­Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Базы Данных (БД)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему

Програмное средство «Справочник контента World of Warships»

БГУИР КП 1-40 01 01 014 ПЗ

Выполнил

студент: гр. 151004 Дубовский А.В.

Руководитель: Новицкая Н. В.

Минск 2024

содержание

[Введение 3](#_Toc165985447)

[1 Аналитический обзор литературы 5](#_Toc165985448)

[1.1 Обзор существующих аналогов 5](#_Toc165985449)

[1.2 Требования к проектируемому программному средству 7](#_Toc165985450)

[1.3 Постановка задачи 8](#_Toc165985451)

[2 Анализ требований к программному средству и разработка функциональных требований 10](#_Toc165985452)

[2.1 Обоснование выбора СУБД и иных средств разработки 10](#_Toc165985453)

[2.2 Спецификация функциональных требований 11](#_Toc165985454)

[2.3 Учитываемые функции и бизнес-процессы 11](#_Toc165985455)

[2.4 Ключевые особенности базы данных 11](#_Toc165985456)

[2.5 Пользователи системы и их роли 11](#_Toc165985457)

[2.6 Параметры поиска информации 12](#_Toc165985458)

[2.7 Бизнес-процессы, связанные с реализацией информационной системы 12](#_Toc165985459)

[3 Инфологическая модель предметной области 13](#_Toc165985460)

[3.1 Описание моделей 13](#_Toc165985461)

[3.2 Сущности и связи 14](#_Toc165985462)

[3.3 Особенности нормализации 19](#_Toc165985463)

[4 Подробное описание бизнес-логики 21](#_Toc165985464)

[4.1 Хранимые процедуры 21](#_Toc165985465)

[4.2 Основные алгоритмы ПС 24](#_Toc165985466)

[5 Тестирование базы данных и приложений системы 30](#_Toc165985467)

[5.1 Тестирование хранимых процедур 30](#_Toc165985468)

[5.2 Тестирование программного средства 30](#_Toc165985469)

[Заключение 33](#_Toc165985470)

[Список использованной литературы 34](#_Toc165985471)

[Приложение А 35](#_Toc165985472)

# Введение

Базы данных – это организованные собрания данных, которые обычно хранятся и обрабатываются компьютерной системой. Они играют решающую роль в организации, хранении и управлении данными в различных сферах, включая бизнес, образование, науку и государственное управление. Базы данных предоставляют эффективный способ организации информации и обеспечивают доступ к данным для различных пользователей и приложений.

История баз данных уходит корнями в 1960-е годы, когда были разработаны первые концепции и технологии управления данными. Одним из ключевых моментов стало появление реляционной модели данных в 1970-х годах, предложенной Эдгаром Коддом. Эта модель стала основой для развития многих современных баз данных.

С тех пор базы данных представляют собой важный инструмент для хранения, организации и управления данными в различных сценариях. Различные типы баз данных предлагают разнообразные способы структурирования данных и обеспечивают уникальные возможности для различных приложений и задач.

1. Реляционные базы данных (RDBMS)

Реляционные базы данных основаны на реляционной модели данных, предложенной Эдгаром Коддом в 1970-х годах. В этой модели данные организованы в виде таблиц, состоящих из строк (кортежей) и столбцов (атрибутов). Каждая таблица представляет собой сущность, а отношения между таблицами устанавливаются с помощью ключей. Реляционные СУБД, такие как Oracle, MySQL, PostgreSQL и Microsoft SQL Server, широко используются в различных областях для управления структурированными данными.

2. Нереляционные базы данных (NoSQL)

Нереляционные базы данных, или NoSQL, предоставляют альтернативные подходы к хранению и организации данных, не ограничиваясь таблицами и отношениями. Вместо этого они могут использовать различные модели данных, такие как ключ-значение, столбцовая, документ-ориентированная или графовая модель. Эти базы данных обычно более гибкие и могут обрабатывать неструктурированные или полуструктурированные данные. Примеры NoSQL баз данных включают MongoDB, Cassandra, Couchbase и Neo4j.

3. Иерархические базы данных

Иерархические базы данных организуют данные в виде иерархии или дерева, где каждый элемент имеет родителя и потомков. Этот тип баз данных был широко использован в прошлом, особенно в системах управления базами данных на ранних этапах развития компьютерных технологий. Примером иерархической базы данных является IMS (Information Management System) от IBM.

4. Сетевые базы данных

Сетевые базы данных расширяют концепцию иерархических баз данных, позволяя каждому элементу иметь несколько родителей и потомков. Это позволяет более сложным структурам данных и более гибким запросам, но такие базы данных стали менее популярными в сравнении с реляционными и NoSQL базами данных.

5. Объектно-ориентированные базы данных

Объектно-ориентированные базы данных разработаны для хранения объектов программирования в базе данных. Они расширяют концепции объектно-ориентированного программирования на уровень хранения данных. Этот тип баз данных обычно используется в приложениях, где объекты имеют сложную структуру или содержат методы. Примеры объектно-ориентированных баз данных включают db4o и ObjectDB.

6. Документ-ориентированные базы данных

Документ-ориентированные базы данных предназначены для хранения и управления документами или JSON-подобными объектами. Каждый документ может иметь свою собственную структуру, и базы данных обычно предоставляют механизмы для индексации и запроса документов. MongoDB является одним из самых популярных примеров документ-ориентированных баз данных.

7. Графовые базы данных

Графовые базы данных используют графовую модель для хранения данных, где узлы представляют сущности, а ребра - отношения между ними. Этот тип баз данных идеально подходит для моделирования связей между объектами, таких как социальные сети, сети дорог или сети компьютеров. Примерами графовых баз данных являются Neo4j, Amazon Neptune и ArangoDB.

Целью данной курсовой работы является разработка программного средства для упрощения знакомства с игрой «World of Warships». Для реализации неообходимо решить следующие задачи:

* получить практические навыки взаимодействие платформы .Net с брокером сообщений RabbitMQ;
* получить практические навыки взаимодействие платформы .Net с базой данный MySQL;
* проектировка и разработка базы данных;
* разработать пользовательский интерфейс.

База данных для программного средства может быть разработана с использованием системы управления базами данных (СУБД), например, MySQL. Это позволяет обеспечить удобное хранение и быстрый доступ к данным.

Основываясь на легкости интеграции MySQL в различные веб-технологиях, а также ее гибкости и производительности, можно создать базу данных, которая не только эффективно поддерживает функциональности программного средства, но и легко интегрируется в существующие веб-сервисы и системы управления контентом.

В результате выполнения курсовой работы будет создано программное средство, позволяющее познакомиться с механиками игры «World of Warships».

# Аналитический обзор литературы

## Обзор существующих аналогов

### Сайт «Azur lane wiki»

Одним из аналогов, взятым за основу, является Wiki-сайт игры Azure Lane, изображённый на рисунке 1.1.

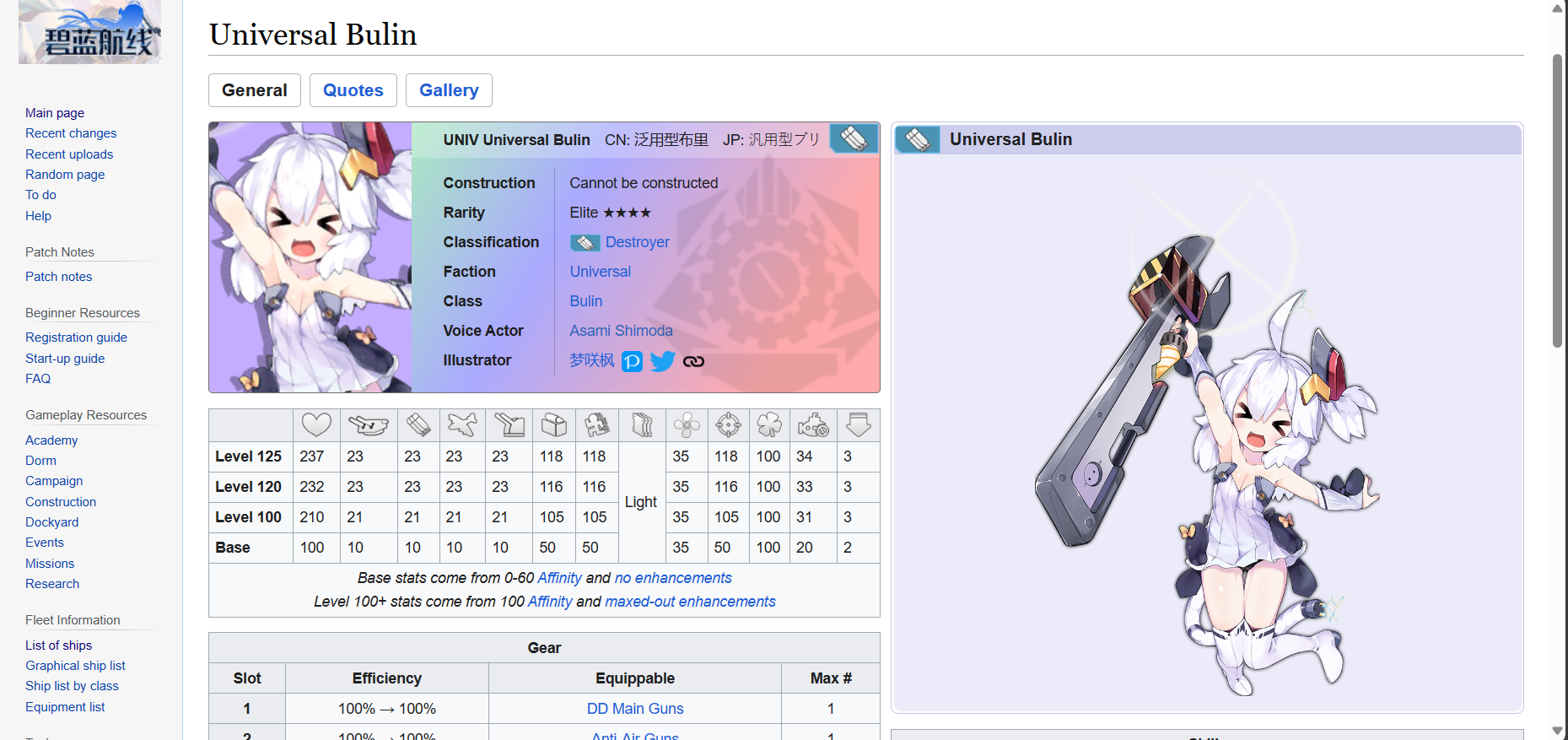


Рисунок 1.1 – Фрагмент сайта «Azur lane wiki»

