*Федеральное государственное бюджетное образовательное учреждение* *высшего образования*

|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(национальный исследовательский университет)*** |

ФАКУЛЬТЕТ \_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_Компьютерные Системы и сети (ИУ6)\_\_\_\_\_\_\_\_\_\_\_

**Отчет**

**по лабораторной работе № 10**

**Название лабораторной работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Дисциплина:** Языки интернет программирования

Студент гр. ИУ6-31б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И С Марчук**

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О. Фамилия)

Москва, 2019

*Федеральное государственное бюджетное образовательное учреждение* *высшего образования*

## *Задание* Вариант 1.16.

## ЛР 8. Ruby on Rails

Разработать веб-приложение, имеющее HTML-страницу с формой ввода данных и HTML-страницу для представления результатов. Результат

расчёта должен быть представлен в форме таблицы, оформленной с помощью элемента table или отдельными ячейками div и имеющей не менее

двух колонок. Если по условию задания результат может быть представлен только в виде одной строки таблицы, необходимо реализовать вывод

промежуточных результатов расчёта в качестве дополнительных строк.

В этом случае первой колонкой таблицы будет порядковый номер итерации.

Под вводом с клавиатуры в тексте заданий следует понимать ввод в

поле ввода данных формы на HTML-странице.

**Текст задания:**

Дано натуральное число m. Написать программу, определяющую такое

натуральное число n, что двоичная запись числа n получается из двоичной записи числа m изменением порядка цифр на обратный порядок их

следования. Например: 6 = 110 , а 3 = 011. Вывести на печать числа и

их двоичное представление.

**ЛР 10. Формирование и отображение XML в HTML средствами сервера и клиента.**

Модифицировать код ЛР 8 таким образом, чтобы по запросу с указанными параметрами выдавался результат в формате XML (средствами стандартной сериализации ActiveSupport).

* Проверить формирование XML и сохранить в файл для отладки XSLT и второго приложения.
* Написать функциональный тест, проверяющий формат выдаваемых данных при запросе RSS.

Разработать XSLT-программу преобразования полученной XML в HTML. Добавить в проверяемый XML-файл строку привязки к преобразованию <?xml-stylesheet type="text/xsl" href="some\_transformer.xslt"?>. Проверить корректность отображения браузером результата преобразования. Проверить на автономной Ruby-программе корректность преобразования, используя следующий фрагмент кода:

require 'nokogiri'

doc = Nokogiri::XML(File.read('some\_file.xml'))

xslt = Nokogiri::XSLT(File.read('some\_transformer.xslt'))

puts xslt.transform(doc)

Разработать второе приложение, являющееся посредником между клиентом и первым приложением, задачей которого является преобразование XML в HTML или передача в неизменном виде браузеру для отображения браузером. Приложение должно запускаться с указанием номера порта TCP, отличным от номера порта первого приложения (например rails server -p 3001)!

* Подготовить каркас приложения, а также форму формирования запроса, форму отображения результата и соответствующие действия контролера.
* Добавить в контроллер преобразование XML в HTML с помощью ранее разработанного XSLT-файла.
* Подключить запрос XML с первого приложения и проверить работу приложений в связке.
* Написать функциональный тест, проверяющий что при различных входных данных результат генерируемой страницы различен.
* Доработать код контроллера и представлений данного приложения для выдачи браузеру XML-потока в неизменном виде (организовать возможность выбора формата выдачи для пользователя).
* Проверить, что браузер получает XML первого приложения в неизменном виде.
* Доработать код контроллера приложения таким образом, чтобы XML-поток первого приложения получал дополнительную строку, указывающую xsl. Модифицировать форму запроса параметров таким образом, чтобы браузер получал в ответ XML. При этом разместить XSLT-файл в директории public.
* Проверить, что браузер производит преобразование XML->HTML в соответствии с xlt.
* Реализовать функциональные тесты второго приложения. Проверить результаты, формируемые приложением, на соответствие выбранному формату выдачи.

