

Кафедра инженерной кибернетики

Квалификация (степень): **магистр**

Направление подготовки: 09.04.03 «Прикладная информатика»

Профиль (программа) «**Инновационные ИТ проекты**»

Дисциплина: Современные инструментальные средства разработки

КУРСОВАЯ РАБОТА

на тему

«Проектирование и разработка клиент – серверного
приложения «Бэнг!»»»

Выполнили:

Альмухаметова Т. С.

Андреев С. Д.

Ефремова М. С.

Москвитина Л. С.

Новицкий Д. А.

Резников К. П.

Группа: МПИ-20-4-2

Проверил: Тарханов И. А.

Оценка: _____

Дата: _____

Москва 2021

ОГЛАВЛЕНИЕ

1. Введение	4
1.1. Тема работы.....	4
1.2. Цель работы.....	4
1.3. Задачи работы	4
1.4. Актуальность темы	5
2. Требования к системе	6
2.1. Описание функциональных требований.....	6
2.2. Описание нефункциональных требований.....	8
3. Архитектура приложения.	13
4. Эскизное проектирование элементов приложения.	14
4.1. Эскизы элементов одиночной игры	14
4.2. Эскизы элементов онлайн игры	17
4.3. Эскизы дополнительных элементов.....	22
5. Разработка БАЗЫ ДАННЫХ	27
6. Test case.....	31
6.1. Тест-кейс №1. Главное меню.....	31
6.2. Тест-кейс №2. Регистрация.....	32
6.3. Тест-кейс №3. Авторизация.....	32
6.4. Тест-кейс №4. Подготовка к онлайн игре	33
6.5. Тест-кейс №5. Подготовка к одиночной игре.....	35
6.6. Тест-кейс №6. Бэнглопедия	36
6.7. Тест-кейс №7. Настройки.....	37
6.8. Тест-кейс №8.1. Подготовка к началу игры	38
6.9. Тест-кейс №8.2. Ход игрока. Розыгрыш карт	39
6.10. Тест-кейс №8.3. Состояния игры	40
6.11. Тест-кейс №9. Игровая статистика	41
6.12. Тест-кейс №10. Обратная связь.....	42

7. Use case	43
7.1. Диаграмма сценария использования.....	43
7.2. Сценарий №1. Регистрация пользователя	43
7.3. Сценарий №2. Авторизация.....	45
7.4. Сценарий №3. Начать онлайн-игру.....	46
7.5. Сценарий №4. Начать одиночную игру.....	49
7.6. Сценарий №5. Реализация игрового процесса.....	50
7.7. Сценарий №6. Выход пользователя.....	51
7.8. Сценарий №7. Работа со справкой/бэнглопедией	52
7.9. Сценарий №8. Работа с настройками.....	53
7.10. Сценарий №9. Осуществление обратной связи.....	54
7.11. Сценарий №10. Просмотр игровой статистики.....	54
8. Инструкция по сборке	56
9. Используемые инструменты.....	57
Заключение.....	58
10. Личный вклад участника команды.....	59
11. Список использованных источников	60
12. ПРИЛОЖЕНИЕ 1. скрипт создания БД.....	61

ВВЕДЕНИЕ

1.1. Тема работы

Разработке мультидисциплинарного проекта в рамках front-разработки и создания базы данных для клиент-серверного приложения «Бэнг!».

1.2. Цель работы

Целью данного проекта является разработка программного продукта, которое представляет собой онлайн версию настольной карточной игры «Бэнг!» [1]. Данное приложение позволит пользователям сыграть в любимую настольную игру на компьютере или ноутбуке и попробовать свои силы как с обученными ботами на различных уровнях сложности, так с другими игроками по всему миру.

1.3. Задачи работы

Задачи данного проекта:

1. Разработать функционал, чтобы пользователь имел возможность сыграть с ботами даже при отсутствии доступа в глобальную сеть. При этом необходимо, чтобы пользователь мог самостоятельно задать параметры игры, такие как:

- уровень сложности ботов;
- количество ботов в игре;
- выбрать роль;
- выбрать персонажа.

2. Разработать функционал для регистрации и авторизации пользователей, чтобы данные пользователи имели возможность поиграть с другими пользователями онлайн (незарегистрированные пользователи не должны иметь такую возможность).

3. Разработать функционал для того, чтобы любой зарегистрированный пользователь имел возможность поиграть в «Бэнг!» с другими пользователями по сети.

4. Разработать функционал, чтобы пользователь в приложении имел возможность регулировать основные параметры игры, а именно:

- включать/выключать звук;
- регулировать громкость звуков;
- регулировать громкость музыки.

5. Разработать функционал для того, чтобы новые пользователи имели возможность ознакомиться с правилами игры «Бэнг!», изучить её особенности, освоить основные

модели поведения в игре, познакомиться с картами и узнать эффективные способы розыгрыша карт.

6. Разработать функционал для сохранения истории результатов игр пользователей с возможностью просмотра своей истории каждым пользователем.

1.4. Актуальность темы

Несомненно, многие знают настольную игру «Bang» («Бэнг» рус.). Эта ролевая игра в жанре вестерн стала популярной по всему миру.

В ходе исследования рынка клиент-серверных игр было обнаружено, что данная игра не представлена в данном формате для пользования. Данное приложение позволит пользователям сыграть в любимую настольную игру на компьютере или ноутбуке и попробовать свои силы как с обученными ботами на различных уровнях сложности, так с другими игроками по всему миру.

Таким образом, разработка приложения для игры «Bang» является актуальной и уникальной.

2. ТРЕБОВАНИЯ К СИСТЕМЕ

2.1. Описание функциональных требований

1. Общий функционал во время игры

- 1.1. Правила игры определяются официальными правилами карточной игры «Бэнг» [2].
- 1.2. Игрок может видеть все свои карты, которые находятся у него «на руке» и в игре, а также видеть карты других игроков, которые находятся в игре. Кроме того, игрок может узнать, сколько карт на руке имеется у других игроков.
- 1.3. Во время игры пользователь может зайти в раздел настроек, чтобы изменить их по своему усмотрению.
- 1.4. Во время игры пользователь может посмотреть внутриигровую справку для того, чтобы узнать или уточнить интересующую его информацию по игре.
- 1.5. Пользователь может покинуть игру и выйти в главное меню по своему усмотрению.

2. Игра с ботами (single-player mode)

Перед и во время игры с ботами необходимо реализовать следующий функционал:

- 2.1. Функционал перед началом игры.
 - 2.1.1. Возможность начать новую игру.
 - 2.1.1.1. Задание количества игроков (от 3-ёх до 8-ми).
 - 2.1.1.2. Задание уровня сложности игры с ботами (лёгкий, средний, тяжёлый).
 - 2.1.1.3. Получение случайной карточки с ролью (шериф, бандит, ренегат, помощник).
 - 2.1.1.4. Выбор одной из трёх случайно выбранных карт персонажей.
 - 2.1.2. Возможность продолжить одну из сохранённых ранее партий.
- 2.2. Функционал во время игры.
 - 2.2.1. Пользователь может поставить игру на паузу.
 - 2.2.2. Пользователь может сохранить текущее состояние игры, чтобы иметь возможность завершить партию в другое время.

3. Вход в аккаунт.

Для того, чтобы пользователю был доступен многопользовательский режим игры, необходимо предоставить пользователю следующие возможности:

3.1. Регистрация.

При регистрации пользователю должна быть предоставлена специальная форма, в которой ему будет предложено ввести почту, логин и пароль.

3.2. Авторизация.

В случае, если пользователь ранее прошёл процедуру регистрации, пользователю будет доступна процедура авторизации. Для этого пользователю должна быть предоставлена специальная форма, в которой ему будет предложено ввести почту и пароль. При верном вводе почты и пароля, авторизация считается успешно пройденной. В противном случае, пользователю будет предложено ввести почту для того, чтобы выслать на неё текущий пароль.

4. Настройки.

Для комфортного использования приложения, а также для разнообразия геймплея и поддержания интереса к игре, должны присутствовать следующие настройки:

4.1. Громкость.

Возможность изменять (а также отключать) громкость различных звуков, таких как:

4.1.1. Громкость фоновой музыки.

4.1.2. Громкость звуковых эффектов.

5. Справка.

Справка необходима, чтобы пользователь получил интересующую его информацию об игре. Всего должно присутствовать 2 типа справок:

5.1. Внутриигровые справки.

Внутриигровые справки помогают пользователю освоиться и разобраться с основными аспектами игры. Внутриигровые справки также делятся на следующие категории:

5.1.1. Справка о картах ролей.

5.1.2. Справка о картах персонажей.

5.1.3. Справка об игровых картах.

5.2. Сторонняя справка.

Данная справка отражает основную информацию о разработчиках приложения.

6. Список достижений.

Список достижений необходим для того, чтобы каждый пользователь смог посмотреть статистику по проведённым играм. При выборе данного пункта из главного меню в новом окне должна появляться следующая информация:

6.1. Количество сыгранных игр в одиночной игре.

6.2. Количество незаконченных игр в одиночной игре.

6.3. Количество побед в одиночной игре.

- 6.4. Количество поражений в одиночной игре.
- 6.5. Процент побед в одиночной игре.
- 6.6. Количество сыгранных игр в онлайн игре.
- 6.7. Количество побед в онлайн игре.
- 6.8. Количество поражений в онлайн игре.
- 6.9. Процент побед в онлайн игре.

7. Обратная связь.

Обратная связь необходима для того, чтобы каждый пользователь мог высказать свои пожелания по поводу игры. При выборе данного пункта из главного меню, должна появляться специальная форма, в которую пользователь сможет ввести свой отзыв о данной игре и затем отправить его на специальную почту, где будут сохраняться отзывы от всех игроков.

2.2. Описание нефункциональных требований

1. Расширяемость.

Поскольку разрабатываемая настольная игра “Бэнг” имеет ряд дополнений, система должна быть расширяема. Под расширяемостью имеется в виду расширяемость разрабатываемой базы данных с возможностью добавления новых элементов, расширяемость программного кода с возможностью добавления новых функций с минимальным количеством изменяемого кода, а также возможность добавления новых языков в игру (единственный на стадии разработки используемый язык — русский).

2. Требования к клиентскому программному обеспечению.

Минимальные системные требования:

- ОС: Windows XP/7/8/10
- Процессор: Pentium II 300 МГц
- Оперативная память: 64 Мб
- Видеокарта: 4 Мб
- Свободное место на жёстком диске: 850 Мб
- Разрешение экрана: 1024*768
- Клавиатура
- Мышь

Рекомендуемые системные требования:

- ОС: Windows XP/7/8/10
- Процессор: Pentium II 550 МГц

- Оперативная память: 128 Мб
- Видеокарта: 4 Мб
- Свободное место на жёстком диске: 850 Мб
- Разрешение экрана: 1024*768
- Клавиатура
- Мышь

Поддерживаемые браузеры:

- Internet Explorer (версия 11+)
- Google Chrome (версия 60+)
- Opera (версия 36+)
- Mozilla Firefox (версия 52+)

Поддерживаемые языки:

- Русский

3. Удобство использования.

Данное приложение должно быть удобно пользователю в процессе эксплуатации. Под удобством имеется в виду, чтобы элементы меню, карты в игре, символы и надписи на них были отчётливо видны среднестатистическому пользователю. Также в игре должен присутствовать проработанный дизайн, отвечающий тематике игры и отражающий дух дикого запада.

4. Требования надёжности к серверному программному обеспечению.

Система должна быть надёжна. Под надёжностью имеется в виду:

- В случае необходимости проведения профилактических работ на сервере либо с базой данных, ранее созданные комнаты не должны быть затронуты, то есть должны работать стабильно.
- В программе должны быть предусмотрены варианты обработки исключений, чтобы минимизировать количество возникающих ошибок в процессе игры, которые приводили бы к сбоям в работе приложения, либо к отказу функционирования.

5. Требования к техническому обеспечению.

Минимальные аппаратные требования к серверам

Сервер приложений:

- Процессор: 4 ядра (8 логических потоков), частота - 2 ГГц и больше
- Оперативная память: 4 Гб и больше
- Свободное дисковое пространство: 50 Гб
- Пропускная способность сетевого интерфейса: 1 Гбит/с

Сервер баз данных:

- Тип накопителя: SSD
- Процессор: 4 ядра (8 логических потоков), частота - 2 ГГц и больше
- Оперативная память: 8 Гб и больше
- Свободное дисковое пространство: 300 Гб и больше
- Пропускная способность сетевого интерфейса: 1 Гбит/с

Рекомендуемые аппаратные требования к серверам

Сервер приложений:

- Процессор: 4 ядра (8 логических потоков), частота - 3-3,5 ГГц и больше
- Оперативная память: 32 Гб и больше
- Свободное дисковое пространство: 128 Гб
- Пропускная способность сетевого интерфейса: 1 Гбит/с

Сервер баз данных:

- Тип накопителя: SSD
- Процессор: 4 ядра (8 логических потоков), частота - 3-3,5 ГГц и больше
- Оперативная память: 32 Гб и больше
- Свободное дисковое пространство: 300 Гб и больше
- Пропускная способность сетевого интерфейса: 1 Гбит/с

6. Требования к производительности

Таблица 1 - Требования к производительности

Тип системы	Требование к производительности	Ограничения окружения
Веб-сервер	Количество одновременно обслуживаемых пользователей - 64.	
Веб-сервер	Время от момента получения запроса до момента создания полного ответа не должно превышать 300 мсек	При условии одновременной обработки не более 32 запросов
Веб-сервер	Объём используемой виртуальной памяти	При условии одновременной обработки

	(включая кэш) не должен превышать 1 Гб	не более 32 запросов
Веб-сервер	Нагрузка на CPU не должна превышать 50%	При условии одновременной обработки не более 32 запросов
База данных	Время отклика от базы данных не должно превышать 200 мсек	
База данных	Ограничение на максимальный размер базы данных отсутствует. Тем не менее рекомендуемый размер базы данных без учёта обновлений - 100 Мб	Максимальный размер базы данных на хостинге gearhost - 100 Мб. В случае возникновения необходимости в увеличении данного объёма, будет рассматриваться возможность перехода на другой хостинг для хранения базы данных
Веб-страница	Время начальной загрузки игры не должно превышать 10000 мсек	С учётом времени для установления соединения с базой данных
Веб-страница	Нагрузка на CPU в режиме простоя приложения не должна превышать 5%	
Веб-страница	Объём памяти для хеширования объектов JavaScript в окне не должен превышать 25 Мб	
Веб-страница	Время от возникновения	

	ошибки до вывода предупреждения не должно превышать 25 мсек	
--	--	--

7. Требования к эргономике

Игра должна обеспечивать быструю и корректную работу. В то же время все действия не должны быть слишком резкими, пестрыми и яркими, дабы не вызвать проблем со здоровьем у людей, страдающими особенными заболеваниями.

3. АРХИТЕКТУРА ПРИЛОЖЕНИЯ.

После определения требований к системе и разработки сценария использования приложения было принято решение разработать клиент-серверное приложение, архитектура которого изображена на рисунке 1.

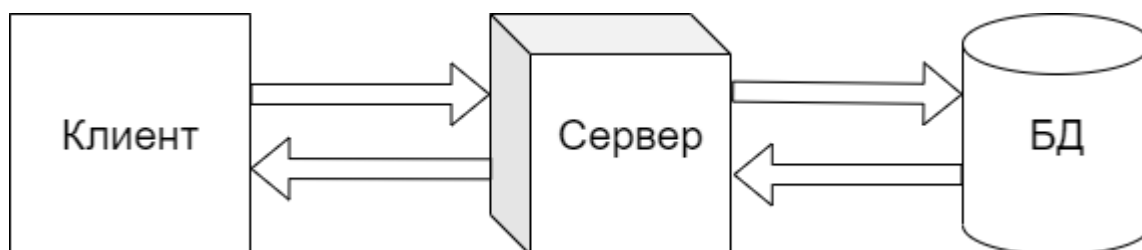


Рисунок 1 - Архитектура клиент-серверного приложения

4. ЭСКИЗНОЕ ПРОЕКТИРОВАНИЕ ЭЛЕМЕНТОВ ПРИЛОЖЕНИЯ.

4.1. Эскизы элементов одиночной игры

В ходе выполнения производственной практики были разработаны эскизы одиночной игры.

