ВВЕДЕНИЕ

Мир компьютерных игр развивается с каждым годом. Всего за каких-то 30 лет простые 2D разработки превратились в полномасштабные проекты, которые поражают своей графической составляющей и невероятным вниманием к деталям. Важной частью этого бурного роста стало появление онлайна в компьютерных играх. Теперь появилась возможность сыграть в любимую игру вместе со своими друзьями со всего мира.

В связи с этим стали появляться специальные сайты, которые собирают информацию обо всех серверах, которые доступны для данной игры и предоставляют пользователю возможность подобрать идеально подходящий для себя сервер, исходя из доступной информации. При этом практически каждый сайт обладает своими проблемами, которые будут описаны позже.

Таким образом, целью данного курсового проекта является создание информационной системы для серверов игры. Для достижения данной цели необходимо решить следующие задачи:

- рассмотреть существующие решения;
- выбрать модель хранения данных;
- разработать базу данных;
- выделить роли пользователей;
- выбрать необходимый набор технологий для разработки;
- создать программный продукт, который решает поставленную цель.

1 Аналитическая часть

В данном разделе будет проведен анализ существующих решений на русском и зарубежном рынках. Также будет произведена формализация задачи и данных, описание типов пользователей, а также обзор существующих типов баз данных.

1.1 Анализ существующих решений

Существует большое количество сайтов, которые предоставляют возможность найти сервера для той или иной онлайн игры.

1.1.1 Российский рынок

Одним из сайтов для поиска серверов для игры «Minecraft» [1] является «MinecaftRating» [2]. При этом интерфейс главной страницы представлен на рисунке 1.1. На нем предоставляется большое количество функциональностей, таких как:

- поиск серверов (с сортировкой по самым популярным, по версии, по количеству игроков);
- получение полной информации о каждом сервере;
- добавление нового сервера;
- для зарегестрированных пользователей — добавление в избранное.

При этом не имеется возможности посмотреть сервера для разных платформ.

Для компьютерной игры «Counter-Strike» [3] также существуют сайты с серверами. Примером такого сайта является «Сервера КС» [4]. На рисунке 1.2 представлен интерфейс главной страницы. Для данной игры рынок развит слабее, поэтому функциональностей куда меньше. При этом предоставляются следующие возможности:

- просмотр серверов (присутсвует лишь список, поиск и сортировка невозможны);

- для зарегестрированных пользователей — добавление нового сервера, добавление в избранное.

Из-за отсутсвия сортировок серверов, имеется возможность купить место в верху таблицы, чтобы пользователи замечали сначала проплаченные сервера.

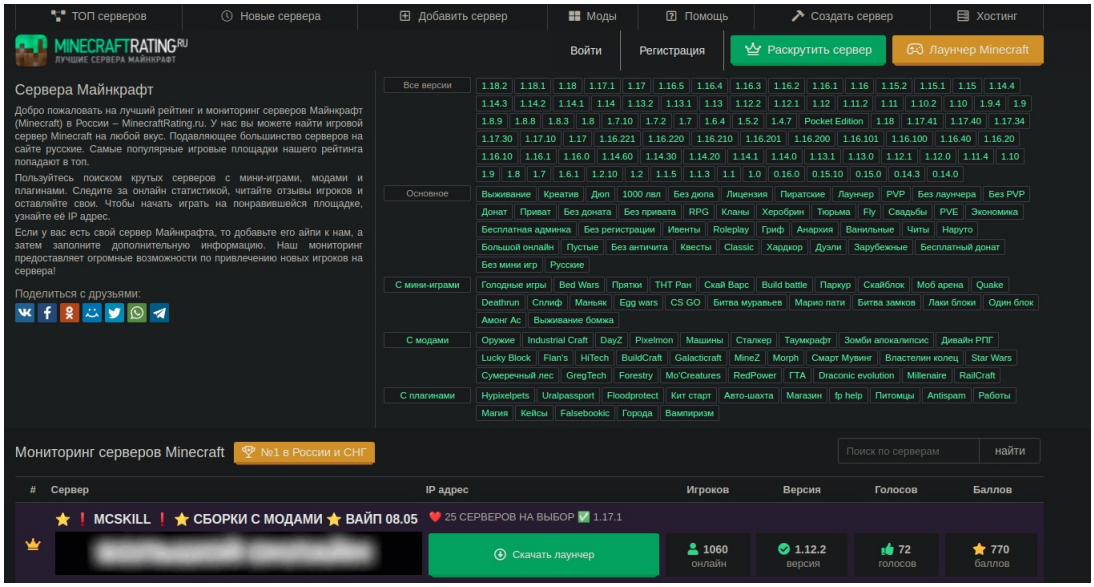

Рисунок 1.1 – Сайт для поиска серверов для игры «Minecraft» на русском рынке

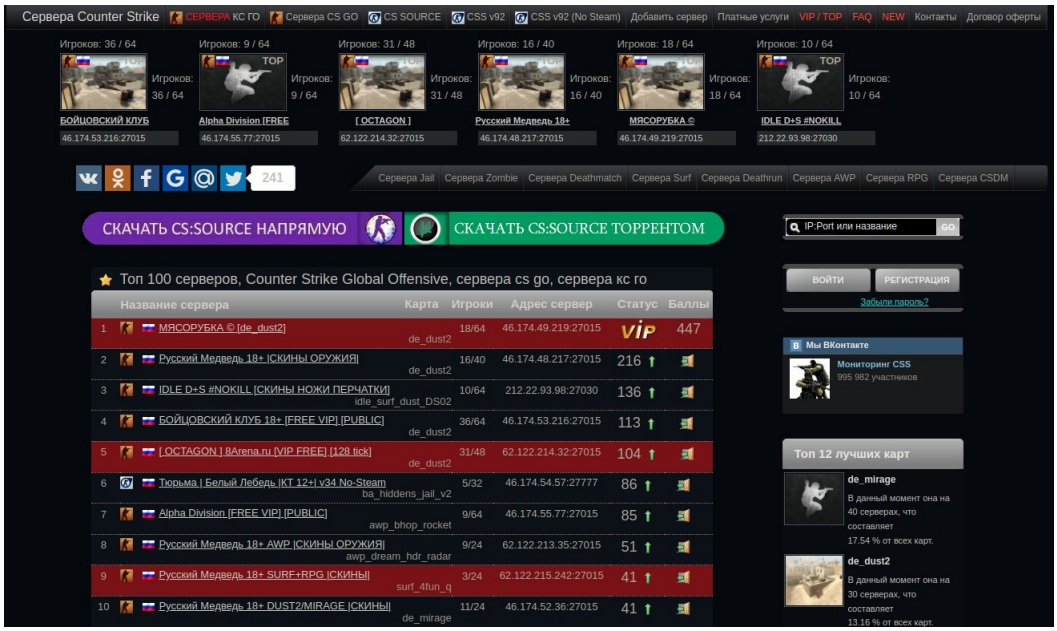

Рисунок 1.2 – Сайт для поиска серверов для игры «Counter-Strike» на русском рынке

1.1.2 Зарубежный рынок

Примером сайта для поиска серверов для игры «Minecraft» [1] является «Minecraft Servers» [5]. На рисунке 1.3 представлен интерфейс главной страницы.Зарубежный аналог обладает тем же самым функционалом, что и пример сайта с российского рынка. При этом поиск серверов для различных платформ также отсутсвует.

Для игры «Counter-Strike» [3] зарубежный рынок развит сильнее. Так, примером является сайт «Game Tracker» [6]. Интерфейс главной страницы представлен на рисунке 1.4. Он обладает тем же самым функционалом, что и его российский аналог, а также:

- присутсвует сортировка по различным параметрам;
- существует возможность выбора серверов для различных стран мира.

При этом возможность покупки приоритетного места в списке отсутсвует.

