



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 ИУ «Информатика и системы управления»

---

КАФЕДРА ИУ-7 «Программное обеспечение ЭВМ и информационные технологии»

---

## РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

### *К КУРСОВОЙ РАБОТЕ*

### *НА ТЕМУ:*

Разработка базы данных мероприятий по направлению  
«BigData»

Студент группы ИУ7-61Б

\_\_\_\_\_  
(Подпись, дата)

**Я. Р. Байрамгалин**

\_\_\_\_\_  
(Н.О. Яковидис)

Руководитель

\_\_\_\_\_  
(Подпись, дата)

**Н. О. Яковидис**

\_\_\_\_\_  
(И.О. Фамилия)

2024 г.

## РЕФЕРАТ

Курсовая работа 29 с., 3 рис., 4 табл., 0 ист.

СУБД, БД, PostgreSQL, C++, Linux, userver.

Цель работы: спроектировать и разработать базу данных мероприятий по направлению «BigData».

Результат работы:

- формализована задача и определен необходимый функционал;
- описана структура объектов базы данных, а также сделан выбор СУБД для ее хранения и взаимодействия;
- спроектирован и разработан триггер, поддерживающий время обновления записи в актуальном состоянии;
- спроектирован и разработан интерфейс взаимодействия с базой данных;
- проведено исследование времени работы операции с использованием индексом, а также без таковых.

# СОДЕРЖАНИЕ

<b>ВВЕДЕНИЕ</b>	<b>8</b>
<b>1 Аналитический раздел</b>	<b>10</b>
1.1 Требования к приложению . . . . .	10
1.2 Анализ существующих решений . . . . .	11
1.3 Пользователи системы . . . . .	11
1.4 Формализация данных . . . . .	12
1.5 Анализ моделей баз данных . . . . .	13
1.5.1 Объектная модель данных . . . . .	13
1.5.2 Иерархическая модель . . . . .	14
1.5.3 Реляционная модель . . . . .	15
1.5.4 Вывод . . . . .	15
<b>2 Конструкторский раздел</b>	<b>17</b>
2.1 Ролевая модель . . . . .	18
2.2 Триггер . . . . .	19
2.3 Проектирование приложения . . . . .	19
2.4 Вывод . . . . .	19
<b>3 Технологический раздел</b>	<b>20</b>
3.1 Выбор объектной базы данных . . . . .	20
3.1.1 Microsoft Azure Blob Storage . . . . .	20
3.1.2 Yandex S3 . . . . .	20
3.1.3 Результат выбора объектной базы данных . . . . .	21
3.2 Выбор реляционной базы данных . . . . .	21
3.2.1 Oracle Database . . . . .	21
3.2.2 MySQL . . . . .	22
3.2.3 PostgreSQL . . . . .	22
3.2.4 Результат выбора реляционной базы данных . . . . .	23
3.3 Сущности объектной базы данных . . . . .	23
3.4 Сущности основной базы данных . . . . .	23
3.4.1 Таблицы . . . . .	23
3.4.2 Индекс . . . . .	23
3.4.3 Триггер . . . . .	23

3.4.4	Роли . . . . .	24
3.5	Интерфейс доступа к данным . . . . .	25
3.6	Вывод . . . . .	26
<b>4</b>	<b>Исследовательский раздел</b>	<b>27</b>
4.1	Цель исследования . . . . .	27
4.2	Описание исследования . . . . .	27
4.3	Вывод . . . . .	29

## ВВЕДЕНИЕ

В современном мире огромное количество данных генерируется ежедневно, и их анализ становится все более важным для принятия обоснованных решений в различных областях науки и бизнеса. Термин «Big Data» определяется как данные огромных объемов и разнообразия, а также методы их обработки, которые помогают в распределенном масштабе анализировать информацию[1]. Обработка данных значительного объема требует использования специальных инструментов и знаний. Именно поэтому передача такого опыта имеет ключевое значение в современном образовании.

Для передачи знаний, а также обсуждения и решения актуальных задач в области обработки больших объемов данных в России проводятся следующие мероприятия[1]: вебинары, воркшопы, выставки, конгрессы, конференции. Организация таких мероприятий связана с рассмотрением организаторами заявок от потенциальных участников. Автоматизация данного процесса может сократить нагрузку на организаторов, а также сделать процесс более понятным и прозрачным для участников.

