Documentazione Progetto  
 Astronomic Picture of the Day

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

1.3 Scopo 3

2 Analisi 4

2.1 Analisi del dominio 4

2.2 Analisi e specifica dei requisiti 4

2.2.1 Spiegazione elementi tabella dei requisiti: 6

2.3 Use case 6

2.4 Pianificazione 7

2.5 Analisi dei mezzi 7

2.5.1 Software 7

2.5.2 Hardware 7

3 Progettazione 8

3.1 Design dell’architettura del sistema 8

3.2 Design dei dati e database 8

3.3 Design delle interfacce 9

3.3.1 Interfaccia Login 9

3.3.2 Interfaccia Home page 9

3.3.3 Interfaccia Preferiti e Cronologia 10

3.4 Design procedurale 10

4 Implementazione 11

4.1 Database 11

4.1.1 Utente 11

4.1.2 Preferiti 11

4.1.3 Cronologia 12

4.2 UML 13

4.2.1 Struttura cartelle 13

4.3 Codice 14

4.3.1 File utils.php 14

4.3.2 Connessione al Database 14

4.3.3 Login 14

4.3.4 Registrazione 15

4.3.5 Logout 18

4.3.6 Mettere una foto nei preferiti 18

4.3.7 Mettere una foto dalla cronologia 18

4.3.8 Rimuovere una foto dalla cronologia e dai preferiti 18

4.3.9 Creazione tabella Preferito e Cronologia 18

favorite.php 18

history.php 18

4.4 Sicurezza 19

4.5 Design delle interfacce 19

5 Test 19

5.1 Protocollo di test 19

5.2 Risultati test 20

5.3 Mancanze/limitazioni conosciute 20

6 Consuntivo 20

7 Conclusioni 20

7.1 Sviluppi futuri 20

7.2 Considerazioni personali 20

8 Glossario 20

9 Bibliografia 21

9.1 Bibliografia per articoli di riviste: 21

9.2 Bibliografia per libri 21

9.3 Sitografia 21

10 Allegati 21

# Introduzione

## Informazioni sul progetto

## Abstract

Il progetto vuole creare una pagina Web dove l’utente può vedere la foto astronomica del giorno, o di un giorno a sua scelta, i crediti, una descrizione dell’immagine, la possibilità di scaricare l’immagine e la cronologia di ricerca.

Può contenere alcuni o tutti gli elementi seguenti:

* **Background/Situazione iniziale**
* **Descrizione del problema e motivazione**: Che problema ho cercato di risolvere? Questa sezione dovrebbe includere l'importanza del vostro lavoro, la difficoltà dell'area e l'effetto che potrebbe avere se portato a termine con successo.
* **Approccio/Metodi**: Come ho ottenuto dei progressi? Come ho risolto il problema (tecniche…)? Quale è stata l’entità del mio lavoro? Che fattori importanti controllo, ignoro o misuro?
* **Risultati**: Quale è la risposta? Quali sono i risultati? Quanto è più veloce, più sicuro, più economico o in qualche altro aspetto migliore di altri prodotti/soluzioni?

## Scopo

Lo scopo didattico del progetto è riuscire a gestire nel modo più ottimale possibile un progetto IT con le risorse datoci dalla scuola. Questo servirà successivamente a prepararmi a futuri progetti e all’esame pratico di fine scuola.

Lo scopo operativo invece riguarda il creare un applicativo Web dove si possano visualizzare le *Astronomic Picture of the Day*, ovvero le foto astronomiche del giorno fatte dalla Nasa, con tutte le informazioni a riguardo. Dovrà esserci la possibilità di ricercare la foto di una specifica data e salvarla. Si dovrà anche visualizzare la cronologia delle immagini ricercate e le foto preferite dell’utente.

# Analisi

## Analisi del dominio

L’Applicativo potrà venir utilizzato da chiunque, permettendo però anche di eseguire il login così da vedere il proprio account, su qualunque Computer o dispositivo mobile. Principalmente si vuole semplificare la visione e la ricerca della Astronomical Picture of the Day (APOD). Attualmente esiste già un sito che fa questo lavoro, però non c’è la possibilità di scegliere l’APOD da vedere senza andare sul sito apposito con tutte le APOD scattate in ordine cronologico e non si può salvare le proprie foto preferite.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-1** | |
| **Nome** | Login |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Sotto requisiti** | |
| **001** | Creare un Database per salvare gli utenti per il sito |

|  |  |
| --- | --- |
| **ID: REQ-2** | |
| **Nome** | Registrazione |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Sotto requisiti** | |
| **001** | Controllare che non esista già un utente con lo stesso nome |
| **002** | Convalidare i dati e proteggersi da SQLInjection |

