|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | **11** |

**Название:**

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

**Дисциплина:** Языки Интернет-программирования



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-35Б |  | 19.12.2023 | В. И. Мамыкин |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Е. Ю. Гаврилова |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

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

**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 /.

**sequences\_controller.rb**

# frozen\_string\_literal: true

require 'json'

class SequencesController < ApplicationController

def input; end

def view

longest\_subsequence = [] # Самая длинная подпоследовательность

current\_subsequence = [] # Текущая подпоследовательность

all\_subsequences = [] # Все подпоследовательности

# unless params[:v2].nil?

sequence = params[:n]&.split(' ')&.map(&:to\_i)

record = Result.find\_by\_input(sequence&.join(' '))

#puts record

if !record

rec\_exists = 0

sequence&.each do |number|

if (Math.sqrt(number) % 1).zero?

# Если число является полным квадратом

current\_subsequence << number

else

# Если число не является полным квадратом

longest\_subsequence = current\_subsequence.clone if current\_subsequence.length > longest\_subsequence.length

cur = current\_subsequence.clone

all\_subsequences << cur.join(' ')

all\_subsequences.pop if all\_subsequences[all\_subsequences.size - 1] == ''

current\_subsequence = []

end

end

cur = current\_subsequence.clone

all\_subsequences << cur.join(' ') if cur.length.positive?

longest\_subsequence = current\_subsequence.clone if current\_subsequence.length > longest\_subsequence.length

all\_subsequences.pop if all\_subsequences[all\_subsequences.size - 1] == ''

subsequence\_count = longest\_subsequence.length

@result = [all\_subsequences, longest\_subsequence.join(' '), subsequence\_count.to\_s, sequence&.join(' ')]

json\_res = @result.to\_json

result = Result.new(:input => sequence&.join(' '), :result => json\_res)

result.save

print 'Запись добалена в БД ', result, "\n"

else

rec\_exists = 1

pp 'Record result:', record.result

@result = JSON.parse(record.result)

pp 'Результат уже посчитан'

end

@table = '<table>' # Начало таблицы

if @result[2] != '0'

@table +=

"<tr><td>Введенная последовательность:</td></tr><tr><td>#{@result[3]}</td></tr><tr><td>Подпоследовательности:</td></tr>"

@result[0].each do |res|

@table += "<tr><td>#{res}</td></tr>"

end

@table +=

"<tr><td>Самая длинная подпоследовательность:</td></tr><tr><td>#{@result[1]}</td></tr><tr><td>Ее длина:</td></tr><tr><td>#{@result[2]}</td></tr>"

else

@table +=

"<tr><td>Введенная последовательность:</td></tr><tr><td>#{@result[3]}</td></tr><tr><td>Подполседовательностей квадратов натуральных чисел нет</td></tr>"

end

@table += '</table>' # Конец таблицы

@tmp = @result.clone

@result = [@tmp, @table, rec\_exists]

end

end

**results\_controller.rb**

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

results = Result.all

rows = ''

result\_saved = true

results.each do |record|

if record.save

rows += "<cd><id>#{record.input}</id><item>#{record.result}</item></cd>"

else

result\_saved = false

break

end

end

if result\_saved

response = "<catalog>#{rows}</catalog>"

render xml: response

else

respond\_to do |format|

format.xml { render 'new', status: :unprocessable\_entity }

end

end

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 'sequences/input'

get 'sequences/view'

root 'sequences#input'

end

**result\_controller\_test.rb**

require 'test\_helper'

class ResultsControllerTest < ActionDispatch::IntegrationTest

setup do

@result = results(:one)

end

test 'should get show\_all' do

get results\_url

assert\_response :success

end

test 'should get new' do

get new\_result\_url

assert\_response :success

end

test 'should create result' do

assert\_difference('Result.count') do

post results\_url, params: { result: { input: @result.input, result: @result.result } }

end

assert\_redirected\_to result\_url(Result.last)

end

test 'should show result' do

get result\_url(@result)

assert\_response :success

end

test 'should get edit' do

get edit\_result\_url(@result)

assert\_response :success

end

test 'should update result' do

patch result\_url(@result), params: { result: { input: @result.input, result: @result.result } }

assert\_redirected\_to result\_url(@result)

end

test 'should destroy result' do

assert\_difference('Result.count', -1) do

delete result\_url(@result)

end

assert\_redirected\_to results\_url

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]]

