ЛАБОРАТОРНАЯ РАБОТА №11

По дисциплине СППР

Тема:

**ВЕБ-САЙТ С ИГРОЙ “BLUFF” (БЛЕФ)**

Выполнили студенты гр. 153505:

Косач Д.А.

Минск 2023

**ВВЕДЕНИЕ**

[**1. Правила игры “Блеф”** 3](#_Toc150187081)

[**2. Требования из лабораторной** 7](#_Toc150187082)

[**3. Ожидание и предложения** 8](#_Toc150187083)

[**3.1. Структура проекта** 8](#_Toc150187084)

[**3.2. Аутентификация** 8](#_Toc150187085)

[**3.3. Функциональные возможности** 8](#_Toc150187086)

**1. Правила игры “Блеф”**

**Игра для 2 - 6 игроков старше 12 лет, в которой участники с удовольствием жульничают, блефуют и стараются предугадывать действия соперников.**

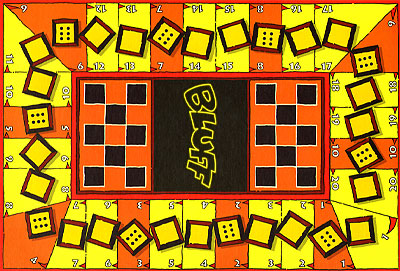
**В комплектацию входит:**

**1. игровое поле (см. рисунок 1.1).**

**2. 6 чашек для игральных кубиков.**

**3. 30 игральных кубиков (в правилах будут желтыми).**

**4. 1 кубик иного цвета (в правил будет красным).**



**Рисунок 1.1 – Игровое поле**

**Начинает игру самый молодой игрок. Он делает начальную ставку и отмечает ее красным в соответствующем месте игрового поля.**

**Ставка игрока – отражает его предположение о том, сколько игральных кубиков в игре (учитывая кубики всех участников) содержат одинаковое число. При этом ему хорошо известно, как выпали его кубики, а также количество кубиков в игре (в начале игры с 3-мя участниками в игре 15 кубиков), что позволяет ему высчитывать среднюю ставку.**

**Каждый игральный кубик имеет 6 сторон. Пять из них с числами 1, 2, 3, 4 и 5, а шестая сторона имеет изображение звезды. Звезда выполняет роль джокера и считается тем числом, на которое сделана ставка.**

**Итак, каждое число имеет два шанса из шести на то, чтобы выпасть на реальном кубике. Например, единица может выпасть в виде собственно единицы и звезды. Если привести эту пропорцию к простому виду, то каждое число может выпасть на кубики в одном из трех случаев.**

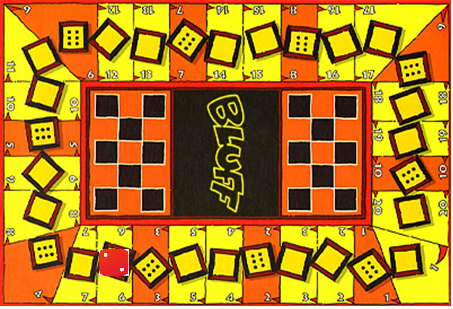
**Предположим, что в игре 15 кубиков, тогда средняя ставка: (15/3)=5 кубиков с одинаковым числом.**

**Поскольку звезда имеет шанс выпасть только в одном случае из шести, то среднее количество звезд в нашем случае равно 2-м или 3-м кубикам.**

**При этом игрок может сделать любую ставку даже на то число, которое не выпало на собственных кубиках. Все же, игра называется “Блеф”.**

**Рассмотрим пример.**

**Игрок делает начальную ставку – “шесть двоек”. После чего игрок берет и поворачивает красный кубик так, чтобы сверху отобразилось число “2”, на которое он сделал ставку, и размещает его в клетке игрового поля с цифрой “6”, которое означает количество кубиков в ставке (см. рисунок 2). Стрелочки указывают направление, в котором игроки могут изменять ставку. Число на кубике соответствует числу, на которое сделана ставка. Цифра на клетке игрового поля соответствует количеству кубиков, на которое сделана ставка.**

