



**«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

\_\_\_\_\_  
(Подпись, дата)

**Дасов Т.Д.**

(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 паттерны «Стратегия» и «Команда», в частности библиотеки dry-system. Добавить поддержку аутентификации с помощью Json Web Tokens (JWT), devise, а также OmniAuth.

## Основная часть

Для адаптивной верстки графической составляющей приложения было принято решение использовать базирующийся на «Bootstrap» CSS фреймворк «Bulma».

### Представления

#### *Листинг шаблона страницы application.html.erb:*

```
<!DOCTYPE html>
<html>
  <head>
    <title>BaumanFinder</title>
    <%= csrf_meta_tags %>
    <%= csp_meta_tag %>
    <%= stylesheet_link_tag 'application', media: 'all', 'data-turbolinks-track': false %>
    <%= javascript_pack_tag 'application', 'data-turbolinks-track': 'reload' %>
    <%= javascript_include_tag "https://kit.fontawesome.com/802bb0066f.js", 'crossorigin':
'anonymous' %>
    <link href="https://fonts.googleapis.com/css?family=IBM+Plex+Mono&display=swap"
rel="stylesheet">
  </head>
  <body>
    <%= render 'layouts/devise_alert' %>
    <nav class="navbar header is-spaced" aria-label="main navigation" role="navigation">
      <div class="container">
        <div class="navbar-brand">
          <a class="navbar-item logo" href="/#{I18n.locale}">
            <p>BaumanFinder API</p>
          </a>
        </div>
        <div id="navbarBasicExample" class="navbar-menu">
          <div class="navbar-start">
            <a class="navbar-item">
              <%= link_to t("home_page.doc"), doc_path %>
            </a>
            <% if user_signed_in? %>
              <a class="navbar-item">
                <%= link_to t("home_page.account"), account_path %>
              </a>
            <% end %>
            <% if user_signed_in? && current_user.is_admin %>
              <a class="navbar-item">
                <%= link_to t("home_page.admin"), admin_path %>
              </a>
            <% end %>
          </div>
          <div class="navbar-item">
            <div class="buttons">
              <%= button_to "RU", "/ru", :method => :get, class: "navbar-item button is-light
is-info is-rounded" %>
              <%= button_to "EN", "/en", :method => :get, class: "navbar-item button is-light
is-danger is-rounded" %>
            </div>
          </div>
          <div class="navbar-end">
            <% if user_signed_in? %>
```



```

        <%= ::Temple::Utils.escape_html((t 'home_page.after_header.users')) %>
    </p>
    <p class="title">&gt; 0</p>
</div>
</div>
</nav>
<div id="wrapper">
    <div class="animated-background">
        <section class="section">
            <div class="container has-text-centered main-page-content">
                <p class="main-page-title">
                    <%= ::Temple::Utils.escape_html((t 'home_page.body.main_txt')) %>
                </p>
                <p class="main-page-subtitle">
                    <%= ::Temple::Utils.escape_html((t 'home_page.body.after_main_txt')) %>
                </p>
            </div>
        </section>
    </div>
</div>

```

### ***Листинг \_documentation.html.erb (документация API):***

```

<section class="hero is-success is-fullheight">
    <div class="hero-body">
        <div class="container">
            <h1 class="title">
                <%= t "account.tutorial_v1_title" %>
            </h1>
        </div>
    </div>
</section>
<section class="hero is-light is-medium">
    <div class="hero-body">
        <div class="container">
            <div class="title"><%= t "doc.v1.requests" %><code>find</code></div>
            <div class="container">
                <%= t "doc.v1.description" %>: <code>name, last_name, mid_name, group,
year_of_admission, exam_scores</code>
            </div>
            <div class="subtitle"><%= t "doc.v1.example" %></div>
            <div class="container">
                <%= t "doc.v1.req" %>: <code>http://baumanfinder.ru/v1/find?
APIKEY=PjxKcDY17dZQSyuqVp2OHO8zn&name=Тит</code>
            </div>
            <div class="container">
                <%= t "doc.v1.response_format" %>: </br><code>
                { </br>
                  "status": "success", </br>
                  "data": [ </br>
                    { </br>
                      "name": "Тит", </br>
                      "last_name": "Дасов", </br>
                      "mid_name": "Дмитриевич", </br>
                      "id_stud": "18Y229", </br>
                      "id_abitur": "K0008", </br>
                      "form_of_study": null, </br>
                      "exam_scores": "230.0", </br>
                      "admitted_group": "ИУ6-33Б", </br>
                      "group": "ИУ6-33Б", </br>
                      "company": null, </br>
                      "subjects_data": { </br>

```

```

"Элективный курс по физической культуре и спорту М": "54",</br>
"Языки интернет-программирования М": "70",</br>
"Теория вероятностей и математическая статистика М": "49",</br>
"Базы данных М": "71",</br>
"Иностранный язык М": "90",</br>
"Правоведение М": "76",</br>
"Электротехника М": "50",</br>
"Физика М": "28",</br>
"Теория вероятностей и математическая статистика КМ": "83",</br>
"Электротехника КМ": "100",</br>
"Базы данных КМ": "100",</br>
"Языки интернет-программирования КМ": "100",</br>
"Иностранный язык КМ": "100",</br>
"Правоведение КМ": "100",</br>
"Физика КМ": "100",</br>
"Правоведение СЗ": "13",</br>
"Языки интернет-программирования СЗ": "1",</br>
"Элективный курс по физической культуре и спорту СЗ": "0",</br>
"Физика СЗ": "6",</br>
"Иностранный язык СЗ": "12",</br>
"Электротехника СЗ": "6",</br>
"Базы данных СЗ": "0",</br>
"Теория вероятностей и математическая статистика СЗ": "14",</br>
"Электротехника ЛР": "4",</br>
"Базы данных ЛР": "4",</br>
"Языки интернет-программирования ЛР": "12",</br>
"Физика ЛР": "4",</br>
"Рейтинг по группе ": "9",</br>
"Рейтинг по кафедре и специальности ": "29",</br>
"Сумма модулей ": "488"</br>
}</br>
}</br>
]</br>
}</br>
</code>
</div>
</div>
</div>
</section>
<section class="hero is-info is-fullheight">
<div class="hero-body">
<div class="container">
<h1 class="title">
<%= t "doc.v2.main-title" %>
</h1>
</div>
</div>
</section>
<section class="hero is-light is-medium">
<div class="hero-body">
<div class="container">
<div class="container">
<%= t "doc.v2.title" %><code>http://baumanfinder.ru/v2/auth</code>
</div>
<div class="container">
<%= t "doc.v2.after-title" %>
</div>
</div>
</div>
</section>

```

### ***Листинг account.html.erb (настройки аккаунта пользователя):***

```
<section class="hero is-medium is-primary is-bold">
  <div class="hero-body">
    <div class="container">
      <h1 class="title">
        <%= t "account.welcome" %>, <%= account_username %>
      </h1>
      <h2 class="subtitle">
        <%= t "account.welcome_subtitle" %>
      </h2>
    </div>
  </div>
</section>
<section class="hero is-light is-medium">
  <div class="hero-body">
    <div class="container">
      <div class="subtitle"><%= t "account.use_this_token" %>:</div>
      <div class="field has-addons">
        <div class="control" id="token-input">
          <input class="input" type="text" readonly="readonly" value="<%=
user_bf_token %>">
        </div>
        <div class="control">
          <%= link_to t('account.create_new_token'), regenerate_token_path, class:
"button is-info", remote: true %>
        </div>
      </div>
    </div>
  </div>
</section>
<%= render 'page/documentation' %>
```

### ***Листинг admin.html.erb (страница администратора):***

```
<section class="hero is-warning">
  <div class="hero-body">
    <div class="container">
      <h1 class="title">
        <%= t "admin_panel.title" %>
      </h1>
    </div>
  </div>
</section>
<div id="wrapper">
  <section class="section">
    <div class="container" id="user-list">
      <%= render 'admin/user_list' %>
    </div>
  </section>
</div>
```

