ОТЧЁТ  
о прохождении стажировки

Центр цифровых решений и искусственного интеллекта

(наименование института)

Веб-приложение на Flask

(наименование ДПП ПП, по которой проходит обучение слушатель)

**Иванов Иван Иванович**

(фамилия, имя, отчество обучающегося)

ООО «Айти Бастион»

(наименование места прохождения)

22-28 июня 2023 года

(сроки стажировки)

Руководитель стажировки

от Академии: Боловцов Сергей Владимирович,

заведующий Лабораторией

интеллектуальной аналитики ЦЦРиИИ

**Москва  
2023**

**ВВЕДЕНИЕ**

1. Краткая характеристика места стажировки, функций организации (подразделения)

**ООО «Айти Бастион»** располагается по адресу: 119501, г. Москва, ул. Староволынская, д. 15, к. 1. Она создана в 2014 г. с целью консультативной деятельности и работы в области компьютерных технологий.

**ЦЕЛИ И ЗАДАЧИ**

1. Цели стажировки (основные задачи стажировки) на основе выбранной темы

Основной **целью** стажировки является получение практического опыта в профессиональной деятельности, освоение и закрепление полученных знаний и навыков работы с базами данных и применения языка программирования Python (в т. ч. скрипты) для решения профессиональных задач под контролем более опытных специалистов.

**Задачи** стажировки:

* овладеть языками Python и SQL на базовом уровне;
* научиться кодировать на языках программирования (с помощью Python) в соответствии с трудовым заданием;
* научиться с помощью Python проводить базовую аналитику многомерных данных;
* самостоятельно написать код (проект с этапами решения кейса) и продемонстрировать выполненное решение.

**ОСНОВНАЯ ЧАСТЬ**

Мини-приложение “Image Processing” предназначено для того, чтобы помочь вам быстро конвертировать изображения и изменять их размер в соответствии с вашими предпочтениями.

Данная тема для разработки выбрана мною, так как она довольна проста как в реализации, так и в понимании.

Для реализации требуется лишь изучить некоторые аспекты новых библиотек и фреймворка, а также грамотно написать структуру и алгоритм работы приложения, так как оно должно выполнять свои задачи без ошибок.

И в качестве понимания работа проста, так как использует основы. Она интуитивно понятна и не вызовет трудностей в понимании, например, у других людей, которые захотят склонировать данный проект с github и использовать его в учебных целях.

Это приложение особенно полезно для студентов разных дисциплин, включая дизайн, менеджмент и другие специальности. Если нужно настроить формат изображения или изменить его размер в соответствии с конкретными требованиями, это приложение упростит ваши задачи по обработке изображений.

Для реализации я выбрал небольшой функционал, так как задачей является ознакомление, но не полное погружение.

Если рассмотреть этапы разработки, то получим следующее:

* Определение требования и функциональные возможности приложения, включая возможности преобразования изображений и изменения размера;
* Заведение желаемых форматов изображений, которые должно поддерживать приложение - jpg, png и gif;
* Разработка дизайна приложения;
* Написание функций обработки изображений;

Приведу пример запуска и работы приложения. Для запуска достаточно одного из IDE для разработки на Python. Я выбрал среду PyCharm. Так как она является бесплатной и удобной в использовании.

Итак, при открытие проекта, нужно запустить сервер, нажав ПКМ по app.py, заранее открыв его, и зайти в терминал. Далее будет написано, что сервер запущен по адресу: http://127.0.0.1:5000.

Перейдя на него, маршрутизатор примет запрос через декоратор @app.route(‘/’), открыв главную страницу, на которой будет основная информация о проекте (рис. 1)