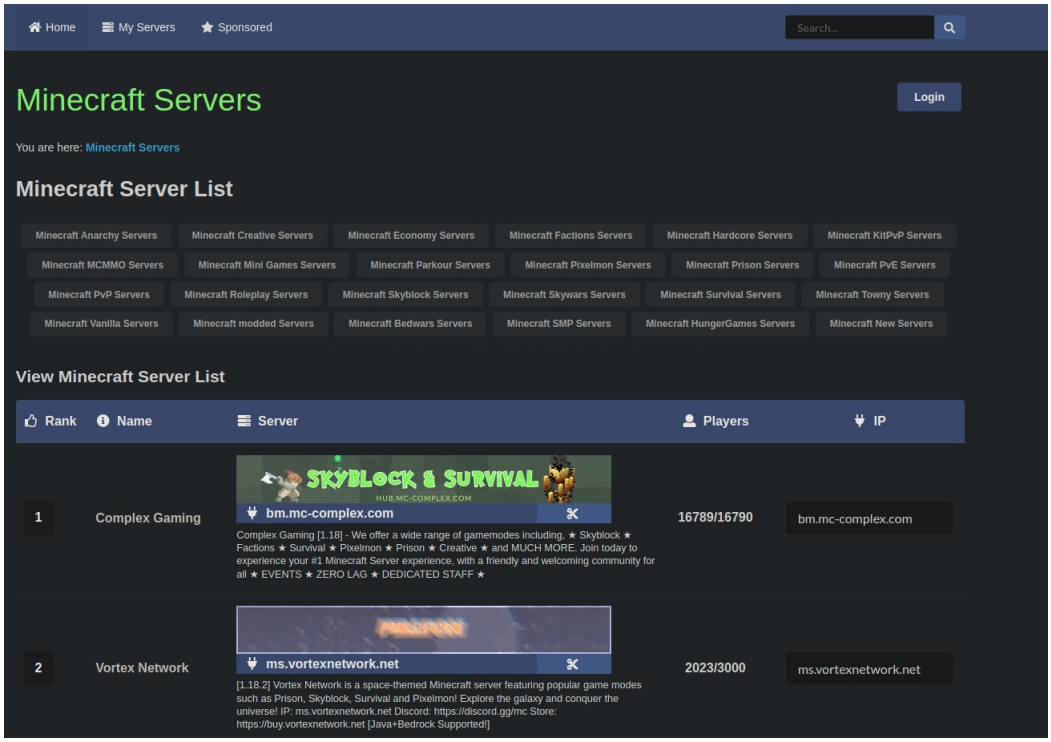

Рисунок 1.3 – Сайт для поиска серверов для игры «Minecraft» на зарубежном рынке

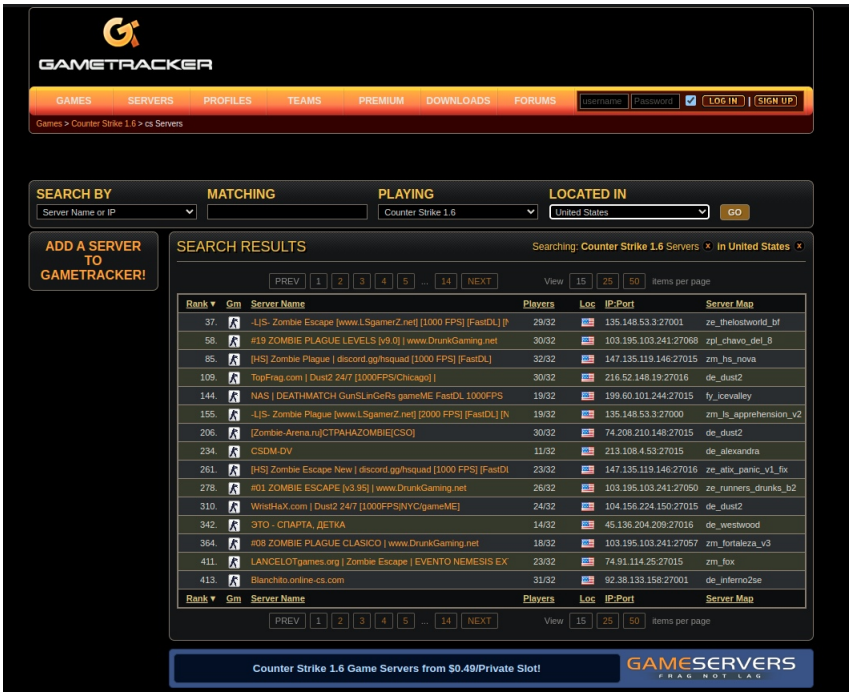

Рисунок 1.4 – Сайт для поиска серверов для игры «Counter-Strike» на зарубежном рынке

Вывод

В таблице 1.1 представлены результаты сравнения существующих решений. Выделенные критерии сравнения:

- К1 – просмотр списка серверов;
- К2 – сортировка списка серверов;
- К3 – просмотр серверов для разных платформ;
- К4 – добавление в избранное;
- К5 – получение подробной информации о сервере.

Таблица 1.1 – Сравнение существующих решений

Название	**К1**	**К2**	**К3**	**К4**	**К5**
«MinecaftRating»	+	+	-	+	+
«Сервера КС»	+	-	-	-	-
«Minecraft Servers»	+	+	-	+	-
«Game Tracker»	+	+	-	+	+

Таким образом, российский и зарубженый рынки предоставляют множество различных сайтов для нахождения серверов для игр. Стоит отметить то, что при этом сайты обладают рядом серьезных проблем.

1. Отсутсвует возможность просмотреть списки серверов для разных платформ одной и той же игры.
2. Недостаточная функциональность для поиска необходимых серверов.
3. Слабая развитость рынка для некоторых игр.

1.2 Формализация задачи

Должно быть разработано веб-приложение для поиска серверов для игры. При этом ПО должно содержать в себе следующие возможности:

- просмотр списка серверов с сортировкой по параметрам;
- авторизация и регистрация на сайте;
- просмотра списка избранных серверов;
- изменение рейтинга серверов;
- просмотр списка серверов для определенной платформы;
- добавление/изменение/удаление серверов администратором;
- изменение ролей пользователей администратором.

1.2.1 Список серверов

Основным процессом является просмотр списка серверов. Список серверов представляет из себя таблицу из всех имеющихся серверов в базе данных. Информационные поля сервера указаны в колонках таблицы, а каждый отдельный сервер — строка этой таблицы.

Также таблица должна иметь возможонсть сортировки по полям сервера (названию, IP-адресу, версии игры, рейтингу сервера) и имени платформы, на которой запущен сервер.

1.2.2 Авторизация и регистрация

Должна быть введена возможность регистрации пользователя на сайте, чтобы открыть ему дополнительные возможности – просмотр информации о хостинге сервера, списка игроков сервера и добавления сервера в список избранного.

1.2.3 Список избранных серверов

Каждый зарегестрированный пользователь должен обладать возможность добавить интересующий его сервер в список избранных. Это необходимо для того, чтобы пользователь не потерял интересующий его сервер и всегда имел быстрый доступ к информации о нем.

1.2.4 Рейтинг серверов

Рейтинг сервера формируется из количества добавлений данного сервера в список избранных серверов отдельно взятого пользователя. Данный рейтинг выводится в качестве поля в таблице серверов, предоставляя возможность пользователям узнать наиболее популярный сервер. При этом, если пользователь сайта удалил данный сервер из своего списка избранных серверов, то рейтинг сервера понизится.

1.2.5 Деление серверов по платформам

Каждый сервер может находится лишь на одной единственной платформе. Поэтому важно разделить список серверов на отдельные списки для каждой платформы, чтобы пользователь мог выбрать именно те сервера, которые подходят для его рабочего устройства.

1.2.6 Добавление/изменение/удаление серверов

Данной возможностью наделен лишь администратор сайта. Должен быть предоставлен интерфейс для данного процесса. При добавлении/изменении должны быть добавлены ограничения на ввод информации, чтобы предотвратить ошибки ввода, а также недопустить появления серверов с таким же названием или на том же IP-адресе. При удалении должно быть реализовано подтверждение удаления сервера, чтобы предотвратить случайные нажатия.

1.2.7 Изменение ролей зарегестрированных пользователей

Должна быть введена возможность изменения роли зарегестрированного пользователя. При этом администратор не может изменить собственную роль, а также администратору с никнеймом «admin» должно быть запрещено изменять роль, чтобы на сайте был всегда, как минимум, один администратор.

1.3 Описание типов пользовтелей

В задаче выделено 3 типа пользователей (таблица 1.2).

Таблица 1.2 – Типы пользователей

Тип	**Функциональность**
Гость (неавторизованный пользователь)	• Просмотр списка серверов • Выбор платформы для списка серверов • Сортировка списка серверов • Регистрация • Авторизация
Авторизованный пользователь	• Добавление сервера в список избранных • Выбор платформы для списка серверов • Просмотр списка серверов • Сортировка списка серверов • Просмотр информации о хостинге сервера, а также списка игроков сервера
Администратор	• Добавление нового сервера на сайт • Удаление сервера с сайта • Изменение информации о сервере • Изменение ролей зарегестрированных пользователей • Добавление сервера в список избранных • Просмотр списка серверов • Выбор платформы для списка серверов • Сортировка списка серверов • Просмотр информации о хостинге сервера, а также списка игроков сервера

На рисунках 1.5-1.7 представлены Use-Case диаграммы выделенных типов пользователей.