### ***Листинг \_user\_list.html.erb (представление списка пользователей):***

```
<% user_list.reject { |u| u == current_user }.each do |u| %>
  <article class="media">
    <% unless u.image_url.blank? %>
      <figure class="media-left">
        <p class="image is-64x64">
          <%= image_tag(u.image_url, class: "is-rounded") %>
        </p>
```

```

</figure>
<% end %>
<div class="media-content">
  <div class="content">
    <p>
      <strong><%= u.bf_username %></strong>
      <% if u.is_admin %>
        <span class="icon is-small has-text-success">
          <i class="fas fa-certificate"></i>
        </span>
      <% end %>
      <% case u.provider&.to_sym %>
      <% when :vkontakte %>
        <span class="icon is-small has-text-info">
          <i class="fab fa-vk"></i>
        </span>
      <% when :facebook %>
        <span class="icon is-small has-text-info">
          <i class="fab fa-facebook-square"></i>
        </span>
      <% when :github %>
        <span class="icon is-small has-text-info">
          <i class="fab fa-github"></i>
        </span>
      <% end %>
    </p>
  </div>
  <div class="content">
    <span><%= t "admin_panel.reg_date" %>: <%= u.created_at %></span>
  </div>
</div>
<div class="media-right">
  <div class="control">
    <%= button_to admin_destroy_user_path(u), method: :delete, remote: true, class: "button is-
danger", data: { confirm: "Are you sure to delete this user?" } do %>
      <span class="icon is-small">
        <i class="fas fa-backspace"></i>
      </span>
      <span><%= t "admin_panel.delete_user" %></span>
    <% end %>
  </div>
  <div class="control">
    <% unless u.is_admin %>
      <%= button_to change_permissions_for_path(u), method: :post, remote: true, class:
"button is-success" do %>
        <span class="icon is-small ">
          <i class="fas fa-tools"></i>
        </span>
        <span><%= t "admin_panel.create_admin" %></span>
      <% end %>
    <% else %>
      <%= button_to change_permissions_for_path(u), method: :post, remote: true, class:
"button is-warning" do %>
        <span class="icon is-small ">
          <i class="fas fa-tools"></i>
        </span>
        <span><%= t "admin_panel.destroy_admin" %></span>
      <% end %>
    <% end %>
  </div>
</div>
</article>
<% end %>

```



## **JSON представления**

### **Листинг *handle\_response.json.jbuilder* (отображение ответа API версии V1):**

```
res = @response

json.status res[:status]
json.cause res[:cause] unless res[:status].eq!?(:success)
json.data res[:data].each do |s|
  json.name s.first_name
  json.last_name s.last_name
  json.mid_name s.mid_name
  json.id_stud s.id_stud
  json.id_abitur s.id_abitur
  json.form_of_study s.form_of_study&.title
  json.examen_scores s.examen_scores
  json.admitted_group s.group_adm&.name
  json.group s.group&.name
  json.company s.company&.company_name
  unless s.subject_data.nil?
    json.subjects_data do
      json.merge! JSON.parse(s.subject_data)
    end
  end
end unless res[:data].nil?
```

### **Листинг *create.json.jbuilder* (предоставление пользователю jwt токена):**

```
json.message "user successfully logged in!"
json.data do
  json.token @user.jwt_payload
end
```

## **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: rgb(221, 221, 221);
}
```

```

.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;
}

.alert-abstract {
  text-align: center;
  padding: 10px 0;
  color: white;
}

.notice {
  @extend .alert-abstract;
  background-color: rgb(20, 204, 100);
}

.alert {
  @extend .alert-abstract;
  background-color: #FF4136;
}

.avatar {
  border: 5px solid black;
  border-radius: 50%;
  width: 2px;
}

.reg-form {
  padding: 30px;
}

```

```

width: 300px;
background-color: rgb(221, 221, 221);
border-radius: 30px;
}

input[type="submit"].sign-in-with {
border-radius: 15px;
background-color: cadetblue;
border: 0;
padding: 0;
}

.sign-in-with {
border-radius: 15px;
background-color: cadetblue;
border: 0;
padding: 0;
}

.control {
margin-top: 10px;
}

.login-form {
padding: 50px;
border-radius: 15px;
border: 1px solid rgb(235, 235, 235);
}

```

### ***Листинг application.js:***

```

require("@rails/ujs").start()
require('jquery')
require('packs/animation')
require('packs/notification')

```

### ***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}_`);
  }
}

```

### ***Листинг notification.js (Обработка удаления оповещения):***

```
$(document).ready(function() {  
  pageLoaded();  
});  
  
function pageLoaded() {  
  $('#delete').click(function() {  
    $(this).parent().remove();  
  });  
}
```

Для реализации асинхронной подгрузки данных на страницу использовалась технология AJAX.

### ***Листинг refresh\_user\_list.js.erb:***

```
$(document).ready(function() {  
  restoreUserList();  
});  
  
function restoreUserList() {  
  let dataToReload = `<%= render 'admin/user_list' %>`  
  $('#user-list').empty().append(dataToReload);  
}
```

В данном примере при изменении админом прав доступа для пользователя не будет происходить полная перезагрузка страницы.

### ***Листинг regenerate\_token.js.erb (Добавление нового токена на страницу):***

```
var new_token = `  
  <input class="input is-success" type="text" readonly="readonly" value="<%=  
current_user.bf_api_token %>">  
  <p class="help is-success"><%= t 'account.token_updated' %></p>  
`;  
  
$('#token-input').empty().append(new_token);
```

### **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 (en.yaml, ru.yaml):***

```

ru:
  home_page:
    doc: 'Документация'
    about_api: 'Об API'
    admin: 'Админка'
    login: 'Войти'
    logout: 'Выйти'
    signup: 'Начать использовать API'
    account: 'Аккаунт'
    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'
    login_form:
      login: "Войти"
      email: "Ваш email"
      pass: "Пароль"
      remember: "Запомнить меня"
      forgot_pass: "Забыли пароль?"
      login_thru: "Войти через"
      signup_mess: "Еще не с нами? Зарегистрируйтесь!"
      form_title: "Вход"
      already_with_us: "Уже зарегистрированы? Войдите!"
      create_pass: "Придумайте пароль"
      repeat_pass: "Повторите пароль"
      signup: "Зарегистрироваться"
      signup_title: "Регистрация"
      search_example: "Пример поиска"
    admin_panel:
      title: "Пользователи"
      reg_date: "Дата регистрации"
      delete_user: "Удалить пользователя"
      create_admin: "Сделать админом"
      destroy_admin: "Убрать права админа"
    account:
      welcome: "Добро пожаловать"
      welcome_subtitle: "Посмотрите краткий tutorial по использованию!"
      use_this_token: "Используйте этот token для работы"
      create_new_token: "Обновить token"
      token_updated: "Token успешно обновлен!"
      tutorial_v1_title: "Использование API с помощью ключа (V1)"
    doc:
      v1:
        requests: "Запросы с помощью функции"

```

```

description: "Данная функция поддерживает 6 аргументов для поиска"
example: "Пример"
req: "Запрос"
response_format: "Формат ответа"
v2:
  main-title: "API V2 (JWT авторизация)"
  title: "Для получения токена необходимо сделать json запрос по адресу:"
  after-title: "В результате этого вы получите токен который необходимо добавлять в
header каждого запроса. Дальнейшая работа аналогична с версией v1."
  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'
    admin: 'Admin panel'
    login: 'Login'
    logout: 'Logout'
    signup: 'Start using API'
    account: 'Account'
    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'
  login_form:
    login: "Login"
    email: "Email"
    pass: "Password"
    remember: "Remember me"
    forgot_pass: "Forgot password?"
    login_thru: "Login thru"
    signup_mess: "Not with us yet? Register now!"
    form_title: "Login!"
    already_with_us: "Already registered? Login!"
    create_pass: "Create password"
    repeat_pass: "Confirm password"
    signup: "Signup"
    signup_title: "Registration"
    search_example: "Example of student searching"
  admin_panel:
    title: "Users"
    reg_date: "Registration date"
    delete_user: "Delete user"
    create_admin: "Give admin rights"
    destroy_admin: "Revoke admin rights"
  account:
    welcome: "Welcome"
    welcome_subtitle: "Check out a quick tutorial on how to use it!"
    use_this_token: "Use this token for V1 API"
    create_new_token: "Refresh token"
    token_updated: "Token successfully updated!"

```

```

tutorial_v1_title: "API V1"
doc:
  v1:
    requests: "Queries Using a Function"
    description: "This function supports 6 arguments to search."
    example: "Example"
    req: "Request"
    response_format: "Response format"
  v2:
    main-title: "API V2 (JWT authorization)"
    title: "To get the token, you need to make a json request at:"
    after-title: "As a result of this, you will receive a token that must be added to the header of
each request. Further work is similar with version v1."

