Documentazione Progetto  
 Astronomic Picture of the Day

1 Introduzione 4

1.1 Informazioni sul progetto 4

1.2 Abstract 4

1.3 Scopo 4

2 Analisi 5

2.1 Analisi del dominio 5

2.2 Analisi e specifica dei requisiti 5

2.2.1 Spiegazione elementi tabella dei requisiti: 7

2.3 Use case 7

2.4 Pianificazione 8

2.5 Analisi dei mezzi 8

2.5.1 Software 8

2.5.2 Hardware 8

3 Progettazione 9

3.1 Design dell’architettura del sistema 9

3.2 Design dei dati e database 10

3.3 Design delle interfacce 11

3.3.1 Interfaccia Login 11

3.3.2 Interfaccia Home page 11

3.3.3 Interfaccia Preferiti e Cronologia 12

3.4 Design procedurale 12

4 Implementazione 13

4.1 Database 13

4.1.1 Utente 13

4.1.2 Preferiti 13

4.1.3 Cronologia 14

4.2 UML 15

4.2.1 Struttura cartelle 15

4.3 Codice 16

4.3.1 File utils.php 16

4.3.2 Connessione al Database 16

4.3.3 Login 16

4.3.4 Registrazione 18

4.3.5 Logout 20

4.3.6 Mettere una foto nei preferiti 21

4.3.7 Rimuovere una foto dalla cronologia e dai preferiti 23

4.3.8 Creazione tabella Preferito e Cronologia 23

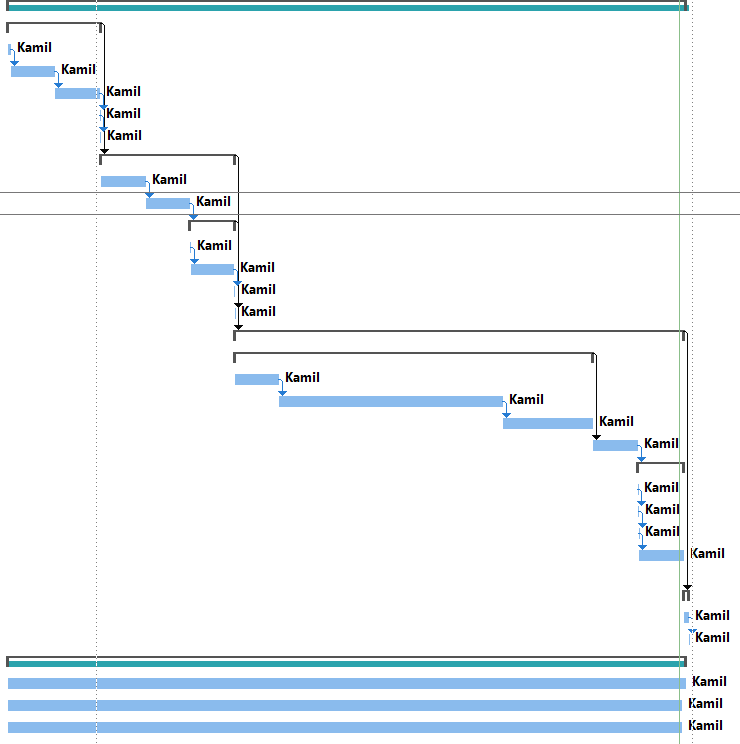
4.3.9 Gestione delle richieste per l’API 25