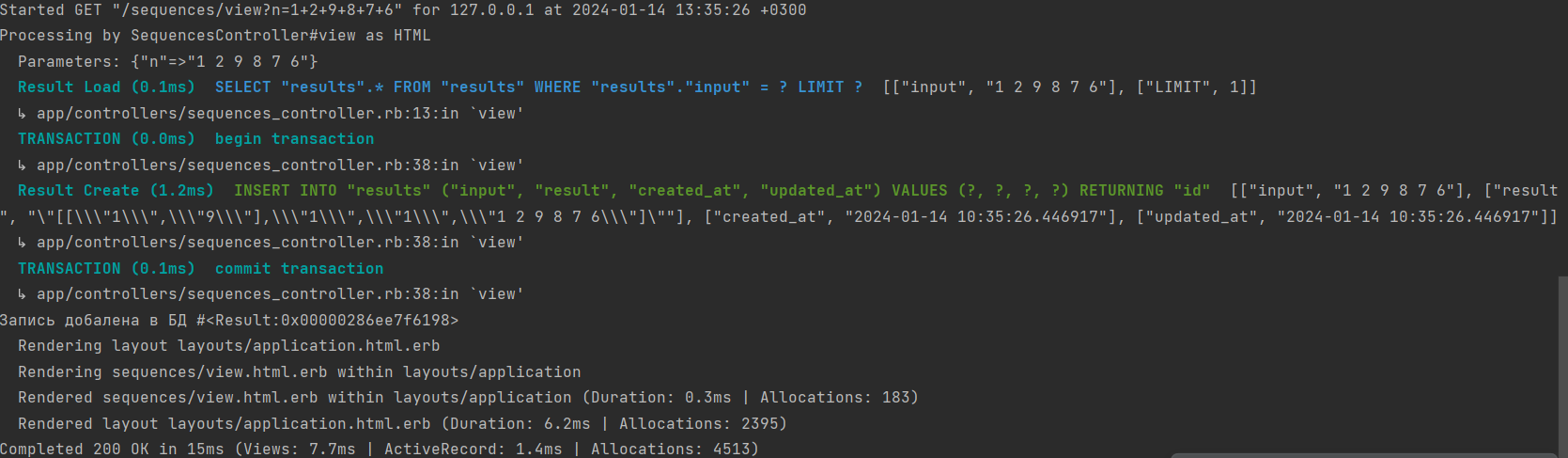


Рисунок 1 – Вывод консоли при работе с БД

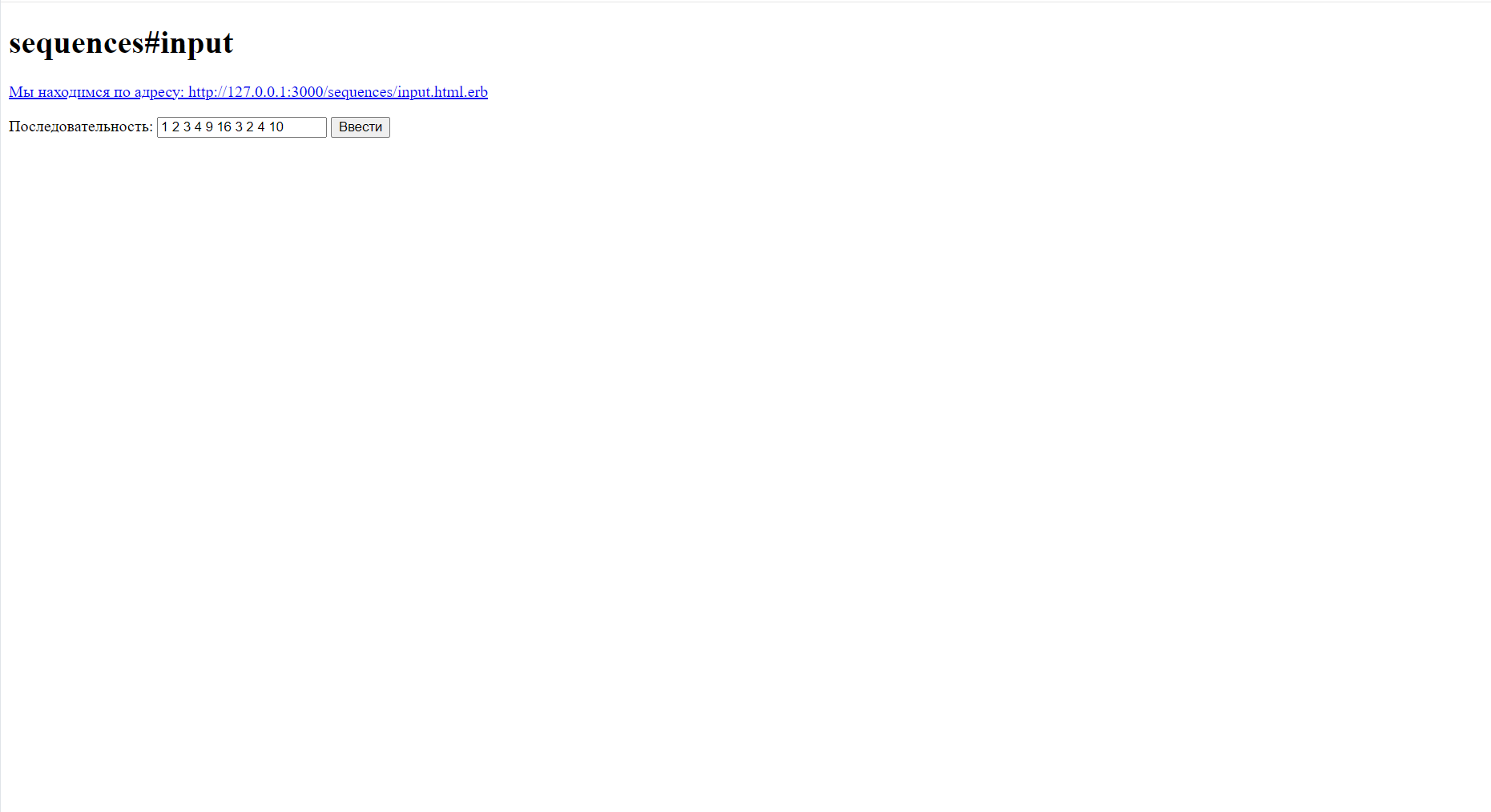
