|  |
| --- |
| Костанайский колледж автомобильного транспорта |
| Web-сервис «Курс» |
| Пояснительная записка |
|  |
| **Выполнил: Степанюк А.А** |
| **Проверил преподаватель: Капизова А.Б** |

|  |
| --- |
| 2017 |

[ВВЕДЕНИЕ 3](#_Toc484751772)

[1 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 4](#_Toc484751773)

[1.1 Введение 4](#_Toc484751774)

[1.2 Основание для разработки 4](#_Toc484751775)

[1.3 Назначение разработки 4](#_Toc484751776)

[2 ОПИСАНИЕ ПРОГРАММЫ 5](#_Toc484751777)

[2.1 Общие сведения 5](#_Toc484751778)

[2.2 Назначение и область применения 5](#_Toc484751779)

[2.3 Функциональное назначение 5](#_Toc484751780)

[2.4 Описание логической структуры 5](#_Toc484751781)

[2.5 Входные данные 7](#_Toc484751782)

[2.6 Используемые программные средства 7](#_Toc484751783)

[3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 8](#_Toc484751784)

[3.1 Общие сведения о веб-сервисе 8](#_Toc484751785)

[3.2 Условия выполнения программы 8](#_Toc484751786)

[3.3 Характеристика сервиса 8](#_Toc484751787)

[ЗАКЛЮЧЕНИЕ 9](#_Toc484751788)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 10](#_Toc484751789)

[ПРИЛОЖЕНИЯ 11](#_Toc484751790)

ВВЕДЕНИЕ

В настоящее время развивается система электронного документооборота. Множество специалистов из различных сфер общества, ежедневно создают множество документов. Чтобы поддерживать их единообразие были разработаны ГОСТ-ы. Они позволили отчасти решить проблему оформления. Но в силу низкой компьютерной грамотности и человеческого фактора на производстве оформление документа по ГОСТу проходит крайне плохо. Было решено создать веб-сервис, который бы сгладил данную проблему, посредством автоматизации проверки соответствия ГОСТу и был доступен через сеть Интернет.

Веб-сервис имеет преимущество перед стационарным приложением, выраженное в доступности множеству людей через сеть Интернет, отсутствии необходимости занимать дисковое пространство компьютера и возможности запуска на любой платформе, которая поддерживает выход в Интернет и имеет средства для графического представления html и css кода.

Среда разработки включала следующие технологии:

* Git - система контроля версий. Позволяет сохранять результаты работы, развёртывать среду разработки на разных компьютерах, а также переключаться между версиями программ.
* Sublime text 3 - универсальный текстовый редактор, поддерживает большинство языков программирования, подстановку и авто сохранение.
* GitKraken - графический клиент для git. Позволяет графическое взаимодействие с git.
* Python 3 - язык логики сервиса

Html, CSS - средства графического отображения сервиса в сети Интернет.

1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1.1 Введение

Веб-сервис «Курс» будет предназначен для проверки документа на соответствие ГОСТ-у. Результатом работы сервиса будет вывод пользователю объектов (абзацев, таблиц, изображений) не соответствующих ГОСТу, с поясняющей надписью, о том какие параметры в нём не соответствуют ГОСТу.

1.2 Основание для разработки

В связи с тенденцией перехода на электронный документооборот, появилась необходимость в средствах автоматического форматирования документов. Так как автоматизирование данного процесса требует специфических знаний, получение которых в пределах данной практики нецелесообразно, данный сервис будет осуществлять автоматическую проверку на соблюдение условий ГОСТа, что будет призвано упростить разработку документации специалистам различных сфер общества.

1.3 Назначение разработки

Сервис будет предназначен для проверки форматирования документа, посредством загрузки его на сервер и проверки через специальный алгоритм на соответствие ГОСТу, с поддержкой информирования пользователя через вывод объектов имеющих неверное форматирование, выделение должно быть представлено в виде подсветки красным и поясняющей надписи, указывающей какие параметры неверны.

