Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных Технологий
институт

Информационные системы
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой ИС
_____ Л.С. Троценко
подпись, дата инициалы, фамилия
«13» июня 2018 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 Информационные системы и технологии

Разработка веб-приложения для проверки документов на соответствие требованиям оформления

Руководитель _____ П.П. Дьячук
подпись, дата инициалы, фамилия

Студент КИ14-13Б, 031403218 _____ А.П. Туров
номер группы, зачетной книжки подпись, дата инициалы, фамилия

Нормоконтролер _____ Ю.В. Шмагрис
подпись, дата инициалы, фамилия

Красноярск 2018

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Информационные системы
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой ИС

_____ С.А. Виденин
подпись, дата инициалы, фамилия

«2» марта 2018г.

ЗАДАНИЕ

НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

в форме бакалаврской работы

Студенту: Турову Андрею Павловичу

Группа: КИ14-13Б Направление: 09.03.02 «Информационные системы и технологии»

Тема выпускной квалификационной работы: Разработка веб-приложения для проверки документов на соответствие требованиям оформления

Утверждена приказом по университету № 4896/с от 05.04.2018г.

Руководитель ВКР: П. П. Дьячук, кандат пед. наук, доцент кафедры «Информационные системы» ИКИТ СФУ

Исходные данные для ВКР: Требования к разрабатываемому веб-приложению, рекомендации руководителя, учебные материалы

Перечень разделов ВКР: Введение, теоритическая часть, анализ задачи, проектирование веб-приложения, разработка веб-приложения, заключение, список использованных источников, приложение А

Руководитель ВКР _____ П. П. Дьячук

подпись

Задание принял к исполнению _____ А. П. Туров

подпись

«2» марта 2018г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Разработка веб-приложения для проверки документов на соответствие требованиям оформления» содержит 46 страниц текстового документа, 15 иллюстраций, 19 использованных источников.

ВЕБ-ПРИЛОЖЕНИЕ, MVC, CODEIGNITER, PHP, HTML, JS_QUERY, BOOTSTRAP, MYSQL.

Цель работы — разработка веб-приложения для проверки документов на соответствие требованиям оформления.

Для достижения поставленной цели были определенны следующие задачи:

– изучение и выбор инструментов для реализации веб-приложения;

– проектирование веб-приложения;

– разработка веб-приложения.

В результате выполнения выпускной квалификационной работы было создано веб-приложение для проверки документов на соответствие требованиям оформления.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность выбранной темы обусловлена необходимостью ручной проверки документов учебной деятельности на соответствие требованиям оформления, установленным в стандартах организации.

Цель данной работы заключается в создании веб-приложения, которое позволит автоматически проверять некоторые требования документов учебной деятельности на соответствие требованиям оформления.

Для достижения цели ставятся следующие задачи:

- осуществить поиск и анализ существующих аналогов разрабатываемого приложения;
- изучить и выбрать инструменты для реализации веб-приложения;
- осуществить проектирование веб-приложения;
- разработать веб-приложение.

1 Теоритическая часть

1.1 Описание предметной области

Для проверки стандартов оформления был выбран формат для хранения электронных документов пакетов офисных приложений «.docx». Причиной такого выбора стали:

- широкое распространение текстового процессора Microsoft Office, для которого формат «.docx» является стандартным начиная с версий 2007 года;

- данный формат является открытым т.е. спецификация данного формата является общедоступной и свободной от лицензионных ограничений при использовании;

- наличие конверторов, способных конвертировать документы других популярных текстовых процессоров в «.docx» и обратно;

DOCX это один из серии форматов Office Open XML, используемый для хранения электронных документов пакетов офисных приложений — в частности, Microsoft Office. Формат представляет собой zip-архив, содержащий текст в виде XML, графику и другие данные.

Первоначально формат создавался как замена прежнему двоичному формату документов, который использовали приложения Microsoft Office вплоть до версии Office 2003 включительно. В 2006 году формат Office Open XML был объявлен свободным и открытым форматом Ecma International.

Структура «.docx» архива после распаковки представлена на рисунке 1.

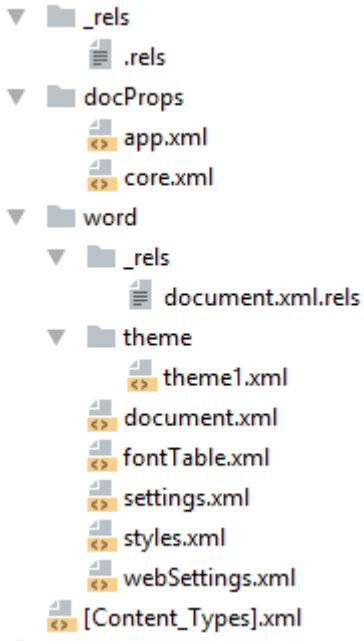

Рисунок 1 – Структура «.docx» архива

Описание основных файлов:

- «docProps/core.xml» — основные метаданные;

- «docProps/app.xml» — общая информация о документе: количество страниц, слов, символов, название приложения в котором был создан документ и т.п.;

- «word/settings.xml» — настройки документа;

- «word/styles.xml» — стили применимые к документу;

- «word/webSettings.xml» — настройки отображения HTML частей документа и настройки того, как конвертировать документ в HTML;

- «word/fontTable.xml» — список шрифтов, используемых в документе;

- «word/theme1.xml» — тема (состоит из цветовой схемы, шрифтов и форматирования);

Основное содержимое документа представлено в файле «word/document.xml». Ниже представлена общая структура этого документа:

- <w:document> — контейнер документа;

- <w:body> — тело документа;

- <w:p> — параграф;

- <w:r> — фрагмент текста;
- <w:t> — текстовое содержимое фрагмента;
- <w:sectPr> — описание страниц;

1.2 Обзор аналогов приложения

Единственным аналогом разрабатываемого приложения, который удалось найти в интернет, является приложение для windows «Нормоконтроль».

1.2.1 Приложение «Нормоконтроль»

Приложение «Нормоконтроль» реализует проверку курсовых работ, отчетов по НИР, творческих работ на соответствие стандартам оформления «Современной Гуманитарной Академии». Приложение предоставляет подробный отчет о ошибках оформления, пример отчёта показан на рисунке 2.

№ п/п	Параметр	Требуемое значение параметра	Соответствует: + Не соответствует: -
1	Вид творческой работы	Творческая работа	+
2	Формат файла творческой работы	MS Word	+
3	Последовательность приведения структурных частей работы	Основные данные о работе, Содержание, Основная часть, Список сокращений, Приложение	+
4	Шрифт	Times New Roman	+
5	Размер шрифта	14	+
6	Интервал межстрочный	1.5	+
7	Верхнее поле	2	+
8	Нижнее поле	2	+
9	Левое поле	2	+
10	Правое поле	1	+
11	Абзац	1.25	+
12	Количество страниц	7-12	+
13	Основная часть	1	+
14	Основные данные о работе	1	+
15	Приложение	0-1	+
16	Содержание	1	+
17	Список сокращений	0-1	+
18	Основная часть Глава	0-1	+
19	Основная часть Количество страниц	5-7	+
20	Стиль структурных частей документа	оформление структурных элементов	+
21	Соразмерность глав	различие не более чем на 5 страниц	+

Рисунок 2 – Отчёт о ошибках оформления

Недостатки данного приложения:

- работа с устаревшим форматом «.doc»;
- необходимость установки на компьютер;
- отсутствие документации для добавления/изменения стандартов;
- отсутствие указания фрагмента текста с ошибкой;

Достоинства приложения «Нормоконтроль»:

- приложение является бесплатным;
- предоставление подробного отчёта по ошибкам оформления;

1.2.2 Выводы

В русскоязычном сегменте интернета было найдено только одно аналогичное приложение, и оно обладает существенными недостатками – необходимостью установки на компьютер и работой только под операционной системой Windows.

2 Анализ задачи

2.1 Требования к функционалу приложения

После проведённого анализа задачи и обзора аналогичных программ, сделаны следующие выводы о требованиях к разрабатываемому веб-приложению:

- возможность проверки документов современного формата;
- лёгкая расширяемость;
- быстрая скорость работы;
- стабильная работа при высокой нагруженности;
- минимальные требования к программным и аппаратным средствам для возможности развёртывания на бюджетном хостинге или сервере;

2.2 Инструментальные средства для разработки

PHP - скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов.

Преимуществом php в области построения веб-сайтов является наличие большого набора встроенных средств для разработки веб-приложений. Основные из них:

- автоматическое извлечение POST и GET-параметров, а также переменных окружения веб-сервера в предопределённые массивы;
- взаимодействие с большим количеством различных систем управления базами данных;
- автоматизированная отправка HTTP-заголовков;
- работа с HTTP-авторизацией;

- работа с cookies и сессиями;

- работа с локальными и удалёнными файлами, сокетами;

- обработка файлов, загружаемых на сервер;

- работа с XForms;

CodeIgniter - это легкий, но при этом достаточно мощный и простой в использовании PHP-фреймворк для разработки веб-приложений. Он предоставляет ряд библиотек и вспомогательных функций, значительно ускоряющих процесс разработки. CodeIgniter основан на технике объектно-ориентированного программирования и архитектуре MVC (Model-View-Controller), что позволяет отделить представление от логики.

К преимуществам CodeIgniter можно отнести:

- бесплатное распространение;

- малый вес;

- быстрота работы;

- использование MVC;

- расширяемость;

MySQL - это свободно распространяемая реляционная система управления базами данных. MySQL является решением для малых и средних приложений.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц, таких как: MyISAM, поддерживающие полнотекстовый поиск, таблиц InnoDB, поддерживающие транзакции на уровне отдельных записей, и других.