```

### **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 worker'ов sidekiq.

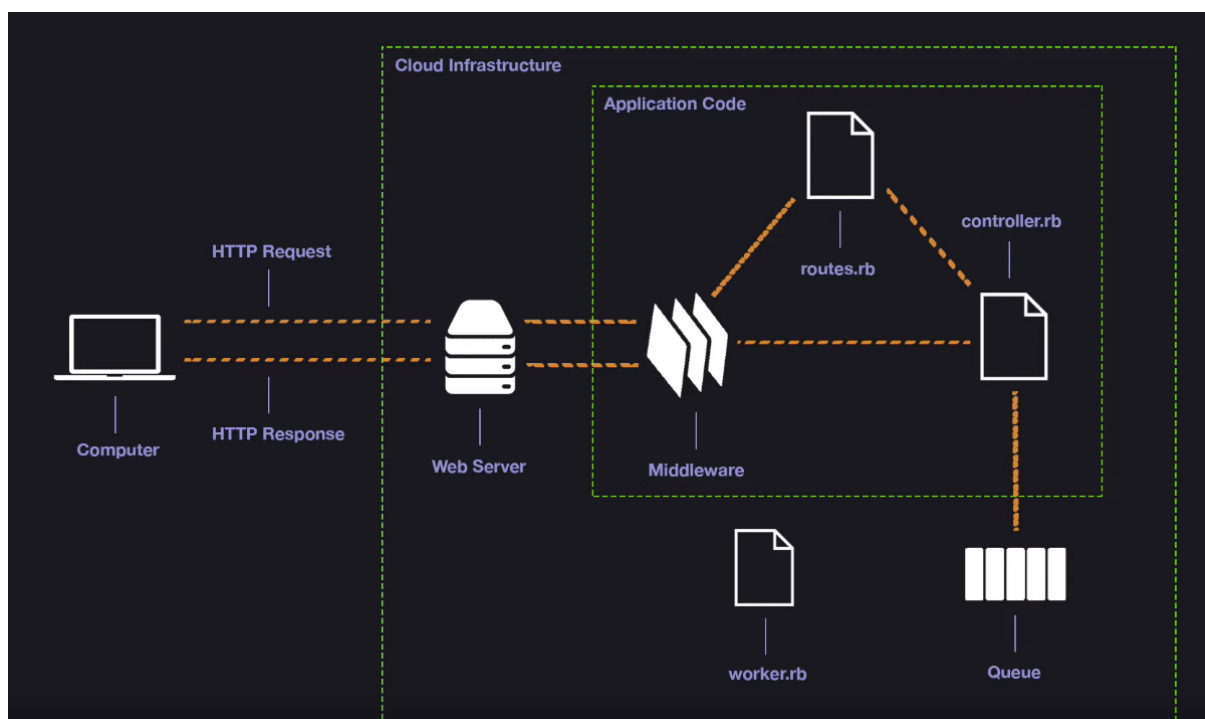


Рис. 1 (Приложение использующее sidekiq и redis)

### ***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
```

### ***Листинг parse\_journal\_worker.rb***

```
class ParseJournalWorker
  include Sidekiq::Worker

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

Для задания интервала вызовов в папке config был создан файл schedule.yaml. В данном случае интервал составляет 15 минут.

### ***Листинг schedule.yaml***

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

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. Каждый сервисный класс возвращает «Монаду».

### ***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 была использована технология «Dependency Injection». 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
  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
```

```

    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_v2' do
      RequestHandlers::RequestHandlerV2
    end

    register 'request_handler_v1' do
      RequestHandlers::RequestHandlerV1
    end

    register 'jwt_manager' do
      Other::JwtDecoder
    end

    register 'token_generator' do
      Other::TokenGenerator
    end

    register 'strategic_finder' do
      Bf::SearchLogic::StrategicFinder
    end

    register 'finder' do
      Bf::Finder
    end
  end
end

```

*Пример добавления зависимости:*

```

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

```

## Архитектура приложения

Для данного API предусмотрено 2 способа взаимодействия именуемые «V1» и «V2». В первом случае, пользователь получает API-ключ который он может бессрочно использовать при каждом запросе. Во втором же случае, пользователю необходимо предоставлять приложению JWT токен полученный при аутентификации через соответствующий JSON запрос.

Обработчиком запросов в случае с версией «V1» является контроллер `api_request_controller`.

### ***Listing `api_request_controller.rb`:***

```
# frozen_string_literal: true

module Api
  module V1
    class ApiRequestController < ApplicationController
      include Dry::Monads[:do, :result]
      include Dry::AutoInject(Container)[
        handler_v1: 'services.request_handler_v1'
      ]

      def handle_request
        @response = handler_v1.call(params).value!
        render :handle_request
      end
    end
  end
end
```

После получения запроса контроллер предоставляет проверку на валидность сервисному классу `request_handler_v1`.

### ***Listing `request_handler_v1.rb`:***

```
# frozen_string_literal: true

module RequestHandlers
  class RequestHandlerV1 < Service
    include Dry::Monads[:result, :do, :maybe, :try]
    include Dry::AutoInject(Container)[
      'services.key_keeper',
      bf: 'services.finder'
    ]

    def call(params = [])
      handle(params).bind do |data|
        Success(
          status: :success,
          data: data
        )
      end.or do |err|
        Success(
          status: :failed,
          cause: err,
          data: []
        )
      end
    end
  end
end
```

```

    end
  end

  private

  attr_accessor :keys

  def setup
    self.keys = yield key_keeper.call

    Success()
  end

  def handle(params)
    params = yield extract_search_params(params)
    yield setup
    yield valid_params?(params)
    yield valid_token?(params)
    data = yield bf.call(
      params[keys['search_method_key_name'].to_sym],
      params.reject { |k| k.eql?(keys['search_method_key_name']) || k.eql?
(keys['token_arg_name']) }
    )

    Success(data)
  end

  # validate income params from user
  def valid_params?(params)
    yield validate_search_function(params)
    yield validate_search_args(params)

    Success(:validation_succeeded)
  end

  def validate_search_function(params)
    return Failure(:missing_search_function) unless params.keys.include?
(keys['search_method_key_name'])

    sup_meth = keys['search_methods']
    return Failure(:invalid_search_method) unless sup_meth.include?
(params[keys['search_method_key_name'].to_sym])

    Success()
  end

  def validate_search_args(params)
    return Failure(:missing_token) unless params.key?(keys['token_arg_name'].to_sym)

    params = params.reject { |k| [keys['search_method_key_name'],
keys['token_arg_name']].include?(k) }
    return Failure(:missing_search_args) if params.empty?

    sup_args = keys['search_methods_args'].map(&.to_sym)
    return Failure(:invalid_search_args) unless params.keys.all? { |a| sup_args.include?
(a.to_sym) }

    Success()
  end

  def extract_search_params(params)
    params = params.reject { |k| %w[format controller action].include?(k) }

```

```

    Success(params)
  end

  def valid_token?(params)
    Maybe(User.find_by(bf_api_token: params[keys['token_arg_name']])).bind
  { Success() }.or(Failure(:invalid_token))
  end
end
end
end

```

В случае успешной проверки на валидность данные передаются в класс Finder.

### ***Листинг finder.rb:***

```

# frozen_string_literal: true

module Bf
  class Finder < Service
    include Dry::Monads[:result, :do, :maybe, :try]
    include Dry::AutoInject(Container)[
      'services.key_keeper',
      sf: 'services.strategic_finder'
    ]

    def call(search_function, search_args = {})
      yield setup
      data = yield find_by_strategy(search_function, search_args)

      Success(data)
    end

    private

    attr_reader :keys

    def setup
      @keys = yield key_keeper.call

      Success()
    end

    def find_by_strategy(search_function, search_args)
      inter_search = Student.all
      search_args.each do |key, value|
        inter_search = yield sf.call(inter_search, key.to_sym, search_function.to_sym, key => value)
      end

      Success(inter_search)
    end
  end
end
end

```

Для поиска студентов используется следующий формат запроса:

«<http://localhost:3000/find?APIKEY=1234&name=Имя>», где find — «поисковая функция» и name - «аргумент поиска». Поскольку аргументы поиска не всегда могут являться простыми полями в интересующей нас таблице, то логика поиска может варьироваться в зависимости от каждого аргумента. Для решения данной проблемы было принято решение использовать паттерн «стратегия» позволивший

предоставить логику поиска различным классам, являющимися потомками специально созданного «интерфейса» «Strategy».

***Листинг strategic\_finder.rb (поиск по стратегии):***

```
# frozen_string_literal: true

module Bf
  module SearchLogic
    class StrategicFinder < Service
      include Dry::Monads[:result, :maybe]

      attr_accessor :strategy
      attr_reader :strategies_list

      def call(inter_search, strategy_name, search_func, arg)
        yield setup(strategy_name)
        res = yield find(inter_search, search_func, arg)

        Success(res)
      end

      private

      def setup(str_name)
        @strategies_list = {
          name: FindByValue,
          last_name: FindByValue,
          mid_name: FindByValue,
          exam_scores: FindByValue,
          group: FindByGroup
        }
        self.strategy = strategies_list[str_name].new

        Success()
      end

      def find(inter_search, search_func, arg)
        Success(strategy.search(inter_search, search_func, arg))
      end
    end

    class Strategy
      include Dry::Monads[:do]

      attr_reader :search_methods, :keys

      def initialize
        @keys = yield KeyKeeper.call
      end

      def search
        raise NotImplementedError, "The class #{self.class} has not implemented #{__method__}"
      end
    end

    class FindByValue < Strategy
      def initialize
        super
        @search_methods = {
          find: ->(data, args) { data.where(args) },

```

```

    find_except: ->(data, args) { data.where.not(args) }
  }
end

def search(data, search_func, args)
  args = args.transform_keys { |key| keys['search_args_matching'][key.to_s] }
  search_methods[search_func.to_sym].call(data, args)
end
end

class FindByGroup < Strategy
  def initialize
    super
    @search_methods = {
      find: ->(data, args) { data.joins(:group).where(groups: { name: args }) },
      find_except: ->(data, args) { data.joins(:group).where.not(groups: { name: args }) }
    }
  end

  def search(data, search_func, args)
    args = args.transform_keys { |key| keys['search_args_matching'][key.to_s] }
    search_methods[search_func].call(data, args.values.first)
  end
end
end
end

```

В приведенном коде (см. листинг `strategic_finder.rb`) можно увидеть, что для поиска студентов по группам, необходимо сделать «join» с таблицей «group».

Поиск через `api/v2` практически идентичен, за исключением одного дополнительного пункта. В случае отсутствия или истечения времени жизни JWT токена пользователю необходимо авторизоваться используя свою почту и пароль.

### ***Листинг `auth_controller.rb` (контроллер аутентификации для версии v2):***