5 Test 28

5.1 Protocollo di test 28

5.2 Risultati test 33

5.3 Mancanze/limitazioni conosciute 40

6 Consunti 40

7 Conclusioni 41

7.1 Sviluppi futuri 41

7.2 Considerazioni personali 41

8 Glossario 41

9 Indice delle figure 42

10 Bibliografia 43

10.1 Sitografia 43

# 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

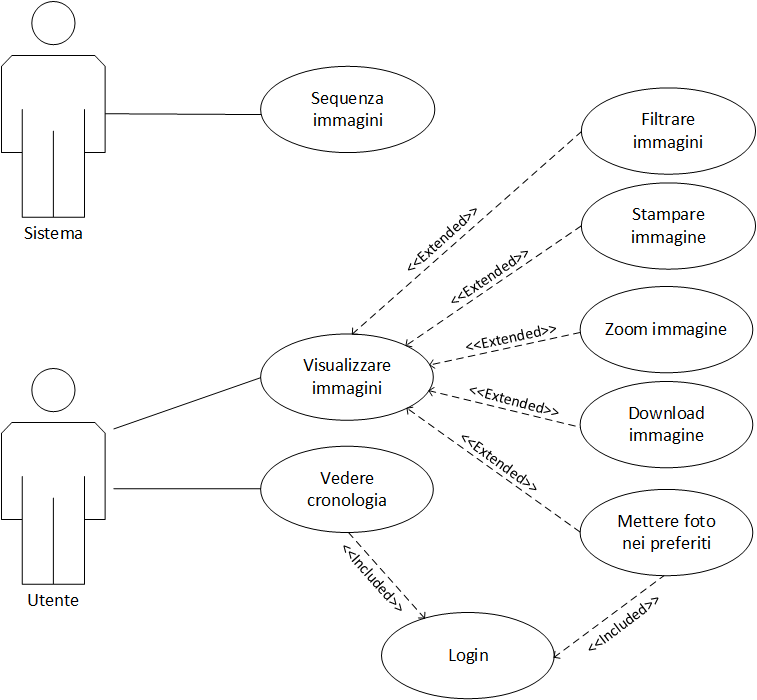


Figura : Use case

## Pianificazione

Figura : Gantt preventivo

## 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

|  |
| --- |
| Figura : Schema E-R |
| C:\Users\kamil.siddiqui\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\DFE556C.tmp  Figura : Schema del Database |