****

**Рисунок 2 – ход игрока**

**Далее ход в порядке часовой стрелки переходит к следующему игроку, который может повысить ставку или оспорить ее.**

**Изменение ставки может происходить следующим образом:**

1. **Назвать более крупное число и:**

* **Оставить при этом кубики на той же клетке.**
* **Передвинуть кубик на другую клетку с более крупным числом.**

1. **Назвать любое другое число ставки или оставить его без изменений:**

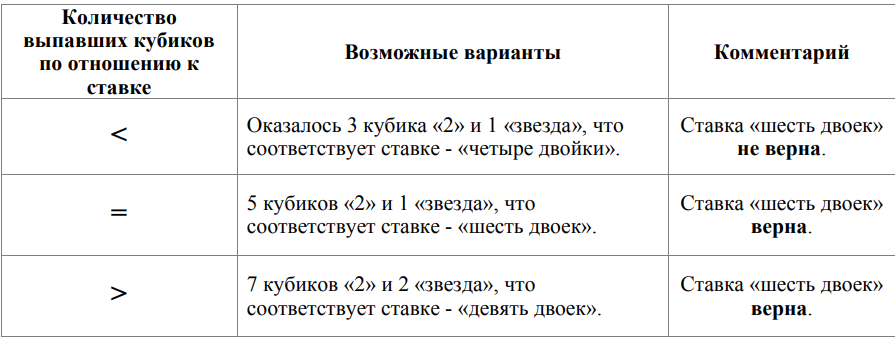
* **Передвинуть кубик на другую клетку с более крупным числом.**

**В данном случае повышение ставки может принять следующие значения:**

**Шесть троек, шесть пятерок, семь двоек, восемь единичек, восемь троек, четыре звезды…**

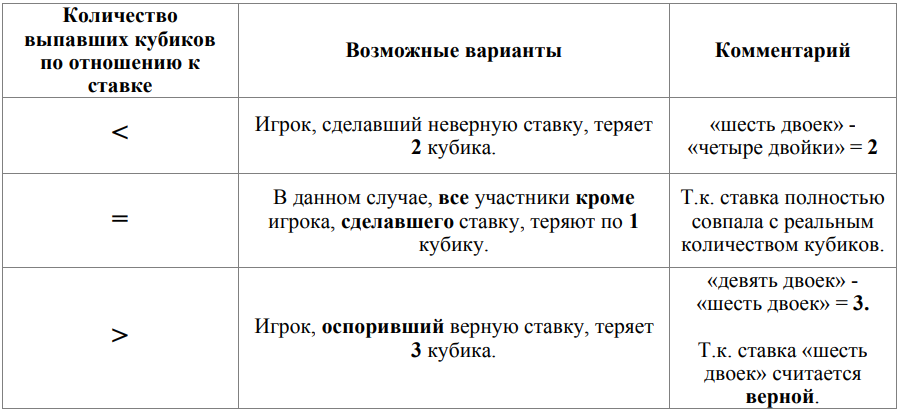
**После того как игрок повысил ставку, ход переходит к следующим игрокам до тех пор, пока один из них не решит оспорить ставку.**

**Чтобы разрешить спор игроки поднимают свои чашки и подсчитывают количество кубиков, соответствующих ставке, + количество кубиков с изображением звезды. Предположим, что ставка для данного раунда – “шесть двоек” и рассмотрим три возможных варианта развития событий:**

****

**Игрок, проигравший в споре, теряет один или несколько своих кубиков. Если игрок, оспаривавший ставку, оказался прав, то игрок, делавший ставку, теряет количество кубиков, и наоборот. Количество кубиков, которых игрок лишается, определяется как разница между количеством кубиков в ставке и реальным количеством кубиков, выпавших в игре.**

**В рассматриваемом примере возможны следующие варианты:**

****

**Игроки должны положить проигранные кубики в центр игрового поля, чтобы все участники могли знать количество кубиков, находящихся в игре. После того, как спор был завершен, начинается новый раунд. Все участники снова трясут свои кубики в чашах, затем переворачивают их на стол и игрок, победивший в споре в предыдущем раунде, делает начальную ставку.**

**Игра заканчивается, когда только у одного игрока остаются игральные кубики. Этот игрок объявляется победителем.**

[Видео с партией](https://www.youtube.com/watch?v=I9d_8mbVsyY)**.**

**Сайт с изначальными** [правилами](https://tesera.ru/images/items/119710/%D0%91%D0%BB%D0%B5%D1%84.pdf)**.**

# **2. Требования из лабораторной**

Требуется разработать игру для не менее 2 игроков. Задание выполняется всей командой. Рекомендуется распределить части разработки проекта между участниками команды (разработка клиентской логики, разработка серверной логики, разработка интерфейса, разработка бизнес-логики и т.д.)