Целью курсовой работы является проектирование и разработка базы данных мероприятий по направлению «BigData».

Для достижения поставленной цели, необходимо решить следующие задачи:

- определить необходимый функционал приложения, предоставляющего доступ к базе данных;
- выделить роли пользователей, а также формализовать данные;
- проанализировать системы управления базами данных и выбрать подходящую систему для хранения данных;
- спроектировать базу данных, описать ее сущности и связи, спроектировать триггер;
- реализовать интерфейс для доступа к базе данных.

Итогом работы станет приложение, предоставляющее интерфейс доступа к базе данных.

## **1 Аналитический раздел**

Аналитический раздел курсовой работы играет ключевую роль в процессе разработки приложения, поскольку он устанавливает основные характеристики системы и выбирает наиболее подходящую модель базы данных. В этой части работы будут изучены различные виды баз данных, их достоинства и недостатки, а также проведен анализ существующих систем управления базами данных. На основе результатов будет выбрана наиболее подходящая модель данных для реализации приложения. Кроме того, в данной секции будут определены акторы системы, будут формализованы данные, а сущности будут представлены отношения между сущностями, что поможет более точно определить требования к приложению и спроектировать его структуру.

### **1.1 Требования к приложению**

Так как подача заявки на конференцию, а также модерация может происходить с разных платформ, решено создать серверное приложение, предоставляющее интерфейс для взаимодействия с данными. Таким образом в дальнейшем для любой платформы можно будет создать клиент, который будет иметь возможность использовать всю функциональность системы.

Система должна поддерживать следующий функционал:

- возможность авторизации через sso[3] для каждого пользователя;
- создание, редактирование, удаление заявки на участие для потенциального докладчика на различные события;
- просмотр списка мероприятий для каждого пользователя;
- добавление приложений объемом до 10МБ каждое к заявке;
- просмотр статуса своих заявок;
- возможность взаимодействия докладчика и модератора через комментарии к заявке;
- утверждение или отклонения заявки модератором.

## 1.2 Анализ существующих решений

Для анализа существующих решений были рассмотрены следующие веб-приложения:

- 1) Онтико — веб-приложение, используемое при проведении конференций highload++, а также некоторых других. Не имеет возможности авторизации через sso, имеет возможность записи на разные конференции.
- 2) YaConf — веб-приложение, используемое при проведении конференций YaConf. Имеется возможность авторизации через аккаунт Яндекса, однако через подать заявку можно только на конференции, проводимые Яндексом.
- 3) Jugru — веб-приложение, используемое для подачи заявок на конференцию сррсоп и некоторые другие. Недоступно для мероприятий в России, не имеет входа через sso, есть возможность записи на разные конференции.

Таким образом, ни одно из существующих решений не удовлетворяет всем требованиям, выдвинутым выше.

## 1.3 Пользователи системы

В приложении выделены следующие роли пользователей:

- 1) Потенциальный докладчик (далее просто докладчик) — пользователь, имеющий возможность авторизоваться, создать заявку на мероприятие, обновить текущую заявку (добавлением комментария к ней), отклонить собственную заявку. Также докладчик может посмотреть список доступных для подачи заявок мероприятий и добавить приложений (файл) к своей заявке.
- 2) Модератор — пользователь, который обладает правами докладчика, а также может просматривать весь список заявок на мероприятия, создавать, редактировать и удалять мероприятия и одобрять или отклонять заявки.
- 3) Администратор — пользователь, обладающий возможностями модерато-



ра, а также возможностью назначать и удалять модераторов.

На рисунке 1 представлена диаграмма использования приложения.