## 

## 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.

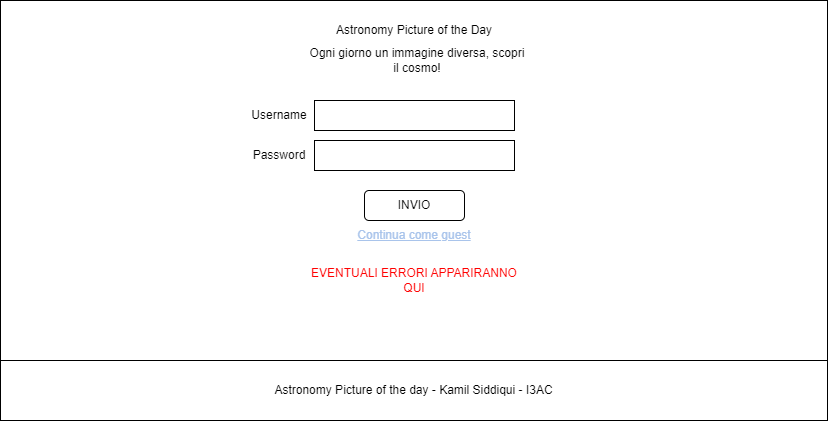


Figura : Pianificazione della GUI del Login

### 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

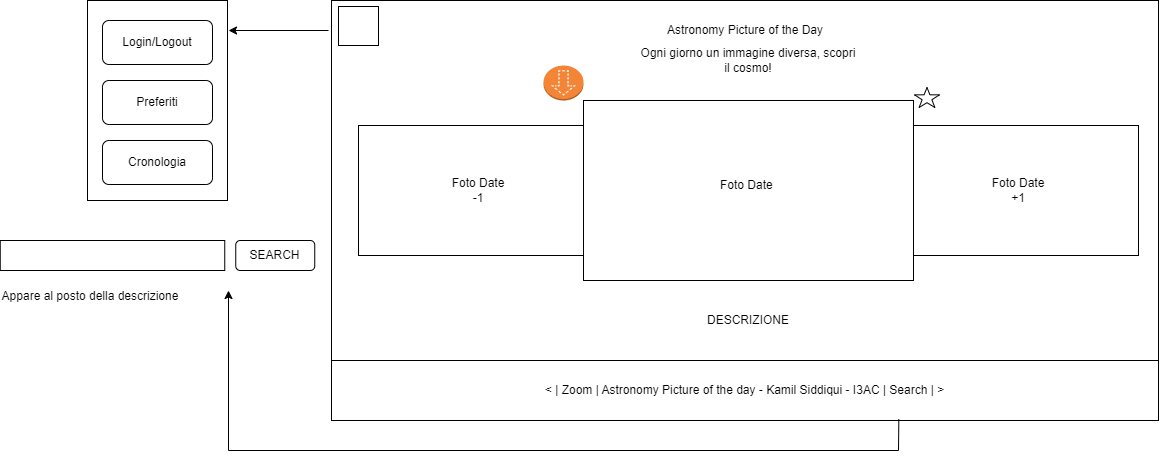


Figura : Pianificazione della GUI della Home page

### 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.

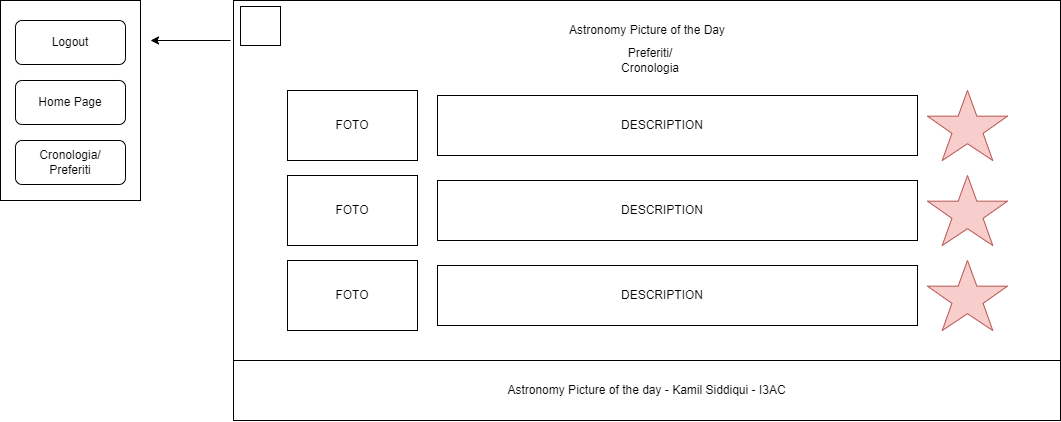


Figura : Pianificazione della page dei Preferiti e cronologia

## Design procedurale

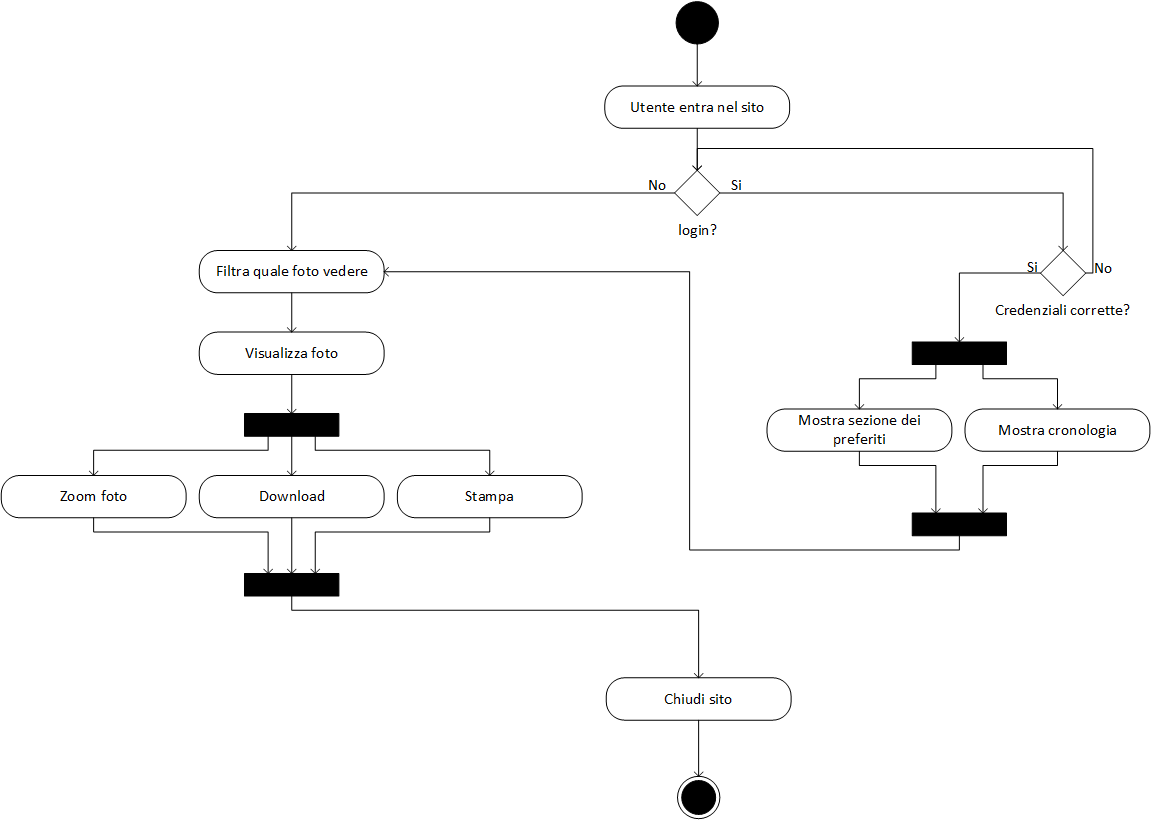


Figura : Design procedurale

# 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

Il file utils contiene metodi che richiamo in diversi punti della pagina, e per poterli definire una sola volta ho creato questo file, contiene solamente il metodo convalida, che serve a convalidare i dati e rimuovere caratteri che non vanno bene.

|  |
| --- |
| <?php  include "db\_conn.php";  function convalida($data){  $data = trim($data); //Rimuove eventuali spazi all'inizio e alla fine  $data = stripslashes($data); //Toglie eventuali slash  $data = htmlspecialchars($data); //Rimuove caratteri speciali  $data=str\_replace("\'","","$data");  return $data;  }  ?> |

### Connessione al Database

|  |
| --- |
| <?php  $sname = "sam-labo-web";  $user = "dbapodk";  $password = "kaR4oo\_Jah";  $dbName = "dbapodk";  $conn = mysqli\_connect($sname, $user, $password, $dbName);  if(!$conn){      exit("Errore nel connettersi al database");  }  ?> |

Questo file viene richiamato dagli altri file che hanno bisogno di connettersi al database.

### 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.

Verifico anche che il risultato della query sia un object oppure mysqli\_result, per evitare SQL injection.  
Se è giusta porto l’utente alla home page *home.php*, altrimenti ritorno un errore nella sezione 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, $sql);  if(gettype($result) == "mysqli\_result" || gettype($result) == "object"){      if(mysqli\_num\_rows($result) == 1) {          $row = mysqli\_fetch\_assoc($result);          if($row['Username'] === $username){ //Se password e nome utente sono IDENTICI a $pass  $uname              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();      }  }  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 la parte HTML della pagina, contenendo un form che manderà i dati in post, il codice PHP è contenuto in registraNewAccount.php, all’interno c’è il codice 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=" registraNewAccount.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 pagina registraNewAccount.php, 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

Per aggiungere le foto nei preferiti, nella home c’è un form con gli input nascosti che si auto riempiono grazie a un codice in javascript, al loro interno inserisco la descrizione, il titolo, la data e l’url dell’immagine.