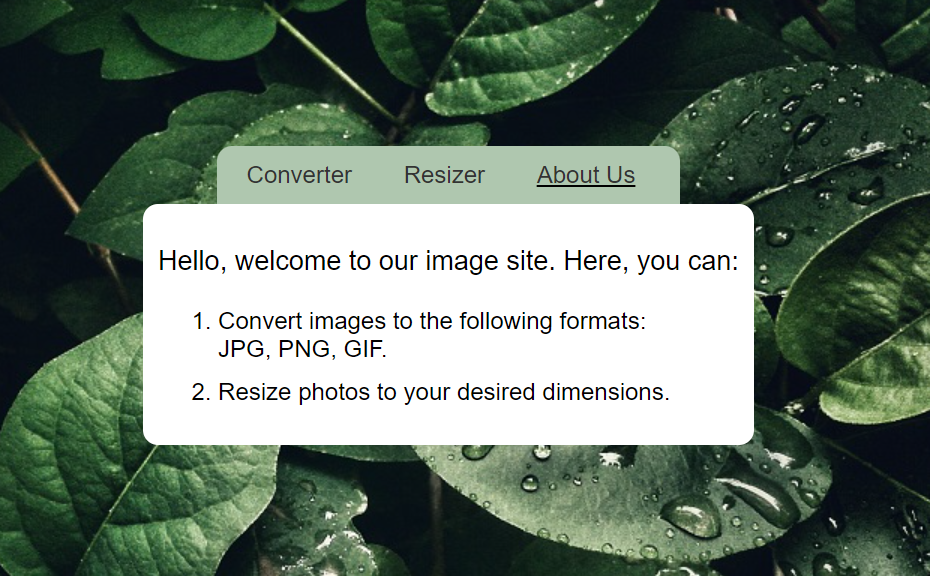


Рис. 1 – главная страница

Для динамичности шаблона, я передавал на страницу параметр, который указывал, что именно эта страница открыта, благодаря чего в меню навигации текущий раздел подчеркнут линией.

После этого у пользователя есть выбор в виде двух функционалов. Если пойти в "Converter”, то увидим окно для изменения формата изображения (рис. 2). При этом сработает обработка запроса @app.route(‘/convert’). В окне пользователь может подгрузить картинку и указать новый формат, который ему нужен. После нажатия “Convert” миниатюра преобразованной картинки появится внизу, и можно будет ее скачать по нажатию кнопки.

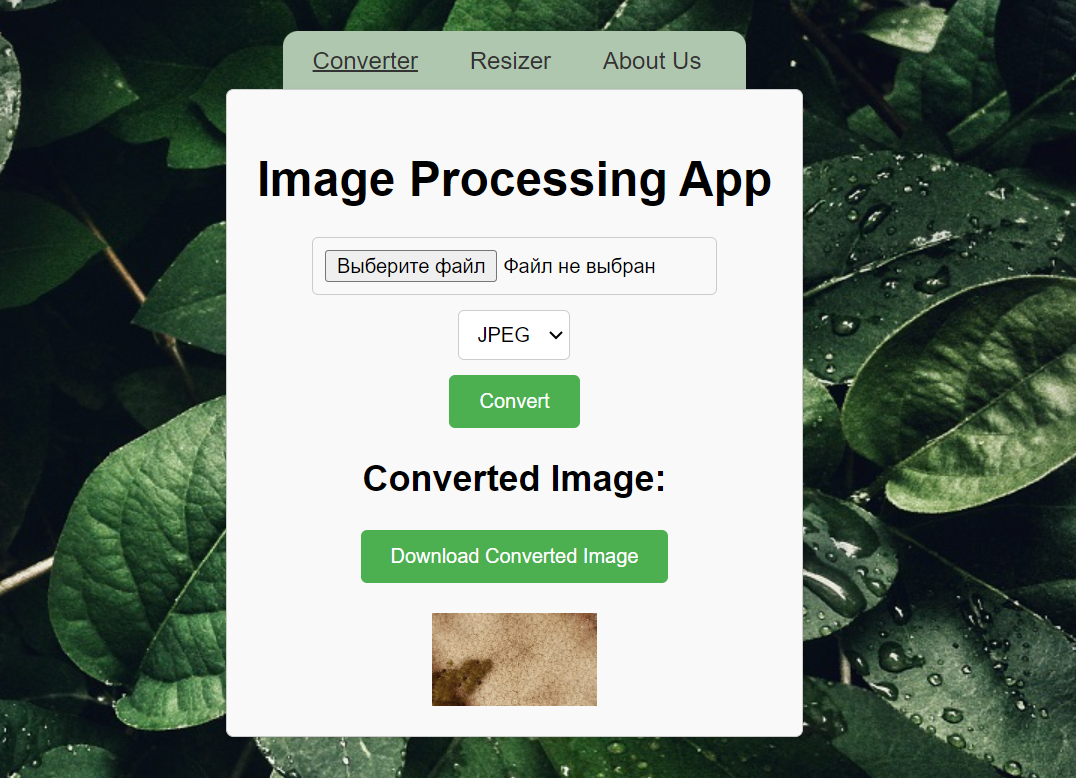


Рис. 2 – окно конвертации

И, наконец, перейдя через меню в “Resizer”, можно изменить размер изображения, указав одну из сторон, размер которой нужно указать в окне “pixels”. При этом маршрутизатор обработает следующий запрос: @app.route(‘/resize). При нажатии “Resize’ изображение получится с одной стороной, длина которой равна указанной, и второй стороной, которая изменилась пропорционально первой. Новую картинку можно также скачать (рис. 3).

