СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc390948204)

[1. Выбор состава технических и программных средств 5](#_Toc390948205)

[1.1. Выбор языка программирования 5](#_Toc390948206)

[2. Разработка базы данных 6](#_Toc390948207)

[2.1. Создание 6](#_Toc390948208)

[2.2. Создание таблиц 6](#_Toc390948209)

[2.2.1. ER-диаграмма 6](#_Toc390948210)

[2.2.1. Структура 7](#_Toc390948211)

[3. Разработка Web-приложения 13](#_Toc390948212)

[3.1. Создание эскиза 13](#_Toc390948213)

[3.2 Меню 13](#_Toc390948214)

[3.3 Главная страница 14](#_Toc390948215)

[3.3.1 Страница клиентов 14](#_Toc390948216)

[3.3.2 Страница клиента 16](#_Toc390948217)

[3.3.3 Страница тренажера 18](#_Toc390948218)

[3.3.4 Страница библиотеки 20](#_Toc390948219)

[3.3.4.1 Добавление изображений 22](#_Toc390948220)

[3.3.4.2 Добавление тегов 23](#_Toc390948221)

[3.3.5 Страница элемента 24](#_Toc390948222)

[3.3.6 Навигация 24](#_Toc390948223)

[3.4 Справочники 26](#_Toc390948224)

[3.4.1 Типы библиотек 26](#_Toc390948225)

[3.4.2 Типы элементов 28](#_Toc390948226)

[3.4.3 Теги элементов 29](#_Toc390948227)

[3.5. Поиск 30](#_Toc390948228)

[3.5.1 Поиск библиотек 30](#_Toc390948229)

[3.5.2 Поиск элементов 33](#_Toc390948230)

[4. Руководство пользователя 37](#_Toc390948231)

[5. Экономическая часть 43](#_Toc390948232)

[ЗАКЛЮЧЕНИЕ 47](#_Toc390948233)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 48](#_Toc390948234)

ВВЕДЕНИЕ

Экспериментальный научно-исследовательский и методический центр “Моделирующие системы” (ЭНИМЦ МС) основан 2 октября 1991 года. Одним из основных направлений деятельности ЭНИМЦ МС является тренажеростроение для обучения и поддержания квалификации оперативного персонала атомных и тепловых станций, а также для обучения студентов.

Для создания тренажера требуется динамические объекты (ДО), которые аналогичны настоящим элементам на атомной станции (АС). Каждый динамический объект взаимодействует с другим объектом, тем самым имитируя реальные действия АС, для отработки внештатных ситуаций.

Тренажеров множество и для автоматизации процесса созданы динамические объекты, которые объединяются в библиотеки. У каждого динамического объекта есть свой номер, состоящий из четырех цифр, первые две цифры определяют номер библиотеки, в которой находится ДО, последние две цифры определяют номер объекта в библиотеки. Библиотеки динамических объектов могут использоваться в разных тренажерах. Для каждого тренажера могут создаваться новые динамические объекты. За 20 лет было создано более 300 объектов и появилась проблема хранения информации ДО.

1. Выбор состава технических и программных средств

1.1. Выбор языка программирования

Базы Данных динамических объектов создано виде web-приложение.

Для создания web-приложения на стороне сервера использовались языки программирования: PHP , SQL.

На стороне клиента для реализации графического пользовательского интерфейса ( GUI ) использовались языки разметки и стилей: HTML , CSS. Для формирования и обработки запросов, создания интерактивного и независимого от браузера интерфейса: JavaScript.

1.2 Выбор системы управления базами данных.

Выбранная реляционная СУБД MySQL является свободным программным обеспечением. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. MySQL является решением для малых и средних приложений. Для администрирования СУБД MySQL выбрано PHPMyAdmin - веб-приложение с открытым кодом, написанное на языке PHP.

1.3. Обоснование

Для решения проблемы хранения динамических объектов требуется:

1. Единая база данных.

2. Просмотр одновременно с нескольких ПК.

3. Удобное редактирование и удаление.

4. Создание поиска.

Все данные требования возможно решить с помощью web-приложения.

2. Разработка базы данных

2.1. Создание

Создать базу данных можно как с помощью SQL запроса, так и с помощью web-интерфейса в phpMyAdmin, как показано на рисунке 1.

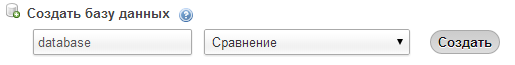


Рисунок 1

Тип базы данных InnoBD , сравнение utf8\_general\_ci.

2.2. Создание таблиц

База данных состоит из 10 таблиц, в ней хранится информация о динамических объектах. Данной базе имеются связи «один ко многим» , «многие ко многим». База данных имеет иерархическое дерево: клиент, тренажер, библиотека, элемент.

2.2.1. ER-диаграмма

ER-диаграмма – модель данных, позволяющая описывать концептуальные схемы предметной области. На рисунке 2 представлена модель данных проекта.

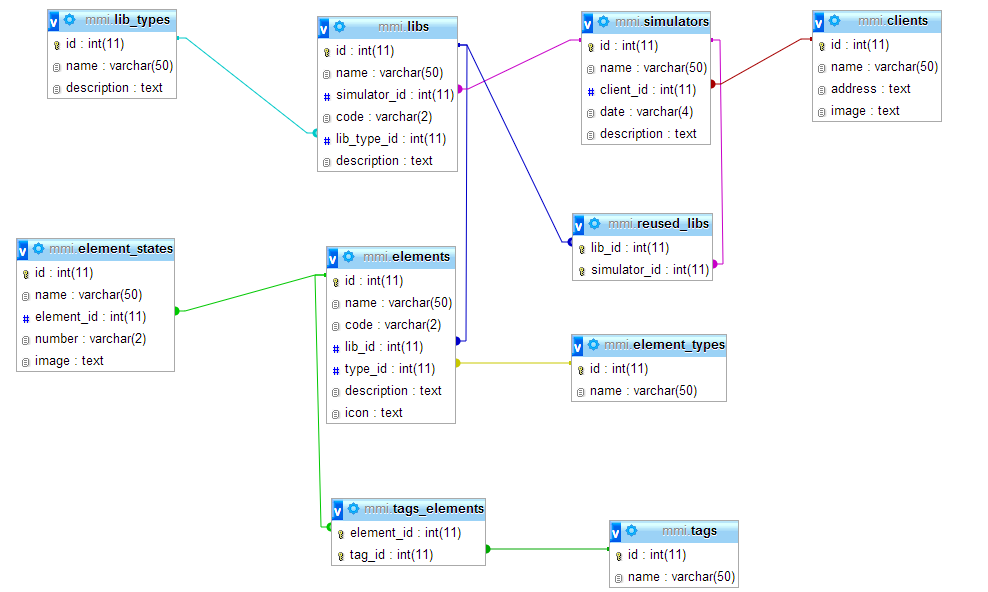


Рисунок 2

2.2.1. Структура

Таблица клиентов - “clients” в которой занесены все заказчики тренажеров.

SQL запрос:

*CREATE TABLE IF NOT EXISTS `clients` (*

*`id` int(11) NOT NULL AUTO\_INCREMENT,*

*`name` varchar(50) NOT NULL,*

*`address` text,*

*`image` text,*

*PRIMARY KEY (`id`)*

*)*

Результат предоставлен на рисунке 3.



Рисунок 3

Id – номер записи, Name – название клиента, Address – адрес клиента, image – картинка клиента.

Таблица тренажеров – “simulators” связанна с таблицей клиентов.

SQL запрос:

*CREATE TABLE IF NOT EXISTS `simulators` (*

*`id` int(11) NOT NULL AUTO\_INCREMENT,*

*`name` varchar(50) NOT NULL,*

*`client\_id` int(11) NOT NULL,*

*`date` varchar(4) NOT NULL,*

*`description` text,*

*PRIMARY KEY (`id`),*

*KEY `client\_id` (`client\_id`),*

*KEY `client\_id\_2` (`client\_id`)*

*)*

Результат предоставлен на рисунке 4.



Рисунок 4

Id – номер записи , Name – название тренажера , Client\_id – номер клиента , date , description – описание.

Таблица библиотеки – “libs” связанна с таблицей тренажеров.

