



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
"Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)"
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И СИСТЕМЫ
УПРАВЛЕНИЯ

КАФЕДРА СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ (ИУ5)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ:

**Автоматизированная информационная система для участников
соревнований - «АИС Фотосервис»**

Студент ИУ5-84Б
(Группа)

(Подпись, дата) **Шпак И.Д.**
(И.О.Фамилия)

Руководитель ВКР

(Подпись, дата) **Антонов А.И.**
(И.О.Фамилия)

Нормоконтролер

(Подпись, дата) **Кротов Ю.Н.**
(И.О.Фамилия)

2022 г.

АННОТАЦИЯ

Расчетно-пояснительная записка 51 с. (с учетом приложений 74 с.), 8 рис., 27 табл., 2 фрагмента кода, 16 источников, 3 приложения.

Выпускная квалификационная работа бакалавра на тему "Автоматизированная информационная система для участников соревнований - «АИС Фотосервис»" посвящена разработке веб-приложения, предназначенного для фотографов спортивных мероприятий с одной стороны и участников соревнований с другой. Первые могут загружать свои фотографии в систему, в то время как вторые могут искать необходимые фото среди загруженных и скачивать их.

Расчетно-пояснительная записка состоит из двух частей.

Постановка задач разработки содержит описание предметной области, функциональных возможностей системы, а также анализ аналогов и сравнение с ними разрабатываемой системы.

Конструкторско-технологическая часть описывает процесс выбора архитектуры приложения, методов взаимодействия различных компонентов, а также выбор программных и аппаратных средств, процесс разработки веб-приложения, включающий в себя разработку интерфейса взаимодействия с пользователем.

СОДЕРЖАНИЕ

АННОТАЦИЯ	2
ВВЕДЕНИЕ	4
1. ПОСТАНОВКА ЗАДАЧ РАЗРАБОТКИ.....	5
1.1. Описание предметной области	5
1.2. Постановка задач и цели проектирования	5
1.3. Функциональные возможности разрабатываемой системы	6
1.4. Выбор критериев качества разрабатываемой системы	7
1.5. Анализ аналогов	8
2. КОНСТРУКТОРСКО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ.....	11
2.1. Выбор архитектуры системы.....	11
2.2. Выбор аппаратных средств.....	27
2.3. Выбор программных средств.....	28
2.3.1. Выбор программных средств для клиентской части системы	28
2.3.2. Выбор программных средств для серверной части системы	30
2.4. Выбор средств развертывания системы	31
2.5. Разработка функциональных диаграмм	32
2.6. Проектирование базы данных	35
2.6.1. Инфологическая модель данных	35
2.6.2. Даталогическая модель базы данных	42
2.7. Разработка интерфейса взаимодействия с пользователем	44
ЗАКЛЮЧЕНИЕ	49
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	50
ПРИЛОЖЕНИЕ А ГРАФИЧЕСКИЕ МАТЕРИАЛЫ	52
ПРИЛОЖЕНИЕ В ТЕХНИЧЕСКОЕ ЗАДАНИЕ.....	59
ПРИЛОЖЕНИЕ С ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ.....	66

ВВЕДЕНИЕ

В современном мире ежедневно проводится масса различных спортивных мероприятий. Большая часть этих событий документируется, в форме различных протоколов, отчетов, а также видео и фотоматериалов.

Количество фотографий, которые делаются на каждом мероприятии может составлять как десятки, так и сотни и даже тысячи. При большом количестве отснятого материала, становится затруднительным поиск изображений конкретных участников, поскольку требуется значительное количество времени для просмотра и отбора нужных фото. Таким образом возникает необходимость отмечать участников на фото. Одним из возможных вариантов решения этой задачи является ручное выделение конкретных участников на фото, но это также требует временных затрат, от пользователя. Другим вариантом решения является применение автоматических систем распознавания номеров участников на изображениях.

Однако даже имея возможность отмечать участников соревнований на конкретном фото, остается необходимость в удобном интерфейсе, для взаимодействия как фотографов, загружающих фото, так и пользователей, желающих просматривать фотографии. Кроме интерфейса взаимодействия, необходимо наличие надежного файлового хранилища, способного хранить большое количество информации с минимально возможным временем доступа к ней, и поддерживающего параллельный доступ к файлам.

Таким образом приложение, выполняющее ряд представленных выше задач, можно выделить в отдельный программный продукт, называемый фотосервисом. В данной выпускной квалификационной работе ставится цель разработать фотосервис, взаимодействующий с двумя внешними системами: системой распознавания номеров участников на фото, и системой учета спортивных мероприятий.

1. ПОСТАНОВКА ЗАДАЧ РАЗРАБОТКИ

1.1. Описание предметной области

Во время спортивных мероприятий создается большое количество медиа контента, в том числе фотографий, которые затем загружаются в публичном доступе в интернет.

Различные пользователи могут желать скачать изображения, сделанные во время этих мероприятий, так участники соревнований часто ищут фото, на которых они изображены. Для поиска можно воспользоваться интернетом, однако количество ресурсов, на которые они загружаются зачастую огромно, что усложняет задачу поиска и фильтрации. Кроме того, поскольку информация о фото составляется людьми, может возникнуть ситуация, когда отсутствуют данные о человеке на конкретном фото, что дополнительно усложняет будущий поиск.

Решением может стать автоматизированная централизованная система, предназначенная как для фотографов, так и для пользователей, желающих скачать фото. Централизованное хранилище, и автоматическое распознавание участников на фотографиях позволит облегчить задачу поиска для пользователей.

Таким образом в рамках предметной области можно выделить 2 основные группы пользователей: фотографы, и пользователи желающие скачать фото. Основными процессами в этом случае будут: загрузка фото, распознавание номеров участников на изображении, поиск конкретного фото среди загруженных и скачивание выбранных изображений.

1.2. Постановка задач и цели проектирования

Цель данной выпускной квалификационной работы заключается в создании автоматизированной системы, поддерживающей разграничение прав

пользовательского доступа и позволяющей выполнять загрузку, выгрузку, поиск и удаление фото из системы.

Для взаимодействия пользователей с системой необходима разработка отдельного интерфейса, являющегося простым и понятным. Таким образом задача проектирования может быть разбить на следующие задачи:

- создание серверного приложения, реализующего логику процессов предметной области, в соответствии с техническим заданием.
- создание клиентского приложения, реализующего интерфейс взаимодействия с разработанным сервером.

1.3. Функциональные возможности разрабатываемой системы

Функциональные возможности разрабатываемой системы описаны в Техническом Задании. Далее представлен список выполняемых системой функций:

1. Авторизация пользователей системы;
2. Просмотр всех проводимых соревнований;
3. Фильтр соревнований по классу пользователя, доступные классы – Медиа и Участник. Также возможен поиск без фильтрации по классу.
4. Просмотр всех фото, относящихся к конкретному соревнованию;
5. Поиск фото по номеру участника;
6. Просмотр номеров изображенных на фото участников;
7. Просмотр превью фото в повышенном качестве с водным знаком;
8. Загрузка фото, зарегистрированными пользователями, с соответствующими правами доступа;
9. Выбор фото для скачивания;
10. Скачивание выбранных пользователем фото;
11. Удаление фото загрузившими их пользователями.

1.4. Выбор критериев качества разрабатываемой системы

В качестве метрик оценки качества работы системы будет использовано несколько критериев:

1. Скорость работы интерфейса;
2. Удобство пользования;
3. Объемы хранимой информации, измеряемое в количестве хранимых объектов;
4. Автоматизированное распознавание номеров.

Оценка коэффициентов важности критериев с использованием метода балльной оценки представлена в таблице 1.

Таблица 1 – Метод балльной оценки.

Код фактора	Название критерия	Балл	Коэффициент важности локального критерия по методу балльной оценки (α_i)
X1	Скорость работы интерфейса	70	0.259
X2	Удобство пользования	70	0.259
X3	Количество хранимых объектов	50	0.185
X4	Автоматизированное распознавание номеров	80	0.296

Коэффициент важности локального критерия по методу балльной оценки (α_i) высчитываются по формуле:

$$\alpha_i = \frac{b_i}{\sum_{i=0}^n b_i} \quad (1)$$

где: b_i – балл критерия,

n – количество критериев.

1.5. Анализ аналогов

В качестве аналогов будут рассмотрены следующие системы:

1. Red Bull Romaniacs – B1;
2. Sport-Images.ru – B2;
3. Sport-Events.ru – B3.

Данные системы были найдены при помощи поисковой системы Google.

Проведем сравнение аналогов с системой (B4), разрабатываемой в рамках этой работы, с использованием метода взвешенной суммы нормированных показателей сравнения [4].

Таблица 2 – Расчет весовых коэффициентов выбранных критериев качества.

Код фактора	Критерий	Весовой коэффициент	B1	B2	B3	B4
X1	Скорость работы интерфейса	0.259	оч. хор.	удовл.	отл.	хор.
X2	Удобство пользования	0.259	оч. хор.	оч. хор.	удовл.	отл.
X3	Количество хранимых объектов	0.185	~10000 0	~10000 00	~ 10000	~ 100000
X4	Автоматизированное распознавание номеров	0.296	неизвестно	есть	нет	есть

Таблица 3 – Сравнение вариантов на Парето-оптимальность.

	B1	B2	B3	B4
B1	0	0	0	0
B2	0	0	0	0
B3	0	0	0	0
B4	0	0	0	0
Результат сравнения	0	0	0	0
Парето-оптимальность варианта	да	да	да	да

Таблица 4 – Вербально-числовая шкала для X1 и X2.

Отлично	Очень хорошо	Хорошо	Удовлетворительно	Плохо
1	0.9	0.75	0.6	0.5

Таблица 5 – Вербально-числовая шкала для X4 (автоматизированное распознавание номеров).

Да	Неизвестно	Нет
1	0,3	0

Таблица 6 – Нормированные значения сравниваемых вариантов.

Код фактора	Весовой коэффициент	Значения показателей			
		B1	B2	B3	B4
X1	0.259	0.2331	0.1554	0.259	0.19425
X2	0.259	0.2331	0.2331	0.1554	0.259
X3	0.185	0.0185	0.185	0.00185	0.0185
X4	0.296	0.0888	0.296	0	0.296
$Y_j = \sum_i^n \alpha_i k_{ij}$		0.5735	0.8695	0.41625	0.76775

Порядок ранжирования вариантов по предпочтительности:

$$B2 \succ B4 \succ B1 \succ B3.$$

В результате проведенного анализа можно отметить, что разрабатываемая система является одним из наиболее оптимальных вариантов среди рассмотренных. Превосходство варианта B2 обусловлено тем, что эта система функционирует на протяжении многих лет, но по прочим рассмотренным параметрам разрабатываемая система превосходит и этот вариант.

2. КОНСТРУКТОРСКО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

2.1. Выбор архитектуры системы

Важным моментом влияющим на дальнейшую разработку системы, является решение интегрировать разрабатываемую систему, в существующую систему организации соревнований и их судейства – MarshalOne [5]. Благодаря подобной интеграции система сможет быстрее обрести первых пользователей, поскольку MarshalOne является устоявшейся и успешно функционирующей системой. С другой стороны пользователи сервиса MarshalOne смогут пользоваться новыми функциями, предоставляемыми фотосервисом, в следствии чего выигрывают обе стороны. Подобная интеграция накладывает определенные ограничения, которые необходимо учитывать в процессе разработки: в частности, необходимо организовать извлечение данных из MarshalOne для использования в функциях фотосервиса, кроме того, в нашей системе не должны храниться личные данные пользователей MarshalOne, в том числе их имена и пароли.

В качестве вариантов архитектуры сервера системы рассматривается 3 варианта: монолитная архитектура, в рамках которой, серверное приложение разрабатывается как большая цельная программа, микросервисная архитектура, в которой функционал системы разделяется между несколькими взаимодействующими между собой меньшими независимыми программами и сервисно-ориентированная архитектура, являющаяся усредняющие первые два варианта.

Преимуществами микросервисной архитектуры над монолитной являются:

- Разделение общей бизнес-логики на части, реализованные в отдельных сервисах. Данная особенность позволяет более удобно организовывать разработку, поскольку при разработке каждого сервиса нет необходимости

держат в уме всю логику предметной области. Кроме того, это облегчает будущую поддержку продукта, а также снижает порог вхождения для новых разработчиков.

- Возможность использования различных технологий, а именно языков программирования и фреймворков, для различных сервисов. Взаимодействие между микросервисами производится общими методами, реализуемыми в различных языках программирования. В связи с этим возникает возможность выбирать наиболее подходящие инструменты для решения каждой поставленной задачи. Напротив, в монолитной архитектуре происходит привязка к конкретному стеку технологий для всего проекта, что лишает мобильности. Взаимодействие между различными частями монолитного приложения осуществляется посредством механизма вызова методов или функций.

- Возможность разбиения базы данных на части доступные только соответствующим сервисам. Точно также как и с разбиением логики предметной области, микросервисный подход подразумевает разбиение хранимых данных на различные контексты. В свою очередь это ведет к уменьшению размеров БД, то есть ускорению доступа к информации, а также большей сохранности данных, поскольку при компрометации одного из сервисов злоумышленниками будет получен доступ лишь к части хранимой информации. Более того появляется возможность выбирать различные СУБД для разных сервисов.

- Способность к вертикальному масштабированию. Благодаря этой способности микросервисного подхода, при масштабировании системы появляется как возможность выбора подходящего аппаратного обеспечения для решения конкретных задач сервиса, так и возможность управления мощностью аппаратного обеспечения в зависимости от нагрузки на систему.