1. ОПИСАНИЕ ПРОГРАММЫ

2.1 Общие сведения

Веб-сервис «Курс» предназначен для проверки документа на соответствие ГОСТ-у. Результатом работы сервиса является вывод пользователю объектов (абзацев, таблиц, изображений) не соответствующих ГОСТу, с поясняющей надписью, о том какие параметры в нём не соответствуют ГОСТу.

Для полноценного функционирования веб-сервису «Курс» необходимы:

* ОС «Ubuntu»
* Python 3 от 3.4 или старше
* LibreOffice Writer 5.3.3.3
* Модуль для python, Django 1,5 или старше

2.2 Назначение и область применения

Сервис предназначен для специалистов различных сфер обществ, к которым предъявлены требования к оформлению документации по какому-либо ГОСТу, к примеру: в учебных заведениях (при оформлении курсовых и дипломных работ), государственных конторах и крупных предприятиях (при оформлении документации).

2.3 Функциональное назначение

Сервис производит проверку документа на соответствие установленному ГОСТу и информирует пользователя о наличии или отсутствии ошибок в форматировании. Сервис ограничен работой с документами .docx формата.

2.4 Описание логической структуры

В соответствии с философией django, логика работы сервиса разбита на модули, исходный код которых помещён в Приложения к пояснительной записке.

Загруженный файл попадает в директорию docx, которая располагается по следующему пути (относительно корневой директории проекта: «./Documents/docx».

Модуль prework (Приложение 1):

* производит подготовку документа к обработке сервисом
  + изменение имени файла в соответствии с формой «doc\_1.docx»
  + конвертирование документа в формат html
  + перенос html файла в директорию «./Documents/html»
* Возвращает имя полученного html файла

Модуль preParser(name\_file) (Приложение 2):

* Вход. Имя html файла
* Удаляет из html файла символы переноса строки
* Возвращает html содержимое html файла без символа переноса строки

Модуль parser(text), (Приложение 3):

* Вход. Содержимое html документа без символа переноса строки.
* Извлекает тело тегов <p>, которые выполняют роль абзаца.
* Возвращает словарь вида: space[порядковый номер элемента] = [Тело тега, тег]

Модуль style(text), (Приложение 4):

* Вход. Содержимое html документа без символа переноса строки.
* Извлекает и возвращает тег <style>.

Модуль test(style, space), (Приложение 5):

* Вход. Стили, извлечённые модулем style и словарь с тегами.
* Извлекает параметры документов
* Возвращает space вида: space[i] = [Тело тега, тег, марка]
  + Марка - массив вида: [Правильность, Справка, что не правильно]

Модуль generator(space), (Приложение 6):

* Вход. Получает space от модуля test.
* Генерирует html страницу.

*И* ***возвращает пользователю*** *html страницу, с результатами проверки.*

2.5 Входные данные

Сервис принимает документ формата .docx, загружаемого пользователем.

* 1. Используемые программные средства

Для преобразования .docx документа используется ПО libreoffice. Libreoffice вызывается посредством возможностей языка программирования python и конвертирует входной документ в html формат.

1. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

3.1 Общие сведения о веб-сервисе

Данный веб-сервис производит проверку переданного ему документа на соответствие ГОСТу для оформления курсовых работ в ККАТ. По окончанию проверки пользователь будет переведён на страницу сгенерированную на основе введённого документа, на которой будут отображены допущенные им ошибки при оформлении работы.

3.2 Условия выполнения программы

Пользователь должен передать серверу документ в формате .docx, после этого начнётся выполнение проверки документа.

3.3 Характеристика сервиса

Сервис находится на стадии разработки. Действующая система может работать некорректно, а визуальное оформление меняться от версии к версии.