SQL запрос:

*CREATE TABLE IF NOT EXISTS `libs` (*

*`id` int(11) NOT NULL AUTO\_INCREMENT,*

*`name` varchar(50) NOT NULL,*

*`simulator\_id` int(11) NOT NULL,*

*`code` varchar(2) NOT NULL,*

*`lib\_type\_id` int(11) DEFAULT NULL,*

*`description` text,*

*PRIMARY KEY (`id`),*

*KEY `simulator\_id` (`simulator\_id`),*

*KEY `lib\_type\_id` (`lib\_type\_id`)*

*)*

Результат предоставлен на рисунке 5.



Рисунок 5

Id – номер записи , Name – название библиотеки, Simulator\_id – номер тренажера , Code - код библиотеки , Lib\_type\_id – номер типа,description – описание.

Таблица используемых библиотек в тренажерах – “ reused\_libs” . Тип связи «многие ко многим».

SQL запрос:

*CREATE TABLE IF NOT EXISTS `reused\_libs` (*

*`lib\_id` int(11) NOT NULL,*

*`simulator\_id` int(11) NOT NULL,*

*PRIMARY KEY (`lib\_id`,`simulator\_id`),*

*KEY `simulator\_id` (`simulator\_id`)*

*)*

Результат предоставлен на рисунке 6.

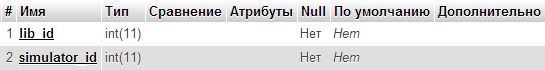


Рисунок 6

Lib\_id – номер библиотеки , Simulator\_id – номер тренажера.

Таблица типов библиотеки – “lib\_types” связана с таблицей библиотек.

SQL запрос:

*CREATE TABLE IF NOT EXISTS `lib\_types` (*

*`id` int(11) NOT NULL AUTO\_INCREMENT,*

*`name` varchar(50) NOT NULL,*

*`description` text NOT NULL,*

*PRIMARY KEY (`id`)*

*)*

Результат предоставлен на рисунке 7.

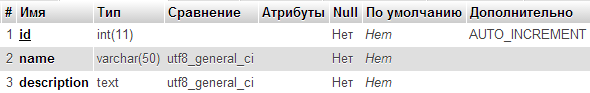


Рисунок 7

Id – номер записи , Name – название типа, Description – описание.

Таблица элементов – “elements” связана с таблицей библиотек.

SQL запрос:

*CREATE TABLE IF NOT EXISTS `elements` (*

*`id` int(11) NOT NULL AUTO\_INCREMENT,*

*`name` varchar(50) NOT NULL,*

*`code` varchar(2) NOT NULL,*

*`lib\_id` int(11) NOT NULL,*

*`type\_id` int(11) DEFAULT NULL,*

*`description` text,*

*`icon` text,*

*PRIMARY KEY (`id`),*

*KEY `type\_id` (`type\_id`),*

*KEY `lib\_id` (`lib\_id`)*

*)*

Результат предоставлен на рисунке 8.

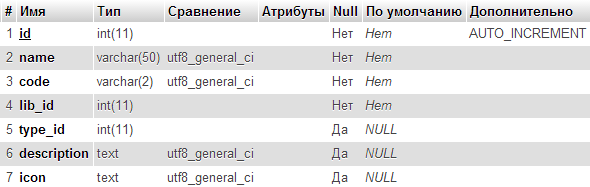


Рисунок 8

Id – номер записи , Name – название элемента, Code – код элемента , Lib\_id – номер библиотеки, Type\_id – номер типа, Description - описание, Icon - картинка.

Таблица типов элемента – “element\_types” связана с таблицей элементов.

SQL запрос:

*CREATE TABLE IF NOT EXISTS `element\_types` (*

*`id` int(11) NOT NULL AUTO\_INCREMENT,*

*`name` varchar(50) NOT NULL,*

*PRIMARY KEY (`id`)*

*)*

Результат предоставлен на рисунке 9.

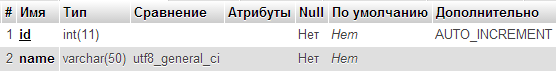


Рисунок 9

Id – номер записи , Name – название типа.

Таблица состояний элемента – “element\_states” связана с таблицей элементов.

SQL запрос:

*CREATE TABLE IF NOT EXISTS `element\_states` (*

*`id` int(11) NOT NULL AUTO\_INCREMENT,*

*`name` varchar(50) NOT NULL,*

*`element\_id` int(11) NOT NULL,*

*`number` varchar(2) DEFAULT NULL,*

*`image` text,*

*PRIMARY KEY (`id`),*

*KEY `element\_id` (`element\_id`)*

*)*

Результат предоставлен на рисунке 10.



Рисунок 10

Id – номер записи , Name – название состояния, Element\_id – номер элемента,

Number – номер позиции, Image – картинка.

Таблица тегов элементов – “tags\_elements” связана с таблицей элементов и тегов. Тип связи «многие ко многим».

SQL запрос:

*CREATE TABLE IF NOT EXISTS `tags\_elements` (*

*`element\_id` int(11) NOT NULL,*

*`tag\_id` int(11) NOT NULL,*

*PRIMARY KEY (`element\_id`,`tag\_id`),*

*KEY `tag\_id` (`tag\_id`)*

*)*

Результат предоставлен на рисунке 11.

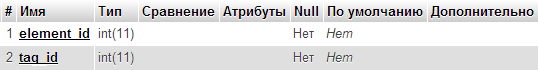


Рисунок 11

Element\_id – номер элемента, Tag\_id – номер тега.

Таблица тегов – “tags” связана с таблицей тегов элементов.

SQL запрос:

*CREATE TABLE IF NOT EXISTS `tags` (*

*`id` int(11) NOT NULL AUTO\_INCREMENT,*

*`name` varchar(50) NOT NULL,*

*PRIMARY KEY (`id`)*

*)*

Результат предоставлен на рисунке 12.

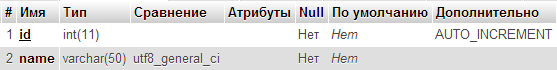


Рисунок 12

Id – номер записи , Name – название тега.

3. Разработка Web-приложения

3.1. Создание эскиза

Требуется создать интуитивно понятный интерфейс. Создание интерфейса началось с главного меню. Далее создание главной страницы, “клиенты” на которой отображались заказчики. При нажатии на заказчика осуществляется переход на страницу “тренажеры” на которой отображен выбранный клиент и его тренажеры. При нажатии на тренажер осуществляется переход на страницу “библиотеки” на которой отображен выбранный тренажер, созданные и добавленные библиотеки. При выборе библиотеки осуществляется переход на вложенные в него элементы, страница элементов. При нажатии на элемент осуществляется переход на страницу элемента на которой отображаются все его характеристики.

3.2 Меню

Меню делится на две части: Справочники и Поиск. В Справочнике можно добавить, просмотреть и редактировать: типы библиотек, типы элементов, теги элементов. В части Поиска осуществляется поиск по двум направлениям: поиск библиотек и поиск элементов. На рисунке 13 представлено меню приложения.

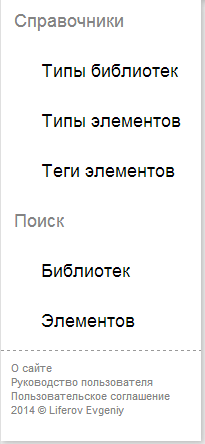


Рисунок 13

3.3 Главная страница

За главную страницу отвечает файл “ index.php ”. В нем определяется какую страницу вывести пользователю в зависимости от данного запроса.

Пример кода index.php:

if ($c == NULL)

{

echo "<div class='nav'><div><a href='/?add=1'>

<div class='nav\_add'></div></a></div><a href='../'>Клиенты</a></div>";

if ($insert != NULL) {include 'clients\_insert.php';} else {

if ($add != NULL) { include 'clients\_add.php'; } else {

include 'clients.php'; }}}

В index.php передается переменная “ c ” которую проверяем на NULL. Переменную получаем из адресной строки браузера.

Код запроса переменной:

$c = empty($\_GET['c'])?'':(int)$\_GET['c'];

3.3.1 Страница клиентов

На данной странице отображены все заказчики, которые занесены в Базу Данных. На каждой “плитке” дана краткая информация по клиенту: название, адрес, картинка. На рисунке 14 отображение клиентов.

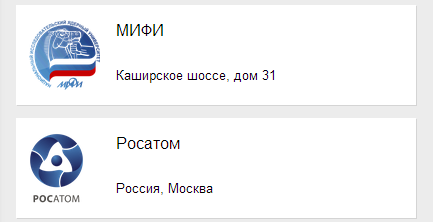


Рисунок 14

Для вывода клиентов осуществляется запрос к базе данных:

SQL запрос:

*SELECT*

*`clients`.`id` AS `client\_id`,*

*`clients`.`name` AS `client\_name`,*

*`clients`.`image` AS `client\_image`,*

*`clients`.`address` AS `client\_address`*

*FROM `clients`*

На данной странице можно добавить клиентов. Для этого требуется нажать на кнопку плюс, рисунок 15.