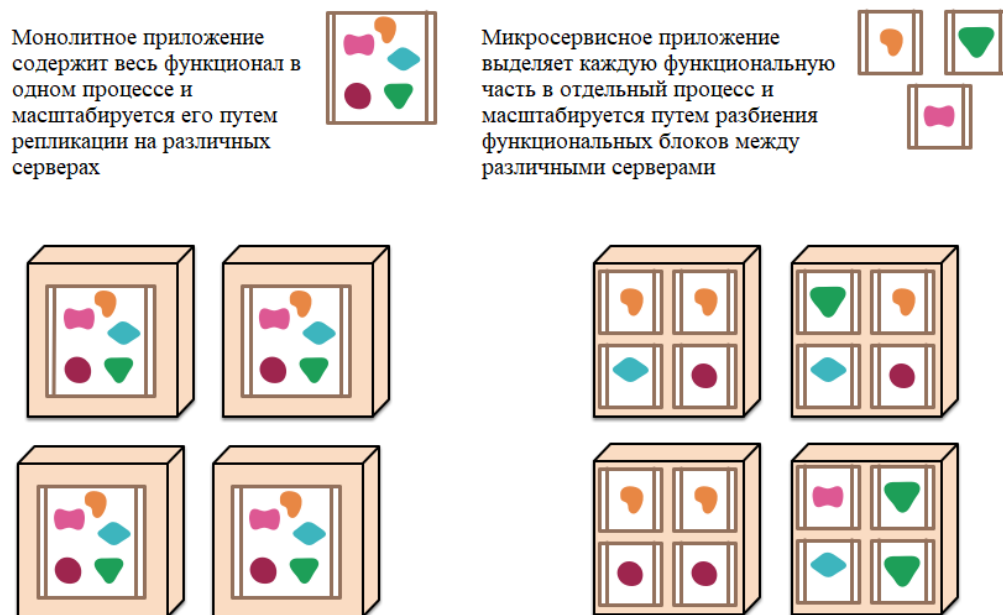


Рисунок 1 – Сравнение монолитной и микросервисной архитектуры.

С другой стороны, монолитная архитектура более эффективно использует аппаратные ресурсы, поскольку обмен информацией происходит внутри оперативной памяти сервера, что уменьшает задержки для конечного пользователя. В общем случае при использовании одинаковых аппаратных ресурсов монолитное приложение превосходит по времени выполнения задач приложение, выполняющее тот же функционал, но с использованием разбиения на микросервисы на величину от 10% до 40% [16]. Кроме того, разработка микросервисного приложения требует больших затрат ресурсов на раннем этапе, поскольку необходимо правильно разбить предметную область на составные части, что увеличивает время разработки. Другим недостатком микросервисной архитектуры является необходимость синхронизации данных в различных базах данных, что полностью отсутствует в монолитной архитектуре.

Сервисно-ориентированная архитектура (SOA), является промежуточным вариантом между первыми двумя, так он объединяет особенность разбиение логики предметной области на составные части, однако части эти зачастую сильнее связаны, что сближает данный подход с монолитной архитектурой, так в рамках этой архитектуры подразумевается совместное использование одних и

тех же ресурсов между различными сервисами, в частности возможно использование общей базы данных.

Теперь определим оптимальную архитектуру, для разрабатываемой системы. Для этого необходимо выполнить следующие задачи:

- удалить варианты, которые не являются Парето-оптимальными;
- произвести оценку показателей важности локальных критериев с использованием: метода базового критерия, метода балльной оценки, метода парного сравнения критериев;
- вычислить средние значения показателей важности локальных критериев;
- произвести сравнение вариантов с использованием интегральных критериев;
- на основании результатов полученных на основе интегральных критериев произвести выбор наилучшего варианта с использованием метода Борда.

Таблица 7 - Исходные значения.

Код критерия	Показатель серверной архитектуры	Значение показателя серверной архитектуры		
		Монолитная архитектура	Микросервисная архитектура	Сервисно-ориентированная архитектура
X1	Масштабируемость	Горизонтальная	Вертикальная и горизонтальная	Вертикальная и горизонтальная
X2	Сложность реализации	Нормальная	Очень высокая	Высокая
X3	Производительность	Высокая	Нормальная	Нормальная

Код критерия	Показатель серверной архитектуры	Значение показателя серверной архитектуры		
		Монолитная архитектура	Микросервисная архитектура	Сервисно-ориентированная архитектура
X4	Сложность модернизации	Высокая	Низкая	Высокая
X5	Актуальность	Низкая	Высокая	Средняя

Таблица 8 - Вербально-числовая шкала для критерия X1 (масштабируемость).

Горизонтальная	Вертикальная	Вертикальная + горизонтальная
0.4	0.6	1

Таблица 9 - Вербально-числовая шкала для критериев X3, и X5 (Производительность и актуальность).

Низкий	Средний / Нормальный	Высокий	Очень высокий
0.4	0.6	0.8	1

Таблица 10 - Вербально-числовая шкала для критериев X2 и X4 (Сложность реализации и модернизации).

Очень высокий	Высокий	Нормальный	Низкий
0.4	0.6	0.8	1

Таблица 11 - Сравнение вариантов на Парето-оптимальность.

Вариант архитектуры	Вариант библиотеки		
	Монолитная	Микросервисная	Сервисно-ориентированная
Монолитная	0	0	0
Микросервисная	0	0	0
Сервисно-ориентированная	0	0	0
Результат сравнения	0	0	0
Парето-оптимальность варианта	Да	Да	Да

Из таблицы 11 видно, что все рассматриваемые варианты являются Парето-оптимальными.

Теперь определим показатели важности локальных критериев с использованием метода базового критерия. Для этого разделим наши показатели на несколько групп важности, где каждая следующая группа в 2 раза более предпочтительней чем предыдущая. В результате получено следующее разбиение по группам:

- Первая группа (менее значимые) – X2;
- Вторая группа – X3, X5;
- Третья группа – X1, X4.

Таким образом сформировано $g = 3$, групп показателей сравнения рассматриваемых архитектур, $n_1 = 1$, $n_2 = 2$, $n_3 = 2$ – количество показателей, которые вошли в состав первой, второй и третьей группы, $k_1 = 1$, $k_2 = 2$, $k_3 = 4$ – коэффициенты, которые показывают степень важности критериев соответствующих групп.

Из представленного ранее вычислим значение базового критерия:

$$1 \cdot \alpha + 2 \cdot 2 \cdot \alpha + 2 \cdot 4 \cdot \alpha = 1$$

$$\alpha = 0,077$$

$$\alpha_1 = 0,077, \quad \alpha_2 = 0,154, \quad \alpha_3 = 0,308$$

Таблица 12 - Расчет весовых коэффициентов методом базового критерия.

Код локального критерия	Коэффициент важности локального критерия α_{i1}
X1	0.308
X2	0.077
X3	0.154
X4	0.308
X5	0.154

Далее произведем расчет весовых коэффициентов критерия методом бального критерия в таблице 13 при помощи формулы 1.

Для получения более точной оценки, произведем её с использованием других критериев, а именно методом бального критерия и парного сравнения.

Расчет с использованием метода бального критерия представлен в таблице 13. Для расчета значений использовалась формула 1.

Таблица 13 - Расчет весовых коэффициентов критерия методом бального критерия.

Код критерия	В, баллы	α_{i2}
X1	9	0.3
X2	1	0.033
X3	5	0.167
X4	9	0.3
X5	6	0.2
	$\sum_{i=1}^5 B_i = 30$	

Расчет с использованием метода попарного сравнения представлен в таблице 14, и используется для расчетов формулы 2 и 3.

Таблица 14 - Расчёт весовых коэффициентов методом парного сравнение критериев.

	X1	X2	X3	X4	X5	C	α_{i3}
X1	1	1.5	1.5	1	1.5	6.5	0.26
X2	0.5	1	0.5	0.5	0.5	3	0.12
X3	0.5	1.5	1	0.5	1	4.5	0.18
X4	1	1.5	1.5	1	1.5	6.5	0.26
X5	0.5	1.5	1	0.5	1	4.5	0.18

$$C_i = \sum_{j=1}^n k_{ij} \quad (2)$$

где: $j = 1 \dots n$;

n – количество критериев.

$$\alpha_{i3} = \frac{C_i}{\sum_{i=1}^n C_i} \quad (3)$$

где: $i = 1 \dots n$;

n – количество критериев.

При этом:

$$\sum_{i=1}^6 C_i = 25$$

Средние значения весовых коэффициентов используемых критериев вычислялись с использованием формулы 4. Вычисленные значения находятся в таблице 15.

$$\alpha_{il} = \frac{\alpha_{i1} + \alpha_{i2} + \alpha_{i3}}{3} \quad (4)$$

где: $l = 1 \dots n, i = 1 \dots 3$;

n – число критериев.

Таблица 15 - Средние значения весовых коэффициентов критериев.

Код локального критерия	α_{i1}	α_{i2}	α_{i3}	$\alpha_{i\text{cp}}$
X1	0.308	0.3	0.26	0.289
X2	0.077	0.033	0.12	0.077
X3	0.154	0.167	0.18	0.167
X4	0.308	0.3	0.26	0.289
X5	0.154	0.2	0.18	0.178

Все рассматриваемые критерии можно разделить на 2 группы: “чем больше, тем лучше”, примером которой является критерий X1 и “чем больше, тем хуже”, примером является критерий X2. В зависимости от группы применяются различные способы нормализации, которые представлены в формулах 5 и 6 соответственно. Нормированные значения критериев записаны в таблицу 16.

$$k_{ij \text{ норм}} = \frac{k_{ij}}{k_{i \max}} \quad (5)$$

$$k_{ij \text{ норм}} = \frac{k_{i \min}}{k_{ij}} \quad (6)$$

Далее необходимо произвести оценку вариантов с использованием интегральных критериев.

Таблица 16 - Локальные критерии сравнения библиотек.

Код локального критерия	Коэффициенты важности локального критерия (α_{ic})	Нормированное значение локального критерия					
		k_{i1}	Ранг	k_{i2}	Ранг	k_{i3}	Ранг
X1	0.289	0.4	2	1	1	1	1
X2	0.077	0.8	1	0.4	3	0.6	2
X3	0.167	0.8	1	0.6	2	0.6	2
X4	0.289	0.6	2	1	1	0.6	2
X5	0.178	0.4	3	0.8	1	0.6	2

Таблица 17 - Критерий 1 - взвешенная сумма показателей сравнения.

Код локального критерия	$\alpha_{i1} * k_{i1}$	$\alpha_{i1} * k_{i2}$	$\alpha_{i1} * k_{i3}$
X1	0.1156	0.289	0.289
X2	0.0616	0.0308	0.0462
X3	0.1336	0.1002	0.1002
X4	0.1734	0.289	0.1734
X5	0.0712	0.1424	0.1068
$Y_1 = \sum_i^n \alpha_i k_{ij}$	0.5554	0.8514	0.7156

$$Y_l = \max \sum_i^n \alpha_i k_{ij} = Y_2 = 0.8514 - \text{микросервисная архитектура.}$$

Таблица 18 - Критерий 2 – мера близости показателей сравниваемых вариантов к идеалу.

Код локального критерия	$\alpha_{il} * (1 - k_i)^2$	$\alpha_{il} * (1 - k_i)^2$	$\alpha_{il} * (1 - k_i)^2$
X1	0.10404	0	0
X2	0.00308	0.02772	0.01232
X3	0.00668	0.02672	0.02672
X4	0.04624	0	0.04624
X5	0.06408	0.00712	0.02848
$Y_l = \sqrt{\sum_{i=1}^n \alpha_i (1 - k_{ij})^2}$	0.22412	0.06156	0.11376

$$Y_l = \min \sqrt{\sum_{i=1}^n \alpha_i (1 - k_{ij})^2} = Y_2 = 0,06156$$

Таблица 19 - Критерий 3 – оценка гарантированного результата сравниваемых вариантов.

Код локального критерия	$\alpha_{il} * k_{i1}$	$\alpha_{il} * k_{i2}$	$\alpha_{il} * k_{i3}$
X1	0.1156	0.289	0.289
X2	0.0616	0.0308	0.0462
X3	0.1336	0.1002	0.1002
X4	0.1734	0.289	0.1734
X5	0.0712	0.1424	0.1068

Код локального критерия	$\alpha_{i1} * k_{i1}$	$\alpha_{i1} * k_{i2}$	$\alpha_{i1} * k_{i3}$
Y_l $= \min (\alpha_i k_{ij})$	0.0712	0.0308	0.0462

$Y_l = \max \min (\alpha_i k_{ij}) = Y_3 = 0,0462$ - сервисно-ориентированная архитектура

Наконец проведем сравнение вариантов и выберем наилучший, применив метод Борда.

Таблица 20 - Критерий 4 – ранжирование по методу Борда.

Код локального критерия	Ранг В1	Ранг В2	Ранг В3
X1	2	1	1
X2	1	3	2
X3	1	2	2
X4	2	1	2
X5	3	1	2
$R_j = \sum_{i=1}^n r_{ij}$	9	8	9

$$R_j = \sum_{i=1}^n r_{ij}$$

$$R_l = \min R_j = Y_2 = 8 - \text{микросервисная архитектура.}$$

Таким образом, все критерии, кроме гарантированного результата указывают на превосходство микросервисной архитектуры, поэтому её и выберем для реализации серверной части системы.

