ESAME DI STATO 2021 – DOCUMENTAZIONE E MANUALE OPERATIVO DELLA SOLUZIONE PROPOSTA

Elaborato di maturità **Progettista:** Andrea Galliano	**Scheda n**. 1 **Data**: 23/05/2021

Titolo del progetto:
Triathlon a distanza per ipovedenti (*Olimpiadi Online*)

Obiettivi:
Proporre una soluzione che garantisca il raggiungimento dei seguenti propositi:

- Dare le stesse possibilità che ha la maggior parte della popolazione di svolgere attività sportive ad un soggetto ipovedente;
- Consentire un'interconnessione a distanza efficace per le scuole partecipanti;
- Realizzare una base di dati che tenga traccia degli aspetti prioritari della competizione;
- Rendere possibile, dalla rete, la visione dei risultati delle gare;
- Garantire l'integrità dei dati trasmessi.

Nota: l'intero progetto è stato sviluppato coerentemente con l'***Agenda 2030***, un programma d'azione per le persone che ha lo scopo di migliorare la qualità di vita dell'intero pianeta. Al suo interno possiamo trovare infatti diverse proposte di miglioramento da raggiungere entro l'anno 2030.
Il triathlon a distanza permette dunque di raggiungere parte dell'**obiettivo n.4**: fornire un'educazione di qualità, equa ed inclusiva, e opportunità di apprendimento per tutti.

Descrizione:
Il progetto nasce, in piena emergenza COVID-19 e con le lezioni in presenza sospese, dal bisogno dell'Ufficio Scolastico Regionale (USR) di mettere a disposizione dei propri alunni non-vedenti un Triathlon a distanza (gareggiando tra scuole), che consenta di competere nelle seguenti discipline:

1. Gara di velocità (100 metri);
2. Corsa a ostacoli;
3. Tiro in porta (calcio di rigore).

Soluzione proposta:
La soluzione proposta prevede lo sviluppo di un'applicazione per smartphone, la cui fattibilità è testata grazie ad un prototipo (il sistema operativo nel quale è stato sviluppato il software è **Android**, grazie all'ambiente di **Unity**, che permette la creazione di codici importabili su vari device).
Questa unica app, però, non basterebbe a dare credibilità totale al progetto ed è necessario capire quali sono le interfacce/software aggiuntivi che danno completezza all'intera soluzione: è stato infatti previsto di sviluppare una vera e propria simulazione delle varie reti delle scuole e dei vari uffici che erogano servizi di connessione in **CISCO Packet Tracer** (anche per dare alla proposta un'impronta sistemistica e non solo informatica).
È stato poi pensato di programmare un **sito Web** che, in rete, ha lo scopo di registrare un utente o farlo accedere con le proprie **credenziali** per poi dargli una chiara visione delle classifiche prodotte dalla competizione alla quale ha partecipato. Per far sì che tutto questo sia possibile realmente, nasce anche il bisogno di sviluppare un **database**, che, di conseguenza, terrà traccia di tutti i dati necessari per il perfetto funzionamento dell'intero sistema (la realizzazione totale della base di dati è, come da prassi, suddivisa nelle canoniche 3 progettazioni: quella concettuale, quella logica e quella fisica).

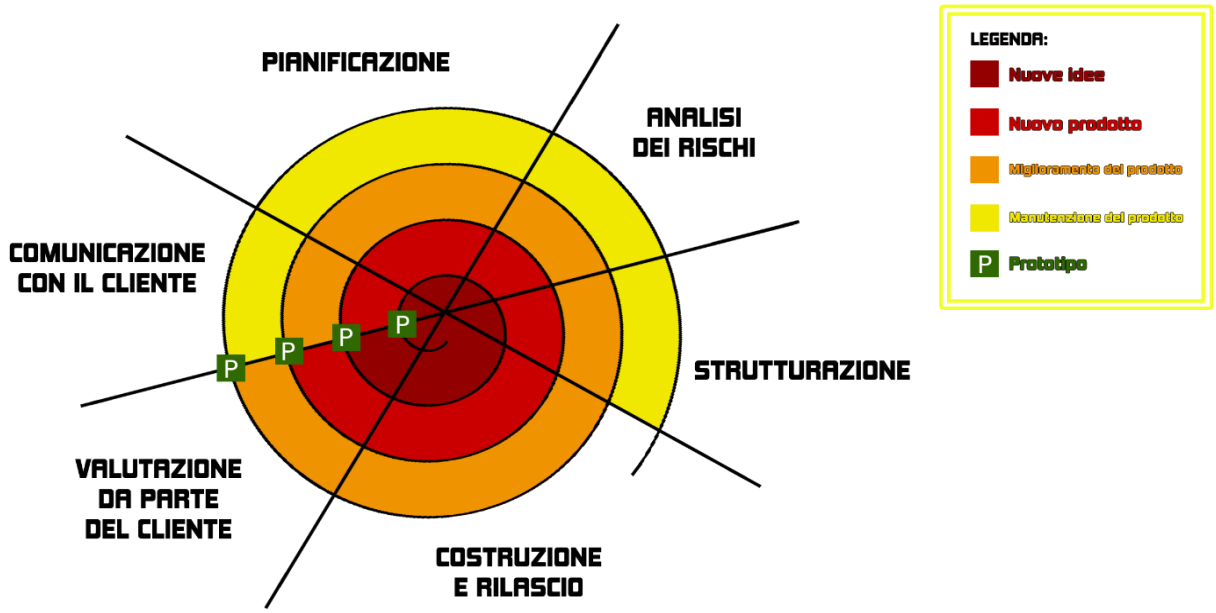
È stato infine deciso di adottare 2 Standard ISO per valutare i vari programmi:

- **ISO 12207** → Viene definito il ciclo di vita del software sviluppato;
- **ISO 9126** → Viene valutata la qualità del software.

Fasi del progetto e tempistica (diagramma allegato):
-Si è optato per dividere lo sviluppo del progetto in fasi delineate e ben precise. Questo per dare un'idea chiara di cosa fare e di come portare avanti la soluzione proposta.

*-Per il compimento di progetti complessi è buona prassi individuare il modello adatto a dividere in fasi il proprio lavoro; per questo motivo è stato scelto un **modello a spirale**, poiché consente, al termine di ogni suo ciclo, la verifica delle soluzioni grazie alla realizzazione di un prototipo (testabile sia dallo sviluppatore che dal cliente).*

Allegato del modello a spirale:

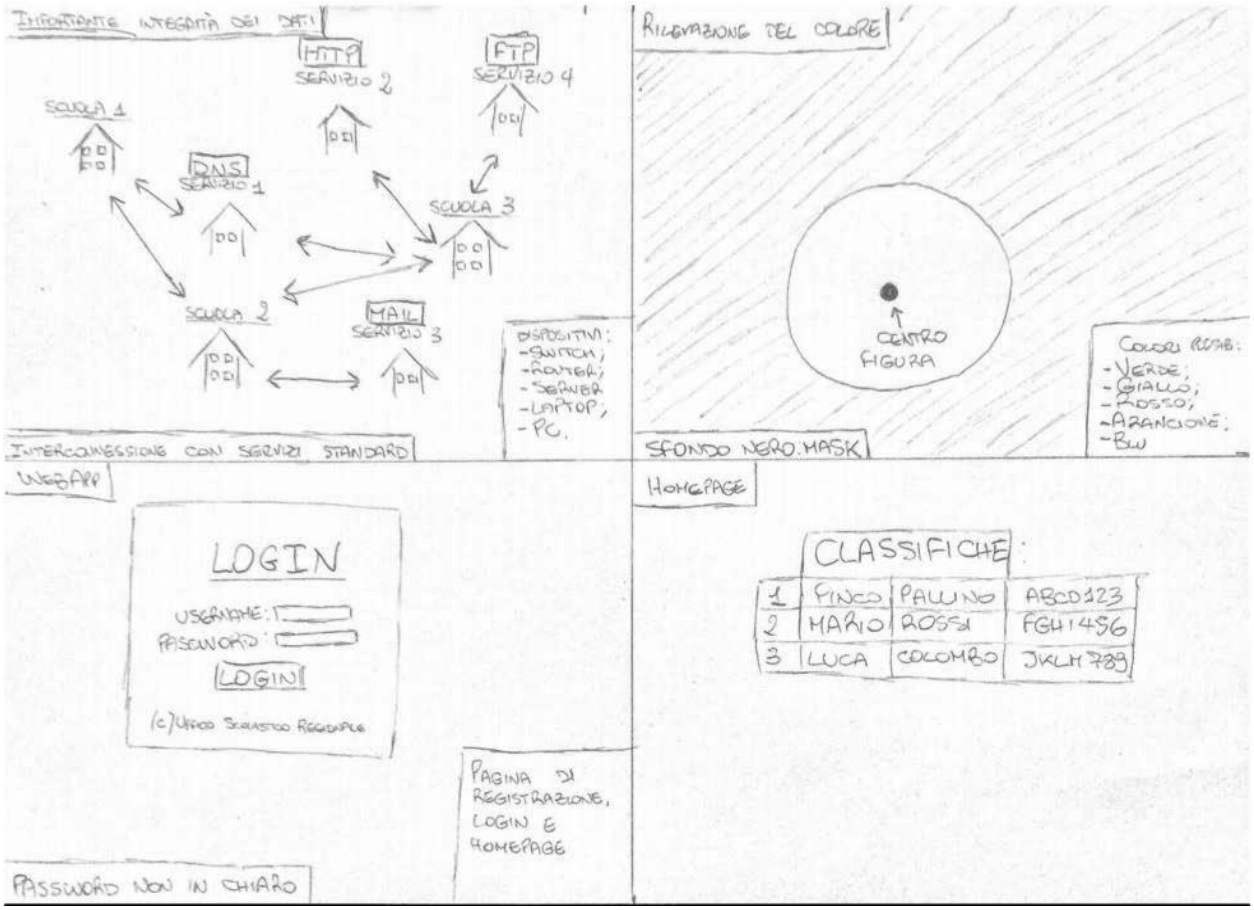


-Fasi del progetto:

1. **Comunicazione con il cliente:** nella primissima fase dello sviluppo di un qualsiasi progetto è essenziale una buona comunicazione con il cliente. L'**Unione Italiana dei Ciechi e degli Ipovedenti ONLUS** ha giocato infatti un ruolo fondamentale nel capire come operare in maniera corretta. Le richieste dell'utente erano dunque chiare a priori ed è stato possibile capire quale fosse il punto centrale dal quale sviluppare tutto: l'**uso dell'udito**.
   Dopo aver raccolto queste prime informazioni è stato utile fare un vero e proprio *brain storming* su ciò che andasse implementato.
   -Allegato del logo della società:

Unione Italiana dei Ciechi e degli Ipovedenti ONLUS - APS
Sezione Territoriale di Milano

-Allegato del disegno di una bozza del primo prototipo da presentare:



- [Primo disegno] → Bozza di come organizzare l'interfaccia dell'interconnessione delle scuole.
- [Secondo disegno] → Bozza sul riconoscimento del colore della palla + calcolo del centro figura.
- [Terzo disegno] → Bozza sulla pagina di login del sito web (prestando attenzione al non trasmettere in chiaro le varie password).
- [Quarto disegno] → Pagina principale del sito con le relative classifiche.

