БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

факультет радиофизики и компьютерных технологий

Разработка технико-экономического обоснования для создания информационной системы

" Создание прототипа интерфейса. Проектирование API и

разработка архитектуры ИС"

Работу выполнили:

Островский И. В.

Шаковец И. А.

Сергиевич В. Д.

4 курс 5 группа

Минск, 2025

1. Проектирование Пользовательского Интерфейса (UI/UX)

Проектирование пользовательского интерфейса (UI) основывалось на анализе утвержденных прототипов: **Панели управления** и **Отчета о верификации**

1.1. Описание Прототипа Интерфейса

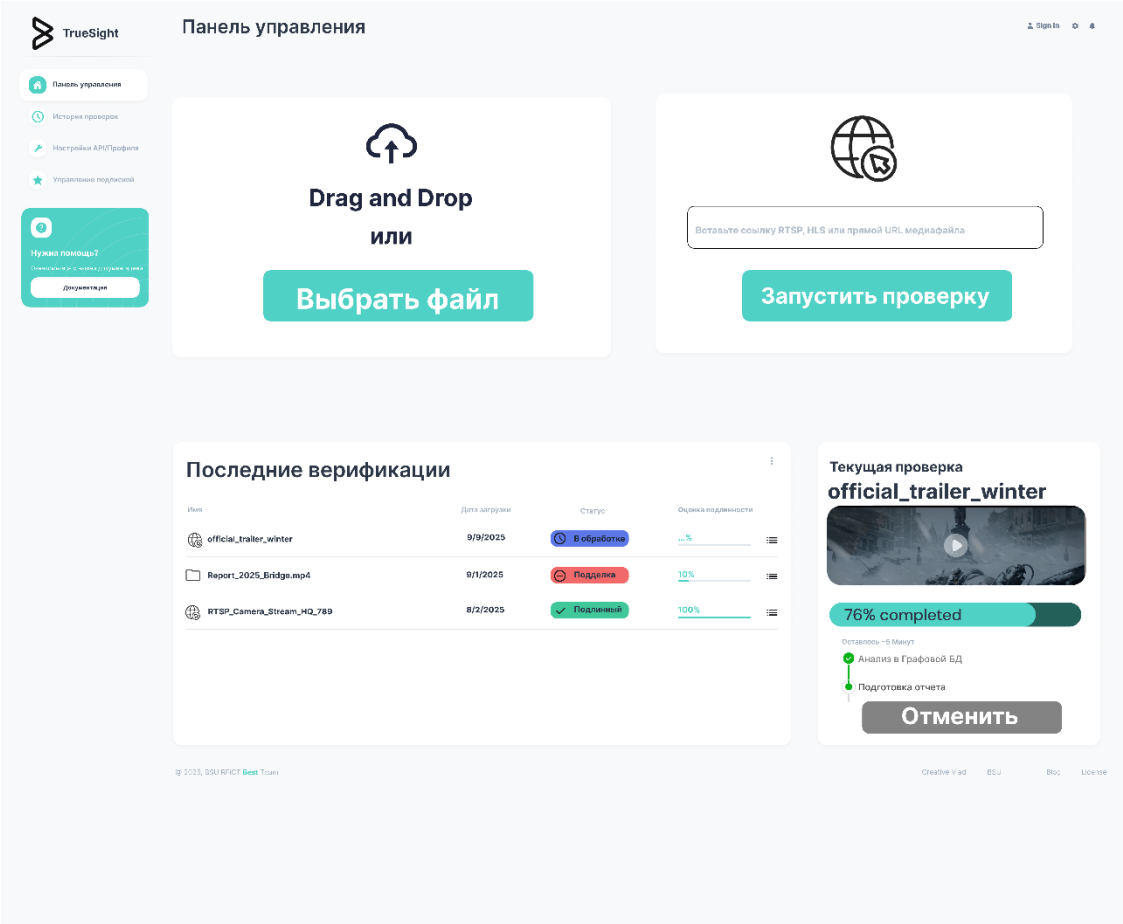

Рисунок 1: Панель управления

Панель управления (Dashboard) спроектирована как центральный хаб для Заказчика, следуя принципам минимализма и функциональной иерархии (эвристика Нильсена: «Простота и эстетика»). Интерфейс четко разделен на

три основные рабочие области (Инициация, История, Мониторинг), что снижает когнитивную нагрузку (Закон Хика).

