Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана»

(национальный исследовательский университет)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

О т ч е т

по зачетной работе

Дисциплина: Языки Интернет-программирования.

Тема: API сервис для поиска студентов МГТУ

Студент гр. ИУ6-33Б _____ **Дасов Т.Д.**

(Подпись, дата) (И.О. Фамилия)

Преподаватель _____ _____

(Подпись, дата) (И.О. Фамилия)

Москва, 2019

Введение

Основная задача зачётной работы - продемонстрировать полученные знания в создании собственного веб-приложения.

Обязательные требования к программе:

- *Для реализации использовать Ruby on Rails.*
- *Необходимо иметь контроллеры, обеспечивающие обработку запросов.*
- *Необходимо использовать модели для хранения данных в БД.*
- *Необходимо обеспечить аутентификацию пользователей.*
- *При реализации клиентской части необходимо применить код на языке Javascript и таблицы стилей CSS.*
- *Провести интернационализацию приложения и обеспечить вывод надписей на русском языке (см. пример в лекции 11).*

Цель: *разработать API сервис используя фреймворк RubyOnRails благодаря которому зарегистрированный пользователь сможет получать имеющиеся в БД данные студента. Для реализации бизнес логики использовать библиотеки dry-systems. Добавить поддержку аутентификации с помощью Json Web Tokens (JWT).*

Основная часть

Для адаптивной верстки графической составляющей приложения было принято решение использовать базирующийся на «Bootstrap» CSS фреймворк «Bulma». В качестве языка разметки использовался язык Slim.

Представления

Листинг шаблона страницы application.html.slim:

```
doctype html
html
  head
    meta content=("text/html; charset=UTF-8") http-equiv="Content-Type" /
    title BaumanFinder
    = csrf_meta_tags
    = csp_meta_tag
    = stylesheet_link_tag 'application', media: 'all', 'data-turbolinks-track': false
    = javascript_pack_tag 'application', 'data-turbolinks-track': 'reload'
    link href="https://fonts.googleapis.com/css?
family=IBM+Plex+Mono&display=swap" rel="stylesheet" /
  body
    nav.navbar.header.is-spaced aria-label=("main navigation") role="navigation"
      .container
        .navbar-brand
          a.navbar-item.logo href="/"
            p BaumanFinder
          a.navbar-burger.burger aria-expanded="false" aria-label="menu" data-
target="navbarBasicExample" role="button"
            span aria-hidden="true"
            span aria-hidden="true"
            span aria-hidden="true"
        #navbarBasicExample.navbar-menu
          .navbar-start
            .navbar-item.has-dropdown.is-hoverable
              a.navbar-link
                = t 'home_page.doc'
              .navbar-dropdown
                a.navbar-item
                  = t :doc
                a.navbar-item
                  | JWT запросы
                a.navbar-item
                  | Запросы с использованием токена
                hr.navbar-divider/
                a.navbar-item
```

```
            | Простейшие запросы
          a.navbar-item
            | Поиск по группам
          a.navbar-item
            | Сортировки
      .navbar-end
        .navbar-item
          .buttons
      .navbar-item
        button.navbar-item.button.is-light.is-link.is-rounded
          = link_to 'RU', region: :ru
        button.navbar-item.button.is-light.is-danger.is-rounded
          = link_to 'EN', region: :en
  = yield
  footer.footer
    .content.has-text-centered
      p
        strong= t 'home_page.footer.work_info'
```

Листинг home.html.slim:

```
nav.level.after-header
  .level-item.has-text-centered
    div
      p.heading= t 'home_page.after_header.students_in'
      p.title= users_in_db
  .level-item.has-text-centered
    div
      p.heading= t 'home_page.after_header.updates'
      p.title= t 'home_page.after_header.updates_txt'
  .level-item.has-text-centered
    div
      p.heading= t 'home_page.after_header.users'
      p.title &gt; 0
#wrapper
  .animated-background
    section.section
      .container.has-text-centered.main-page-content
        p.main-page-title= t 'home_page.body.main_txt'
        p.main-page-subtitle= t 'home_page.body.after_main_txt'
```

JS сценарии и CSS стили приложения

Листинг application.scss:

```
@import 'bulma';
* {
   padding: 0;
   margin: 0;
   font-family: 'IBM Plex Mono', monospace;
}

body {
   display: flex;
   min-height: 100vh;
   flex-direction: column;
}

#wrapper {
   flex: 1;
}

.form_element {
   background-color: blue;
}

.form-line {
   margin-bottom: 15px;
}

.logo {
   font-weight: bolder;
   margin-right: 30px;
}

.navbar {
   padding: {
      top: 15px;
      bottom: 15px;
   }
   border: {
      bottom: 1px solid rgb(235, 235, 235);
   }
   font-size: 20px;
}

.main-page-content {
   .main-page-title {
      font-weight: bolder;
      font-size: 80px;
   }
   .main-page-subtitle {
```

```
      font-weight: bold;
      color: grey;
      font-size: 25px;
   }
}

.after-header {
   background-color: rgb(241, 241, 241);
   padding-top: 15px;
   padding-bottom: 15px;
}
```

Листинг application.js:

```
require('jquery')
require('packs/animation')
```

Листинг animation.js (анимация курсора на главной странице):

```
$(document).ready(function() {
    pageLoaded();
});

function pageLoaded() {
    setInterval(changeCursor, 500);
}

function changeCursor() {
    if ($('.main-page-subtitle').text().slice(-1) == '_') {
        oldText = $('.main-page-subtitle').text();
        $('.main-page-subtitle').text(`${oldText.substring(0,
oldText.length - 1)}`);
    } else {
        oldText = $('.main-page-subtitle').text();
        $('.main-page-subtitle').text(`${oldText}_`);
    }
}
```

Локализация приложения

Для поддержки двух языков использовался гем I18n.

