ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

СОГЛАСОВАНО
Руководитель курсового проекта, Senior MLE в Digital Finance International

_____ В. А. Ахмедов
«__» _____2025 г.

УТВЕРЖДАЮ
Академический руководитель образовательной программы «Программная инженерия» старший преподаватель департамента программной инженерии

_____ Н. А. Павлочев
«__» _____2025 г.

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	

Приложение для управления фотоархивом с интеллектуальным поиском изображений

Пояснительная записка

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.10.03-01 ТЗ 01-1-ЛУ

Исполнитель:
студент группы БПИ2310
_____ Безруков Г.А.
« __ » _____ 2025 г.

2024

УТВЕРЖДЁН

RU.17701729.05.05-01 51 01-1-ЛУ

Приложение для управления фотоархивом с интеллектуальным поиском изображений

Пояснительная записка
RU.17701729.05.05-01 ТЗ 01-1
Листов: 41

Инв. № подл	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата
RU.17701729.05.05-01 ТЗ 01-1-ЛУ				

2024

АННОТАЦИЯ

Настоящий документ является пояснительной запиской к программному продукту **«Приложение для управления фотоархивом с интеллектуальным поиском изображений» (Smart Gallery)** и включает следующие разделы «Введение», «Назначение и область применения», «Технические характеристики», «Ожидаемые технико-экономические показатели» и приложения.

В разделе *«Введение»* указано наименование программного продукта, цели разработки, а также документальное основание для выполнения работ.

В разделе *«Назначение и область применения»* определено функциональное и эксплуатационное назначение приложения, а также его предполагаемая сфера использования.

Раздел *«Технические характеристики»* включает описание задачи, архитектурного решения, алгоритмов работы и организации хранения данных, обоснование выбора технологий и состава используемых технических и программных средств.

Раздел *«Ожидаемые технико-экономические показатели»* содержит сведения о целевой аудитории, предполагаемой пользе и отличиях от аналогов, а также анализ преимуществ разработки по сравнению с существующими решениями.

Документ составлен в соответствии с нормативными требованиями:

1) ГОСТ 19.101-77 Виды программ и программных документов [1].
2) ГОСТ 19.102-77 Стадии разработки [2].
3) ГОСТ 19.103-77 Обозначения программ и программных документов [3].
4) ГОСТ 19.104-78 Основные надписи [4].
5) ГОСТ 19.105-78 Общие требования к программным документам [5].
6) ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом [6].
7) ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению [7]

Перед прочтением настоятельно рекомендуется ознакомиться с глоссарием (см. Приложение 2).

СОДЕРЖАНИЕ

1. Введение

1.1.Наименование программы

Наименование программы – «Умная галерея».

Наименование программы на английском языке – «Smart Gallery».

1.2.Краткая характеристика области применения программы

«Умная галерея» — это приложение для локального и интуитивно понятного управления коллекциями изображений. Основной функционал включает текстовый поиск изображений с использованием нейросетевых эмбеддингов (например, по запросу «море»), просмотр и удаление фотографий. Приложение применяет технологии машинного обучения для анализа содержимого изображений и организации поиска. Удобный пользовательский интерфейс обеспечивает простой доступ к функциям даже при работе с большими объёмами данных. Вся обработка и хранение выполняются локально, что повышает уровень конфиденциальности и надёжности системы.

1. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

1.1. Назначение разработки

1.1.1. Функциональное назначение

Программный продукт **Smart Gallery** представляет собой систему управления изображениями, работающую в офлайн-режиме и включающую пользовательский интерфейс, backend и ML-сервис. Система реализует следующие функциональные возможности:

1. Загрузка изображений пользователем;
2. Автоматическая генерация предпросмотров и эмбеддингов изображений;
3. Хранение изображений в объектном хранилище (MinIO), метаданных — в PostgreSQL, эмбеддингов — в Qdrant;
4. Поиск изображений по текстовому запросу с использованием модели ruCLIP;
5. Просмотр изображений в галерее;
6. Удаление изображений;
7. Взаимодействие между компонентами через REST API.

1.1.2. Эксплуатационное назначение

Приложение ориентировано на пользователей, работающих с большими коллекциями изображений в условиях ограниченного или отсутствующего доступа к сети Интернет. Система обеспечивает локальную работу, высокую конфиденциальность и интуитивно понятный интерфейс. Она предназначена как для индивидуальных пользователей (например, фотографов, дизайнеров), так и для организаций, которым требуется удобный инструмент для хранения, сортировки и поиска изображений на локальных устройствах.

