

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «МИСИС»**

*ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ  
И КОМПЬЮТЕРНЫХ НАУК  
КАФЕДРА ИНЖЕНЕРНОЙ КИБЕРНЕТИКИ*

**Отчет о производственной практике**

Выполнил учащийся:

Савенков Д. В.

группа МПИ-23-1-2

Руководитель практики:

к.э.н., доцент кафедры  
инженерной кибернетики

Бакулов К.С.

Москва 2024

# **ОГЛАВЛЕНИЕ**

<b>1. Введение.....</b>	<b>3</b>
1. Цели и сроки проведения производственной практики согласно рабочей программе дисциплины.....	3
2. Тема производственной практики в 2024 году .....	3
3. Задачи производственной практики .....	3
4. Актуальность темы.....	4
5. Термины и определения.....	4
6. Перечень сокращений и обозначений .....	4
<b>2. СОЗДАНИЕ ПРОТОТИПА ПРОГРАММНОГО ПРОДУКТА В ОБЛАСТИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА И МАШИННОГО ОБУЧЕНИЯ .....</b>	<b>5</b>
<b>3. Заключение .....</b>	<b>7</b>
<b>4. Список литературы.....</b>	<b>8</b>
<b>5. Приложение 1 .....</b>	<b>10</b>
<b>6. Приложение 2 .....</b>	<b>19</b>
<b>7. Приложение 3 .....</b>	<b>23</b>

## **1. ВВЕДЕНИЕ**

### **1. Цели и сроки проведения производственной практики согласно рабочей программе дисциплины**

- закрепление и углубление теоретической подготовки обучающихся в области информатики, связанной с искусственным интеллектом и машинным обучением,
- приобретение и закрепление обучающимися умений в рамках профессиональных компетенций, таких как:

- способность осуществлять критический анализ новых и сложных инженерных объектов, процессов и систем в междисциплинарном контексте, проблемных ситуаций на основе системного подхода;
- способность интегрировать знания и принимать решения в сложных ситуациях, формулировать суждения на основе неполной или ограниченной информации, управлять проектом на различных этапах его жизненного цикла;
- осуществлять эффективное управление разработкой программных средств и проектов, демонстрировать практические навыки для решения сложных задач, выполнения сложного проектирования, а также проведения комплексных исследований, и т.п.

В процессе прохождения производственной практики происходит изучение управлеченческих моделей и производственных стандартов ООО “Гарпикс” по тестированию и внедрению дополнений (обновлений) к существующему ПО.

Практика проводится для обучающихся I курса магистратуры по направлению “Прикладная информатика” профиль “Искусственный интеллект и машинное обучение” (кафедра инженерной кибернетики НИТУ МИСИС).

Сроки практики: 01.07.2024 – 22.07.2024.

### **2. Тема производственной практики в 2024 году**

Создание прототипа программного продукта в области искусственного интеллекта и машинного обучения с учетом ключевых требований индустриальной разработки на основе управлеченческих моделей и производственных стандартов ООО “Гарпикс” с целью приобретения практического опыта.

### **3. Задачи производственной практики**

В процессе прохождения практики практиканты должны решить следующие задачи:

1. Знакомство с производственным циклом, стандартами и требованиями безопасности ООО “Гарпикс”.
2. Изучение принципов организации и управления научно-исследовательских и производственных процессов производства ИТ-продуктов предприятия ООО “Гарпикс”.
3. Создание прототипа программного продукта в области искусственного интеллекта и машинного обучения с учетом ключевых требований индустриальной разработки на основе управлеченческих моделей и производственных стандартов ООО “Гарпикс”

4. Организационно-производственное взаимодействие с представителями компании-организатора практики в формате Agile
5. Защита итоговых результатов практики перед руководителями практики от производства и от кафедры.