Исходя из представленного выше анализа для разработки сервера выбрана микросервисная архитектура. Далее необходимо решить следующие задачи:

- Разбить предметную область на части, выделяемые в микросервисы;
- Выбрать способ межсервисного взаимодействия;
- Выбрать способ организации файлового хранилища;
- Выбрать способ взаимодействия между пользовательским приложением и файлами в хранилище.

Разрабатываемую систему предлагается разбить на 3 части:

1. Контекст взаимодействия с MarshalOne. Данная часть отвечает за получение и кэширование информации о пользователе и его правах доступа к различным мероприятиям, запрашиваемых из системы MarshalOne.
2. Контекст пользовательских фотографий. Данная часть отвечает за хранение и обработку информации о загружаемых фото. Функции данного сервиса: загрузка фото, удаление фото, поиск фото, отправка запроса на распознавание участников на фото. В процессе работы данный сервис может взаимодействовать с предыдущим сервисом, для валидации прав пользователя на совершение операции. Также этот сервис отвечает за взаимодействие пользователя с хранилищем фотографий.
3. Сервис распознавания участников на фотографиях. Данный сервис представляет собой готовую разработку, позволяющую предоставлять фотографии с целью получения списка номеров изображенных участников для дальнейшего использования [14].

Существующие варианты организации взаимодействия между микросервисами включают в себя:

- Использование REST API;

- Использование технологии RPC;
- Использование брокера сообщений.

Рассмотрим представленные методы. REST API представляет собой способ создания интерфейса взаимодействия посредством HTTP. Данный подход широко используется в системах любого масштаба, а также поддерживается на уровне стандартных библиотек большинством современных языков программирования. Обмен данных производится при помощи структурированных данных, чаще всего в формате JSON или XML.

RPC (Remote Procedure Calling) – удаленный вызов процедур представляет собой схожий с REST механизм взаимодействия посредством HTTP, однако если REST рассматривает всю доступную информацию как ресурсы, которые запрашивает клиент, RPC рассматривает её как набор действий (процедур), которые необходимо выполнить. Современные реализации подхода RPC, в частности GRPC от компании Google, позволяют ускорить взаимодействие между различными узлами сети, путем использования более современного протокола HTTP/2, а также специального бинарного формата данных Protocol Buffers (ProtoBuf), что способствует более эффективному использованию сети и, как следствие, уменьшению времени отклика.

Третий вариант – использование брокеров сообщений. Популярным вариантом на текущий момент является RabbitMQ использующий протокол AMQP. Данный подход подразумевает использование отдельного приложения, отвечающего как за передачу запросов между сервисами, так и за взаимодействие с клиентом. Внедрение брокера сообщений целесообразно в системах предполагающих большое количество узлов, в которые могут поступать запросы.

Сравнивая представленные выше три метода, стоит отметить превосходство во времени отклика систем, использующих механизм RPC либо брокеры сообщений, однако их эффективность проявляется в системах с большим количеством запросов и взаимодействующих микросервисов, а затраты на реализацию, в рамках разрабатываемой системы, несоизмеримы с ожидаемым

выигрышем в скорости для конечного клиента. В связи с этим принято решение реализации взаимодействия посредством REST API. Кроме того, одним из требований к системе является простота обслуживания, из-за чего использование более сложных подходов не целесообразно.

Другим важным моментом в процессе разработки архитектуры системы является способ организации файлового хранилища. Основным предъявляемым требованием к нему является масштабируемость, что ведет к использованию облачных решений. Для использования в нашей системе отлично подходят S3 облачные хранилища, поскольку они обладают автоматическим масштабированием в зависимости от количества хранимых объектов. Другим преимуществом S3 является независимость от стандарта POSIX, что позволяет обеспечивать стабильное время отклика независимо от количества запросов к конкретному файлу, соответствующее исследование представлено в [6]. С другой стороны, платой за скорость является ограниченный интерфейс взаимодействия с хранимыми файлами, так S3 не поддерживает операцию изменения уже загруженных файлов.

Наконец последней задачей по разработке архитектуры сервера является выбор способа взаимодействия пользователя с S3.

Существует несколько альтернативных подходов, каждый из которых обладает своими преимуществами перед остальными:

1. Использование потоковой передачи данных посредством микросервиса. В рамках этого подхода взаимодействие пользователя с хранилищем осуществляется целиком при посредстве микросервиса. Процессы загрузки и выгрузки изображений реализованы при помощи потоковой передачи данных. Так, по запросу клиентом фото, микросервис-посредник инициирует загрузку этого фото из S3 и записывает данные, считываемые из получаемого потока в ответ на клиентский запрос, что позволяет не хранить целое изображение в оперативной памяти. Во время загрузки новых изображений дополнительно происходит считывание и проверка конфигурации файла, с целью отсеивания некорректных файлов.

2. Модификация предыдущего способа путем реализации локального кэша внутри оперативной памяти микросервиса посредника. Этот подход позволяем снизить время доступа при запросе одних и тех же файлов, но требует дополнительных аппаратных ресурсов.

3. Предоставление прямого доступа на чтение к файлам внутри S3 хранилища. Таким образом просматривать файлы могут любые пользователи, имеющие ссылку на файл, загрузка производится с использованием других представленных методов. Недостатком этого метода является то, что S3 подразумевает плату за использование трафика, объемы которого, при использовании этого метода, невозможно контролировать. С другой стороны, снижается время нагрузка на сервер, а также время доступа к файлам для конечного клиента при большом количестве запросов. Кроме того, данный лишает возможности ограничивать доступ к файлам для различных групп пользователей, что ограничивает потенциал раскрытия предметной области.

4. Предоставление временных ключей для доступа к файлам. Данный метод поддерживаемый API S3, позволяет создавать временные ключи для доступа клиентов к файлам. Подход также позволяет снизить нагрузку на сервер, как и предыдущий метод, с другой стороны растет время доступа к файлам, при отсутствии высокой нагрузки, поскольку сначала сервер запрашивает у S3 ключ, затем передает его клиенту и только после этого клиент может скачать желаемый файл [7].

Интерфейс взаимодействия клиента с сервером представляет собой отдельное веб-приложение, получающее информацию из определенных частей системы посредством HTTP запросов. Таким образом общая структура системы соответствует рисунку 2.

составными частями системы. В качестве провайдера был выбран VK Cloud Solutions, поскольку он предоставляет одновременно услуги VPS (выделенные виртуальные сервера) и S3, а также использует современное оборудование. Кроме того, сервера провайдера располагаются в России, что позволит дополнительно ускорить работу системы для пользователей. Выбранная конфигурация сервера имеет следующие параметры:

- Одноядерный процессор, точная модель процессора неизвестна;
- 2 Гб ОЗУ;
- Тип системы – 64-разрядная операционная система Ubuntu Server.

Преимуществом использования выделенного сервера, является возможность смены конфигурации при увеличении нагрузки, что позволяет дополнительно повысить масштабируемость системы. Максимальная поддерживаемая конфигурация одного выделенного сервера имеет следующие параметры:

- 16 ядер CPU, точная модель процессора неизвестна;
- 64 Гб ОЗУ.

2.3. Выбор программных средств

2.3.1. Выбор программных средств для клиентской части системы

Клиентское приложение написано с использованием языка TypeScript и фреймворков React и Bootstrap. Данные инструменты активно используются для разработки веб-приложений и обладают хорошей документацией.

TypeScript [10] - язык веб-разработки расширяющий возможности обычного JavaScript. В частности, TypeScript реализует функционал объектно-ориентированного программирования, а также строгую типизацию данных, сохраняя совместимость с библиотеками для JavaScript.

Особенностью же React [11] является использование AJAX, для фоновых обновлений информации, в следствии чего страница с приложением выглядит

более отзывчиво, а также наличие JSX – расширения синтаксиса JavaScript и TypeScript, облегчающее интеграцию программного кода в HTML страницы.

Bootstrap представляет собой популярную библиотеку содержащую готовые компоненты и стили для разработки веб-приложений, что позволяет ускорить разработку.

Наиболее важными библиотеками для клиентской части ПО являются React-Router-Dom [12] и Mobx [13]. Первая позволяет организовать перемещение пользователя внутри приложения с использованием страниц, что позволяет создать удобную структуру клиентского приложения с использованием страниц, т.е. набора JSX документов, между которыми может переключаться пользователь. Вторая отвечает за хранение глобальной информации приложения, что позволяет передавать данные между различными страницами, то есть выполняет функции локального хранилища данных приложения. Таким образом общую структуру клиентского приложения можно представить согласно следующей картинке:

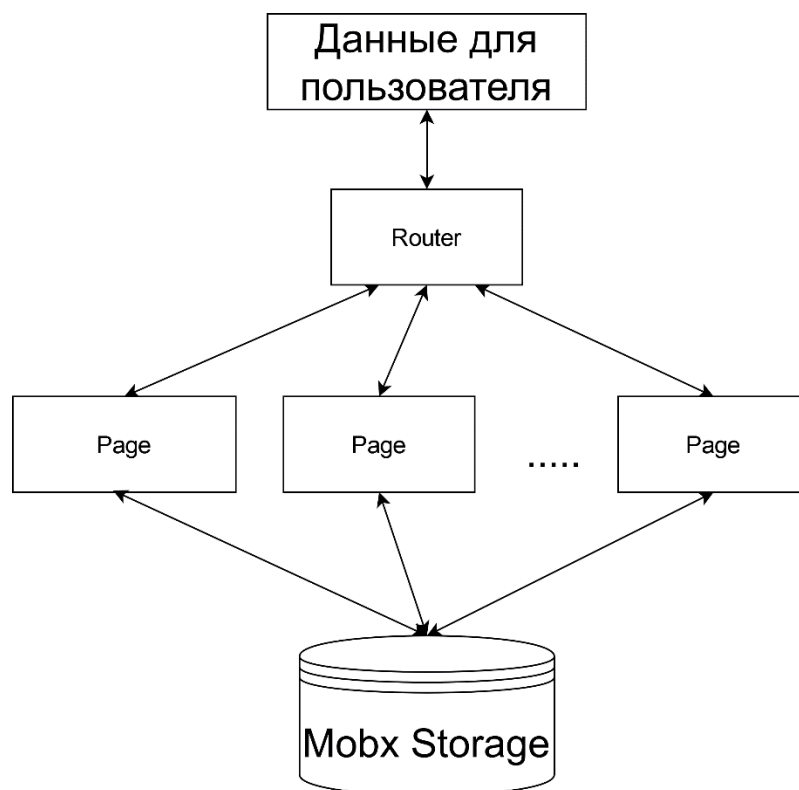


Рисунок 3 – Архитектура клиентского приложения.

2.3.2. Выбор программных средств для серверной части системы

Для серверной части системы необходимо выбрать:

- язык и фреймворки для написания сервера, отвечающего на запросы, в том числе ORM для взаимодействия с СУБД и библиотеку для взаимодействия с S3;
- систему управления базами данных.

В качестве языка программирования для серверной части, было принято решение использовать Golang [8]. Данный язык позволяет разрабатывать высокопроизводительные HTTP сервера, что полностью соответствует поставленной задаче. В качестве библиотеки для взаимодействия с СУБД было решено использовать GORM [9], в виду его простоты и качества документации, а для взаимодействия с S3 – Minio [15], поскольку эта библиотека предоставляет удобный интерфейс, и является высокопроизводительной, что важно при передаче большого количества данных.

Также важной задачей при разработке серверного ПО является журналирование событий, поскольку это способствует быстрому поиску программных ошибок и их исправлению. В системе оно реализовано с использованием библиотеки lumberjack, которая позволяет создавать перезаписываемые журнальные файлы, что позволяет экономить место на диске.

Несмотря на то, что существует возможность использовать различные СУБД для микросервисов, было принято решение использовать единую СУБД, но организовать в ней несколько баз данных. В итоге выбрана PostgreSQL, которая представляет собой реляционную СУБД. Её можно использовать для создания баз данных, реализующих связи между сущностями предметной области любой сложности. Кроме того, PostgreSQL официально поддерживается GORM, что позволяет избежать большого количества программных ошибок, в процессе разработки.

2.4. Выбор средств развертывания системы

Поскольку разработка представляет собой цельную систему, состоящую из множества частей, необходимо выбрать решение для задачи развертывания ПО на выделенном сервере. Особую популярность, в данном направлении приобрел Docker, который представляет собой программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, иначе говоря контейнеризатор приложений.

Контейнеризация — это упаковка программного кода только с библиотеками операционной системы (ОС) и зависимостями, необходимыми для выполнения кода, для создания одного легкого исполняемого файла, называемого контейнером, который стабильно работает на любой инфраструктуре. Более портативные и ресурсоэффективные, чем виртуальные машины (ВМ), контейнеры становятся вычислительными единицами современных облачных приложений.

Упаковка приложения в контейнер позволяет свободно перемещать его между различными платформами, поскольку в нем содержится абсолютно все необходимые для работы компоненты.

Для своей работы Docker использует файлы специального формата, содержащие наборы команд для развертывания программ внутри контейнера. Организация работы приложений из нескольких контейнеров осуществляется с использованием инструмента Docker-Compose, который также использует файлы специального формата, описывающего связи между контейнерами, а также указывающего на Docker файлы, используемые при сборке. Непосредственно сборка программного кода в запускаемые программы осуществляется при помощи системы сборки Make и специальных Makefile, которые содержат набор команд для исполнения. Использование Docker-Compose, Docker и Make позволяет сократить количество действий при обновлении программного кода, а также упростить операцию развертывания программного обеспечения, состоящего из множества частей на сервере.

