ПРИЛОЖЕНИЕ А
(обязательное)

Исходный текст управляющей программы

Файл control_script.py:

```
import requests
from picamera import PiCamera
from time import sleep
SERVER_URL = 'http://192.168.43.138:3000'
USER_ID = 1
TITLE = 'Home'
PHOTO_FILENAME = 'home_photo'
DELAY = 300
IS_TEMP = True
PREVIEW_DELAY = 4
MODE = 'cycle' # or 'single'
class MyCamera:
    def __init__(self, is_temp, filename, preview_delay):
        self.is_temp = is_temp
        self.counter = 0
        self.filename = filename
        self.preview_delay = preview_delay
        self.camera = PiCamera()
    def capture(self):
        self.camera.start_preview()
        sleep(self.preview_delay)
        self.camera.capture(self.filename + str(self.counter) + '.jpg')
        self.camera.stop_preview()
        if not self.is_temp:
            self.counter += 1
        return self.filename
class PostRequest:
    def __init__(self, url, user_id, title):
        self.url = url
        self.params = (('image[user_id]', str(user_id)),('image[title]', str(title)))
    def make_request(self, filename):
        self.files = ('image[photo]', open(filename, 'rb'))
        requests.post(self.url, files = self.files, data = self.params)
        self.files['image[photo]'].close()
class Client:
    TEMP_FILENAME = 'temp_photo'
    def __init__(self, server_url, photo_filename=TEMP_FILENAME, delay=0, preview_delay=2,
is_temp=True, user_id=1, title='Title'):
        self.server_url = server_url
        self.photo_filename = photo_filename
        self.delay = delay
        self.preview_delay = preview_delay
        self.is_temp = is_temp
        self.user_id = user_id
        self.title = title
        self.my_camera = MyCamera(self.is_temp, self.photo_filename, self.preview_delay)
        self.post_request = PostRequest(self.server_url, self.user_id, self.title)
    def single_capture_and_post(self):
        filename = self.my_camera.capture()
        self.post_request.make_request(filename)
    def cycle_capture_and_post(self):
        while True:
            filename = self.my_camera.capture()
            self.post_request.make_request(filename)
            sleep(self.delay)
if __name__ == '__main__':
    client = Client(SERVER_URL, PHOTO_FILENAME, DELAY, PREVIEW_DELAY, IS_TEMP, USER_ID, TITLE)
    if MODE == 'single':
        client.single_capture_and_post()
    elif  MODE == 'cycle':
        client.cycle_capture_and_post()
```

ПРИЛОЖЕНИЕ Б
(обязательное)

Исходный текст веб-приложения

Файл confirmation_controller.rb:

```
 frozen_string_literal: true

class Users::ConfirmationsController < Devise::ConfirmationsController
   # GET /resource/confirmation/new
   def new
     super
   end

   # POST /resource/confirmation
   def create
     super
   end

   # GET /resource/confirmation?confirmation_token=abcdef
   def show
     super
   end

   protected

  # The path used after resending confirmation instructions.
   def after_resending_confirmation_instructions_path_for(resource_name)
     super(resource_name)
   end

  # The path used after confirmation.
   def after_confirmation_path_for(resource_name, resource)
     super(resource_name, resource)
   end
end
```

Файл omniauth_callback_controller.rb:

```
 frozen_string_literal: true

class Users::OmniauthCallbacksController < Devise::OmniauthCallbacksController
  # You should configure your model like this:
   devise :omniauthable, omniauth_providers: [:twitter]

  # You should also create an action method in this controller like this:
   def twitter
   end

  # GET|POST /resource/auth/twitter
   def passthru
     super
   end

  # GET|POST /users/auth/twitter/callback
   def failure
     super
   end

   protected

  # The path used when OmniAuth fails
   def after_omniauth_failure_path_for(scope)
     super(scope)
   end
end
```

