МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет Компьютерных Наук

Кафедра технологий обработки и защиты информации

Веб-приложение обнаружения объектов «Object Detection»

Курсовая работа по дисциплине
«Технологии программирования»

09.03.02 Информационные системы и технологии
Обработка информации и машинное обучение

Зав. кафедрой _____д.т.н., профессор А.А.Сирота __.__20__

Обучающийся _____К.А. Ветров, 4 курс, д/о

Обучающийся _____К.А. Иванов, 4 курс, д/о

Обучающийся _____И.Г. Буслаев, 4 курс, д/о

Обучающийся _____Р.И. Князев, 4 курс, д/о

Руководитель _____ В.С. Тарасов, ст. преподаватель

Воронеж 2023

Содержание

Введение

В настоящее время технологического прогресса, виртуализации и обработки данных, технология Object Detection становится одной из ключевых областей исследований и разработок. Она играет важную роль в автоматизации, безопасности и оптимизации различных сфер человеческой деятельности, начиная от медицины и промышленности и заканчивая транспортом и образованием. Одной из фундаментальных задач в области компьютерного зрения является обнаружение и идентификация объектов в изображениях и видео.

Технология обнаружения объектов (Object Detection) представляет собой мощный инструмент, позволяющий компьютерам автоматически распознавать и выделять объекты различных классов на изображениях и в потоке видеоданных.

В данной курсовой работе был реализовано веб-приложение, позволяющее каждому прикоснуться к данной технологии.

1 Постановка задачи

Данное веб-приложение создается для предоставления возможности использовать модель обнаружения объектов без необходимости развёртывать её вручную.

1.1 Требования к разрабатываемой системе

1.1.1 Функциональные требования

К разрабатываемому приложению выдвигаются следующие функциональные требования:

- осуществление обнаружения и классификации объектов на загруженных изображениях;
- просмотр истории ранее обработанных изображений авторизованным пользователем.

1.1.2 Нефункциональные требования

К разрабатываемому приложению выдвигаются следующие нефункциональные требования:

- приложение должно отвечать на запросы пользователей в течение нескольких секунд;
- наличие интерфейса, выполненного в едином стиле со всем необходимым набором функций, чтобы с ним могли работать пользователи различных возрастных и культурных групп;
- использование современных технологий и инструментов разработки.

1.2 Требования к архитектуре

Список требований к архитектуре:

- приложение должно быть построено с использованием протоколов HTTP;
- для хранения информации необходимо использовать реляционную базу данных;

— клиентская часть приложения должна быть написана с использованием технологий frontend разработки, таких как HTML, CSS, JavaScript;
— серверная часть приложения должна быть написана с использованием технологий backend разработки, таких как Python и Django [1] на основе архитектурного паттерна MVC. Выбор этого фреймворка объясняется тем, что он включает в себя большое количество готового функционала. И, как правило, проекты, написанные на данном фреймворке, обладают быстрой загрузкой, могут хранить огромные данные на сервере и по умолчанию создают панель администратора для редактирования информации на сайте.

1.3 Задачи, решаемые в процессе разработки

Процесс организации данного веб-приложения построен на основе гибкой методологии Kanban.

В процессе разработки веб-сайта обнаружения объектов будут решаться следующие задачи:

— анализ предметной области: необходимо изучить особенности работы механизма обнаружения объектов;
— проектирование базы данных: на основе полученных требований необходимо разработать структуру базы данных, которая будет использоваться в приложении;
— разработка серверной части приложения: на этом этапе необходимо разработать серверную часть приложения, которая будет отвечать за обработку запросов клиента и взаимодействие с базой данных. Для этого используется фреймворк Django;
— разработка клиентской части приложения: клиентская часть приложения должна быть написана с использованием современных технологий frontend разработки, таких как HTML, CSS, JavaScript;

— тестирование и отладка: на этом этапе необходимо провести тестирование и отладку приложения, чтобы убедиться, что оно соответствует требованиям, определенным в начале проекта.

2 Анализ предметной области

2.1 Терминология (глоссарий) предметной области

Веб-приложение — клиент-серверное приложение, в котором клиент взаимодействует с веб-сервером при помощи браузера.

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

Клиент (клиентская сторона) – сайт, который предоставляет пользователю взаимодействовать со всей системой.

Сервер (серверная часть) – компьютер, обслуживающий другие устройства (клиентов) и предоставляющий им свои ресурсы для выполнения определенных задач.

Backend – логика работы сайта, внутренняя часть продукта, которая находится на сервере и скрыта от пользователя.

Frontend – презентационная часть информационной или программной системы, ее пользовательский интерфейс и связанные с ним компоненты.

MVC – схема разделения данных приложения и управляющей логики на три отдельных компонента: модель, представление и контроллер - таким образом, что модификация каждого компонента может осуществляться независимо.

GitHub – веб-сервис для хостинга IT-проектов и их совместной разработки.

CSS – формальный язык описания внешнего вида документа, написанного с использованием языка разметки (HTML, XHTML, XML).

HTML – стандартизированный язык гипертекстовой разметки веб-страниц в браузере.

JavaScript – язык программирования высокого уровня, который используется для написания frontend- и backend-частей сайтов, а также мобильных приложений.

SQLite – встраиваемая реляционная СУБД с открытым кодом.

Авторизация – предоставление определенному лицу или группе лиц прав на выполнение определенных действий, а также процесс проверки данных прав при попытке выполнения этих действий.

Регистрация – действия, направленные на создание личной учетной записи в приложении, с целью получения доступа к его полной функциональности.

Пользователь – лицо, которое использует действующую систему для выполнения конкретной функции.

Неавторизованный пользователь — пользователь, не прошедший авторизацию или не зарегистрированный в системе.

Авторизованный пользователь — пользователь, прошедший авторизацию в системе.

Сериализация — это процесс преобразования объектов Python в поток байтов, который может быть сохранен в файле или передан по сети.

Десериализация — это процесс получения потока байтов и преобразования его в объект Python.

Docker — это открытая платформа для разработки, доставки и работы с приложениями. С ее помощью можно создавать и развертывать приложения в легковесных, подключаемых контейнерах, которые могут быть запущены и масштабированы на любой платформе.

Обнаружение объектов – компьютерная технология, относящаяся к компьютерному зрению и обработке изображений, которая занимается обнаружением экземпляров семантических объектов определенного класса в цифровых изображениях и видео.

Ограничивающая рамка (bounding box) – прямоугольник, который выделяет объект на изображении.

Точность обнаружения объектов – вероятность, которая показывает, насколько модель для задач обнаружения объектов уверена в предсказанном объекте.

2.2 Обзор аналогов

2.2.1 Aspose

Компания «Aspose» на своём сайте предоставляет инструменты для создания, редактирования, обработки, преобразования и конвертирования различных файлов, включая текстовые, аудио- и видеофайлы. Продуктами Aspose пользуются множества компаний, например Oracle, Lulu, Ubisoft, Red Gate. Согласно статистике [2], из сотни крупнейших компаний в США (Fortune 100) 77% компаний пользуются продуктами Aspose. Одним из инструментов для обработки изображений, доступный на сайте «Aspose», является инструмент для распознавания объектов на изображении. Интерфейс сайта представлен на Рисунке 1.

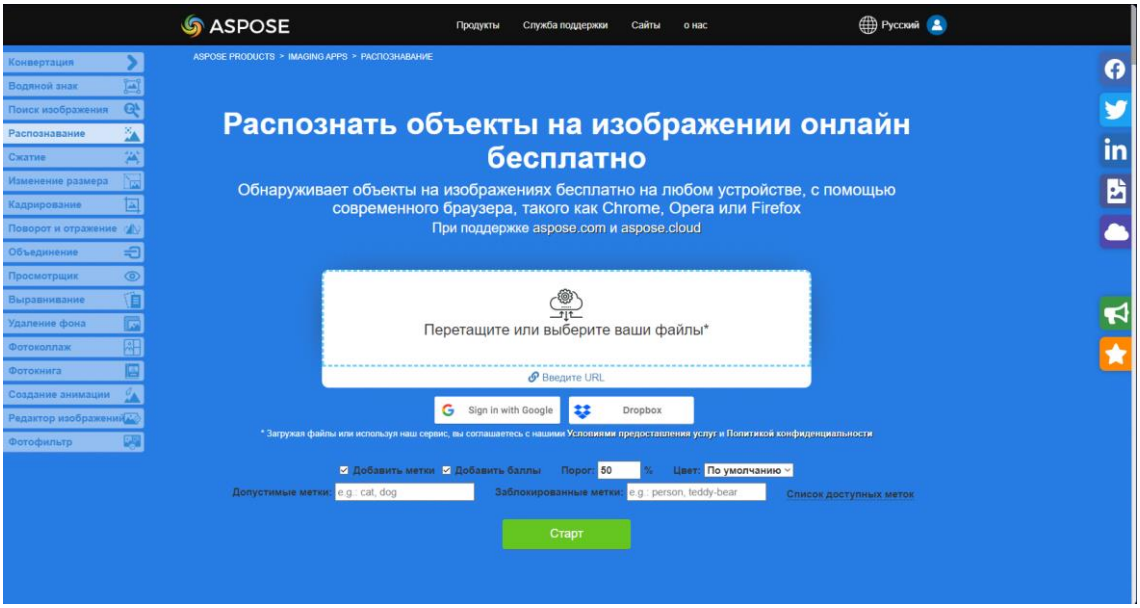

Рисунок 1 - Внешний вид сайта «Aspose»

Сайт «Aspose» обладает следующим рядом преимуществ:

— русскоязычный интерфейс;

— возможность выставить допустимые и заблокированные метки;

— выбор цвета ограничивающей рамки;

— содержит много информации о принципах работы сайта.

И в свою очередь следующим рядом недостатков:

— отсутствует история обработанных изображений;

— при выставлении меток необходимо вводить название меток вручную.

2.2.2 Image Recognize

Сайт «Image Recognize» предоставляет инструменты компьютерного зрения. Инструменты сайта позволяют распознать объекты, контекст, знаменитостей, возраст, пол, лицевые эмоции и наличие небезопасного контента на изображениях. Интерфейс сайта представлен на Рисунке 2.