Также необходимо выбрать веб-сервер, который будет передавать информацию, получаемую сервером от клиента соответствующим программам, то есть выполнять функцию проксирования запросов. Популярным веб-сервером, реализующим требуемый функционал является Nginx. Его преимуществами является удобство настройки, и большое количество конфигурируемых параметров, включая кэширование данных, максимальный размер запроса и количество запросов в единицу времени, что позволит ограничивать нагрузку на систему.

2.5. Разработка функциональных диаграмм

Для создания функциональных диаграмм применена нотация IDEF0[5]. Используемая в этой нотации декомпозиция блоков на более простые, позволила описать систему на различных уровнях. Диаграммы описывают функции согласно требуемым в техническом задании и реализованные в системе. Построенные диаграммы содержатся в приложении А расчетно-пояснительной записки.

Кроме того, в процессе разработки потребовалось создание дополнительных диаграмм, описывающих процесс взаимодействия системы с сервисом распознавания номеров участников, а также процесс загрузки изображений в систему. Диаграммы реализующие соответствующие диаграммы представлены на рисунках 4 и 5.

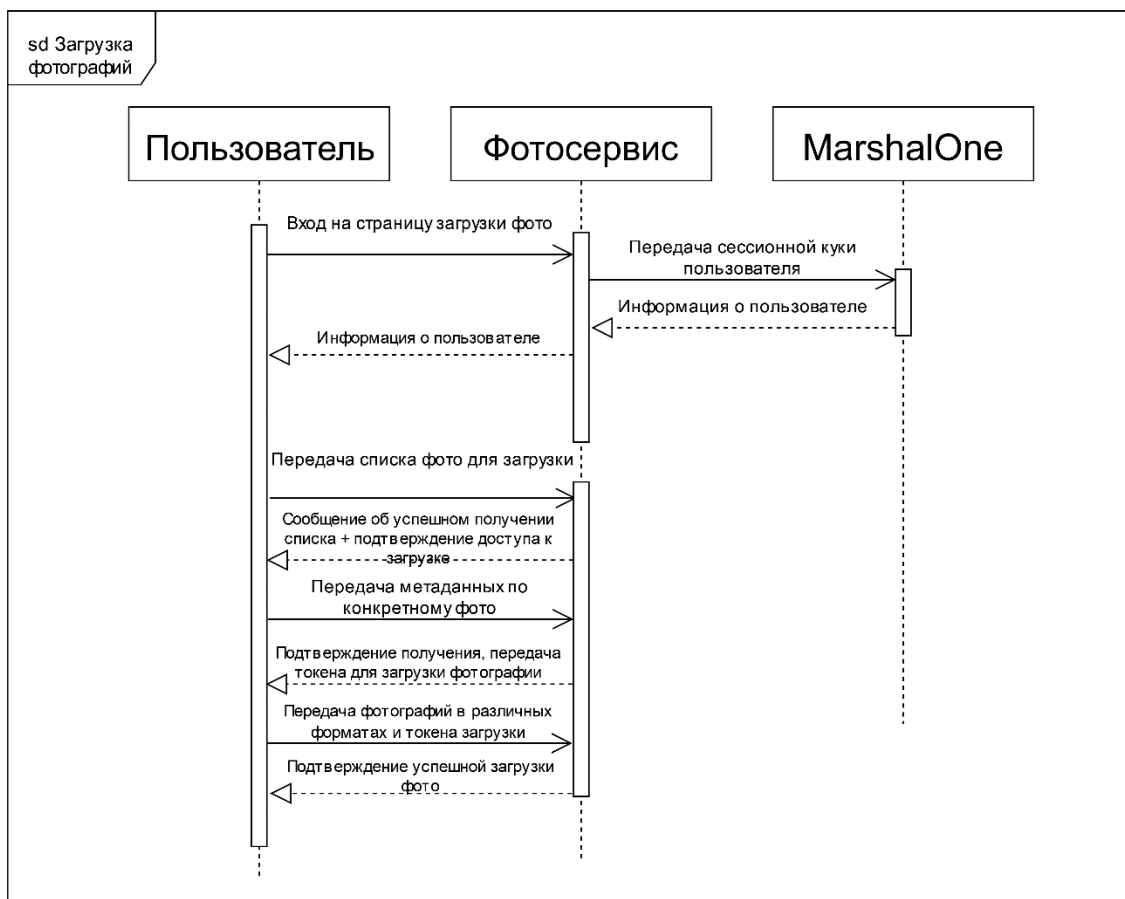


Рисунок 4 – Диаграмма последовательности загрузки фотографий.

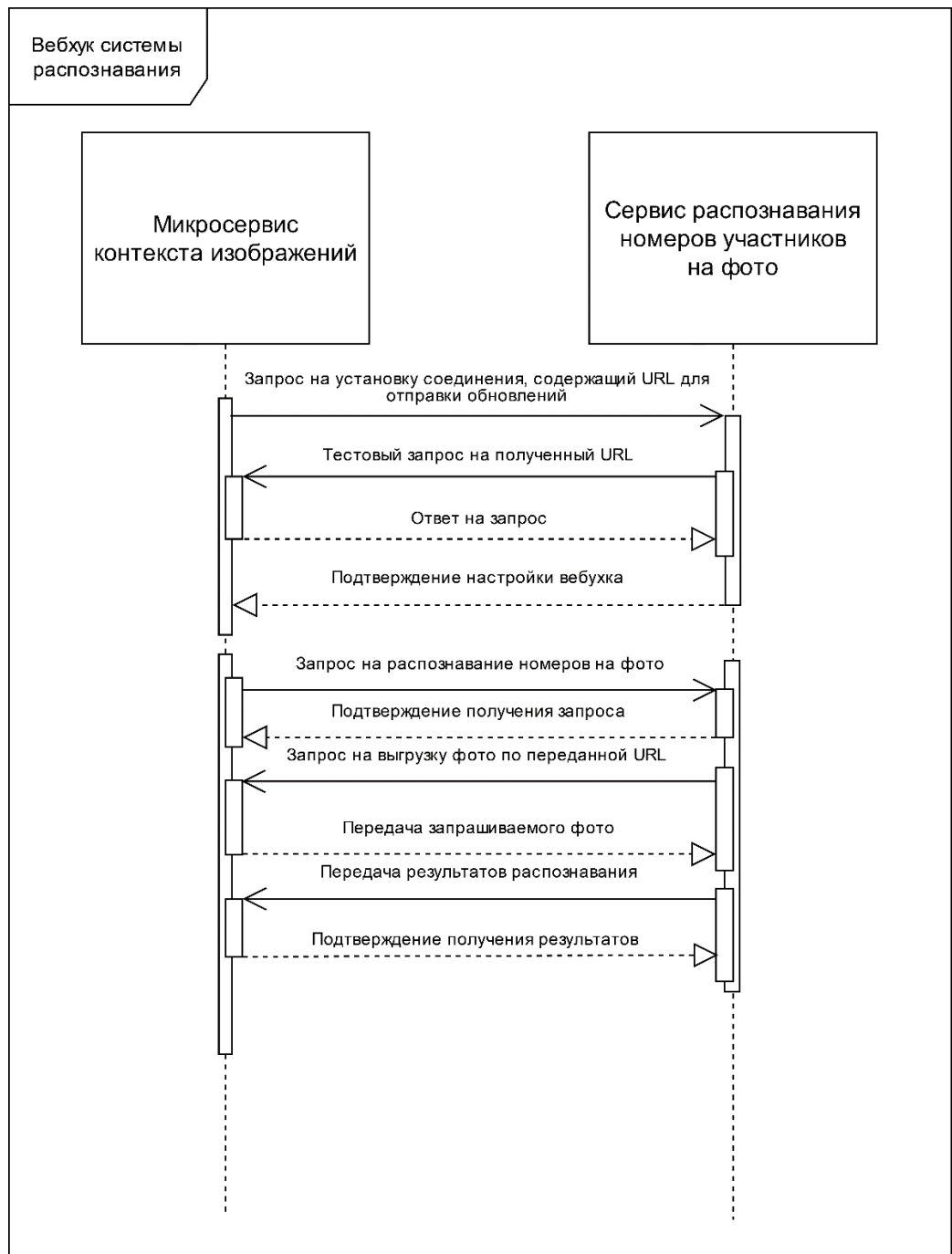


Рисунок 5 – Диаграмма последовательности взаимодействия с системой распознавания номеров участников.

2.6. Проектирование базы данных

2.6.1. Инфологическая модель данных

Как уже было сказано ранее, планируется использовать различные базы данных для микросервисов. Таким образом будут представлены данные для 2х баз данных: БД микросервиса фотографий, и БД микросервиса отвечающего за интеграцию с MarshalOne.

Для проектирования инфологической модели опишем сущности и их атрибуты.

Сущности микросервиса контекста фотографий:

Таблица 21 - Сущность «Фото».

Логическое имя атрибута	Тип данных	Ключ
Идентификатор фото	Числовой	Первичный (РК)
Размер фото	Числовой	
Ширина	Числовой	
Высота	Числовой	
Название фото	Текстовый	
Внешний идентификатор фото	Текстовый	
Время загрузки	Дата и время	

Таблица 22 - Сущность «Тэг».

Логическое имя атрибута	Тип данных	Ключ
Идентификатор тэга	Числовой	Первичный (РК)
Идентификатор фото	Числовой	
Тэг	Текстовый	

Логическое имя атрибута	Тип данных	Ключ
Идентификатор гонки	Текстовый	
Флаг ввода пользователем	Логический	

Таблица 23 - Сущность «Загрузка фото».

Логическое имя атрибута	Тип данных	Ключ
Идентификатор загрузки	Числовой	Первичный (РК)
Название загружаемого файла	Текстовый	
Внешний идентификатор пользователя	Текстовый	
Внешний идентификатор гонки	Текстовый	
Ширина	Числовой	
Высота	Числовой	
Размер полного фото	Числовой	
Размер фото с водным знаком	Числовой	
Размер фото для системы распознавания	Числовой	
Размер превью фото	Числовой	
Флаг загрузки полного фото	Логический	

Логическое имя атрибута	Тип данных	Ключ
Флаг загрузки фото с водным знаком	Логический	
Флаг загрузки фото для системы распознавания	Логический	
Флаг загрузки превью фото	Логический	

Таблица 24 – Описание связей сущностей.

Связь	От какой сущности	К какой сущности	Тип связи
относится	Тэг	Фото	М-1

Теперь рассмотрим сущности БД микросервиса взаимодействия с М1. Особенностью этого микросервиса является то, что он необходим для получения и хранения данных из БД внешней системы MarshalOne посредством предоставляемого API, что определяет архитектуру и особенности БД, разрабатываемой для этого сервиса. В частности, информация должна кэшироваться, но не храниться постоянно, то есть записи должны удаляться при истечении определенного времени, и обновляться по необходимости. Кроме того, поскольку запрашиваемая информация независима, и поступает в случайном порядке, определяемым запросами пользователей к нашей системе, в рамках этого сервиса отсутствуют связи между данным разных сущностей, что позволяет в будущем реализовать её с использованием не реляционной СУБД, использование которой в текущий момент нецелесообразно ввиду разрастания количества необходимого для работы системы ПО.

Таблица 25 - Сущность «Гонка».

Логическое имя атрибута	Тип данных	Ключ
Идентификатор гонки	Числовой	Первичный (РК)
Внешний идентификатор гонки	Текстовый	
Название гонки	Текстовый	
Дата проведения	Дата и время	
Место проведения	Текстовый	

Таблица 26 - Сущность «Права пользователя».

Логическое имя атрибута	Тип данных	Ключ
Идентификатор записи	Числовой	Первичный (РК)
Внешний идентификатор пользовательской сессии	Текстовый	
Идентификатор соревнования	Логический	
Флаг принадлежности к организаторам	Логический	
Флаг принадлежности к участнику	Логический	
Флаг принадлежности к фотографиям	Логический	

Таблица 27 - Сущность «Сессия пользователя».

Логическое имя атрибута	Тип данных	Ключ
Идентификатор сессии	Числовой	Первичный (РК)
Внешний идентификатор пользовательской сессии	Текстовый	
Внешний идентификатор пользователя	Текстовый	

Разработанная инфологическая модель БД системы представлена на рисунках 6 и 7.

