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

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

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

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

О Т Ч Е Т

По домашней работе

Название: Зачетная работа.

Дисциплина: Языки интернет программирования

Студент ИУ6-32Б _____ Баканов Р.В.
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ Самарев Р.С.
(Подпись, дата) (И.О. Фамилия)

Москва, 2020

Содержание

Задание

Продемонстрировать полученные знания в создании собственного веб-приложения. Тему зачётной работы допустимо выбрать самостоятельно, но обязательно согласовать с преподавателем.

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.

Что из себя должен представлять проект:

Приложение, позволяющее зарегистрированному пользователю создавать карточки, содержащие описание дел, которые надо сделать в обозримом будущем. Данные карточки можно группировать по трем основным категориям: «В планах», «В работе», «Выполнено»; редактировать, устанавливать статус видимости для других пользователей, а также перемещать в корзину, где в будущем их можно будет окончательно удалить или, наоборот, восстановить. Также можно изменять данные самого пользователя (имя, электронную почту, пароль и фото). Реализована возможность просмотра списка всех пользователей и открытого для просмотра списка дел конкретного пользователя.

Исходный код

Контроллеры:

```ruby
class ApplicationController < ActionController::Base
  helper_method :current_user
  before_action :authenticate, except: %i[new create] # :signup, :login,

  def current_user
    if session[:user_id]
      @current_user ||= User.find(session[:user_id])
    else
      @current_user = nil
    end
  end

  private

  def authenticate
    unless current_user
      redirect_to login_path
    end
  end
end
```

Листинг 1 – application_controller.rb

```ruby
class UsersController < ApplicationController
  before_action :set_user, only: %i[show edit update destroy]
  skip_before_action :authenticate, only: :index

  # GET /users
  # GET /users.json
  def index
    @users = User.all
    redirect_to signup_path
  end

  # GET /users/1
  # GET /users/1.json
  def show; end

  # GET /users/new
  def new
    @user = User.new
  end

  # GET /users/1/edit
  def edit; end

  # POST /users
  # POST /users.json
  def create
    @user = User.new(user_params)

    respond_to do |format|
      if @user.save
        format.html { redirect_to login_path, notice: 'Успешная регистрация!' }
        format.json { render :show, status: :created, location: @user }
      else
        format.html { render :new }
        format.json { render json: @user.errors, status: :unprocessable_entity }
      end
```

```
    end
  end

  # PATCH/PUT /users/1
  # PATCH/PUT /users/1.json
  def update
    respond_to do |format|
      if @user.update(new_user_params)
        format.html { redirect_to root_path, notice: 'Данные обновлены!' }
        format.json { render :show, status: :ok, location: @user }
      else
        format.html { render :edit }
        format.json { render json: @user.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /users/1
  # DELETE /users/1.json
  def destroy
    Task.where(owner: @user.id).destroy_all
    @user.destroy
    respond_to do |format|
      format.html { redirect_to login_path, notice: 'Аккаунт удален' }
      format.json { head :no_content }
    end
  end

  private

  # Use callbacks to share common setup or constraints between actions.
  def set_user
    if !User.exists?(params[:id]) || User.find(params[:id]) != current_user
      redirect_to warning_path
    else
      @user = User.find(params[:id])
    end
  end

  # Only allow a list of trusted parameters through.
  def user_params
    params.require(:user).permit(:name, :email, :password, :password_confirmation)
  end

  def new_user_params
    params.require(:user).permit(:name, :email, :password, :password_confirmation,
:avatar)
  end
end
```

Листинг 2 – users_controller.rb

```
class SessionsController < ApplicationController
  before_action :logout_current, only: :new
  # skip_before_action :authenticate, only: :destroy

  def new; end

  def create
    user = User.find_by_email(params[:email])
    if user&.authenticate(params[:password])
      session[:user_id] = user.id
      redirect_to root_path, notice: 'Успешный вход!'
    else
      redirect_to login_path, alert: 'Неверная почта или пароль'
```

```
      # flash.now[:alert] = 'Email or password is invalid'
      # render 'new'
    end
  end

  def destroy
    session[:user_id] = nil
    redirect_to login_path, notice: 'Logged out!'
  end

  private

  def logout_current
    session[:user_id] = nil if session[:user_id]
  end
end
```

Листинг 3 – sessions_controller.rb

```
class TodoerController < ApplicationController
  before_action :set_task, only: %i[destroy move_to to_trash edit update]

  def index
    all_tasks false
  end

  def warning; end

  def add_task
    @task = Task.new
  end

  def create
    @task = Task.new(task_params)
    @task.owner = current_user.id
    @task.is_in_trash = false
    if @task.save
      redirect_to root_path, notice: 'TODO успешно создан!'
    else
      render 'todoer/add_task'
    end
  end

  def destroy
    @task.destroy
    respond_to do |format|
      format.html { redirect_to trash_can_path }
      format.js { render 'to_trash' }
    end
  end

  def clear_trash
    Task.where(is_in_trash: true).destroy_all
    respond_to do |format|
      format.html { redirect_to trash_can_path, notice: 'Корзина очищена!' }
      format.js
    end
  end

  def to_trash
    @task.update is_in_trash: !@task.is_in_trash
    respond_to do |format|
      format.html { redirect_to root_path }
      format.js
    end
```

```ruby
  end

  def edit; end

  def update
    # if Task.find(params[:id]).update(task params)
    if @task.update(task_params)
      redirect_to root_path, notice: 'TODO успешно обновлен!'
    else
      render 'todoer/edit'
    end
  end

  def trash_can
    all_tasks true
  end

  def move_to
    set_new_group
    @task.update group: @group
    respond_to do |format|
      format.html { redirect_to root_path }
      format.js
    end
  end

  private

  def all_tasks(status)
    @card_plans = Task.where({ owner: current_user.id, group: 'В планах',
is_in_trash: status })
    @card_in_work = Task.where({ owner: current_user.id, group: 'В работе',
is_in_trash: status })
    @card_done = Task.where({ owner: current_user.id, group: 'Выполнено',
is_in_trash: status })
  end

  def set_task
    @task = Task.find(params[:id])
  end

  def task_params
    params.require(:task).permit(:title, :content, :is_private, :group)
  end

  def update_params
    [:params[:title, :content, :is_private, :group]]
  end

  def set_new_group
    case @task.group
    when 'В планах'
      @group = 'В работе'
      @group_id = 'in_work'
    when 'В работе'
      @group = 'Выполнено'
      @group_id = 'done'
    else
      @group = @task.group
    end
  end
end
```

Листинг 4 – todoer_controller.rb

```ruby
class ColleaguesController < ApplicationController
  before_action :check_colleague, only: :colleague

  def index
    @colleagues = User.all.reject { |elem| elem.id == current_user.id }
  end

  def colleague
    all_tasks
  end

  def not_found; end

  def info; end

  private

  def check_colleague
    @colleague = User.find_by(id: params[:id])
    if @colleague.nil?
      redirect_to not_found_path
    elsif @colleague.id == current_user.id
      redirect_to root_path
    end
  end

  def all_tasks
    @card_plans = Task.where({ owner: @colleague.id, group: 'В планах',
is_in_trash: false, is_private: false })
    @card_in_work = Task.where({ owner: @colleague.id, group: 'В работе',
is_in_trash: false, is_private: false })
    @card_done = Task.where({ owner: @colleague.id, group: 'Выполнено',
is_in_trash: false, is_private: false })
  end
end
```

Листинг 5 – colleagues_controller.rb

Модели:

```ruby
class User < ApplicationRecord
  has_secure_password

  validates :password, length: { minimum: 8 }, if: -> { new_record? ||
!password.nil? }
  validates :name, presence: true
  validates :email, presence: true, uniqueness: true
  validates_format_of :email, with: /\A([^@\s]+)@((?:[-a-z0-9]+\.)+[a-z]{2,})\Z/i
  mount_uploader :avatar, AvatarUploader
end
```

Листинг 6 – user.rb

```ruby
class Task < ApplicationRecord
  validates :title, presence: true
  validates :group, inclusion: { in: ['В планах', 'В работе', 'Выполнено'] }
  validates :owner, numericality: true, presence: true
end
```

Листинг 7 – task.rb

Представления:

```
<%= form_with(model: @user, local: true, html: { class: "reg_form", autocomplete:
"off" }) do |form| %>
  <div class="error_div reg_error">
    <% if @user.errors.any? %>
      <div id="error_explanation">
        <h2>Не удалось зарегистрироваться</h2>
        <ul>
          <% @user.errors.full_messages.each do |message| %>
            <% case message %>
            <% when "Password is too short (minimum is 8 characters)" %>
              <% message = "Пароль не может быть короче 8 символов" %>
            <% when "Name can't be blank" %>
              <% message = "Введите имя" %>
            <% when "Email is invalid" %>
              <% message = "Введите корректный адрес электронной почты" %>
            <% when "Password confirmation doesn't match Password" %>
              <% message = "Пароли не совпадают" %>
            <% when "Email has already been taken" %>
              <% message = "Указанный адрес электронной почты уже зарегистрирован"
%>
            <% else %>
              <% message = nil %>
            <% end %>
            <% if message %>
              <li><%= message %></li>
            <% end %>
          <% end %>
        </ul>
      </div>
    <% end %>
  </div>
  <div class="signup_div get_info_div">
    <div class="field">
      <%#= form.label :name %>
      <%= form.text_field :name, placeholder: "Имя" %>
    </div>

    <div class="field">
      <%#= form.label :email %>
      <%= form.text_field :email, placeholder: "Электронная почта" %>
    </div>

    <div class="field">
      <%#= form.label :password %>
      <%= form.password_field :password, placeholder: "Пароль" %>
    </div>

    <div class="field">
      <%#= form.label :password_confirmation %>
      <%= form.password_field :password_confirmation, placeholder: "Повторите
пароль" %>
    </div>

    <div class="actions">
      <%= form.submit "Зарегистрироваться" %>
    </div>
  <% end %>

  <%= link_to 'У меня уже есть аккаунт', login_path %>
  </div>
```

Листинг 8 – user / new.html.erb

```
<%= render 'todoer/head' %>

<div class="card container" id="user_info">
  <table class="table table-borderless" id="info_table">
    <tr>
      <% if @user.avatar.file.nil? %>
        <td rowspan="4" id="avatar_store"><img id="avatar" src="<%=
image_path('none.jpg') %>"></td>
      <% else %>
        <td rowspan="4" id="avatar_store"><%= image_tag @user.avatar.url, id:
"avatar" %></td>
      <% end %>
      <td class="caption">Имя:</td>
    </tr>
    <tr>
      <td class="value"><%= @user.name %></td>
    </tr>
    <tr>
      <td class="caption">Электронная почта:</td>
    </tr>
    <tr>
      <td class="value"><%= @user.email %></td>
    </tr>
  </table>
</div>
```