#### **4. Актуальность темы**

Актуальность темы практики обусловлена поэтапностью подготовки студентов-магистрантов направления «Прикладная информатика» к трудуоустройству в ИТ-компаниях или ИТ-подразделениях компаний производственной, финансовой и иных сфер экономики в качестве управленцев нижнего и среднего звена. В процессе производственной практики студенты получают более детальные представления о специфике организационно-производственного взаимодействия в ИТ-компаниях (или ИТ-подразделениях отраслевых компаний) и пробуют себя в управлении решением производственных задач, проявляя индивидуальные знания и умения, а также развивая навыки командно-проектного взаимодействия внутри производственного коллектива.

#### **5. Термины и определения**

Серверное приложение – это программное обеспечение, которое выполняется на сервере и предоставляет функциональность или услуги клиентским приложениям через сеть.

Программный модуль – отдельная часть или компонент программы, который выполняет определённую функцию или набор функций.

Искусственный интеллект – область компьютерных наук, занимающаяся созданием систем, способных выполнять задачи, которые обычно требуют человеческого интеллекта.

Framework – программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

REST API (Representational State Transfer API) — это интерфейс программирования, использующий принципы REST для взаимодействия с веб-сервисами через HTTP. Он определяет набор стандартных методов (GET, POST, PUT, DELETE, PATCH) для обмена данными между клиентом и сервером, работающих с представлениями ресурсов и идентифицируемых через URL.

Unit-тесты – это тесты, нацеленные на проверку работоспособности отдельных функциональных модулей, а также фрагментов кода программного обеспечения или его процессов.

#### **6. Перечень сокращений и обозначений**

IT – Information Technology

НИТУ – Национальный исследовательский технологический университет

МИСИС – Московский институт стали и сплавов

ООО – общество с ограниченной ответственностью

## **2. СОЗДАНИЕ ПРОТОТИПА ПРОГРАММНОГО ПРОДУКТА В ОБЛАСТИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА И МАШИННОГО ОБУЧЕНИЯ**

Для практического закрепления управленческих компетенций ИТ-руководителя низшего и среднего звена были использованы задания, предоставленные продуктным подразделением ООО “Гарпикс”.

В рамках создания прототипа программного продукта в области искусственного интеллекта и машинного обучения с учетом ключевых требований индустриальной разработки на основе управленческих моделей и производственных стандартов ООО “Гарпикс” организаторами практики было предложено решить следующую задачу: создания программного продукта (серверного приложения) для детекции «точек-интереса».

Требования к функционалу продукта: на RGB-изображениях детектирует “точки-интереса”. В качестве “точек интереса” (Point-of-Interest) выступают углы прямоугольников (или близких к ним фигур), присутствующих на изображениях. Прямоугольники образуются гранями параллелепипедов перпендикулярные (или “почти перпендикулярные”) к оптической оси камеры.

В процессе работы были выполнены следующие этапы:

### **1) Изучение технологий для реализации задачи:**

Были изучены технологии и инструменты, необходимые для реализации задачи: Python, библиотека OpenCV2, фреймворк Django [1], интерфейс программирования REST API [2], Docker [3], и инструменты для тестирования и документирования API.

### **2) Реализация алгоритма, способного детектировать «точки-интереса» на RGB-изображениях:**

Алгоритм реализован на языке программирования Python с использованием библиотеки OpenCV2. Этот алгоритм детектирует углы (точки интереса) на RGB-изображениях, образованные гранями параллелепипедов. Результат возвращается в виде JSON-строки с координатами точек интереса.

### **3) Создание серверного приложения с доступом в виде REST API:**

Серверное приложение создано на фреймворке Django. Оно обеспечивает доступ к функционалу через REST API с авторизацией по Bearer Token. Основной метод API обрабатывает изображение, получает его в теле запроса и возвращает JSON-строку с координатами точек интереса. Приложение контейнеризировано с использованием Docker для упрощения развертывания и управления средой выполнения.

### **4) Покрытие unit-тестами:**

Составлены unit-тесты для проверки основных функций программы. Для обеспечения качества и надежности кода добавлена автоматизация тестирования с использованием библиотеки eqator [4]. Программа проходит тесты на соответствие стандартам кодирования (flake8), вычисление метрик кода (radon), проверку безопасности (bandit) и покрытие тестами (coverage).