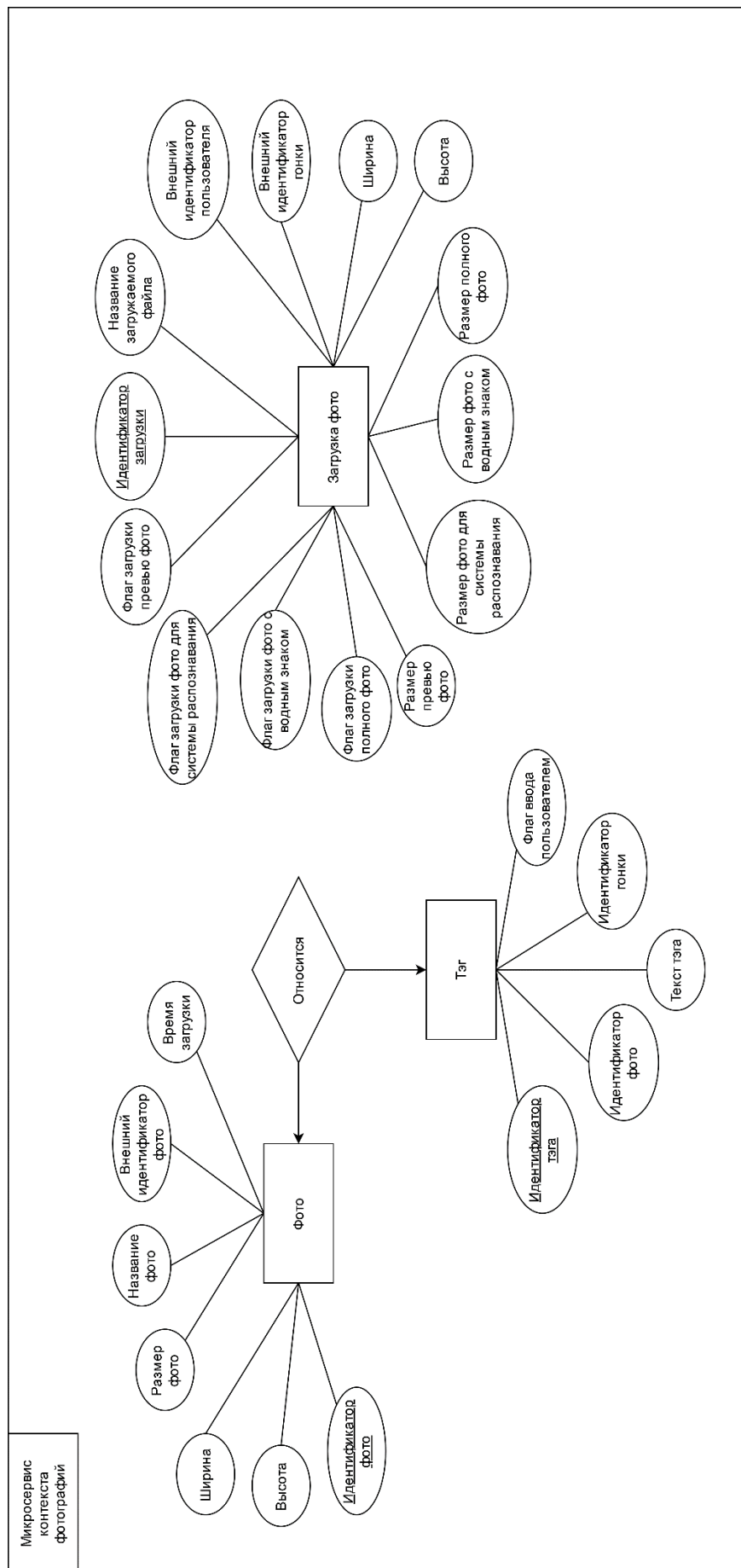


Рисунок 6 - Инфологическая модель микросервиса контекста изображений.

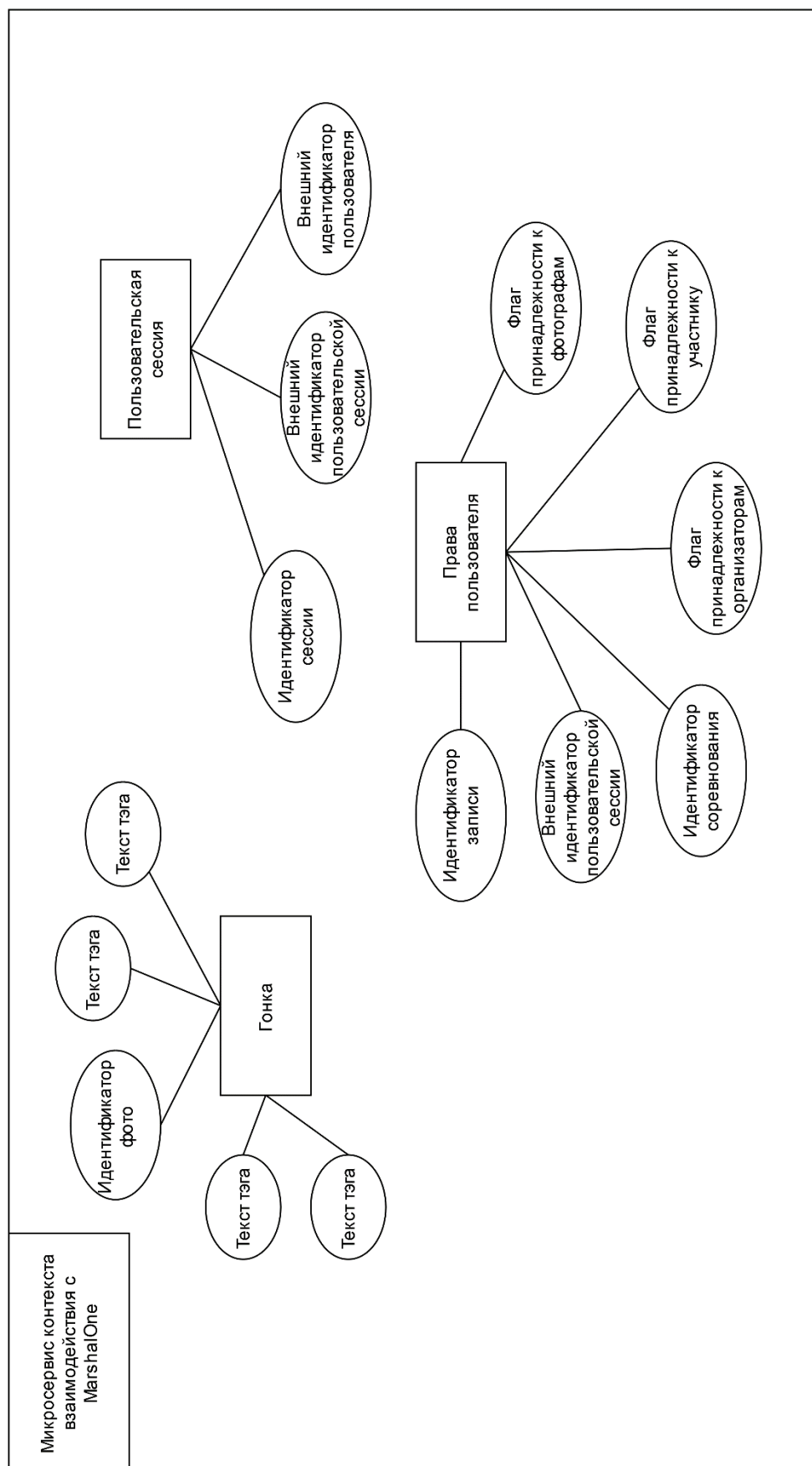


Рисунок 7 - Инфологическая модель микросервиса контекста взаимодействия с MarshalOne.

Отсутствие явных связей внутри БД, приводит к необходимости реализации механизмов, характерных для нереляционных БД, например 2-phase commit(2PC) и механизм обратных транзакций, с целью обеспечения постоянной согласованности данных.

2.6.2. Даталогическая модель базы данных

Учитывая особенности представленной в прошлом пункте инфологической модели базы данных, была построена её даталогическая модель. Специфичными для неё особенностями являются поля `created_at`, `updated_at` и `deleted_at` добавляемые к каждой таблице базы данных выбранной ORM и хранящие информацию о создании записи, её удалении и изменении. Кроме того, в БД используется отдельный тип данных: `UUID`, используемый для создания уникальных текстовых идентификаторов записей. `UUID` генерируются специальным алгоритмом, задачей которого является создание неповторяющихся идентификаторов, что позволяет создавать уникальные записи в разных экземплярах БД. `UUID` также передается по внешние таблицы других БД, что позволяет скрыть настоящие идентификаторы записи от пользователя, то есть выступают в роли внешнего идентификатора записи. Поскольку `UUID` представляет собой строку, то для таблиц, в которых используются внешние идентификаторы, например в таблице `photo` это `race_id` и `user_id`, получаемые из других БД вместо типа `UUID` используется тип `Text`, для упрощения работы с конкретными полями.

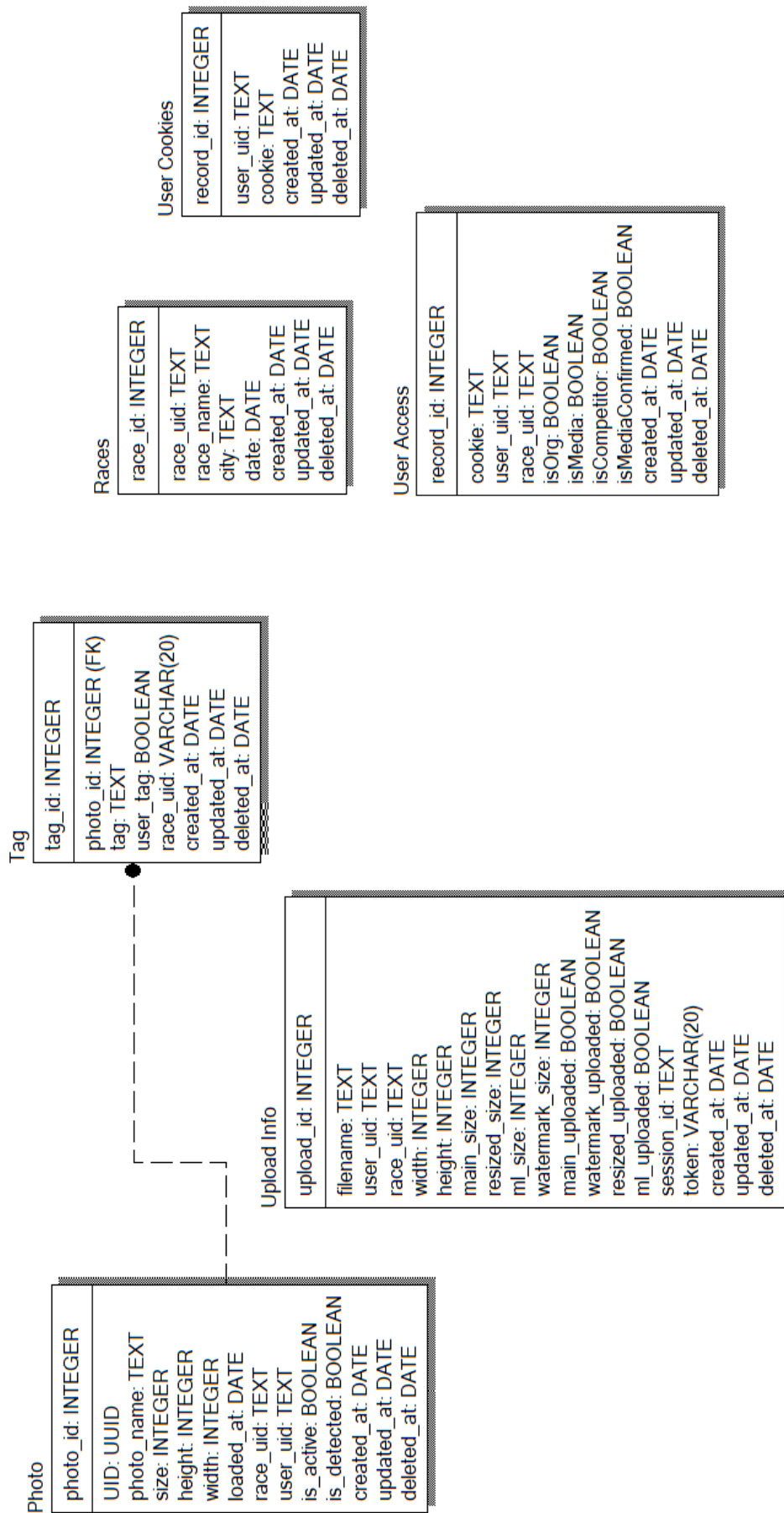


Рисунок 8 - Даталогическая модель.

2.7. Разработка интерфейса взаимодействия с пользователем

Для организации доступа к функционалу системы для конечных пользователей необходимо создать отдельный интерфейс, поэтому было разработано веб приложение с использованием инструментов, указанных в пункте 2.3.1.

В приложении предусмотрены следующие страницы:

- Страница авторизации. Пользователь заполняет свои данные и входит в систему. На этой же странице незарегистрированный пользователь может получить пароль, нажатием на соответствующую кнопку;
- Страница со всеми проводимыми мероприятиями. На этой странице пользователь может просмотреть все мероприятия, и отфильтровать их по дате и классу пользователя;
- Страница конкретного мероприятия. На этой странице находится информация о выбранном мероприятии, и если оно уже началось, то и список загруженных фото. На странице предусмотрен поиск фото по номерам участников, а также просмотр изображений, на которых их не удалось распознать. Для удобного просмотра большого количества фото предусмотрена пагинация;
- Страница конкретного фото. На этой странице можно просмотреть выбранное фото, номера изображенных на нем участников, а также похожие фото;
- Страница с пользовательской галереей. На этой странице можно просмотреть выбранные для фото и скачать их;
- Странице загрузки фото. На этой странице при наличии прав доступа можно загрузить фотографии с мероприятия, а также просмотреть и удалить уже загруженные пользователем фотографии.

Изображения, демонстрирующие интерфейс содержатся в приложении А расчетно-пояснительной записки.

Особенностью разработанного клиентского приложения является то, что оно выполняет часть функций сервера, поскольку используемый сервер не обладает необходимыми характеристиками для выполнения большого количества параллельных операций с изображениями. В частности, на клиентское приложение вынесены следующие функции:

- Сжатие изображения;
- Нанесение водного знака;
- Сохранение нескольких изображений в Zip архив.