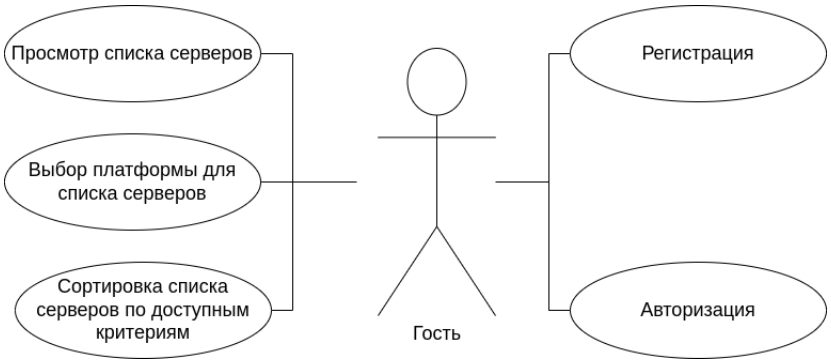

Рисунок 1.5 – Use-Case диаграмма для неавторизованного пользователя

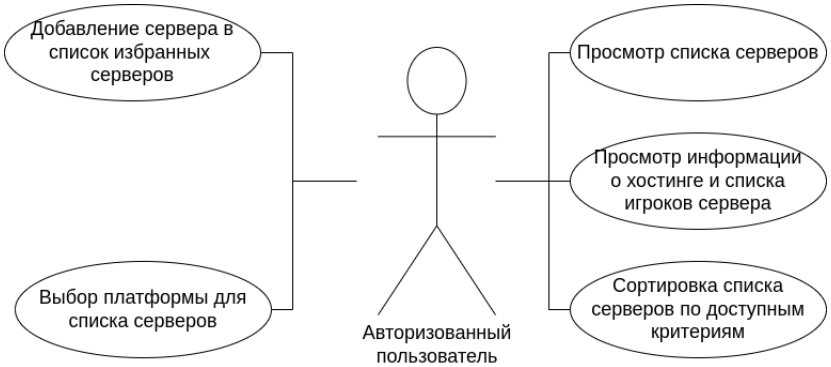

Рисунок 1.6 – Use-Case диаграмма для авторизованного пользователя

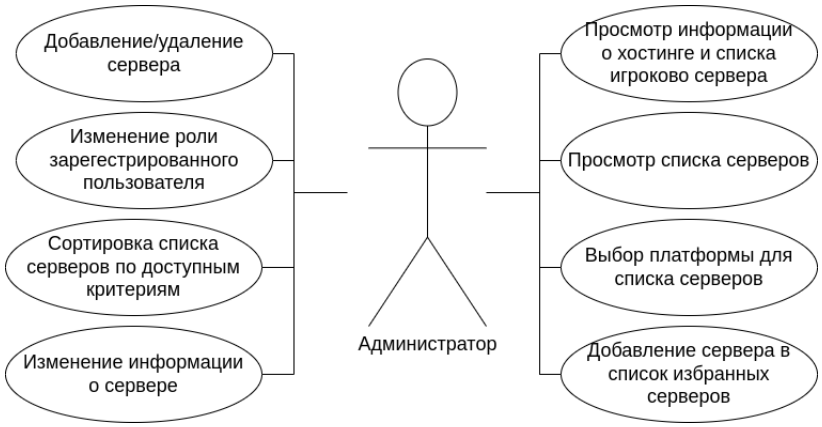

Рисунок 1.7 – Use-Case диаграмма для администратора

1.4 Формализация данных

В соответствии с задачей и типами пользователей, база данных должна содержать следующие модели (таблица 1.3):

- сервер;
- пользователь;
- платформа;
- игрок;
- хостинг.

Таблица 1.3 – Модели базы данных

Тип	**Функциональность**
Серевер	• ID • Название • IP-адрес • Версия игры • Рейтинг • ID платформы • ID хостинга
Платформа	• ID • Название • Популярность • Стоимость
Хостинг	• ID • Название • Плата в месяц
Игрок	• ID • Никнейм • Сыграно часов на сервере • Дата последнего захода на сервер
Пользователь	• ID • Логин • Пароль • Роль

Также на рисунке 1.8 представлена ER-диаграмма [7] разрабатываемой системы в нотации Чена.

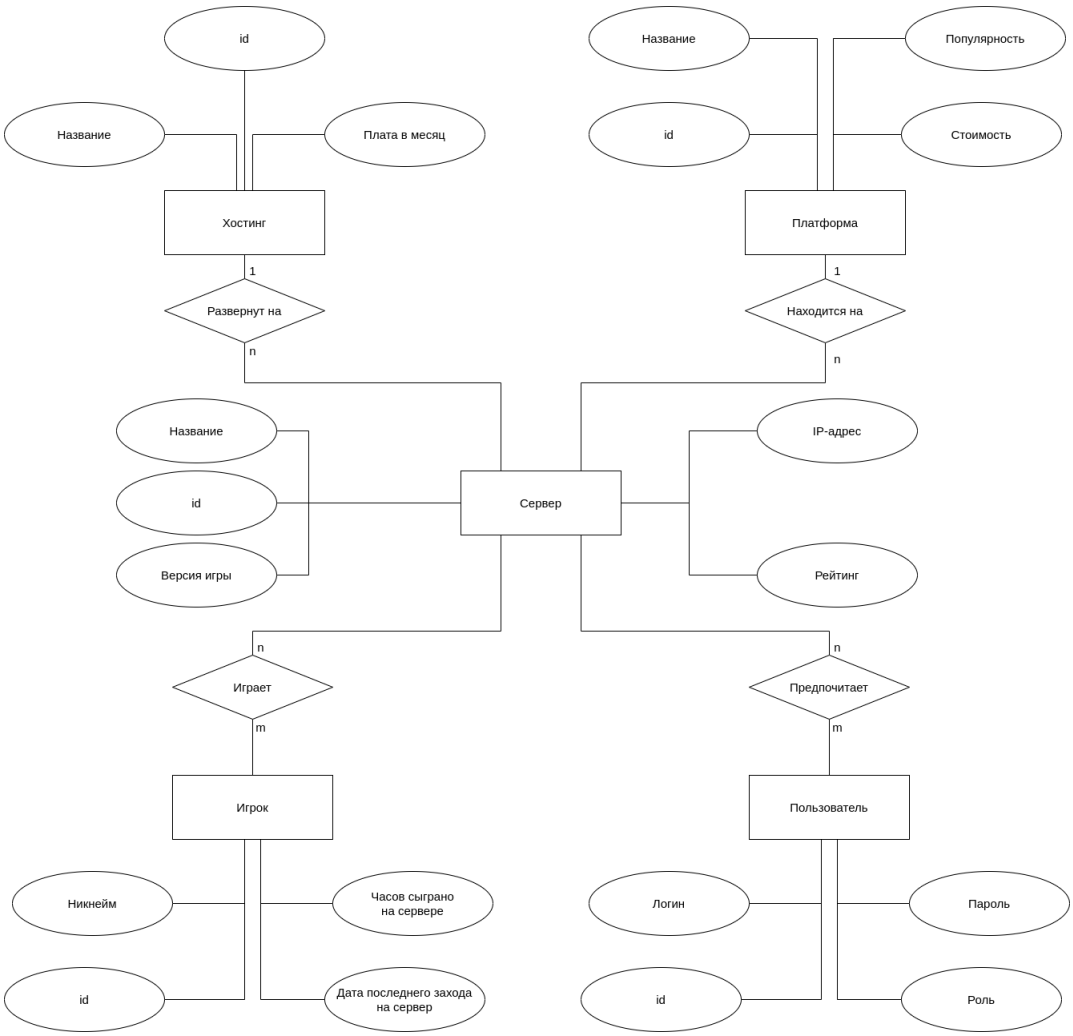

Рисунок 1.8 – ER-диаграмма в нотации Чена

1.5 Выбор модели базы данных

База данных [8] — упорядоченный набор структурированныой информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. База данных обычно управляется системой управления базами данных (СУБД). Данные вместе с СУБД, а также приложения, которые с ними связаны, называются базой данных.

Данные в наиболее распространенных типах современных баз данных обычно хранятся в виде строк и столбцов формирующих таблицу [8]. Этими данными можно управлять, изменять, обновлять, контролировать и упорядочивать.

Базы данных делятся на три модели организации данных:

- дореляционные;

- реляционные;
- постреляционные.

1.5.1 Дореляционные модели

Дореляционные модели базы данных [9] предоставляли доступ на уровне записей, которые располагались в виде древовидной структуры со связями предок-потомок. При этом взаимодействие с базой данных происходило с использованием языков программирования, которые были расширены функциями дореляционных СУБД.

Главными недостатками является то, что оптимизация доступа к данным со стороны системы отсутствует, а также древовидная структура является весьма трудоемкой.

1.5.2 Реляционные модели

Реляционная база данных [10] — совокупность отношений, которые содержат всю информацию, которая должна храниться в базе данных. Каждое отношение — двумерная таблица, в каждой строке которой хранится запись об объекте, а в каждом столбце — свойства данного объекта.

Реляционные базы данных обладают несомненным преимуществом – благодаря стандартизированного языка запросов SQL, существует возможность подмены СУБД.

1.5.3 Постреляционные модели

В постреляционных моделях баз данных [11] не используется табличная схема строк и столбцов. В этих базах данных применяется модель хранения, оптимизированная под конкретные требования типа хранимых данных.

При этом они делятся на следующие основные категории:

- коллекции — документы, упорядоченные по группам;
- ключ-значение — хэш-таблица, в которой по ключу находится значение;
- колоночная — хранит информацию в виде разреженной матрицы, строки и столбцы которой используются как ключи;

- графовые — сетевая база, использующие узлы и ребра для хранения данных.

Данные модели используются для спефических задач, где явно подходит одна из приведенных выше категорий, что явно ускоряет работу программного продукта благодаря грамотной работе с данными.

Вывод

Для поставленной задачи разработки информационной системы наилучшим обрзаом подходит реляционная модель хранения данных. Выбор обусловлен необходимостью хранить данные в виде структированных таблиц, между которыми проведена связь.

1.6 Вывод

В данном разделе была проанализирована выполняемая задача — была проведена ее формализация, проведена формализация данных, описаны типы пользователей. Также были расммотрены модели базы данных и выбрана реляционная модель.

2 Конструкторская часть

В данном разделе приведено описание таблиц разрабатываемой базы данных, а также описаны функции, триггеры и роли базы данных.

2.1 Описание таблиц базы данных

Схема разрабатываемой базы данных приведена на рисунке 2.1.