### **5) Документирование:**

Разработано техническое задание в соответствии с ГОСТ 19.201 [5]. Создана и добавлена Postman-коллекция для взаимодействия с API в виде POST-запросов.

Разработана документация API в формате Swagger [6], что обеспечивает удобство работы и тестирования запросов.

В результате выполнения поставленной задачи был приобретен практический опыт создания серверного приложения с учетом ключевых требований индустриальной разработки. Успешно реализован алгоритм детектирования «точек-интереса» на RGB-изображениях. Созданное серверное приложение на фреймворке Django обеспечило надежное и эффективное взаимодействие с клиентами через REST API. Контейнеризация с использованием Docker значительно упростила процесс развертывания и эксплуатации приложения. Разработанные и внедренные unit-тесты, а также автоматизация тестирования, гарантировали высокое качество кода и стабильность работы приложения. Полная и подробная документация, включая техническое задание, Postman-коллекцию и Swagger-документацию, обеспечила удобство взаимодействия и интеграции с другими системами. Результат работы загружен в репозиторий и направлен руководителю практики для проверки и включения в итоговый отчет.

### **3. ЗАКЛЮЧЕНИЕ**

В ходе практики были выполнены следующие задачи и получены следующие результаты:

- изучены принципы организации и управления научно-исследовательских и производственных процессов предприятия в рамках продуктового направления ООО “Гарпикс”;
- реализован алгоритм детекции углов Харриса, способный детектировать «точки-интереса» на фотографиях (RGB-изображения);
- разработано серверное приложение с доступом в виде REST API;
- создано сопровождение в виде unit-тестов, а также автоматизировано тестирование;
- документирование серверного приложения.

#### **4. СПИСОК ЛИТЕРАТУРЫ**

1. <https://docs.djangoproject.com/en/5.0/>
2. <https://rapidapi.com/guides/api-docs-rest>
3. <https://docs.docker.com/>
4. <https://github.com/garpixcms/eqator>
5. Готовый проект - <https://github.com/D-Zane/DetectorApi>
6. <https://protect.gost.ru/v.aspx?control=8&baseC=-1&page=0&month=-1&year=-1&search=&RegNum=1&DocOnPageCount=15&id=147258>
7. <https://docs.swagger.io/spec.html>



## **5. ПРИЛОЖЕНИЕ 1**

### **Техническое задание, составленное в соответствии с ГОСТ 19.201**

#### **АННОТАЦИЯ**

В данном программном документе приведено техническое задание на разработку программного продукта для детекции «точек-интереса» на фотографиях.

В данном программном документе, в разделе «Введение» указано наименование, краткая характеристика области применения программы (программного изделия).

В разделе «Основания для разработки» указаны документы, на основании которых ведется разработка, наименование и условное обозначение темы разработки.

В данном программном документе, в разделе «Назначение разработки» указано функциональное и эксплуатационное назначение программы (программного изделия).

Раздел «Требования к программе» содержит следующие подразделы:

- требования к функциональным характеристикам;
- требования к надежности;
- условия эксплуатации;
- требования к составу и параметрам технических средств;
- требования к информационной и программной совместимости;
- специальные требования.

В данном программном документе, в разделе «Требования к программной документации» указаны предварительный состав программной документации и специальные требования к ней.

В разделе «Технико-экономические показатели» указаны: ориентировочная экономическая эффективность, предполагаемая годовая потребность, экономические преимущества разработки.

В данном программном документе, в разделе «Стадии и этапы разработки» установлены необходимые стадии разработки, этапы и содержание работ.

В разделе «Порядок контроля и приемки» должны быть указаны виды испытаний и общие требования к приемке работы.

Оформление программного документа «Руководство оператора» произведено по требованиям ЕСПД (ГОСТ 19.101-77 1), ГОСТ 19.103-77 2), ГОСТ 19.104-78\* 3), ГОСТ 19.105-78\* 4), ГОСТ 19.106-78\* 5), ГОСТ 19.201-78 6), ГОСТ 19.604-78\* 7)).