Исходный код функций, выполняющих задачи по обработке изображений представлен ниже:

```
const resizeFile = (file: File) => {
  return new Promise(resolve => {
    var height: number
    var width: number
    var contentType = "image/jpeg";
    var canvas = document.createElement('canvas');
    var ctx = canvas.getContext('2d');
    const reader = new FileReader();
    reader.onload = () => {
      var image = new Image();
      image.onload = function () {
        if (image.height > image.width) {
          height = 480
          width = 480 * (image.width /
image.height)
        } else {
          height = 480 * (image.height /
image.width)
          width = 480
        }
        canvas.height = height
```

```

        canvas.width = width
        if (ctx !== null) {
            ctx.drawImage(image, 0, 0,
canvas.width, canvas.height);
            canvas.toBlob(function (blob) {
                if (blob !== null) {
                    var newFile = new
File([blob], file.name, {
                                type: contentType,
                                lastModified:
Math.floor(new Date().getTime() / 1000)
                                })
                    resolve(newFile)
                }
            }, 'image/jpeg', 1)
        }
    }
    if (reader.result?.toString() !== null) {
        image.src = reader.result?.toString();
    }
}
reader.readAsDataURL(file);
}))
}

```

Фрагмент кода 1. Функция «Сжатие исходного изображения»

```

const addWatermark = (file: File) => {
    return new Promise(resolve => {
        var height: number
        var width: number
        var contentType = "image/jpeg";
        var canvas = document.createElement('canvas');
        var ctx = canvas.getContext('2d');
        const reader = new FileReader();
        reader.onload = () => {

```

```

var image = new Image();
image.onload = () => {
    if (image.height > image.width) {
        height = 1024
        width = 1024 * (image.width /
image.height)

    } else {
        height = 1024 * (image.height /
image.width)

        width = 1024
    }
    canvas.height = height
    canvas.width = width
    ctx?.drawImage(image, 0, 0, canvas.width,
canvas.height)

    var watermarkImage = new Image()
    watermarkImage.src = watermark
    watermarkImage.onload = () => {
        if (ctx !== null) {
            ctx.globalAlpha = 1;
            ctx?.drawImage(watermarkImage, 0,
0, width / 5, height / 5)

        }
        canvas.toBlob(function (blob) {
            if (blob !== null) {
                var newFile = new
File([blob], file.name, {
                    type: contentType,
                    lastModified:
Math.floor(new Date().getTime() / 1000)
                })
                resolve(newFile)
            }
        }, 'image/jpeg', 1)
    }
}

```

```
        if (reader.result?.toString() != null) {  
            image.src = reader.result?.toString();  
        }  
    }  
    reader.readAsDataURL(file);  
}}}
```

Фрагмент кода 2. Функция «Нанесения водного знака»

ЗАКЛЮЧЕНИЕ

В результате выполнения Выпускной квалификационной работы бакалавра было разработано веб-приложение, выполняющее функционал фотосервиса для различных спортивных мероприятий.

В ходе разработки были рассмотрены различные способы организации архитектуры нагруженных систем, проведен их сравнительный анализ. Кроме того, архитектура, с лучшими параметрами была использован при разработке системы.

В процессе разработки были выполнены следующие шаги:

- Исследование предметной области;
- Анализ и обработка данных;
- Разработка инфологической и даталогической модели;
- Разработка сервера обработки запросов;
- Разработка интерфейса взаимодействия с сервером;
- Разработка комплекта технической документации и графической части конструкторской документации [1][2][3];
- Подготовка графических материалов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 19.201-78. ЕСПД. Техническое задание. Требования к содержанию и оформлению.
2. ГОСТ 19.301-79. ЕСПД. Программа и методика испытаний. Требования к содержанию и оформлению.
3. ГОСТ 19.404-79. ЕСПД. Пояснительная записка. Требования к содержанию и оформлению.
4. В.М. Постников. Основы эксплуатации АСОИиУ. Ч.1: Техническое обслуживание // М.: Изд-во МГТУ им. Н. Э. Баумана, 2015. – 191 с.
5. Система проведения и судейства спортивных соревнований – MarshalOne [Электронный ресурс]. – URL: <https://marshalone.ru> (дата обращения: 6.04.2022)
6. M. Fowler, J. Lewis. Microservices - a definition of this new architectural term [Электронный ресурс]. – URL: <https://martinfowler.com/articles/microservices.html> (дата обращения: 6.04.2022)
6. Плавшич, В. Отклик облачных хранилищ при загрузке и выгрузке файлов / В. Плавшич, А. Н. Зейн // Молодежный исследовательский потенциал: сборник статей Международного учебно-исследовательского конкурса, Петрозаводск, 25 апреля 2021 года. – Петрозаводск: Международный центр научного партнерства «Новая Наука» (ИП Ивановская Ирина Игоревна), 2021. – С. 9-14. – EDN ROJNXK.
7. И.Д. Шпак, Е.М. Сысойкин, А.И. Антонов. Использование микросервисов для организации взаимодействия клиента с S3 // Искусственный интеллект в автоматизированных системах управления и обработки данных. ИИАСУ'22. Сборник статей всероссийской научной конференции. – М.: ИНФРА-М. – 2022. – Т.1. – С. 391-396.
8. Документация Golang [Электронный ресурс]. – URL: <https://go.dev/doc/> (дата обращения: 12.12.2021)

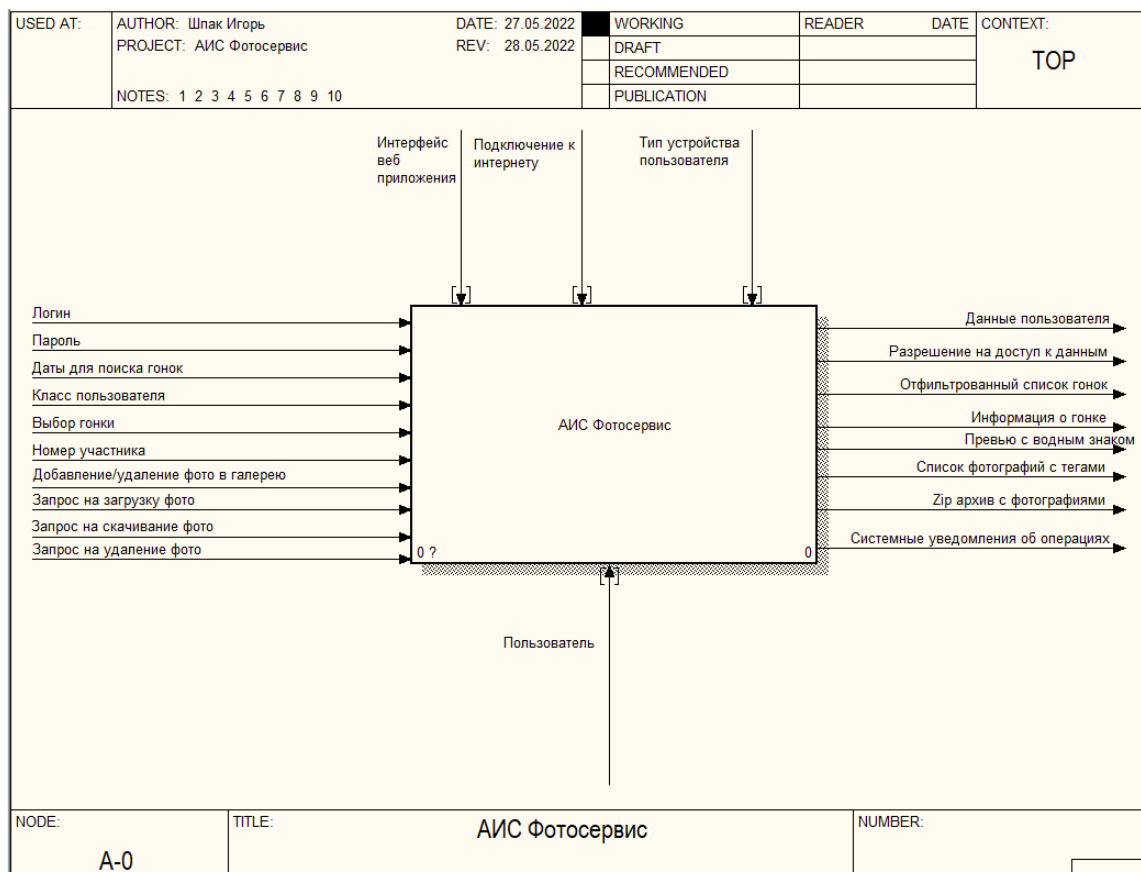
9. Документация библиотеки Gorm [Электронный ресурс]. - URL: <https://gorm.io/docs> (дата обращения: 12.12.2021)
10. Документация и спецификации языка TypeScript [Электронный ресурс]. - URL: <https://www.typescriptlang.org/docs> (дата обращения: 12.03.2022)
11. Документация React URL: <https://ru.reactjs.org/docs/getting-started.html> (дата обращения: 12.03.2022)
12. Документация библиотеки React-Router-DOM [Электронный ресурс]. - URL: <https://reactrouter.com/docs/en/v6> (дата обращения: 12.03.2022)
13. Документация библиотеки Mobx [Электронный ресурс]. - URL: <https://mobx.js.org/api.html> (дата обращения: 12.03.2022)
14. Е.М. Сысойкин, И.Д. Шпак, А.И. Антонов, А.М. Самойлов, А.О. Енин. Распознавание номеров участников соревнований с помощью ансамбля моделей глубокого обучения // Искусственный интеллект в автоматизированных системах управления и обработки данных. ИИАСУ'22. Сборник статей всероссийской научной конференции. – М.: ИНФРА-М. – 2022. – Т.1. – С. 247-254.
15. Документация API Minio для Golang [Электронный ресурс]. - URL: <https://docs.min.io/docs/golang-client-api-reference.html> (дата обращения 12.12.2021)
16. R. Flygare, A. Holmqvist, Performance characteristics between monolithic and microservice-based systems [Электронный ресурс]. – URL: <http://www.diva-portal.org/smash/get/diva2:1119785/FULLTEXT03.pdf>

ПРИЛОЖЕНИЕ А

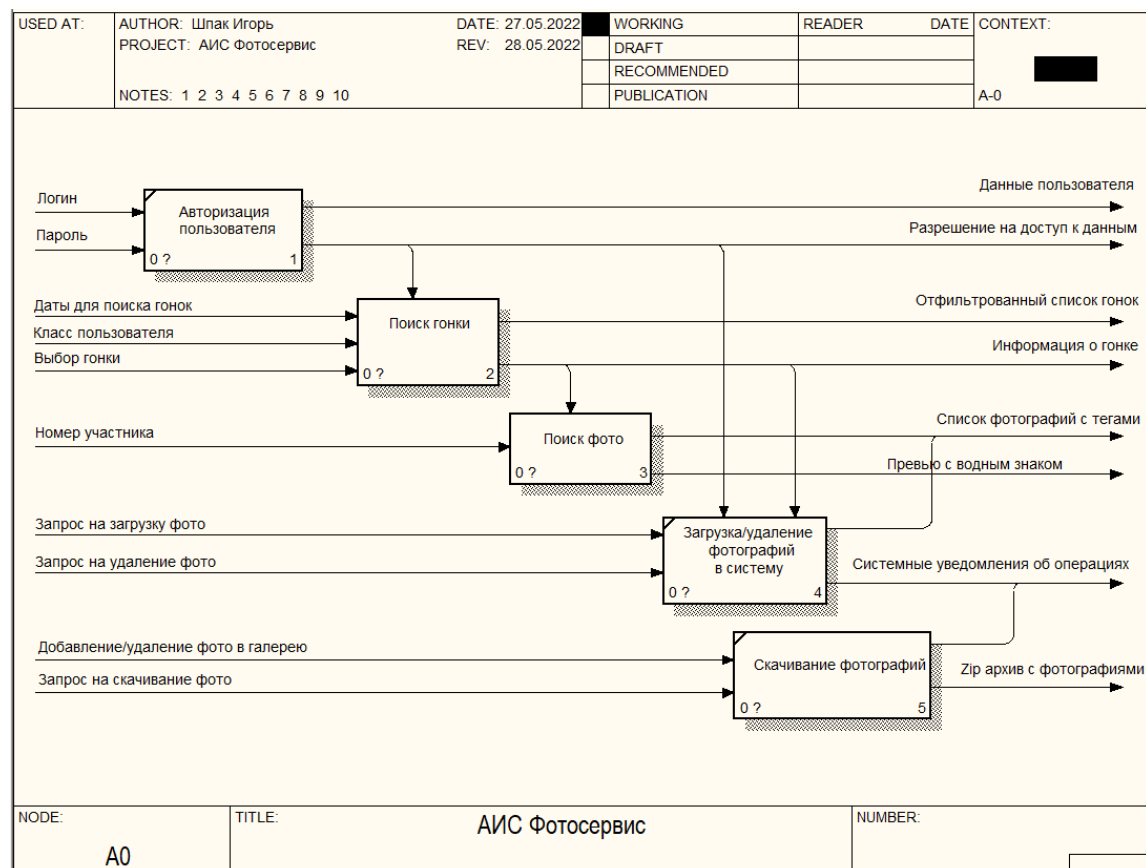
ГРАФИЧЕСКИЕ МАТЕРИАЛЫ

1. Интерфейс пользователя;
2. Работа функций приложения.