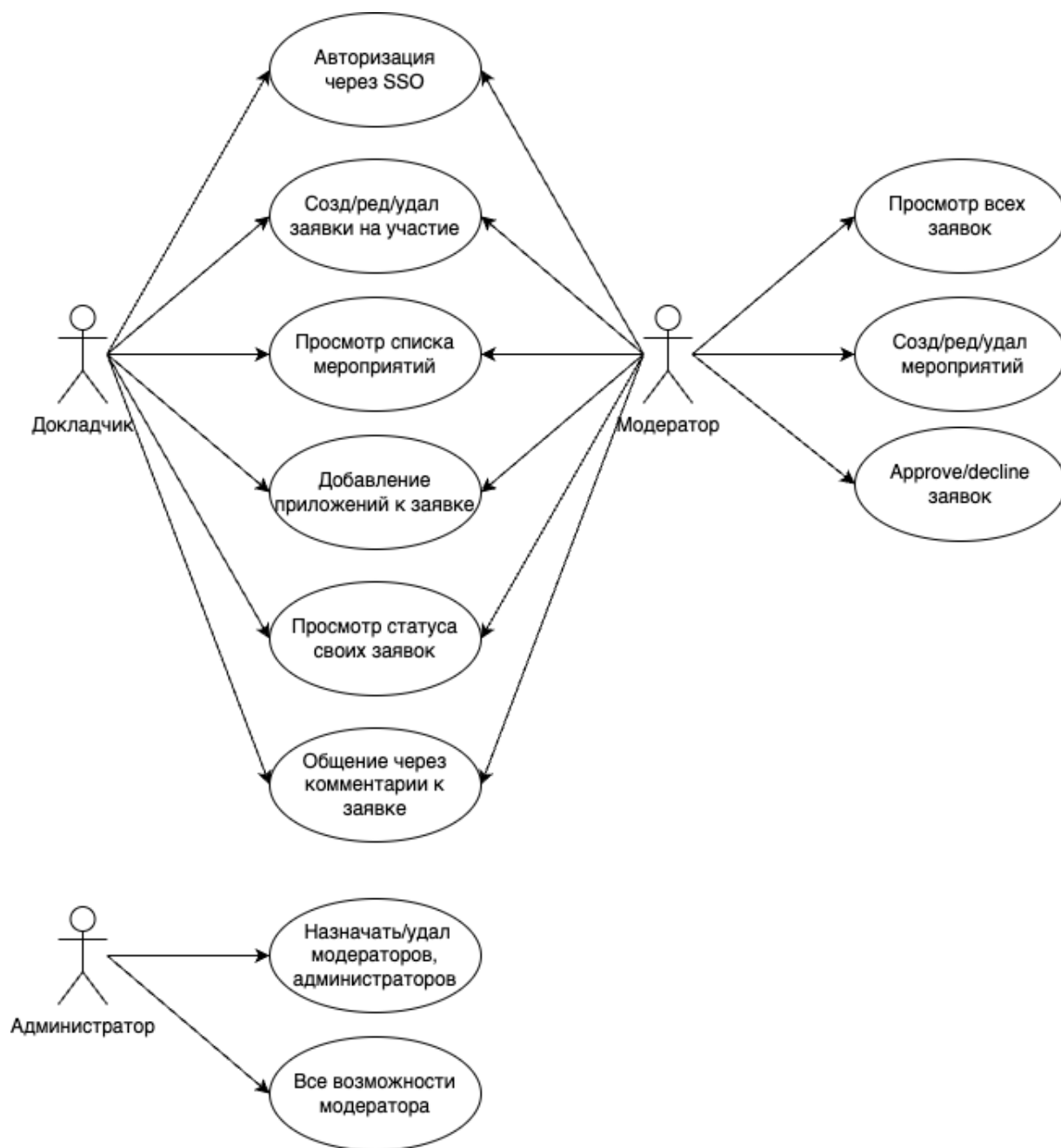


Рисунок 1 – Диаграмма использования приложения

#### 1.4 Формализация данных

База данных должна хранить информацию о следующих сущностях:

- мероприятие;
- пользователь;
- группа доступов;

- заявка;
- комментарий к заявке;
- приложение (бинарный файл);
- мета-информация о приложении.

В таблице 1 представлены сущности и сведения о них.

Таблица 1 – Сущности и сведения о них

<b>Сущность</b>	<b>Сведения</b>
Мероприятие	Название, описание, дата проведения, продолжительность
Пользователь	Имя, пол, телефон, email-адрес, метка времени первого входа
Группа доступов	Название, описание
Заяка	Событие, автор, описание, статус, метка времени создания
Комментарий к заявке	Текст, событие, автор, время создания
Приложение	Бинарные данные
Мета-информация о приложении	Имя файла, хеш, размер, метка времени загрузки

## 1.5 Анализ моделей баз данных

Перед выбором подходящей системы управления базами данных были рассмотрены 3 модели базы данных.