2. **Pianificazione**: Dopo aver dato libero spazio alle idee, la fase successiva di pianificazione riguarda tutto ciò che concerne i linguaggi di programmazione ed i linguaggi di mark-up adatti al caso preso in esame.
   Linguaggi di programmazione:
   - **Python** → Utilizzato per effettuare il riconoscimento del colore da fotocamera, la rilevazione dei contorni dell'oggetto, il calcolo del centro di quest'ultimo e l'invio dei dati all'applicazione Android. Per scrivere il programma che ha portato al completo funzionamento di questa porzione di lavoro è stato necessario scaricare una libreria software multipiattaforma (insieme di codici importabili tramite un download dalla rete) essenziale: **OpenCV**, acronimo di *Open Source Computer Vision Library*.
   - **C#** → Adottato per la scrittura di codici script in **Unity**. Permette di creare un ambiente virtuale all'interno del quale è possibile sviluppare un vero e proprio contesto videoludico. Nel nostro caso è stato adottato per dare al partecipante la possibilità di muoversi all'interno dello spazio in cui si svolge la disciplina, fargli capire quando un oggetto è vicino (il pallone o l'ostacolo) grazie all'ausilio di un suono ed è inoltre stato il linguaggio adatto a creare un codice che ricevesse i dati inviati dall'algoritmo in python.
   - **PHP** → Utilizzato per l'implementazione di algoritmi di ***back-end*** riguardanti il sito web.

- **JavaScript** → Implementato per la gestione delle **richieste asincrone http** (sempre utilizzato quando si clicca su un qualsiasi bottone del sito). Necessaria l'importazione di **Ajax** dalla pagina iniziale html per garantirne il perfetto funzionamento.
- **SQL** → Linguaggio che consente di creare su **workbench** ("*piano di lavoro*") il database, di effettuare le interrogazioni (**query**) necessarie e di creare eventuali utenti dandogli i dovuti **permessi** e **privilegi**.

Linguaggi di mark-up:

- **HTML** → Linguaggio utilizzato per creare la base di ***front-end*** del sito (utilizzo di tag).
- **CSS** → Linguaggio adoperato per la cura dell'estetica del sito (fa sempre parte della sezione dei linguaggi di ***front-end***).

3. **Analisi dei rischi**: uno degli obiettivi primari da rispettare per assicurare la buona riuscita del prodotto è quello di andare a valutare i potenziali rischi e le minacce che potrebbero intaccare la sicurezza del progetto. Particolare attenzione sarà posta infatti sul **preservare l'accessibilità e l'integrità dei dati** trasmessi in rete (prioritario l'aspetto riguardante la **privacy**) e le soluzioni apportate, all'interno del sito web, per scongiurare qualsiasi tipo di attacco sono le seguenti:
   - ***Prepared Statements*** → Interrogazione al DB ***parametrizzata*** all'interno della quale è possibile dividere il corpo della query dai parametri che vengono passati dall'utente. Scopo: prevenire un attacco di tipo **SQL Injection**.

   -Allegato screenshot di un codice del sito web implementato con questa tecnica:

scriptLoginStud.php

```php
<?php
  $ris = 0;
  $idConnessione = mysqli_connect("localhost", "root", "", "Olimpiadi_Online")
  or die("Connessione fallita con il database: ".mysqli_connect_error());

  $query = "select password from studenti where id = ?";
  $stmt = mysqli_stmt_init($idConnessione);
  if(mysqli_stmt_prepare($stmt, $query)){
    mysqli_stmt_bind_param($stmt, "s", $_POST["nomeUtente"]);
    if(! mysqli_stmt_execute($stmt)){
      $error = mysqli_stmt_error($stmt);
    }
    $ris = mysqli_fetch_assoc(mysqli_stmt_get_result($stmt));
    //echo json_encode($ris);
    if(password_verify($_POST["password"], $ris["password"])){
      echo "OK";
    }
    else{
      echo "FAILED";
    }
    mysqli_stmt_close($stmt);
  }
  else{
    echo mysqli_error($idConnessione);
  }
  mysqli_close($idConnessione);
?>
```

   - ***Digest*** → Di fondamentale importanza, nel momento in cui si vanno a memorizzare dati sensibili all'interno di una base di dati come le password, è **NON** salvarle in chiaro per evitare che utenti malintenzionati possano averne visione. In questo caso, la soluzione proposta è quella di avvalersi degli **algoritmi di *Hash*,** un insieme di procedure di matematica complessa che permettono di convertire una password in una stringa di caratteri crittati, che prende dunque il nome di *digest*. Ciò che rende particolarmente efficaci i digest è l'essere **irreversibili**: una volta generato, non si può tornare alla stringa di partenza.

-Allegato di una password di prova convertita:

password_hash(ProvaHash , PASSWORD_DEFAULT)
= $2y$10$vFuJebFyIsVO..ONSvCPC.1MQtnueJCQV0CtWuSAMSTNXA/GG8eSy

- Validazione dell'**input** (*lato client*) → Un'altra caratteristica dell'applicazione Web è il controllo di ciò che l'utente inserisce nei campi di input (aree di testo all'interno delle quali potrebbe essere richiesto l'inserimento di dati personali come il nome utente, la password, il nome, il cognome, il sesso…). Questi controlli sono molto semplici, ma consentono comunque di avere in ingresso dati in un formato corretto rispetto a come deve essere memorizzato nella base di dati.

-Allegato di alcuni controlli fatti all'interno della pagina di registrazione per lo studente:

registratiStudente.php

```
<body>
  <div id="main3">
    <h2>Inserisci le credenziali personali per la registrazione:</h2>
    <br>
    <form method="post">
      <h4>Username:</h4>
      <input type="text" placeholder="Inserisci username" id="user" required>
      <br>
      <br>
      <h4>Password:</h4>
      <input type="password" placeholder="Inserisci la password" id="pass" required>
      <br>
      <br>
      <h4>Cognome:</h4>
      <input type="text" placeholder="Inserisci il cognome" id="cognome" required>
      <br>
      <br>
      <h4>Nome:</h4>
      <input type="text" placeholder="Inserisci il nome" id="nome" required>
      <br>
      <br>
      <h4>Sesso:</h4>
      <select name="sesso" id="sesso">
        <option value="maschio">Maschio</option>
        <option value="femmina">Femmina</option>
        <option value="non specificato">Non specificato</option>
      </select>
      <br>
```

I controlli sulla validazione dell'input, all'interno di questo codice, si possono notare grazie a tre elementi all'interno dei **tag** input:

- *Type* → Specifica il **tipo di dato** che deve essere mandato in ingresso ("text", "password" o "*select*");
- *Placeholder* → Dà una chiara idea all'utente di **cosa richiede** il sito ("Inserisci il nome", "Inserisci il cognome");
- *Required* → Specifica che il dato in input è richiesto ed **obbligatorio**.

- Creazione virtuale di una **VPN** → Nell'andare a simulare l'interconnessione tra le varie scuole è possibile progettare, all'interno dell'ambiente di Packet Tracer, una *Virtual*

Private Network; il fine è quello di realizzare una vera e propria rete privata virtuale che garantisca **anonimato** (per una questione di privacy) e **sicurezza maggiore dei dati** trasmessi. Ciò è possibile grazie all'ausilio di un **canale di comunicazione riservato** tra i diversi dispositivi in rete.
NOTA: Nella fase di "*strutturazione*" del progetto sarà possibile visionare in chiaro come è avvenuta la realizzazione di questa VPN.

4. **Strutturazione**: la strutturazione del progetto è divisa nelle seguenti macro-fasi:
   - Studio della **realtà di interesse**;
   - Progettazione **concettuale**, **logica** e **fisica** del database;
   - Creazione della **Web App**;
   - Creazione dell'**interfaccia di comunicazione** tra le scuole;
   - Creazione dell'**applicazione per smartphone** all'interno dell'ambiente *Unity*.

*Studio della **realtà d'interesse**:*

Analisi del contesto che ha lo scopo di iniziare la progettazione del database relazionale coerentemente con ciò che è descritto all'interno della traccia.

Per uno studio corretto della realtà di interesse, ci si pone le seguenti domande: qual è il contesto nel quale mi trovo? Quali sono gli utenti (*target*) ai quali è rivolto il progetto? Come dovrà utilizzare il database l'utente? Cosa dovrà vedere affinché venga implementato un servizio valido?
Occorre dunque un'analisi corretta dei requisiti: individuare correttamente lo scopo del sistema e le funzionalità che deve fornire.