Листинг 9 – users / show.html.erb

```
<%= render 'todoer/head' %>

<%= form_with(model: @user, local: true, html: { class: "reg_form", autocomplete:
"off" }) do |form| %>
  <div class="error_div edit_error">
    <% if @user.errors.any? %>
      <div id="error_explanation">
        <h2>Не удалось обновить данные</h2>
        <ul>
          <% @user.errors.full_messages.each do |message| %>
            <% case message %>
          <% when "Password can't be blank" %>
              <% message = "Введите пароль" %>
            <% when "Name can't be blank" %>
              <% message = "Введите имя" %>
            <% when "Email is invalid" %>
              <% message = "Введите корректный адрес электронной почты" %>
            <% when "Password confirmation doesn't match Password" %>
              <% message = "Пароли не совпадают" %>
            <% when "Email has already been taken" %>
              <% message = "Указанный адрес электронной почты уже зарегистрирован"
%>
            <% when "Password is too short (minimum is 8 characters)" %>
              <% message = "Пароль не может быть короче 8 символов" %>
            <% else %>
              <% message = nil %>
            <% end %>
            <% if message %>
              <li><%= message %></li>
            <% end %>
          <% end %>
        </ul>
      </div>
    <% end %>
  </div>
  <div class="get_info_div edit_user_info_div">
    <p id="notice"><%= notice %></p>
```

```
    <div class="field">
      <%#= form.label :name %>
      <%= form.text_field :name, placeholder: "Имя", class: "edit_input" %>
    </div>

    <div class="field">
      <%#= form.label :email %>
      <%= form.text_field :email, placeholder: "Электронная почта", class:
"edit_input" %>
    </div>

    <div class="field">
      <%#= form.label :password %>
      <%= form.password_field :password, placeholder: "Пароль", class:
"edit_input" %>
    </div>

    <div class="field">
      <%#= form.label :password_confirmation %>
      <%= form.password_field :password_confirmation, placeholder: "Повторите
пароль", class: "edit_input" %>
    </div>

    <div class="field">
      <%= form.file_field :avatar %>
    </div>

    <div class="actions">
      <%= form.submit "Обновить данные" %>
    </div>
<% end %>
<div id="space"></div>
<%= link_to 'Удалить аккаунт', user_path(current_user), method: :delete, data: {
confirm: 'Вы точно хотите удалить свой аккаунт?' } %>

</div>
```

Листинг 10 – users / edit.html.erb

```
<div class="login_div">
  <div class="msg">
    <p id="alert"><%= alert %></p>
    <p id="notice"><%= notice %></p>
  </div>
  <div class="form_div">
    <%= form_tag sessions_path, :autocomplete => "off" do |form| %>
      <div class="field">
        <%= text_field_tag :email, nil, placeholder: "Электронная почта" %>
      </div>

      <div class="field">
        <%= password_field_tag :password, nil, placeholder: "Пароль" %>
      </div>

      <div class="actions">
        <%= submit_tag "Войти" %>
      </div>
    <% end %>

    <%= link_to "Я тут первый раз", signup_path %>
  </div>
</div>
```

Листинг 11 – sessions / new.html.erb

```
<header id="header_nav">
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="<%= root_path %>">TODO</a>
    <a class="nav-link" href="<%= info_path %>">Что это?</a>
    <a class="nav-link" href="<%= colleagues_path %>">Все пользователи</a>
    <p class="navbar-text" id="notice"><%= notice %></p>
      <div class="nav-item dropdown">
        <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
          <%= current_user.name %>
        </a>
        <div class="dropdown-menu" aria-labelledby="navbarDropdown">
          <a class="dropdown-item" href="<%= user_path(current_user)
%>">Информация</a>
          <a class="dropdown-item" href="<%= edit_user_path(current_user)
%>">Редактировать</a>
          <a class="dropdown-item" href="<%= login_path %>">Выйти</a>
        </div>
      </div>
  </nav>
</header>
```

Листинг 11 – todoer / _head.html.erb

```
<%= render 'head' %>
<%= form_with(model: @task, local: true, html: { class: "reg_form", autocomplete:
"off" }) do |form| %>
  <div class="error_div edit_error">
    <% if @task.errors.any? %>
      <div id="error_explanation">
        <h2>Не удалось создать TODO</h2>
        <ul>
          <% @task.errors.full_messages.each do |message| %>
            <% case message %>
            <% when "Title can't be blank" %>
              <% message = "Укажите, что вы хотите добавить" %>
            <% else %>
              <% message = nil %>
            <% end %>
            <% if message %>
              <li><%= message %></li>
            <% end %>
          <% end %>
        </ul>
      </div>
    <% end %>
  </div>
  <div class="signup_div get_info_div">
      <div class="field">
        <%= form.text_field :title, placeholder: "Что надо сделать?" %>
      </div>
      <div class="field">
        <%= form.text_field :content, placeholder: "Немного конкретики" %>
      </div>
    <div class="row field">
      <div id="select_caption">Скрытый TODO?  </div>
      <%= form.check_box :is_private, :checked => true %>
    </div>
      <div class="field">
        <%= form.select :group, ["В планах", "В работе"] %>
      </div>
    <div class="actions">
      <%= form.submit "Создать TODO" %>
    </div>
```

```
  </div>
<% end %>
```

Листинг 12 – todoer / add_task.html.erb

```
<%= render 'head' %>
<div id="main_content">
  <div class="row buttons">
    <div class="col-6 btn_add">
      <a href="<%= tasks_path %>">
        <button class="btn add_del col-12">Добавить</button>
      </a>
    </div>
    <div class="col-6 btn_del">
      <a href="<%= trash_can_path %>">
        <button class="btn add_del col-12">Корзина</button>
      </a>
    </div>
  </div>
  <div class="row">
    <div class="col-4" id="in_plans">
      <div class="card"><p class="group_title">В планах</p></div>
      <% @card_plans.each do |card| %>
      <div class="card" id="card_<%= card.id %>">
        <div class="row card_row">
        <div class="card-body col-11">
          <h4 class="card-title"><%= card.title %></h4>
          <p class="card-text"><%= card.content %></p>
        </div>
        <div class="col-1 card_menu">
          <div class="card_menu_elem card_delete">
            <%= link_to image_tag(image_path('close.svg'), class: "img_mini"),
trash_path(params: { id: card.id }), remote: true %>
          </div>
          <div class="card_menu_elem move_to_right">
            <%= link_to image_tag(image_path('chevron-right.svg'), class:
"img_mini"), move_path(params: { id: card.id }), remote: true %>
          </div>
          <div class="card_menu_elem card_edit">
            <%= link_to image_tag(image_path('search.svg'), class: "img_mini"),
edit_path(params: { id: card.id }) %>
          </div>
        </div>
        </div>
      </div>
      <% end %>
    </div>
    <div class="col-4" id="in_work">
      <div class="card"><p class="group_title">В работе</p></div>
      <% @card_in_work.each do |card| %>
        <div class="card" id="card_<%= card.id %>">
          <div class="row card_row">
            <div class="card-body col-11">
              <h4 class="card-title"><%= card.title %></h4>
              <p class="card-text"><%= card.content %></p>
            </div>
            <div class="col-1 card_menu">
              <div class="card_menu_elem card_delete">
                <%= link_to image_tag(image_path('close.svg'), class: "img_mini"),
trash_path(params: { id: card.id }), remote: true %>
              </div>
              <div class="card_menu_elem move_to_right">
                <%= link_to image_tag(image_path('chevron-right.svg'), class:
"img_mini"), move_path(params: { id: card.id }), remote: true %>
              </div>
```

```
              <div class="card_menu_elem card_edit">
                <%= link_to image_tag(image_path('search.svg'), class:
"img_mini"), edit_path(params: { id: card.id }) %>
              </div>
            </div>
          </div>
        </div>
      <% end %>
    </div>
    <div class="col-4" id="done">
      <div class="card"><p class="group_title">Выполнено</p></div>
      <% @card_done.each do |card| %>
        <div class="card" id="card_<%= card.id %>">
          <div class="row card_row">
            <div class="card-body col-11">
              <h4 class="card-title"><%= card.title %></h4>
              <p class="card-text"><%= card.content %></p>
            </div>
            <div class="col-1 card_menu">
              <div class="card_menu_elem card_done_delete">
                <%= link_to image_tag(image_path('close.svg'), class: "img_mini"),
trash_path(params: { id: card.id }), remote: true %>
              </div>
              <div class="card_menu_elem card_done_edit">
                <%= link_to image_tag(image_path('search.svg'), class:
"img_mini"), edit_path(params: { id: card.id }) %>
              </div>
            </div>
          </div>
        </div>
      <% end %>
    </div>
  </div>
</div>
```

Листинг 13 – todoer / index.html.erb

```
<%= render 'head' %>
<%= form_with(model: @task, local: true, url: update_path(id: @task.id), html: {
class: "reg_form", autocomplete: "off" }) do |form| %>
  <div class="error_div edit_error">
    <% if @task.errors.any? %>
      <div id="error_explanation">
        <h2>Не удалось изменить TODO</h2>
        <ul>
          <% @task.errors.full_messages.each do |message| %>
            <% case message %>
            <% when "Title can't be blank" %>
              <% message = "Данное поле не может быть пустым" %>
            <% else %>
              <% message = nil %>
            <% end %>
            <% if message %>
              <li><%= message %></li>
            <% end %>
          <% end %>
        </ul>
      </div>
    <% end %>
  </div>
  <div class="signup_div get_info_div">
    <div class="field">
      <%= form.text_field :title, placeholder: "Что надо сделать?" %>
    </div>
    <div class="field">
```

```
      <%= form.text_field :content, placeholder: "Немного конкретики" %>
    </div>
    <div class="row field">
      <div id="select_caption">Скрытый TODO?  </div>
      <%= form.check_box :is_private %>
    </div>
    <div class="field">
      <%= form.select :group, ["В планах", "В работе", "Выполнено"] %>
    </div>
    <div class="actions">
      <%= form.submit "Подтвердить изменения" %>
    </div>
  </div>
<% end %>
```

Листинг 14 – todoer / edit.html.erb

```
<div id="warning_msg">
  <p id="warning">Вы пытаетесь получить доступ к данным другого пользователя.</br>
  Зачем? Вы что, злоумышленник?</p>
  <a href="<%= root_path %>">Конечно же нет!</a>
</div>
```