Система запускается в контейнеризированной среде (Docker) и доступна через веб-браузер. Все данные обрабатываются локально, что исключает необходимость передачи информации во внешние сервисы

1.2. Область применения

Программный продукт **Smart Gallery** может использоваться:

- фотографами, дизайнерами и блогерами для хранения и интеллектуального поиска изображений.
- в исследовательских институтах и лабораториях для работы с визуальными архивами.

- в организациях с повышенными требованиями к автономности и конфиденциальности (например, в закрытых сетях);
- в образовательных и культурных учреждениях, где необходимо управление цифровыми фотоархивами без доступа к облачным решениям

Благодаря модульной архитектуре, система может быть адаптирована под конкретные задачи, расширена или интегрирована с другими решениями, работающими по REST API

2. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

2.1. Постановка задачи на разработку программы

Разрабатываемый программный продукт **Smart Gallery** представляет собой кроссплатформенное приложение для управления изображениями с возможностью локального хранения, интеллектуального поиска и интуитивно понятного взаимодействия. Система предназначена для автономной работы без подключения к интернету и развёртывается в контейнеризированной среде:

В рамках проекта реализуются следующие компоненты:

- серверная часть (backend) на FastAPI;
- ML-сервис для генерации эмбеддингов изображений и текстов (на базе ruCLIP);
- пользовательский интерфейс на Flet.

Программа должна обеспечивать выполнение следующих задач:

1. Загрузка изображений

Пользователь загружает одно или несколько изображений через интерфейс. При этом система:

- сохраняет оригинал изображения в объектное хранилище MinIO;
- создаёт и сохраняет уменьшенную копию (превью);
- отправляет изображение в ML-сервис для генерации эмбеддинга;
- сохраняет метаданные (размер, дата, путь) в PostgreSQL, эмбеддинг — в Qdrant.

2. Просмотр изображений

Пользователь может:

- просматривать изображения в виде сетки предпросмотров;
- открывать каждое изображение в отдельном окне;
- получать информацию о дате загрузки, размере и пути к файлу.

3. Поиск изображений по текстовому запросу

После ввода текстового описания (например, «море», «вечеринка»):

- ML-сервис генерирует текстовый эмбеддинг;
- осуществляется сравнение с эмбеддингами изображений в Qdrant;
- возвращается список наиболее релевантных изображений.

4. Удаление изображений

Пользователь может выбрать одно или несколько изображений для удаления. Система:

- удаляет оригинал и предпросмотр из MinIO;
- удаляет соответствующий эмбеддинг из Qdrant;
- удаляет метаданные из PostgreSQL.

5. REST API

Backend предоставляет следующий набор REST-эндпоинтов:

- GET /images/ — получение всех изображений;
- POST /images/ — загрузка изображений;
- DELETE /images/ — удаление всех изображений;
- GET /images/search/{prompt} — поиск по текстовому описанию;
- GET /image/{image_id} — получение конкретного изображения;
- DELETE /image/{image_id} — удаление конкретного изображения;
- GET /health/ — проверка состояния сервера.

6. ML-сервис (CLIP API)

Отдельный ML-сервис предоставляет следующие функции:

- POST /embed/image — генерация эмбеддинга изображения;
- POST /embed/text — генерация эмбеддинга текста;
- GET /health/ — проверка доступности ML-сервиса.

7. Хранение данных

Программа использует следующие компоненты для хранения информации

- MinIO — объектное хранилище для изображений и превью;
- PostgreSQL — реляционная база для хранения метаданных;
- Qdrant — векторная база данных для хранения эмбеддингов;
- Все компоненты развёрнуты в контейнерах Docker и взаимодействуют внутри локальной сети.

8. Пользовательский интерфейс

Пользовательский интерфейс системы **Smart Gallery** реализован с использованием библиотеки **Flet** — кроссплатформенного инструмента на языке Python для построения интерактивных приложений. Фронтенд представляет собой локальное desktop/web-приложение, развёртываемое в контейнере Docker:

- **Многостраничную навигацию**:
  - главная страница с сеткой изображений (галерея);
  - страница просмотра изображения в полном размере;
  - режим поиска изображений по текстовому описанию;
  - режим удаления выбранных изображений.

- **Интерактивное взаимодействие с backend-сервисом**:
  - загрузка изображений;
  - отправка текстовых запросов для поиска;
  - удаление изображений;
  - отображение результата в реальном времени.

- **Локальный запуск в изолированной среде**:
  - приложение развёртывается в контейнере и не требует подключения к внешним ресурсам;
  - взаимодействие с backend и ML-сервисом осуществляется через внутреннюю сеть Docker.