# **1. ВВЕДЕНИЕ**

## **1.1   Наименование программы**

Наименование - «Программа для детекции «точек-интереса» на RGB-изображениях».

## **1.2 Краткая характеристика области применения программы**

Программный продукт может использоваться в системах компьютерного зрения, в автоматизированных системах контроля и управления, а также в любых других приложениях, требующих детектирования "точек интереса" на фотографиях.

# **2. ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ**

## **2.1. Основание для проведения разработки**

Основанием для разработки данного приложения является задание руководителя практики.

## **2.2. Наименование и условное обозначение темы разработки**

Наименование темы разработки - «Разработка программного продукта (серверного приложения) для детекции «точек-интереса»».

# **3. НАЗНАЧЕНИЕ РАЗРАБОТКИ**

## **3.1. Функциональное назначение программы**

Функциональным назначением программы является детектирование "точек интереса" на RGB-изображениях, выявляя углы прямоугольников или подобных фигур, перпендикулярных или почти перпендикулярных к оптической оси камеры. Программа возвращает координаты углов в формате JSON. Взаимодействие осуществляется через защищенный REST API с авторизацией по Bearer Token. API документируется в формате Swagger. Программа включает unit-тесты, статическую проверку кода с использованием flake8, radon, bandit и coverage, а также автоматизацию тестирования с помощью инструмента equator.

## **3.2. Эксплуатационное назначение программы**

Программа должна эксплуатироваться в профильных подразделениях. Конечными пользователями программы должны являться сотрудники профильных подразделений.

Программа работает как серверное приложение на Django. Контейнеризуется с помощью Docker. Поддерживает сценарии CI/CD для автоматического развертывания и тестирования.

Совместима с существующими системами через стандартизованные API. Пользователям предоставляются документация и руководства

## **4. ТРЕБОВАНИЯ К ПРОГРАММНОМУ ПРОДУКТУ**

### **4.1. Требования к функциональным характеристикам**

#### **4.1.1. Требования к составу выполняемых функций**

Программа должна обеспечивать возможность выполнения следующих функций:

- а) Прием RGB-изображений в формате JPEG или PNG.
- б) Обработка изображений для детектирования углов прямоугольников или подобных фигур, перпендикулярных или почти перпендикулярных к оптической оси камеры.
- в) Вывод координат найденных "точек интереса" в формате JSON.
- г) Предоставление доступа к функционалу через защищенный REST API с авторизацией по Bearer Token.
- д) Документирование API в формате Swagger.
- е) Выполнение unit-тестов для проверки основных модулей.
- ж) Проведение статической проверки кода.
- з) Автоматизация тестирования.

#### **4.1.2. Требования к организации входных данных**

Входные данные программы должны быть организованы в виде изображений в формате JPEG или PNG, передаваемых через HTTP-запросы в теле POST-запроса.

#### **4.1.3. Требования к организации выходных данных**

Выходные данные программы должны быть организованы в виде JSON-объекта, содержащего координаты "точек интереса". Формат ответа должен включать массив объектов, где каждый объект представляет собой координаты угла в виде пар значений

#### **4.1.4. Требования к временным характеристикам**

Требования к временным характеристикам программы не предъявляются.

### **4.2. Требования к надежности**

#### **4.2.1. Требования к обеспечению надежного (устойчивого) функционирования программы**

Надежное (устойчивое) функционирование программы должно быть обеспечено выполнением совокупности организационно-технических мероприятий, перечень которых приведен ниже:

- a) организацией бесперебойного питания технических средств;
  - б) регулярным выполнением рекомендаций Министерства труда и социального развития РФ, изложенных в Постановлении от 23 июля 1998 г. «Об утверждении межотраслевых типовых норм времени на работы по сервисному обслуживанию ПЭВМ и оргтехники и сопровождению программных средств»;
  - в) регулярным выполнением требований ГОСТ 51188–98. Защита информации.
- Испытания программных средств на наличие компьютерных вирусов;
- г) необходимым уровнем квалификации сотрудников профильных подразделений.