### 1.5.1 Объектная модель данных

Объектные хранилища (Object Stores) представляют собой специализированные хранилища данных, в которых информация хранится в виде объектов. Эти объекты могут содержать не только сами данные, но и дополнительные метаданные, методы и связи с другими объектами. Объектные хранилища широко

применяются в различных областях информационных технологий, таких как разработка программного обеспечения, облачные вычисления, хранение данных больших объемов и др.

В данной работе объектная модель данных подходит для хранения приложений (бинарных файлов). Ключом (путем к файлу) может выступать уникальный идентификатор файла, создаваемый на уровне приложения, а объектом — сами набор байтов (содержимое файла).

### **1.5.2 Иерархическая модель**

Иерархическая модель данных представляет базу данных как древовидную структуру, где объекты различных уровней связаны между собой. Каждый объект может содержать несколько объектов более низкого уровня, устанавливая отношение предка к потомку. Близнецами называются объекты, имеющие общего предка, а в программировании структура дерева также известна как "братья".

Иерархическую модель данных стоит использовать в случаях, когда данные могут быть организованы в виде древовидной структуры с четко выраженными отношениями предок-потомок. Это может быть полезно, когда данные имеют естественную иерархию, например, в организационных структурах, классификациях товаров, семейных деревьях и т. д. Иерархическая модель также подходит для ситуаций, где данные имеют фиксированную структуру и не требуют сложных запросов или изменений в структуре данных. Однако следует учитывать, что иерархическая модель может ограничить гибкость и эффективность обработки данных в случае изменений в их структуре или потребности в сложных запросах.

В данных формализованных выше отношение предок-потомок имеют только сущности заявка-комментарий, поэтому иерархическая модель не подходит для применения в данной курсовой работе.

### **1.5.3 Реляционная модель**

Реляционная модель данных - основана на понятии математических отношений. В реляционной модели данные и связи представлены в виде таблиц, каждая из которых имеет несколько столбцов с уникальными именами[2]. Основной идеей такой модели является использование ключей для связывания таблиц между собой. Ключ - это уникальный идентификатор, который позволяет однозначно определить каждую строку в таблице. Путем использования ключей можно устанавливать связи между таблицами для получения более сложной информации.

Модель баз данных в форме таблиц обладает рядом преимуществ. Во-первых, при правильном использовании гарантируется высокая надежность и целостность данных. Во-вторых большое количество самых популярных современных СУБД (Postgres, Oracle, MySql и другие) основаны на реляционной модели данных.

Однако у этой модели также есть недостатки. Например, она не всегда эффективна при работе с большими объемами информации, так как выполнение сложных запросов может занимать много времени. Кроме того, модель в форме таблиц не всегда подходит для хранения и обработки неструктурированных данных, таких как длинные тексты и бинарные файлы.

В рамках данной работы реляционная модель данных подходит лучше остальным рассмотренным для всех данных кроме сущности «Приложение». Во-первых, все оставшиеся сущности легко представимы в табличной форме, во-вторых использование внешних ключей поможет гарантировать целостность данных.

### **1.5.4 Вывод**

Для хранения сущности «Приложение» лучше подходит объектная модель, содержание файла не находится в каких-либо отношениях с другими сущностями, а значит использование реляционной модели данных не является целесообразным. Также содержимое файла не образует какую-либо иерархию,

что говорит о невозможности применения иерархической модели.

Для хранения остальных сущностей реляционная модель подходит лучшим образом, так как данные легко представимы в табличном виде, а также реляционная модель лучше остальных рассмотренных гарантирует целостность данных, что позволит избежать ошибок во время эксплуатации приложения.

## 2 Конструкторский раздел

В данном разделе курсовой работы будет рассмотрен процесс проектирования базы данных для разрабатываемого приложения. Будут выделены ключевые этапы проектирования, подробно проанализированы действия в рамках ролевой модели, спроектирован триггер и описаны основные принципы проектирования приложения. Целью данного раздела будет создание базы данных, которая обеспечит стабильную работу приложения и удовлетворит потребности пользователей.

На основе выделенных ранее сущностей спроектированы следующие объекты базы данных.

1) Users — содержит информацию об участнике и включает следующие поля:

- user\_id — уникальный идентификатор пользователя;
- token — авторизационный токен пользователя;
- login — логин пользователя;
- name — имя пользователя;
- phone — телефон пользователя;
- created\_at — метка времени первой авторизации.