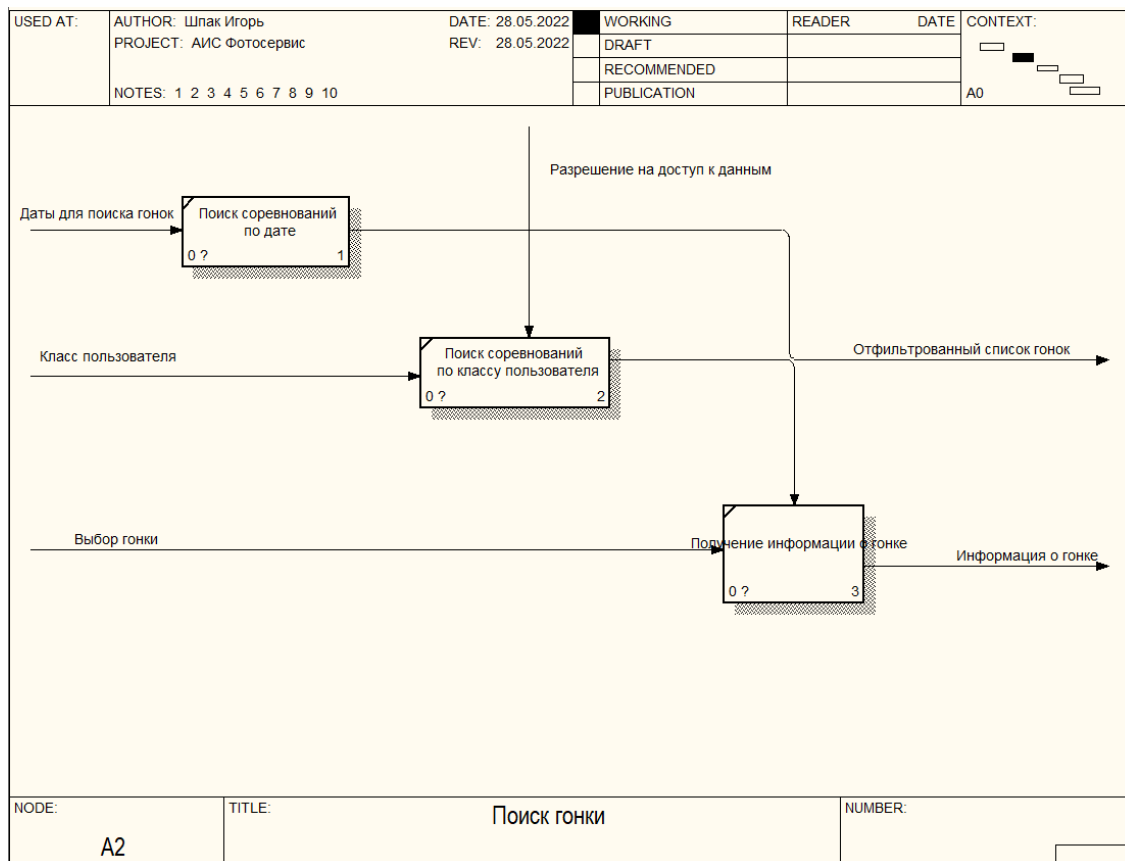
А.1 Контекстная диаграмма разрабатываемой системы



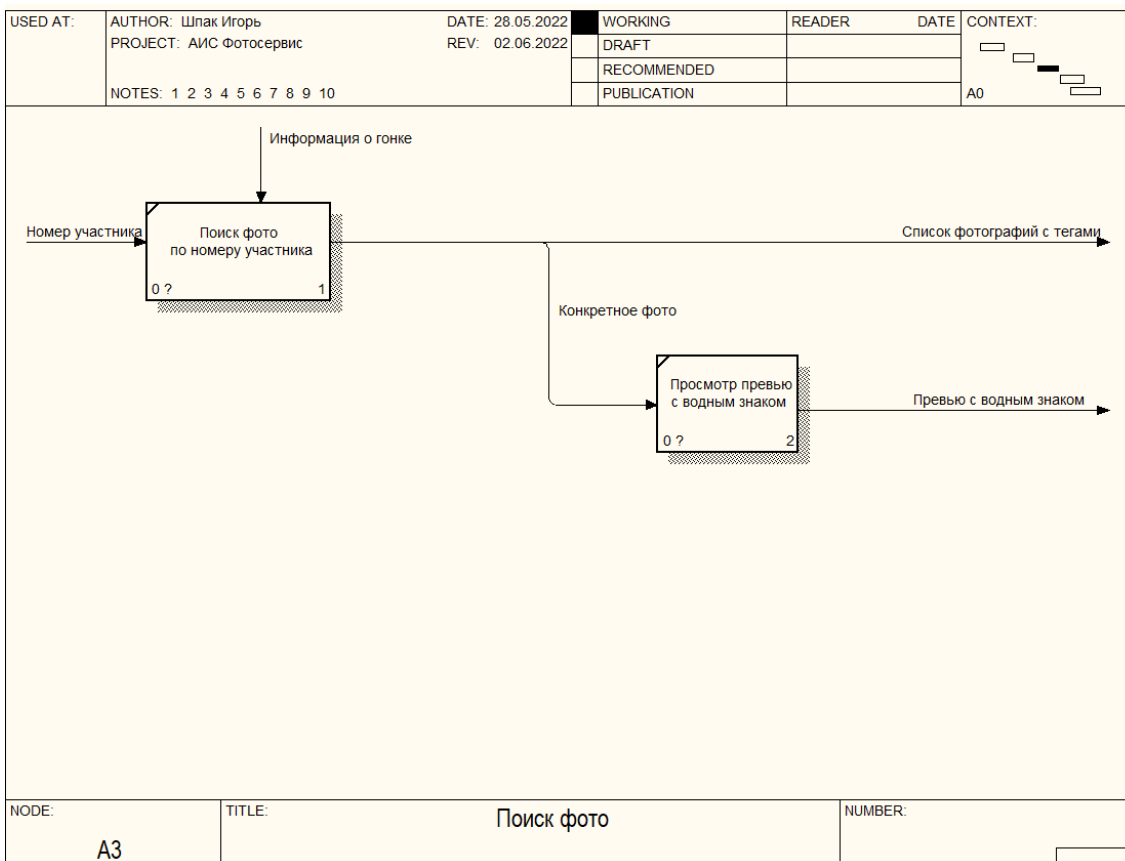
А.2 Подробное описание функций системы



А.3 Функция «Поиск Гонки»




А.4 Блок поиска фото



А.5 Форма авторизации пользователя

Номер телефона



8 (912) 345-67-89

Пароль

Получить пароль

Ok

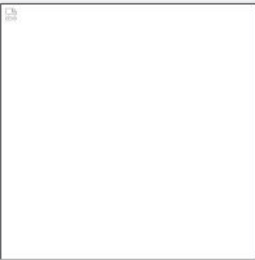
А.6 Страница со списком гонок

PHOTO.MARSHALONE


Шляк Игоря

Период дат: 02/28/2022 - 08/28/2022


По типу: Все гонки




Way offroad
Город: Кострома
Дата проведения: 2022-07-09




Беной - II этап Чемпионата России по экстрим эндуро
Город: Беной (Чечня)
Дата проведения: 2022-07-02




"Тропа Кабана". Заруба!
Город: Ульяновск
Дата проведения: 2022-06-18




Поворот не туда 2022
Город: Куцевская
Дата проведения: 2022-06-11




ВелоФест
Город: Саратов
Дата проведения: 2022-06-11




The Wall Race
Город: Уфа
Дата проведения: 2022-06-11




Enduro Titans
Город: Дальнегоorsk
Дата проведения: 2022-06-04




Шурале
Город: Зеленодольск
Дата проведения: 2022-06-04



Husky Fest
Город: Молькино
Дата проведения: 2022-05-28



ГОЙКА МПГ 2022
Город: Находка
Дата проведения: 2022-05-22



EVORACE 2DAY
Город: Находка
Дата проведения: 2022-05-22


А.7 Отфильтрованный по классу Медиа список гонок

PHOTO.MARSHALONE

Шлак Игорь

Период дат: 07/01/2021 - 08/05/2021

По типу: Медиа



OFRR 2021
Город: Венёв Тульская область
Дата проведения: 2021-07-03

А.8 Страница соревнования

←

PHOTO.MARSHALONE

Шлак Игорь

Распознанные фото: ☒

Номер участника:

К пользовательской галерее

I этап ЧР по экстрим эндуро Полуфиналы

12-го апреля 2022
Энгельс (Парк покорителей космоса)

<

1

2

3

4

5

6

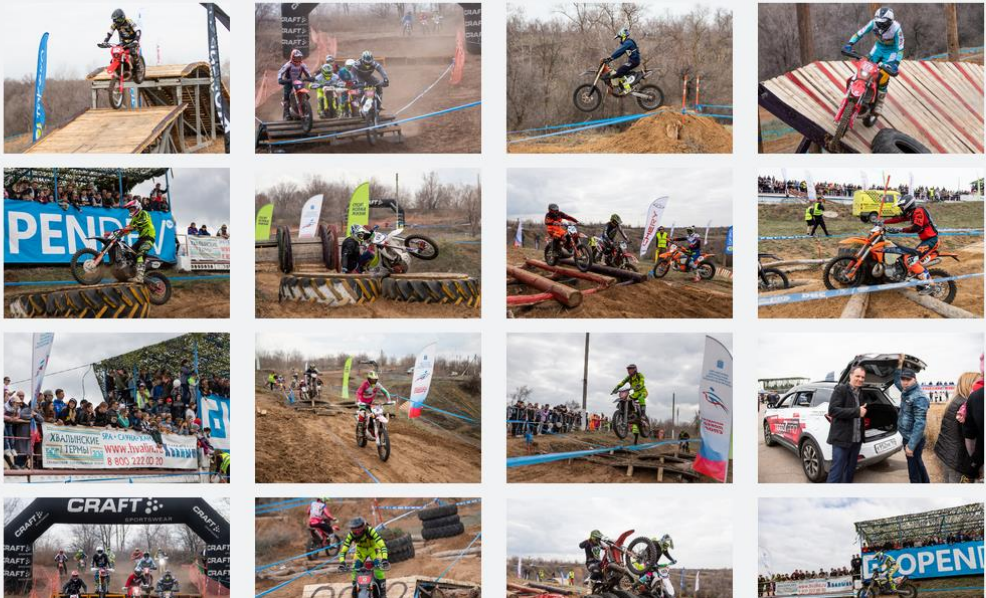
7

8

...

17

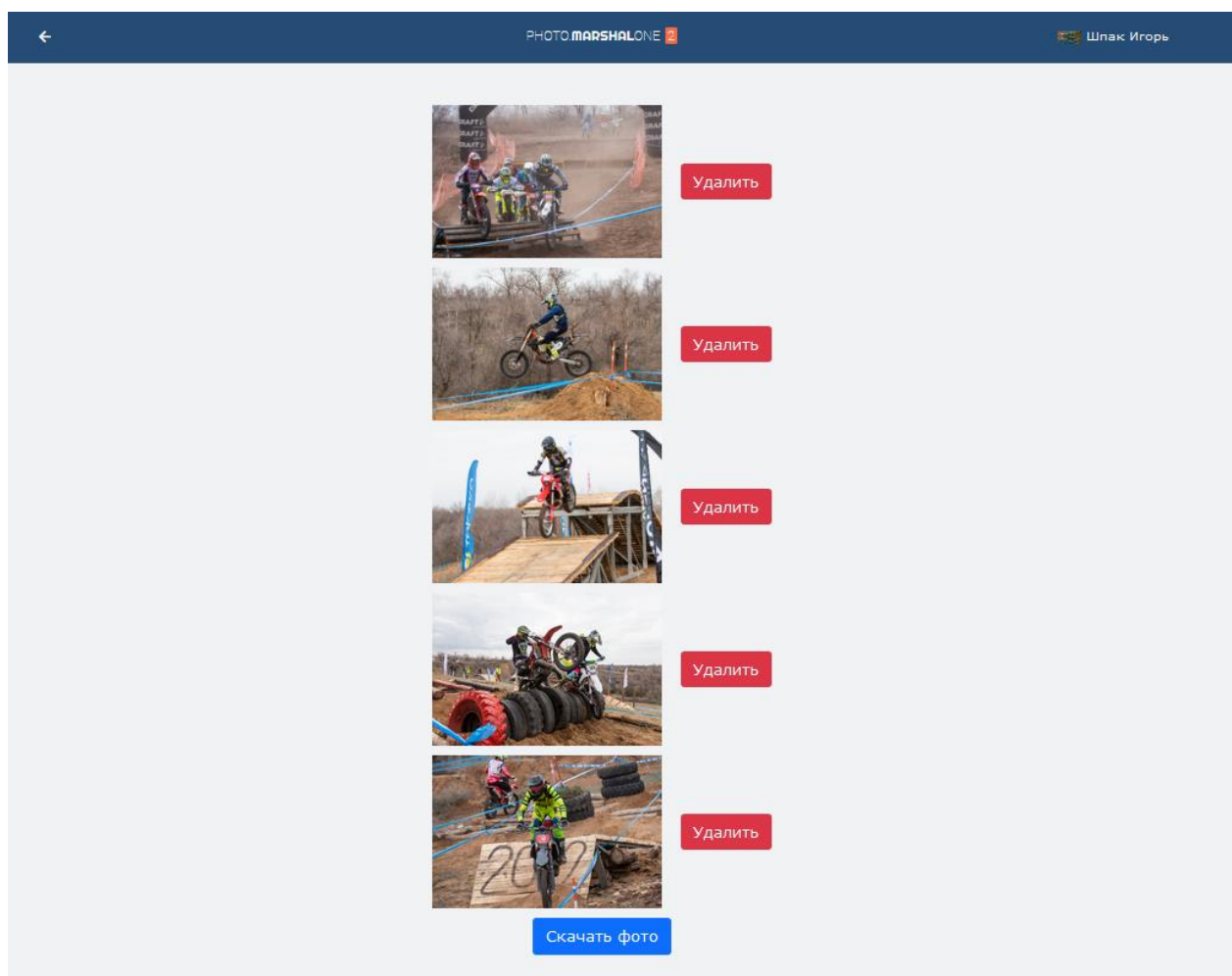
>



А.9 Страница просмотра фотографии



А.10 Пользовательская галерея фотографий



А.11 Страница загрузки изображений в систему

←

PHOTO.MARSHALONE





Шлак Игорь

Выбор фото для загрузки

Название соревнования: Выберите файлы для загрузки

Отправить изображения

Загруженные изображения:



ПРИЛОЖЕНИЕ В
ТЕХНИЧЕСКОЕ ЗАДАНИЕ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет «Информатика, искусственный интеллект и системы управления»
Кафедра «Системы обработки информации и управления»

Утверждаю
Научный руководитель
_____ Антонов А.И.
" __ " _____ 2022 г.

