



---

# C.V.E.B MANUALE SVILUPPATORE

Versione 1.0.1

---

## INTRODUZIONE

Questo manuale<sup>1</sup> è stato realizzato con lo scopo di provvedere alle informazioni essenziali per comprendere in pieno il corretto funzionamento dell'algoritmo del software e gli strumenti utilizzati per la realizzazione e l'implementazione di quest'ultimo.

La conoscenza dei seguenti linguaggi di programmazione è richiesta per poter sviluppare il software:

- HTML5 (<https://www.html.it/guide/guida-html5/>)
- CSS3 (<https://www.html.it/guide/guida-css-di-base/>)
- JavaScript (<https://www.html.it/guide/guida-javascript-di-base/>)

Inoltre è consigliata la conoscenza dei concetti base del web storage di Javascript e con precisione la proprietà `localStorage`.

---

<sup>1</sup> Se hai qualche suggerimento per migliorare questo manuale, sentiti libero di segnalare problemi. Se riscontri un errore di ortografia o di battitura, un bug o incoerenza non esitare a riportarcelo.

# 1.Setup

## Requisiti minimi di sistema

- Sistema operativo a 64-bit: Windows, Linux o MacOS.
- 1GB RAM.

## Software di sviluppo

I seguenti IDE facilitano la visione e lo sviluppo del codice di gioco.

- **Visual Studio Code** - un editor di codice sorgente sviluppato da Microsoft per Windows, Linux e macOS, include il supporto per debugging e un controllo per Git integrato, dall'uso intuitivo, molto personalizzabile e facile da usare, per scrivere codice ed eseguire comandi direttamente dall'editor di testo. Per scoprire maggiori informazioni visitare <https://code.visualstudio.com/docs/>
- **Replit** - è un semplice ma potente IDE, Editor, Compiler, e REPL online. Strumento molto utile per visualizzare codice anche su dispositivi mobili o fissi se sprovvisti di IDE. Maggiori informazioni su <https://docs.replit.com/>

## Web browser

L'interfaccia web del gioco supporta tutti i browser più diffusi, tra cui Google Chrome, Mozilla Firefox, Safari, Opera 15+, Internet Explorer 11 e Microsoft Edge.

## Librerie

Per la realizzazione del software sono state implementate le seguenti librerie:

- **jQuery** - una libreria javascript che permette ai Designer/Developer di sviluppare pagine web con funzionalità interattive in maniera semplice. API all'indirizzo <https://api.jquery.com/category/version/1.6/>
- **FontAwesome** - un toolkit di icone distribuito sotto licenze libere. Manuale utente su <https://fontawesome.com/how-to-use/on-the-web/referencing-icons/basic-use/>

## 2. Utente

### 2.1 Interfaccia utente

C.V.E.B è un gioco replica del famoso programma televisivo *Chi vuole essere milionario?* e ne ricalca sia l'impostazione grafica che il meccanismo di gioco. In C.V.E.B l'utente, dopo aver iniziato una partita dalla schermata principale<sup>2</sup>, sceglie di concorrere per accaparrarsi il premio di 512 bitcoin. Durante la sessione di gioco il software mostrerà all'utente una domanda generata casualmente e le sue relative 4 risposte, inoltre verrà mostrato il montepremi per il momento accumulato ed i tre aiuti se non ancora utilizzati.



### Layout base

C.V.E.B ha un layout semplice, intuitivo ed è diviso in cinque aree:

