Санкт-Петербургское государственное бюджетное

профессиональное образовательное учреждение

«Ижорский колледж»

КУРСОВАЯ РАБОТА

ТЕМА: «Платформа для симуляции финансовых стратегий – игр с реальными данными»

по ПМ.11 МДК.11.01 Технология разработки и защиты баз данных

Выполнил

обучающийся группы 232с

Фофанов Максим Дмитриевич

Проверил

преподаватель спец. дисциплин

специальности 09.02.07

Информационные системы и программирование

Венедиктов Д.В.

Санкт-Петербург, 2025ЗАДАНИЕ ДЛЯ ВЫПОЛНЕНИЯ КУРСОВОЙ РАБОТЫ

1. Изучить теоретические основы технологии разработки баз данных, включая этапы проектирования, нормализации, реализации и тестирования.

2. Рассмотреть методы и средства защиты баз данных от несанкционированного доступа, атак и вредоносных программ.

3. Провести анализ существующих систем управления базами данных (СУБД) с точки зрения их функциональности, безопасности и удобства использования.

4. Разработать проект базы данных для конкретной предметной области, учитывая требования к безопасности и конфиденциальности данных.

5. Реализовать разработанный проект в выбранной СУБД.

6. Протестировать разработанную базу данных на предмет соответствия требованиям безопасности и надёжности.

7. Оценить эффективность разработанной системы защиты базы данных и предложить рекомендации по её улучшению.

8. Оформить результаты работы в виде курсовой работы, включающей введение, основную часть, заключение, список использованных источников и приложения.

В основной части курсовой работы необходимо подробно рассмотреть следующие вопросы:

* основные этапы разработки базы данных;
* методы обеспечения целостности и безопасности данных;
* механизмы аутентификации и авторизации пользователей;
* шифрование данных и защита от вредоносного ПО;
* особенности реализации проекта базы данных в выбранной СУБД;
* результаты тестирования и оценка эффективности системы защиты.

Для выполнения задания рекомендуется использовать современные научные источники, а также практический опыт разработки и защиты баз данных.

ПЛАН-ГРАФИК ВЫПОЛНЕНИЯ КУРСОВОЙ РАБОТЫ

|  |  |  |
| --- | --- | --- |
| Дата | Содержание работ | Отметка о выполнении |
| 10.09 | Выбор темы курсовой работы |  |
| 17.09 | Знакомство с методическими рекомендациями по выполнению курсовой работы |  |
| 24.09 | Знакомство с планом-графиком выполнения курсовой работы |  |
| 08.10 | Работа с нормативно-правовыми документами, учебной литературой |  |
| 22.10 | Анализ предметной области. Постановка задачи |  |
| 29.10 | Составление введения к курсовой работе |  |
| 05.11 | Описание предметной области и функции решаемых задач |  |
| 12.11 | Выбор средств для выполнения курсовой работы. Выбор среды разработки |  |
| 19.11 | Концептуально-логическое проектирование. Составление ER-диаграммы |  |
| 10.12 | Создание и заполнение базы данных |  |
| 24.12 | Представления в базе данных |  |
| 21.01 | Процедуры в базе данных |  |
| 04.02 | Создание ролей в базе данных |  |
| 18.02 | Аутентификация пользователей в базе данных |  |
| 25.02 | Импорт и экспорт базы данных |  |
| 04.03 | Разработка стратегии резервного копирования базы данных |  |
| 11.03 | Тестирование базы данных |  |
| 18.03 | Оптимизация базы данных |  |
| 25.03 | Составление заключения к курсовой работе |  |
| 01.04 | Составление списка источников информации, используемых при выполнении курсовой работы |  |
| 08.04 | Подготовка курсовой работы к защите |  |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 7](#_Toc180594792)

[ГЛАВА 1. ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ БАЗЫ ДАННЫХ 8](#_Toc180594793)

[1.1. Анализ предметной области 8](#_Toc180594794)

[1.2. Анализ технологий для разработки базы данных 8](#_Toc180594795)

[1.3. Постановка задачи курсовой работы 10](#_Toc180594796)

[1.3.1. Цели курсовой работы 10](#_Toc180594797)

[1.3.2. Задачи курсовой работы 11](#_Toc180594798)

[1.4.1. Функциональные требования к системе 11](#_Toc180594799)

[1.4.2. Нефункциональные требования к системе 11](#_Toc180594800)

[1.5. Выбор программных средств для разработки базы данных 11](#_Toc180594801)

[1.6. Выбор среды для разработки базы данных 14](#_Toc180594802)

[ГЛАВА 2. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ 17](#_Toc180594803)

[2.1. Концептуально-логическое моделирование 17](#_Toc180594804)

[2.1.1 Концептуальное моделирование 17](#_Toc180594805)

[2.1.2. Логическое моделирование 18](#_Toc180594806)

[2.2. Описание информационных объектов базы данных 20](#_Toc180594807)

[2.3.1. Создание базы данных 25](#_Toc180594808)

[2.3.2. Заполнение базы данных 25](#_Toc180594809)

[2.4. Представления в базе данных 27](#_Toc180594810)

[2.5. Процедуры в базе данных 30](#_Toc180594811)

[2.6. Создание ролей в базе данных 30](#_Toc180594812)

[2.7. Аутентификация пользователей в базе данных 30](#_Toc180594813)

[2.8. Импорт и экспорт базы данных 30](#_Toc180594814)

[2.9. Разработка стратегии резервного копирования базы данных 30](#_Toc180594815)

[2.10.1. Тестирование базы данных 31](#_Toc180594816)

[2.10.2. Оптимизация базы данных 31](#_Toc180594817)

[ЗАКЛЮЧЕНИЕ 32](#_Toc180594818)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 33](#_Toc180594819)

# ВВЕДЕНИЕ

В современном цифровом мире данные представляют собой один из ключевых ресурсов, влияющих на стратегические решения и операционные процессы организаций. Базы данных стали основным инструментом для хранения, обработки и управления информацией, обеспечивая эффективное взаимодействие с данными. Разработка баз данных – это сложный процесс, требующий комплексного подхода, включающего анализ требований, проектирование, реализацию и поддержку.

С учетом растущего объема информации и усложнения бизнес-процессов, значимость качественного проектирования баз данных трудно переоценить. Правильная структура базы данных обеспечивает не только ее функциональность и производительность, но и легкость в управлении данными, что в свою очередь влияет на оперативность принятия решений.