#### **4.2.2. Время восстановления после отказа**

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать времени, необходимого на перезагрузку операционной системы и запуск программы, при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

### **4.3. Условия эксплуатации**

#### **4.3.1. Климатические условия эксплуатации**

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

#### **4.3.2. Требования к видам обслуживания**

См. Требования к обеспечению надежного (устойчивого) функционирования программы.

#### **4.3.3. Требования к численности и квалификации персонала**

Минимальное количество персонала, требуемого для работы программы, должно составлять не менее 2 штатных единиц - системный программист и конечный пользователь программы - оператор.

Системный программист должен иметь минимум среднее техническое образование.

В перечень задач, выполняемых системным программистом, должны входить:

- а) задача поддержания работоспособности технических средств;
- б) задачи установки (инсталляции) и поддержания работоспособности системных программных средств - операционной системы;
- в) задача установки (инсталляции) программы.

Конечный пользователь программы (оператор) должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы.

Персонал должен быть аттестован минимум на II квалификационную группу по электробезопасности (для работы с конторским оборудованием).

### **4.4. Требования к составу и параметрам технических средств**

В состав технических средств должен входить сервер или рабочая станция со следующими параметрами:

- а) процессор: Intel Core i5 или аналогичный AMD Ryzen 5 с тактовой частотой не менее 2.5 ГГц;
- б) оперативная память: 8 ГБ DDR4 или выше;
- в) жесткий диск: SSD объемом 256 ГБ или выше для быстрой загрузки и обработки данных;
- г) сетевое подключение: Ethernet-адаптер с поддержкой Gigabit Ethernet (1000 Mbps) для стабильного и быстрого обмена данными;
- д) операционная система: Поддержка Linux (Ubuntu 20.04 LTS или выше) или Windows Server 2019;

е) контейнеризация: Установленная среда Docker для развертывания и управления контейнерами;

ж) программное обеспечение:

и. Python версии 3.8 или выше.

ii. Django версии 4.0 или выше.

iii. Библиотеки для обработки изображений: OpenCV, NumPy.

При развертывании серверного приложения требуется обеспечить доступ к интернету для установки обновлений и интеграции с внешними сервисами.

## **4.5. Требования к информационной и программной совместимости**

### **4.5.1. Требования к информационным структурам и методам решения**

Требования к информационным структурам (файлов) на входе и выходе, а также к методам решения. Входные данные: изображения в формате JPEG или PNG передаются через HTTP POST-запрос. Выходные данные: результаты обработки представляются в формате JSON с массивом объектов, где каждый объект содержит координаты углов "точек интереса".

Для обработки изображений используются алгоритмы компьютерного зрения для детектирования углов и контуров. API предоставляет результаты через REST API с авторизацией по Bearer Token. Документация API выполнена в формате Swagger, код включает unit-тесты.

### **4.5.2. Требования к исходным кодам и языкам программирования**

Исходные коды программы должны быть реализованы на языке Python. В качестве фреймворка для разработки серверного приложения должна быть использована Django версии 4.0 или выше. В качестве интегрированной среды разработки программы должна быть использована среда Visual Studio Code (локализованная, русская версия).

### **4.5.3. Требования к программным средствам, используемым программой**

Системные программные средства, используемые программой, должны быть представлены локализованной версией операционной системы Linux (Ubuntu 20.04 LTS или выше) или Windows Server 2019.

## **4.6. Требования к маркировке и упаковке**

Требования к маркировке и упаковке программы не предъявляются.

## **4.7. Требования к транспортированию и хранению**

Требования к транспортированию и хранению программы не предъявляются.

## **4.8. Специальные требования**

Специальные требования к программе не предъявляются.

# **5. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ**

## **5.1. Предварительный состав программной документации**

Состав программной документации должен включать в себя:

- 1) техническое задание, составленное в соответствии с ГОСТ 19.201;
- 2) документация для взаимодействия с API в формате postman-коллекции;
- 3) документация API в формате Swagger;
- 4) описание программы;

## **5.2. Специальные требования к программной документации**

Специальные требования к программной документации не предъявляются.