Листинг application_controller.rb:

```
class ApplicationController < ActionController::Base
  layout 'application'
  protect_from_forgery with: :null_session
  before_action :language_set
  helper_method :users_in_db

  def language_set
    I18n.locale = params[:region] || I18n.default_locale
  end

  def default_url_options
    { region: I18n.locale }
  end

  def users_in_db
    Student.count
  end
end
```

Листинг page_controller.rb:

```
# frozen_string_literal: true

# Page controller class
class PageController < ApplicationController
  def home; end

  def doc; end
end
```

Листинг (en.yaml, ru.yaml):

```
ru:
  home_page:
    doc: 'Документация'
    about_api: 'Об API'
    login: 'Войти'
    logout: 'Выйти'
    signin: 'Зарегистрироваться'
    after_header:
      students_in: 'Студентов в базе данных'
      updates: 'Данные обновляются каждые'
      updates_txt: '15 мин'
      users: 'людей использующих данное API'
    body:
      main_txt: 'Найдутся все!'
      after_main_txt: 'И даже отчисленные.'
    footer:
      work_info: 'Зачетная работа'
en:
  home_page:
    doc: 'Documentation'
    about_api: 'About API'
    login: 'Login'
    logout: 'Logout'
    signin: 'Signin'
    after_header:
      students_in: 'Students in the database'
      updates: 'Data is updated every'
      updates_txt: '15 min'
      users: 'People using this API'
    body:
      main_txt: 'There are all!'
      after_main_txt: 'And even expelled.'
    footer:
      work_info: 'Zachyotnaya rabota'
```

Бизнес логика приложения

Основой всего API сервиса являются два парсера позволяющие извлекать данные из приказов о зачислении, а также регулярно получать успеваемость студентов с сайта webvpn.bmstu.ru. Поскольку данные операции достаточно ресурсоемкие, то они могут приостанавливать работу всего веб-приложения во время своей работы, поэтому, для корректной работы всего приложения вся бизнес логика связанная с парсерами должна выполняться в отдельном потоке. Для этого были использованы планировщик задач «Sidekiq», а также NoSQL база данных «Redis». Поскольку работа парсеров должна выполняться регулярно с заданным интервалом, был использован гем «Sidekiq-Cron» позволяющий создавать расписание для вызова workeров sidekiqa.

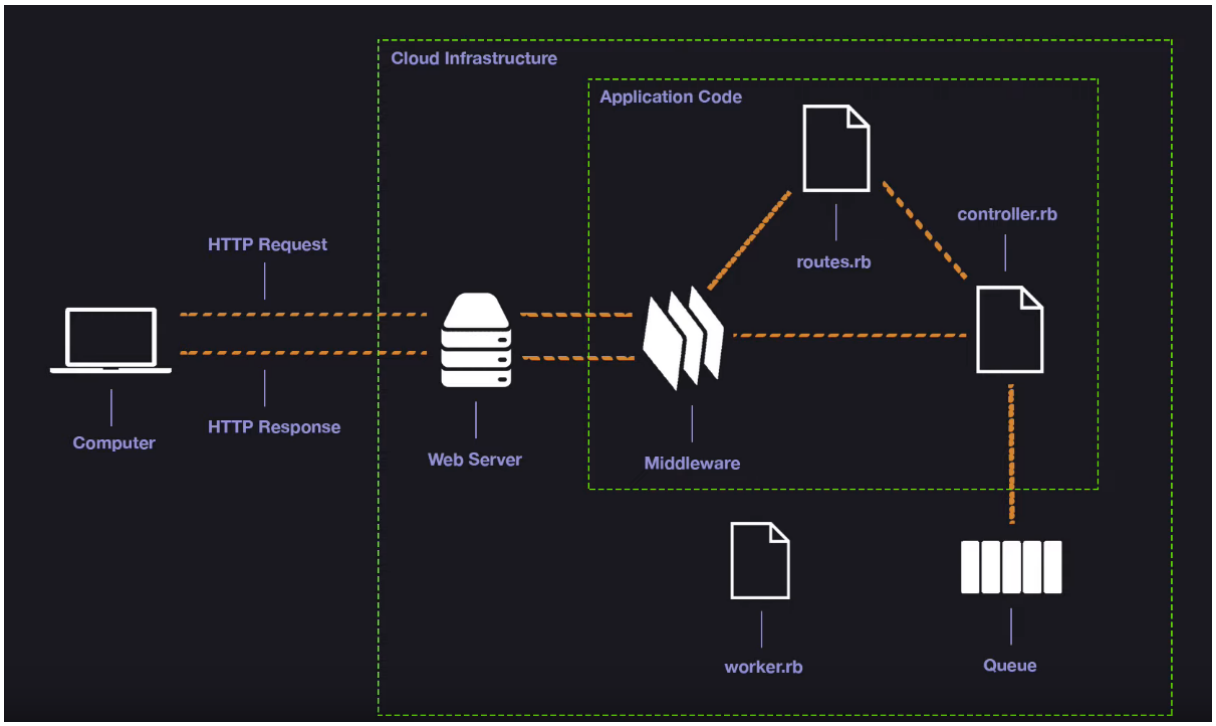

Рис. 1 (Приложение использующее sidekiq и redis)

Листинг sidekiq.rb (инициализации для Sidekiq):

```
Sidekiq.configure_server do |config|
  config.redis = { url: 'redis://localhost:3003/0' }
  schedule_file = "config/schedule.yaml"
  if File.exist?(schedule_file) && Sidekiq.server?
    Sidekiq::Cron::Job.load_from_hash YAML.load_file(schedule_file)
  end
end

Sidekiq.configure_client do |config|
  config.redis = { url: 'redis://localhost:3003/0' }
end
```

Расписание вызовов

```
parse_journal_job:
  cron: "*/15 * * * *"
  class: "ParseJournalWorker"
```

Листинг parse_journal_worker.rb

```
class ParseJournalWorker
  include Sidekiq::Worker

  def perform(*args)
    Parsers::ParserManager.call.update_webvpn_data
  end
end
```

Для реализации корректной работы всей бизнес логики в целом использовался паттерн «команда». Каждый класс должен выполнять определенную последовательность действий и вызываться через метод класса «call». Для этого все сервисные классы были унаследованы от класса service.

Листинг service.rb:

```
# frozen_string_literal: true
# Parent class for service objects
class Service
  include Dry::Monads[:maybe, :result, :do, :try]
  class << self
    def call(*data, &block)
      new.call(*data, &block)
    end
  end
end
```

Для поддержки модульности, а также для того чтобы упростить поддержку кода в будущем все зависимости использующиеся в каждом классе были вынесены в класс «контейнер» для дальнейшего подключения в каждый сервисный класс.