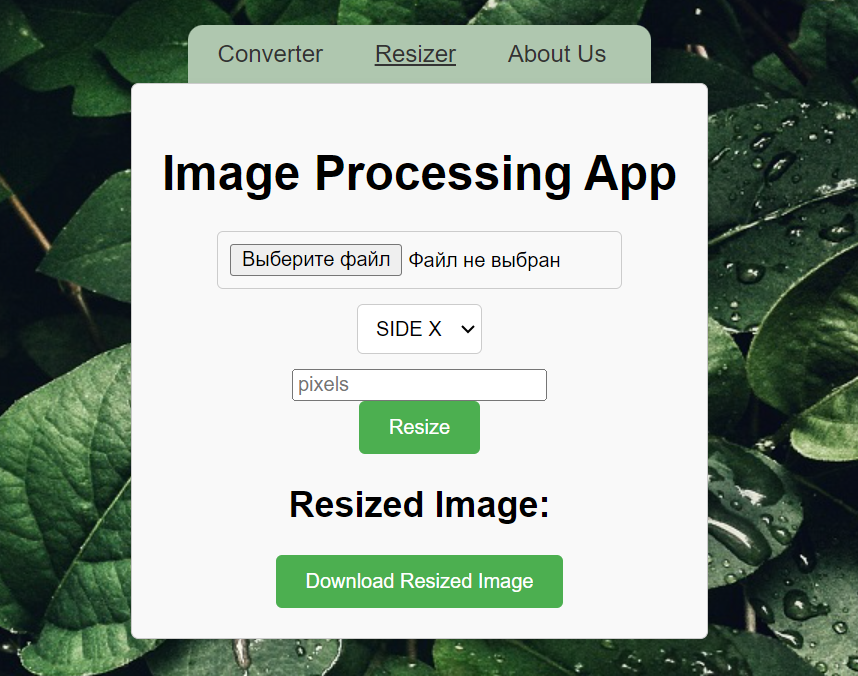


Рис. 3 – окно изменения размера.

Приложение обработки изображений призвано упростить процесс преобразования изображений, предлагая при этом гибкость и возможности настройки. Независимо от того, работаете ли вы с фотографиями, иллюстрациями или другим визуальным контентом, это приложение поможет вам без особых усилий добиться желаемого формата или размера.

Используя современные методы обработки изображений, можно эффективно преобразовывать изображения и изменять их размер для различных целей, в том числе:

* Архивирование и хранение: преобразование изображений в различные форматы, подходящие для долгосрочного архивирования или хранения;
* Совместимость и доступность: адаптация форматов изображений, для обеспечения совместимости с различными платформами, устройствами или программными приложениями;
* Веб-оптимизация: изменение размера изображений для оптимизации времени загрузки и повышения производительности веб-сайта;
* Настройка контента: настраивание размеров изображений в соответствии с конкретными требованиями, например изображения профиля в социальных сетях или иллюстрации сообщений в блогах;
* Визуальная согласованность: поддержка согласованности форматов изображений на протяжении всего проекта или презентации;

Приложение обработки изображений представляет собой простое в использовании решение для учащихся, позволяющее им эффективно обрабатывать задачи преобразования изображений и изменения размера. Упрощая эти процессы, учащиеся могут сосредоточиться на своих основных целях, улучшая визуальный контент, чтобы он соответствовал профессиональным стандартам.

**ЗАКЛЮЧЕНИЕ**

Приложение для обработки изображений — это веб-приложение, которое позволяет пользователям выполнять задачи по преобразованию изображений.

Построенное приложение предоставляет две основные функции: преобразование изображения и изменение размера изображения. Данные функции реализованы в классах, объекты которых вызываются для обработки.

Работа является заделом для дальнейшей разработки, здесь может быть и кадрирование, изменение цветовых характеристик. Таких приложений много, но для получения начального опыта оно очень полезно.