```

# frozen_string_literal: true

require 'json'

module Api
  # module Api
  module V2
    # Authorization controller
    class AuthController < ApplicationController
      include Dry::Monads[:try, :maybe]
      include Dry::AutoInject(Container)[
        jwt: 'services.jwt_manager'
      ]

      def create
        if data = params['auth']
          @user = User.find_by(email: data['email'])
          if @user&.valid_password?(data['password'])
            @user.update_token
            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']
        BlackList.destroy_token(token)
        render json: { message: 'token successfully destroyed!' }
      else
        render json: { message: 'invalid params!' }
      end
    end
  end
end
end
end
end

```

### ***Листинг find\_controller.rb (контроллер обработки запросов v2):***

```

# frozen_string_literal: true

module Api
  # Api module
  module V2
    # FindController class
    class FindController < ApplicationController
      include RequestHandlers
      include Dry::AutoInject(Container)[
        'models.black_list',
        'services.request_handler_v2'
      ]

      def find
        if black_list.check_token(request.headers['token'])
          @response = request_handler_v2.call(params['search']).value!
          render :find, content_type: 'application/json'
        else
          render json: { message: 'authentication failed' }
        end
      end
    end
  end
end
end
end

```

### ***Листинг request\_handler\_v2.rb:***

```

# frozen_string_literal: true

# Request handler class
module RequestHandlers
  # Request handler class
  class RequestHandlerV2 < Service
    include Dry::Monads[:result, :do, :maybe, :try]
    include Dry::AutoInject(Container)[
      'services.key_keeper',
      bf: 'services.finder'
    ]

    def call(params = {})
      handle(params).bind do |data|
        Success(
          status: :success,

```



```

      data: data
    )
  end.or do |err|
    Success(
      status: :failed,
      cause: err,
      data: []
    )
  end
end

private

attr_accessor :keys

def handle(params)
  yield setup
  yield validate_search_args(params)
  data = yield bf.call(:find, params)

  Success(data)
end

def setup
  self.keys = yield key_keeper.call

  Success()
end

def validate_search_args(params)
  unless params && params.keys.all? { |k| keys['search_methods_args'].include?(k.to_s) }
    return Failure(:invalid_args)
  end

  Success()
end
end
end

```

### **Регистрация пользователей**

Регистрация пользователей реализована путем использования гема «devise». Также была добавлена поддержка «OAuth» авторизаций через vk, facebook и github.

### ***Листинг user.rb (Модель User):***

```

# frozen_string_literal: true

# User model
class User < ApplicationRecord
  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :validatable, :confirmable,
         :omniauthable, omniauth_providers: %i[vkontakte facebook github]

  validates :email, uniqueness: true
  validates :bf_api_token, uniqueness: true
  before_create :generate_token

  # updates jwt token

```

```

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

# get token part for user
def jwt_payload
  jwt_token.match(/\.(\w+\.\w+)/)[1]
end

# parse username from email
def username
  email.match(/^[^@]+/).to_s
end

# generates bf token for v1 api
def generate_token
  begin
    token = Other::TokenGenerator.call(25).value_or('')
  end until !User.where(token: token).nil?
  self.bf_api_token = token
end

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

  def new_with_session(params, session)
    super.tap do |user|
      if data = session['devise.facebook_data'] &&
        session['devise.facebook_data']['extra']['raw_info']
        user.email = data['email'] if user.email.blank?
      end
    end
  end

  def from_omniauth_facebook(auth)
    where(provider: auth.provider, uid: auth.uid).first_or_initialize do |user|
      user.email = auth.info.email
      user.password = Devise.friendly_token[0, 20]
      user.bf_username = auth.info.name
      user.image_url = auth.info.image
    end
  end

  def from_omniauth_vk(auth)
    where(provider: auth.provider, uid: auth.uid).first_or_initialize do |user|
      user.uid = auth.uid
      user.provider = auth.provider
      user.email = auth.info.email
      user.password = Other::TokenGenerator.call(6).value_or('123456')
      user.image_url = auth.extra.raw_info.photo_400_orig
      user.bf_username = auth.info.name || 'nameless'
    end
  end
end

```

```

end

def from_omniauth_github(auth)
  where(provider: auth.provider, uid: auth.uid).first_or_initialize do |user|
    user.uid = auth.uid
    user.provider = auth.provider
    user.email = auth.info.email
    user.password = Other::TokenGenerator.call(6).value_or('123456')
  end
end

end

end

end

```

### ***Листинг devise.rb (конфигурация devise):***

```

...
config.omniauth :facebook, oauth_secrets[:facebook_app_id],
  oauth_secrets[:facebook_app_secret], scope: 'email', token_params:
  { parse: :json }, :provider_ignores_state => true
  config.omniauth :vkontakte, oauth_secrets[:vk_app_id],
  oauth_secrets[:vk_app_secret],
    scope: 'email', display: 'popup', token_params:
  { parse: :json }, :provider_ignores_state => true
  config.omniauth :github, oauth_secrets[:github_app_id],
  oauth_secrets[:github_app_id], scope: "user,repo,gist"
...

```

### ***Листинг omniauth\_callbacks\_controller.rb:***

```

# frozen_string_literal: true

module Users
  class OmniauthCallbacksController < Devise::OmniauthCallbacksController
    def facebook
      @user = User.from_omniauth_facebook(request.env['omniauth.auth'])
      omniauth_handle_user(@user, :facebook)
    end

    def vkontakte
      @user = User.from_omniauth_vk(request.env['omniauth.auth'])
      omniauth_handle_user(@user, :vkontakte)
    end

    def github
      @user = User.from_omniauth_github(request.env['omniauth.auth'])
      omniauth_handle_user(@user, :github)
    end

    def omniauth_handle_user(user, provider)
      if user.persisted?
        sign_in_and_redirect @user, event: :authentication
        set_flash_message(:notice, :success, kind: provider.to_s) if
is_navigational_format?
      else
        if user.valid?
          user.tap do |u|
            u.skip_confirmation!
            u.save!
          end
          sign_in_and_redirect user, event: :authentication
          set_flash_message(:notice, :success, kind: provider.to_s) if
is_navigational_format?
        else

```

```

        redirect_to new_user_registration_url, notice: 'Пользователь с такой
почтой уже существует!'
      end
    end
  end
end

def failure
  puts 'Error while omniauth callback handling'
  redirect_to doc_path
end
end
end

```

Приложение предусматривает наличие администраторов, которые имеют возможность удалять, а также менять права доступа у других пользователей.

### ***Листинг admin\_controller.rb:***

```

# frozen_string_literal: true

class AdminController < ApplicationController
  include Devise
  before_action :check_admin, only: [:admin]
  REFRESH_USERS_JS = 'refresh_user_list.js.erb'

  def admin; end

  def destroy_user
    usr = User.find_by(id: params[:id])
    User.destroy(usr.id) unless usr.nil?
    respond_to { |format| format.js { render action: REFRESH_USERS_JS } }
  end

  def change_permissions
    usr = User.find_by(id: params[:id])
    usr&.tap do |u|
      u.is_admin = !u.is_admin
      u.save!
    end
    respond_to { |format| format.js { render action: REFRESH_USERS_JS } }
  end

  private

  def check_admin
    redirect_to home_path unless current_user.is_admin
  end
end

```

### ***Листинг db/seeds.rb:***

```

User.new.tap do |admin|
  admin.email = Rails.application.credentials.default_admin_email!
  admin.password = Rails.application.credentials.default_admin_pass!
  admin.bf_api_token = Other::TokenGenerator.call(6).value_or("123456")
  admin.is_admin = true
  admin.skip_confirmation!
  admin.save!
end

```

## Служебные функции приложения

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+1075676763663A2F2F636265676E79332E72682E6F7A666768
    2E6568++/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+1h75676763663A2F2F72682E6F7A6667682E6568+
  +/modules/progress3/2019-01/'
  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]'
  search_method_key_name: 'search_meth'
  search_methods:
    - 'find'
    - 'find_first'
    - 'find_except'
    - 'find_except_first'
  token_arg_name: 'APIKEY'
  search_methods_args:
    - 'name'
    - 'last_name'
    - 'mid_name'
    - 'group'
    - 'year_of_admission'
    - 'exam_scores'
  search_args_matching:
    'name': 'first_name'
    'last_name': 'last_name'
    'mid_name': 'mid_name'
    'group': 'group'
    'exam_scores': 'exam_scores'
```

### ***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 token\_generator.rb (генератор случайных токенов):***

```
# frozen_string_literal: true

# All service objects
module Other
  # Class for encode/decode jwt key
  class TokenGenerator < Service
    include Dry::Monads[:result, :do]

    def call(size)
      result = yield generate_by_size(size)

      Success(result)
    end

    private

    def generate_by_size(size)
      token = 1.upto(size).map { (('a'..'z').to_a + ('A'..'Z').to_a + ('0'..'9')).to_a[rand(62)] }.join
      Success(token)
    end
  end
end
```

```

    end
  end
end

```

### ***Листинг jwt\_decoder.rb (класс для генерации и расшифровки jwt токенов):***

```

# 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

```

Удаление jwt токенов происходит путем добавления их в черный список.