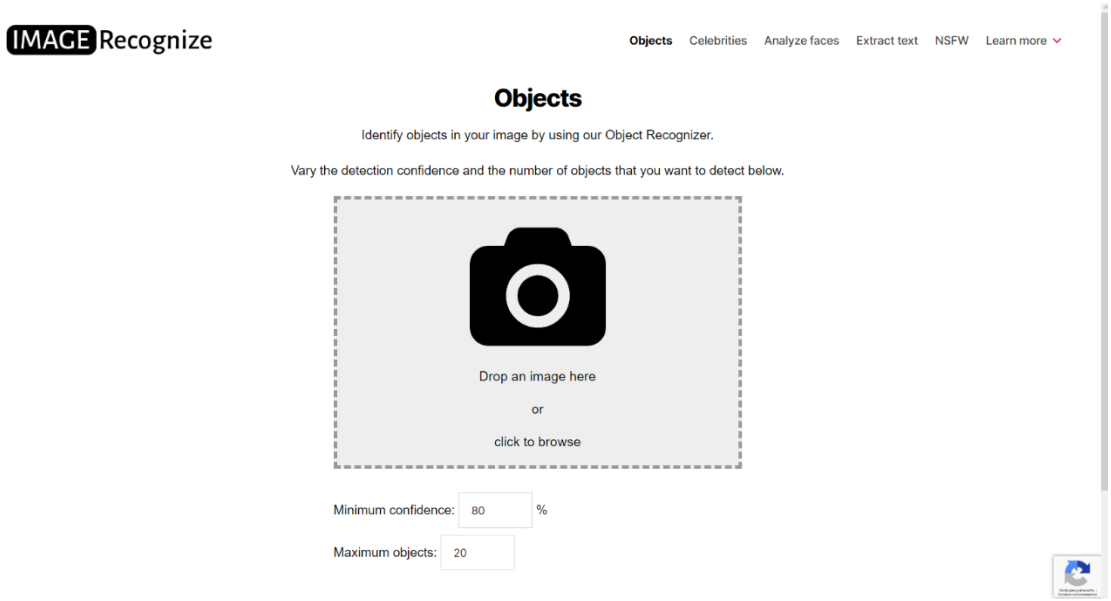

Рисунок 2 - Внешний вид сайта «Image Recognize»

Сайт «Image Recognize» обладает следующим рядом преимуществ:

— возможность настроить параметры точности и числа объектов;

— присутствует большое количество статей об обнаружении объектов;

— облегчённый интерфейс;

— есть отдельный список обнаруженных классов;

— описывает контекст изображения.

И в свою очередь следующим рядом недостатков:

— отсутствует история обработанных изображений;

— нет русскоязычного интерфейса;

— не предусмотрен выбор цвета ограничивающей рамки;

— присутствует ограничение на количество обработок.

2.2.3 Astica

Компания «Astica» разрабатывает инструменты на основе технологий искусственного интеллекта. На своём сайте компания предоставляет такие инструменты, как генерация голоса, обнаружение объектов и лиц, описание изображения. Интерфейс сайта представлен на Рисунке 3.

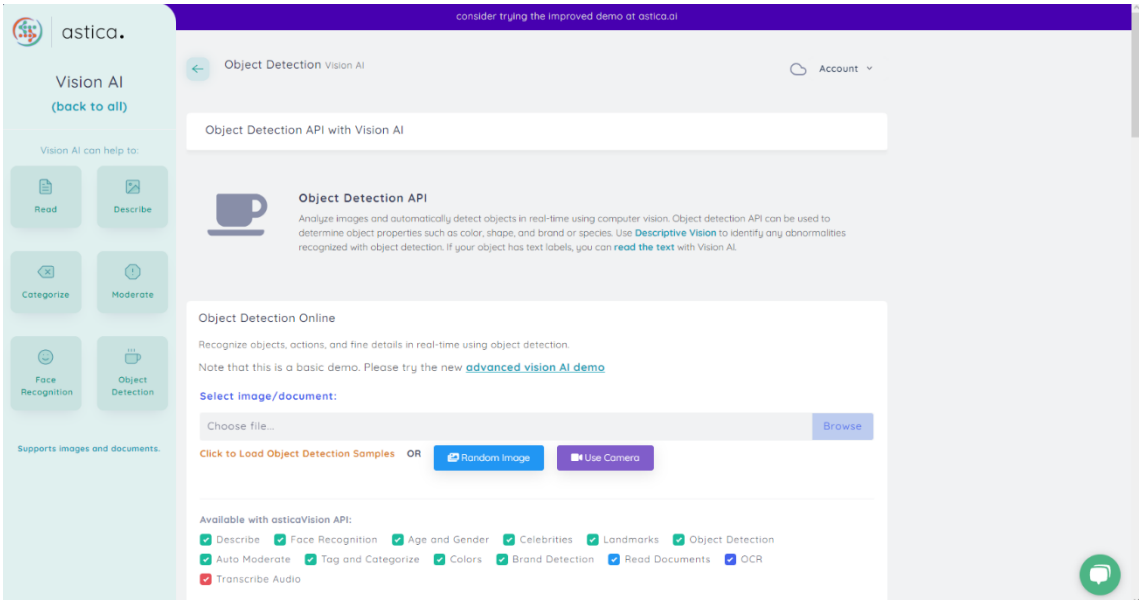

Рисунок 3 - Внешний вид сайта «Astica»

Сайт «Astica» обладает следующим рядом преимуществ:

— детально описывает контекст и части изображения;
— определяет пол и возраст людей на изображении;
— проставляет теги для изображения;
— обнаруживает имеющиеся цвета на изображении и преимущественный цвет переднего и заднего плана.

И в свою очередь следующим рядом недостатков:

— отсутствует история обработанных изображений;
— нет русскоязычного интерфейса;
— не предусмотрен выбор цвета ограничивающей рамки;
— не тонкая настройка параметров;
— перегруженный элементами интерфейс;
— необходимо платить после использования доступных попыток.

2.3 Диаграммы, иллюстрирующие работу системы

2.3.1 Диаграмма прецедентов (Use case)

Диаграмма прецедентов (Use case) представлена для двух типов актёров: неавторизованного пользователя и авторизованного пользователя. У каждого из них своя модель поведения, которую можно проследить на Рисунке 4.

Неавторизованный пользователь может:

— зарегистрироваться в системе;

— авторизоваться в системе;

— осуществлять обнаружение объектов на картинке.

Авторизованный пользователь помимо функций, доступных неавторизованному пользователю, может:

— просматривать историю обработанных картинок;

— осуществлять выход из аккаунта.

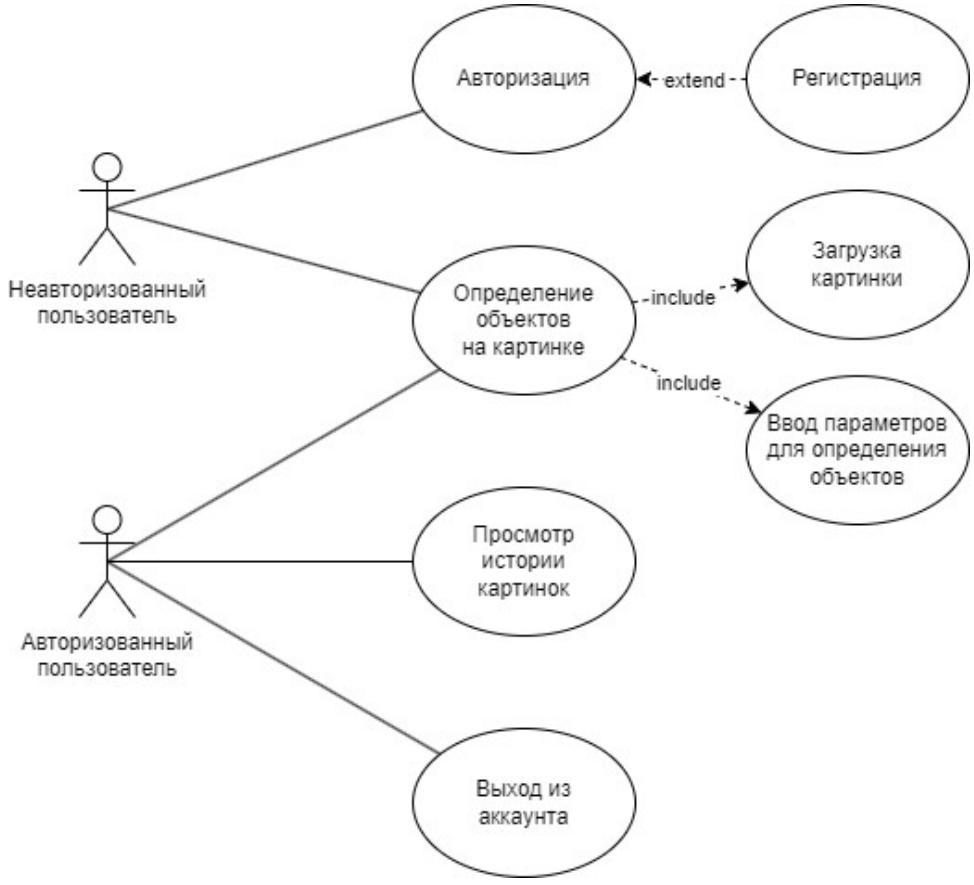

Рисунок 4 - Диаграмма прецедентов (Use case)

2.3.2 Диаграмма последовательности (Sequence diagram)

Существует также диаграмма последовательностей (Рисунки 5-6), на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта и взаимодействие актеров информационной системы в рамках прецедента [3]. Участником данной системы является пользователь, а объектами – клиент, сервер и база данных.