Per capire quali sono le entità necessarie per la creazione del database si prova a rispondere alle domande precedenti: il contesto nel quale ci si trova è una gara virtuale di Triathlon, in cui le **scuole** che vi partecipano sono interconnesse tra di loro e possono visualizzare i vari risultati grazie ad una applicazione web con la quale è possibile interfacciarsi. Deve inoltre essere possibile reperire, oltre ai dati delle scuole, anche i dati del singolo **studente**. Chiaramente sarà possibile avere a disposizione i dati sul **Triathlon** (senza trascurare in alcun modo le discipline che esso prevede: **gara di velocità**, **corsa a ostacoli** e **calcio di rigore**) con le relative **classifiche**.
Alla luce di questa prima analisi è già possibile capire quali saranno le entità dello schema concettuale:

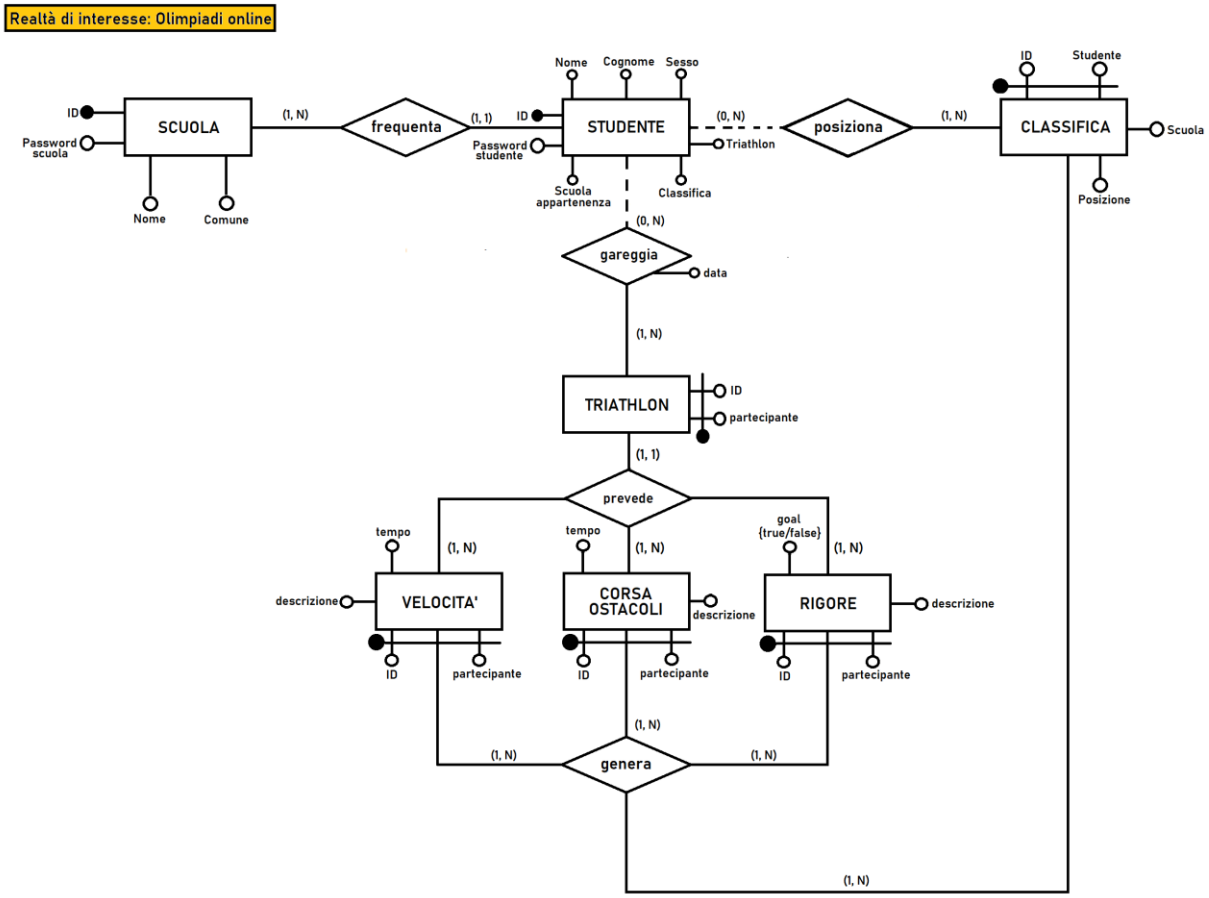
- Scuola;
- Studente;
- Triathlon;
- Velocità;
- Corsa ostacoli;
- Rigore;
- Classifica.

-Di ciascuna scuola si vuole sapere l'ID, la password, il nome ed il comune di appartenenza.
-Di ciascuno studente si vuole sapere l'ID, la password, il nome, il cognome, il sesso, la scuola di appartenenza e il codice della competizione alla quale ha partecipato.
-Per la competizione (Triathlon) si vuole sapere l'ID ed i partecipanti.

-Delle tre discipline diverse, che saranno strettamente correlate con il Triathlon all'interno dello schema concettuale, si vuole conoscere l'ID, il risultato (tempo, goal o no goal) e la descrizione.
-Di ciascuna classifica si conoscono infine l'ID, le scuole, gli studenti che hanno partecipato alle varie discipline e le relative posizioni.
-Deve inoltre essere nota la data nella quale vengono svolte le Olimpiadi.

*A questo punto è possibile realizzare lo **schema concettuale (ER)** ed analizzarlo*:

Analisi descrittiva dell'ER volta a motivare tutte le scelte attuate



Dopo aver studiato correttamente la realtà di interesse, per poter creare uno schema concettuale adatto al progetto, è necessario andare ad individuare altri punti fondamentali: le associazioni tra le entità (individuate precedentemente proprio grazie allo studio della realtà d'interesse), le chiavi primarie che compongono queste entità, le relative chiavi esterne, identificazione di eventuali generalizzazioni (classificandole correttamente in base al caso preso in esame), i vincoli intrarelazionali e le cardinalità che caratterizzano l'intero schema.

-Primo punto: individuazione delle associazioni

- Scuola → **FREQUENTA** → Studente
- Studente → **POSIZIONA** → Classifica
- Studente → **GAREGGIA** → Triathlon
- Triathlon → **PREVEDE** → Velocità
- Triathlon → **PREVEDE** → Corsa Ostacoli
- Triathlon → **PREVEDE** → Rigore

- ❖ Velocità → **GENERA** → Classifica
- ❖ Corsa Ostacoli → **GENERA** → Classifica
- ❖ Rigore → **GENERA** → Classifica

-Secondo punto: individuazione delle chiavi primarie (Primary Key)

Per l'entità *Scuola* e per l'entità *Studente* è stata scelta come chiave primaria un ID, che permette dunque un'identificazione **univoca** per ogni record.
Per tutte le altre entità che compongono lo schema concettuale (*Classifica*, *Triathlon*, *Velocità*, *Corsa a ostacoli* e *Rigore*), invece, è stato deciso di creare delle **chiavi composte**, poiché nel database che sarà creato in seguito, differenza di *Scuola* e *Studente*, l'ID non basterà per identificare univocamente ogni record.
Nello specifico, per tutte le entità citate poc'anzi la coppia di attributi che forma la chiave primaria sono l'ID dell'entità stessa + l'ID dello studente partecipante.

-Terzo punto: individuazione delle chiavi esterne (Foreign Key)

Le chiavi esterne sono state scelte in modo tale che, nel momento in cui verrà definitivamente creato il database relazionale, le query potranno essere comprensibili e facilmente codificabili.
Va comunque ricordato che le chiavi esterne potrebbero essere aggiunte più avanti, nel momento in cui si passerà alla fase di scrittura dello schema logico.
Nello specifico, per il momento le chiavi esterne sono le seguenti:

- Attributo "*triathlon*" dell'entità Studente, che è in integrità referenziale con l'ID dell'entità triathlon per identificare a che singola Olimpiade ha preso parte lo studente;
- Attributo "*scuola appartenenza*" dell'entità Studente, che è in integrità referenziale con l'ID dell'entità Scuola, questo per capire che scuola è frequentata dall'alunno;
- Attributo "*scuola*" dell'entità Classifica, che è in integrità referenziale con l'ID dell'entità Scuola, per una maggiore chiarezza ed identificazione di una singola scuola all'interno della classifica che la singola competizione va a creare;
- Attributo "s*tudente*" dell'entità Classifica, che è in integrità referenziale con l'ID dell'entità Studente, in modo tale da identificare, all'interno della classifica creata, lo studente;
- Attributo "*partecipante*" dell'entità Triathlon. Questo attributo è in integrità referenziale con l'ID dell'entità Studente, ciò permette di identificare il singolo studente nell'Olimpiadi alla quale decide di prendere parte.
- Attributo "*partecipante*" dell'entità Velocità. Questo attributo è in integrità referenziale con l'ID dell'entità Studente, ciò permette di identificare il singolo studente nell'Olimpiadi alla quale decide di prendere parte.
- Attributo "*partecipante*" dell'entità Corsa a Ostacoli. Questo attributo è in integrità referenziale con l'ID dell'entità Studente, ciò permette di identificare il singolo studente nell'Olimpiadi alla quale decide di prendere parte.
- Attributo "*partecipante*" dell'entità Rigore. Questo attributo è in integrità referenziale con l'ID dell'entità Studente, ciò permette di identificare il singolo studente nell'Olimpiadi alla quale decide di prendere parte.

-Quarto punto: individuazione dei vincoli intrarelazionali

- Scuola:
  - ID != NULL && ID == String
  - Nome == String
  - Comune == String
  - Password Scuola == String
- Studente:
  - ID != NULL && ID == String
  - Nome == String
  - Cognome == String

- Password Studente == String
- Sesso == Enumerativo
- Triathlon == String
- Scuola appartenenza == String
- Classifica == String

- Classifica:
  - ID != NULL && ID == String
  - Studente != NULL && Studente == String
  - Scuola == String
  - Posizione == int
- Triathlon:
  - ID != NULL && ID == String
  - Partecipante != NULL && Partecipante == String
- Velocità:
  - ID != NULL && ID == String
  - Partecipante != NULL && Partecipante == String
  - Tempo == time
  - Descrizione == String
- Corsa ostacoli:
  - ID != NULL && ID == String
  - Partecipante != NULL && Partecipante == String
  - Tempo == time
  - Descrizione == String
- Rigore:
  - ID != NULL && ID == String
  - Partecipante != NULL && Partecipante == String
  - Goal == boolean
  - Descrizione == String

-Quinto punto: identificazione di eventuali generalizzazioni

Generalizzazioni utilizzate per lo schema concettuale: 0.
Non avendo adottato alcuna generalizzazione all'interno dello schema concettuale, non c'è necessità di un'eventuale ristrutturazione.