### ***Листинг token\_manager.rb (менеджер токенов)***

```

# 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)
      token = restore_jwt_token(token)
      Other::JwtDecoder.call.decode_key(token).bind do |data|
        exp = data['expires'].to_time
      end
    end
  end
end

```

```

    return true if (exp - Time.now).positive? && !BlackList.in_black_list?(token)

    return false
  end
  false
end

def destroy_token(token)
  token = restore_jwt_token(token)
  BlackList.find_or_create_by(token: token)
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)
  token = restore_jwt_token(token)
  Other::JwtDecoder.call.decode_key(token).bind do |val|
    return User.find_by(email: val['user_email'])
  end
end

def restore_jwt_token(token)
  "eyJhbGciOiJIUzI1NiJ9.#{token}"
end
end
end

```

### ***Листинг page\_helper.rb:***

```

# frozen_string_literal: true

module PageHelper
  def account_username
    return 'nameless' unless current_user

    current_user.tap do |usr|
      usr.update(bf_username: usr.username) if usr.bf_username.blank?
    end.bf_username
  end

  def user_bf_token
    current_user&.update_token if current_user&.bf_api_token.blank?
    current_user&.bf_api_token
  end

  def random_student(mode = :all)
    case mode
    when :with_subj_data
      Student.all.reject { |s| s.subject_data.blank? }.sample
    else
      Student.all.sample
    end
  end
end

```



### ***Листинг admin\_helper.rb:***

```
module AdminHelper
  def user_list
    User.all
  end
end
```

### **Парсеры:**

### ***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 parsing
    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']
          end
        end
      end
    end
  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
end

```

Для более удобного использования парсера приказов был создан файл конфигурации.

### ***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]+\s)*;. +иента\s*№:\s*([^\s;]+\s)*;. +ента\s*№:\s*([^\s;]+\s)*;. +ппа:\s*([а-яА-Я]+[^\s;]+\s)*. +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]+\s)*;. +иента\s*№:\s*([^\s;]+\s)*;. +ента\s*№:\s*([^\s;]+\s)*;. +ппа:\s*([а-яА-Я]+[^\s;]+\s)*. +лов:\s*(\d+);'
      model:
        - 'last_name'
        - 'first_name'
        - 'mid_name'
        - 'id_abitur'
        - 'id_stud'
        - 'group_adm'
        - 'exam_scores'
...

```

### ***Листинг web\_scraping.rb (написан 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?('сместр')

        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
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
  end
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

```

***Листинг parser\_manager.rb (управление всеми парсерами ):***

```

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

    def call
      self
    end

    def update_decreed_data
      decreed_data = yield decreed_parser.call
      data_validator.call(decreed_data, :decreed_data)
    end

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

```

## **Модели**

### ***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
```

## Пример работы приложения

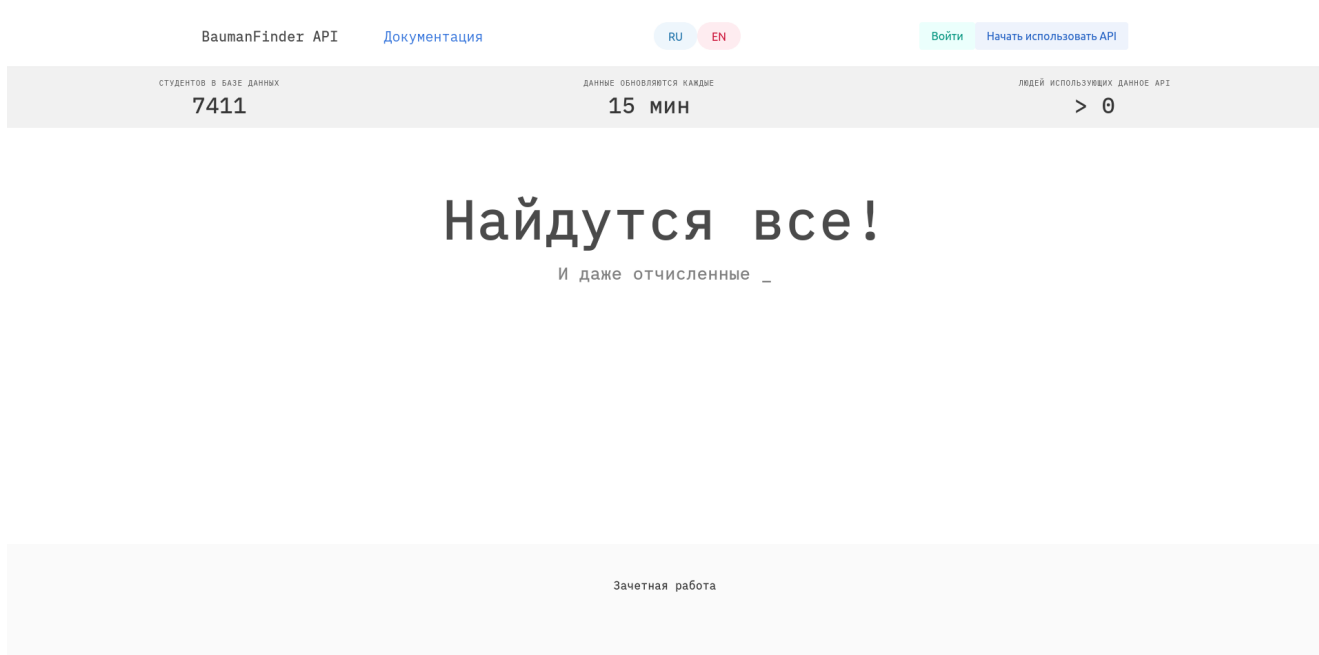


Рис. 2 (Главная страница приложения)

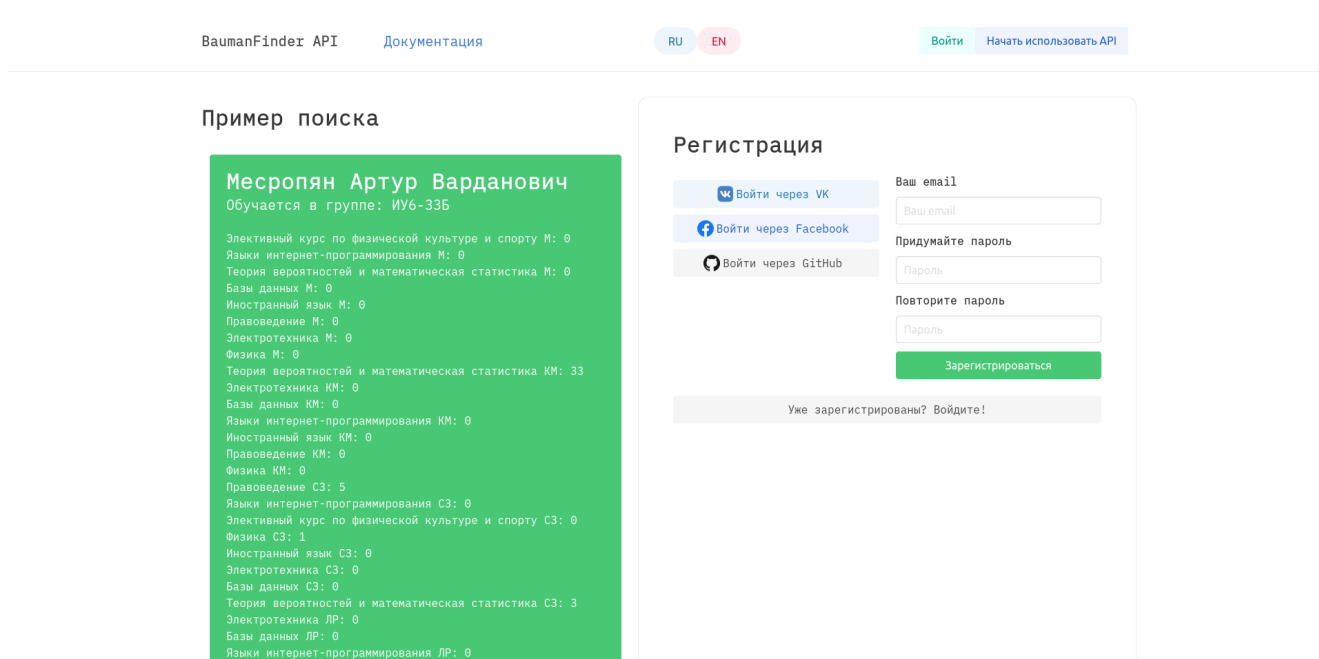


Рис. 3 (Страница регистрации)