Достоинства:

* удобная навигация;
* удалённая доступ к данным, без необходимости устанавливать программу большого размера.

Недостатки:

* интерфейс перегружен информацией.

### Cайт «World of warship wiki»

Одним из аналогов, взятым за основу, является оригинальный Wiki-сайт игры World of Warships, изображённый на рисунке 1.1.

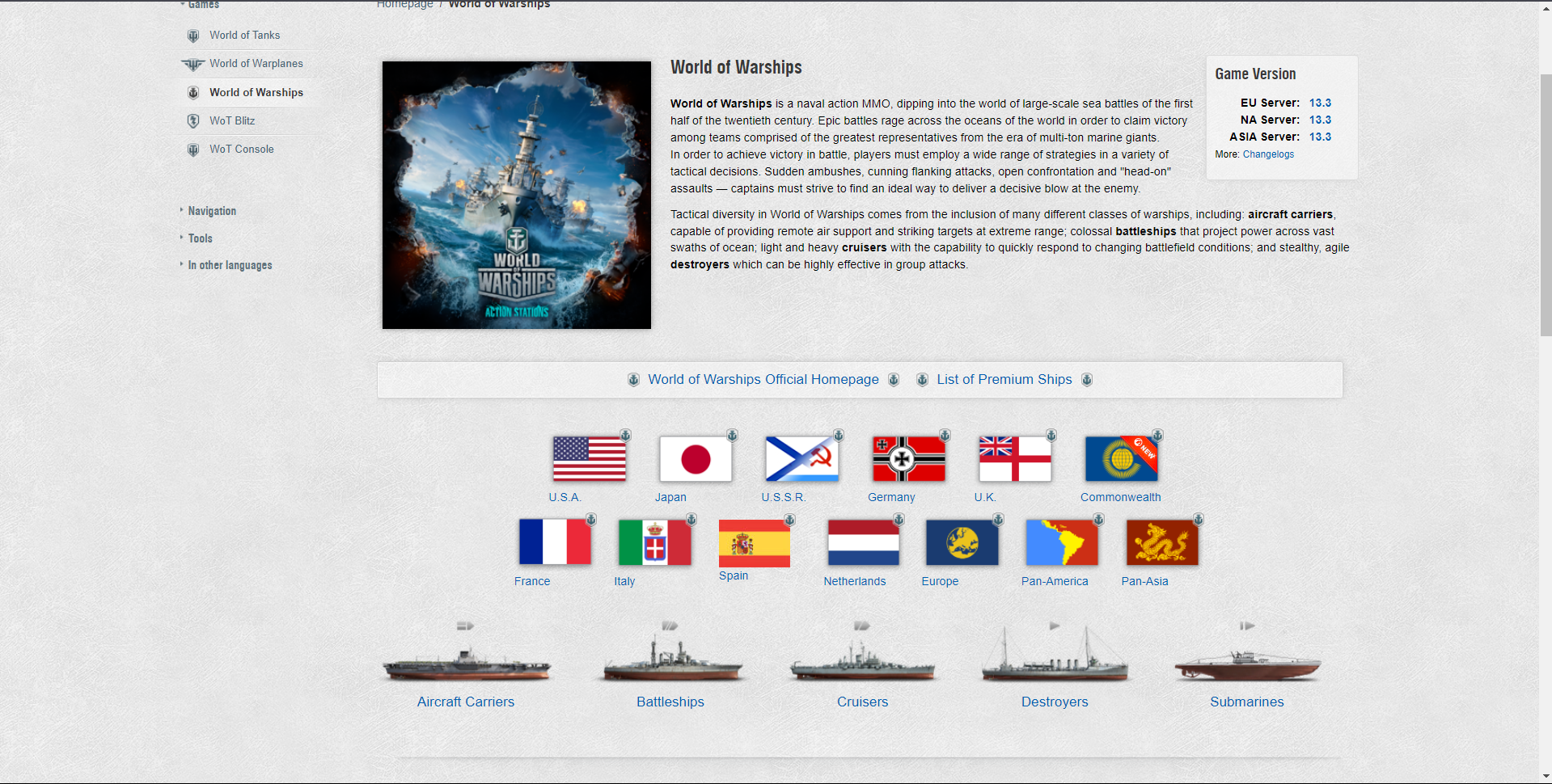


Рисунок 1.2 – Фрагмент Главной страницы wiki-сайта World of Warships

Достоинства:

* простой и понятный интерфейс;
* удобная навигация по всем разделам с главного экрана.

Недостатки:

Явных недостатков обнаружено не было.

## Требования к проектируемому программному средству

### Назначение разработки

Целью данной разработки является создание приложения справочник для игры «World of Warships». Приложение предназначено для предоставления игрокам полной и достоверной информации о мире игры, его механиках, событиях, кораблях, стратегиях и тактиках игры, а также для обмена опытом и советами между игроками.

Основная задача приложения - обеспечить пользователей всей необходимой информацией для успешного прохождения игры, улучшения игрового опыта и раскрытия потенциала их игровых кораблей.

### Состав выполняемых функций

В рамках данного курсового проекта планируется разработка базы данных для справочника приложения игры «World of Warships». Основные функциональные возможности базы данных:

* Учет кораблей: База данных будет содержать подробные описания и характеристики всех доступных кораблей в игре "World of Warships", чтобы пользователи могли получить полезную информацию о каждом из них.
* Учет игровых механик: Будут представлены данные о различных игровых механиках, таких как управление кораблем, стрельба, маневрирование и тактические приемы, чтобы игроки могли лучше понять игровой процесс.
* Учет информация о странах производства кораблей: БД хранит информацию о странах происхождения кораблей, включая название. ПС позволяет просматривать информацию о странах происхождения кораблей.
* Учет обновлений и патчей: Пользователи смогут получать информацию о последних обновлениях, изменениях и исправлениях в игре, чтобы быть в курсе последних событий.

### Входные данные

Информация о:

* Странах;
* Кораблях;
* Командирах;
* Картах;
* Уровнях игроков;
* Достижениях;
* Контейнерах.

### Выходные данные

* Информация о кораблях: уровень, название, характеристики;
* Информация о странах: флаг, описание;
* Информация о командирах: изображение, описание;
* Информация о картах: размеры, уровни доступа;
* Информация о уровнях игроков: доступные функции;
* Информация о достижениях: название, описание;
* Информация о контейнерах: получаемый лут.

### Требования к составу и параметрам технических и программных средств

В состав технических средств входит сервер, включающий в себя:

* процессор Pentium 3 и выше;
* 256 Мбайт оперативной памяти;
* 512 Мбайт свободного пространства на жестком диске.

### Требования к информационной и программной совместимости

Программа рассчитывается на работу на сервере и требует иметь .NET 8, RabbitMQ, WAMP на устройстве сервера. А для клиентской стороны достаточно наличие поддерживаемой операционной системы: Windows, Android.

### Обоснование выбора языка и сред разработки

Для разработки программного средства были выбраны язык C# и среда разработки Visual Studio 2022, так как — это мощный и популярный инструментарий для разработки приложений. C# - современный язык программирования с понятным синтаксисом, а Visual Studio 2022 предлагает удобную интегрированную среду разработки с множеством функций. Они обеспечивают разработчикам доступ к богатой экосистеме, высокую производительность и надежность приложений, а также поддержку и обновления со стороны Microsoft.

## Постановка задачи

Разработать справочник приложение для игры «World of Warhips» для получения данных об игре из базы данных. Справочник должен предоставлять пользователям возможность получать актуальную информацию по игровым аспектам. База данных должна содержать информацию о игровых объектах, включая название, описание, путь к картинкам и другие связанные данные для обеспечения полноценного справочника.

# Анализ требований к программному средству и разработка функциональных требований

## Обоснование выбора СУБД и иных средств разработки

Выбор в качестве системы управления базами данных (СУБД) был сделан в пользу MySQL, свободной реляционной СУБД, разрабатываемой корпорацией Oracle. Этот продукт распространяется под лицензией GNU General Public License и коммерческой лицензией, и обычно применяется в малых и средних приложениях, включая серверы WAMP и LAMP.