|  |  |
| --- | --- |
| **ID: REQ-3** | |
| **Nome** | API |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Bisogna utilizzare l’API APOD della Nasa per ricavare i dati delle foto |

|  |  |
| --- | --- |
| **ID: REQ-4** | |
| **Nome** | Cronologia |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Sotto requisiti** | |
| **001** | Aggiungere le foto con le relative informazioni nel DB, associandole all’Id dell’utente |
| **002** | Eliminare le foto in automatico dopo aver raggiunto un certo numero (Max 20 per utente) |
| **003** | Permettere all’utene di eliminare le foto dalla cronologia |

|  |  |
| --- | --- |
| **ID: REQ-5** | |
| **Nome** | Preferiti |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Sotto requisiti** | |
| **001** | Aggiungere le foto con le relative informazioni nel DB, associandole all’Id dell’utente |
| **002** | Permettere di eliminare le foto dal DB |

|  |  |
| --- | --- |
| **ID: REQ-6** | |
| **Nome** | Filtraggio |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Permettere di cercare una foto per data |

|  |  |
| --- | --- |
| **ID: REQ-7** | |
| **Nome** | Sequenza immagini |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Nella home page si vede la foto del giorno desiderato, a sinistra del giorno precedente e a destra del giorno seguente, se non disponibile esce una foto di default |

### Spiegazione elementi tabella dei requisiti:

**ID**: identificativo univoco del requisito

**Nome**: breve descrizione del requisito

**Priorità**: indica l’importanza di un requisito nell’insieme del progetto, definita assieme al committente. Ad esempio, poter disporre di report con colonne di colori diversi ha priorità minore rispetto al fatto di avere un database con gli elementi al suo interno. Solitamente si definiscono al massimo di 2-3 livelli di priorità.

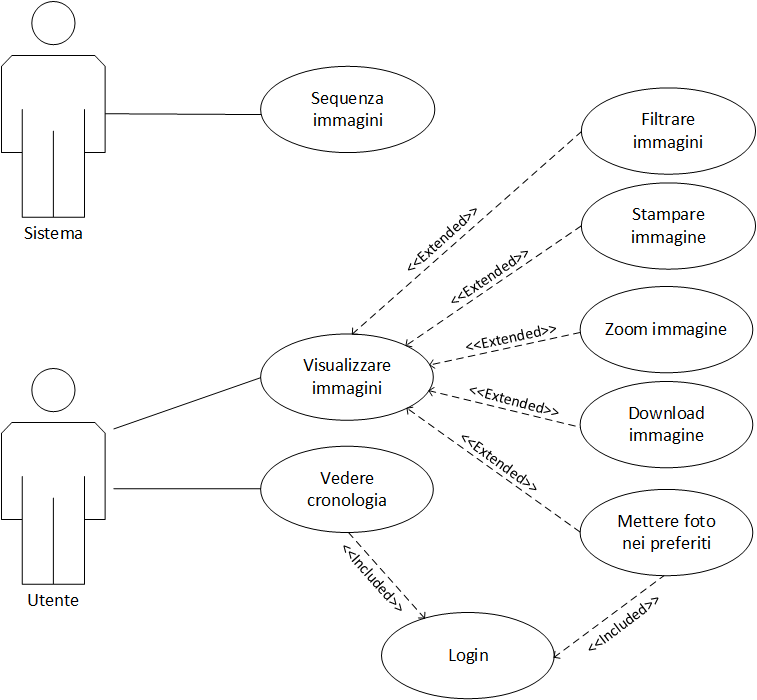
**Versione**: indica la versione del requisito. Ogni modifica del requisito avrà una versione aggiornata.

Sulla documentazione apparirà solamente l’ultima versione, mentre le vecchie dovranno essere inserite nei diari.

**Note**: eventuali osservazioni importanti o riferimenti ad altri requisiti.

**Sotto requisiti**: elementi che compongono il requisito.

## Use case



## Pianificazione

## Analisi dei mezzi

### Software

Per questo progetto avrò bisogno dell’API APOD della Nasa, MySQL e Apache (XAMPP), Project per il gantt e Visual studio Code per la parte di programmazione.

### Hardware

Per questo progetto avrò bisogno del PC di scuola e un Server dove mettere il Database.

# Progettazione

Questo capitolo descrive esaustivamente come deve essere realizzato il prodotto fin nei suoi dettagli. Una buona progettazione permette all’esecutore di evitare fraintendimenti e imprecisioni nell’implementazione del prodotto.