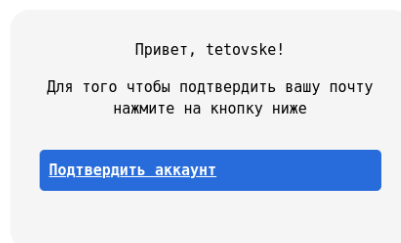


Рис. 4 (Письмо с подтверждением регистрации)

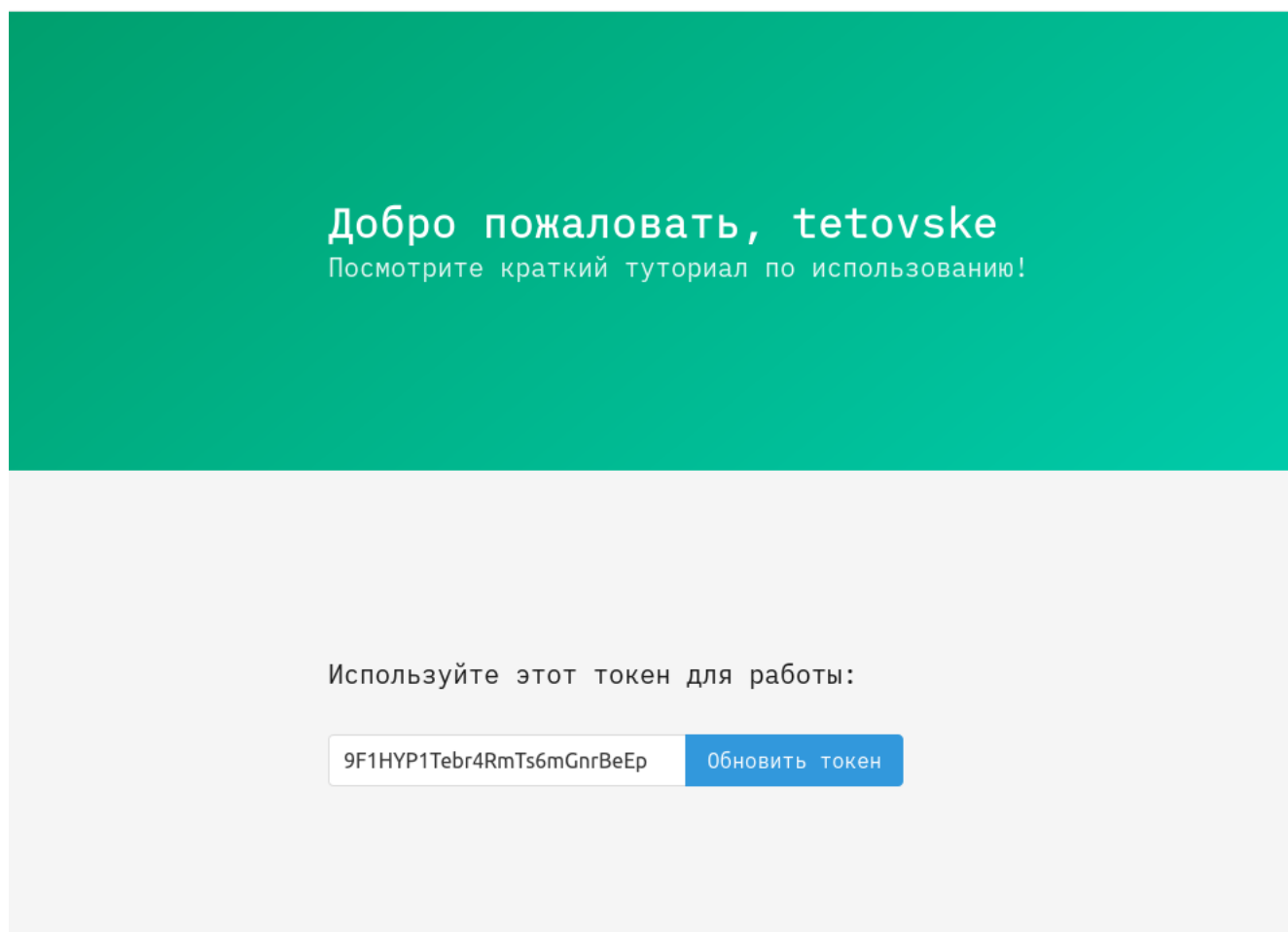


Рис. 5 (Личный кабинет)

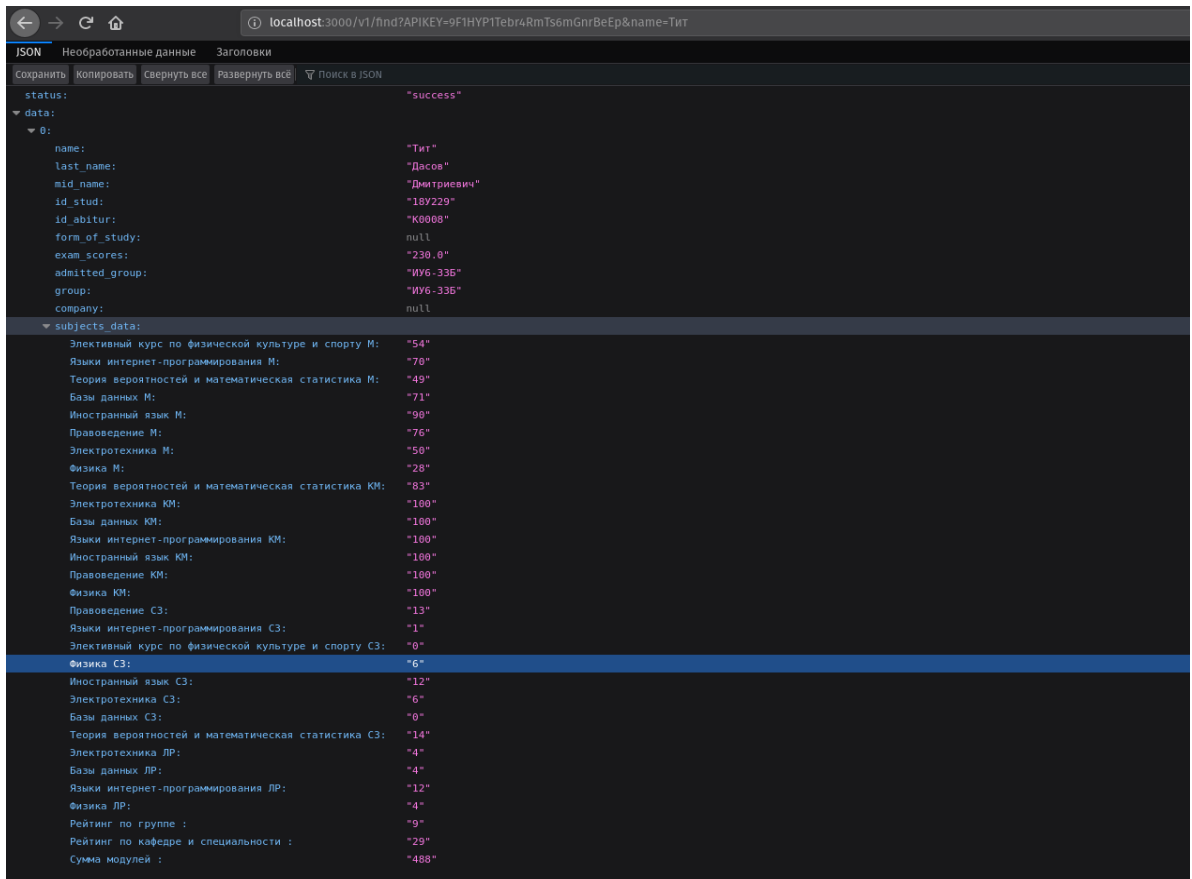


Рис. 6 (Пример ответа API/V1)