MySQL может действовать как сервер, обслуживающий локальных или удаленных клиентов, и включает в себя внутренний сервер для встраивания в автономные программы. СУБД обладает гибкостью благодаря поддержке различных типов таблиц, таких как MyISAM и InnoDB, с различными характеристиками, включая поддержку полнотекстового поиска и транзакций.

MySQL имеет ограничения по размеру таблиц, зависящие от их типа. Например, MyISAM ограничен размером файловой системы, тогда как InnoDB может использовать несколько файлов для одной таблицы, увеличивая общее табличное пространство.

Созданы различные форки MySQL, включая MariaDB. MySQL поддерживает разнообразные типы данных, операторы и функции SQL, а также обеспечивает функциональности, такие как репликация, транзакции, работа с внешними ключами и другие. СУБД пользуется популярностью благодаря своей простоте, обширным функциональным возможностям, системам безопасности и поддержке SQL.

MySQL также реализует функцию разграничения доступа, позволяя определить уровни доступа для разных пользователей. Эта особенность усиливает безопасность и дает возможность организовать управление данными в соответствии с ролями.

Важным аспектом является также активное сообщество разработчиков и поддержка со стороны Oracle, что гарантирует постоянное обновление и совершенствование СУБД. MySQL остается одним из ведущих решений для различных проектов, включая веб-приложения, корпоративные системы и приложения с невысокой нагрузкой.

Таким образом, MySQL представляет собой оптимальное решение для курсового проекта, обеспечивая легкость в настройке, надежность и масштабируемость при необходимости.

## Спецификация функциональных требований

Для хранения информации о кораблях в игре «World of Warships», база данных должна предоставлять возможность хранить разнообразные характеристики каждого танка, такие как название, тип (эсминец, крейсер, линкор и т. д.), уровень (Tier), страна производства, основные технические параметры (например, броня, мощность двигателя, скорость), исторические данные, а также другую дополнительную информацию, включая особенности внешнего вида.

Кроме того, база данных должна предоставлять возможность хранить информацию о командире каждого корабля, включая их навыки, опыт, статус и принадлежность к конкретным кораблям. Это позволит пользователям эффективно управлять своими командирами и оптимизировать их навыки для каждой конкретной техники.

Для обеспечения гибкости и удобства использования, база данных должна также поддерживать хранение информации о модулях и оборудовании, которые могут быть установлены на корабли. Это включает в себя данные о различных видах орудий, брони, двигателей, модулей управления и других элементов, которые могут влиять на боевые характеристики и тактику каждого корабля.

Для удобства пользователей база данных должна обеспечивать возможность поиска информации о кораблях по различным критериям, таким как тип, уровень, страна производства, технические характеристики и другие параметры. Также важно предоставить возможность просмотра и анализа дополнительной информации о технике, такой как исторические сведения, особенности игрового процесса и т. д.

Использование базы данных для хранения информации о кораблях позволит пользователям эффективно планировать и развивать свои игровые стратегии, а также получать удовольствие от игрового процесса благодаря более глубокому пониманию и использованию различных аспектов игрового мира.

## Учитываемые функции и бизнес-процессы

База данных для справочника должна иметь возможность хранить информацию о игровых объектах, включая название, описание, путь картинки и другие характеристики.

## Ключевые особенности базы данных

Программное средство должно позволять работать одновременно большому числу пользователей, что требует хорошей оптимизации как самого программного средства, так и базы данных. База данных должна содержать в себе сущности для хранения таких важных данных, как посты, комментарии, информация о пользователях. Для обеспечения эффективной работы программного средства и базы данных в условиях большой нагрузки можно применить следующие оптимизационные подходы:

* Индексирование: Создание индексов на часто используемых полях в базе данных позволяет ускорить поиск и фильтрацию данных. Например, можно создать индексы для полей, используемых при поиске фильмов по названию, жанру или режиссеру, а также для полей, связанных с пользователями, таких как имена или адреса электронной почты.
* Оптимизация запросов: Анализ и оптимизация запросов к базе данных могут значительно повысить производительность системы. Использование индексов, правильное написание запросов с использованием оптимальных операторов и избегание излишних запросов могут сократить время выполнения запросов и уменьшить нагрузку на базу данных.

## Пользователи системы и их роли

Гость: может просматривать информацию об игровых объектах.

Авторизованный пользователь: может редактировать информацию об игровых объектах.

## Параметры поиска информации

Данные поступают в базу данных через API, который реализован с использованием RabbitMQ. RabbitMQ служит для связи клиента и сервера. Для общения сервера и MySql используется MySql коннектор.

Когда клиент отправляет запрос через API, соответствующий контроллер получает запрос из RabbitMQ и выполняет получение данных из MySql.

После получения данных из базы данных сервис возвращает их контроллеру, который в свою очередь формирует ответ и отправляет его обратно клиенту через RabbitMQ. Клиентский интерфейс отображает полученные данные пользователю.

## Бизнес-процессы, связанные с реализацией информационной системы

Основные особенности реализации информационной системы:

* Множество запросов на чтение: предполагается, что пользователи базы данных часто выполняют запросы на чтение данных, например, для аналитических целей или отображения информации, это может потребовать оптимизации процесса чтения данных.
* Большое количество параллельных запросов: база данных получаетбольшое количество параллельных запросов от разных пользователей. Поэтому следует разработать план оптимизации как запросов к базе данных, так и саму базу данных. Также для программного средства тоже следует оптимизировать архитектуру, для более быстрого выполнения запросов.

# Инфологическая модель предметной области

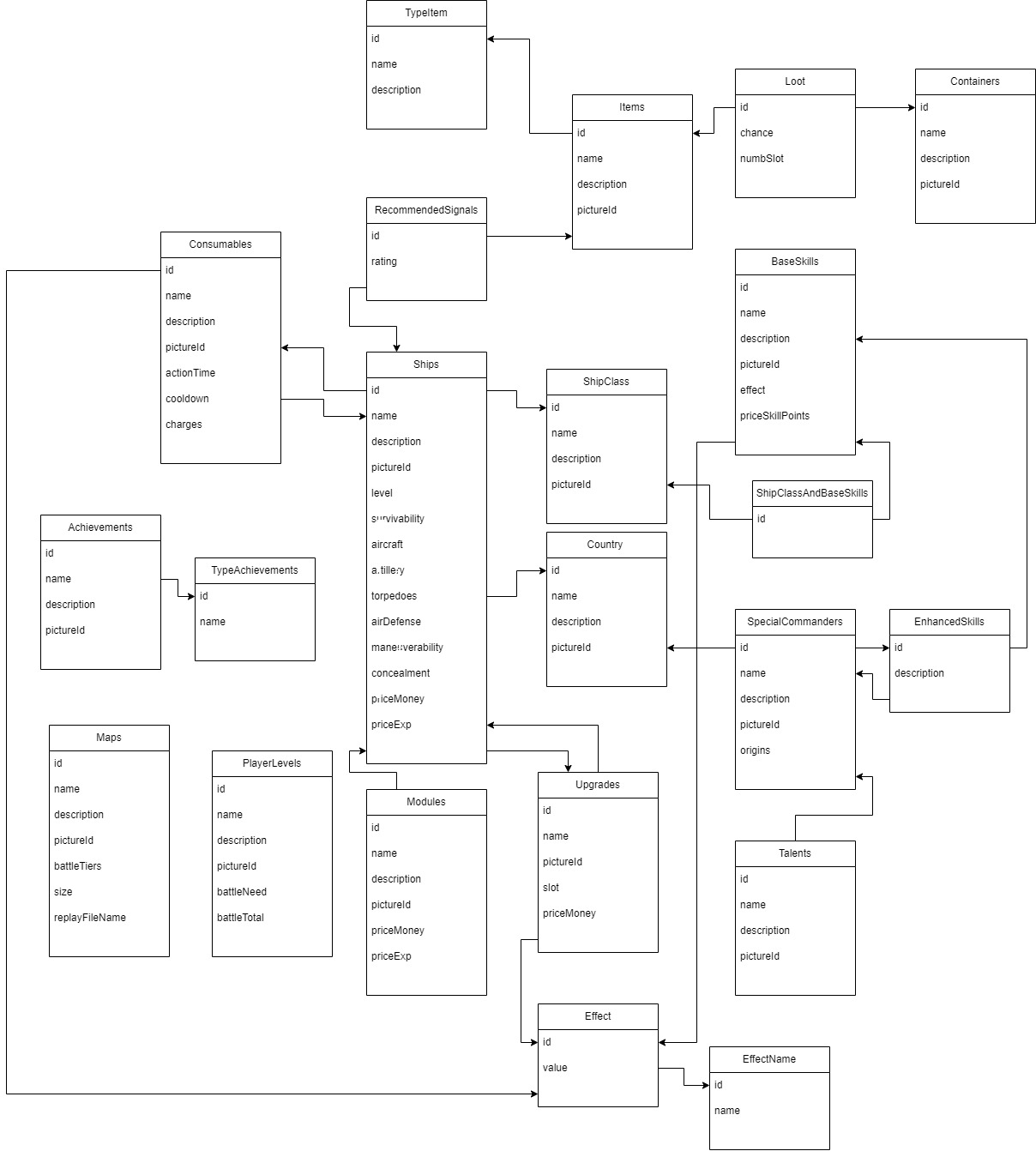


Рисунок . – Инфологическая модель базы данных

## Описание моделей

База данных должна содержать в себе информацию о кораблях, капитанах, способностях, предметах, достижений, эффектов и связанных с ними объектах.

