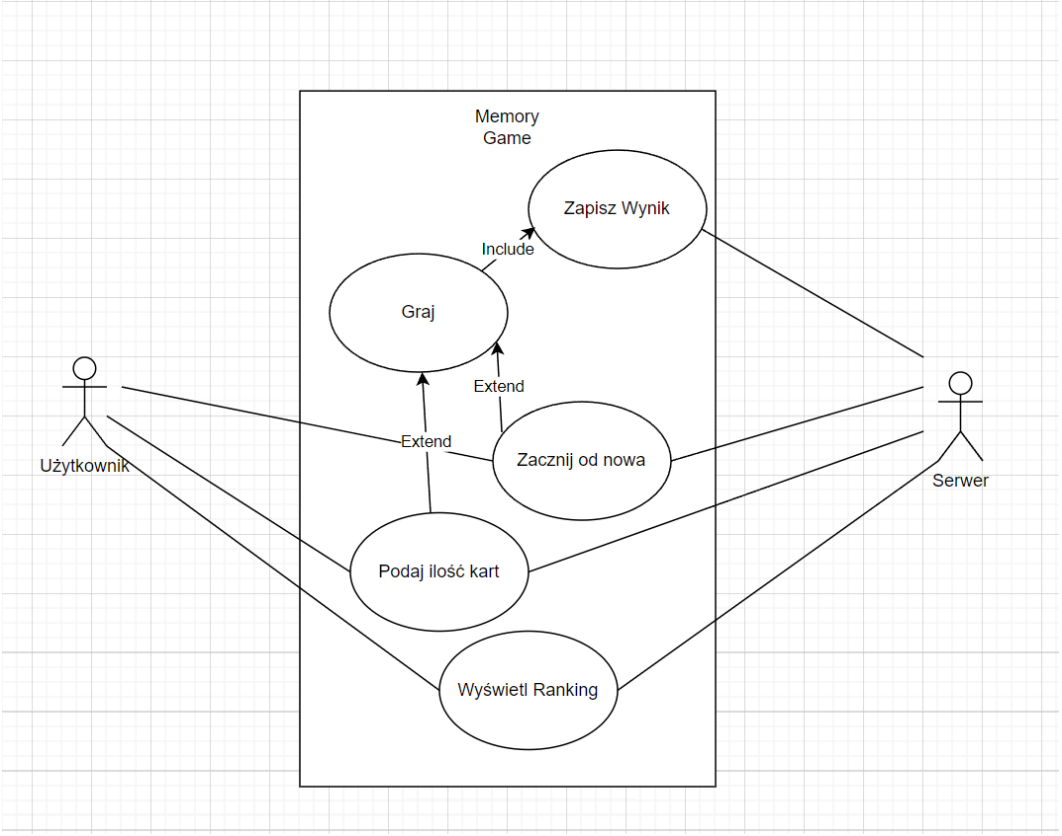
Technologie internetowe w programowaniu
Projekt zaliczeniowy: Memory Game

Wykonał: Artur Kompała ISI 1

1. **Krótki opis aplikacji.**
   Gra ,,Memory” to prosta gra polegająca na zapamiętywaniu pozycji kart rozłożonych na planszy. Gracz za zadanie ma znalezienie wszystkich par kart o takich samych rysunkach. Jednocześnie tylko 2 karty mogą być odkryte. Przed rozpoczęciem gry wybierana jest ilość kart do odkrycia, minimalnie to 2 pary czyli 4 karty,maksymalnie 18 par czyli 36 kart. Po znalezieniu wszystkich kart gra się kończy i zapisywany jest wynik czyli ilość potrzebnych ruchów do odkrycia wszystkich kart. Wynik czym niższy tym lepszy.

2. **Lista funkcjonalności aplikacji.**
   - losowe generowanie planszy składającej się z macierzy kafelek, pod którymi znajdują się obrazki wybranych zwierząt,
   - odkrywanie wyłącznie dwóch kafelek naraz,
   - płynna animacja obrotu przy odkrywaniu kafelki,
   - informacja o liczbie wykonanych ruchów (odkryciu par kafelek),
   - możliwość ustawienia wymiarów planszy,
   - możliwość zapisu wyników danego gracza do pliku tekstowego z wykorzystaniem technologii PHP
   - możliwość wyświetlenia listy graczy z największą punktacją,
   - dostosowanie do wielkości obszaru roboczego – aplikacja powinna reagować na zmniejszenie bądź zwiększenie rozmiaru okna, zmieniając rozmiar lub położenie wyświetlanych komponentów (zachowanie responsywności).
3. **Diagram przypadków użycia.**



4. **Opis najważniejszych zastosowanych algorytmów**

Algorytm mieszania kart:

```
23   const shuffle = (array) => {
24       let currentIndex = array.length, randomIndex;
25       while (currentIndex != 0) {
26
27           randomIndex = Math.floor(Math.random() * currentIndex);
28           currentIndex--;
29           [array[currentIndex], array[randomIndex]] = [
30               array[randomIndex], array[currentIndex]];
31       }
32       return array;
33   }
```