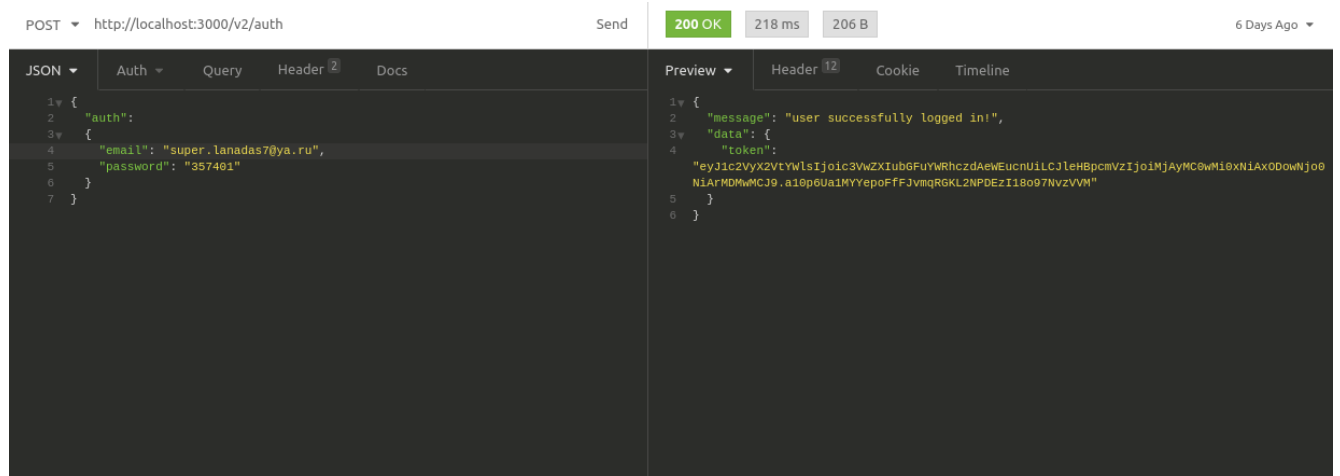


Рис. 7 (Авторизация API/V2)

GET http://localhost:3000/v2/find Send 200 OK 23.5 ms 3.6 KB Just Now

JSON Auth Query Header 2 Docs Preview Header 12 Cookie Timeline

```

1 {
2   "search":
3   {
4     "group": "ИУ6-335",
5     "name": "Иван"
6   }
7 }

```

```

1 {
2   "status": "success",
3   "data": {
4     {
5       "name": "Иван",
6       "last_name": "Кравцов",
7       "mid_name": "Константинович",
8       "id_stud": "18v331",
9       "id_abitur": "И2289",
10      "form_of_study": null,
11      "exam_scores": "219.0",
12      "admitted_group": "ИУ6-335",
13      "group": "ИУ6-335",
14      "company": null,
15      "subjects_data": {
16        "Элективный курс по физической культуре и спорту М": "56",
17        "Языки интернет-программирования М": "69",
18        "Теория вероятностей и математическая статистика М": "48",
19        "Базы данных М": "71",
20        "Иностранный язык М": "16",
21        "Правоведение М": "70",
22        "Электротехника М": "43",
23        "Физика М": "45",
24        "Теория вероятностей и математическая статистика КМ": "83",
25        "Электротехника КМ": "67",
26        "Базы данных КМ": "100",
27        "Языки интернет-программирования КМ": "100",
28        "Иностранный язык КМ": "100",
29        "Правоведение КМ": "100",
30        "Физика КМ": "100",
31        "Правоведение СЗ": "13",
32        "Языки интернет-программирования СЗ": "1",
33        "Элективный курс по физической культуре и спорту СЗ": "0",
34        "Физика СЗ": "8",
35        "Иностранный язык СЗ": "15",
36        "Электротехника СЗ": "8",
37        "Базы данных СЗ": "0",
38        "Теория вероятностей и математическая статистика СЗ": "15",
39        "Электротехника ЛР": "4",
40        "Базы данных ЛР": "4",
41        "Языки интернет-программирования ЛР": "12",
42        "Физика ЛР": "6",
43        "Рейтинг по группе ": "13",
44        "Рейтинг по кафедре и специальности ": "42",
45        "Сумма модулей ": "418"
46      }
47    }
48  }
49 }

```

Рис. 8 (Пример поиска API/V2)

## Листинг routes.rb:

```

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

  mount Sidekiq::Web => '/sidekiq'
  devise_for :users, only: :omniauth_callbacks, :controllers => { :omniauth_callbacks =>
"users/omniauth_callbacks" }

  scope "(:region)", region: /#{l18n.available_locales.join('|')}/ do
    root 'page#home', :as => 'home'
    devise_for :users, skip: :omniauth_callbacks, :controllers => { :omniauth_callbacks =>
"users/omniauth_callbacks" }

    get 'test/output', :as => 'output'
    get '/documentation' => 'page#doc', :as => 'doc'
    get '/account' => 'page#account', :as => 'account'
    get '/regenerate' => 'page#regenerate_token', :as => 'regenerate_token'

    scope 'admin' do
      get '/' => 'admin#admin', :as => 'admin'
      delete '/destroy_user/:id' => 'admin#destroy_user', :as => 'admin_destroy_user'
      post '/change_permissions/:id' => 'admin#change_permissions', :as =>
'change_permissions_for'
    end
  end

  scope module: 'api', defaults: { format: 'json' } do
    scope module: 'v1', path: 'v1' do

```

```

    get '/:search_meth' => 'api_request#handle_request', :as => 'v1_handler'
  end

  scope module: 'v2', path: 'v2' do
    resources :auth, only: [:create]
    delete '/signout' => 'auth#signout'
    get '/find' => 'find#find'
  end
end
end
end

```

## **БД и миграции**

Для хранения всех данных приложения была использована СУБД «postgresql».

Пример миграции:

### ***Листинг add\_some\_information\_to\_profile.rb (файл миграции):***

```

class AddSomeInformationToProfile < ActiveRecord::Migration[6.0]
  def change
    reversible do |direction|
      change_table :users do |t|
        direction.up do
          t.string :image_url, :bf_username
          t.change :email, :string, default: nil, null: true
        end

        direction.down do
          t.remove :image_url, :bf_username
          t.change :email, :string, default: "", null: false
        end
      end
    end
  end
end

```

## **Тесты приложения**

### ***Листинг login\_spec.rb (тест регистрации):***

```

require 'rails_helper'

RSpec.feature "Logins", type: :feature do
  let(:fake_user) { FactoryBot.build(:default_user) }
  let(:not_hacker) { FactoryBot.create(:default_user) }
  let(:admin) { FactoryBot.create(:admin) }

  describe "user authentication thru devise" do
    scenario "hacker trying to login" do
      locale = I18n.default_locale
      visit "#{locale}#{new_user_session_path}"
      within(:xpath, '//*[@id="new_user"]') do
        fill_in "user[email]", with: fake_user.email
        fill_in "user[password]", with: fake_user.password
      end
      click_button "commit"
    end
  end
end

```

```

    expect(page).to have_content(I18n.t('devise.failure.not_found_in_database',
locale: locale))
  end

  scenario "default user trying to login" do
    locale = I18n.default_locale
    visit "#{locale}#{new_user_session_path}"
    within(:xpath, '//*[@id="new_user"]') do
      fill_in "user[email]", with: not_hacker.email
      fill_in "user[password]", with: not_hacker.password
    end
    click_button "commit"
    expect(page).to have_content(I18n.t('devise.sessions.signed_in', locale:
locale))
  end

  scenario "admin trying to login" do
    locale = I18n.default_locale
    visit "#{locale}#{new_user_session_path}"
    within(:xpath, '//*[@id="new_user"]') do
      fill_in "user[email]", with: admin.email
      fill_in "user[password]", with: admin.password
    end
    click_button "commit"
    expect(page).to have_content(I18n.t('home_page.admin', locale: locale))
  end
end
end
end

```

### ***Листинг internationalization\_spec.rb (мест перевода):***

```

require 'rails_helper'

RSpec.feature "Internationalizations", type: :feature do
  describe "if we are going to home page" do
    context "content of header" do
      let(:xpath_to_header) do
        {
          'div[1]/div/p[1]' => 'home_page.after_header.students_in',
          'div[2]/div/p[1]' => 'home_page.after_header.updates'
        }
      end

      it "should contain correct translated header" do
        I18n.available_locales.each do |locale|
          visit "#{locale}"
          within(:xpath, "/html/body/nav[2]") do
            xpath_to_header.each do |xpath, content|
              expect(page).to have_xpath(xpath, text: I18n.t(content, locale:
locale))
            end
          end
        end
      end
    end
  end

  context "content of body" do
    let(:xpath_to_body) do
      {
        'div/p[1]' => 'home_page.body.main_txt',
        'div/p[2]' => 'home_page.body.after_main_txt'
      }
    end
  end
end

```

```

end

it "should contain correct translated body" do
  I18n.available_locales.each do |locale|
    visit "#{locale}"
    within(:xpath, "/html/body/div/div/section") do
      xpath_to_body.each do |xpath, content|
        expect(page).to have_xpath(xpath, text: I18n.t(content, locale:
locale))
      end
    end
  end
end
end
end
end
end
end
end

```

### ***Листинг api\_search\_spec.rb (тест корректности поиска данных):***

```

require 'rails_helper'

RSpec.feature "ApiSearches", type: :feature do

  describe "v1 api" do
    let(:create_user) { FactoryBot.create(:default_user) }

    context "if we are submit options without APIKEY" do
      let(:result_without_key) do
        {
          'status': 'failed',
          'cause': 'missing_token',
          'data': []
        }
      end

      it "should return empty list because there is no APIKEY" do
        visit v1_handler_path(search_meth: 'find')
        expect(JSON.parse(page.body).transform_keys(&:to_sym)).to
        eq(result_without_key)
      end
    end

    context "if we are submitting options with APIKEY" do
      context "if we submitting invalid arguments with valid APIKEY" do
        let(:missing_search_args) do
          {
            'status': 'failed',
            'cause': 'missing_search_args',
            'data': []
          }
        end

        let(:invalid_search_meth) do
          {
            'status': 'failed',
            'cause': 'invalid_search_method',
            'data': []
          }
        end

        it "should return cause = missing_search_args" do
          visit v1_handler_path(search_meth: 'find', APIKEY:
create_user.bf_api_token)