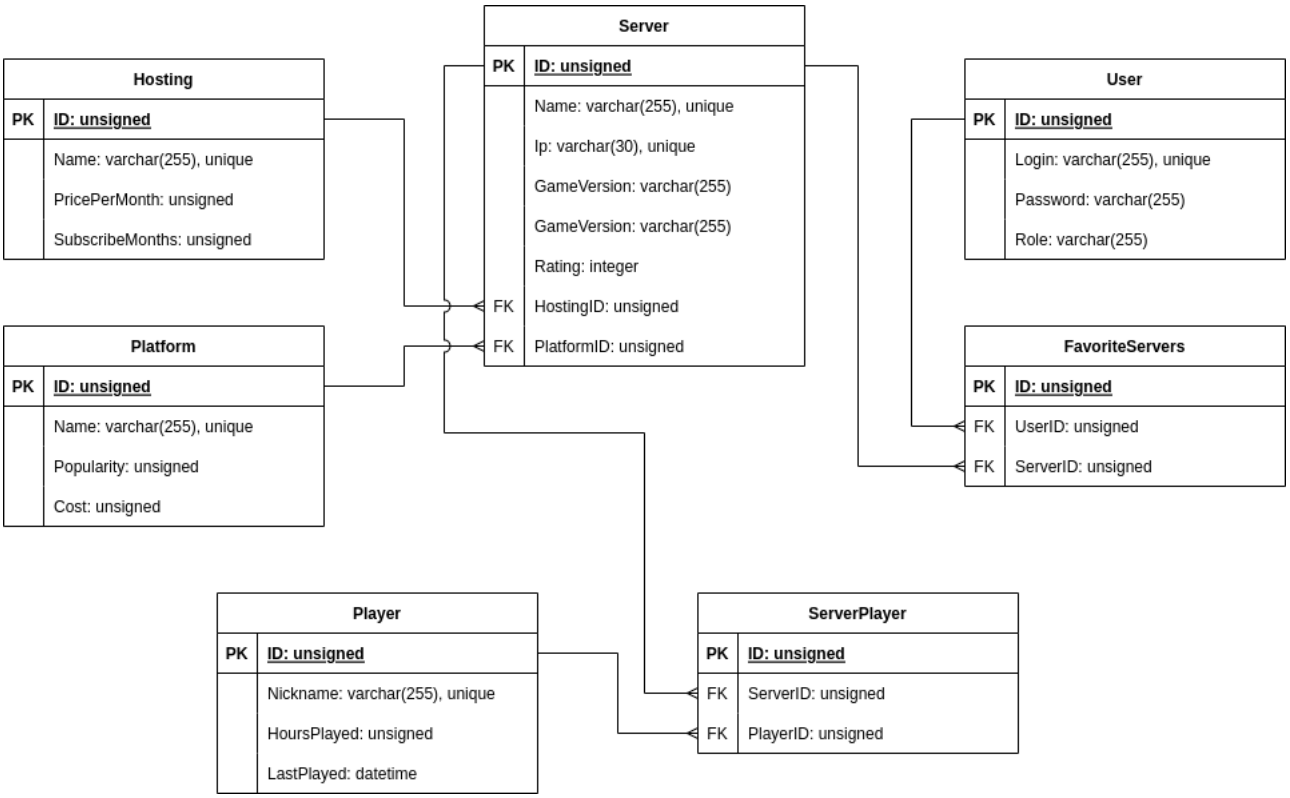

Рисунок 2.1 – Диаграмма базы данных

Реализуемая модель базы данных содержит 7 таблиц.

1. Таблица «Server» хранит информацию о серверах. Содержит следующие поля:

   - ID – первичный ключ; тип данных – unsigned;
   - Name – уникальное название сервера; тип данных – varchar(255);
   - IP – уникальный IP-адрес сервера; тип данных – varchar(30);
   - GameVersion – версия игры, для которой открыт данный сервер; тип данных – varchar(255);
   - Rating – рейтинг сервера; тип данных – integer;

- HostingID – идентификатор хостинга; тип данных – unsigned;
- PlatformID – идентификатор платформы; тип данных – unsigned;

2. Таблица «User» хранит информацию о пользователе сайта. Содержит следующие поля:
   - ID – первичный ключ; тип данных – unsigned;
   - Login – уникальный логин на сайте; тип данных – varchar(255);
   - Password – пароль на сайте; тип данных – varchar(255);
   - Role – роль пользователя; тип данных – varchar(255);
3. Таблица «FavoriteServer» хранит информацию об избранных серверах игроков. Является связующей таблицей между таблицами «Server» и «User». Содержит следующие поля:
   - ID – первичный ключ; тип данных – unsigned;
   - UserID – идентификатор пользователя; тип данных – unsigned;
   - ServerID – идентификатор сервера; тип данных – unsigned;
4. Таблица «Player» хранит информацию об игроке сервера. Содержит следующие поля:
   - ID – первичный ключ; тип данных – unsigned;
   - Nickname – уникальный никнейм игрока; тип данных – varchar(255);
   - HoursPlayed – количество часов, сыгранных на серверах; тип данных – unsigned;
   - LastPlayed – дата крайнего раза захода на сервер; тип данных – unsigned;
5. Таблица «FavoriteServer» хранит информацию об игроках и на каких серверах они играют. Является связующей таблицей между таблицами «Server» и «Player». Содержит следующие поля:
   - ID – первичный ключ; тип данных – unsigned;
   - PlayerID – идентификатор игрока; тип данных – unsigned;

- ServerID – идентификатор сервера; тип данных – unsigned;

6. Таблица «Hosting» хранит информацию о хостинге, на котором находится сервер. Содержит следующие поля:

   - ID – первичный ключ; тип данных – unsigned;
   - Name – уникальное название хостинга; тип данных – varchar(255);
   - PricePerMonth – плата за размещение сервера на хостинге в месяц; тип данных – unsigned;
   - SubMonts – количество месяцев подписки сервера на хостинг; тип данных – unsigned;

7. Таблица «Platform» хранит информацию о платформах, на которых может быть запущена игра. Содержит следующие поля:

   - ID – первичный ключ; тип данных – unsigned;
   - Name – уникальное название платформы; тип данных – varchar(255);
   - Popularity – популярность платформы; тип данных – unsigned;
   - Cost – стоимость платформы; тип данных – unsigned;

2.2 Функции базы данных

В проектируемой базе данных определены следующие функции.

1. **Функция выбора серверов**. Данная функция выбирает или все сервера из таиблицы «Server», или только избранные сервера пользователя по его идентификатору. Схема алгоритма функции представлена на рисунке 2.2.

2. **Функция фильтрации серверов**. Данная функция фильтрует сервера из таблицы «Server», полученные после работы функции выбора серверов. Фильтрация происходит по названию сервера и/или идентификатору платформы. Схема алгоритма функции представлена на рисунке 2.5.

3. **Функция сортировки серверов**. Данная функция сортирует сервера из таблицы «Server», полученные после работы функции фильтрации серверов. Сортировка происходит по полям: название сервера, IP-адрес сервера, версия игры сервера, рейтинг сервера, названию платформы (все – по

возрастанию и по убыванию). Схема алгоритма функции представлена на рисунке 2.4.

4. **Функция парсинга серверов**. Данная функция является некой «прослойкой», которая возвращает все сервера из таблицы «Server», полученных после работы функции сортировки серверов. Схема алгоритма функции представлена на рисунке 2.3.