Файл images_controller.rb:

```
class ImagesController < ApplicationController
  before_action :set_image, only: [:show, :edit, :update, :destroy]
  skip_before_action :authenticate_user!
  skip_before_action :verify_authenticity_token

  # GET /images
  # GET /images.json
  def index
    @images = Image.all
  end

  # GET /images/1
  # GET /images/1.json
  def show
  end

  # GET /images/new
  def new
    @image = Image.new
  end

  # GET /images/1/edit
  def edit
  end

  # POST /images
  # POST /images.json
  def create
    @image = Image.new(image_params)

    respond_to do |format|
      if @image.save
        format.html { redirect_to @image, notice: 'Image was successfully created.' }
        format.json { render :show, status: :created, location: @image }
      else
        format.html { render :new }
        format.json { render json: @image.errors, status: :unprocessable_entity }
      end
    end
  end

  # PATCH/PUT /images/1
  # PATCH/PUT /images/1.json
  def update
    respond_to do |format|
      if @image.update(image_params)
        format.html { redirect_to @image, notice: 'Image was successfully updated.' }
        format.json { render :show, status: :ok, location: @image }
      else
        format.html { render :edit }
        format.json { render json: @image.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /images/1
  # DELETE /images/1.json
  def destroy
    @image.destroy
    respond_to do |format|
      format.html { redirect_to images_url, notice: 'Image was successfully destroyed.' }
      format.json { head :no_content }
    end
  end

  private
    # Use callbacks to share common setup or constraints between actions.
    def set_image
      @image = Image.find(params[:id])
    end
    # Never trust parameters from the scary internet, only allow the white list through.
    def image_params
      params.require(:image).permit(:user_id, :title, :photo)
    end
end
```

Файл passwords_controller.rb:

```
 frozen_string_literal: true

class Users::PasswordsController < Devise::PasswordsController
  # GET /resource/password/new
   def new
     super
   end

  # POST /resource/password
   def create
     super
   end

  # GET /resource/password/edit?reset_password_token=abcdef
  # def edit
     super
   end

  # PUT /resource/password
   def update
     super
   end

   protected

  # def after_resetting_password_path_for(resource)
     super(resource)
   end

  # The path used after sending reset password instructions
   def after_sending_reset_password_instructions_path_for(resource_name)
     super(resource_name)
   end
end
```

Файл registrations_controller.rb:

```
 frozen_string_literal: true

class Users::RegistrationsController < Devise::RegistrationsController
   before_action :configure_sign_up_params, only: [:create]
   before_action :configure_account_update_params, only: [:update]

  # GET /resource/sign_up
   def new
     super
   end

  # POST /resource
   def create
     super
   end

  # GET /resource/edit
   def edit
     super
   end

  # PUT /resource
```

```
   def update
     super
   end

  # DELETE /resource
   def destroy
     super
   end

  # GET /resource/cancel
  # Forces the session data which is usually expired after sign
  # in to be expired now. This is useful if the user wants to
  # cancel oauth signing in/up in the middle of the process,
  # removing all OAuth session data.
   def cancel
     super
   end

   protected

  # If you have extra params to permit, append them to the sanitizer.
   def configure_sign_up_params
     devise_parameter_sanitizer.permit(:sign_up, keys: [:attribute])
   end

  # If you have extra params to permit, append them to the sanitizer.
   def configure_account_update_params
     devise_parameter_sanitizer.permit(:account_update, keys: [:attribute])
   end

  # The path used after sign up.
   def after_sign_up_path_for(resource)
     super(resource)
   end

  # The path used after sign up for inactive accounts.
   def after_inactive_sign_up_path_for(resource)
     super(resource)
   end
end
```

Файл registrations_controller.rb:

```
  frozen_string_literal: true

class Users::UnlocksController < Devise::UnlocksController
  # GET /resource/unlock/new
   def new
     super
   end

  # POST /resource/unlock
   def create
     super
   end

  # GET /resource/unlock?unlock_token=abcdef
   def show
     super
   end

   protected

   def after_sending_unlock_instructions_path_for(resource)
     super(resource)
   end

   def after_unlock_path_for(resource)
     super(resource)
   end
end
```

ПРИЛОЖЕНИЕ В
(обязательное)

Спецификация проекта

ПРИЛОЖЕНИЕ Г
(обязательное)

Ведомость документов