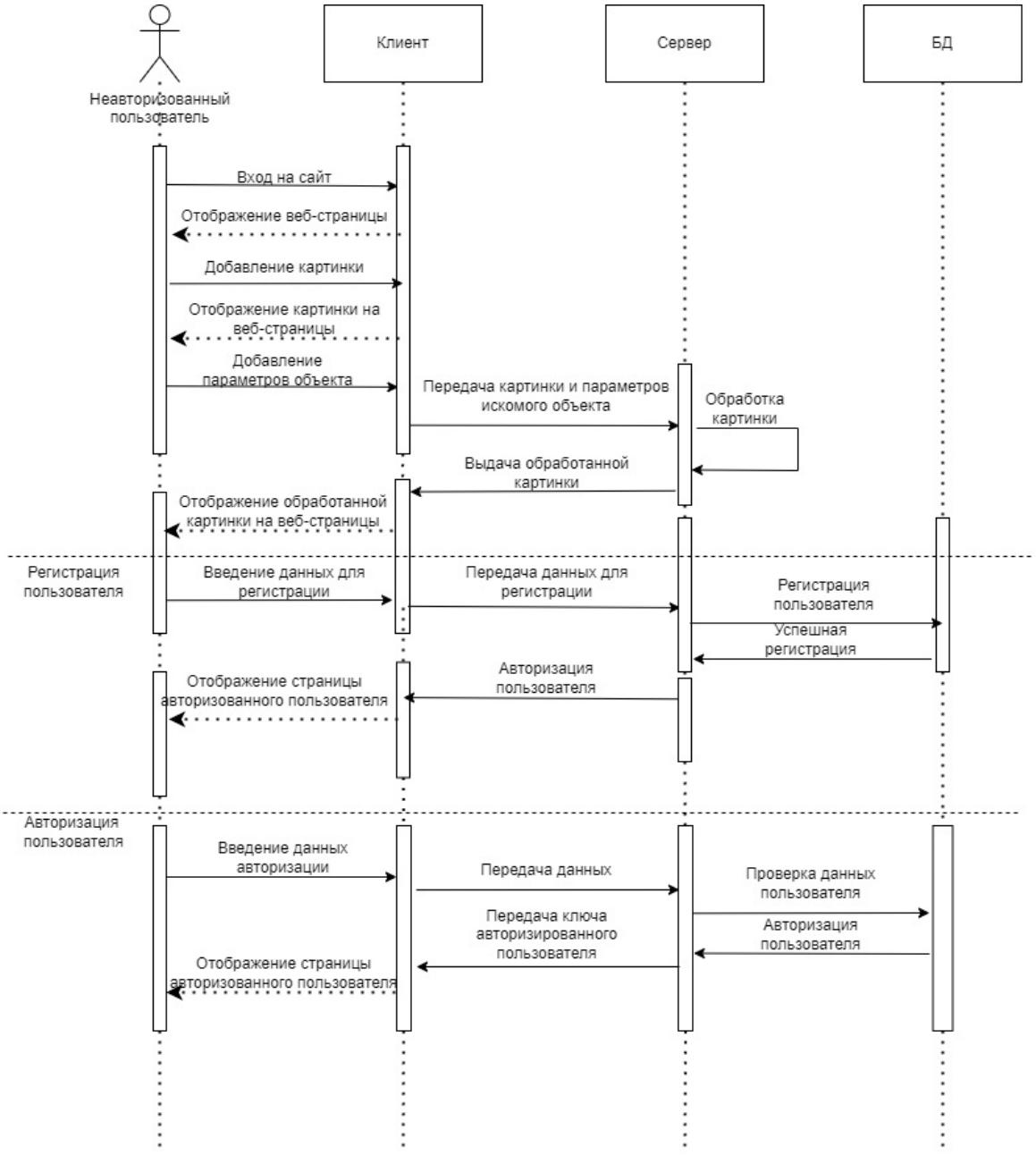

Рисунок 5 - Диаграмма последовательности для неавторизованного пользователя

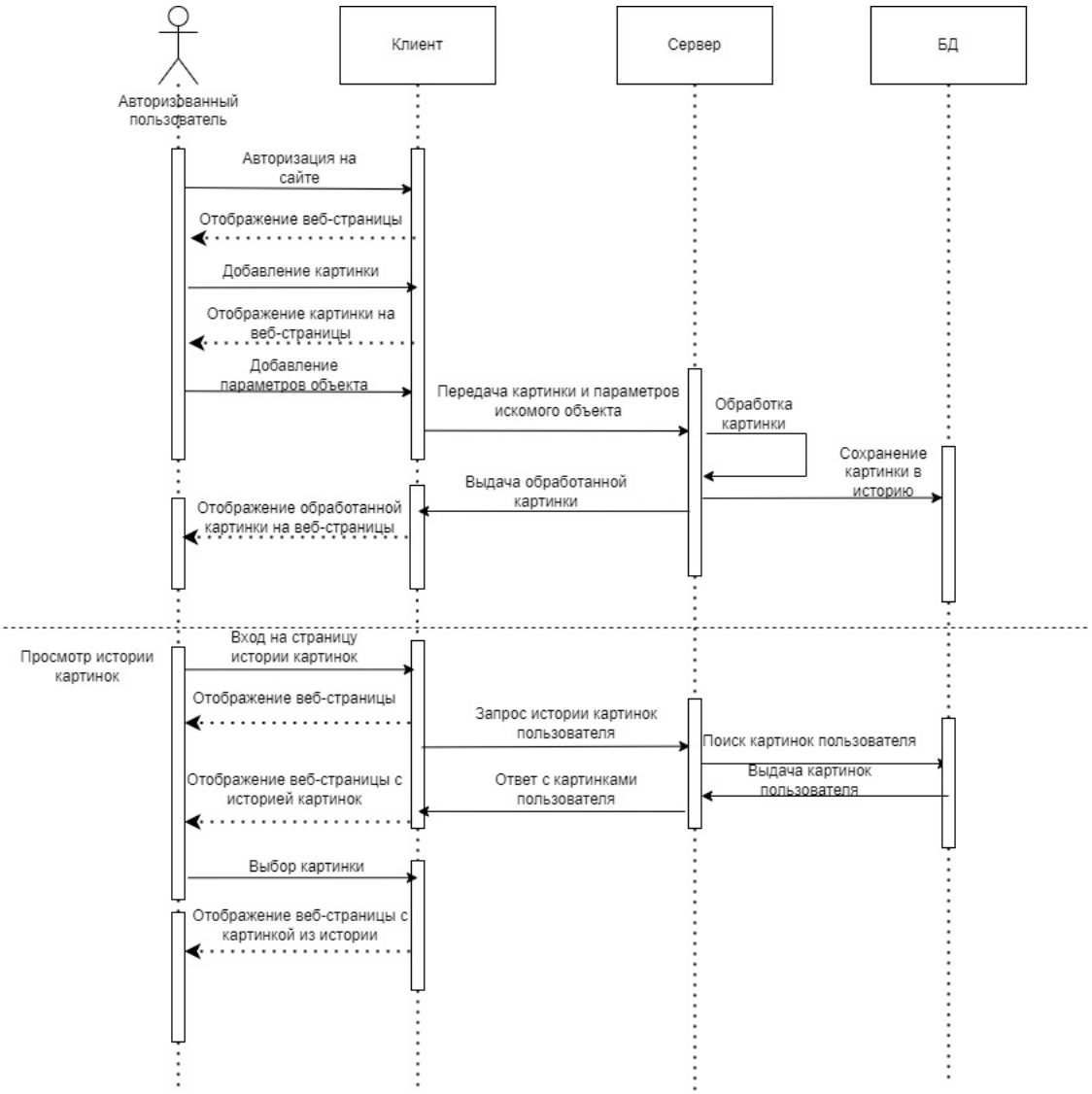

Рисунок 6 - Диаграмма последовательности для авторизованного пользователя

2.3.3 Диаграмма состояний (Statechart diagram)

Диаграмма состояний (Рисунок 7) отражает внутренние состояния объекта в течение его жизненного цикла от момента создания до разрушения [3]. На данной диаграмме рассмотрены состояния от момента входа в систему до полного выхода из нее.

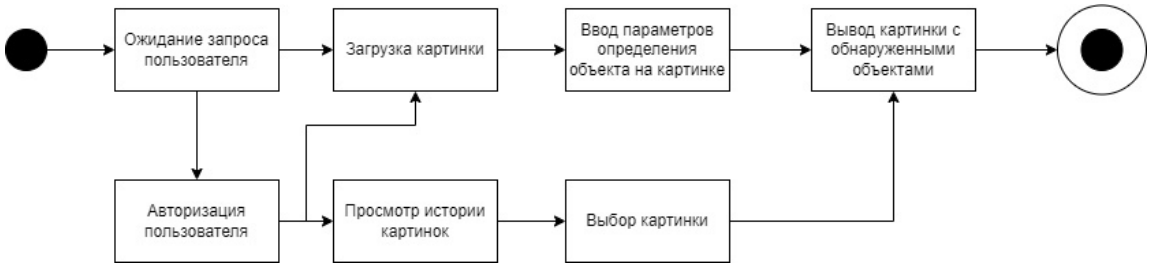

Рисунок 7 - Диаграмма состояний

2.3.4 Диаграмма активности (Activity diagram)

Диаграмма активности (Рисунок 8) представляет собой диаграмму, на которой показаны действия, состояния которых описаны на диаграмме состояний. Она описывает действия системы или людей, выполняющих действия, и последовательный поток этих действий [3].

Диаграмма показывает, что пользователь, находясь в неавторизованной зоне системы может обнаруживать объекты на изображении и авторизоваться, но не может просматривать историю изображений.

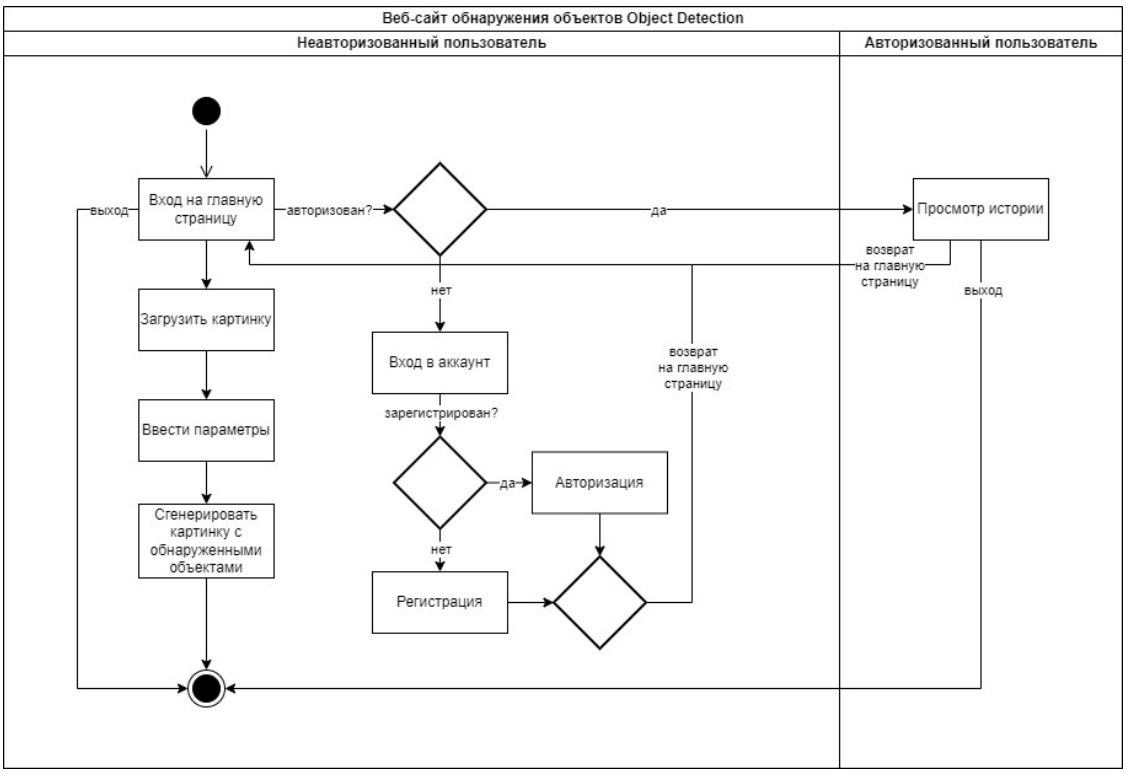

Рисунок 8 - Диаграмма активности

2.3.5 Диаграмма классов (Class diagram)

Диаграмма классов (Рисунок 9) демонстрирует общую структуру иерархии классов системы, их кооперацией, атрибутов, методов, интерфейсов и взаимосвязей между ними. В данной системе рассмотрены следующие классы:

— класс «Пользователь»;

— класс «Изображение»;

— класс «Ограничивающая рамка».