1)Блок Инициации Проверки (Upload Zone)

Этот блок занимает центральное место, подчеркивая его роль как основного действия Заказчика. Он предоставляет два четких и равноправных способа инициировать верификацию:

- **Загрузка файла:** Представлена иконкой облака и крупной надписью «Drag and Drop или Выбрать файл». Это интуитивно понятный шаблон, соответствующий ментальной модели пользователя (Jakob's Law).
- **Проверка потока (URL/RTSP/HLS):** Представлена иконкой глобуса и полем ввода с инструкцией «Вставьте ссылку RTSP, HLS или прямой URL медиафайла».

Такое явное разделение (файл vs. ссылка) является формой **Предотвращения ошибок** (Error Prevention) , поскольку оно минимизирует вероятность некорректного ввода. Кнопки действий, такие как «Выбрать файл» и «Запустить проверку», выполнены крупными и заметными, что соответствует **Закону Фиттса** и ускоряет взаимодействие.

2) Блок Последних Верификаций (History Table)

Этот блок обеспечивает доступ к истории всех ранее выполненных проверок и реализует принцип **Распознавание, а не запоминание** (Recognition rather than Recall). Таблица содержит:

- **Имя:** Название проверенного контента.
- **Дата загрузки.**
- **Статус:** Четкий, цветокодированный статус («В обработке», «Подделка», «Подлинный»).
- **Оценка подлинности.** Наличие истории и четкого статуса для каждой записи поддерживает **Видимость статуса системы**.

3). Блок Текущей Проверки

Блок «Текущая проверка» (на правой стороне Dashboard) является критически важным для реализации Видимости статуса системы (Visibility of

System Status) в сценарии длительной асинхронной задачи. Он предназначен для управления ожиданием Заказчика и предоставления полной прозрачности хода выполнения фоновой задачи.

A. Визуальное Представление Контента и Прогресса

- Превью: Отображается миниатюра или стоп-кадр из проверяемого медиафайла.
- Прогресс-бар и процент: Главный элемент, отображающий ход выполнения:
- Крупное число 76% completed информирует пользователя.
- Визуальный прогресс-бар и оценка оставшегося времени (Осталось ~5 минут) управляют восприятием ожидания (Doherty Threshold), что критически важно, так как верификация может занимать продолжительное время .

B. Детализация Этапов (Execution Flow) Ниже прогресс-бара отображается последовательность архитектурных этапов процесса верификации:

Завершенные этапы: Отмечены зелеными галочками.

Текущий этап: Выделен и находится в процессе выполнения.

Предстоящие этапы: Видны, но не активны (например, Подготовка отчета). Эта детализация гарантирует, что пользователь получает постоянную обратную связь о работе сложных бэкенд-сервисов (AI Processor, Aggregator).

C. Управление Задачей

Кнопка «Отменить»: Обеспечивает Контроль и Свободу пользователя (User Control and Freedom), позволяя Заказчику прервать длительную операцию, что является важным «аварийным выходом» из асинхронного процесса.