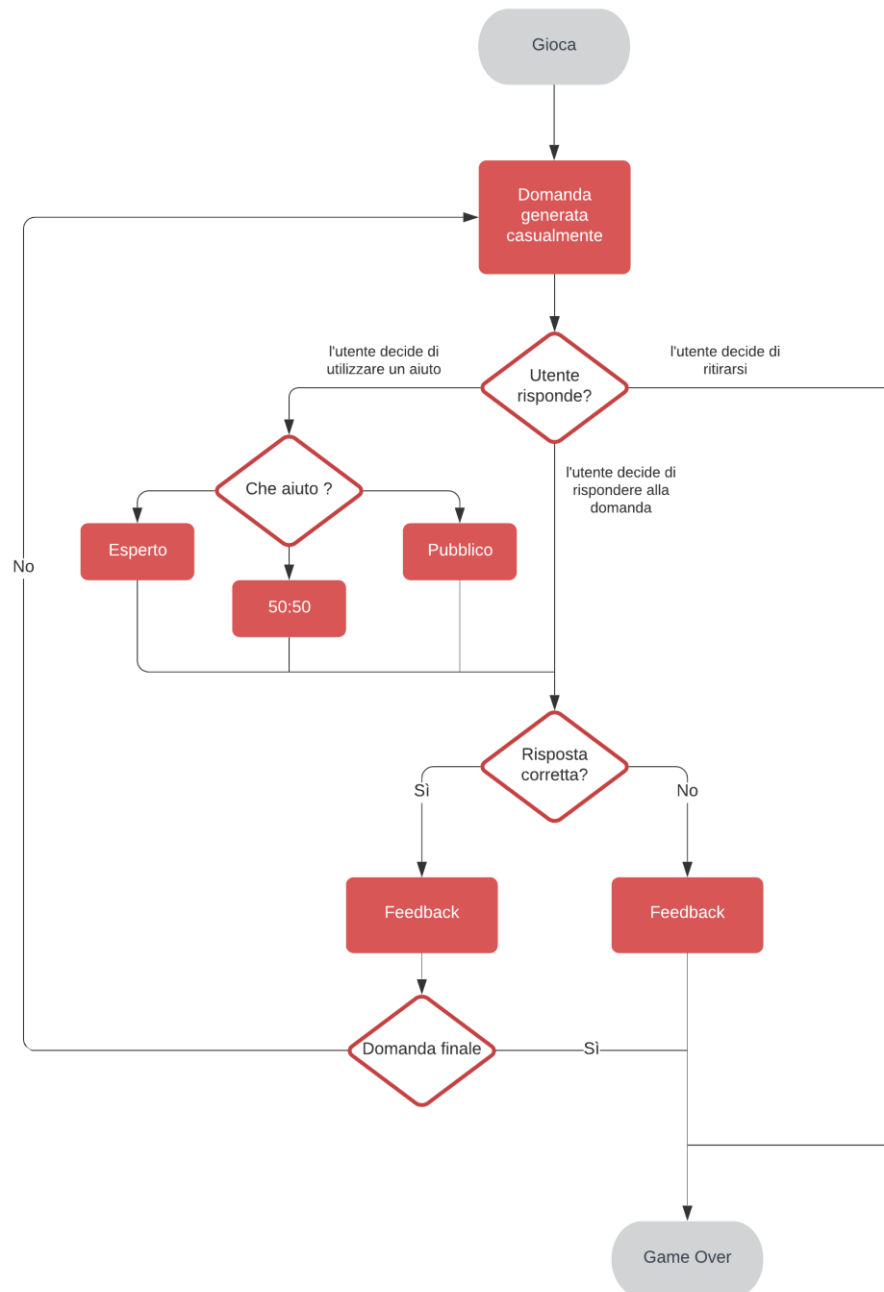
- **Domanda** – viene mostrata una domanda generata casualmente rispettando il grado di difficoltà congruo al relativo step del montepremi.
- **Risposte** – sono presenti quattro risposte per ogni domanda generata, tra cui tre errate e soltanto una corretta.
- **Aiuti** – sono le tre possibilità di aiuto che l'utente può ricevere dal software una volta sola per ciascun aiuto in ogni sessione di gioco.
- **Montepremi** – a destra dell'interfaccia è mostrata la classifica con evidenziati i premi per ogni domanda a cui l'utente abbia risposto correttamente.
- **Exit o Prossima domanda** – le due opzioni disponibili all'utente sul proseguimento del gioco.

---

<sup>2</sup> L'interfaccia grafica della schermata principale è riportata nel manuale utente