У каждого из классов существуют свои атрибуты.

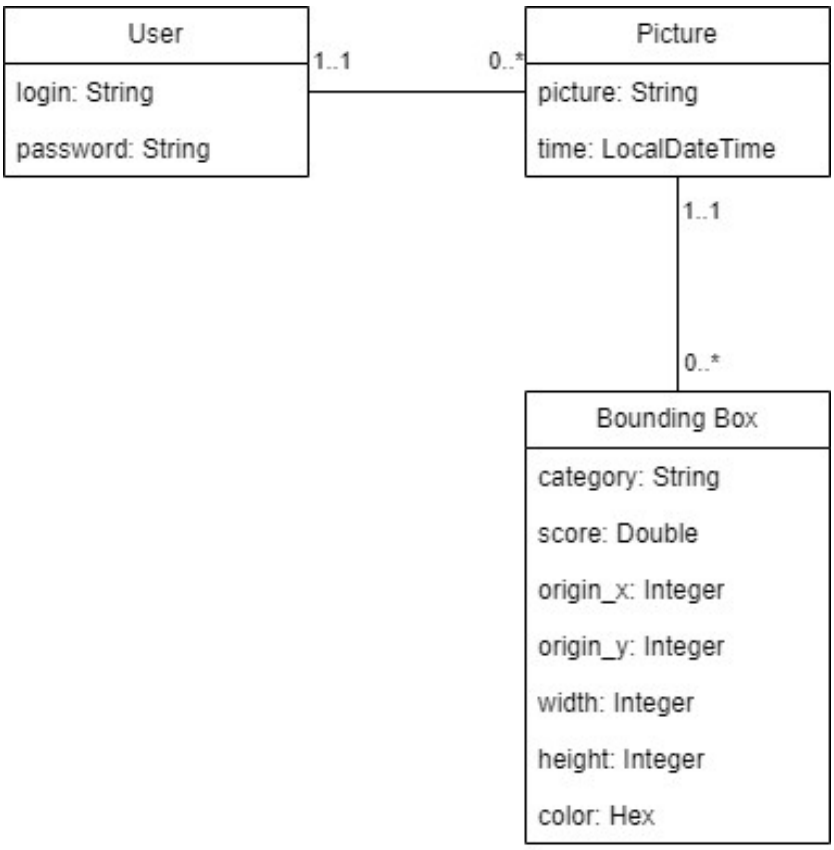

Рисунок 9 - Диаграмма классов

2.3.6 Диаграмма объектов (Object diagram)

По подобию диаграммы классов была построена диаграмма объектов. (Рисунок 10).

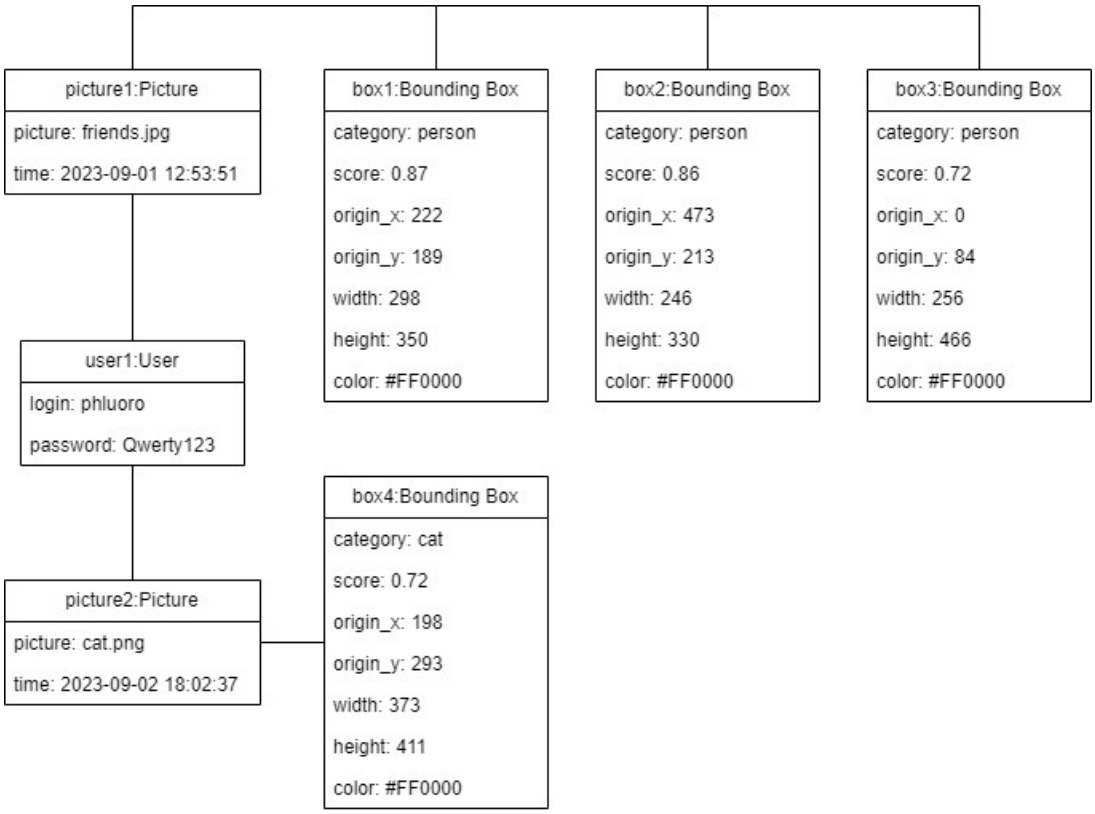

Рисунок 10 - Диаграмма объектов

2.3.7 Диаграмма развертывания (Deployment diagram)

Диаграмма развертывания (Рисунок 11) предназначена для представления общей конфигурации или топологии распределенной программной системы [3].

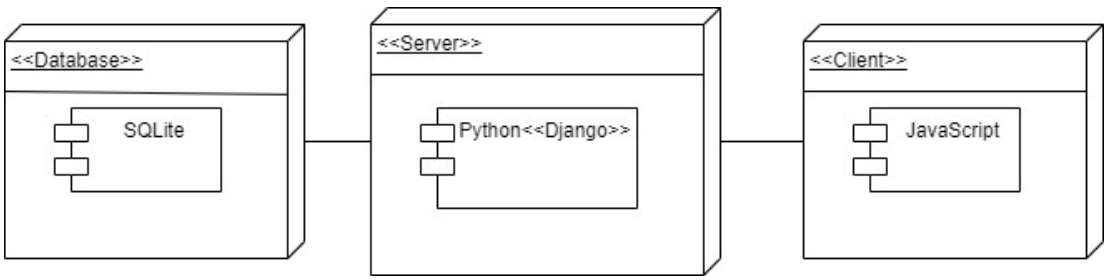

Рисунок 11 - Диаграмма развертывания

2.3.8 Диаграмма сотрудничества (Collaboration diagram)

Диаграмма сотрудничества (Рисунки 12-15) — это вид диаграммы взаимодействия, в котором основное внимание сосредоточено на структуре взаимосвязей объектов, принимающих и отправляющих сообщения [4].