## Design dell’architettura del sistema

Descrive:

* La struttura del programma/sistema lo schema di rete...
* Gli oggetti/moduli/componenti che lo compongono.
* I flussi di informazione in ingresso ed in uscita e le relative elaborazioni. Può utilizzare *diagrammi di flusso dei dati* (DFD).
* Eventuale sitemap

## Design dei dati e database

|  |
| --- |
|  |
| C:\Users\kamil.siddiqui\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\DFE556C.tmp |

## Design delle interfacce

### Interfaccia Login

Questa sarà la prima pagina che l’utente vedrà quando apre il sito, se accede potrà vedere anche la pagina preferiti e cronologia, altrimenti avrà solo accesso alla home page.

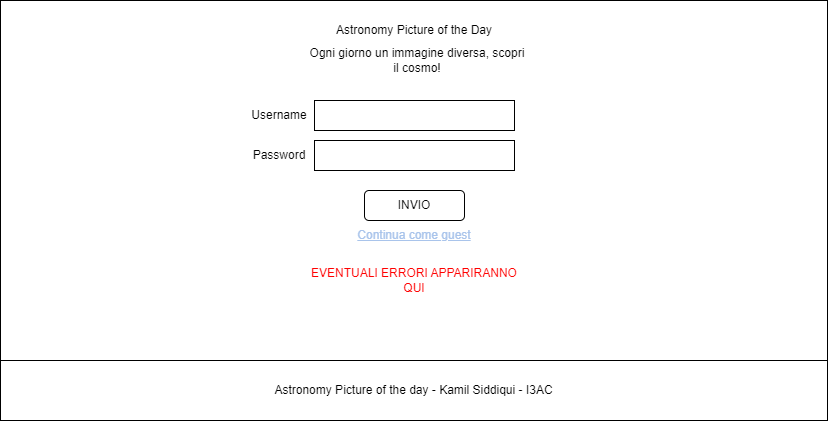


Figura 1: Pianificazione della GUI del Login

### Interfaccia Home page

Questa sarà la home page, da cui si potrà vedere l’immagine del giorno o quella scelta dall’utente, e se disponibili quella del giorno precedente e successivo.

Da questa pagina si potrà andare alla pagina di login premendo il menu a scomparsa in alto a sinistra.  
Da qui si aprirà un menu che permetterà anche di andare alla pagina dei preferiti o della cronologia

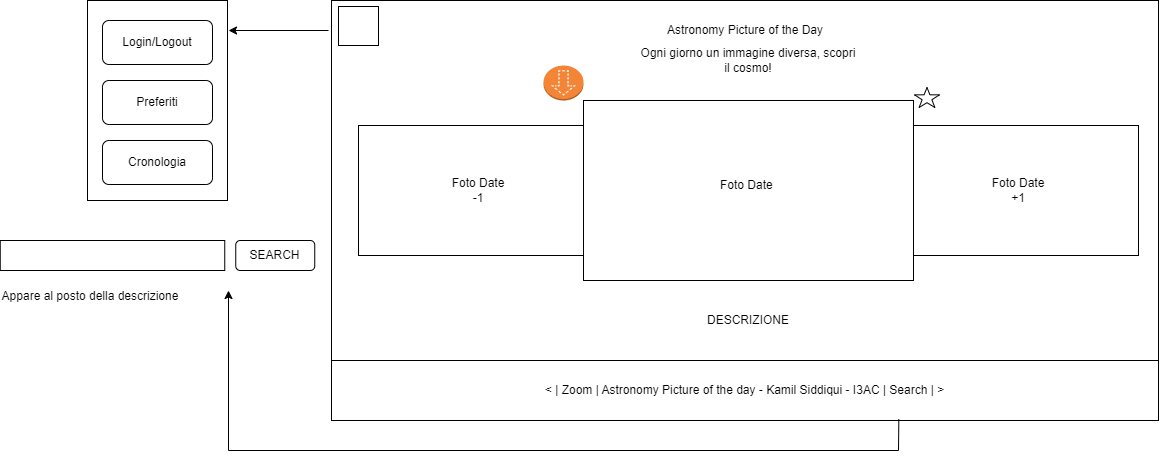


Figura 2: Pianificazione della GUI della Home page

### Interfaccia Preferiti e Cronologia

Questa pagina mostrerà le foto preferite dell’utente oppure le ultime 10 foto visualizzate dall’utente, da qui si potrà sempre andare alla home page,alla pagina preferiti o cronologia ed effettuare il logout, essendo questa una pagina solo per utenti loggati.