# **6. Технико-экономические показатели**

## **6.1. Ориентировочная экономическая эффективность**

Ориентировочная экономическая эффективность не рассчитываются.

## **6.2. Предполагаемая годовая потребность**

Предполагаемое число использования программы в год – круглосуточная работа программы на одном рабочем месте.

## **6.3. Экономические преимущества разработки**

Разработка продукта позволяет снизить затраты на обработку изображений за счет автоматизации, повысить производительность и уменьшить ошибки. Интеграция через REST API упрощает соединение с другими системами, что сокращает затраты на разработку интерфейсов. Использование стандартных инструментов и автоматизированного тестирования снижает затраты на поддержку. Гибкость и масштабируемость системы через контейнеризацию и CI/CD уменьшают затраты на инфраструктуру.

## **7. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ**

### **7.1. Стадии разработки**

Разработка должна быть проведена в три стадии:

- 1) разработка технического задания;
- 2) реализация алгоритма и создание серверного приложения с доступом в виде REST API;
- 3) сопровождение unit-тестами, автоматизация тестирования.

### **7.2. Этапы разработки**

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

- 1) разработка алгоритма;
- 2) разработка программной документации;
- 3) испытания программы.

На стадии внедрения должен быть выполнен этап разработки - подготовка и передача программы.

### **7.3. Содержание работ по этапам**

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

- 1) постановка задачи;
- 2) определение и уточнение требований к техническим средствам;
- 3) определение требований к программе;
- 4) определение стадий, этапов и сроков разработки программы и документации на неё;
- 5) выбор языков программирования;
- 6) согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с требованиями ГОСТ 19.201 и требованием п. «Предварительный состав программной документации» настоящего технического задания.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

- 1) разработка, согласование и утверждение программы и методики испытаний;
- 2) проведение приемо-сдаточных испытаний;

3) корректировка программы и программной документации по результатам испытаний.

На этапе подготовки и передачи программы должна быть выполнена работа по подготовке и передаче программы и программной документации в эксплуатацию.

#### **7.4. Исполнители**

Руководитель практики

от кафедры

Бакулов К.С.

Руководитель практики

от профильной организации

Мишурин С.С.

Исполнитель

Студент

Савенков Д.В.

## **8. ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ**

### **8.1. Виды испытаний**

Функциональные испытания включают проверку выполнения всех требований программы. Тестирование производительности оценивает время обработки и ресурсы, необходимые для выполнения программы. Тестирование безопасности анализирует уязвимости кода и проверяет защищенность API. Интеграционные испытания проверяют корректность работы REST API и взаимодействие программы с клиентскими приложениями. Тестирование качества кода включает проверку стандартов кодирования, метрик кода и покрытия тестами.

Приемо-сдаточные испытания проводятся по согласованной методике и проверяют выполнение всех требований технического задания. Документальные испытания включают проверку соответствия и актуальности всей необходимой документации.

### **8.2. Общие требования к приемке работы**

Программа должна полностью соответствовать требованиям технического задания. Все документы должны быть предоставлены в соответствии с установленными стандартами и быть актуальными. Код должен соответствовать установленным стандартам качества и демонстрировать адекватное покрытие тестами. Программа должна успешно развертываться и функционировать в Docker-среде, показывая стабильность и удовлетворительную производительность.

Результаты испытаний должны быть документированы и согласованы с заказчиком перед окончательной приемкой.

