МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Факультет безопасности информационных технологий**

**Направление подготовки: 10.03.01.**

**Образовательная программа: Безопасность информационных технологий.**

**Дисциплина:**

**«**Информационная безопасность баз данных**»**

**КУРСОВАЯ РАБОТА***.*

**на тему *«***Информационная система подготовительного учебного центра для поступления в высшее учебное заведение***»***

**Выполнил студент(ы):**

Ахраров Али Рустмович, N3350, ИББД1.1



*ФИО                               Подпись*

**Проверил:**

Салихов Максим Русланович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_

*ФИО                               Подпись*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Отметка о выполнении (один из вариантов:*

*отлично, хорошо, удовлетворительно)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Дата*

Санкт-Петербург

2024г.

# ВВЕДЕНИЕ

В современных условиях информационная безопасность играет важную роль во всех аспектах работы с данными. Особенно это касается баз данных, в которых хранится конфиденциальная информация. Курс информационной безопасности баз данных охватывает различные аспекты защиты данных от несанкционированного доступа, манипуляций и потерь. Ключевым элементом курса является практическое выполнение лабораторных работ, направленных на получение навыков защиты данных на уровне баз данных и серверных приложений.

Данная курсовая работа основана на лабораторных работах, выполненных в рамках курса. Она включает в себя создание и реализацию безопасной системы управления базой данных (СУБД), способной обеспечивать конфиденциальность, целостность и доступность данных, а также защиту от внешних и внутренних угроз. В рамках работы была разработана база данных для учебного центра, создано API для взаимодействия с данными, внедрены механизмы логирования, резервного копирования и восстановления данных.

**1.** ПОСТАНОВКА ЗАДАЧИ

Целью курсовой работы является разработка системы, способной обеспечить безопасное хранение и обработку данных учебного центра. Для достижения этой цели были поставлены следующие задачи:

1. Разработать базу данных, соответствующую требованиям учебного центра, с учетом аспектов безопасности и защиты данных.
2. Создать API для взаимодействия с базой данных, реализующий функциональность CRUD (создание, чтение, обновление и удаление) с учетом разграничения прав доступа.
3. Внедрить систему логирования, позволяющую отслеживать действия пользователей в системе для обеспечения аудита и контроля безопасности.
4. Настроить механизмы резервного копирования и восстановления данных, включая тестирование отката к контрольным точкам в случае возникновения непредвиденных изменений.
5. Провести анализ реализованных методов защиты данных, выявить их достоинства и недостатки.

**2.** АНАЛИЗ СУЩЕСТВУЮЩИХ РЕШЕНИЙ

На сегодняшний день существует множество решений для обеспечения безопасности баз данных. Среди них можно выделить следующие подходы:

* **Роль и политика доступа:** Внедрение четкой политики доступа, которая ограничивает права пользователей в зависимости от их роли в системе.
* **Логирование действий пользователей:** Система регистрации и отслеживания действий пользователей позволяет выявлять подозрительные активности и предотвращать злоупотребления.
* **Резервное копирование и восстановление данных:** Системы резервного копирования позволяют быстро восстановить данные в случае их повреждения или утраты.
* **Обезличивание данных:** Используется для защиты конфиденциальности пользователей, путем удаления личных данных, если это необходимо.
* **Методы шифрования данных:** Применяются для защиты данных как при хранении, так и при передаче по сети.

Каждое из этих решений имеет свои достоинства и недостатки, поэтому для обеспечения комплексной безопасности необходимо использовать несколько методов одновременно.

**3.** РАЗРАБОТКА БАЗЫ ДАННЫХ

Для реализации курсовой работы была разработана база данных, включающая таблицы для хранения информации о студентах, преподавателях, курсах, посещаемости, оплате и результатах тестов. Структура базы данных представлена следующими основными сущностями:

1. **Teachers** – содержит информацию о преподавателях, включая их контактные данные и специализацию.
2. **Courses** – включает данные о курсах, их описании, продолжительности и стоимости.
3. **Students** – хранит информацию о студентах, включая их личные данные и специальности.
4. **Attendance** – фиксирует посещаемость студентов на курсах.
5. **Payment** – отражает информацию об оплатах студентов за курсы.
6. **Schedule** – отображает расписание занятий и привязку к аудиториям.
7. **Assignment** и **Assignment\_Result** – содержат данные о заданиях и результатах выполнения студентами.
8. **Main\_Log** – таблица для логирования действий пользователей, что позволяет отслеживать изменения в системе.