Листинг container.rb

```
# frozen_string_literal: true

require 'yaml'
require 'pdf-reader'
require 'selenium-webdriver'
require 'nokogiri'
require 'mechanize'
require 'jwt'

# Main container with all dependencies
class Container
  extend Dry::Container::Mixin

  namespace 'parsers' do
    namespace 'doc_parser' do
      register 'decree_parser' do
        Parsers::DocParser::DecreeParser
      end
    end

    namespace 'web_parser' do
      register 'web_vpn_parser' do
        Parsers::WebParser::WebVpnParser
      end
    end

    register 'data_validator' do
      Parsers::DataValidator
    end

    register 'scraper' do
      Parsers::WebParser::WebScraping
    end
  end

  namespace 'models' do
    register 'student' do
      Student
    end

    register 'company' do
      Company
    end
```

```
    register 'group' do
      Group
    end

    register 'form_of_study' do
      FormOfStudy
    end

    register 'black_list' do
      BlackList
    end
  end

  namespace 'services' do
    register 'key_keeper' do
      KeyKeeper
    end
    register 'yaml_parser' do
      YAML
    end
    register 'pdf_reader' do
      PDF::Reader
    end
    register 'selenium' do
      Selenium::WebDriver
    end
    register 'scraping_api' do
      Mechanize
    end
    register 'jwt' do
      JWT
    end
    register 'request_handler' do
      RequestHandlers::RequestHandler
    end
    register 'jwt_manager' do
      Other::JwtDecoder
    end
  end
end
```

Пример добавления зависимости:

```
class WebScraping < Service
    include Dry::AutoInject(Container)[
      scraping: 'services.scraping_api',
      key_keeper: 'services.key_keeper'
    ]
end
```

Все необходимые ключи и ссылки также были вынесены в отдельный yaml файл, доступ к которому осуществляется через класс KeyKeeper.

Листинг key_keeper.yaml:

```
decree_parser:
  doc_parser_config: '/config/doc_parser_config.yaml'
  decrees_docs_path: '/storage/data/decrees'
web:
  groups:
    'ИУ6-31Б': 'group/8e243850-3d75-11e8-9f9b-005056960017/'
    'ИУ6-32Б': 'group/8e282802-3d75-11e8-8869-005056960017/'
    'ИУ6-33Б': 'group/8e2955e2-3d75-11e8-9b85-005056960017/'
    'ИУ6-34Б': 'group/8e2c2f2e-3d75-11e8-a507-005056960017/'
    'ИУ6-35Б': 'group/8e2d07c8-3d75-11e8-94be-005056960017/'
  login1_u: 'https://webvpn.bmstu.ru/+CSCOE+/logon.html'
  login1_1_u:
'https://webvpn.bmstu.ru/+CSCO+1075676763663A2F2F636265676E79332E726
82E6F7A6667682E6568++/portal3/login1?back=https://eu.bmstu.ru/'
  login2_x: '/html/body/div[1]/div/a[1]'
  form1:
    usr: 'username'
    pass: 'password_input'
  form2:
    usr: 'username'
    pass: 'password'
  home_u:
'https://webvpn.bmstu.ru/+CSCO+1h75676763663A2F2F72682E6F7A6667682E6
568++/modules/progress3/'
  subj_x: '/html/body/div[1]/div[7]/div/div[3]/div/div[2]/div/table/thead/tr/th'
  stud_c: 'table.standart_table:nth-child(1) > tbody:nth-child(3) > tr'
  name: 'td[2]/a/nobr/span[3]'
  stud_id: 'td[3]'
```

Листинг key_keeper.rb:

```
# frozen_string_literal: true

# Class for extracting service data from a key_keeper.yaml file
class KeyKeeper < Service
  include Dry::AutoInject(Container)[
    'services.yaml_parser'
  ]

  KEEPER_PATH = "#{Rails.root}/config/key_keeper.yaml"

  def call
    file = yield parse_keeper_file
```

```
    Success(file)
  end

  def get_key(resource_key)
    parsed_file = yield parse_keeper_file
    result_data = yield find_by_key(parsed_file, resource_key)

    Success(result_data)
  end

  private

  def parse_keeper_file
    Try { yaml_parser.load_file(KEEPER_PATH) }
      .bind { |file| Success(file) }
      .or(Failure(:file_exception))
  end

  def find_by_key(data, resource_key)
    Maybe(data[resource_key]).bind do |value|
      Success(value)
    end.or(Failure(:key_does_not_exists))
  end
end
```

Листинг decree_parser.rb (парсер приказов о зачислении):

```
# frozen_string_literal: true

# Parsers module
module Parsers
  # Decree parser module
  module DocParser
    # Service object for decree pasring
    class DecreeParser < Service
      include Dry::AutoInject(Container)[
        key_keeper: 'services.key_keeper',
        pdf_parser: 'services.pdf_reader',
        yaml: 'services.yaml_parser'
      ]

      attr_accessor :keys

      def call
        yield init_keys
        config_path = self.keys['doc_parser_config']
        decrees_path = self.keys['decrees_docs_path']
        config = yield init_parser("#{Rails.root}#{config_path}")
        stud_resords = yield parse_doc(config, "#{Rails.root}#{decrees_path}")
        Success(stud_resords)
      end
```

```
    def init_keys
      keys = yield key_keeper.call
      self.keys = keys['decree_parser']

      Success()
    end

    def init_parser(config_path)
      Try { yaml.load_file(config_path)['docs'] }
        .bind { |file| Success(file) }
        .or(Failure(:init_parser_fail))
    end

    def parse_doc(docs_config, doc_path)
      data = docs_config.map do |cnf|
        txt = yield parse_pdf("#{doc_path}/#{cnf['year']}/#{cnf['file_name']}")
        students = yield find_by_regex(txt, cnf['regexes'])
        students.map do |stud|
          stud[:year] = cnf['year']
          stud[:form_of_study] = cnf['form_of_study']
          stud
        end
      end
      Success(data.reduce { |a, b| a + b })
    end

    def find_by_regex(data, regex)
      students = regex.map do |reg|
        data.scan(Regexp.new(reg['regex'])).map do |el|
          model = reg['model'].map(&:to_sym)
          return Failure(:regex_error) unless el.size == model.size

          model.zip(el).to_h
        end
      end
      Success(students.reduce { |a, b| a + b })
    end

    def parse_pdf(file_path)
      Try { pdf_parser.new(file_path).pages.map(&:text).join }
        .bind { |data| Success(data) }
        .or(Failure(:file_reading))
    end
  end
 end
end
```