Листинг 15 – todoer / warining.html.erb

```
<%= render 'head' %>
<div id="main_content">
  <div class="card del_all">
    <%= button_to "Очистить корзину", del_all_path, remote: true, data: { confirm:
'Вы точно хотите очистить корзину?' }, class: "btn col-12" %>
  </div>
  <div class="row">
    <div class="col-4" id="in_plans">
      <div class="card"><p class="group_title">Было в планах</p></div>
      <% @card_plans.each do |card| %>
        <div class="card card-task" id="card_<%= card.id %>">
          <div class="card-body">
            <h4 class="card-title trash_can_title"><%= card.title %></h4>
          </div>
          <div class="row card_row">
            <div class="col-6 del_forever card_menu_elem trash_can_menu">
              <%= link_to image_tag(image_path('close.svg'), class: "img_mini"),
del_path(params: { id: card.id }), remote: true %>
            </div>
            <div class="col-6 trash_can_menu card_menu_elem">
              <%= link_to image_tag(image_path('reply.svg'), class: "img_mini"),
trash_path(params: { id: card.id }), remote: true %>
            </div>
          </div>
        </div>
      <% end %>
    </div>
    <div class="col-4" id="in_work">
      <div class="card"><p class="group_title">Было в работе</p></div>
      <% @card_in_work.each do |card| %>
        <div class="card card-task" id="card_<%= card.id %>">
          <div class="card-body">
            <h4 class="card-title trash_can_title"><%= card.title %></h4>
          </div>
          <div class="row card_row">
            <div class="col-6 del_forever card_menu_elem trash_can_menu">
              <%= link_to image_tag(image_path('close.svg'), class: "img_mini"),
del_path(params: { id: card.id }), remote: true %>
            </div>
```

```
            <div class="col-6 trash_can_menu card_menu_elem">
              <%= link_to image_tag(image_path('reply.svg'), class: "img_mini"),
trash_path(params: { id: card.id }), remote: true %>
            </div>
          </div>
        </div>
      <% end %>
    </div>
    <div class="col-4" id="done">
      <div class="card"><p class="group_title">Было выполнено</p></div>
      <% @card_done.each do |card| %>
        <div class="card card-task" id="card_<%= card.id %>">
          <div class="card-body">
            <h4 class="card-title trash_can_title"><%= card.title %></h4>
          </div>
          <div class="row card_row">
            <div class="col-6 del_forever card_menu_elem trash_can_menu">
              <%= link_to image_tag(image_path('close.svg'), class: "img_mini"),
del_path(params: { id: card.id }), remote: true %>
            </div>
            <div class="col-6 trash_can_menu card_menu_elem">
              <%= link_to image_tag(image_path('reply.svg'), class: "img_mini"),
trash_path(params: { id: card.id }), remote: true %>
            </div>
          </div>
        </div>
      <% end %>
    </div>
  </div>
</div>
```

Листинг 16 – todoer / trash_can.html.erb

```
elems_to_del = document.querySelectorAll(".card-task")
elems_to_del.forEach(elem => elem.remove())
```

Листинг 17 – todoer / clear_trash.js.erb

```
elem_to_move = document.querySelector("#card_<%= @task.id %>")
parent_elem = document.querySelector("#<%= @group_id %>")
parent_elem.append(elem_to_move)
if (parent_elem.id === "done") {
    move_menu = elem_to_move.querySelector(".move_to_right")
    move_menu.remove()
    del_menu = elem_to_move.querySelector(".card_delete")
    del_menu.classList.remove("card_delete")
    del_menu.classList.add("card_done_delete")
    edit_menu = elem_to_move.querySelector(".card_edit")
    edit_menu.classList.remove("card_edit")
    edit_menu.classList.add("card_done_edit")
}
```

Листинг 18 – todoer / move_to.js.erb

```
elem_to_del = document.querySelector("#card_<%= @task.id %>")
elem_to_del.remove()
```

Листинг 19 – todoer / to_trash.js.erb

Таблицы стилей:

```
.get_info_div {
  margin: auto;
  text-align: center;
}

.signup_div {
  width: 183px;
}

.edit_user_info_div {
  width: 200px;
}

.edit_input {
  width: 200px;
}

div#error_explanation {
  margin: auto;
  text-align: left;
  border-radius: 3px;
}

.error_div {
  margin-top: 20px;
}

.reg_error {
  height: 200px;
}

.edit_error {
  height: 150px;
}

.field_with_errors {
  border-radius: 3px;
  padding: 0 4px 0 0;
}

#warning_msg {
  width: 650px;
  margin: 20% auto auto;
  text-align: center;
}

#info_table {
  margin: 0;
}

#warning {
  font-weight: bold;
  font-size: 20px;
}

#user info {
  width: 500px;
  margin: 8% auto auto;
}

.caption {
  width: 50%;
  font-size: 20px;
```

```
  line-height: 57px;
}

.value {
  text-align: left;
  font-size: 20px;
  font-weight: bold;
  line-height: 57px;
}

#avatar_store {
  text-align: right;
  vertical-align: middle;
}

#avatar {
  max-width: 235px;
  max-height: 300px;
}

#space {
  height: 20px;
}
```

Листинг 20 – users.scss

```
p#alert
{
  color: #cc0000;
}

div.login_div
{
  width: 20%;
  //height: 130px;
  margin: 20% auto auto;
  text-align: center;
}

div.msg
{
  height: 20px;
}
```

Листинг 21 – sessions.scss

```
#main_content {
  margin: 20px;
  min-height: auto;
}

.row.buttons {
  padding-bottom: 10px;
  font-size: 18px;
}

.card_row {
  margin: 0;
}

.card_menu {
  border-left: 1px solid lightgrey;
  padding: 0;
}
```

```
.card_delete {
  height: 32.65px;
}

.trash_can_title {
  margin: 0;
}

.trash_can_menu {
  height: 25px;
  text-align: center;
  border-top: 1px solid lightgrey;
  padding: 0;
}

.del_forever {
  border-right: 1px solid lightgrey;
}

.del_all {
  border: 2px solid lightgrey;
}

.card_done_delete {
  height: 50%;
  border-bottom: 1px solid lightgrey;
}

.card_done_edit {
  height: 50%;
}

.img_mini {
  width: 60%;
  height: 60%;
}

.move_to_right {
  border-top: 1px solid lightgrey;
  border-bottom: 1px solid lightgrey;
  height: calc(100% - 65.3px);
}

.card_edit {
  position: absolute;
  bottom: 0;
  width: 100%;
  height: 32.65px;
}

div.card_menu_elem a {
  display: flex;
  align-items: center;
  justify-content: center;
  height: 100%;
  width: 100%;
}

div.card_menu_elem button {
  width: 100%;
  height: 100%;
  padding: 0;
}

#select_caption {
```

```
  width: 160px;
}

.group_title {
  text-align: center;
  margin: 0;
  font-size: 18px;
  height: 25px;
  line-height: 25px;
}

.btn_add {
  padding-left: 0;
  padding-right: 10px;
}

#task_group {
  width: 100%;
}

.add_del {
  border: 2px solid lightgrey;
  border-radius: 5px;
}

.btn_del {
  padding-left: 10px;
  padding-right: 0;
}

.navbar {
  border-radius: 6px;
}

.buttons {
  padding-left: 20px;
  padding-right: 20px;
}

.card-text {
  text-indent: 20px;
  text-align: justify ;
}

.card {
  margin-top: 10px;
  margin-left: 5px;
  margin-right: 5px;
}

body {
  margin: 20px;
}

a:hover {
  background-color: transparent;
  color: black;
}

.session_info {
  margin-left: auto;
  margin-right: 10px;
}

div.dropdown {
  margin-left: 0;
```

```
  margin-right: 0;
}

div.dropdown-menu {
  right: 0;
  left: auto;
  font-size: 13px;
}

.navbar-text {
  margin-bottom: 0;
  margin-right: 0;
  margin-left: auto;
}
```

Листинг 22 – todoer.scss

```
#list_of_coll {
  margin: 10px 20px 20px;
}

.coll_title {
  text-align: left;
}

#mini_avatar {
  display: table-cell;
  max-width: 70px;
  padding: 5px;
}

.mini_avatar_store {
  text-align: right;
}

.user_link {
  display: inline-block;
  height: 100%;
  width: 100%;
}

.colleague_info {
  text-align: center;
}

#colleague {
  text-align: center;
  padding-top: 5px;
  padding-bottom: 5px;
  font-size: 18px;
}
```

Листинг 23 – colleagues.scss

Пути:

```
Rails.application.routes.draw do
  get 'colleagues', to: 'colleagues#index'
  resources :sessions, only: %i[new create destroy]
  resources :users
  root 'todoer#index'
  get 'signup', to: 'users#new', as: 'signup'
  get 'login', to: 'sessions#new', as: 'login'
  get 'logout', to: 'sessions#destroy', as: 'logout'
  get 'warning', to: 'todoer#warning'
  post 'tasks', to: 'todoer#create'
  patch 'tasks', to: 'todoer#update'
  get 'tasks', to: 'todoer#add_task'
  get 'del', to: 'todoer#destroy'
  get 'move', to: 'todoer#move_to'
  get 'trash', to: 'todoer#to_trash'
  get 'trash_can', to: 'todoer#trash_can'
  get 'del', to: 'todoer#destroy'
  post 'del_all', to: 'todoer#clear_trash'
  get 'edit', to: 'todoer#edit'
  patch 'update', to: 'todoer#update'
  get 'update', to: 'todoer#edit'
  get 'colleague', to: 'colleagues#colleague'
  get 'info', to: 'colleagues#info'
  get 'not_found', to: 'colleagues#not_found'
  # For details on the DSL available within this file, see
https://guides.rubyonrails.org/routing.html
end
```

Листинг 24 – routes.rb

Внешний вид пользовательского интерфейса:

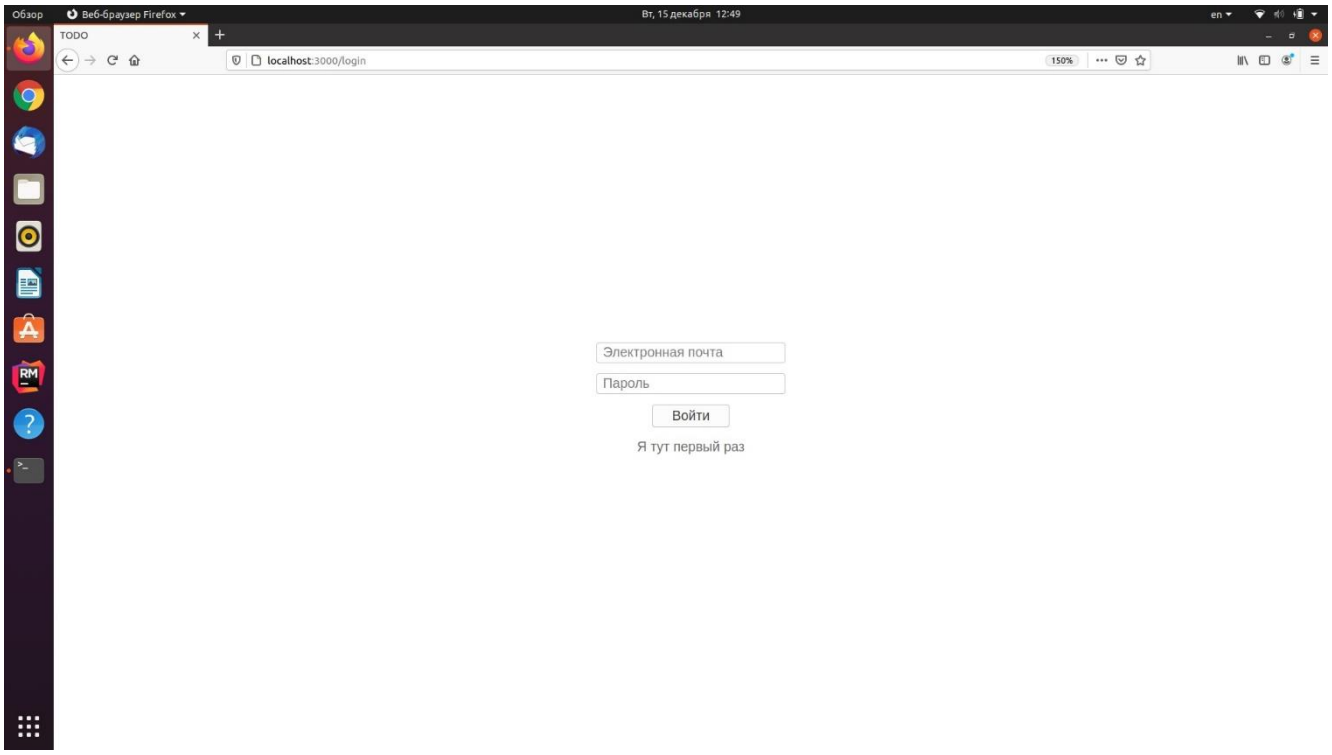


Рисунок 1 - Страница авторизации

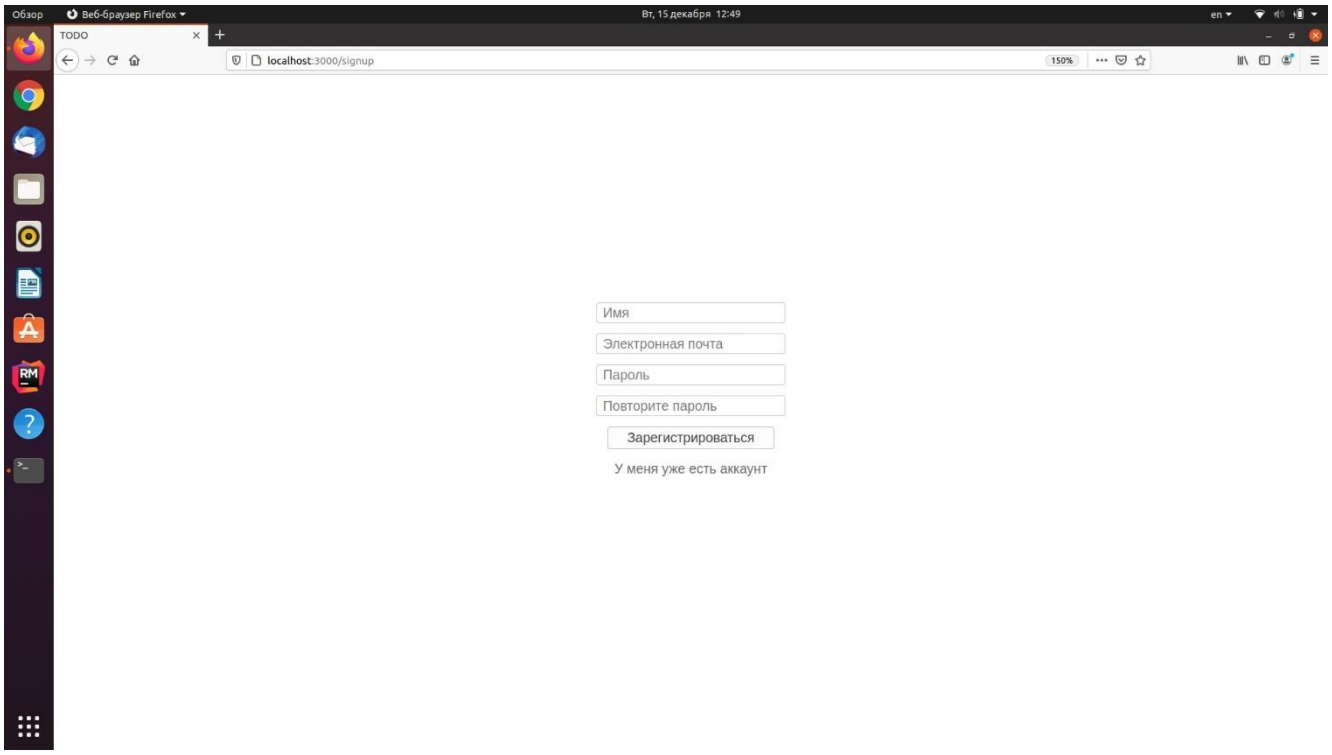


Рисунок 2 - Страница создания нового пользователя

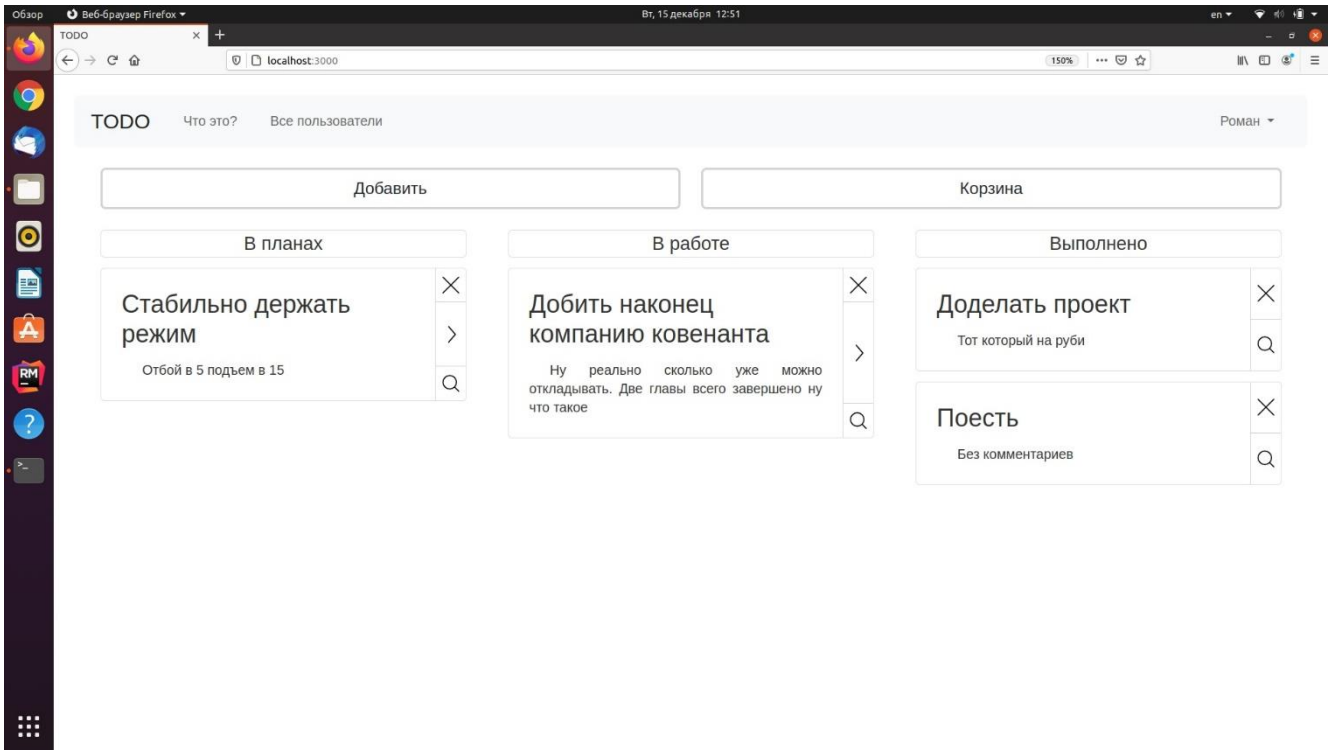


Рисунок 3 - Главная страница

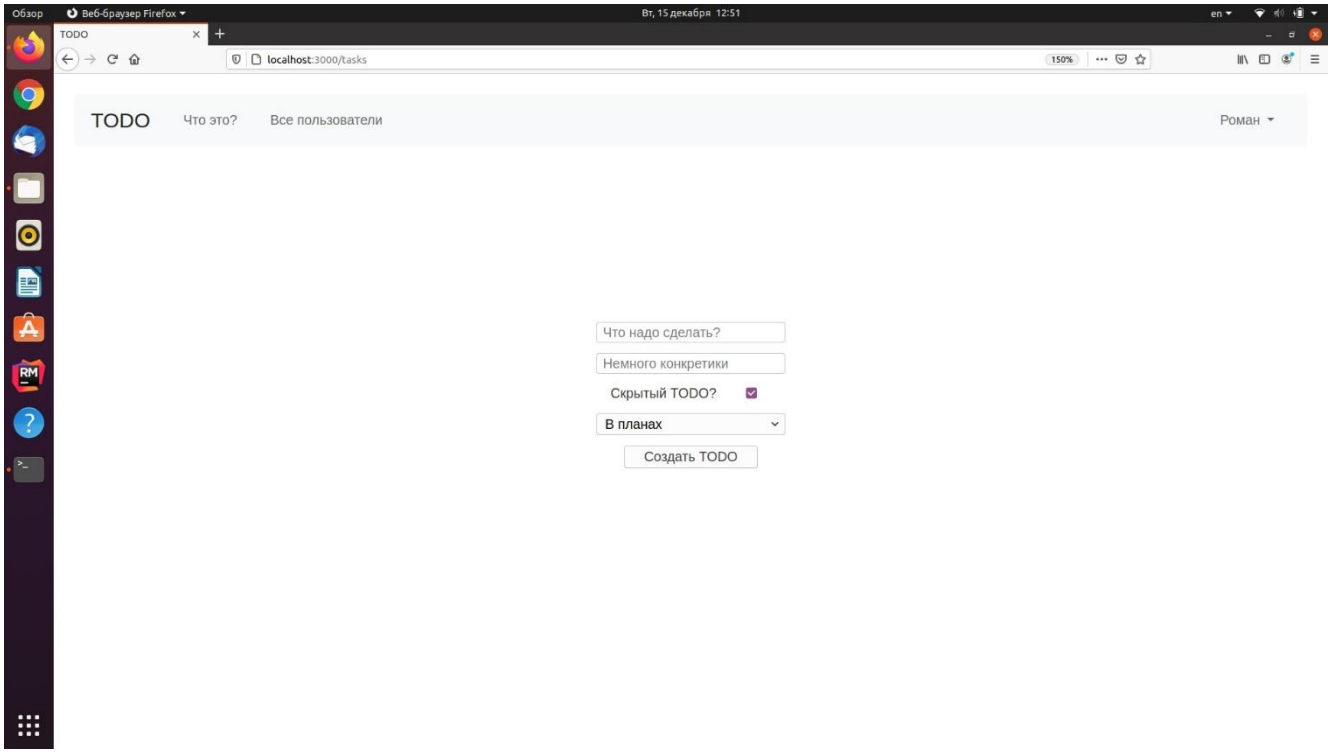