Questo form si può visualizzare solo se l’utente ha effettuato il login.

|  |
| --- |
| **Home.php** |

Il metodo che modifica i valori è contenuto nel file *format.js* e viene richiamata ogni volta che richiamiamo il metodo *richiesta(d),* contenuta nel file *api.js*

|  |
| --- |
| **Format.js** |

Quando si preme il button, viene richiamato il file *addPreferiti.php*

|  |
| --- |
|  |

Prendo i dati in POST e li convalido con la funzione convalida, preparo uno statement per verificare se la foto è già stata aggiunta tra le preferite.  
Se il risultato è <0, quindi non è ancora stata aggiunta, preparo un alto statement dove aggiungo nei preferiti queta immagine, la eseguo e imposto il $\_COOKIE[‘date’] come la data dell’immagine, così quando si ritorna alla home aprirà la foto che stavo guardando.

### Rimuovere una foto dalla cronologia e dai preferiti

Il codice è identico per enrambi, cambiano solamente le query e gli header

|  |
| --- |
|  |

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.

### Gestione delle richieste per l’API

Le richieste per le tre foto vengono gestite da un metodo Javascript nel file *api.js* chiamato *richiesta(d),* che prende in entrata una data.

Come prima cosa analizziamo la parte di presa e verifica dei dati

|  |
| --- |
| function richiesta(d){ //d sarà una stringa      var url = "https://api.nasa.gov/planetary/apod?api\_key=";      var api\_key = "CH0wsKM4d6YOpvI7WI5tsulO3snZ5ybxueUIyb7l";      document.getElementsByClassName("error").innerHTML="";      defaultSize("immagine", "iframe");      defaultSize("immagine-1", "iframe-1");      defaultSize("immagine1", "iframe1");      var req1 = new XMLHttpRequest(); //Request per la foto centrale      var req2 = new XMLHttpRequest(); //Request per la foto centrale      var req3 = new XMLHttpRequest(); //Request per la foto centrale      if(d==''){          d = new Date();      }      else{          d = new Date(d);      }      var oggi = new Date();      var primaFoto = "1995-06-16";      var dataPrimaFoto = new Date(primaFoto);      if(d.getTime() < dataPrimaFoto.getTime() || d.getTime() > oggi.getTime()){          window.alert("Data non valida");          d = new Date();          dataCentrale = calcoloDate(d,0);      }      var dataCentrale = calcoloDate(d,0);      var dataSinistra = calcoloDate(d,1);      var dataDestra = calcoloDate(d,2);      var dataUtente = "&date="+ dataCentrale;      var dataIeri = "&date="+ dataSinistra;      var dataDomani = "&date="+ dataDestra;        d = d.toISOString(); //Formatto d perchè dovrò fare un controllo futuro      d = d.slice(0,10); //prendo solo le info a me necessarie, "yyyy-mm-dd"      oggi = oggi.toISOString();      oggi = oggi.slice(0,10); |

Inizializzo delle variabili che userò più tardi, poi verifico se d è vuoto oppure ha un valore, se è vuoto creo un obect data uguale oggi.

Dopodichè verifico se la data passata è valida, ovvero se viene prima del 16.06.1995 e dopo di oggi (compresi)

Dopodichè calcolo che giorno dovrà mostrare la foto del giorno seguente e quella del giorno dopo, usando il metodo *calcoloDate(d, numeroGiorno)*.

Formatto d così da renderlo più facilmente leggibile all’utene, trasformandolo da formato americano a formato “normale” (giorno.mese.anno).

Ora effettuo le 3 richieste per le foto

|  |
| --- |
| **Foto 1, centrale:** |

Creo una richiesta, se ritorna il codice 200, allora inserisco i valori ritornati dal JSON direttamente sull’html, cambiando la source della foto e le varie informazioni, altrimenti restituisco un errore e come source metto una foto di default.

Il metodo *ScegliMedia()* controlla il valore di media, capendo se mostrare un iframe con source un video oppure una foto con source una foto.

|  |
| --- |
| **Foto 2, sinistra:** |

Molto simile alla prima foto, unica differenza che prendo molte meno informazioni essendo che ho bisogno solo della source della foto

|  |
| --- |
| **Foto 3, destra:** |

Identico al codice per la foto 2.