Однако с увеличением объема данных и многообразием технологий, связанных с их хранением, возникает необходимость уделять особое внимание защите данных. Утечки информации и кибератаки могут привести к значительным финансовым потерям и подрыву доверия со стороны клиентов. Поэтому технологии защиты баз данных становятся критически важными для бизнеса всех размеров.

Эта работа сосредоточится на ключевых аспектах разработки и защиты баз данных, включая методы проектирования и элементы безопасности, которые должны быть внедрены для обеспечения надежности и защиты данных в любой организации. Изучение этих тем поможет определить лучшие практики, которые должны применяться для достижения устойчивости и эффективности работы с информацией в условиях динамично меняющейся технологической среды.

# ГЛАВА 1. ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ БАЗЫ ДАННЫХ

## 1.1. Анализ предметной области

Технология разработки и защиты баз данных охватывает широкий спектр процессов, методов и инструментов, необходимых для создания, управления и обеспечения безопасности баз данных. Это область, которая активно развивается в ответ на растущие требования к хранению и защите информации, что делает её актуальной для различных отраслей, таких как финансы, здравоохранение, образование и многие другие.

## 1.2. Анализ технологий для разработки базы данных

Разработка баз данных включает в себя использование различных технологий и инструментов, которые помогают осуществлять проектирование, реализацию, управление и оптимизацию баз данных. Рассмотрим основные технологии и подходы, которые широко применяются в этой сфере.

**Системы управления базами данных (СУБД)**

СУБД являются основными инструментами для создания и управления базами данных. Они могут быть разделены на несколько категорий:

* реляционные СУБД (RDBMS): MySQL, PostgreSQL, Oracle, Microsoft SQL Server;
* нереляционные СУБД (NoSQL): MongoDB, Cassandra, Redis;
* гибридные СУБД: Microsoft Azure Cosmos DB.

**Языки программирования**

* SQL (Structured Query Language): основной язык для работы с реляционными базами данных. Позволяет создавать, изменять, запрашивать и управлять данными;
* Python и JavaScript: часто используются для взаимодействия с базами данных в веб-приложениях. Библиотеки, такие как SQLAlchemy или Sequelize, упрощают работу с базами данных;
* Java и C#: используются в крупных корпоративных приложениях, особенно с реляционными базами данных.

**Инструменты для проектирования баз данных**

* ER-моделирование: Инструменты, такие как Lucidchart, Draw.io или ER/Studio, позволяют создавать диаграммы сущностей и взаимосвязей (ER-диаграммы), которые помогают визуализировать структуру базы данных;
* системы управления версиями: Git и другие системы позволяют контролировать изменения в схемах баз данных и процессах миграции.

**Инструменты для обеспечения безопасности баз данных**

* шифрование: использование технологий шифрования данных (например, AES) для защиты информации как «в покое», так и «в движении»;
* аутентификация и авторизация: применение OAuth, OpenID Connect и других протоколов для управления доступом к базе данных;
* мониторинг и аудит: инструменты, такие как Splunk, могут использоваться для отслеживания запросов и анализа событий безопасности.

**Методы оптимизации производительности**

* индексация: создание индексов для ускорения выполнения запросов, особенно в реляционных базах данных.
* кэширование: использование кэшей, таких как Redis или Memcached, для ускорения доступа к часто запрашиваемым данным.
* балансировка нагрузки: распределение запросов между несколькими серверами баз данных для повышения производительности и отказоустойчивости.

Выбор технологий для разработки базы данных зависит от специфики проекта, объема данных, ожидаемых нагрузок и требований к безопасности. Комплексный подход к выбору инструментов и методов способствует созданию эффективных и надежных систем управления данными, которые могут адаптироваться к изменяющимся условиям и требованиям.

## 1.3. Постановка задачи курсовой работы

С ростом объемов данных, а также увеличением угроз безопасности информации, эффективная разработка и защита баз данных становятся крайне важными задачами для организаций. Безопасность данных имеет критическое значение для решения вопросов конфиденциальности, целостности и доступности информации.

Успешное выполнение работы будет способствовать более глубокому пониманию актуальных проблем и новых подходов в области безопасного управления данными.

### 1.3.1. Цели курсовой работы

Основная цель заключается в создании эффективной и безопасной базы данных, способной обеспечить надежный доступ к информации и защиту от потери или несанкционированного доступа.

### 1.3.2. Задачи курсовой работы

* проектирование архитектуры баз данных;
* разработка схемы данных и моделей;
* обеспечение безопасности данных через шифрование, аутентификацию и авторизацию;
* реализация механизмов резервного копирования и восстановления;
* мониторинг и аудит доступа к данным.

## 1.4.1. Функциональные требования к системе

* возможность создания, обновления и удаления данных;
* поддержка сложных запросов и отчетов;
* обеспечение многопользовательского доступа.

## 1.4.2. Нефункциональные требования к системе

* высокая производительность и надежность;
* защита от несанкционированного доступа;
* удобный интерфейс для пользователей.

## 1.5. Выбор программных средств для разработки базы данных

Выбор программных средств для разработки баз данных является ключевым этапом, который влияет на функциональность, производительность и безопасность системы.

При выполнении данной курсовой работы я планирую использовать следующие программные средства:

* система управления базами данных – PostgreSQL;
* языки программирования – Java JavaScript;
* средства проектирования базы данных – dbdiagram.io Flyway;
* инструменты для обеспечения безопасности данных – AWS KMS Splunk;
* средства тестирования и оптимизации приложений – Apache Bench EXPLAIN

Выбор программных средств для разработки базы данных зависит от специфических требований проекта, таких как объем данных, тип доступа, требования к безопасности и производительности. Важно учитывать будущую масштабируемость и поддерживаемость решений, чтобы создать надежную и эффективную систему.

## 1.6. Выбор среды для разработки базы данных

Выбор подходящей среды для разработки базы данных является критически важным этапом проекта, который может оказать значительное влияние на эффективность работы команды и качество конечного продукта.

При выполнении данной курсовой работы я планирую использовать следующие среды и инструменты:

* IDE и редактор кода – pgAdmin;
* система контейнеризации – Kubernetes;
* система управления версиями – git;
* фреймворк и библиотека – Hibernate (Java)

Выбор среды для разработки базы данных требует комплексного подхода с учетом специфики проекта, требований к производительности, безопасности и интеграции. Правильный выбор инструментария и технологии существенно влияет на успех проекта, поэтому следует тщательно проанализировать свои потребности и доступные решения.