Рассмотрим основные достоинства СУБД MySQL:

- MySQL характеризуется стабильностью работы;

- Наряду с Oracle, MySQL считается одной из самых быстрых СУБД в мире;

- Открытый код доступен для просмотра и модернизации, что позволяет постоянно улучшать программный продукт;

- поддержка API для популярных языков программирования;

- широкий выбор типов таблиц, в том числе и сторонних разработчиков, что позволяет реализовать оптимальную для решаемой задачи производительность и функциональность;

- Корректная локализация - как правило, не возникает проблем при обработке русского содержимого БД;

Фреймворк Bootstrap - это свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения.

3 Проектирование приложения

3.1 Блок-схема работы приложения

Блок-схема работы приложения на CodeIgniter представлена на рисунке 3.

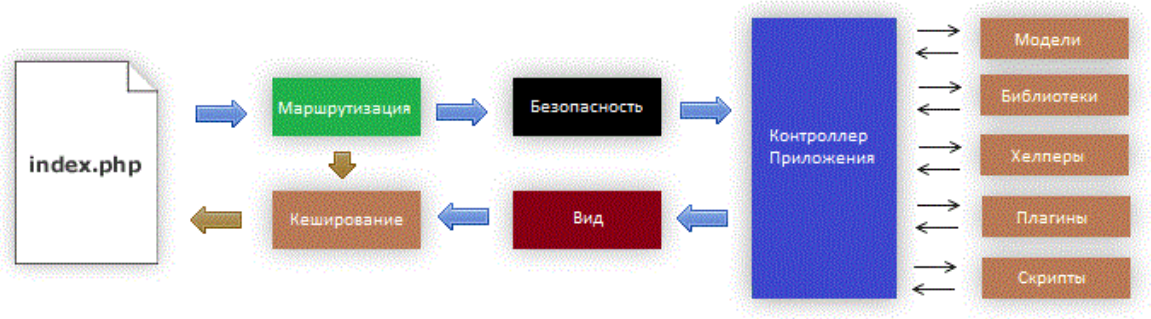

Рисунок 3 – Блок-схема работы приложения

Описание элементов блок-схемы:

1. Index.php служит как основной контроллер, инициализирующий необходимые для работы CodeIgniter ресурсы.

2. Маршрутизатор проверяет HTTP запрос, чтобы определить его дальнейшую судьбу.

3. Если файлы кеша существуют, то они отправляются сразу в браузер, минуя исполнительные процессы системы.

4. Безопасность. Перед загрузкой контроллера приложения, HTTP запрос и любая подтвержденная информация пользователя проходит фильтрацию безопасности.

5. Контроллер загружает Модель, ядро библиотек, хелперы, и любые другие необходимые ресурсы для выполнения запроса.

6. Завершающий Вид отправляет в браузер для отображения полученную информацию. Если кеширование доступно, то Вид сначала кеширует чтобы в дальнейшем можно было повторно использовать.

3.2 Проектирование моделей в базе данных

Модель в CodeIgniter представляет структуру данных. Классы моделей содержат функции, помогающие получить, вставить и обновить информацию из базы данных.

Для создания функций работы с БД CodeIgniter дает доступ к классу конструктора запросов. Этот шаблон позволяет информации в базе данных быть обработанной вставкой и обновлением с минимальными трудозатратами. В некоторых случаях достаточно только одной или двух строк код для выполнения действий в базе. CodeIgniter не требует, чтобы каждая таблица базы данных имела свой собственный файл класса. Вместо этого он обеспечивает более упрощенный интерфейс.

Определим перечень объектов, имеющих влияние на информационную систему, и установим между ними связи:

- Свойства оформления документов
- Проверяемые свойства
- Список стандартов, для которых проверяются требования
- Типы проверяемых документов

Вспомогательными являются сущности аккаунтов для административной панели приложения.

На рисунке 4 показана структура базы данных MySQL.

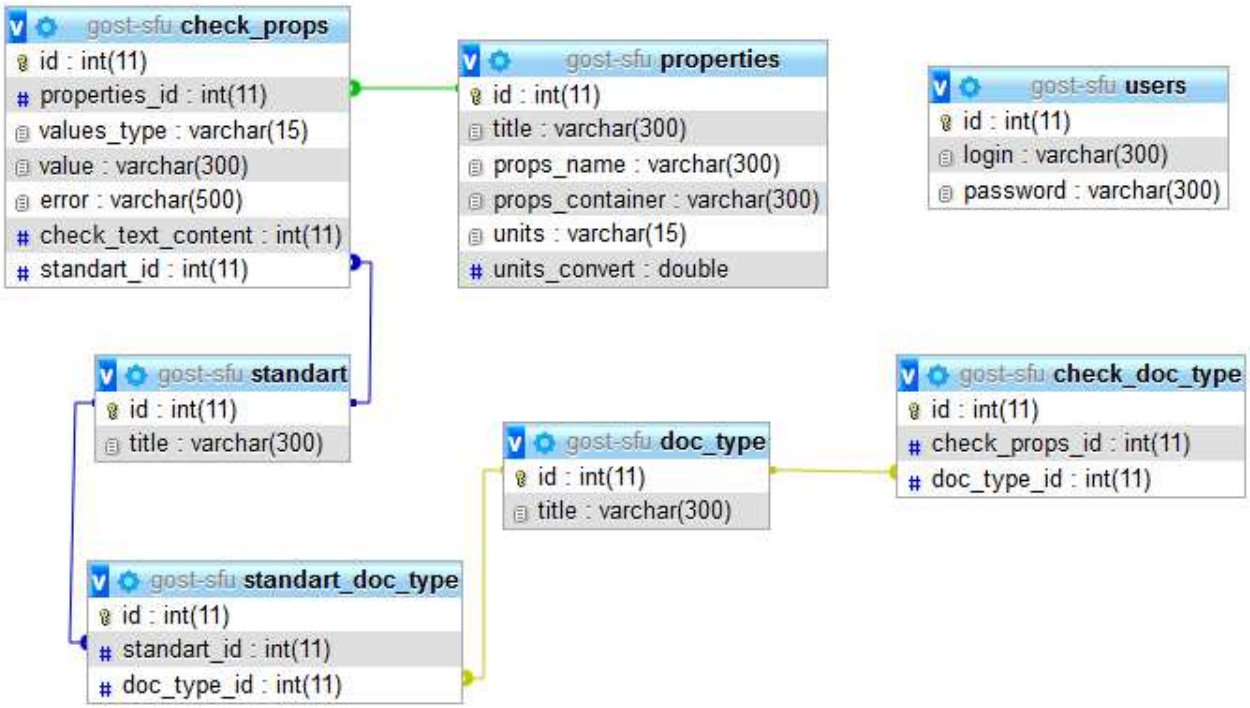

Рисунок 4 – Структура базы данных

Сущности приложения:

- properties, свойства оформления документа;
- standart, стандарт для которых проверяются требования;
- doc_type, типы проверяемых документов;
- check_props, проверяемые свойства;
- users, аккаунты для административной панели

3.3 Представления приложения

3.3.1 Главная страница

Представление главной страницы должно содержать следующие элементы:

- заголовок;
- форма для загрузки документа;

Данные необходимые для работы представления главной страницы:

- массив стандартов оформления;
- массив типов документов, связанных с стандартом оформления;
- массив ошибок валидации формы или загрузки документа, в случае их возникновения;

3.3.2 Страница отчёта об ошибках

Страница отчёта об ошибках необходима для вывода ошибок оформления после проверки документа и должна содержать следующие элементы:

- заголовок;
- таблица для вывода ошибок;
- кнопка для перехода на главную страницу;

Данные необходимые для работы представления главной страницы:

- массив ошибок оформления;

3.3.3 Страница административной панели

Страница административной панели необходима для добавления и редактирования свойства документа, проверяемых свойств, стандартов и типов документов. Представление административной панели должна содержать следующие элементы:

- заголовок;
- таблица свойства документа;
- таблица проверяемых свойств;
- таблица стандартов;
- таблица типов документов;
- кнопки для добавления свойства документа, проверяемых свойств, стандартов и типов документов;
- кнопки для редактирования элементов таблиц;

Данные необходимые для работы представления главной страницы:

- массив свойства документа;
- массив проверяемых свойств;
- массив стандартов;
- массив типов документов;

3.4 Контроллер приложения

Для реализации работы приложения, построенного на архитектуре MVC необходимы контроллеры. При проектировании было принято решение использовать один контроллер в связи с несложной структурой приложения.

Структура контроллера должна содержать следующие методы для работы с видами:

- index – для работы с представлением главной страницы;
- report – для работы с представлением страницы отчёта об ошибках;
- add_properties – для работы с представлением старницы добавления свойств документов;
- add_check_props – для работы с представлением страницы добавления проверяемых свойств;
- add_doc_type – для работы с представлением страницы добавления типов документов;
- add_standart – для работы с представлением страницы добавления стандартов;
- admin – для работы с представлением страницы административной панели;

4 Реализация приложения

4.1 Файловая структура разработанного веб-приложения

Файловая структура корневого каталога разработанного приложения представлена на рисунке 5.

