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

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

# РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

## *К КУРСОВОЙ РАБОТЕ*

## *НА ТЕМУ:*

«Создание информационной системы настольных игр и игротек»

Студент: ИУ7-63Б _____ М. Д. Маслова
(группа) (подпись, дата) (И. О. Фамилия)

Руководитель: _____ О. В. Кузнецова
(подпись, дата) (И. О. Фамилия)

*2022 г.*

# РЕФЕРАТ

Расчетно-пояснительная записка 49 с., 13 рис., 5 табл., 29 источн., 0 прил.

Ключевые слова: база данных, система управления базами данных, PostgeSQL, настольные игры, игротеки.

В данной работе проведено проектирование и разработка базы данных настольных игр и игротек.

В разделе 1 проведен анализ существующих решений, выявлен их основной недостаток: отсутствие возможности для игроков регистрироваться на игротеки разных организаторов через единую систему. Также проведена формализация задачи, данных и типов пользователей. В качестве модели базы данных выбрана релияционная.

В разделе 2 проведено проектирование базы данных: необходимые таблицы, хранимые процедуры и функции, триггеры и роли, — а также проведно проектирование приложения.

В разделе 3 в качестве используемой СУБД выбрана PostgreSQL, а в качестве средств реалзиации выбраны: C#, Entity Framework, Blazor и Visual Studio. Также приведены детали реализации и примеры работы программы.

В разделе 4 проведено исследование зависимости времени выполнения запроса от индексации при различном количестве записей.

К разделам сделаны выводы, в заключении подведены итоги.

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Настольные игры — это игры, основанные на манипуляции инвентарем (карточками, кубиками, фишками и т. п.), размещаемым на обычном или специально сделанном столе [1].

На сегодняшний день рынок настольных игр представляет собой одну из наиболее молодых и стремительно развивающихся областей мировой экономики [2]. Это связно с тем, что несмотря на современные условия информатизации настольные игры набирают популярность и продолжают сохранять востребованность в широкой среде потребителей.

Отечественный рынок настольных игр молод, но стремительно усиливает свои позиции: за счет сегментации рынка по возрасту и категориям (семейные, стратегические, для вечеринок и др.) настольные игры отделились от рынка детских товаров и стали самостоятельным сектором экономики.

В отличие от других рынков прямая реклама настольных игр работает не эффективно. При этом наилучший способ прорекламировать игру — посадить человека играть. Поэтому в сфере настольных игры приобрели популярность игротеки — мероприятия, на котором гости могут поиграть в различные игры, завести знакомства, опробовать новинки или ознакомиться с ассортиментом [3].

В силу молодости рынка программные решения в сфере настольных игр, как будет показано в одном из разделов данной работы, по большей части представлены в виде интернет-магазинов, в которых информация об игротеках, регистрации на них, получения информации об проводимых играх либо представляется текстовым списком на отдельной странице, либо появляется в новостном разделе, либо вовсе отсутствует. Поэтому поиск необходимых игротек и регистрация на них становятся проблематичными.

Таким образом, **целью данной работы** является проектирование и реализация базы данных настольных игр и игротек, а также разработка приложения доступа к базе данных с возможностью просмотра, добавления, удаления и редактирования информации о настольных играх и игротеках, регистрации на игротеки и составления списка предпочитаемых игр.

Для достижения поставленной цели необходимо решить следующие **задачи**:

– провести анализ предметной области;

– формализовать задачу;
– провести анализ баз данных и систем управления базами данных;
– спроектировать базу данных и архитектуру приложения;
– реализовать базу данных и приложение для доступа к ней;
– провести эксперимент по сравнению времени выполнения запросов с использованием индексов и без них.

# 1 Аналитическая часть

В данном разделе проводится анализ предметной области: на основе рассмотрения существующих решений делаются выводы о представлении информации о настольных играх и игротеках, а также о необходимых возможностях пользователей. На базе анализа предметной области формулируются требования к информационной системе, формализуются данные и типы пользователей. Также проводится сравнение существующих моделей баз данных и систем управления базами данных, из которых выбираются оптимальные для решения поставленной задачи.

## 1.1 Анализ существующих аналогов

В силу молодости рынка настольных игр анализ предметной области проводится на основе рассмотрения программных решений, представляющих интернет-магазины крупных компаний настольных игр или тематических сайтов.

Так как рынки настольных игр в России и за рубежом имеют разные уровни развития существующие программные решения рассматриваются отдельно для российского и зарубежного рынков.

### 1.1.1 Российский рынок

На российском рынке программные решения в сфере настольных игр представлены интернет-магазинами крупных продавцов: Hobby Games [4], «Мосигра» [5], «Низа Гамс» [6] и др. Их веб-сайты ориентированы именно на продажу игр, в то время как анонсы игротек являются второстепенными и располагаются либо в специализированном разделе сайта, либо в новостях, при этом регистрация проходит либо через сторонний сервис, либо через сообщения в социальных сетях.

### 1.1.2 Зарубежный рынок

На зарубежном рынке преобладают тематические сайты. Например, BoardGameGeek [7] — многофункциональный сайт для любителей настольных игр, — содержит информацию о многочисленных настольных играх, их авторах, предоставляет ссылки на интернет-магазины, в которых можно купить

игру, а также возможность продать свою игру и пообщаться на форуме. Данный сайт также содержит страницу с актуальными данными по игровым конференциям всего мира, на которых проводятся игротеки. Аналогичными проектами, но с меньшими наборами функций являются Top Tier Board Games [8] и Board Game Halv [9], они представляют информацию о настольных играх и игротеках в формате новостей.

Существует также программное решение, предоставляющее возможность играть в популярные настольные игры онлайн — Board Game Arena [10]. Здесь реализованы виртуальные копии многих настольных игр, ведутся рейтинги по каждой игре и по всем играм в целом. Благодаря возможности перевода сайта пользователями, он доступен на нескольких языках, в том числе и на русском.

### 1.1.3 Сравнение существующих решений

Для сравнения существующих решений можно выделить следующие критерии:

- **К1** — наличие информации о настольных играх;
- **К2** — наличие информации об игротеках;
- **К3** — наличие информации об игротеках различных организаторов;
- **К4** — возможность регистрации на игротеку;
- **К5** — ведение рейтингов игроков;
- **К6** — поддержка русского языка.

Сравнение решений по данным критериям представлено в таблице 1.1.

Таблица 1.1 – Сравнение существующих решений

| **Решение** | **К1** | **К2** | **К3** | **К4** | **К5** | **К6** |
|---|---|---|---|---|---|---|
| Hobby Games | + | + | - | через сторонний ресурс | - | + |
| Мосигра | + | - | - | нет | - | + |
| Низа Гамс | + | + | - | через сторонний ресурс | - | + |
| BoardGameGeek | + | + | + | через сторонний ресурс | - | - |
| Board Game Arena | + | - | - | нет | + | + |

Данные из приведенной таблицы показывают, что все описанные существующие решения предоставляют пользователю возможность просмотра информации о настольных играх, однако при этом возможноть работы с игротеками в них либо вовсе отсутствует, либо предоставляется через сторонний ресурс. Также стоит отметить, что только одно решения предоставляет возможность работы с игротеками разных организаторов и только одно возможность отслеживания рейтинга игрока.

### Вывод

Таким образом, программные решения для поиска информации о настольных играх широко распространены, однако решение задачи поиска необходимой игротеки для игрока требует многих усилий. Если организаторы игротек, используя свои сайты, имеют возможность в любом удобном для них формате выкладывать информацию об игротеках, то для поиска нужной игротеки игроку придется посетить многочисленные сайты организаторов, найти раздел или страницу игротек на каждом из них и зарегистрироваться через сторонний ресурс.

То есть встает необходимость реализации информационной системы, в которой информация о всех игротеках будет собрана в одном месте, а регистрация будет происходить единообразно в самой системе. При этом в данной системе должна содержатся информация о настольных играх, так как на их основе происходит выбор игротеки. А в силу того, что реализация регистрации на игротеки, потребует регистрации в системе, как организаторов, так и игроков, появится возможность включения в систему рейтинговых списков.

## 1.2 Формализация задачи

Для создания информационной системы необходимо формализовать процесс проведения игротеки. Для этого выделим необходимые и возможные действия пользователей, которые будут соответствовать требованиям, предъявляемым к создаваемому программному обеспечению.

И так, при проведении игротеки сначала организатор:

1) выбирает время и место проведения игротеки, а также настольные игры, по которым она проводится;
2) определяет другие характеристики игротеки (название, продолжитель-

ность, стоимость участия, возможность купить игры);

3) регистрирует игротеку.

Далее игрок — участник игротеки:

1) просматривает список настольных игр;
2) выбирает понравившиеся игры, добавляя их в свой список избранных;
3) выбирает игротеку по понравившимся играм или из всего списка зарегистрированных игротек;
4) регистрируется на игротеку.

При этом игрок может пропустить этапы просмотра и выбора настольных игр и сразу перейти к выбору игротеки.

Описанные выше действия субъектов игротеки соответствуют необходимым функциям, которые являются обязательными требованиями к разрабатываемой информационной системе и должны быть реализованы в первую очередь. Также различия в возможных действиях пользователей формируют еще одно обязательное требование: регистрацию различных типов пользователей в системе.

Также у организатора должна быть возможность:

- просмотреть список участников, зарегистрированных на игротеку;
- отменить игротеку;
- перенести игротеку (изменить место проведения или время начала);
- изменить другую информацию о ней;
- сформировать заявку на добавление в базу места проведения или настольной игры при их отсутствии.

Игрок, в свою очередь, должен иметь возможность:

- удалить игру из списка понравившихся игр;
- отменить регистрацию на игротеку.

Данные требования являются желательными и реализуются сразу после обязательных.

Возможными требования к реализуемой системе являются:

- подтверждение организатором посещения игроком игротеки;
- изменение организатором рейтинга игрока после посещения им игротеки;

– уменьшение рейтинга игрока вследствие пропуска игротеки, на которую он был зарегистрирован;
– оценка игроком посещенной игротеки и организатора.