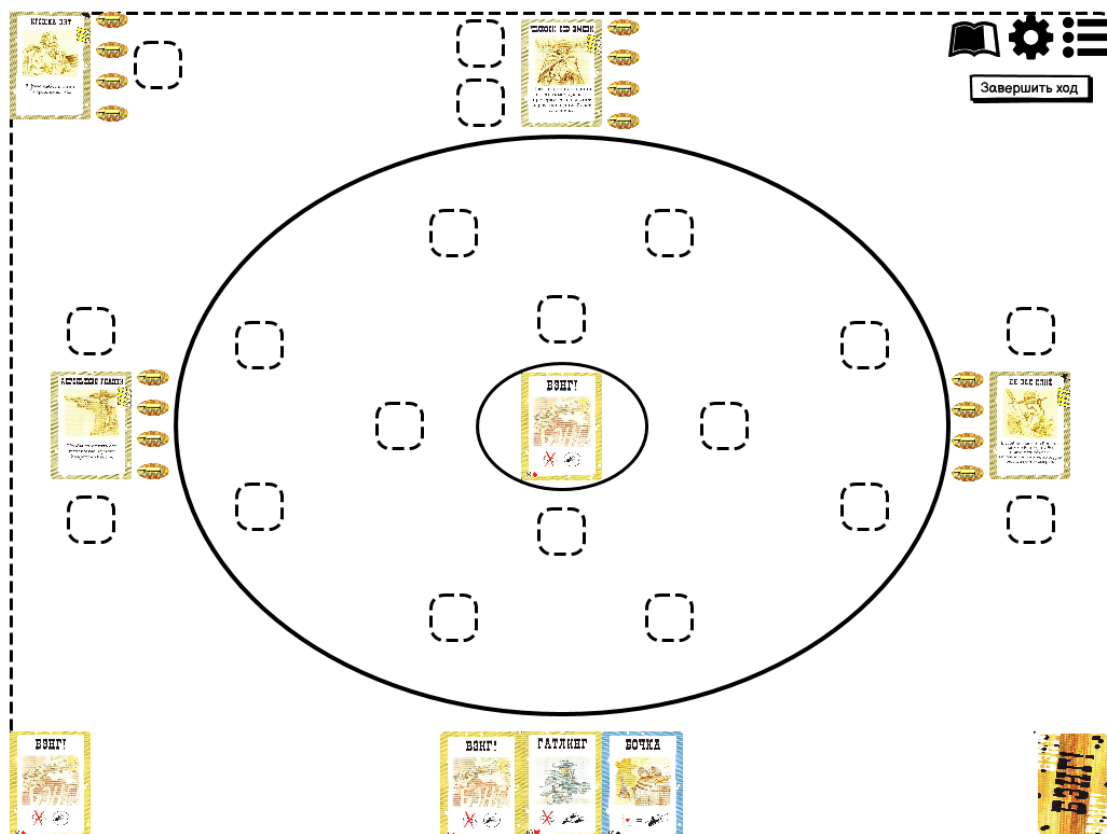


Рисунок 2 - Одиночная игра



Рисунок 3 - Одиночная игра. Выбор персонажа



Рисунок 4 - Одиночная игра. Получена случайная роль

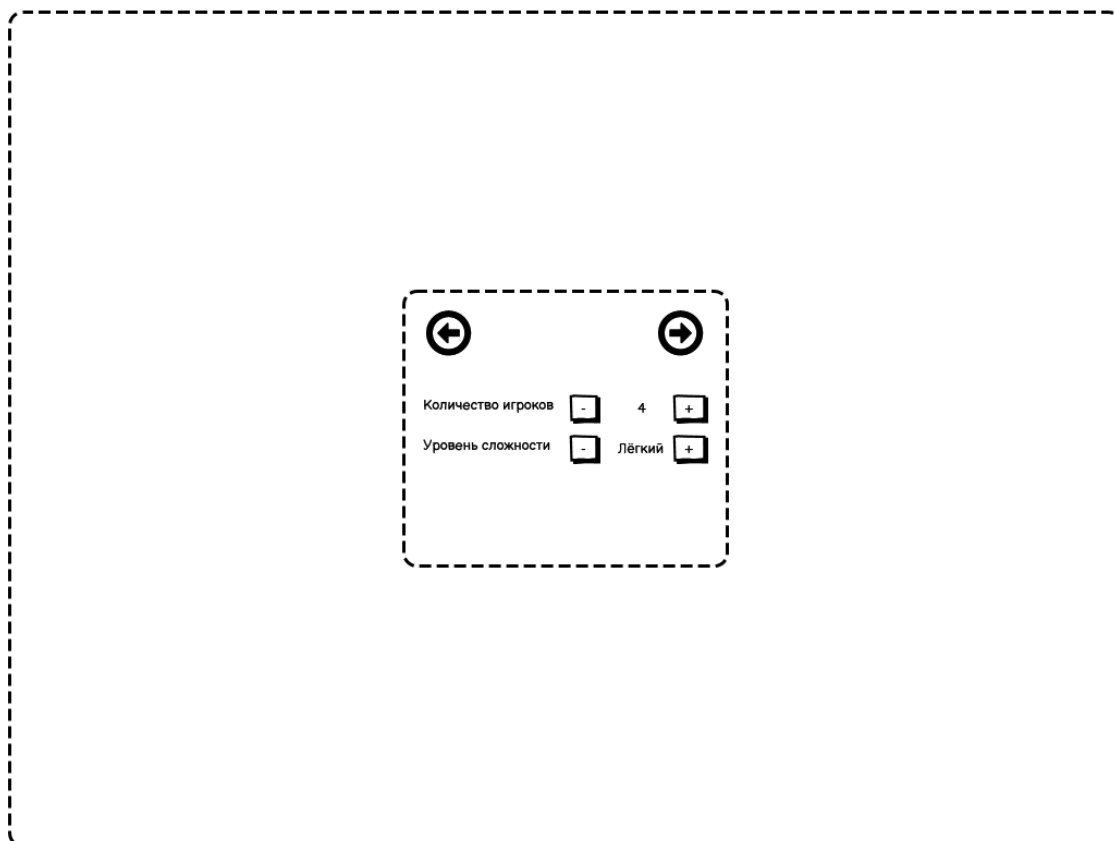


Рисунок 5 - Одиночная игра. Окно 1



Рисунок 6 - Одиночная игра. Окно 2

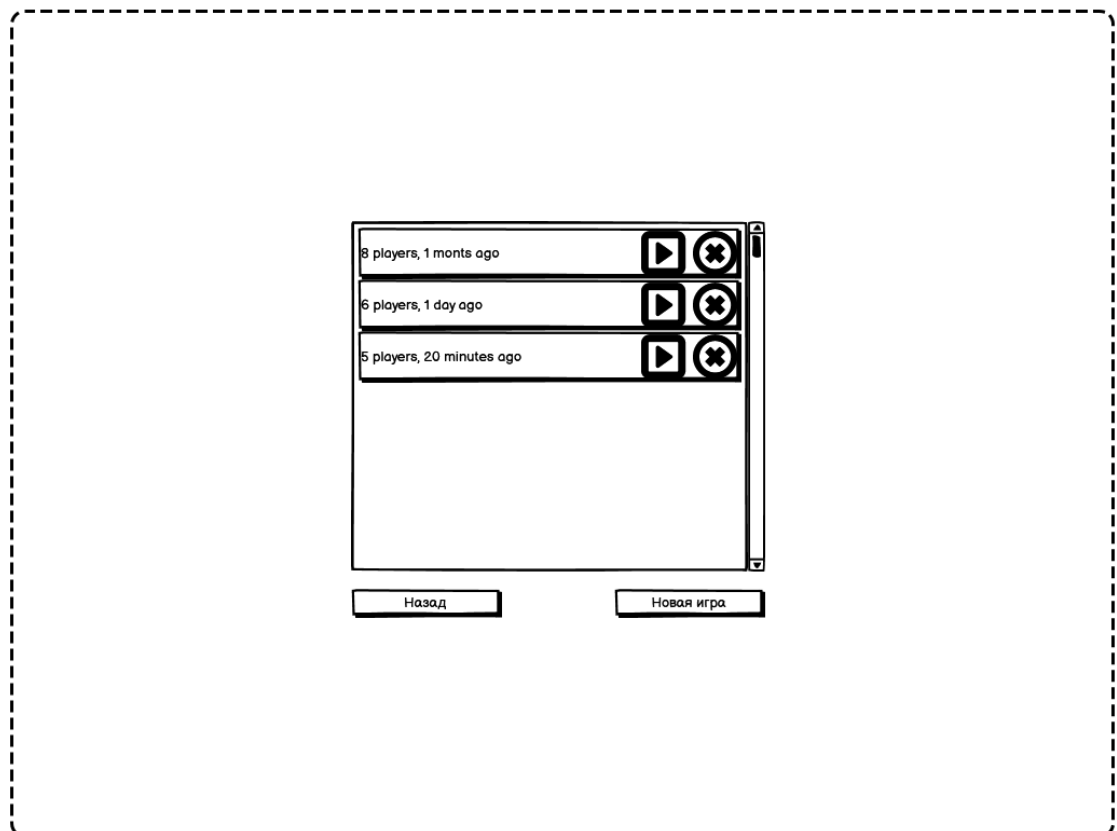


Рисунок 7 - Одиночная игра. Загрузка

4.2. Эскизы элементов онлайн игры

В ходе выполнения производственной практики были разработаны эскизы элементов онлайн игры.

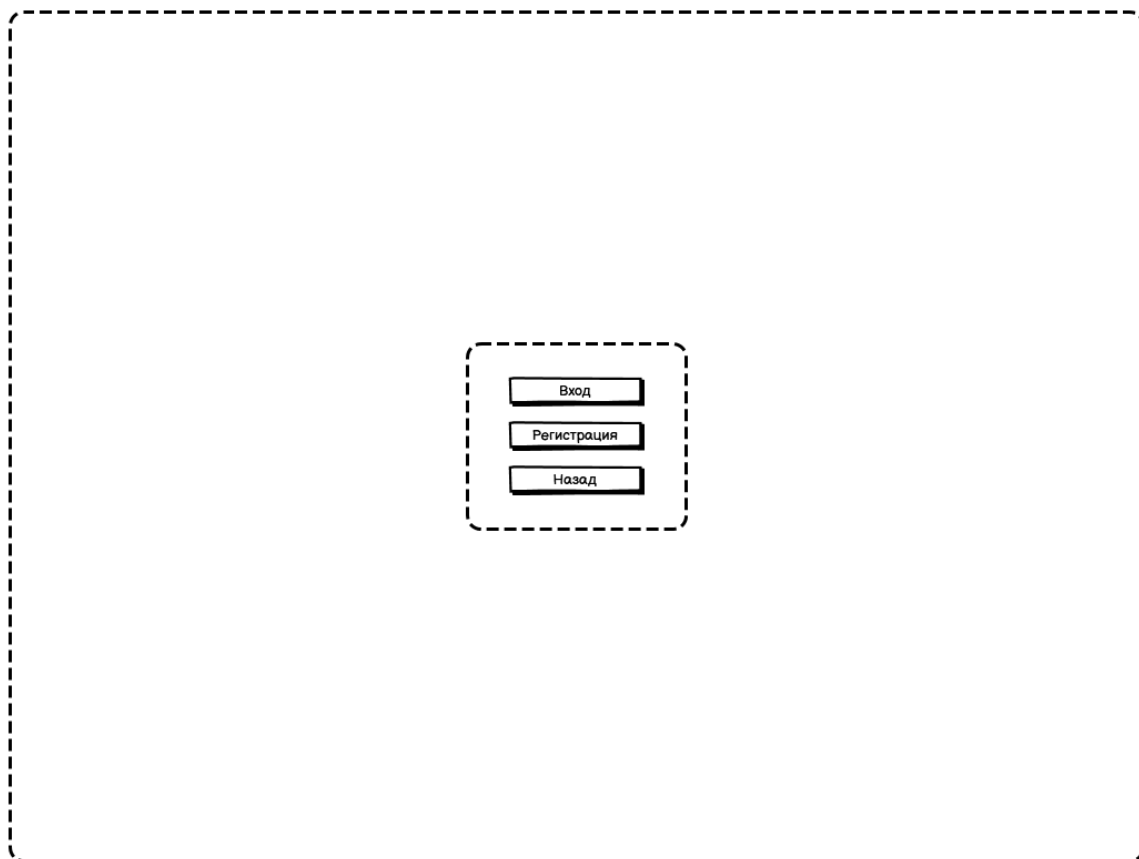


Рисунок 8 - Онлайн игра. Окно 1

Регистрация нового пользователя

Create a username

Create a password

Re-enter password

< Back

Next >

Рисунок 9 - Онлайн игра. Регистрация

A login form titled "Вход" (Login) enclosed in a dashed rectangular border. The form has a title bar with the text "Вход". Below the title bar, there are two input fields. The first is labeled "Enter a username" and the second is labeled "Enter a password". At the bottom of the form, there are two buttons: "< Back" on the left and "Next >" on the right.

Рисунок 10 - Онлайн игра. Вход

A room selection screen for an online game. It features a list of three rooms, each with a player count and a "Присоединиться" (Join) button. The rooms are: "7/8 players", "5/8 players", and "3/4 players". Below the list, there are two buttons: "Назад" (Back) and "Создать комнату" (Create room). The entire screen is enclosed in a dashed rectangular border.

Рисунок 11 - Онлайн игра. Выбор комнаты

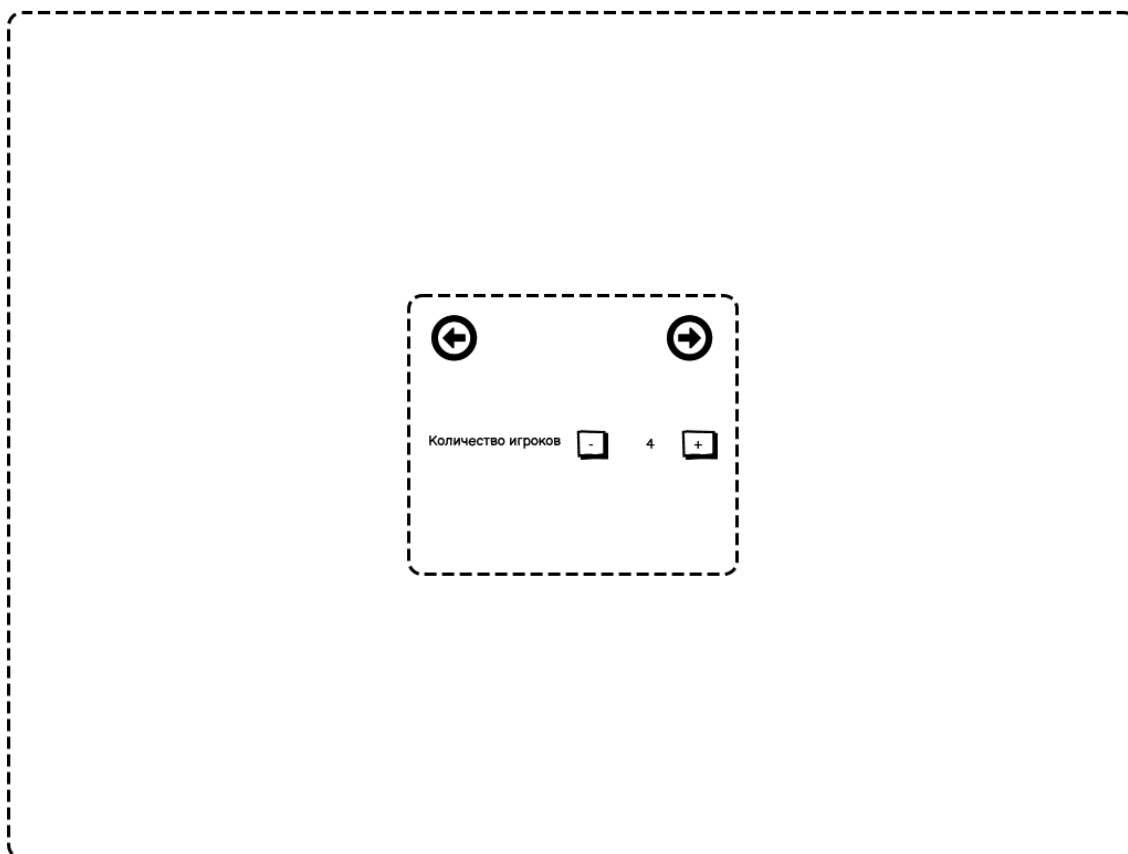


Рисунок 12 - Онлайн игра. Создание комнаты

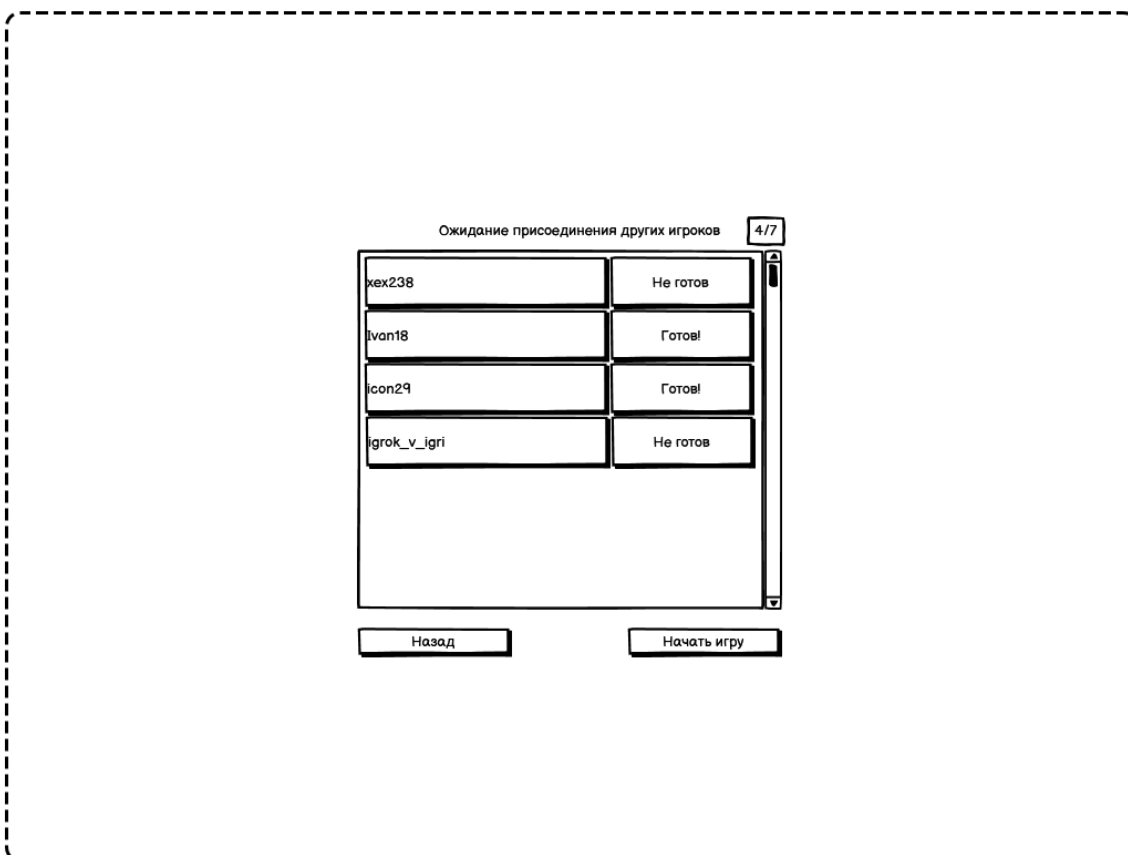


Рисунок 13 - Онлайн игра. Комната

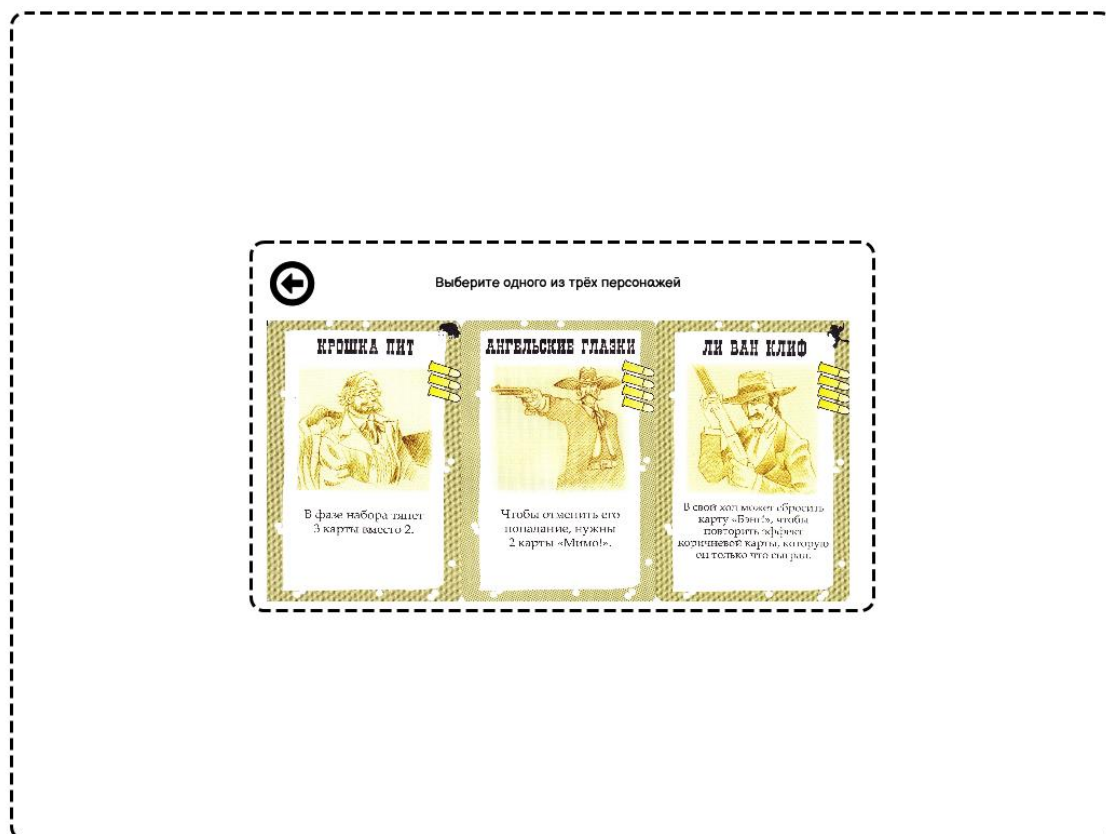


Рисунок 14 - Онлайн игра. Выбор персонажа



Рисунок 15 - Онлайн игра. Получена случайная роль

4.3. Эскизы дополнительных элементов

В ходе выполнения производственной практики были разработаны эскизы дополнительных элементов игры.