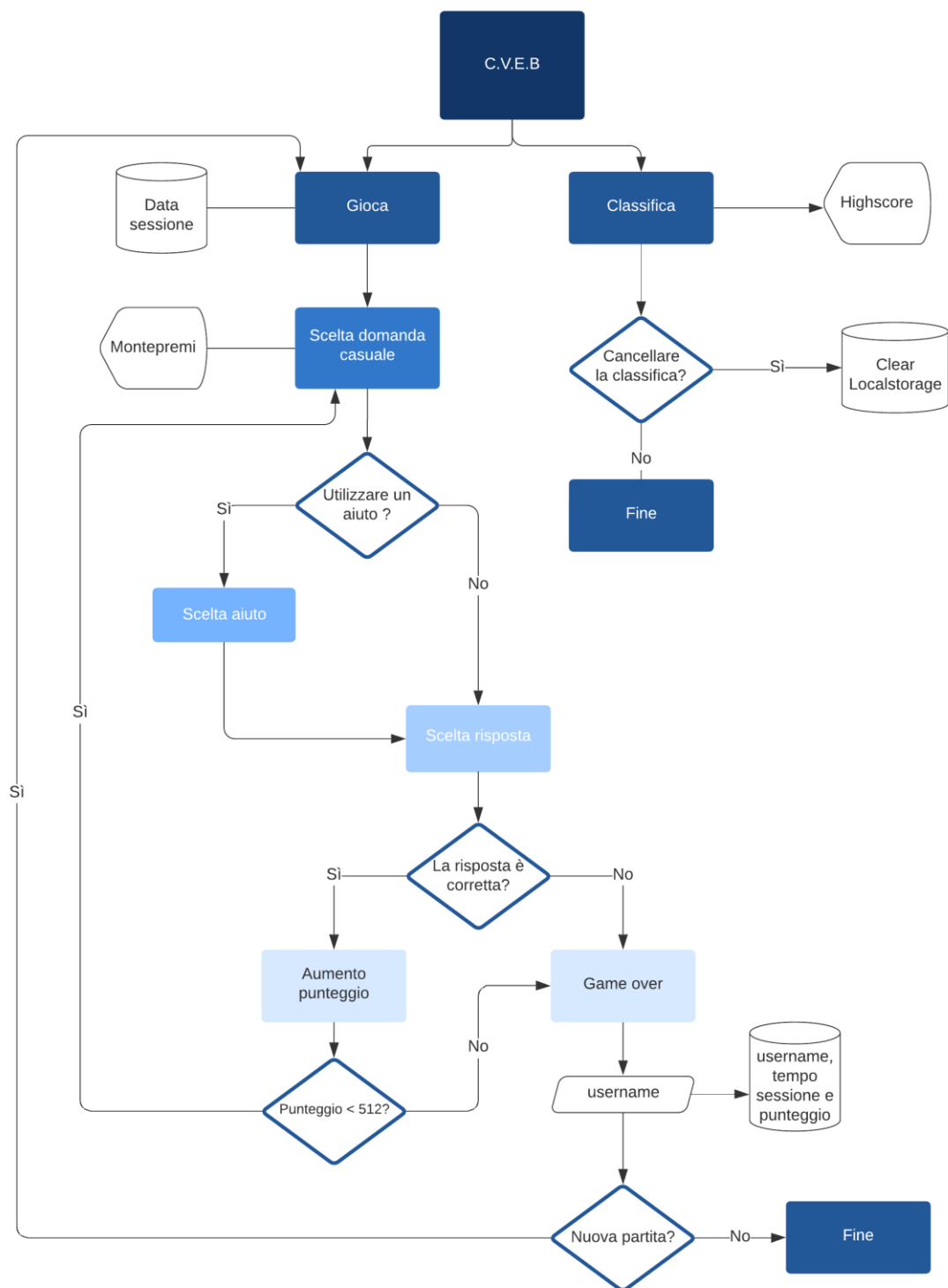
## 2.2 Azioni dell'utente

Il seguente diagramma di flusso riporta le modalità in cui l'utente può interagire con l'interfaccia se quest'ultimo ha avviato una sessione di gioco.



## 3.Architettura e API

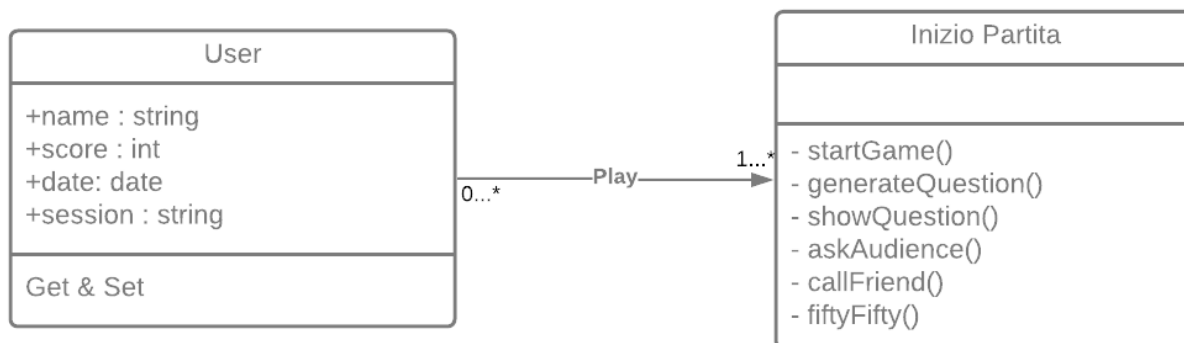
Questa sezione è dedicata all'organizzazione base del software espressa dalle sue componenti, dalle relazioni tra di loro e con l'ambiente



### 3.1 Inizio della partita

L'utente scegliendo di iniziare una sessione di gioco invia una richiesta al server che elabora la risposta tramite il linguaggio Javascript attraverso le seguenti funzioni:

- startGame()
- generateQuestion()
- showQuestion()



Elaborata la risposta, l'utente potrà visualizzare tramite il client l'interfaccia grafica tramite la quale potrà iniziare la propria sessione di gioco. Nello specifico verrà estrapolata una domanda dall'array:

```
const questions = [
  { dif:1,
    question: "Quanti sono i giocatori titolari di una squadra rugby?",
    answers: [
      { text: "A. 15", correct: true },
      { text: "B. 11", correct: false },
      { text: "C. 12", correct: false },
      { text: "D. 8", correct: false },
    ],
  },
  . . .
]
```