Бизнес-логика игры (очередность хода, подсчет результатов, формирование данных игрового поля и т.д.) реализуется на серверной части.

Бизнес-логику описать в отдельном классе.

Взаимодействие между клиентом и сервером реализовать по технологии signalR.

Использовать строго типизированный Hub.

Предусмотреть возможность проведения нескольких игр одновременно (использовать свойство Group).

Клиентскую часть выполнить на основе технологии Blazor Webassembly.

Бизнес-логику клиента описать в отдельном классе.

Отрисовку игрового поля реализовать в отдельном компоненте Blazor.

Возможно, полезные ссылочки (я так, немного глянул, что вообще есть):

[Долгое видео, как создают Дурака](https://www.youtube.com/watch?v=13RtxapWZoY)

[Покер, который у меня не заработал)](https://github.com/JonasSiuksteris/Poker/tree/master)

[FlipGame (тут лучше зайти и увидеть, что там)](https://github.com/KristofferStrube/FlipGame)

# **3. Ожидание и предложения**

**В данном разделе будут описаны ожидания от данного проекта и предложения по тому, что нужно для реализации, как это реализовывать, где посмотреть информацию.**

## **3.1. Структура проекта**

**Будет 2 проекта: сервер, на котором будут реализованы технология SignalR и аутентификация, и клиентская часть, которая будет реализовывать пользовательский интерфейс при помощи фреймворка Blazor Webassembly и подключение к серверу.**

**Сервер должен реализовывать логику игры, очередность ходов, несколько игр одновременно (т.е. настроить группы SignalR).**

## **3.2. Аутентификация**

**Не надо!**

**Ожидается возможность войти в аккаунт, создать новый аккаунт, выйти с аккаунта. Пользователь должен хранить в себе следующую информацию: логин, пароль, дата рождения (можно еще аватар, если не лень, и почту по приколу, мб восстановление пароля сделать). Скорее всего, сделаем через jwt-токен, будем с нуля свою создавать, чтоб не было много лишних полей у юзера (это надо про аутентификацию почитать дополнительно).**

## **3.3. Функциональные возможности**

**Было бы прям круто дизайн разрабатывать и для телефонов, сделать список с открытыми играми, где есть еще места для пользователей. Ожидается страница с правилами, страница для подключения к игре, страница логина и регистрации (как по мне, можно прикольно сделать это на одной странице).**

**Наверное, самый простой вариант по реализации интерактива:**

**Делаем только для пк. Не позволим уменьшать окно приложения, чтоб не запариваться по поводу внешнего вида. Пример игрового поля уже был в правилах. Делаем каждый блок поля – отдельный типа div, на который можно нажать. Допустим, игрок ходит. Вначале у него появятся все стороны кубика, на которые он может поменять. Пускай обязательно выбирает. Затем выбирает ячейку, на которую может поставить (запрещенные грани кубика и поля можно сделать тусклыми). Там на случай, когда можно не двигать, добавим кнопку. У пользователей будут не чашки, а крышки, которые будут закрывать выставленные в ряд кубики. У игрока его крышка будет прозрачной, чтоб он видел свои кубики. Над крышкой 2 кнопки – оспорить и повысить ставку.**

**Наверное, 2 страницы: главная с кратким описанием игры и доступными играми и полное руководство по игре.**

**На главной сделать кнопку по созданию лобби. После создания лобби сразу предлагаем ввести имя и перекидываем на страницу с игрой. Сделать ограничение на количество лобби, чтоб дофига не создали. Удалять пустые лобби.**