Рисунок 4 - Страница создания TODO

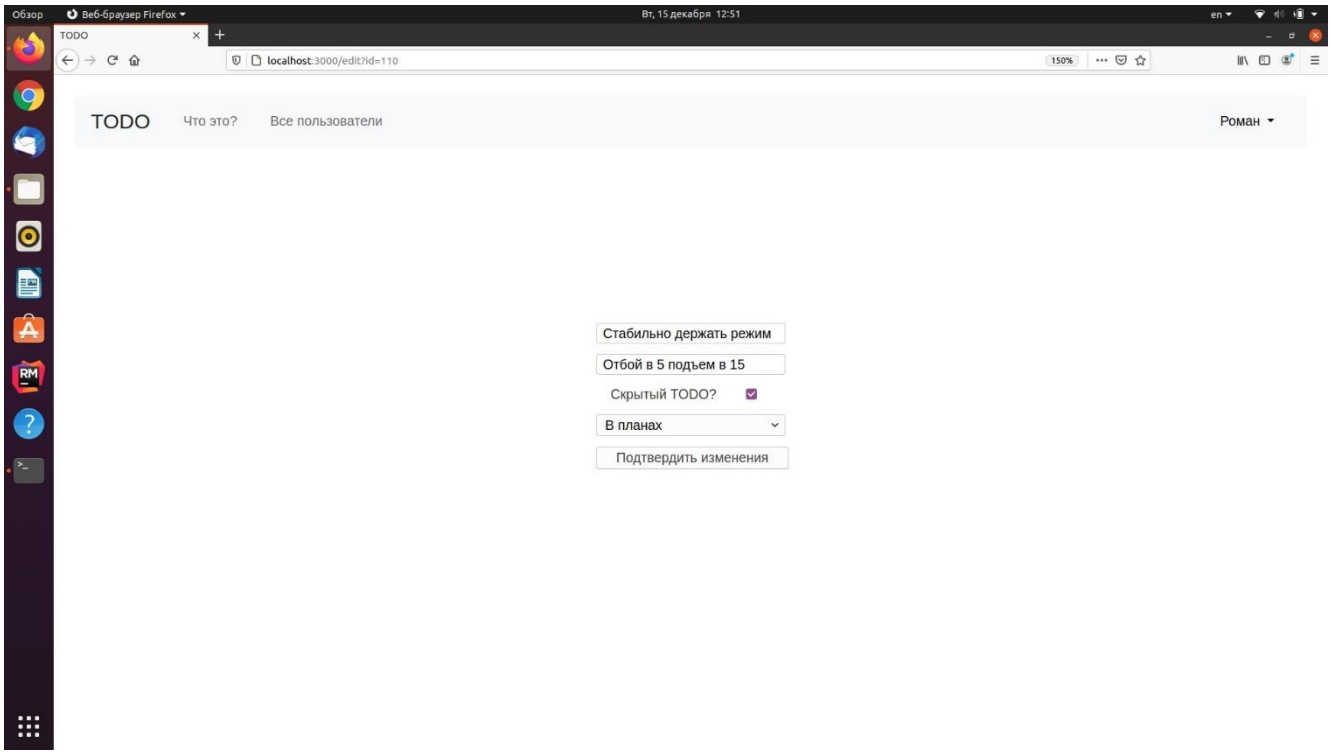


Рисунок 5 - Страница редактирования TODO

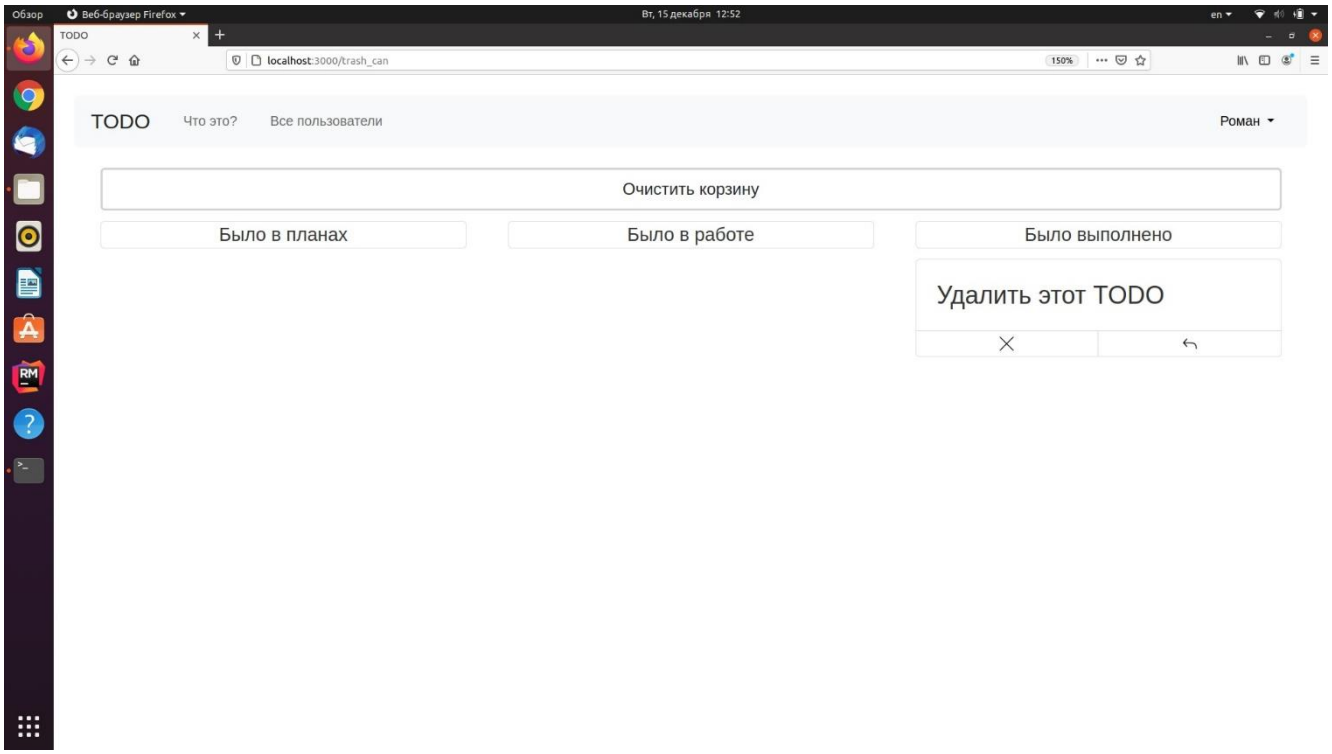


Рисунок 6 - Корзина

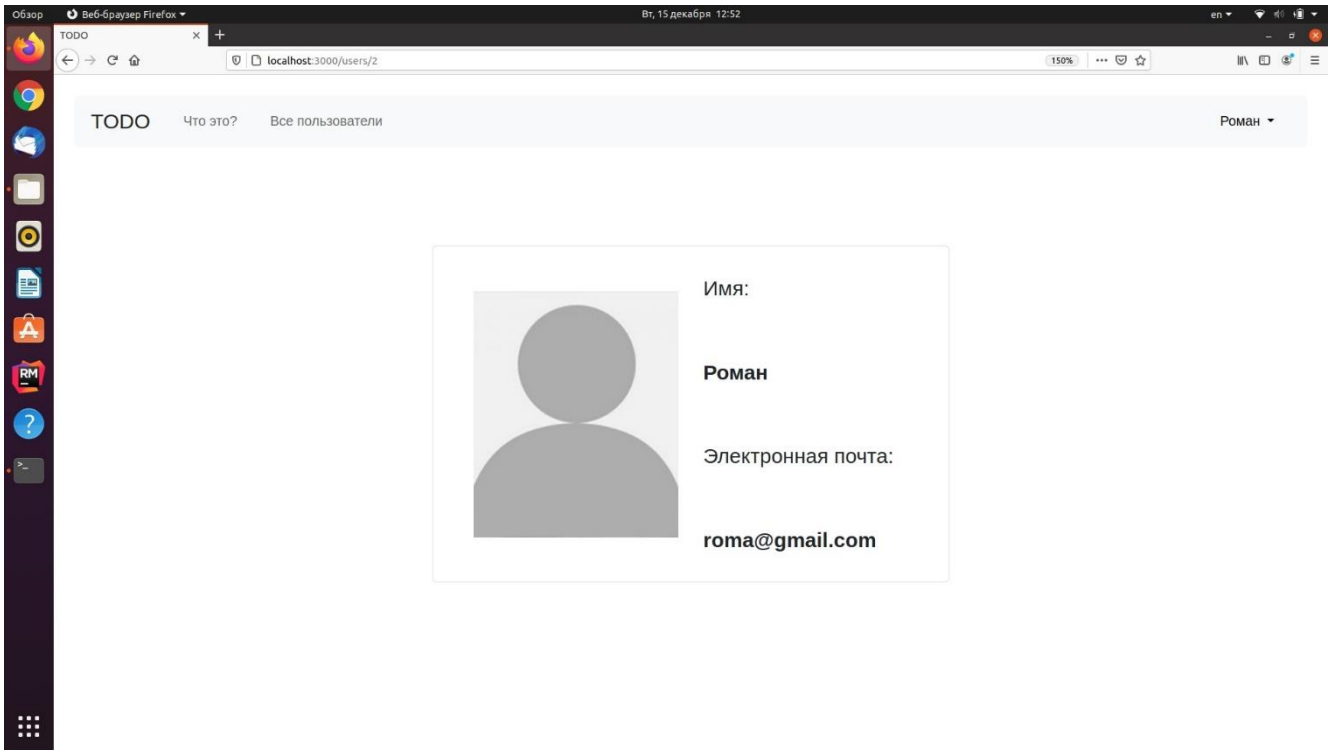


Рисунок 7 - Информация о пользователе