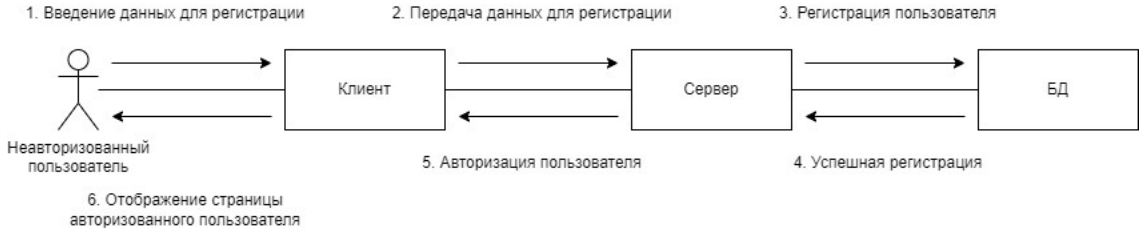

Рисунок 12 - Диаграмма сотрудничества при авторизации

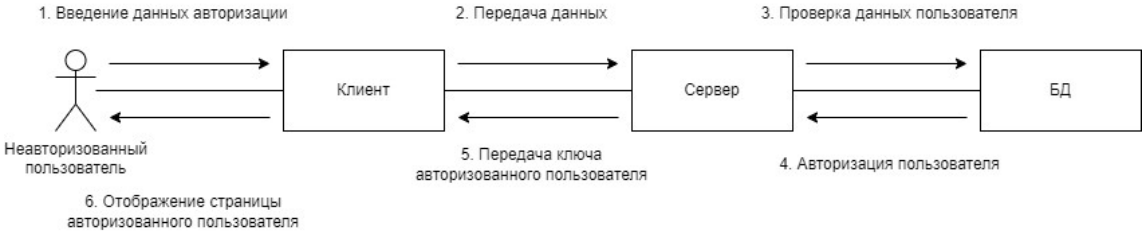

Рисунок 13 - Диаграмма сотрудничества при регистрации

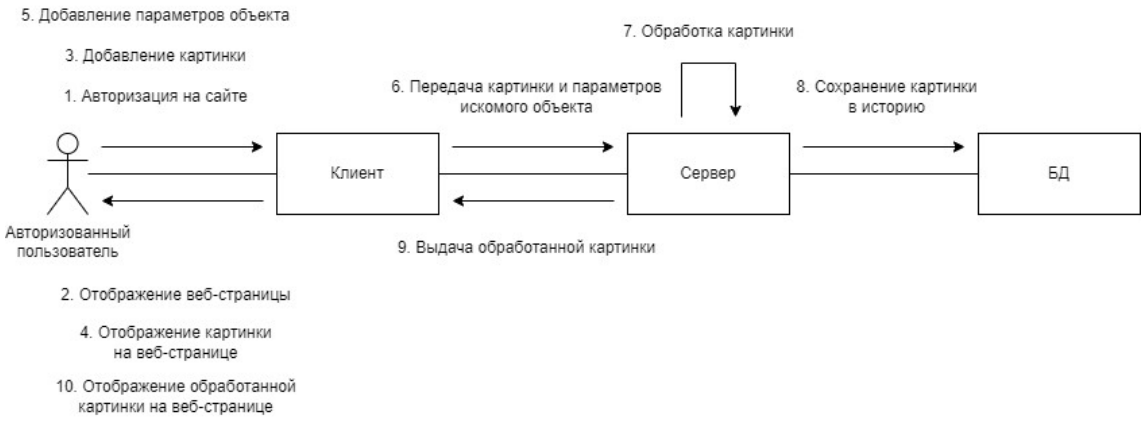

Рисунок 14 - Диаграмма сотрудничества при обнаружении объектов

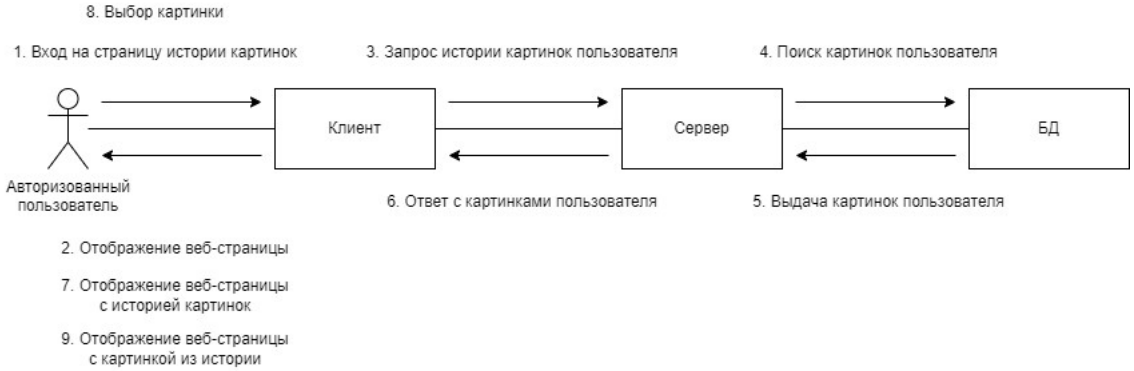

Рисунок 15 - Диаграмма сотрудничества при просмотре истории

2.3.9 Диаграмма IDEF0

IDEF0 используется для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальные объекты, связывающие эти функции.

На Рисунке 16 представлена контекстная диаграмма системы. На вход системе поступает пользователь и изображение. Работу системы регулирует законодательство РФ. Как ресурсы, необходимые для работы системы, в неё поступает сайт. На выходе системы мы имеем удовлетворённого пользователя, изображение с классифицированными объектами и список обработанных изображений. Далее представлена декомпозиция диаграммы по уровням (Рисунки 17-18).