# 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-1 | **Nome:** | Login |
| **Descrizione:** | Effettuare il login con un nome utente e password corretti | | |
| **Prerequisiti:** | Nella tabella degli utenti deve essere presente l’utente | | |
| **Procedura:** | 1. Aprire il sito <http://apod-kamil.labosam.cpt.local> e compilare il form 2. Inserire come username e password ‘admin’ 3. Inviare il form | | |
| **Risultati attesi:** | Arrivare nella home page <http://apod-kamil.labosam.cpt.local/home.php> con il login effettuato con successo | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-002  REQ-1 | **Nome:** | Login fallito |
| **Descrizione:** | Effettuare il login con un nome utente o password sbagliati | | |
| **Prerequisiti:** | Nella tabella degli utenti non deve essere presente l’utente | | |
| **Procedura:** | 1. Aprire il sito <http://apod-kamil.labosam.cpt.local> e compilare il form 2. Inserire come username e password ‘sbagliato’ 3. Inviare il form | | |
| **Risultati attesi:** | Rimanere nella stessa pagina e ricevere l’errore “Username o pasword sbagliati” | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-003  REQ-1 | **Nome:** | Protezione da un SQL Injection nel form di login |
| **Descrizione:** | Verificare se il form è a prova di SQL Injection | | |
| **Prerequisiti:** | Tabella degli utenti | | |
| **Procedura:** | 1. Andare nella pagina di login e compilare il form provando ad eseguire un SQL Injection 2. Inserire come username e password: ‘ or ’x’=’x’ 3. Inviare il form | | |
| **Risultati attesi:** | Errore e nessuna restituzione dei dati delle tabelle del database | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-004  REQ-2 | **Nome:** | Protezione da un SQL Injection nel form di registrazione |
| **Descrizione:** | Verificare se il form è a prova di SQL Injection | | |
| **Prerequisiti:** | Tabella degli utenti | | |
| **Procedura:** | 1. Andare nella pagina di registrazione e compilare il form provando ad eseguire un SQL Injection 2. Inserire come username e le due password: ‘ or ’x’=’x’ 3. Inviare il form | | |
| **Risultati attesi:** | Errore e nessuna restituzione dei dati delle tabelle del database | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento:** | TC-005  REQ-2 | **Nome:** | Registrarsi con un nuovo account |
| **Descrizione:** | Crear un nuovo utente ed effettuare il login | | |
| **Prerequisiti:** | Tabella degli utenti | | |
| **Procedura:** | 1. Dalla pagina di login premere il pulsante “Crea un account” 2. Inserire come username “user” e le due password “user” 3. Inviare il form | | |
| **Risultati attesi:** | Creazione dell’utente nella tabella utente ed arrivare alla pagina home con il login effettuato con successo | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-006  REQ-2 | **Nome:** | Creare account con lo stesso username |
| **Descrizione:** | Verificare che ci siano adeguati controlli e sia impossibile creare due utenti con lo stesso username | | |
| **Prerequisiti:** | Tabella degli utenti | | |
| **Procedura:** | 1. Andare nella pagina di register e compilare il form con il nome di un utente già esistente 2. Inserire le due password identiche a scelta 3. Inviare il form | | |
| **Risultati attesi:** | Errore e nessuna creazione di un nuovo utente | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-007  REQ-3 | **Nome:** | Restituzione immagine di default se API restituisce un errore |
| **Descrizione:** | Se l’API dovesse restituire un errore o nessun url, nella home page nella sezione della foto sarà presente una foto di default | | |
| **Prerequisiti:** | Non mandare le richieste all’API | | |
| **Procedura:** | 1. Effettuare il login o continuare come guest 2. Non dovrebbe essere possibili vedere le foto della nasa, saranno visibili solo le foto di default | | |
| **Risultati attesi:** | Tutte le foto devono essere quella di default | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-008  REQ-4 | **Nome:** | Aggiungere foto alla cronologia |
| **Descrizione:** | Aggiungere una foto alla cronologia | | |
| **Prerequisiti:** | Effettuare il login  Tabella cronologia | | |