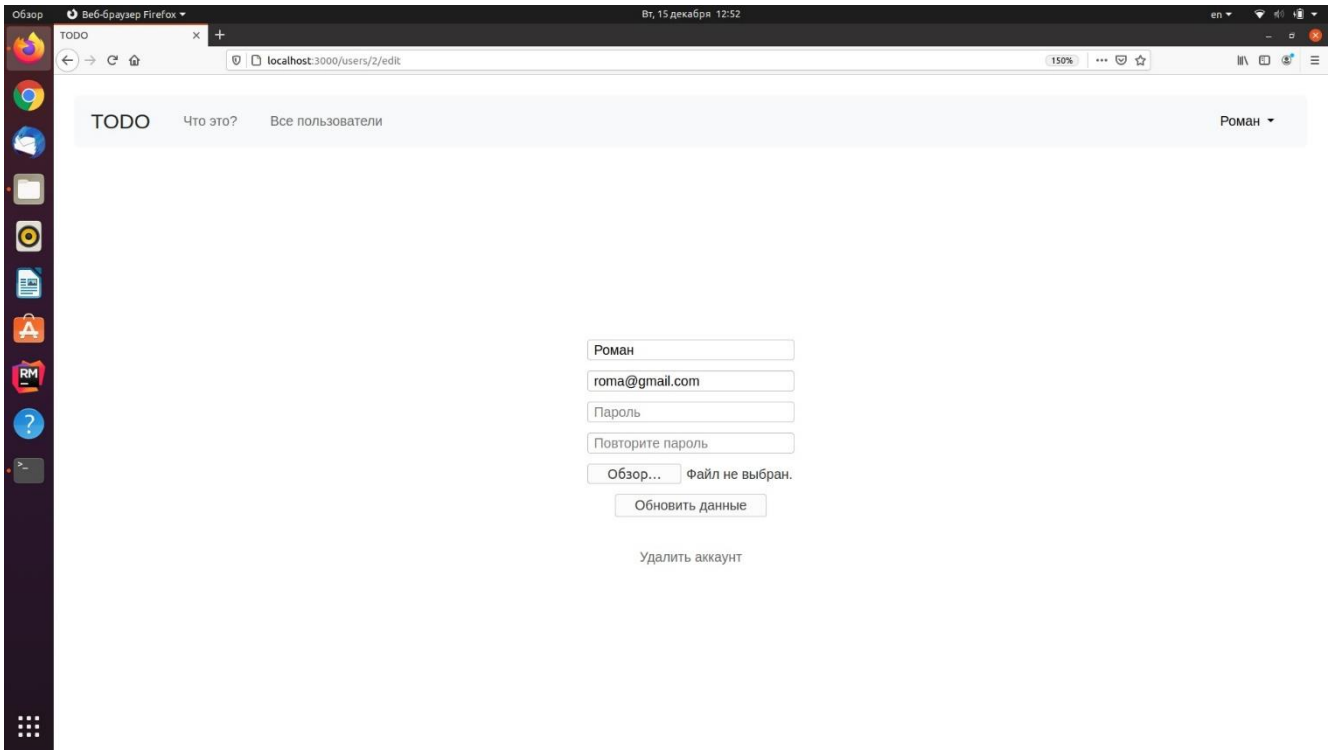


Рисунок 8 - Страница редактирования пользователя

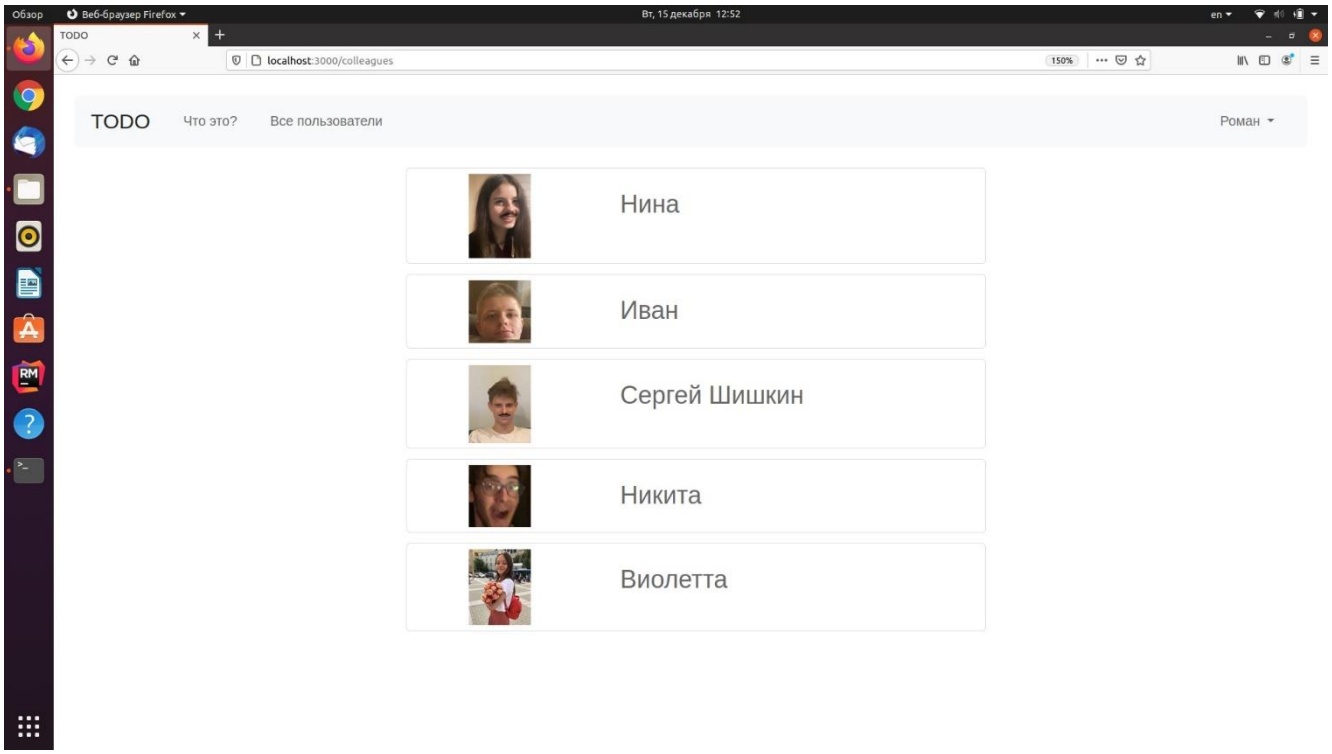


Рисунок 9 - Список всех пользователей

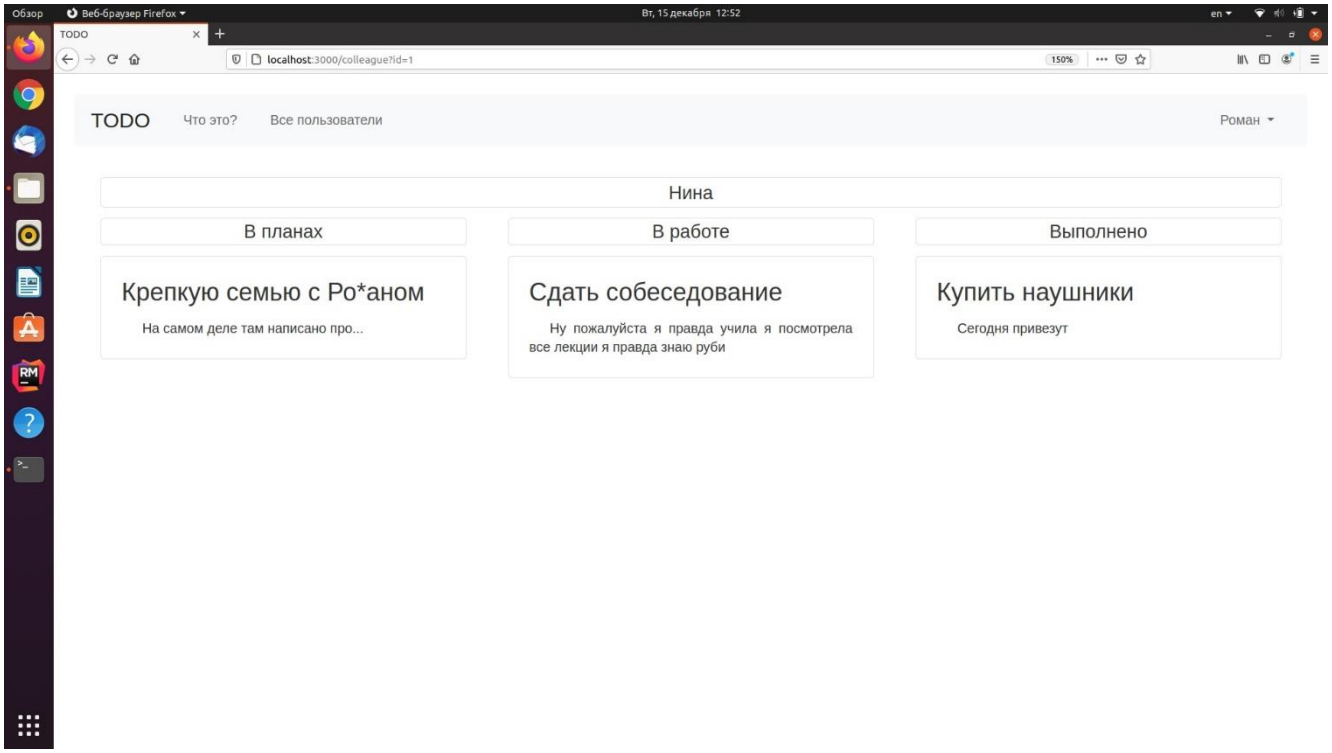


Рисунок 10 - Доступные для просмотра TODO конкретного пользователя

Тестирование:

Тесты моделей:

```
class UserTest < ActiveSupport::TestCase
  # проверка корректного сохранения записи
  test 'correct save' do
    instance = User.new(name: 'Oda Nobunaga', email: 'oda@gmail.com', password:
'12345678')
    assert instance.save
  end

  # проверка корректного доступа к полям записи
  test 'correct get' do
    instance = User.new(name: 'Isoroku Yamamoto', email: 'isoroku@mail.ru',
password: '12345678')
    assert_equal instance.name, 'Isoroku Yamamoto'
  end

  # проверка корректного удаления записи
  test 'correct destroy' do
    instance = User.new(name: 'Heitaro Kimura', email: 'heitaro@bmstu.ru',
password: '12345678')
    assert instance.destroy
  end

  # проверка на выдачу ошибки при попытке создания записи с неуникальным полем
email
  test 'error not uniq email' do
    f_instance = User.new(name: 'Masatane Kanda', email: 'masa@yandex.ru',
password: '12345678')
    f_instance.save
    s_instance = User.new(name: 'Masaki Honda', email: 'masa@yandex.ru', password:
'12345678')
    s_instance.validate
    assert_equal s_instance.errors[:email], ['has already been taken']
  end

  # проверка на выдачу ошибки при попытке создания записи с невалидным полем email
  test 'error invalid email' do
    instance = User.new(name: 'Toru Okabe', email: 'toru.com', password:
'12345678')
    instance.validate
    assert_equal instance.errors[:email], ['is invalid']
  end

  # проверка на выдачу ошибки при попытке создания записи со слишком коротким
полем password
  test 'error short password' do
    instance = User.new(name: 'Tateo Katō', email: 'kato@gmail.com', password:
'123')
    instance.validate
    assert_equal instance.errors[:password], ['is too short (minimum is 8
characters)']
  end

  # проверка на выдачу ошибки при попытке создания записи с пустым полем name
  test 'error name blank' do
    instance = User.new(email: 'Kuroda@gmail.com', password: '12345678')
    instance.validate
    assert_equal instance.errors[:name], ['can\'t be blank']
  end

  # проверка на выдачу ошибки при попытке создания записи со пустым полем email
```

```
  test 'error email blank' do
    instance = User.new(name: 'Samejima Shigeo', password: '12345678')
    instance.validate
    assert_equal instance.errors[:email], ['can\'t be blank', 'is invalid']
  end

  # проверка на выдачу ошибки при попытке создания записи с пустым полем password
  test 'error password blank' do
    instance = User.new(name: 'Shunji Sato', email: 'shunji@rambler.ru')
    instance.validate
    assert_equal instance.errors[:password], ['can\'t be blank', 'is too short
(minimum is 8 characters)']
  end
end
```

Листинг 25 - user_test.rb

```
class TaskTest < ActiveSupport::TestCase
  # проверка корректного сохранения записи
  test 'correct save' do
    instance = Task.new(owner: 1, title: 'Do something', is_private: true,
is_in_trash: false, group: 'В планах')
    assert instance.save
  end

  # проверка корректного доступа к полям записи
  test 'correct get' do
    instance = Task.new(owner: 1, title: 'Do something', is_private: true,
is_in_trash: false, group: 'В работе')
    assert_equal instance.title, 'Do something'
  end

  # проверка корректного удаления записи
  test 'correct destroy' do
    instance = Task.new(owner: 1, title: 'Do something', is_private: true,
is_in_trash: false, group: 'В планах')
    assert instance.destroy
  end

  # проверка на выдачу ошибки при попытке создания записи с нечисловым полем owner
  test 'error owner not num' do
    instance = Task.new(owner: 'Someone', title: 'Do it', is_private: true,
is_in_trash: false, group: 'В планах')
    instance.validate
    assert_equal instance.errors[:owner], ['is not a number']
  end

  # проверка на выдачу ошибки при попытке создания записи со значением поля group,
не входящим в набор
  test 'error incorrect group' do
    instance = Task.new(owner: 1, title: 'Do something', is_private: true,
is_in_trash: false, group: 'В корзине')
    instance.validate
    assert_equal instance.errors[:group], ['is not included in the list']
  end

  # проверка на выдачу ошибки при попытке создания записи с пустым полем title
  test 'error title blank' do
    instance = Task.new(owner: 1, is_private: true, is_in_trash: false, group: 'В
планах')
    instance.validate
    assert_equal instance.errors[:title], ['can\'t be blank']
  end

  # проверка на выдачу ошибки при попытке создания записи с пустым полем owner
```

```
  test 'error owner blank' do
    instance = Task.new(title: 'Do something', is_private: true, is_in_trash:
false, group: 'В планах')
    instance.validate
    assert_equal instance.errors[:owner], ['is not a number', 'can\'t be blank']
  end
end
```

Листинг 26 - task_test.rb

Тесты контроллеров:

```
class UsersControllerTest < ActionDispatch::IntegrationTest
  # проверка корректного захода на страницу создания нового пользователя
  test "should get new" do
    get new_user_url
    assert_response :success
  end

  # проверка корректного создания нового пользователя и перенаправления на
страницу авторизации
  test "should create user" do
    assert_difference('User.count') do
      post users_url, params: { user: { email: 'nikita@mail.ru', name: 'Nikita',
password: '12345678', password_confirmation: '12345678' } }
    end

    assert_redirected_to login_path
  end

  # проверка корректного перенаправления на страницу авторизации при попытке зайти
на страницу пользователя
  test "redirect to login from user" do
    get '/users/1'
    assert_response :redirect
    assert_redirected_to login_path
  end

  # проверка корректного перенаправления на страницу авторизации при попытке зайти
на страницу редактирования пользователя
  test "redirect to login from edit user" do
    get '/users/1/edit'
    assert_response :redirect
    assert_redirected_to login_path
  end
end
```

Листинг 27 - user_controller_test.rb

```
class SessionsControllerTest < ActionDispatch::IntegrationTest
  # проверка корректного захода на страницу авторизации
  test 'get login' do
    get login_path
    assert_response :success
  end
end
```

Листинг 28 - sessions_controller_test.rb

```
class TodoerControllerTest < ActionDispatch::IntegrationTest
  # проверка корректного перенаправления на страницу авторизации при попытке зайти
на главную страницу
```

```
  test 'redirect to login' do
    get root_path
    assert_response :redirect
    assert_redirected_to login_path
  end
end
```

Листинг 29 - todoer_controller_test.rb

```
class ColleaguesControllerTest < ActionDispatch::IntegrationTest
  # проверка корректного перенаправления на страницу авторизации при попытке
посмотреть список всех пользователей
  test 'redirect to login' do
    get colleagues_path
    assert_response :redirect
    assert_redirected_to login_path
  end
end
```

Листинг 30 - colleagues_controller_test.rb

Интеграционные тесты:

```
class SessionTest < ActionDispatch::IntegrationTest
  # проверка корректной регистрации и авторизации
  test 'should reg and log' do
    post users_path, params: { user: { name: 'Nina', email: 'nina@yandex.ru',
password: '12345678', password_confirmation: '12345678' } }
    assert_response :redirect
    assert_redirected_to login_path
    post sessions_path, params: { email: 'nina@yandex.ru', password: '12345678' }
    assert_response :redirect
    assert_redirected_to root_path
  end

  # проверка корректного перенаправления на страницу авторизации
  test 'should redirect to login' do
    get root_path
    assert_response :redirect
    assert_redirected_to login_path
  end

  # проверка корректного выхода из аккаунта
  test 'should logout' do
    post users_path, params: { user: { name: 'Nina', email: 'nina@yandex.ru',
password: '12345678', password_confirmation: '12345678' } }
    post sessions_path, params: { email: 'nina@yandex.ru', password: '12345678' }
    get logout_path
    assert_response :redirect
    assert_redirected_to login_path
  end
end
```

Листинг 31 - session_test.rb

```
class TodoerBaseTest < ActionDispatch::IntegrationTest
  def setup
    post users_path, params: { user: { name: 'Nina', email: 'nina@yandex.ru',
password: '12345678', password_confirmation: '12345678' } }
    post sessions_path, params: { email: 'nina@yandex.ru', password: '12345678' }
    @id = User.find_by(email: 'nina@yandex.ru').id
  end
```

```
  # проверка корректного перехода на страницу с информацией о пользователе
  test 'correct get info' do
    get "/users/#{@id}"
    assert_response :success
  end

  # проверка корректного перехода на страницу редактирования пользователя
  test 'correct get info edit' do
    get "/users/#{@id}/edit"
    assert_response :success
  end

  # проверка корректного редактирования пользователя
  test 'correct user edit' do
    patch "/users/#{@id}", params: { user: { name: 'NeNina', email:
'nina@yandex.ru' } }
    assert_response :redirect
    assert_redirected_to root_path
    assert_equal User.find(@id).name, 'NeNina'
  end

  # проверка корректного удаления пользователя
  test 'correct delete user' do
    delete "/users/#{@id}"
    assert_response :redirect
    assert_redirected_to login_path
    assert_nil User.find_by(id: @id)
  end

  # проверка корректного перехода на страницу корзины
  test 'correct get trash can' do
    get trash_can_path
    assert_response :success
  end

  # проверка корректного выхода из аккаунта
  test 'correct get logout' do
    get logout_path
    assert_response :redirect
    assert_redirected_to login_path
  end

  # проверка корректного перенаправления на страницу с предупреждением при попытке
получить информацию о другом пользователе
  test 'redirect from user to warning' do
    get "/users/#{@id - 1}"
    assert_response :redirect
    assert_redirected_to warning_path
  end

  # проверка корректного перенаправления на страницу с предупреждением при попытке
редактировать другого пользователя
  test 'redirect from edit user to warning' do
    get "/users/#{@id - 1}/edit"
    assert_response :redirect
    assert_redirected_to warning_path
  end

  # проверка корректного перехода на страницу редактирования TODO
  test 'correct get edit task' do
    get tasks_path
    assert_response :success
  end

  # проверка корректного создания нового TODO
```

```
  test 'correct create new task' do
    post tasks_path, params: { task: { title: "Do something", is_private: "1",
group: "В планах" } }
    assert_response :redirect
    assert_redirected_to root_path
  end
end
```

Листинг 32 - todoer_base_test.rb

```
class TodoerCardTest < ActionDispatch::IntegrationTest
  def setup
    post users_path, params: { user: { name: 'Nina', email: 'nina@yandex.ru',
password: '12345678', password_confirmation: '12345678' } }
    post sessions_path, params: { email: 'nina@yandex.ru', password: '12345678' }
    post tasks_path, params: { task: { title: "Do something", is_private: "1",
group: "В планах" } }
    @id = Task.find_by(title: "Do something").id
  end

  # проверка корректного перемещения TODO из одной группы в другую
  test 'correct move to' do
    get move_path, params: { id: @id }
    assert_response :redirect
    assert_redirected_to root_path
    assert_equal Task.find(@id).group, "В работе"
  end

  # проверка корректного перехода на страницу редактирования TODO
  test 'correct get edit card' do
    get edit_path, params: { id: @id }
    assert_response :success
  end

  # проверка корректного изменения TODO
  test 'correct edit card' do
    patch update_path, params: { task: { title: "Do nothing", is_private: "0",
group: "В планах" }, id: @id }
    assert_response :redirect
    assert_redirected_to root_path
    assert_equal Task.find(@id).title, "Do nothing"
  end

  # проверка корректного перемещения TODO в корзину
  test 'correct send to trash' do
    get trash_path, params: { id: @id }
    assert_response :redirect
    assert_redirected_to root_path
    assert_equal Task.find(@id).is_in_trash, true
  end
end
```