Итоговая форма ввода параметра должна содержать кнопки или селектор, позволяющие проверить два варианта преобразования:

* Серверное xml+xslt->html
* Клиентское xml+xslt->html

Текст программ приложения-сервера с api

***routes.rb:***

Rails.application.routes.draw do

root 'start\_page#index'

end

***start\_page\_controller.rb:***

class StartPageController < ApplicationController

def number\_or\_zero(string)

Integer(string || 0)

rescue ArgumentError

0

end

def my\_calc\_method(qestion\_m)

m = qestion\_m

bin\_m = ' '

temp\_arr = Array.new()

ans = 0

bin\_ans = ' '

i = 0

temp\_m = m

while temp\_m > 0 do

ans = ans \* 2 + (temp\_m % 2)

bin\_ans = bin\_ans + (temp\_m % 2).to\_s

bin\_m = (temp\_m % 2).to\_s + bin\_m

# новую строку в матрицу

temp\_arr.push Array.new(3)

# ячейки строки

temp\_arr[-1][0] = ans

temp\_arr[-1][1] = bin\_ans

temp\_arr[-1][2] = bin\_m

temp\_m = temp\_m / 2

i = i + 1

end

return {qestion\_m: m, bin\_m: bin\_m, temp\_arr: temp\_arr, ans: ans, bin\_ans: bin\_ans}

end

# начальная страничка считающая значения

before\_action :parse\_params, only: :index

def index ()

respond\_to do |format|

format.xml {

@data = my\_calc\_method(@m)

render xml: @data.to\_xml

}

format.rss {

@data = my\_calc\_method(@m)

render xml: @data.to\_xml

}

end

end

def parse\_params

@m = number\_or\_zero(params[:pm])

@format = params[:format]

end

end

Текст программ приложения-посредника

***routes.rb:***

Rails.application.routes.draw do

root 'proxy#input'

get 'proxy/input'

get 'proxy/output'

# For details on the DSL available within this file, see https://guides.rubyonrails.org/routing.html

end

***proxy\_controller.rb:***

require 'nokogiri'

require 'open-uri'

class ProxyController < ApplicationController

def input

end

before\_action :parse\_params, only: :output

def output

my\_url = "http://localhost:3000/?&pm=#{@cl\_pm}&format=xml"

api\_response = open(my\_url)

if @side == 'server-xslt'

@result = xslt\_transform(api\_response).to\_html

elsif @side == 'client-xslt'

render xml: insert\_browser\_xslt(api\_response).to\_xml

else

render xml: api\_response

end

end

private

def parse\_params

@cl\_pm = params[:pm\_enter]

@side = params[:side]

end

def xslt\_transform(data, transform: "#{Rails.root}/public/server\_transform.xslt")

doc = Nokogiri::XML(data)

xslt = Nokogiri::XSLT(File.read(transform))

xslt.transform(doc)

end

def insert\_browser\_xslt(data, transform: '/browser\_transform.xslt')

doc = Nokogiri::XML(data)

xslt = Nokogiri::XML::ProcessingInstruction.new(doc, 'xml-stylesheet', 'type="text/xsl" href="' + transform + '"')

doc.root.add\_previous\_sibling(xslt)

doc

end

end

***input.html.erb:***

<%= form\_tag url\_for(controller: :proxy, action: :output), method: 'get' do %>

<label>Введите число m:</label>

<input type = "number" pattern = "^[\d]+$" name = "pm\_enter" id = "pm" placeholder="144" />