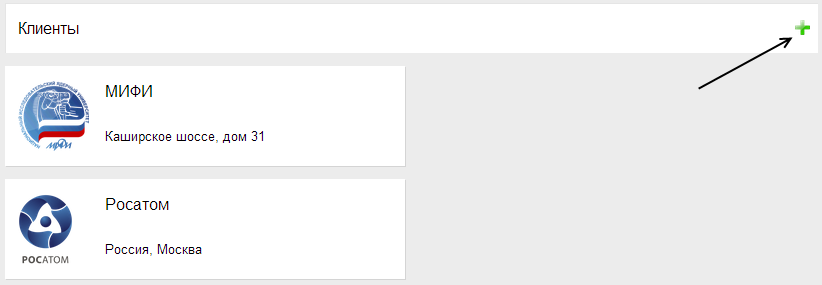


Рисунок 15

При нажатии в адресной строке добавиться еще одна переменная add, которая не будет равно NULL. При данной ссылке: http://ssl.com/?add=1 ,отрабатывает файл clients\_add.php .

Даны две строки для заполнения текстом и одна для добавления картинки.

При выборе картинки срабатывает JavaScript который отображает выбранное изображение ( рисунок 16 ).

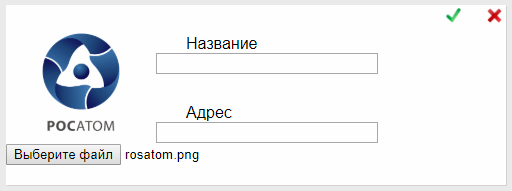


Рисунок 16

На форме имеются две кнопки, галочка – сохранить, крестик – отменить редактирование.

При нажатии на крестик происходит переход на страницу клиентов. При нажатии на галочку происходит SQL запрос добавления в Базу Данных. Необходимо добавить ссылку на картинку в БД и поместить файл в определенную папку.

Определяем пришедшие данные с формы. Создаем переменную time и заносим в нее время сервера. Заносим эти данные в БД. Выбираем данные из БД которые равны переменной time. Присваиваем названию файла id выбранного поля, определяем расширение файла и присоединяем к названию. Импортируем файл в назначенную директорию. Производим SQL запрос на обновление БД, где вместо time присваиваем путь к файлу. Пример кода на рисунке 17.

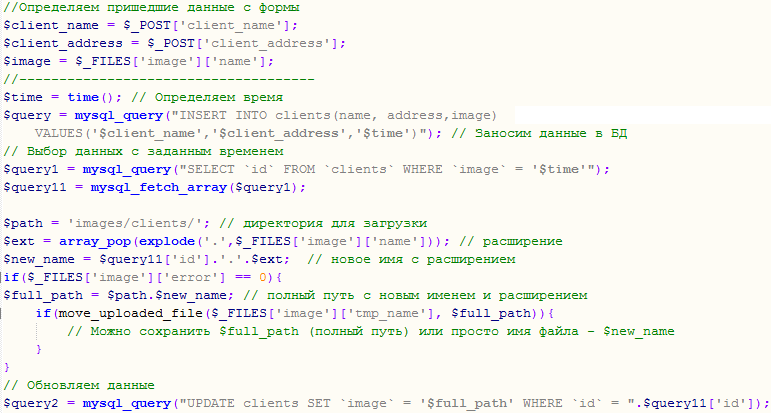


Рисунок 17

3.3.2 Страница клиента

На данной странице отображается выбранный клиент и вложенные в него тренажеры, также можно отредактировать клиента и добавить новый тренажер.

Редактирование клиента осуществляется наведение на плитку, где появится иконка настройки. При редактирование будут доступны поля для редактирование: изображение, название, адрес. Для сохранение требуется нажать на кнопку сохранить она изображена как галочка. При сохранение будет отправлен SQL запрос на обновление. Также имеются две кнопки корзинка ( удалить ), крестик ( закрыть ).

При нажатии на крестик, закроется страница редактирования, а при нажатии на корзинку будет выполнен SQL запрос на удаление. Форма редактирования представлена на рисунке 18.

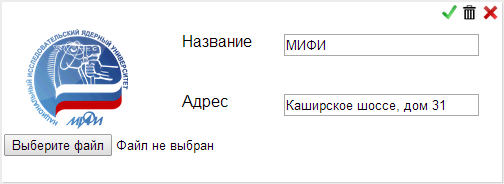


Рисунок 18

Ниже плитки клиента расположены тренажеры. У которых выведены такие данные как: название, год создания, краткое описание ( рисунок 19 ).

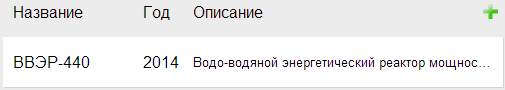


Рисунок 19

Слева расположена кнопка в форме полюса, при нажатии на которую можно добавить тренажер.

На странице добавления тренажера есть три поля для заполнения: название, дата, описание. Имеется две кнопки: сохранить и закрыть. Форма заполнения данных представлена на рисунке 20.



Рисунок 20

Для добавления получаются данные с формы и с помощью SQL запроса добавляются в БД ( рисунок 21 ).

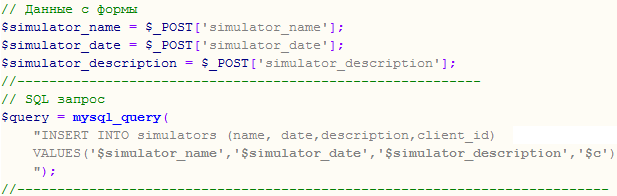


Рисунок 21

После добавление выведется информация добавление тренажера, добавлены или произошла ошибка. ( рисунок 22 ).

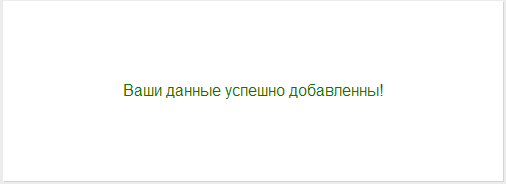


Рисунок 22

3.3.3 Страница тренажера

На данной странице отображается полная информация о выбранном тренажере. Ниже расположены, находящиеся в нем библиотеки, они делятся на два типа: библиотеки, созданные в данном тренажере и используемые из других тренажеров.

Созданные библиотеки – это библиотеки, которые относятся к данному тренажеру и могут быть использованы в других тренажерах только как используемые.

Используемые библиотеки – это библиотеки, которые созданы в другом тренажере и используются для создания тренажера.

Каждая библиотека имеет свой код состоящий из двух цифр.

На рисунке 23 изображена страница тренажера.

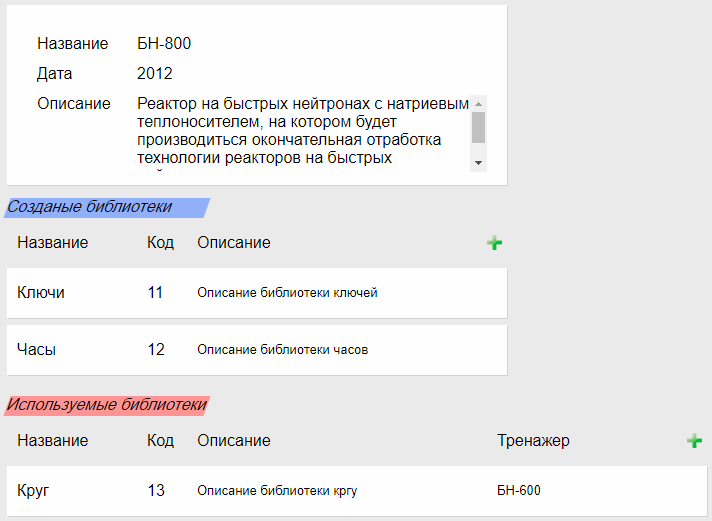


Рисунок 23

Для создания библиотеки нужно нажать на кнопку в виде плюса в разделе “Созданные библиотеки”. Появится форма, в которой будет четыре поля: название, код, тип и описание. Для сохранения нажать на кнопку в виде галочки, для отмены на крестик. В данной форме имеется поле “тип” – это тип библиотеки, он выбирается из списка предложенных типов. Вид формы на рисунке 24.