2) Events — содержит информацию о событиях и включает следующие поля:

- event\_id — уникальный идентификатор события;
- name — название события.

3) Requests — содержит информацию заявках на участие и включает следующие поля:

- request\_id — уникальный идентификатор заявки;
- event\_id — идентификатор конференции, на которую подана заявка;
- user\_id — пользователь, подавший заявку;
- description — описание заявки;
- status — статус заявки;

- `created_at` — метка времени создания заявки;
  - `updated_at` — метка времени обновления заявки.
- 4) `Permissions` — содержит информацию о правах, доступных пользователям, и включает следующие поля:
- `slug` — строковый идентификатор заявки;
  - `user_id` — идентификатор пользователя
- 5) `FileMeta` — содержит мета-информация о файлах и включает следующие поля:
- `uuid` — уникальный идентификатор файла;
  - `source_name` — изначальное имя файла;
  - `hash` — хеш файла;
  - `created_at` — время загрузки файла.
- 6) `File` — содержит информацию об улове участника и включает следующие поля:
- `uuid` — уникальный идентификатор файла;
  - `binary_data` — бинарные данные.

## 2.1 Ролевая модель

Ролевая модель предполагает наличие трех ролей: участника, модератора и администратора. Стоит отметить, что модератор обладает всеми правами участника, а администратор — всеми правами модератора.

Для реализации поставленной выше задачи программа должна предоставлять следующие возможности всем пользователям:

- авторизация через sso;
- создание, редактирование, удаление заявки на участие;
- просмотр списка мероприятий;
- добавление приложений к заявке;
- просмотр статуса своих заявок;
- общение через комментарии к заявке.

Модератор также имеет право:

- просматривать все заявки;
- создавать, редактировать, удалять мероприятия;
- одобрять и отклонять заявки;
- назначать и удалять модераторов, администраторов;

Администратор обладает всеми правами модератора, но кроме того может назначать и удалять модераторов, других администраторов.

## **2.2 Триггер**

Триггер — это хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено действием по изменению данных: добавлением, модификацией, удалением строки в заданной таблице.

В сущности «Request» имеется поле `updated_at` — время последнего изменения заявки. Для гарантии корректности данного поля и исключения ошибок при `update` - запросах разумно создать триггер, который будет проставлять данному полю текущее время после совершения операции `update` над записью.

## **2.3 Проектирование приложения**

- 1) слой бизнес логики — обработка основной логики работы приложения;
- 2) слой доступа к данным — подключение к базе данных, отправка запросов, получение информации из базы данных;
- 3) слой программного интерфейса — обработка запросов от пользователей, делегирование выполнения соответствующим сервисам бизнес-логики.

## **2.4 Вывод**

В результате проектирования базы данных для разрабатываемого приложения спроектированы сущности базы данных и их связи, ролевая модель и триггер.



### 3 Технологический раздел

В данном разделе выбраны и обоснованы средства реализации, работана база данных, а также будет разработан интерфейс доступа к данным. Выполнено покрытие тестами публичного API приложение, протестирован триггер.

#### 3.1 Выбор объектной базы данных

При выборе объектной базы данных стоит помнить, что использоваться она будет для хранения приложений (бинарных файлов) объемом до 10МБ. Ниже представлен обзор существующих решений и обоснование выбора базы данных.

##### 3.1.1 Microsoft Azure Blob Storage

Microsoft Azure Blob Storage - это облачное хранилище данных, предоставляемое компанией Microsoft в рамках облачного сервиса Azure. Azure Blob Storage предназначено для хранения больших объемов неструктурированных данных, таких как файлы, изображения, видео, аудио, резервные копии, журналы и другие типы информации. Это расширяемое и высокодоступное хранилище, которое обеспечивает надежное хранение данных в облаке с возможностью масштабирования по требованию.

Основной недостаток – недоступна для использования в России.

##### 3.1.2 Yandex S3

Amazon Simple Storage Service (Amazon S3) — это облачное хранилище данных, предоставляемое Amazon Web Services (AWS). S3 предоставляет возможность хранить и извлекать любое количество данных в Интернете, обеспечивая высокую доступность, надежность и масштабируемость.