3.4 Выполнение программы

1. Пользователь переходит по адресу: kurskkat.kz. Перед Вами откроется окно как на рисунке 3.1

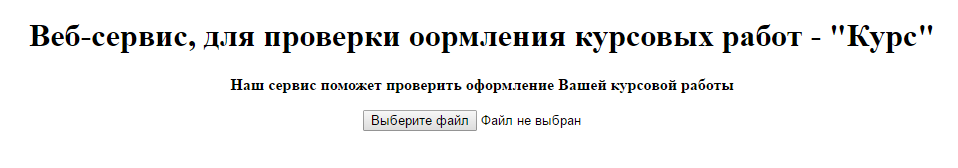


Рисунок 3.1 - Главная страница сайта

1. Для загрузки документа необходимо нажать «Выберите файл»
2. Перед Вами откроется окно как на Рисунке 3.2. Выбрав документ, необходимо нажать на кнопку «Открыть»

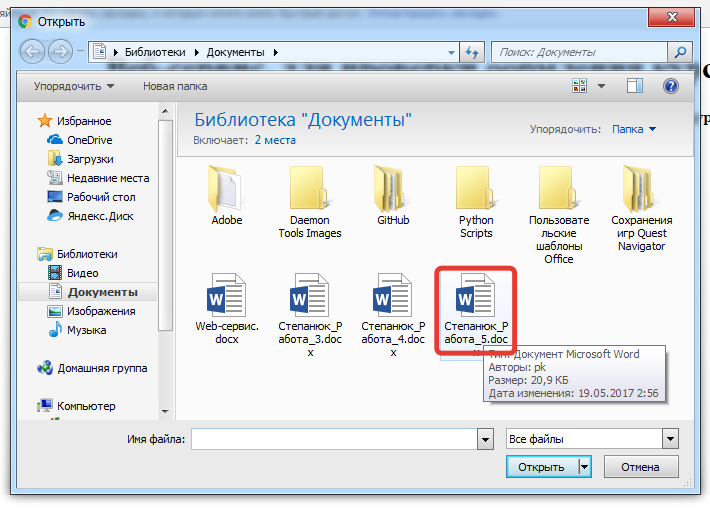


Рисунок 3.2 Выбор файла

1. После этого произойдёт загрузка документа на сервер
2. После небольшого ожидания Вам будут выведены результаты проверки как на Рисунке 3.3.

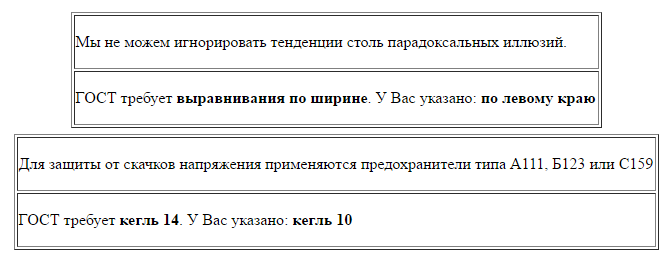


Рисунок 3.3 – Результат работы сервиса

ЗАКЛЮЧЕНИЕ

На настоящий момент разработка программы находится на стадии альфа. Требуется множество доработок (в том числе и по оформлению сайта). Визуальное оформление будет доработано в процессе.

За время разработки мною были приобретены навыки:

* Работы с системой контроля версий git (знание которого, встречается в каждой второй вакансии в интернете, на должность разработчика)
* Проектирования архитектуры серверной части (backend) веб-сервисов
* Преодоления «творческого кризиса»

Поиска и чтения документации, как на русском, так и на английском языке

Так же было усвоено, что проекты такого масштаба следует брать в качестве домашней работы, но не как курсовой проект на практику, так как время необходимое на реализацию данного проекта превышает доступное во время практики время.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. habrahabr.ru/hub/django/
2. habrahabr.ru/hub/python/
3. djbook.ru
4. github.com
5. ilnurgi1.ru/docs/python/modules/os.html
6. htmlbook.ru
7. python.org
8. djangoproject.com

ПРИЛОЖЕНИЯ

Приложение 1:

**import** os

**def** prework():

i = 0; file = ""

link = "libreoffice --convert-to html "