Листинг decree_parser.yaml (конфигурация для decre_parser.rb):

```
docs:
  - year: 2016
    form_of_study: "budget"
    file_name: "03082016.pdf"
    regexes:
      - regex: '§\d+\.\s*([а-яА-Я]+)\s*([а-яА-Я]+)\s*([а-яА-Я]+);.+иента\s*№:\
s*([^;]+);.+ента\s*№:\s*([^;]+);.+ппа:\s*([а-яА-Я]+[^;]+).+лов:\s*(\d+);'
        model:
          - 'last_name'
          - 'first_name'
          - 'mid_name'
          - 'id_abitur'
          - 'id_stud'
          - 'group_adm'
          - 'exam_scores'

  - year: 2016
    form_of_study: "budget"
    file_name: "08082016 бюджет.pdf"
    regexes:
      - regex: '§\d+\.\s*([а-яА-Я]+)\s*([а-яА-Я]+)\s*([а-яА-Я]+);.+иента\s*№:\
s*([^;]+);.+ента\s*№:\s*([^;]+);.+ппа:\s*([а-яА-Я]+[^;]+).+лов:\s*(\d+);'
        model:
          - 'last_name'
          - 'first_name'
          - 'mid_name'
          - 'id_abitur'
          - 'id_stud'
          - 'group_adm'
          - 'exam_scores'
...
```

Листинг web_scraping.rb (парсер webvpn bmstu):

```
# frozen_string_literal: true

# Parsers
module Parsers
  # Web parser scripts
  module WebParser
    # Mechanize parser
    class WebScraping < Service
      include Dry::AutoInject(Container)[
        scraping: 'services.scraping_api',
        key_keeper: 'services.key_keeper'
      ]

      def call
        yield setup
        agn = agent
```

```
  yield login1(agn)
  yield login2(agn)
  data = yield parse_group(agn)

  Success(data)
end

private

attr_accessor :agent, :key

def setup
  self.agent = scraping.new(user_agent_alias: 'Mac Safari 4')
  keys = yield key_keeper.call
  self.key = keys['web']
  return Success(agent) unless agent.nil?

  Failure(:parser_setup_failed)
end

def login1(agn)
  lg1 = agn.get key['login1_u']
  yield form_filler(
    lg1.forms.first,
    get_form_data('form1')
  )
  Success()
end

def login2(agn)
  lg1 = agn.get key['login1_1_u']
  lg2 = agn.get(lg1.at(key['login2_x'])['href'])
  page_r = yield form_filler(
    lg2.forms.first,
    get_form_data('form2')
  )
  page_f = page_r.links[14].click
  return Success(page_f) if page_f.links.last.text.include?('семестр')

  Failure(:error_while_login)
end

def get_form_data(form_id)
  {
    key[form_id]['usr'] => Rails.application.credentials.webvpn_login!,
    key[form_id]['pass'] => Rails.application.credentials.webvpn_pass!
  }
end

def parse_group(agn)
  data = key['groups'].map do |grp|
    page = agn.get "#{key['home_u']}#{grp}"
```

```
        page.css(key['stud_c']).map { |std| yield parse_std(std, grp, page) }
      end
      Success(data.flatten)
    end

    def parse_std(std, group, page)
      ini = std.at(key['name']).content.to_s.scan(/[а-яА-Я-]+/)
      stud_id = std.at(key['stud_id']).content
      scores = (4..33).map do |id|
        rgx = "td[#{id}]/span"
        id < 12 ? std.at("#{rgx}/span").content : std.at(rgx.to_s).content
      end
      subj_info = get_subj(page, key['subj_x']).zip(scores).to_h
      return Failure(:parse_error) if [ini, stud_id, group.first, subj_info].any?(nil)

      Success(generate_record(ini, stud_id, group.first, subj_info))
    end

    def generate_record(name, id, group, subj_info)
      {
        last_name: name.first,
        first_name: name.second,
        mid_name: name.third,
        id_stud: id,
        group: group,
        subject_data: subj_info
      }
    end

    def get_subj(page, xpath)
      (4..33).map do |num|
        path = "#{xpath}[#{num}]"
        name = page.at(path)
        postfix = %w[КМ М СЗ ЛР].detect { |id| name.content.include?(id) }
        "#{name['title']} #{postfix}"
      end
    end

    def form_filler(form, data)
      data.each { |k, v| form.field_with(id: k).value = v }
      res_page = form.submit
      return Success(res_page) unless res_page.nil?

      Failure(:error_while_form_sub)
    end
  end
 end
end
```

Листинг data_validator.rb:

```
# frozen_string_literal: true

# Parsers module
module Parsers
  # Fill tables with data coming from parsers
  class DataValidator < Service
    include Dry::AutoInject(Container)[
      'models.student',
      'models.company',
      'models.group',
      study_f: 'models.form_of_study'
    ]

    def call(data, data_type)
      case data_type
      when :decree_data
        update_groups_and_companies(data)
        update_by_decrees(data)
      when :web_data
        update_webvpn_data(data)
      else
        Failure(:unsupported_data_type)
      end
    end

    private

    def update_by_decrees(data)
      data.each do |stud|
        record = student.new(
          first_name: stud[:first_name],
          last_name: stud[:last_name],
          mid_name: stud[:mid_name],
          id_stud: stud[:id_stud],
          id_abitur: stud[:id_abitur],
          exam_scores: stud[:exam_scores].to_i,
          form_of_study: study_f.find_by(title: stud[:form_of_study]),
          group_adm: group.find_by(name: stud[:group_adm])
        )
        if record.valid?
          record.save
        else
          upd = student.find_by(id_stud: stud[:id_stud])
          upd.update(
            first_name: stud[:first_name],
            last_name: stud[:last_name],
            mid_name: stud[:mid_name],
            exam_scores: stud[:exam_scores].to_i,
            form_of_study: study_f.find_by(title: stud[:form_of_study]),
            group_adm: group.find_by(name: stud[:group_adm])
```

```
        ) unless upd.nil?
      end
    end
  end

  def update_groups_and_companies(data)
    data.map do |stud|
      company.new(company_name: stud[:company]).save
      group.new(name: stud[:group]).save
    end
  end

  def update_webvpn_data(data)
    data.each do |rec|
      record = student.new(
        first_name: rec[:first_name],
        last_name: rec[:last_name],
        mid_name: rec[:mid_name],
        id_stud: rec[:id_stud],
        group: detect_group(rec[:group]),
        subject_data: rec[:subject_data].to_json
      )
      if record.valid?
        record.save
      else
        stud = student.find_by(id_stud: rec[:id_stud])
        student.update(
          stud.id,
          subject_data: rec[:subject_data].to_json,
          group: detect_group(rec[:group])
        ) unless stud.nil?
      end
    end
  end

  def detect_group(name)
    grp = group.find_by(name: name)
    grp = group.new(name: name).save if grp.nil?
    grp
  end
 end
end
```

