ZIQUIZ

# Indice

1 Indice 2

2 Introduzione 4

2.1 Informazioni sul progetto 4

2.2 Abstract 4

2.3 Scopo 4

3 Analisi 5

3.1 Analisi del dominio 5

3.2 Analisi e specifica dei requisiti 5

3.3 Use case 8

3.4 Pianificazione 9

3.5 Descrizione Gantt 10

3.5.1 Analisi 10

3.5.2 Progettazione 10

3.5.3 Implementazione 10

3.5.4 Test 10

3.5.5 Documentazione 10

3.6 Analisi dei mezzi 11

3.6.1 Software 11

3.6.2 Hardware 11

4 Progettazione 12

4.1 Design dei dati e database 12

4.2 Design dell’architettura del sistema 12

4.3 Design delle interfacce 13

4.4 Design procedurale 19

5 Implementazione 21

5.1 HTML 21

5.1.1 Menù 21

5.2 Javascript 23

5.2.1 Tasto back 23

5.2.2 Shuffle 23

5.3 PHP 23

5.3.1 Mandare i dati a php 23

5.4 PHP salvataggio record 25

5.5 Prendere i quiz generali 26

5.6 Prendere i quiz personali 27

5.7 Login 28

5.7.1 Firebsae 28

5.7.2 Motivo scelta 28

5.7.3 Configurazione 29

5.8 Configurazione Autenticazione 30

6 Test 34

6.1 Protocollo di test 34

6.2 Risultati test 37

6.3 Mancanze/limitazioni conosciute 41

7 Consuntivo 42

7.1 Descrizione Gantt Consuntivo 42

7.1.1 Analisi 42

7.1.2 Progettazione 43

7.1.3 Implementazione 43

7.1.4 Test 43

7.1.5 Documentazione 43

8 Conclusioni 44

8.1 Sviluppi futuri 44

8.2 Considerazioni personali 44

9 Bibliografia 45

9.1 Sitografia 45

10 Indice delle figure 45

11 Allegati 46

# Introduzione

## Informazioni sul progetto

Allievo: Robin Sartore

Docente: Michel Palucci

Scuola: SAM Trevano

Sezione: Informatica

Modulo:306

Data inizio:11.09.2024

Data fine: 18.12.2024

## Abstract

*The project involved creating a web-based general knowledge quiz application where users can study, play, and create quizzes. The platform includes email registration to track records and allows users to challenge friends and have fun.*

*The development required skills in PHP, JavaScript, HTML, Firebase for database management, and user authentication. Records are saved in JSON files on the server, and both personal and general scores are displayed. Users can also create and share custom quizzes.*

*This project utilized knowledge from my studies and introduced new skills. Although some features were not implemented, the application is well-designed, functional, and visually appealing.*

## Scopo

Lo scopo del progetto consiste nella creazione di un’applicazione per un quiz di nome Ziquiz, con varie modalità di domande, difficoltà, e tipologie e si può usare per creare un quiz personalizzato e nel caso usarlo per studiare si visualizzerà anche un proprio punteggio per le varie categorie e difficoltà e verrà salvato localmente un top score.

Un altro scopo è quello di iniziare a fare progetti dato che l’anno prossimo dovrò fare il LPI e quindi è meglio iniziare a prepararsi prima per arrivare al meglio preparati e con abbastanza esperienza per riuscire a fare un buon progetto.

# Analisi

## Analisi del dominio

Il prodotto potrà essere usato come svago nel tempo libero e sfidare gli amici a chi fa il punteggio più alto e vedere chi ha più conoscenze sulle varie categorie o un allenamento per studiare una qualsiasi materia o formazione con domande personalizzate, chiaramente esistono già vari quiz online su cose diverse ma questo si differenzia dagli altri perché oltre ad avere una modalità studio i quiz sono su varie categorie con varie difficoltà e mi sembra molto innovativo.

Gli utenti potranno migliorare e sviluppare una conoscenza su vari argomenti e migliorarsi ogni volta superando il loro top score.

I quiz che ci sono già nel web sono pieni di animazioni e effetti quindi devo provare ad avere anch’io qualche animazione e non qualcosa di monotono.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **Req-01** | |
| **Nome** | Creazione interfaccia Menu |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Comprensibile all’utente |

|  |  |
| --- | --- |
| **Req-02** | |
| **Nome** | Possibilità di scegliere la tipologia di quiz |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Comprensibile all’utente |

|  |  |
| --- | --- |
| **Req-03** | |
| **Nome** | Possibilità di creare un quiz |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Comprensibile all’utente |

|  |  |
| --- | --- |
| **Req-04** | |
| **Nome** | Possibilità di scegliere la difficolta del quiz |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Comprensibile all’utente |

|  |  |
| --- | --- |
| **Req-05** | |
| **Nome** | Possibilita di vedere il Topscore |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Comprensibile all’utente |

|  |  |
| --- | --- |
| **Req-06** | |
| **Nome** | Studiare su quiz creati |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Comprensibile all’utente |

|  |  |
| --- | --- |
| **Req-07** | |
| **Nome** | Musica di sottofondo |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Comprensibile all’utente |

|  |  |
| --- | --- |
| **Req-08** | |
| **Nome** | Visualizzare se si ha sbagliato |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Comprensibile all’utente |

|  |  |
| --- | --- |
| **Req-09** | |
| **Nome** | Aggiunta animazioni |
| **Priorità** | 2 |
| **Versione** | 1.0 |

|  |  |
| --- | --- |
| **Req-010** | |
| **Nome** | Scegliere una modalità di studio |
| **Priorità** | 2 |
| **Versione** | 1.0 |

## Use case

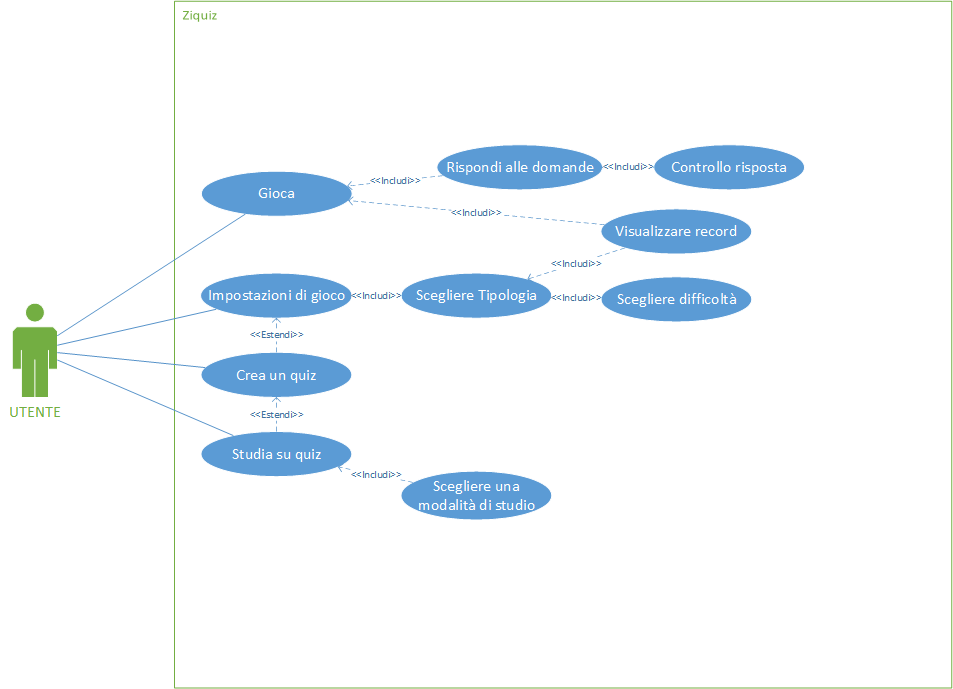


Figura 1 use case

Questo è lo schema delle use case del progetto. In pratica l’utente può scegliere le impostazioni del quiz, la tipologia e la difficolta di cui può vedere il suo record in quella specifica modalità, può anche creare un quiz e studiare sul quiz con varie diverse modalità di studio e può anche giocare rispondendo alle domande e visualizzando gli errori.

## Pianificazione

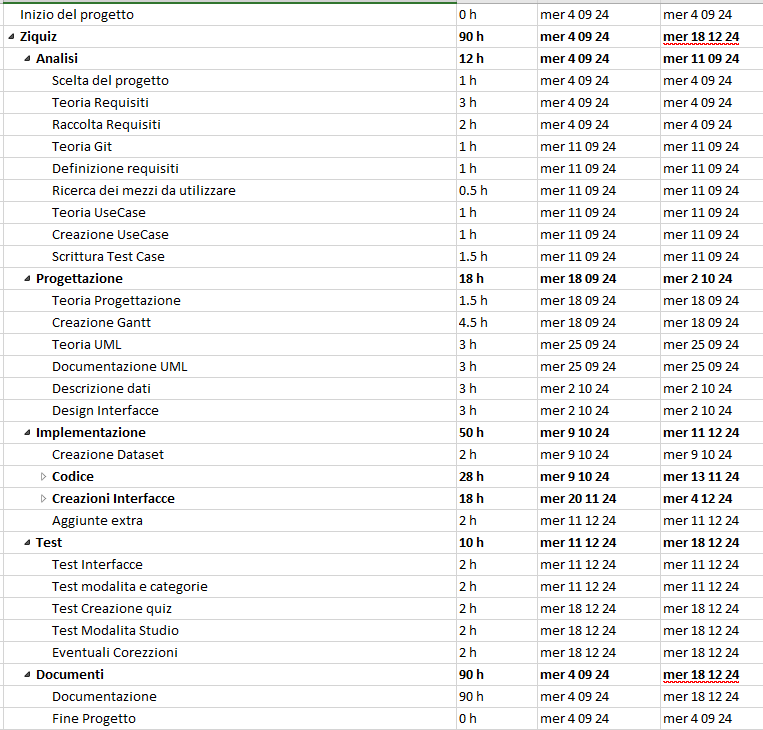


Figura 2 Esempio di diagramma di Gantt

Questo è il Gantt di 90 ore del mio progetto, ho applicato una metologia waterfall.

## Descrizione Gantt

### Analisi

Per l’analisi ho programmato 2 giornate lavorative da 6 ore includendo la teoria fatta a lezione e ho ipotizzato dei tempi per le varie parti.

### Progettazione

Per la progettazione ho calcolato più ore della Analisi, ho messo una lunga tempistica soprattutto per la costruzione del gantt.

### Implementazione

L’implementazione è la parte più importante del progetto per questo ho previsto 50 ore di lavoro dato che devo ancora bene capire come leggere le domande e come crearle.

