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