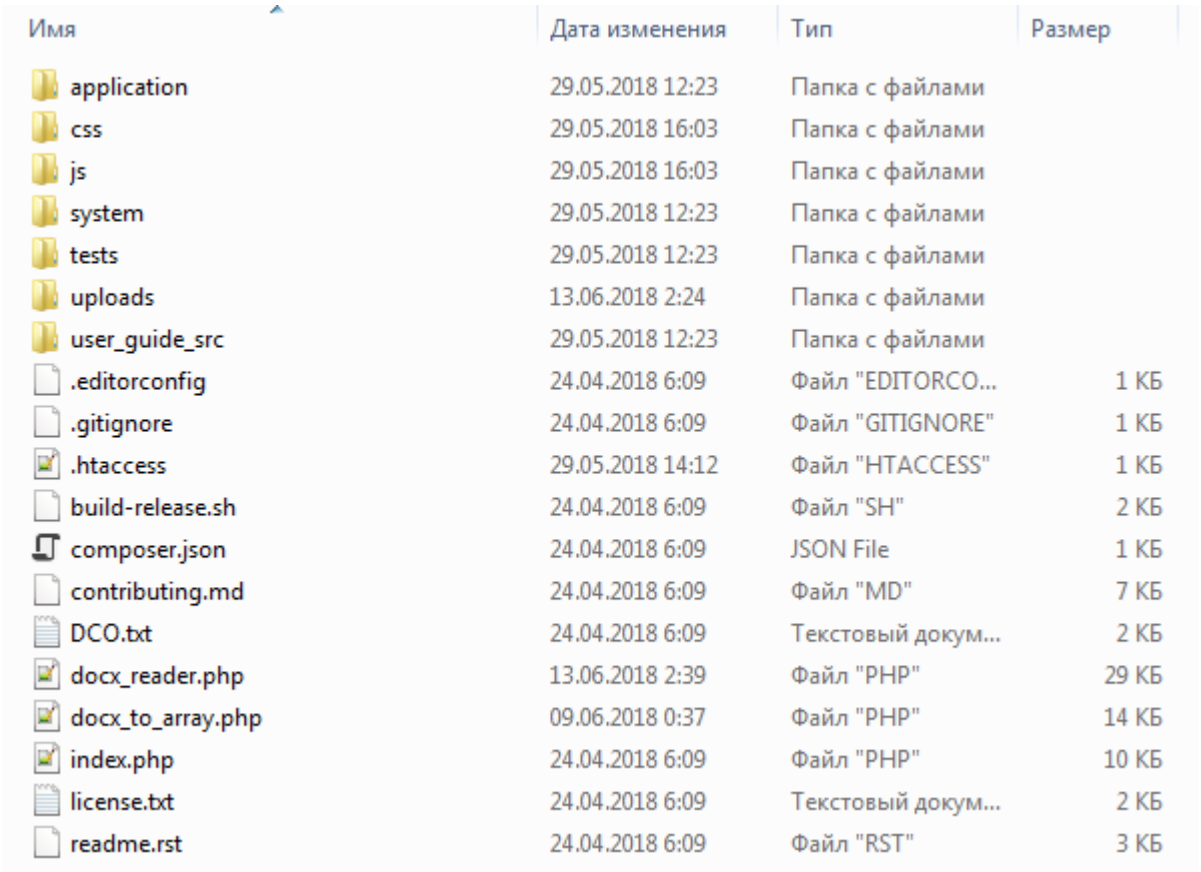

Рисунок 5 – Файловая структура корневого каталога

Описание и структура основных каталогов и файлов представлена ниже:

- каталог «application»
    - каталог «models» - каталог с моделями приложения
    - каталог «views» - каталог с представлениями приложения
    - каталог «controllers» - каталог с контроллерами приложения
    - каталог «config» - каталог файлами настроек приложения

- каталог «css» - каталог с таблицами стилей
- каталог «js» - каталог скриптами javascript
- файл «index.php» - точка входа в приложение
- каталог «uploads» - каталог используемый для загрузки документов

4.2 Реализация загрузки документа

Интерфейс загрузки документа состоит из заголовка и формы загрузки. Форма загрузки в свою очередь состоит из двух селекторов (для выбора стандарта и для выбора типа проверяемого документа), поля для загрузки документа и кнопки для отправки формы. Внешний вид интерфейса загрузки документа представлен на рисунке 6.

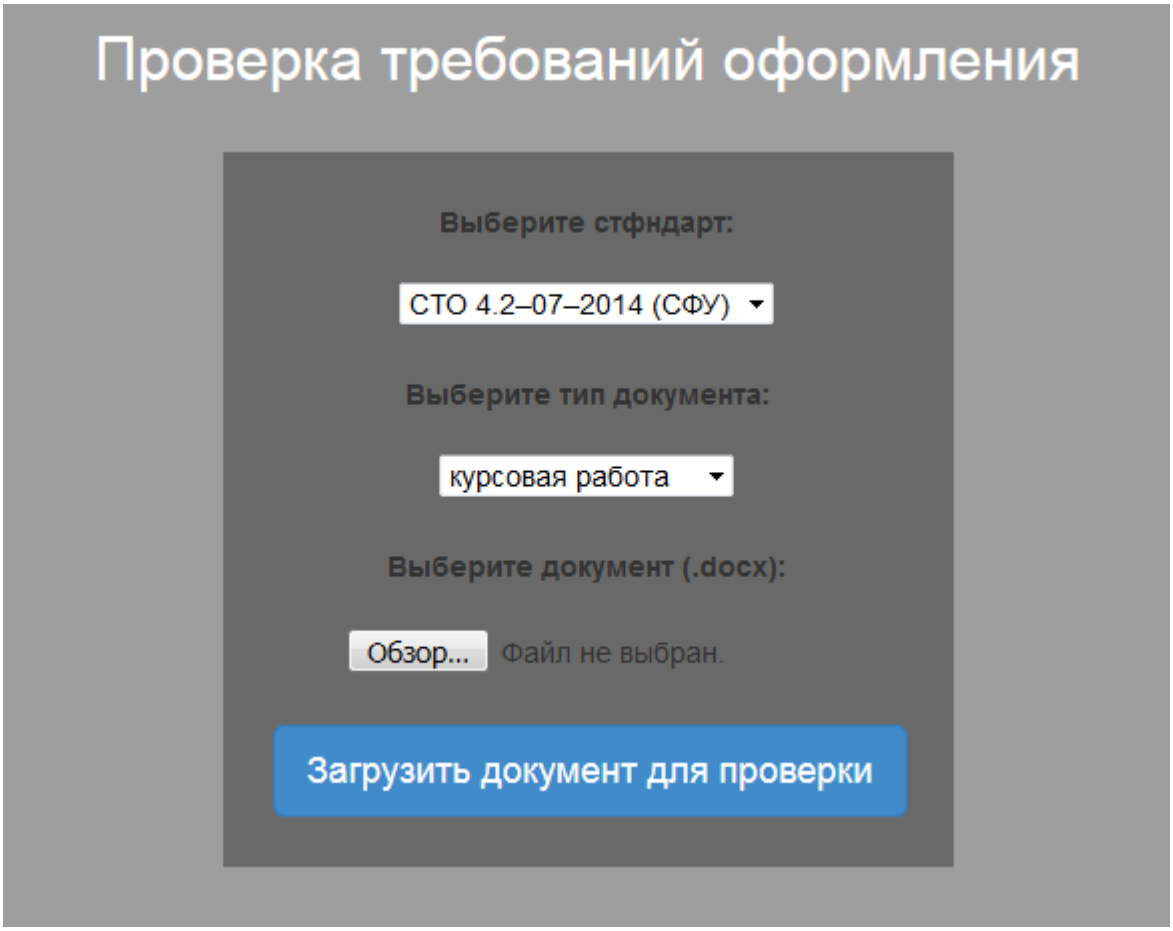

Рисунок 6 – Интерфейс загрузки документа

Для обеспечения безопасности работы приложения необходимо проверять поля формы на соответствие ожидаемой информации. Для этой цели в CodeIgniter существует класс form_validation. Данный класс позволяет задать правила проверки для нужного поля и вывести ошибки в случае их возникновения. Например, при отсутствии документа в форме, form_validation выдаст ошибку, изображённую на рисунке 7.

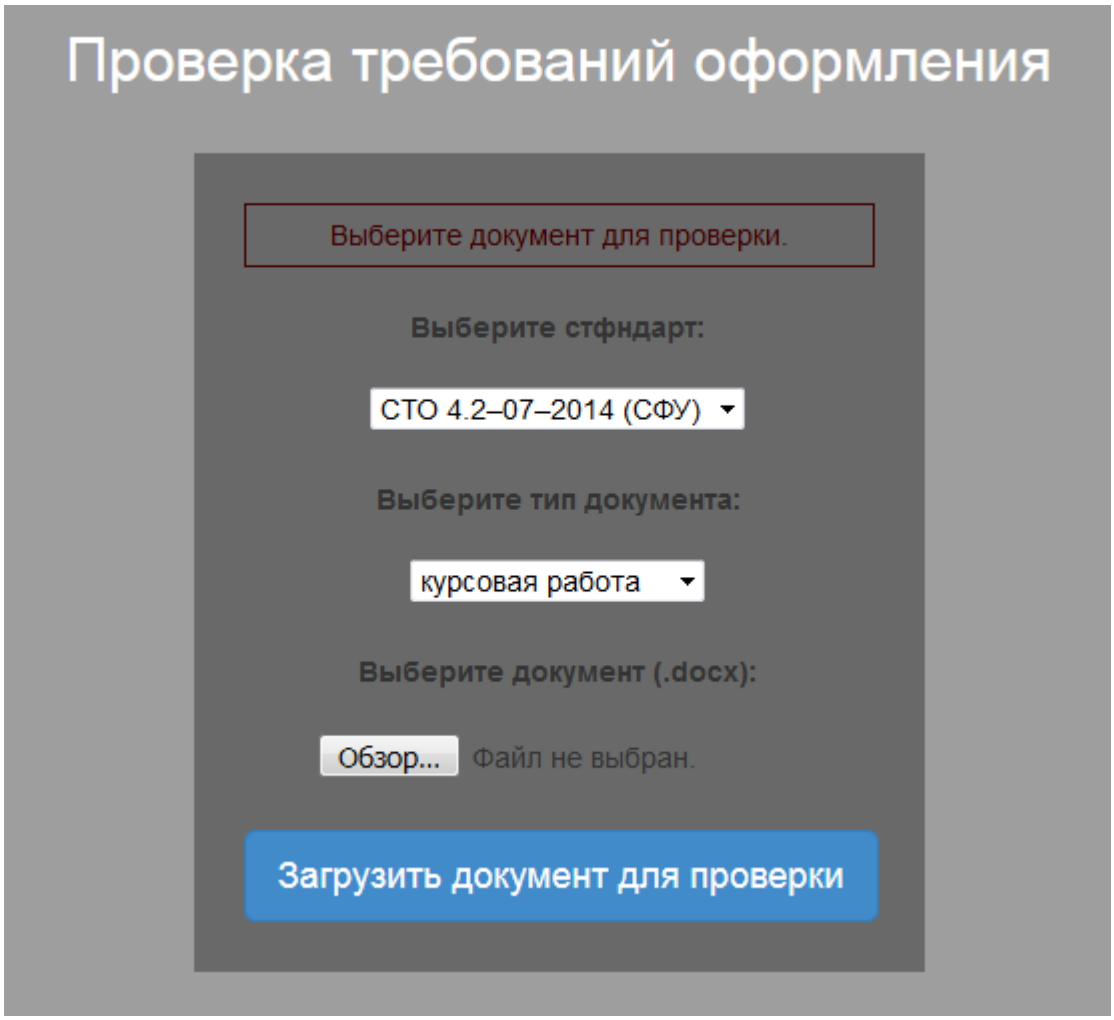

Рисунок 7 – Ошибка в форме загрузки

4.3 Реализация административной панели

На рисунке 8 изображён интерфейс административной панели.