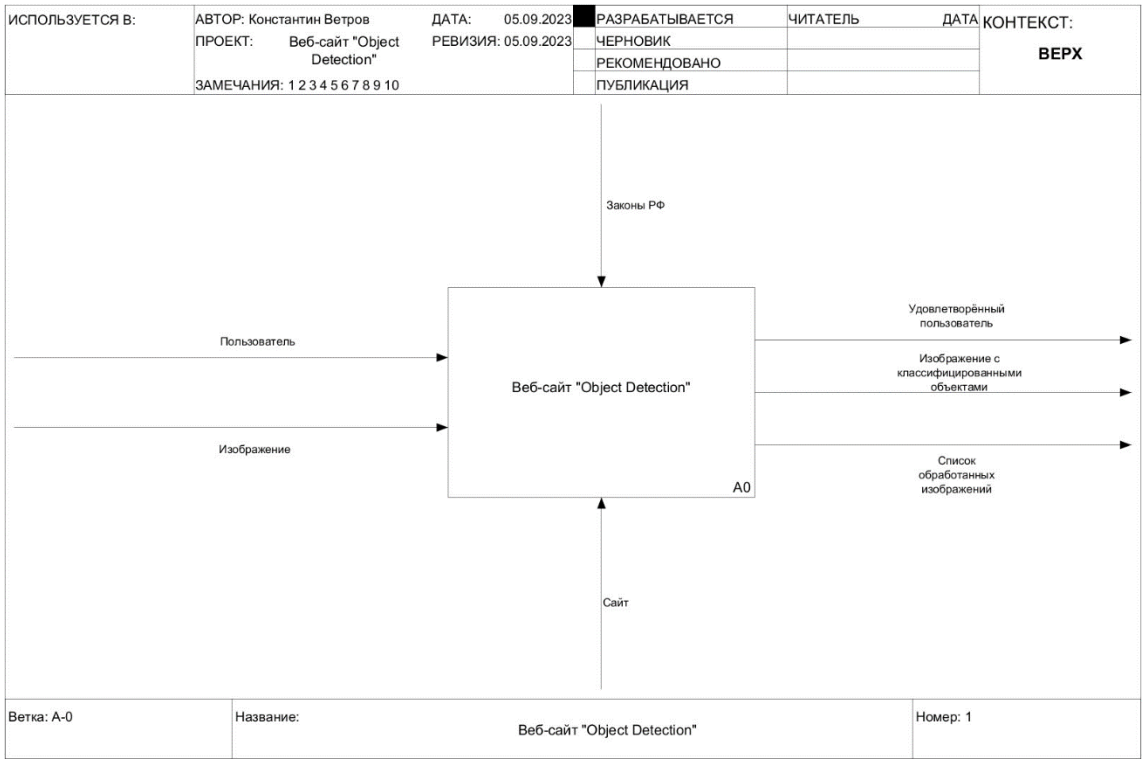

Рисунок 16 - Контекстная диаграмма системы

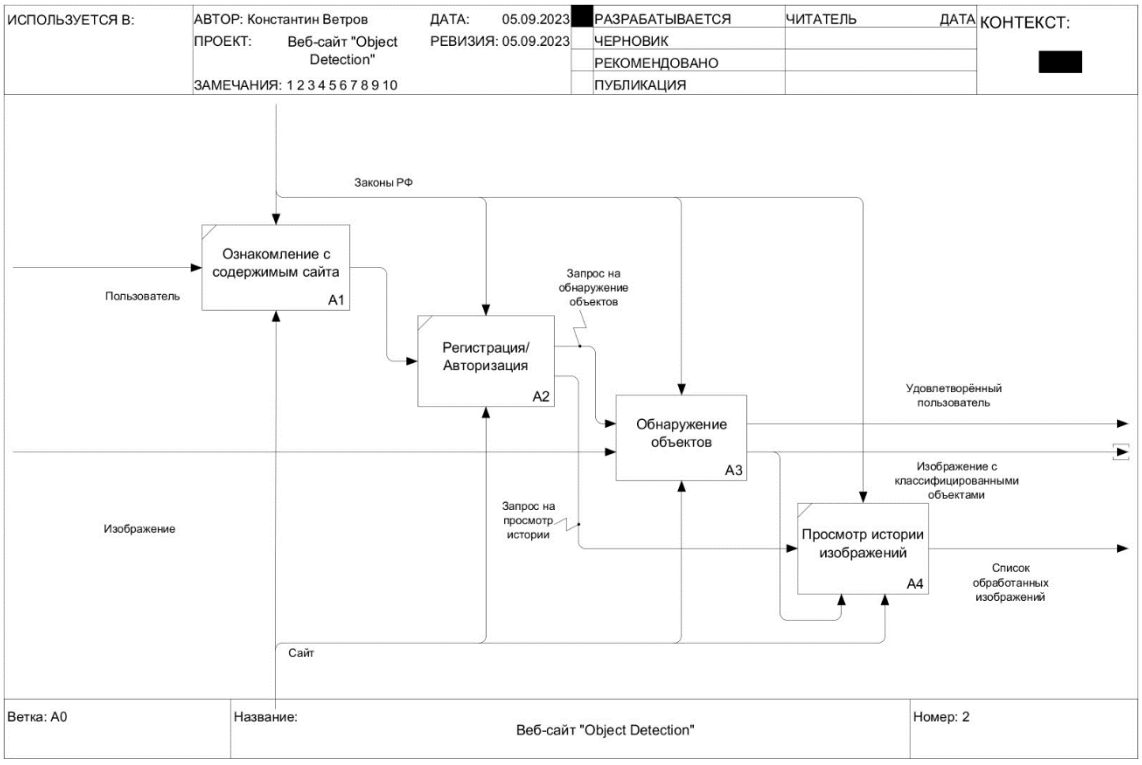

Рисунок 17 - Декомпозиция работы веб-сайта

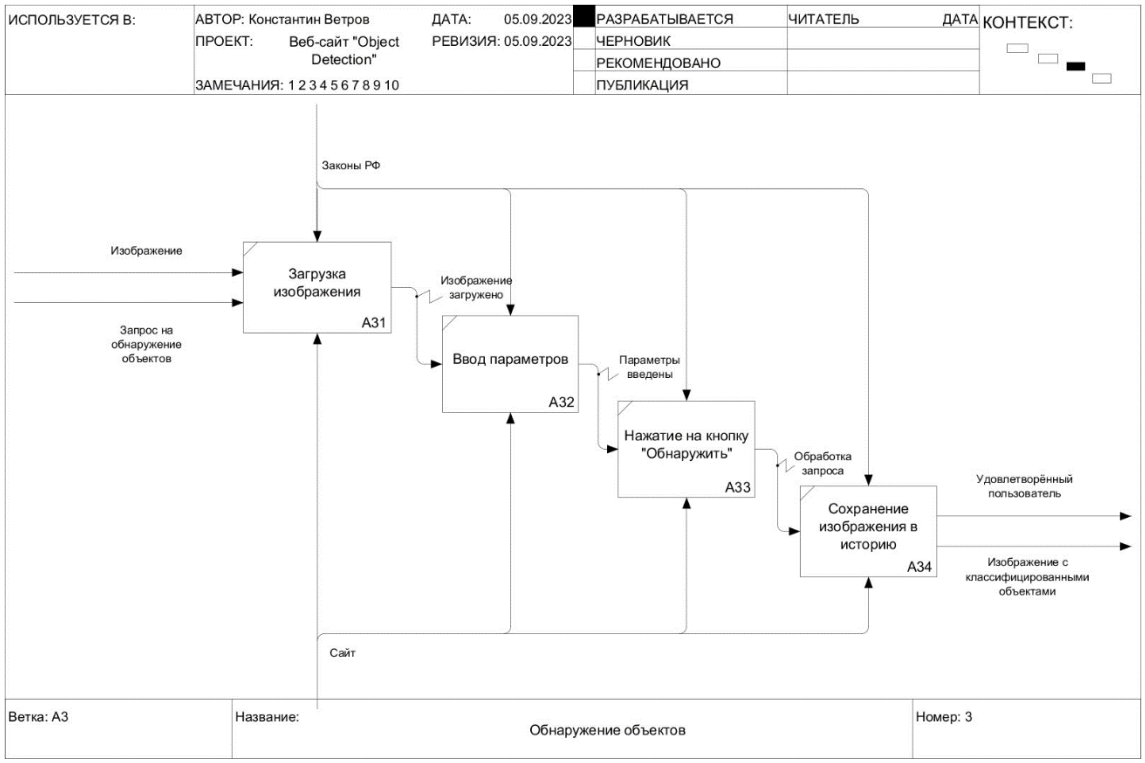

Рисунок 18 - Декомпозиция обнаружения объектов

2.3.10 ER-диаграмма

ER-диаграмма — это графическое представление модели данных, которая используется для описания концептуальной структуры базы данных. В такой диаграмме есть сущности, которые представляют объекты, с которыми работает система, и связи между сущностями, описывающие их взаимодействия. ER-диаграмма помогает наглядно описать структуру базы данных и увидеть связи между ее элементами (Рисунок 19).

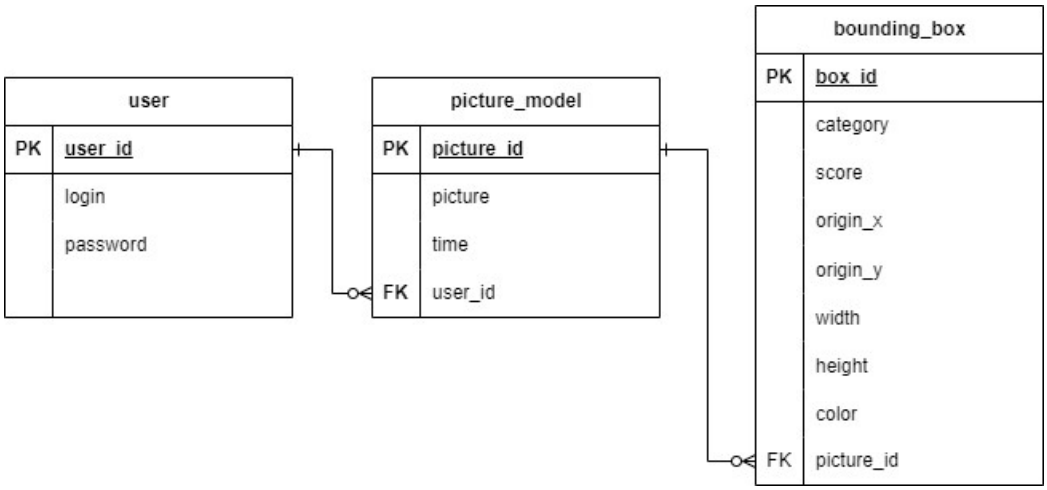

Рисунок 19 - ER-диаграмма

3 Реализация

3.1 Средства реализации

3.1.1 Средства реализации серверной части приложения

Для разработки серверной (backend) части приложения был выбран следующий стэк технологий:

— Django — это высокоуровневый веб-фреймворк на языке Python. Он предоставляет набор инструментов и библиотек для упрощения и автоматизации различных аспектов веб-разработки, таких как работа с базами данных, обработка форм, работа с шаблонами, управление сессиями пользователей и многие другие функции. Он является открытым и бесплатным инструментом, доступным каждому разработчику;

— SQLite — это быстрая и легкая встраиваемая однофайловая СУБД на языке C. Хранит данные в локальном файле, не требует отдельного сервера для выполнения запросов или управления данными: вместо этого она использует библиотеку, которая работает внутри приложения. SQLite можно использовать для мобильных, настольных и веб-приложений;

— Swagger — инструмент для документирования и тестирования API. Он позволяет создавать интерактивную документацию для вебсервисов, что упрощает их использование и интеграцию. Swagger автоматически генерирует документацию на основе аннотаций и комментариев в коде, что позволяет разработчикам сосредоточиться на написании логики приложения, а не на создании и поддержке документации. Благодаря Swagger, разработчики могут изучить доступные параметры, модели данных и примеры запросов и ответов.

3.1.2 Средства реализации клиентской части приложения

Для разработки клиентской (frontend) части приложения был выбран следующий стэк технологий:

— JavaScript — это один из наиболее популярных языков программирования, который используется для разработки веб-приложений, игр, мобильных приложений и других приложений. JavaScript является интерпретируемым языком скриптов, то есть он выполняется в среде браузера, что позволяет создавать динамические и интерактивные веб-сайты, а также управлять внешним поведением страницы. Помимо этого, он обладает следующими преимуществами: широкая поддержка, гибкость, доступ к различным библиотекам и фреймворкам;

— CSS (Cascading Style Sheets) — это язык стилей, используемый для задания внешнего вида веб-страниц. CSS позволяет разработчикам отделить визуальное представление веб-сайта от содержания, тем самым обеспечивая большую гибкость и управляемость веб-ресурсов. С помощью CSS можно задавать шрифты, цвета, расположение, размеры элементов, оформление фона и др. CSS является основным инструментом для создания визуального дизайна веб-сайтов и позволяет создавать уникальные дизайны и согласовывать внешний вид контента на всем сайте;

— HTML (HyperText Markup Language) — это язык разметки, используемый для создания веб-страниц. С помощью HTML разработчики определяют структуру и содержание веб-страниц, позволяя браузеру правильно интерпретировать и отображать контент для пользователей. HTML использует теги для форматирования текста, вставки изображений, оформления списков, аудио и видео, ссылок и многого другого. Он обладает следующими преимуществами: читаемость для разработчиков, возможность создания структурированного контента с использованием семантических элементов, доступность и адаптивности для различных устройств и браузеров. HTML является основным языком для создания веб-страниц и работает в сочетании с CSS для

определения внешнего вида и JavaScript для добавления интерактивности.

3.2 Реализация серверной (backend) части приложения

Серверная (backend) часть приложения была написана на языке Python с использованием фреймворка Django. Структура проекта представляет собой корневую папку object_detection и дополнительные — main и object_detection (Рисунок 20).

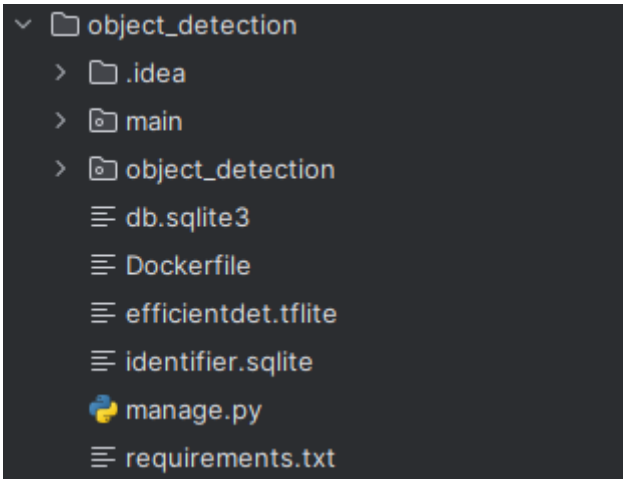

Рисунок 20 - Структура проекта

В корневой папке (Рисунок 21) особое внимание стоит обратить на файл settings.py, который содержит настройки проекта. Он содержит множество параметров для настройки работоспособности приложения.

- DEBUG — параметр, отвечающий за включение/выключение режима отладки;
- SECRET_KEY — секретный ключ приложения, который применяется для генерации токенов, а также для шифрования паролей пользователей;
- DATABASES — настройки подключения к базе данных, в которой хранятся данные приложения;
- INSTALLED_APPS — список приложений, установленных в проекте. Здесь указываются все приложения, которые будут использованы в проекте;

— MIDDLEWARE — список всех промежуточных компонентов, используемых в Django для обработки запросов и ответов между сервером и клиентом;
— TEMPLATES — настройки для генерации HTML-шаблонов, используемых для визуализации данных пользователя в БД;
— AUTH_PASSWORD_VALIDATORS — список компонентов, используемых для проверки безопасности паролей пользователей.

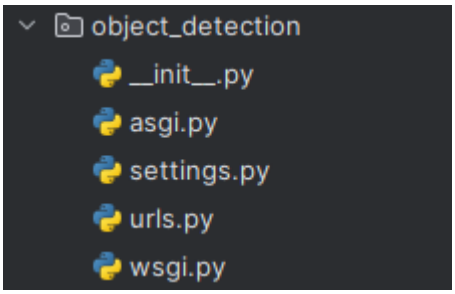

Рисунок 21 - Структура корневой папки проекта

Так же стоит обратить внимание на файл urls.py — это механизм маршрутизации запросов на определенные представления. Django использует файл urls.py для определения соответствующей представлению URL-адреса.

При этом фреймворк Django подразумевает разделение приложения на несколько основных компонентов (Рисунок 22).