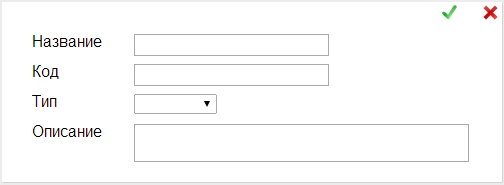


Рисунок 24

При сохранении выполняется SQL запрос на добавление. При отмене переход на страницу тренажера.

Добавление используемых библиотек используется кнопка в виде плюса. При нажатии на кнопку появится форма, в которой есть список доступных для использования библиотек ( рисунок 25 ).

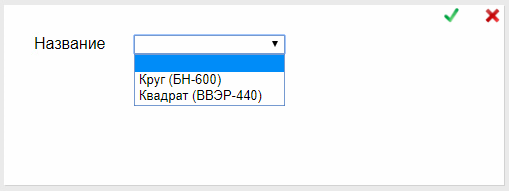


Рисунок 25

При сохранении, осуществляется добавлен

ие в таблице БД reused\_libs. В эту таблицу заносится номер используемого тренажера и номер используемой библиотеки.

Чтобы удалить используемую библиотеку, нужно навести курсор на удаляемую библиотеку и справа появится кнопка в виде крестика при нажатии на который произойдет удаление данной библиотеки ( рисунок 26 ).



Рисунок 26

3.3.4 Страница библиотеки

На данной странице отображается полная информация о выбранной библиотеке. Ниже расположены, находящиеся в ней элементы. При на ведении на плитку библиотеки, появится иконка редактирования. На странице редактирование можно изменить имеющуюся информацию. С этой странице можно и удалить библиотеку, нажав на иконку корзинки. Сохранить на галочку, отметить на крестик.

Перед удалением выпадает окно подтверждения. Форма редактирования на рисунке 27.

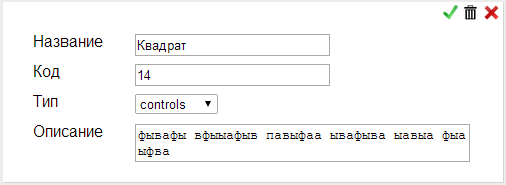


Рисунок 27

Снизу выводится краткая информация о элементе: название, код и описание.

Слева расположена кнопка добавить элемента. При нажатии на эту кнопку осуществляется переход на страницу добавления элемента (рисунок 28).

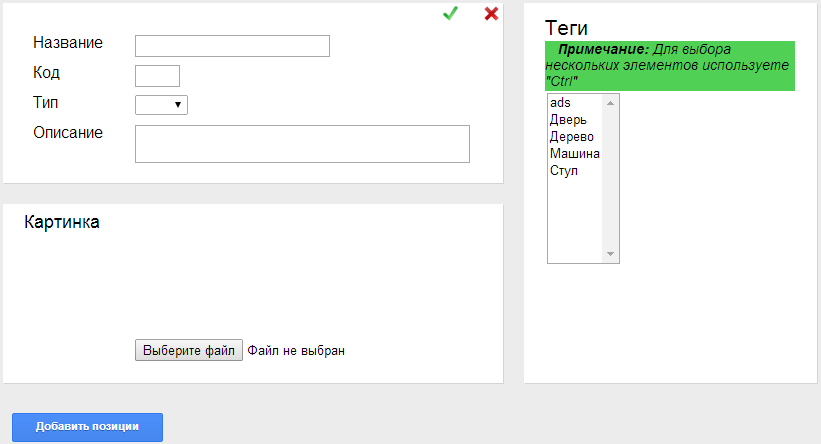


Рисунок 28

В форме редактирования имеется четыре типа информации: общая, теги, картинки, позиции.

К общей информации относятся поля: название, код, тип, описание. Тип элемента выбирается из справочника.

Теги – ключевые слова элемента. Теги можно выбрать или добавить свои в справочники.

Картинка – это главный вид элемента.

Позиции – каждый элемент может выглядеть по-разному, в зависимости от размеров, поворота и т.д. Для этого добавляются различные позиции элемента.

Для добавления новой позиции нужно нажать на кнопку “Добавление позиции”. Каждый раз при нажатии на кнопку срабатывает JavaScript который, добавляет поле для ввода информации ( рисунок 29 ).

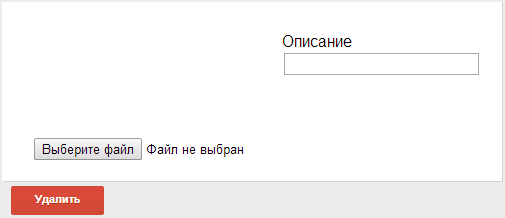


Рисунок 30

На данной форме есть два пункта, выбрать файл и после для описание. Снизу расположена кнопка удалить, при нажатии на которую данная форма удалиться.

Количество добавляемых позиций не ограничено.

3.3.4.1 Добавление изображений

Изображения из добавленных позиций группируются в массив. После передачи данных, массив с помощью цикла разбивается и получается отдельная информация о каждой позиции. Вместо названия файла добавляется в БД время сервера занесенное в переменную time. Делаем SQL запрос на вывод id строки, где столбец image равен переменной time. После определения id подставляем номер в название картинки и добавляем в БД полный путь файла. С помощью цикла добавляем файл в нужную директорию подставляя в название id соответствующей строки. Фрагмент кода на рисунке 31.

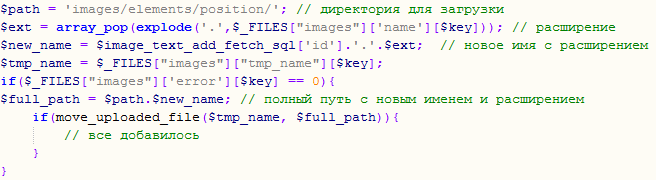


Рисунок 31

3.3.4.2 Добавление тегов

Теги формируются в массив. С помощью конструкции foreach перебирается массив.

Конструкция foreach предоставляет простой способ перебора массивов. Foreach работает только с массивами и объектами, и будет генерировать ошибку при попытке использования с переменными других типов или неинициализированными переменными.

В данной конструкции происходит выбор данных, где массив равен столбцу `tage`.`name`. Полученный результат заносится в отдельную переменную с название “tag\_id”, для дальнейшей подстановки в SQL запросы.

Каждый тег с помощью SQL запроса INSERT заносит данные в таблицу `tags\_elements` . Теги заносятся в базу данных с номером добавляемого элемента.

Код добавления тегов в базу данных представлен на рисунке 32.

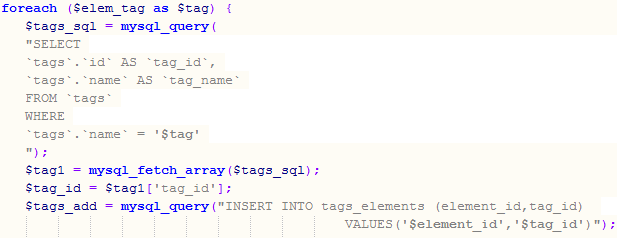


Рисунок 32

3.3.5 Страница элемента

На странице элемента представлена вся информация о выбранном элементе.

Данная страница разбита на 4 блока, где в верхним блоке: название, код, тип, описание. Во втором теги элемента. Третьем основное изображение элемента. Четвертый блок расположен ниже остальных блоков и является не обязательным, так как позиции у элемента может не быть.

Все блоки были созданы в едином стиле, удобном для просмотра пользователя.

На рисунке 31 представлена страница элемента.