### Test

Per i test ho previsto 10 ore per i vari test e le eventuali correzioni ho preso più di un giorno perché non riuscirei magari a correggere molti errori in un giorno.

### Documentazione

Per la documentazione ho calcolato la durata intera quindi 90 ore anche se in realtà prenderà il 305 del tempo ma verrà fatta man mano.

## Analisi dei mezzi

Per la realizzazione del progetto è stato utilizzato un PC scolastico con Visual Studio 2022

Elencare e descrivere i mezzi disponibili per la realizzazione del progetto. Ricordarsi di sempre descrivere nel dettaglio le versioni e il modello di riferimento.

### Software

Visual studio 2022

Hosting servito dalla scuola

### Hardware

Computer fornito dalla scuola:

32 gb di RAM

Windows 10 Education 22H2

Processore 13th Gen Intel(R) Core(TM) i7-13700 2.10 GHz

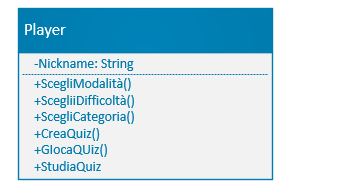
# Progettazione

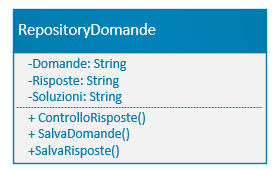
## Design dei dati e database

L’architettura del sistema si basa su un file json in cui ci saranno le domande e le risposte per le domande e un altro file in cui verrano salvate localmente le domande e le risposte del utente

## Design dell’architettura del sistema

Diagramma di classi





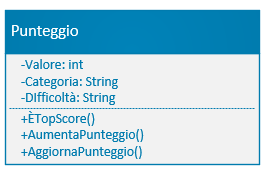


Figura 3 Diagramma di classi

## Design delle interfacce

Figura 4 Interfaccia menù

Questa è l’interfaccia del menu principale qui l’utente potrà scegliere cosa fare e inserire il suo nickname, e si può leggere il titolo dell’applicazione.

Per vedere quali colori stavano meglio insieme ho usato questo sito [link](https://color.adobe.com/it/create/color-wheel).

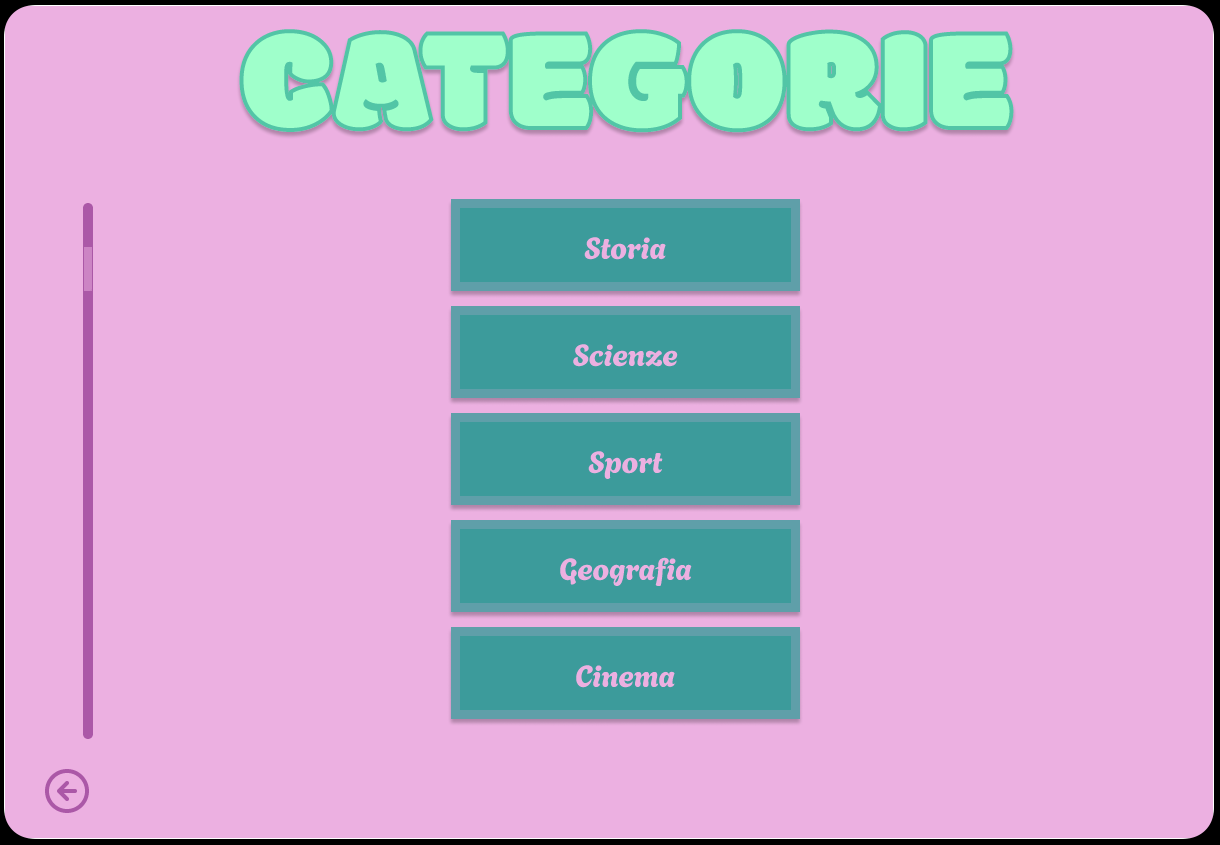


Figura 5 Categoria

Qua si potrà selezionare la categoria che più ci piace, si potrà scegliere anche un quiz creato dall’utente.



Figura 6 Difficoltà

Qui invece si sceglierà la difficoltà e si potrà vedere il topscore in base a ogni difficoltà.



Figura 7 Esempio domanda

In Questa interfaccia si può vedere come viene mostrato il quiz con un punteggio in alto a destra



Figura 8 Esempio errore

Qua si può vedere come viene mostrato un errore e la risposta correttà.

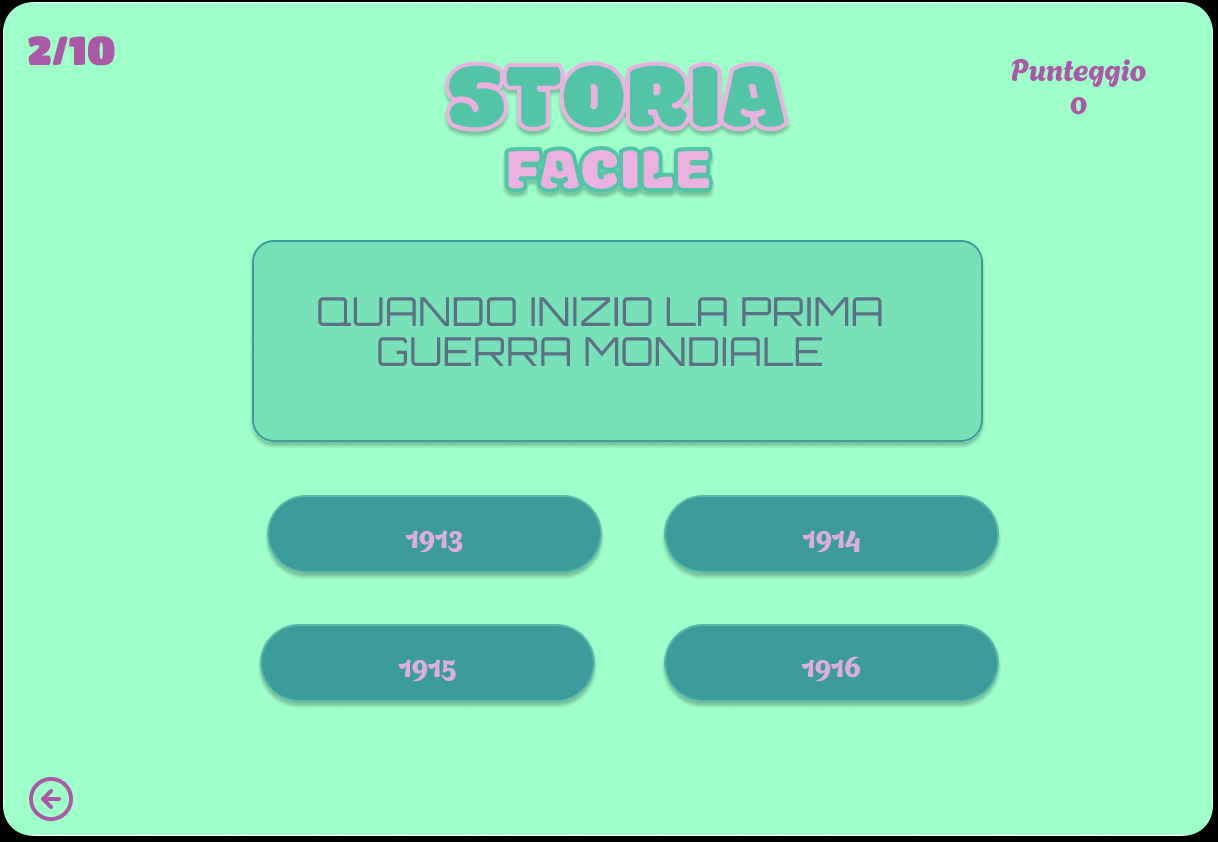


Figura 9 Esempio domanda 2

Questa è un'altra domanda e si può vedere in alto a sinistra che è stat modificato il counter delle domande.