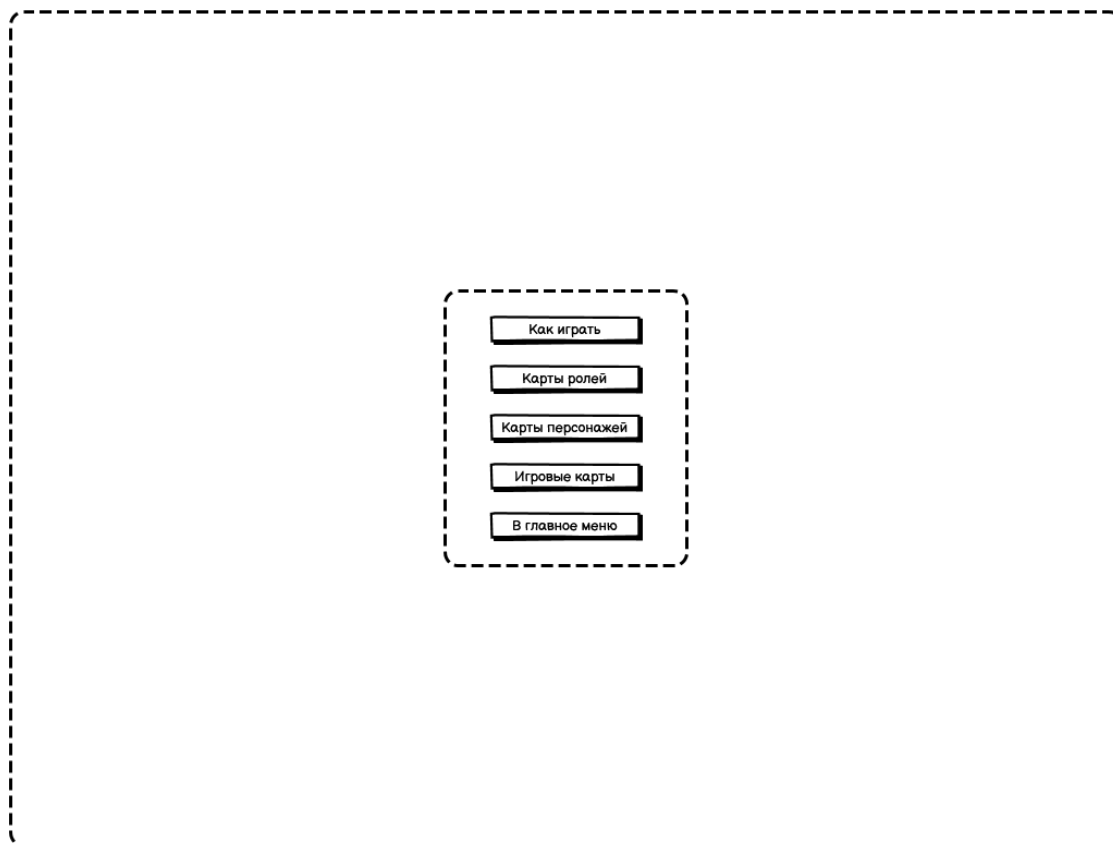


Рисунок 16 – Бэнглопедия

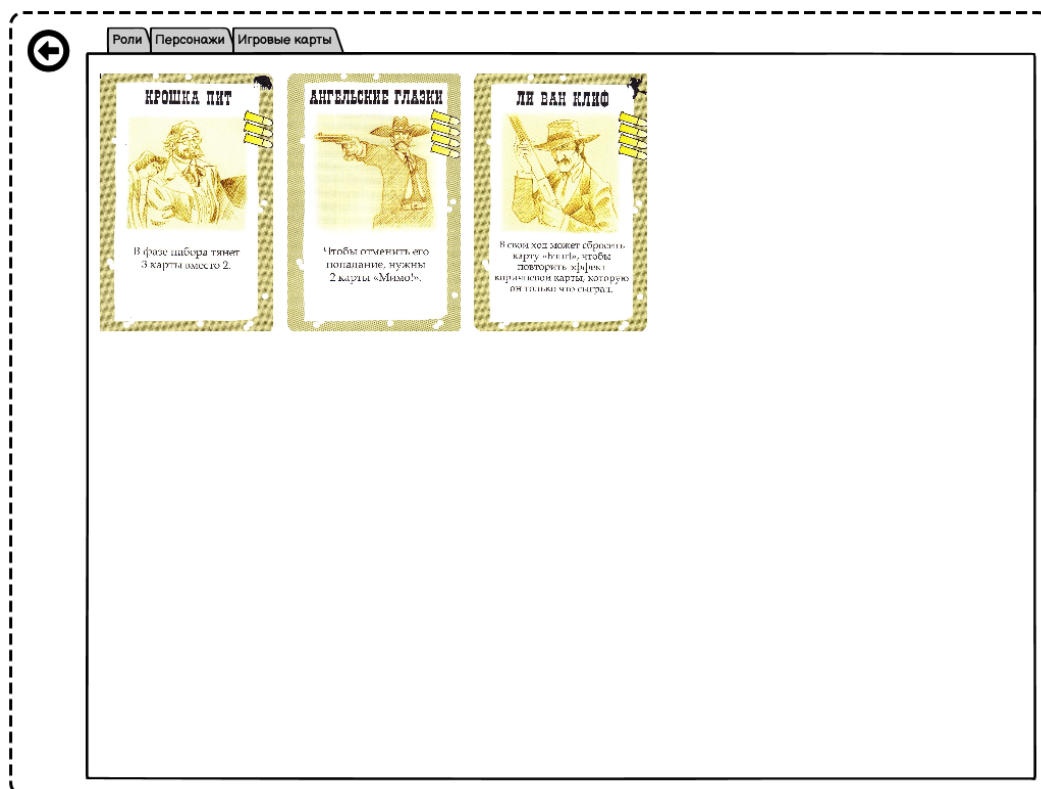


Рисунок 17 – Карты

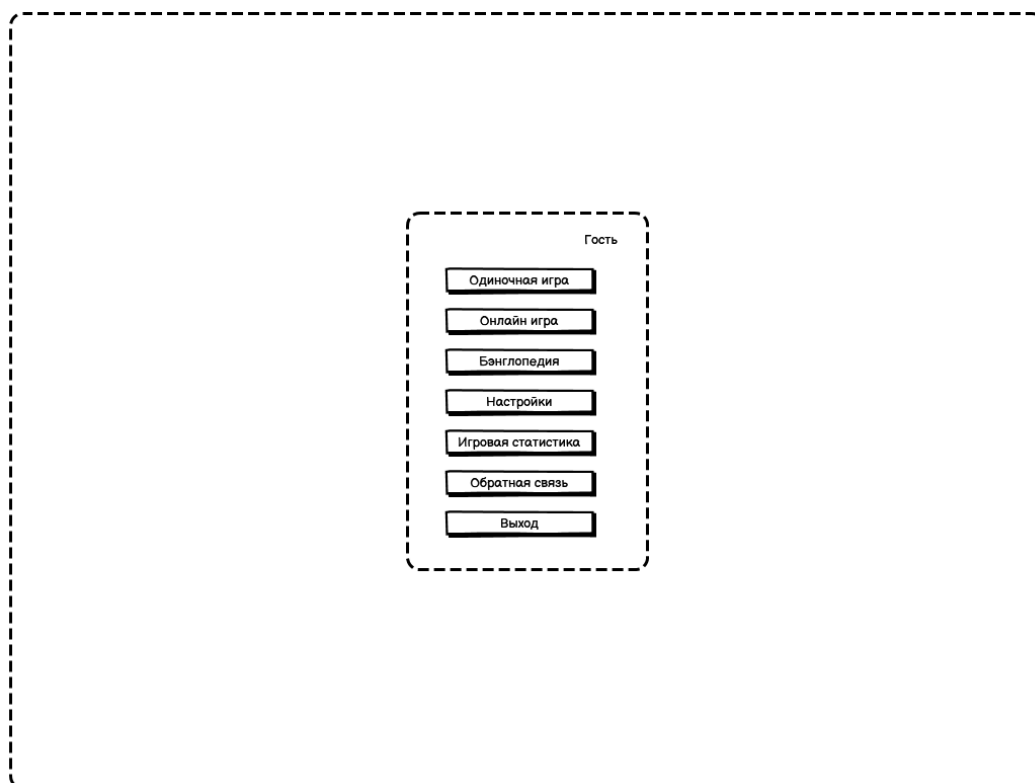


Рисунок 18 - Главное меню. Гость

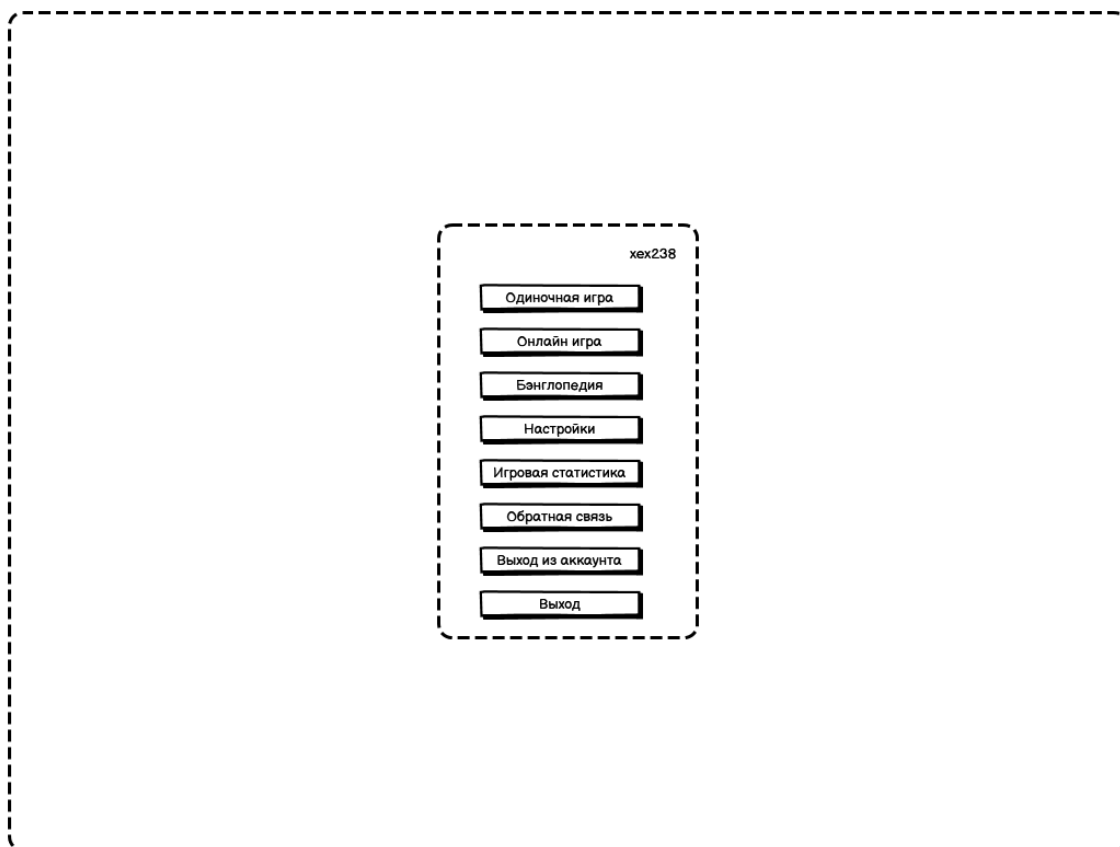


Рисунок 19 - Главное меню. Пользователь

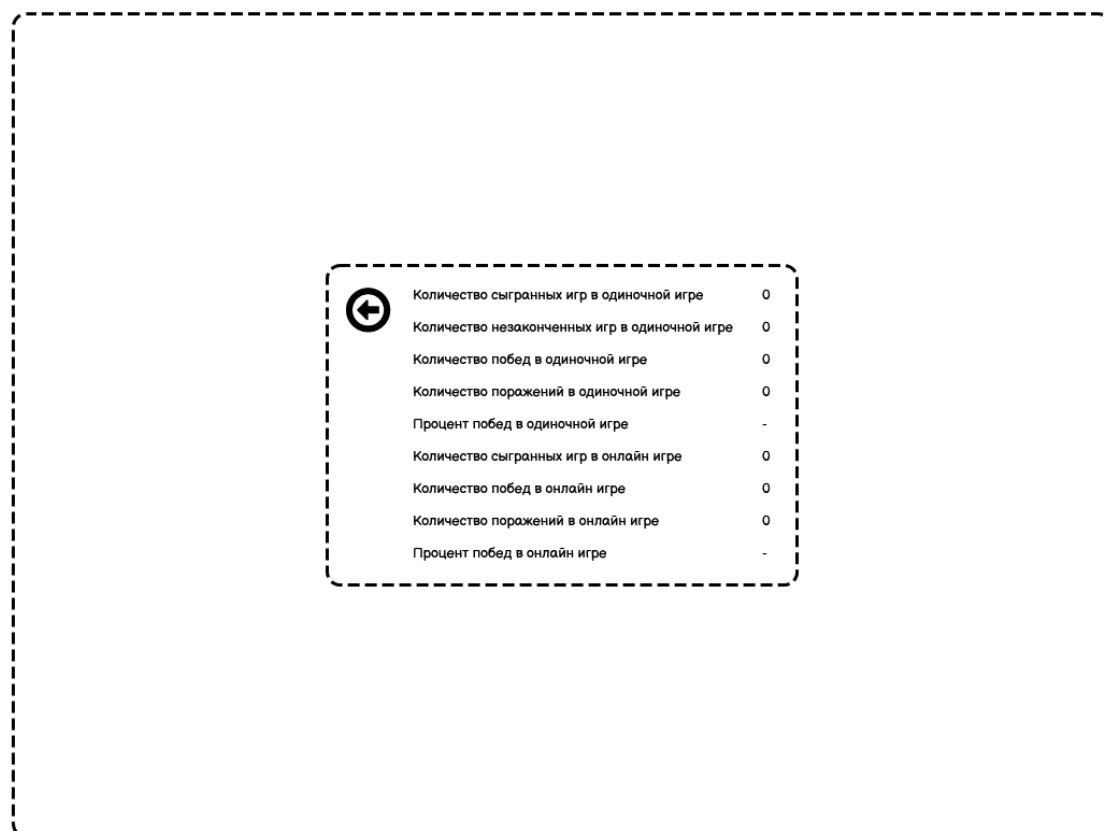


Рисунок 20 - Игровая статистика

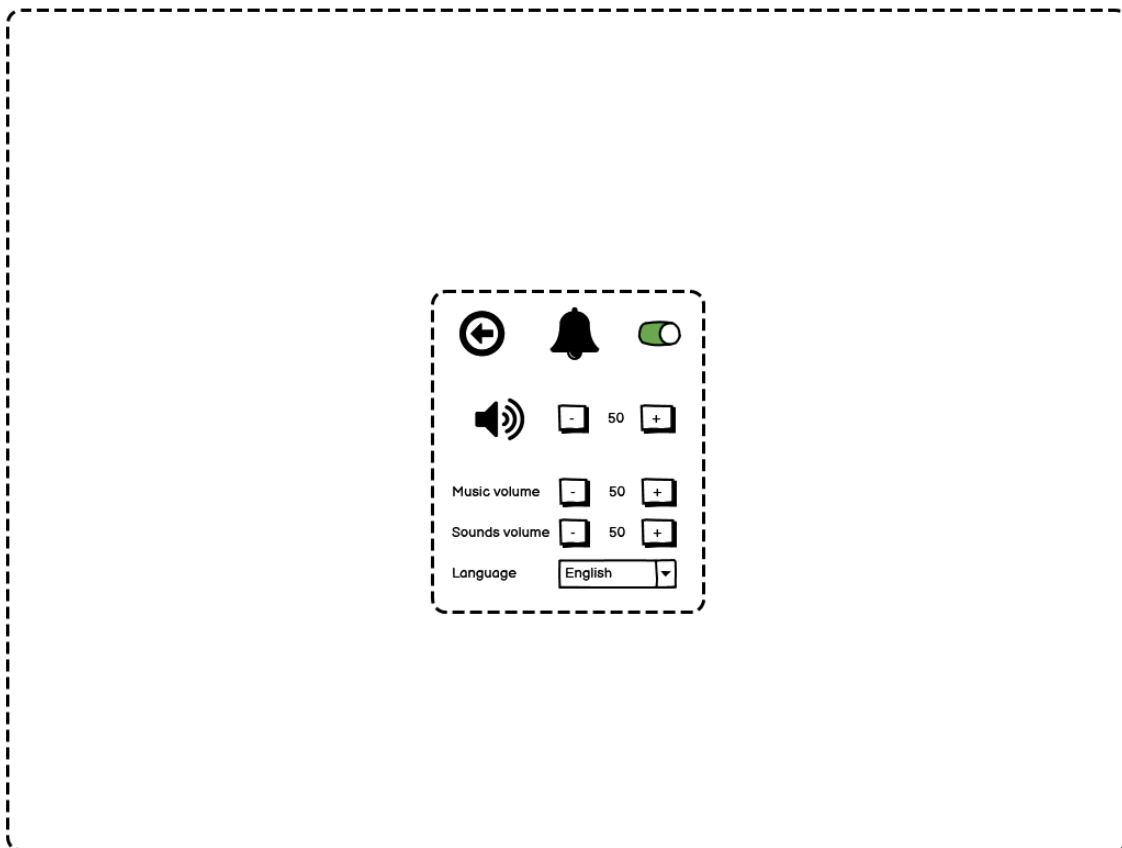


Рисунок 21 – Настройки

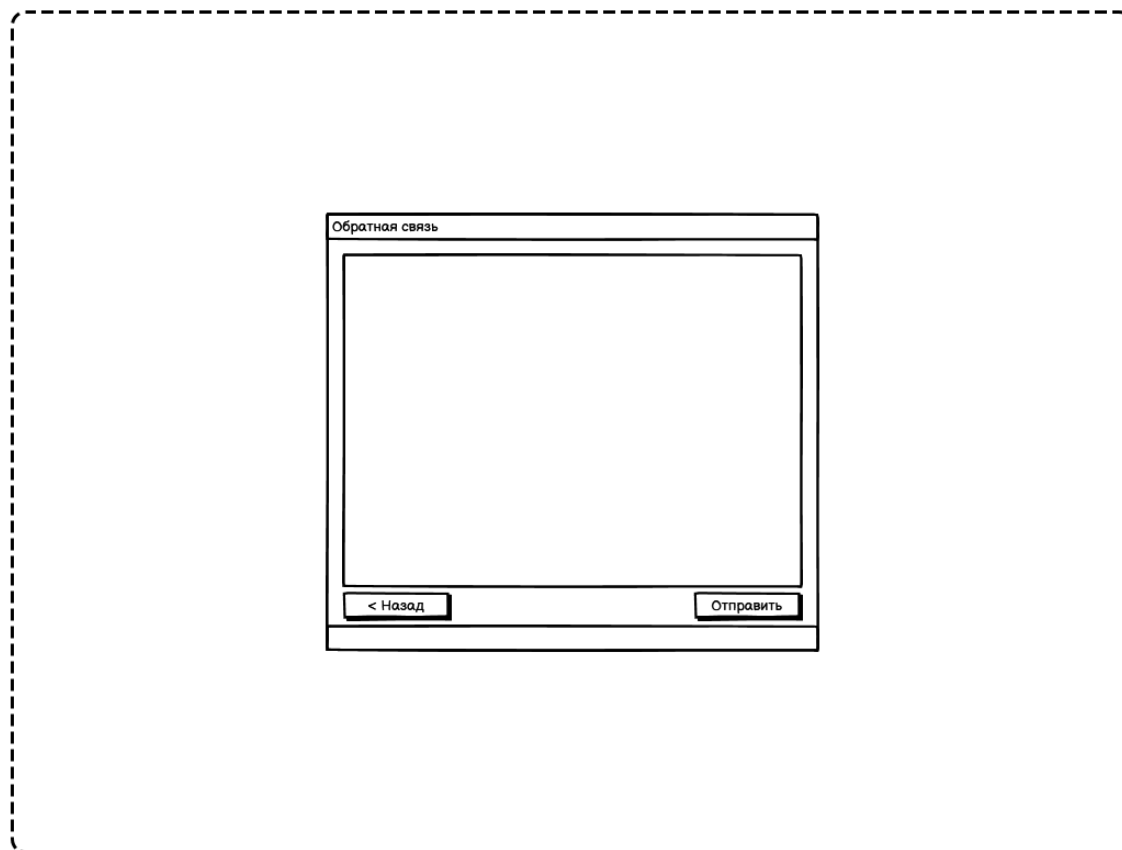


Рисунок 22 - Обратная связь



Рисунок 23 - Правила



Рисунок 24 - Результат игры

5. РАЗРАБОТКА БАЗЫ ДАННЫХ

В ходе данной производственной практики была разработана ER-модель и база данных Bang с использованием свободного кроссплатформенного программного обеспечения SQL Server Management Studio.