Основные характеристики Amazon S3:

- 1) доступность и надежность: S3 обеспечивает высокую доступность данных и надежность, гарантируя, что ваши данные будут доступны в любое время;
- 2) масштабируемость: S3 позволяет хранить огромные объемы данных без

необходимости заботиться о масштабировании инфраструктуры;

- 3) управление данными: Вы можете управлять доступом к вашим данным, устанавливать права доступа, шифровать данные и использовать другие функции для обеспечения безопасности информации;
- 4) стоимость: Вы платите только за использование, что делает S3 экономически выгодным решением для хранения данных.

Amazon S3 широко используется для хранения резервных копий, статических сайтов, медиафайлов, архивов данных, а также для обработки и анализа больших объемов информации.

Yandex S3 предоставляет совместимый с «оригинальным» Amazon S3 программный интерфейс и доступен в России. В современной мирк s3 является де-факто стандартом хранения бинарных файлов.

### **3.1.3 Резальтат выбора объектной базы данных**

Для реализации проекта среди 2 рассмотренных вариантов была выбрана база данных Yandex S3, так как она доступна в России и удовлетворяет всем требованиям.

## **3.2 Выбор реляционной базы данных**

Ниже представлен обзор существующих решений обоснование выбора базы данных.

### **3.2.1 Oracle Database**

Oracle Database - это одна из самых популярных и мощных реляционных баз данных, разработанная компанией Oracle Corporation. Она широко используется в корпоративных средах для хранения и управления данными, обеспечивая высокую производительность, надежность и масштабируемость. Основным недостатком, который делает невозможным применения данной базы данных является отсутствие поддержки этой базы данных выбранным для разработки фреймворком userver.

### 3.2.2 MySQL

MySQL - это одна из самых популярных открытых реляционных баз данных, широко используемая веб-разработчиками и предприятиями для хранения и управления данными. Тем не менее, в рамках данной работы получилось выделить следующие минусы данной системы управления базами данных:

- 1) ограниченная поддержка даты и времени: MySQL имеет ограниченный набор типов данных для работы с датой и временем, что может вызывать сложности при работе с различными форматами даты и времени или при работе с часовыми поясами;
- 2) ограниченные возможности хранения текстовых данных: MySQL имеет ограничения на размер текстовых полей, что может быть проблемой при работе с большими объемами текстовых данных, таких как скрипты выступлений, отправленные через комментарии.

Хотя MySQL является мощной и гибкой базой данных, ограниченный набор типов данных может стать препятствием к применению в данном проекте.

### 3.2.3 PostgreSQL

PostgreSQL - это мощная и расширяемая объектно-реляционная система управления базами данных (СУБД), которая широко используется в различных проектах и приложениях. Она является открытой и бесплатной для использования, что делает ее популярным выбором среди разработчиков и организаций.

PostgreSQL предлагает широкий набор встроенных и сторонних расширений, которые позволяют расширять функциональность базы данных с помощью пользовательских функций, типов данных, операторов и других возможностей. PostgreSQL обеспечивает высокий уровень транзакционной безопасности благодаря поддержке ACID-свойств (атомарность, согласованность, изолированность, долговечность) и механизмам контроля целостности данных.

Из минусов PostgreSQL обычно выделяют отсутствие встроенной возможности шардирования, а также сложность настройки кластера базы данных. В рамках данной работы шардирование не планируется к использованию, а

PostgreSQL будет развернута на единичной инсталляции, а не в кластере.

### **3.2.4 Результат выбора реляционной базы данных**

В качестве основной базы данных принято использовать PostgreSQL, так как она удовлетворяет заданным требованиям и, кроме того, выбранный для разработки фреймворк `userver` имеет лучшую поддержку именно этой базы данных.

## **3.3 Сущности объектной базы данных**

Единственная сущность объектной базы данных — бинарный файл. Доступ к нему осуществляется по имени файла. Имя файла — его идентификатор (полу в таблице `file_meta`).

Проектирование ролевой модели для объектной базы данных разрабатываемого приложения нецелесообразно.

## **3.4 Сущности основной базы данных**

Ниже представлены разработанные сущности базы данных.

### **3.4.1 Таблицы**

На рисунке 2 представлены сущности-таблицы полученной базы данных. Для обеспечения целостности данных используются внешние ключи, ограничения на уникальность первичного ключа.

### **3.4.2 Индекс**

В таблице `requests` используется стандартный b-tree индекс на поле `user_id`. Использование необходимо обеспечения быстрого действия запроса из листинга 1 и для проведения исследования.

