Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

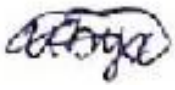
НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 Прикладная информатика**

О Т Ч Е Т

по лабораторной работе № 1 1

Название: Добавление модели. ORM. Разработка БД, подключение, хранение и поиск данных.

Дисциплина: Языки Интернет-программирования

Студент	ИУ6-35Б		10.11.2023	И.А. Буханцев
	(Группа)		(Подпись, дата)	(И.О. Фамилия)
Преподаватель				Е.Ю. Гаврилова
			(Подпись, дата)	(И.О. Фамилия)

Москва, 2023

Задание:

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 Puma/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 /.

Цель: Получение практических навыков в создании веб-приложений, использующих базы данных.

twins_controller.rb

```
# frozen_string_literal: true
require 'json'

class TwinsController < ApplicationController
  def input; end

  def view
    n = params[:n].to_i
    @result = twins(n)
  end

  def is_prime(num)
    return false if num <= 1
    Math.sqrt(num).to_i.downto(2).each { |i| return false if num % i == 0 }
    true
  end

  def twins(n)
    record = Result.find_by_input(n)

    if !record
```

```
    pairs = {}
    (n..2 * n).each do |first|
      (first..2 * n).each do |second|
        if second - first == 2 and is_prime(first) and is_prime(second)
          pairs[first] = second
        end
      end
    end

    # Записывем результат выполнения в БД
    json_res = pairs.to_json

    result = Result.new(:input => n, :result => json_res)
    result.save
    print 'Запись добалена в БД ', result, "\n"
  else
    pairs = JSON.parse(record.result)
    pp 'Результат уже посчитан'
  end

  [pairs, to_table(pairs)]
end

def to_table(pairs = @result, table_class = 'table')
  @table = 'Unknown!'

  unless pairs.empty?
    rows = ''
    pairs.each do |key, value|
      rows += "<tr><td>#{key}</td><td>#{value}</td></tr>"
```

```
    end
    @table = "<table class=\"#{table_class}\"><tbody>#{rows}</tbody></table>"
  end

  @table
 end
end
```

results_controller.rb

```
require 'nokogiri'

class ResultsController < ApplicationController
 before_action :set_result, only: %i[ show edit update destroy ]
 XSLT_TRANSFORM = "#{Rails.root}/public/xslt_transformer.xslt".freeze #
Путь до xslt файла

 # GET /results or /results.json
 def index
  @results = Result.all
 end

 def show_all
  respond_to do |format|
   results = Result.all
   rows = ''

   results.each do |record|
    rows +=
"<cd><id>#{record.input}</id><item>#{record.result}</item></cd>"
   end
```

```
    responce = "<catalog>#{rows}</catalog>"

    format.xml { render xml: xslt_transform(responce).to_xml }
  end
end

def xslt_transform(data, transform: XSLT_TRANSFORM)
  # Функция преобразования
  print data, transform, "\n"

  doc = Nokogiri::XML(data)
  xslt = Nokogiri::XSLT(File.read(transform))
  xslt.transform(doc)
end

# GET /results/1 or /results/1.json
def show
end

# GET /results/new
def new
  @result = Result.new
end

# GET /results/1/edit
def edit
end

# POST /results or /results.json
```

```
  def create
    @result = Result.new(result_params)

    respond_to do |format|
      if @result.save
        format.html { redirect_to result_url(@result), notice: "Result was
successfully created." }
        format.json { render :show, status: :created, location: @result }
      else
        format.html { render :new, status: :unprocessable_entity }
        format.json { render json: @result.errors, status: :unprocessable_entity }
      end
    end
  end

  # PATCH/PUT /results/1 or /results/1.json
  def update
    respond_to do |format|
      if @result.update(result_params)
        format.html { redirect_to result_url(@result), notice: "Result was
successfully updated." }
        format.json { render :show, status: :ok, location: @result }
      else
        format.html { render :edit, status: :unprocessable_entity }
        format.json { render json: @result.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /results/1 or /results/1.json
```

```
  def destroy
    @result.destroy

    respond_to do |format|
      format.html { redirect_to results_url, notice: "Result was successfully
destroyed." }
      format.json { head :no_content }
    end
  end

  private

  # Use callbacks to share common setup or constraints between actions.
  def set_result
    @result = Result.find(params[:id])
  end

  # Only allow a list of trusted parameters through.
  def result_params
    params.require(:result).permit(:input, :result)
  end
end
```