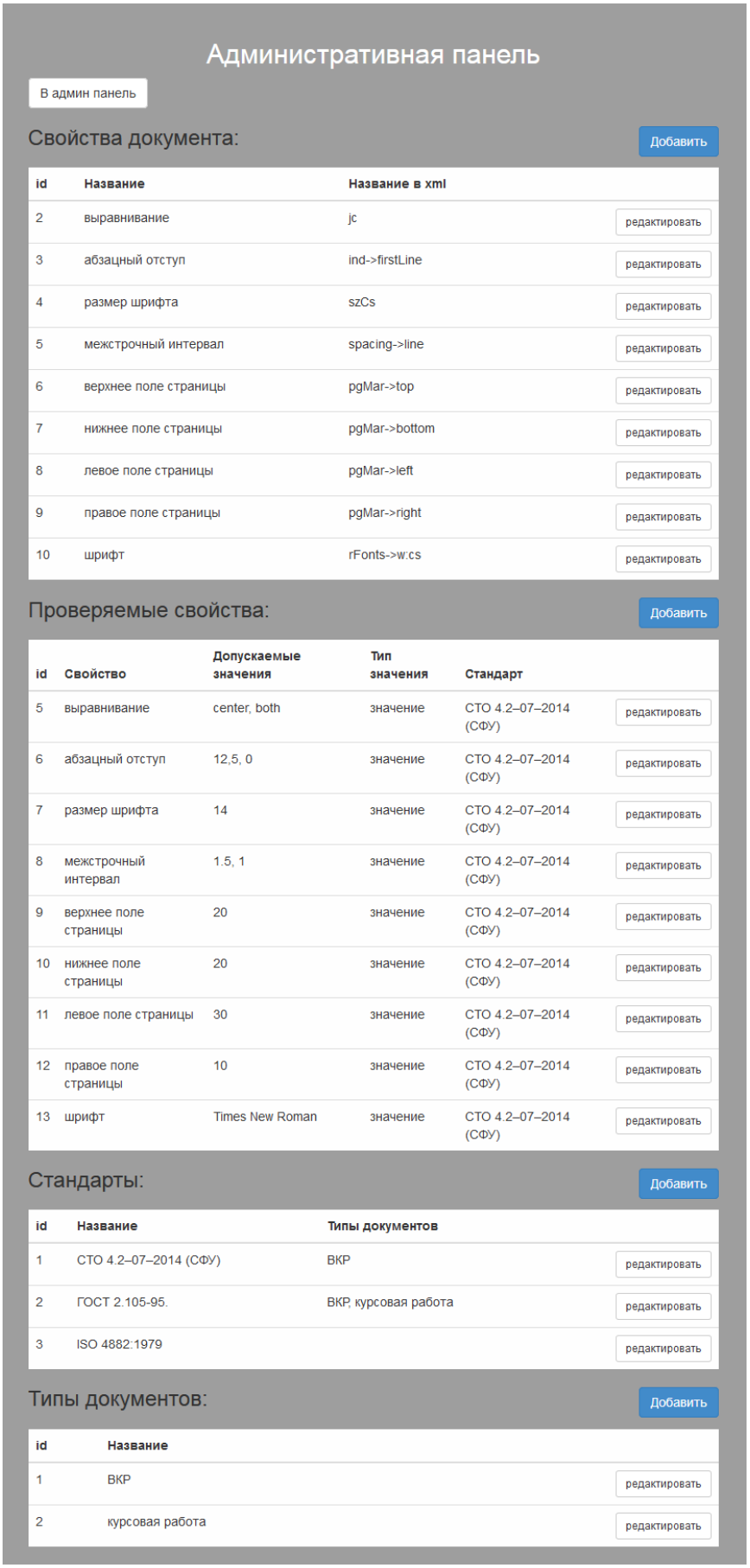

Рисунок 8 – Интерфейс административной панели

Административная панель используется для добавления и редактирования всех сущностей приложения. Кнопки «добавить» служат для перехода на страницу добавления соответствующей сущности. Кнопки редактировать предназначены для перехода на страницу редактирования соответствующего элемента таблицы.

4.4 Реализация добавления стандарта

На рисунке 9 изображён интерфейс добавления нового стандарта для добавления новых типов документов, к которым будут привязаны проверяемые стандарты.

Поля данной формы:

1. Наименование стандарта
   1.1. Ограничения: длинна 300 символов
   1.2. Тип поля: текстовый
2. Типы проверяемых документов по этому стандарту
   2.1. Ограничения: проверка наличия значения флажка в базе данных, для предотвращения записи в базу данных неверных значений
   2.2. Тип поля: флажок

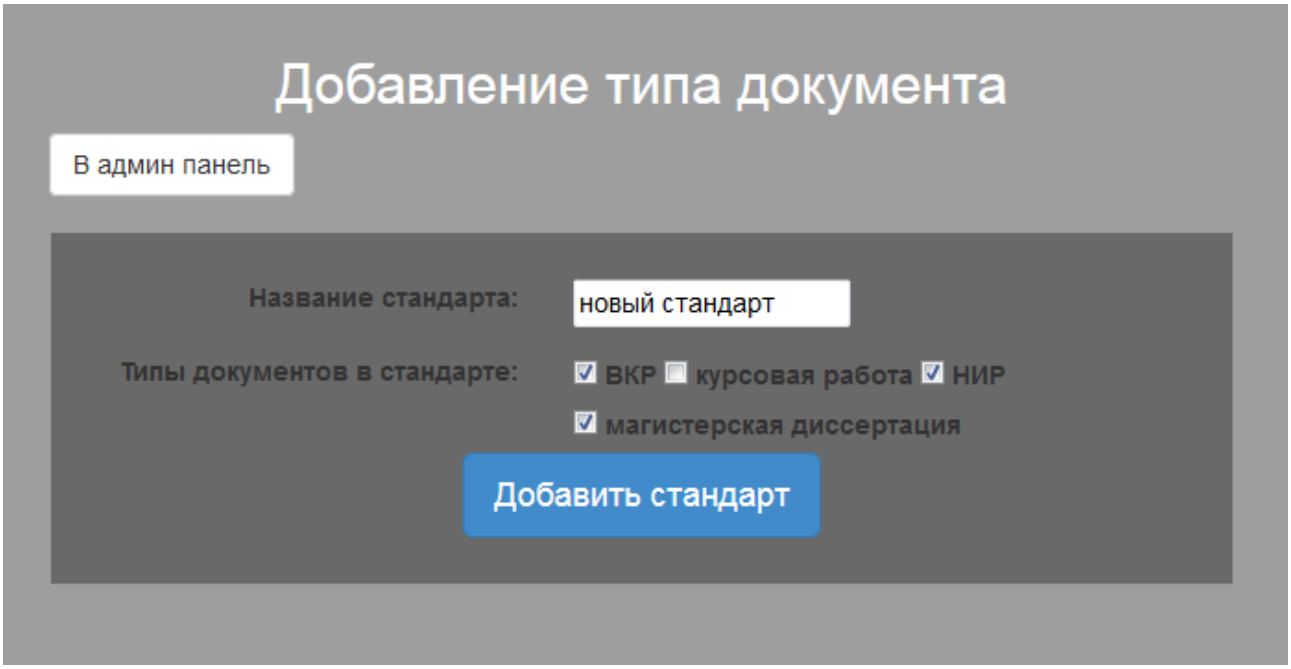

Рисунок 9 – Форма добавления стандарта

4.5 Реализация интерфейса добавления типов документов

На рисунке 10 изображён интерфейс добавления нового типа документа для привязки к какому-либо из стандартов.