| **Procedura:** | 1. Effettuare il login 2. Effettuare una ricerca tramite la pagina “filtro.php”, raggiungibile tramite la sezione “Menu” in alto a sinistra, premendo il button con scritto “Filtro” che porterà alla pagina <http://apod-kamil.labosam.cpt.local/filtro.php> 3. Andare nella pagina “history.php”, raggiungibile tramite la sezione “Menu” in alto a sinistra, premendo il button con scritto “Cronologia” che porterà alla pagina <http://apod-kamil.labosam.cpt.local/history.php> 4. Verificare che l’immagine sia stata aggiunta e sia visibile | | |
| **Risultati attesi:** | Aggiunta con successo dell’immagine e corretta visualizzazione nella pagina apposita | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-009  REQ-4 | **Nome:** | Rimuovere foto dalla cronologia |
| **Descrizione:** | Rimuovere una foto dalla cronologia | | |
| **Prerequisiti:** | Effettuare il login  Tabella cronologia | | |
| **Procedura:** | 1. Effettuare il login 2. Andare nella pagina “history.php” e premere il pulsante “Rimuovi” 3. Verificare che l’immagine sia stata rimossa dalla tabella e non più visualizzabile | | |
| **Risultati attesi:** | Rimozione con successo dell’immagine nella pagina apposita | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-010  REQ-5 | **Nome:** | Aggiungere foto ai preferiti |
| **Descrizione:** | Aggiungere una foto nei preferiti | | |
| **Prerequisiti:** | Effettuare il login  Tabella preferiti | | |
| **Procedura:** | 1. Effettuare il login 2. Nella home page premere il pulsante “Metti nei preferiti” 3. Andare nella pagina “favorite.php”, raggiungibile tramite la sezione “Menu” in alto a sinistra, premendo il button con scritto “Preferiti” che porterà alla pagina <http://apod-kamil.labosam.cpt.local/favorite.php> 4. Verificare che l’immagine sia stata aggiunta e sia visibile | | |
| **Risultati attesi:** | Aggiunta con successo dell’immagine e corretta visualizzazione nella pagina apposita | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-011  REQ-5 | **Nome:** | Rimuovere foto dai preferiti |
| **Descrizione:** | Rimuovere una foto dai preferiti | | |
| **Prerequisiti:** | Effettuare il login  Tabella preferiti | | |
| **Procedura:** | 1. Effettuare il login 2. Andare nella pagina “favorite.php” e premere il pulsante “Rimuovi” 3. Verificare che l’immagine sia stata rimossa dalla tabella e non più visualizzabile | | |
| **Risultati attesi:** | Rimozione con successo dell’immagine nella pagina apposita | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-012  REQ-6 | **Nome:** | Effettuare una ricerca valida |
| **Descrizione:** | Effettuare una ricerca valida tramite la pagina <http://apod-kamil.labosam.cpt.local/filtro.php> | | |
| **Prerequisiti:** | - | | |
| **Procedura:** | 1. Dal menu in alto sinistra andare nella pagina per eseguire la ricerca 2. Inserire una data valida all’interno del form e premere il pulsante “Cerca” | | |
| **Risultati attesi:** | Venir reindirizzati alla home page e visualizzare la foto della data ricercata | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-013  REQ-6 | **Nome:** | Effettuare una ricerca invalida |
| **Descrizione:** | Effettuare una ricerca invalida tramite la pagina <http://apod-kamil.labosam.cpt.local/filtro.php> | | |
| **Prerequisiti:** | - | | |
| **Procedura:** | 1. Dal menu in alto sinistra andare nella pagina per eseguire la ricerca 2. Inserire una data invalida all’interno del form e premere il pulsante “Cerca” | | |
| **Risultati attesi:** | rimanere nella stessa pagina e visualizzare l’errore “Data invalida” | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-014  REQ-7 | **Nome:** | Navigare grazie alla sequenza delle immagini |
| **Descrizione:** | Premere sulle foto ai lati per navigare tra i giorni | | |
| **Prerequisiti:** | - | | |
| **Procedura:** | 1. Nella Home page, premere su una delle immagini ai lati e navigare tra i vari giorni | | |
| **Risultati attesi:** | L’immagine centrale cambia e diventa quella premuta, di conseguenza anche quelle ai lati | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-015  REQ-7 | **Nome:** | Provare a navigare premendo un immagine invalida |
| **Descrizione:** | Premere su una foto invalida per navigare, come per esempio quella el giorno seguente (se ancora inesistente) | | |
| **Prerequisiti:** | - | | |
| **Procedura:** | 1. Nella Home page, premere sulla foto del giorno seguente | | |
| **Risultati attesi:** | Restituizione dell’errore “Data non valida” e nessun cambiamento | | |