Также необходимо учесть ряд правил делового регламента:

– организатор не может зарегистрировать игротеку, дата и время начала которой прошли или соответствуют текущему дню;
– игротека считается прошедшей, если наступила дата проведения и время ее окончания;
– игрок не может зарегистрироваться на прошедшую или отмененную игротеку;
– игрок может зарегистрироваться на игротеку не позднее, чем за установленное организатором время до ее начала;
– при отмене игротеки список зарегистрированных на нее участников сохраняется;
– регистрация игротеки считается незавершенной до тех пор, пока не будут приняты заявки организатора на добавление в базу места проведения и хотя бы одной игры, если такие оформлялись.

Таким образом, необходимо разработать приложение, отвечающее выше перечисленным требованиям. В силу необходимости хранения информации о различных объектах предметной области также требуется спроектировать и реализовать базу данных, которая должна хранить информацию о настольных играх и игротеках, и взаимодействие с которой должно осуществляться с помощью интерфейса разрабатываемого приложения.

## 1.3 Описание типов пользователей

В соответствии с поставленными требованиями необходимо выделить 4 типа пользователей, описание которых представлено в таблице 4.1.

Таблица 1.2 – Описание типов пользователей

| **Тип** | **Функциональность** |
|---|---|
| Гость (неавторизованный пользователь) | – Просмотр списка настольных игр и игротек<br>– Поиск игротек по настольной игре<br>– Авторизация<br>– Регистрация |
| Игрок (авторизованный пользователь) | – Просмотр списка настольных игр и игротек<br>– Составление списка предпочитаемых игр<br>– Поиск игротек по настольным играм<br>– Регистрация на игротеки и ее отмена |
| Организатор (авторизованный пользователь) | – Просмотр списка настольных игр, игротек, мест проведения<br>– Регистрация новых игротек и их отмена<br>– Изменение информации о зарегистрированных игротеках<br>– Просмотр списка зарегистрированных на игротеку участников<br>– Формирование заявки на добавление нового места проведения при его отсутствии в базе<br>– Формирование заявки на добавление игры при ее отсутствии в базе |
| Администратор (авторизованный пользователь с повышенным уровнем полномочий) | – Просмотр списка настольных игр, игротек, мест проведения, организаторов, пользователей<br>– Изменение информации о настольных играх, местах проведения, организаторах, игротеках<br>– Одобрение или отклонение заявок от организаторов<br>– Изменение прав доступа пользователей<br>– Удаление пользователей |

Диаграммы вариантов использования для Гостя, Игрока, Организатора и Администратора представлены на рисунках 1.1, 1.2, 1.3, 1.4 соответственно.

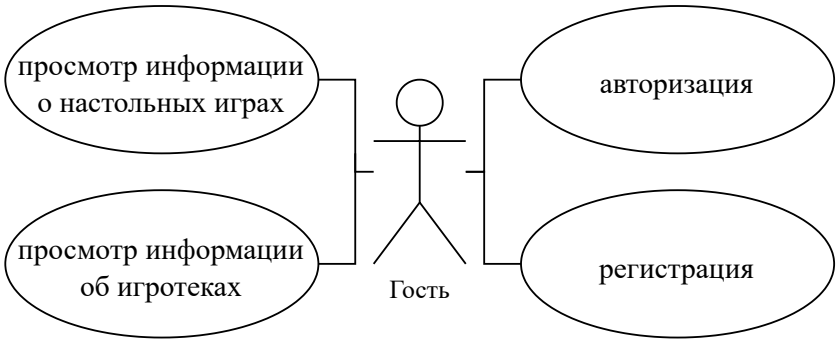


Рисунок 1.1 – Use-Case диаграмма для Гостя

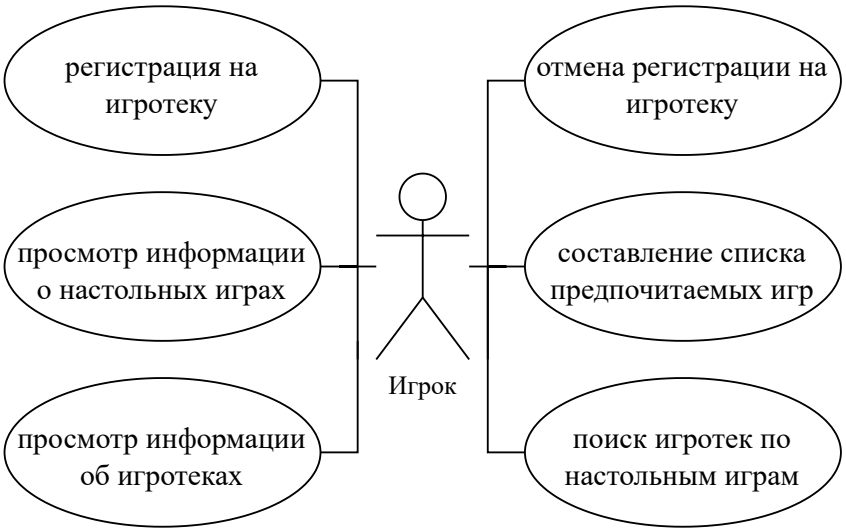


Рисунок 1.2 – Use-Case диаграмма для Игрока

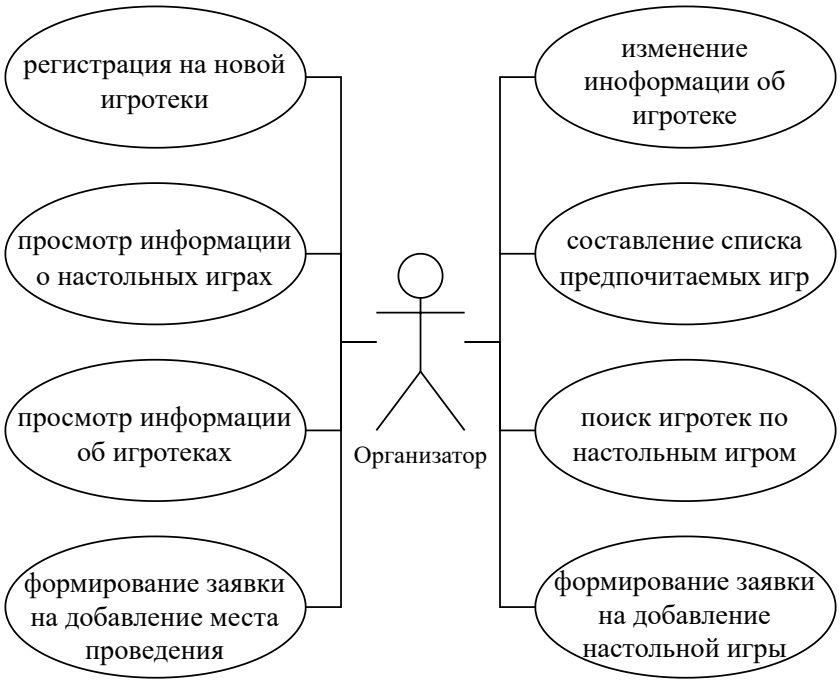


Рисунок 1.3 – Use-Case диаграмма для Организатора

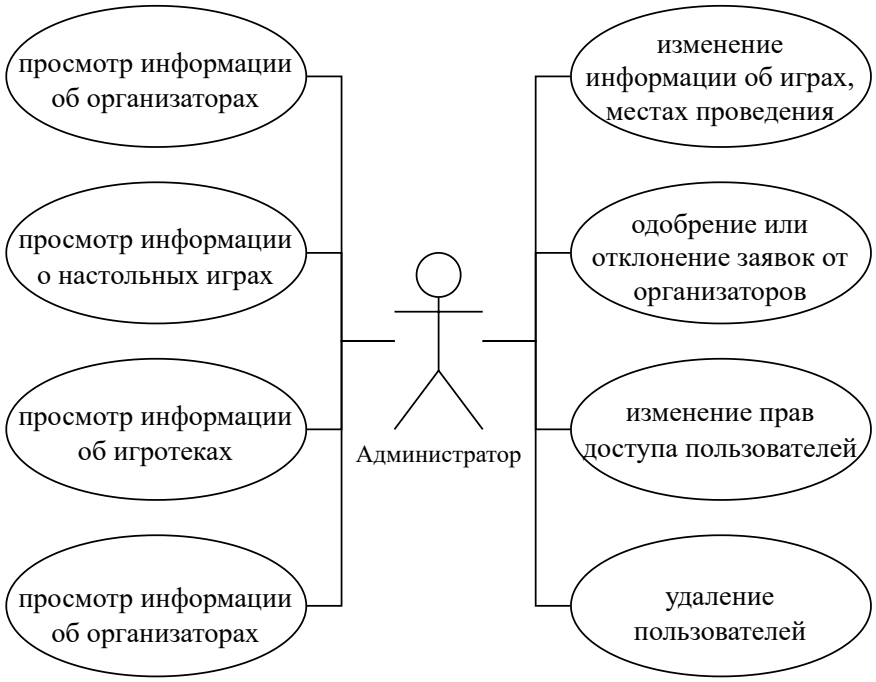


Рисунок 1.4 – Use-Case диаграмма для Администратора

## 1.4 Формализация данных

На основании сформулированных требований база данных должна содержать информацию о:

- игротеках;
- настольных играх, по которым проводятся игротеки;
- организаторах;
- местах проведения;
- игроках;
- пользователях.

Последние четыре сущности характерны для многих предметных областей, поэтому для них приведено только краткое описание (таблица 1.3), а подробный анализ приводится для первых двух сущностей, являющихся основными в рассматриваемой предметной области (их краткое описание также представлено в таблице 1.3).

Таблица 1.3 – Категории данных и сведения о них

| **Категория** | **Сведения** | |
|---|---|---|
| Игротека | ID<br>Название<br>Организатор<br>Время начала<br>Дата проведения | Возможность покупки игр<br>Место проведения<br>Продолжительность<br>Стоимость участия<br>Время регистрации |
| Организатор | ID<br>Название<br>Адрес | Сайт<br>Email<br>Телефон |
| Место проведения | ID<br>Адрес<br>Сайт<br>Название | Тип<br>Email<br>Телефон |
| Настольная игра | ID<br>Название<br>Производитель<br>Год выпуска<br>Мин. число игроков | Макс. число игроков<br>Мин. возраст<br>Макс. возраст<br>Мин. время игры<br>Макс. время игры |
| Игрок | ID<br>Имя | Лига<br>Рейтинг |
| Пользователь | ID<br>Роль | Логин<br>Пароль |

На основе анализа существующих решений можно выявить основные сведения о настольных играх и игротеках, которые используются при их описании. Так, для описания настольной игры используют следующие характеристики:

- название;
- производитель;
- год выпуска;
- возможное количество игроков в формате «от ... до ...»;
- время игры также в формате «от ... до ...»;
- рекомендованный возраст участников в формате возрастных категорий («6+», «12+», «18+» и т. п.) или диапазона возрастов («от ... до ...»);

Для описания игротеки в рассмотренных решениях используются:

- название;

– адрес места проведения;
– время начала и продолжительность;
– стоимость участия;
– ссылка на регистрацию.