-Sesto punto: cardinalità

- Scuola → **FREQUENTA** → Studente
  - Il ramo che va da Scuola a Frequenta ha cardinalità (1, N) perché una scuola è frequentata almeno da 1 o più studenti;
  - Il ramo che va da Studente a Frequenta ha cardinalità (1, 1) obbligatoria, perché uno studente frequenta obbligatoriamente una e una sola scuola.
- Studente → **POSIZIONA** → Classifica
  - Il ramo che va da Studente a Posiziona ha cardinalità (0, N) opzionale, perché uno studente non è obbligato a posizionarsi in una classifica nel caso in cui non partecipasse alle Olimpiadi, ma il numero di studenti che intende partecipare e classificarsi è pari a N;
  - Il ramo che va da Classifica a Posiziona ha cardinalità (1, N), perché una classifica può posizionare almeno da 1 a N studenti.
- Studente → **GAREGGIA** → Triathlon
  - Il ramo che va da Studente a Gareggia ha cardinalità (0, N) opzionale, perché, come per quanto riguarda il posizionamento in classifica, un alunno non è obbligato a partecipare alla competizione, ma il numero di ragazzi che possono aderire è comunque pari a N;

- Il ramo che va da Triathlon a Gareggia ha cardinalità (1, N), perché una disciplina può essere svolta a partire da 1 studente fino ad un massimo di N.

- Triathlon → **PREVEDE** → Velocità
- Triathlon → **PREVEDE** → Corsa Ostacoli
- Triathlon → **PREVEDE** → Rigore
  - Il ramo che va da Triathlon a ognuna delle tre discipline che prevede la competizione è obbligatorio (1, 1), poiché un triathlon prevede una e una sola gara di velocità, una e una sola corsa ad ostacoli e uno e un solo calcio di rigore.
  - Il ramo che va dalle tre discipline a Triathlon è invece (1, N): una qualsiasi delle tre discipline, infatti, fa obbligatoriamente parte del triathlon, ma allo stesso tempo è possibile fare più gare della stessa disciplina, ognuna della quali appartiene inevitabilmente a triathlon differenti.
- Velocità → **GENERA** → Classifica
- Corsa Ostacoli → **GENERA** → Classifica
- Rigore → **GENERA** → Classifica
  - Il ramo che va dalle tre discipline figlie a Genera ha cardinalità (1, N), perché le tre discipline generano almeno una classifica fino ad arrivare ad un massimo di N;
  - Il ramo che va da Classifica a Genera ha cardinalità (1, N), perché una classifica è generata da un minimo di una disciplina fino ad un massimo di N.

*-**Progettazione logica** della base di dati:*

Scrittura dello schema logico partendo dal risultato finale della progettazione concettuale del database relazionale.

Alla luce di ciò che abbiamo di fronte dopo l'analisi dettagliata dell'Entity Relationship, per la composizione del database necessario a portare a termine il DB, bisogna passare alla progettazione logica. Le entità diverranno relazioni e, eventualmente, ne saranno aggiunte altre (usando i nomi delle associazioni) per coerenza con le cardinalità inserite.
La composizione dello schema si divide in due fasi:

1. Formazione delle relazioni secondo le regole delle cardinalità;
2. Segnalazione di eventuali chiavi esterne in integrità referenziale.

Fase 1: formazione delle relazioni

Scuole (ID, Nome, Comune, Password Scuola)
Studenti (ID, Nome, Cognome, Password Studente, Scuola appartenenza, Sesso, Classifica)

Tra le due entità, è possibile osservare dall'ER che siamo di fronte ad una cardinalità (1: N), di conseguenza si necessita di una chiave esterna, all'interno della relazione con cardinalità obbligatoria (Studenti) che faccia riferimento alla relazione Scuole: "*Scuola appartenenza*".

Classifiche (ID, Scuola, Studente, Posizione)
Posiziona (Studente, Classifica)
Caso preso in considerazione: associazione tra Studenti e Classifiche. Si nota che la cardinalità è (N: N), con un ramo opzionale (quello che va da Studente a posiziona) con cardinalità (0, N). In questo modo è possibile creare una nuova relazione, con il nome della associazione, che contenga al suo interno due chiavi primarie con i nomi delle entità associate (studente e classifica).

Gareggia (Triathlon, Studente, data)
Triathlon (ID, partecipante, velocità, corsa_ostacoli, rigore)
Velocità (ID, partecipante, tempo velocità, descrizione, Classifica)
Corse Ostacoli (ID, partecipante, tempo corsa, descrizione, Classifica)
Rigori (ID, partecipante, goal, descrizione, Classifica)

La porzione di schema concettuale che parte dall'entità Studente e arriva alle tre diverse discipline è stato tradotto in schema logico come riportato sopra ed eccone i motivi: è stata creata la relazione "*Gareggia*", dato che abbiamo una cardinalità (N: N) da Studente a Triathlon. La chiave primaria di questa entità è composta, cioè formata dall'attributo "*Triathlon*" (che è anche in integrità referenziale) e dall'attributo "*data*".
Inoltre è necessario rivedere gli attributi dell'entità Triathlon, perché, trovandoci di fronte ad una cardinalità (1, N), con un ramo obbligatorio (1, 1) da Triathlon all'associazione "*prevede*", vanno inserite le varie chiavi esterne delle discipline.
Viene infine inserita una foreign key "*Classifica*" in ognuna delle tre discipline in modo tale da rispettare la cardinalità (N: N) di questa ultima associazione e, soprattutto, andare ad identificare di che classifica fa parte la gara specifica.

Fase 2: Integrità Referenziali

Scuole (ID, Nome, Password Scuola, Comune)
Studenti (ID, Nome, Cognome, Password Studente, Scuola appartenenza, Sesso, Classifica)
Classifiche (ID, Scuola, Studente, Posizione)
Posiziona (Studente, Classifica)
Gareggia (Triathlon, Studente, data)
Triathlon (ID, partecipante, velocità, corsa_ostacoli, rigore)
Velocità (ID, partecipante, tempo velocità, descrizione, Classifica)
Corse Ostacoli (ID, partecipante, tempo corsa, descrizione, Classifica)
Rigori (ID, partecipante, goal, descrizione, Classifica)

Studenti.Scuola_appartenenza è in integrità referenziale con Scuole.ID
Studenti.Classifica è in integrità referenziale con Classifiche.ID
Classifiche.Scuola è in integrità referenziale con Scuole.ID
Classifiche.Studente è in integrità referenziale con Studenti.ID
Posiziona.Studente è in integrità referenziale con Studenti.ID
Posiziona.Classifica è in integrità referenziale con Classifiche.ID
Gareggia.Thriathlon è in integrità referenziale con Triathlon.ID
Gareggia.Studente è in integrità referenziale con Studente.ID
Triathlon.Partecipante è in integrità referenziale con Studente.ID
Trathlon.Velocità è in integrità referenziale con Velocità.ID
Triathlon.Corsa_Ostacoli è in integrità referenziale con Corse_Ostacoli.ID
Triathlon.Rigore è in integrità referenziale con Rigori.ID
Velocità.Partecipante è in integrità referenziale con Studente.ID
Velocità.Classifica è in integrità referenziale con Classiche.ID
Corse_Ostacoli.Partecipante è in integrità referenziale con Studente.ID
Corse_Ostacoli.Classifica è in integrità referenziale con Classiche.ID
Rigori.Partecipante è in integrità referenziale con Studente.ID
Rigori.Classifica è in integrità referenziale con Classiche.ID

*-Ultima fase della progettazione logica: individuare se sono necessarie operazioni di **normalizzazione**:*

Fase della progettazione logica del Database che ne garantisce a pieno la qualità e l'efficienza.

In questa parte del progetto ci si andrà ad occupare di come sistemare lo schema logico del database relazionale per scongiurare ognuno di questi 3 specifici problemi:

1. **Anomalie di inserimento**: impossibilità di inserire un dato nel caso in cui non venissero inseriti anche i correlati;
2. **Anomalie di cancellazione**: eliminazione non corretta di alcuni dati a causa dell'eliminazione di eventuali dati correlati;
3. **Anomalie di aggiornamento**: problema frequente che si verifica nel momento in cui un database presenta al suo interno delle ridondanze e non vengono dunque aggiornate tutte le occorrenze dello stesso dato.

Anche questa fase è divisa in due passaggi fondamentali:

1. Normalizzazione dello schema adottando le regole che vanno dalla prima forma normale fino ad arrivare a quella di Boyce-Codd;
2. Riscrivere eventuali nuove integrità referenziali per rendere maggiormente chiaro lo schema logico finale che si andrà a comporre.

Fase 1: Normalizzazione

- 1NF: Lo schema logico è in prima forma normale se tutti gli attributi delle varie entità sono atomici (cioè non scomponibili).

  Scuole (ID, Nome, Password Scuola, Comune)
  Studenti (ID, Nome, Cognome, Password Studente, Scuola appartenenza, Sesso, Classifica)
  Classifiche (ID, Scuola, Studente, Posizione)
  Posiziona (Studente, Classifica)
  Gareggia (Velocità, Corsa Ostacoli, Rigore)
  Velocità (ID, Partecipante, tempo, descrizione, Classifica)
  Corse Ostacoli (ID, Partecipante, tempo, descrizione, Classifica)
  Rigori (ID, Partecipante, goal, descrizione, Classifica)

  Come possiamo notare, non c'è alcun attributo **NON atomico** all'interno delle varie relazioni che compongono lo schema logico. Di conseguenza il nostro database rispetterà a tutti gli effetti la prima forma normale.

- 2NF: Due requisiti da soddisfare:
  - Lo schema deve essere in prima forma normale;
  - Tutti gli attributi non chiave devono dipendere funzionalmente dall'intera chiave primaria.

  In cosa consiste la dipendenza funzionale: descrizione di un legame di funzione tra gli attributi della relazione.
  Esempio: X → Y, significa che Y dipende funzionalmente da X e/o X determina Y.

In cosa consiste la seconda forma di normalizzazione, esempio:
T1 (A1, A2, A3, A4, A5) con {A1, A2} → A3, con A1 → 14 e con A2 → A5.
Qui, per normalizzare in seconda forma, dovremmo trasformare la relazione in più relazioni nel seguente modo:
T1 (A1, A2, A3)
T2 (A1, A4)
T3 (A2, A5)
T1.A1 è in integrità referenziale con T2.A1
T1.A2 è in integrità referenziale con T3.A2

Nel nostro caso abbiamo le seguenti dipendenze funzionali:

- Relazione *Scuole:*
  - ID → Nome, Password, Comune
- Relazione *Studenti:*
  - ID → Nome, Cognome, Sesso, Password, Scuola appartenenza, Classifica
- Relazione *Classifiche*:
  - ID → Scuola, Studente, Posizione
- Relazione *Gareggia*:
  - Triathlon, Data → Studente
- Relazione *Triathlon*:
  - ID → Partecipante, Velocità, Corsa Ostacoli, Rigore
- Relazione *Velocità*:
  - ID → Partecipante, tempo velocità, descrizione, Classifica
- Relazione *Corse Ostacoli*:
  - ID → Partecipante, descrizione, tempo corsa, classifica
- Relazione *Rigori*:
  - ID → Partecipante, descrizione, goal, classifica

Tutte le relazioni sono in 2NF.