База данных и содержащиеся в ней таблицы были разработаны согласно архитектуре программного продукта.

Загружается у
пользователя
Хранится только на
сервере

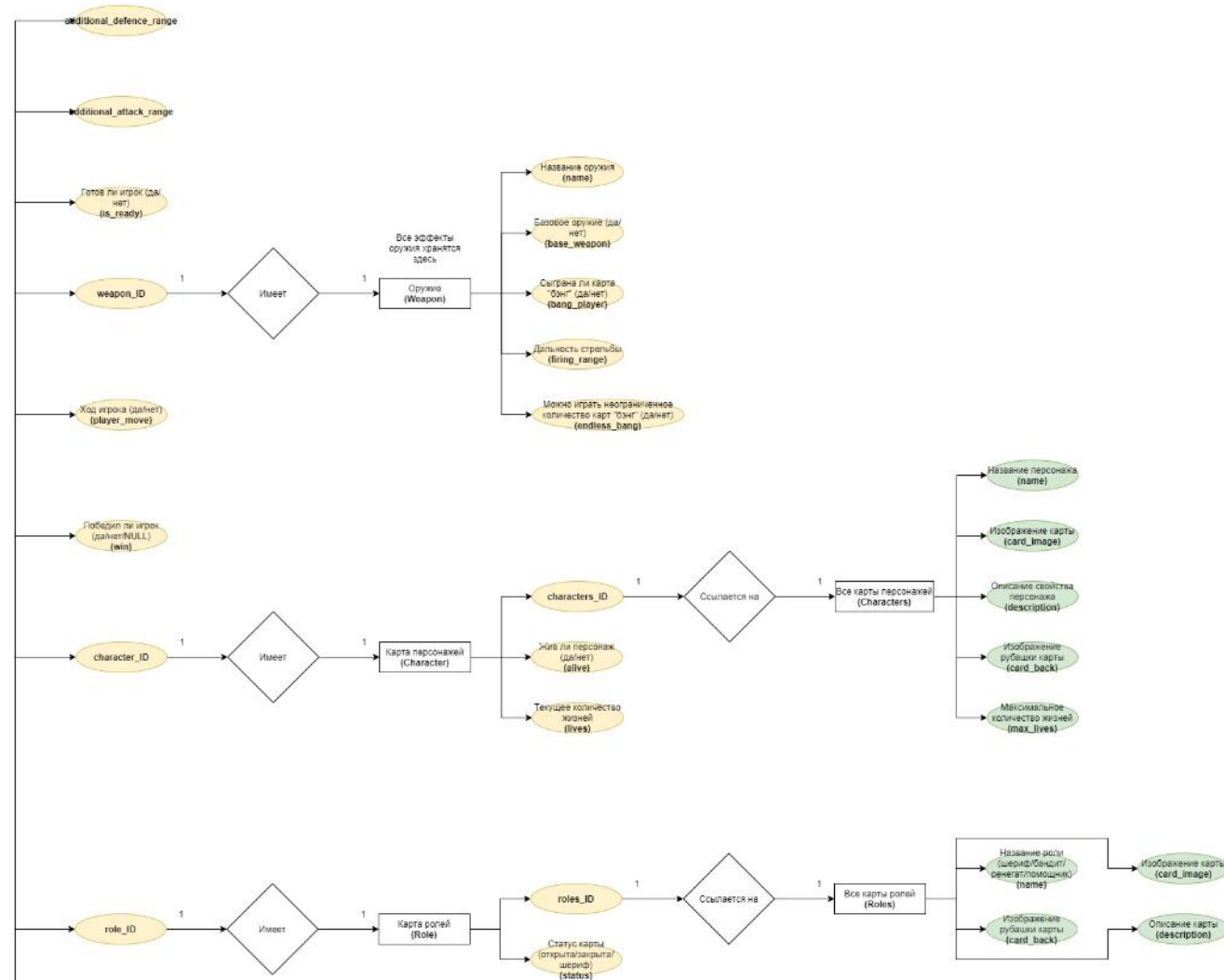


Рисунок 25 - ER-модель. Часты1

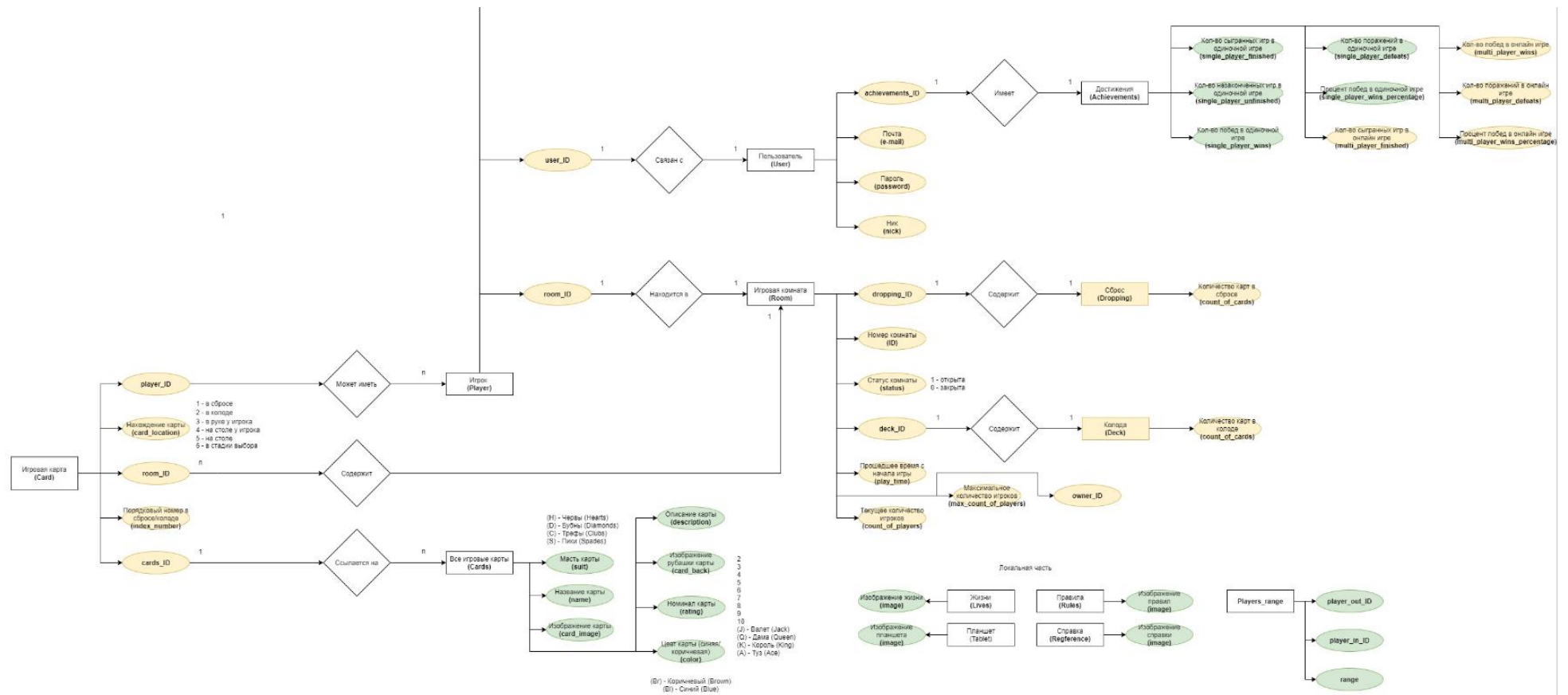


Рисунок 26 - ER-модель. Часть2

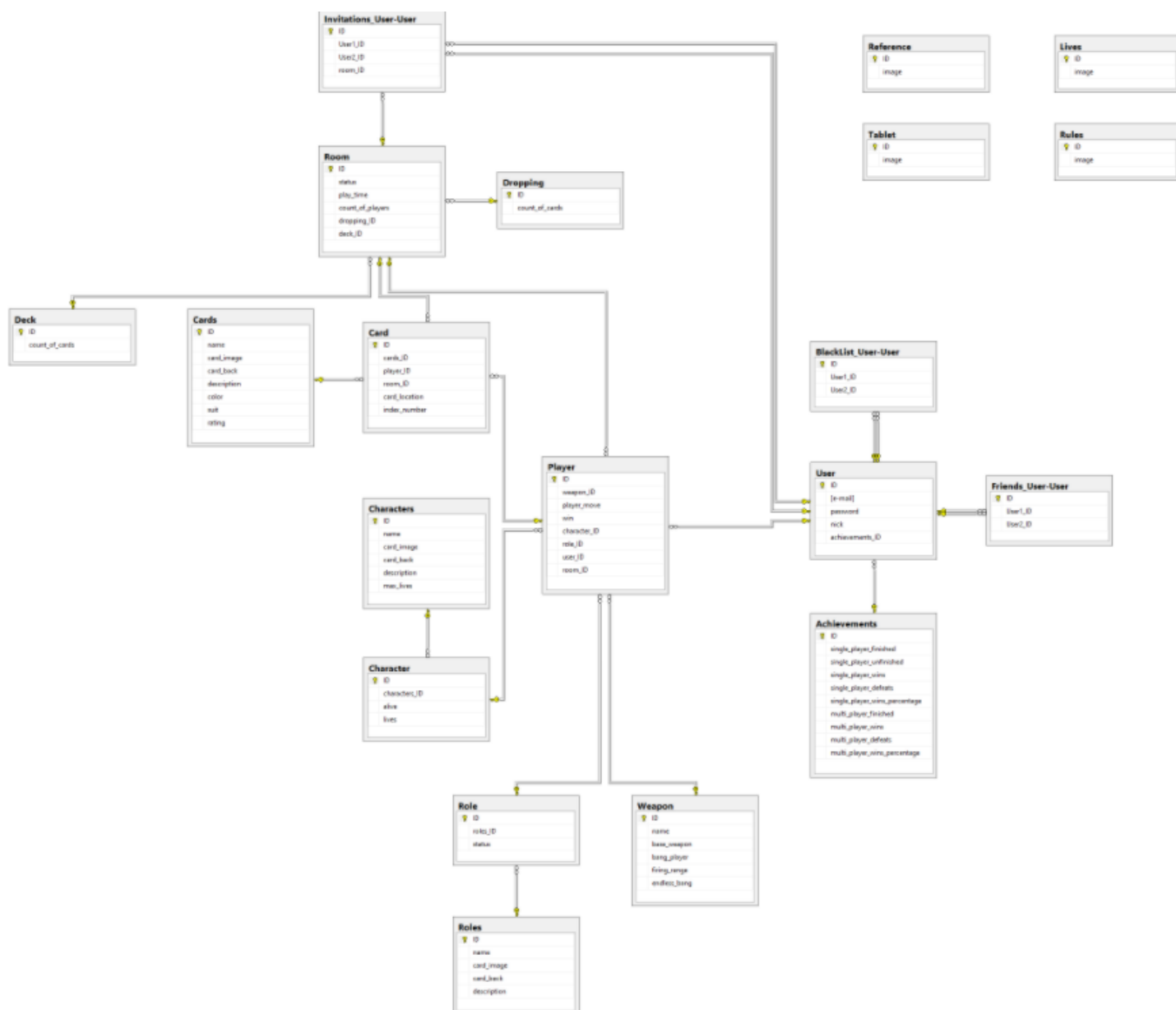


Рисунок 27 - База данных

6. TEST CASE

6.1. Тест-кейс №1. Главное меню

Название: **Управление главным меню**

Предусловие: **Открыто главное меню**

Таблица 2 - Тест-кейс №1. Главное меню

Шаг	Ожидаемый результат
Предусловие: пользователь не авторизован, роль гостя	
Нажать на кнопку «Одиночная игра»	Переход на страницу, где отображаются ранее созданные игры и можно создать новую (см. Тест-кейс №4)
Нажать на кнопку «Онлайн игра»	Переход на страницу с комнатами(см. Тест-кейс №3)
Нажать на кнопку «Бэнглопедия»	Переход к меню Бэнглопедии(см. Тест-кейс №5)
Нажать на кнопку «Настройки»	Переход на страницу настроек(см. Тест-кейс №6)
Нажать на кнопку «Выход»	Выход из приложения
Предусловие: пользователь авторизован	
Нажать на кнопку «Одиночная игра»	Переход на страницу, где отображаются ранее созданные игры и можно создать новую (см. Тест-кейс №4)
Нажать на кнопку «Онлайн игра»	Переход на страницу с комнатами(см. Тест-кейс №3)
Нажать на кнопку «Бэнглопедия»	Переход к меню Бэнглопедии(см. Тест-кейс №5)
Нажать на кнопку «Настройки»	Переход на страницу настроек(см. Тест-кейс №6)
Нажать кнопку на «Выход из аккаунта»	Произошел выход из аккаунта и в правом верхнем углу отображается надпись «Гость»
Нажать на кнопку «Выход»	Выход из приложения

6.2. Тест-кейс №2. Регистрация

Название: **Регистрация**

Предусловие: **Открыто меню входа/регистрации**

Таблица 3 - Тест-кейс №2. Регистрация

Шаг	Ожидаемый результат
Выбрать опцию «Регистрация» в главном меню	Появляется форма с полями для ввода
Заполнить поле «Создать логин» латинскими буквами	В поле «Логин» отображается введенное значение
Ввести в поле «Создать логин» буквы не латинского алфавита	Ошибка - «Это поле может содержать буквы только латинского алфавита»
Заполнить поле «Создать логин» неуникальным значением	Ошибка - «Данный логин уже используется»
Ввести корректный e-mail в поле «Введите e-mail»	В поле «Введите e-mail» отображается введенный e-mail
Ввести e-mail, уже зарегистрированный в системе	Ошибка – «Для данной почты уже существует аккаунт»
Ввести некорректный e-mail	Ошибка - «Введите корректный e-mail»
Заполнить поле «Создайте пароль»	В поле «Создайте пароль» отображается пароль, зашифрованный символом «*» .
Заполнить поле «Подтвердите пароль» теми же символами, что и поле «Создайте пароль»	В поле «Подтвердите пароль» отображается пароль, зашифрованный символом «*».
Заполнить поля «Подтвердите пароль» и «Создайте пароль» не совпадающими значениями	Ошибка - «Пароли не совпадают»
Оставить любое поле пустым	Ошибка - «Поле не может быть пустым»
Нажать на кнопку «Далее»	Переход на страницу авторизации
Нажать на кнопку «Назад»	Произойдет переход в главное меню

6.3. Тест-кейс №3. Авторизация

Название: **Авторизация**

Предусловие: **Открыто меню входа/регистрации**

Таблица 4 - Тест-кейс №3. Авторизация

Шаг	Ожидаемый результат
Выбрать опцию «Вход» в главном меню	Появляется форма с полями для ввода
Ввести в поле «Введите логин»	В поле «Введите логин» отображается введенный логин
Ввести в поле «Введите пароль»	В поле «Введите пароль» отображается введенный пароль, зашифрованный символом «*»
Нажать на кнопку «Далее» & введенный логин и пароль не совпадают с данными в БД	Ошибка - «Неверный логин/пароль»
Нажать на кнопку «Далее» & Оставить любое поле пустым	Ошибка - «Поле не может быть пустым»
Нажать на кнопку «Далее» & введенный логин и пароль совпадают с данными в БД	Переход на страницу с выбором комнат/игр
Нажать на кнопку «Назад»	Переход в главное меню

6.4. Тест-кейс №4. Подготовка к онлайн игре

Название: **Подготовка к онлайн игре**

Предусловие: **Открыто главное меню**

Таблица 5 - Тест-кейс №4. Подготовка к онлайн игре

Шаг	Ожидаемый результат
Предусловие: пользователь не зашел в систему/не зарегистрирован.	
В главном меню нажать кнопку «Онлайн игра»	Переход на страницу меню с опциями «Вход», «Регистрация», «Назад»
Нажать на кнопку «Вход»	Переход на страницу авторизации (см. Тест-кейс №2)
Нажать на кнопку «Регистрация»	Переход на страницу регистрации (см. Тест-кейс №1)

Нажать на кнопку «Назад»	Возврат к главному меню
Предусловие: пользователь зашел в систему. Выбор уже созданной комнаты	
В главном меню нажать кнопку «Онлайн игра»	Переход к выбору комнат
Нажать кнопку «Назад»	Возврат к главному меню
Нажать кнопку «Присоединиться» к комнате	Отображение карточки с ролью, полученной автоматически
Нажать на стрелку назад	Возвращение к выбору комнаты
Нажать на стрелку вперед	Переход на страницу с выбором персонажа
Нажать на карточку выбранного персонажа	Переход к игровому процессу
Предусловие: пользователь зашел в систему. Создание комнаты	
В главном меню нажать кнопку «Онлайн-игра»	Переход к выбору комнат
Нажать кнопку «Назад»	Переход в главное меню
Нажать кнопку «Создать комнату»	Переход к выбору количества игроков
Нажать на кнопку «-» при количестве игроков = 4	Количество игроков не меняется
Нажать на кнопку «+» при количестве игроков = 8	Количество игроков не меняется
Выбрать количество игроков от 4 до 8	Отображается выбранное количество игроков
Нажать на стрелку назад	Переход к странице с выбором комнат
Нажать на стрелку вперед	Переход к странице с игроками, которые зашли в комнату
Нажать кнопку «Назад»	Переход к выбору количества игроков
Нажать «Начать игру», когда комната не полная	Кнопка неактивна, нажать на нее нельзя
Нажать «Начать игру», когда не все	Кнопка неактивна, нажать на нее нельзя

пользователи готовы	
Нажать «Начать игру», когда готовы все пользователи и комната полная	Отображение карточки с ролью, полученной автоматически
Нажать на стрелку назад	Возвращение к странице с игроками, которые зашли в комнату
Нажать на стрелку вперед	Переход на страницу с выбором персонажа
Нажать на карточку выбранного персонажа	Переход к игровому процессу
Нажать на стрелку назад	Отображение карточки с ролью

6.5. Тест-кейс №5. Подготовка к одиночной игре

Название: **Подготовка к одиночной игре**

Предусловие: **Открыто главное меню**

Таблица 6 - Тест-кейс №5. Подготовка к одиночной игре

Шаг	Ожидаемый результат
В главном меню нажать кнопку «Одиночная игра»	Переход на страницу, где отображаются ранее созданные игры и можно создать новую
Нажать кнопку «Назад»	Переход в главное меню
Нажать кнопку запуска ранее созданной игры	Переход к игровому процессу
Нажать кнопку удаления ранее созданной игры	Игра успешно удалена
Нажать кнопку «Новая игра»	Переход к выбору количества игроков и уровня сложности
Нажать на кнопку «-» при количестве игроков = 4	Количество игроков не меняется
Нажать на кнопку «+» при количестве игроков = 8	Количество игроков не меняется
Нажать на кнопку «-»	Отображается выбранное количество игроков уменьшается на 1