Так как регистрация на игротеку будет происходить в самой системе, ссылка на регистрацию, как сведение об игротеке храниться не будет, но в соответствии с сформулированными требованиями будет добавлено время, за которое можно зарегистрироваться на игротеку.

Связи между описанными сущностями представлены на ER-диаграмме (рисунок 1.5).

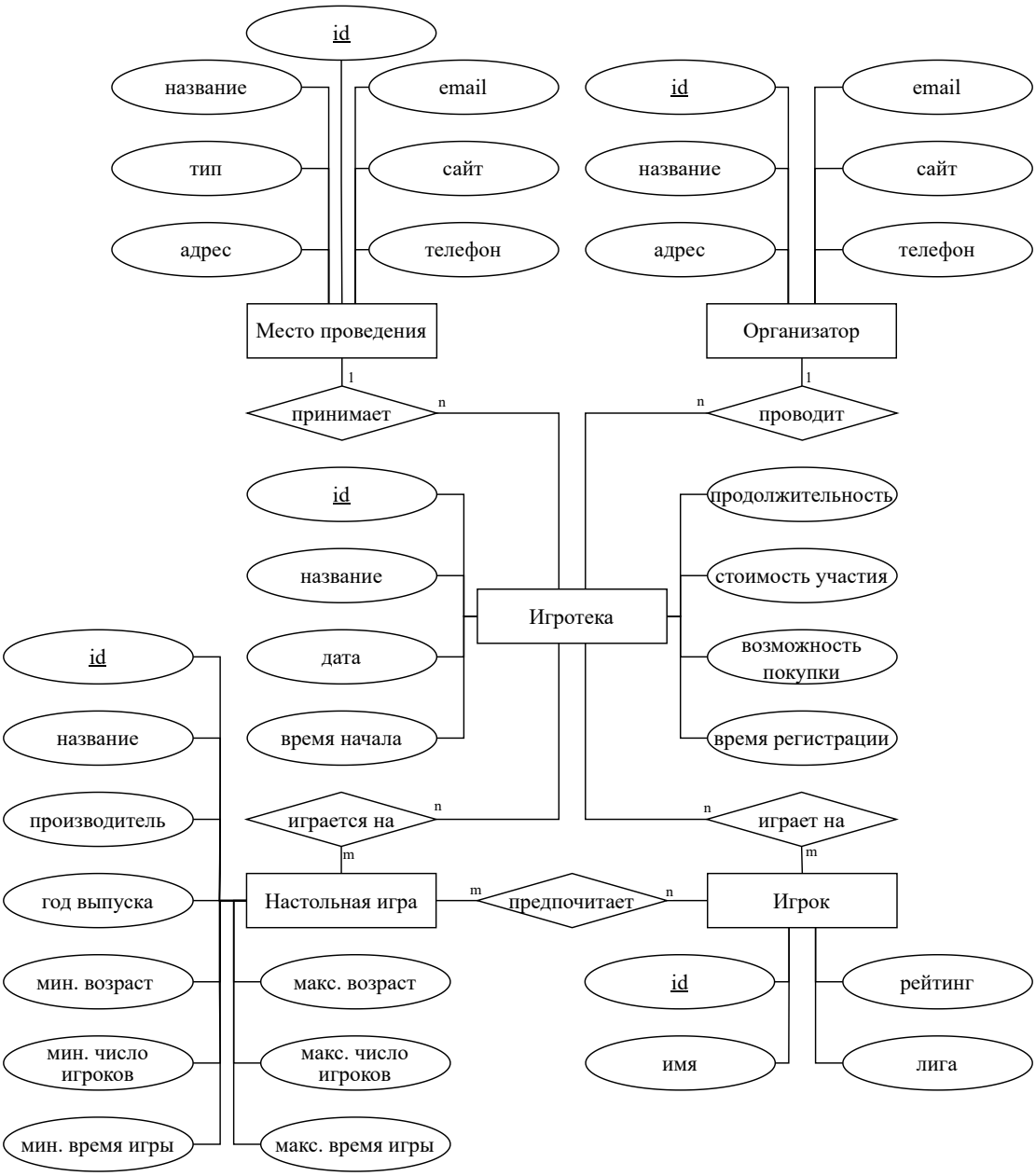


Рисунок 1.5 – ER-диаграмма

## 1.5 Выбор модели базы данных

На основании сформулированных требований система должна включать базу данных для хранения информации обо всех сущностях, описанных в подразделе 1.4.

База данных (БД) — это совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимая от прикладных программ [11].

Основой любой базы данных является модель данных — формализованное описание структур единиц информации и операций над ними в информационной системе. Модель данных определяет логическую структуру базы данных, то есть способ хранения, организации и обработки информации [12].

Существуют различные модели данных, каждая из которых имеет свои достоинства и недостатки, поэтому необходимо провести анализ моделей и выбрать модель базы данных, соответствующая модель данных которой наиболее полно удовлетворяет требованиям реализуемой системы.

Модели баз данных можно разделить на три основные категории: дореляционные, реляционные и постреляционные.

### 1.5.1 Дореляционные базы данных

К дореляционным моделям [13] относят модели, предшедствующие реляционной: иерархическую, сетевую и модель на основе инвертированных списков.

**Иерархическая модель** — представление базы данных в виде древовидной структуры, состоящей из объектов разных уровней, где связи между записями выражаются в виде отношений предок/потомок, а у каждой записи есть ровно одна родительская запись.

Достоинствами иерархической модели являются:

- простота модели (схожая, например, со структурой компании или генеалогическим деревом);
- быстродействие за счет реализации отношения предок/потомок в виде физических указателей.

Недостатками иерархической модели являются:

- необходимость дублирования деревьев для связи многие-ко-многим;

– отсутствие ссылок между записями, не входящими в одну иерархию и, как следствие, невозможность хранения в базе данных порожденного узла без соответсвующего исходного.

Для устранения недостатков иехрархической модели была разработана **сетевая модель**, в которой одна запись может участвовать в нескольких отношениях предок/потомок.

К достоинствам сетевой модели можно отнести:

– уменьшение дублирования данных по сравнению с иерархической моделью;
– гибкость за счет возможности создания дополнительных связей.

Недостатками сетевой модели являются:

– сложность и запутанной структуры;
– ослабление контроля целостности вследствие допустимости установления произвольных связей между записями.

В силу сложности практического использования иерархической и сетевой модели появилась **модель на основе инвертированных списках**. Организация доступа к данным на основе инверитированных списков используется в современных реляционных системах, однако в них пользователи не имеют непосредственного доступа к инвертированным спискам.

Недостатками такой модели являются:

– отсутствие строгого математического аппарата;
– отсутствие средств для описания ограничений целостности базы данных;
– большая трудоемкость программирования запросов к базе данных.

Общим недостатком дореляционных баз данных является поддержка доступа к данным только путем создания прикладных программ и, как следствие, оптимизация доступа к данным пользователем без какой-либо поддержки системы.

### 1.5.2 Реляционные базы данных

Реляционные базы данных основаны на реляционной модели, представляющей сущности и связи между ними представлены в виде отношений —

двумерных таблиц, каждая строка (кортеж) является записью, содержащей данные о конкретном объекте данной сущности, а столбцы ее свойствам, или атрибутам. То есть реляционная база данных — это совокупность отношений, содержащих всю информацию, которая должна храниться в базе данных.

В каждом отношении выделяют первичный ключ — атрибут или набор атрибутов, однозначно идентифицирующий каждый из кортежей отношения. В этом же или другом отношении может быть создан столбец (их набор), ссылающийся на первичный ключ — внешний ключ, с помощью которого реализуются связи между таблицами базы данных [13].

При этом для поддержания целостности данных в реляционных БД соблюдаются требованиям ACID:

- атомарность (atomicity) — для каждой транзакции либо выполняются все операции внутри нее, либо не выполняется ни одной, то есть транзакции являются атомарными и работает принцип «все или ничего»;
- согласованность (consistency) — выполнение транзакции не может перевести систему в несогласованное состояние, то есть база данных всегда остается в согласованном состоянии;
- изолированность (isolation) — на результат транзакции не влияют другие транзакции, которые происходят параллельно с ней;
- долговечность (durabiliry) — любые изменения сохраняются в базе данных несмотря на сбои и действия пользователей.

Реляционные базы данных поддерживают SQL — язык структурированных запросов для определения и обработки данных. SQL является одним из наиболее гибких и стандартизированных языков запросов, что позволяет минимизировать ряд рисков для разработчиков [14].

Достоинствами реляционных баз данных являются:

- единый способ представления сущностей и связей между ними: все — отношение;
- соответствие требованиям ACID;
- использование стандартизированного языка запросов SQL.

Недостатками реляционных баз данных являются:

- вертикальная масштабируемость;
- отсутствие возможности представить некоторые данные в виде отно-

шения;
– увеличение времени работы при увеличении числа отношений (например, при нормализации);
– трудозатраты на проектирование базы данных.

### 1.5.3 Постреляционные базы данных

Постреляционные базы данных — базы данных с динамическим представлением структуры данных [14]. В таких базах данных используются гибкие модели данных, которые определяют следующие типы баз данных [15]:

– колоночные БД хранят данные по столбцам, число которых от строки к строке может изменяться;
– хранилища «ключ-значение» для хранения объектов используют хеш-таблицу, в которой находятся пары из уникального ключа и указателя на конкретный объект данных;
– документноориентированные БД хранят данные в виде коллекций документов (например, в форматах JSON, XML и др.), причем каждая запись может содержать различные по количеству и типам данные.
– графовые БД используются для хранения данных с большим числом связей и представляют элементы базы данных в виде вершин графа, а отношения между ними — в виде ребер между соответствующими вершинами.