2.2. Постановка задачи на разработку программы

2.2.1. Описание архитектуры системы

Программный продукт **Smart Gallery** реализован в виде модульной микросервисной архитектуры, включающей три основных компонента.

- **Frontend** — пользовательский интерфейс на Flet;
- **Backend** — серверная часть на FastAPI;
- **ML-сервис** — выделенный компонент для генерации эмбеддингов на базе модели ruCLIP.

Вся система разворачивается локально в контейнеризированной среде с использованием Docker и взаимодействует через REST API. Компоненты работают в изолированной сети и не требуют подключения к интернету.

Общая схема взаимодействия компонентов:

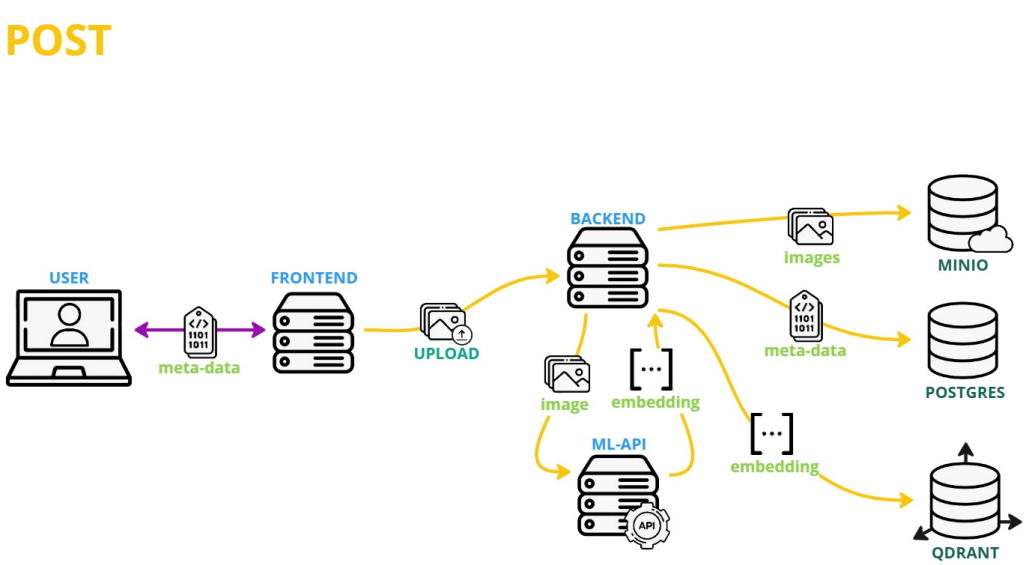
Для развертывания БД используется Docker. У него есть несколько преимуществ, но основное – он сильно облегчает развертывание приложения на любом хосте, для подключения к БД достаточно запустить докер-контейнер.

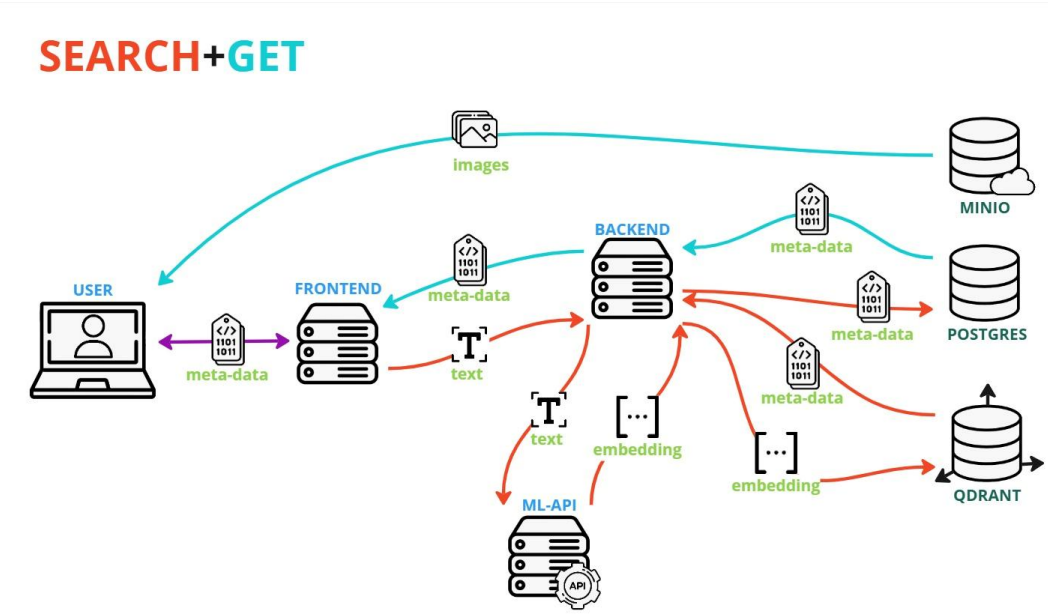
1. Пользователь работает с визуальным интерфейсом (Flet).
2. Интерфейс формирует HTTP-запросы к backend-сервису.
3. Backend обрабатывает запросы: взаимодействует с базами данных (PostgreSQL, Qdrant), файловым хранилищем (MinIO) и, при необходимости, вызывает ML-сервис.
4. ML-сервис возвращает эмбеддинги изображений или текстов.
5. Backend формирует ответ и передаёт его клиенту.
6. Интерфейс отображает результат пользователю.

