***Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования***

|  |  |
| --- | --- |
|  | ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Отчет**

**по лабораторной работе № \_\_8\_\_**

**Дисциплина: \_\_Языки Интернет-программирования\_\_\_\_\_**

**Название лабораторной работы:** \_\_\_\_Ruby on Rails\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент гр. \_ИУ6-33Б**\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_Яныгин Д.С.\_\_\_**

(Подпись, дата) (И.О. Фамилия)

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

(Подпись, дата) (И.О. Фамилия)

Москва, 2020

**Задание:**

1. Модифицировать код ЛР 8 таким образом, чтобы запросы, которые были ранее выполнены, сохранялись в БД и при следующем запросе не требовали повтора вычислений.

* Сформировать модель в соответствии с потребностями хранения данных. Входные параметры являются ключами, по которым извлекается результат.
* Выполнить создание БД и миграцию соответствующими запросами rake.
* Написать тест на добавление и поиск данных с помощью модели. Проверить выполнение теста.
* Модифицировать код приложения таким образом, чтобы результат вычислений преобразовывался в строковый или бинарный формат (на выбор: json, xml, и пр.). Проверить через отладочную печать в консоль, что преобразование выполняется корректно.
* Вставить код для сохранения данных в БД и запрос на поиск предыдущего результата вычислений.
* Добавить действие в контроллер, позволяющее определить, что хранится в БД через сериализацию в XML.
* Проверить, что при выполнении запроса, данные добавляются в БД.
* При помощи консоли сообщений Webrick определить, производится ли поиск результата предыдущего запроса в БД и не повторяются ли одни и те же вычисления.
* Модифицировать модель таким образом, чтобы добавление записей с одинаковыми параметрами было невозможно.
* Реализовать тест модели, проверяющий невозможность повторного добавления одних и тех же результатов вычислений.
* Реализовать функциональный тест, проверяющий, что результаты вычислений различны при различных входных параметрах.
* Проверить маршруты приложения с помощью rake routes и убрать лишние. Обеспечить доступ при обращении по адресу /.

Написать программу, определяющую количество целых чисел ≤ n (n

вводится с клавиатуры), квадрат которых является палиндромом. На-

пример: 26 2 = 676. Вывести на печать числа, квадраты и их количество.

**Часть 1**

**method\_controller.rb**

# frozen\_string\_literal: true  
  
# controllers  
class MethodController < ApplicationController  
 def input; end  
  
 def main\_logic(number)  
 # stuck on 98798674657  
 calc = Enumerator.new do |check\_func|  
 index = *0*  
loop do  
 number = number + reverse\_number(number)  
 check\_func.yield number  
 index += *1*  
if (index > *10000*)   
 return "Execution time expired" + " (> 10 000 iterations)"  
 end  
 end  
 end  
 result = calc.take\_while { |calc\_value, index| (not palindrome?(calc\_value)) }  
 result.push(number)  
 end  
  
 def view  
 number = params[*:number*]  
  
 #---------------validation-----------------#  
 if (number.to\_i == *0*) and (number != "0")   
 @result = "Please, input valid number"  
 else  
 #---------------end-validation-----------------#  
 if (res = Result.find\_by\_value(number.to\_i))  
 @result = ActiveSupport::JSON.decode(res.result)  
 else  
 @result = []  
 @intres = []  
 @ans = *0*  
 n = number.to\_i  
 for i in *1*..n  
 s = (i\*\**2*).to\_s  
 if s.reverse == s  
 @ans += *1*  
@result.append(s)  
 @intres.append(i.to\_s)  
 end  
 end  
 res = Result.create *value:* number.to\_i, *result:* ActiveSupport::JSON.encode(@result)  
 res.save  
 end   
 end  
  
 respond\_to do |format|  
 format.html   
 format.json do  
 render *json:*  
{ *type\_result:* @result.class.to\_s, *value\_result:* @result }  
 end  
 end   
 end  
  
 def results  
 result = Result.all.map { |el| { *value:* el.value, *result:* ActiveSupport::JSON.decode(el.result) } }  
  