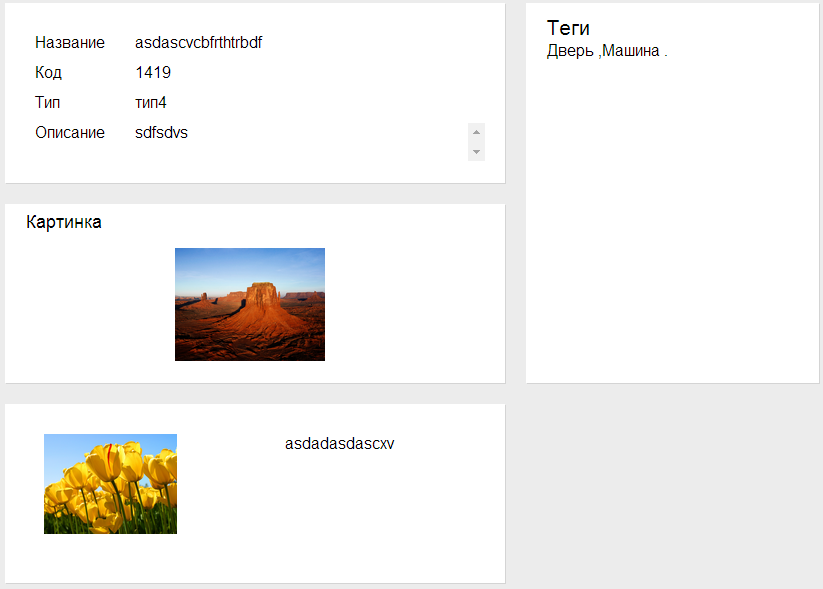


Рисунок 31

3.3.6 Навигация

В навигации отображается выбранные страницы сайта. С помощью навигации можно разобрать в какой библиотеки вы находитесь или в тренажере. Навигация выполнена в упрощенном стиле. Для отображения нужной странице берутся значения переменных из адресной строки: ssl.com/?c=2&s=3&l=4&e=93

Полученные значения выполняются отдельным запросом и выводят названия.

Навигация представлена на рисунке 32.



Рисунок 32

Каждый раздел обозначен свой картинкой для более простого понимания пользователя.

Рисунок 33 обозначает домашнюю страницу.



Рисунок 33

Рисунок 34 обозначает выбранного клиента.



Рисунок 34

Рисунок 35 обозначает выбранный тренажер.

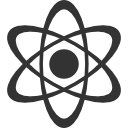


Рисунок 35

Рисунок 36 обозначает выбранную библиотеку.



Рисунок 36

Рисунок 37 обозначает выбранный элемент.

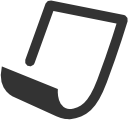


Рисунок 37

Данные изображения достаточно ясно помогают разобраться пользователю, в какой директории он находится в данный момент.

3.4 Справочники

В справочниках содержатся содержится информация о типах библиотек, типах и тегах элементов.

3.4.1 Типы библиотек

Каждая библиотека имеет свой тип, эти типы создаются в справочнике, где могут быть добавлены, редактированы и удалены.

Типы библиотек хранятся в таблице ‘lib\_types’.

На рисунке 32 представлен вид страницы.

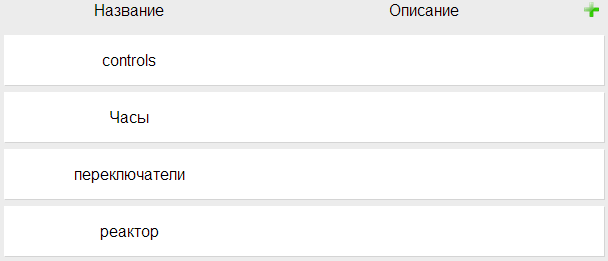


Рисунок 32

При нажатии на кнопку добавить ( кнопка в виде плюса ). Отроется страница добавления. Обязательным полем является название, а описание может быть не заполнено, так как оно является не обязательным.

Страница добавления представлена на рисунке 33.

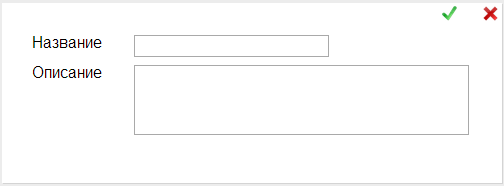


Рисунок 33

При нажатии на кнопку сохранить в виде галочки, будет отправлен SQL запрос на добавление в базу данных. Теперь занесенный тип можно выбрать при добавление библиотеки.

Для удаления требуется перейти в редактирование и нажать на кнопку удалить в виде корзинки. Выполнится SQL запрос на удаление и данный тип будет успешно удален.

3.4.2 Типы элементов

Каждый элемент имеет свой тип, этот тип создается в справочнике, где может быть добавлен, редактирован и удален.

Типы элементов хранятся в таблице “element\_types”.

На рисунке 34 представлен вид страницы.

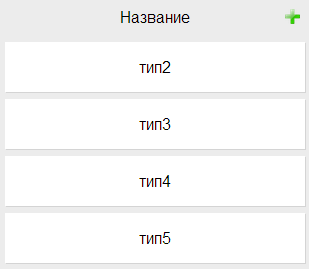


Рисунок 34

При нажатии на кнопку добавить ( кнопка в виде плюса ). Отроется страница добавления. В добавление имеется только одно поле – название.

При нажатии на кнопку сохранить в виде галочки, будет отправлен SQL запрос на добавление в базу данных. Теперь занесенный тип можно выбрать при добавление элемента.

Редактирование типа осуществляется нажатием на нужный тип. При нажатии откроется страница редактирования.

Страница редактирования представлена на рисунке 35.



Рисунок 35

Все типы страницы типов выполнены в едином стиле для упрощенной работы.

3.4.3 Теги элементов

Каждый элемент может иметь множество тегов, теги создаются в справочнике, где могут быть добавлены, редактированы и удалены. Теги нужны для поиска элементов.

Теги элементов хранятся в таблице “tags”.

На рисунке 36 представлен вид страницы.

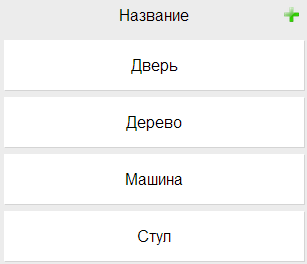


Рисунок 36

При нажатии на кнопку добавить ( кнопка в виде плюса ). Отроется страница добавления. В добавление имеется только одно поле – название.

При нажатии на кнопку сохранить в виде галочки, будет отправлен SQL запрос на добавление в базу данных. Теперь занесенный тег можно выбрать при добавление элемента.

Редактирование тега осуществляется нажатием на нужный тег. При нажатии откроется страница редактирования.

Страница редактирования представлена на рисунке 37.

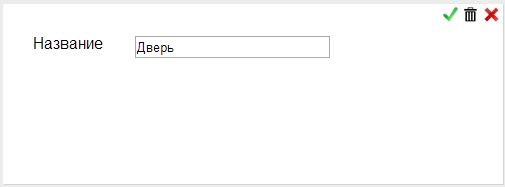


Рисунок 37

3.5. Поиск

Поиск помогает найти пользователю нужную для него информацию, если он выберет какие либо критерии, по которым можно найти искомый объект.

3.5.1 Поиск библиотек

Поиск библиотеки осуществляется по нескольким критериям: код, текст, тип.

Критерии поиска вносятся в соответствующие ячейки ввода, и при нажатии на кнопку “поиск” осуществляется обработка формы.

На рисунке 38 представлена форма поиска.

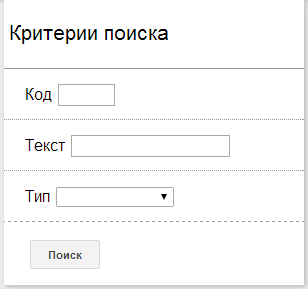


Рисунок 38

Занесенные критерии поиска помещаются в переменные, далее происходит запрос на вывод всех строк из таблицы “lib\_type” удовлетворяющее условию, столбец “name” равен выбранному типу “$type\_name”.

На рисунке 39 представлен фрагмент кода.

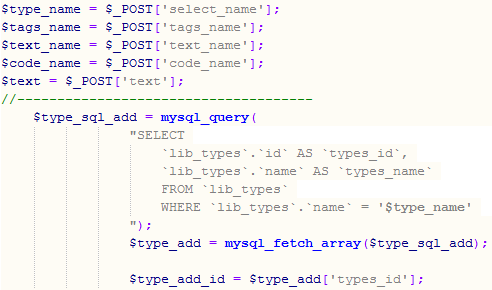


Рисунок 39

При разном выборе критериев будет разный SQL запрос и требуется подставлять условия поиска. Проверяем переменные, если они не равные нулю, тогда в новую переменную записываем условия поиска, если равно, то делаем новую переменную NULL. После каждого условия требуется поставить “ AND ”. Проверяем несколько переменных на NULL и заносим в новые переменные “ AND ” иначе присваиваем NULL.

На рисунке 40 представлен фрагмент кода.

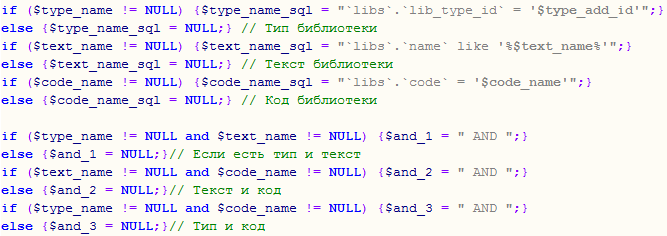


Рисунок 40

Подставляем в SQL запрос переменные, рисунок 41.