Поля данной формы:

1. Название типа документов

1.1. Ограничения: длинна 300 символов

1.2. Тип поля: текстовый

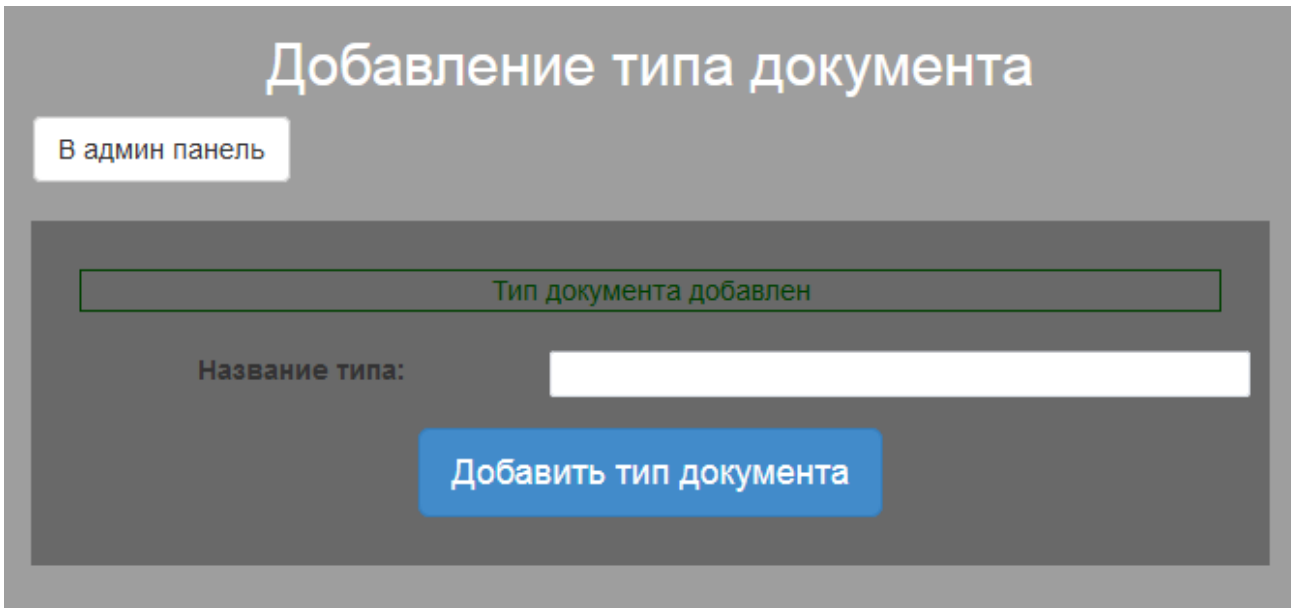

Рисунок 10 – Форма добавления типа документа

На рисунке выше представлено сообщение об успешном добавлении нового типа документа. При возникновении ошибок добавления какой-либо из форм административной части приложения будет отображена ошибка, подобная ошибке, представленной на рисунке 11.

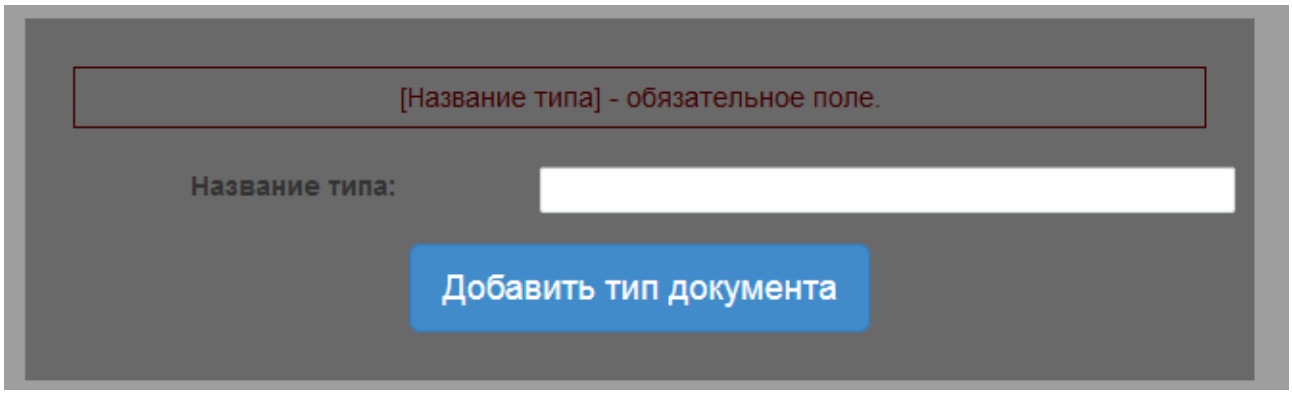

Рисунок 11 – Форма добавления типа документа

4.6 Реализация добавления свойств документа

Для того, чтобы проверять свойства документа, необходимо описать данные свойства. Для этого был реализован функционал добавления свойств документа, интерфейс которого представлен на рисунке 12.

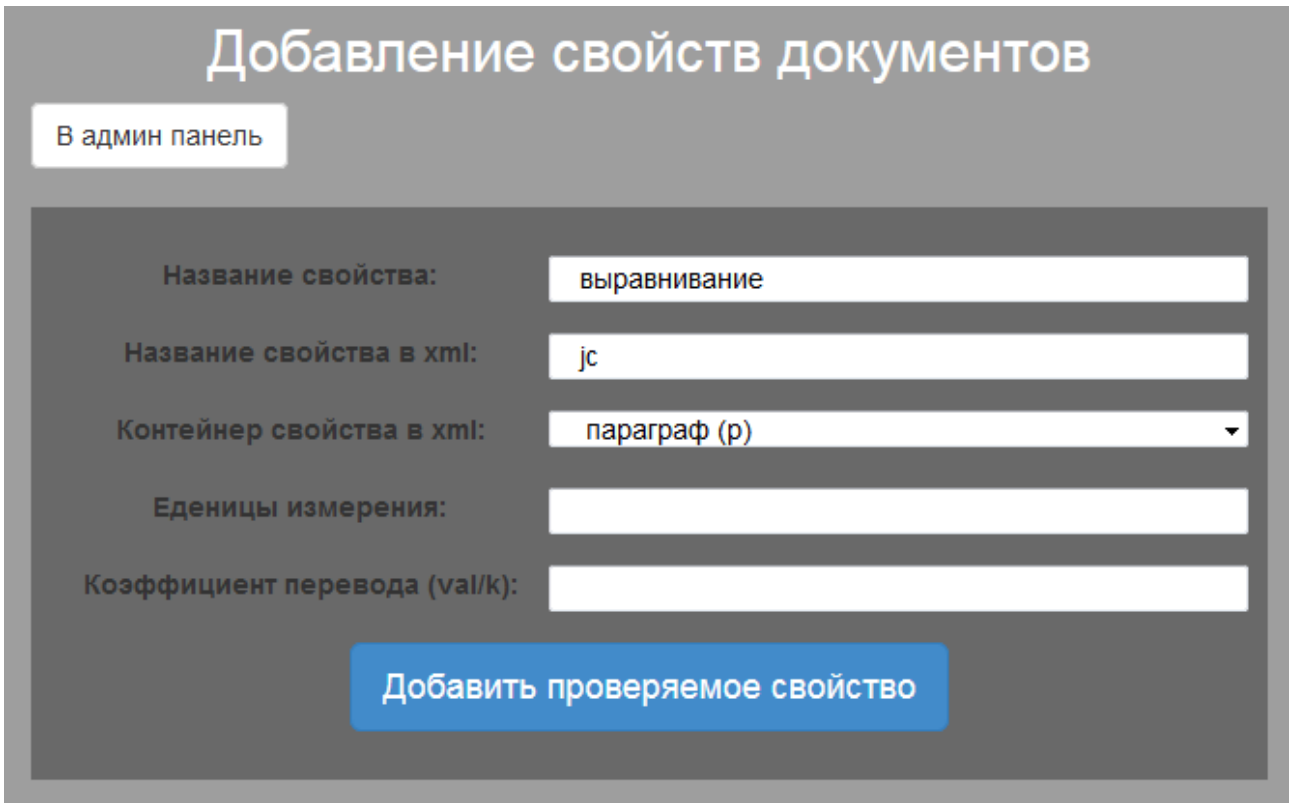

Рисунок 12 – Форма добавления свойств документов

Описание полей формы добавления свойств документов:

- «Название свойства» - используется для формирования отчёта об ошибках оформления документа;

- «Название свойства в xml» - поле, котором необходимо указать имя проверяемого параметра, используемое в xml разметке документа. В случае вложенных параметров, например как в фрагменте «<w:pgSz w:w="11906" w:h="16838" />», свойство «w», необходимо указать с учётом вложенности этого свойства через запятую, сначала родительское свойство, затем дочернее;

- «Контейнер свойства» - это поле указывает в каком контейнере располагается свойство, в документе стандарта Open Office XML, такими контейнерами могут служить, например, параграфы («p»), прогоны («run») или контейнер параметров страницы («sectPr»);

- «Единицы измерения» - поле, для указания названия единиц измерения параметра, например, «мм» или «пт»;

- «Коэффициент перевода» - поле, служащее для указания коэффициента перевода для единиц измерения свойства. Данное поле необходимо, так как многие свойства в xml документе измеряются в единицах отличных от используемых в описании стандартов;

4.7 Реализация добавления проверяемых свойств

Для проверки свойств документа необходимо внести в базу данных значения допустимые для того или иного стандарта. Для этих целей был реализован функционал добавления значений проверяемых свойств.

Страница с формой для добавления проверяемых значений свойств представлена на рисунке 13.