При проектировании базы данных на инфологическом уровне можно выделить следующие сущности и их атрибуты:

1. Achievements: Таблица для содержания id достижения, название, описание и фотографию достижения.
2. TypeAchievements: Таблица для содержания id типа достижения и названия типа достижения.
3. Maps: Таблица для содержания id карты, название карты, описание и связанные с ней поля.
4. Consumables: Таблица для содержания информации о активных навыках, их id времени активации, время на перезарядку и связанных с ними объектах.
5. Ships: Таблица для содержания информации о кораблях, их id, названию, описания, фотографии, характеристиках и связанных с ними объектах.
6. Modules: Таблица для содержания информации о модулях корабля, id модуля, названия модуля, описания, цены и фотографии модуля.
7. RecommendedSignals: Таблица для хранения информации о рекомендованных сигналах.
8. Items: Таблица для хранения информации о предметах, их id название, описание и фотографии.
9. TypeItem: Таблица для хранения типа предмета, содержит id типа, название и описание.
10. ShipClass: Таблица для хранения информации о классе корабля, их id название, описание и фотографии.
11. Country: Таблица для хранения информации о городе корабля, их id название, описание и фотографии.
12. Upgrades: Таблица для хранения информации об улучшениях корабля, их id название, фотографии, цене и месте улучшения.
13. Effect: Таблица для хранения значения эффекта.
14. EffectName: Таблица для хранения названии эффекта.
15. Loot: Таблица для хранения информации о луте, его id, шанса выпадения и номере слота.
16. Containers: Таблица для хранения информации о лутбоксах, их id название, описание и фотографии.
17. Talents: Таблица для хранения информации о талантах, их id название, описание и фотографии.
18. SpecialCommanders: Таблица для хранения информации о талантах, их id название, описание, id фотографии и способе получения.
19. EnhancedSkills: Таблица для хранения информации улучшенных способностей их id и описание.
20. BaseSkills: Таблица для хранения информации о базовых навыках, их id название, описание, id фотографии, эффекте и стоимости использования.

## Сущности и связи

Таблица 1 – Сущности и связи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Отношение (имя на русском языке и латиницей) | Описание | Основные атрибуты | Краткое описание связей с другими отношениями | Особенности |
| ships(корабли) | Содержит информацию о корабле | ship\_id (INT, PRIMARY KEY): Уникальный идентификатор корабла.  name (VARCHAR(200)): Название корабла.  description (VARCHAR(1000)): Описание корабла.  picturepath (VARCHAR(45)): Путь к изображению корабла.  level (INT): Уровень корабла.  survivability (INT): Показатель живучести корабла.  aircraft (INT): Показатель палубной авиации (для игровых целей, если применимо).  artillery (INT): Показатель артиллерии (для игровых целей).  torpedoes (INT): Показатель торпед (для игровых целей, если применимо).  airdefense (INT): Показатель противовоздушной обороны (для игровых целей).  maneuverability (INT): Показатель маневренности корабля (для игровых целей).  concealment (INT): Показатель незаметности корабля.  priceexp (INT): Цена корабля в опыте (внутриигровая валюта).  modules\_id (INT): Идентификатор модулей, связанных с данным кораблём.  modules\_type\_of\_price\_id (INT): Идентификатор типа цены для модулей корабля.  country\_id (INT): Идентификатор страны, к которой принадлежит корабля.  Ship\_type\_id (INT): Идентификатор класса корабля (если применимо, для организации по типам). | Связано с отношениями: country,  upgrades,  modules,  consumables,  recommended-signals,  ship\_type, |  |
| country (страна) | Содержит информацию о стране-производителе корабля | id (INT): идентификатор страны (первичный ключ)  name (VARCHAR(100)): название страны  description (VARCHAR(1000)): описание страны  picturepath (VARCHAR(200)): путь до изображения флага страны | Связано с отношениями: ships,  specialcommanders | id является уникальным идентификатором |
| Special\_commanders(специальные командиры) | Содержит информацию об особых командирах | id (INT): идентификатор командира (первичный ключ)  name (VARCHAR(100)): имя командира  description (VARCHAR(1000)): описание командира  picturepath (VARCHAR(200)): путь до изображения командира  origins(VARCHAR(200)):способ получения  country\_id(INT): айди страны | Связано с отношениями:  Talents,  Country,  enhencedskills | id является уникальным идентификатором |
| ship\_type(тип корабла) | Информация о типах кораблей | id (INT): идентификатор командира (первичный ключ)  name (VARCHAR(100)): имя командира  description (VARCHAR(1000)): описание командира  picturepath (VARCHAR(200)): путь до изображения командира | Связано отношениями:  Ships,  baseskills | id является уникальным идентификатором |
| Baseskills(базовые умения) | Информация о умениях экипажа | id (INT): идентификатор умения(первичный ключ)  name (VARCHAR(100)):название умения  description (VARCHAR(1000)): описание умения  picturepath (VARCHAR(200)): путь до изображения с умением  pricekillpoints(TINYINT(1)):  стоимость очков зачисленных дополнительно  effect\_id INT: id дополнительного эффекта | Связано отношениями:  enhenced\_skills,  ship\_type,  effect | id является уникальным идентификатором |
| Enhencedskills(расширенные умение) | Информация о расширенных умениях | id (INT): идентификатор умения(первичный ключ)  name (VARCHAR(100)):название умения  description (VARCHAR(1000)): описание умения  picturepath (VARCHAR(200)): путь до изображения с умением  priceskillpoints(VARCHAR(1))  pricekillpoints(TINYINT(1)):  effect\_id INT | Связано отношениями: | id является уникальным идентификатором |
| Talents(умение) | Инофрмация о умениях командира | id (INT): идентификатор умений командира(первичный ключ)  name (VARCHAR(100)):название умения  description (VARCHAR(1000)): описание умения  picturepath (VARCHAR(200)): путь до изображения с умением | Связано отношениями: | id является уникальным идентификатором |
| Effectname(название эффекта) | Названия эффектов | id (INT): идентификатор названия эффекта(первичный ключ)  name (VARCHAR(100)):название эффекта | Связано отношениями: | id является уникальным идентификатором |
| Effect(эффект) | Значения эффектов | id (INT): идентификатор умений командира(первичный ключ)  value (VARCHAR(100)):значение эффекта | Связано отношениями: | id является уникальным идентификатором |
| Consumables(расходные материалы) | Расходные материалы | id (INT): идентификатор расходных материалов(первичный ключ)  name (VARCHAR(100)): название расходного материала  description (VARCHAR(1000)): описание расходного материала  actiontime(VARCHAR(5)): время действия расходника  cooldown(VARCHAR(5)): время перезарядки умения  charges(VARCHAR(5)): затраты  picturepath (VARCHAR(200)): путь до изображения расходного материала | Связано отношениями:  Ships, effect | id является уникальным идентификатором |
| Modules(модули) | Модули кораблов | id (INT): идентификатор модулей(первичный ключ)  name (VARCHAR(100)): название модулей  description (VARCHAR(1000)): описание модулей  picturepath (VARCHAR(200)): путь до изображения модуля  pricemoney(INT): стоимость в кредитах  priceexp(INT): Стоимость в опыте | Связано отношениями:  Ships | id является уникальным идентификатором |
| Player\_levels(уровни игроков) | Уровни | id (INT): идентификатор уровней(первичный ключ)  name (VARCHAR(100)): название уровней  description (VARCHAR(1000)): описание уровней  battleneed(INT): необходимое количество опыта до следующего уровня  battletotal(INT): необходимое количество опыта от начала до уровня |  | id является уникальным идентификатором |
| Maps(карты) | Таблица карт игры | id (INT): идентификатор карты(первичный ключ)  name (VARCHAR(100)): название карты  description (VARCHAR(1000)): описание карты  picturepath  (VARCHAR(45)): путь до карты  Battletiers(VARCHAR(45):количество боевых ярусов  size(VARCHAR(45): размер карты  replyfilename(VARCHAR(45)): хранение повторов |  | id является уникальным идентификатором |
| Achievements(достижения) | Таблица достижений в игре | id (INT): идентификатор достижения(первичный ключ)  name (VARCHAR(100)): название достижения  description (VARCHAR(1000)): описание достижения  picturepath  (VARCHAR(45)): путь до изображения достижения | Связано отношениями:  Typeachivements | id является уникальным идентификатором |
| Typeachivements(тип достижения) | Таблица типов достижений в игре | id (INT): идентификатор тиап достижения(первичный ключ)  name (VARCHAR(100)): название типа достижения | Связано отношениями:  Achievements | id является уникальным идентификатором |
| Recommendedsignals(рекомендуемые сигналы) | Таблица флагов, наиболее подоходящих кораблу | id (INT): идентификатор типа сигнала(первичный ключ)  raiting (INT): значения, для флагов | Связано отношениями: | id является уникальным идентификатором |
| Items(предметы) | Предметы, доступные для выигрыша | id (INT): идентификатор предмета(первичный ключ)  name (VARCHAR(100)): название предметов  description (VARCHAR(1000)): описание предметов  picturepath  (VARCHAR(45)): путь до изображения предметов | Связано отношениями:  Type\_item, loot, Recommendedsignals | id является уникальным идентификатором |
| Typeitem(типы предметов) | Типы предметов | id (INT): идентификатор типа предмета(первичный ключ)  name (VARCHAR(100)): названия типов предметов  description (VARCHAR(1000)): описания типов предметов | Связано отношениями:  items | id является уникальным идентификатором |
| Loot(выигрыш) | Шансы выпадения предметов в контейнерах | id (INT): идентификатор шансов (первичный ключ)  chance(TYNINT): значение шанса | Связано отношениями:  Items,  Containers | id является уникальным идентификатором |
| Containers(контейнеры) | Контейнеры, разыгрываеммые в игре | id (INT): идентификатор типа контейнера(первичный ключ)  name (VARCHAR(100)): названия контейнеров  description (VARCHAR(1000)): описания контейнеров  picturepath  (VARCHAR(45)): путь до изображения предметов | Связано отношениями:  Loot | id является уникальным идентификатором |
| Upgrades(улучшения) | Улучшения, доступные для корабла | id (INT): идентификатор типа контейнера(первичный ключ)  name (VARCHAR(100)): названия контейнеров  description (VARCHAR(1000)): описания контейнеров  picturepath  (VARCHAR(45)): путь до изображений улучшений  Slot(TINYINT(1)): Слот, в который можно произвести улучшение  pricemoney(INT): стоимость улучшения | Связано отношениями:  Effect,  ships | id является уникальным идентификатором |