Такой подход обеспечивает:

- масштабируемость;
- изоляцию компонентов;
- независимое обновление и тестирование;

- удобство развертывания и отладки благодаря Docker.





2.2.2. Обоснование архитектурных решений

Выбор микросервисной архитектуры с REST API связан со следующими преимуществами:

- **Гибкость** — каждый компонент может быть запущен, протестирован и масштабирован независимо;
- **Безопасность и конфиденциальность** — данные обрабатываются полностью локально, исключая передачу изображений в сторонние облака;
- **Кроссплатформенность** — система работает на Windows, Linux и macOS благодаря Docker;
- **Масштабируемость** — возможно горизонтальное масштабирование компонентов при росте нагрузки;
- **Совместимость** — REST API позволяет интегрировать Smart Gallery с другими системами (например, корпоративными DAM или архивами).

3. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

3.1. Ориентировочная экономическая эффективность

В рамках данного проекта **расчёт экономической эффективности не проводится**, так как разработка выполняется в рамках учебного процесса. Вместе с тем, проект демонстрирует практическое применение технологий машинного обучения, микросервисной архитектуры и контейнеризации, что способствует повышению квалификации разработчика и может быть адаптировано под реальные коммерческие задачи.

3.2. Предполагаемая потребность

Программный продукт **Smart Gallery** ориентирован на следующие категории пользователей:

- фотографов, блогеров и дизайнеров, работающих с большими коллекциями изображений;
- научные, культурные и образовательные учреждения, ведущие локальные архивы;
- организации, использующие визуальные данные в условиях ограниченного доступа к сети;
- разработчиков, заинтересованных в создании автономных систем управления изображениями.

Система особенно востребована в условиях, где критичны приватность, автономность, быстрая навигация и локальный доступ к визуальным данным.

3.3. Экономические преимущества разработки по сравнению с отечественными и зарубежными аналогами

На момент начала разработки **отсутствовали прямые аналоги**, обладающие следующими характеристиками в совокупности:

- полностью **автономная работа без доступа к интернету**;
- использование **технологий CLIP/ruCLIP** для интеллектуального поиска изображений;
- удобный, локально запускаемый **интерфейс на Python (Flet)**;
- **контейнеризация** всей системы для упрощённого развёртывания.

Таким образом, **Smart Gallery** предоставляет уникальное решение, сочетающее современные ML-подходы и простоту использования в локальной среде, при этом не зависящее от облачных платформ и сервисов, что снижает затраты на обслуживание и защищает пользовательские данные.

ПРИЛОЖЕНИЕ 1

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. ГОСТ 19.101-77: Виды программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2. ГОСТ 19.102-77: Стадии разработки. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.103-77: Обозначения программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.104-78: Основные надписи. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.105-78: Общие требования к программным документам. // Единая система программнойдокументации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.106-78: Требования к программным документам, выполненным печатным способом. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.201-78: Техническое задание. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. ГОСТ 19.603-78: Общие правила внесения изменений. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9. ГОСТ 19.604-78: Правила внесения изменений в программные документы, выполненные печатнымспособом. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

