об объекте*

#### *check\_null(start, end)*

*Проверка на пустоту заполняемых значений*

Параметры:

start: integer - Начальное значение идентификатора поля заполнения данных

end: integer - Конечное значение идентификатор поля заполнения данных

### Файл **dynamic.js**

*Модуль, отвечающий за динамическую подгрузку данных в табличном отображении*

#### *checkLastElement()*

*Выполнение проверок и подгрузка данных, если пользователь долистал до последнего элемента*

### Файл **edit\_form.js**

*Модуль режима редактирования объекта*

#### *edit()*

*Переключение на вид редактирования, вызов функций для отправки отредактированных элементов*

#### postChangedObject()

#### Вызов функция для отправки данных об объекте

#### *selectList(name)*

*Формирование выпадающих списков для формы редактирования*

Параметры:

name: string - Идентификатор поля, для которого необходимо формирование выпадающего списка

#### *getFieldName(obj)*

*Поиск поля, где написано имя для выпадающего списка*

Параметры:

obj: array\_like - Объект, в котором необходимо найти имя

#### *editView()*

*Генерация полей режима редактирования*

#### *findID(name)*

*Поиск ID объекта*

Параметры:

name: DOM-object - Объект, в котором необходимо найти id

#### *removeID()*

*Удалить ID из заголовка формы просмотра/редактирования*

### Файл **excel.js**

*Модуль экспорта табличного представления в .xlsx*

#### createMockBody()

*Формирование тела экспортной таблицы*

#### toExcel()

*Функция onclick для кнопки, обработка данных и условий экспорта*

### Файл **functions\_for\_buttons.js**

*Модуль, содержащий callback функции для кнопок основного окна*

#### *close\_edit()*

*Закрыть форму редактирования/просмотра*

#### *undo\_edit()*

*Перейти в режим просмотра из режима редактирования без изменений*

#### *reload\_cache()*

*Обновление кэша*

#### *delete\_object()*

*Запрос на удаление объекта*

#### *accept\_filters()*

*Принять выбранные фильтры и сортировки*

#### *filter\_open()*

*Открыть блок фильтров*

#### *filter\_close()*

*Скрыть блок фильтров*

*drop\_filters()*

*Сбросить фильтры*

### Файл **search.js**

*Модуль, отвечающий за поиск и отображение искомых данных*

#### search()

*Функция поиска, валидация ввода, onclick для кнопки “Найти”*

#### clear(input)

*Очистка поля ввода и возврат к виду таблицы/дерева по умолчанию*

Параметры:

input: DOM-object - Поле поиска

#### inputEvent(element)

*Обработчик событий при вводе в строку поиска*

Параметры:

element: DOM-object - Поле поиска

### Файл **show\_edit.js**

*Модуль, содержащий методы отображения информации по существующему объекту*

#### *selectItemEnterprise(id)*

*Загрузить форму просмотра для предприятия с идентификатором id*

Параметры:

id: integer - Идентификатор предприятия для просмотра

#### *selectItemDivision(id)*

*Загрузить форму просмотра для подразделения с идентификатором id*

Параметры:

id: integer - Идентификатор подразделения для просмотра

#### *findName(id, obj)*

*Поиск имени для подставления по ID в выпадающих списках*

Параметры:

id: integer - Идентификатор необходимого имени

obj: array\_like - Массив объектов для поиска

#### *getType(object)*

*Определение типа объекта (подразделение, предприятие)*

Параметры:

object: DOM-object - Объект для определения

#### *blockSelect(freeze)*

*Запрет на генерацию событий указателя*

Параметры:

freeze: bool - Текущее состояние запрета

### Файл **table.js**

*Модуль формирования табличного представления данных*

#### *isAllow(item)*

*Проверка содержимого объекта данных на соответствие фильтрам*

Параметры:

item: array\_like - Объект данных, полученный по api

#### *createHead()*

*Формировании заголовков таблицы*

#### *createBody(data, tbody, backlightPattern)*

*Формирование тела таблицы, либо добавление новых данных в существующее тело таблицы*

Параметры:

data: array\_like - Данные для добавления в таблицу

tbody: DOM-object, optional - Существующее тело таблицы

backlightPattern: string, optional - Искомая строка в подстроке

#### *table(data, backlightPattern)*

*Формирование таблицы*

Параметры:

data: array\_like, optional - Данные для добавления в таблицу

backlightPattern: string, optional - Подсветка искомой подстроки

### Файл **tree.js**

*Модуль формирования древовидного представления данных*

#### clearTreeData()

*Очистка информации о наличии детей у объекта, очистка кэша с объектами данных, очистка хранилища сессии*