## Особенности нормализации

Особенности нормализации:

* Разбиение таблицы с информацией о достижениях на TypeAchievements позволяет контролировать добавление и изменение данных, так как теперь нельзя записать данные, не находящиеся в данной таблице.
* Разбиение таблицы с информацией о корабле на ShipClass, Country и Modules позволяет контролировать добавление и изменение данных, так как теперь нельзя записать данные, не находящиеся в данных таблицах.
* Разбиение таблицы с информацией о предметах на TypeItem позволяет контролировать добавление и изменение данных, так как теперь нельзя записать данные, не находящиеся в данной таблице.

# Подробное описание бизнес-логики

Изображение выглядит как текст, рукописный текст, диаграмма, Шрифт

Автоматически созданное описание

Рисунок 4.1 – Физическая модель базы данных.

## Хранимые процедуры

### Процедура на получение списков стран.

Процедура get\_countries возвращает список стран при получении запроса.

DROP procedure IF EXISTS `get\_countries`;

DELIMITER $$

USE `mydb`$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `get\_countries`()

BEGIN

SELECT `id`,`name`,`picturepath` FROM `countries`;

END$$

DELIMITER ;

### Процедура на получение корабля

Процедура get\_ship возвращает элементы из таблицы корабля по заданному id.

DROP procedure IF EXISTS `get\_ship`;

DELIMITER $$

USE `mydb`$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `get\_ship`(IN `object\_id` INT)

BEGIN

SELECT `ships`.`name` AS `ship\_name`, `ships`.`description` AS `ship\_description`, `ships`.`picturepath` AS `ship\_picturepath`,`ships`.`level` AS `ship\_level`,`ships`.`survivability` AS `ship\_survivability`,`ships`.`aircraft` AS `ship\_aircraft`,`ships`.`artillery` AS `ship\_artillery`,`ships`.`torpedoes` AS `ship\_torpedoes`,`ships`.`airdefense` AS `ship\_airdefense`,`ships`.`maneuverability` AS `ship\_maneuverability`,`ships`.`concealment` AS `ship\_concealment`,`ships`.`pricemoney` AS `ship\_pricemoney`,`ships`.`priceexp` AS `ship\_priceexp`,

`ship\_class`.`name` AS `ship\_class\_name`,`ship\_class`.`picturepath` AS `ship\_class\_picturepath`,

`modules`.`name` AS `modules\_name`, `modules`.`description` AS `modules\_description`, `modules`.`picturepath` AS `modules\_picturepath`,`modules`.`pricemoney` AS `modules\_pricemoney`,`modules`.`priceexp` AS `modules\_priceexp`

FROM `ships`

LEFT JOIN `ship\_class` ON `ships`.`shipclass\_id` = `ship\_class`.`id`

LEFT JOIN `modules` ON `modules`.`ships\_id` = `ships`.`id`

WHERE `ships`.`id` = `object\_id`;

END$$

DELIMITER ;

### Процедура на получение командира

Процедура get\_commander возвращает элементы из таблицы командиров по заданному id.

DELIMITER $$

USE `mydb`$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `get\_commander`(IN `object\_id` INT)

BEGIN

SELECT `special\_commanders`.`name` AS `special\_commanders\_name`,`special\_commanders`.`description`AS `special\_commanders\_description`,`special\_commanders`.`picturepath`AS `special\_commanders\_picturepath`, `special\_commanders`.`origins`AS `special\_commanders\_origins`,

`talents`.`name` AS `talents\_name`,`talents`.`description` AS `talents\_description`,`talents`.`picturepath` AS `talents\_picturepath`

FROM `special\_commanders`

LEFT JOIN

`talents` ON `talents`.`specialcommanders\_id` = `special\_commanders`.`id`

WHERE `special\_commanders`.`id` = `object\_id`;

END$$

DELIMITER ;

### Процедура на получение командира

Процедура get\_container возвращает элементы из таблицы лутбоксов по заданному id.

DROP procedure IF EXISTS `get\_container`;

DELIMITER $$

USE `mydb`$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `get\_container`(IN `object\_id` INT)

BEGIN

SELECT `containers`.`name` AS `container\_name`, `containers`.`description` AS `container\_description`, `containers`.`picturepath` AS `container\_picturepath`,

`loot`.`chance` AS `loot\_chance`,

`items`.`name` AS `item\_name`,

`type\_item`.`name` AS `type\_item\_name`

FROM `containers`

JOIN `loot` ON `loot`.`containers\_id` = `containers`.`id`

JOIN `items` ON `loot`.`items\_id` = `items`.`id`

JOIN `type\_item` ON `items`.`typeitem\_id` = `type\_item`.`id`

WHERE `containers`.`id` = `object\_id`;

END$$

DELIMITER ;

## Основные алгоритмы ПС

* + 1. Алгоритмы вывода информации

Алгоритм GetObjectOfListPage выводит списки игровых объектов.

Алгоритм GetBasePartOfObjectPage выводит базовую часть страницы любого игрового элемента.

Алгоритм GetObjectOfListPage:

public static ScrollView GetObjectOfListPage(GeneralConstant.GeneralObjectFromDB typeObjectShow, TapGestureRecognizer funcGoToNextPage, int? contryId = null)

{

var request = new RequestListMessage()

{

Action = GeneralConstant.GeneralServerActions.Get,

TopicFromServer = RabbitMQ.TopicFromServer,

ObjectName = typeObjectShow,

CountryId = contryId

};

RabbitMQ.Publisher.SendMessage(request.ToJson());

var json = RabbitMQ.Consumer.GetMessage();

var messageList = JsonConvert.DeserializeObject<DBListMessage>(json);

var vStack = new VerticalStackLayout()

{

HorizontalOptions = LayoutOptions.Center,

};

var namePageText = "";

switch (typeObjectShow)

{

case GeneralConstant.GeneralObjectFromDB.Countries:

namePageText = "Список стран";

break;

case GeneralConstant.GeneralObjectFromDB.Ships:

namePageText = "Список кораблей";

break;

case GeneralConstant.GeneralObjectFromDB.Commanders:

namePageText = "Список уникальных командиров";

break;

case GeneralConstant.GeneralObjectFromDB.Maps:

namePageText = "Список карт";

break;

case GeneralConstant.GeneralObjectFromDB.PlayerLevels:

namePageText = "Список уровней игрока";

break;

case GeneralConstant.GeneralObjectFromDB.Achievements:

namePageText = "Список достижений";

break;

case GeneralConstant.GeneralObjectFromDB.Containers:

namePageText = "Список контейнеров";

break;

}

var namePage = new Label()

{

HorizontalOptions = LayoutOptions.Center,

FontSize = 20,

Text = namePageText,

};

vStack.Add(namePage);

foreach (var message in messageList.ItemList)

{

var vObjectStack = new VerticalStackLayout()

{

HorizontalOptions = LayoutOptions.Center,

};

if (message.Name != null)

{

vObjectStack.Add(new Label()

{

HorizontalOptions = LayoutOptions.Center,

Text = message.Name,

});

}

if (message.PicturePath != null)

{

var image = new Image()

{

Source = GetUrlImageFromPath(message.PicturePath, typeObjectShow),

WidthRequest = PagesConstants.SIZE\_IMAGE\_IN\_LIST\_PAGE,

HeightRequest = PagesConstants.SIZE\_IMAGE\_IN\_LIST\_PAGE,

BindingContext = message.Id

};

image.GestureRecognizers.Add(funcGoToNextPage);

vObjectStack.Add(image);

}

vStack.Add(vObjectStack);

}

var scrollView = new ScrollView

{

Content = vStack

};

return scrollView;

}

Алгоритм GetBasePartOfObjectPage :

public static VerticalStackLayout GetBasePartOfObjectPage(DBObjectMessage message, GeneralConstant.GeneralObjectFromDB typeObjectShow)

{

var vStack = new VerticalStackLayout()

{

Padding = 10,

HorizontalOptions = LayoutOptions.Center,

};

if (message.Name != null)

{

vStack.Add(new Label()

{

HorizontalOptions = LayoutOptions.Center,

Text = message.Name,

});

}

if (message.PicturePath != null)

{

vStack.Add(new Image()

{

Source = GetUrlImageFromPath(message.PicturePath, typeObjectShow),

WidthRequest = PagesConstants.SIZE\_IMAGE\_IN\_OBJECT\_PAGE,

HeightRequest = PagesConstants.SIZE\_IMAGE\_IN\_OBJECT\_PAGE,

});

}

if (message.Description != null)

{

vStack.Add(new Label()

{

Text = message.Description,

});

}

return vStack;

}

* + 1. Алгоритм получения данных о корабле из MySQL

Алгоритм GetShip получает данные из таблицы ship и связанных таблиц и отправляет их в формате json клиенту.