# Удаляем все файлы в директории html

os.chdir("./Documents/html")

**for** w **in** os.listdir():

os.remove(w)

# и возвращаемся назад

os.chdir("../docx") #\* Перейти в директорию с документом

**for** w **in** os.listdir(): #\* Обнаружить документ .docx

q = w[-5: len(w)]

**if** q == ".docx" **or** ".doc" **in** q:

i+=1; file = "doc\_" + str(i) + ".html"

os.rename(w, "doc\_"+ str(i) + ".docx") # \* Переименовать по форме

os.system(link + "doc\_"+ str(i) + ".docx") # \* Конвертировать в html

os.system("mv " + file + " ../html") # \* Переместить полученный документ в папку html

os. chdir("../html")

**return** file

Приложение 2:

def preParser(name\_file):

doc = open(name\_file, 'r') # Подключаем файл с проектом

doc\_text = doc.readlines() # Записываем его в переменную

new\_doc\_text = ""

**for** w in doc\_text:

**for** d in w:

**if** d == "\n":

new\_doc\_text += " "

**else**:

new\_doc\_text += d

doc.close()

f = open("new.html", "w")

f.write(new\_doc\_text)

f.close()

**return** new\_doc\_text

Приложение 3:

def parser(doc\_text):

f\_1 = False; f\_2 = False; f\_3 = False; f\_4 = False; f\_5 = False; f\_6 = False# Объявляем флаги

i = 0; text = ""; teg = ""; c\_teg = ""; space = {} # Объявляем вспомагательные переменные

g = 0

**for** w in doc\_text:

# Если во-второй раз встретили <, сообщаем об этом

**if** w == "<" and f\_1 == True and f\_2 == False: f\_2 = True

# Если в теле тега много других тегов. Когда проверяем будет ли этот тег нашим закрывающим тегом

**if** w == "<" and f\_2 == True: c\_teg = "";

# Если наткнулись на знак начала тега запускаем запись

**if** w == "<" and f\_1 == False: f\_1 = True

**if** f\_3 == True: text += w

# Если это не пробел, мы нашли второй тег и тег не найден

**if** w != " " and f\_2 == True and f\_6 == False:

c\_teg += w

**if** c\_teg == "</p" and teg == "<p": f\_6 = True

**if** c\_teg == "</img" and teg == "<img": f\_6 = True

**if** c\_teg == "</table" and teg == "<table": f\_6 = True

**if** len(c\_teg) > 7: f\_2 = False; c\_teg = ""

# Если запись разрешена и тег не определён

**if** f\_1 == True and f\_3 == False:

**if** w in "<pimgtableh123456": # Добавляем только те символы из которыз состоят наши теги

teg += w

**if** teg == "<p": f\_3 = True; text += " "

**if** teg == "<img": f\_3 = True; text += " "

**if** teg == "<table": f\_3 = True; text += " "

**if** len(teg) > 6: teg = ""; f\_1 = False

# Если Нашли знак закрытия тега и тег закрытия найден кончаем

**if** w == ">" and (f\_6 == True or teg == "<img"):

i+=1; space[i] = [teg + text, teg]

f\_1 = False; f\_2 = False; f\_3 = False; f\_4 = False; f\_5 = False; f\_6 = False

teg = ""; c\_teg = ""; text = ""; g = 0;

g += 1

**return** space

Приложение 3:

def styleParser(name\_file):

f\_1 = False; f\_2 = False; f\_3 = False; f\_4 = False; f\_5 = False; f\_6 = False# Объявляем флаги

i = 0; text = ""; teg = ""; c\_teg = ""; space = {} # Объявляем вспомагательные переменные

g = 0

**for** w in name\_file:

**if** w == "<" and f\_2 == False: f\_2 = True

**if** f\_3 == True: text += w

**if** w != " " and f\_2 == True and f\_3 == False:

teg += w

**if** teg == "<style": f\_3 = True;

**if** len(teg) > 6: f\_2 = False; teg = ""

**if** w == "<" and f\_3 == True: f\_4 = True

**if** w != " " and f\_4 == True and f\_5 == False:

c\_teg += w

**if** c\_teg == "</style": f\_5 = True

**if** f\_5 == True: text += ">";**break**

g+=1

**return** teg + text

Приложение 5:

f = open("./../Documents/html/data.html", "r")

data = f.readlines()

f.close()

def test\_style(teg):

teg\_text = ""; data\_style = {}; h =[" ", ":", ";", "\n", "\t", "{", "}", ""]

tt = []