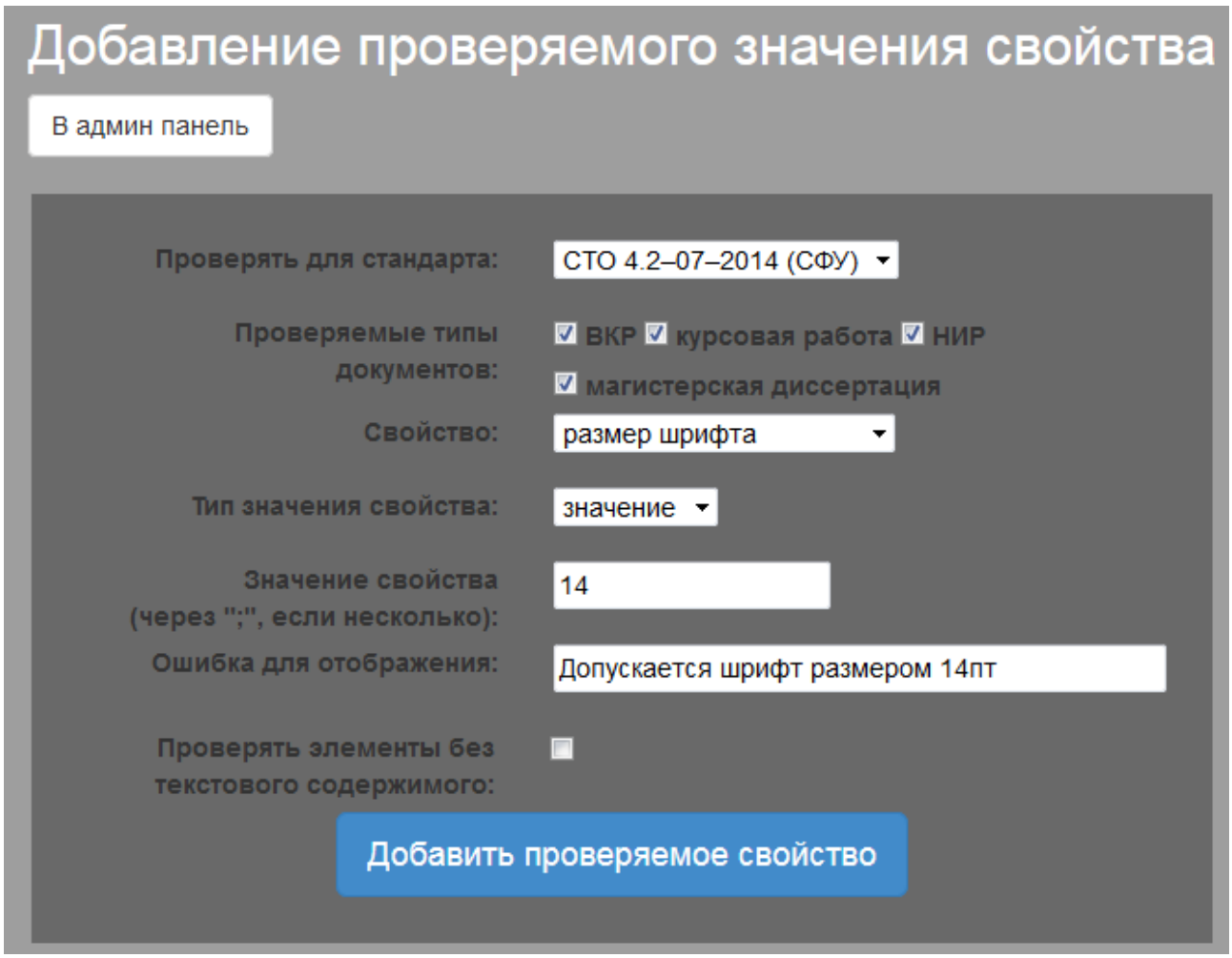

Рисунок 13 – Форма добавления проверяемых свойств

Поля формы:

- «Проверять для стандарта» - селектор для выбора стандарта, для которого проверяется значение свойства;

- «Проверять типы документов» - флажки для выбора типов документов, для которых будут проверяться значения;

- «Свойство» - селектор для выбора свойства, которое было добавлено и описано с помощью функционала добавления свойств документа;

- «Тип значения свойства» - селектор для выбора типа значения. Может принимать следующие варианты: «значение» и «диапазон»;

- «Значение свойства» - поле, в котором непосредственно записываются значения, которые указаны в стандарте. Если допускаются несколько значений, то их можно указать через «;»;

- «Ошибка для отображения» - в этом поле указывается ошибка, которая будет выведена для пояснения на странице отчёта об ошибках после проверки документа;

- «Проверять элементы без текстового содержимого» - флажок для указания, необходимо ли проверять элементы внутри которых нет текстового содержимого;

4.8 Реализация проверки документа

На этапе тестирования приложения документ проверялся на соответствие следующих требований:

- используемый шрифт;
- размер шрифта;
- межстрочный интервал;
- верхнее поле;
- нижнее поле;
- левое поле;
- правое поле;
- абзацный отступ;

Первым шагом после загрузки документа на сервер осуществляется распаковка «.docx» файла посредствам функции open встроенного в php класса «ZipArchive». После распаковки архива, интерпретируем XML файлы word/styles.xml и word/document.xml в объекты для последующей работы ними.

Функция выполняющая вышеперечисленные действия load_document получает на вход путь к архиву .docx и возвращает содержимое файла word/document.xml внутри архива.

Далее необходимо преобразовать .xml файл в массив параграфов. Функция выполняющее данное действие get_p получает на вход файл .xml и возвращает массив параграфов с его свойствами и содержимым.

Структура возвращаемого массива параграфов:

- «props» - содержит свойства параграфа

- «runs» - содержит прогоны внутри параграфа

  - «props» - содержит свойства прогона

  - «text» - содержит текстовое содержимое прогона

На этом подготовительный этап проверки документа завершается.

Проверка документа осуществляется за счёт перебора массива параграфов, который был создан на подготовительном этапе и проверке их стилей на соответствие значениям, заданным в поле «value» таблицы «check_props».

Функция, выполняющая проверку и формирующая данные для отчёта об ошибках оформления get_format_error получает на вход массив параграфов с их свойствами и содержимым, массив проверяемых параметров и их значений, массив стилей, для получения свойств параграфов и внутренних элементов, если они не указаны явно, а на выходе формирует массив с следующей структурой:

- «title» - название проверяемого свойства;

- «error» - текст ошибки;

- «gost_val» - требуемые значения свойств, указанные в стандарте;

- «doc_val» - значения свойств, встречающиеся в документе без повторов;

- «p_text» - содержимое фрагментов текста с ошибкой;

- «doc_val_not_group» - значения свойств, встречающихся в документе в порядке, в котором находятся содержимое фрагментов текста с ошибкой;

Проверяемые требования и их значения выбираются в соответствии с выбранным стандартом и типом документа.

После проверки документа формируется отчёт. Внешний вид отчёта при загрузке эталонного документа представлен на рисунке 14.

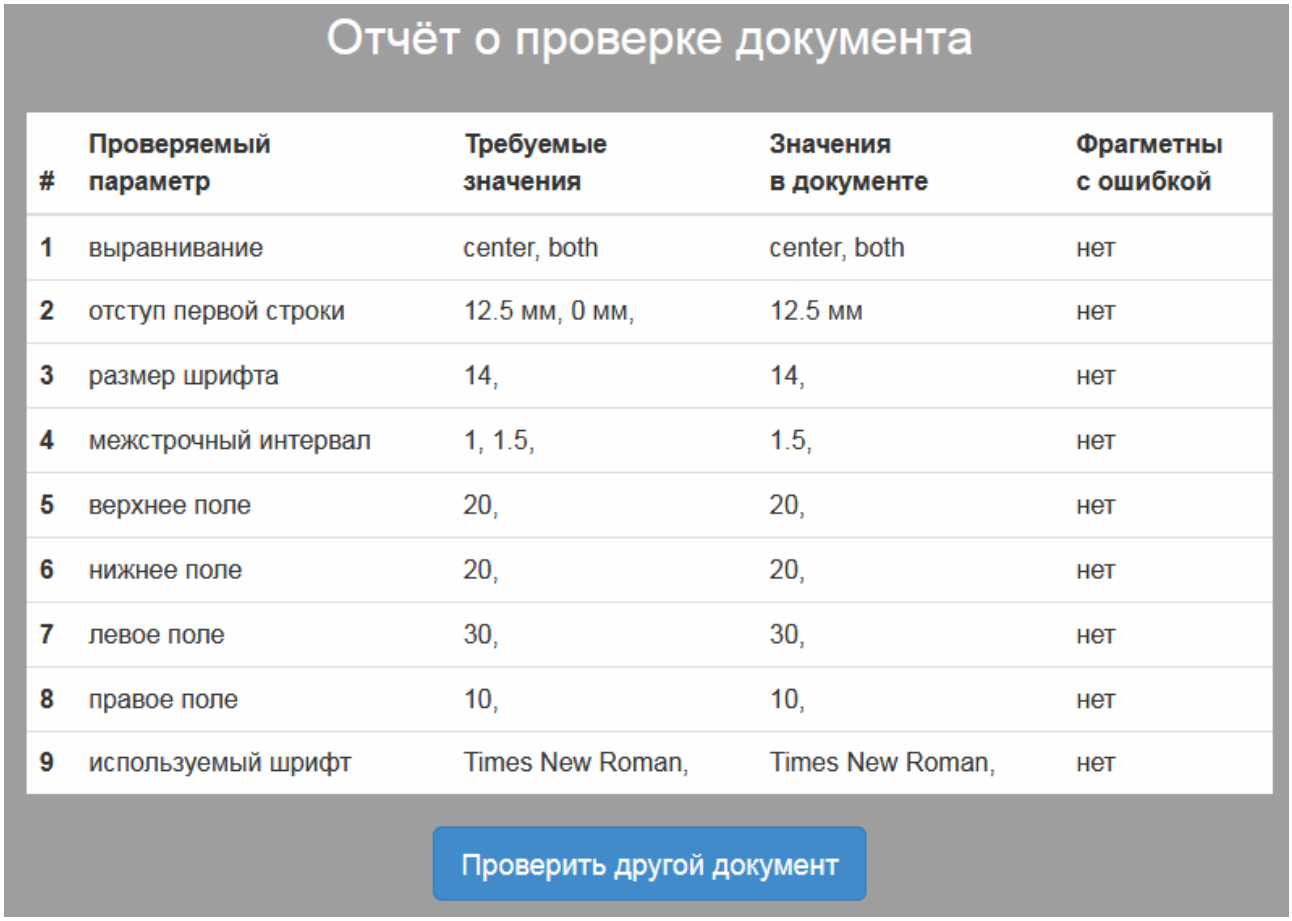

Рисунок 14 – Отчёт о проверке эталонного документа

Для проверки работы приложения изменим шрифт одного из абзацев и увеличим межстрочный интервал в эталонном документе. Результаты проверки показаны на рисунке 15. Текст в строках таблицы, где требования не соответствуют стандартам, выделяется красным цветом, а в поле «соответствие» этой строки появляется значение этого свойства в документе и фрагмент текст, которому соответствует это свойство.