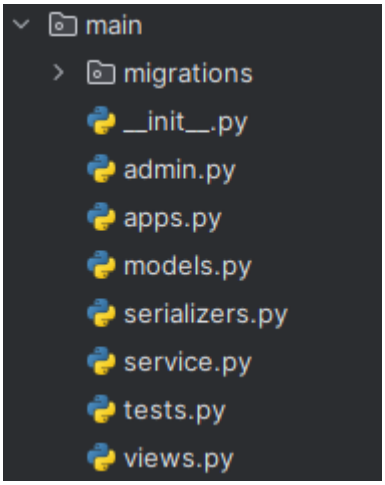

Рисунок 22 - Основные компоненты Django

— инициализация (__init__.py) — это специальный файл в Python, определяющий пакет. В Django он используется для определения пакета для приложения;
— административная панель (admin.py) — это административная панель Django, которая обеспечивает автоматическую генерацию форм и предоставляет удобный интерфейс для работы с данными;
— конфигурационный файл (apps.py) — это Python файл, который определяет основные настройки для приложения. Он содержит метаданные, относящиеся к данному приложению, и позволяет настроить приложение в соответствии с требованиями и лучше контролировать его работу;
— модели данных (models.py) — это классы Python, которые определяют структуру базы данных и могут быть использованы для создания или обновления схемы базы данных. Модели могут содержать поля для хранения данных (текстовые, числовые, даты, файлы и др.), а также методы для работы с этими данными;
— сериализация (serializers.py) — файл, обеспечивающий сериализацию и десериализацию данных, передаваемых через приложение. Он используется в Django для работы с данными, передаваемыми через HTTP в API-модулях. Файл содержит классы сериализаторов, которые определяют, какие поля модели должны быть преобразованы в JSON, XML и т.д. Они также могут обеспечивать валидацию данных во время десериализации;
— тесты (test.py) — это файл в приложении, который содержит модульные тесты для этого приложения. Эти тесты используются, чтобы убедиться, что функциональные возможности приложения работают должным образом, и выявить любые ошибки или ошибки до того, как приложение будет развернуто в рабочей среде;

— представления (views.py) — это функции Python, которые обрабатывают запросы от клиента и возвращают HTTP-ответы; представления могут включать в себя логику приложения, обработку данных из модели, взаимодействие с другими системами.

Серверная часть приложения Django также может использовать множество дополнительных библиотек и компонентов.

Сервер был развернут на виртуальной машине с помощью системы контейнеризации Docker. Этот процесс был выполнен с использования системы контроля версий Git. После загрузки потребовалось настроить переменные окружения, представляющие собой конфиденциальные настройки, такие как данные аутентификации для базы данных. Клиентская часть была загружена в другом Docker-контейнере.

Для описания спецификации API использовался Swagger [5]. Для того чтобы интегрировать его в проект, нужно было внести изменения в код, включая добавление необходимой зависимости (drf_yasg), добавить ее в INSTALLED_APPS и описать спецификации всех методов.

3.3 Реализация клиентской (frontend) части приложения

Клиентская часть приложения (frontend) разработана на языке JavaScript. Структура проекта представляет собой корневую папку od-frontend в которой находятся разметка страниц (html) и их логика (js), а так же стили в папке styles (css) (Рисунок 23).

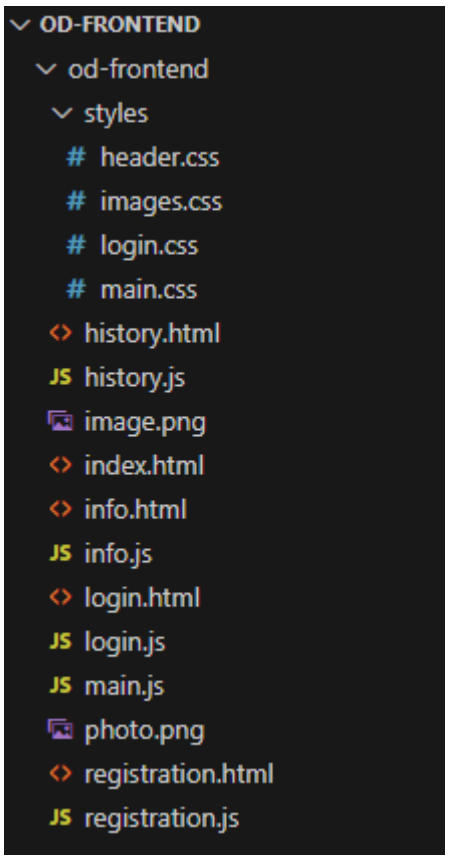

Рисунок 23 - Основные компоненты клиентской части

— history.html — разметка страницы истории картинок;

— history.js — логика страницы истории картинок;

— image.png — это изображение для страницы с описанием сайта;

— index.html — разметка главной страницы;

— info.html — разметка страницы с описанием сайта;

— info.js — логика страницы с описанием сайта;

— login.html — разметка страницы авторизации;

— login.js — логика страницы авторизации;

— main.js — логика главной страницы;

— registration.html — разметка страницы регистрации;

— registration.js — логика страницы регистрации.

В папке styles находятся стили для компонентов программы (Рисунок 24).

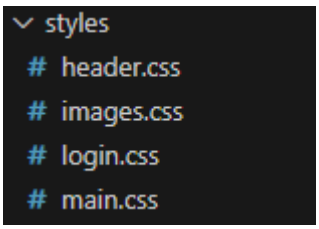

Рисунок 24 - Содержание папки стилей

— header.css — определяет стили для навигационной панели;

— images.css — определяет стили для картинок;

— login.css — определяет стили для страницы авторизации;

— main.css — определяет стили для главной страницы.

3.4 Навигация по приложению

3.4.1 Для неавторизованного пользователя

Первое, что видит пользователь — главный экран (Рисунок 25).

Далее у пользователя возникает выбор: он может войти в личный кабинет при помощи авторизации (Рисунок 26), если нет аккаунта – при помощи регистрации (Рисунок 27) или продолжить пользоваться сайтом в режиме неавторизованного пользователя, но с ограниченной функциональностью (без возможности просмотра истории запросов обработки изображений).