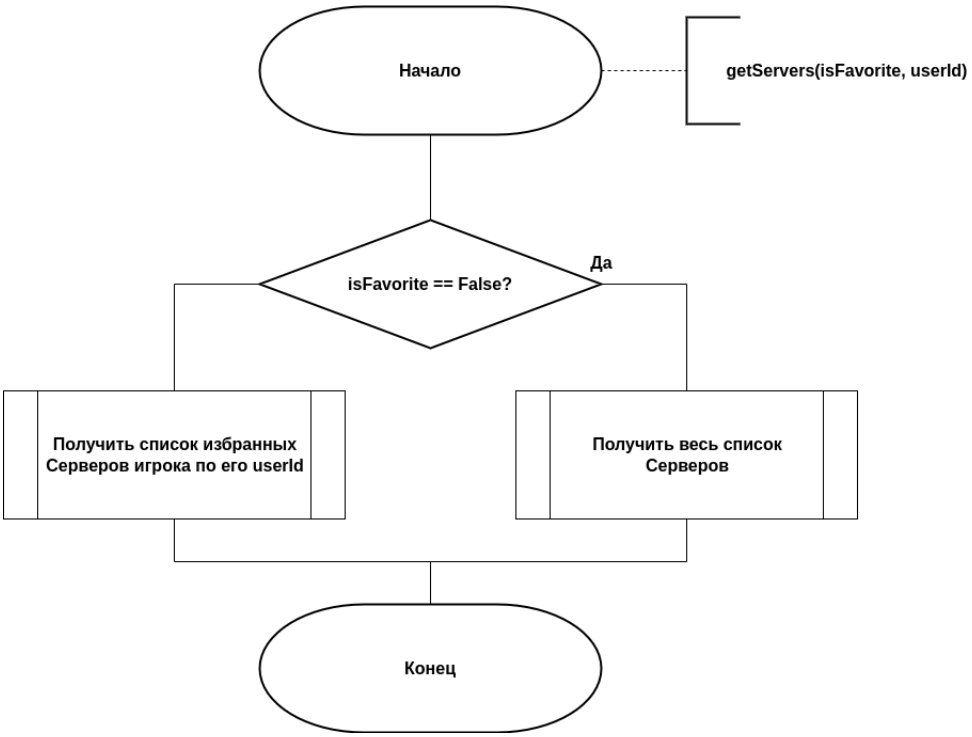

Рисунок 2.2 – Функция выбора серверов

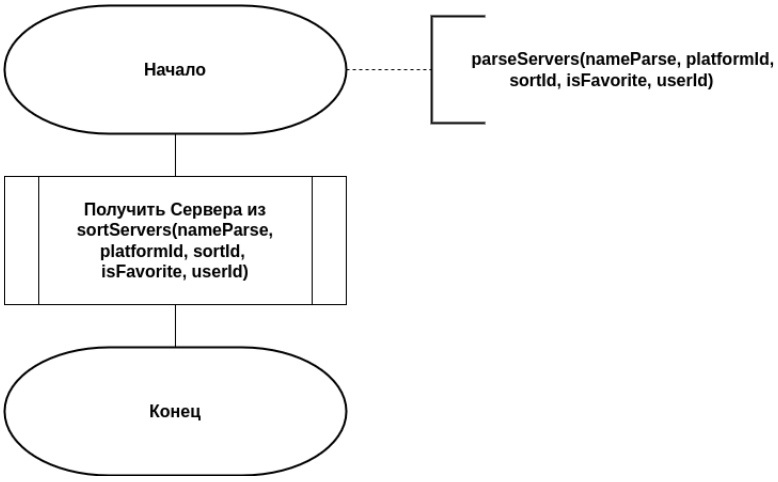

Рисунок 2.3 – Функция парсинга серверов

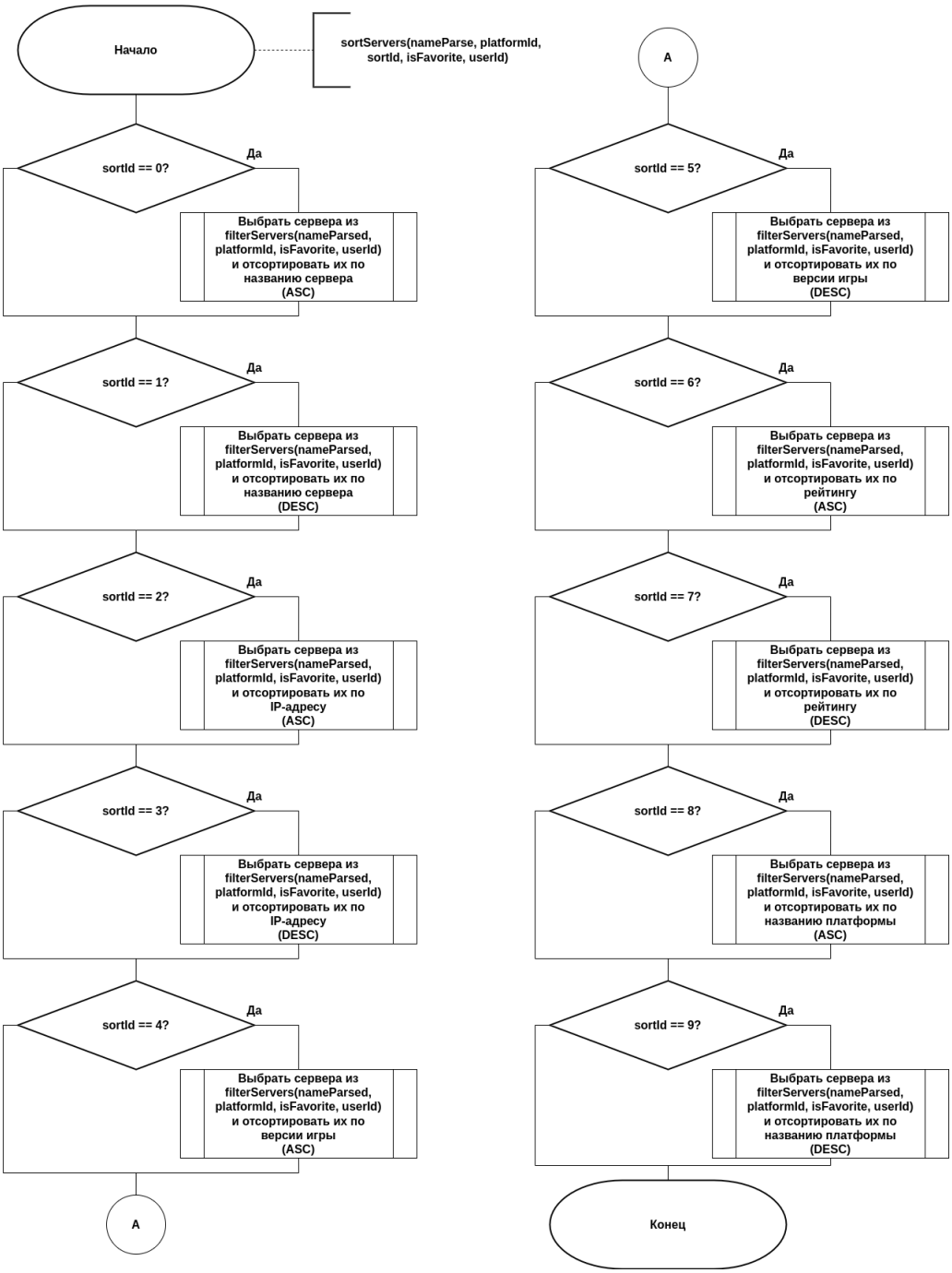

Рисунок 2.4 – Функция сортировки серверов

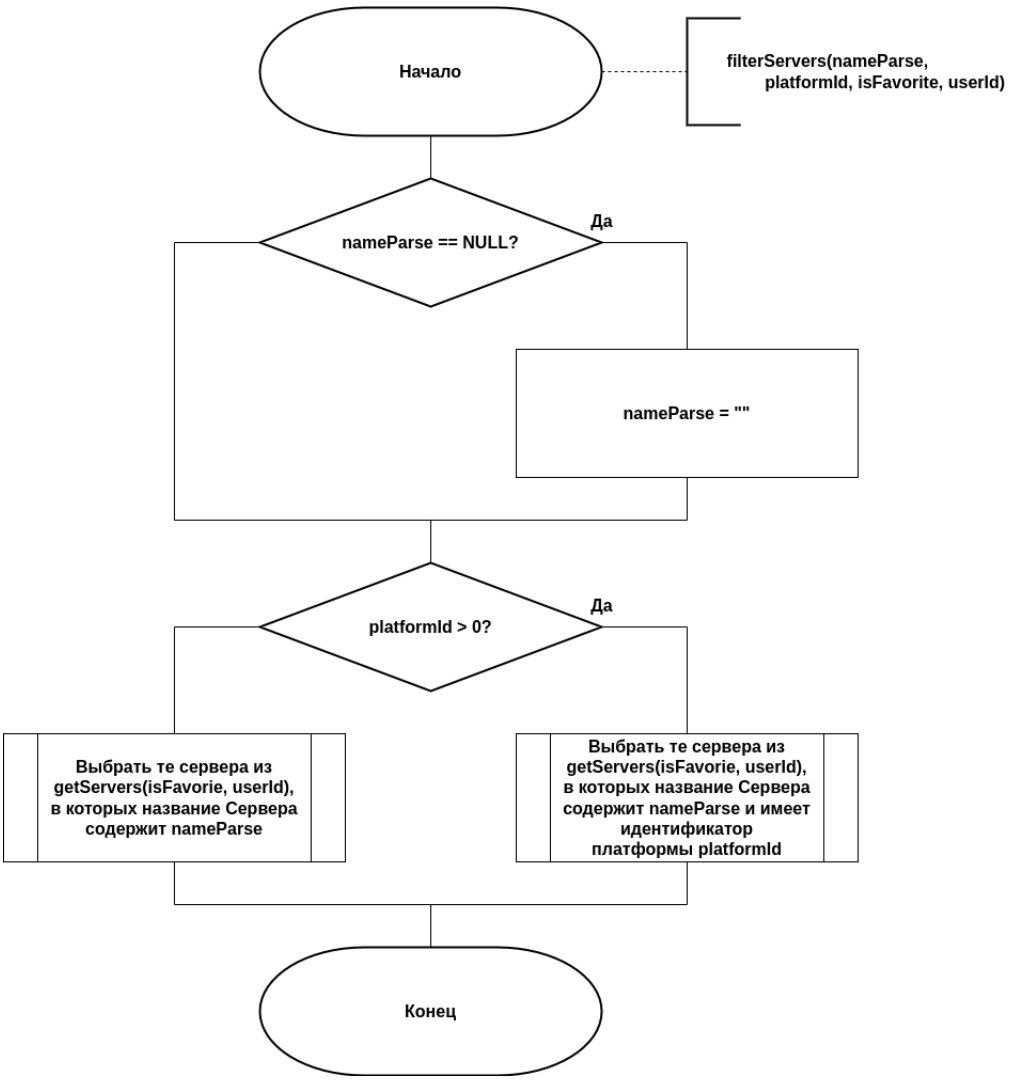

Рисунок 2.5 – Функция фильтрации серверов

2.3 Триггеры базы данных

В проектируемой базе данных определены следующие триггеры.