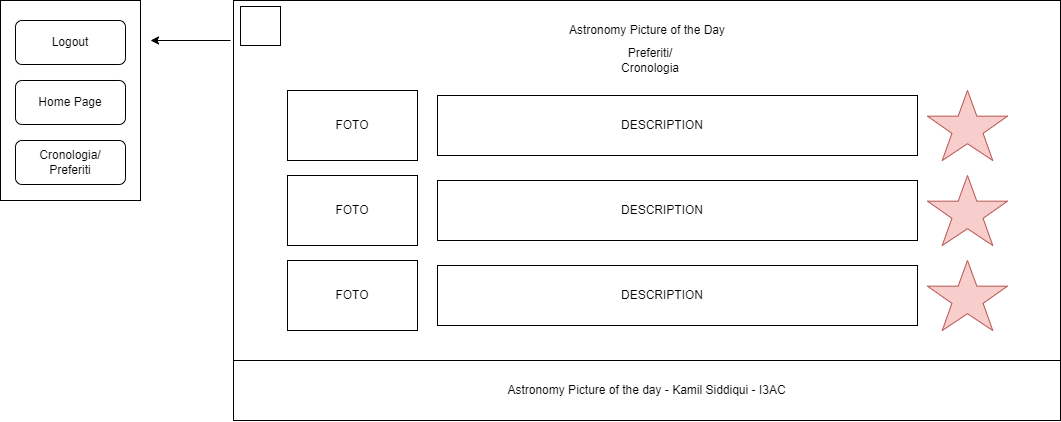
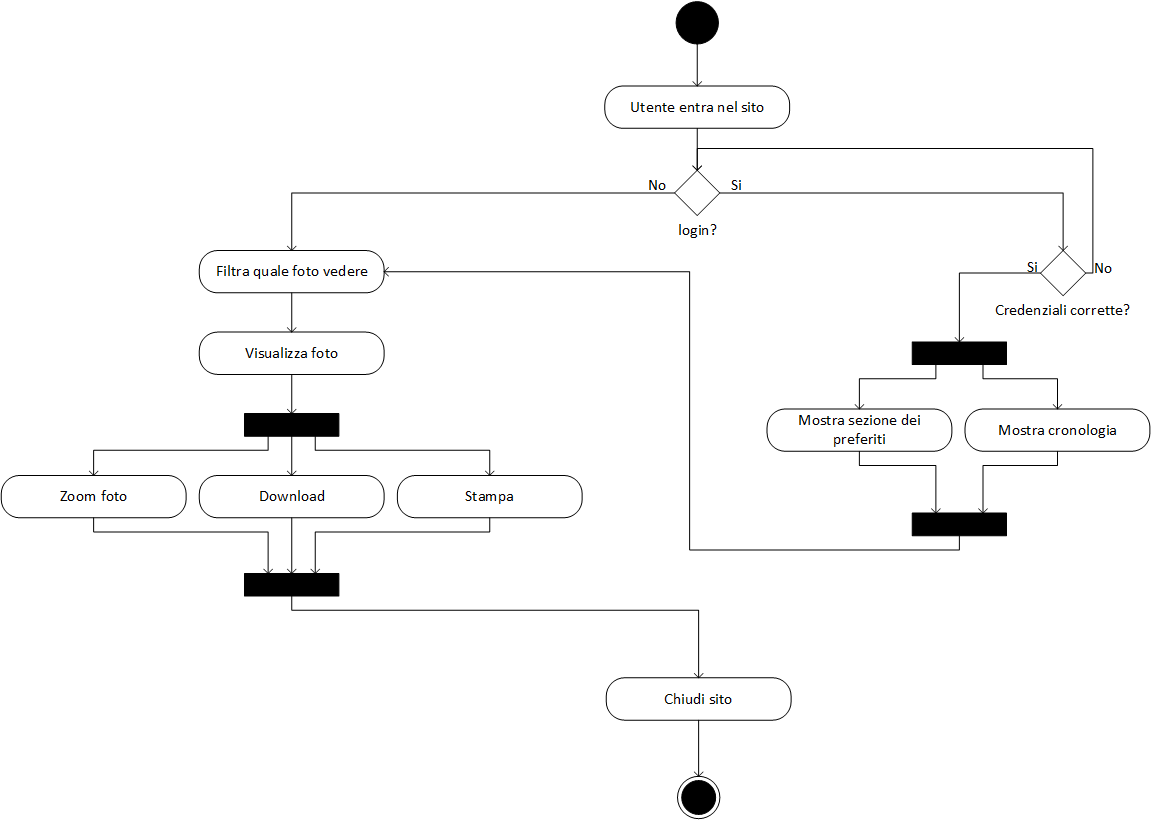


Figura 3: Pianificazione della page dei Preferiti e cronologia

## Design procedurale



# Implementazione

## Database

### Utente

|  |
| --- |
|  |

Il sito viene utilizzato dagli utenti, questi devono poter vedere le loro foto preferite e le foto viste recentemente all’interno del sito, per permettere questo ho create la tabella utente, in modo che ogni utente possa avere un account personale. Per fare il login utilizzo un Username e una password che verrà hashata usando MD5, tutti gli utenti vengono aggiunti direttamente dalla pagina di registrazione di PHP. ID viene usato come key e verrà anche usato come foreign key per le altre tabelle del Database.