#### *getData(url)*

*Получить данные из кэша для дерева*

Параметры:

url: string - Url для получения детей объектов

#### *createElemWithAttr(item, attributes)*

*Создание DOM-объекта, с заданными атрибутами*

Параметры:

item: string - Тип объекта, который создается

attributes: object - Свойства, присваемые к объекту

#### *hasChilds(id)*

*Есть ли дочерние элементы у обьекта*

Параметры:

id: integer - Идентификатор родителя

#### *openChilds(element, button)*

*Загрузка детей объекта, раскрытие/закрытие узла*

Параметры:

element: DOM-object - Родитель

button: DOM-object - Кнопка раскрытия/закрытия детей

#### *createLi(classItem, item, searchPattern)*

*Создание элементарного List Item элемента дерева и его необходимого заполнения*

Параметры:

classItem: string - Класс узла дерева (root, child)

item: object - Объект данных

searchPattern: string, optional - Пользовательская поисковая строка, используется для подсветки вхождений

#### *createUl(classItem, arr, searchPattern)*

*Создание списка элементов дерева*

Параметры:

classItem: string - Класс узла дерева (root, child)

arr: array\_like - Объект данных

searchPattern: string, optional - Пользовательская поисковая строка, используется для подсветки вхождений

#### *createView(className, arr, searchPattern)*

*Создание корневых элементов для дерева*

Параметры:

className: string - Класс узла дерева (root, child)

arr: array\_like, optional - Объект данных

searchPattern: string, optional - Пользовательская поисковая строка

#### *tree(data, backlightPattern)*

*Создание дерева*

Параметры:

data: array\_like, optional - Данные для отображения

backlightPattern: string, optional - Искомая строка

### Файл **sortTable.js**

*Модуль, отвечающий за сортировку таблицы*

#### sort(header)

*Сортировка таблицы по указанному столбцу*

Параметры:

header: DOM-element - Элемент заголовка столбца таблицы

### Файл **urls.js**

*Модуль конфигурации списка url*

## Структура директории **libs**

### Файл **dragtable.js**

*Модуль, отвечающий за функционал перетаскивания столбцов таблицы*

### Файл **jquery-ui.min.js, jquery.min.js**

*Библиотека jquery*

### Файл **selectize.min.js**

*Модуль для выпадающего списка с динамическим поиском*

### Файл **tableToExcel.js**

*Модуль для формирования таблицы excel*

## Структура директории **php**

### Файл **api.php**

*Модуль, содержащий реализацию методов обращения к API заказчика*

#### get( $url )

*Запрос данных по url*

Параметры:

$url: адрес сервера, ресурса

#### post( $url, $data = array() )

*Отправка данных по url*

Параметры:

$url: string - Адрес сервера, ресурса

$data: array - Набор передаваемых данных

### Файл **auth.php**

*Модуль, содержащий интеграцию с системой аутентификации заказчика*

### Файл **cache.php**

*Модуль, реализующий помещение шифрованных данных в кэш*

#### cache( &$data )

*Помещение data в локальное хранилище, осуществляя шифрование данных*

Параметры:

$data: array - Набор данных

### Файл **get\_cache.php**

*Модуль, осуществляющий дешифровку и получение данных из кэша в зависимости от типа отображения (табличное или древовидное). Для табличного - из кэша происходит запрос по 25 элементов, для древовидного - запрос кэшированного ответа от сервера по API url ключу*

### Файл **reload\_cache.php**

*Модуль, осуществляющий запрос актуальных данных и их обновление в кэше*

### Файл **sort\_cache.php**

*Модуль, осуществляющий сортировку кэша*

#### sort\_high\_low( $left, $right )

*Быстрая сортировка кэша по возрастанию/убыванию*

Параметры:

$left: int - левая “граница” словаря

$right: int - правая “граница” словаря

#### 

#### sort\_null\_not( $left, $right )

*Быстрая сортировка кэша по пустые/непустые*

Параметры:

$left: int - левая “граница” словаря

$right: int - правая “граница” словаря

## Структура директории **templates**

### Файл **index.php**

*Отображение главной страницы, содержит определение всех компонент сайта.*

Заголовок файла (тег head) - Содержит ссылки на основные CSS стили

Содержание веб-страницы (тег body)

Блок "container-fluid" - Блок шапки сайта

Блок "container" - Элементы панели управления

Блок "mySidebar" - Боковая панель от контента для размещения параметров фильтрации и сортировки

Блок "main" - Основной функционал сайта

Блок "container\_information" - Заголовки функционала сайта

Блок "block\_edit" - Форма для создания нового объекта

#### tree(data, backlightPattern)

*Создание дерева*

Параметры:

data: array\_like, optional - Данные для отображения

backlightPattern: string, optional - Искомая строка

## Структура директории **styles**

### Файл **css.css**

*Общие стили*

## Структура директории **tools**

### Файл **README.md**

*Руководство по проекту*