Достоинствами постреляционных баз данных являются:

– горизонтальная масштабируемость;
– гибкость моделей данных;
– возможность хранения неструктурированной информации.

Недостатками постреляционных баз данных являются:

– смягчение требований ACID;
– привязанность приложения к конкретной СУБД в силу собственных языков запросов (отсутствие поддержки SQL).

### Вывод

В разрабатываемой базе:

– данные являются структурированными, причем их структура не под-

вержена частым изменениям;
- необходима поддержка целостности данных при наличии нескольких пользователей, то есть соответствие требованиям к транзакционным системам;
- необходимо выполнение сложных запросов.

Таким образом, в соответствии с перечисленными требованиями к базе данных и свойствами основных категорий моделей данных необходимо выбрать реляционную базу данных.

## Вывод

В данном разделе был проведен анализ предметной области настольных игр и игротек с помощью исследования существующих решений. На основе анализа предметной области была формализована задача и данные и описаны типы пользователей. Также по результатам сравнения моделей баз данных для решения задачи была выбрана реляционная модель данных.

# 2 Конструкторская часть

## 2.1 Проектирование базы данных

### 2.1.1 Таблицы базы данных

Реализуемая база данных должна включать десять таблиц: шесть из них соответсвуют сущностям, описанным в подразделе 1.4, а оставшиеся четыре реализуют связь между сущностями. Диаграмма базы данных представлена на рисунке 2.1.

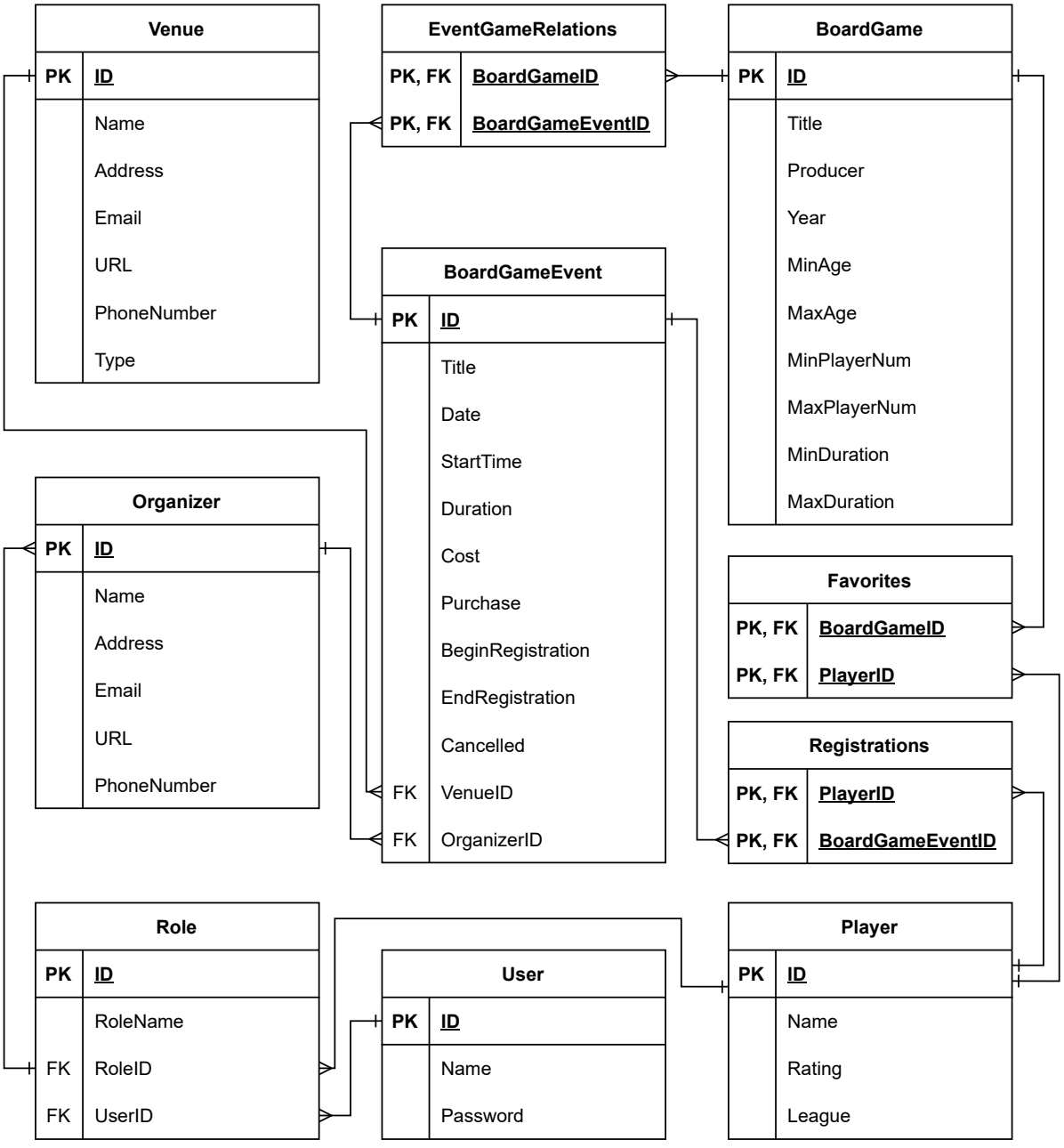


Рисунок 2.1 – Диаграмма базы данных

Приведем, подробное описание каждой таблицы.

1. Таблица BoardGameEvent содержит информацию об игротеках и включает следующие поля:
   - ID — первичный ключ, целочисленный тип;
   - Title — название, строковый тип;
   - Date — дата проведения, тип даты;
   - StartTime — время начала, тип времени;
   - Duration — продолжительность в минутах, целочисленный беззнаковый тип;
   - Cost — стоимость посещения, целочисленный беззнаковый тип;
   - Purchase — возможность покупки игр, логический тип;
   - BeginRegistration — время начала регистрации, тип даты-времени (timestamp);
   - EndRegistration — время окончания регистрации, тип даты-времени (timestamp);
   - Cancelled — состояние отмены, логический тип;
   - VenueID — ID места проведения, внешний ключ;
   - OrganizerID — ID организатора, внешний ключ.
2. Таблица BoardGame содержит информацию о настольных играх и включает следующие поля:
   - ID — первичный ключ, целочисленный тип;
   - Title — название, строковый тип;
   - Producer — издатель, строковый тип;
   - Year — год издания, целочисленный беззнаковый тип;
   - MinAge — минимальный возраст игроков, целочисленный беззнаковый тип;
   - MaxAge — максимальный возраст игроков, целочисленный беззнаковый тип;
   - MinPlayerNum — минимальное число игроков, целочисленный беззнаковый тип;
   - MaxPlayerNum — максимальное число игроков, целочисленный беззнаковый тип;
   - MinDuration — минимальное время партии в минутах, целочис-

ленный беззнаковый тип;
   - MaxDuration — максимальное время партии, целочисленный беззнаковый тип.

3. Таблица Venue содержит информацию о местах проведения и включает следующие поля:
   - ID — первичный ключ, целочисленный тип;
   - Name — название, строковый тип;
   - Address — адрес, строковый тип;
   - Email — адрес электронной почты, строковый тип;
   - URL — интернет-адрес, строковый тип;
   - PhoneNumber — номер телефона, строковый тип;
   - Type — тип места проведения, строковый тип.
4. Таблица Organzer содержит информацию об организаторах и включает следующие поля:
   - ID — первичный ключ, целочисленный тип;
   - Name — название, строковый тип;
   - Address — адрес, строковый тип;
   - Email — адрес электронной почты, строковый тип;
   - URL — интернет-адрес, строковый тип;
   - PhoneNumber — номер телефона, строковый тип;
5. Таблица Player содержит информацию об игроках и включает следующие поля:
   - ID — первичный ключ, целочисленный тип;
   - Name — имя игрока, строковый тип;
   - Rating — рейтинг, целочисленный тип;
   - League — лига игрока, строковый тип.
6. Таблица User содержит информацию об пользователях и включает следующие поля:
   - ID — первичный ключ, целочисленный тип;
   - Name — имя пользователя, строковый тип;
   - Password — хеш пароля, строковый тип.
7. Таблица Role содержит информацию о ролях пользователей и включает

следующие поля:

- ID — первичный ключ, целочисленный тип;
- RoleName — название роли («guest», «player», «organizer», «admin»), строковый тип;
- RoleID — ID в таблице игроков или организаторов, внешний ключ;
- UserID — ID пользователя, внешний ключ.

8. Таблица EventGameRelations реализует связь многие-ко-многим таблиц BoardGameEvent и BoardGame и включает следующие поля:
   - BoardGameID — ID настольной игры, часть составного первичного ключа, внешний ключ;
   - BoardGameEventID — ID игротеки, часть составного первичного ключа, внешний ключ.
9. Таблица Favorites реализует связь многие-ко-многим таблиц Player и BoardGame и включает следующие поля:
   - BoardGameID — ID настольной игры, часть составного первичного ключа, внешний ключ;
   - PlayerID — ID игрока, часть составного первичного ключа, внешний ключ.
10. Таблица Registrations реализует связь многие-ко-многим таблиц BoardGameEvent и Player и включает следующие поля:
    - PlayerID — ID игрока, часть составного первичного ключа, внешний ключ;
    - BoardGameEventID — ID игротеки, часть составного первичного ключа, внешний ключ.

### 2.1.2 Хранимые процедуры и функции базы данных

Для выполнения правил делового регламента в базе данных должны быть определены следующие хранимые процедуры и функции:

- функция, которая реализует алгоритм получения состояния игротеки, представленный на рисунке 2.2;
- процедура обновления таблицы-связи игр и игротек, реализующая

алгоритм, представленный на рисунке 2.3.