# ГЛАВА 2. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

## 2.1. Концептуально-логическое моделирование

Концептуально-логическое моделирование – это важный этап в проектировании базы данных, который позволяет структурировать данные и определить их взаимосвязи. Этот процесс разделяется на два ключевых этапа: концептуальное моделирование и логическое моделирование.

### 2.1.1 Концептуальное моделирование

Целью концептуального моделирования является определение основных сущностей и их связей в предметной области, не вдаваясь в детали реализации.

**Идентификация сущностей**

1. Пользователь
2. Пользователи сессии
3. Карты пользователя
4. Друзья пользователя
5. Чат
6. Игровая сессия
7. Игровое поле

**Определение атрибутов**

1. **Пользователь:**

* User\_id (PK) — уникальный идентификатор.
* username — логин.
* email — электронная почта.
* password — хэш пароля.
* created\_at — дата регистрации.
* Age — возрасть.

1. **Пользователи сессии:**

* player\_session\_id (PK) — уникальный идентификатор.
* user\_id — ссылка на пользователя.
* session\_id — ссылка на игровую сессию.
* money — текущий баланс игрока.
* position — позиция на игровом поле (1-40).

1. **Карты пользователя:**

* card\_id (PK) — уникальный идентификатор.
* user\_id — владелец карты.
* card\_type — тип карты.
* description — описание действия карты.
* action — эффект.

1. **Друзья пользователя:**

* friendship\_id (PK) — уникальный идентификатор.
* user\_id — пользователь.
* friend\_id — друг.
* status — статус связи ("запрос", "принято", "отклонено").

1. **Чат:**

* message\_id (PK) — уникальный идентификатор.
* sender\_id — отправитель.
* receiver\_id — получатель (или session\_id, если чат групповой).
* message — текст сообщения.
* timestamp — время отправки.

1. **Игровая сессия:**

* session\_id (PK) — уникальный идентификатор.
* creator\_id — создатель сессии.
* status — статус ("ожидание", "в процессе", "завершена").
* current\_player\_id — текущий игрок.
* started\_at — дата начала.
* ended\_at — дата завершения.

1. **Игровое поле:**

* space\_id (PK) — уникальный идентификатор.
* session\_id — игровая сессия.
* name — название клетки (например, "Старт", "Тюрьма").
* type — тип ("собственность", "шанс", "налог").
* position — номер клетки (1-40).
* price — стоимость покупки.
* rent — базовая арендная плата.

**Установление связей**

1. **Пользователи → Пользователи сессии**  
   **Тип**: Один ко многим  
   **Описание**: Один пользователь может участвовать в нескольких игровых сессиях.
2. **Игровая сессия → Пользователи сессии**  
   **Тип**: Один ко многим  
   **Описание**: В одной игровой сессии участвуют несколько игроков (пользователей).
3. **Пользователи ↔ Друзья пользователя**  
   **Тип**: Многие ко многим  
   **Описание**: Пользователи могут иметь друзей через промежуточную таблицу (связь между user\_id и friend\_id).
4. **Пользователи → Карты пользователя**  
   **Тип**: Один ко многим  
   **Описание**: У одного пользователя может быть несколько карт (например, "Шанс", "Общественная казна").
5. **Пользователи → Чат (как отправитель)**  
   **Тип**: Один ко многим  
   **Описание**: Пользователь может отправить несколько сообщений в чат.
6. **Пользователи → Чат (как получатель)**  
   **Тип**: Один ко многим  
   **Описание**: Сообщения могут быть адресованы конкретному пользователю или группе (через session\_id).
7. **Игровая сессия → Игровое поле**  
   **Тип**: Один ко многим  
   **Описание**: Каждая игровая сессия имеет уникальное состояние игрового поля.
8. **Игровая сессия → Пользователи (создатель)**  
   **Тип**: Один к одному  
   **Описание**: Сессию создает один пользователь (creator\_id).
9. **Пользователи сессии → Игровое поле**  
   **Тип**: Один ко многим  
   **Описание**: Позиция игрока (position) связана с клеткой игрового поля.

**Создание ER-диаграммы**



Рисунок 1 – ER-диаграмма разрабатываемой базы данных

### 2.1.2. Логическое моделирование

Целью логического моделирования является перевод концептуальной модели в логическую, с уточнением структуры базы данных и обязательств для хранения данных.

**Определение первичных ключей**

Таблица пользователь – поле User\_id PRIMARY KEY

Таблица Пользователи сессии – поле player\_session\_id PRIMARY KEY

Таблица Карты пользователя – поле card\_id PRIMARY KEY

Таблица Друзья пользователя – поле friendship\_id PRIMARY KEY

Таблица Чат – поле message\_id PRIMARY KEY

Таблица Игровая сессия – поле session\_id PRIMARY KEY

Таблица Игровое поле – поле space\_id PRIMARY KEY

**Формализация связей**

Пользователи (Users) → Пользователи сессии (PlayerSessions)

* **Тип**: *1 ко многим*
* **Реализация**:  
  В таблице PlayerSessions добавить поле user\_id (FK → Users.user\_id).

Игровая сессия (GameSessions) → Пользователи сессии (PlayerSessions)

* **Тип**: *1 ко многим*
* **Реализация**:  
  В таблице PlayerSessions добавить поле session\_id (FK → GameSessions.session\_id).

Пользователи (Users) ↔ Друзья пользователя

* **Тип**: *Многие ко многим*
* **Реализация без отдельной таблицы**:  
  Использовать **массив идентификаторов** в таблице Users:

Пользователи (Users) → Карты пользователя (UserCards)

* **Тип**: *1 ко многим*
* **Реализация**:  
  В таблице UserCards добавить поле user\_id (FK → Users.user\_id).

Пользователи (Users) → Чат (ChatMessages)

* **Тип**: *1 ко многим* (для отправителя и получателя).
* **Реализация**:  
  В таблице ChatMessages добавить:
  + sender\_id (FK → Users.user\_id),
  + receiver\_id (FK → Users.user\_id или GameSessions.session\_id для группового чата).

Игровая сессия (GameSessions) → Игровое поле (BoardSpaces)

* **Тип**: *1 ко многим*
* **Реализация**:  
  В таблице BoardSpaces добавить поле session\_id (FK → GameSessions.session\_id).

Игровая сессия (GameSessions) → Пользователи (Users)

* **Тип**: *1 к 1* (создатель сессии).
* **Реализация**:  
  В таблице GameSessions добавить поле creator\_id (FK → Users.user\_id).