1. **Триггер установки роли пользователя**. Данный триггер срабатывает при операции «Insert» в таблицу «User» и вызывает функцию установки роли пользователя, которая выдает пользователю роль «User» или «Admin», если его логин «admin». Таким образом, в базе данных точно будет один администратор для управления сайтом. Схема алгоритма вызываемой функции представлена на рисунке 2.6.

2. **Триггеры изменения рейтинга сервера**. Один триггер срабатывает при операции «Insert» в таблицу «FavoriteServer» и вызывает функцию *увеличения* рейтинга добавленного сервера на единицу, а второй триггер –

при операции «Delete» из таблицы «FavoriteServer» и вызывает функцию *уменьшения* рейтинга удаляемого сервера на единицу. Схема алгоритма вызываемой функции представлена на рисунке 2.6.

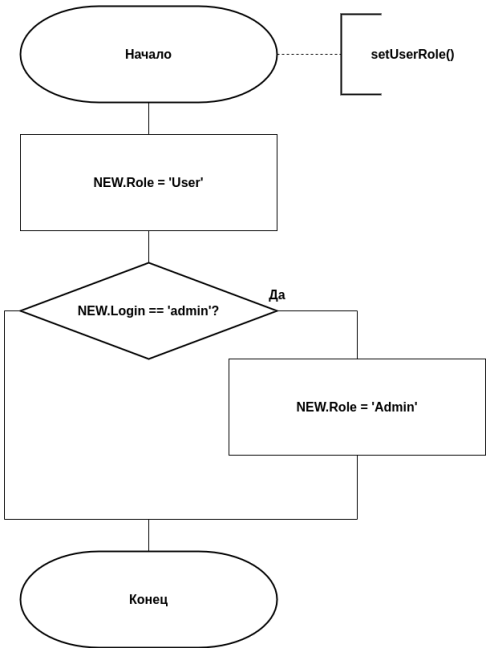

Рисунок 2.6 – Функция выдачи роли пользователю

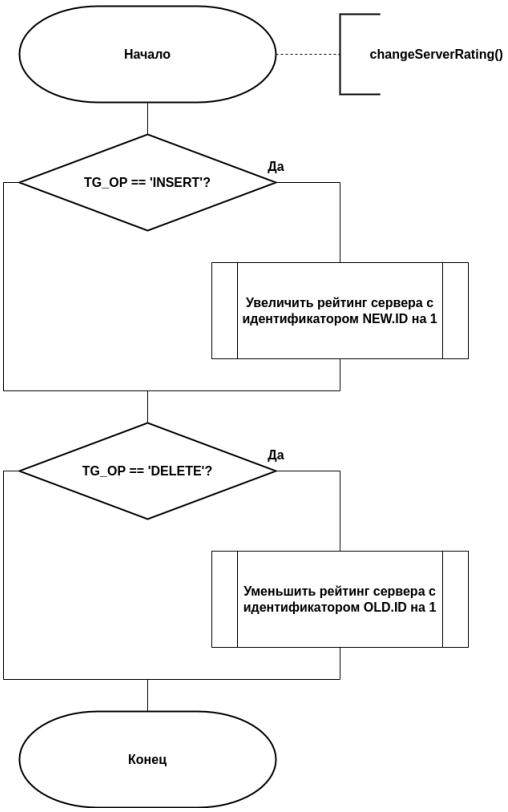

Рисунок 2.7 – Функция изменения рейтинга сервера

2.4 Роли базы данных

В проектируемой базе данных определены следующие роли.

1. **Неавторизованный пользователь**. Данная роль обладает возможность просмотра таблиц серверов и платформ, так как общий список серверов доступен всем видам пользователей. Также обладает возможностью добавлять в таблицу «User» для регистрации нового аккаунта.

2. **Авторизованный пользователь**. Данная роль обладает возможность просмотра таблиц серверов, платформ для списка серверов; просмотра таблиц «Hosting», «Player» и «ServerPlayer» для получения детальной информации о сервере; просмотра таблицы «FavoriteServer» для просмотра серверов из списка избранного. Также обладает возможностью добавлять и удаления в таблице «FavoriteServer» для добавления и удаления серверов из списка изрбанного.

3. **Администратор сайта**. Данная роль обладает возможность просмотра, добавления и удаления на все определенные в базе данных таблицы.

Вывод

В данном разделе были подробно описаны все поля всех таблиц проектирумой базы данных, ее функции, триггеры и роли.

3 Технологическая часть

В данном разделе будут рассмотрены средства разработки программного обеспечения, детали реализации, а также диаграмма классов.

3.1 Средства реализации

В качестве языка программирования был выбран язык *C#* [**csharp**]. Он обладает возможность создания веб-приложений. Также данный язык программирования поддерживает принципы ООП, что крайне важно для поставленной задачи – имеется возможность создать понятную и простую структуру программного кода.

Фреймворком для работы с СУБД был выбран *EntityFramework* [**ef**]. Данный фреймворк является основным для языка C# и имеет множество учебной литературы.

Visual Studio [**vs**] был выбран в качестве среды разработки, так как имеется беспалтная версия для студентов, хороший дебаггер и полноценная поддержка языка C#.

Для разработки непосредственно веб-приложения используется фреймворк *ASP.NET* [**asp**]. Данный фреймворк является абсолютно бесплатным и обладает широким функционалом для разработки. Имеет поддержку мультистраничных сайтов.

3.2 Архитектура приложения

Архитектура приложения построена на основе схемы MVC (модель-представление-контроллер) [**mvc**]. Использование данной схемы позволяет разрабатывать каждый из компонентов независимо от других. Это имеет следующие преимущества:

- изменения одного из компонентов не влияют на работоспособность других компонентов;
- имеется возможность подмены (веб-интерфейс может быть заменен на интерфейс десктопного приложения).

Схема взаимодействие компонентов MVC представлена на рисунке 3.1

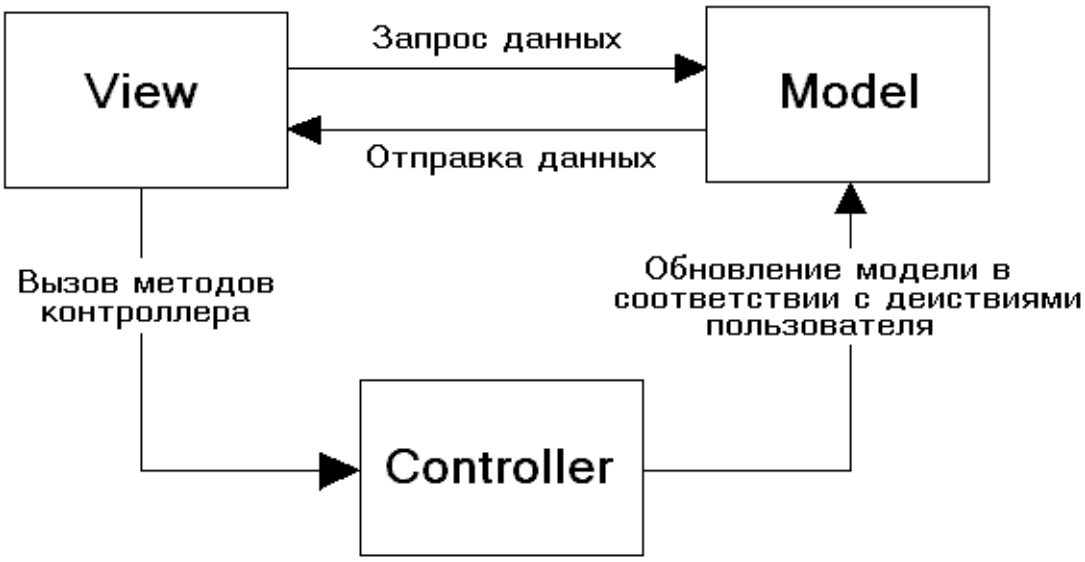

Рисунок 3.1 – Схема взаимодействие компонентов MVC

3.3 Структура классов

Классы приложения разбиты на 3 основных слоя:

- слой доступа к данным, который состоит из классов моделей и репозиториев, для работы с данными;
- слой бизнес логики, состоящих из классов-сервисов, которые реализуют основную логику веб-приложения;
- слой контроллеров, которые, в ответ на действия пользователя, вызывают соответствующие методы слоя бизнес логки.

На рисунке 3.2 представлена схема взимодействия классов между слоем доступа к данным и слоем бизнес логики приложения.