База данных была спроектирована с учетом необходимости обеспечения целостности и согласованности данных. Использование внешних ключей и ограничений позволило достичь требуемого уровня структурной целостности.

# 3. ПОДРОБНАЯ АНАЛИЗ И РЕАЛИЗАЦИЯ АСПЕКТОВ БЕЗОПАСНОСТИ В КУРСОВОЙ РАБОТЕ

Реализация сервиса для взаимодействия с базой данных требует всестороннего анализа и внедрения множества технических решений. Ключевым этапом стало обеспечение безопасности данных как на уровне сервиса, так и на уровне базы данных (СУБД). В данном разделе мы детально рассмотрим архитектуру системы, функциональность сервисов, внедренные меры безопасности и проведем анализ их эффективности.

# 4. АРХИТЕКТУРА СИСТЕМЫ

#### Структура сервиса

Разработанный сервис базируется на трех основных компонентах:

1. **API на основе Flask**:
   * Реализовано RESTful API, поддерживающее стандартные CRUD-операции.
   * Каждая операция защищена системой аутентификации и авторизации.
   * Обеспечена возможность взаимодействия с основными таблицами базы данных через HTTP-запросы.
2. **База данных PostgreSQL**:
   * Содержит 10 таблиц, структурированных в соответствии с нормализованной моделью данных.
   * Реализованы строгие ограничения целостности (внешние ключи, ограничения NOT NULL, UNIQUE).
   * Логика работы таблиц поддерживает жесткую иерархию зависимостей для обеспечения целостности данных.
3. **Система логирования**:
   * Фиксирует все операции взаимодействия с базой данных.
   * Логи содержат подробную информацию: тип операции, пользователя, дату/время, измененные данные.
   * Логи хранятся в таблице main\_log и доступны только администраторам.

## Пример структуры базы данных

База данных содержит следующие таблицы:

* **teachers:** хранит данные о преподавателях (ФИО, специализация, контактные данные).
* **courses:** описывает курсы (название, описание, продолжительность, стоимость).
* **students:** содержит информацию о студентах (ФИО, специальность, дата регистрации).
* **attendance:** фиксирует посещаемость занятий.
* **main\_log:** журнал операций.

Каждая таблица связана с другими через внешние ключи. Например, courses.teacher\_id связан с teachers.teacher\_id, а attendance.course\_id связан с courses.course\_id. Это обеспечивает строгую целостность данных и исключает дублирование.

# ФУНКЦИОНАЛЬНОСТЬ API

#### Реализованные операции API

1. **Чтение данных (GET):**
   * Возвращает записи из таблиц базы данных.
   * Поддерживает фильтрацию и сортировку.
   * Пример: запрос GET /teachers возвращает список всех преподавателей.
2. **Создание записей (POST):**
   * Добавляет новые записи в таблицы.
   * Перед добавлением проверяет уникальность ключевых полей.
   * Пример: запрос POST /courses создает новый курс.
3. **Обновление записей (PUT/PATCH):**
   * Изменяет существующие записи.
   * Проверяет права пользователя перед внесением изменений.
   * Пример: запрос PATCH /students/{id} обновляет данные студента.
4. **Удаление записей (DELETE):**
   * Удаляет записи из таблиц.
   * Включает механизм каскадного удаления, чтобы гарантировать, что связанные данные также удаляются.
   * Пример: запрос DELETE /teachers/{id} удаляет преподавателя.

#### Пример запроса к API

Для получения данных о всех курсах отправляется запрос:

GET /courses HTTP/1.1

Host: localhost:5000

Authorization: Bearer <токен>

Ответ:

[

{

"course\_id": 1,

"course\_name": "Основы программирования",

"description": "Курс для начинающих",

"duration": "3 месяца",

"cost": 1000,

"teacher\_id": 2

},

...

]

# МЕРЫ БЕЗОПАСНОСТИ

#### Уровень сервиса

1. **Разграничение прав доступа:**
   * Реализованы три роли пользователей: студент, преподаватель, администратор.
   * Каждой роли назначены определенные права доступа:
     + Студенты могут просматривать свои данные и данные курса.
     + Преподаватели могут управлять курсами, которые они ведут.
     + Администраторы имеют полный доступ ко всем таблицам.
2. **Аутентификация:**
   * Для доступа к API пользователь должен предоставить токен аутентификации.
   * Реализована базовая токенизация. В будущем возможно внедрение OAuth2.
3. **Защита от SQL-инъекций:**
   * Использование SQLAlchemy предотвращает прямой ввод SQL-кода.
   * Все параметры запросов валидируются перед передачей в базу данных.
4. **Логирование:**
   * Каждый запрос к API записывается в логи.
   * Логи содержат тип операции, пользователя, дату/время, переданные параметры.
   * Логирование помогает отслеживать аномалии, такие как частые запросы на удаление данных.