L'array presenta 50 domande nelle rispettive 50 celle, le domande sono ordinate in ordine di difficoltà secondo lo schema:

```
const questions = [  
  { dif:1,  
    question: ...,  
    answers: [...],  
  }, { dif:2,  
    ... },  
  { dif:3,  
    ... },  
  { dif:4,  
    ... },  
  { dif:5,  
    ... },  
  { dif:6,  
    ... },  
  { dif:7,  
    ... },  
  { dif:8,  
    ... },  
  { dif:9,  
    ... },  
  { dif:10,  
    ... },  
  { dif:1,  
    ... },  
  ...  
]
```



La funzione:

```
(function ($) {  
    $.rand = function (arg) {  
        if ($.isArray(arg)) {  
            return arg[$.rand(arg.length)];  
        } else if (typeof arg == "number") {  
            return Math.floor(Math.random() * arg);  
        } else {  
            return 4;  
        }  
    };  
})(jQuery);
```

Grazie alla libreria jQuery (<https://blog.jquery.com/2011/05/03/jquery-16-released/>), permette di ottenere un numero casuale da una serie di numeri specifici contenuti all'interno di un array che in questo caso corrisponde a:

```
var items = [0, 10, 20, 30, 40];
```

Ovvero, un array contenente l'indice delle domande di difficoltà 1. Grazie alla quale la funzione:

```
function generateQuestion() {  
  
    nextButton.classList.remove("hide");  
    exitButton.classList.remove("hide");  
    currentQuestionIndex = $.rand(items);  
    currentWinningIndex = 9;  
  
    showQuestion(questions[currentQuestionIndex]);  
}
```

Può dare inizio alla generazione delle domande casuali in ordine di difficoltà.

Per ogni risposta corretta, il software genera la successiva domanda seguendo l'ordine delle celle dell'array `questions`:

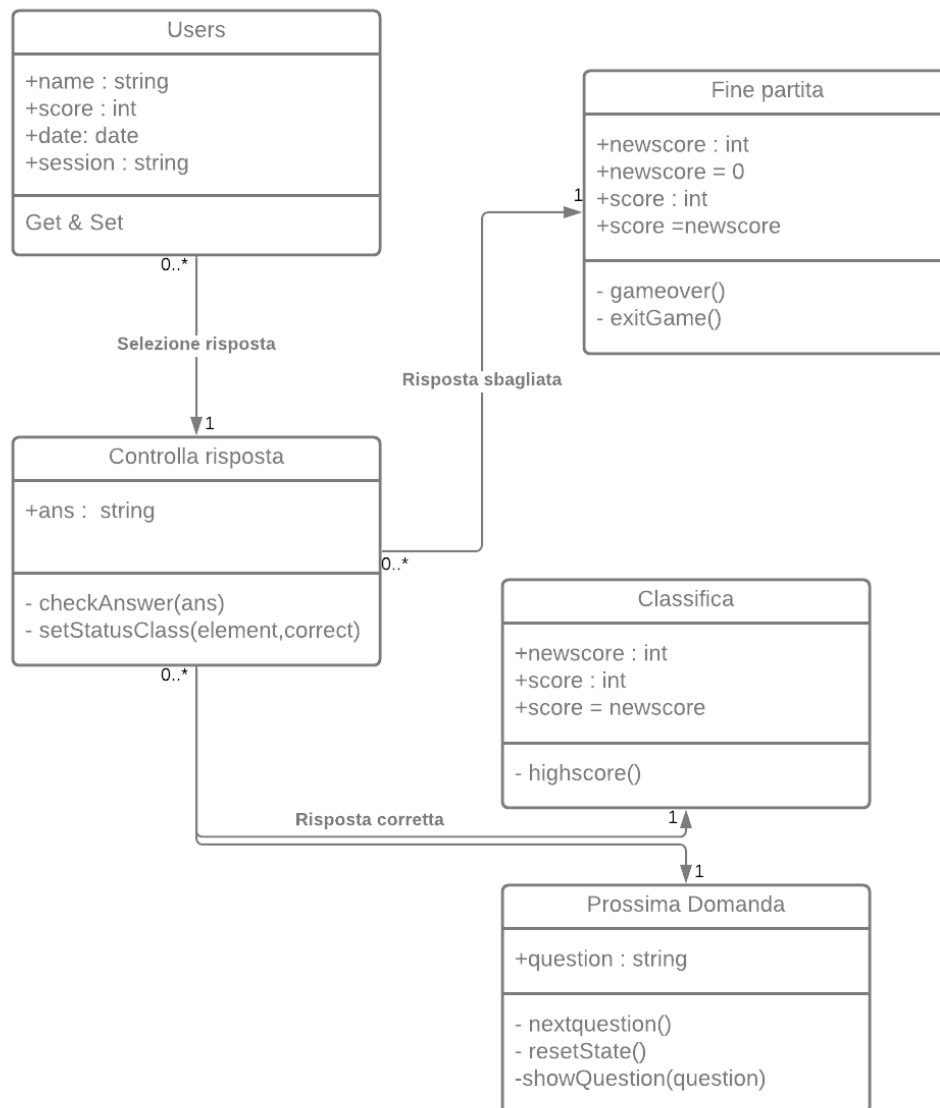
```
function nextquestion() {  
  if (nxtqst == true) {  
    currentQuestionIndex++;  
    currentWinningIndex--;  
    nxtqst = false;  
    resetState();  
    showQuestion(questions[currentQuestionIndex]);  
  }  
}
```