<div>

<label>Отдать с обработкой на стороне</label>

<select class="custom-select" name = "side">

<option value = "server-xslt">Server XSLT</option>

<option value = "client-xslt">Client XSLT</option>

<option value = "client">RAW</option>

</select>

</div>

<input type="submit" class = "btn btn-primary" value="субмит\_баттон"/>

<% end %>

***output.html.erb:***

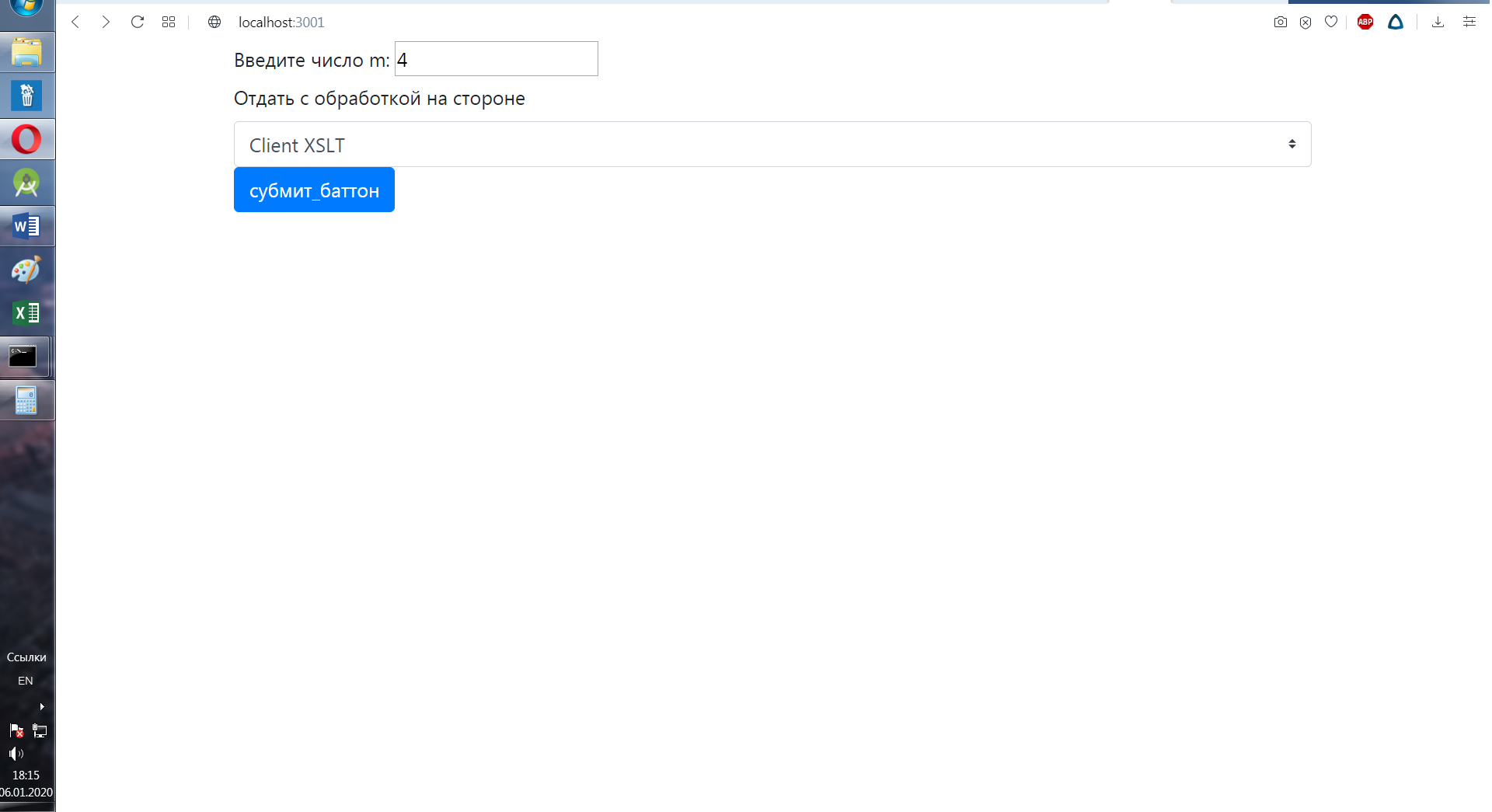
<h1>Cервер:</h1>

<%= render inline: @result %>

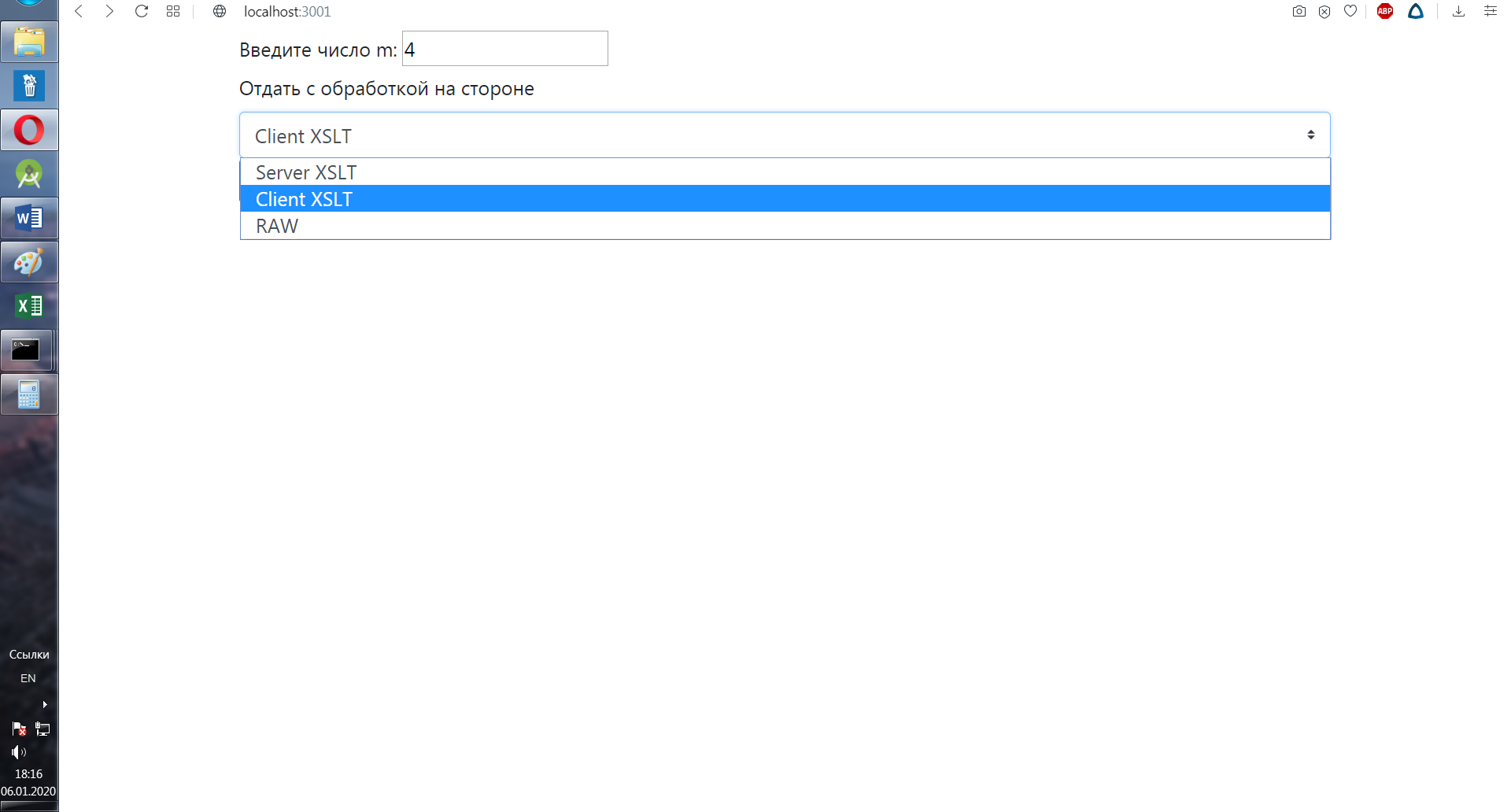
***также в gemfile была добавлена библиотека bootstrap:***

gem 'bootstrap', '~> 4.0.0.alpha6'