 # http://127.0.0.1:3000/method/results.xml  
 respond\_to do |format|  
 format.xml { render *xml:* result.to\_xml }  
 end  
 end  
  
end

**models/result.rb**

# frozen\_string\_literal: true  
  
# Model result  
class Result < ApplicationRecord  
 validates *:value*, *presence: true*, *numericality:* { *greater\_than\_or\_equal\_to: 0* }, *uniqueness: true*  
 validates *:answer*, *presence: false*  
 validates *:result*, *presence: true*  
end

**input.html.erb**

<h2>Страница для ввода</h2>  
<%= form\_tag("/method/view", :method => "get") do %>  
 <%= label\_tag("Число:") %>  
 <%= text\_field\_tag(:number, "1000") %> <br/>  
 <br/>  
 <%= submit\_tag("Calc result") %>  
<% end %>  
  
  
<%= form\_tag("/method/results.xml", :method => "get") do %>  
<hr>  
 <%= submit\_tag("return DB contents") %>  
<% end %>

**view.html.erb**

<h3>Страница для вывода результатов подсчета</h3>  
<p id="result">  
 <!-- <%= debug(@result) %> -->  
 <% if @result.is\_a?(String) %>  
 <%= @result%>  
 <%else%>  
 <table>  
 <% @result.each\_with\_index do |elem, index| %>  
 <tr>  
 <td><%= index+1%></td>  
 <td><%= (elem.to\_i\*\*0.5).to\_i %></td>  
 <td><%= elem%></td>  
  
 </tr>  
 <% end %>  
 </table>  
 <% end %>  
</p>

**method\_controller\_test.rb**

# frozen\_string\_literal: true  
  
require 'test\_helper'  
  
class MethodControllerTest < ActionDispatch::IntegrationTest  
 test 'should get input' do  
 get method\_input\_url  
 assert\_response *:success*  
end  
  
 test 'should get view' do  
 get method\_view\_url  
 assert\_response *:success*  
end  
  
 test 'test\_return\_json' do  
 get method\_view\_url, *params:* { *number: 100*, *format:* 'json' }  
 assert\_response *:success*  
 assert\_includes @response.headers['Content-Type'], 'application/json'  
 end  
  
 test 'test\_record\_in\_db' do  
 before = Result.count  
 get method\_view\_url, *params:* { *number: 10000* }  
 after = Result.count  
  
 assert\_equal before + *1*, after  
 end  
  
 test 'test\_different\_response\_json' do  
 get method\_view\_url, *params:* { *number: 1000*, *format:* 'json' }  
 first = JSON.parse @response.body  
  
 get method\_view\_url, *params:* { *number: 10000*, *format:* 'json' }  
 second = JSON.parse @response.body  
  
 refute\_equal first, second  
 end  
  
end

**Результат:**

Рис. 1 Изначальная страница

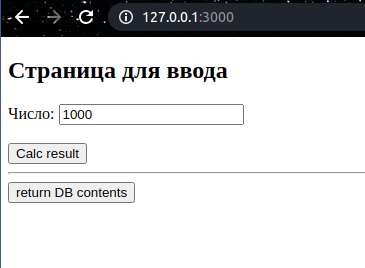


Рис. 2 Вывод

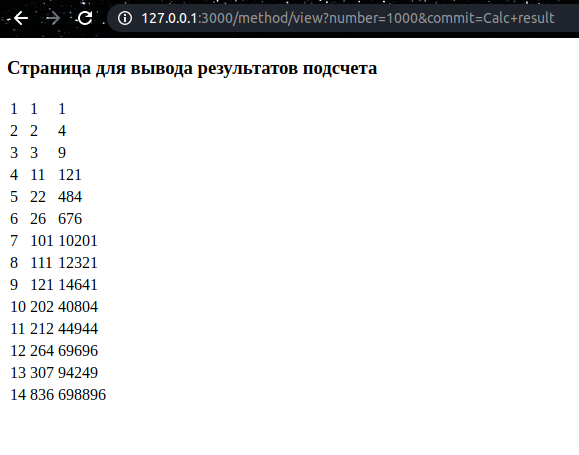
****

Рис. 3 База данных