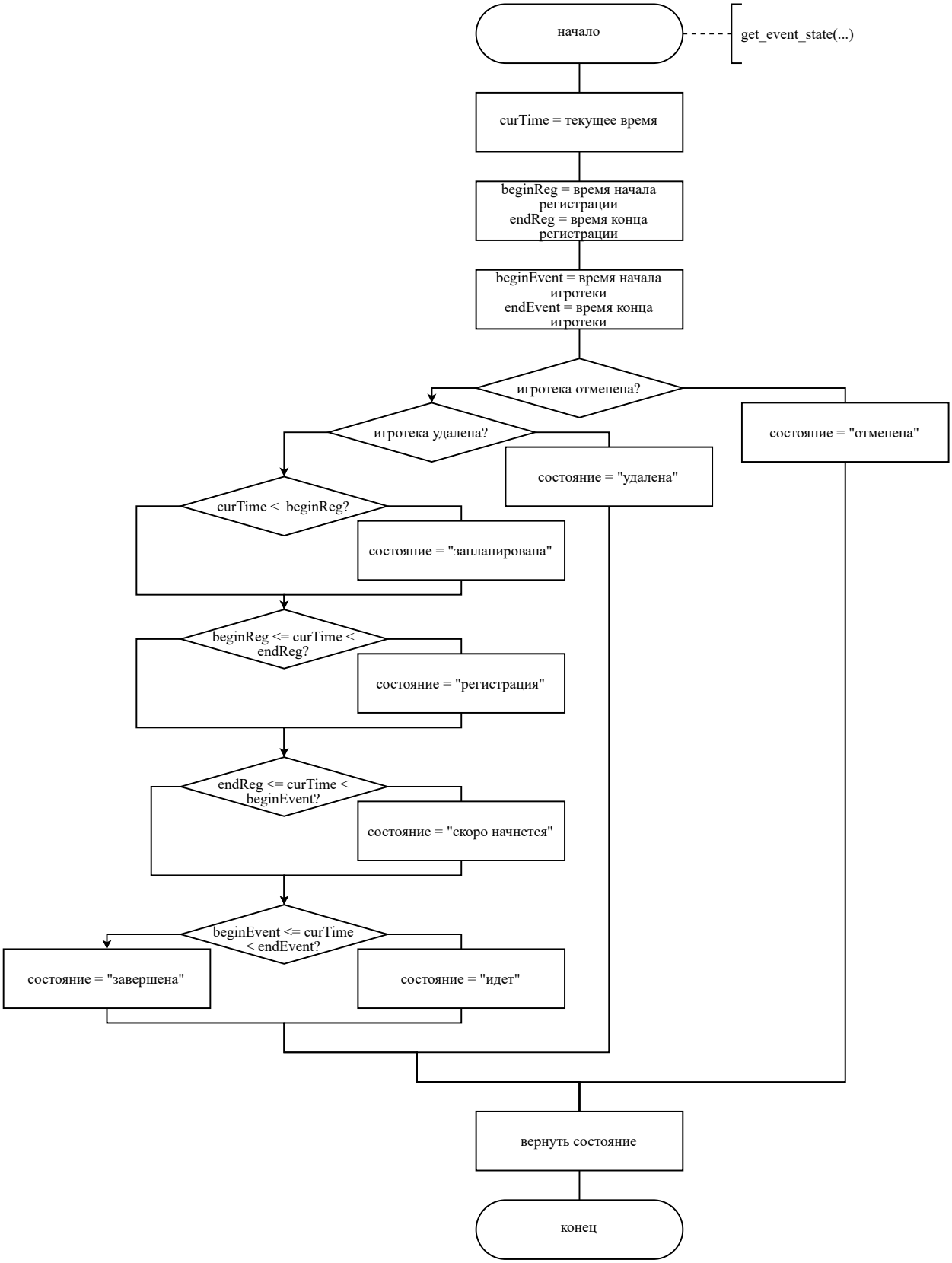


Рисунок 2.2 – Алгоритм получения состояния игротеки

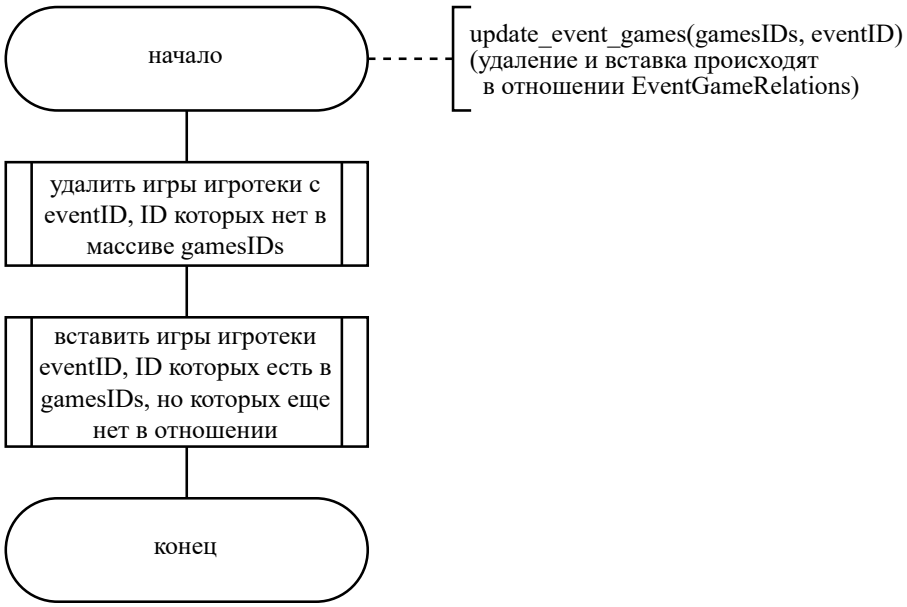


Рисунок 2.3 – Алгоритм обновления таблицы-связи игр и игротек

### 2.1.3 Триггеры базы данных

Для запрета игроку добавления своей регистрации на игротеку, которая уже завершилась, в базе данных должен быть определен «BEFORE INSERT» триггер, алгоритм работы которого представлен на рисунке 2.4.

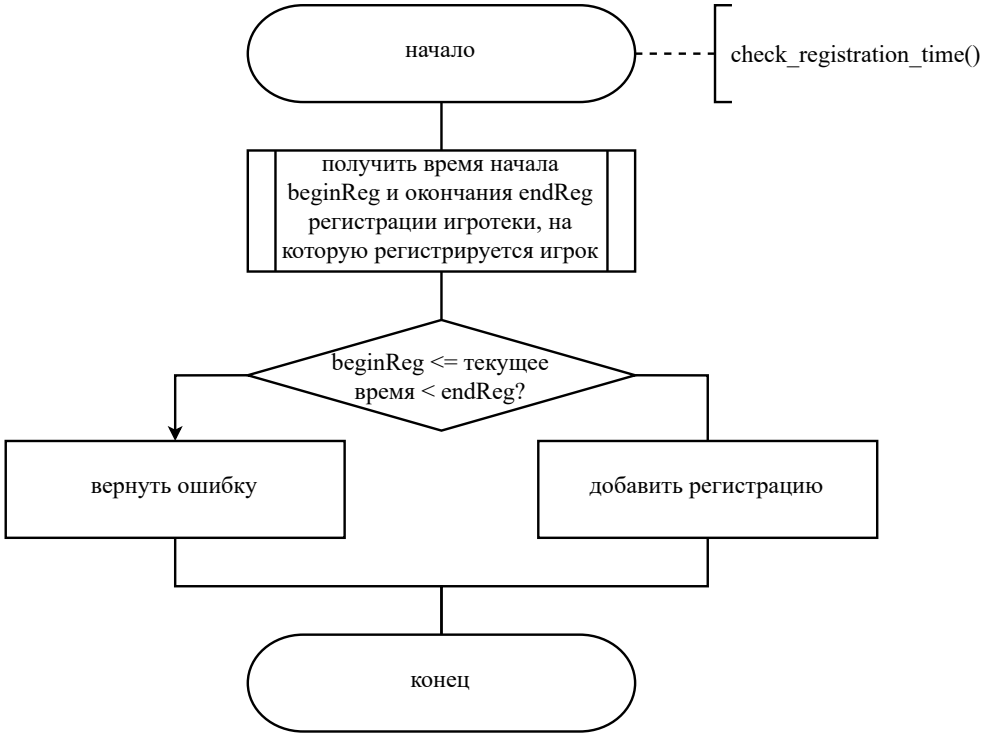


Рисунок 2.4 – Алгоритм проверки времени регистрации

### 2.1.4 Роли базы данных

Для обеспечения безопасности сервера базы данных должны быть определены следующие роли пользователей:

- **Гость или неавторизованный пользователь**, который должен иметь возможность чтения записей из таблиц игротек, настольных игр, их таблицы-связи, а также таблиц организаторов и мест проведения.
- **Игрок**, который должен иметь все возможности неавторизованного пользователя, а также возможность чтения таблиц игроков, избранных игр и регистраций, возможность обновления первой из них и вставки/удаления из двух последних.
- **Организатор** должен обладать теми же правами, что и гость, а также иметь возможность чтения таблиц игроков и регистраций, вставки в таблицу игротек и обновления записей в таблице игротек и организаторов.
- **Администратор** должен иметь возможность проводить все возможные операции над всеми таблицами в базе данных.

## 2.2 Проектирование приложения

Программного обеспечение для работы с базой данных будет разрабатываться как монолитное web-приложение. Структурно приложение строится на парадигмах ООП. На верхнем уровне выделено три компонента: компонент доступа к данным, через который происходит обращение к базе данных, компонент бизнес-логики и компонент пользовательского интерфеса. На рисунке 2.5 показаны классы первых двух компонентов. Связь между компонентом доступа к данным и компонентом бизнес-логики построена на основе паттерна репозиторий, обращение же компонента пользовательского интерфейса к сервисам происходит через их интерфейсы.

### Вывод

В данном разделе была спроектирована база данных: описаны таблицы, хранимые процедуры и функции, триггеры и роли, — а также была описана структура приложения.