Пользователи сессии (PlayerSessions) → Игровое поле (BoardSpaces)

* **Тип**: *1 ко многим* (через позицию).
* **Реализация**:  
  В таблице PlayerSessions хранить position (номер клетки), который логически связан с BoardSpaces.position.

**Определение типов данных**

пользователь:

* user\_id — serial
* username — varchar(50)
* email — varchar(255)
* password\_hash — varchar(60)
* created\_at — timestamp
* age — integer

пользователи сессии:

* player\_session\_id — serial
* user\_id — bigint
* session\_id — bigint
* money — integer
* position — integer

карты пользователя

* card\_id — serial
* user\_id — bigint
* card\_type — varchar(20)
* description — text
* action — varchar(100)

друзья пользователя:

* friendship\_id — serial
* user\_id — bigint
* friend\_id — bigint
* status — varchar(15)

чат:

* message\_id — serial
* sender\_id — bigint
* receiver\_id — bigint
* message — text
* timestamp — timestamp

игровая сессия:

* session\_id — serial
* creator\_id — bigint
* status — varchar(20)
* current\_player\_id — bigint
* started\_at — timestamp
* ended\_at — timestamp

игровое поле:

* space\_id — serial
* session\_id — bigint
* name — varchar(50)
* type — varchar(20)
* position — integer
* price — integer
* rent — integer

**Создание логической схемы базы данных**

пользователь:

user\_id (PK), username, email, password\_hash, created\_at, age

пользователи сессии:

player\_session\_id (PK), user\_id (FK), session\_id (FK), money, position

карты пользователя

card\_id (PK), user\_id (FK), card\_type, description, action

друзья пользователя:

friendship\_id (PK), user\_id (FK), friend\_id (FK), status

чат:

message\_id (PK), sender\_id (FK), receiver\_id (FK), message, timestamp

игровая сессия:

session\_id (PK), creator\_id (FK), status, current\_player\_id (FK), started\_at, ended\_at

игровое поле:

space\_id (PK), session\_id (FK), name, type, position, price, rent

Концептуально-логическое моделирование является важным этапом разработки базы данных, который позволяет углубиться в детализацию структуры данных и обеспечить целостность и согласованность информации. Правильное моделирование позволяет избежать проблем на следующих этапах разработки и эксплуатации базы данных.

## 2.2. Описание информационных объектов базы данных

Информационные объекты базы данных представляют собой основные строительные блоки, на которых базируется структура любой базы данных. Они включают в себя различные сущности и их атрибуты, которые позволяют организовать, хранить и управлять данными.

Таблица 1 – информационные объекты разрабатываемой базы данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Объект | Атрибуты | Связи | Ограничения | Тип данных |
| Пользователь | User\_id, username, email, password, created\_at, age | 1:N с PlayerSessions, М:N с Friends, 1:N с UserCards, 1:N с ChatMessages | Уникальность id пользователя | serial, varchar(), integer, timestamp |
| Пользователи сессии | player\_session\_id, user\_id, session\_id, moneyposition | 1:N с PlayerSessions, 1:N с BoardSpaces, 1:1 с Users | Уникальность id Пользователей сессии | serial, varchar(), timestamp, bigint |
| Карты пользователя | card\_id, user\_id, card\_type, description, action | N:1 с Users, N:1 с GameSessions, 1:N с BoardSpaces | Уникальность id карты | serial, integer, timestamp, bigint, boolean |
| Друзья пользователя | friendship\_id, user\_id, friend\_id, status | N:1 с GameSessions, 1:N с PlayerSessions | Уникальность id друга | serial, varchar(), integer, bigint |
| Чат | message\_id, sender\_id, receiver\_id, message, timestamp | N:1 с Users | Уникальность id чата | serial, varchar(), text, bigint |
| Игровая сессия | session\_id, creator\_id, status, current\_player\_id, started\_at, ended\_at | 1:N с Users (отправитель), 1:N с Users (получатель) | Уникальность id сессии | serial, text, timestamp, bigint |
| Игровое поле | space\_id, session\_id, name, type, position, price, rent | М:N между Users | Уникальность id поля | serial, varchar(), bigint |

Описание информационных объектов базы данных позволяет структурировать и организовать информацию, обеспечивая эффективное хранение и доступ к данным. Правильное определение сущностей, их атрибутов и связей является ключом к успешному проектированию базы данных и ее функциональной полноте.

## 2.3.1. Создание базы данных



Рисунок 2 – созданная база данных

## 2.3.2. Заполнение базы данных

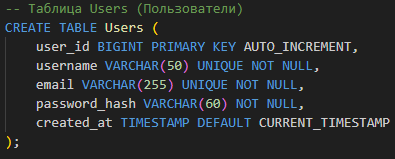


Рисунок 3 – добавление объекта в базу данных



Рисунок 4 – добавление атрибутов объекту в базе данных

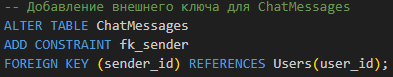


Рисунок 5 – установление связей между объектами в базе данных



Рисунок 6 – выбор типа данных для объекта в базе данных

## 2.4. Представления в базе данных

Представление в базе данных – это виртуальная таблица, которая формируется на основе результата выполнения SQL-запроса. Она не содержит данных, а лишь определяет, как данные будут отображаться пользователям.

Функции представлений:

* упрощение сложных запросов – позволяют объединять данные из нескольких таблиц и показывать только нужные поля, что делает работу с данными более удобной;
* безопасность – можно ограничить доступ пользователей к определённым столбцам или строкам баз данных, предоставляя доступ только к представлениям;
* логическая независимость – изменения в базах данных не влияют на внешний интерфейс, так как представления могут скрывать изменения структуры таблиц;
* кодовая реиспользуемость – позволяют сократить код, избавляя от необходимости повторно писать одни и те же запросы.