Мною были изучены библиотеки Flask и PIL. Благодаря ним приложение быстро пишется без написаний сложных функций.

Также было разобрано/закреплено:

* OOP в python для написания классов обработки изображений;
* Маршрутизация запросов для обработки приходящих запросов;
* Работа с данными сессии для запоминания пути изображения и его формата;
* HTML, CSS для написания шаблонов страниц;
* Jinja2 для шаблонизации и передачи параметров на страницу;
* Кодирование данных для передачи файла на страницу для отображения;

В заключение, приложение для обработки изображений демонстрирует реализацию функций преобразования изображений и изменения размера в веб-среде. Он служит практичным инструментом для пользователей, которым требуются быстрые и удобные операции по обработке изображений.

Цели и задачи, поставленные в рамках данной стажировки, выполнены мною в полном объёме.

Руководитель стажировки (наставник):

Заведующий Лаборатории

Интеллектуальной аналитики ЦЦРиИИ С.В. Боловцов

Стажер:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ФИО) ( подпись) ( расшифровка подписи)

«28» июня 2023 г.

**Приложение 1**.

app.py:

**import** base64  
**import** logging  
**import** os  
**import** time  
  
**from** flask **import** Flask, request, render\_template, session, send\_file  
**from** classes.converter **import** ImageConverter  
**from** classes.resizer **import** ImageResizer  
  
app = Flask(\_\_name\_\_)  
app.secret\_key = **"11111"***# Configure logging*logging.basicConfig(level=logging.INFO)  
  
  
@app.route(**'/'**)  
@app.route(**'/about\_us'**)  
**def** main():  
 request\_path = **'/about\_us'  
 return** render\_template(**'about\_us.html'**,request\_path=request\_path)  
  
  
@app.route(**'/convert'**)  
**def** convert\_page():  
 request\_path = **'/convert'  
 return** render\_template(**'convert.html'**,request\_path=request\_path)  
  
  
@app.route(**'/resize'**)  
**def** resize\_page():  
 request\_path = **'/resize'  
 return** render\_template(**'resize.html'**,request\_path=request\_path)  
  
  
@app.route(**'/convert'**, methods=[**'POST'**])  
**def** convert():  
 request\_path = **'/convert'  
 try**:  
 image = request.files[**'image'**]  
 format = request.form[**'format'**]  
 **except** Exception **as** e:  
 logging.error(**f'Just a page or error retrieving image or format: {**e**}'**)  
 **return** render\_template(**'convert.html'**,request\_path=request\_path)  
  
 converter = ImageConverter(image, format)  
 converted\_image = converter.convert\_to\_new\_format()  
 media\_dir = os.path.join(app.root\_path, **'media'**)  
 **if not** os.path.exists(media\_dir):  
 os.makedirs(media\_dir)  
  
 *# создаем путь для файла с уникальным идентификатором* path = os.path.join(media\_dir, **"image\_%s.%s"** % ((str(int(time.time() \* 1000))), format))  
 converter.save\_to\_file(path)  
  
 session[**'path'**] = path  
 session[**'format'**] = format  
 logging.info(path)  
 logging.info(**'Image conversion successful'**)  
 converted\_image\_base64 = base64.b64encode(converted\_image.getvalue()).decode(**'utf-8'**)  
 **return** render\_template(**'convert.html'**, converted\_image=converted\_image\_base64, request\_path=request\_path)  
  
  
@app.route(**'/resize'**, methods=[**'POST'**])  
**def** resize():  
 request\_path = **'/resize'  
 try**:  
 image = request.files[**'image'**]  
 side = request.form[**'side'**]  
 px\_value = int(request.form.get(**'px'**))  
 logging.info(side)  
 logging.info(px\_value)  
 **if** px\_value < 50 **or** px\_value > 1000:  
 **raise** Exception(**"size in px is not valid"**)  
 **except** Exception **as** e:  
 logging.error(**f'Just a page or error retrieving image or format: {**e**}'**)  
 **return** render\_template(**'resize.html'**, request\_path=request\_path)  
  
 image\_extension = os.path.splitext(image.filename)[1][1:].lower()  
  