**Автоматизированная информационная система для
участников соревнований - «АИС Фотосервис»**

Техническое задание
(вид документа)

писчая бумага
(вид носителя)

6
(количество листов)

ИСПОЛНИТЕЛЬ:

_____ Шпак Игорь Денисович
" __ " _____ 2022 г.

1. Наименование

Автоматизированная информационная система для участников соревнований - «АИС Фотосервис»

2. Основание для разработки

Основанием для разработки является задание на выпускную работу, подписанное руководителем выпускной работы и утверждённое заведующим кафедрой ИУ5 МГТУ им.Баумана.

3. Исполнитель

Исполнителем данной работы является студент четвёртого курса, группы ИУ5-84Б Шпак И.Д.

4. Назначение и цель работы

Целью работы является проектирование и разработка системы, позволяющей осуществлять поиск фотографий участников соревнований, загруженных фотографиями, по данным участника.

Назначением работы является создание приложения, позволяющего выполнять целевой функционал.

5. Содержание работы

5.1 Задачи

1. Исследовать предметную область, определить функциональные задачи.
2. Разработать архитектуру программного обеспечения.
3. Собрать данные предметной области.
4. Структурировать и подготовить данные предметной области.
5. Разработать алгоритмы взаимодействия с пользователем.
6. Реализовать веб-приложение и сервер обработки запросов.
7. Провести тестирование информационно - программного продукта.
8. Провести отладку программного продукта.
9. Оформить техническую документацию.

5.2 Требования к функциональным характеристикам

Разрабатываемая система должна выполнять следующие функции:

12. Авторизация пользователей системы.
13. Просмотр всех проводимых соревнований.
14. Фильтр соревнований по классу пользователя.
15. Просмотр всех фото, относящихся к конкретному соревнованию.
16. Поиск фото по номеру участника
17. Просмотр номеров изображенных на фото участников
18. Просмотр превью фото в повышенном качестве с водным знаком
19. Загрузка фото, зарегистрированными пользователями, с соответствующими правами доступа
20. Выбор фото для скачивания
21. Скачивание выбранных пользователем фото
22. Удаление фото загрузившими их пользователями

5.3 Требования к архитектуре программного изделия

Программный продукт представляет собой веб-приложение, реализующее интерфейс взаимодействия с сервером, а также собственный сервер обработки запросов.

5.4 Требования к входным и выходным данным

5.4.1 Требования к входным данным

Данные, получаемые от пользователя:

- Изображения участников, сделанные во время соревнования
- Информация для поиска фотографий, включающая номер участника и идентификатор соревнования

5.4.2 Требования к выходным данным

Выходные данные представляют результат обработки входных данных, а именно фото, с номерами изображенных на нем участников.

5.5 Требования к надежности

Не должна выдавать ошибок, не предусмотренных работой; программа должна надежно и устойчиво функционировать. Должна иметь возможность быть восстановленной в течение часа, в случае непредвиденных проблем функционирования.

5.6 Лингвистические требования

Клиентская часть веб-приложения должна быть русифицирована. Вся документация к программному продукту должна быть русифицирована.

5.7 Требования к составу программных средств

Для работы клиентской части данного приложения необходимо, чтобы на компьютере были установлены следующие программные продукты:

1. Microsoft Windows 7 и выше, Mac OS X 10.6 и выше, Ubuntu 10.04 и выше;
2. Google Chrome версии 96 и выше, либо браузер с аналогичным функционалом

Для работы серверной части данного приложения необходимо, чтобы на компьютере были установлены следующие программные продукты:

1. ОС на базе ядра GNU/Linux версии 5.4 и выше, либо Windows 10 и старше.
2. Docker версии 20.10.14 и выше.
3. Docker-compose версии 3.7 и выше.
4. Nginx версии 1.21 и выше.

5.8 Требования к составу технических средств

- Клиентская часть
 1. Процессор с тактовой частотой – 2 ГГц;
 2. Оперативная память – 4 Гб;
 3. Видеоадаптер и монитор, способные обеспечить графический режим 1024*768 точек с 32-битной цветопередачей;
 4. Манипулятор “мышь” или другое указывающее устройство;
 5. Клавиатура.
 6. Сетевой адаптер.
- Серверная часть
 1. Процессор с тактовой частотой – 1 ГГц;
 2. Оперативная память – 1 Гб;

3. Жёсткий диск со свободным объемом памяти не менее 10 Гб;
4. Сетевой адаптер.

6. Этапы работы

График выполнения отдельных этапов работ приведен в соответствии с приказом об организации учебного процесса в 2021/2022 учебном году.

Таблица 1: Этапы разработки

№ п/п	Наименование этапа и содержание работ	Сроки исполнения
1	Исследование предметной области	Март 2022 г.
2	Разработка и утверждение ТЗ	Апрель 2022 г.
3	Разработка архитектуры программного обеспечения	Апрель 2022 г.
4	Создание программ	Апрель — Май 2022 г.
5	Тестирование и отладка	Май 2022 г.
6	Оформление документации	Май — Июнь 2022 г.
7	Защита работы	Июнь 2022 г.

7. Техническая документация

По окончании работы предъявляется следующая техническая документация:

1. Техническое задание;
2. Расчётно-пояснительная записка;
3. Программа и методика испытаний;
4. Графический материал по проекту в формате презентации.

8. Порядок приема работы

Приём и контроль программного изделия осуществляется в соответствии с методикой испытаний (см. документ «Программа и методика испытаний»).

9. Дополнительные условия

Данное техническое задание может уточняться в установленном порядке.

ПРИЛОЖЕНИЕ С
ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

Факультет «Информатика, искусственный интеллект и системы управления»
Кафедра «Системы обработки информации и управления»

Утверждаю
Научный руководитель
_____ Антонов А.И.
" __ " _____ 2022 г.

**Автоматизированная информационная система для
участников соревнований - «АИС Фотосервис»**

Программа и методика испытаний
(вид документа)

писчая бумага
(вид носителя)

8
(количество листов)

ИСПОЛНИТЕЛЬ:

_____ Шпак Игорь Денисович
" __ " _____ 2022 г.

1. Объект испытаний

Объектом испытаний является автоматизированная информационная система для участников соревнований, состоящая из веб-приложения и сервера обработки запросов.

2. Цель испытаний

Испытания проводятся с целью проверки соответствия клиента требованиям к функциональным характеристикам, описанным в п. 5.2 Технического задания.

3. Состав предъявляемой документации

На испытания программного продукта представляются следующие документы:

- 1) Техническое задание
- 2) Программа и методика испытаний

4. Технические требования

4.1. Требования к программной документации

Комплектность программной документации должна удовлетворять разделу данного документа “Состав предъявляемой документации”.

4.2. Требования к техническим характеристикам

4.2.1. Требования к составу аппаратного обеспечения

Данная программа должна работать на компьютере следующей конфигурации:

- Клиентская часть
- 7. Процессор с тактовой частотой – 2 ГГц;
- 8. Оперативная память – 4 Гб;

9. Видеоадаптер и монитор, способные обеспечить графический режим 1024*768 точек с 32-битной цветопередачей;
10. Манипулятор “мышь” или другое указывающее устройство;
11. Клавиатура;
12. Сетевой адаптер.

- Серверная часть

Минимальные требования для работы серверной части:

5. Процессор с тактовой частотой – 1 ГГц;
6. Оперативная память – 1 Гб;
7. Жёсткий диск со свободным объемом памяти не менее 10 Гб;
8. Сетевой адаптер.

Требования к составу программного обеспечения

Для работы данного модуля необходимо, чтобы на компьютере были установлены следующие программные продукты:

- Клиентская часть

3. Microsoft Windows 7 и выше, Mac OS X 10.6 и выше, Ubuntu 10.04 и выше;
4. Google Chrome версии 96 и выше, либо браузер с аналогичным функционалом

- Серверная часть

1. ОС на базе ядра GNU/Linux версии 5.4 и выше, либо Windows 10 и старше.
2. Docker версии 20.10.14 и выше.
3. Docker-compose версии 3.7 и выше.
4. Nginx версии 1.21 и выше.

5. Порядок проведения испытаний

Испытание данного программного продукта будут проводиться в следующем порядке:

- 1) Запуск клиентского приложения, заполнение формы авторизации.
- 2) Тестирование базовых операций системы.

В процессе проведения приемочных испытаний должна быть протестирована работоспособность следующих компонентов клиента:

1. Модуль авторизации;
2. Модуль просмотра гонок;
3. Модуль просмотр фото;
4. Модуль загрузки фото;
5. Модуль скачивания фото.

Все компоненты испытываются одновременно на корректность и влияние друг на друга, т.е. испытания проводятся комплексно.

6. Методы испытаний

№	Наименование / № пункта в ТЗ	Выполняемые действия	Результат
1	5.2.1 Авторизация зарегистрированных пользователей	Нажать кнопку «Войти», после чего заполнить форму авторизации и нажать кнопку «Ок».	Переход на главный экран авторизованного пользователя.
2	5.2.2 Просмотр всех проводимых соревнований	На главной странице установить флаг «Все гонки» и в календаре	Отображаются все гонки за указанный период.

		выбрать период поиска	
3	5.2.2 Просмотр всех проводимых соревнований	Прокрутка главной страницы вниз	Подгружаются новые гонки за более старый период, с учетом выбранного класса пользователя.
4	5.2.3 Фильтр соревнований по классу пользователя.	На главной странице изменить флаг «Все гонки» на «Участник» или «Медиа», в календаре указать период поиска	Отображаются все гонки за указанный период, в котором пользователь является участником, либо представляет класс медиа
5	5.2.4 Просмотр всех фото, относящихся к конкретному соревнованию	На главной странице выбрать гонку, дата проведения которой меньше текущей даты	Переход на страницу с гонкой
6	5.2.4 Просмотр всех фото, относящихся к конкретному соревнованию	На странице гонки переключения флага «Распознанные фото»	Изменяется список отображаемых фото
7	5.2.4 Просмотр всех фото, относящихся к конкретному соревнованию	Переключение страниц в компоненте пагинации.	Отображение выбранной страницы с фото.

8	5.2.5 Поиск фото по номеру участника	На странице с гонкой и установленным флагом «Распознанные фото» ввести в поле «Номер участника» желаемый номер	Вывод фотографий, на которых изображен участник с выбранным номером
9	5.2.6 Просмотр номеров изображенных на фото участников 5.2.7 Просмотр превью фото в повышенном качестве с водным знаком	На странице с гонкой нажать на превью фото	Отображение диалогового окна содержащего фото в повышенном качестве с водным знаком и набор номеров изображенных на фото участников
10	5.2.8 Загрузка фото зарегистрированными пользователями с соответствующими правами доступа	На странице с гонкой, нажать кнопку загрузить фото	Переход на страницу загрузки фото
11	5.2.8 Загрузка фото зарегистрированными пользователями с соответствующими правами доступа	На странице загрузки фото нажать на кнопку «Выбрать фото для загрузки», выбрать желаемые фото и нажать кнопку «Отправить	Выбранные фото загрузятся и отобразятся на странице соответствующей гонки

		изображения»	
12	5.2.9 Выбор фото для скачивания	На странице гонки навести курсор на выбираемое фото и нажать кнопку «Добавить фото»	Выбранные фото добавятся в корзину, для последующей загрузки
13	5.2.9 Выбор фото для скачивания	Нажать на иконку корзины в верхнем правом углу	Переход на страницу с корзиной, содержащей список выбранных фото
14	5.2.9 Выбор фото для скачивания	На странице с корзиной нажать кнопку «Удалить» рядом с фото, которое необходимо удалить из корзины	Выбранное фото будет удалено из корзины
15	5.2.10 Скачивание выбранных пользователем фото	На странице с корзиной нажать кнопку «Загрузить выбранные фото»	Выбранные фото будут загружены в виде zip архива.
16	5.2.11 Удаление фото загружившими их пользователями	На странице с загрузкой фото нажать кнопку удалить напротив	Вывод сообщения об удалении фото / ошибке удаления фото

		соответствующего фото	
--	--	--------------------------	--

7. Результат испытаний

В результате испытаний должна быть успешное выполнение всех действий, описанных в пункте 6 данного документа и подтверждение соответствия системы требованиям, описанным в техническом задании.