### **3.4.3 Триггер**

Ниже на рисунке представлена диаграмма спроектированного триггера.

Триггер гарантирует корректность метки времени `updated_at` таблицы `requests`. Для тестирования триггера написаны функциональные тесты на языке Python с использованием библиотеки `pytest`. В таблице 2 приведены сценарии тестирования.

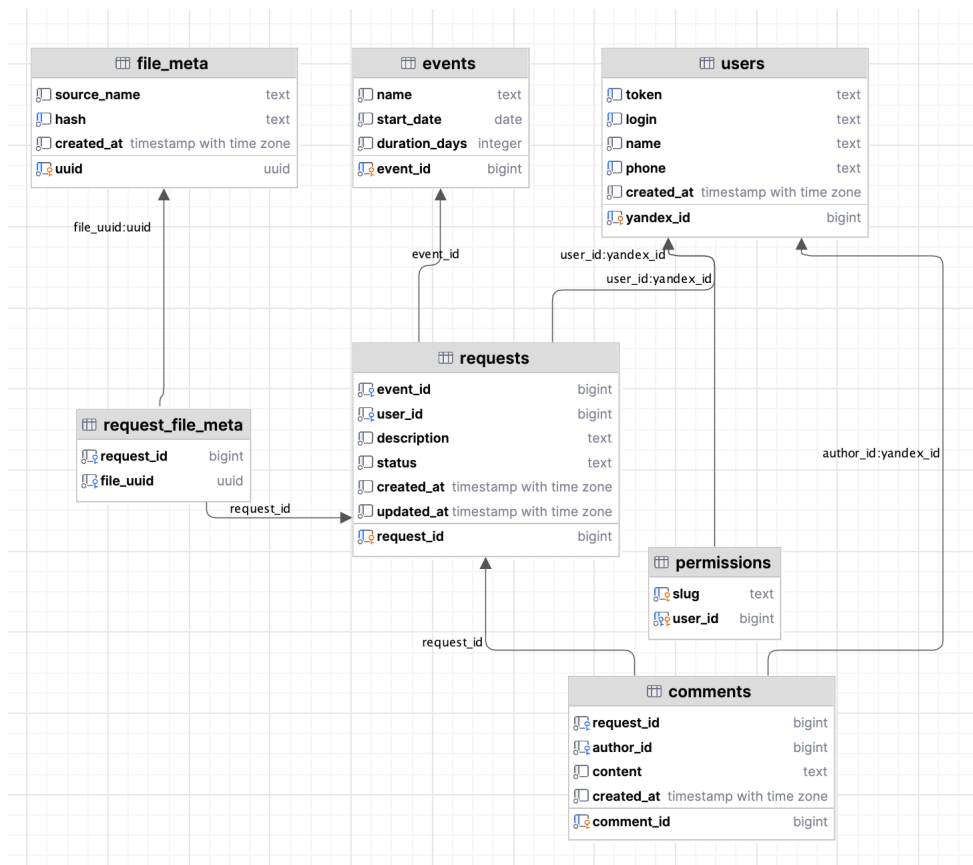


Рисунок 2 – ER-диаграмма базы данных

Таблица 2 – Сценарии тестирования триггера

Событие	Ожидаемый результат	Рельный результат
Изменилось поле description	updated_at = now()	updated_at = now()
Изменилось поле status	updated_at = now()	updated_at = now()
Попытка изменить updated_at	updated_at = now()	updated_at =now()

#### 3.4.4 Роли

Помимо ролевой модели на уровне приложения разработана для дополнительных гарантий безопасности дополнительно разработа ролевая модель на уровне приложения. Права пользователей представлены в таблице 3.

Такая ролевая модель на уровне базы данных обеспечит пользователей минимальными необходимыми правами для доступа к просмотру и модификации данных.

Таблица 3 – Ролевая модель на уровне базы данных

Действие	Таблица	everybody	modarator	admin
select	все таблицы	да	да	да
insert, update	requests	да	да	да
insert, update, delete	events	нет	да	да
insert, update	users	да	да	да
insert, update,	requests_file_meta	да	да	да
insert, update	file_meta	да	да	да
insert, update, delete	permisisions	нет	нет	да
insert, update	comments	да	да	да

### 3.5 Интерфейс доступа к данным

Доступ к данным осуществляется методом отправки http запроса на сервер, на котором развернуто приложение. Ниже приведено описание эндпоинтов.