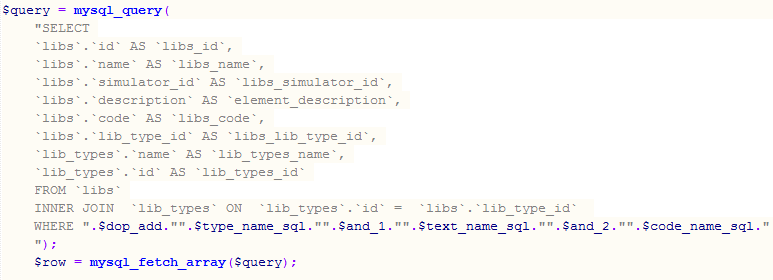


Рисунок 41

Создаем цикл do-while. Если после SQL запрос не выдал NULL, тогда делаем новый SQL запрос для выбора требуемой библиотеки.

На рисунке 42 представлен SQL запрос.

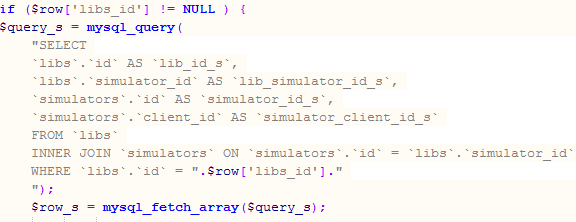


Рисунок 42.

Далее выводим все строки из таблицы “libs” удовлетворяющих условия поиска.

Закрываем цикл do- while, рисунок 43.



Рисунок 43

3.5.2 Поиск элементов

Поиск элемента осуществляется по нескольким критериям: код, текст, тег, тип.

Критерии поиска вносятся в соответствующие ячейки ввода, и при нажатии на кнопку “поиск” осуществляется обработка формы.

На рисунке 44 представлена форма поиска.

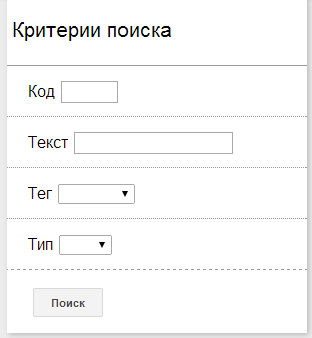


Рисунок 44

Занесенные критерии поиска помещаются в переменные, далее происходит запрос на вывод всех строк из таблицы “element\_types ” удовлетворяющее условию, столбец “name” равен выбранному типу “$type\_name”.

Определяем номер тега. Осуществляем SQL запрос на поиск тега в таблице “tags”, где столбец “name” равен “$tags\_name”. Заносим id найденного тега в переменную “$tags\_add\_id”.

На рисунке 45 представлен фрагмент кода.

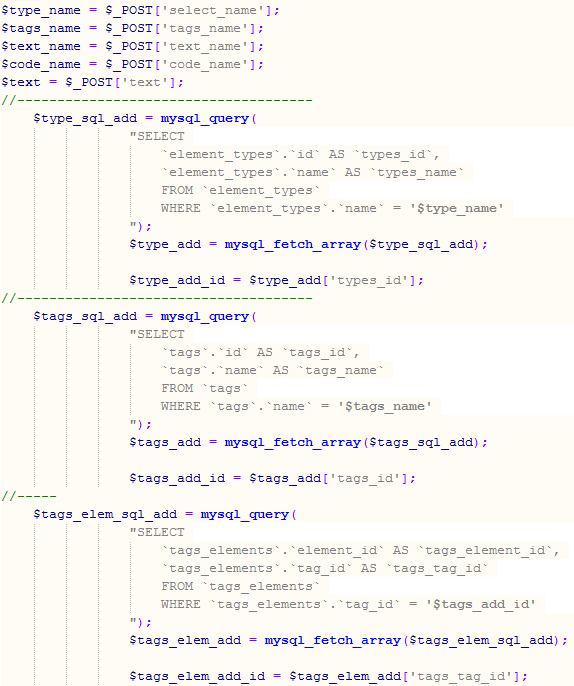


Рисунок 45

Определяем номер элементов с данным тегом. Осуществляем SQL запрос на поиск элементов в таблице “tags\_elements”, где столбец “tag\_id” равен переменной “$tags\_add\_id”. Заносим id элементов в переменную “tags\_elem\_add\_id”.

При разном выборе критериев будет разный SQL запрос и требуется подставлять условия поиска. Проверяем переменные, если они не равные нулю, тогда в новую переменную записываем условия поиска, если равно, то делаем новую переменную NULL. После каждого условия требуется поставить “ AND ”. Проверяем несколько переменных на NULL и заносим в новые переменные “ AND ” иначе присваиваем NULL.

На рисунке 46 представлен фрагмент кода.

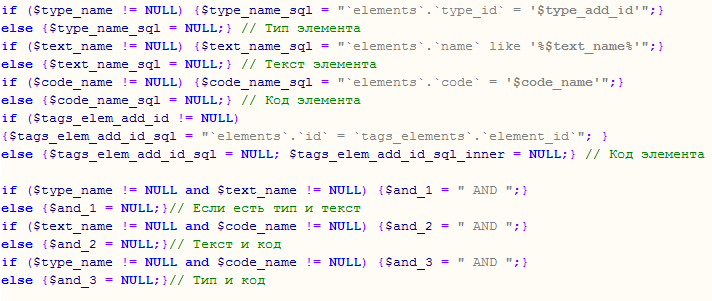


Рисунок 46

Если не осуществляется поиск по тегу, тогда выбираем SQL запрос без тега.

В SQL запрос подставляем условия поиска, рисунок 47.

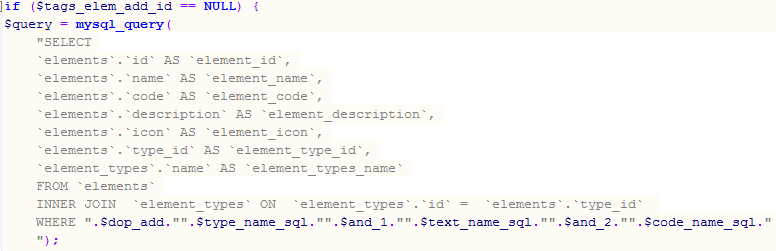


Рисунок 47

Если выбран тег, тогда осуществляем SQL запрос с добавленным условием.

На рисунке 48 представлен фрагмент кода.

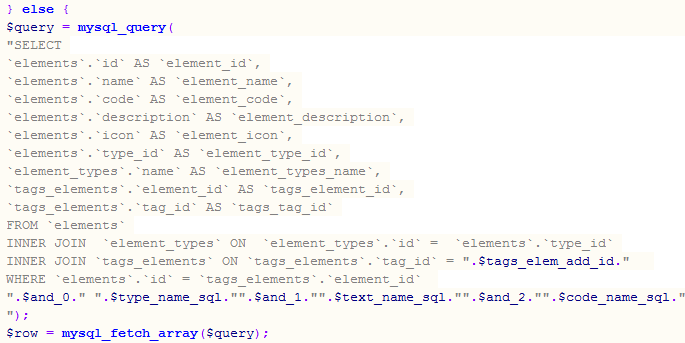


Рисунок 48

Создаем цикл do-while. Определяем переменную id элемента, если она не NULL, тогда осуществляем SQL запрос из таблицы “elements” , где столбец id равен номеру “$row[‘element\_id’]”. Заносим данный результат запроса в переменную “$row\_s” и подставляем значения в html код, для определения страницы элемента.

Фрагмент кода представлен на рисунке 49.

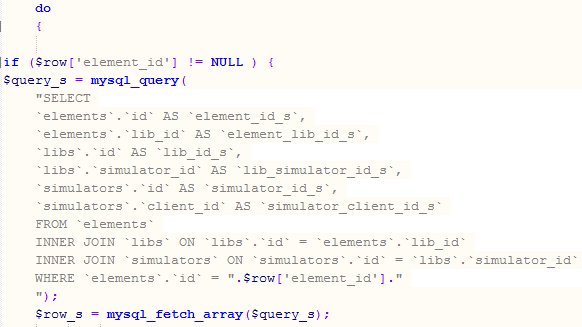


Рисунок 49

4. Руководство пользователя

В данном web-приложении все будет в едином стиле.

Кнопка редактирования представлена на рисунке 50.