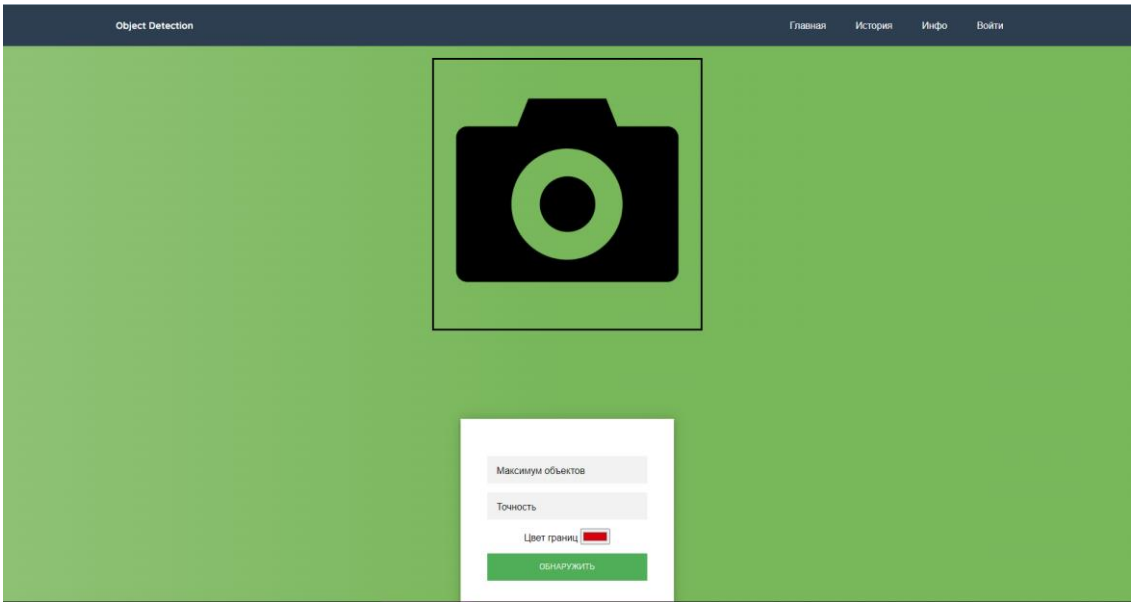

Рисунок 25 - Главная страница

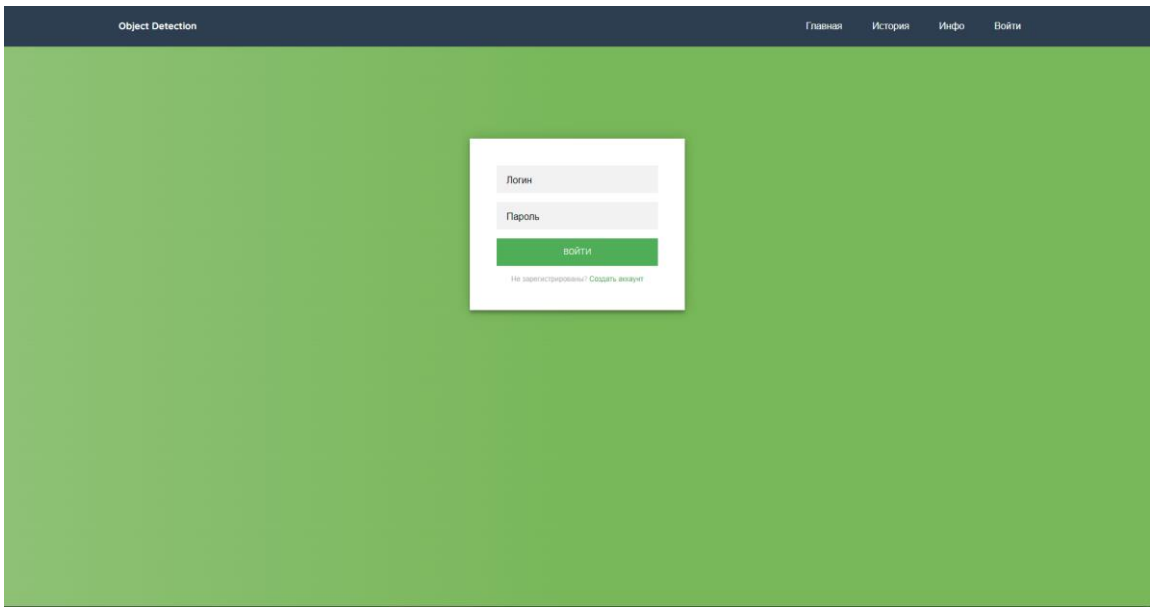

Рисунок 26 - Страница авторизации

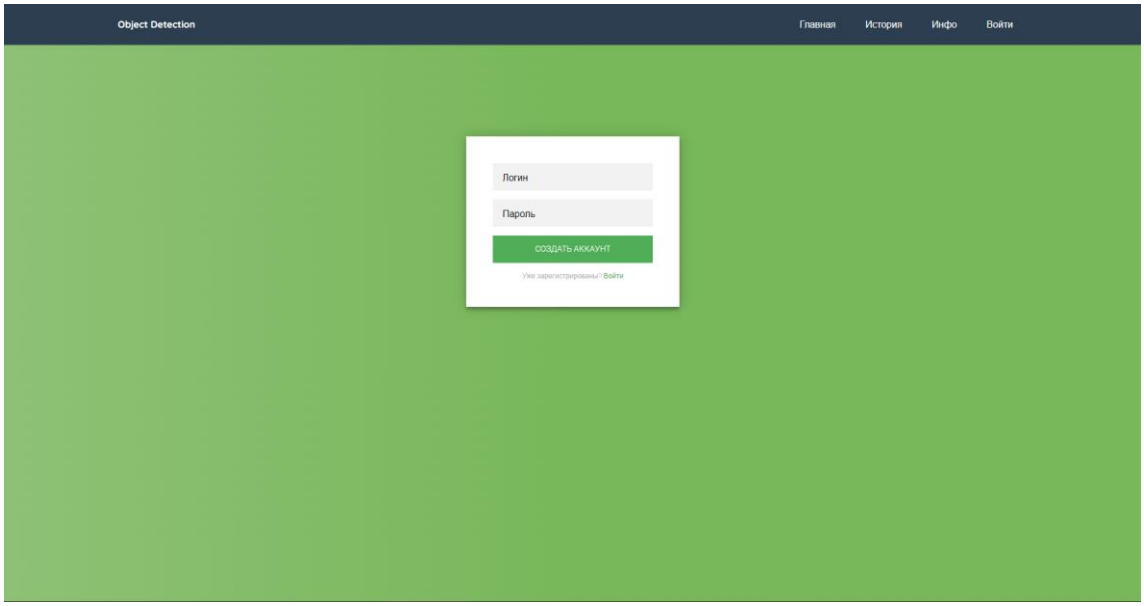

Рисунок 27 - Страница регистрации

Неавторизованному пользователю доступна главная страница, на которой присутствует возможность использования основной функцией веб-приложения (использование технологии Object Detection) (Рисунок 25).

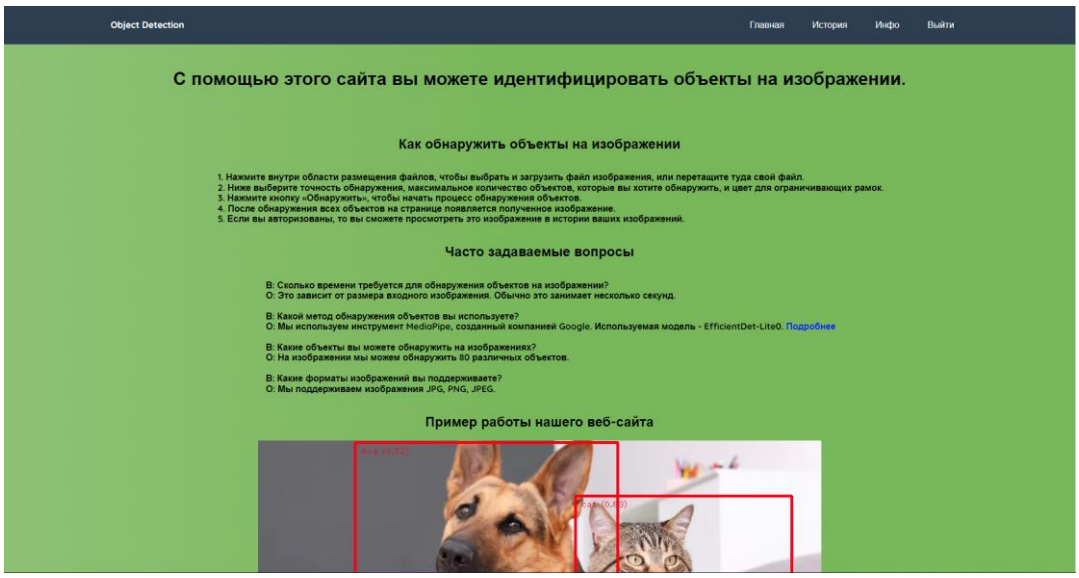

Рисунок 28 - Страница информации

Помимо этого, можно перейти на страницу информации, где наглядно показывается как использовать систему обнаружения объектов, а также ответы на часто задаваемые вопросы (Рисунок 28).

3.4.2 Для авторизованного пользователя

После авторизации пользователю становится доступной возможность просмотра истории запросов обработки изображений. Для этого на главной странице необходимо нажать на кнопку «История» (Рисунок 29).

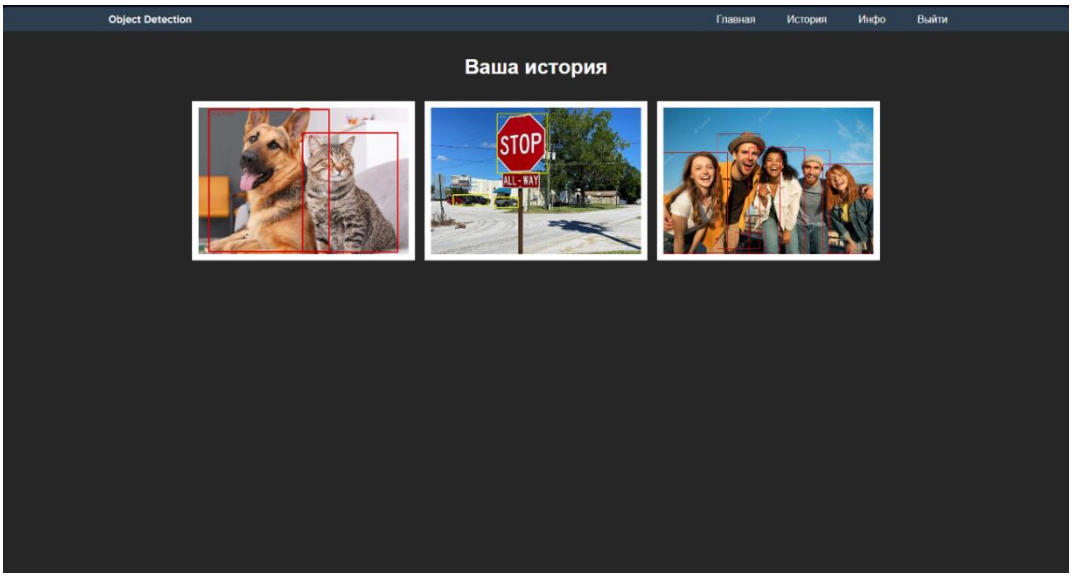

Рисунок 29 - Страница истории

4 Тестирование

4.1 Дымовое тестирование

Дымовое тестирование – это форма предварительного тестирования программного продукта или системы с целью проверки базовой функциональности и выявления критических ошибок. Оно выполняется перед более полным и подробным тестированием для обнаружения серьезных проблем, которые могут препятствовать дальнейшему тестированию или работы продукта.

Дымовое тестирование включает в себя выполнение небольшого набора тестовых сценариев или функциональных проверок, которые охватывают основные компоненты, функции и возможности программного продукта. Целью дымового тестирования является убедиться, что основные функции продукта работают стабильно и не возникают критические проблемы.

Если при дымовом тестировании обнаруживаются ошибки или неполадки, то это может стать причиной отклонения продукта от плана и дальнейшего детального тестирования до устранения проблем. Дымовое тестирование помогает предотвратить возможные задержки и проблем. Далее представлены результаты дымового тестирования для основных сценариев неавторизованного, авторизованного пользователей и администратора (Таблица 1-2).

Таблица 1 - Результаты дымового тестирования для неавторизованного пользователя

Тестовый сценарий	**Результат теста**
Регистрация	Пройден
Авторизация	Пройден
Осуществление детекции объектов	Пройден
Просмотр информации	Пройден

Таблица 2 - Результаты дымового тестирования для авторизованного пользователя

Тестовый сценарий	**Результат теста**
Регистрация	Пройден
Авторизация	Пройден
Просмотр истории	Пройден
Осуществление детекции объектов	Пройден
Просмотр информации	Пройден

Заключение

В ходе выполнения курсовой работы была выполнены поставленные задачи. Было разработано веб-приложение, поддерживающее технологию Object Detection. Оно предоставляет пользователям использовать данную систему и соответственно возможность обнаружения и классификации объектов на загруженных изображениях. А также просмотр истории ранее обработанных изображений авторизованным пользователем.

В начале разработки был проведен анализ предметной области, определены основные требования к разрабатываемой системе, определены основные сценарии веб-приложения.

Таким образом, итоги разработки, проверенные в ходе тестирования, позволяют достигнуть поставленных заказчиком целей и решают сформулированные в начале разработки задачи.

Список использованной литературы

1. Django - что это за фреймворк на Python: возможности и преимущества [Электронный ресурс]. — Режим доступа: https://blog.skillfactory.ru/glossary/django/.— Заглавие с экрана. — (Дата обращения: 21.08.2023).

2. Aspose API формата файла High Code [Электронный ресурс]. — Режим доступа: https://www.aspose.com/ru/. — Заглавие с экрана. — (Дата обращения: 21.08.2023).

3. Полное руководство по 14 типам диаграмм UML [Электронный ресурс]. — Режим доступа: https://www.cybermedian.com/ru/a-comprehensive-guide-to-14-types-of-uml-diagram/. — Заглавие с экрана. — (Дата обращения: 24.08.2023).

4. Диаграммы сотрудничества [Электронный ресурс]. — Режим доступа: https://helpiks.org/9-53448.html. — Заглавие с экрана. — (Дата обращения: 24.08.2023).

5. Тестирование API с помощью Swagger: особенности и преимущества [Электронный ресурс]. — Режим доступа: https://blog.ithillel.ua/ru/articles/apitesting-with-swagger. — Заглавие с экрана. — (Дата обращения: 28.08.2023).

6. Всё про дымовое тестирование для начинающего тестировщика. 2022 [Электронный ресурс]. — Режим доступа: https://vc.ru/u/1263669-vasiliy-volgin/570260-vse-pro-dymovoe-testirovanie-dlya-nachinayushchego-testirovshchika-2022.— Заглавие с экрана. — (Дата обращения: 29.08.2023).