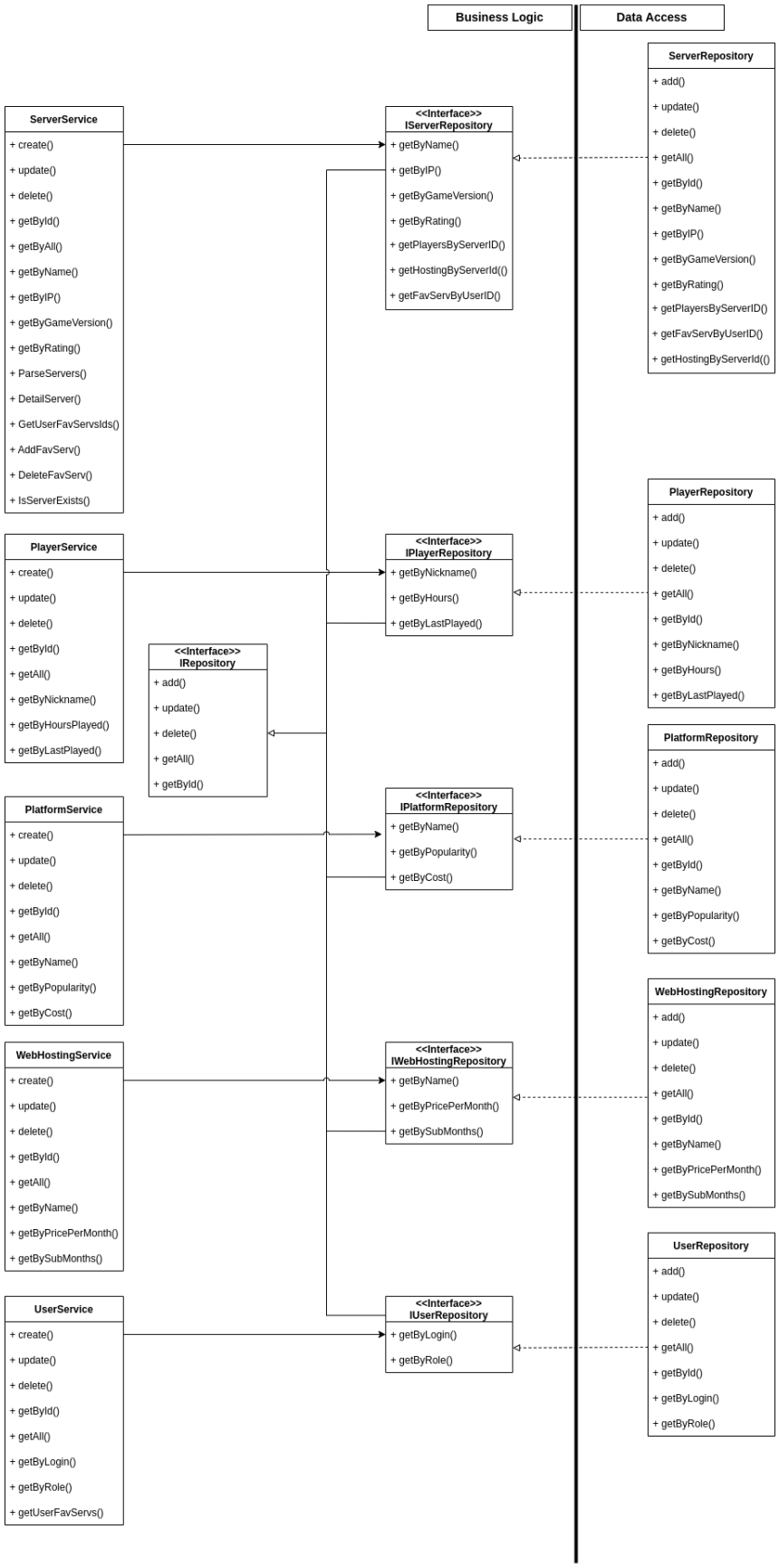

Рисунок 3.2 – Взаимодействи классов доступа к данным и бизнес логики

3.4 Реализация функций

При разработке были реализованы следующие функции базы данных, описанные в разделе 2.2.

1. Функция выбора серверов (листинг 3.1).

Листинг 3.1 — Функция выбора серверов

```
CREATE OR REPLACE FUNCTION getServers(isFavorite boolean, userId integer)
RETURNS TABLE (SrvId integer, SrvName varchar, SrvIp varchar, SrvGameVersion varchar, SrvRating integer, SrvHostingID integer, SrvPlatformID integer)
AS $$
  BEGIN
    IF (isFavorite is False) THEN
      return query
      SELECT *
      FROM public."Server";
    ELSE
      return query
      SELECT Fav."ServerID", Fav."Name", Fav."Ip", Fav."GameVersion", Fav."Rating", Fav."HostingID", Fav."PlatformID"
      FROM (public."Server" AS S JOIN public."FavoriteServer" AS FS ON S."Id" = FS."ServerID") AS Fav
      WHERE Fav."UserID" = userId;
    END IF;
  END;
$$ LANGUAGE plpgsql;
```

2. Функция фильтрации серверов (листинг 3.2).

Листинг 3.2 — Функция фильтрации серверов

```
CREATE OR REPLACE FUNCTION filterServers(nameParse varchar, platform_id integer, isFavorite boolean, userId integer)
RETURNS TABLE (FltId integer, FltName varchar, FltIp varchar, FltGameVersion varchar, FltRating integer, FltHostingID integer, FltPlatformID integer)
AS $$
  BEGIN
    IF (nameParse IS NULL) THEN
      nameParse = '';
    END IF;

    IF (platform_id > 0) THEN
```

Продолжение листинга 3.2

```
10        return query
11        SELECT *
12        FROM getServers(isFavorite, userId) as S
13        WHERE S.SrvPlatformID = platform_id AND S.SrvName LIKE '%' ||
    ↪  nameParse || '%';
14      ELSE
15        return query
16        SELECT *
17        FROM getServers(isFavorite, userId) as S
18        WHERE S.SrvName LIKE '%' || nameParse || '%';
19      END IF;
20    END;
21  $$ LANGUAGE plpgsql;
```

3. Функция сортировки серверов (листинг 3.3).

Листинг 3.3 — Функция сортировки серверов

```
1   CREATE OR REPLACE FUNCTION sortServers(nameParse varchar, platform_id
    ↪  integer, sortId integer, isFavorite boolean, userId integer)
2   RETURNS TABLE (Id integer, Name varchar, Ip varchar, GameVersion
    ↪  varchar, Rating integer, HostingID integer, PlatformID integer)
3   AS $$
4     BEGIN
5       IF (sortId = 0) THEN -- Name ASC
6         return query
7         SELECT *
8         FROM filterServers(nameParse, platform_id, isFavorite, userId)
    ↪  as S
9         ORDER BY S.FltName;
10      ELSEIF (sortId = 1) THEN -- Name DESC
11        return query
12        SELECT *
13        FROM filterServers(nameParse, platform_id, isFavorite, userId)
    ↪  as S
14        ORDER BY S.FltName DESC;
15      ELSEIF (sortId = 2) THEN -- Ip ASC
16        return query
17        SELECT *
18        FROM filterServers(nameParse, platform_id, isFavorite, userId)
    ↪  as S
19        ORDER BY S.FltIp;
20      ELSEIF (sortId = 3) THEN -- Ip DESC
21        return query
22        SELECT *
```

Продолжение листинга 3.3

```
23        FROM filterServers(nameParse, platform_id, isFavorite, userId)
    ↪   as S
24        ORDER BY S.FltIp DESC;
25      ELSEIF (sortId = 4) THEN -- GameVersion ASC
26        return query
27        SELECT *
28        FROM filterServers(nameParse, platform_id, isFavorite, userId)
    ↪   as S
29        ORDER BY S.FltGameVersion;
30      ELSEIF (sortId = 5) THEN -- GameVersion DESC
31        return query
32        SELECT *
33        FROM filterServers(nameParse, platform_id, isFavorite, userId)
    ↪   as S
34        ORDER BY S.FltGameVersion DESC;
35      ELSEIF (sortId = 6) THEN -- Rating ASC
36        return query
37        SELECT *
38        FROM filterServers(nameParse, platform_id, isFavorite, userId)
    ↪   as S
39        ORDER BY S.FltRating;
40      ELSEIF (sortId = 7) THEN -- Rating DESC
41        return query
42        SELECT *
43        FROM filterServers(nameParse, platform_id, isFavorite, userId)
    ↪   as S
44        ORDER BY S.FltRating DESC;
45      ELSEIF (sortId = 8) THEN -- Platform ASC
46        return query
47        SELECT PS.FltId, PS.FltName, PS.FltIp, PS.FltGameVersion,
    ↪   PS.FltRating, PS.FltHostingID, PS.FltPlatformID
48        FROM (filterServers(nameParse, platform_id, isFavorite, userId)
    ↪   as S JOIN public."Platform" as P on S.FltPlatformID = P."Id") as
    ↪   PS
49        ORDER BY PS."Name";
50      ELSEIF (sortId = 9) THEN -- Platform DESC
51        return query
52        SELECT PS.FltId, PS.FltName, PS.FltIp, PS.FltGameVersion,
    ↪   PS.FltRating, PS.FltHostingID, PS.FltPlatformID
53        FROM (filterServers(nameParse, platform_id, isFavorite, userId)
    ↪   as S JOIN public."Platform" as P on S.FltPlatformID = P."Id") as
    ↪   PS
54        ORDER BY PS."Name" DESC;
55      ELSE -- Default: Name ASC
56        return query
57        SELECT *
```

Продолжение листинга 3.3

```
58        FROM filterServers(nameParse, platform_id, isFavorite, userId)
    ↪   as S
59        ORDER BY S.FltName;
60      END IF;
61    END;
62  $$ LANGUAGE plpgsql;
```

4. Функция парсинга серверов (обертка) (листинг 3.4).

Листинг 3.4 — Функция парсинга серверов

```
1   CREATE OR REPLACE FUNCTION parseServers(nameParse varchar,
    ↪   platform_id integer, sortId integer, isFavorite boolean, userId
    ↪   integer)
2   RETURNS TABLE (Id integer, Name varchar, Ip varchar, GameVersion
    ↪   varchar, Rating integer, HostingID integer, PlatformID integer)
3   AS $$
4     BEGIN
5       return query
6       SELECT * FROM sortServers(nameParse, platform_id, sortId,
    ↪   isFavorite, userId);
7     END;
8   $$ LANGUAGE plpgsql;
```

3.5 Реализация триггеров

В проектируемой базе данных определены следующие триггеры, которые описаны в разделе 2.2.