При нажатии кнопки «показать фрагменты» отображаются фрагменты текста с ошибкой.

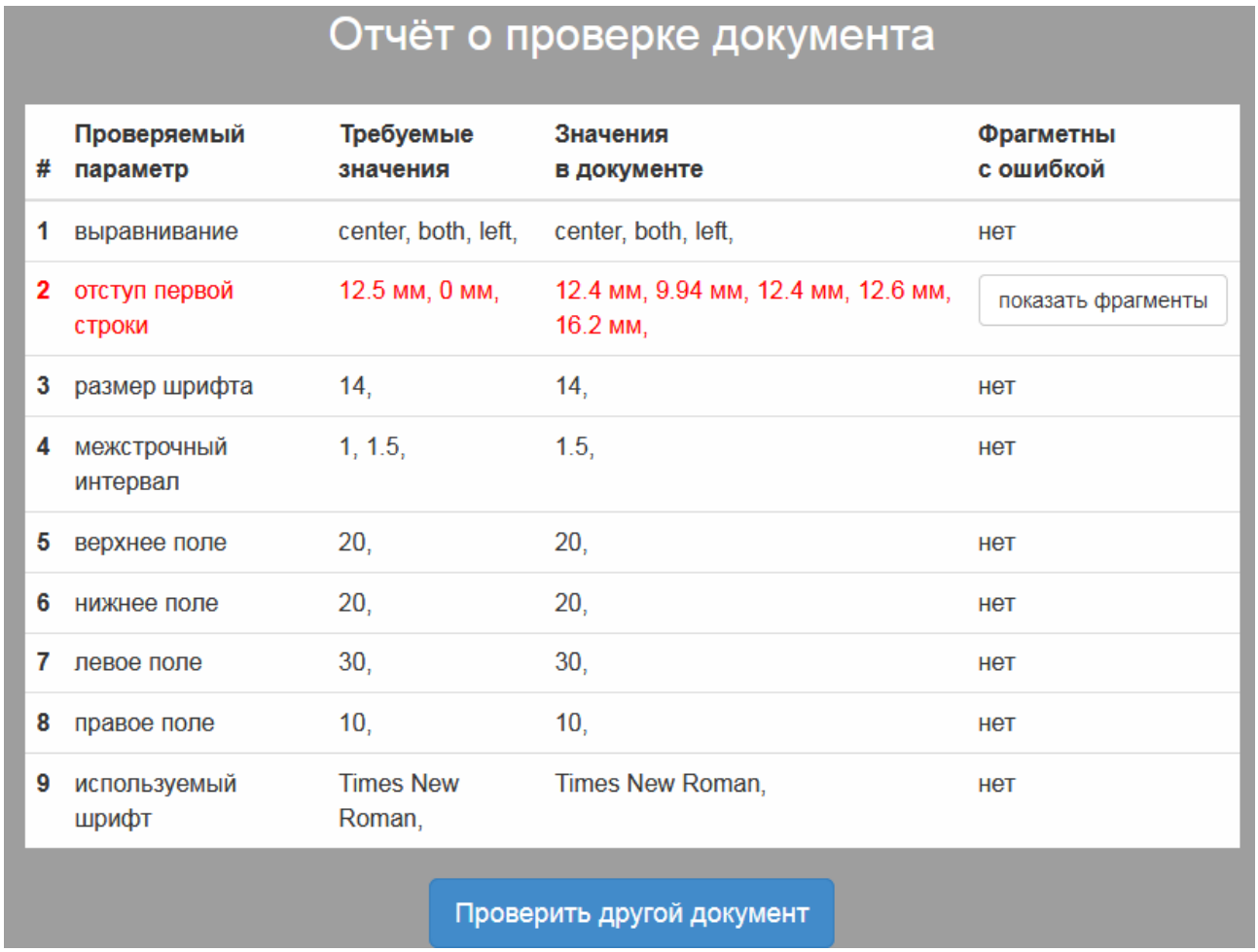

Рисунок 15 – Отчёт о проверке документа с ошибками

ЗАКЛЮЧЕНИЕ

В результате выполнения бакалаврской работы было разработано веб-приложение, позволяющее автоматически проверять электронные документы на соответствие требованиям оформления.

Для достижения результатов работы были решены следующие задачи:

- поиск и анализ существующих аналогов разрабатываемого приложения;
- изучение и выбор инструментов для реализации веб-приложения;
- проектирование веб-приложения;
- разработка веб-приложения.

После изучения и сравнении инструментальных средств для реализации, был подобран набор средств, позволяющих эффективно выполнить поставленную задачу по разработке приложения.

На этапе проектирования, были обозначены модели, контролер и представления разрабатываемого приложения.

Разработка велась с учётом информации полученной, при выборе инструментальных средств и проектировании приложения. Исходный код разработанного программного продукта представлен в приложении А.

В дальнейшем планируется внедрение веб-приложения и расширение его функционала.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. СТО 4.2-07-2014 Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности. – Введ. 9.01.2014. – Красноярск : ИПК СФУ, 2014. – 60 с.

2. Свободная энциклопедия Википедия [Электронный ресурс]: Office Open XML – Режим доступа: https://ru.wikipedia.org/wiki/Office_Open_XML

3. Нормоконтроль СГА. Программа для проверки курсовых, дипломных (ВКР), творческих работ. [Электронный ресурс]. – Режим доступа – http://stydentamsga.ru/load/fajly/nuzhnoe/

4. Руководство Пользователя CodeIgniter 3.0 [Электронный ресурс]. – Режим доступа – http://codeigniter3.info/guide/

5. Технические статьи по Office [Электронный ресурс]. – Режим доступа – https://technet.microsoft.com/ru-ru/library/cc508891.aspx

6. Руководство по PHP [Электронный ресурс]. – Режим доступа – http://php.net/manual/ru/index.php

7. Русская документация по API jQuery [Электронный ресурс]. – Режим доступа – https://jquery-docs.ru/

8. Справочник Javascript [Электронный ресурс]. – Режим доступа – http://javascript.ru/manual

9. Олищук, А. Разработка Web-приложений на PHP / А. Олищук. – Москва : Вильямс, 2006. – 352 с.

10. Руководство по phpMyAdmin [Электронный ресурс]. – Режим доступа – https://php-myadmin.ru/doc/

11. Справочное руководство по MySQL [Электронный ресурс]. – Режим доступа – http://www.mysql.ru/docs/man/

12. Описание стандарта OOXML [Электронный ресурс]. – Режим доступа – http://officeopenxml.com/

13. Open Server – руководство пользователя [Электронный ресурс]. – Режим доступа – https://ospanel.io/docs/

14. Основы Bootstrap [Электронный ресурс]. – Режим доступа – http://bootstrap-3.ru/getting-started.php

15. Руководство по CSS [Электронный ресурс]. – Режим доступа – https://developer.mozilla.org/ru/docs/Web/CSS/Reference

16. Справочник по HTML [Электронный ресурс]. – Режим доступа – http://htmlbook.ru/html

17. Пирогов, В. Ю. Информационные системы и базы данных: организация и проектирование / В. Ю. Пирогов. – Санкт-Петербург : БХВ-Петербург, 2009. – 87 с.

18. Архитектурный паттерн MVC [Электронный ресурс]. – Режим доступа – http://artanovy.com/2011/03/arhitekturnyj-pattern-mvc/

19. Строительство Web-сайтов / В. А. Фридман, А. В. Александров, Г. Г. Сергеев, С. П. Костин. – Москва : Триумф, 2011. – 288 с.