1.2. Отчет о верификации

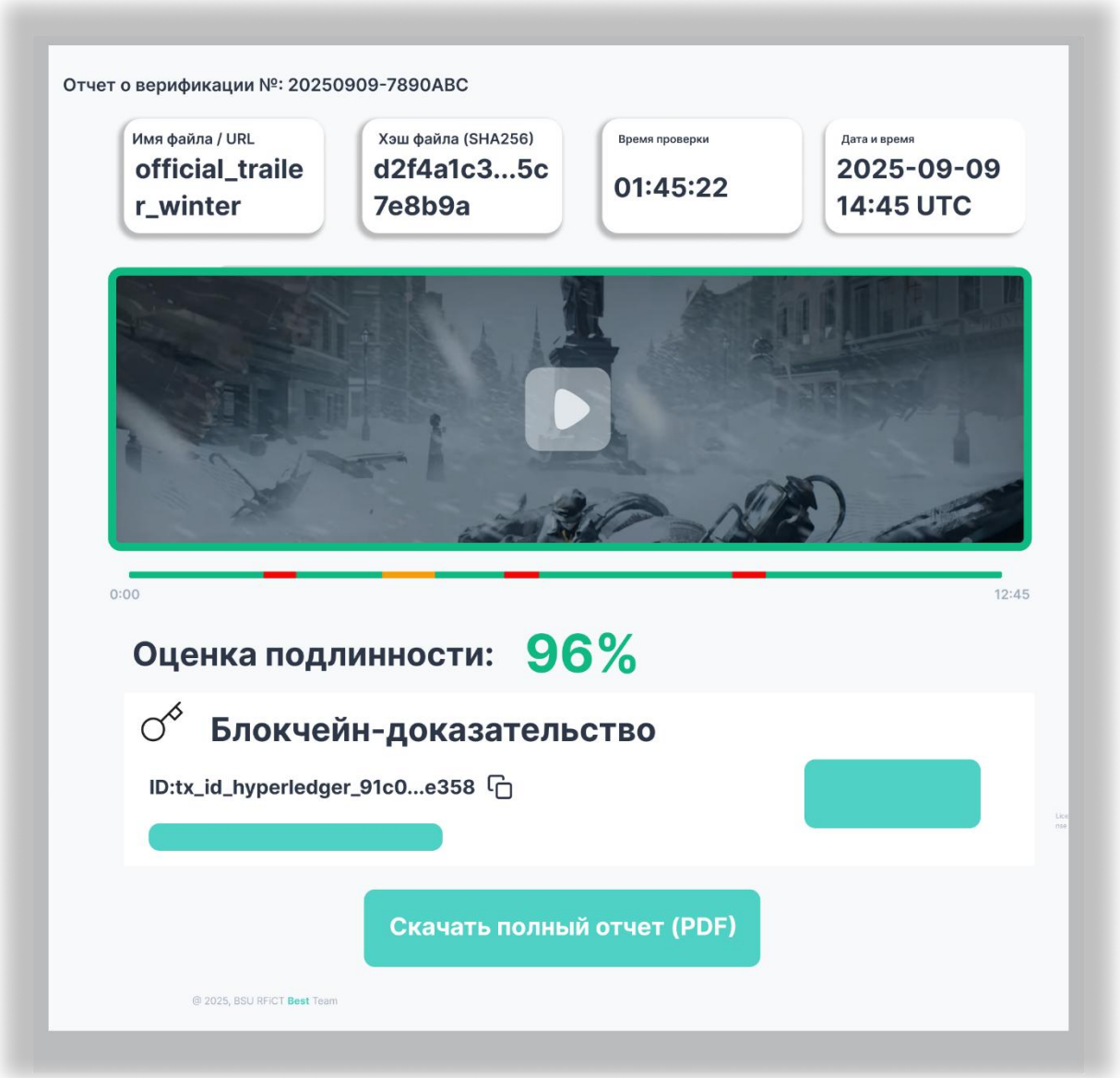

Рисунок 2: Отчет о верификации

Прототип **Отчета о верификации** представляет собой финальное, неизменяемое представление результата работы всей системы, реализуя этап **«Просмотр отчета»**. Дизайн отчета сосредоточен на максимальной ясности, достоверности и доказуемости (Robustness) результата.

1)Идентификационные Данные и Аудит (Header Block)

Верхняя часть отчета содержит метаданные, необходимые для аудита и однозначной идентификации проверенного контента и самого процесса:

- **Номер Отчета:** уникальный идентификатор, связывающий отчет с внутренней системой учета.
- **Имя файла / URL** источник контента.
- **Хэш файла (SHA256):** криптографический хэш медиафайла, гарантирующий, что проверялись именно эти данные. Он является основой для фиксации в блокчейне.
- **Время проверки и Дата и время:** отображаются в часовом поясе UTC, обеспечивая универсальную точку отсчета для юридически значимого аудита.
- **Видеопревью:** непосредственно отображает превью проверенного контента, что немедленно связывает цифровые метаданные с визуальной реальностью, поддерживая принцип **Соответствие системе и реальному миру** (Match Between the System and the Real World).

2) Ключевой Результат: Оценка Подлинности

Основной акцент в отчете сделан на центральной метрике - **Оценке подлинности.**

- **Визуальное выделение:** Оценка выделена крупным, контрастным шрифтом и цветом (зеленый для высокого процента), что обеспечивает мгновенное восприятие результата и соответствует требованию к **Контрасту**.
- **Интерпретация:** Эта метрика является результатом работы **Result Aggregator**, который собирает данные от всех AI Processor'ов и сводит их к единому, понятному числу. Высокий процент 96% соответствует нефункциональному требованию к точности классификатора (не ниже 92%).

3) Блокчейн-Доказательство (Blockchain Proof)

Этот блок является уникальной особенностью системы, подтверждающей ее **Robustness** (Надежность) и **Доказуемость**:

ID транзакции - это идентификатор транзакции, зафиксированной в **Blockchain Node (Hyperledger Fabric)**.