### Preferiti

|  |
| --- |
|  |

La tabella “Preferito” conterrà le informazioni importanti sulle foto che l’utente metterà tra i preferiti.

Data e Utente\_Id sono le chiavi essendo che un utente può avere più foto preferite diverse, e la stessa foto può essere la preferità di più utenti.

Url serve per salvare l’URL della foto e mettere un’anteprima della foto.  
Titolo e descrizione servono per mettere qualche informazione nella pagina dei preferiti.   
Date, oltre che venir usato come chiave, serve per fare un eventuale richiesta all’API e avere tutte le informazioni sull’immagine.

### Cronologia

|  |
| --- |
|  |

La tabella “Cronologia” conterrà le informazioni importanti sulle ultime 15/20 foto viste dall’utente. A differenza della tabella “Preferito”, questa come chiave usa Id e Utente\_Id, questo perché a differenza di “Preferito”, se superi il numero massimo di foto, sarà il sistema a eliminare la foto più vecchia vista (quella con l’ID più basso).

Url serve per salvare l’URL della foto e mettere un’anteprima della foto.   
Titolo e descrizione servono per mettere qualche informazione nella pagina dei preferiti.   
Date serve per fare un eventuale richiesta all’API e avere tutte le informazioni sull’immagine.

## UML

### Struttura cartelle

|  |
| --- |
| * APOD\_API-2024-2025   + 1\_Qdc   + 2\_Abstract   + 3\_Documentazione (word e pdf)   + 4\_Diari     - 12. Dicembre     - 11. Novembre     - 10. Ottobre     - 9. Settembre   + 5\_Applicativo     - css     - img     - js   + 7\_Allegati     - DB     - Pianificazione\_Pagine   + 8\_Manuali |

Questa è la strutta delle cartelle. Inizialmente c’è la cartella del progetto **APOD\_API-2024-2025**, questa cartella contiene 7 cartelle, **1\_Qdc** che contiene il quaderno dei compiti. **2\_Abstract** contiene l’abstract del progetto. **3\_Documentazione (word e pdf)** contiene la documentazione del progetto in formato .docx e .pdf.  
**4\_Diari** contiene una cartella per ogni mese, dentro di esse ci sono i relativi diari di quel mese. **5\_Applicativo** contiene delle sottocartelle, css contiene tutti i file di stile per le pagine php, img contiene tutte le immagini usate nel sito, come la favicon, lo sfondo e l’immagine che uso se manca l’url di una foto e js contiene due file di tipo javascript, uno che gestise il layout della pagina html e uno che gestisce le richieste all’api.

## Codice

### File utils.php

|  |
| --- |
| CODICE QUA |

DESCRIZIONE QUA.

### Connessione al Database

|  |
| --- |
| CODICE QUA |

DESCRIZIONE QUA.

### Login

Per effettuare il login, bisogna compilare il form presente in *index.php,* che poi passerà tutti i dati in post a *login.*php, che farà tutti i controlli per effettuare il login.

|  |
| --- |
| <form action="login.php" method="post">  <label>Username</label>      <input type="text" placeholder="Enter Username" name="uname">      <br><br>      <label>Password</label>      <input type="password" placeholder="Enter Password" name="password">      <br><br>  <!-- QUESTA SEZIONE SERVE PER UN EVENTUALE DISPLAY DEGLI ERRORI -->  <?php if(isset($\_GET['error'])) { ?>  <p class="error"> <?php echo $\_GET['error'];?></p>  <?php } ?>  <button id="btnLogin" type="submit">Login</button><br><br>  </form> |

Ora in *login.php* è necessario iniziare una sessione e includere *db\_conn.php.*

Creo un form che richiede un username e una password.chiamo la funzione *convalida($data)* da *utils.php* che pulisce il dato in entrata, così da evitare attacchi malevoli, controllo anche che username e password non siano lasciati vuoti

|  |
| --- |
| session\_start();  include "db\_conn.php";  include "utils.php";  $username = convalida($\_POST['uname']);  $pass = convalida($\_POST['password']);  //CONTROLLO CHE USERNAME E PASSWORD NON SIANO VUOTI; IN CASO DISPLAY UN ERRORE  if(empty($username)){      header("Location: index.php?error=Username è richiesto");      exit();  }  else if(empty($pass)){      header("Location: index.php?error=Password è richiesta");      exit();  }  … |