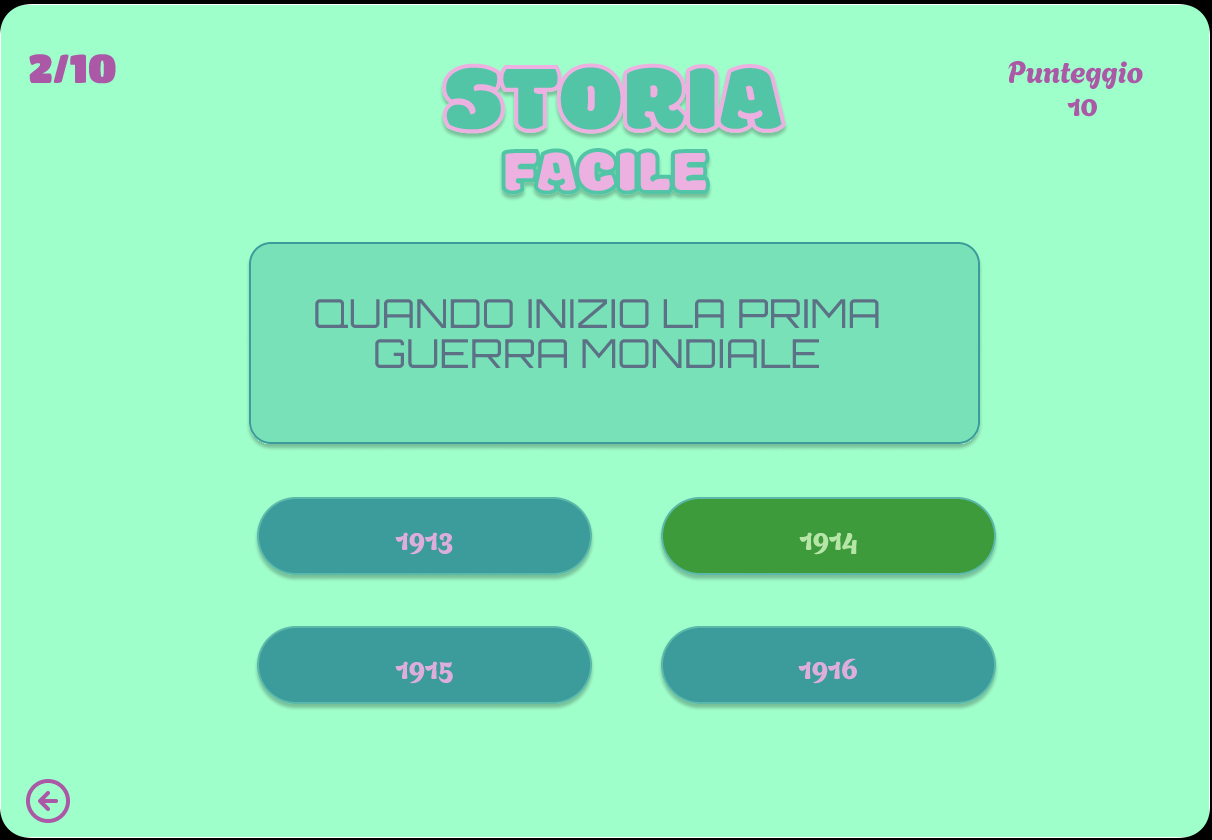


Figura 10 Esempio correzione 2

In questa interfaccia si può vedere invece quando viene inserita una risposta corretta , e si vede anche il punteggio aumentare.



Figura 11 Uscita da quiz

Qui viene mostrato il messaggio di quando si vuole uscire dal quiz



Figura 12 Modalità Crea

Qua si vede la schermata principale in cui si vedono i vari dati da inserire per creare un quiz.

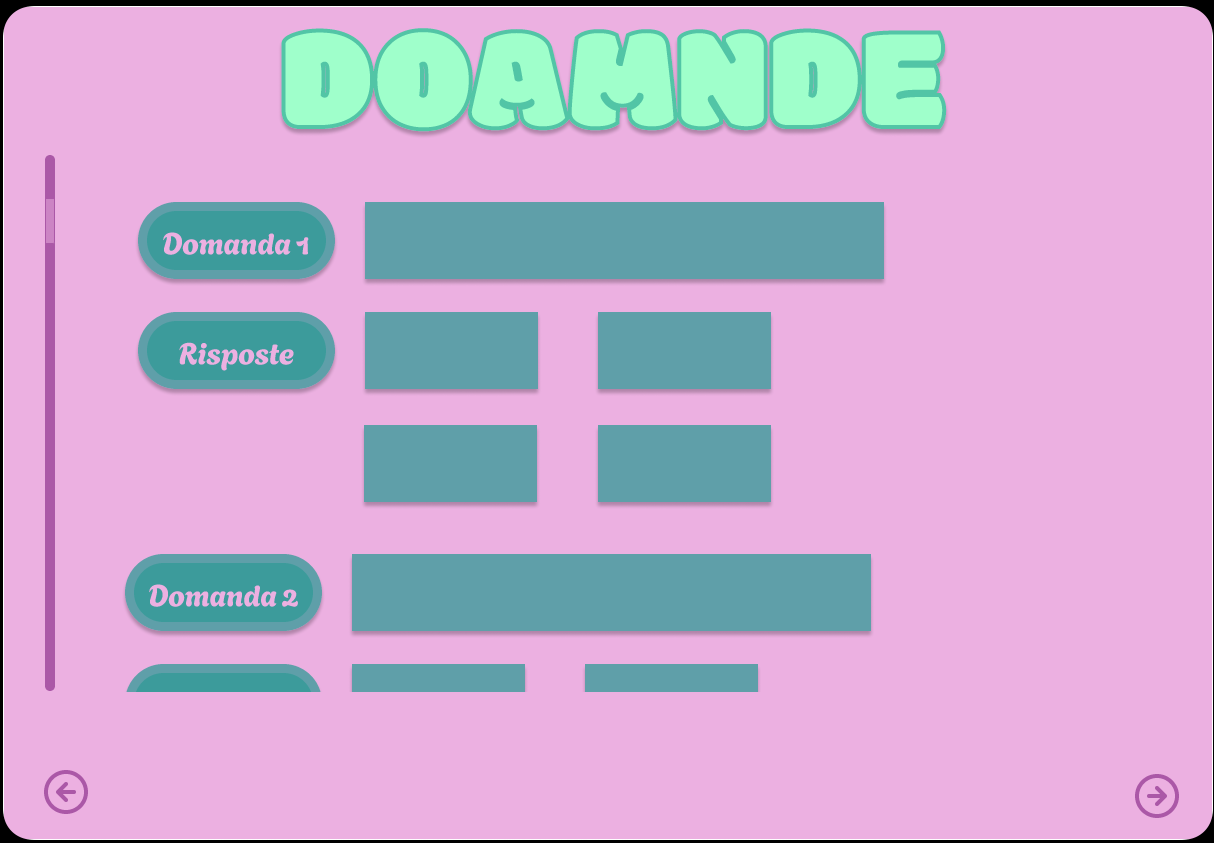


Figura 13 Domande modalità crea

Qua invece viene mostrata la schermata in cui si selezioneranno le domande e le risposte da inserire nel nostro quiz personalizzato.