ПРИЛОЖЕНИЕ А

Контроллер

```
<?php
class Pages extends CI_Controller {

        public function __construct()
   {
      parent::__construct();
      $this->load->helper(array('form', 'url'));
                $this->load->model('properties_model');
                $this->load->model('check_props_model');
                $this->load->model('doc_type_model');
                $this->load->model('standart_model');
   }

        public function about($error = 0)
        {
                $data['title'] = 'Отчёт о проверке курсовой работы;
                $data['error'] = $error;

                $this->load->view('templates/header', $data);
                $this->load->view('pages/about', $data);
                $this->load->view('templates/footer', $data);
        }

        //
        //ГЛАВНАЯ СТРАНИЦА
        //

        public function index($page = 'home')
        {
                if ( ! file_exists(APPPATH.'/views/pages/'.$page.'.php')){
                        //нет такой страницы!
                        show_404();
                        echo "test";
                }

                $data['title'] = 'Проверка требований оформления';
                $data['error'] = '';
```

```
                $this->load->view('templates/header', $data);
                $this->load->view('pages/'.$page, $data);
                $this->load->view('templates/footer', $data);
        }

        //
        //СТРАНИЦА ОТЧЁТА О ПРОВЕРКЕ
        //

        public function do_upload()
{
        $config['upload_path']          =  './uploads/';
        $config['allowed_types']        = 'docx|doc';
        $config['max_size']             = 10000; //kb

        $this->load->library('upload', $config);

        if ( ! $this->upload->do_upload('userfile'))
        {
                $data['error'] = $this->upload->display_errors();
                        $data['title'] = 'Проверка требований оформления';
                        $this->load->view('templates/header', $data);
                        $this->load->view('pages/home', $data);
                        $this->load->view('templates/footer', $data);
        }
        else
        {
                $data['upload_data'] = $this->upload->data();
                        $data['title'] = 'Документ успешно загружен';
                        $this->load->view('templates/header', $data);
                        $this->load->view('pages/upload_success', $data);
                        $this->load->view('templates/footer', $data);
        }
}

        //
        //Добавление свойств документов
        //

        public function add_properties($success = '')
        {
                $this->load->helper('form');
                $this->load->library('form_validation');
```

```
		$data['content'] =" ";
		$data['title'] = 'Добавление свойств документов';
		$data['success'] = $success;


		$this->form_validation->set_rules('title', 'Название свойства', 'required');
		$this->form_validation->set_rules('props_name', 'Название свойства в xml', 'required');
		$this->form_validation->set_rules('props_container', 'Контейнер свойства в xml', 'required');

		if ($this->form_validation->run() === FALSE)
		{
			$this->load->view('templates/header', $data);
			$this->load->view('pages/add_properties', $data);
			$this->load->view('templates/footer', $data);
		}
		else
		{
			$this->properties_model->set_properties();
			$data['success'] = "свойство успешно добавлено";
			$this->load->view('templates/header', $data);
			$this->load->view('pages/add_properties', $data);
			$this->load->view('templates/footer', $data);
		}
	}

	//
	//Добавление проверяемого значения свойства
	//

	public function add_check_props($success = ''){
		$this->load->helper('form');
		$this->load->library('form_validation');

		$data['content'] =" ";
		$data['title'] = 'Добавление проверяемого значения свойства';
		$data['success'] = $success;

		$data['properties'] = $this->properties_model->get_properties();
		$data['doc_type'] = $this->doc_type_model->get_doc_type();
		$data['standart'] = $this->standart_model->get_standart();
```

```
            $this->form_validation->set_rules('value', 'Значение свойства',
'required');
            $this->form_validation->set_rules('error', 'Ошибка для отображения',
'required');

            if ($this->form_validation->run() === FALSE)
            {
                $this->load->view('templates/header', $data);
                $this->load->view('pages/add_check_props', $data);
                $this->load->view('templates/footer', $data);
            }
            else
            {
                $this->check_props_model->set_check_props();
                $data['success'] = "проверяемое значение свойства добавлено";
                $this->load->view('templates/header', $data);
                $this->load->view('pages/add_check_props', $data);
                $this->load->view('templates/footer', $data);
            }
        }

        //
        //Добавление типа документа
        //

        public function add_doc_type($success = ''){
            $this->load->helper('form');
            $this->load->library('form_validation');

            $data['content'] =" ";
            $data['title'] = 'Добавление типа документа';
            $data['success'] = $success;

            $this->form_validation->set_rules('title', 'Название типа', 'required');

            if ($this->form_validation->run() === FALSE)
            {
                $this->load->view('templates/header', $data);
                $this->load->view('pages/add_doc_type', $data);
                $this->load->view('templates/footer', $data);
            }
            else
            {
                $this->doc_type_model->set_doc_type();
```

```
                $data['success'] = "Тип документа добавлен";
                $this->load->view('templates/header', $data);
                $this->load->view('pages/add_doc_type', $data);
                $this->load->view('templates/footer', $data);
            }
        }

        //
        //Добавление стандарта
        //

        public function add_standart($success = ''){
            $this->load->helper('form');
            $this->load->library('form_validation');

            $data['content'] =" ";
            $data['title'] = 'Добавление типа документа';
            $data['success'] = $success;
            $data['doc_type'] = $this->doc_type_model->get_doc_type();

            $this->form_validation->set_rules('title', 'Название стандарта',
'required');

            if ($this->form_validation->run() === FALSE)
            {
                $this->load->view('templates/header', $data);
                $this->load->view('pages/add_standart', $data);
                $this->load->view('templates/footer', $data);
            }
            else
            {
                $this->standart_model->set_standart();
                $data['success'] = "Стандарт добавлен";
                $this->load->view('templates/header', $data);
                $this->load->view('pages/add_standart', $data);
                $this->load->view('templates/footer', $data);
            }
        }

        //
        //Административаня панель
        //

        public function admin($success = ''){
```

```
                $data['content'] =" ";
                $data['title'] = 'Административная панель';
                $data['success'] = $success;

                $data['properties'] = $this->properties_model->get_properties();
                $data['check_props'] = $this->check_props_model->get_check_props();
                $data['doc_type'] = $this->doc_type_model->get_doc_type();
                $data['standart'] = $this->standart_model->get_standart();

                $this->load->view('templates/header', $data);
                $this->load->view('pages/admin', $data);
                $this->load->view('templates/footer', $data);

        }
}
```

Модель Check_props_model

```
<?php
class Check_props_model extends CI_Model {

   public function __construct()
   {
      $this->load->database();
   }

        public function get_check_props($props_name = FALSE){
      if ($props_name === FALSE)
      {
                        $this->db->select('standart.title as s_title, properties.title as p_title,
check_props.value, check_props.id, check_props.values_type');
                        $this->db->join('properties', 'properties.id =
check_props.properties_id', 'left');
                        $this->db->join('standart', 'standart.id = check_props.standart_id',
'left');

                        $query = $this->db->get('check_props');
```

```
                $props_tmp = $query->result_array();
                $check_props = array();
                foreach($props_tmp as $p){
                        $p['value'] = unserialize($p['value']);
                        $check_props[] = $p;
                }

          return $check_props;
        }
        $query = $this->db->get_where('check_props', array('props_name' =>
$props_name));
        return $query->row_array();
          }

          public function set_check_props(){
                $value = serialize(explode(";", $this->input->post('value')));

                $data = array(
                        'properties_id' => $this->input->post('properties'),
                        'values_type ' => $this->input->post('values_type'),
                        'value' => $value,
                        'error' => $this->input->post('error'),
                        'check_text_content' => ($this->input->post('check_text_content')?
1 : 0),
                        'standart_id' => $this->input->post('standart'),
                );

                $this->db->insert('check_props', $data);
                $id = $this->db->insert_id();

                if(null !== $this->input->post('doc_type')){
                        $doc_type = $this->input->post('doc_type');
                        foreach($doc_type as $d){
                                $this->db->insert('check_doc_type',
['check_props_id'=>$id, 'doc_type_id'=>$d]);
                        }
                }

                return $this->db->error();
        }
}
```

Модель Doc_type_model

```
<?php
class Doc_type_model extends CI_Model {

   public function __construct()
   {
      $this->load->database();
   }

        public function get_doc_type($id = FALSE){
      if ($id === FALSE)
      {
            $query = $this->db->get('doc_type');
            return $query->result_array();
      }
      $query = $this->db->get_where('doc_type', array('id' => $id));
      return $query->row_array();
        }

        public function set_doc_type(){
                $data = array(
                        'title' => $this->input->post('title'),
                );
                return $this->db->insert('doc_type', $data);
        }
}
```

Модель Properties_model

```
<?php
class Properties_model extends CI_Model {

   public function __construct()
   {
      $this->load->database();
   }

        public function get_properties($props_name = FALSE)
        {
      if ($props_name === FALSE)
      {
            $query = $this->db->get('properties');
            //return $query->result_array();
```

```
        }
                else{
                        $query = $this->db->get_where('properties', array('props_name'
=> $props_name));
                }
                $props_tmp = $query->result_array();
                $props = array();
                foreach($props_tmp as $p){
                        $p['props_name'] = unserialize($p['props_name']);
                        $props[] = $p;
                }
        return $props;
          }

          public function set_properties()
          {
                $props_name = serialize(explode(",", $this->input-
>post('props_name')));

                $data = array(
                        'title' => $this->input->post('title'),
                        'props_name' => $props_name,
                        'props_container' => $this->input->post('props_container'),
                        'units' => $this->input->post('units'),
                        'units_convert' => $this->input->post('units_convert'),
                );
                return $this->db->insert('properties', $data);
          }
}
```

Модель Standart_model

```
<?php
class Standart_model extends CI_Model {

   public function __construct()
   {
      $this->load->database();
   }

        public function get_standart($id = FALSE){
      if ($id === FALSE)
      {
```

```
        //$query = $this->db->join('standart_doc_type', 'standart_doc_type.standart_id
= standart.id', 'left')->get('standart');
                $query = $this->db->get('standart');
                $standart_tmp = $query->result_array();
                $standart = array();

                foreach($standart_tmp as $s){
                        $query = $this->db->join('doc_type', 'doc_type.id =
standart_doc_type.doc_type_id', 'left')->get_where('standart_doc_type',
array('standart_id' => $s['id']));
                        $doc_type = $query->result_array();
                        $s['doc_type'] = $doc_type;
                        $standart[] = $s;
                }

        //return $query->result_array();
                return $standart;
    }
    $query = $this->db->get_where('standart', array('id' => $id));

            //$this->db->join('comments', 'comments.id = blogs.id', 'left');

    return $query->row_array();
      }

      public function set_standart(){
            $data = array(
                'title' => $this->input->post('title'),
                //'doc_type' => serialize($this->input->post('doc_type')),
            );
            $this->db->insert('standart', $data);
            $id = $this->db->insert_id();

            $doc_type = $this->input->post('doc_type');

            foreach($doc_type as $d){
                $this->db->insert('standart_doc_type', ['standart_id'=>$id,
'doc_type_id'=>$d]);
            }
      }
}
```

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных Технологий
институт

Информационные системы
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой ИС
_____ Л.С. Троценко
подпись, дата инициалы, фамилия
«13» июня 2018 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 Информационные системы и технологии

Разработка веб-приложения для проверки документов на соответствие требованиям оформления

Руководитель _____ 13.06.18 П.П. Дьячук
подпись, дата инициалы, фамилия

Студент КИ14-13Б, 031403218 _____ 13.06.18 А.П. Туров
номер группы, зачетной книжки подпись, дата инициалы, фамилия

Нормоконтролер _____ 13.06.18 Ю.В. Шмагрис
подпись, дата инициалы, фамилия

Красноярск 2018