Нажать на кнопку «+»	Отображается выбранное количество игроков увеличивается на 1
Нажать на кнопку «-» при уровне сложности = «лёгкий»	Уровень сложности не меняется
Нажать на кнопку «+» при уровне сложности = «тяжёлый»	Уровень сложности не меняется
Нажать на кнопку «-»	Отображаемый выбранный уровень сложности уменьшается на «1» («тяжёлый» -> «средний» -> «лёгкий»)
Нажать на кнопку «+»	Отображаемый выбранный уровень сложности увеличивается на «1» («лёгкий» -> «средний» -> «тяжёлый»)
Нажать на стрелку назад	Переход к странице с выбором игр
Нажать на стрелку вперед	Переход к выбору роли
Нажать на карточку выбранной роли	Отображение выбранной роли
Нажать на кнопку «Получить случайную роль»	Отображение полученной случайной роли
Нажать на стрелку назад	Возвращение к выбору количества персонажей и уровня сложности
Нажать на стрелку вперед	Переход на страницу с выбором персонажа
Нажать на карточку выбранного персонажа	Переход к игровому процессу
Нажать на стрелку назад	Отображение выбранной роли

6.6. Тест-кейс №6. Бэнглопедия

Название: **Бэнглопедия**

Предусловие: **Открыто главное меню**

Таблица 7 - Тест-кейс №6. Бэнглопедия

Шаг	Ожидаемый результат
В главном меню нажать кнопку «Бэнглопедия»	Переход к меню Бэнглопедии
Нажать на кнопку «Как играть»	Отображение правил игры

Нажать на стрелки перелистывания страниц	Страницы успешно перелистываются
Нажать на стрелку назад	Переход к меню Бэнглопедии
Нажать на кнопку «Карты ролей»	Отображение карт ролей и верхнего меню, из которого можно выбрать карты персонажей и игровые карты
Нажать в верхнем меню на «Персонажи»	Отображение карт персонажей
Нажать в верхнем меню на «Игровые карты»	Отображение игровых карт
Нажать на стрелку назад	Переход к меню Бэнглопедии
Нажать на кнопку «Карты персонажей»	Отображение карт персонажей и верхнего меню, из которого можно выбрать карты ролей и игровые карты
Переключить в верхнем меню виды карт	Карты успешно меняются
Нажать на стрелку назад	Переход к меню Бэнглопедии
Нажать на кнопку «Игровые карты»	Отображение игровых карт и верхнего меню, из которого можно выбрать карты ролей и карты персонажей
Переключить в верхнем меню виды карт	Карты успешно меняются
Нажать на стрелку назад	Переход к меню Бэнглопедии
Нажать на кнопку «В главное меню»	Переход в главное меню

6.7. Тест-кейс №7. Настройки

Название: **Настройки**

Предусловие: **Открыто главное меню**

Таблица 8 - Тест-кейс №7. Настройки

Шаг	Ожидаемый результат
В главном меню нажать кнопку «Настройки»	Переход на страницу настроек
Нажать на переключатель включения/выключения громкости	Громкость выключится/включится в зависимости от положения переключателя

Увеличить/уменьшить общий звук	Новые настройки звука применились
Увеличить/уменьшить «Громкость музыки»	Новые настройки звука применились
Увеличить/уменьшить «Громкость звуков»	Новые настройки звука применились

6.8. Тест-кейс №8.1. Подготовка к началу игры

Название: **Игровой процесс. Подготовка к началу игры**

Предусловие: **Запущена новая одиночная или онлайн игра**

Таблица 9 - Тест-кейс №8.1. Подготовка к началу игры

Шаг	Ожидаемый результат
Нажать в верхнем правом углу на иконку меню	Переход к меню, где можно осуществить выход из игры
Нажать в верхнем правом углу на иконку настроек	Переход к настройкам игры
Нажать в верхнем правом углу на иконку Бэнгопедии	Переход к меню Бэнгопедии
Подготовка к началу игры	У каждого игрока присутствует одна карта роли.
	Карта роли «Шериф» держится в открытую, карты ролей остальных игроков скрыты
	У каждого игрока присутствует одна карта персонажей. Все карты персонажей держатся в открытую
	У каждого персонажа в начале игры столько жизней, сколько патронов указано на его карте персонажа. У шерифа плюс одна жизнь в начале игры
	Игроки получили в закрытую столько игровых карт, сколько у них жизней (патронов) в начале игры
Ход игры	Первый ход делает шериф, игра идет по часовой стрелке

	В зависимости от того, куда игрок делает ход в ту ячейку и кладется карта
	Количество патронов игрока уменьшается при попадании в игрока

6.9. Тест-кейс №8.2. Ход игрока. Розыгрыш карт

Название: **Игровой процесс. Ход игрока. Розыгрыш карт**

Предусловие: **Запущена одиночная или онлайн игра**

Таблица 10 - Тест-кейс №8.2. Ход игрока. Розыгрыш карт

Шаг	Ожидаемый результат
Ход переходит игроку	Две карты из колоды в закрытую переходят игроку
Игрок делает ход и хочет использовать две карты «Бэнг» за ход	Карта неактивна, её нельзя сыграть
Игрок делает ход картой, которая уже есть в игре	Карта неактивна, её нельзя сыграть
Игрок делает ход картой оружие, хотя карта с оружием уже есть в игре	Карта неактивна, её нельзя сыграть
Игрок делает ход картой «Пиво», но в игре осталось только два игрока	Карта неактивна, её нельзя сыграть
В нижней центральной части панели нажать на одну из карт, находящуюся на руках	При нажатии на выбранную карту, она увеличивается
Перенести одну из карт с синей рамкой, которую планируется применить, на панель перед игроком	Применяется действие карты на игрока. В зависимости от типа карты, она попадает в один из трёх закруглённых квадратов на поле. Значение в данном квадрате увеличивается на 1
Перенести одну из карт с синей рамкой, которую планируется применить на другого игрока, в область данного игрока	Применяется действие карты на выбранного игрока. В зависимости от типа карты, она попадает в один из трёх закруглённых квадратов на поле. Значение в данном квадрате увеличивается на 1
Перенести одну из карт с коричневой рамкой, которую планируется	Применяется эффект карты на игрока. После розыгрыша карты, она попадает в

использовать, на панель перед игроком	стопку сброса
Перенести одну из карт с коричневой рамкой, которую планируется применить на другого игрока, в область данного игрока	Применяется эффект карты на выбранного игрока. После розыгрыша карты, она попадает в стопку сброса
Игрок нажимает кнопку «Завершить ход»	Если у игрока на руке карт больше, чем жизней (патронов) в данный момент, то игрок должен сбросить лишние карты. В противном случае, ход переходит следующему игроку
Игрок сбрасывает карты после хода	Игрок должен сбросить лишние карты. В конце хода у игрока должно остаться на руке столько карт, сколько у него жизней (патронов) в данный момент

6.10. Тест-кейс №8.3. Состояния игры

Название: **Игровой процесс. Состояния игры**

Предусловие: **Запущена одиночная или онлайн игра**

Таблица 11 - Тест-кейс №8.3. Состояния игры

Шаг	Ожидаемый результат
Игрок теряет последнюю единицу жизни	Раскрывается карта роли игрока и все его карты уходят в сброс. Игрок выбывает из игры, но может наблюдать за процессом игры
Игрок с ролью «Шериф» убивает игрока с ролью «Помощник»	Все карты игрока с ролью «Шериф» уходят в сброс
Любой игрок убивает игрока с ролью «Бандит»	Игрок, убивший игрока с ролью «Бандит» забирает три карты с колоды
Игрок с ролью «Шериф» убит	Игра окончена победой бандитов (в случае если последним выжившим остался ренегат, то игра заканчивается победой ренегата). Результаты игры отображаются в специальном окне. Также раскрываются роли всех игроков
Игроки с ролью «Бандит» и «Ренегат» убиты	Игра окончена победой шерифа и его помощников. Результаты игры отображаются в специальном окне. Также раскрываются роли всех игроков

Игровые карты	Все игровые карты отображаются корректно. Эффект от карт и действия игрока соответствуют карте.
Карты персонажей	Все карты отображаются корректно. Количество здоровья соответствует карте. Возможности персонажа и действия игрока соответствуют карте.
Игрок выходит из онлайн-игры, нажав на кнопку меню	Раскрывается карта роли игрока и сбрасываются все его карты
Игрок выходит из одиночной игры, нажав кнопку меню	Игра автоматически сохраняется, игрок позже может вернуться к игре без потери данных

6.11. Тест-кейс №9. Игровая статистика

Название: **Игровая статистика**

Предусловие: **Открыто главное меню**

Таблица 12 - Тест-кейс №9. Игровая статистика

Шаг	Ожидаемый результат
В главном меню нажать на кнопку «Игровая статистика»	Переход к игровой статистике
Просмотреть игровую статистику	<p>Отображаются следующие пункты игровой статистики:</p> <ol style="list-style-type: none"> 1) Количество сыгранных игр в одиночной игре 2) Количество незаконченных игр в одиночной игре 3) Количество побед в одиночной игре 4) Количество поражений в одиночной игре 5) Процент побед в одиночной игре 6) Количество сыгранных игр в онлайн игре 7) Количество побед в онлайн игре 8) Количество поражений в онлайн игре 9) Процент побед в онлайн игре <p>Количество игр отображается корректно, если игр не было то отображается цифра 0. Процент побед в играх отображается корректно, если игр не было, то отображается символ «-».</p>
Нажать на стрелку назад	Переход в главное меню

6.12. Тест-кейс №10. Обратная связь

Название: **Обратная связь**

Предусловие: **Открыто главное меню**

Таблица 13 - Тест-кейс №10. Обратная связь

Шаг	Ожидаемый результат
В главном меню нажать на кнопку «Обратная связь»	Переход к обратной связи. Перед пользователем появляется специальное окно для ввода обратной связи
Поле для ввода текста оставить пустым и нажать на кнопку «Отправить»	Ничего не происходит: кнопка «Отправить» неактивна
В окне ввода ввести сообщение с описанием ошибки/отзыва/предложения об игре	В окне ввода отображается текстовое сообщение
Ввести текст в поле ввода и нажать на кнопку «Отправить»	«Ваш запрос отправлен технической поддержке»
Нажать на кнопку «Назад»	Переход в главное меню

7. USE CASE

7.1. Диаграмма сценария использования

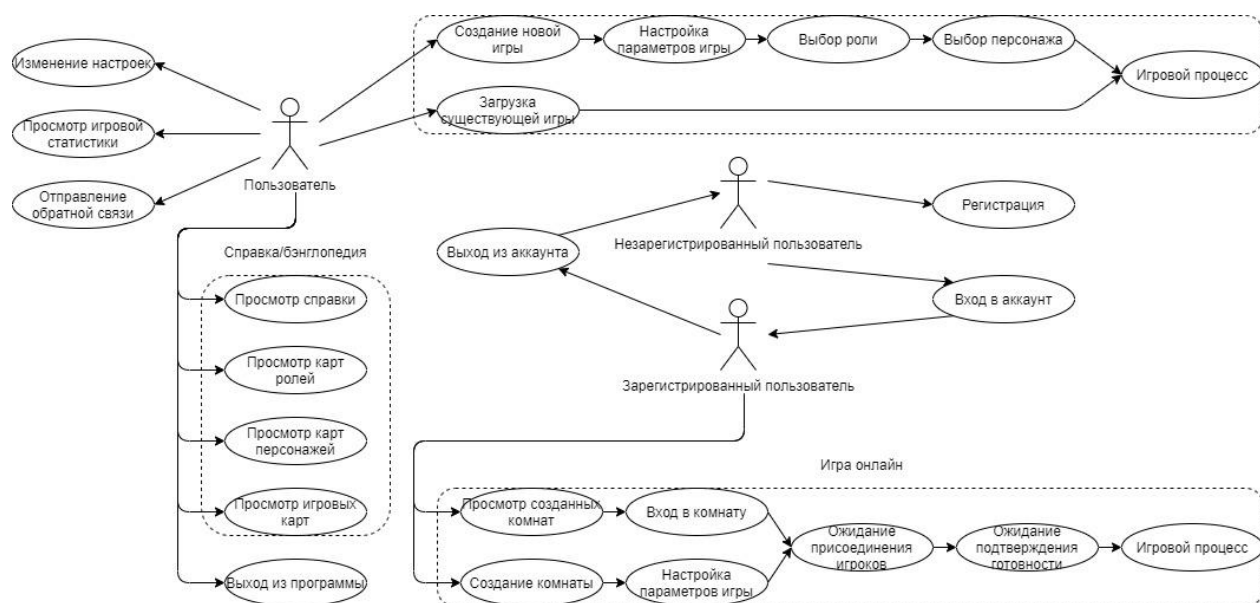


Рисунок 28 - Сценарий использования

7.2. Сценарий №1. Регистрация пользователя

Таблица 14 - Сценарий №1. Регистрация пользователя

Имя сценария	Регистрация пользователя
Цель	Создание аккаунта в онлайн-игре «Бэнг»
Действующее лицо	Незарегистрированный пользователь
Предварительные условия	Запуск карточной игры «Бэнг»

<p>Основной сценарий</p>	<ol style="list-style-type: none"> 1. Незарегистрированный пользователь выбирает пункт «Онлайн игра» главного меню 2. Незарегистрированный пользователь выбирает пункт «Регистрация» в меню 3. Незарегистрированный пользователь заполняет поле «Создать логин» латинскими буквами 4. Незарегистрированный пользователь вводит корректный e-mail в поле «Введите e-mail» 5. Незарегистрированный пользователь заполняет поле «Создайте пароль» 6. Незарегистрированный пользователь заполняет поле «Подтвердите пароль» теми же символами, что и поле «Создайте пароль» 7. Незарегистрированный пользователь нажимает кнопку «Далее» 8. Регистрация прошла успешно
<p>Альтернативный сценарий 1</p>	<ol style="list-style-type: none"> 3. Незарегистрированный пользователь заполняет поле «Создать логин» не латинскими буквами 8. Система выдает ошибку «Это поле может содержать буквы только латинского алфавита»
<p>Альтернативный сценарий 2</p>	<ol style="list-style-type: none"> 3. Незарегистрированный пользователь заполняет поле «Создать логин» неуникальным значением 8. Система выдает ошибку «Данный логин уже используется»

Альтернативный сценарий 3	<p>4. Незарегистрированный пользователь вводит e-mail, зарегистрированный в системе ранее</p> <p>8. Система выдает ошибку «Для данной почты уже существует аккаунт»</p>
Альтернативный сценарий 4	<p>4. Незарегистрированный пользователь вводит некорректный e-mail</p> <p>8. Система выдает ошибку «Введите корректный e-mail»</p>
Альтернативный сценарий 3	<p>6. Незарегистрированный пользователь заполняет поля «Подтвердите пароль» и «Создайте пароль» не совпадающими значениями</p> <p>8. Система выдает ошибку «Пароли не совпадают»</p>

7.3. Сценарий №2. Авторизация

Таблица 15 - Сценарий №2. Авторизация

Имя сценария	Авторизация пользователя
Цель	Вход пользователя в созданный аккаунт в онлайн-игре «Бэнг»
Действующее лицо	Зарегистрированный пользователь
Предварительные условия	Запуск карточной игры «Бэнг»

Основной сценарий	<ol style="list-style-type: none"> 1. Зарегистрированный пользователь выбирает опцию «Вход» в меню 2. Зарегистрированный пользователь заполняет поле «Введите логин» 3. Зарегистрированный пользователь заполняет поле «Введите пароль» 4. Зарегистрированный пользователь нажимает кнопку «Далее» 5. Зарегистрированный пользователь успешно входит в аккаунт игры
Альтернативный сценарий 1	<ol style="list-style-type: none"> 2. Зарегистрированный пользователь некорректно заполняет поле «Введите логин» <p>и/или</p> <ol style="list-style-type: none"> 3. Зарегистрированный пользователь некорректно заполняет поле «Введите пароль» 5. Система выдает ошибку «Неверный логин/пароль»
Альтернативный сценарий 2	<ol style="list-style-type: none"> 2. Зарегистрированный пользователь оставляет поле «Введите логин» пустым <p>и/или</p> <ol style="list-style-type: none"> 3. Зарегистрированный пользователь оставляет поле «Введите пароль» пустым 5. Система выдает ошибку «Поле не может быть пустым»

7.4. Сценарий №3. Начать онлайн-игру

Таблица 16 - Сценарий №3. Начать онлайн-игру

Имя сценария	Начать онлайн-игру
Цель	Успешное осуществление начала игрового процесса в онлайн-игре

	«Бэнг»
Действующие лица	Зарегистрированный пользователь
Предварительные условия	Запуск карточной игры «Бэнг»
Основной сценарий	<ol style="list-style-type: none"> 1. Зарегистрированный пользователь нажимает кнопку «Онлайн игра» 3. Зарегистрированный пользователь просматривает созданные комнаты 4. Зарегистрированный пользователь входит в комнату, нажав кнопку «Присоединиться» 5. Зарегистрированный пользователь ожидает присоединения игроков 6. Зарегистрированный пользователь ожидает подтверждения игроков 7. Зарегистрированный пользователь выбирает роль 8. Зарегистрированный пользователь выбирает персонажа 9. Зарегистрированный пользователь начинает игровой процесс
Альтернативный сценарий 1	<ol style="list-style-type: none"> 1. Зарегистрированный пользователь нажимает кнопку «Онлайн игра» 2. Зарегистрированный пользователь создает комнату, нажав кнопку «Создать комнату» 4. Зарегистрированный пользователь выбирает количество игроков от 4 до 8 5. Зарегистрированный пользователь ожидает присоединения игроков 6. Зарегистрированный пользователь ожидает подтверждения игроков 7. Зарегистрированный пользователь выбирает роль 8. Зарегистрированный пользователь выбирает персонажа

	9. Зарегистрированный пользователь начинает игровой процесс
Альтернативный сценарий 2	<p>4.1 Зарегистрированный пользователь нажимает на кнопку «-» при количестве игроков = 4</p> <p>и/или</p> <p>4.1 Зарегистрированный пользователь нажимает на кнопку «+» при количестве игроков = 8</p> <p>4.2 Количество игроков не меняется</p>
Альтернативный сценарий 3	<p>5.1 Зарегистрированный пользователь нажимает «Начать игру», когда комната не полная</p> <p>5.2 Система выдает ошибку «Дождитесь присоединения всех пользователей»</p>
Альтернативный сценарий 4	<p>6.1 Зарегистрированный пользователь нажимает «Начать игру», когда не все пользователи готовы</p> <p>6.2 Система выдает ошибку «Дождитесь готовности всех пользователей»</p>

7.5. Сценарий №4. Начать одиночную игру

Таблица 17 - Сценарий №4. Начать одиночную игру

Имя сценария	Начать одиночную игру
Цель	Успешное осуществление одиночной игры
Действующее лицо	Пользователь
Предварительные условия	Запуск карточной игры «Бэнг»
Основной сценарий	<ol style="list-style-type: none">1. Пользователь выбирает пункт “Одиночная игра” в главном меню2. Пользователь выбирает пункт “Создать комнату” в появившемся окне3. Пользователь выбирает количество игроков4. Пользователь выбирает уровень сложности5. Пользователь нажимает стрелку “Вперёд”6. Пользователь выбирает одну из предложенных ролей и/или6. Пользователь нажимает на кнопку “Получить случайную роль”7. Пользователю отображается полученная роль, после чего пользователь нажимает стрелку “Далее”8. Пользователю отображаются три случайные карты персонажей после чего пользователь выбирает одну из предложенных карт персонажей9. Пользователь начинает игровой процесс