**for** w in teg:

**if** w not in h:

teg\_text += w

**if** w == " ":

tt += [teg\_text]

teg\_text = ""

**if** w == "{" or w == "}": tt += w

dd = []; ww = 0; www = 0

f1 = False

i = 0; tq = []; ttq = []

**for** w in tt:

**if** w == "<style" or w == ***'type="text/css">'*** or w == ***''***: **continue**

www = ww; ww = w

**if** w == "{": f1 = True; dd = www

**if** f1:

i+=1;

**if** w == "{" or w == "}":

pass

**else**: tq += [w]

**if** i == 2: i = 0; ttq += tq; tq = []

**if** w == "}": f1 = False; i = 0; tq = []; data\_style[dd] = ttq; ttq = []

**for** w in data\_style:

kt = {}; i = 0; r = ""

**for** d in data\_style[w]:

i+=1;

**if** **int**(i) == 1: r = d

**if** i == 2:

kt[r] = d; i = 0

data\_style[w] = kt

**return** data\_style

def test\_paragraph(teg):

nur = []

teg\_text = ""; data\_style = {}; h =[" ", ":", ";", "\n", "\t", "{", "}", "", "=", "\'", "\"", "<", ">"]

progon\_1 = []

**for** w in teg:

**if** w not in h:

teg\_text += w

**if** w == " " or w == ">" or w == "<" or w == "=":

progon\_1 += [teg\_text]

teg\_text = ""

**if** w == "{" or w == "}" or w == ">" or w == "<": progon\_1 += w

progon\_2 = {}; body = []

f1 = False; f2 = False; key = ""

**for** w in progon\_1:

**if** f2: body+=[w];

**if** f1: key = w; f1 = False; f2 = True

**if** w == ">": f2 = False; progon\_2[key] = body; body = []

**if** w == "<": f1 = True

ty = []

**for** w in progon\_2:

**if** w[0] == "/" or w == ">": ty += [w]

**for** w in ty:

del progon\_2[w]

**for** w in progon\_2:

kt = []

**for** d in progon\_2[w]:

**if** not (d == ">" or d == "style" or d == ’’): kt += [d]

progon\_2[w] = kt

**for** w in progon\_2: # Получаем список ключей

kt = {}; i = 0; r = "";

**for** d in progon\_2[w]: # Получаем с

i+=1

**if** **int**(i) == 1: r = d

**if** i == 2: i = 0; kt[r] = d; r = ""

progon\_2[w] = kt

**return** progon\_2

g=0; data\_style = []; data\_paragraph = {}

**for** w in data:

g+=1

**if** g == 1: data\_style = test\_style(w)

**if** g > 1: data\_paragraph[g] = [test\_paragraph(w)]

f = open("data\_new.html", 'w')

**for** w in data\_style:

text = w + " --> " + str(data\_style[w])

f.write(text)

f.write("\n")

**for** w in data\_paragraph:

text = str(w) + " --> " + str(data\_paragraph[w])

f.write(text)

f.write("\n")

f.close()

Приложение 6:

def generator(space):

i = 0; html = """

<style>

    .brd {

    border: 4px double black; /\* Параметры границы \*/

     padding: 10px; /\* Поля вокруг текста \*/

    }

   </style>

"""

#print(html)

**while** i < len(space):

i+=1

line = '<div class = "brd">' + space[i][0] + '</div>\n'

html += line

doc = open("dinamic.html", "w")

doc.write(html)