Le domande selezionate sono poi riportate nell'interfaccia utente attraverso la funzione:

```
function showQuestion(question) {  
  questionElement.innerHTML = question.question;  
  question.answers.forEach((answer) => {  
    const newButton = document.createElement(  
      "button"  
    ); /*create a new button elements for the answers*/  
    newButton.classList.add("btn");  
    newButton.innerHTML = answer.text;  
    if (answer.correct) {  
      newButton.dataset.correct =  
        answer.correct; /*check if the answer is right or wrong */  
    }  
    answerButtonsElement.appendChild(newButton);  
    newButton.addEventListener("click", checkAnswer);  
  });  
}
```

### 3.2 Sviluppo della partita

Avviata la sessione di gioco l'utente può compiere varie azione come descritto nel punto 2.2 *Azioni dell'utente*. In caso di una o più azioni da parte dell'utente il software restituisce uno o più feedback nel caso non riscontrasse errori.



Selezionata la risposta, il software elabora il feedback da restituire all'utente.

```
function checkAnswer(ans) {
  const selectedButton = ans.target;
  const answer =
    selectedButton.dataset
      .correct; /*Chechinkg if the answer is right or wrong using data
attributes*/
  const totalWinnings = document.createElement(
    "div"
  ); /*create a new elements to represent total winnings for the answers*/

  if (answer && score < 11) {
    /*Chech if the answer is correct or wrong and if there are still
questions unanswered*/
    var audio = new Audio("/sounds/sound_right.mp3");
    audio.play();
    nxtqst = true;
    selectedButton.classList.add("correct");
    winnings.children[currentWinningIndex].classList.add("correctly");
    score++;
  } else if (!answer) {
    var audio = new Audio("/sounds/sound_wrong.mp3");
    audio.play();
    selectedButton.classList.add("wrong");
    winnings.children[currentWinningIndex].classList.add("wrongly");

    Array.from(answerButtonsElement.children).forEach((button) => {
      setStatusClass(button, button.dataset.correct);
    });
  }
}
```

Nel caso in cui la risposta selezionata sia corretta, l'utente sarà avvisato da un segnale acustico, dall'aggiornamento del montepremi e dal cambio del layout della casella della risposta esatta attraverso:

```
function setStatusClass(element, correct) {
  if (correct) {
    element.classList.add("correct");
  }
}
```

Un' eventuale risposta sbagliata porta alla conclusione della sessione di gioco. Verrà subito dopo richiesto l'username dell'utente il quale verrà salvato insieme agli altri dati della sessione all'interno della specifica classifica. Una risposta sbagliata azzerà il montepremi guadagnato facendo automaticamente realizzare 0 punti:

```
function exitGame() {  
  const totalWinnings = document.createElement("div");  
  if (score === 0) {  
    winningMessage = document.createTextNode(  
      `Mi dispiace vai a casa con 0 btc. Il nulla!! Ritenta la prossima  
volta`  
    );  
    totalWinnings.appendChild(winningMessage);  
    endElement.insertBefore(totalWinnings, child);  
    setTimeout(gameOver, 1000);  
  }  
  ...  
}
```

## Funzionamento degli aiuti

In caso l'utente voglia ricevere un aiuto, da parte del software, quest'ultimo mette a disposizione dell'utente tre aiuti possibili: il 50/50, l'aiuto del pubblico e la chiamata all'esperto.