Альтернативный сценарий 1	<ol style="list-style-type: none"> 1. Пользователь выбирает пункт “Одиночная игра” в главном меню 2. Пользователь загружает существующую игру 3. Пользователь начинает игровой процесс
---------------------------	---

7.6. Сценарий №5. Реализация игрового процесса

Таблица 18 - Сценарий №5. Реализация игрового процесса

Имя сценария	Реализация игрового процесса
Цель	Провести игровой процесс до выполнения цели любого игрока
Действующее лицо	Пользователь
Предварительные условия	Осуществление предыгрового процесса

Основной сценарий	<ol style="list-style-type: none"> 1. Пользователю раздается определенное количество карт, в соответствии с количеством жизни персонажа 2. Пользователь ожидает очереди для выполнения хода 3. Пользователь разыгрывает разовую (коричневую) карту, перенося ее на поле соперника 4. Пользователь нажимает кнопку «Окончить ход» 5. Пользователь ожидает очереди для выполнения следующего хода 6. Пользователь теряет последнюю единицу жизни - игра завершается 7. Пользователь выходит из игры, нажав на кнопку меню
Альтернативный сценарий 1	<ol style="list-style-type: none"> 3.1 Пользователь разыгрывает разовую (коричневую) карту, перенося ее на свое поле 3.2 Пользователь разыгрывает постоянную (синюю) карту, перенося ее на поле соперника 3.3 Пользователь разыгрывает постоянную (синюю) карту, перенося ее на свое поле
Альтернативный сценарий 2	<ol style="list-style-type: none"> 6.1 Пользователь игравший за роль «Шериф» убит – игра завершается 6.1 Пользователи игравшие за роль «Бандиты» и «Регенераты» убиты – игра завершается

7.7. Сценарий №6. Выход пользователя

Таблица 19 - Сценарий №6. Выход пользователя

Имя сценария	Выход пользователя
Цель	Выход пользователя из карточной игры «Бэнг»

Действующее лицо	Пользователь
Предварительные условия	Запуск карточной игры «Бэнг»
Основной сценарий	1. Зарегистрированный пользователь выходит из онлайн-игры, нажав кнопку «Выход»

7.8. Сценарий №7. Работа со справкой/бэнглопедией

Таблица 20 - Сценарий №7. Работа со справкой/бэнглопедией

Имя сценария	Работа со справкой/бэнглопедией
Цель	Изучение «бэнглопедии» в карточной игре «Бэнг». Ознакомление со справками и картами игры.
Действующее лицо	Пользователь
Предварительные условия	Запуск карточной игры «Бэнг»
Основной сценарий	<ol style="list-style-type: none"> 1. Пользователь выбирает пункт «Бэнглопедия» главного меню 2. Пользователь выбирает пункт «Как играть» меню 3. Пользователь просматривает правила игры
Альтернативный сценарий 1	<ol style="list-style-type: none"> 1. Пользователь выбирает пункт «Бэнглопедия» главного меню 2. Пользователь выбирает пункт «Карты ролей» меню 3. Пользователь просматривает карты ролей

Альтернативный сценарий 2	<ol style="list-style-type: none"> 1. Пользователь выбирает пункт «Бэнглопедия» главного меню 2. Пользователь выбирает пункт «Карты персонажей» меню 3. Пользователь просматривает карты персонажей
Альтернативный сценарий 3	<ol style="list-style-type: none"> 1. Пользователь выбирает пункт «Бэнглопедия» главного меню 2. Пользователь выбирает пункт «Игровые карты» меню 3. Пользователь просматривает игровые карты

7.9. Сценарий №8. Работа с настройками

Таблица 21 - Сценарий №8. Работа с настройками

Имя сценария	Работа с настройками
Цель	Изменить настройки карточной игры «Бэнг»
Действующее лицо	Пользователь
Предварительные условия	Запуск карточной игры «Бэнг»
Основной сценарий	<ol style="list-style-type: none"> 1. Пользователь переходит в настройки, нажав кнопку «Настройки» 2. Пользователь нажимает на переключатель включения/выключения громкости 3. Пользователь увеличивает/уменьшает общий звук 4. Пользователь увеличивает/уменьшает «Громкость музыки» 5. Пользователь увеличивает/уменьшает «Громкость звуков» 6. Пользователь изменяет язык на русский/английский

7.10. Сценарий №9. Осуществление обратной связи

Таблица 22 - Сценарий №9. Осуществление обратной связи

Имя сценария	Осуществление обратной связи
Цель	Получение отзывов и пожеланий от пользователей игры
Действующее лицо	Пользователь
Предварительные условия	Вход пользователя в аккаунт онлайн-игры «Бэнг»
Основной сценарий	<ol style="list-style-type: none">1. Пользователь выбирает пункт «Обратная связь» главного меню2. Пользователь вводит отзыв в специальную форму3. Пользователь нажимает кнопку «Отправить» отправляет отзыв на почту для комментариев игроков

7.11. Сценарий №10. Просмотр игровой статистики

Таблица 23 - Сценарий №10. Просмотр игровой статистики

Имя сценария	Просмотр игровой статистики
Цель	Ознакомление пользователя со статистическими данными по проведенным играм

Действующее лицо	Пользователь
Предварительные условия	Вход пользователя в аккаунт онлайн-игры «Бэнг»
Основной сценарий	<ol style="list-style-type: none"> 1. Пользователь выбирает пункт «Игровая статистика» главного меню 2. Пользователь просматривает статистические данные игры: <ol style="list-style-type: none"> 1) Количество сыгранных игр в одиночной игре 2) Количество незаконченных игр в одиночной игре 3) Количество побед в одиночной игре 4) Количество поражений в одиночной игре 5) Процент побед в одиночной игре 6) Количество сыгранных игр в онлайн игре 7) Количество побед в онлайн игре 8) Количество поражений в онлайн игре 9) Процент побед в онлайн игре

8. ИНСТРУКЦИЯ ПО СБОРКЕ

Чтобы осуществить сборку и установку на свое устройство, нужно иметь глобально установленный `node.js` на устройстве, и в командной строке папки `Project` написать `npm i` (дождаться установки всех необходимых пакетов) а затем `npm start` чтобы запустить локальный сервер по адресу `localhost:3000`. Node Package Manager осуществит загрузку и установку всех необходимых пакетов для работы среды. Готовый вариант игры доступен на reznikov.beget.tech.

9. ИСПОЛЬЗУЕМЫЕ ИНСТРУМЕНТЫ

Тип программы: приложение с установочным пакетом

Приложение для создания прототипа: Balsamiq Wireframes

Front-end: JavaScript

Back-end: Python

База данных: MS SQL

Хостинг: <http://reznikov.beget.tech>

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы:

- разработано архитектурное решение для реализации клиент-серверного приложения игры «Бэнг!».
- Определены пути реализации программного обеспечения в процессе реализации сложных проектов.
- Созданы макеты основных компонентов программного обеспечения

В процессе прохождения практики, были решены поставленные задачи. А именно:

- Разработана архитектура выбранной системы;
- Выделены компоненты, реализация которых сложна или неочевидна;
- Проведено эскизное проектирование выбранных компонентов;
- Задokumentированы полученные результаты.

Для достижения поставленных целей и задач были сформулированы требования к системе, составлен сценарий использования данного продукта. Было произведено исследование, в ходе которого были определены инструменты разработки: разработка веб-сервиса реализовывалась на языке JavaScript. Back - end осуществлен на языке Python. Также была разработана ER-модель системы и с помощью SQL Server Management Studio будет реализована база данных.

Таким образом, поставленные цели и задачи были успешно выполнены.

10. ЛИЧНЫЙ ВКЛАД УЧАСТНИКА КОМАНДЫ.

Альмухаметова Т. С. –Аналитик/Тестировщик/Доки:

- Предварительная обработка данных
- Составление тестов
- Документация отчетов

Андреев С. Д. – Разработчик (базы данных):

- Разработка схемы БД;
- Разработка серверной части системы;
- Проведение тестирования БД.

Ефремова М. С. – Тестировщик:

- Разработка части требований к системе;
- Разработка test case;
- Проведение тестирования процедур БД;
- Проведение ручного тестирования.

Москвитина Л. С. –Аналитик/Тестировщик/Доки

- Работа с изображениями
- Написание сценария use-case
- Разработка class request

Новицкий Д. А. – Team-lead /разработчик (back-end):

- Организация командной деятельности;
- Разработка прототипа программы;
- Разработка базы данных;
- Разработка back-end
- Консультирование с участниками команды.

Резников К. П. – Разработчик (front-end):

- Разработка front-end
- Разработка логики игры

11. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Настольная игра “Бэнг” — Википедия. URL: <https://ru.wikipedia.org/wiki/Бэнг!> (дата обращения 21.03.21).
2. Настольная игра “Бэнг”. Правила. URL: https://docviewer.yandex.ru/view/103700023/?page=1&*=hrJzGKhrhOEnLens2sAjGEdV2yB7InVybcI6Imh0dHBzOi8vd3d3Lmlncm92ZWQucnUvcnVsZXNmvcnVsZXNfYmFuZy1ydXNfcnVzLnBkZiIsInRpdGx1IjoicnVsZXNfYmFuZy1ydXNfcnVzLnBkZiIsIm5vaWZyYW11Ijp0cnVILCJ1aWQiOiIxMDM3MDAwMjMiLCJ0cyI6MTYxNjM0ODk1NzMzMzMCwieXUiOiIzNDg4MzIwOTcxNjEyNTQzMzA4Iiwic2VycFBhcmFtcyI6Imxhbmc9cnUmdG09MTYxNjM0ODk1MyZ0bGQ9cnUmbmFtZT1ydWxlc19iYW5nLXJ1c19ydXMucGRmJnRleHQ9JUQwJUIBQUQwJUIwJUQxJTgwJUQxJTgyJUQwJUJFJUQxJTg3JUQwJUJEJUQwJUIwJUQxJTg3JUQwJUJGJUQxJTgwJUQwJUIwJUQwJUJyJUQwJUI4JUQwJUJCJUQwJUIwJnVyYmFtZD1odHRwcyUzQS8vd3d3Lmlncm92ZWQucnUvcnVsZXNmvcnVsZXNfYmFuZy1ydXNfcnVzLnBkZiZscj0yMTMmbWltZT1wZGYmbDEwbj1ydSZzaWduPWFiYzYxNmY3ZmEzMmM0NDdkODYxNWQ3ZTIyNWl3ZGMzMjtleW5vPTAifQ%3D%3D&lang=ru (дата обращения: 21.03.21).

12. ПРИЛОЖЕНИЕ 1. СКРИПТ СОЗДАНИЯ БД.

```
CREATE TABLE [dbo].[Player]
(
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [weapon_ID] [int] NULL,
    [player_move] [bit] NULL,
    [win] [bit] NULL,
    [character_ID] [int] NULL,
    [role_ID] [int] NULL,
    [user_ID] [int] NULL,
    [room_ID][int] NULL,
    [is_ready] [bit] NOT NULL,
    [queue] [int] NULL,
    [additional_attack_range] [int] NOT NULL,
    [additional_defence_range] [int] NOT NULL
)
GO
ALTER TABLE [dbo].[Player]
ADD CONSTRAINT [PK_Player_ID] PRIMARY KEY CLUSTERED ([ID])
GO

CREATE TABLE [dbo].[Weapon]
(
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [name] [nvarchar](50) NOT NULL,
    [base_weapon] [bit] NOT NULL,
    [bang_player] [bit] NOT NULL,
    [firing_range] [int] NOT NULL,
    [endless_bang] [bit] NOT NULL,
    [weapon_card_ID] [int] NULL
)
GO
ALTER TABLE [dbo].[Weapon]
ADD CONSTRAINT [PK_Weapon_ID] PRIMARY KEY CLUSTERED ([ID])
GO

CREATE TABLE [dbo].[Character]
(
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [characters_ID] [int] NOT NULL,
    [alive] [bit] NOT NULL,
```

```

        [lives][int] NOT NULL

    )
GO
ALTER TABLE [dbo].[Character]
ADD CONSTRAINT [PK_Character_ID] PRIMARY KEY CLUSTERED ([ID])
GO

CREATE TABLE [dbo].[Characters]
(
    [ID] [int] NOT NULL,
    [name] [nvarchar](50) NOT NULL,
    [card_image] [bit] NULL,
    [card_back] [bit] NULL,
    [description][nvarchar](500) NULL,
    [max_lives] [int] NOT NULL
)
GO
ALTER TABLE [dbo].[Characters]
ADD CONSTRAINT [PK_Characters_ID] PRIMARY KEY CLUSTERED ([ID])
GO

CREATE TABLE [dbo].[Role]
(
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [roles_ID] [int] NOT NULL,
    [status] [nvarchar](50) NOT NULL
)
GO
ALTER TABLE [dbo].[Role]
ADD CONSTRAINT [PK_Role_ID] PRIMARY KEY CLUSTERED ([ID])
GO

CREATE TABLE [dbo].[Roles]
(
    [ID] [int] NOT NULL,
    [name] [nvarchar](50) NOT NULL,
    [card_image] [bit] NULL,
    [card_back] [bit] NULL,
    [description] [nvarchar](500) NULL
)

```

```

GO
ALTER TABLE [dbo].[Roles]
ADD CONSTRAINT [PK_Roles_ID] PRIMARY KEY CLUSTERED ([ID])
GO

CREATE TABLE [dbo].[User]
(
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [e-mail] [nvarchar](50) NOT NULL,
    [password] [nvarchar](50) NOT NULL,
    [nick] [nvarchar](50) NOT NULL,
    [achievements_ID] [int] NULL
)
GO
ALTER TABLE [dbo].[User]
ADD CONSTRAINT [PK_User_ID] PRIMARY KEY CLUSTERED ([ID])
GO

CREATE TABLE [dbo].[Achievements]
(
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [single_player_finished] [int] NOT NULL,
    [single_player_unfinished] [int] NOT NULL,
    [single_player_wins] [int] NOT NULL,
    [single_player_defeats] [int] NOT NULL,
    [single_player_wins_percentage] [real] NULL,
    [multi_player_finished] [int] NOT NULL,
    [multi_player_wins] [int] NOT NULL,
    [multi_player_defeats] [int] NOT NULL,
    [multi_player_wins_percentage] [real] NULL
)
GO
ALTER TABLE [dbo].[Achievements]
ADD CONSTRAINT [PK_Achievements_ID] PRIMARY KEY CLUSTERED ([ID])
GO

CREATE TABLE [dbo].[Room]
(
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [status] [int] NOT NULL,
    [play_time] [time] NULL,
    [count_of_players] [int] NOT NULL,

```

```

        [dropping_ID] [int] NULL,
        [deck_ID] [int] NULL,
        [max_count_of_players] [int] NOT NULL,
        [owner_ID] [int] NULL
    )
GO
ALTER TABLE [dbo].[Room]
ADD CONSTRAINT [PK_Room_ID] PRIMARY KEY CLUSTERED ([ID])
GO

CREATE TABLE [dbo].[Dropping]
(
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [count_of_cards] [int] NOT NULL
)
GO
ALTER TABLE [dbo].[Dropping]
ADD CONSTRAINT [PK_Dropping_ID] PRIMARY KEY CLUSTERED ([ID])
GO

CREATE TABLE [dbo].[Deck]
(
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [count_of_cards] [int] NOT NULL
)
GO
ALTER TABLE [dbo].[Deck]
ADD CONSTRAINT [PK_Deck_ID] PRIMARY KEY CLUSTERED ([ID])
GO

CREATE TABLE [dbo].[Card]
(
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [cards_ID] [int] NOT NULL,
    [player_ID] [int] NULL,
    [room_ID] [int] NOT NULL,
    [card_location] [int] NOT NULL,
    [index_number] [int] NOT NULL
)
GO
ALTER TABLE [dbo].[Card]
ADD CONSTRAINT [PK_Card_ID] PRIMARY KEY CLUSTERED ([ID])

```


GO

CREATE TABLE [dbo].[Cards]

(

[ID] [int] NOT NULL,
[name] [nvarchar](50) NOT NULL,
[card_image] [bit] NULL,
[card_back] [bit] NULL,
[description] [nvarchar](500) NULL,
[color] [nvarchar](50) NOT NULL,
[suit] [nvarchar](50) NOT NULL,
[rating] [nvarchar](50) NOT NULL

)

GO

ALTER TABLE [dbo].[Cards]

ADD CONSTRAINT [PK_Cards_ID] PRIMARY KEY CLUSTERED ([ID])

GO

/*-----*/

/*-----TABLES WITHOUT LINKS -----*/

/*-----*/

CREATE TABLE [dbo].[Players_range]

(

[ID] [int] IDENTITY(1,1) NOT NULL,
[player_out_ID] [int] NOT NULL,
[player_in_ID] [int] NOT NULL,
[range] [int] NOT NULL

)

GO

ALTER TABLE [dbo].[Players_range]

ADD CONSTRAINT [PK_Players_range_ID] PRIMARY KEY CLUSTERED ([ID])

GO

/*-----*/

/*-----TABLES END -----*/

/*-----*/

/*-----*/

/*-----FOREIGN KEYS-----*/

/*-----*/

```
ALTER TABLE [dbo].[Player]
WITH CHECK ADD CONSTRAINT [FK_Player_weapon_ID] FOREIGN KEY([weapon_ID])
REFERENCES [dbo].[Weapon] ([ID])
GO
```

```
ALTER TABLE [dbo].[Player]
WITH CHECK ADD CONSTRAINT [FK_Player_character_ID] FOREIGN KEY([character_ID])
REFERENCES [dbo].[Character] ([ID])
GO
```

```
ALTER TABLE [dbo].[Character]
WITH CHECK ADD CONSTRAINT [FK_Character_characters_ID] FOREIGN KEY([characters_ID])
REFERENCES [dbo].[Characters] ([ID])
GO
```

```
ALTER TABLE [dbo].[Player]
WITH CHECK ADD CONSTRAINT [FK_Player_role_ID] FOREIGN KEY([role_ID])
REFERENCES [dbo].[Role] ([ID])
GO
```

```
ALTER TABLE [dbo].[Role]
WITH CHECK ADD CONSTRAINT [FK_Role_roles_ID] FOREIGN KEY([roles_ID])
REFERENCES [dbo].[Roles] ([ID])
GO
```

```
ALTER TABLE [dbo].[Player]
WITH CHECK ADD CONSTRAINT [FK_Player_user_ID] FOREIGN KEY([user_ID])
REFERENCES [dbo].[User] ([ID])
GO
```

```
ALTER TABLE [dbo].[User]
WITH CHECK ADD CONSTRAINT [FK_User_achievements_ID] FOREIGN KEY([achievements_ID])
REFERENCES [dbo].[Achievements] ([ID])
GO
```

```
ALTER TABLE [dbo].[Player]
WITH CHECK ADD CONSTRAINT [FK_Player_room_ID] FOREIGN KEY([room_ID])
REFERENCES [dbo].[Room] ([ID])
GO
```

```
ALTER TABLE [dbo].[Room]
```

```
WITH CHECK ADD CONSTRAINT [FK_Room_dropping_ID] FOREIGN KEY([dropping_ID])
REFERENCES [dbo].[Dropping] ([ID])
GO
```

```
ALTER TABLE [dbo].[Room]
WITH CHECK ADD CONSTRAINT [FK_Room_deck_ID] FOREIGN KEY([deck_ID])
REFERENCES [dbo].[Deck] ([ID])
GO
```

```
ALTER TABLE [dbo].[Card]
WITH CHECK ADD CONSTRAINT [FK_Card_player_ID] FOREIGN KEY([player_ID])
REFERENCES [dbo].[Player] ([ID])
GO
```

```
ALTER TABLE [dbo].[Card]
WITH CHECK ADD CONSTRAINT [FK_Card_room_ID] FOREIGN KEY([room_ID])
REFERENCES [dbo].[Room] ([ID])
GO
```

```
ALTER TABLE [dbo].[Card]
WITH CHECK ADD CONSTRAINT [FK_Card_cards_ID] FOREIGN KEY([cards_ID])
REFERENCES [dbo].[Cards] ([ID])
GO
```

```
/*-----*/
/*-----FOREIGN KEYS END-----*/
/*-----*/
```

```
/*-----*/
/*-----FUNCTIONS/PROCEDURES-----*/
/*-----*/
```