Представление создается с помощью команды CREATE VIEW.

|  |
| --- |
| sql  CREATE VIEW UserPublicInfo AS  SELECT user\_id, username, created\_at, age  FROM Users; |

После создания представления его можно использовать так же, как и таблицу:

|  |
| --- |
| sql  SELECT \* FROM UserPublicInfo WHERE username = sigmaboy2016; |

Представления могут быть обновляемыми или не обновляемыми. Обновляемые представления позволяют выполнять операции INSERT, UPDATE и DELETE, при этом изменения будут применяться к базовой таблице.

|  |
| --- |
| sql  UPDATE UserPublicInfo  SET username = 'pikmi'  WHERE id = 2; |

Представления можно изменять с помощью команды CREATE OR REPLACE VIEW или удалять с помощью DROP VIEW

|  |
| --- |
| sql  CREATE OR REPLACE VIEW UserPublicInfo AS  SELECT  username AS "daivinchik",  FROM Users  WHERE username = ‘sigmaboy2016’; |

|  |
| --- |
| sql  DROP VIEW UserPublicInfo; |

Представления в базах данных являются мощным инструментом для работы с данными. Они обеспечивают большую гибкость, безопасность и легкость в использовании, способствуя более эффективной организации и обработке информации. Понимание работы с представлениями – важный аспект для разработчиков и администраторов баз данных.

## 2.5. Процедуры в базе данных

Хранимые процедуры — это заранее написанные SQL-скрипты, которые сохраняются в базе данных и выполняются по запросу. Их использование решает следующие задачи:

1. **Инкапсуляция логики**  
   Процедуры объединяют несколько операций в одну транзакцию, что упрощает взаимодействие с данными. Например:
   * Добавление пользователя с проверкой уникальности.
   * Обновление баланса игрока и запись транзакции.
2. **Безопасность**  
   Процедуры позволяют ограничить прямой доступ к таблицам, снижая риск SQL-инъекций.
3. **Производительность**  
   Предкомпилированные процедуры выполняются быстрее, чем последовательные запросы.

## 2.6. Создание ролей в базе данных

Роли в базе данных — это механизм управления правами доступа, который позволяет группировать пользователей по уровням привилегий. Их использование решает следующие задачи:

1. **Безопасность данных**  
   Роли ограничивают доступ к критически важным операциям (например, удаление таблиц) и защищают данные от несанкционированного изменения. Например:
   * **Администратор** — полный доступ.
   * **Пользователь** — чтение и запись.
   * **Гость** — только чтение.
2. **Удобство администрирования**  
   Вместо назначения прав каждому пользователю отдельно, права задаются роли, которую затем присваивают группе пользователей. Это упрощает масштабирование системы.
3. **Минимизация ошибок**  
   Централизованное управление правами через роли снижает риск случайного предоставления избыточных привилегий.

## 2.7. Аутентификация пользователей в базе данных

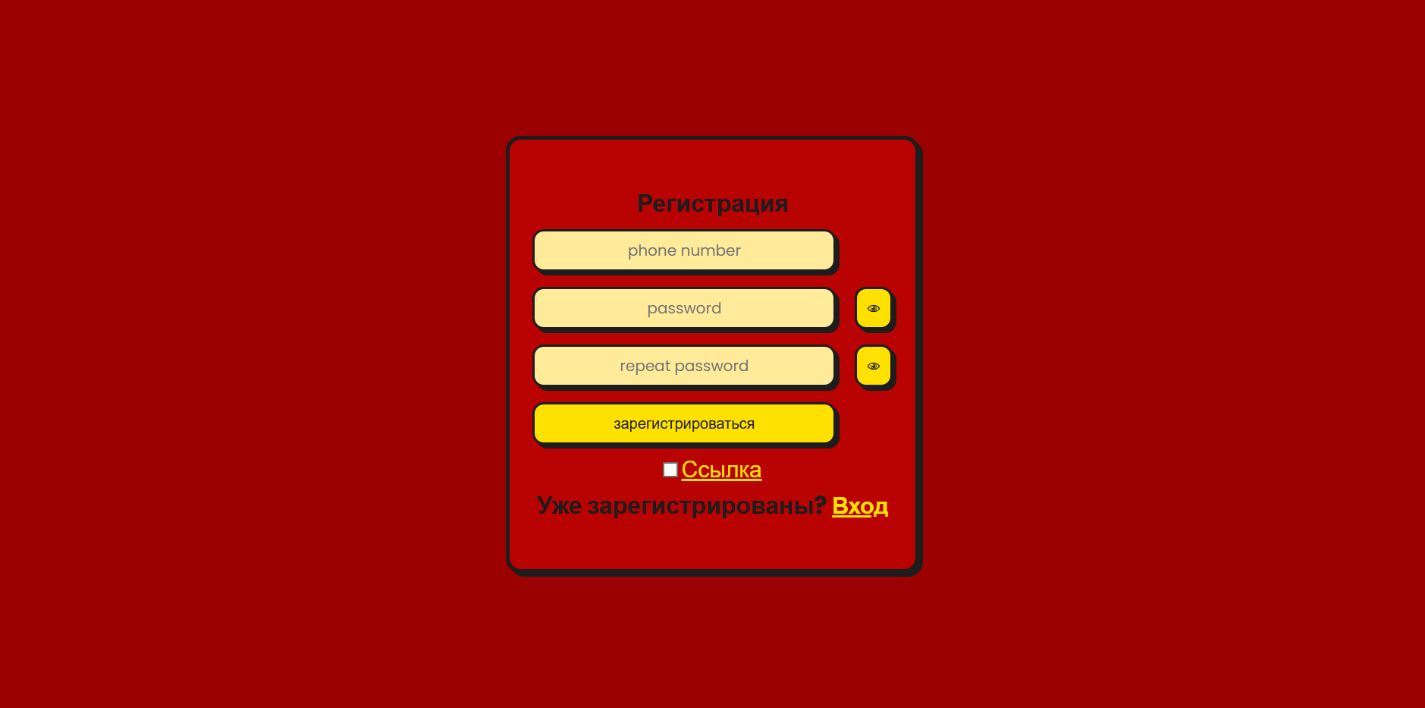
Чтобы зарегистрировать аккаунт на сайте, нужно пройти регистрацию. 

Рисунок 8 – форма регистрации

После регистрации перейти на страницу входа и ввести данные, которые вписывали в полях для регистрации.