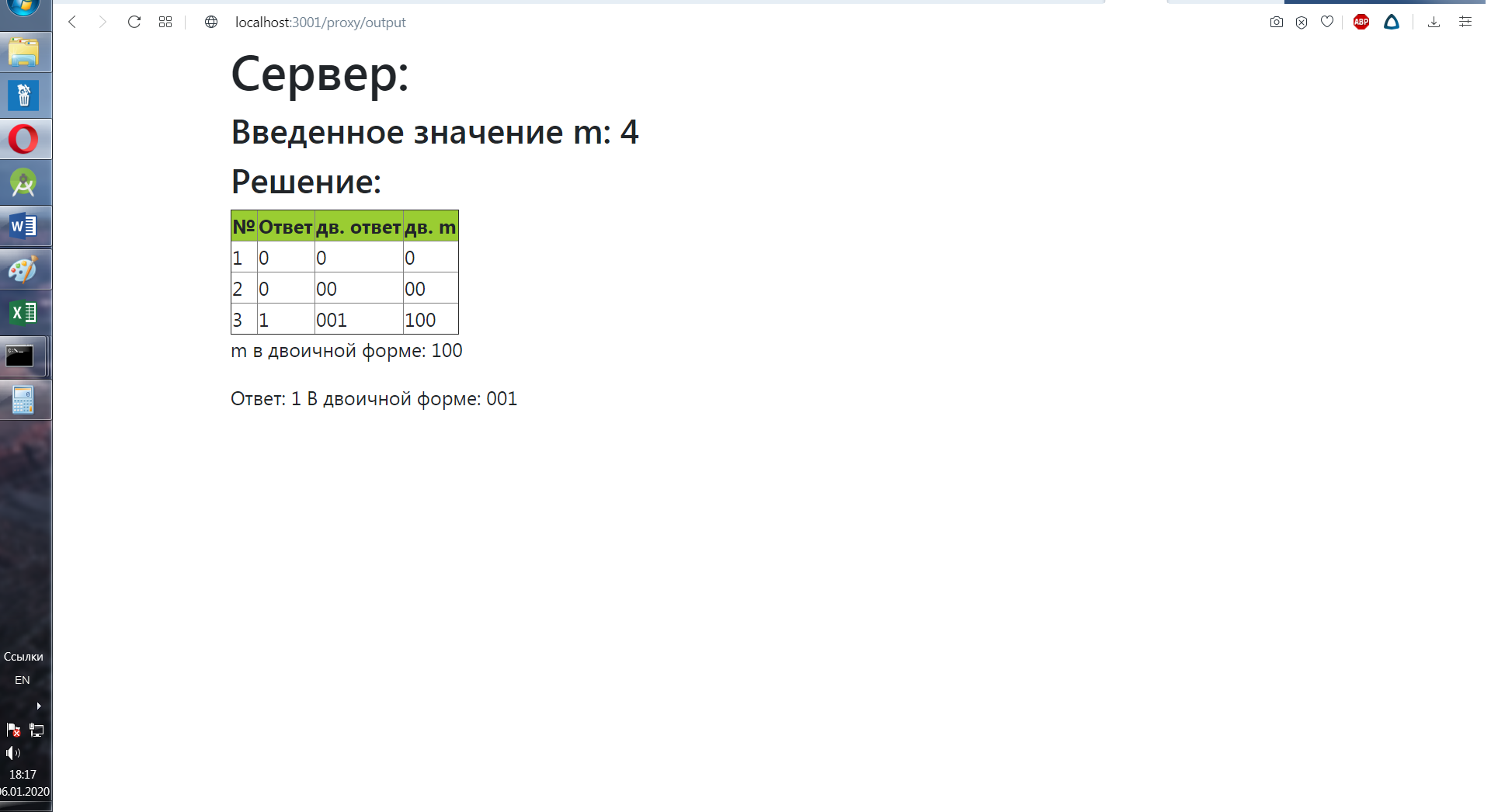
Скриншоты



Страница ввода данных



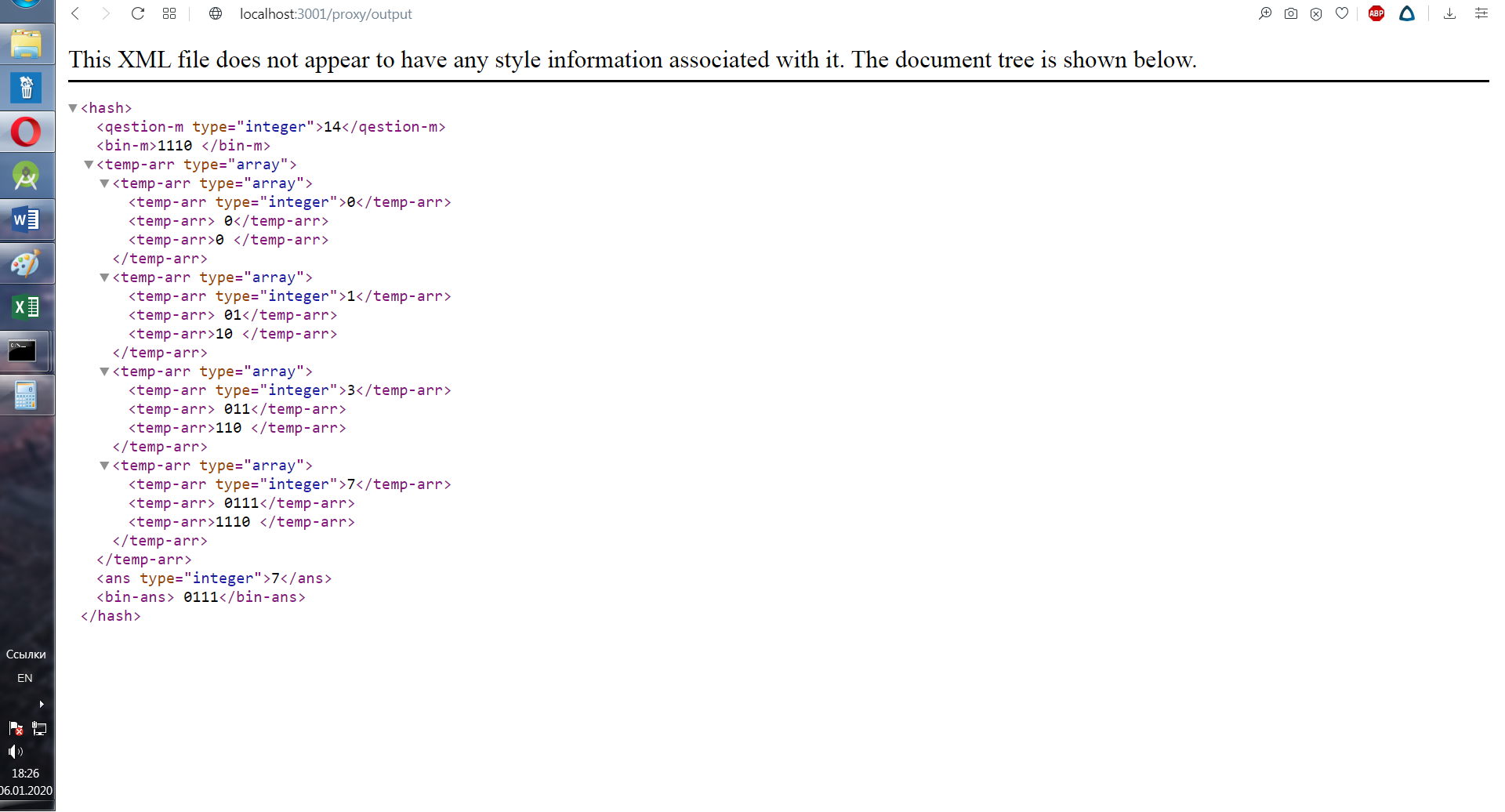
Селектор на странице ввода данных



Страница обработанная на приложении посреднике при выборе пункта “server XSLT”

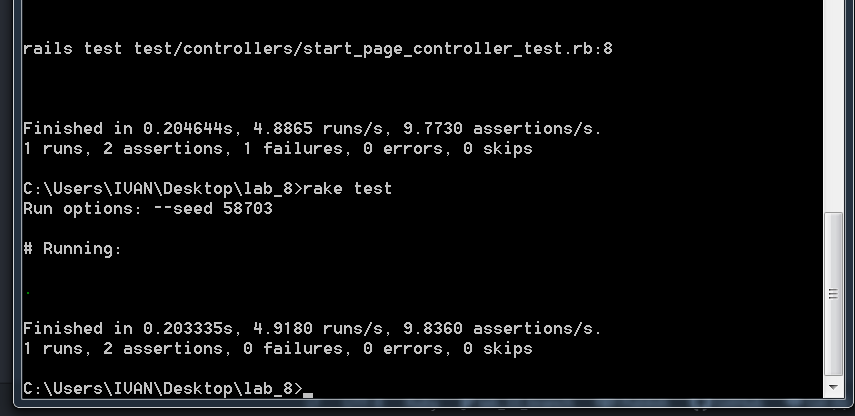


Страница обработанная в браузере при выборе пункта “client XSLT”



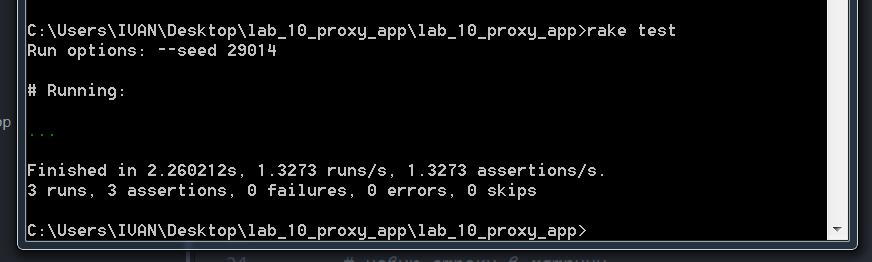
XML с серверного приложения без обработки при выборе пункта “raw”

Тестирование серверного приложения



Результаты теста

Тестирование приложения посредника



Результаты теста

**Вывод:**

Для решения поставленной задачи создано два web-приложения, приложение “сервер” реализующее логику приложения из восьмой лабораторной и выдающее результаты в XML и приложение посредник, преобразующее XML в HTML с помощью XSLT