Dopo questi controlli, faccio un hash della password in MD5, poi eseguo la query per verificare che l’utente esista e che la password sia giusta.  
Se è giusta porto l’utente alla home page *home.php*, altrimenti ritorno un errore nella sezion creata in *index.php*.

|  |
| --- |
| $pass = md5($\_POST['password']);  $sql = "SELECT \* FROM utente WHERE Username='$username' AND  Password='$pass'"; //QUERY DA FARE AL DATABASE  $result = mysqli\_query($conn, $sql);  if(mysqli\_num\_rows($result) == 1) {      $row = mysqli\_fetch\_assoc($result);      if($row['Username'] === $username){ //Se username corretto          echo "Login effetuato con successo!";          $\_SESSION['Username'] = $row['Username'];          $\_SESSION['Password'] = $row['Password'];          $\_SESSION['Id'] = $row['Id'];          header("Location: home.php");          exit();      }      else{          header("Location: index.php?error=Password errata ");          exit();      }  }  else{      header("Location: index.php?error=Username o password errata");      exit();  } |

### Registrazione

Nella pagina *index.php*, c’è anche la possibilità di registrarsi, premendo un button presente nella pagina verremo portati ad un'altra pagina, *register.php.*

|  |
| --- |
| *Index.php*  …  <form action="register.php" method="post">  <button id="btnRegister" type="submit">Crea un account</button><br><br>  </form>  … |

La pagina *register.php* gestisce sia la parte HTML della pagina, contenendo un form che manderà i dati in post, e contenendo anche il codice PHP per convalidare i dati e aggiungere l’utente al DB.

Per essere sicuro che l’utene non sbagli password, deve inserirla due volte.

|  |
| --- |
| <form action="register.php" method="post">      <label>Username</label>      <input type="text" placeholder="Enter Username" name="uname">      <br><br>      <label>Password</label>      <input type="password" placeholder="Enter Password" name="password">      <br><br>      <label>Ripeti password</label>      <input type="password" placeholder="Enter Password again" name="repeatP">      <br><br>        <!-- QUESTA SEZIONE SERVE PER UN EVENTUALE DISPLAY DEGLI ERRORI -->      <?php if(isset($\_GET['error'])) { ?>          <p class="error"> <?php echo $\_GET['error'];?></p>      <?php } ?>      <button id="btnRegister" type="submit">Login</button><br><br>  </form> |

Inviato il form, nella stessa pagina, apro il tag php, avvio una sessione e includo i file *db\_conn.php* e *utils.php* che contiene la funzione convalida($data) per pulire i dati in entrata.  
Verifico che username e le due password non siano vuote e che le due password combacino.

|  |
| --- |
| session\_start();  include "db\_conn.php";  include 'utils.php';  $username = convalida($\_POST['uname']);  $pass = convalida($\_POST['password']);  $rPass = convalida($\_POST['repeatP']);  //CONTROLLO CHE USERNAME E PASSWORD NON SIANO VUOTI; IN CASO DISPLAY UN ERRORE  if(empty($username)){      header("Location: register.php?error=Username è richiesto");      exit();  }  else if(empty($pass) || empty($rPass)){      header("Location: register.php?error=Password è richiesta");      exit();  }  else if($pass!=$rPass){      header("Location: register.php?error=Password non coincidono");      exit();  } |

Dopo questi controlli, se passati con successo, faccio una query per verificare che questo utente non sia già presente nel DB, se non è presente allora faccio la query per inserire l’utente nel DB, altrimenti restituisco un errore.

|  |
| --- |
| if($stmt = $conn->prepare('SELECT Id, Password FROM utente WHERE Username =?')){      $stmt->bind\_param('s', $\_POST['uname']);      $stmt->execute();      $stmt->store\_result();      if($stmt->num\_rows>0){          header("Location: register.php?error=Username già esistente");          exit();      }      else{          if($stmt = $conn->prepare('INSERT INTO utente (Username, Password) VALUE(?, ?)')){              $password = md5($\_POST['password']);              $stmt->bind\_param('ss', $username, $password);              $stmt->execute();  … |

Dopo aver aggiunto l’utente al DB, per evitare di dovergli far mettere di nuovo le credenziali per fare il login, direttamente da qui facciamo un login. Facciamo quindi una query, così da avere anche l’Id dell’utente.

|  |
| --- |
| $sql = "SELECT \* FROM utente WHERE Username='$username' AND  Password='$password'"; //QUERY DA FARE AL DATABASE              $result = mysqli\_query($conn, $sql);              $row = mysqli\_fetch\_assoc($result);              echo "Register effetuato con successo!";              $\_SESSION['Username'] = $row['Username'];              $\_SESSION['Password'] = $row['Password'];              $\_SESSION['Id'] = $row['Id'];              header("Location: home.php");              exit();          }      }      $stmt->close();  }  else{      header("Location: register.php?=error= ERRORE");      exit();  } |

Così facendo arriviamo direttamente alla *home.php* loggati.