Листинг 33 - todoer_card_test.rb

```
class TodoerTrashTest < ActionDispatch::IntegrationTest
  def setup
    post users_path, params: { user: { name: 'Nina', email: 'nina@yandex.ru',
password: '12345678', password_confirmation: '12345678' } }
    post sessions_path, params: { email: 'nina@yandex.ru', password: '12345678' }
    post tasks_path, params: { task: { title: "Do something", is_private: "1",
group: "В планах" } }
    @id = Task.find_by(title: "Do something").id
    get trash_path, params: { id: @id }
  end
```

```
  # проверка корректного возвращения на главную страницу
  test 'correct return to root' do
    get root_path
    assert_response :success
  end

  # проверка корректного окончательного удаления TODO
  test 'correct destroy' do
    get del_path, params: { id: @id }
    assert_response :redirect
    assert_redirected_to trash_can_path
    assert_nil Task.find_by(id: @id)
  end

  # проверка корректного возвращения TODO из корзины
  test 'correct return from trash' do
    get trash_path, params: { id: @id }
    assert_response :redirect
    assert_redirected_to root_path
    assert_equal Task.find(@id).is_in_trash, false
  end

  # проверка корректной очистки корзины
  test 'correct clear trash' do
    post del_all_path
    assert_response :redirect
    assert_redirected_to trash_can_path
    assert_nil Task.find_by(id: @id)
  end
end
```

Листинг 34 - todoer_trash_test.rb

```
class TodoerColleaguesTest < ActionDispatch::IntegrationTest
  def setup
    post users_path, params: { user: { name: 'Nina', email: 'nina@yandex.ru',
password: '12345678', password_confirmation: '12345678' } }
    post users_path, params: { user: { name: 'Roma', email: 'roma@gmail.com',
password: '12345678', password_confirmation: '12345678' } }
    post sessions_path, params: { email: 'nina@yandex.ru', password: '12345678' }
  end

  # проверка корректноого перехода на страницу редактирования информации о проекте
  test 'correct get project info' do
    get info_path
    assert_response :success
  end

  # проверка корректноого перехода на страницу со списком всех пользователей
  test 'correct get all' do
    get colleagues_path do
      assert_response :success
    end
  end

  # проверка корректноого перехода на страницу с доступными для простмотра TODO
конкретного пользователя
  test 'correct get one' do
    get colleague_path, params: { id: User.find_by(name: 'Roma').id }
    assert_response :success
  end

  # проверка корректноого перенаправления на страницу ошибки при попытке найти
несуществующего пользователя
```

```
  test 'redirect trying to see none' do
    get colleague_path, params: { id: 2 }
    assert_response :redirect
    assert_redirected_to not_found_path
  end

  # проверка корректноого перенаправления на главную страницу при попытке найти
себя в списке пользователей
  test 'redirect to root trying to see self' do
    get colleague_path, params: { id: User.find_by(name: 'Nina').id }
    assert_response :redirect
    assert_redirected_to root_path
  end
end
```

Листинг 35 - colleagues_test.rb

Системные тесты:

```
class SessionsTest < ApplicationSystemTestCase
  # проверка получения сообщения об ошибке при попытке авторизоваться с
некорректными данными
  test 'error login' do
    visit login_path
    fill_in 'email', with: 'Incorrect email'
    fill_in 'password', with: '123'
    click_on 'commit'
    assert_text 'Неверная почта или пароль'
  end

  # проверка корректной авторизации
  test 'correct login' do
    User.new(name: 'Ivan', email: 'ivan@gmail.com', password: '12345678').save
    visit login_path
    fill_in 'email', with: 'ivan@gmail.com'
    fill_in 'password', with: '12345678'
    click_on 'commit'
    assert_text 'Успешный вход!'
  end

  # проверка получения сообщения об ошибке при попытке создания пользователя с
некорректными данными
  test 'error registration' do
    visit signup_path
    fill_in 'user[name]', with: 'Ivan'
    fill_in 'user[email]', with: 'Incorrect email'
    fill_in 'user[password]', with: '12345678'
    fill_in 'user[password_confirmation]', with: '12345678'
    click_on 'commit'
    assert_text 'Не удалось зарегистрироваться'
  end

  # проверка корректной регистрации
  test 'correct registration' do
    visit signup_path
    fill_in 'user[name]', with: 'Sergei'
    fill_in 'user[email]', with: 'Sergei@gmail.com'
    fill_in 'user[password]', with: '12345678'
    fill_in 'user[password_confirmation]', with: '12345678'
    click_on 'commit'
    assert_text 'Успешная регистрация!'
  end

  # проверка корректного перехода со страницы авторизации на страницу регистрации
```

```
  test 'correct way to registration' do
    visit login_path
    find(:xpath, "//a[@href='/signup']").click
    assert_text 'У меня уже есть аккаунт'
  end

  # проверка корректного перехода со страницы регистрации на страницу авторизации
  test 'correct way to login' do
    visit signup_path
    find(:xpath, "//a[@href='/login']").click
    assert_text 'Я тут первый раз'
  end
end
```

Листинг 36 - sessions_test.rb

```
class TodoerTest < ApplicationSystemTestCase
  def setup
    User.new(name: 'Ivan', email: 'ivan@gmail.com', password: '12345678').save
    visit login_path
    fill_in 'email', with: 'ivan@gmail.com'
    fill_in 'password', with: '12345678'
    click_on 'commit'
  end

  # проверка корректного перехода на страницу с информацией о проекте
  test 'visit project info' do
    find(:xpath, "//a[@href='/info']").click
    assert_text 'Я все понял'
  end

  # проверка корректного перехода на страницу со списком всех пользователей
  test 'visit colleagues' do
    find(:xpath, "//a[@href='/colleagues']").click
    assert_text 'User_from_fixtures'
    find(:xpath, "//a[@href='/colleague?id=1']").click
    assert_text 'В планах'
  end

  # проверка корректного выхода из аккаунта
  test 'correct logout' do
    find(:xpath, "//a[@href='#']").click
    find(:xpath, "//a[@href='/login']").click
    assert_text 'Я тут первый раз'
  end

  # проверка корректного добавления нового TODO
  test 'correct add card' do
    find(:xpath, "//a[@href='/tasks']").click
    fill_in 'task[title]', with: 'Do something'
    click_on 'commit'
    assert_text 'TODO успешно создан!'
  end

  # проверка корректного редактирования пользователя
  test 'correct edit user' do
    find(:xpath, "//a[@href='#']").click
    find(:xpath, "//a[@href='/users/#{User.find_by(name:
'Ivan').id}/edit']").click
    fill_in 'user[name]', with: 'Vanya'
    click_on 'commit'
    assert_text 'Данные обновлены'
  end

  # проверка корректного перехода на страницу с информацией о пользователе
```

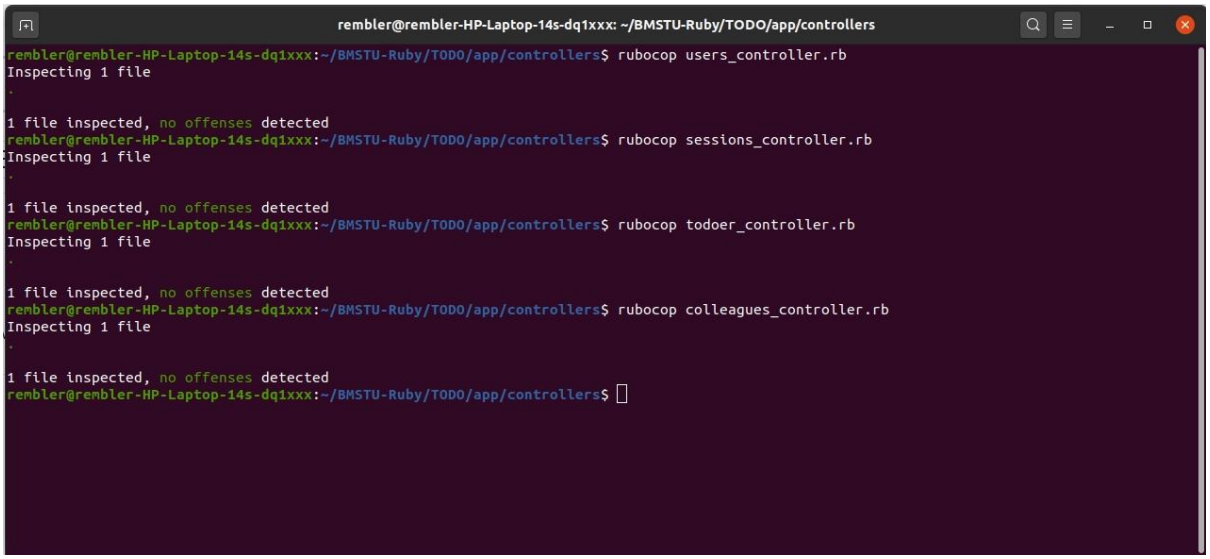
```
  test 'visit user info' do
    find(:xpath, "//a[@href='#']").click
    find(:xpath, "//a[@href='/users/#{User.find_by(name: 'Ivan').id}']").click
    assert_text 'Имя'
  end

  # проверка корректного перехода на страницу корзины
  test 'visit trash can' do
      find(:xpath, "//a[@href='/trash_can']").click
      assert_equal find(:xpath, "//input").value, "Очистить корзину"
    end
end
```

Листинг 37 - todoer_test.rb

```
class TodoerCardTest < ApplicationSystemTestCase
  def setup
    User.new(name: 'Ivan', email: 'ivan@gmail.com', password: '12345678').save
    visit login_path
    fill_in 'email', with: 'ivan@gmail.com'
    fill_in 'password', with: '12345678'
    click_on 'commit'
    find(:xpath, "//a[@href='/tasks']").click
    fill_in 'task[title]', with: 'Do something'
    click_on 'commit'
    @id = Task.find_by(title: 'Do something').id
  end

  # проверка корректного перемещения TODO из одной группы в другую
  test 'correct move card' do
    find(:xpath, "//a[@href='/move?id=#{@id}']").click
    flag = false
    flag = true if find(:xpath, "//div[@id='in_work']/div[@id='card_#{@id}']")
    assert flag
  end

  # проверка корректного перемещения TODO в корзину
  test 'correct del card' do
    find(:xpath, "//a[@href='/trash?id=#{@id}']").click
    find(:xpath, "//a[@href='/trash_can']").click
    assert_text 'Do something'
  end

  # проверка корректного редактирования TODO
  test 'correct card edit' do
    find(:xpath, "//a[@href='/edit?id=#{@id}']").click
    fill_in 'task[title]', with: 'Do nothing'
    click_on 'commit'
    assert_text 'TODO успешно обновлен!'
  end
end
```

Листинг 38 - todoer_card_test.rb

Результат проверки контроллеров анализатором Rubocop:



Вывод:

В данной работе было продемонстрировано применение на практике – при создании собственного веб-приложения, знаний, полученных в ходе изучения курса «Языки интернет-программирования».