routes.rb

```
Rails.application.routes.draw do
  resources :results

  get 'db_index', to: 'results#index'
  get 'db_new', to: 'results#new'
  get 'show_all', to: 'results#show_all'
```

```
  get 'twins/input'
  get 'twins/view'
  root 'twins#input'
end
```

result_test.rb

```
require "test_helper"

class ResultTest < ActiveSupport::TestCase
  def add_record(n = 10, data = { "11": 13, "17": 19 })
    record = Result.new(:input => n, :result => data)
    record.save
    record
  end

  test "Add data" do
    res = { "11": 13, "17": 19 }
    record = add_record(10, res)
    assert record
  end

  test "Find data" do
    add_record
    record = Result.find_by_input(10)

    assert record
  end

  test "Add same result" do
```

```
    add_record
    assert_raises(ActiveRecord::RecordNotUnique) do
      add_record
    end
  end
end
```

Распечатка данных БД таблицы result:

results id integer PRIMARY KEY PRIMARY KEY AUTOINCREMENT NOT NULL NOT NULL input integer result json created_at datetime NOT NULL NOT NULL updated_at datetime NOT NULL NOT NULL 1 10 "{\"11\":13,\"17\":19}" 2023-11-08 16:33:04.789709 2023-11-08 16:33:04.789709 2 11 "{\"11\":13,\"17\":19}" 2023-11-08 16:33:07.187956 2023-11-08 16:33:07.187956 3 15 "{\"17\":19}" 2023-11-08 16:33:09.666411 2023-11-08 16:33:09.666411

Вставка данных в таблицу:

INSERT INTO "results" ("input", "result", "created_at", "updated_at") VALUES (?, ?, ?, ?) [["input", 15], ["result", "\"{\\\"17\\\":19}\""], ["created_at", "2023-11-08 16:33:09.666411"], ["updated_at", "2023-11-08 16:33:09.666411"]]

Вывод данных из таблицы:

SELECT "results".* FROM "results" WHERE "results"."input" = ? LIMIT ? [["input", 15], ["LIMIT", 1]]

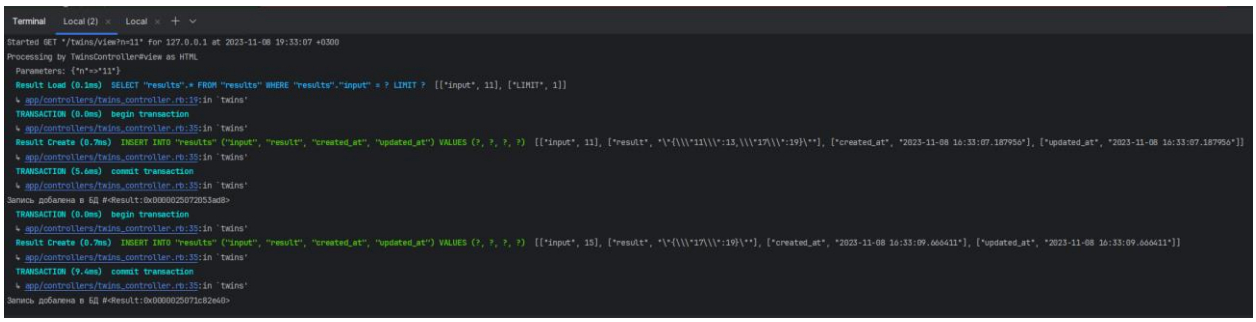


Рисунок 1 – Вывод консоли при работе с БД

Рисунок 2 – Ввод данных в поле

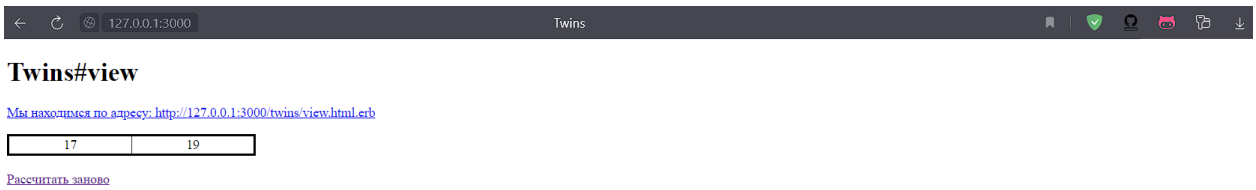


Рисунок 3 – Вывод результата

Вывод: Получили практические навыки в создании веб-приложений, использующих базы данных