### Logout

Per effettuare il logout, da qualunque pagina bisogna aprire il menu e premere il button con scritto logout, partirà questo codice che libererà tutte le variabili e distruggera le sessioni, riportando l’utente a *index.php*, pagina di login.

|  |
| --- |
| session\_unset();  //libera tutte le variabili di sessione attualmente registrate.  session\_destroy();  header("Location: index.php"); |

### Mettere una foto nei preferiti

|  |
| --- |
| CODICE QUA |

DESCRIZIONE QUA.

### Mettere una foto dalla cronologia

|  |
| --- |
| CODICE QUA |

DESCRIZIONE QUA.

### Rimuovere una foto dalla cronologia e dai preferiti

|  |
| --- |
| CODICE QUA |

DESCRIZIONE QUA.

### Creazione tabella Preferito e Cronologia

Per visualizzare con più facilità le foto e le loro descrizioni, ho deciso di creare una tabella che conterrà tutti i dati, ed essendo che entrambe le pagine hanno lo stesso formato di tabella, ho usato lo stesso codice.

Come prima cosa prendo i dati dalla tabella.

|  |
| --- |
| favorite.php include "db\_conn.php";      $id = $\_SESSION['Id'];  $sql = "SELECT \* FROM preferito WHERE Utente\_Id = '$id'"; //QUERY DATABASE      $result = mysqli\_query($conn, $sql);  … |
| history.php include "db\_conn.php";      $id = $\_SESSION['Id'];  $sql = "SELECT \* FROM cronologia WHERE Utente\_Id = '$id'"; //QUERY DATABASE      $result = mysqli\_query($conn, $sql); |

Dopodiché, creo delle stringhe per le regole css e poi creo la stringa per la prima riga della tabella, in questo caso per entrambi i file è uguale.

|  |
| --- |
| $css = "style='border: 1px solid white;border-collapse: collapse;padding:10px;'";  $cssWider = "style='border: 1px solid white;border-collapse: collapse; padding:10px; width:400px;'";  $cssImg = "style='border: 1px solid white;border-collapse: collapse;'";  $table ="<center><table ".$css.">          <tr>                <td ".$css.">Data</td>                  <td ".$css.">Immagine</td>                  <td ".$css.">Titolo</td>                  <td ".$css.">Descrizione</td>                  <td ".$css.">Delete</td>              </tr>"; |

Ora popoliamo la tabella, prima controlliamo se la query ha restituito un valore, altrimenti stampiamo solamente “Nessuna foto preferita” nel caso di *favorite.php* e “Nessuna foto visualizzata” nel caso di *history.php*, se ha restituito un valore popoliamo la tabella con un ciclo while.

Nel ciclo if a riga 7, verifico se l’url appartiene a una immagine, che contiene sempre quel prefisso, oppure un video, così so se metere un tag iframe o img nella tabella.

|  |
| --- |
| if(mysqli\_num\_rows($result) > 0) {  while ($row = $result->fetch\_array(MYSQLI\_ASSOC)) {  $date = $row['Data'];  $url = $row['url'];  $titolo = $row['Titolo'];  $desc = $row['Descrizione'];  if(str\_contains($url, "apod.nasa.gov")){          $media = "<img src=".$url."style='width:auto;height:auto;border:none;'>";          }          else{          $media = "<iframe src=".$url."style='width:400px;height:250px;border:none;'></iframe>";  }          $table = $table."<tr>  <th ".$css.">".$date."</th>  <th ".$cssImg.">".$media."</th>              <th ".$css.">".$titolo."</th>              <th ".$cssWider.">".$desc."... <br><a href='home.php'>See more</a></th>  <th ".$css.">  <form action='removePreferiti.php' method='post'>  //Nel caso di history.php sarà removeCronologia.php  <input type='text' id='dataInput' name='dataInput' value=".$date." style='visibility:hidden;'><br>                      <button id='button\_remover' type='submit'>Rimuovi dai preferiti</button><br><br>  </form>  </th>  </tr>";  } //Chiusura while      $table=$table."</table></center>";      echo $table;  } //Chiusura if num\_row > 0  else{  echo "<hr><br><br><center><h1>Nessuna foto preferita!</h1><center><br><br><hr>";  }//Nel caso di history.php sarà Nessuna foto visualizzata! |

Ora abbiamo una tabella con tutte le informazioni necessarie per la foto.