Назначение: Запись хэша файла и результата в блокчейн гарантирует **неизменность** полученной оценки подлинности. В случае любого спора или необходимости аудита, транзакцию можно проверить в распределенном реестре, доказывая, что результат не был изменен после агрегации.

4) Финальное Действие

Кнопка «Скачать полный отчет (PDF)»: представляет собой основное завершающее действие для Заказчика. Это позволяет получить формализованный, нередактируемый документ, необходимый для внутренних или юридических целей.

2. Проектирование Интерфейса Программирования (API)

2.1. Анализ Взаимодействия и Выбор Подхода к API

Архитектура системы TrueSight требует взаимодействия между фронтендом и бэкенд-сервисами, в частности, между **REST API** и **Ingest Service**. Учитывая, что процесс верификации является длительной, асинхронной задачей (до 5 минут), невозможно выполнить его в рамках одного синхронного HTTP-запроса, поскольку необходимо соблюсти нефункциональное требование к Latency <3 с для времени отклика.

Для решения этой проблемы был выбран подход **Асинхронного REST API**, реализующий **Паттерн Ресурса Состояния (Status Resource Pattern)** в сочетании с **Webhooks (callback)**:

- **Инициация (POST):** Клиент отправляет запрос. REST API быстро отвечает статусом 202 Accepted и предоставляет URL для мониторинга (Location header).

- **Мониторинг (Polling):** UI Заказчика (Dashboard) периодически проверяет статус задачи через предоставленный endpoint (GET /jobs/{job_id}) для отображения прогресс-бара.
- **Уведомление (Webhook):** после завершения обработки Result Aggregator отправляет уведомление (Webhook) на указанный клиентом URL, обеспечивая мгновенное информирование о готовности отчета.

2.2. Прототип Спецификации API

Ниже представлена спецификация ключевых REST API-операций.

Таблица 2. Прототип Спецификации Асинхронного REST API

Элемент Спецификации	**Инициирование: POST /api/v1/verify/file**	**Мониторинг: GET /api/v1/jobs/{job_id}**	**Результат: GET /api/v1/reports/{report_id}**
Назначение	Запуск верификации медиафайла или потока.	Получение текущего статуса обработки по ID задачи.	Получение готового финального отчета.
HTTP Метод	POST	GET	GET
Параметры пути	N/A	job_id (string, UUID задачи)	report_id (string, ID отчета)
Успешный Ответ	202 Accepted	200 OK	200 OK (возвращает PDF/JSON отчет)
Неуспешный Ответ	400 Bad Request, 401	404 Not Found, 401 Unauthorized	404 Not Found, 401 Unauthorized

	Unauthorized, 403 Forbidden		

Формат Ответа Мониторинга (GET /api/v1/jobs/{job_id})

Этот ответ используется UI для наполнения блока «Текущая проверка» на Dashboard .

JSON


```
{
  "job_id": "tx_job_abc123",
  "status": "processing",
  "progress_percent": 76,
  "stage": "Анализ в Графовой БД",
  "estimated_time_remaining_minutes": 5,
  "start_time_utc": "2025-09-09T14:45:00Z",
  "result_report_url": null
}
```


3. Проектирование Архитектуры: Применение Паттерна

3.1. Выбор Архитектурного Паттерна: Микросервисная Архитектура (MSA)

Выбор **Микросервисной Архитектуры (MSA)** обусловлен сложными нефункциональными требованиями к платформе верификации мультимедиа, а именно: необходимостью горизонтального масштабирования, отказоустойчивостью и требованиями реального времени.Система состоит из

набора слабосвязанных, независимо развертываемых сервисов (Ingest Service, AI Processor, Result Aggregator, Blockchain Node).

3.2. Обоснование Выбора Паттерна (MSA)

MSA является оптимальным выбором для проекта, поскольку она напрямую решает проблемы, возникающие при обработке мультимедиа:

Таблица 3. Обоснование Микросервисной Архитектуры на основе Требований ТЗ

Критерий Требования	Требование	Решение, Обеспечиваемое MSA	Значимость
Масштабируемость	Горизонтальное масштабирование для сотен конкурентных проверок.	Независимый Scale-Out AI-кластеров: Сервисы AI Processor, требующие GPU-ускорителей, могут масштабироваться отдельно от API-сервисов.	Обеспечивает оптимальную стоимость эксплуатации и поддерживает требование коммерческой эффективности.
Надёжность и Отказоустойчивость	Доступность автоматическое восстановление при сбоях.	Изоляция отказов (Fault Isolation): Сбой в одном компоненте не распространяется на всю систему.	Использование Очереди сообщений гарантирует, что задачи не будут потеряны.
Интеграция	Использование гибридного подхода: AI-	Polyglot Persistence: MSA позволяет каждому	Изоляция Blockchain Node обеспечивает

	анализ + Блокчейн-логирование.	компоненту использовать наиболее подходящее хранилище (Postgres, GraphDB, Hyperledger Fabric).	защиту и подтверждает неизменность результатов верификации.