****

Рисунок 2 – Ввод данных в поле

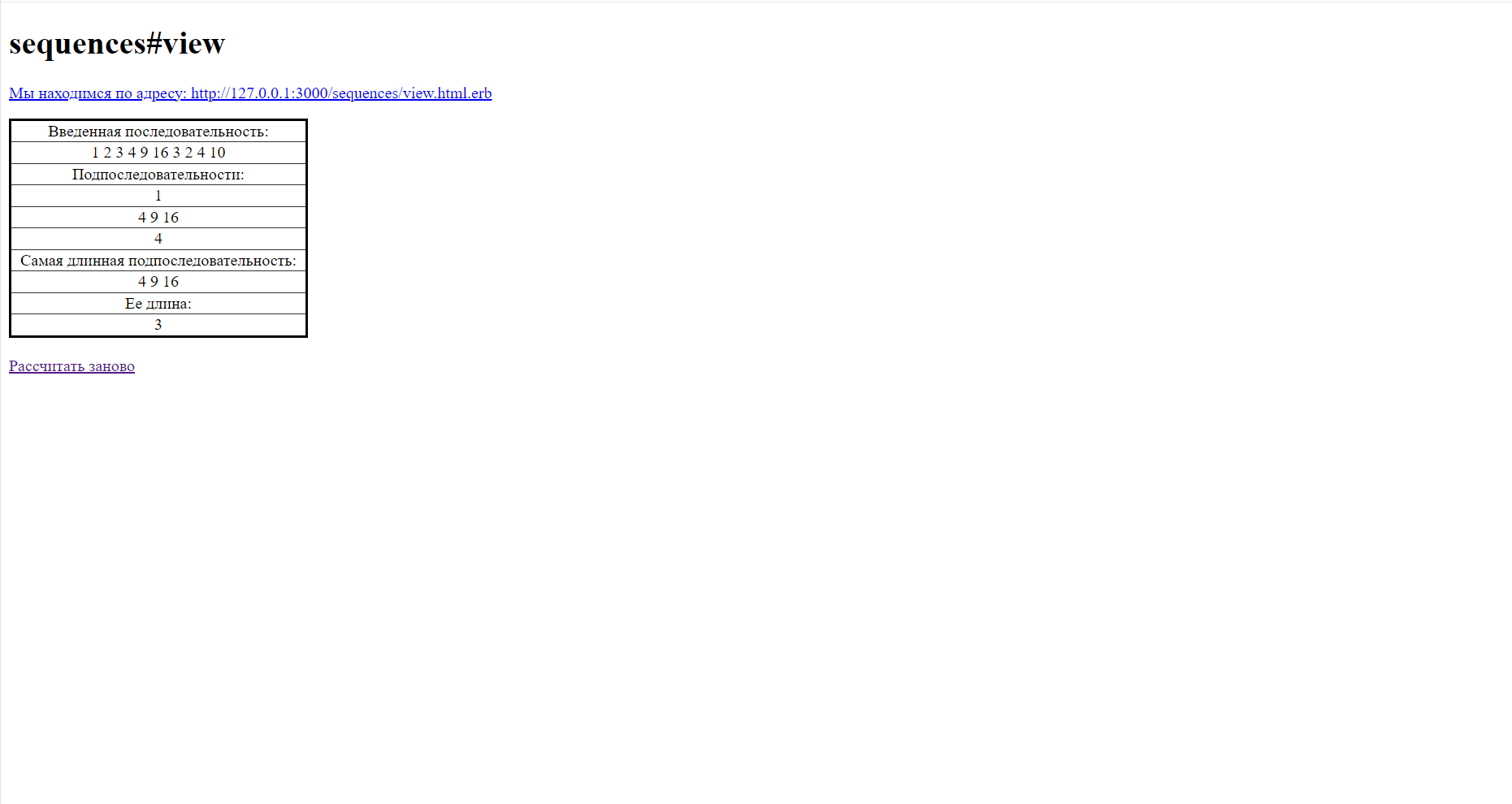
****

Рисунок 3 – Вывод результата

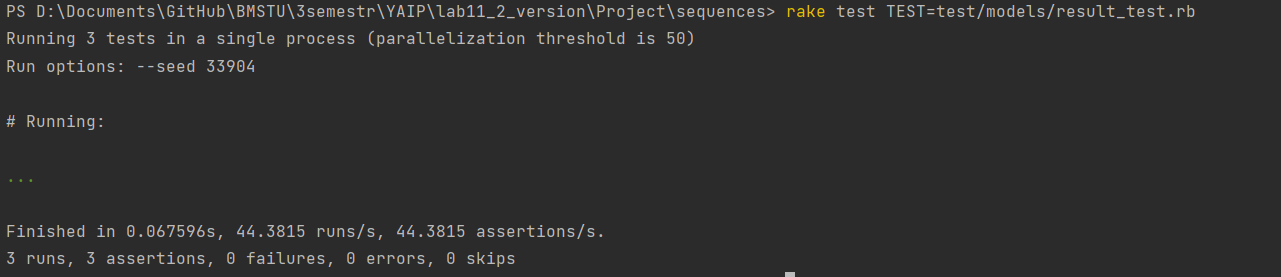
****

Рисунок 4 – Тесты модели

**Вывод:** были получены практических навыков в создании веб-приложений, использующих базы данных.