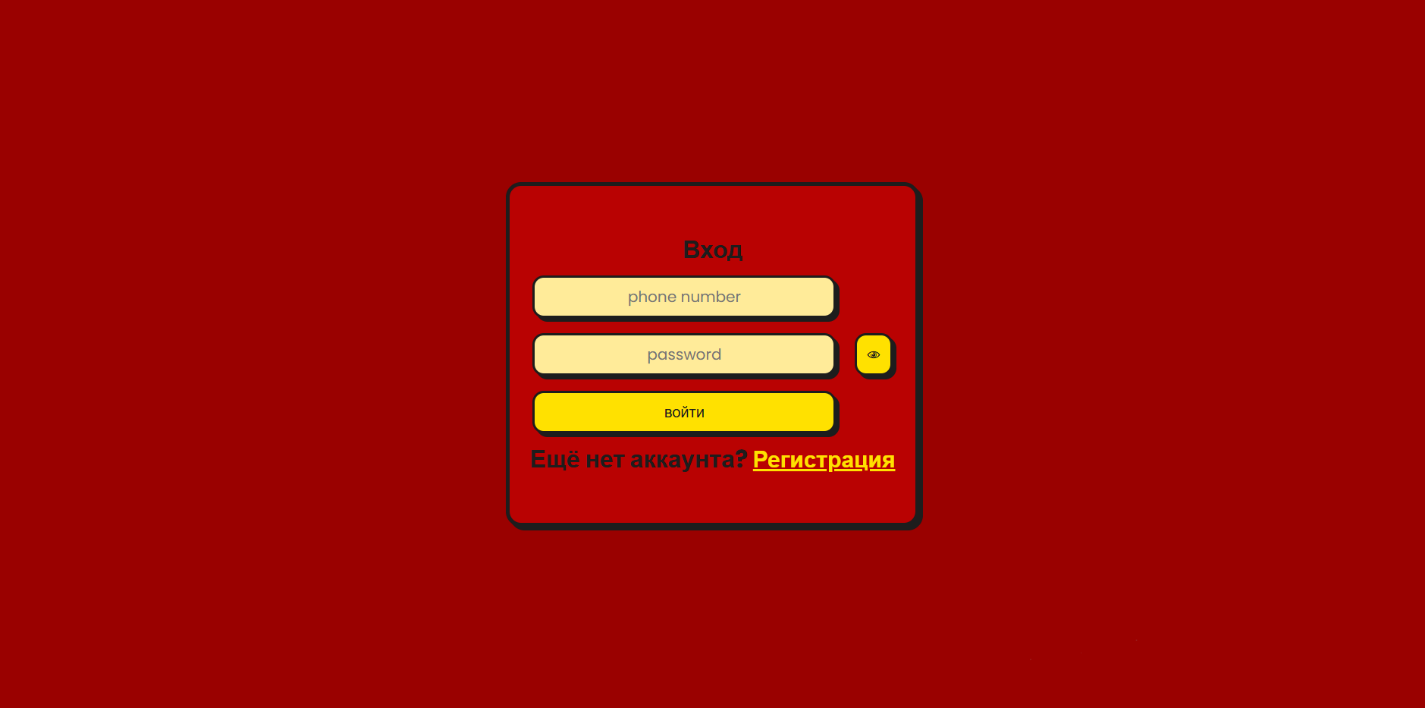


Рисунок 7 – форма входа

Если все поля заполнены верно, то переходит на главную страницу сайта. 

Рисунок 9 – главная страница сайта

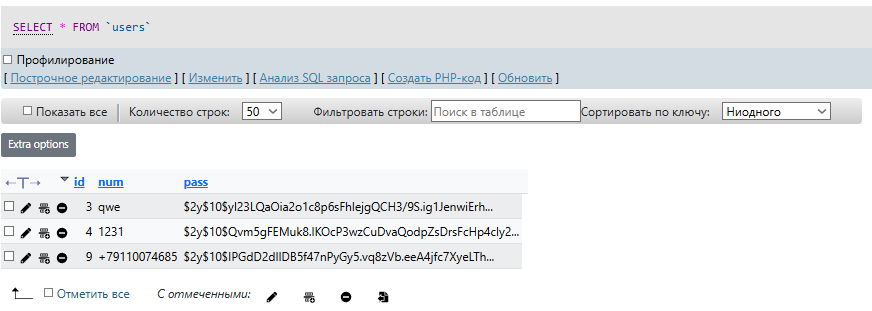
Ваш аккаунт появится во вкладке “users”. Пароля видно не будет, так как используется “**Хеширование”.** 

Рисунок 10 – пример таблицы

## 2.8. Импорт и экспорт базы данных

**Экспорт базы данных**. Чтобы осуществить быстрый экспорт базы данных, нужно выполнить следующие действия:

1. В области слева выбрать необходимую базу данных (должен раскрыться список таблиц базы данных).
2. Перейти на вкладку «Экспорт» в меню phpMyAdmin.
3. Нажать кнопку «Экспорт», и файл дампа базы данных загрузится на локальный компьютер.