 **if** side == **'x'**:  
 resizer = ImageResizer(image, px\_value, **None**)  
 **else**:  
 resizer = ImageResizer(image, **None**, px\_value)  
 resized\_image = resizer.resize\_image()  
  
 media\_dir = os.path.join(app.root\_path, **'media'**)  
 **if not** os.path.exists(media\_dir):  
 os.makedirs(media\_dir)  
  
 path = os.path.join(media\_dir, **"image\_%s.%s"** % (str(int(time.time() \* 1000)), image\_extension))  
 resizer.save\_to\_file(path)  
  
 *# сохраняем в сессии данные* session[**'path'**] = path  
 session[**'format'**] = image\_extension  
 logging.info(path)  
 logging.info(**'Image resizing successful'**)  
 resized\_image\_base64 = base64.b64encode(resized\_image.tobytes()).decode(**'utf-8'**)  
 **return** render\_template(**'resize.html'**, resized\_image=resized\_image\_base64, request\_path=request\_path)  
  
  
@app.route(**'/download'**)  
**def** download():  
 logging.info(session)  
 **if 'path' in** session:  
 image\_path = session[**'path'**]  
 format = session[**'format'**]  
 *# Set the appropriate MIME type based on the format* mime\_types = {  
 **'jpeg'**: **'image/jpeg'**,  
 **'jpg'**: **'image/jpeg'**,  
 **'png'**: **'image/png'**,  
 **'gif'**: **'image/gif'** }  
 *# указываем формат документа, который отправляем для скачивания  
 # благодаря application/octet-stream браузер предложит скачать файл* mime\_type = mime\_types.get(format, **'application/octet-stream'**)  
  
 file\_name\_session = session[**'path'**].split(**'\\'**)[-1]  
 media\_dir = os.path.join(app.root\_path, **'media'**)  
 *# удаляем все файлы кроме файла сессии  
 # чтобы не копился мусор* **for** file\_name **in** os.listdir(media\_dir):  
 file\_path = os.path.join(media\_dir, file\_name)  
 **if** os.path.isfile(file\_path) **and** file\_name != file\_name\_session:  
 os.remove(file\_path)  
  
 **return** send\_file(image\_path, mimetype=mime\_type, as\_attachment=**True**)  
 **else**:  
 **return 'No processed image found.'  
  
  
if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 logging.info(**'app start'**)  
 app.run()

**Приложение 2**.

converter.py:

**from** PIL **import** Image  
**import** io  
  
  
**class** ImageConverter:  
 **def** \_\_init\_\_(self, image\_file, format):  
 self.image\_file = image\_file  
 self.format = format  
 self.converted\_image = **None  
  
 def** convert\_to\_new\_format(self):  
 image = Image.open(self.image\_file)  
 converted\_image = io.BytesIO()  
 **if** image.mode == **'RGBA' and** self.format != **'PNG'**:  
 *# конвертируем RGBA в RGB if если новый формат не png  
 # иначе файл не сохранится в другом формате из-за альфа-канала* image = image.convert(**'RGB'**)  
 image.save(converted\_image, format=self.format)  
 converted\_image.seek(0)  
 self.converted\_image = converted\_image  
 **return** converted\_image  
  
 **def** save\_to\_file(self, file\_path):  
 converted\_image = self.converted\_image  
 **with** open(file\_path, **'wb'**) **as** file:  
 file.write(converted\_image.getvalue())

**Приложение 3**.

resizer.py:

**from** PIL **import** Image  
  
  
**class** ImageResizer:  
 **def** \_\_init\_\_(self, image\_file, x, y):  
 self.image\_file = image\_file  
 self.side\_x = **None if** x **is None else** int(x)  
 self.side\_y = **None if** y **is None else** int(y)  
 self.resized\_image = **None  
  
 def** resize\_image(self):  
  
 im1 = Image.open(self.image\_file)  
  
 width, height = im1.size  
 coef\_ = 1  
  
 *# считаем коэффициент как отношение нового и старого размера* **if not** self.side\_x **is None**:  
 coef\_ = self.side\_x / width  
  
 **elif not** self.side\_y **is None**:  
 coef\_ = self.side\_y / height  
  
 width\_new = width \* coef\_  
 height\_new = height \* coef\_  
 *# resize image* self.resized\_image = im1.resize((int(width\_new), int(height\_new)))  
 **return** self.resized\_image  
  
 **def** save\_to\_file(self, file\_path):  
 *# сохраняем файл без кодировки в base64, так как мы не показываем миниатюру* resized\_image = self.resized\_image  
 resized\_image.save(file\_path)