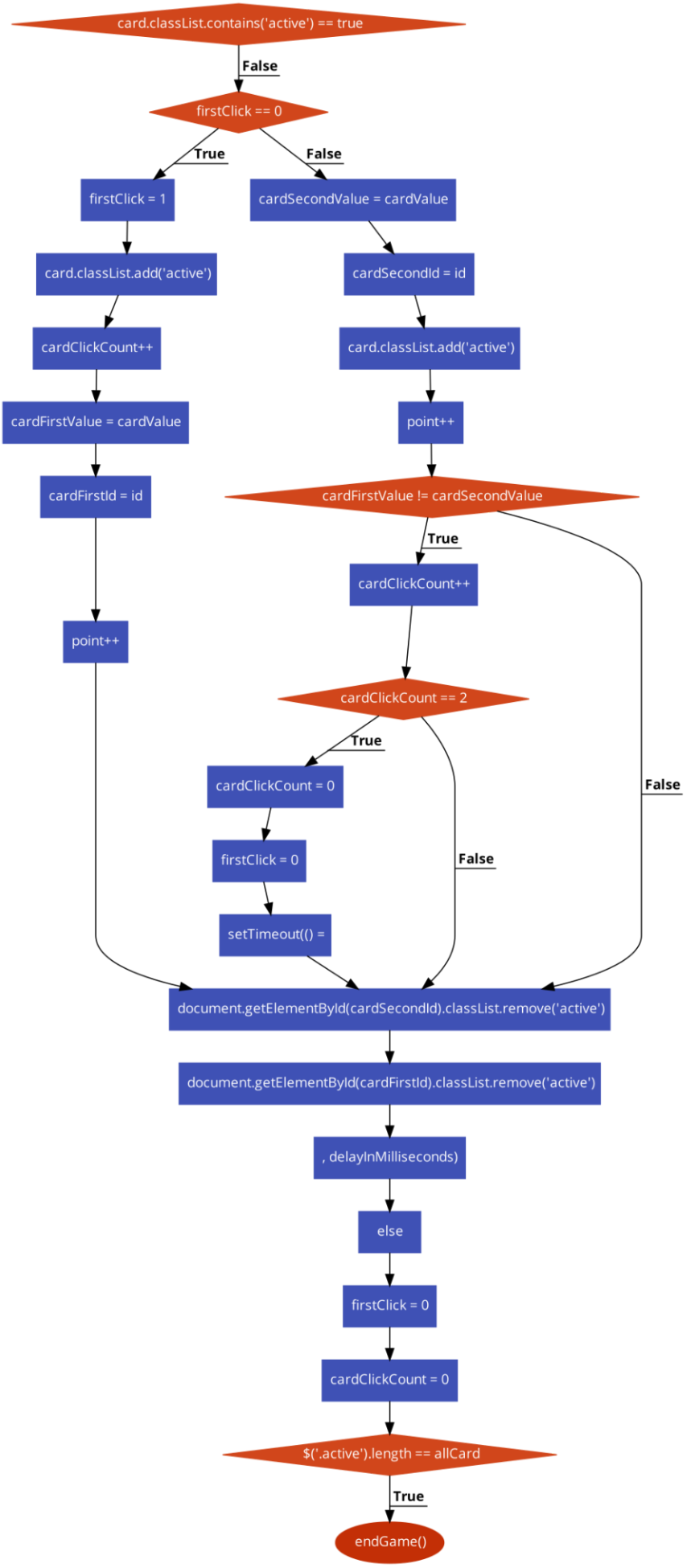
Do swojego programu wykorzystałem algorytm tasowania ,,Fisher - Yates shuffle”
Opis algorytmu:

1. Zapisz liczby od 1 do *N.*
2. Wybierz losową liczbę *k* z przedziału od jeden do liczby pozostałych niewylosowanych liczb (włącznie).
3. Licząc od końca, wykreśl *k* - tą liczbę, która jeszcze nie została wykreślona, i zapisz ją na końcu oddzielnej listy.
4. Powtarzaj od kroku 2, aż wszystkie cyfry zostaną wykreślone.
5. Sekwencja liczb zapisanych w kroku 3 jest teraz losową permutacją oryginalnych liczb

Algorytm sprawdzania kart:

```
46  const checkCard = (id, cardValue) => {
47      let delayInMilliseconds = 500;
48      let card = document.getElementById(id);
49
50      if (card.classList.contains('active') == true) {
51          return;
52      }
53      if (firstClick == 0) {
54          firstClick = 1;
55          card.classList.add('active');
56          cardClickCount++;
57          cardFirstValue = cardValue;
58          cardFirstId = id;
59          point++;
60      } else {
61          cardSecondValue = cardValue;
62          cardSecondId = id;
63          card.classList.add('active');
64          point++;
65          if (cardFirstValue != cardSecondValue) {
66              cardClickCount++;
67
68              if (cardClickCount == 2) {
69                  cardClickCount = 0;
70                  firstClick = 0;
71                  setTimeout(() => {
72                      document.getElementById(cardSecondId).classList.remove('active');
73                      document.getElementById(cardFirstId).classList.remove('active');
74                  }, delayInMilliseconds);
75
76              }
77
78          } else {
79              firstClick = 0;
80              cardClickCount = 0;
81
82          }
83      }
84      if($('.active').length == allCard){
85          endGame();
86      }
87
88      points.textContent = `Liczba ruchów: ${point}`;
89
90  }
```

Schemat blokowy:


card.classList.contains('active') == true
False
firstClick == 0
True
False
firstClick = 1
cardSecondValue = cardValue
card.classList.add('active')
cardSecondId = id
cardClickCount++
card.classList.add('active')
cardFirstValue = cardValue
point++
cardFirstId = id
cardFirstValue != cardSecondValue
True
cardClickCount++
point++
cardClickCount == 2
True
cardClickCount = 0
False
firstClick = 0
False
setTimeout(() =
document.getElementById(cardSecondId).classList.remove('active')
document.getElementById(cardFirstId).classList.remove('active')
, delayInMilliseconds)
else
firstClick = 0
cardClickCount = 0
$('.active').length == allCard
True
endGame()