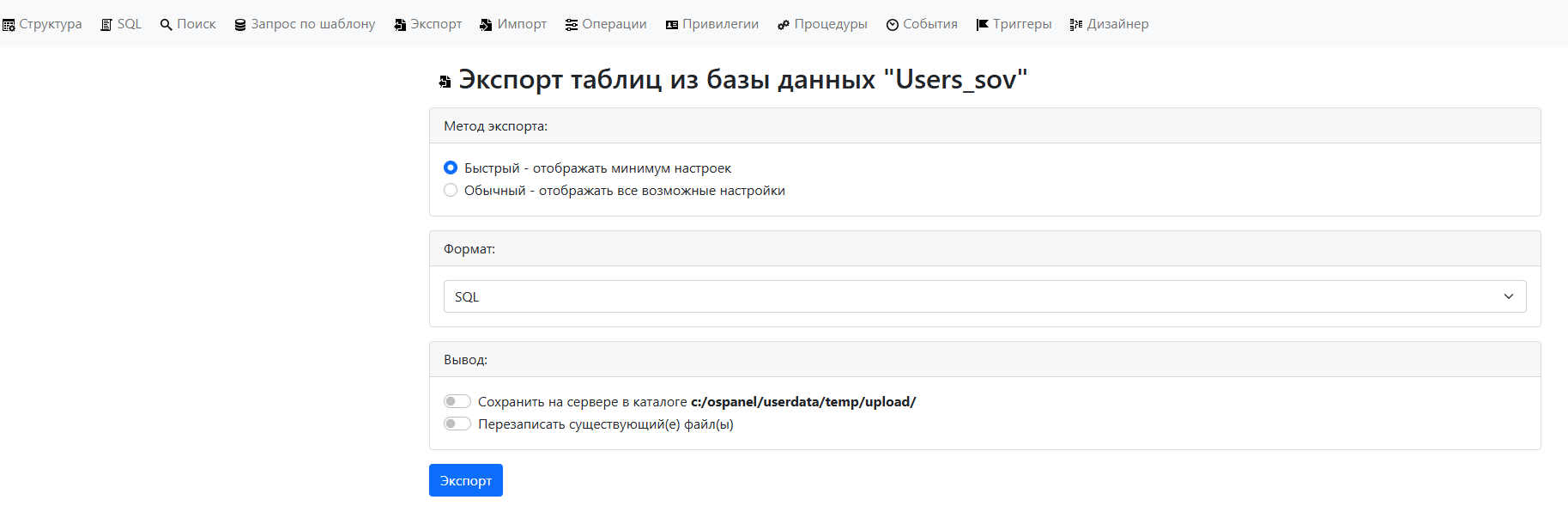


Рисунок 11 – импорт таблицы

**Импорт базы данных**. Чтобы осуществить импорт, нужно выполнить следующие действия:

1. В области слева выбрать необходимую базу данных (должен раскрыться список таблиц базы данных).
2. Перейти на закладку «Импорт» в меню phpMyAdmin.
3. Загрузить импортируемый файл (дамп базы данных), нажав на кнопку «Выбрать файл» и указав файл на локальном компьютере.
4. Выбрать кодировку, формат и проверить другие настройки импорта (можно оставить по умолчанию).
5. Нажать кнопку «Импорт», и дамп будет импортирован в базу данных.

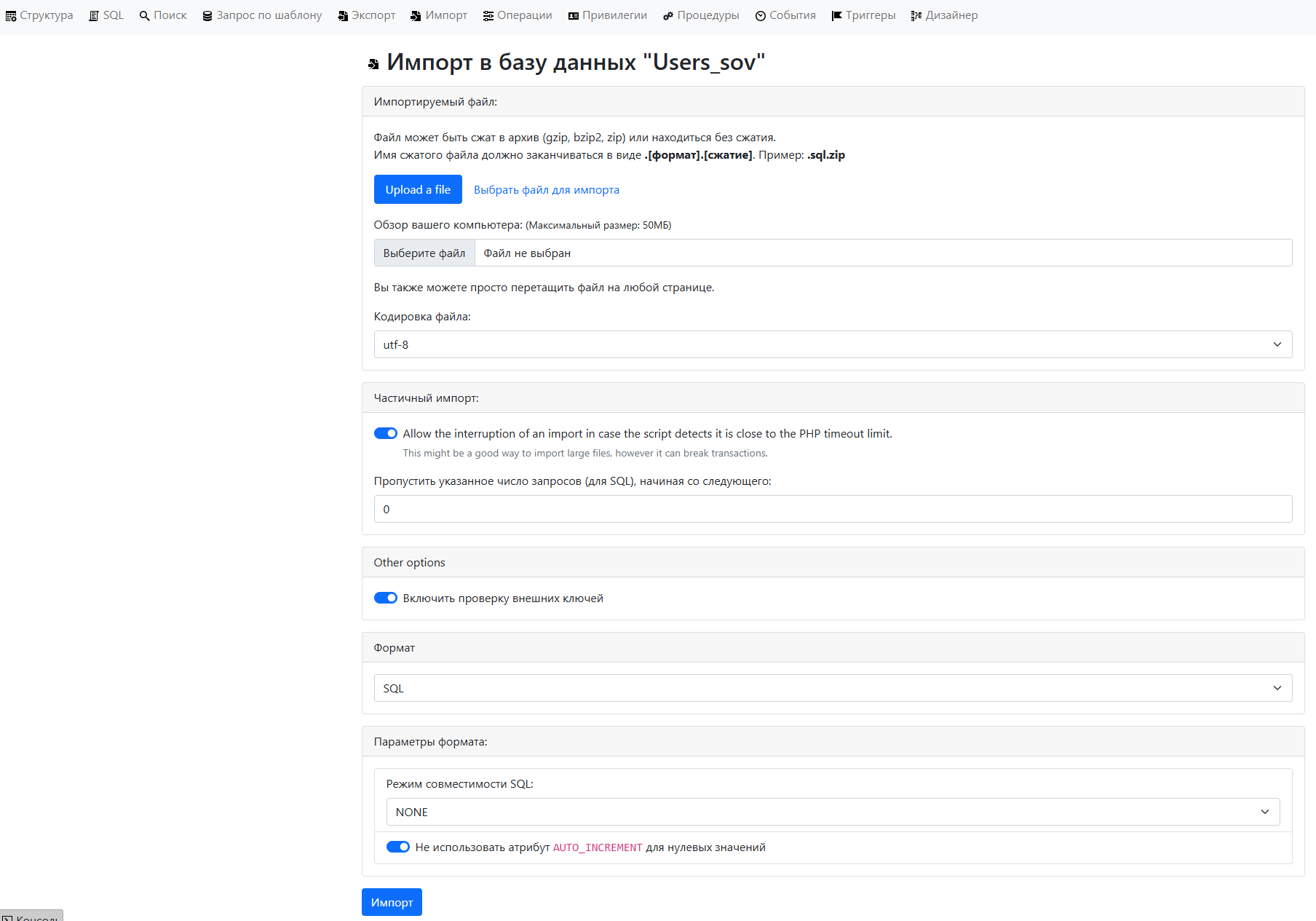


Рисунок 12 – экспорт таблицы

## 2.9. Разработка стратегии резервного копирования базы данных

**Цели резервного копирования**:

* + Восстановление после сбоев (аварийное восстановление).
  + Сохранение данных на случай человеческих ошибок (например, удаление данных).

1. **Полное резервное копирование**: Каждый Понедельник в 00:00.

* Копируются вся база данных.
* Требует больше времени и места, но обеспечивает быстрое восстановление.

1. **Инкрементное резервное копирование**: Ежедневно в 02:00.

* Копируются только изменения, сделанные с момента последнего резервного копирования (полного или инкрементного).
* Требует меньше места и времени, но восстановление занимает больше времени

1. **Транзакционные логи**: Каждые 15 минут.

* Транзакционные логи позволяют восстановить данные до последней успешной транзакции.
* Это особенно важно для систем, где критически важно минимизировать потерю данных (например, уровень пользователя).

1. **Хранение**: локально на сервере и в облаке (AWS S3 или Google Cloud Storage).

* Быстрый доступ к резервным копиям.
* Риск потери данных при физическом повреждении сервера.

1. **Тестирование восстановления**: Ежемесячно.

* Регулярное тестирование восстановления данных из резервных копий.
* Проверка целостности данных после восстановления.
* Документирование процесса восстановления.