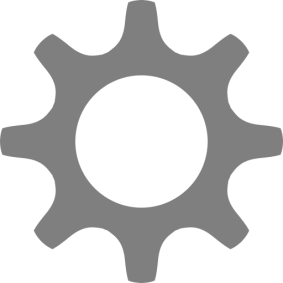


Рисунок 50

Кнопка сохранить представлена на рисунке 51.



Рисунок 51

Кнопка удалить представлена на рисунке 52.



Рисунок 52

Кнопка закрыть/отменить представлена на рисунке 53.



Рисунок 53

Кнопка добавить представлена на рисунке 54.



Рисунок 54

На странице клиентов, для добавления клиента требуется нажать на кнопку ”добавить”. Форма добавления полностью идентичные с остальными ( рисунок 55 ).

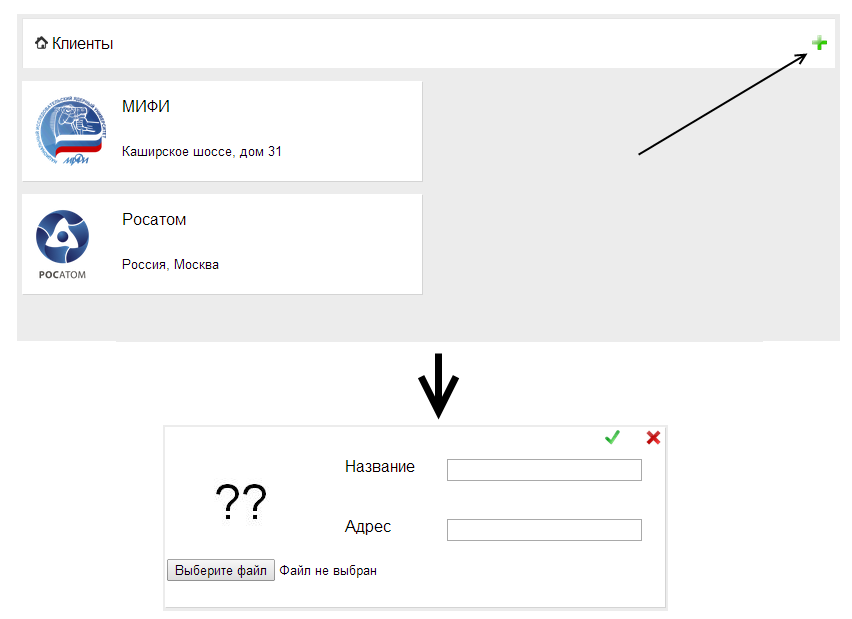


Рисунок 55

При нажатии на кнопку сохранить, форма сохраняется. Для закрытия, нажать на кнопку закрыть.

На странице клиента, можно редактировать о нем информацию ( рисунок 56 ).

При нажатии на кнопку редактировании, откроется форма редактирования, при сохранении выведется информация о добавлении.



Рисунок 56

Добавление тренажера осуществляется при нажатии на кнопку добавить на странице клиента ( рисунок 57 ).

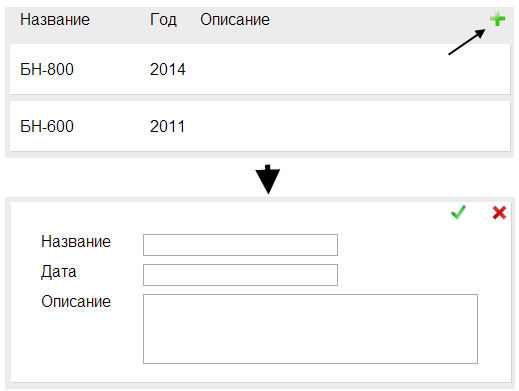


Рисунок 57

На странице тренажера, редактирование, происходить идентичен с остальными.

Добавление библиотеки осуществляется при нажатии на кнопку добавить, форма добавление аналогична форме тренажера. Для добавления используемых библиотек требуется нажать на кнопку добавить и на странице добавления выбрать нужную библиотеку ( рисунок 58 ).

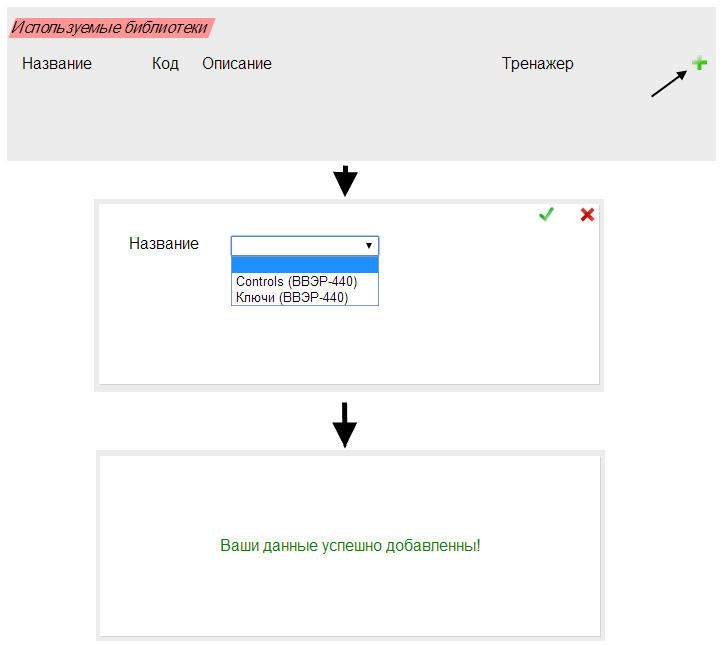


Рисунок 58

На странице библиотеки редактирование полностью идентично. Добавить информацию об элементе можно с помощью кнопки “добавить” . После добавление информации требуется нажать на кнопку сохранить ( Рисунок 59 ).

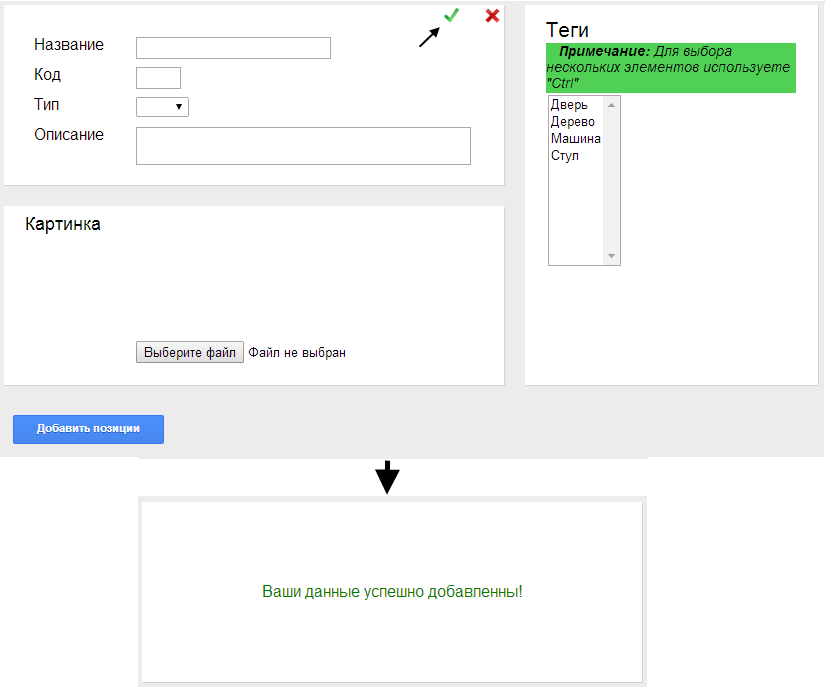


Рисунок 59

Добавление типов библиотек осуществляется кнопкой добавить ( рисунок 60 ). Форма добавления, редактировании, полностью идентичны предыдущем.

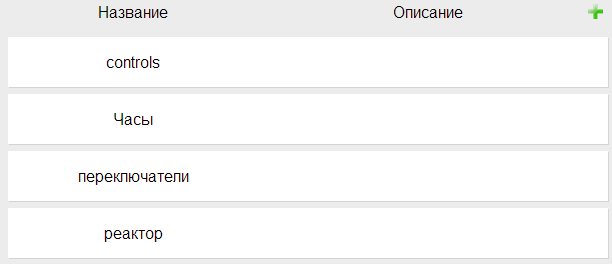


Рисунок 60.

Типы и теги элементов выполнены в едином стиле и не отличаются друг от друга по формам добавления и редактирования. Редактирование тип или тега осуществляется нажатием на форму ( рисунок 61 ).