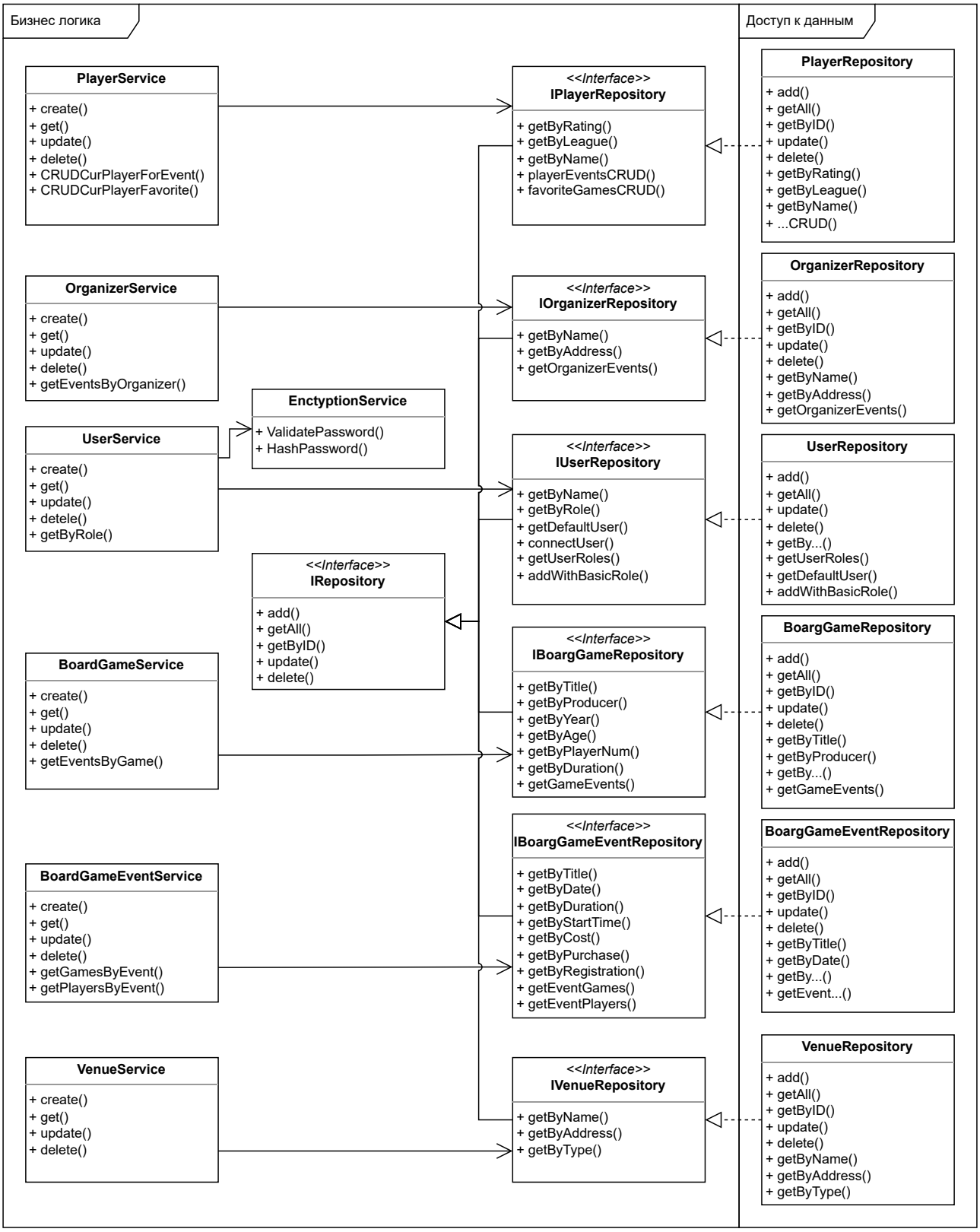


Рисунок 2.5 – Компоненты бизнес-логики и доступа к данным

# 3 Технологическая часть

## 3.1 Выбор системы управления базами данных

Система управления базами данных (СУБД) — это совокупность программ и языковых средств, предназначенных для управления данными в базе данных, ведения базы данных и обеспечения взаимодействия ее с прикладными программами [11].

В аналитической части для реализуемой системы были выбраны реляционные базы данных, следовательно, необходимо рассмотреть СУБД, предоставляющие возможность работы с ними:

- **Microsoft SQL Server** — реляционная система управления базами данных, разработанная корпорацией Microsoft, основным языком запросов которой является Transact-SQL [16]. Являясь разработкой Microsoft имеет возможность интеграции с другими продуктами компании, в том числе с облачными хранилищами, а также подробную документацию [17]. Однако при этом поддерживает меньшее число операционных систем по сравнению с другими рассматриваемыми СУБД [18] и имеет высокую стоимость корпоративной версии [19].
- **PostgreSQL** — это объектно-реляционная система управления базами данных с открытым исходным кодом, соответсвующая стандартам ANSI SQL [20]. Данная СУБД поддерживает пользовательские типы данных, в том числе неструктурированные (JSON и XML), а также интеграцию сторонних интрументов [21].
- **Oracle** — это объектно-реляционная система управления базами данных, созданная корпорацией Oracle [22]. Поддерживает работу с крупными БД и большим числом пользователей, имеет подробную документацию, однако версии без ограничения функциональностей имеют высокую стоимость [23].
- **MySQL** — это бесплатная реляционная система управления базами данных, разработку и поддержку которой осуществляет корпорация Oracle [16]. Хотя и распроняется бесплатно имеет платную поддержку, и неполностью соответсвует стандартам SQL [21].

### 3.1.1 Сравнение описанных СУБД

Основными критериями для выбора СУБД являются:

– **К1** — бесплатное распространение (логический критерий);
– **К2** — подробная открытая документация (логический критерий);
– **К3** — производительность (рейтинг на основе источника [24]).

Сравнение СУБД по этим критериям приведено в таблице 4.1.

Таблица 3.1 – Сравнение СУБД

| Решение | К1 | К2 | К3 |
|---|---|---|---|
| Microsoft SQL Server | - | + | 3 |
| PostgreSQL | + | + | 1 |
| Oracle | - | + | 2 |
| MySQL | + | - | 4 |

### Вывод

Для решения задачи была выбрана система управления базами данных PostgreSQL, так как она:

– наиболее производительна из рассматриваемых СУБД;
– имеет открытый исходный код;
– имеет подробную открытую документацию.

## 3.2 Выбор средств реализации приложения

Для реализации приложения выбран язык C# [25], так как он поддерживает принципы ООП, на которых основана структура разрабатываемого программного обеспечения.

Для реализации компонента доступа к базе данных выбран Entity Framework [26], позволяющий автоматизировать процесс создания базы данных и ее таблиц, а также поддерживающий работу с многими СУБД, в том числе и PostgreSQL.

Для реализации web-интерфейса выбран фреймворк Blazor [27] и библиотека компонентов Blazorise [28], для быстрой разработки интерфейса.

В качестве среды разработки выбрана Visual Studio [29], так как она поддерживает работу со всеми выше описанными средствами в том числе и их

установку, а также обладает встроенными средствами тестирования и отладки.

## 3.3 Детали реализации

### 3.3.1 Создание таблиц

Создание таблиц базы данных и соответствующих ограничений полей представленно на листингах 3.1-3.10.

Листинг 3.1 – Создание таблицы мест проведения

```
CREATE TABLE "Venues" (
  "ID" int8 NOT NULL GENERATED BY DEFAULT AS IDENTITY,
  "Name" text NOT NULL,
  "Type" text NOT NULL,
  "Address" text NOT NULL,
  "Email" text NULL,
  "URL" text NULL,
  "PhoneNumber" text NULL,
  "Deleted" bool NOT NULL,
  CONSTRAINT "PK_Venues" PRIMARY KEY ("ID")
);
```

Листинг 3.2 – Создание таблицы организаторов

```
CREATE TABLE "Organizers" (
  "ID" int8 NOT NULL GENERATED BY DEFAULT AS IDENTITY,
  "Name" text NOT NULL,
  "Address" text NOT NULL,
  "Email" text NULL,
  "URL" text NULL,
  "PhoneNumber" text NULL,
  "Deleted" bool NOT NULL,
  CONSTRAINT "PK_Organizers" PRIMARY KEY ("ID")
);
```

Листинг 3.3 – Создание таблицы игроков

```
CREATE TABLE "Players" (
  "ID" int8 NOT NULL GENERATED BY DEFAULT AS IDENTITY,
  "Name" text NOT NULL,
  "League" text NOT NULL,
  "Rating" int8 NOT NULL,
  "Deleted" bool NOT NULL,
  CONSTRAINT "PK_Players" PRIMARY KEY ("ID")
);
```

Листинг 3.4 – Создание таблицы игротек

```
CREATE TABLE "Events" (
  "ID" int8 NOT NULL GENERATED BY DEFAULT AS IDENTITY,
  "Title" text NOT NULL,
  "Date" date NOT NULL,
  "StartTime" time NOT NULL,
  "Duration" int8 NOT NULL,
  "Cost" int8 NOT NULL,
  "Purchase" bool NOT NULL,
  "OrganizerID" int8 NOT NULL,
  "VenueID" int8 NOT NULL,
  "Deleted" bool NOT NULL,
  "BeginRegistration" timestamp NOT NULL,
  "EndRegistration" timestamp NOT NULL,
  "Cancelled" bool NOT NULL DEFAULT false,0,
  CONSTRAINT "PK_Events" PRIMARY KEY ("ID")
);

ALTER TABLE public."Events"
ADD CONSTRAINT "FK_Events_Organizers_OrganizerID"
FOREIGN KEY ("OrganizerID") REFERENCES "Organizers"("ID")
ON DELETE CASCADE;

ALTER TABLE public."Events"
ADD CONSTRAINT "FK_Events_Venues_VenueID"
FOREIGN KEY ("VenueID") REFERENCES "Venues"("ID")
ON DELETE CASCADE;
```

Листинг 3.5 – Создание таблицы связи игротек и игроков

```
CREATE TABLE "Registrations" (
  "PlayerID" int8 NOT NULL,
  "BoardGameEventID" int8 NOT NULL,
  CONSTRAINT "PK_Registrations"
  PRIMARY KEY ("BoardGameEventID", "PlayerID")
);

ALTER TABLE public."Registrations"
ADD CONSTRAINT "FK_Registrations_Events_BoardGameEventID"
FOREIGN KEY ("BoardGameEventID") REFERENCES "Events"("ID")
ON DELETE CASCADE;

ALTER TABLE public."Registrations"
ADD CONSTRAINT "FK_Registrations_Players_PlayerID"
FOREIGN KEY ("PlayerID") REFERENCES "Players"("ID")ON DELETE
   CASCADE;
```