## 2.10.1. Тестирование базы данных

Тестирование базы данных (БД) — важная часть проверки корректности работы системы. В данном документе представлены тесты для проверки взаимодействия PHP-кода с MySQL в рамках системы регистрации и входа пользователей. Тестирование включает:

* Проверку структуры БД.
* Тестирование SQL-запросов.
* Проверку целостности данных.
* Тестирование обработки ошибок.

**Проверку структуры БД**

Проверка структуры таблицы

|  |
| --- |
| sql  DESCRIBE users; --выведит структура бд |

Результат вывода: Структура корректно выводятся.

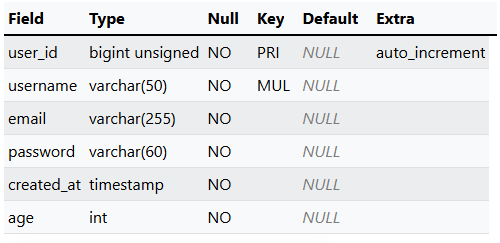


Рисунок 13 – структура таблицы

**Тестирование регистрации пользователя**

Добавляем нового пользователя.

|  |
| --- |
| sql  INSERT INTO `users` (`user\_id`, `username`, `email`, `password`, `created\_at`, `age`) VALUES (NULL, 'lain1r', 'aidmax1234@gmail.com', '1234', NOW(), '17'); |

Получаем результат

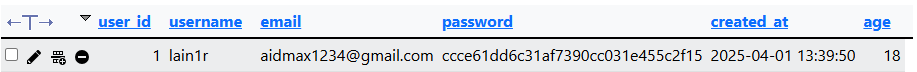


Рисунок 14 – результат регистрации

**Тестирование дублирования номера**

Проверяем, что система предотвращает регистрацию с уже существующим пользователя.

**SQL-запрос для проверки:**

|  |
| --- |
| sql  INSERT INTO `users` (`user\_id`, `username`, `email`, `password`, `created\_at`, `age`) VALUES (NULL, 'lain1r', 'aidmax1234@gmail.com', '1234', NOW(), '17'); |

Результат: сообщение «Пользователь с таким именим уже существует»

## 2.10.2. Оптимизация базы данных

Оптимизация базы данных представляет собой комплекс мер, направленных на повышение эффективности работы системы, сокращение времени выполнения запросов и рациональное использование вычислительных ресурсов. Ключевые методы оптимизации включают:

**1. Применение индексов**

Индексы являются одним из основных инструментов ускорения операций поиска и сортировки данных. Их использование позволяет сократить время доступа к информации за счет уменьшения объема сканируемых данных. Однако избыточное количество индексов может негативно сказаться на производительности операций модификации данных (вставка, обновление, удаление). Рекомендуется:

* Создавать индексы для столбцов, участвующих в условиях фильтрации (WHERE) и соединениях (JOIN).
* Анализировать и удалять неиспользуемые или дублирующие индексы.

**2. Нормализация структуры данных**

Нормализация обеспечивает устранение избыточности данных и минимизацию аномалий при их обработке. Разделение данных на логически связанные таблицы (например, выделение отдельных сущностей для пользователей, игровых сессий и транзакций) способствует соблюдению принципов целостности и непротиворечивости информации.

**3. Анализ и модификация запросов**

Производительность базы данных напрямую зависит от эффективности выполняемых запросов. Для выявления «узких мест» рекомендуется:

* Использовать встроенные средства СУБД, такие как EXPLAIN в PostgreSQL, для анализа плана выполнения запросов.
* Упрощать сложные запросы, разбивая их на несколько этапов.
* Ограничивать объем возвращаемых данных с помощью пагинации или выборки конкретных поле

**4. Оптимизация структуры таблиц**

Правильный выбор типов данных и организация хранения информации позволяют снизить нагрузку на систему. Основные рекомендации:

* Использовать минимально достаточные типы данных (например, SMALLINT вместо INT для числовых значений с малым диапазоном).
* Применять секционирование таблиц для распределения данных по логическим группам (например, по дате или категориям).

**5. Настройка параметров СУБД**

Конфигурация системы управления базами данных должна соответствовать текущим нагрузкам. Примеры настроек:

* Увеличение объема оперативной памяти, выделяемой для кэширования часто используемых данных.
* Активация автоматических процессов обслуживания, таких как очистка устаревших записей.

**6. Регулярное техническое обслуживание**

Периодическое обслуживание базы данных необходимо для поддержания ее производительности. К основным процедурам относятся:

* Перестроение фрагментированных индексов.
* Архивация или удаление устаревших данных.

# ЗАКЛЮЧЕНИЕ

В ходе изучения технологии разработки и защиты баз данных было выявлено, что успешная работа с данными в современных условиях требует комплексного подхода, включающего как аспекты проектирования, так и обеспечения безопасности. Принимая во внимание стремительное развитие информационных технологий, применение современных методов и инструментов становится неизбежным. Это позволяет не только оптимизировать процессы хранения и обработки данных, но и защитить их от возможных угроз.

Особое внимание следует уделить вопросам шифрования данных, аутентификации пользователей и резервного копирования. Эффективная защита баз данных требует активного мониторинга и регулярного обновления систем безопасности.

Таким образом, подходы к разработке и защите баз данных должны быть гибкими и адаптивными, чтобы соответствовать быстро меняющимся требованиям и угрозам. В будущем важно продолжать исследовать и внедрять новые технологии, что позволит обеспечить надежность и безопасность информации на всех уровнях.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Волик, М.В. Разработка базы данных в Access / М.В. Волик. – Издательство «Прометей», 2021. – 88 с.
2. Даккет, Дж. PHP и MySQL. Серверная веб-разработка / Дж. Даккет. – Издательство «Эксмо», 2023. – 688 с.
3. Дьяков, И.А. Базы данных. Язык SQL : учебное пособие / И.А. Дьяков. – Тамбов : Тамбовский государственный технический университет (ТГТУ), 2022. – 82 с.
4. Ла Рокка, М. Продвинутые и структуры данных. / М. ла Рокка. – Издательство «Питер», 2024. – 848 с.
5. Никсон, Р. Создаём динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. 6-е изд. / Р. Никсон. – Издательство «Питер»: Санкт-Петербург, 2023. – 832 с.
6. https://www.figma.com/ (дата обращения: 15.03.2025).
7. https://www.github.com/ (дата обращения: 16.03.2025).

ПРИЛОЖЕНИЯ

(Скриншоты разработанной базы данных)