Алгоритм GetShip:

case GeneralConstant.GeneralObjectFromDB.Ship:

dataReader = mySQLConnector.GetDataByIdUseDBFunc("get\_ship", messageObjectGet.ObjectId.Value);

var messageShipSend = new DBShipMessage();

if (dataReader.Read())

{

messageShipSend.Name = dataReader.GetString("ship\_name");

messageShipSend.Description = dataReader.GetString("ship\_description");

messageShipSend.PicturePath = dataReader.GetString("ship\_picturepath");

messageShipSend.Level = dataReader.IsDBNull(dataReader.GetOrdinal("ship\_level")) ? 0 : dataReader.GetInt32("ship\_level");

messageShipSend.Survivability = dataReader.IsDBNull(dataReader.GetOrdinal("ship\_survivability")) ? 0 : dataReader.GetInt32("ship\_survivability");

messageShipSend.Aircraft = dataReader.IsDBNull(dataReader.GetOrdinal("ship\_aircraft")) ? 0 : dataReader.GetInt32("ship\_aircraft");

messageShipSend.Artillery = dataReader.IsDBNull(dataReader.GetOrdinal("ship\_artillery")) ? 0 : dataReader.GetInt32("ship\_artillery");

messageShipSend.Torpedoes = dataReader.IsDBNull(dataReader.GetOrdinal("ship\_torpedoes")) ? 0 : dataReader.GetInt32("ship\_torpedoes");

messageShipSend.Airdefense = dataReader.IsDBNull(dataReader.GetOrdinal("ship\_airdefense")) ? 0 : dataReader.GetInt32("ship\_airdefense");

messageShipSend.Maneuverability = dataReader.IsDBNull(dataReader.GetOrdinal("ship\_maneuverability")) ? 0 : dataReader.GetInt32("ship\_maneuverability");

messageShipSend.Concealment = dataReader.IsDBNull(dataReader.GetOrdinal("ship\_concealment")) ? 0 : dataReader.GetInt32("ship\_concealment");

messageShipSend.PriceMoney = dataReader.IsDBNull(dataReader.GetOrdinal("ship\_pricemoney")) ? 0 : dataReader.GetInt32("ship\_pricemoney");

messageShipSend.PriceExp = dataReader.IsDBNull(dataReader.GetOrdinal("ship\_priceexp")) ? 0 : dataReader.GetInt32("ship\_priceexp");

messageShipSend.ShipClassName = dataReader.GetString("ship\_class\_name");

messageShipSend.ShipClassPicturePath = dataReader.GetString("ship\_class\_picturepath");

var modulesFromShip = new DBShipMessage.DBModules();

modulesFromShip.Name = dataReader.IsDBNull(dataReader.GetOrdinal("modules\_name")) ? null : dataReader.GetString("modules\_name");

modulesFromShip.Description = dataReader.IsDBNull(dataReader.GetOrdinal("modules\_description")) ? null : dataReader.GetString("modules\_description");

modulesFromShip.PicturePath = dataReader.IsDBNull(dataReader.GetOrdinal("modules\_picturepath")) ? null : dataReader.GetString("modules\_picturepath");

modulesFromShip.PriceMoney = dataReader.IsDBNull(dataReader.GetOrdinal("modules\_pricemoney")) ? 0 : dataReader.GetInt32("modules\_pricemoney");

modulesFromShip.PriceExp = dataReader.IsDBNull(dataReader.GetOrdinal("modules\_priceexp")) ? 0 : dataReader.GetInt32("modules\_priceexp");

if(modulesFromShip.Name != null)

{

messageShipSend.ModulesList.Add(modulesFromShip);

}

}

while (dataReader.Read())

{

var modulesFromShip = new DBShipMessage.DBModules();

modulesFromShip.Name = dataReader.IsDBNull(dataReader.GetOrdinal("modules\_name")) ? null : dataReader.GetString("modules\_name");

modulesFromShip.Description = dataReader.IsDBNull(dataReader.GetOrdinal("modules\_description")) ? null : dataReader.GetString("modules\_description");

modulesFromShip.PicturePath = dataReader.IsDBNull(dataReader.GetOrdinal("modules\_picturepath")) ? null : dataReader.GetString("modules\_picturepath");

modulesFromShip.PriceMoney = dataReader.IsDBNull(dataReader.GetOrdinal("modules\_pricemoney")) ? 0 : dataReader.GetInt32("modules\_pricemoney");

modulesFromShip.PriceExp = dataReader.IsDBNull(dataReader.GetOrdinal("modules\_priceexp")) ? 0 : dataReader.GetInt32("modules\_priceexp");

if (modulesFromShip.Name != null)

{

messageShipSend.ModulesList.Add(modulesFromShip);

}

}

dataReader.Close();

if (!publishers.TryGetValue(basePartOfMessage.TopicFromServer, out publisher))

{

break;

}

json = messageShipSend.ToJson();

Console.WriteLine("Send: " + json);

Console.WriteLine();

publisher.SendMessage(json);

break;

# Тестирование базы данных и приложений системы

## Тестирование хранимых процедур

### процедура: `get\_country`

Исходные данные: вызов процедуры с передачей id страны.

Ожидаемый результат: получение информации о стране.

Полученный результат:



Рисунок 5.1.1 – Вывод информации о стране

## Тестирование программного средства

### Вывод информации о стране

Исходный набор данных: запрос на получение данных о стране.

Ожидаемый результат: экран с информацией о стране.

Полученный результат:

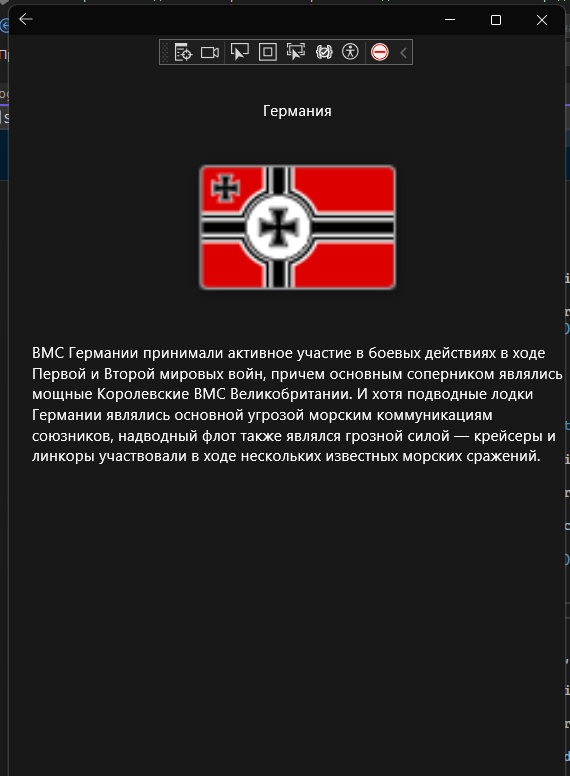


Рисунок 5.2.1 – Вывод информации о стране

### Вывод информации о корабле

Исходный набор данных: запрос на получение данных о корабле.

Ожидаемый результат: экран с информацией о корабле.

Полученный результат:

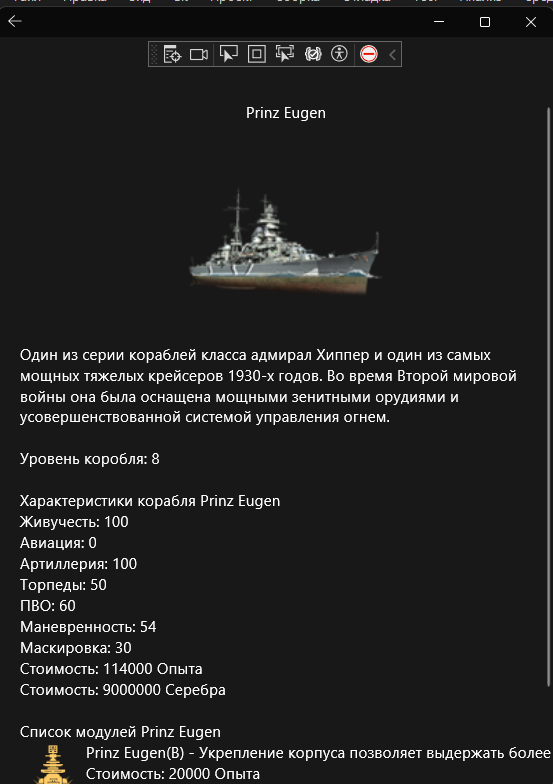


Рисунок 5.2.1 – Вывод информации о корабле

### Вывод информации о кораблях

Исходный набор данных: запрос на получение данных о кораблях Германии.