#### Уровень базы данных

1. **Ограничения целостности:**
   * Внешние ключи и каскадное удаление.
   * Ограничения уникальности для ключевых полей, таких как email.
2. **Роли и права:**
   * Студенты могут только читать свои данные.
   * Преподаватели могут читать и изменять данные курсов.
   * Администраторы могут выполнять любые операции.
3. **Резервное копирование:**
   * Настроено автоматическое создание бэкапов базы данных.
   * Бэкапы создаются каждую неделю и хранятся в защищенной директории.

# ПРОЦЕСС ВОССТАНОВЛЕНИЯ ДАННЫХ

#### Резервное копирование

Резервное копирование настроено через pgAdmin:

* Формат бэкапа: .backup.
* Расписание: еженедельно.
* Директория хранения: /backups.

#### Откат изменений

Откат изменений выполняется с использованием последнего бэкапа:

1. Создание резервной копии текущего состояния базы данных.
2. Загрузка последнего бэкапа.
3. Восстановление базы данных с помощью команды:

pg\_restore -U postgres -d training\_center /path/to/backup.backup

#### Анализ изменений

Для анализа откатанных изменений используются логи в таблице main\_log. Логи содержат подробную информацию о каждой операции:

* Тип операции (INSERT, UPDATE, DELETE).
* Дата и время.
* Пользователь.
* Измененные данные.

Пример записи в логе:

INSERT INTO main\_log (operation\_type, operation\_date, user\_operator, changed\_data)

VALUES ('DELETE', '2024-11-14 14:35:00', 'admin', 'Удален курс ID 3');

# АНАЛИЗ БЕЗОПАСНОСТИ

#### Достижения

1. **Высокий уровень защиты данных:**
   * Реализовано разграничение доступа.
   * Логирование операций.
   * Настроено резервное копирование.
2. **Гибкость системы:**
   * Простота добавления новых ролей и прав.
   * Поддержка расширения API.
3. **Интеграция с современными инструментами:**
   * Использование SQLAlchemy для защиты от SQL-инъекций.
   * Автоматическое создание и проверка бэкапов.

# РИМЕНЕНИЕ И АНАЛИЗ СЛОЖНЫХ МЕТОДОВ РЕАЛИЗАЦИИ БЕЗОПАСНОСТИ

В данном разделе мы углубленно рассмотрим реализованные методы информационной безопасности в сервисе, а также дадим рекомендации по их усовершенствованию. Основное внимание будет уделено защите данных на уровне сервиса и СУБД, а также оценке применяемых подходов с точки зрения современного уровня угроз.

# ДЕТАЛЬНЫЙ АНАЛИЗ РЕАЛИЗОВАННЫХ МЕТОДОВ

#### Логирование операций

Система логирования является одним из наиболее важных компонентов обеспечения безопасности. Логирование позволяет:

* Отслеживать попытки несанкционированного доступа.
* Анализировать действия пользователей в случае обнаружения нарушений.
* Фиксировать изменения данных для дальнейшего анализа.

##### Реализация логирования в сервисе

Система логирования реализована через таблицу main\_log в базе данных. Пример записи в логе:

@app.before\_request

def verify\_token():

token = request.headers.get('Authorization')

if not token or not is\_valid\_token(token):

return jsonify({'message': 'Unauthorized'}), 401

#### Резервное копирование и восстановление

Для обеспечения сохранности данных разработана система резервного копирования. Бэкапы создаются с помощью инструмента pg\_dump и сохраняются в директорию /backups. Восстановление выполняется с использованием pg\_restore.

##### Процесс резервного копирования

1. Настроено автоматическое создание бэкапов через планировщик задач.
2. Бэкапы включают всю схему базы данных и данные всех таблиц.
3. Формат хранения: .backup.

##### Пример команды резервного копирования

pg\_dump -U postgres -F c -b -v -f /backups/training\_center\_2024-11-14.backup training\_center

##### Процесс восстановления

1. Создание резервной копии текущей базы.
2. Остановка сервиса.
3. Восстановление из бэкапа:

pg\_restore -U postgres -d training\_center /backups/training\_center\_2024-11-14.backup

# ФУНКЦИОНАЛЬНОСТЬ РАЗРАБОТАННОГО СЕРВИСА

Разработанный сервис взаимодействия с базой данных training\_center включает в себя следующие основные функциональные модули:

1. **API для управления данными**  
   API предоставляет интерфейс для выполнения CRUD-операций (создание, чтение, обновление, удаление) над данными в базе. Реализованы маршруты для работы с основными сущностями, такими как teachers, courses, students, и другие.