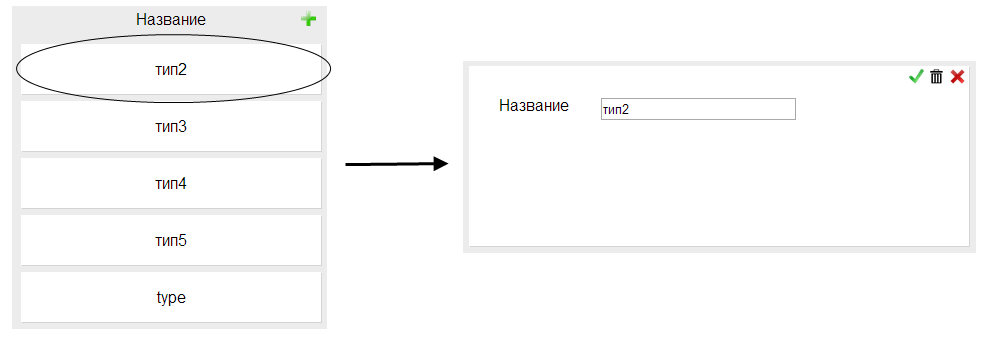


Рисунок 61

Поиск библиотек и элементов сделан в едином стиле. На странице поиска элементов, требуется ввести или выбрать критерии поиска и нажать на кнопку поиск.

Работа формы представлена на рисунке 62.

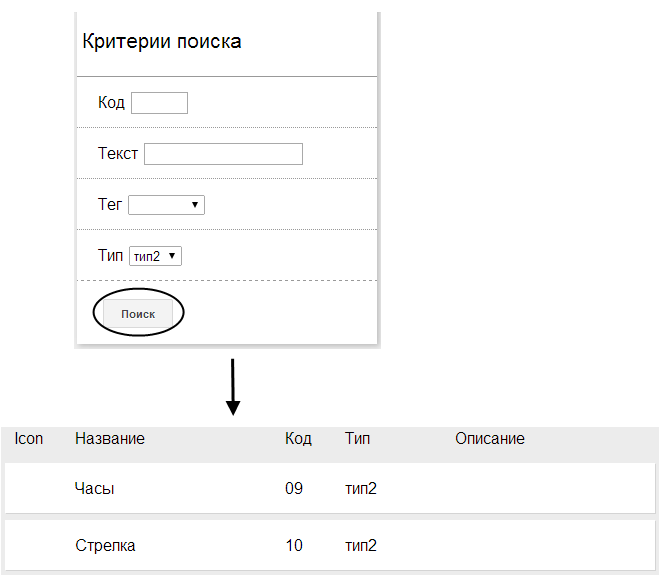


Рисунок 62

5. Экономическая часть

5.1. В экономической части дипломного проекта производил расчёт себестоимости создания проекта «Разработка базы данных динамических объектов человеко-машинного интерфейса функционально аналитического тренажеров для подготовки оперативного персонала АЭС».

**1**. **Затраты на материалы, руб.:**

Таблица 1. Спецификация на материалы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Показатель | Цена за единицу,руб | Кол-во | Сумма, руб |
| 1 | Бумага А4 | 200 | 1 | 200 |
| 2 | Канцелярские товары | 25 | 4 | 100 |
|  | Итог: | | | 300 |

**2. Основная заработная плата, руб.:**

Над проектом работал один человек, и его заработная плата составила 21000 рублей за 21 рабочий день, но так как он работал три месяца апрель, май, начало июня (51 рабочий день), его основная заработная плата составила 51000 рублей.

ЗПₒ. = 21000/21\*51 = 51000

**3. Дополнительная заработная плата (оплата отпусков, командировок, выполнения гособязанностей), руб.:**

ЗПдоп. = ЗПо. ˟ Кдоп.

где:

Кдоп.- коэффициент дополнительной заработной платы, Кдоп. = 10%

ЗПдоп.=51000 ˟ 0,1 = 5100

**4. Отчисления на социальные нужды, руб.:**

Ос.н. = (ЗПо.) ˟ Кс.н.,

Кс.н = 15%

О с.н. =(51000) ˟ 0,15 = 7650

**5. Содержание и эксплуатация оборудования, руб.:**

Таблица 2. Спецификация на информационное оборудование.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Показатели** | **Единица Измерения** | **Цена за единицу** | **Количество** | **Сумма, руб.** |
| **1** | **Компьютер** | **Шт.** | **30000** | **1** | **30000** |
| **2** | **Принтер** | **Шт.** | **6000** | **1** | **6000** |
|  | **Итого** | | | | **36000** |

Амортизационный период предположительно будет равен трём годам. Следовательно данную сумму делим на три года, затем на 12 месяцев и умножаем на 2 месяца, чтобы узнать расходы на содержание и эксплуатацию оборудования за 2 месяца.

36000/2/12\*2 = 3000

**6. Затраты на интернет, руб.:**

В течении разработки диплома использовался безлимитный интернет стоимостью 1000 рублей за месяц.

1000\*3 = 3000

**7. Затраты на электроэнергию, руб.:**

В среднем компьютер потребляет 500Вт/час. Работал по 3 часа в день.

Зэл. = Р ˟ Цквт·ч Т

где:

Т – время

Р – потребляемая мощность

Цквт·ч – стоимость 1 кВт·ч электроэнергии, 4.54 руб.:

500 Вт/час \* 3 часа = 1.5 кВт/день

1.5 кВт \* 51 день = 76.5 кВт/3мес

76.5 кВт \* 4.54 = 347

**8. Программное обеспечение, руб.:**

Лицензионное программное обеспечение пособие действует сроком на один год, но мы будем его использовать в течение 51 рабочего дня.

Лицензионное программное обеспечение Windows 7 Professional.

10353/12/22\*51 = 2000

**9. Накладные расходы, руб.:**

Рнакл. = К накл. ˟ (ЗПо.),

где:

Кнакл. – коэффициент накладных расходов, Кнакл. = 2,7.

Рнакл. = 2,7 ˟ 7000 = 18900

Все затраты на разработку программы сводим в таблицу № 3.

Таблица 3. Калькуляция себестоимости.

|  |  |  |
| --- | --- | --- |
| № | Статья затрата | Сумма (руб.) |
| 1 | Материалы | 300 |
| 2 | Основная заработная плата | 51000 |
| 3 | Дополнительная заработная плата | 5100 |
| 4 | Отчисления на социальное страхование | 7650 |
| 5 | Содержание и эксплуатация оборудования | 3000 |
| 6 | Интернет | 3000 |
| 7 | Электроэнергия | 347 |
| 8 | Программное обеспечение | 2000 |
| 9 | Накладные расходы | 18900 |
|  | Итого | 91297 |

Таким образом расходы на создание проекта сос тавили 91297 рублей. Договорная цена складывается из оптовой цены продукта плюс НДС(18%). Оптовая цена продукта равна полной себестоимости разработки + прибыль (30%).

Прибыль = 91297 \* 30% = 27389 (руб.)

Оптовая цена продукта = 91297 + 27389 = 118686 (руб.)

НДС = 118686 \* 18% = 21363 (руб.)

Договорная цена = 118686 + 21363 = 140050 (руб.)

ЗАКЛЮЧЕНИЕ

Данная дипломная работа была выполнена в Экспериментально научно-исследовательском и методическом центре “Моделирующие системы” (ЭНИМЦ МС).

За время своего существования компания зарекомендовала себя как стабильное предприятие, специализирующееся в области прикладных научно-исследовательских и опытно-конструкторских работ (НИОКР) для атомной энергетики и промышленности, а также иных высокотехнологичных производств.

Одним из основных направлений деятельности ЭНИМЦ МС является тренажеростроение для обучения и поддержания квалификации оперативного персонала атомных и тепловых станций, а также для обучения студентов.

Данное web-приложение успешно используется в компании ЭНИМЦ МС. В базу данных, занесены более 10 клиентов, 20 тренажеров, 100 библиотек и свыше 500 динамических объектов. Web-приложение протестировано и успешно выполняет поставленные задачи.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Руководство по PHP. http://www.php.net/

2. Сайт компании ЭНИМЦ МС. <http://www.ssl.obninsk.ru/>

3. Статьи по программированию Хабрахабр. <http://habrahabr.ru/>

4. Руководство по обеспечению доступности Web-контента. <http://www.w3.org/>

5. Сайт про SQL. http://www.sql.ru/