## Design procedurale

Diagramma Swimlanes

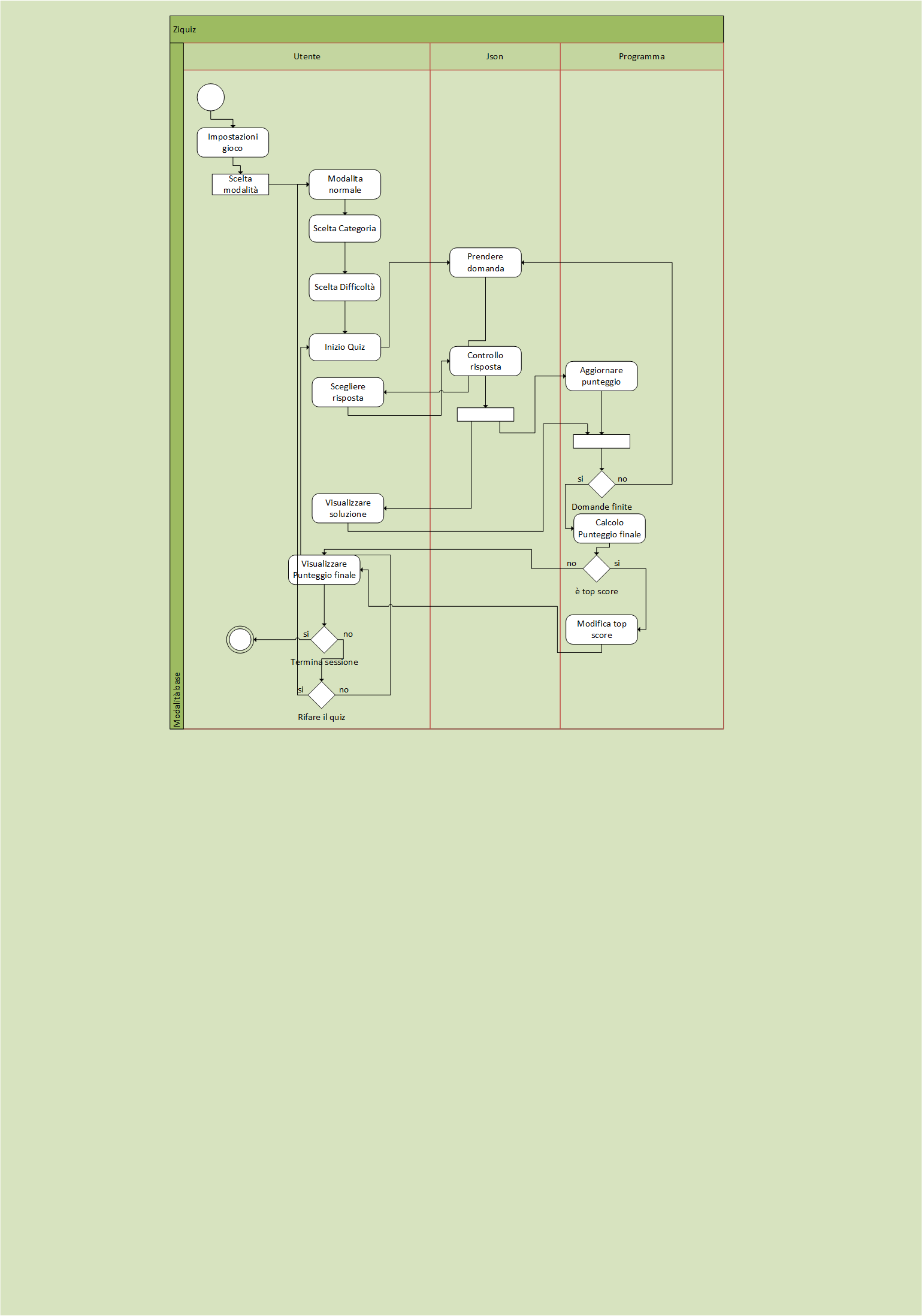


Figura 14 Diagramma Swimlanes base

Diagramma Swimlanes 2

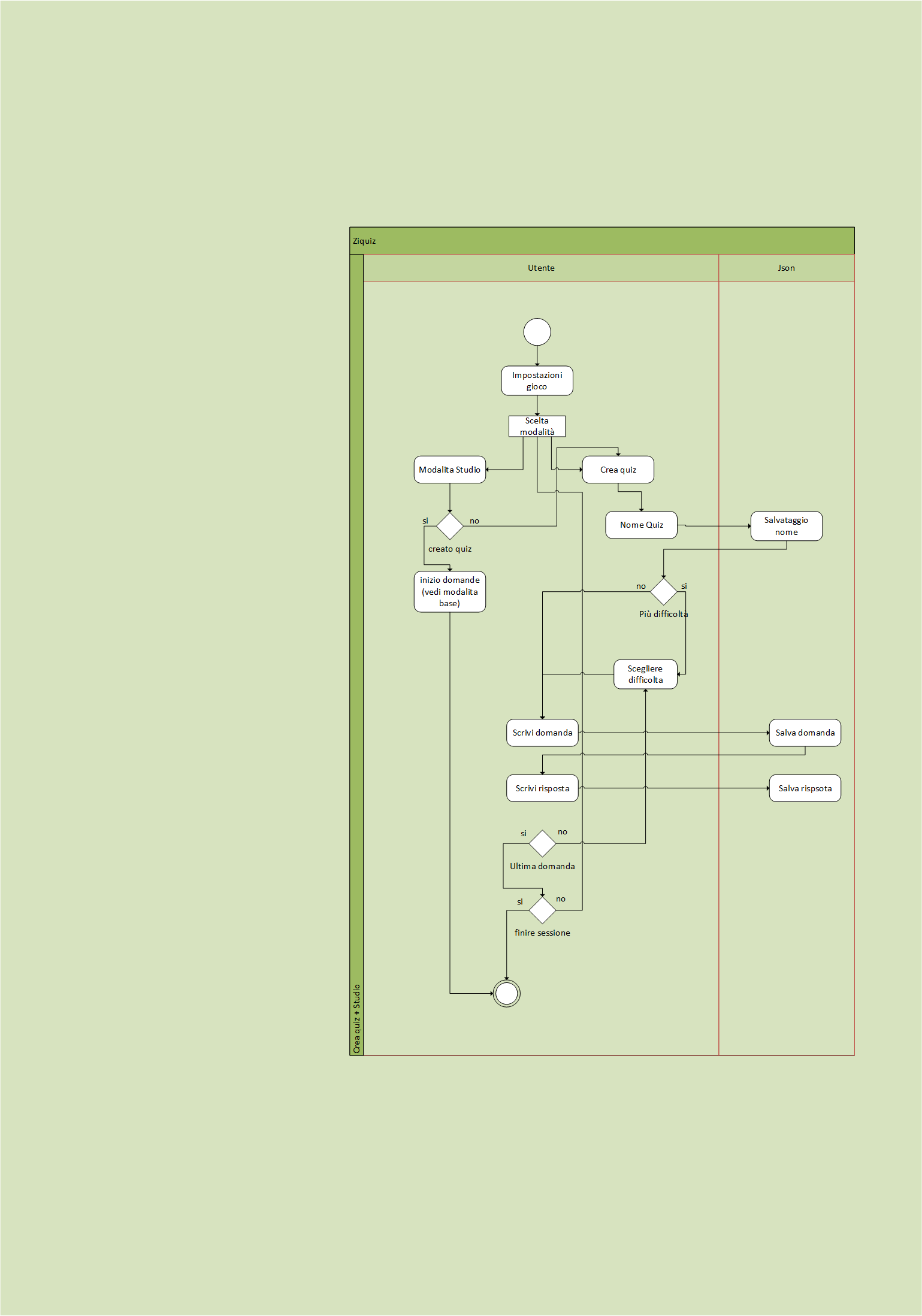
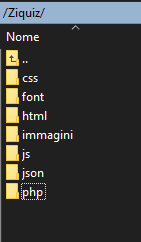


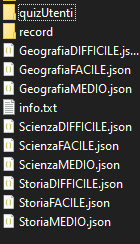
Figura 15 Diagramma Swimlanes (studio + creaquiz)

# Implementazione

## Struttura



### Struttura Json



## HTML

### Menù

Per ora il menù lo strutturato così:

<!DOCTYPE HTML>

<html lang="it">

<head>

  <meta charset="UTF-8">

  <meta name="description" content="Quiz">

  <meta name="keywords" content="Quiz">

  <meta name="author" content="Robin Sartore">

  <title>ROBIN SARTORE</title>

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">

  <link href="../css/menu.css" rel="stylesheet" type="text/css">

  <link rel="icon" type="image/x-icon" href="image/favicon.ico">

<script src="../js/menu.js"></script>

<script src="../js/nickname.js"></script>

<script src="../js/logout.js"></script>

<script type="module" src="../js/firebaseConfig.js"></script>

<script type="module" src="../js/auth.js"></script>

</head>

<body onload="refresh(),checklgbt()">

    <div id="nicknamefinal" class="nickname">

    </div>

    <div class="titolo">

        <SPAN>Z<D>I</D>Q<D>U</D>I<D>Z</D>!</SPAN>

    </div>

    <button id="loginButton" class="nicknamebutton" onclick="accesso()" >Login</button>

    <button id="logoutButton" class="nicknamebutton">Logout</button>

    <button class="leaderboard" onclick="leaderboard()" >leaderboard</button>

    <button class="gioca" onclick="gioca()">GIOCA</button>

    <br>

    <button class="crea" onclick="crea()">CREA</button>

    <h6>version 1.5</h6>

</body>



Figura 16 Home

### Crea

Questo è il codice per la view della creazione di un quiz:

<!--VERSIONE:

1.0.0

-->

<!DOCTYPE HTML>

<html lang="it">

<head>

  <meta charset="UTF-8">

  <meta name="description" content="Quiz">

  <meta name="keywords" content="Quiz">

  <meta name="author" content="Robin Sartore">

  <title>ROBIN SARTORE</title>

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">

  <link href="../css/crea.css" rel="stylesheet" type="text/css">

  <link rel="icon" type="image/x-icon" href="image/favicon.ico">

<script src="../js/nickname.js"></script>

<script src="../js/crea.js"></script>

</head>

<body onload="refresh()">

    <div id="nicknamefinal" class="nickname">

    </div>

    <div class="titolo">

     CREA QUIZ

    </div>

    <button class="btnCrea" onclick="info()">Info</button>

<br>

    <input class="nomeQuiz" type="text" id="nomeQuiz" placeholder="Nome Quiz">

    <input class="" type="file" id="file" name="">

<center>

  <input type="radio" id="publico" name="pub" value="Pubblico">

  <label for="publico">Pubblico</label>

  <br>

  <input type="radio" id="Facile" name="difficolta" value="FACILE">

  <label for="FACILE">FACILE</label>

  <input type="radio" id="Medio" name="difficolta" value="MEDIO">

  <label for="MEDIO">MEDIO</label>

  <input type="radio" id="Difficile" name="difficolta" value="DIFFICILE">

  <label for="DIFFICILE">DIFFICILE</label>

</center>

    <button class="btnCrea" onclick="crea()">Crea</button>

    <div class="back" onclick="back()">

      <img src="../immagini/back.png" width="50px">

  </div>

</body>

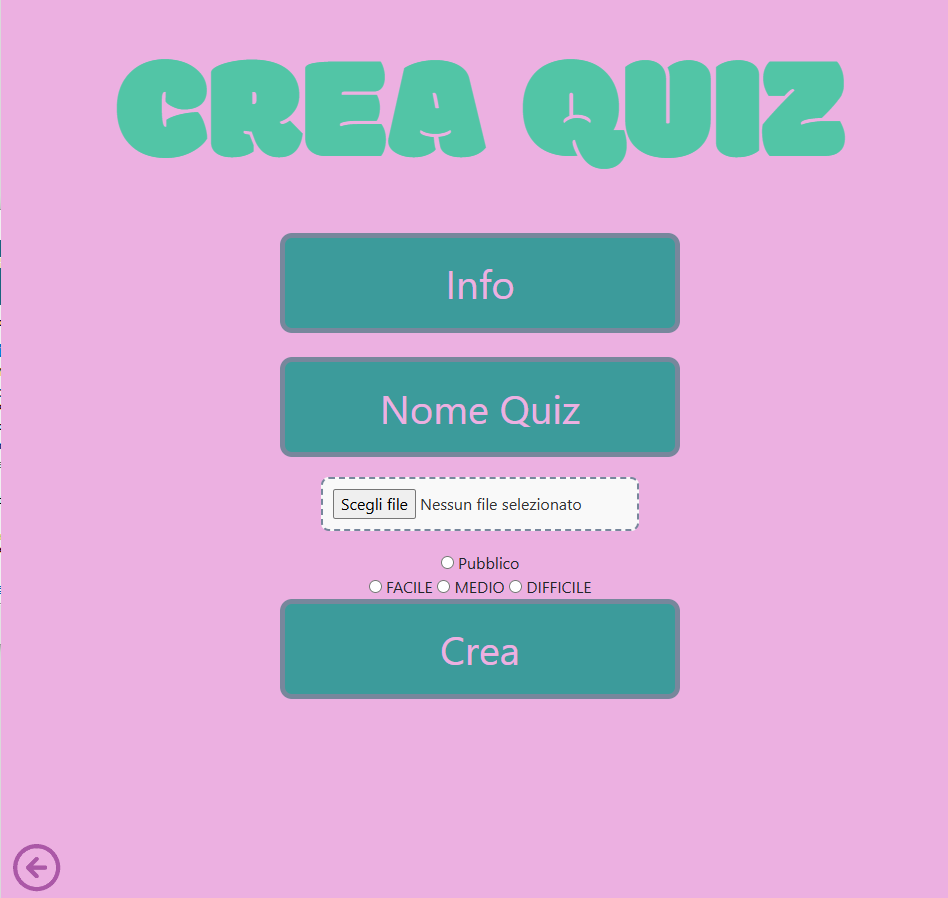


Figura 17 Crea Quiz

## Javascript

### Tasto back

Per il tasto back ho inserito un png e quando viene cliccato torna alla pagina precedente

function back(){

    close();

    var pag=window.open("Ziquiz.html");

}



Figura 18 Back button

### Shuffle

Per avere le risposte in modo casuale ho creato un array da uno a quattro e ho associato i numeri alle classi dei bottoni contenenti le risposte per assegnarlo sempre casuale ho creato un metodo mescola che mescola l’array

function mescola(arr) {

    arr.sort(() => Math.random() - 0.5);

    }

Il metodo sort ordina l’array in base a una funzione di confronto, viene generato un numero casuale tra -0.5 e 0.5. Però questo metodo non è totalmente casuale, ma provando un altro metodo non funzionava e non sono riuscito a capire il perché.