- 3NF: Due requisiti da soddisfare:
  1. Lo schema deve essere in seconda forma normale;
  2. Tutti gli attributi non chiave dipendono dall'intera chiave primaria, ma non devono essere presenti dipendenze transitive.

In cosa consiste la dipendenza transitiva, esempio:
T1 (A1, A2, A3, A4) con A2 → A4.
In questo caso, i passaggi per normalizzare in terza forma sono i seguenti: T1 (A1, A2, A3), con A2 che diventa chiave esterna T2 (A2, A4).
T1.A2 è in integrità referenziale con T2.A2

Nel nostro caso, non sono presenti alcune dipendenze transitive, poiché lo schema logico, al momento, si presenta nel seguente modo:
Scuole (ID, Nome, Password Scuola, Comune)
Indirizzi (Comune, Provincia, Via, Numero Civico)
Studenti (ID, Nome, Cognome, Password Studente, Scuola appartenenza, Sesso, Classifica)
Classifiche (ID, Scuola, Studente, Posizione)

Posiziona (Studente, Classifica)
Gareggia (Triathlon, Studente, data)
Triathlon (ID, Partecipante, velocità, corsa_ostacoli, rigore)
Velocità (ID, Partecipante, tempo, descrizione, Classifica)
Corse Ostacoli (ID, Partecipante, tempo, descrizione, Classifica)
Rigori (ID, Partecipante, goal, descrizione, Classifica)

- BCNF: Due requisiti da soddisfare:
  1. Lo schema deve essere in prima forma normale;
  2. Tutti gli attributi che compongono la chiave candidata di una relazione devono essere determinanti degli altri attributi rimanenti all'interno della stessa relazione.

  Riflessione da fare per quanto riguarda la BCNF: nel caso in cui l'operazione di normalizzazione (partizionamento della relazione che non presenta i requisiti necessari a soddisfare la forma normale di cui ci stiamo occupando al momento) portasse ad una perdita di dati, è buona prassi evitare di normalizzare e lasciare la relazione invariata.

Fase 2: Integrità Referenziali

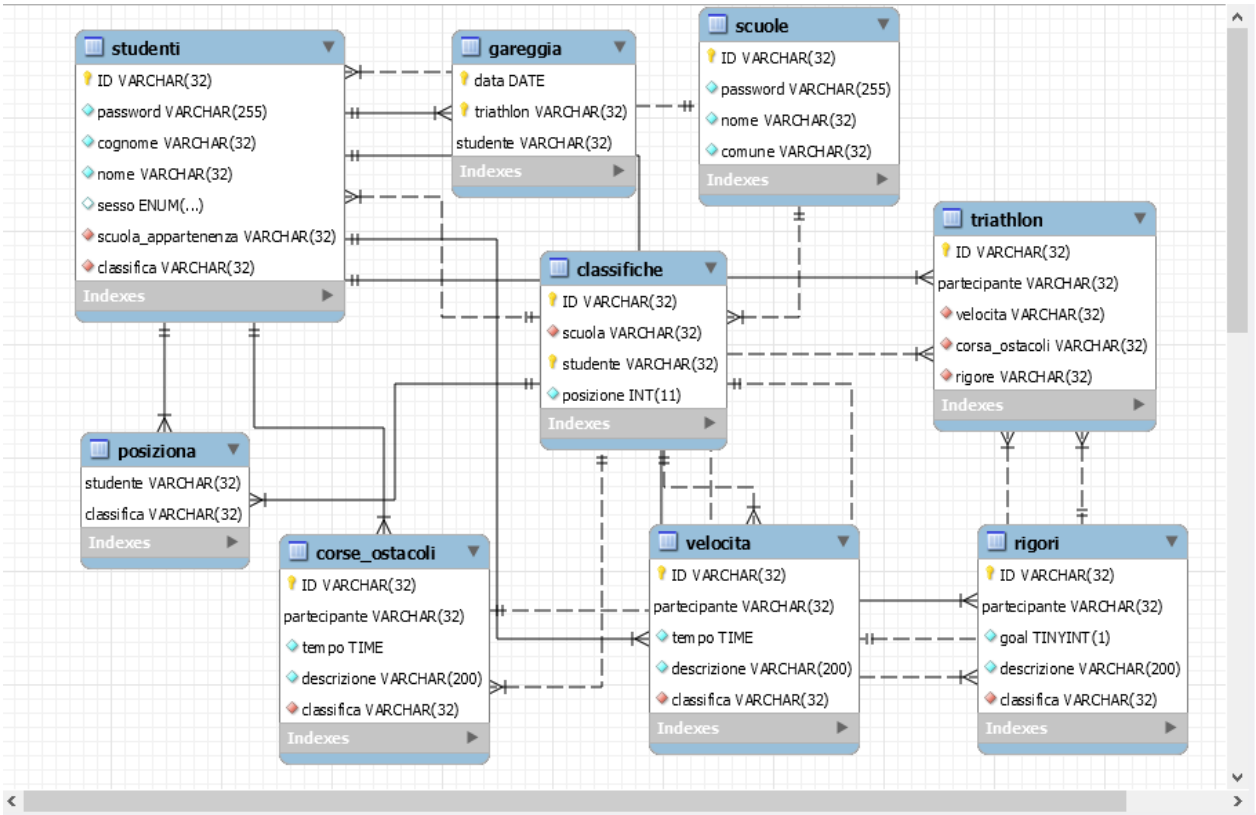
Apportate le dovute modifiche ed individuate tutte le integrità referenziali, è possibile scrivere per esteso tutto lo schema, concludendo dunque la progettazione logica del database relazionale.

Scuole (ID, Nome, Password Scuola, Comune)
Studenti (ID, Nome, Cognome, Password Studente, Scuola appartenenza, Sesso, Classifica)
Classifiche (ID, Scuola, Studente, Posizione)
Posiziona (Studente, Classifica)
Gareggia (Triathlon, Studente, data)
Triathlon (ID, Partecipante, velocità, corsa_ostacoli, rigore)
Velocità (ID, Partecipante, tempo, descrizione, Classifica)
Corse Ostacoli (ID, Partecipante, tempo, descrizione, Classifica)
Rigori (ID, Partecipante, goal, descrizione, Classifica)

Studenti.Scuola_appartenenza è in integrità referenziale con Scuole.ID
Studenti.Classifica è in integrità referenziale con Classifiche.ID
Classifiche.Scuola è in integrità referenziale con Scuole.ID
Classifiche.Studente è in integrità referenziale con Studenti.ID
Posiziona.Studente è in integrità referenziale con Studente.ID
Posiziona.Classifica è in integrità referenziale con Classifiche.ID
Gareggia.Triathlon è in integrità referenziale con Triathlon.ID
Gareggia.Studente è in integrità referenziale con Studenti.ID
Triathlon.Partecipante è in integrità referenziale con Studenti.ID
Triathlon.Velocità è in integrità referenziale con Velocità.ID
Triathlon.Corsa_Ostacoli è in integrità referenziale con Corse_Ostacoli.ID
Triathlon.Rigore è in integrità referenziale con Rigori.ID
Velocità.Partecipante è in integrità referenziale con Studenti.ID
Velocità.Classifica è in integrità referenziale con Classifiche.ID
Corse_Ostacoli.Partecipante è in integrità referenziale con Studenti.ID
Corse_Ostacoli.Classifica è in integrità referenziale con Classifiche.ID

Rigori.Partecipante è in integrità referenziale con Studenti.ID
Rigori.Classifica è in integrità referenziale con Classifiche.ID

*-Ultima fase: **progettazione fisica** del database in MySQL:*



-Altre informazioni sul database e sulla **privacy** – utenti, permessi e privilegi (**GRANT**):

All'interno della base di dati che è stata implementata, da sviluppatore, si è pensato di creare un utente ("*ProfAdmin*"), identificato opportunamente con una password e in grado di effettuare tre operazioni principali sulle tabelle "*studenti*", "*scuole*" e "*classifiche*": operazioni di **insert**, di **update** e **delete**.
All'interno della sezione "*Users and Privileges*" di MySQL è infatti possibile visionare la tabella degli utenti e controllare effettivamente se la creazione del Prof. Amministratore è andata a buon fine o meno.