Ожидаемый результат: экран с информацией о кораблях Германии.

Полученный результат:



Рисунок 5.2.2 – Вывод информации о кораблях Германии

### Вывод информации о командирах

Исходный набор данных: запрос на получение данных о командирах Германии.

Ожидаемый результат: экран с информацией о командирах Германии.

Полученный результат:



Рисунок 5.2.3 – Вывод информации о корабле

# Заключение

В рамках курсового проекта была разработана база данных для справочника по игре World of Warships. Данная разработка позволяет хранить и обеспечивать управление информацией о кораблях, капитанах, способностях, предметах, достижений, эффектов и связанных с ними объектах. Была обеспечена функциональность по вводу, изменению и извлечению данных, а также поддерживать связи и целостность данных.

Также было разработано приложение для взаимодействия с базой данных. Пользователи могут искать корабли, командирах, просматривать информацию о них. Приложение обеспечивает удобный пользовательский интерфейс для навигации и взаимодействия с приложением.

В процессе разработки исследовались различные варианты решения задач, связанных со справочником. Была изучена модель базы данных, специфичная для справочника, а также язык запросов для работы с базой данных. В данном проекте в качестве системы управления базами данных (СУБД) был применен MySQL Server, который обеспечивает надежное хранение и эффективное извлечение данных для справочника.

В результате были достигнуты поставленные цели, и приложение взаимодействует успешно с базой данных, обеспечивая надежное хранение и управление информацией, а также удобный интерфейс для пользователей. Проект прошел все тестирования и готов для использования.

# Список использованной литературы

1. Куликов, Святослав. Реляционные базы данных в примерах. EPAM Systems, 2020–2023. – 424 с.
2. Куликов, Святослав. Работа с MySQL, MS SQL Server и Oracle в примерах. EPAM Systems, 2020–2023. – 590 с.
3. Б. Новиков, Е. Горшкова, Н. Графеева. Основы технологий баз данных: учебное пособие. ДМК Пресс, 2020. – 582 с.
4. Руководство по Entity Framework 8 [Электронный ресурс] – Режим доступа: <https://metanit.com/sharp/efcore>
5. Руководство по ASP.NET Core 8 [Электронный ресурс] – Режим доступа: <https://metanit.com/sharp/aspnet6/>
6. Руководство по ASP.NET Core MVC [Электронный ресурс] – Режим доступа: <https://metanit.com/sharp/aspnetmvc/>

Приложение А

(обязательное)

Схема базы данных на языке SQL

-- MySQL Script generated by MySQL Workbench

-- Tue May 7 12:06:31 2024

-- Model: New Model Version: 1.0

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;

USE `mydb` ;

-- -----------------------------------------------------

-- Table `mydb`.`type\_item`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`type\_item` (

`id` INT NOT NULL,

`name` VARCHAR(100) NOT NULL,

`description` VARCHAR(1000) NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`items`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`items` (

`id` INT NOT NULL,

`name` VARCHAR(100) NULL,

`description` VARCHAR(1000) NOT NULL,

`picturepath` VARCHAR(45) NULL,

`typeitem\_id` INT NOT NULL,

PRIMARY KEY (`id`, `typeitem\_id`),

INDEX `fk\_items\_typeitem1\_idx` (`typeitem\_id` ASC) VISIBLE,

CONSTRAINT `fk\_items\_typeitem1`

FOREIGN KEY (`typeitem\_id`)

REFERENCES `mydb`.`type\_item` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`containers`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`containers` (

`id` INT NOT NULL,

`name` VARCHAR(100) NOT NULL,

`description` VARCHAR(1000) NULL,

`picturepath` VARCHAR(45) NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`loot`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`loot` (

`id` INT NOT NULL,

`chance` TINYINT NOT NULL,

`containers\_id` INT NOT NULL,

`items\_id` INT NOT NULL,

PRIMARY KEY (`id`, `containers\_id`, `items\_id`),

INDEX `fk\_loot\_containers\_idx` (`containers\_id` ASC) VISIBLE,

INDEX `fk\_loot\_items1\_idx` (`items\_id` ASC) VISIBLE,

CONSTRAINT `fk\_loot\_containers`

FOREIGN KEY (`containers\_id`)

REFERENCES `mydb`.`containers` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_loot\_items1`

FOREIGN KEY (`items\_id`)

REFERENCES `mydb`.`items` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`countries`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`countries` (

`id` INT NOT NULL,

`name` VARCHAR(100) NOT NULL,

`description` VARCHAR(1000) NULL,

`picturepath` VARCHAR(45) NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`ship\_class`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`ship\_class` (

`id` INT NOT NULL,

`name` VARCHAR(100) NOT NULL,

`description` VARCHAR(1000) NULL,

`picturepath` VARCHAR(45) NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`ships`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`ships` (

`id` INT NOT NULL,

`name` VARCHAR(100) NOT NULL,

`description` VARCHAR(1000) NULL,

`picturepath` VARCHAR(45) NULL,

`level` INT NULL,

`survivability` INT NULL,

`aircraft` INT NULL,

`artillery` INT NULL,

`torpedoes` INT NULL,

`airdefense` INT NULL,

`maneuverability` INT NULL,

`concealment` INT NULL,

`pricemoney` INT NULL,

`priceexp` INT NULL,

`countries\_id` INT NOT NULL,

`shipclass\_id` INT NOT NULL,

PRIMARY KEY (`id`, `countries\_id`, `shipclass\_id`),

INDEX `fk\_ships\_country1\_idx` (`countries\_id` ASC) VISIBLE,

INDEX `fk\_ships\_shipclass1\_idx` (`shipclass\_id` ASC) VISIBLE,

CONSTRAINT `fk\_ships\_country1`

FOREIGN KEY (`countries\_id`)

REFERENCES `mydb`.`countries` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_ships\_shipclass1`

FOREIGN KEY (`shipclass\_id`)

REFERENCES `mydb`.`ship\_class` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`recommended\_signals`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`recommended\_signals` (

`id` INT NOT NULL,

`raiting` INT NOT NULL,

`items\_id` INT NOT NULL,

`ships\_id` INT NOT NULL,

PRIMARY KEY (`id`, `items\_id`, `ships\_id`),

INDEX `fk\_recommendedsignals\_items1\_idx` (`items\_id` ASC) VISIBLE,

INDEX `fk\_recommendedsignals\_ships1\_idx` (`ships\_id` ASC) VISIBLE,

CONSTRAINT `fk\_recommendedsignals\_items1`

FOREIGN KEY (`items\_id`)

REFERENCES `mydb`.`items` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_recommendedsignals\_ships1`

FOREIGN KEY (`ships\_id`)

REFERENCES `mydb`.`ships` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`consumables`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`consumables` (

`id` INT NOT NULL,

`name` VARCHAR(100) NOT NULL,

`description` VARCHAR(1000) NULL,

`actiontime` VARCHAR(5) NULL,

`cooldown` VARCHAR(5) NULL,

`charges` VARCHAR(5) NULL,

`picturepath` VARCHAR(45) NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`consumables\_has\_ships`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`consumables\_has\_ships` (

`consumables\_id` INT NOT NULL,

`ships\_id` INT NOT NULL,

PRIMARY KEY (`consumables\_id`, `ships\_id`),

INDEX `fk\_consumables\_has\_ships\_ships1\_idx` (`ships\_id` ASC) VISIBLE,

INDEX `fk\_consumables\_has\_ships\_consumables1\_idx` (`consumables\_id` ASC) VISIBLE,

CONSTRAINT `fk\_consumables\_has\_ships\_consumables1`

FOREIGN KEY (`consumables\_id`)

REFERENCES `mydb`.`consumables` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_consumables\_has\_ships\_ships1`

FOREIGN KEY (`ships\_id`)

REFERENCES `mydb`.`ships` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`type\_achievements`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`type\_achievements` (

`id` INT NOT NULL,

`name` VARCHAR(100) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`achievements`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`achievements` (

`id` INT NOT NULL,

`name` VARCHAR(100) NOT NULL,

`description` VARCHAR(1000) NULL,

`picturepath` VARCHAR(45) NULL,

`typeachivements\_id` INT NOT NULL,

PRIMARY KEY (`id`, `typeachivements\_id`),

INDEX `fk\_achievements\_typeachivements1\_idx` (`typeachivements\_id` ASC) VISIBLE,

CONSTRAINT `fk\_achievements\_typeachivements1`

FOREIGN KEY (`typeachivements\_id`)

REFERENCES `mydb`.`type\_achievements` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`maps`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`maps` (

`id` INT NOT NULL,

`name` VARCHAR(100) NOT NULL,

`description` VARCHAR(1000) NULL,

`picturepath` VARCHAR(45) NULL,

`battletiers` VARCHAR(45) NULL,

`size` VARCHAR(45) NULL,

`replyfilename` VARCHAR(45) NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`player\_levels`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`player\_levels` (