## 6. ПРИЛОЖЕНИЕ 2

**Документация для взаимодействия с API в форме postman-коллекции.**

```
{  
  "info": {  
    "_postman_id": "c3500210-8653-4f51-8301-337f71597e39",  
    "name": "Postman коллекция для API",  
    "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",  
    "_exporter_id": "28795049"  
  },  
  "item": [  
    {  
      "name": "Регистрация",  
      "request": {  
        "method": "POST",  
        "header": [  
          {  
            "key": "Content-Type",  
            "value": "application/json"  
          }  
        ],  
        "body": {  
          "mode": "raw",  
          "raw": "{\"username\": \"Testuser123456\", \"password\": \"password123\"}",  
          "options": {  
            "raw": {  
              "language": "json"  
            }  
          }  
        },  
        "url": {  
          "raw": "http://127.0.0.1:8000/api/register/",  
          "protocol": "http",  
          "host": [  
            "127",  
            "0",  
            "0",  
            "1"  
          ],  
          "port": "8000",  
          "path": [  
            "api",  
            "register",  
            ""  
          ]  
        }  
      },  
      "response": []  
    },  
    {  
      "name": "Получение токена авторизации",  
      "request": {  
        "method": "POST",  
        "header": [  
          {  
            "key": "Content-Type",  
            "value": "application/json"  
          }  
        ],  
        "body": {  
          "mode": "raw",  
          "raw": "{\"username\": \"Testuser123456\", \"password\": \"password123\"}",  
          "options": {  
            "raw": {  
              "language": "json"  
            }  
          }  
        },  
        "url": {  
          "raw": "http://127.0.0.1:8000/api/auth/",  
          "protocol": "http",  
          "host": [  
            "127",  
            "0",  
            "0",  
            "1"  
          ],  
          "port": "8000",  
          "path": [  
            "api",  
            "auth",  
            ""  
          ]  
        }  
      },  
      "response": []  
    }  
  ]  
}
```

```

        "method": "POST",
        "header": [],
        "body": {
            "mode": "urlencoded",
            "urlencoded": [
                {
                    "key": "username",
                    "value": "Corgi",
                    "type": "text"
                },
                {
                    "key": "password",
                    "value": "testpassword",
                    "type": "text"
                }
            ]
        },
        "url": {
            "raw": "http://127.0.0.1:8000/api/token/",
            "protocol": "http",
            "host": [
                "127",
                "0",
                "0",
                "1"
            ],
            "port": "8000",
            "path": [
                "api",
                "token",
                ""
            ]
        }
    },
    "response": []
},
{
    "name": "Обработка изображений и получение результата в формате json строки",
    "request": {
        "auth": {
            "type": "bearer",
            "bearer": [
                {
                    "key": "token",
                    "value": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9eyJ0b2tlbl90eXBIIjoiYWNjZXNzIwiZXhwIjoxNzIxNTAyMDQxLCJpYXQiOjE3MjE1MDExNDEsImp0aSI6IjRiYTliYTZhZjI3NjRmOWZiYmQ2MGQ2MTZkODZlZjc1IiwidXNlcI9pZCI6Nn0.-k81_aHRMEP6LeHpdSyAUySUIZ3dNnF-kURD1UauWHY",
                    "type": "string"
                }
            ]
        },
        "method": "POST",
        "header": [
            {
                "key": "Authorization",

```

```

        "value": "Token YOUR_TOKEN"
    }
],
"body": {
    "mode": "formdata",
    "formdata": [
        {
            "key": "image",
            "type": "file",
            "src": [
                "/C:/Users/Дмитрий/Documents/vscode/practicapi/point_detector/input/1_Color.png",
                "/C:/Users/Дмитрий/Documents/vscode/practicapi/point_detector/input/2_Color.png",
                "/C:/Users/Дмитрий/Documents/vscode/practicapi/point_detector/input/3_Color.png",
                "/C:/Users/Дмитрий/Documents/vscode/practicapi/point_detector/input/4_Color.png",
                "/C:/Users/Дмитрий/Documents/vscode/practicapi/point_detector/input/5_Color.png",
                "/C:/Users/Дмитрий/Documents/vscode/practicapi/point_detector/input/6_Color.png",
                "/C:/Users/Дмитрий/Documents/vscode/practicapi/point_detector/input/7_Color.png",
                "/C:/Users/Дмитрий/Documents/vscode/practicapi/point_detector/input/8_Color.png",
                "/C:/Users/Дмитрий/Documents/vscode/practicapi/point_detector/input/9_Color.png",
                "/C:/Users/Дмитрий/Documents/vscode/practicapi/point_detector/input/10_Color.png",
                "/C:/Users/Дмитрий/Documents/vscode/practicapi/point_detector/input/11_Color.png",
                "/C:/Users/Дмитрий/Documents/vscode/practicapi/point_detector/input/12_Color.png",
                "/C:/Users/Дмитрий/Documents/vscode/practicapi/point_detector/input/13_Color.png",
                "/C:/Users/Дмитрий/Documents/vscode/practicapi/point_detector/input/14_Color.png",
                "/C:/Users/Дмитрий/Documents/vscode/practicapi/point_detector/input/15_Color.png",
                "/C:/Users/Дмитрий/Documents/vscode/practicapi/point_detector/input/16_Color.png",
                "/C:/Users/Дмитрий/Documents/vscode/practicapi/point_detector/input/17_Color.png"
            ]
        }
    ],
    "url": {
        "raw": "http://127.0.0.1:8000/api/process-image/",
        "protocol": "http",
        "host": [
            "127",
            "0",
            "0",
            "1"
        ],
        "port": null
    }
}