*-Allegato screenshot della **creazione** dei **GRANT** da codice SQL:*

-

DB_Olimpiadi | SQL File 3*

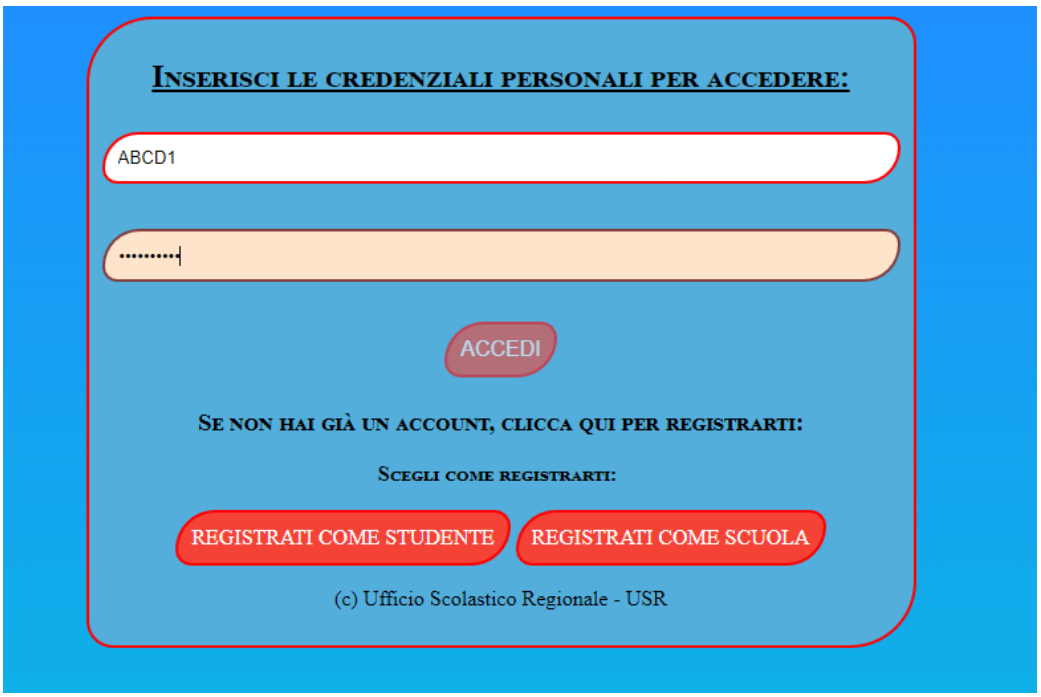
Limit to 300 rows

```
247     #Creazione di un nuovo utente:
248     DROP USER IF EXISTS 'ProfAdmin'@'localhost';
249     CREATE USER 'ProfAdmin'@'localhost' IDENTIFIED BY 'PasswordProvaProf';
250
251     #Creazione dei vari permessi (GRANT):
252     GRANT delete ON studenti TO 'ProfAdmin'@'localhost';
253     GRANT delete ON scuole TO 'ProfAdmin'@'localhost';
254
255     GRANT insert ON studenti TO 'ProfAdmin'@'localhost';
256     GRANT insert ON scuole TO 'ProfAdmin'@'localhost';
257
258     GRANT update ON studenti TO 'ProfAdmin'@'localhost';
259     GRANT update ON scuole TO 'ProfAdmin'@'localhost';
260
261     GRANT insert ON classifiche TO 'ProfAdmin'@'localhost';
262     GRANT update ON classifiche TO 'ProfAdmin'@'localhost';
263
```

*Creazione del **sito Web**:*

L'applicazione Web prevista per il progetto ha come scopo principale la divulgazione delle **classifiche** delle gare effettuate tra gli alunni delle varie scuole partecipanti delle Olimpiadi Online. Per rendere possibile questa funzionalità, è stato previsto un **login** (o un'eventuale **registrazione**) per accedere alla pagina iniziale riguardante le classifiche.

-Pagina iniziale di login:



-Pagina con le classifiche:

Classifiche Olimpiadi Online - USR

Home Login Contact

Posizione	ID Studente	Cognome	Nome	Istituto di appartenenza	Tempo velocità	Tempo corsa a ostacoli	Rigore
1	ABCD1	Rola	Andrea	Istituto Facchinetti	00:02:36	00:02:36	Goal
2	DDRR4	Rossi	Luca	Istituto Newton	00:03:02	00:03:02	Goal
3	MNBV4	Colombo	Giacomo	Istituto Parini	00:03:05	00:03:05	Goal
4	QAWS6	Bolletta	Chiara	Istituto Verga	00:03:17	00:03:17	Goal

*-Creazione dell'interfaccia di **collegamento delle scuole**:*

Per la creazione di un prototipo che faccia capire immediatamente all'utente come funziona l'intero sistema e dare una sorta di visione generale di come si comporterebbe una parte della città impegnata nelle Olimpiadi, è stato deciso di utilizzare il software **CISCO Packet Tracer**. All'interno della porzione "*Physical*" è infatti possibile visionare una piccola cittadina dall'alto, che ipoteticamente ha al suo interno 3 scuole che hanno aderito alla proposta delle Olimpiadi Online per studenti ipovedenti. Le 3 scuole (Istituto Leibniz, Istituto Facchinetti ed Istituto Pascoli) sono più precisamente 3 generici cluster resi maggiormente realistici mediante l'ausilio delle immagini delle palestre nelle quali si svolge la competizione (è stata infatti sfruttata una funzionalità di Packet Tracer che dà la possibilità di importare un qualsiasi tipo di foto per personalizzare i propri lavori).
È inoltre possibile vedere un quarto, un quinto, un sesto ed un settimo cluster (*Fornitori di Servizi)* contenenti **Server** che hanno la funzionalità di erogare i servizi principali per la comunicazione online tra le scuole.
Per rendere chiara la simulazione è di fondamentale importanza configurare le varie reti in modo tale che i principali protocolli della pila ISO/OSI vengano correttamente rispettati durante la spedizione e la ricezione di pacchetti che vengono inviati e/o arrivano dalla rete.
Entrando più nello specifico, i protocolli (ed i servizi) che prevede la simulazione sono i seguenti:

- **DHCP** → Per l'assegnazione in modo dinamico degli indirizzi IP di una rete (solo il server che eroga il servizio viene configurato staticamente). Questa assegnazione dinamica di IP avviene all'interno delle tre scuole, gli unici indirizzi IP statici (quelli di ogni Server DHCP) con i relativi Default Gateway sono i seguenti:
  - Istituto Leibniz:
    - Indirizzo IP Server: 192.168.0.2;
    - Default Gateway: 192.168.0.1;
  - Istituto Facchinetti:
    - Indirizzo IP Server: 172.16.1.2;
    - Default Gateway: 172.16.1.1;

  - Istituto Pascoli:
    - Indirizzo IP Server: 192.168.1.2;
    - Default Gateway: 192.168.1.1
- **DNS** → Ufficio "Fornitore Servizio 1". Per richiedere (fare una vera e propria query) ad un server per ottenere informazioni sulle risorse, come nomi, indirizzi IP ed altre informazioni aggiuntive. Indirizzo IPv4 e Default Gateway del Server DNS:
  - IPv4: 10.0.0.1;
  - Default Gateway: 10.0.0.254;
- **HTTP** → Ufficio "Fornitore Servizio 2". Per rendere maggiormente chiara la simulazione nel momento in cui un qualsiasi client proveniente dalle scuole richiede la visione del sito sviluppato apposta per le Olimpiadi Online (solo visione generale lato client, quindi HTML e CSS, per dare un'idea chiara all'utente inesperto di ciò che sarà il prototipo applicato). Indirizzo IPv4 e Default Gateway del Server HTTP:
  - IPv4: 130.0.0.1;
  - Default Gateway: 130.0.0.254;
- **FTP** → Per accettare eventuali richieste client (autenticato correttamente con uno username ed una password) che potrebbe avere intenzione di trasferire un qualsiasi tipo di file all'interno di un server remoto. Nel nostro caso il file remoto è "prova1FTP.txt" e tutti gli utenti possono accedervi una volta immesse le credenziali (username: User 1 e Password: 123). Indirizzo IPv4 e Default Gateway del Server FTP:
  - IPv4: 130.1.0.1;
  - Default Gateway: 130.1.0.254;
- **EMAIL** → Per dare la possibilità ad un client fittizio di inviare e ricevere e-mail da un altro client. Configurazione del Server MAIL con apposito dominio ("gmail.com", per garantire la realisticità della simulazione), username e password. Indirizzo IPv4 e Default Gateway del Server MAIL:
  - IPv4: 18.0.0.1;
  - Default Gateway: 18.0.0.254;

  Credenziali per i primi 2 utenti:
  - Utente 1 → Username: User 1 – Password: 123
  - Utente 2 → Username: User 2 – Password: 456

- **VPN** → Un aspetto fondamentale dell'intero progetto è quello di garantire la sicurezza dei dati trasmessi, ecco perché è necessario, anche nella simulazione effettuata su CISCO Packet Tracer, andare a mostrare delle soluzioni valide che certifichino la confidenzialità, l'integrità e la disponibilità (*triangolo **CIA***) dei dati che viaggiano in rete da una scuola all'altra. La soluzione che si deve apportare in questo caso è la creazione di una VPN (*Virtual Private Network*). Per renderla possibile, andrebbero scritte alcune righe di codice nella sezione "*CLI*" dei vari router, ma per semplicità supponiamo di creare la VPN all'interno del cluster "Fornitore Servizio 2" e all'interno dell'Istituto Leibniz.

-Ecco tutti i passaggi da fare:

Router 3> enable
Router 3# configure terminal
Router 3(config)# license boot module c2900 technology-package securityk9
Router 3(config)# end
Router 3# copy running-config startup-config
Router 3# reload

Questo primo passaggio è necessario al fine di installare il pacchetto software per la sicurezza "*SECURITYK9*" sui 2 router che fungono da Security Gateway.

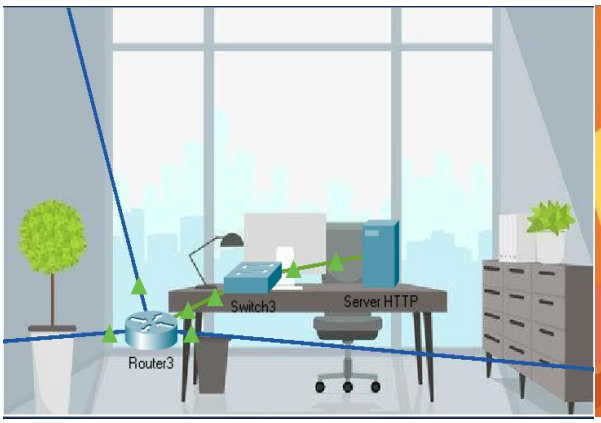
Ripetere l'operazione anche sull'altro router con il quale sarà consentita la comunicazione: "Router Leibniz".

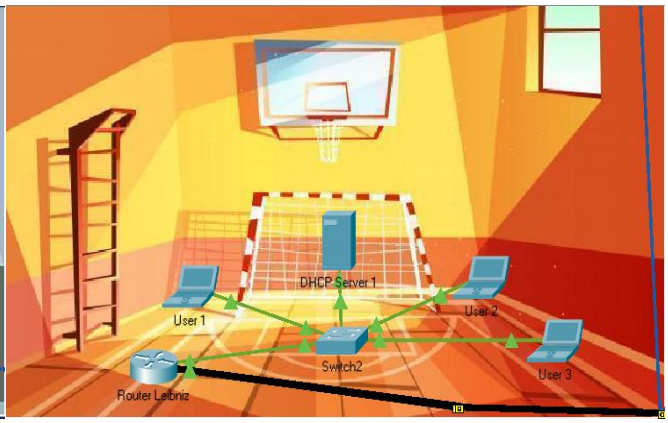
Router Leibniz> enable
Router Leibniz# configure terminal
Router Leibniz(config)# license boot module c2900 technology-package securityk9
Router Leibniz(config)# end
Router Leibniz# copy running-config startup-config
Router Leibniz# reload

Dopo aver effettuato queste due operazioni di vitale importanza, si può procedere con la vera e propria configurazione delle due VPN (come in precedenza, va fatto su entrambi i router).