Per fornire all'utente l'aiuto desiderato, il software lavora con le quattro possibili risposte in base al loro contenuto booleano che le identifica come corrette od errate:

```
...  
answers: [  
  { text: "A. Nessuno", correct: false },  
  { text: "B. Pacifico", correct: false },  
  { text: "C. Indiano", correct: false },  
  { text: "D. Atlantico", correct: true },  
],  
...
```

L'aiuto 50/50 elimina in maniera casuale due delle tre risposte errate all'interno dell'array **answer**.  
L'aiuto potrà essere utilizzato solo una volta e dopo la sua esecuzione verrà nascosto.

```
function fiftyFifty() {
  var audio = new Audio("/sounds/50-50.mp3");
  audio.play();
  let rand = Math.floor(Math.random() * 3);
  let newArr = []; //Create a new array to push 2 wrongs options into
  audienceVoters.classList.remove("show");
  alert.classList.remove("show");
  for (let i = 0; i < 2; i++) {
    for (let j = 0; j < 4; j++) {
      if (
        !answerButtonsElement.children[j].dataset.correct &&
        newArr.length < 2
      ) {
        newArr.push(answerButtonsElement.children[j]);
      }
    }
  }
  for (let i = 0; i < newArr.length; i++) {
    newArr[i].classList.add("nothing");
  }
  fiftyButton.classList.add("hide");
}
```

L'aiuto *Chiamata all'esperto* consiglierà all'utente la risposta corretta all'interno di un box di testo.  
I possibili consigli dell'esperto sono scelti casualmente da un array di possibili consigli. L'aiuto potrà essere utilizzato solo una volta e dopo la sua esecuzione verrà nascosto.

```
function callFriend() {
  var audio = new Audio("/sounds/help.mp3");
  audio.play();
  audienceVoters.classList.remove("show");
  alert.classList.add("show");
  for (let i = 0; i < 4; i++) {
    if (answerButtonsElement.children[i].dataset.correct) {
      answer = answerButtonsElement.children[i].innerText;
      //create an array of possible answer and generate a random one
      friendsAnswer = [
        `Ma hai letto bene la domanda? \n Veramente mi chiami per una domanda
del genere?\n La risposta è sicuramente ${answer}`,
        `Molto probabilmente la risposta è ${answer}`,
        `Questa la so! La risposta è ${answer}`,
        `Sono indeciso forse è ${answer}`,
      ];
      alert.innerHTML =
        friendsAnswer[Math.floor(Math.random() * friendsAnswer.length)];
    }
  }
  phoneButton.classList.add("hide");
}
```

Infine l'*aiuto del pubblico* simula la percentuale delle risposte scelte da un ipotetico pubblico della sala, queste percentuali sono rappresentate attraverso una rappresentazione grafica ad istogramma. La lunghezza delle colonne del grafico è scelta casualmente da un array contenente varie lunghezze possibili in pixel. Ovviamente la risposta corretta sarà rappresentata nel grafico come la risposta che ha ottenuto più voti. L'aiuto potrà essere utilizzato solo una volta e dopo la sua esecuzione verrà nascosto.

```
function askAudience() {
  var audio = new Audio("/sounds/help.mp3");
  audio.play();
  alert.classList.remove("show");
  audienceVoters.classList.add("show");
  let A = document.getElementById("a");
  let B = document.getElementById("b");
  let C = document.getElementById("c");
  let D = document.getElementById("d");
  let correctHeight = ["150px", "120px", "165px", "110px", "145px"]; //Create
  an array of height so you can have a random value
  let wrongHeight = [
    "30px", "20px", "90px", "45px", "80px", "25px", "40px", "100px", "85px", "55px",
    "43px",
  ];

  if (answerButtonsElement.children[0].dataset.correct) {
    A.style.height =
      correctHeight[Math.floor(Math.random() * correctHeight.length)];
    B.style.height =
      wrongHeight[Math.floor(Math.random() * wrongHeight.length)];
    C.style.height =
      wrongHeight[Math.floor(Math.random() * wrongHeight.length)];
    D.style.height =
      wrongHeight[Math.floor(Math.random() * wrongHeight.length)];
  } else if (answerButtonsElement.children[1].dataset.correct) {
    A.style.height =
      wrongHeight[Math.floor(Math.random() * wrongHeight.length)];
    B.style.height =
      correctHeight[Math.floor(Math.random() * correctHeight.length)];
    . . .
  } else if (answerButtonsElement.children[2].dataset.correct) {
    . . .
  } else if (answerButtonsElement.children[3].dataset.correct) {
    . . .
  }
  audienceButton.classList.add("hide");
}
```

### 3.3 Fine della partita

La sessione di gioco termina con il ritiro dell'utente, la selezione di una risposta errata o il raggiungimento del massimo montepremi. La conclusione della sessione coincide con il salvataggio dei dati dell'utente all'interno del `localStorage`.