1. Триггеры изменения рейтинга сервера и функция, реализующая работу этих триггеров (листинг 3.5).

Листинг 3.5 — Триггеры изменения рейтинга сервера

```
1   CREATE OR REPLACE FUNCTION changeServerRating() RETURNS trigger AS $$
2     BEGIN
3       IF (TG_OP = 'INSERT') THEN
4         UPDATE public."Server"
5         SET "Rating" = "Rating" + 1
6         WHERE "Id" = NEW."ServerID";
7
8         RETURN NEW;
9       ELSIF (TG_OP = 'DELETE') THEN
10        UPDATE public."Server"
11        SET "Rating" = "Rating" - 1
```

Продолжение листинга 3.5

```
      WHERE "Id" = OLD."ServerID";

      RETURN OLD;
    END IF;
  END;
$$ LANGUAGE plpgsql;


-- Триггер повышения рейтинга сервера
CREATE OR REPLACE TRIGGER incServerRatingTrigger AFTER INSERT ON
↪  public."FavoriteServer"
  FOR EACH ROW EXECUTE PROCEDURE changeServerRating();

-- Триггер понижения рейтинга сервера
CREATE OR REPLACE TRIGGER decServerRatingTrigger AFTER DELETE ON
↪  public."FavoriteServer"
  FOR EACH ROW EXECUTE PROCEDURE changeServerRating();
```

2. Триггер установки роли пользователя и функция, реализующая его работу (листинг 3.6).

Листинг 3.6 — Триггер установки роли пользователя

```
CREATE OR REPLACE FUNCTION setUserRole() RETURNS trigger AS $$
  BEGIN
    NEW."Role" := 'User';

    -- Если ник админ, то роль выдать админа
    IF (NEW."Login" = 'admin') THEN
      NEW."Role" := 'Admin';
    END IF;

    RETURN NEW;
  END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER setUserRoleTrigger BEFORE INSERT ON
↪  public."User"
  FOR EACH ROW EXECUTE PROCEDURE setUserRole();
```

3.6 Реализация ролевой модели

В базу данных были введены следующие роли, рассмотренные в разделе 2.4.

1. Неавторизованный пользователь. В листинге 3.7 представлено создание роли и раздача необходимых прав.

Листинг 3.7 — Неавторизованный пользователь

```
CREATE ROLE non_auth_user WITH
  LOGIN
  NOSUPERUSER
  NOCREATEDB
  NOCREATEROLE
  NOREPLICATION
  PASSWORD 'non_auth_user'
  CONNECTION LIMIT  -1;

-- Привелегии
GRANT SELECT ON public."Server" TO non_auth_user;
GRANT SELECT ON public."Platform" TO non_auth_user;
GRANT SELECT ON public."User" TO non_auth_user;

GRANT INSERT ON public."User" TO non_auth_user;
```

2. Авторизованный пользователь. В листинге 3.8 представлено создание роли и раздача необходимых прав.

Листинг 3.8 — Авторизованный пользователь

```
CREATE ROLE auth_user WITH
  LOGIN
  NOSUPERUSER
  NOCREATEDB
  NOCREATEROLE
  NOREPLICATION
  PASSWORD 'auth_user'
  CONNECTION LIMIT  -1;

-- Привелегии
GRANT SELECT ON public."Server" TO auth_user;
GRANT SELECT ON public."Platform" TO auth_user;
GRANT SELECT ON public."FavoriteServer" TO auth_user;
GRANT SELECT ON public."ServerPlayer" TO auth_user;
GRANT SELECT ON public."Player" TO auth_user;
GRANT SELECT ON public."User" TO auth_user;
GRANT SELECT ON public."WebHosting" TO auth_user;

GRANT INSERT ON public."FavoriteServer" TO auth_user;
GRANT INSERT ON public."User" TO auth_user;
```

Продолжение листинга 3.8

```
21
22  GRANT UPDATE ON public."Server" TO auth_user;
23
24  GRANT DELETE ON public."FavoriteServer" TO auth_user;
```

3. Администратор сайта. В листинге 3.9 представлено создание роли и раздача необходимых прав.

Листинг 3.9 — Администратор сайта

```
1   CREATE ROLE admin WITH
2     LOGIN
3     SUPERUSER
4     CREATEDB
5     CREATEROLE
6     NOREPLICATION
7     PASSWORD 'admin'
8     CONNECTION LIMIT  -1;
9
10  -- Привелегии
11  GRANT ALL PRIVILEGES ON ALL TABLES in schema public TO admin;
```

3.7 Выбор СУБД

Выбор системы управления базой даных является важной частью разработки программного продукта. Выбранная СУБД должна соответствовать всем требованиям, которые предъявляются при разработке. Рассмотрим наиболее популярные СУБД:

- MySQL;
- Microsoft SQL Server;
- Oracle;
- PostgresSQL.

3.7.1 MySQL

MySQL [**mysql**] – является одной из самых популярных СУБД. Распространяется бесплатно. Из-за высокого спроса, часто выходят новые обновления,

исправляющие ошибки и добавляющие много нового функционала. Не подходит для простых задач из-за трудной настройки. Не имеет ряда функционала, который в других СУБД реализован по умолчанию.

3.7.2 Microsoft SQL Server

Microsoft SQL Server [**sqlserver**] – собственная разработка компании Microsoft. Имеется поддержка для работы на операционной системе Linux, но наилучшую поддержку имеет для операционной системы Windows и иных продуктов компании Microsoft. Имеет проблемы с оптимизацией использования ресурсов. Проста в использовании.

3.7.3 Oracle

.

Oracle [**oracle**] – крайне популярна в крупных компаниях. Распространяется по подписке. Является одной из самых безопасных СУБД, так как каждая транзакция полностью изолирована друг от друга. Требует больших вычислительных ресурсов, тем самым не подходит для небольших проектов.

3.7.4 PostgresSQL

PostgresSQL [**postgres**] – одна из самых популярных СУБД, которые распространяются бесплатно и имеют высокое качество. Имеет хорошую оптимизацию, особенно под операционную систему Linux. Часто использует при разработке веб-приложений и хорошо подходит для небольших проектов. Имеет трудную настройку для неподготовленного пользователя.

Вывод

Выбор СУБД будет произведен, исходя из следующих критериев:

- К1 – подробная документация;
- К2 – высокий уровень оптимизации;
- К3 – подходит для разработки небольших проектов;
- К4 – поддержка различных форматов файлов;

- К5 – распространяется бесплатно.

Таблица 3.1 – Сравнение СУБД

Название	**К1**	**К2**	**К3**	**К4**	**К5**
«MySQL»	+	+	-	-	+
«Microsoft SQL Server»	-	-	+	+	-
«Oracle»	+	-	-	+	-
«PostgresSQL»	-	+	+	+	+

Исходя из того, что разрабатывается небольшое веб-приложение, то наилучшим образом подходит СУБД PostgresSQL, так как она обладает всеми необходимыми требованиями.

3.8 Полученный результат

...

Вывод

В данном разделе были рассмотрены средства, с помощью которых было реализовано ПО, описана структура классов проекта, а также представлен листинг алгоритма перемещения частицы водопада за кадр.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Официальный сайт Minecraft [Электронный ресурс]. — URL: https://www.minecraft.net/ru-ru (дата обращения: 5.05.2022).
2. Сервера Майнкрафт «MinecraftRating» [Электронный ресурс]. — URL: https://minecraftrating.ru/ (дата обращения: 5.05.2022).
3. Официальный сайт Counter-Strike [Электронный ресурс]. — URL: https://blog.counter-strike.net/ (дата обращения: 5.05.2022).
4. Сервера Counter-Strike «Сервера КС» [Электронный ресурс]. — URL: https://servera-csgo.ru/ (дата обращения: 5.05.2022).
5. Сервера Майнкрафт «Best Minecraft Servers» [Электронный ресурс]. — URL: https://best-minecraft-servers.co/ (дата обращения: 5.05.2022).
6. Сервера Counter-Strike «Game Tracker» [Электронный ресурс]. — URL: https://www.gametracker.com/search/cs/ (дата обращения: 5.05.2022).
7. Нотации модели сущность-связь (ER диаграммы) [Электронный ресурс]. — URL: https://pro-prof.com/archives/8126 (дата обращения: 5.05.2022).
8. Что такое база данных? [Электронный ресурс]. — URL: https://www.oracle.com/cis/database/what-is-database/ (дата обращения: 5.05.2022).
9. Дореляционные модели данных [Электронный ресурс]. — URL: https://spravochnick.ru/bazy_dannyh/dorelyacionnye_modeli_dannyh/ (дата обращения: 5.05.2022).
10. Что такое реляционная база данных? [Электронный ресурс]. — URL: https://aws.amazon.com/ru/relational-database (дата обращения: 5.05.2022).
11. Нереляционные данные и базы данных NoSQL [Электронный ресурс]. — URL: https://docs.microsoft.com/ru-ru/azure/architecture/data-guide/big-data/non-relational-data (дата обращения: 5.05.2022).

12. SQL [Электронный ресурс]. — URL: https://blog.skillfactory.ru/glossary/sql/ (дата обращения: 5.05.2022).