**Что сделать с подключенными после начала игры игроками?**

**Как понять, что игру можно начать? (Мб для каждого сделать флажок, что он готов. После того, как все в лобби готовы, остальных не учитывать при определении следующего).**

**Вопросики:**

**Можно ли, как в js напрямую взаимодействовать с элементами html? Например, поменять стили, добавить элемент и т.к. Или надо вызывать функции js? (вот** [статья](https://learn.microsoft.com/ru-ru/aspnet/core/blazor/javascript-interoperability/?view=aspnetcore-7.0)**)**

**Только через привязки**

**Делать ли логин и регистрацию страницами на сервере, или делать страницы в клиентском проекте, который будет отправлять запросы к серверу?**

**Не надо авторизацию**

# **4. Логика игры на стороне сервера**

**Объект игры, который будет передаваться (возможно), объект доски и кубиков (для отрисовки, наверное). В игре хранится нынешняя ставка, в которой выбранный кубик и ячейка на площадке. Также надо иметь активные кубики всех игроков, когда проверяем ставку (активные для того, чтоб определять количество кубиков в игре).**

**Объект кубика может хранить id и путь к изображению.**

**Для хранения кубиков игроков можно использовать словарь: имя пользователя – ключ, количество (или значения, т.к. для проверки победителя нужны значения и отправлять в игру должны значения кубиков, которые будет видеть игрок) кубиков – значение.**

**При подключении к игре надо будет, наверное, сделать коллекцию с пользователями, где будет имя и ConnectionId.**

**Для работы с клиентами можно использовать (все это лежит Context класса Hub):**

**ConnectionId - возвращает уникальный идентификатор подключения в виде строки.**

**ConnectionAbborted - возвращает объект CancellationToken, который извещает о закрытии подключения. Может пригодиться в случае, если игрок отключился и его не надо учитывать при выборе игрока, который делает ход.**

**Abort() - принудительно завершает текущее подключение. Может пригодиться при завершении игры и если заходит лишний клиент.**

**Если игрок отключается, то проверять, последний ли он. Если последний, то удалять этот сервер у всех. Т.е. для этого надо в меню с серверами всех подписывать на метод в хабе, который будет выдавать информацию о группах (т.е. серверах).**

**1. Начало**

**Ожидаем всех игроков, ждем 30 секунд для подтверждения.**

**Первый ход делает случайный игрок.**

**2. Сама игра**

**Рандомятся кубики. Игрок свои кубики видит снизу вместе с кнопками. Остальные игроки по часовой располагаются и ходят. Надо сделать таймер на ход каждого игрока, который будет отображаться всем.**

**Первый ход точно делают ставку. Дальше есть выбор: повысить и оспорить.**

**Ставка:**

**1) выбираем грань кубика**

**2) выбираем поле, на которое ставить кубик**

**3) переход хода к следующему игроку**

**Оспорить:**

**1) отобразить всем кубики каждого**

**2) сделать подсчет**

**3) убрать кубики у игрока (если кубики закончились, то убираем игрока из очереди на ход. Если остался 1 игрок, то он победил).**

**4) начинаем этап партии заново**

**3. Конец**

**Выводим победителя с большой кнопкой “ВЫЙТИ”.**

**Что нужно игре?**

**Туда-сюда:**

**Сервер:**

**Клиент, который ходит, со ставкой. Если без ставки, то это оспаривание. Сервер хранит последний ход, чтоб при оспаривании знать значение, на котором остановилась ставка.**

**Клиент:**

**Сервер после ставки должен вернуть сущность ставки, чтоб отрисовать у всех новую ставку.**

**При оспаривании возвращает список клиентов. Со стороны клиента будет находиться приватная переменная с уникальным id, чтоб находить нужного клиента из списка.**

**Сущности:**

**Клиент (уникальный id, имя, кубики, количество кубиков)**

**Кубики являются массивом, где индекс + 1 – это значения кубиков. Соответственно, элемент массива хранит количество кубиков с данным значением. При проверке ставки мы будем получать из массива всех игроков по индексу с кубика для ставки – 1 и 5 (это звездочка).**

**Ставка (значение кубика и поля)**