```

```
        "port": "8000",
        "path": [
            "api",
            "process-image",
            ""
        ]
    },
    "response": []
}
]
```

## 7. ПРИЛОЖЕНИЕ 3

### Документация API в формате Swagger.

```
openapi: 3.0.0
info:
  title: API для детекции “точек-интереса”
  description: Swagger документация для API
  version: 1.0.0
  contact:
    name: Савенков Дмитрий
    email: theangryzane@gmail.com
    url: https://github.com/D-Zane

servers:
  - url: http://127.0.0.1:8000/api
    description: Для локального тестирования
  - url: http://localhost:8000/api
    description: Для тестирования через Docker

paths:
  /register:
    post:
      summary: Регистрация нового пользователя
      description: Зарегистрируйте нового пользователя и получите токен аутентификации.
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                username:
                  type: string
                  example: testuser
                password:
                  type: string
                  example: testpassword
      responses:
        '201':
          description: Пользователь успешно зарегистрирован
          content:
            application/json:
              schema:
                type: object
                properties:
                  token:
                    type: string
                    example: abcdef123456
        '400':
          description: Недопустимый ввод
          content:
            application/json:
              schema:
```

```

type: object
properties:
  detail:
    type: string
    example: "Имя пользователя и/или пароль отсутствуют."

```

/token/:

post:

summary: Получение токена аутентификации

description: Получите токен аутентификации, указав имя пользователя и пароль.

requestBody:

required: true

content:

application/json:

schema:

type: object

properties:

username:

type: string

example: testuser

password:

type: string

example: testpassword

responses:

'200':

description: Токен успешно получен

content:

application/json:

schema:

type: object

properties:

token:

type: string

example: abcdef123456

'400':

description: Неверные данные учетной записи

content:

application/json:

schema:

type: object

properties:

error:

type: string

example: "Неверные данные учетной записи"

/process-image/:

post:

summary: Детекция “точек-интереса”

description: Обработайте загруженное изображение и верните результат в виде json строки

security:

- BearerAuth: []

requestBody:

required: true

content:

multipart/form-data:

schema:

type: object

```
properties:
  image:
    type: string
    format: binary
responses:
  '200':
    description: Изображение обработано успешно
    content:
      application/json:
        schema:
          type: object
          properties:
            points_of_interest:
              type: array
              items:
                type: object
                properties:
                  x:
                    type: integer
                  y:
                    type: integer
  '400':
    description: Изображение не предоставлено или его формат недопустим
    content:
      application/json:
        schema:
          type: object
          properties:
            error:
              type: string
            example: "Изображение не предоставлено или его формат недопустим."
  '500':
    description: Внутренняя ошибка сервера
    content:
      application/json:
        schema:
          type: object
          properties:
            error:
              type: string
            example: "Внутренняя ошибка сервера."
components:
  securitySchemes:
    BearerAuth:
      type: http
      scheme: bearer
      bearerFormat: JWT
```