Router 3 (config)# access-list 110 permit ip 192.168.0.0 0.0.0.255 130.0.0.0 0.0.255.255
Router 3(config)# crypto isakmp policy 10
Router 3(config-isakmp)# encryption aes
Router 3(config-isakmp)# authentication pre-share
Router 3(config-isakmp)# group 2
Router 3(config-isakmp)# exit
Router 3(config)# crypto isakmp key cisco address 24.0.0.2
Router 3 (config)# crypto ipsec transform-set VPN-SET esp-3des esp-sha-hmac
Router 3 (config)# crypto map VPN-MAP 10 ipsec-isakmp
Router 3(config-crypto-map)# description VPN connection to Router 3
Router 3(config-crypto-map)# set peer 24.0.0.2
Router 3 (config-crypto-map)# set transform-set VPN-SET
Router 3 (config-crypto-map)# match address 110
Router 3 (config-crypto-map)# exit
Router 3 (config)# interface S1/0/0
Router 3(config-if)# crypto map VPN-MAP

È possibile effettuare le stesse identiche operazioni per la configurazione del Router Leibniz.



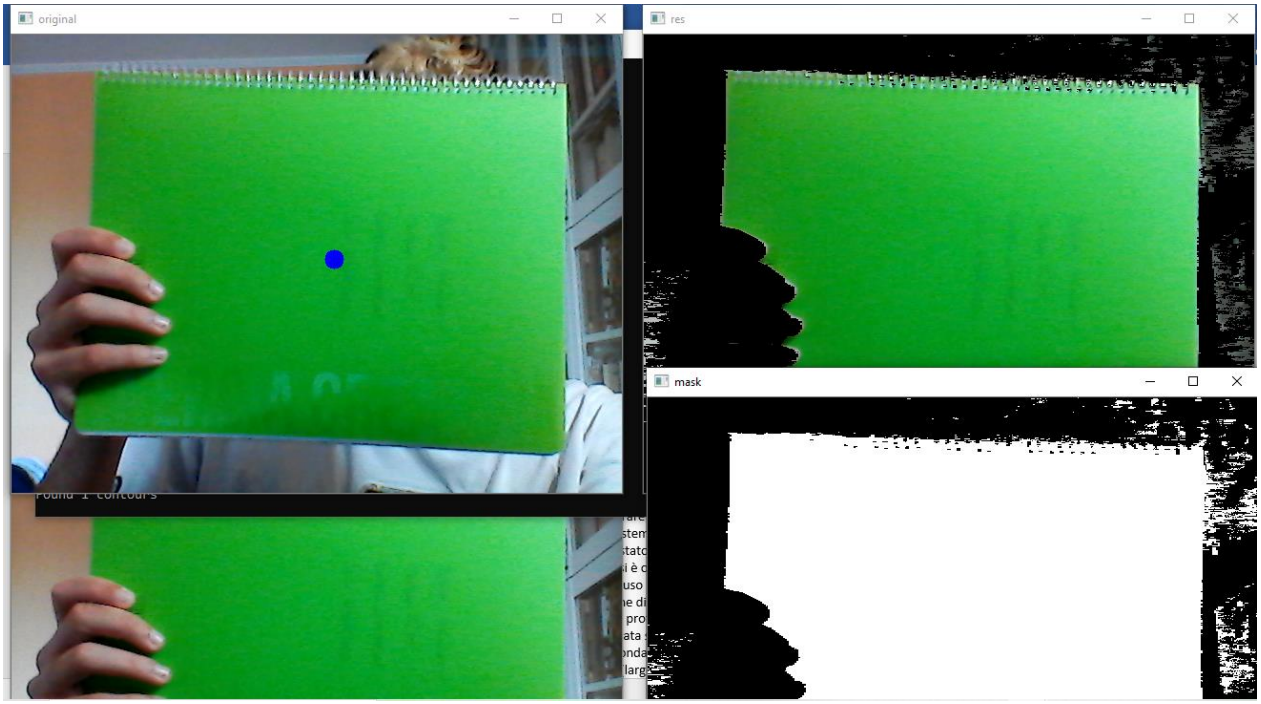


*-Allegato della città con **interconnessione** di **scuole** ed uffici di servizi:*

*-Creazione dell'**applicazione** per le **3 discipline**:*

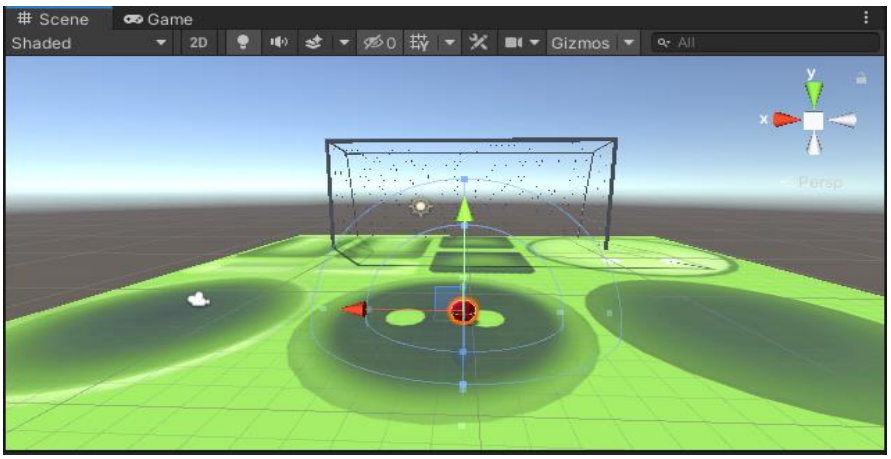
L'applicazione prototipo, che ampliata nel corso del tempo consente il normale svolgimento del triathlon, è stata pensata e divisa in due parti fondamentali:

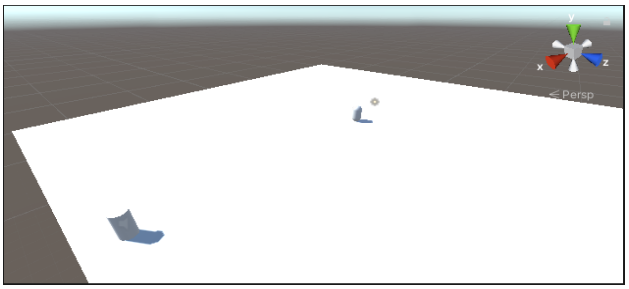
- Un file di codici **Python** che consenta, tramite fotocamera, la chiara rilevazione di un oggetto "protagonista" della scena di **colore verde** (tra gli oggetti possiamo avere il pallone da calciare o l'ostacolo da superare durante la corsa) e ne consenta il calcolo di contorni e centro per raccogliere questi dati ed inviarli ad un altro terminale grazie all'ausilio di una **socket UDP**:

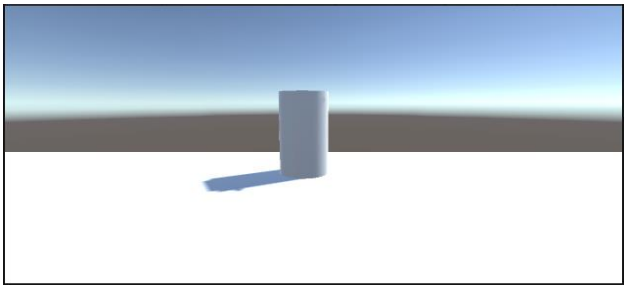


- Una serie di file in ambiente *Unity* che, grazie ad un'interfaccia grafica adatta e alla scrittura di alcuni script in **C#,** permetta di compiere le seguenti operazioni:

- **Ricevere** in ingresso i **dati** provenienti dalla socket UDP in Python circa la posizione esatta dell'oggetto rilevato dalla fotocamera;
- **Muovere** l'eventuale **palla** e **visuali** dei giocatori;
- Avvalersi dell'**audio 3D** per segnalare e far percepire, nella maniera più precisa possibile, un oggetto in prossimità alla persona ipovedente (in modo tale da capire DOVE calciare il pallone e/o QUANDO saltare l'ostacolo).







Nella prima foto (pagina precedente), è possibile vedere un vero e proprio campo da calcio virtuale. All'interno di esso sono presenti una porta ed una palla, contornata opportunamente da un'area entro la quale si attiva il suono 3D di una goccia d'acqua (che aumenta e/o diminuisce di intensità a seconda della vicinanza all'oggetto).

Nella seconda e nella terza foto, invece, è possibile vedere una sorta di *beta* della corsa ad ostacoli: il gioco è sviluppato in prima persona e al suo interno sono implementati script che permettono ogni tipo di movenza al personaggio. Il cilindro della terza foto, in particolare, rappresenta una versione primitiva di un ostacolo: con lo stesso principio di funzionamento della palla del calcio di rigore, infatti, permette l'attivazione di un suono tridimensionale in base a quanto siamo in prossimità dell'entità.