Листинг parser_manager.rb:

```
# Parsers module
module Parsers
  # Main class for manage all parsers
  class ParserManager < Service
    include Dry::AutoInject(Container)[
      'parsers.doc_parser.decree_parser',
      'parsers.scraper',
      'parsers.data_validator'
    ]

    def call
      self
    end

    def update_decree_data
      decree_data = yield decree_parser.call
      data_validator.call(decree_data, :decree_data)
    end

    def update_webvpn_data
      data = yield scraper.call
      data_validator.call(data, :web_data)
    end
  end
end
```

Листинг request_handler.rb:

```
# frozen_string_literal: true

# Request handler class
module RequestHandlers
  class RequestHandler < Service
    include Dry::Monads[:result, :do, :maybe, :try]

    def call(params)
      yield check_params(params)
      options = yield filter_req(params)
      data = yield find(options)

      Success(data)
    end

    private

    def filter_req(params)
      search_params = JSON.parse(params.to_s.gsub('=>', ':'))
      return Success(search_params) unless search_params.empty?

      Failure(:invalid_request)
```

```
    end

    def find(params)
      res = Student.where(params).to_a
      return Failure(:students_not_found) if res.nil?

      Success(res)
    end

    def check_params(params)
      return Failure(:invalid_params) if params.nil?

      opt = %w[first_name last_name id_stud]
      res = opt.any? { |par| !params[par].nil? }
      return Failure(:invalid_params) unless res

      Success()
    end
  end
end
```

Листинг token_manager.rb:

```
# frozen_string_literal: true

# module for handling requests
module RequestHandlers
  # Token helper module
  module TokenManager
    def in_black_list?(token)
      return true unless find_by(:token => token).nil?
      false
    end

    def check_token(token)
      Other::JwtDecoder.call.decode_key(token).bind do |data|
        exp = data['expires'].to_time
        return true if (exp - Time.now > 0) && !BlackList.in_black_list?(token)
        return false
      end
      false
    end

    def destroy_token(token)
      rec = BlackList.new(token: token)
      if rec.valid?
        rec.save
        return true
      end
      false
    end
```

```
    def generate_token(email)
      data = {
        user_email: email,
        expires: Time.now + 1.hours.to_i
      }
      Other::JwtDecoder.call.encode_key(data).value!
    end

    def find_by_token(token)
      Other::JwtDecoder.call.decode_key(token).bind do |val|
        return User.where(email: val['user_email']).first
      end
    end
  end
end
```

Листинг jwt_decoder.rb:

```
# frozen_string_literal: true

# All service objects
module Other
  # Class for encode/decode jwt key
  class JwtDecoder < Service
    include Dry::Monads[:result, :do]
    include Dry::AutoInject(Container)[
      'services.jwt'
    ]

    SECRET_KEY = Rails.application.secrets.secret_key_base
    ALGORITHM = 'HS256'

    def call
      self
    end

    def encode_key(data)
      return Failure(:arg_isnt_hash) unless data.is_a?(Hash)

      data = jwt.encode data, SECRET_KEY, ALGORITHM
      return Success(data) unless data.nil?

      Failure(:failed_to_create_key)
    end

    def decode_key(token)
      Try { jwt.decode token, SECRET_KEY, ALGORITHM }
        .bind { |data| Success(data.first) }
        .or(Failure(:invalid_token))
    end
  end
end
```

Модели

Листинг user.rb:

```
class User < ApplicationRecord
  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :validatable

  validates :token, :email, uniqueness: true

  def update_token
    BlackList.destroy_token(self.token)
    new_token = BlackList.generate_token(self.email)
    self.update(token: new_token)
  end

  def get_username
    self.email.match(/[^@]+/).to_s
  end

  class << self
    def create_user(params)
      user = User.new(
        email: params['email'],
        password: params['password'],
        password_confirmation: params['pasword_confirmation'])
      user.token = BlackList.generate_token(user.email)
      user.save if user.valid?
      user.valid? ? user : false
    end
  end
end
```

Листинг student.rb:

```
class Student < ApplicationRecord
  belongs_to :company, optional: true
  belongs_to :group, :class_name => 'Group', :foreign_key => 'group_id',
optional: true
  belongs_to :group_adm, :class_name => 'Group', :foreign_key => 'group_id',
optional: true
  belongs_to :form_of_study, optional: true
  validates :first_name, :last_name, :id_stud, presence: true
  validates :id_stud, uniqueness: true
end
```

Листинг group.rb:

```
class Group < ApplicationRecord
  has_many :students
  validates :name, uniqueness: true
end
```

Листинг from_of_study.rb:

```
class FormOfStudy < ApplicationRecord
  has_many :students
end
```

Листинг company.rb:

```
class Company < ApplicationRecord
  has_many :students
  validates :company_name, uniqueness: true
end
```

Листинг black_list.rb:

```
class BlackList < ApplicationRecord
  extend RequestHandlers::TokenManager
  validates :token, uniqueness: true
end
```

Контроллеры

Листинг user_controller.rb:

```
# frozen_string_literal: true

module Api
  # UserController class
  class UserController < ApplicationController
    include Dry::Monads[:maybe]

    def create
      if @user = User.create_user(user_params)
        render :create, content_type: 'application/json'
      else
        render :json => { message: "user params is not valid!" }
      end
    end

    def destroy
      if user = check_admin(request.headers['token']).value_or(nil)
        User.where(user).first.destroy
        render json: { message: "user successfully deleted!" }
      else
        render json: { message: "permission denied!" }
      end
    end

    private
```

```
    def user_params
      params.require('signup').permit('email', 'password', 'password_confirmation')
    end

    def check_admin(token)
      Maybe(BlackList.find_by_token(token)) do |user|
        return user if user.admin
      end
    end
  end
end
```

Листинг auth_controller.rb:

```
# frozen_string_literal: true

module Api
  # Authorization controller
  class AuthController < ApplicationController
    include Dry::Monads[:try, :maybe]
    include Dry::AutoInject(Container)[
      jwt: 'services.jwt_manager'
    ]

    def create
      if data = params['auth']
        usr = User.find_by(email: data['email'])
        if usr&.valid_password?(data['password'])
          usr.update_token
          @user = usr
          render :create, content_type: 'application/json'
        else
          render json: { message: 'unauthorized user!' }
        end
      else
        render json: { message: 'missing arguments!' }
      end
    end

    def signout
      if token = request.headers['token']
        Maybe(BlackList.find_by_token(token)).bind do |user|
          msg = BlackList.destroy_token(token) ? 'token destroyed!' : 'token already
destroyed!'
          render :json => { message: msg }
        end
      else
        render :json => { message: 'invalid params!' }
      end
    end
  end
end
```