3.3. Диаграмма Применения Паттерна: Асинхронный Поток Верификации

Для управления асинхронным процессом верификации в MSA используется архитектурный подход, основанный на событиях (Event-Driven Architecture, EDA), где центром взаимодействия является очередь сообщений (Message Queue).

- **Инициация (Client to REST API):** Клиент отправляет запрос. REST API отвечает 202 Accepted и передает запрос в Ingest Service.
- **Прием данных (Ingest Service):** Ingest Service сохраняет медиафайл и **публикует сообщение (событие) в Очередь** (Kafka/RabbitMQ).
- **Изоляция и Надежность (Message Queue):** Очередь выступает буфером, изолируя Ingest Service от AI Processor. Это обеспечивает гарантированную доставку задачи.
- **Обработка (AI Processor):** Worker Preprocessor забирает задачу из очереди и передает в AI Processor, который выполняет ресурсоемкий анализ на GPU-кластере.
- **Агрегация (Result Aggregator):** Aggregator собирает результаты, формирует итоговую оценку, обновляет данные в Postgres/GraphDB.
- **Фиксация Доказательства (Blockchain Node):** Aggregator вычисляет хэш результата и отправляет транзакцию в **Blockchain Node (Hyperledger Fabric)** для неизменяемой фиксации.[1]

- **Уведомление (Notification Service):** Result Aggregator через Notification Service отправляет Webhook-уведомление (callback) клиенту, сигнализируя о готовности отчета.

4. Синтез Решений и Заключение

4.1. Взаимосвязь и Взаимное Влияние UI, API и Архитектуры

Разработанные пользовательский интерфейс, спецификация API и архитектурные решения в системе не существуют изолированно, а образуют целостную, взаимозависимую систему, где требования одного компонента напрямую диктуют реализацию другого.

Взаимосвязь	**Причина**	**Влияние**
UI to API	Длительность верификации (до 5 минут) противоречит требованию Latency < 3 с для отклика	**UI-требование** к «Видимости статуса» и «Управлению ожиданием» (Nielsen, Doherty Threshold) **привело к выбору** асинхронного API-контракта (HTTP 202 Accepted и ресурс мониторинга /jobs/{job_id}).
API to Архитектура	Необходимость поддержки асинхронного контракта и высокой пропускной способности.	**Асинхронный API-контракт потребовал** использования Event-Driven Architecture (EDA) с **Message Queue** (Kafka/RabbitMQ) для надежного приема и

		гарантированной доставки задач в фоновом режиме.
Архитектура to UI/API	Требования к масштабируемости (сотни конкурентных проверок) и отказоустойчивости (Доступность для AI-анализа.	**Микросервисная Архитектура (MSA) обеспечила** независимое масштабирование дорогих AI Processor’ов , что позволяет системе всегда быстро принимать запросы (через Ingest Service) и поддерживать API-контракт, независимо от нагрузки на GPU-кластеры.

Вывод

Проектирование информационной системы основано на принципах системного подхода.

Пользовательский интерфейс успешно решает проблему длительных фоновых задач, используя принципы UX (Законы Хика и Фиттса, Эвристики Нильсена [3]) для управления ожиданиями и снижения когнитивной нагрузки.

API-контракт (Асинхронный REST) был разработан как мост между требованиями UX (быстрый отклик) и техническими ограничениями (длительная обработка медиа), обеспечивая надежный механизм мониторинга и уведомлений.

Архитектурное решение (MSA с EDA) выступает фундаментом, который обеспечивает выполнение жестких нефункциональных требований. Изоляция компонентов и использование **Blockchain Node** гарантирует не только масштабируемость и отказоустойчивость, но и ключевое конкурентное преимущество - **доказуемость** и неизменность финального Отчета о верификации.

Таким образом, целостность и согласованность разработанного UI, API и архитектуры гарантирует, что платформа соответствует всем функциональным и нефункциональным критериям, необходимым для успешного достижения коммерческих целей и обеспечения высокого уровня доверия к верифицируемому контенту.