La funzione:

```
function gameOver() {  
  gameElement.classList.add("hide");  
  endElement.classList.remove("hide");  
  highscoreContainer.classList.add("hide");  
  highscoreInputName.value = ("");  
  var n = new Date();  
  endTime = n.getTime();  
  sessionTimeMin = Math.floor((endTime - startTime) / 1000 / 60);  
  sessionTimeSec = Math.floor((endTime - startTime) / 1000);  
  temp = 1;  
}
```

Oltre che permettere di visualizzare le risposte esatte ed i bitcoin guadagnati, è adibito alla ricezione in input dell'username dell'utente che viene associato ai dati della sessione. Il tempo totale della sessione è calcolato grazie alla funzione `new Date()` che permette di ricavare il tempo trascorso dal 1 Gennaio 1970 ad oggi in millisecondi. Sottraendo dal `endTime` il `startTime` si trova il tempo trascorso all'interno della sessione di gioco in millisecondi. Ottenuti i quattro valori di username, punteggio, data e tempo della sessione, il software li salva all'interno del `LocalStorage`.

## Il LocalStorage

HTML5 ha introdotto il concetto di Web Storage, ovvero un contenitore di tipo chiave-valore dove poter salvare dei dati all'interno del browser. A differenza dei cookie, gli webstorage sono più capienti e soprattutto le informazioni contenute non vengono mai inviate al server. Ci sono solo due vincoli: il limite di spazio disponibile (fino a 5MB) e il tipo di dato supportato. Si possono salvare infatti solo coppie di stringhe, ma questa difficoltà viene facilmente superata con la serializzazione di un oggetto JSON ed il suo salvataggio. Gli Web Storage si dividono in due categorie:

- **LocalStorage:** può contenere dati, senza scadenza, accessibili da tutte le pagine di uno stesso dominio.
- **SessionStorage:** può contenere dati relativi alla sessione utente, ovvero isolati nel tab in cui vi si fa accesso. Vengono cancellati alla chiusura del tab stesso.

Abbiamo scelto la prima opzione poiché persiste tra più finestre e nel tempo.



Le funzioni che generano la classifica, salvano i dati in LocalStorage e serializzano gli oggetti JSON sono le seguenti:

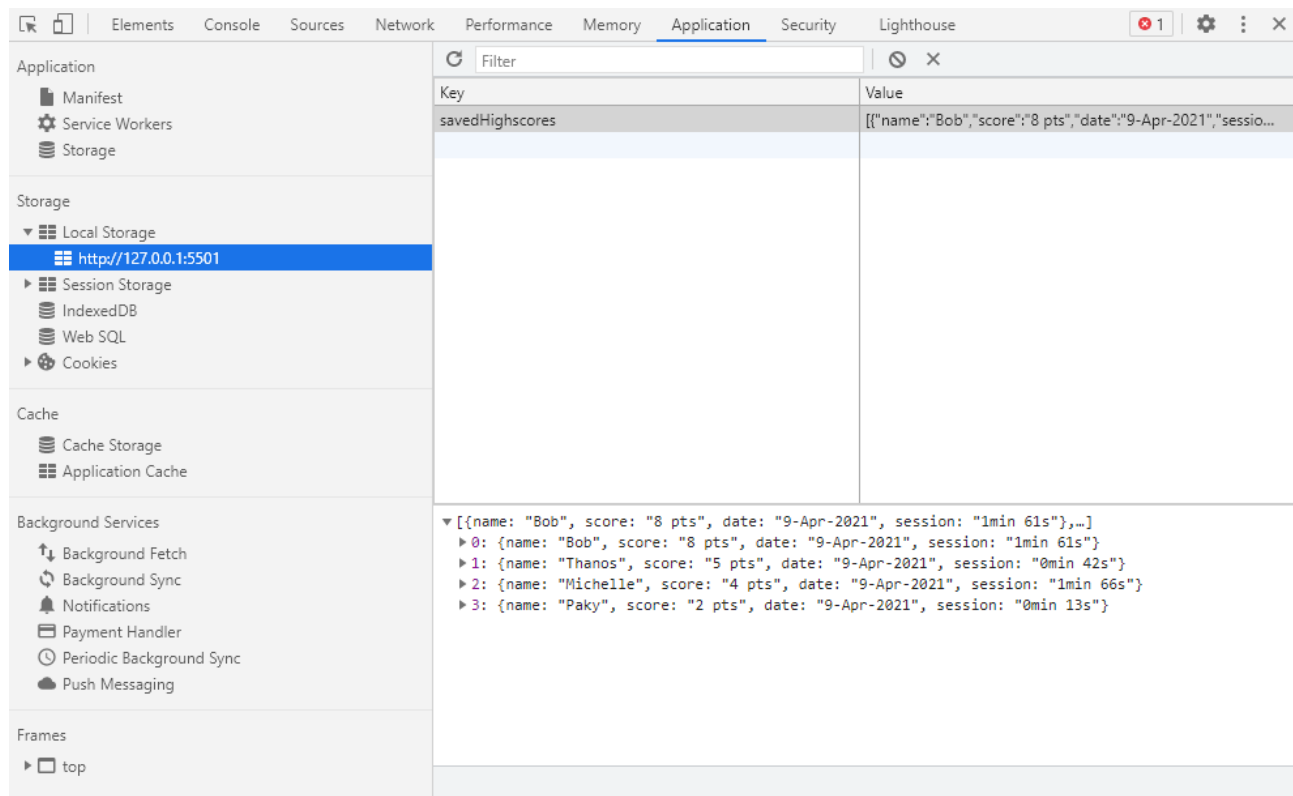
```
function highscore() {
  if (highscoreInputName.value === "") {
    alert("Initials cannot be blank");
    return false;
  } else {
    var savedHighscores =
      JSON.parse(localStorage.getItem("savedHighscores")) || [];
    var currentUser = highscoreInputName.value.trim();
    var currentHighscore = {
      name: currentUser,
      score: score + ' pts',
      date : join(new Date, a, '-'),
      session : sessionTimeMin + 'min' + ' ' + sessionTimeSec + 's'
    };

    highscoreContainer.style.display = "flex";
    highscoreDiv.style.display = "block";

    savedHighscores.push(currentHighscore);
    localStorage.setItem("savedHighscores", JSON.stringify(savedHighscores));
    generateHighscores();
  }
  endElement.classList.add("hide");
}
```

```
function generateHighscores() {
  highscoreDisplayName.innerHTML = "";
  highscoreDisplayScore.innerHTML = "";
  highscoreDisplayDate.innerHTML = "";
  highscoreDisplaySession.innerHTML = "";
  var highscores = JSON.parse(localStorage.getItem("savedHighscores")) || [];
  for (i = 0; i < highscores.length; i++) {
    var newNameSpan = document.createElement("ul");
    var newScoreSpan = document.createElement("ul");
    var newDateSpan = document.createElement("ul");
    var newDateSession = document.createElement("ul");
    newNameSpan.textContent = highscores[i].name;
    newScoreSpan.textContent = highscores[i].score;
    newDateSpan.textContent = highscores[i].date;
    newDateSession.textContent = highscores[i].session;
    highscoreDisplayName.appendChild(newNameSpan);
    highscoreDisplayScore.appendChild(newScoreSpan);
    highscoreDisplayDate.appendChild(newDateSpan);
    highscoreDisplaySession.appendChild(newDateSession);
  }
}
```