Листинг find_controller.rb:

```
# Api module
module Api
  # FindController class
  class FindController < ApplicationController
    include RequestHandlers
    include Dry::AutoInject(Container)[
      'models.black_list',
      'services.request_handler']
    def find
      if black_list.check_token(request.headers['token'])
        @data = request_handler.call(params['search']).value_or([])
        render :find, content_type: 'application/json'
      else
        render :json => { message: 'authentication failed' }
      end
    end
  end
end
```

Листинг create.json.jbuilder (пользователь вошел в систему):

```
json.message "user successfully logged in!"
json.data do
  json.username @user.email.match(/[^@]+/).to_s
  json.token @user.token
end
```

Листинг find.json.jbuilder (студенты найдены):

```
 json.students @data.each do |student|
  subj = JSON.parse(student.subject_data) unless student.subject_data.nil?
  json.first_name student.first_name
  json.second_name student.last_name
  json.mid_name student.mid_name
  json.student_id student.id_stud
  json.group_upon_admission student.group_adm.name unless
student.group_adm.nil?
  json.group student.group.name unless student.group.nil?
  json.exam_scores student.exam_scores
  unless subj.nil?
    json.group_rating subj.values[-3]
    json.flow_rating subj.values[-2]
    json.module_points_sum subj.values[-1]
    json.subject_data do
      json.merge! subj
    end
  end
end
```

Листинг create.json.jbuilder (пользователь зарегистрирован):

```
json.message "user successfully registered!"
json.data do
  json.username @user.email.match(/[^@]+/).to_s
  json.token @user.token
end
```

Пример работы приложения

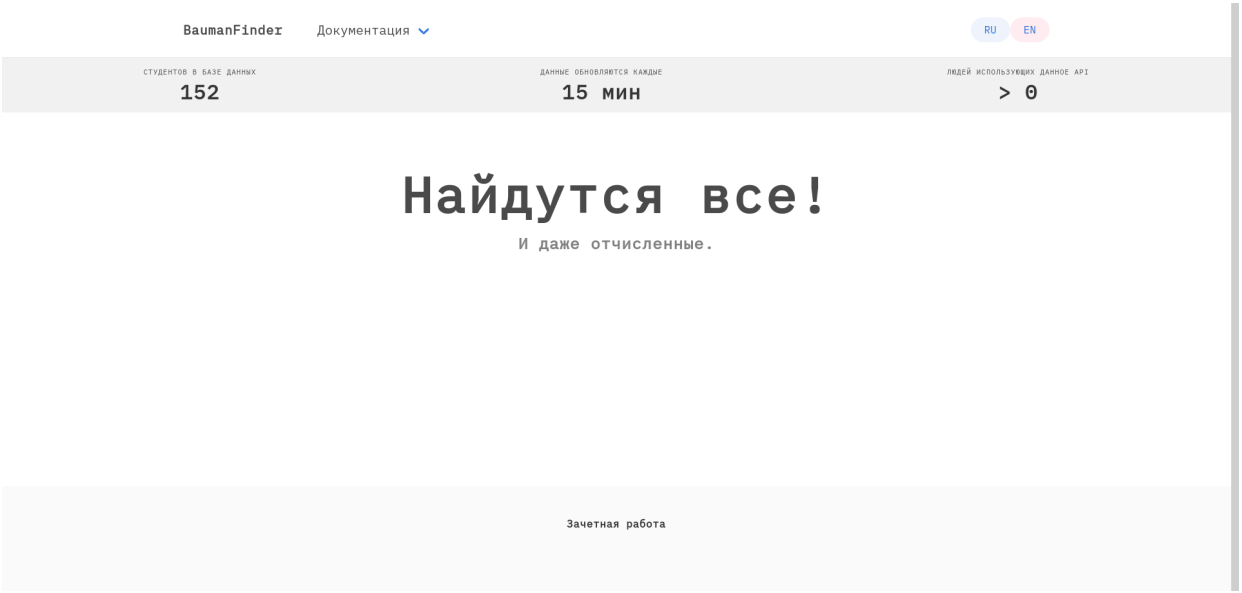

Рис. 2 (Главная страница приложения)

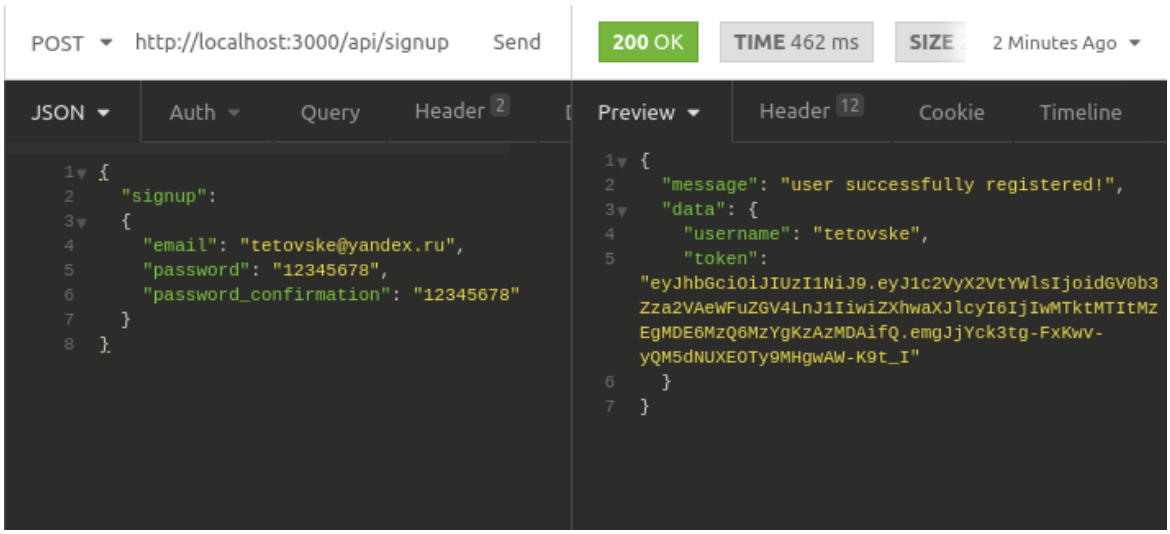

Рис. 3 (Регистрация нового пользователя)