-- 1.1) Çàïðñ òà ðààèñòðàöèþ - àñòü èè ñëüçüààòàëü â ààçà ààíüõ

```
CREATE FUNCTION dbo.Registration_request(@mail nvarchar(50), @password nvarchar(50), @login
nvarchar(50))
RETURNS INT
AS
BEGIN
```

```
DECLARE @Code int
SELECT @Code = 0
```

```

DECLARE @CheckMail nvarchar(50)
SELECT @CheckMail = [e-mail]
FROM dbo.[User]
WHERE [e-mail] = @mail
IF @CheckMail IS NOT NULL
BEGIN
SELECT @Code = 2
END

```

```

DECLARE @CheckNick nvarchar(50)
SELECT @CheckNick = [nick]
FROM dbo.[User]
WHERE [nick] = @login
IF @CheckNick IS NOT NULL
BEGIN
SELECT @Code = 1
END

```

```

RETURN @Code
END
GO

```

-- 1.2) Ðääñððàðëÿ ñëüçîàððäÿ

```

CREATE PROCEDURE dbo.Registration(@mail nvarchar(50), @password nvarchar(50), @login nvarchar(50))
AS
BEGIN
SET NOCOUNT ON;

DECLARE @message int = 0
INSERT INTO [User] ([e-mail], [password], [nick])
VALUES (@mail, @password, @login)
RETURN @message

SET NOCOUNT OFF;
END

```

/*-----END REGISTRATION-----*/

-- 1.3)

/*-----AUTHORIZATION REQUEST-----*/

```

/*-----FUNCTION 2 (3)-----*/
CREATE FUNCTION dbo.Authorization_request (@mail nvarchar(50), @password nvarchar(50))
RETURNS INT
AS
BEGIN

DECLARE @message int = 1

DECLARE @CheckMail nvarchar(50)
SELECT @CheckMail = [e-mail]
FROM dbo.[User]
WHERE [e-mail] = @mail

DECLARE @CheckPassword nvarchar(50)
SELECT @CheckPassword = [password]
FROM dbo.[User]
WHERE [password] = @password

IF @CheckMail IS NOT NULL AND @CheckPassword IS NOT NULL
BEGIN
SELECT @message = 0
END

RETURN @message
END
GO

-- 1.3) Àâôîðèçàòëý ñëüçîâàðåý - åñðü èè ñëüçîâàðåü â áàçå äàííûõ
CREATE FUNCTION dbo.Authorization_request(@mail nvarchar(50), @password nvarchar(50))
RETURNS INT
AS
BEGIN

DECLARE @message int = 1

DECLARE @CheckMail nvarchar(50)
SELECT @CheckMail = [e-mail]
FROM dbo.[User]
WHERE [e-mail] = @mail

DECLARE @CheckPassword nvarchar(50)
SELECT @CheckPassword = [password]

```

```
FROM dbo.[User]
WHERE [password] = @password
```

```
IF @CheckMail IS NOT NULL AND @CheckPassword IS NOT NULL
BEGIN
SELECT @message = 0
END
```

```
RETURN @message
END
GO
```

```
-- 1.4) Ĭřěó÷âĕâ ĩîĕñĕâ êñîâò
```

```
CREATE PROCEDURE dbo.Available_rooms
```

```
AS
```

```
BEGIN
```

```
SET NOCOUNT ON;
```

```
SELECT [ID], [count_of_players], [max_count_of_players]
```

```
from dbo.Room
```

```
WHERE dbo.Room.status = 1 AND (dbo.Room.max_count_of_players - dbo.Room.count_of_players) > 0
```

```
SET NOCOUNT OFF;
```

```
END
```

```
/*-----END AVAILABLE ROOMS REQUEST-----*/
```

```
-- 1.5)
```

```
/*-----USER'S ACHIVEMENTS-----*/
```

```
CREATE PROCEDURE dbo.Achivements_request (@mail nvarchar(50), @password nvarchar(50))
```

```
AS
```

```
BEGIN
```

```
SET NOCOUNT ON;
```

```
DECLARE @message int
```

```
SELECT @message = -1
```

```
DECLARE @CheckMail nvarchar(50)
```

```
SELECT @CheckMail = [e-mail]
```

```

FROM dbo.[User]
WHERE [e-mail] = @mail

DECLARE @CheckPassword nvarchar(50)
SELECT @CheckPassword = [password]
FROM dbo.[User]
WHERE [password] = @password

DECLARE @Achievement_ID int

IF @CheckMail IS NOT NULL AND @CheckPassword IS NOT NULL
BEGIN
SELECT @Achievement_ID = achievements_ID FROM [dbo].[User] WHERE [e-mail] = @mail AND password
= @password
SELECT *
FROM [Achievements]
WHERE ID = @Achievement_ID
END

SET NOCOUNT OFF;
END
GO

-- 1.5) Ыіє6÷âĖâ ħĕĕââ âĭĭðèæâĖé äĕÿ âûáðâĭŕâĭ ħĕŭçĭââðâĕÿ
CREATE PROCEDURE dbo.Achivements_request(@mail nvarchar(50), @password nvarchar(50))
AS
BEGIN
SET NOCOUNT ON;

DECLARE @message int
SELECT @message = -1

DECLARE @CheckMail nvarchar(50)
SELECT @CheckMail = [e-mail]
FROM dbo.[User]
WHERE [e-mail] = @mail

DECLARE @CheckPassword nvarchar(50)
SELECT @CheckPassword = [password]
FROM dbo.[User]
WHERE [password] = @password

```

```

DECLARE @Achievement_ID int

IF @CheckMail IS NOT NULL AND @CheckPassword IS NOT NULL
BEGIN
SELECT @Achievement_ID = achievements_ID FROM [dbo].[User] WHERE [e-mail] = @mail AND password
= @password
SELECT *
FROM [Achievements]
WHERE ID = @Achievement_ID
END

SET NOCOUNT OFF;
END
GO

-- 1.6) Țișăreanu ămădău
CREATE PROCEDURE dbo.Creating_room(@mail nvarchar(50), @password nvarchar(50),
@max_count_of_players int)
AS
BEGIN
SET NOCOUNT ON;

declare @owner_ID int = NULL
set @owner_ID =
(
    select [ID]
    from [User]
    where ([User].[e-mail] = @mail) and ([User].[password] = @password)
)

INSERT INTO [Player] ([user_ID], [is_ready], [additional_attack_range], [additional_defence_range])
VALUES (@owner_ID, 0, 0, 0)

declare @player_ID int = IDENT_CURRENT('Player')

INSERT INTO [Room] ([status], [play_time], [count_of_players], [max_count_of_players], [owner_ID])
VALUES (1, '00:00:00', 1, @max_count_of_players, @player_ID)

declare @room_ID [int] = IDENT_CURRENT('Room')

UPDATE [Player]
set [room_ID] = @room_ID where ID = @player_ID

```



```

SELECT @room_ID

SET NOCOUNT OFF;

END

GO

-- 2.1.1) Äíääæâîèà èãðîèà â êñîàðó
CREATE PROCEDURE dbo.Add_player_to_room(@mail nvarchar(50), @password nvarchar(50), @room_ID int)
AS
BEGIN
SET NOCOUNT ON;

declare @user_ID int = NULL
set @user_ID =
(
    select [ID]
    from [User]
    where ([User].[e-mail] = @mail) and ([User].[password] = @password)
)

INSERT INTO [Player] ([user_ID], [is_ready], [additional_attack_range], [additional_defence_range], [room_ID])
VALUES (@user_ID, 0, 0, 0, @room_ID)

SELECT [Room].count_of_players, [Room].max_count_of_players
FROM [Room]
WHERE ([Room].ID = @room_ID)

SET NOCOUNT OFF;

END

GO

-- 2.1.2) Ĭîëó÷âîèà n ñëó÷âéíûõ îäññîâæâé (ääç îîàòðâîéé)
CREATE TABLE [dbo].[For_get_characters]
(
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [cahr_ID] [int] NOT NULL
)
GO
ALTER TABLE [dbo].[For_get_characters]
ADD CONSTRAINT [PK_For_get_characters_ID] PRIMARY KEY CLUSTERED ([ID])
GO

```

```

CREATE PROCEDURE dbo.Getting_characters(@n int)
AS
BEGIN
SET NOCOUNT ON;

TRUNCATE TABLE [For_get_characters]

DECLARE @count_of_characters [int] = 0
DECLARE @This_variables [int] = 0
SELECT @count_of_characters = COUNT(*) FROM [Characters]

IF @n < @count_of_characters
BEGIN
WHILE @n > 0
    BEGIN
        SELECT @This_variables = FLOOR(RAND()*(@count_of_characters-0)+0);
        IF NOT EXISTS(SELECT [cahr_ID] FROM [For_get_characters] WHERE [cahr_ID] =
@This_variables)
            BEGIN
                INSERT INTO [For_get_characters] (cahr_ID)
                VALUES (@This_variables)
                SELECT @n = @n - 1
            END
        END
    END
END
SELECT (cahr_ID) FROM [For_get_characters]

SET NOCOUNT OFF;
END
GO

```

-- 2.1.4) Äíáâëäåæ çà âúáðáíñîî ïððññîàæà è èãðîéó

```

CREATE PROCEDURE dbo.Add_character_to_player(@player_ID int, @characters_ID int)
AS
BEGIN
SET NOCOUNT ON;

INSERT INTO [Character] (characters_ID, alive, lives)
VALUES (@characters_ID, 1, 4)

declare @Character_ID [int] = IDENT_CURRENT('Character')

```

```

UPDATE [Player]
SET [character_ID] = @Character_ID where ID = @player_ID

SET NOCOUNT OFF;
END
GO

-- 2.1.5) Äíáääëâíêâ ðíëè ê èãðíêó
CREATE PROCEDURE dbo.Add_role_to_player(@player_ID int, @roles_ID int)
AS
BEGIN
SET NOCOUNT ON;

INSERT INTO [Role] (roles_ID, [status])
VALUES (@roles_ID, 1)

declare @role_ID [int] = IDENT_CURRENT('Role')

UPDATE [Player]
SET [role_ID] = @role_ID where ID = @player_ID

SET NOCOUNT OFF;
END
GO

-- 2.1.7) Ííëó÷âíêâ ID ííëüçíââðåëý íí ââí íí÷ðå è ìàðíëþ
CREATE PROCEDURE dbo.Get_user_ID(@mail nvarchar(50), @password nvarchar(50))
AS
BEGIN
SET NOCOUNT ON;

SELECT [User].ID
FROM [User]
WHERE ([User].[e-mail] = @mail) and ([User].[password] = @password)

SET NOCOUNT OFF;
END
GO

-- 2.1.8) Ííëó÷âíêâ ID èãðíêâ íí ââí íí÷ðå è ìàðíëþ
CREATE PROCEDURE dbo.Get_player_ID(@mail nvarchar(50), @password nvarchar(50))

```

```

AS
BEGIN
SET NOCOUNT ON;

SELECT [Player].ID
FROM [User]
join [Player]
on [Player].[user_ID] = [User].ID
WHERE ([User].[e-mail] = @mail) and ([User].[password] = @password)

SET NOCOUNT OFF;
END
GO

```

-- 2.1.9) Îñëó÷áñêà ìàêñèìàèüííàî êñêë÷áñòàà æèçíàé ãðñííàæà èãðíêà

```
CREATE PROCEDURE dbo.Get_max_lives(@plater_ID int)
```

```

AS
BEGIN
SET NOCOUNT ON;

```

```

select [Characters].max_lives
from [Player]
join [Character]
on [Player].character_ID = [Character].ID
join [Characters]
on [Character].characters_ID = [Characters].ID
where [Player].ID = @player_ID

```

```

SET NOCOUNT OFF;
END
GO

```

-- 3.1) Óñòàñíàêà çíà÷áñêý òíàà æëý çàääáñíàî èãðíêà

```
CREATE PROCEDURE dbo.Player_turn_state(@player_ID int, @state bit)
```

```

AS
BEGIN
SET NOCOUNT ON;

```

```

UPDATE [Player]
SET [player_move] = @state WHERE ID = @player_ID

```

```
SET NOCOUNT OFF;
```

END

GO

-- 3.2) Îđîääðêà îà îàëë÷èå çàääîííé êàðòòú ó èãđîêà

CREATE FUNCTION dbo.Check_card_availability(@player_ID int, @card_ID int)

RETURNS INT

AS

BEGIN

DECLARE @Flag int

DECLARE @Card_player int

SELECT @Card_player = player_ID from [Card] where ID = @card_ID

IF @Card_player = @player_ID

BEGIN

SELECT @Flag = 1

END

ELSE

BEGIN

SELECT @Flag = 0

END

RETURN @Flag

END

GO

-- 3.3) Îđîääðêà îà îàëë÷èå çàääîííé êàðòòú îãđîíîæà ó èãđîêà

CREATE FUNCTION dbo.Check_character_availability(@player_ID int, @character_ID int)

RETURNS INT

AS

BEGIN

DECLARE @Flag int

DECLARE @Character_player int

SELECT @Character_player = character_ID from [Player] where ID = @player_ID

IF @Character_player = @character_ID

BEGIN

SELECT @Flag = 1

END

```

ELSE
    BEGIN
        SELECT @Flag = 0
    END

RETURN @Flag
END
GO

-- 3.4) Țăiașdêà âîçîîæîñdè âûñòđăăèdòî â èăđîèà (ó÷èòúâââdñý đăññòîýîèă îăæăó èăđîèàîè è ààèüîñdòî ñòđăèüáû èç
îđóæèÿ)
CREATE FUNCTION dbo.Check_shoot_opportunity(@player_ID int, @target_ID int)
RETURNS INT
AS
BEGIN

    DECLARE @Flag int

    DECLARE @firing_range int

    SELECT @firing_range = [firing_range] FROM [Weapon] WHERE [ID] = (SELECT [weapon_ID] FROM [Player]
    WHERE [ID] = @player_ID)
    SELECT @firing_range = @firing_range + [additional_attack_range] FROM [Player] WHERE ID = @player_ID

    DECLARE @range int
    SELECT @range = [range] FROM [Players_range] WHERE [player_in_ID] = @player_ID AND [player_out_ID] =
    @target_ID

    DECLARE @defence_range int

    SELECT @defence_range = [additional_defence_range] FROM [Player] WHERE [ID] = @target_ID
    SELECT @defence_range = @defence_range + @range

    IF @firing_range = @defence_range
        BEGIN
            SELECT @Flag = 1
        END
    ELSE
        BEGIN
            SELECT @Flag = 0
        END
    END

```

END

RETURN @Flag

END

GO

-- 3.5) Àúää÷à èàðòù èãðíéó èç êíëíäü

CREATE PROCEDURE dbo.Set_card_to_player_from_Deck(@player_ID int, @room_ID int)

AS

BEGIN

SET NOCOUNT ON;

declare @card_ID int =

(

select [Card].ID

from [Card]

where (card_location = 2) and (room_ID = @room_ID) and (index_number =

(

select max(index_number)

from [Card]

where (card_location = 2) and (room_ID = @room_ID)

))

)

UPDATE [Card]

SET player_ID = @player_ID,

card_location = 3

WHERE [Card].ID = @card_ID

SELECT @card_ID

SET NOCOUNT OFF;

END

GO

-- 3.6) Àúää÷à èàðòù èãðíéó èç ñáðíñà

CREATE PROCEDURE dbo.Set_card_to_player_from_Dropping(@player_ID int, @room_ID int)

AS

BEGIN

SET NOCOUNT ON;

declare @card_ID int =

```
(
    select [Card].ID
    from [Card]
    where (card_location = 1) and (room_ID = @room_ID) and (index_number =
        (
            select max(index_number)
            from [Card]
            where (card_location = 1) and (room_ID = @room_ID)
        ))
)
```

```
SELECT @card_ID
```

```
UPDATE [Card]
SET player_ID = @player_ID,
card_location = 3
WHERE [Card].ID = @card_ID
```

```
SET NOCOUNT OFF;
END
GO
```

```
-- 3.7) Īāðāōīā èàððú èāðîèà â ñāðîñ
```

```
CREATE PROCEDURE dbo.Send_card_to_Dropping(@card_ID int, @room_ID int)
```

```
AS
```

```
BEGIN
```

```
SET NOCOUNT ON;
```

```
UPDATE [Card]
```

```
SET player_ID = NULL,
```

```
card_location = 1,
```

```
index_number =
```

```
(
    select max([Card].index_number) + 1
    from [Card]
    where (card_location = 1) and (room_ID = @room_ID)
)
```

```
WHERE [Card].ID = @card_ID
```

```
SET NOCOUNT OFF;
END
GO
```



```

-- 3.8) Áúáðáíúé èãðîê ðáðýåð 1 æèçíü
CREATE PROCEDURE dbo.Lose_health(@player_ID int)
AS
BEGIN
SET NOCOUNT ON;

declare @code int = 0
declare @lives int =
(
    select [Character].lives
    from [Player]
    join [Character]
    on [Player].character_ID = [Character].ID
    where [Player].ID = @player_ID
)

UPDATE [Character]
SET lives = lives - 1
WHERE ID =
(
    select [Character].ID
    from [Player]
    join [Character]
    on [Player].character_ID = [Character].ID
    where [Player].ID = @player_ID
)

if(@lives - 1 = 0)
BEGIN
    UPDATE [Character]
    SET alive = 0
    WHERE ID =
    (
        select [Character].ID
        from [Player]
        join [Character]
        on [Player].character_ID = [Character].ID
        where [Player].ID = @player_ID
    )

    SET @code = 10

```

```

END

select @code

SET NOCOUNT OFF;
END
GO

-- 3.9) Ȧúáðáííúé èãðíê âîññòàíáâèèääàò ääèèèöö çäíðíâüÿ
CREATE PROCEDURE dbo.Recovery_health(@player_ID int)
AS
BEGIN
SET NOCOUNT ON;

declare @code int = 1
declare @max_lives int =
(
    select [Characters].max_lives
    from [Player]
    join [Character]
    on [Player].character_ID = [Character].ID
    join [Characters]
    on [Character].characters_ID = [Characters].ID
    where [Player].ID = @player_ID
)
declare @lives int =
(
    select [Character].lives
    from [Player]
    join [Character]
    on [Player].character_ID = [Character].ID
    where ([Player].ID = @player_ID) and ([Character].alive = 1)
)

if(@lives < @max_lives)
BEGIN
    UPDATE [Character]
    SET lives = lives + 1
    where ID =
    (
        select [Character].ID
        from [Player]