Листинг 3.6 – Создание таблицы игр

```
CREATE TABLE "Games" (
  "ID" int8 NOT NULL GENERATED BY DEFAULT AS IDENTITY,
  "Title" text NOT NULL, "Producer" text NULL,
  "Year" int8 NOT NULL, "MaxAge" int8 NOT NULL,
  "MinAge" int8 NOT NULL, "MaxPlayerNum" int8 NOT NULL,
  "MinPlayerNum" int8 NOT NULL,
  "MaxDuration" int8 NOT NULL,
  "MinDuration" int8 NOT NULL,
  "Deleted" bool NOT NULL,
  CONSTRAINT "PK_Games" PRIMARY KEY ("ID"));
```

Листинг 3.7 – Создание таблицы связи игр и игротек

```
CREATE TABLE "EventGameRelations" (
  "BoardGameID" int8 NOT NULL,
  "BoardGameEventID" int8 NOT NULL,
  CONSTRAINT "PK_EventGameRelations"
  PRIMARY KEY ("BoardGameID", "BoardGameEventID"));

ALTER TABLE public."EventGameRelations"
ADD CONSTRAINT "FK_EventGameRelations_Events_BoardGameEventID"
FOREIGN KEY ("BoardGameEventID") REFERENCES "Events"("ID")
ON DELETE CASCADE;

ALTER TABLE public."EventGameRelations"
ADD CONSTRAINT "FK_EventGameRelations_Games_BoardGameID"
FOREIGN KEY ("BoardGameID") REFERENCES "Games"("ID")
ON DELETE CASCADE;
```

Листинг 3.8 – Создание таблицы связи игр и игроков

```
CREATE TABLE "Favorites" (
  "BoardGameID" int8 NOT NULL,
  "PlayerID" int8 NOT NULL,
  CONSTRAINT "PK_Favorites"
  PRIMARY KEY ("BoardGameID", "PlayerID"));

ALTER TABLE public."Favorites"
ADD CONSTRAINT "FK_Favorites_Games_BoardGameID"
FOREIGN KEY ("BoardGameID") REFERENCES "Games"("ID")
ON DELETE CASCADE;

ALTER TABLE public."Favorites"
ADD CONSTRAINT "FK_Favorites_Players_PlayerID"
FOREIGN KEY ("PlayerID") REFERENCES "Players"("ID")
ON DELETE CASCADE;
```

Листинг 3.9 – Создание таблицы пользователей

```
CREATE TABLE "Users" (
  "ID" int8 NOT NULL GENERATED BY DEFAULT AS IDENTITY,
  "Name" text NOT NULL,
  "Password" text NOT NULL,
  CONSTRAINT "PK_Users" PRIMARY KEY ("ID")
);
```

Листинг 3.10 – Создание таблицы ролей

```
CREATE TABLE "Roles" (
  "ID" int8 NOT NULL GENERATED BY DEFAULT AS IDENTITY,
  "RoleName" text NOT NULL,
  "RoleID" int8 NOT NULL,
  "UserID" int8 NOT NULL,
  CONSTRAINT "PK_Roles" PRIMARY KEY ("ID")
);

ALTER TABLE public."Roles"
ADD CONSTRAINT "FK_Roles_Users_UserID"
FOREIGN KEY ("UserID") REFERENCES "Users"("ID")
ON DELETE CASCADE;
```

### 3.3.2 Создание ролей

Создание ролей Гостя, Игрока, Организатора и Администратора представлены на листингах 3.11, 3.12, 3.13, 3.14 соответсвтенно.

Листинг 3.11 – Создание роли гостя

```
create role guest with
  login
  nosuperuser
  nocreatedb
  nocreaterole
  noreplication
  password 'guest'
  connection limit -1;

grant select on public."EventGameRelations"
        , public."Events"
        , public."Games"
        , public."Organizers"
        , public."Venues"
to guest;
```

Листинг 3.12 – Создание роли игрока

```
create role player with
  login
  nosuperuser
  nocreatedb
  nocreaterole
  noreplication
  password 'player'
  connection limit -1;

grant select on public."EventGameRelations"
        , public."Events"
        , public."Games"
        , public."Organizers"
        , public."Venues"
        , public."Favorites"
        , public."Players"
        , public."Registrations"
to player;

grant insert on public."Favorites"
          , public."Registrations"
to player;

grant update ("Name") on public."Players"
to player;

grant delete on public."Favorites"
        , public."Registrations"
to player;
```

Листинг 3.13 – Создание роли организатора

```
create role organizer with
  login
  nosuperuser
  nocreatedb
  nocreaterole
  noreplication
  password 'organizer'
  connection limit -1;

grant select on public."EventGameRelations"
        , public."Events"
        , public."Games"
        , public."Organizers"
        , public."Venues"
        , public."Players"
        , public."Registrations"
to organizer;

grant insert on public."Events"
to organizer;

grant update on public."Events"
              , public."Organizers"
to organizer;
```

Листинг 3.14 – Создание роли администратора

```
create role "admin" with
  login
  nosuperuser
  nocreatedb
  nocreaterole
  noreplication
  password 'admin'
  connection limit -1;

grant all privileges on all tables in schema public to "admin";

select * from pg_catalog.pg_roles;

SELECT *
FROM information_schema.role_table_grants
where grantee = 'admin';
```

### 3.3.3 Создание хранимых процедур и функций

Создание функции получения соостояния игротеки представлено на листинге 3.15, процедура обновления списка игр, по которым проводится игротека, представлена на листинге 3.16.

Листинг 3.15 – Функция получение состояния игротеки

```
create or replace
function get_event_state(eventDate date
                         , startTime time
                         , duration bigint
                         , beginReg  timestamp
                         , endReg timestamp
                         , cancelled bool
                         , deleted bool)
returns integer as
$$
declare
  curTime timestamp = now();
  beginEvent timestamp := eventDate + startTime;
  endEvent timestamp := beginEvent
                           + (duration * interval '1 minute');
begin
  case
    when cancelled then
      return 5;
    when deleted then
      return 6;
    when curTime < beginReg then
      return 0;
    when curTime < endReg then
      return 1;
    when curTime < beginEvent then
      return 2;
    when curTime < endEvent then
      return 3;
    else
      return 4;
  end case;
end;
```

### 3.3.4 Создание триггеров

Триггер «BEFORE INSERT» на таблице Registrations, реализующий проверку времени регистрации на игротеку представлен на листинге 3.17.

Листинг 3.16 – Процедура обновления игр игротеки

```
create or replace
procedure update_event_games("gamesIDs" bigint[], "eventID"
   bigint)
language plpgsql as
$$
begin
  delete from "EventGameRelations" egr
  where egr."BoardGameEventID" = "eventID"
        and egr."BoardGameID" <> all("gamesIDs");

  insert into "EventGameRelations"("BoardGameEventID" , "
     BoardGameID")
  select "eventID", unnest("gamesIDs")
  on conflict ("BoardGameEventID", "BoardGameID") do nothing;
end
$$
```

Листинг 3.17 – Триггер проверки завершения времени регистрации на игротеку

```
create or replace function check_registration_time()
returns trigger as
$$
declare
  beginReg timestamp;
  endReg   timestamp;
begin
  select e."BeginRegistration", e."EndRegistration"
  from "Events" as e
  where e."ID" = new."BoardGameEventID"
  into beginReg, endReg;

  if beginReg <= current_timestamp
     and current_timestamp < endReg then
    return new;
  else
    raise exception 'Registration is over!';
  end if;
end
$$ language plpgsql;

create or replace trigger check_registration_time
before insert on public."Registrations"
for each row execute function check_registration_time();
```

## 3.4 Пример работы программы

Начальная страница приложения представлена на рисунке **??**. Эту страницу видит гость при открытии приложения, на нее происходит переход при входе, регистрации и смене роли пользователя с игрока на организатора и наоборот.

Неавторизованный пользователь имеет доступ к спискам игротек (рисунок **??**) и настольных игр (рисунок **??**), также предоставляется доступ к просмотру полной информации о конкретной игротеке или игре на отдельных страницах. К этим страницам имеют доступ и пользователи с другими ролями. На странице игротеки гостю доступно нажатие на кнопку «Зарегистрироваться» (если регистрация на игротеку идет), но при нажатии на нее отображается всплывающее окно с предложением зарегистрироваться или войти **??**.

Гостю также предлагается возможность входа и регистрации. Соответствующие формы представлены на рисунках **??**, **??**. При вводе в поля форм осуществляется валидация введенных данных, ее пример представлен на рисунке **??**.

При успешном входе или регистрации пользователю выдается базовая роль игрока. Игрок имеет возможность зарегистрироваться на игроку нажатием на кнопку «Зарегистрироваться» на странице игротеки и получить подтверждение (рисунок **??**). После чего имеет возможность отменить регистрацию нажатием на кнопку «Отменить». Аналогично происходит добавление настольной игры в избранное: при посещении страницы настольной игры в роли игрока на ней отображается кнопка добавления в избранное (рисунок **??**). Игротеки, на которое игрок зарегистрировался и игры, которые он добалит в избранное, игрок может посмотреть на соответсвующих страницах, доступ к которым осуществляется через выпадающее меню (**??**). Там же пользователь может изменить роль.

Изначально у пользователя нет роли организатора, чтобы им стать игрок должен заполнить дополнительную форму с информацией об организаторе (рисунок **??**). После переключения на роль организатора пользователь может создать свою игротеку (рисунок **??**), изменить ее или отменить на странице «Мои игротеки» (рисунок **??**).

Также в системе предусмотрена роль администратора. Изначально только один пользователь имеет эту роль. Администратор имеет возможность изменять

информацию об играх и местах проведения (пример представлен на рисунке **??**), выдавать и отзывать роль администратора у других пользователей **??**, а также удалять их **??**. Отозвать роль администратора у тогоп пользователя, который изначально имел эту роль нельзя.

**КАРТИНКИ**

# 4 Исследовательская часть

## 4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялось тестирование:

- Операционная система: Windows 10 Home.
- Память: 8 GiB.
- Процессор: Intel® Core™ i5-8265U, 4 физических ядра, 8 логических ядра.