### Carica domande

Questa funzione prende con il fetch il json:

function caricaDomande(){

    console.log('../json/'+categoria+difficolta+'.json');

    fetch('https://samtinfo.ch/i22RobinSartore/Ziquiz/json/'+categoria+difficolta+'.json')

        .then(response => {

            if (!response.ok) {

                throw new Error('Errore nell\'apertura');

            }

            return response.json();

        })

        .then(data => {

            impostaDomanda(data[nrdomanda].domanda, "domanda");

            impostaDomanda(data[nrdomanda].risposta1, "risposta"+posrisposte[0]);

            impostaDomanda(data[nrdomanda].risposta2, "risposta"+posrisposte[1]);

            impostaDomanda(data[nrdomanda].risposta3, "risposta"+posrisposte[2]);

            impostaDomanda(data[nrdomanda].rispostaGiusta, "risposta"+posrisposte[3]);

            domande=data;

        })

        .catch(error => {

            console.error('Problema con il fetch', error);

        });

}

In questo codice prendo le domande e per ogni domanda faccio imposta domanda che è la funzione per impostare le domande.

### ImpostaDomanda

function impostaDomanda(x, y) {

    document.getElementById(y).innerHTML = x;

}

Prende il nome che sarebbe l’indice del json e trova l’elemento a cui associarlo e glielo associa.

### Prossima Domanda

Questo è il codice per impostare le domande:

function domandaSuccesiva(){

    document.body.style.backgroundColor = "#52C5A6";

    mescola(posrisposte);

    cdomande++;

    document.getElementById("risposta1").disabled = false;

    document.getElementById("risposta2").disabled = false;

    document.getElementById("risposta3").disabled = false;

    document.getElementById("risposta4").disabled = false;

    if(difficolta=="FACILE"){

        if(cdomande<=10){

            totdomande=cdomande+"/10";

        }

        document.getElementById("nrdomande").innerHTML= totdomande;

        if(nrdomande.length<10){

            nrdomanda=Math.floor(Math.random() \* 10);

            if(!nrdomande.includes(nrdomanda)){

                nrdomande.push(nrdomanda);

            }

            else{

                while(nrdomande.includes(nrdomanda)){

                    nrdomanda=Math.floor(Math.random() \* 10);

                }

                nrdomande.push(nrdomanda);

            }

            impostaDomanda(domande[nrdomanda].domanda, "domanda");

            impostaDomanda(domande[nrdomanda].risposta1, "risposta"+posrisposte[0]);

            impostaDomanda(domande[nrdomanda].risposta2, "risposta"+posrisposte[1]);

            impostaDomanda(domande[nrdomanda].risposta3, "risposta"+posrisposte[2]);

            impostaDomanda(domande[nrdomanda].rispostaGiusta, "risposta"+posrisposte[3]);

        }

        else{

            salvaPunteggio()

            setTimeout(risultato,1000);

        }

    }

    else if(difficolta=="MEDIO"){

        if(cdomande<=20){

            totdomande=cdomande+"/20";

        }

        document.getElementById("nrdomande").innerHTML= totdomande;

        if(nrdomande.length<20){

            nrdomanda=Math.floor(Math.random() \* 20);

            if(!nrdomande.includes(nrdomanda)){

                nrdomande.push(nrdomanda);

            }

            else{

                while(nrdomande.includes(nrdomanda)){

                    nrdomanda=Math.floor(Math.random() \* 20);

                }

                nrdomande.push(nrdomanda);

            }

            impostaDomanda(domande[nrdomanda].domanda, "domanda");

            impostaDomanda(domande[nrdomanda].risposta1, "risposta"+posrisposte[0]);

            impostaDomanda(domande[nrdomanda].risposta2, "risposta"+posrisposte[1]);

            impostaDomanda(domande[nrdomanda].risposta3, "risposta"+posrisposte[2]);

            impostaDomanda(domande[nrdomanda].rispostaGiusta, "risposta"+posrisposte[3]);

        }

        else{

            salvaPunteggio()

            setTimeout(risultato,1000);

        }

    }

    else{

        if(cdomande<=30){

            totdomande=cdomande+"/30";

        }

        document.getElementById("nrdomande").innerHTML=totdomande;

        if(nrdomande.length<30){

            nrdomanda=Math.floor(Math.random() \* 30);

            console.log(nrdomanda);

            if(!nrdomande.includes(nrdomanda)){

                nrdomande.push(nrdomanda);

            }

            else{

                while(nrdomande.includes(nrdomanda)){

                    nrdomanda=Math.floor(Math.random() \* 30);

                }

                nrdomande.push(nrdomanda);

            }

            console.log(nrdomanda);

            impostaDomanda(domande[nrdomanda].domanda, "domanda");

            impostaDomanda(domande[nrdomanda].risposta1, "risposta"+posrisposte[0]);

            impostaDomanda(domande[nrdomanda].risposta2, "risposta"+posrisposte[1]);

            impostaDomanda(domande[nrdomanda].risposta3, "risposta"+posrisposte[2]);

            impostaDomanda(domande[nrdomanda].rispostaGiusta, "risposta"+posrisposte[3]);

        }

        else{

            salvaPunteggio()

            setTimeout(risultato,1000);

        }

    }

}

In questa funzione in base alla difficoltà carico le domande in modo casuale usando il metodo mescola per mescolare l’ordine e controllo se le domande sono finite.

### Verifica Domanda

Per verificare le domande uso questa funzione:

function verificaDomanda(x){

    if(x==posrisposte[3]){

        document.body.style.backgroundColor = "green";

        document.getElementById("punteggio").innerHTML =punti;

        sessionStorage.setItem(4,punti);

        if(difficolta=="FACILE"){

           if(cdomande<10) {

            punti+=10;

           }

        }

        else if(difficolta=="MEDIO"){

                if(cdomande<20) {

                 punti+=20;

                }

            }

        else{

            if(cdomande<30) {

                punti+=30;

               }

        }

    }

    else{

        document.body.style.backgroundColor = "red";

    }

    document.getElementById("risposta1").disabled = true;

    document.getElementById("risposta2").disabled = true;

    document.getElementById("risposta3").disabled = true;

    document.getElementById("risposta4").disabled = true;

    setTimeout(domandaSuccesiva,1000);

}

Nella funzione aumento l’indice della domanda cambio il colore se è giusta o no e richiamo la domanda succesiva.

### Mostra record

Questa Funzione Mostra i record personali in base al nome:

function mostraDatiSalvati() {

    // Ottieni il nickname da sessionStorage

    let nickname = sessionStorage.getItem("1");

    // Verifica se il nickname è valido

    if (!nickname) {

        console.error("Errore: il nickname non è stato fornito.");

        return;

    }

    // Esegui la richiesta POST per cercare il nome nel JSON

    fetch('../php/prendi.php', {

        method: 'POST',

        headers: {

            'Content-Type': 'application/json'

        },

        body: JSON.stringify({ name: nickname })  // Invia il nome nel corpo della richiesta

    })

    .then(response => {

        if (!response.ok) {

            throw new Error(`Errore nella risposta: ${response.status}`);

        }

        return response.json(); // Recupera la risposta come JSON

    })

    .then(data => {

        console.log("Risultati trovati:", data);

        const resultDiv = document.getElementById("results");

        resultDiv.innerHTML = ""; // Svuota i risultati precedenti

        if (data.length > 0) {

            let currentCategory = "";

            let currentDifficulty = "";

            data.forEach(item => {

                // Mostra il titolo per ogni nuova categoria e difficoltà

                if (item.categoria !== currentCategory || item.difficolta !== currentDifficulty) {

                    const categoryTitle = document.createElement("h3");

                    categoryTitle.textContent = `Categoria: ${item.categoria}, Difficoltà: ${item.difficolta}`;

                    resultDiv.appendChild(categoryTitle);

                    // Aggiorna le variabili per tenere traccia della categoria e difficoltà

                    currentCategory = item.categoria;

                    currentDifficulty = item.difficolta;

                }

                // Aggiungi i record sotto la categoria e difficoltà

                const recordRow = document.createElement("div");

                recordRow.textContent = `Nome: ${item.nome}, Punteggio: ${item.punteggio}`;

                resultDiv.appendChild(recordRow);

            });

        } else {

            resultDiv.textContent = "Nessun risultato trovato.";

        }

    })

    .catch(error => {

        console.error('Errore nella richiesta:', error);

    });

}

Questo invece in base alla categoria e diffcolta:

function mostraDatiSalvati() {

    const categoria = sessionStorage.getItem("catlead");

    const difficolta = sessionStorage.getItem("diflead");

    // Controllo che categoria e difficoltà siano definiti

    if (!categoria || !difficolta) {

        console.error("Errore: categoria o difficoltà mancanti.");

        return;

    }

    // Esegui la richiesta GET

    fetch(`../php/searchfinal.php?categoria=${categoria}&difficolta=${difficolta}`, {

        method: 'GET',

        headers: {

            'Content-Type': 'application/json'

        }

    })

    .then(response => {

        if (!response.ok) {

            throw new Error(`Errore nella risposta: ${response.status}`);

        }

        return response.json(); // Recupera la risposta come JSON

    })

    .then(data => {

        console.log('Dati salvati:', data);

        const container = document.getElementById('datiSalvati');

        container.innerHTML = ''; // Svuota il contenitore

        if (data.length > 0) {

            data.forEach(record => {

                const elemento = document.createElement('div');

                elemento.textContent = `Nome: ${record.nome}, Punteggio: ${record.punteggio}`;

                container.appendChild(elemento);

            });

        } else {

            container.textContent = "Nessun risultato trovato.";

        }

    })

    .catch(error => {

        console.error('Errore nella richiesta:', error);

    });

}

### Salva punteggio

Per salvare il record faccio un POST con php:

 function salvaPunteggio() {

        // Verifica se le variabili sono impostate correttamente

        if (!nickname || !punti || !categoria || !difficolta) {

            alert("Errore: dati incompleti.");

            return;

        }

        // Dati da inviare

        const dati = {

            nome: nickname,

            punteggio: punti,

            categoria: categoria,

            difficolta: difficolta

        };

     // Invia i dati al server tramite fetch

     fetch('../php/salvaQuiz.php', {

        method: 'POST',

        body: formData

    })

    .then(response => response.json())

    .then(data => {

        if (data.status === 'success') {

            alert("Quiz salvato con successo!");

            // Dopo aver mostrato l'alert, reindirizza alla pagina Ziquiz.html

            window.location.href = "Ziquiz.html";

        } else {

            alert("Errore nel salvataggio del quiz: " + data.message);

        }

    })

    .catch(error => {

        console.error('Errore nella connessione con il server:', error);

        alert("Errore nella connessione con il server.");

    });

}

Nel codice prendo i dati e li mando al server con php.