## Sicurezza

## Design delle interfacce

# Test

## Protocollo di test

Definire in modo accurato tutti i test che devono essere realizzati per garantire l’adempimento delle richieste formulate nei requisiti. I test fungono da garanzia di qualità del prodotto. Ogni test deve essere ripetibile alle stesse condizioni.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-012 | **Nome:** | Import a card with KIC, KID and KIK keys, but not shown with the GUI |
| **Descrizione:** | Import a card with KIC, KID and KIK keys with no obfuscation, but not shown with the GUI | | |
| **Prerequisiti:** | Store on local PC: Profile\_1.2.001.xml (appendix n\_n) and Cards\_1.2.001.txt (appendix n\_n).  PIN (OTA\_VIEW\_PIN\_PUK\_KEY) and ADM (OTA\_VIEW\_ADM\_KEY) user right not set. | | |
| **Procedura:** | 1. Go to “Cards manager” menu,  in main page click “Import Profiles” link, Select the “1.2.001.xml” file, Import the Profile 2. Go to “Cards manager” menu,  in main page click “Import Cards” link, Select the “1.2.001.txt” file, Delete the cards,  Select the “1.2.001.txt” file, Import the cards 3. Research the “41795924770” Card, Click the imsi card link Check the card details 4. Execute the SQL: SELECT imsi, dir, keyset, cntr, rawtohex(kickey), rawtohex(kidkey), rawtohex(kikkey), rawtohex(chv), rawtohex(dap)FROM otacardkey a where imsi='340041795924770' ORDER BY keyset; | | |
| **Risultati attesi:** | Keys visible in the DB (OtaCardKey) but not visible in the GUI (Card details) | | |

## Risultati test

Tabella riassuntiva in cui si inseriscono i test riusciti e non del prodotto finale. Se un test non riesce e viene corretto l’errore, questo dovrà risultare nel documento finale come riuscito (la procedura della correzione apparirà nel diario), altrimenti dovrà essere descritto l’errore con eventuali ipotesi di correzione.

## Mancanze/limitazioni conosciute

Descrizione con motivazione di eventuali elementi mancanti o non completamente implementati, al di fuori dei test case. Non devono essere riportati gli errori e i problemi riscontrati e poi risolti durante il progetto.

# Consuntivo

Consuntivo del tempo di lavoro effettivo e considerazioni riguardo le differenze rispetto alla pianificazione (cap. 1.7) (ad esempio Gantt consuntivo).

# Conclusioni

Quali sono le implicazioni della mia soluzione? Che impatto avrà? Cambierà il mondo? È un successo importante? È solo un’aggiunta marginale o è semplicemente servita per scoprire che questo percorso è stato una perdita di tempo? I risultati ottenuti sono generali, facilmente generalizzabili o sono specifici di un caso particolare? ecc.

## Sviluppi futuri

Migliorie o estensioni che possono essere sviluppate sul prodotto.

## Considerazioni personali

Cosa ho imparato in questo progetto? ecc.

# Glossario

Inserite una semplice tabella con due colonne che spieghi i termini specifici del progetto (lista dei termini in ordine alfabetico A-Z)

Esempio:

|  |  |
| --- | --- |
| **Termine** | **Descrizione** |
| AJAX | **Asynchronous JavaScript And XML**: una tecnica che permette di eseguire richieste ed ottenere dati da una pagina web in modo asincrono. |
| CSS | **Cascading Style Sheets**: linguaggio che permette di definire il layout e la grafica di una pagina web. |

# Bibliografia

## Bibliografia per articoli di riviste:

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo dell’articolo (tra virgolette),
3. Titolo della rivista (in italico),
4. Anno e numero
5. Pagina iniziale dell’articolo.

## Bibliografia per libri

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo del libro (in italico),
3. ev. Numero di edizione,
4. Nome dell’editore,
5. Anno di pubblicazione,
6. ISBN.

## Sitografia

1. URL del sito (se troppo lungo solo dominio, evt completo nel diario),
2. Eventuale titolo della pagina (in italico),
3. Data di consultazione (GG-MM-AAAA).

**Esempio:**

* http://standards.ieee.org/guides/style/section7.html, *IEEE Standards Style Manual*, 07-06-2008.

# Allegati

Elenco degli allegati, esempio:

* Diari di lavoro
* Codici sorgente/documentazione macchine virtuali
* Istruzioni di installazione del prodotto (con credenziali di accesso) e/o di eventuali prodotti terzi
* Documentazione di prodotti di terzi
* Eventuali guide utente / Manuali di utilizzo
* Mandato e/o QdC
* Prodotto
* …