`id` INT NOT NULL,

`name` VARCHAR(100) NOT NULL,

`description` VARCHAR(1000) NULL,

`picturepath` VARCHAR(45) NULL,

`battleneed` TINYINT NULL,

`battletotal` TINYINT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`modules`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`modules` (

`id` INT NOT NULL,

`name` VARCHAR(45) NULL,

`description` VARCHAR(1000) NULL,

`picturepath` VARCHAR(45) NULL,

`pricemoney` INT NULL,

`priceexp` INT NULL,

`ships\_id` INT NOT NULL,

PRIMARY KEY (`id`, `ships\_id`),

INDEX `fk\_modules\_ships1\_idx` (`ships\_id` ASC) VISIBLE,

CONSTRAINT `fk\_modules\_ships1`

FOREIGN KEY (`ships\_id`)

REFERENCES `mydb`.`ships` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`effect\_name`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`effect\_name` (

`id` INT NOT NULL,

`name` VARCHAR(100) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`effect`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`effect` (

`id` INT NOT NULL,

`value` TINYINT NOT NULL,

`consumables\_id` INT NOT NULL,

`effectname\_id` INT NOT NULL,

PRIMARY KEY (`id`, `consumables\_id`, `effectname\_id`),

INDEX `fk\_effect\_consumables1\_idx` (`consumables\_id` ASC) VISIBLE,

INDEX `fk\_effect\_effectname1\_idx` (`effectname\_id` ASC) VISIBLE,

CONSTRAINT `fk\_effect\_consumables1`

FOREIGN KEY (`consumables\_id`)

REFERENCES `mydb`.`consumables` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_effect\_effectname1`

FOREIGN KEY (`effectname\_id`)

REFERENCES `mydb`.`effect\_name` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`upgrades`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`upgrades` (

`id` INT NOT NULL,

`name` VARCHAR(100) NOT NULL,

`discription` VARCHAR(1000) NULL,

`picturepath` VARCHAR(45) NULL,

`slot` TINYINT(1) NULL,

`pricemoney` INT NULL,

`effect\_id` INT NOT NULL,

`effect\_consumables\_id` INT NOT NULL,

`effect\_effectname\_id` INT NOT NULL,

PRIMARY KEY (`id`, `effect\_id`, `effect\_consumables\_id`, `effect\_effectname\_id`),

INDEX `fk\_upgrades\_effect1\_idx` (`effect\_id` ASC, `effect\_consumables\_id` ASC, `effect\_effectname\_id` ASC) VISIBLE,

CONSTRAINT `fk\_upgrades\_effect1`

FOREIGN KEY (`effect\_id` , `effect\_consumables\_id` , `effect\_effectname\_id`)

REFERENCES `mydb`.`effect` (`id` , `consumables\_id` , `effectname\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`ships\_has\_upgrades`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`ships\_has\_upgrades` (

`ships\_id` INT NOT NULL,

`ships\_modules\_id` INT NOT NULL,

`ships\_modules\_type\_of\_price\_id` INT NOT NULL,

`upgrades\_id` INT NOT NULL,

PRIMARY KEY (`ships\_id`, `ships\_modules\_id`, `ships\_modules\_type\_of\_price\_id`, `upgrades\_id`),

INDEX `fk\_ships\_has\_upgrades\_upgrades1\_idx` (`upgrades\_id` ASC) VISIBLE,

INDEX `fk\_ships\_has\_upgrades\_ships1\_idx` (`ships\_id` ASC, `ships\_modules\_id` ASC, `ships\_modules\_type\_of\_price\_id` ASC) VISIBLE,

CONSTRAINT `fk\_ships\_has\_upgrades\_ships1`

FOREIGN KEY (`ships\_id`)

REFERENCES `mydb`.`ships` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_ships\_has\_upgrades\_upgrades1`

FOREIGN KEY (`upgrades\_id`)

REFERENCES `mydb`.`upgrades` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`special\_commanders`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`special\_commanders` (

`id` INT NOT NULL,

`name` VARCHAR(100) NOT NULL,

`description` VARCHAR(1000) NULL,

`picturepath` VARCHAR(45) NULL,

`origins` VARCHAR(100) NULL,

`countries\_id` INT NOT NULL,

PRIMARY KEY (`id`, `countries\_id`),

INDEX `fk\_specialcommanders\_country1\_idx` (`countries\_id` ASC) VISIBLE,

CONSTRAINT `fk\_specialcommanders\_country1`

FOREIGN KEY (`countries\_id`)

REFERENCES `mydb`.`countries` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`base\_skills`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`base\_skills` (

`id` INT NOT NULL,

`name` VARCHAR(100) NOT NULL,

`description` VARCHAR(1000) NULL,

`picturepath` VARCHAR(45) NULL,

`priceskillpoints` TINYINT(1) NULL,

`effect\_id` INT NOT NULL,

`effect\_consumables\_id` INT NOT NULL,

`effect\_effectname\_id` INT NOT NULL,

PRIMARY KEY (`id`, `effect\_id`, `effect\_consumables\_id`, `effect\_effectname\_id`),

INDEX `fk\_baseskills\_effect1\_idx` (`effect\_id` ASC, `effect\_consumables\_id` ASC, `effect\_effectname\_id` ASC) VISIBLE,

CONSTRAINT `fk\_baseskills\_effect1`

FOREIGN KEY (`effect\_id` , `effect\_consumables\_id` , `effect\_effectname\_id`)

REFERENCES `mydb`.`effect` (`id` , `consumables\_id` , `effectname\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`shipclass\_has\_baseskills`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`shipclass\_has\_baseskills` (

`shipclass\_id` INT NOT NULL,

`baseskills\_id` INT NOT NULL,

PRIMARY KEY (`shipclass\_id`, `baseskills\_id`),

INDEX `fk\_shipclass\_has\_baseskills\_baseskills1\_idx` (`baseskills\_id` ASC) VISIBLE,

INDEX `fk\_shipclass\_has\_baseskills\_shipclass1\_idx` (`shipclass\_id` ASC) VISIBLE,

CONSTRAINT `fk\_shipclass\_has\_baseskills\_shipclass1`

FOREIGN KEY (`shipclass\_id`)

REFERENCES `mydb`.`ship\_class` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_shipclass\_has\_baseskills\_baseskills1`

FOREIGN KEY (`baseskills\_id`)

REFERENCES `mydb`.`base\_skills` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`enhenced\_skills`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`enhenced\_skills` (

`id` INT NOT NULL,

`description` VARCHAR(1000) NOT NULL,

`baseskills\_id` INT NOT NULL,

`baseskills\_effect\_id` INT NOT NULL,

`baseskills\_effect\_consumables\_id` INT NOT NULL,

`baseskills\_effect\_effectname\_id` INT NOT NULL,

PRIMARY KEY (`id`, `baseskills\_id`, `baseskills\_effect\_id`, `baseskills\_effect\_consumables\_id`, `baseskills\_effect\_effectname\_id`),

INDEX `fk\_enhencedskills\_baseskills1\_idx` (`baseskills\_id` ASC, `baseskills\_effect\_id` ASC, `baseskills\_effect\_consumables\_id` ASC, `baseskills\_effect\_effectname\_id` ASC) VISIBLE,

CONSTRAINT `fk\_enhencedskills\_baseskills1`

FOREIGN KEY (`baseskills\_id` , `baseskills\_effect\_id` , `baseskills\_effect\_consumables\_id` , `baseskills\_effect\_effectname\_id`)

REFERENCES `mydb`.`base\_skills` (`id` , `effect\_id` , `effect\_consumables\_id` , `effect\_effectname\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`talents`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`talents` (

`id` INT NOT NULL,

`name` VARCHAR(100) NOT NULL,

`description` VARCHAR(1000) NULL,

`picturepath` VARCHAR(45) NULL,

`specialcommanders\_id` INT NOT NULL,

PRIMARY KEY (`id`, `specialcommanders\_id`),

INDEX `fk\_talents\_specialcommanders1\_idx` (`specialcommanders\_id` ASC) VISIBLE,

CONSTRAINT `fk\_talents\_specialcommanders1`

FOREIGN KEY (`specialcommanders\_id`)

REFERENCES `mydb`.`special\_commanders` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`enhencedskills\_has\_specialcommanders`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`enhencedskills\_has\_specialcommanders` (

`enhencedskills\_id` INT NOT NULL,

`specialcommanders\_id` INT NOT NULL,

PRIMARY KEY (`enhencedskills\_id`, `specialcommanders\_id`),

INDEX `fk\_enhencedskills\_has\_specialcommanders\_specialcommanders1\_idx` (`specialcommanders\_id` ASC) VISIBLE,

INDEX `fk\_enhencedskills\_has\_specialcommanders\_enhencedskills1\_idx` (`enhencedskills\_id` ASC) VISIBLE,

CONSTRAINT `fk\_enhencedskills\_has\_specialcommanders\_enhencedskills1`

FOREIGN KEY (`enhencedskills\_id`)

REFERENCES `mydb`.`enhenced\_skills` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_enhencedskills\_has\_specialcommanders\_specialcommanders1`

FOREIGN KEY (`specialcommanders\_id`)

REFERENCES `mydb`.`special\_commanders` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;