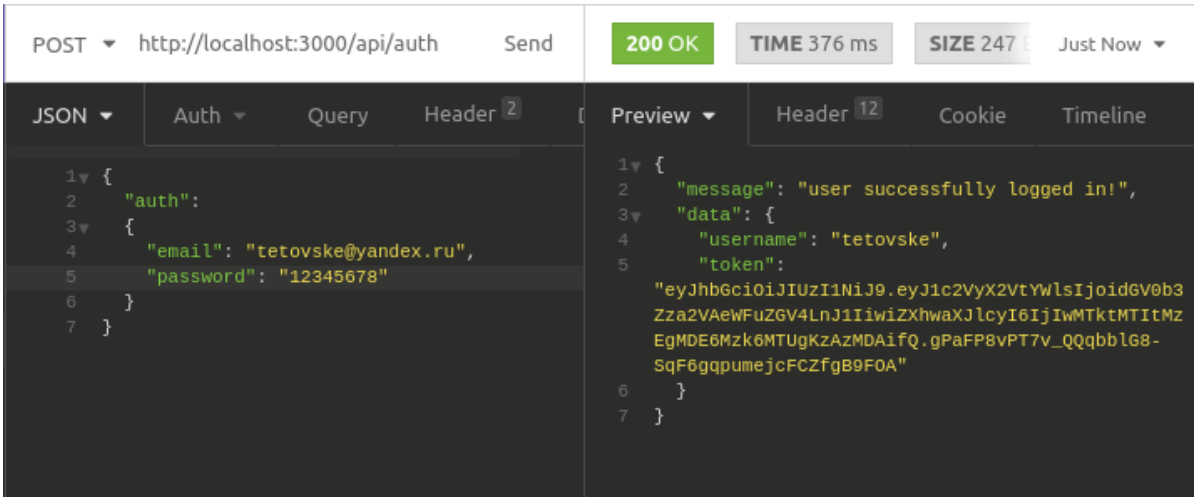

Рис. 4 (Пользователь успешно авторизовался (токен находится в заголовке))

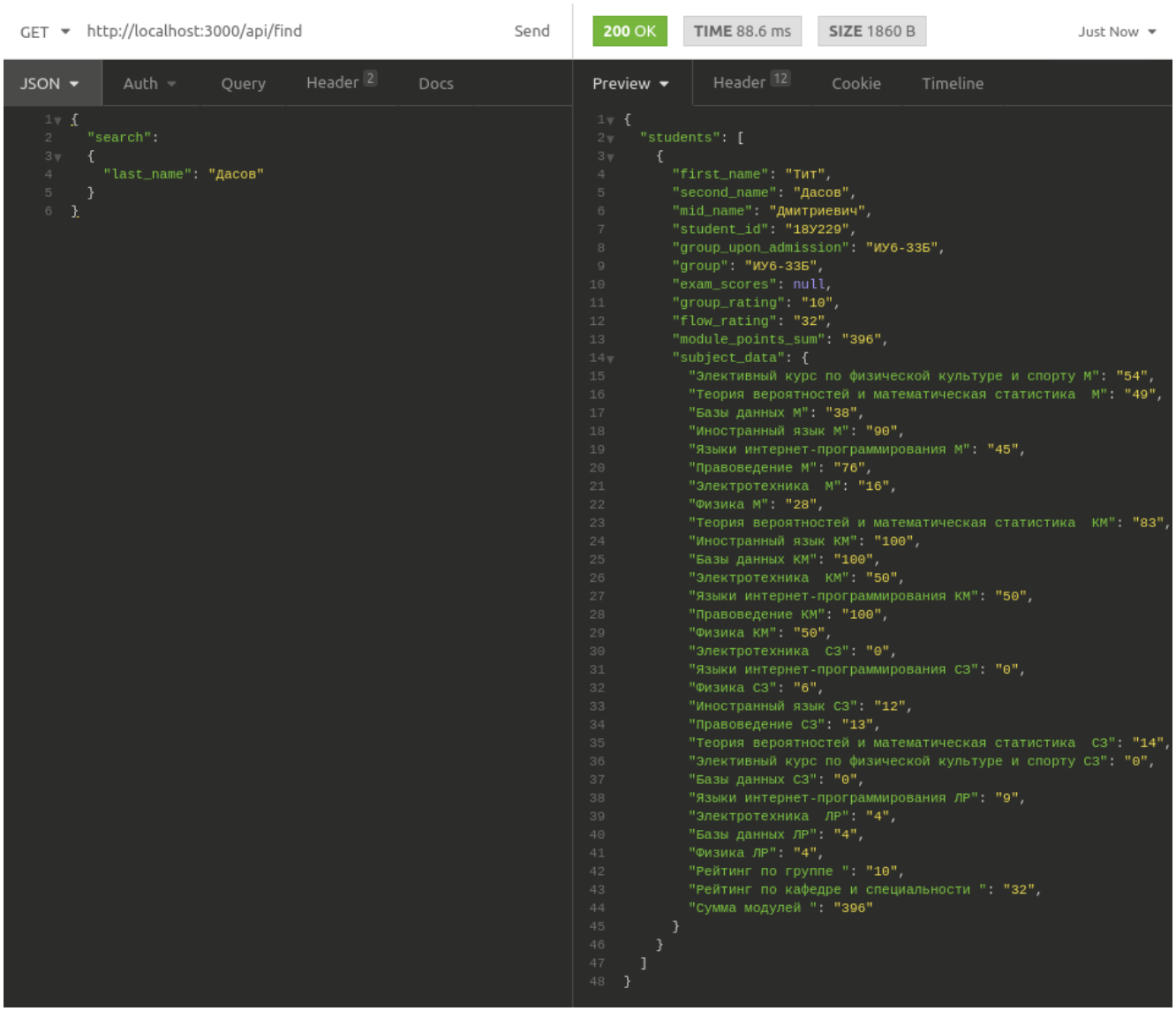

Рис. 5 (Результат поиска)

Выход пользователя из системы происходит путем добавления токена в черный список, после чего данный токен больше не может использоваться. Также следует отметить, что у каждого токена есть срок жизни.