```

```

        expect(JSON.parse(page.body).transform_keys(&:to_sym)).to
eq(missing_search_args)
      end

      it "should return cause = invalid_search_method beacause we have
invalid_search_meth!" do
        visit v1_handler_path(search_meth: 'finds', APIKEY:
create_user.bf_api_token)
        expect(JSON.parse(page.body).transform_keys(&:to_sym)).to
eq(nvalid_search_meth)
      end
    end

    context "if we are submitting valid args" do

      it "should have status 'success'" do
        visit v1_handler_path(search_meth: 'find', APIKEY:
create_user.bf_api_token, name: 'Name')
        expect(JSON.parse(page.body)['status']).to eq('success')
      end

      it "should return student" do
        student = FactoryBot.create(:student)
        visit v1_handler_path(search_meth: 'find', APIKEY:
create_user.bf_api_token, name: student.first_name)
        expect(JSON.parse(page.body)['status']).to eq('success')
      end
    end

    context "and want to print all students" do
      create_student

      it "should return not empty list" do
        visit v1_handler_path(search_meth: 'find_except', APIKEY:
create_user.bf_api_token, name: '')
        expect(JSON.parse(page.body)['status']).to eq('success')
        expect(JSON.parse(page.body)['data']).not_to be_empty
      end
    end
  end
end
end
end

```

### ***Листинг user\_spec.rb (тест модели user):***

```

require 'rails_helper'

RSpec.describe User, type: :model do
  context 'if user is trying to add existing variables' do
    it 'should return false because email is uniq' do
      fake_email = Faker::Internet.email
      create(:default_user, email: fake_email)
      expect(build(:default_user, email: fake_email).valid?).to be_falsy
    end

    it 'should also return false as tokens should not be repeated!' do
      fake_token = Faker::Internet.password
      create(:default_user).tap do |u|
        u.bf_api_token = fake_token
        u.save!
      end
    end
  end
end

```

```

      expect(build(:default_user).tap { |u| u.bf_api_token =
fake_token }.valid?).to be_falsy
    end
  end
end

```

### ***Листинг student\_spec.rb (тест модели student):***

```

require 'rails_helper'

RSpec.describe Student, type: :model do
  context 'if we are trying to add student with exsisting stud_id' do
    it 'should return false because stud_id is uniq!' do
      fake_student_id = Faker::Internet.password
      create(:student, id_stud: fake_student_id)
      expect(build(:student, id_stud: fake_student_id).valid?).to be_falsy
    end
  end
end

```

Для очистки тестовой бд перед каждым тестом использовался гем «DatabaseCleaner»

### ***Листинг support/database\_cleaner.rb (конфигурация гема):***

```

RSpec.configure do |config|
  config.before(:suite) do
    DatabaseCleaner.clean_with(:truncation)
  end

  config.before(:each) do
    DatabaseCleaner.strategy = :transaction
  end

  config.before(:each, :js => true) do
    DatabaseCleaner.strategy = :truncation
  end

  config.before(:each) do
    DatabaseCleaner.start
  end

  config.after(:each) do
    DatabaseCleaner.clean
  end
end

```

Для удобной генерации пользователей для тестов был также использован гем FactoryBot.

### ***Листинг user\_factory.rb:***

```

FactoryBot.define do
  factory :default_user, class: User do

```



```

    email { Faker::Internet.email }
    password { Faker::Internet.password }
    bf_api_token { Faker::Internet.password }
    confirmed_at { Date.today }
  end

  factory :admin, class: User do
    email { Faker::Internet.email }
    password { Faker::Internet.password }
    bf_api_token { Faker::Internet.password }
    is_admin { true }
    confirmed_at { Date.today }
  end
end
end

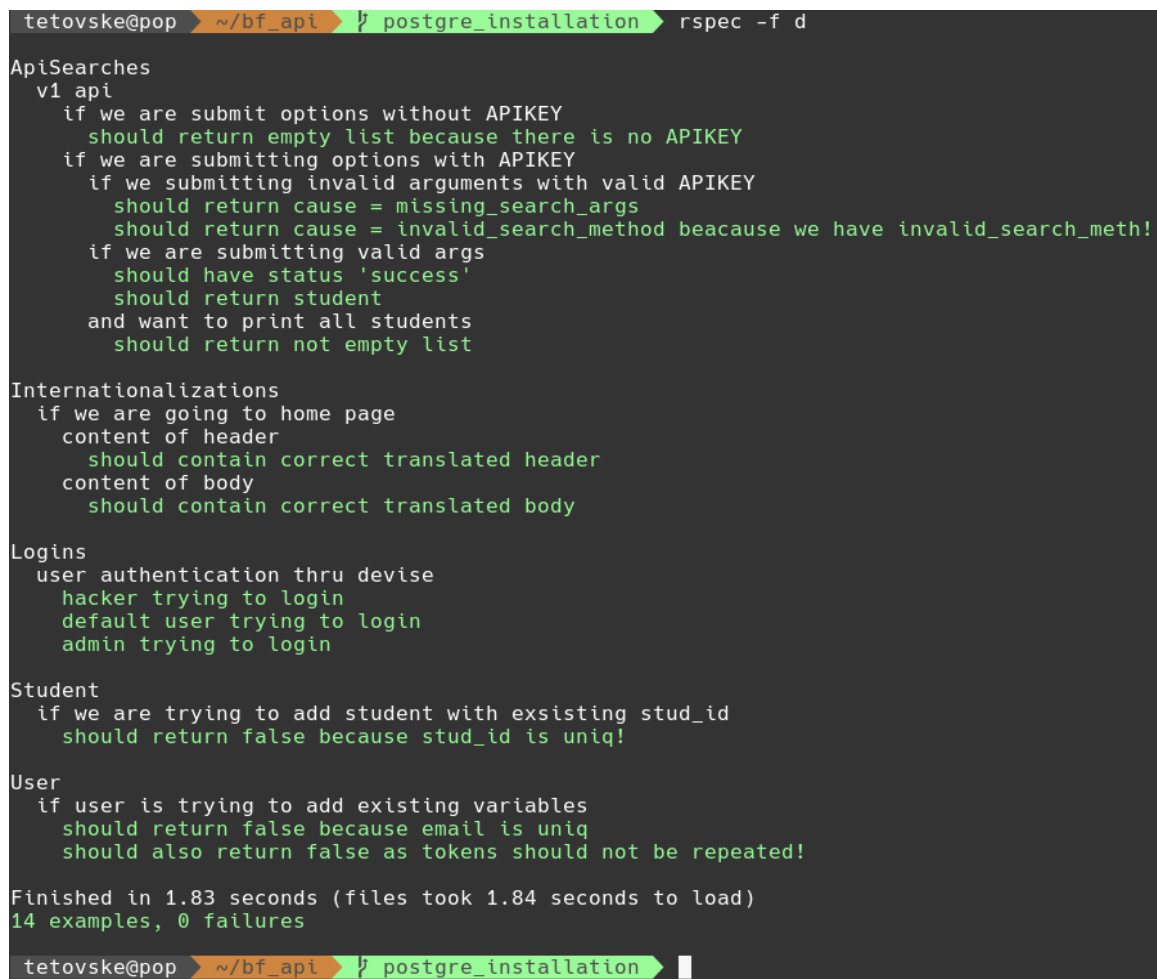
```

### ***Листинг student\_factory.rb:***

```

FactoryBot.define do
  factory :student, class: Student do
    first_name { Faker::Name.first_name }
    last_name { Faker::Name.last_name }
    id_stud { Faker::Internet.password }
  end
end
end

```



```

tetovske@pop > ~/bf_api > postgres_installation > rspec -f d
ApiSearches
  v1 api
    if we are submit options without APIKEY
      should return empty list because there is no APIKEY
    if we are submitting options with APIKEY
      if we submitting invalid arguments with valid APIKEY
        should return cause = missing_search_args
        should return cause = invalid_search_method because we have invalid_search_meth!
      if we are submitting valid args
        should have status 'success'
        should return student
      and want to print all students
        should return not empty list
  Internationalizations
    if we are going to home page
      content of header
        should contain correct translated header
      content of body
        should contain correct translated body
  Logins
    user authentication thru devise
      hacker trying to login
      default user trying to login
      admin trying to login
  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 because email is uniq
      should also return false as tokens should not be repeated!
Finished in 1.83 seconds (files took 1.84 seconds to load)
14 examples, 0 failures

tetovske@pop > ~/bf_api > postgres_installation >

```

*Рис. 9 (Результаты RSpec тестов)*

**Листинг .rubocop.yml (Файл конфигурации rubocop):**

```
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. 10 (Отчет Rubocop)*

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