Пример функционального маршрута:

@app.route('/teachers', methods=['GET'])

def get\_teachers():

db = get\_db()

teachers = db.query(Teacher).all()

return jsonify([{

'teacher\_id': teacher.teacher\_id,

'firstname': teacher.teacher\_firstname,

'lastname': teacher.teacher\_lastname,

'email': teacher.teacher\_email,

'specialization': teacher.teacher\_specialization

} for teacher in teachers])

**Преимущества**:

* + Удобный доступ к данным через стандартные HTTP-методы.
  + Универсальность, подходящая как для веб-интерфейсов, так и для мобильных приложений.

Пример настройки маршрута с проверкой прав:

@app.before\_request

def check\_permissions():

user\_role = get\_user\_role\_from\_token(request.headers.get('Authorization'))

if not has\_permission(user\_role, request.endpoint):

return jsonify({'message': 'Forbidden'}), 403

**Логирование действий пользователей**  
Все операции фиксируются в логах для дальнейшего анализа. Логи записываются как в файл сервера, так и в таблицу main\_log базы данных.

Пример записи в лог:

logger.info(f"Пользователь {user} запросил список преподавателей.")

1. **Резервное копирование данных**  
   Настроена автоматическая система резервного копирования, которая сохраняет всю структуру базы данных и данные в зашифрованный файл. Система резервирования активируется на основе расписания, что минимизирует риск потери данных.
2. **Обработка ошибок и уведомления**  
   Реализована система обработки исключений, которая перехватывает ошибки, информирует пользователя о проблеме и записывает информацию об ошибке в лог. Это повышает устойчивость системы к сбоям и улучшает пользовательский опыт.

# ПРЕИМЕНЕНИЕ РЕАЛИЗОВАННОГО СЕРВИСА

#### Взаимодействие с базой данных

Сервис обеспечивает централизованный доступ к базе данных training\_center, предлагая унифицированный API для управления данными. Основные действия включают:

* **Получение данных**: запросы данных по различным параметрам, включая сложные фильтры.
* **Управление курсами**: добавление, удаление и обновление записей о курсах.

#### Пример работы API

1. **Запрос списка курсов**:

curl -X GET <http://localhost:5000/courses>

Ответ:

[

{

"course\_id": 1,

"course\_name": "Основы программирования",

"description": "Курс для начинающих",

"duration": "3 месяца",

"cost": 15000

},

{

"course\_id": 2,

"course\_name": "Веб-разработка",

"description": "Создание сайтов с нуля",

"duration": "4 месяца",

"cost": 20000

}

]

**Добавление нового преподавателя**:

curl -X POST http://localhost:5000/teachers -H "Content-Type: application/json" -d '{

"firstname": "Александр",

"lastname": "Иванов",

"phone": "+79031234567",

"email": "alex.ivanov@example.com",

"specialization": "Математика"

}'

Ответ:

{

"message": "Teacher added successfully"

}

# ИТОГОВЫЕ ВЫВОДЫ

Разработанный сервис демонстрирует высокий уровень интеграции с базой данных training\_center, предоставляя полный набор инструментов для работы с данными. Сочетание удобного API, системы резервирования и логирования делает сервис мощным и надежным инструментом для работы в реальных условиях.

# Заключение

В рамках выполнения курсовой работы была разработана информационная система для управления учебным процессом в тренировочном центре, включающая базу данных и сервис для взаимодействия с ней. Работа над проектом охватывала этапы инфологического моделирования, проектирования структуры БД, реализации системы в выбранной СУБД, а также разработку безопасного сервиса для работы с данными.

Реализованный сервис предоставил возможность:

* управления данными о преподавателях, курсах, группах и расписаниях через веб-интерфейс;
* создания резервных копий базы данных и откатов к контрольным точкам;
* обеспечения разграниченного доступа для разных ролей пользователей.

Особое внимание уделено вопросам безопасности. В ходе работы были применены механизмы аутентификации, защиты данных при передаче и хранении, а также внедрена система логирования для мониторинга операций в БД.

Данная работа продемонстрировала возможности оптимального подхода к проектированию и реализации современных информационных систем с учетом требований информационной безопасности. Полученные результаты показывают перспективы дальнейшего развития системы, включая интеграцию с внешними API, внедрение дополнительных методов защиты данных и повышение производительности за счет оптимизации запросов и структуры БД.