Il salvataggio si può verificare attraverso il comando `ispeziona -> application -> storage`  
`-> Local Storage`



## Classifica

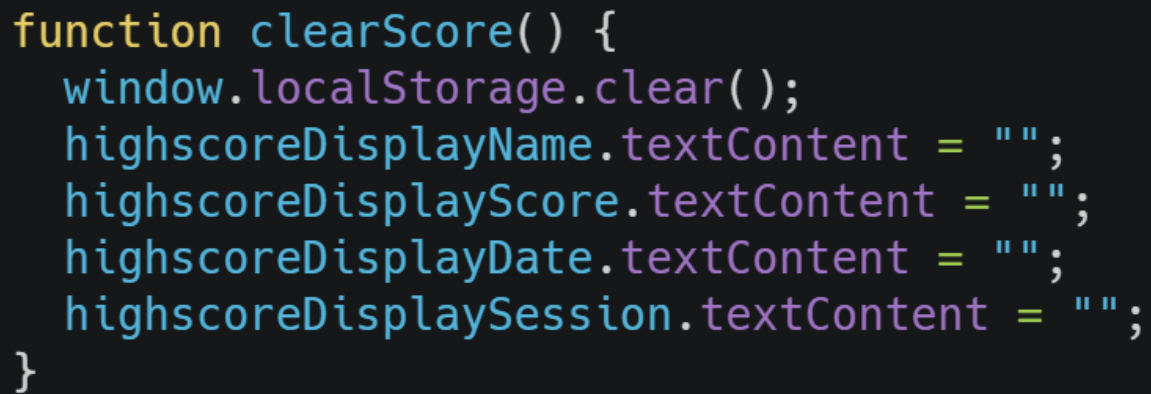
Dopo aver effettuato la partita è possibile visualizzare le sessioni salvate all'interno della classifica generale. Si accede tramite la schermata principale selezionando il pulsante "Classifica"

Nome Utente	Punteggio	Data	Tempo Sessione
Bob	8 pts	9-Apr-2021	1min 61s
Thanos	5 pts	9-Apr-2021	0min 42s
Michelle	4 pts	9-Apr-2021	1min 66s
Paky	2 pts	9-Apr-2021	0min 13s

Cancellla classifica

indietro

Infine all'interno della sezione classifica è possibile svuotare i dati salvati all'interno del LocalStorage.



```
function clearScore() {  
    window.localStorage.clear();  
    highscoreDisplayName.textContent = "";  
    highscoreDisplayScore.textContent = "";  
    highscoreDisplayDate.textContent = "";  
    highscoreDisplaySession.textContent = "";  
}
```