## PHP

### Mandare i dati a php

Al posto del Top score ho deciso di fare un intera classifica.

Per creare la leaderboard con javascript mando i dati del nickname e del punteggio del giocatore e la difficolta e la categoria per sapere in quale leaderboard salvarlo.

Per passare questi dati ho usato un fetch

 function salvaPunteggio() {

        // Verifica se le variabili sono impostate correttamente

        if (!nickname || !punti || !categoria || !difficolta) {

            alert("Errore: dati incompleti.");

            return;

        }

        // Dati da inviare

        const dati = {

            nome: nickname,

            punteggio: punti,

            categoria: categoria,

            difficolta: difficolta

        };

     // Invia i dati al server tramite fetch

     fetch('../php/salvaQuiz.php', {

        method: 'POST',

        body: formData

    })

    .then(response => response.json())

    .then(data => {

        if (data.status === 'success') {

            alert("Quiz salvato con successo!");

            // Dopo aver mostrato l'alert, reindirizza alla pagina Ziquiz.html

            window.location.href = "Ziquiz.html";

        } else {

            alert("Errore nel salvataggio del quiz: " + data.message);

        }

    })

    .catch(error => {

        console.error('Errore nella connessione con il server:', error);

        alert("Errore nella connessione con il server.");

    });

}

## PHP salvataggio record

Questo è il codice php che uso per prendere i nuovi record:

<?php

// Impostazioni per evitare caching

header("Cache-Control: no-store, no-cache, must-revalidate, max-age=0");

header("Pragma: no-cache");

header("Expires: Thu, 01 Jan 1970 00:00:00 GMT");

// Funzione per ottenere il percorso del file JSON

function prendiFilePath ($categoria, $difficolta) {

    return \_\_DIR\_\_ . '/../json/record/' . $categoria . '\_' . $difficolta . '\_dati.json';

}

// Se è una richiesta POST (per aggiungere punteggi)

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

    $data = json\_decode(file\_get\_contents('php://input'), true); // Prende i dati dal corpo della richiesta

    // Controlla se i dati necessari sono presenti

    if ($data && isset($data['nome']) && isset($data['punteggio']) && isset($data['categoria']) && isset($data['difficolta'])) {

        $nome = $data['nome'];

        $punteggio = $data['punteggio'];

        $categoria = $data['categoria'];

        $difficolta = $data['difficolta'];

        // Percorso del file JSON

        $filePath = prendiFilePath($categoria, $difficolta);

        // Crea la directory se non esiste

        $dirPath = dirname($filePath);

        if (!is\_dir($dirPath)) {

            mkdir($dirPath, 0777, true);

        }

        // Crea il file se non esiste

        if (!file\_exists($filePath)) {

            $created = file\_put\_contents($filePath, json\_encode([]));  // Crea un file vuoto

            if ($created === false) {

                echo json\_encode(['status' => 'error', 'message' => 'Errore nella creazione del file']);

                exit;

            }

        }

        // Leggi i dati esistenti

        $datiCorrenti = json\_decode(file\_get\_contents($filePath), true);

        if (!is\_array($datiCorrenti)) {

            $datiCorrenti = [];

        }

        // Aggiungi il nuovo punteggio al file JSON

        $datiCorrenti[] = [

            'nome' => $nome,

            'punteggio' => $punteggio

        ];

        // Salva i dati nel file JSON

        $result = file\_put\_contents($filePath, json\_encode($datiCorrenti, JSON\_PRETTY\_PRINT));

        // Risposta JSON

        if ($result === false) {

            echo json\_encode(['status' => 'error', 'message' => 'Errore nella scrittura del file']);

        } else {

            echo json\_encode(['status' => 'success', 'message' => 'Punteggio salvato correttamente']);

            exit;

        }

    } else {

        // Se i dati sono incompleti

        echo json\_encode(['status' => 'error', 'message' => 'Dati mancanti o non validi']);

    }

}

?>

Prima di tutto elimino le cache per evitare errori, prendo gli elementi che mi servono e che ho passato tramite il post e verifico se sono tutti settati, poi prendo tramite alla funzione prendiFilePath( ) che ritorna lui la path dove salvare i record se non esiste la creo, poi controllo se esiste un file con i record se no lo creo, poi prendo i dati nel file json e inserisco il nuovo record e salvo tutti i dati.

## Prendere i quiz generali

<?php

// Leggi i parametri dalla query string

$categoria = isset($\_GET['categoria']) ? $\_GET['categoria'] : '';

$difficolta = isset($\_GET['difficolta']) ? $\_GET['difficolta'] : '';

if ($categoria && $difficolta) {

    $filePath = \_\_DIR\_\_ . "/../json/record/{$categoria}\_{$difficolta}\_dati.json"; // Costruisci il percorso del file

    // Se il file esiste, carica i dati

    if (file\_exists($filePath)) {

        $datiCorrenti = json\_decode(file\_get\_contents($filePath), true); // Carica i dati dal file JSON

        // Ordina i dati per punteggio in ordine decrescente

        usort($datiCorrenti, function($a, $b) {

            return $b['punteggio'] - $a['punteggio'];

        });

        // Rispondi con i dati in formato JSON

        echo json\_encode($datiCorrenti);

    } else {

        // Rispondi con errore se il file non esiste

        echo json\_encode(['status' => 'error', 'message' => 'File non trovato']);

    }

} else {

    // Se categoria o difficoltà non sono specificati, rispondi con errore

    echo json\_encode(['status' => 'error', 'message' => 'Categoria o difficoltà mancanti']);

}

?>

Per prendere i quiz generali prendo categoria e difficolta e controllo se sono settati poi prendo la path e nel file prendo tutti i record poi li ordino in modo crescente per record e restituisco i json.

### SalvareQuiz

Quando un utente crea un quiz uso questa funzione per salvarlo:

<?php

// Abilitare la visualizzazione degli errori per il debug

ini\_set('display\_errors', 1);

error\_reporting(E\_ALL);

// Verifica se i dati sono stati inviati via POST

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

    // Ottieni i dati dal form

    $nickname = $\_POST['nickname'];

    $nomeQuiz = $\_POST['nomeQuiz'];

    $publico = $\_POST['publico'];

    $difficolta = $\_POST['difficolta'];

    // Verifica la validità del valore di difficoltà

    if (empty($difficolta)) {

        echo json\_encode(['status' => 'error', 'message' => 'La difficoltà non è stata specificata']);

        exit;

    }

    // Ottieni il file caricato

    if (isset($\_FILES['file'])) {

        $file = $\_FILES['file'];

        // Controlla se il file è un JSON

        if (pathinfo($file['name'], PATHINFO\_EXTENSION) != 'json') {

            echo json\_encode(['status' => 'error', 'message' => 'Il file deve essere un JSON']);

            exit;

        }

        // Determina la destinazione e il nome del file

        if ($publico == 'Pubblico') {

            // Salva con il nome "nickname\_nomeQuiz\_difficolta.json" nella cartella json/quizUtenti

            $destination = \_\_DIR\_\_ . '/../json/quizUtenti/' . $nickname . '\_' . $nomeQuiz . '\_' . $difficolta . '.json';

        } else {

            // Crea una nuova cartella per il quiz privato e salva come "nomeQuiz\_difficolta.json"

            $folderPath = \_\_DIR\_\_ . '/../json/quizUtenti/' . $nickname;

            if (!is\_dir($folderPath)) {

                mkdir($folderPath, 0777, true);

            }

            $destination = $folderPath . '/' . $nomeQuiz . '\_' . $difficolta . '.json';

        }

        // Muovi il file caricato nella destinazione corretta

        if (move\_uploaded\_file($file['tmp\_name'], $destination)) {

            echo json\_encode(['status' => 'success', 'message' => 'File caricato correttamente']);

        } else {

            echo json\_encode(['status' => 'error', 'message' => 'Errore nel caricamento del file']);

        }

    } else {

        echo json\_encode(['status' => 'error', 'message' => 'File non selezionato']);

    }

} else {

    echo json\_encode(['status' => 'error', 'message' => 'Metodo non valido']);

}

?>

In questo php creo un file json con il file che mi passa l’utente e lo salvo nel server.

## Login

Per avere un unicita dei record ho deciso di fare un login cosi ogni utente puo vedere i propri record e accedere da piu dispositivi. Per fare ciò ho usato Firebase che mi permetteva di avere un databse con gli utenti e potergli fare resettare la password.

### Firebsae

Firebase è una in possesso da google che permette ai suoi utenti di creare app mobile e web.

È stata fondata da Envolve nel 2011 e nel 20014 acquistata da google. Il primo prodotto che fecero il databases real-time, che è quello che utilizzo per avere i miei utenti salvati.

### Motivo scelta

Ho scelto di utilizzare Firebase perché rende disponibile un database i cuoi potevo salvare gli utenti ed è facilmente usabile dato che ha molte funzionalita integrate come per esempio mettere un recupero della password

### Configurazione

Dopo aver fatto il login con l’account di google su firbase ho collegato il mio progetto per farlo ho dovuto mettere questa configurazione in un file js nel mio applicativo:

// Firebase configuration

const firebaseConfig = {

  apiKey: "AIzaSyBVAPvEBg-9KnKMgrE5cFqxZrQiaCIlGsU",

  authDomain: "ziquiz-cb376.firebaseapp.com",

  databaseURL:"https://ziquiz-cb376-default-rtdb.europe-west1.firebasedatabase.app,

  projectId: "ziquiz-cb376",

  storageBucket: "ziquiz-cb376.firebasestorage.app",

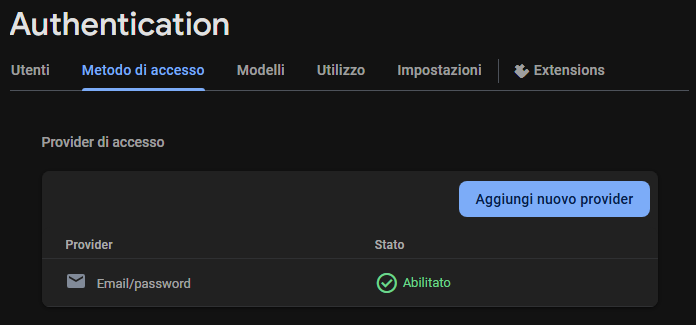
  messagingSenderId: "971536143420",

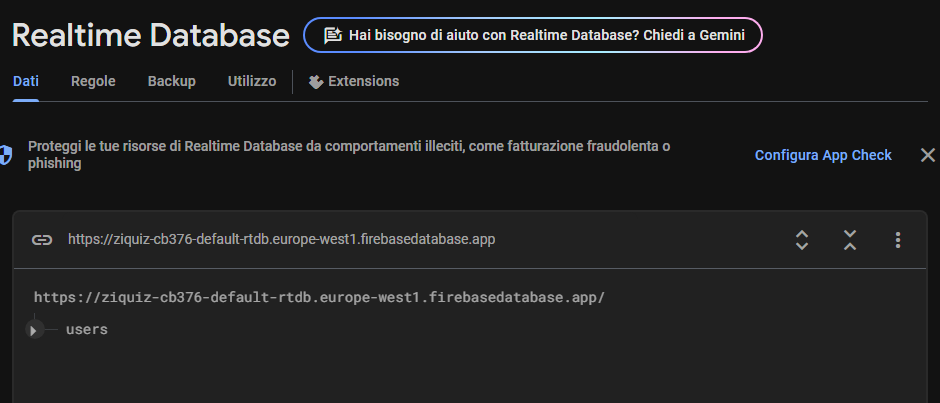
  appId: "1:971536143420:web:9534385c390bc939339f39",

  measurementId: "G-G3W2P5JLVZ"