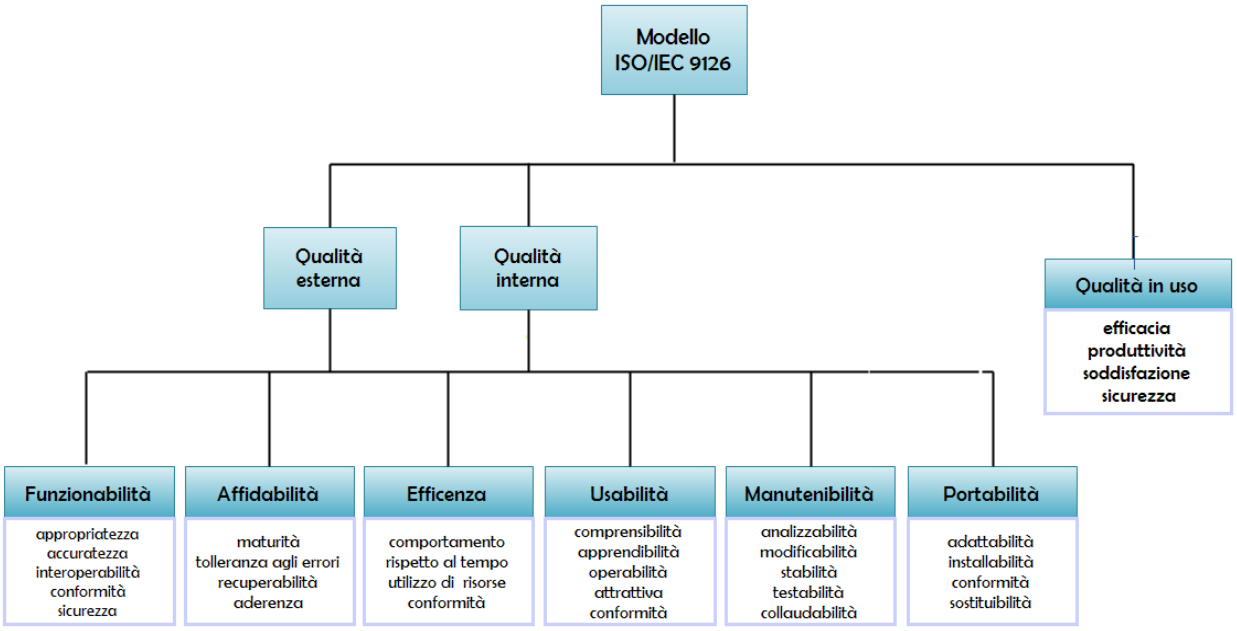
5. **Costruzione e rilascio:** nella fase finale di costruzione del progetto è stato possibile rilasciare un prototipo funzionante dell'applicazione *Unity* ad un'azienda che, dopo aver raccolto tutte le informazioni essenziali sui programmi da realizzare, si è realmente mostrata interessata al progetto: la **DSC** di **Legnano**. Questo permette inevitabilmente di dare credibilità e continuità alle soluzioni proposte e la realizzazione di altri prototipi.

6. **Valutazione da parte del cliente:** avendo scelto di sviluppare il lavoro secondo le rigorose fasi del modello a spirale, alla fine del ciclo è assolutamente necessario ascoltare il parere del cliente che meglio di tutti saprà indicare quali sono le migliorie da apportare. Questo è di fondamentale importanza per riempire di idee corrette la sezione "*possibili implementazioni future*".

Analisi dei software (Standard ISO):
Tutti i software sviluppati all'interno del progetto sono stati realizzati nel rispetto assoluto delle principali **normative** che un qualsiasi sviluppatore di programmi è tenuto a rispettare: gli **standard ISO 12207 e 9126**.

- ISO **12207** → Si sono andate a definire tutte le fasi del ciclo di vita dei software (dalla fase di acquisizione alla fase di dismissione). Procedendo con ordine:
  1. ***Acquisizione***: *identificazione delle richieste del cliente* → Perché sono stati implementati determinati codici per applicazioni varie e Web App.
  2. ***Fornitura***: *compilazione di un vero e proprio piano di progetto* → Nel nostro caso, il piano di progetto adattato è stato quello del modello a spirale.
  3. ***Sviluppo***: *Realizzazione effettiva del software* → Scrittura ed implementazione di codici JS, PHP, html, CSS e scripts in C#.
  4. ***Uso:*** *Installazione ed utilizzo del software* → Utilizzo, anche solo di un prototipo funzionante, degli algoritmi e codici realizzati dopo l'opportuna installazione.
  5. ***Manutenzione:*** *Correzione di eventuali errori ed aumento delle prestazioni* → Fase di *debugging* che garantisce il controllo ed il monitoraggio delle debolezze di tutti i software.
  6. ***Dismissione:*** *Disinstallazione o sostituzione del prodotto software* → Le potenziali migliorie da apportare al progetto con la realizzazione continua di nuovi prototipi porterà inevitabilmente ad una vera e propria sostituzione dei codici originali.

- ISO **9126** → Utilizzo di una serie di metriche atte a valutare il grado di qualità dei software sviluppati. La verifica è basata sulla seguente tabella:



Per la valutazione di ognuna di queste qualità è stata utilizzata una **metrica**: strumenti che forniscono misure quantitative del prodotto e delle sue componenti.
Tipologie di metriche adottate nei software di realizzazione del triathlon Online:

- **Dimensionali** → Misurazione della dimensione del software grazie alla tecnica LOC (*Lines Of Code*), con un risultato che è fortemente legato allo stile di programmazione dello sviluppatore;
- **Strutturali** → Misurazione della qualità del singolo software in base ai suoi punti decisionali, che sono rappresentati dai costrutti e dai cicli presenti al suo interno (*IF, ELSE, FOR, SWITCH…*);
- **Funzionali** → Individuazione di quali sono le vere e proprie funzionalità del software, con lo scopo di poter inquadrare nel modo più preciso possibile il bacino di utenza al quale è rivolta la soluzione proposta.

Possibili implementazioni future:

Non è da trascurare il fatto che, essendo di fronte ad un prototipo, in futuro possano essere implementate ulteriori funzionalità atte a migliorare l'efficienza, la sicurezza e, più in generale, tutte quelle che potrebbero essere le debolezze del sistema creato. Vediamone alcune:

- Uso dell'**intelligenza artificiale**: come è stato riportato, per la rilevazione del colore, dei contorni e del centro del pallone e degli ostacoli si è optato per l'utilizzo delle librerie *OpenCV*, ma sarebbe comunque possibile sostituire l'uso di questi algoritmi con l'intelligenza artificiale, che permetterebbe una riduzione del margine di errore sui vari riconoscimenti dalla fotocamera;
- Uso di una **fotocamera 3D**: per testare il prototipo è bastato l'utilizzo di una semplice WebCam integrata all'interno del PC sul quale è stata sviluppata l'applicazione, ma per garantire il perfetto svolgimento delle tre discipline sarà essenziale adottare una telecamera a tre dimensioni che invii, oltre ai valori **x** e **y** (larghezza e altezza), il valore **z** (profondità);
- Migliorie sull'**audio 3D**: è possibile che l'audio 3D, implementabile tramite funzioni di *Unity*, sia ulteriormente migliorabile nel corso del tempo. Grazie al settaggio dei valori del suono e ad un'**ulteriore comunicazione con il cliente** ipovedente efficace, infatti, sono sempre attuabili upgrade che possano rendere l'applicazione sempre più precisa.
- Uso della funzione **htmlspecialchars**: uno degli attacchi maggiormente diffusi in rete è l'**XSS** (Cross-Site Scripting), un attacco lato server in cui un utente malintenzionato (detto **cracker**) inserisce codici script malevoli. In questo caso ci si potrebbe avvalere dell' "*htmlspecialchars*", una funzione che permette di effettuare controlli all'interno di tutti i campi di input dell'html, convertendo tutti i caratteri *a rischio* (ad esempio i caratteri di apertura e chiusura di un tag: **<>**).
- Uso della funzione **filter_val**: per assicurarsi che l'utente, nei diversi campi di input presenti all'interno della pagina html, inserisca valori validi è possibile avvalersi della funzione di PHP "*filter_val*", da mettere, ad esempio, per il monitoraggio degli array associativi (**$_GET** o **$_POST**) che si formeranno all'interno della porzione di back-end del sito.
- Utilizzo dei **cookie di sessione**: un altro aspetto ulteriormente migliorabile all'interno dell'applicazione web è l'uso corretto dei cookie di sessione, per evitare, ad esempio, di far mettere nuovamente le credenziali ad un utente che ha chiuso il sito da poco tempo.
- Implementazioni di altre misure di sicurezza come i **firewall**: all'interno dell'interfaccia che mostra l'interconnessione tra le scuole nella città, è possibile aggiungere un firewall, un ulteriore elemento di difesa perimetrale alla rete, in grado di svolgere delle vere e proprie funzioni di collegamento tra due o più segmenti di rete. Va inoltre fatto presente che, in caso di attacco, il firewall avrebbe un'altissima probabilità di essere il primo componente della rete a difenderci e ad essere attaccato (sistema *a strati*).
- Implementazioni dei **sensori** di rilevazione: per avere precisione sui tempi delle gare di velocità, di corse a ostacoli e sui goal sbagliati o realizzati nei rigori, è auspicabile inviare questi dati in tempo reale al database con assoluta precisione grazie all'installazione di sensori vari (cronometri, rilevatori all'interno della porta da calcio…).

Note Aggiuntive:

Esecuzione e verifica dei codici che compongono il progetto e link di collegamento.

1. Parte di "Application":
   - Codice Python: "*main.py*" --> download open CV di Python ed esecuzione del codice dal prompt dei comandi (CMD).

- Link per prendere visione del codice Python: https://www.dropbox.com/s/z7yt9w6c19p388q/main.py?dl=0
- Codici C#: download "*Unity*" (versione utilizzata: 2019) con la possibilità di eseguire i vari script che compongono l'applicazione.
- Link per prendere visione degli script dell'applicazione del calcio di rigore: https://www.dropbox.com/sh/t65830c7uycqs0v/AAB9pASU-34IA9pyv1KH_F2Va?dl=0
- Link per prendere visione degli script dell'applicazione della gara di velocità e della corsa a ostacoli: https://www.dropbox.com/sh/he56wr8nbxjp6nm/AABHfRMh-VRnQR7e1c1FeDZTa?dl=0

2. Web App:
   - Sviluppata su localhost --> esecuzione possibile nella directory "htdocs" di "xampp".
   - Link per prendere visione dei codici che compongono l'intero sito: https://www.dropbox.com/sh/19hum4goyeekrg3/AABKenHIDFwGTDt1UaL3pR7-a?dl=0
   - Link per prendere visione del database scritto in MySQL: https://www.dropbox.com/s/c7q3go7jtcie8tn/DB_Olimpiadi.sql?dl=0
   - Credenziali per effettuare il login nel sito:
     - Come studente: **USERNAME**: ABCD1 **PASSWORD**: PassProva01
     - Come scuola: **USERNAME**: ASD123 **PASSWORD**: PassProva02

3. Interfaccia scuole:
   - Possibilità di verificare la buona riuscita del viaggio di pacchetti in rete con le operazioni di *ping* tra i vari terminali che compongono la Smart City.
   - Link per prendere visione del file Packet Tracer: https://www.dropbox.com/s/zby3zrbdwltgeui/city.pkt?dl=0