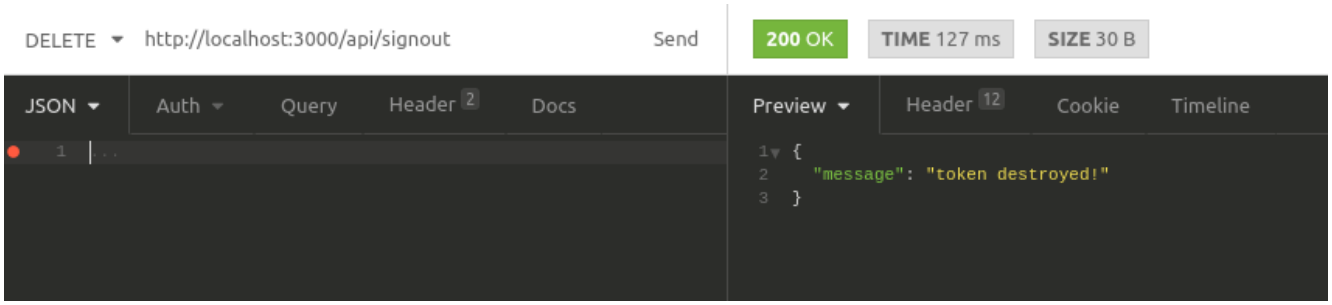

Рис. 6 (Пользователь вышел из системы)

Листинг routes.rb:

```
Rails.application.routes.draw do
  require 'sidekiq/web'
  require 'sidekiq/cron/web'

  mount Sidekiq::Web => '/sidekiq'
  devise_for :users

  scope "(:region)", region: /#{I18n.available_locales.join('|')}/ do
    root 'page#home', :as => 'home'
    get 'test/output', :as => 'output'
    get '/documentation' => 'page#doc', :as => 'doc'
  end

  scope module: 'api', path: 'api' do
    resources :auth, only: [:create]
    resources :user, only: [:destroy]
    delete '/signout' => 'auth#signout'
    post '/signup' => 'user#create'
    get '/find' => 'find#find'
  end
end
```

Тесты приложения

Листинг user_spec.rb (Тест на регистрацию существующих пользователей):

```
require 'rails_helper'

RSpec.describe User, type: :model do
  context 'if user is trying to add existing variables' do
    let(:user_data) do
      {
        :email => 'test@email.com',
        :password => '123456',
        :password_confirmation => '123456',
        :token => '1'
      }
    end

    it 'should return false' do
      fake_user = User.new(user_data)
      fake_user.token = '11'
      fake_user.save if exs = User.find_by(:email => 'test@email.com').nil?
      another_fake = User.new(user_data)
      expect(another_fake.valid?).to be_falsy
      fake_user.destroy if exs.nil?
    end

    it 'should also return false as tokens should not be repeated!' do
      fake_user = User.new(user_data)
      fake_user.email = 'another@email.com'
      fake_user.save if exs = User.find_by(:token => '1').nil?
      another_fake = User.new(user_data)
      expect(another_fake.valid?).to be_falsy
      User.find_by(:email => 'test@email.com').destroy if exs.nil?
    end
  end
end
```

Листинг student_spec.rb (Тест на добавление существующих студентов):

```
require 'rails_helper'
RSpec.describe Student, type: :model do
  context 'if we are trying to add student with exsisting stud_id' do
    let(:stud_data) do
      {
        :first_name => 'Александр',
        :last_name => 'Сидоров',
        :id_stud => 'ИУ6'
      }
```

```
    end

    it 'should return false because stud_id is uniq!' do
      student = Student.new(stud_data)
      student.save if exs = Student.find_by(:id_stud => 'ИУ6').nil?
      another_student = Student.new(stud_data)
      expect(another_student.valid?).to be_falsy
      another_student.destroy if exs.nil?
    end
  end
end
```

Листинг home_page_spec.rb (Тест на работоспособность локализации, а также корректность отображения различных элементов страницы):

```
require 'rails_helper'

RSpec.describe 'home page' do
  describe 'home page content' do
    it 'should check presense of main inscription' do
      visit '/ru'
      expect(page).to have_content('Найдутся все!')
    end

    it 'should check presence of header elements' do
      visit '/ru'
      elems = %w[.container .navbar .header .is-spaced .navbar-brand]
      elems.each { |elem| expect(page).to have_css(elem) }
    end
  end

  describe 'some internationalization tests' do
    it 'should be translated into English' do
      visit '/en'
      sel1 = '.main-page-title'
      sel2 = 'div.level-item:nth-child(2) > div:nth-child(1) > p:nth-child(1)'
      expect(page.find(sel1)).to have_content 'There are all!'
      expect(page.find(sel2)).to have_content 'Data is updated every'
    end

    it 'should be translated into Russian' do
      visit '/ru'
      sel1 = '.main-page-title'
      sel2 = 'div.level-item:nth-child(2) > div:nth-child(1) > p:nth-child(1)'
      expect(page.find(sel1)).to have_content 'Найдутся все!'
      expect(page.find(sel2)).to have_content 'Данные обновляются каждые'
    end
  end
end
```

```
tetovske@pop-os ~/YAIP/bauman_finder_api  jwt ● rspec -f d

home page
  home page content
    should check presense of main inscription
    should check presence of header elements
  some internationalization tests
    should be translated into English
    should be translated into Russian

Student
  if we are trying to add student with exsisting stud_id
    should return false because stud_id is uniq!

User
  if user is trying to add existing variables
    should return false
    should also return false as tokens should not be repeated!

Finished in 0.537 seconds (files took 6.88 seconds to load)
7 examples, 0 failures

tetovske@pop-os ~/YAIP/bauman_finder_api  jwt ●
```

Рис. 7 (Результаты тестов)

Листинг .rubocop.yaml (Файл конфигурации):

```
Metrics/LineLength:
  Max: 120

Metrics/MethodLength:
  Max: 30

Metrics/AbcSize:
  Enabled: false

Lint/AssignmentInCondition:
  Enabled: false
```

```
tetovske@pop-os ~/YAIP/bauman_finder_api/app  jwt ● rubocop controllers models service
s
Inspecting 25 files
.........................

25 files inspected, no offenses detected
tetovske@pop-os ~/YAIP/bauman_finder_api/app  jwt ●
```

Рис. 8 (Отчет Rubocop)

Вывод: *в ходе выполнения зачетной работы было разработан API сервис позволяющий получать различные данные каждого студента. Приложение поддерживает JWT авторизацию, что свойственно всем API сервисам. Для корректной работы парсеров использовались планировщик задач «Sidekiq» и NoSQL база данных «Redis». Приложение было протестировано, а также проверено на соответствие стилю программой Rubocop.*