- 1) /v1/request GET — получение информации о запросе по его идентификатору.
- 2) /v1/requests GET — получение информации о всех запросах, доступных для просмотра.
- 3) /v1/requests POST — создание/обновление запроса.
- 4) /v1/request/status PATCH — обновление статуса заявки.
- 5) /v1/request/comment POST — добавление комментария к заявке.
- 6) /v1/file GET — получение файла по его идентификтору.
- 7) /v1/file POST — добавление файла (приложения).
- 8) /v1/events GET — просмотр списка мероприятий.
- 9) /v1/event POST — создание мероприятия.
- 10) /v1/event DELETE — удаление мероприятия.

### **3.6 Вывод**

Разработаны и протестированы сущности базы данных, разработан интерфейс взаимодействия с данными.

## 4 Исследовательский раздел

### 4.1 Цель исследования

Целью исследования является оценка зависимости времени выполнения запроса от наличия индексов в базе данных. Для оценки использован запрос из листинга 1. Необходимо определить, влияет ли использование стандартного b-tree индекса на user\_id на время выполнения данного запроса при разном количестве записей.

Листинг 1 – Запрос для исследования

```
1 select event_id, description, status, created_at, updated_at
2 from service.requests
3 where user_id={user_id};
```

### 4.2 Описание исследования

На первом этапе исследования производится замер времени выполнения запросов при количестве записей от 100 до 1000 с шагом 100 в таблице без использования индекса.

На втором этапе исследования производится замер времени выполнения запросов при том же количестве записей, что и на предыдущем этапе, однако с использованием индексов.

Для уменьшения погрешности вычислений каждый запрос отправляется 1000 раз, а за результат берется медианное время 1000 запросов.

Для уменьшения временных издержек, связанных с работой сети база данных и скрипт отправки запросов запускаются на одной физической машине.

Результаты измерений приведены в таблице 4. График изображен на рисунке 3.



Таблица 4 – Результат исследования

Количество записей	Без индексов, мс	С индексами, мс
100	9.03	6.91
200	12.50	8.73
300	13.41	10.3
400	14.12	11.78
500	15.46	12.13
600	16.12	12.72
700	16.42	12.34
800	16.91	13.4
900	17.07	14.23
1000	17.39	14.10

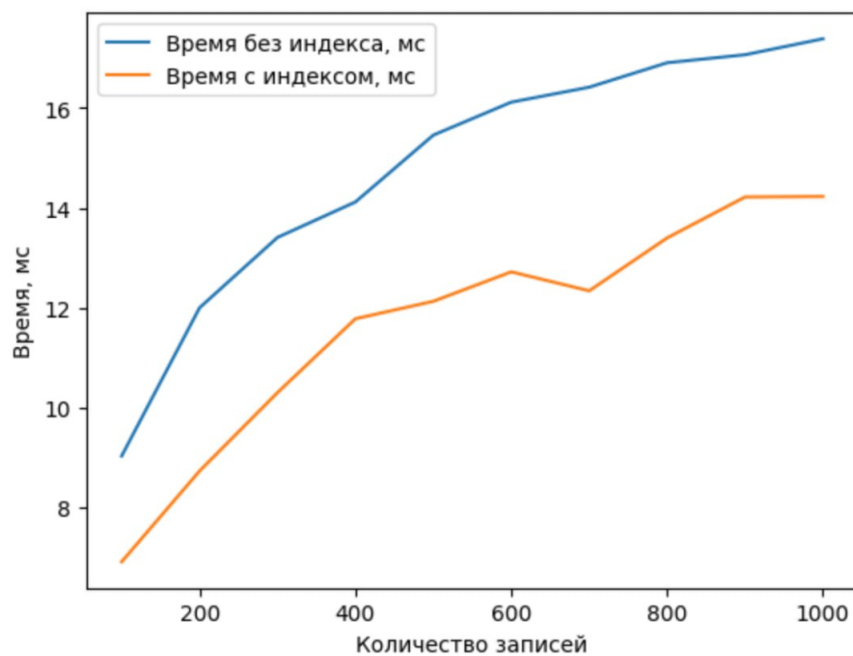


Рисунок 3 – Графие измерений

### **4.3 Вывод**

В результате исследования установлено, что для запроса из листинга 1 время выполнения меньше при использовании индексов.