};

poi ho dovuto creare un database Realtime e configurare come metodo di accesso Email/password





## Configurazione Autenticazione

Dopo aver creato le pagine html per login registrazione e reset password ho dovuto scrivere il codice in javascript che con firebase per farlo bisogna usare SDK che ho dovuto importare nelle configurazioni

import { getAuth, onAuthStateChanged, signOut } from "https://www.gstatic.com/firebasejs/9.6.8/firebase-auth.js";

il codice nel file js autenticazione che per richiamarlo negli altri file dato che si usa il modulo SDK bisogna scrivere questa riga di comando nel header:

  <script type="module" src="../js/autenticazione.js"></script>

Una volta messo quando i submit dei form nel html verrano cliccati farà partire le dovute azioni. Il codice autenticazione è il seguente:

//Importazione Librerie firebase

import { auth, db } from './firebaseConfig.js';

import { signInWithEmailAndPassword, createUserWithEmailAndPassword, signOut, onAuthStateChanged } from "https://www.gstatic.com/firebasejs/9.6.8/firebase-auth.js";

import { get, ref, set, child } from "https://www.gstatic.com/firebasejs/9.6.8/firebase-database.js";

All’inizio ho importato delle librerie di firebase cioè, auth e db rappresentano il servizio di qutenticazione e il db.

signInWithEmailAndPassword, createUserWithEmailAndPassword, signOut e onAuthStateChanged sono metodi che importo direttamente da firebase perché mi servono per interagire con il db; Invece get, ref, set e child sono funzioni che mi servono per interagire con il db.

//Gestione dei form

const loginForm = document.getElementById("login-form");

const signupForm = document.getElementById("signup-form");

if (loginForm) {

    loginForm.addEventListener("submit", logIn);

}

if (signupForm) {

    signupForm.addEventListener("submit", signUp);

}

Gestisce i Form e in base al form usato fa partire una funzione.

//funzione accesso

async function logIn(e) {

    e.preventDefault();

    const email = document.getElementById("login-identifier").value;

    const password = document.getElementById("password").value;

    console.log("Login in corso...");

    try {

        const userCredential = await signInWithEmailAndPassword(auth, email, password);

        const user = userCredential.user;

        // Recupera il nome utente dal database Firebase

        const userRef = ref(db, 'users/' + user.uid);

        const userSnapshot = await get(userRef);

        const userData = userSnapshot.val();

        if (userData) {

            const username = userData.username;

            sessionStorage.setItem("1", username);

            // Puoi anche visualizzare il nome utente sulla pagina, ad esempio:

        }

        console.log("Login funzionato!");

        window.location.href = "../html/Ziquiz.html"; // Redirige alla pagina principale

    } catch (error) {

        console.error("Errore:", error.message);

        document.getElementById("error-message").textContent = "Errore nel login, inserisci le credenziali corrette.";

    }

}

La funzione per il login fa il login con un metodo di firebase che ho importato salva l’username associato all’utente per farlo devo usare le funzioni importate che mi permettono di trovare nel db l’utente e prendere il suo username, dopo averlo preso lo salvo nelle sessioni.

Se funziona lo rindirizzo alla home altrimenti mostro gli errori.

//Funzione Registrazione

async function signUp(e) {

    e.preventDefault();

const username = document.getElementById("signup-username").value;

    const email = document.getElementById("signup-email").value;

    const password = document.getElementById("signup-password").value;

    try {

        // Controlla se il nome utente esiste già nel database

        const usernameExists = await checkUsernameExist(username);

        if (usernameExists) {

            document.getElementById("error-message").textContent = "Nome utente già esistente, prova con un altro.";

            return;  // Non proseguire con la registrazione se il nome utente esiste già

        }

        // Crea un nuovo utente con email e password

        const userCredential = await createUserWithEmailAndPassword(auth, email, password);

        // Ottieni l'oggetto dell'utente appena creato

        const user = userCredential.user;

        sessionStorage.setItem("1", username);

        // Autentica l'utente prima di scrivere nel database

        await set(ref(db, "users/" + user.uid), {

            username,

            email

        });

        // Redirige l'utente alla pagina principale dopo la registrazione

        window.location.href = "../html/Ziquiz.html";

    } catch (error) {

        console.error("Signup error:", error.message);

        document.getElementById("error-message").textContent = "Errore nella registrazione: " + error.message;

    }

}

La funzione di registrazione oltre a fare le cose precedenti salva l’utente creato nel db ma prima di farlo controlla l’unicità con questa funzione:

// Funzione per controllare se un nome utente esiste già nel database

async function checkUsernameExist(username) {

    const usersRef = ref(db, 'users/');

    const snapshot = await get(usersRef);

    const data = snapshot.val();

    // Controlla se qualche utente ha lo stesso nome utente

    for (const key in data) {

        if (data[key].username === username) {

            return true;  // Il nome utente esiste già

        }

    }

    return false;  // Il nome utente non esiste

}

In questa funzione prendo tutti gli utenti e li ciclo per vedere se il loro username è uguale a quello nuovo e nel caso restituisco false.

//Funzione per il logout

const logoutButton = document.getElementById("logoutButton");

if (logoutButton) {

    logoutButton.addEventListener("click", async () => {

        try {

            await signOut(auth);  // Effettua il logout

            sessionStorage.setItem("1", "");

            window.location.href = "../html/accesso.html";  // Redirige alla pagina di login

        } catch (error) {

            console.error("Errore nel logout:", error.message);

        }

    });

}

Per fare il logout appena il bottone viene cliccato faccio il logout con una funzione di firebase e tolgo il nickname e reindirizzo l’utente al login.

import { getAuth, sendPasswordResetEmail } from

"https://www.gstatic.com/firebasejs/9.6.8/firebase-auth.js";

// Aggiungi un ascoltatore per il modulo di reset della password

const resetPasswordForm = document.getElementById("reset-password-form");

if (resetPasswordForm) {

    resetPasswordForm.addEventListener("submit", resetPassword);

}

Per il reset della password importo le funzioni già implementate da firebase e aggiungo il controllore per il form.

//Funzione del reset della password

async function resetPassword(e) {

    e.preventDefault();  // Previeni l'invio del form

    const email = document.getElementById("reset-email").value;

    const auth = getAuth();

    try {

        // Invia la richiesta di reset della password

        await sendPasswordResetEmail(auth, email);

        document.getElementById("reset-message").textContent = "Controlla la tua email per resettare la password!";

        document.getElementById("reset-message").style.color = "green"; // Messaggio di successo

    } catch (error) {

        console.error("Errore nel reset della password:", error.message);

        document.getElementById("reset-message").textContent = "Errore nell'invio dell'email di reset: " + error.message;

        document.getElementById("reset-message").style.color = "red"; // Messaggio di errore

    }

}

Nel codice per resettare la password invio l’email e gli dico di controllarla, il resto lo farà firebase che si occupa di inviare l’email e cambiare la password.

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-001  REQ-001 | **Nome** | Verifica funzionalità Menu |
| **Descrizione** | Si controlla se il menu sia utilizzabile non facendo un login | | |
| **Prerequisiti** | - | | |
| **Procedura** | 1. Accedere al sito 2. Premere bottone gioca   3.tornare indietro | | |
| **Risultati attesi** | Il bottone del menu funziona e ti fa andare avanti però ti avverte che i dati non verranno salvati e poi si può tornare indietro. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-002  REQ-001 | **Nome** | Verifica funzionalità Registrazione con login |
| **Descrizione** | Si controlla se il nickname si imposta . | | |
| **Prerequisiti** | - | | |
| **Procedura** | 1. Accedere al sito 2. Premere il bottone accedi e cliccare su registrati e riuscire a registrarsi. | | |
| **Risultati attesi** | Riuscirò a registrarmi e il nickname verrà salvato. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-003  REQ-002 | **Nome** | Verifica funzionalità scelta categorie |
| **Descrizione** | Si controlla se è possibile scegliere tra le varie categorie | | |
| **Prerequisiti** | - | | |
| **Procedura** | 1. Accedere al sito 2. Premere gioca 3. Premere su una categoria 4. Tornare indietro e premerne un altra | | |
| **Risultati attesi** | L’utente potrà scegliere varie categorie e tornare indietro quando vuole | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-004  REQ-004 | **Nome** | Verifica funzionalità scelta difficoltà |
| **Descrizione** | Si controlla se è possibile scegliere tra le varie categorie | | |
| **Prerequisiti** | - | | |
| **Procedura** | 1. Accedere al sito 2. Premere gioca 3. Premere su una categoria 4. Scegliere la difficoltà 5. Tornare indietro | | |
| **Risultati attesi** | L’utente potrà scegliere varie difficoltà e tornare indietro al menù | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-005  REQ-006 | **Nome** | Verifica corezzione domanda |
| **Descrizione** | Si controlla se si visualizza la corezzione | | |
| **Prerequisiti** | - | | |
| **Procedura** | 1. Accedere al sito 2. Premere gioca 3. Premere su una categoria 4. Scegliere una difficoltà facile 5. Sbagliare apposta una domanda | | |
| **Risultati attesi** | Si dovrà visualizzare l’errore | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-006  REQ-005 | **Nome** | Verifica funzionalità top score |
| **Descrizione** | Si controlla se il top score viene salvato | | |
| **Prerequisiti** | - | | |
| **Procedura** | 1. Accedere al sito 2. Premere gioca 3. Premere su una categoria 4. Scegliere una difficoltà 5. Finire il quiz 6. Tornare alla pagina dela scelta delle dificoltà | | |
| **Risultati attesi** | Ci dovrà essere salvato il top score | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-007  REQ-003 | **Nome** | Verifica creazione di quiz |
| **Descrizione** | Si controlla se il top score viene salvato | | |
| **Prerequisiti** | Essere loggato | | |
| **Procedura** | 1. Accedere al sito 2. Premere Crea 3. Creare un quiz 4. Tornare alla home 5. Cliccare gioca 6. Visualizzare le categorie | | |
| **Risultati attesi** | Ci dovrà essere la categoria appena creata | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-008  REQ-006 | **Nome** | Verifica modalità studio |
| **Descrizione** | Si controlla se il top score viene salvato | | |
| **Prerequisiti** | Essere loggato | | |
| **Procedura** | 1. Accedere al sito 2. Premere Studia 3. Inizare la modalità | | |
| **Risultati attesi** | Si potrà studiare su un quiz creato | | |