ПРИЛОЖЕНИЕ 2

ГЛОССАРИЙ

Таблица 1

Понятие	Определение
REST	Архитектурный стиль взаимодействия компонентов через стандартные HTTP-(GET, POST, DELETE и др.).
API	Интерфейс программирования приложений — набор методов взаимодействия различными компонентами программного обеспечения.
Frontend	Клиентская часть приложения, с которой взаимодействует пользователь. В данном проекте реализована с использованием библиотеки **Flet**
Backend	Серверная часть приложения, обрабатывающая запросы и выполняющая бизнес-логику. В проекте реализована с помощью **FastAPI**.
ML-сервис	Отдельный микросервис, отвечающий за генерацию эмбеддингов изображений и поиск по ним. Использует модель **ruCLIP**
CLIP / ruCLIP	Нейросеть, обученная сопоставлять текстовые описания и изображения в одном пространстве признаков
Эмбеддинг	Векторное представление изображения или текста, получаемое с помощью нейросети, используемое для поиска и сравнения
MinIO	Объектное хранилище файлов, аналогичное Amazon S3, используемое для хранения изображений и их превью
Qdrant	Векторная база данных, предназначенная для быстрого поиска по эмбеддингам
Docker	Платформа для упаковки, доставки и запуска приложений в виде изолированных контейнеров
Docker Compose	Инструмент для управления многоконтейнерными приложениями Docker с помощью одного конфигурационного файла
Превью	Уменьшенное изображение, отображаемое в галерее для ускорения загрузки и отображения.
Ручное тестирование	Метод проверки работы системы вручную через пользовательский интерфейс использования автоматизированных средств.

ПРИЛОЖЕНИЕ 3

ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ ФАЙЛОВ

Таблица 2

Папка	Файл / Модуль	Назначение
Smart-Gallery/	.env.dev	Переменные окружения для разработки
	docker-compose.yml	Конфигурация всех Docker-сервисов (backend, frontend, ml_api, postgres, minio, nginx, qdrant)
	nginx.conf	Конфигурация обратного прокси-сервера NGINX
backend/	.dockerignore, Dockerfile, requirements.txt	Служебные файлы сборки backend-сервиса
backend/app/	main.py	Точка входа в FastAPI-приложение
	config.py	Конфигурация приложения (переменные окружения, пути, настройки)
	models.py	SQLAlchemy-модели и схемы таблиц БД
	router.py	Главный маршрутизатор API
	schemas.py	Pydantic-схемы для валидации запросов и ответов
backend/app/api/	ml_api.py	Эндпоинты взаимодействия с ML-сервисом
backend/app/database/	minio_client.py	Инициализация клиента для MinIO-хранилища изображений
	postgres_client.py	Подключение к PostgreSQL
	qdrant_client.py	Подключение к Qdrant-векторной БД
	test_data.py	Тестовые данные для отладки и разработки
backend/app/repository/	base_repository.py	Базовый интерфейс репозиториев
	minio_repository.py	Репозиторий для MinIO
	qdrant_repository.py	Репозиторий для Qdrant
	postgres_repository.py	Репозиторий для работы с PostgreSQL
	repository.py	Агрегация всех репозиториев для использования в логике приложения
frontend/	.dockerignore, Dockerfile, requirements.txt	Файлы сборки Flet-приложения
frontend/app/	main.py	Точка входа в Flet-приложение

	config.py	Константы и конфигурация интерфейса
	routes.py	Навигация между страницами
frontend/app/api/	images_api.py	Вызовы API для массовых операций над изображениями
	image_api.py	Вызовы API для работы с одним изображением
frontend/app/data/	image_data.py	Структуры данных и логика работы с изображениями
frontend/app/views/	base_view.py	Базовый класс всех экранов
	home_view.py	Экран с описанием проекта
	images_view.py	Экран с несколькими изображениями
	image_view.py	Экран одного изображения
	delete_images_view.py	Экран удаления выбранных изображений
	search_images_view.py	Экран поиска изображений по тексту
frontend/app/mixins/	app_bar_mixin.py	Mixin для верхней панели
	grid_mixin.py	Mixin для сетки изображений
	nav_bar_mixin.py	Mixin для нижней навигационной панели
ml_api/	.dockerignore, Dockerfile, requirements.txt	Служебные файлы ML-сервиса
ml_api/app/	main.py	Точка входа в ML-сервис на FastAPI
	config.py	Конфигурация сервиса и модели
	router.py	Эндпоинты для поиска по эмбеддингу и тексту
	schemas.py	Pydantic-схемы для запросов и ответов
ml_api/app/sm_clip/	base_clip.py	Базовый класс для CLIP-моделей
	clip_vit_b_32.py	Реализация модели CLIP ViT-B/32
	ruclip_clip993.py	Реализация дообученной модели RuCLIP на основе clip993

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц в докум.)	№ документа	Входящий № сопроводительного докум. и дата	Подп.	Дата
Изм.	Изменённых	Заменённых	Новых	Аннулированных					