```

```

        join [Character]
        on [Player].character_ID = [Character].ID
        where [Player].ID = @player_ID
    )

    SET @code = 0

END

SELECT @code

SET NOCOUNT OFF;

END

GO

-- 3.10) Εἰσαγάγετε ἐνὸς παίκτη ἑνὸς κάρτι
CREATE PROCEDURE dbo.Stealing_card_from_player(@player_ID_to int, @card_ID int)
AS
BEGIN
SET NOCOUNT ON;

UPDATE [Card]
SET player_ID = @player_ID_to,
card_location = 3
WHERE ID = @card_ID

SET NOCOUNT OFF;

END

GO

-- 3.11) Πᾶν κάρτι τοῦ παίκτη, ὁποῦ εἶναι ἐνὸς παίκτη
CREATE PROCEDURE dbo.Get_player_cards(@player_ID int)
AS
BEGIN
SET NOCOUNT ON;

SELECT [Card].ID
FROM [Card]
join [Player]
on [Card].player_ID = [Player].ID
where ([Player].ID = @player_ID) and ([Card].card_location = 3)

SET NOCOUNT OFF;

```

END

GO

-- 3.12) Óñòàñîâêà ïîâîî ïðóæèý äèý èäðîèà

```
CREATE PROCEDURE dbo.Set_weapon(@player_ID int, @name nvarchar(50), @base_weapon bit,  
@firing_range int, @endless_bang bit, @weapon_card_ID int, @room_ID int)
```

```
AS
```

```
BEGIN
```

```
SET NOCOUNT ON;
```

```
declare @old_weapon_card_ID int = NULL
```

```
set @old_weapon_card_ID =
```

```
(
```

```
    select [Weapon].[weapon_card_ID]
```

```
    from [Player]
```

```
    join [Weapon]
```

```
    on [Player].weapon_ID = [Weapon].ID
```

```
    where [Player].ID = @player_ID
```

```
)
```

```
if(@old_weapon_card_ID IS NOT NULL)
```

```
    BEGIN
```

```
        exec dbo.Send_card_to_Dropping
```

```
        @card_ID = @old_weapon_card_ID,
```

```
        @room_ID = @room_ID
```

```
    END
```

```
UPDATE Weapon
```

```
SET [name] = @name,
```

```
[base_weapon] = @base_weapon,
```

```
[firing_range] = @firing_range,
```

```
[endless_bang] = @endless_bang,
```

```
[weapon_card_ID] = @weapon_card_ID
```

```
WHERE ID =
```

```
(
```

```
    select [Weapon].ID
```

```
    from [Weapon]
```

```
    join [Player]
```

```
    on [Player].weapon_ID = [Weapon].ID
```

```
    where [Player].ID = @player_ID
```

```

)

SET NOCOUNT OFF;
END
GO

-- 3.13) Ħieó÷âieâ êaðòó èç êieŕiáú äëý iðíááðèè (iðíááðáííäý êaðòà óóíæò â ñáðíñ)
CREATE PROCEDURE dbo.Get_card_for_checking(@room_ID int)
AS
BEGIN
SET NOCOUNT ON;

declare @next_index_number int = NULL
set @next_index_number =
(
    select max(index_number) + 1
    from [Card]
    where (card_location = 1) and (room_ID = @room_ID)
)
if(@next_index_number IS NULL)
    set @next_index_number = 1

SELECT [Card].[ID], [Cards].[ID], [suit], [rating]
FROM [Cards]
join [Card]
on [Card].cards_ID = [Cards].ID
WHERE (card_location = 2) and (room_ID = @room_ID) and (index_number =
(
    select max(index_number)
    from [Card]
    where (card_location = 2) and (room_ID = @room_ID)
));

WITH Main_select(Card_ID, Cards_ID, suit, rating) as
(
    SELECT [Card].[ID], [Cards].[ID], [suit], [rating]
    FROM [Cards]
    join [Card]
    on [Card].cards_ID = [Cards].ID
    WHERE (card_location = 2) and (room_ID = @room_ID) and (index_number =
(
    select max(index_number)

```

```

        from [Card]
        where (card_location = 2) and (room_ID = @room_ID)
    ))
)

```

```

UPDATE [Card]
SET card_location = 1,
index_number = @next_index_number
where ID =
(
    select Card_ID
    from Main_select
)

```

```

SET NOCOUNT OFF;
END
GO

```

-- 3.14) Èçìáíèòü äñîëíèðàëëüíóþ çàùèòó èãðíêà íà n

```

CREATE PROCEDURE dbo.Change_additional_defence_range(@player_ID int, @n int)
AS
BEGIN
SET NOCOUNT ON;

```

```

UPDATE [Player]
SET additional_defence_range = additional_defence_range + @n
WHERE ID = @player_ID

```

```

SET NOCOUNT OFF;
END
GO

```

-- 3.15) Èçìáíèòü äñîëíèðàëëüíóþ äàëëüíñòü àðàêè èãðíêà íà n

```

CREATE PROCEDURE dbo.Change_additional_attack_range(@player_ID int, @n int)
AS
BEGIN
SET NOCOUNT ON;

```

```

UPDATE [Player]
SET additional_attack_range = additional_attack_range + @n
WHERE ID = @player_ID

```

```

SET NOCOUNT OFF;
END
GO

-- 3.16) Îđĩăăđêă, âñòü èè ó èăđîêă îă ñòîěă êăđòă ñ àîăîîă÷îîî îăçăăîêăî
CREATE PROCEDURE dbo.Check_player_name_card(@player_ID int, @name nvarchar(50))
AS
BEGIN
SET NOCOUNT ON;

SELECT COUNT(*)
FROM [Card]
join [Cards]
on [Card].cards_ID = [Cards].ID
WHERE (player_ID = @player_ID) and (card_location = 4) and ([Cards].[name] = @name)

SET NOCOUNT OFF;
END
GO

-- 3.17) Îěó÷èòü êăđòó èç êîîîăü è äîăăăèòü â, îă ñòîě
CREATE PROCEDURE dbo.Set_cards_to_table(@room_ID int)
AS
BEGIN
SET NOCOUNT ON;

declare @card_ID int =
(
    select [ID]
    from [Card]
    WHERE ([index_number] =
        (
            select max(index_number)
            from [Card]
            where ([room_ID] = @room_ID) and ([card_location] = 2))
        ) and
        ([room_ID] = @room_ID) and ([card_location] = 2)
)

UPDATE [Card]
SET [card_location] = 5
WHERE [ID] = @card_ID

```

```
SELECT @card_ID
```

```
SET NOCOUNT OFF;
```

```
END
```

```
GO
```

```
-- 3.18) Îăđăăă÷à êăđôû êăđîêó
```

```
CREATE PROCEDURE dbo.Passing_card_to_player(@player_ID int, @card_ID int, @card_location int)
```

```
AS
```

```
BEGIN
```

```
SET NOCOUNT ON;
```

```
declare @result int = 0
```

```
UPDATE [Card]
```

```
SET card_location = @card_location,
```

```
player_ID = @player_ID
```

```
WHERE ID = @card_ID
```

```
SELECT @result
```

```
SET NOCOUNT OFF;
```

```
END
```

```
GO
```

```
-- 3.19) Âîññòàîîăăîêă äêîêôû æèçîê âñîî êăđîêâî, âñêê ýòî âîçîîæîî
```

```
CREATE PROCEDURE dbo.Recovery_health_all_players(@room_ID int)
```

```
AS
```

```
BEGIN
```

```
SET NOCOUNT ON;
```

```
select [Player].ID
```

```
from [Player]
```

```
join [Character]
```

```
on [Player].character_ID = [Character].ID
```

```
join [Characters]
```

```
on [Character].characters_ID = [Characters].ID
```

```
where ([room_ID] = @room_ID) and ([Character].alive = 1) and ([Character].lives < [Characters].max_lives)
```

```
with Helper(ID) as
```

```
(
```



```

select [Player].ID
from [Player]
join [Character]
on [Player].character_ID = [Character].ID
join [Characters]
on [Character].characters_ID = [Characters].ID
where ([room_ID] = @room_ID) and ([Character].alive = 1) and ([Character].lives <
[Characters].max_lives)
)

```

```

UPDATE [Character]
SET [lives] = [lives] + 1
WHERE [ID] in (select * from Helper)

```

```

SET NOCOUNT OFF;
END
GO

```

```

-- 3.20) Ыёо÷èòü ââððíþþ êàððó èç êîëîâü è äîääâèòü â, â ñòääèþ âüâíðà
CREATE PROCEDURE dbo.Set_cards_to_selection_stage(@room_ID int)
AS
BEGIN
SET NOCOUNT ON;

```

```

declare @card_ID int =
(
    select [ID]
    from [Card]
    WHERE ([index_number] =
        (
            select max(index_number)
            from [Card]
            where ([room_ID] = @room_ID) and ([card_location] = 2))
        ) and
        ([room_ID] = @room_ID) and ([card_location] = 2)
)

```

```

UPDATE [Card]
SET [card_location] = 6
WHERE [ID] = @card_ID

```

```

SELECT @card_ID

```

```

SET NOCOUNT OFF;
END
GO

-- 3.21) Âîçãðàùàíêà èàðòó à êîñîäó
CREATE PROCEDURE dbo.Return_card_to_Deck(@card_ID int, @room_ID int)
AS
BEGIN
SET NOCOUNT ON;

declare @index_number int = NULL
set @index_number =
(
    select max(index_number) + 1
    from [Card]
    where ([room_ID] = @room_ID) and ([card_location] = 2)
)

if(@index_number is NULL)
    set @index_number = 1

UPDATE [Card]
SET [player_ID] = NULL,
[card_location] = 2,
[index_number] = @index_number
WHERE [Card].ID = @card_ID

SET NOCOUNT OFF;
END
GO

-- 3.22) Îãðàíàðøèâêà èàðò èç ñáðîñà è áðàâåâêà èõ à êîñîäó
CREATE PROCEDURE dbo.Shuffle_cards_in_Dropping(@room_ID int)
AS
BEGIN
SET NOCOUNT ON;

-- Îñóáðåâ êîñîäàðò à ñáðîñ
declare @max_index_number int =
(
    select max(index_number)

```

```

        from [Card]
        where (card_location = 1) and (room_ID = @room_ID)
    );

WITH Series(a, b) AS
(
    SELECT 1, cast ( rand( cast ( newid() as varbinary(16) ) ) * 1000000 + 1 as int )
    UNION ALL
    SELECT a+1, cast ( rand( cast ( newid() as varbinary(16) ) ) * 1000000 + 1 as int )
    FROM Series
    WHERE a < @max_index_number
),
Helper(a1, a2) as
(
    select [Series_1].a as a1, [Series_2].a as a2
    from
    (
        select a, row_number()over(order by [b]) npp
        from Series
    ) Series_1
    left join
    (
        select a, row_number()over(order by [b]) npp
        from Series
    ) Series_2 on Series_1.npp = Series_2.npp
)

UPDATE [Card_]
SET [Card_].index_number = [Helper_].a2,
[Card_].card_location = 2
from [Card] as [Card_]
join [Helper] as [Helper_]
on [Card_].cards_ID = [Helper_].a2

SET NOCOUNT OFF;
END
GO

-- 3.23) Ыїё÷ăřăă çăăăăřăă řřÿ [name] â ðăăăëðă [Card] äÿ çăăăřřé ID
CREATE PROCEDURE dbo.Get_card_name(@card_ID int)
AS
BEGIN

```

```
SET NOCOUNT ON;
```

```
select [Card].[name]
from [Card]
where [Card].ID = @card_ID
```

```
SET NOCOUNT OFF;
```

```
END
```

```
GO
```

```
-- 3.24) Ўқувчига ўқув картасининг ўрнига [card_location] дағдирати [Card] га қўйишга қўйиш
```

```
CREATE PROCEDURE dbo.Change_card_location(@card_ID int, @card_location int)
```

```
AS
```

```
BEGIN
```

```
SET NOCOUNT ON;
```

```
update [Card]
set card_location = @card_location
where ID = @card_ID
```

```
SET NOCOUNT OFF;
```

```
END
```

```
GO
```

```
-- 3.25) Ўқувчига ўқув картасининг ўрнига [card_location] дағдирати [Card] га қўйишга қўйиш
```

```
CREATE PROCEDURE dbo.Lose_all_cards(@player_ID int, @room_ID int)
```

```
AS
```

```
BEGIN
```

```
SET NOCOUNT ON;
```

```
select [Cards].ID
from [Card]
join [Cards]
on [Card].cards_ID = [Cards].ID
where [Card].player_ID = @player_ID
```

```
with Player_cards_ID(ID) as
```

```
(
```

```
    select [Card].ID
    from [Card]
    where [Card].player_ID = @player_ID
```

```
)
```

```

UPDATE [Card]
SET player_ID = NULL,
card_location = 1,
index_number =
(
    select max([Card].index_number) + 1
    from [Card]
    where (card_location = 1) and (room_ID = @room_ID)
)
WHERE [Card].ID in (select * from Player_cards_ID)

```

```

SET NOCOUNT OFF;
END
GO

```

-- 3.26) Âñà èàðòù èãðîèà óôîäò â ðóéó äðóâîó èãðîéó

```

CREATE PROCEDURE dbo.Send_all_cards_to_player(@player_ID_from int, @player_ID_to int)
AS
BEGIN
SET NOCOUNT ON;

```

```

select [Cards].ID
from [Card]
join [Cards]
on [Card].cards_ID = [Cards].ID
where [Card].player_ID = @player_ID_from

```

```

with Player_cards_ID(ID) as
(
    select [Card].ID
    from [Card]
    where [Card].player_ID = @player_ID_from
)

```

```

UPDATE [Card]
SET player_ID = @player_ID_to,
card_location = 3
WHERE [Card].ID in (select * from Player_cards_ID)

```

```

SET NOCOUNT OFF;
END

```

GO

-- 3.27) Îñö÷èòü ID ðñää æèâüõ èãðîêî

CREATE PROCEDURE dbo.Get_alive_roles_ID(@room_ID int)

AS

BEGIN

SET NOCOUNT ON;

with Room_players(ID) as

(

select [Player].ID

from [Player]

where [Player].room_ID = @room_ID

)

select [Roles].ID

from [Player]

join [Role]

on [Player].role_ID = [Role].ID

join [Roles]

on [Role].roles_ID = [Roles].ID

join [Character]

on [Player].character_ID = [Character].ID

where ([Player].ID in (select * from Room_players)) and ([Character].alive = 1)

SET NOCOUNT OFF;

END

GO

/*-----*/

/*-----TRIGGERS-----*/

/*-----*/

-- 5.1) Ñîçääîèà çàîèñè â òàáëèöâ [Achievements] îðè ñîçääàîèè çàîèñè â òàáëèöâ [User]

CREATE TRIGGER [dbo].[Create user achievements]

ON [dbo].[User]

AFTER INSERT

AS

BEGIN

INSERT INTO [Achievements] ([single_player_finished], [single_player_unfinished], [single_player_wins],
[single_player_defeats], [multi_player_finished], [multi_player_wins], [multi_player_defeats])

```
VALUES (0,0,0,0,0,0,0);
```

```
declare @achievements_ID [int] = IDENT_CURRENT('Achievements')
```

```
BEGIN
```

```
UPDATE [User]
```

```
SET [achievements_ID] = @achievements_ID where ID = IDENT_CURRENT('User')
```

```
END
```

```
END
```

```
GO
```

```
-- 5.2) Nîçääîfêå çàîrêñê â òàáääèöå [Deck] è [Dropping] îðê nîçääîfêå çàîrêñê â òàáääèöå [Room]
```

```
CREATE TRIGGER [dbo].[Create Dropping and Deck default record]
```

```
ON [dbo].[Room]
```

```
AFTER INSERT
```

```
AS
```

```
BEGIN
```

```
INSERT INTO [Dropping] ([count_of_cards])
```

```
VALUES (0);
```

```
INSERT INTO [Deck] ([count_of_cards])
```

```
VALUES (80);
```

```
declare @Dropping_ID [int] = IDENT_CURRENT('Dropping')
```

```
declare @Deck_ID [int] = IDENT_CURRENT('Deck')
```

```
BEGIN
```

```
UPDATE [Room]
```

```
SET [dropping_ID] = @Dropping_ID where ID = IDENT_CURRENT('Room')
```

```
UPDATE [Room]
```

```
SET [Deck_ID] = @Deck_ID where ID = IDENT_CURRENT('Room')
```

```
END
```

```
END
```

```
GO
```

```
-- 5.3) Cîçääîfêå çàîrêñê â òàáääèöå [Weapon] îðê nîçääîfêå çàîrêñê â òàáääèöå [Player]
```

```
CREATE TRIGGER [dbo].[Create weapon]
```

```
ON [dbo].[Player]
```

```
AFTER INSERT
```

```
AS
```

```

BEGIN

declare @Player_ID [int] = (select ID from inserted)

INSERT INTO [Weapon] ([name], [base_weapon], [bang_player], [firing_range], [endless_bang])
VALUES ('colt', 1, 0, 1, 0)

declare @Weapon_ID [int] = SCOPE_IDENTITY()

update [Player]
set [weapon_ID] = @Weapon_ID
where [ID] = @Player_ID

END
GO

-- 5.4) Êîèðîâîèà èäîíóð èç òàáëèöü [Cards] â òàáëèöü [Card] ïðå ñîçäàíèå êîíöîâ
CREATE TRIGGER [dbo].[Duplicate cards]
ON [dbo].[Room]
AFTER INSERT
AS
BEGIN

WITH Series(a, b) AS
(
SELECT 1, cast ( rand( cast ( newid() as varbinary(16) ) ) * 1000000 + 1 as int )
UNION ALL
SELECT a+1, cast ( rand( cast ( newid() as varbinary(16) ) ) * 1000000 + 1 as int )
FROM Series
WHERE a < 80
)
INSERT INTO [Card] ([cards_ID], [room_ID], [card_location], [index_number])
select Cards_.ID, [inserted].[ID], 2, Series_.a
from [inserted], (
select [ID], row_number()over(order by [ID]) npp
from Cards
)Cards_
left join(
select a, row_number()over(order by [b]) npp
from Series
)Series_ on Series_.npp=Cards_.npp

```


END

GO

-- 5.5) Ĩăđăĩăøèâăîēă êàðð èç ñáđîñà è äîäâäēăîēă èõ â êîēîäó

CREATE TRIGGER [dbo].[Shuffle_cards_in_Dropping_trigger]

ON [dbo].[Card]

AFTER UPDATE

AS

BEGIN

declare @count_of_cards_in_Deck int = NULL

set @count_of_cards_in_Deck =

(

select max(index_number)

from [Card]

where (card_location = 2) and (room_ID = (select room_ID from [inserted]))

)

if(@count_of_cards_in_Deck is NULL)

BEGIN

declare @max_index_number int =

(

select max(index_number)

from [Card]

where (card_location = 1) and (room_ID = (select room_ID from [inserted]))

);

WITH Series(a, b) AS

(

SELECT 1, cast (rand(cast (newid() as varbinary(16))) * 1000000 + 1 as int)

UNION ALL

SELECT a+1, cast (rand(cast (newid() as varbinary(16))) * 1000000 + 1 as int)

FROM Series

WHERE a < @max_index_number

),

Helper(a1, a2) as

(

select [Series_1].a as a1, [Series_2].a as a2

from

(

select a, row_number()over(order by [b]) npp

```

        from Series
    ) Series_1
left join
(
    select a, row_number()over(order by [b]) npp
    from Series
) Series_2 on Series_1.npp = Series_2.npp
)

UPDATE [Card_]
SET [Card_].index_number = [Helper_].a2,
[Card_].card_location = 2
from [Card] as [Card_]
join [Helper] as [Helper_]
on [Card_].cards_ID = [Helper_].a2

END

END
GO

/*-----*/
/*-----TRIGGERS END-----*/
/*-----*/

```