## Risultati test

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Risultato ottenuto** | **Stato** |
| TC-001 | Il menu funziona si riesce ad accedere e tutti i bottoni vanno e quando clicco gioca mi avverte che non sono registrato e non salverà i record | Passato |
| TC-002 | Riesco a registrarmi con un nuovo utente e appena mi registro mi reindirizza al menu impostando il mio nickname. | Passato |
| TC-003 | Si riesce a cambiare categoria facilmente    Come si vede la categoria funziona e le domande sono diverse per categoria. | Passato |
| TC-004 | Anche il test 4 funziona e si può vedere che in base alla difficolta cambiano il numero di domande nel quiz | Passato |
| TC-005 | Si vede chiaramente che la domanda è corretta e sbagliata sia dal aumentare del punteggio e sia dal cambio dello sfondo | Passato |
| TC-006 | Il top score viene visualizzato correttamente | Passato |
| TC-007 | I quiz creati si vedono corretamente | Passato |
| TC-008 | Non è passato perché non ho avuto abbastanza tempo per implementarlo | Non passato |

## 

## Mancanze/limitazioni conosciute

Ci sono vari problemi la prima è la mancanza di alcune funzionalità dei requisiti di livello basso come la modalità studio o i suoni di sfondo ciò è dovuto dal fatto che non ho avuto abbastanza tempo ma sicuramente se il progetto avrà un futuro queste saranno le prime cose da implementare.

Un’altra mancanza è quella che non ho avuto tempo per trovare un modo per bloccare i file json nel server e quindi si potrebbe vedere la path e trovare le soluzioni ai quiz.

# Consuntivo

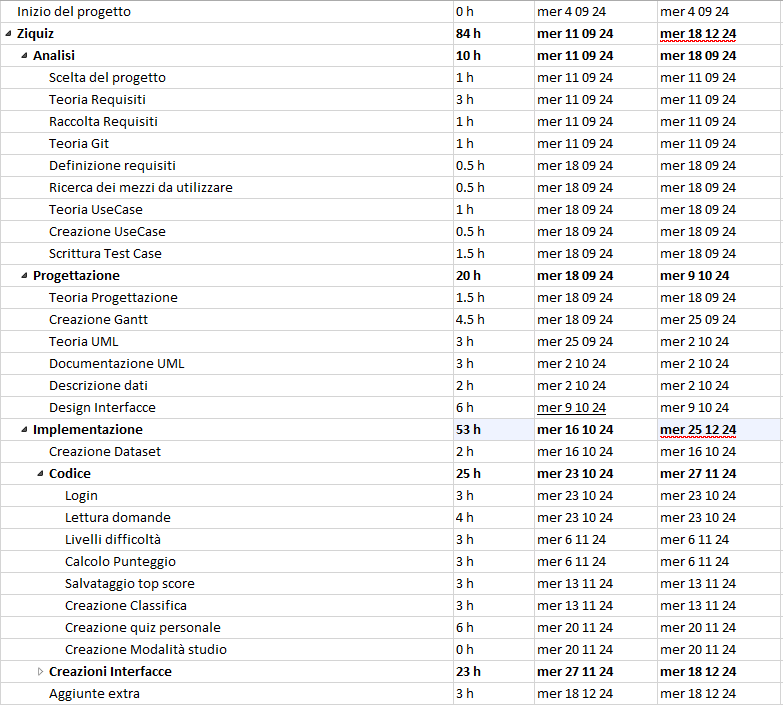
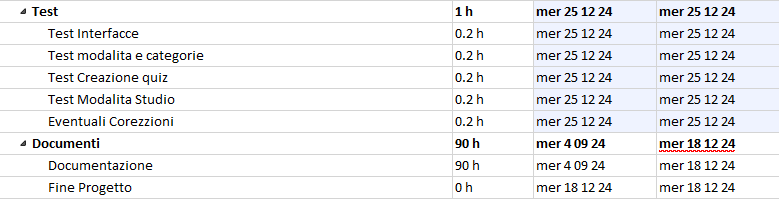
 

Figura 7 Gantt consuntivo

## Descrizione Gantt Consuntivo

### Analisi

Il Primo Giorno di lavoro non c’ero quindi ho perso un po’ di tempo e di fatti nel analisi ci ho messo solo 10h facendo un po’ più veloce la scelta dei requisiti.

### Progettazione

Nella progettazione ci ho messo 2 ore in più da quelle pianificate perché ho avuto difficoltà a creare i mockup.

### Implementazione

Per la progettazione ho impiegato 3 ore in più ma comunque non sono riuscito a fare alcune funzioni che mi ero prefissato, tutto ciò è dovuto dal fatto che ho deciso di fare un login per salvare gli utenti.

### Test

Non avendo più tempo i Test li ho fatti in modo rapido ma sono comunque riuscito a farli.

### Documentazione

Per la documentazione sono sempre rimasto un po’ indietro perché preferivo continuare l’implementazione alla documentazione quindi dal mio punto di vista non sono riuscito a farla al meglio.

# Conclusioni

Le mie conclusioni sono che l’applicativo web che ho creato è molto divertente anche se ne esistono simili e si può ancora migliorare spero che verrà usato dagli utenti per divertirsi e battere ogni record.

## Sviluppi futuri

Ci sono molti possibili sviluppi futuri e miglioramenti come ad esempio finire le cose che non sono riuscito a implementare come la possibilità di studio, migliorare la creazione di studio e mettere animazioni e suoni per rendere l’esperienza di gioco più interattiva.

Oltre a ciò si potrebbero mettere nuove modalità e funzionalità come quiz con le immagini o difficoltà a tempo.

## Considerazioni personali

A me creare questo applicativo ha insegnato molte nuove cose ad esempio ho scoperto firebase, e altre funzioni che non conoscevo e sono migliorato con js e php, e credo di aver imparato che devo migliorare la gestione del tempo.

# Bibliografia

## Sitografia

* [**https://www.w3schools.com/**](https://www.w3schools.com/) *w3school*  consultato in più date
* [W3school json](https://www.w3schools.com/whatis/whatis_json.asp) *w3school json* consultato il 23.10.’24
* [link fetch json](https://www.geeksforgeeks.org/read-json-file-using-javascript/) *Fetch con json* consultato il 23.10.’24
* [Link per funzione di shuffle](https://javascript.info/task/shuffle#:~:text=sort%20(%20(%20)%20%3D%3E%20Math,sorting%20function%20reorders%20elements%20randomly.) *javascript*.*info* consultato il 6.11.2024
* [**https://chatgpt.com/**](https://chatgpt.com/) *chatgpt* consultato in più date
* <https://www.php.net/> *php* consultato in più date
* <https://firebase.google.com/> *firebase* consultato in più date
* [link youtube per firebase](https://www.youtube.com/watch?v=nVhPqPpgndM) *youtube* consultato l’ 11.12.2024

Glossario

|  |  |
| --- | --- |
| **Termine** | **Significato** |
| PHP | PHP è un linguaggio di programmazione lato server usato per creare siti web dinamici, gestire database e sessioni utente. È facilmente integrabile con HTML ed è open-source. |
| Javascript | JavaScript è un linguaggio di programmazione utilizzato per creare interattività nelle pagine web. Consente di manipolare il contenuto, gestire eventi e comunicare con i server senza ricaricare la pagina. |
| Html | HTML (Hypertext Markup Language) è il linguaggio di markup utilizzato per strutturare e presentare contenuti web. Definisce la struttura delle pagine web tramite elementi come testo, immagini, link e tabelle. |
| Css | CSS (Cascading Style Sheets) è un linguaggio utilizzato per definire lo stile e l'aspetto visivo delle pagine web. Permette di controllare layout, colori, font e altre proprietà estetiche separando la struttura (HTML) dalla presentazione |
| Firebase | Firebase è una piattaforma di sviluppo di applicazioni web e mobili offerta da Google. Fornisce servizi come database in tempo reale, autenticazione, hosting, analytics e notifiche push, permettendo di creare e gestire applicazioni in modo semplice e scalabile. |

# Indice delle figure

[Figura 1 use case 8](#_Toc185426631)

[Figura 2 Esempio di diagramma di Gantt 9](#_Toc185426632)

[Figura 3 Diagramma di classi 12](file:///D:\Scuola\I3\M306\Ziquiz\3_Documentazione\Documentazione_Ziquiz.docx#_Toc185426633)

[Figura 4 Interfaccia menù 13](file:///D:\Scuola\I3\M306\Ziquiz\3_Documentazione\Documentazione_Ziquiz.docx#_Toc185426634)

[Figura 5 Categoria 14](#_Toc185426635)

[Figura 6 Difficoltà 14](#_Toc185426636)

[Figura 7 Esempio domanda 15](#_Toc185426637)

[Figura 8 Esempio errore 15](#_Toc185426638)

[Figura 9 Esempio domanda 2 16](#_Toc185426639)

[Figura 10 Esempio correzione 2 16](#_Toc185426640)

[Figura 11 Uscita da quiz 17](#_Toc185426641)

[Figura 12 Modalità Crea 17](#_Toc185426642)

[Figura 13 Domande modalità crea 18](#_Toc185426643)

[Figura 14 Diagramma Swimlanes base 19](#_Toc185426644)

[Figura 15 Diagramma Swimlanes (studio + creaquiz) 20](#_Toc185426645)

[Figura 16 Home 23](#_Toc185426646)

[Figura 17 Back button 26](#_Toc185426647)

# Allegati

Elenco degli allegati, esempio:

* Diari di lavoro
* Codici sorgente
* Mandato
* Prodotto
* Abstract
* Activity Diagrams
* Use Case
* QdC