Эксперимент проводился на ноутбуке, включенном в сеть электропитания. Во время тестирования ноутбук был нагружен только встроенными приложениями окружения, окружением, а также непосредственно системой тестирования.

## 4.2 Описание эксперимента

Целью эксперимента является проверка зависимости времени выполнения запроса от индексации на различных количествах записей в таблице.

Для проведения эксперимента выбрана таблица игротек как реализация основной сущености в системе. Для индексации выбрано поле даты как наиболее востребованное при поиске и сортировке. Тестирование проводилось на времени выполнения запроса сортировки при количествах записей от 10 до 1000.

Создание индекса приведено на листинге 4.1.

Листинг 4.1 – Создание таблицы мест проведения

```
1 create index "DateIndex" on public."Events" using btree("Date");
```

В ходе эксперимента время выполнения запросы измерялось с помощью встроенных инструментов СУБД PostgreSQL (листинг 4.2).

Листинг 4.2 – Создание таблицы мест проведения

```
1 explain analyse select * from public."Events" order by "Date";
```

Примеры выполнения данного запроса с индексацией и без нее представлены на рисунках 4.1, 4.2 соответственно.

| | QUERY PLAN |
|---|---|
| 1 | Index Scan using "DateIndex" on "Events" e  (cost=10000000000.27..10000000122.97 rows=1000 width=113) (actual time=0.028..0.964 rows=1000 loops=1) |
| 2 | Planning Time: 0.152 ms |
| 3 | Execution Time: 1.035 ms |

Рисунок 4.1 – План запроса на сортировку по дате с индексированием

| | QUERY PLAN |
|---|---|
| 1 | Sort  (cost=81.83..84.33 rows=1000 width=113) (actual time=1.055..1.139 rows=1000 loops=1) |
| 2 | Sort Key: "Date" |
| 3 | Sort Method: quicksort  Memory: 279kB |
| 4 | ->  Seq Scan on "Events" e  (cost=0.00..32.00 rows=1000 width=113) (actual time=0.016..0.485 rows=1000 loops=1) |
| 5 | Planning Time: 0.100 ms |
| 6 | Execution Time: 1.232 ms |

Рисунок 4.2 – План запроса на сортировку по дате без индексирования

## 4.3 Результат эксперимента

В таблице 1.3 приведены результаты измерений. График, иллюстрирующий данную таблицу приведен на рисунке 4.3.

Таблица 4.1 – Сравнение СУБД

| **Число записей** | **Без индекса, мс** | **С индексом, мс** |
|---|---|---|
| 10 | | |
| 30 | | |
| 60 | | |
| 100 | | |
| 300 | | |
| 600 | | |
| 1000 | | |

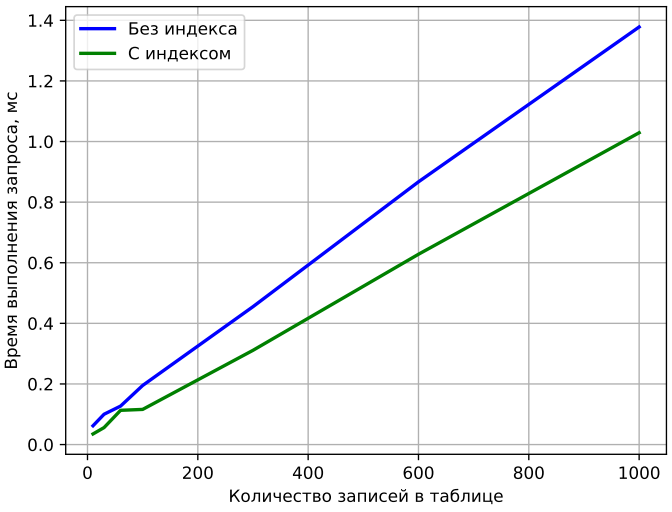


Рисунок 4.3 – График зависимости времени выполнения запроса от числа записей в таблице без и с индексированием

## Вывод

Из полученных результатов можно сделать вывод, что использование индексирования при малых количествах записей (до 100) дает уменьшение времени выполнения запроса в/на ..., при больших количествах (от 100 до 100) уменьшение времени в 1.4 раза.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была спроектирована и реализована база данных настольных игр и игротек, также разработано приложение предоставляющее отсутствующую в аналогах функциональность: возможность регистрации игроков на игротеки разных организаторов без использования сторонних ресурсов.

В результате эксперимента было выяснено, что с помощью использование индексов при запросах на упорядочение записей приводит к уменьшению времени выполнения запроса в x раз на малых количества записей (до 100) и в 1.4 раза на количествах записей от 100 до 1000.

В процессе выполнения курсовой работы:

- проведен анализ предметной области;
- формализована задача;
- проведен анализ баз данных и систем управления базами данных;
- спроектирована база данных и архитектура приложения;
- реализована база данных и приложение для доступа к ней;
- проведен эксперимент по сравнению времени выполнения запросов с использованием индексов и без них, результат которого описан выше.

Таким образом, все поставленные задачи были выполнены, а цель достигнута.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. *Своротова Ю. В.* Использование настольных игр в образовательном процессе // Развитие современного образования: от теории к практике. Сборник материалов VII Всероссийской научно-практической конференции с международным участием, Чебоксары, 14 июня 2019 г. — 2019. — С. 145—150.
2. *Журавков Д. Д.* Роль маркетинга в процессе формирования новых типов бизнеса на примере мирового рынка настольных игр // ЭТАП: Экономическая Теория, Анализ, Практика. — 2013. — № 1. — С. 139—148.
3. *Герасикова Е. Н.*, *Родина Е. Н.*, *Шпакова Г. А.* Рынок настольных игр: тенденции и способы продвижения // Инновации в науке. — 2018. — 6 (82). — С. 58—60.
4. Магазины настольных игр Hobby Games [Электронный ресурс]. — URL: `https://hobbygames.ru/` (дата обращения: 24.04.2022).
5. Магазин настольных игр и подарков Мосигра [Электронный ресурс]. — URL: `https://www.mosigra.ru/` (дата обращения: 24.04.2022).
6. Интернет-магазин настольных игр Низа Гамс [Электронный ресурс]. — URL: `https://nizagams.ru/` (дата обращения: 24.04.2022).
7. BoardGameGeek – Gaming Unplugged Since 2000. — URL: `https://boardgamegeek.com/` (дата обращения: 26.04.2022).
8. Top Tier Board Games. — URL: `https://toptiergaming.com/` (дата обращения: 26.04.2022).
9. Board Game Halv | Board Game Culture and Enthusiasm. — URL: `https://www.boardgamehalv.com/` (дата обращения: 26.04.2022).
10. Board Game Arena. — URL: `https://boardgamearena.com/` (дата обращения: 26.04.2022).
11. ГОСТ 20886-85. Организация данных в системах обработки данных. Термины и определения [Электронный ресурс]. — URL: `https://docs.cntd.ru/document/1200015708` (дата обращения: 28.04.2022).
12. *Аврунев О. Е.*, *Стасышин В. М.* Модели баз данных: учебное пособие. — Новосибирск : Издательство НГТУ, 2018. — С. 124.

13. Базы данных: учебное пособие / В. И. Халимон [и др.]. — СПб. : СПбГТИ(ТУ), 2017. — С. 118.

14. *Саидов Н. В.* Основные преимущества реляционных SQL баз данных над нереляционными NoSQL // «Научное сообщество студентов XXI столетия. Технические науки»: Электронный сборник статей по материалам XCVIII студенческой международной научно-практической конференции. — Новосибирск, 2021. — 2 (97). — С. 17–21.

15. *Зенцов Д. А.*, *Галеева А. И.*, *Додонов М. В.* Сравнение реляционных и нереляционных (NoSQL) баз данных // «Научное сообщество студентов XXI столетия. Междисциплинарные исследования»: Электронный сборник статей по материалам XLIV студенческой международной научно-практической конференции. — Новосибирск, 2018. — 9 (44). — С. 76–80.

16. *Vershinin I. S.*, *Mustafina A. R.* Performance Analysis of PostgreSQL, MySQL, Microsoft SQL Server Systems Based on TPC-H Tests // 2021 International Russian Automation Conference (RusAutoCon). — Sochi, 2021. — С. 683–687. — DOI: 10.1109/RusAutoCon52004.2021.9537400.

17. Техническая документация по SQL Server. — URL: https://docs.microsoft.com/ru-ru/sql/sql-server/?view=sql-server-ver15 (дата обращения: 12.05.2022).

18. Microsoft SQL Server and Oracle: Comparative Performance Analysis / M. Ilic [и др.] // The 7th International conference Knowledge management and informatics. — Vrnjacka Banja, 2021. — С. 33–40.

19. SQL Server 2019 — Цены | Microsoft. — URL: https://www.microsoft.com/ru-ru/sql-server/sql-server-2019-pricing?rtc=1 (дата обращения: 12.05.2022).

20. PostgreSQL: About [Электронный ресурс]. — URL: https://www.postgresql.org/about/ (дата обращения: 13.05.2022).

21. Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch, and others. — URL: https://www.altexsoft.com/blog/business/comparing-database-management-systems-mysql-postgresql-mssql-server-

mongodb - elasticsearch - and - others/ (дата обращения: 12.05.2022).

22. Технологии баз данных Oracle. — URL: https://www.oracle.com/cis/database/technologies/ (дата обращения: 12.05.2022).
23. Oracle Price List. — URL: https://www.oracle.com/cloud/price-list/ (дата обращения: 12.05.2022).
24. *Hossain M. I.*, *Mahmud S.*, *Santa T. D.* Oracle, MySQL, PostgreSQL, SQLite, SQL Server: Performance based competitive analysis. — 2019.
25. Документация по C#. — URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/ (дата обращения: 14.09.2022).
26. Документация по Entity Framework. — URL: https://docs.microsoft.com/ru-ru/ef/ (дата обращения: 14.09.2022).
27. Blazor Framework. — URL: https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor (дата обращения: 14.09.2022).
28. Blazorise Component Library. — URL: https://blazorise.com/ (дата обращения: 14.09.2022).
29. Visual Studio. — URL: https://visualstudio.microsoft.com/ru/ (дата обращения: 14.09.2022).