## Risultati test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test**  **Case** | **Risultato** | **Descrizione** | **Data di test** |
| 001 | Passato | **Risultato atteso:** Arrivare nella home page <http://apod-kamil.labosam.cpt.local/home.php> con il login effettuato con successo  **Risultato effettivo:**  Dopo aver compilato il form inserendo ‘admin’ come username e password e premuto il pulsante ‘Login’, il login viene effettuato con successo.    Figura : Test1, login corretto  Dopo venir reindirizzati nella home page, in alto in centro vediamo ‘ciao admin’, segno che siamo riusciti ad accedere con successo.    Figura : Login effettuato con successo | 18.12.2024 |
| 002 | Passato | **Risultato atteso:**  Rimanere nella stessa pagina e ricevere l’errore “Username o pasword sbagliati”  **Risultato effettivo:**  Dopo aver compilato il form inserendo ‘user come username e password e premuto il pulsante ‘Login’, ci viene restituito l’errore “Username o password errata”    Figura : Login con dati non validi fallito | 18.12.2024 |
| 003 | Passato | **Risultato atteso:**  Errore e nessuna restituzione dei dati delle tabelle del database  **Risultato effettivo:**  Compilando il form con ‘ or ‘x’=’x’, così che la query sia  SELECT \* FROM utente WHERE Username= "" or ‘x’=’x’ AND Password= "" or ‘x’=’x’    Figura : Form Login compilato con tentata SQL Injection  A una prima prova, mi restituiva un errore, ma dopo aver aggiunto un controllo aggiuntivo, mi restituisce un semplice errore “Username o password errata”  Prima:      Dopo:  Ho aggiunto un controllo al tipo di $result, se è diverso da mysqli\_result ritorna errore    Figura : SQL Injection non riuscita | 18.12.2024 |
| 004 | Superato parzialmente | **Risultato atteso:** Errore e nessuna restituzione dei dati delle tabelle del database **Risultato effettivo:**  Provare a registrarsi mettendo ‘ or ‘x’=’x’ come username non restituisce nessun dato sensibile, ma crea un utente con quell’username, che poi ovviamente non può venir utilizzato    Figura : SQL Injection nel form di registrazione    Figura : Creazione dell'utente con username la SQL injection | 18.12.2024 |
| 005 | Passato | **Risultato atteso:** Creazione dell’utente nella tabella utente ed arrivare alla pagina home con il login effettuato con successo **Risultato effettivo:**  Riempio il form inserendo come username “Jenny” e password “abcd”    Figura : Creazione di un nuovo utente  L’account viene creato con successo e veniamo automatiamente loggati con il nuovo utente    Figura : Creazione e login del nuovo user effettuato con successo | 18.12.2024 |
| 006 | Passato | **Risultato atteso:** Errore e nessuna creazione di un nuovo utente **Risultato effettivo:**  Provando a creare un utente con username ‘admin’ restituisce l’errore ‘Username già esistente’    Figura : Provare a crear un account con un username già esistente | 18.12.2024 |
| 007 | Passato | **Risultato atteso:** Tutte le foto devono essere quella di default  **Risultato effettivo:**  Dopo aver rimosso la richiesta all’API dal codice, nella home page vediamo solo le foto di default:    Figura : Home con foto default | 18.12.2024 | |
| 008 | Superato parzialmente | **Risultato atteso:** Aggiunta con successo dell’immagine e corretta visualizzazione nella pagina apposit **Risultato effettivo:**  La ricerca tramite la pagina filtro va a buon fine, portandomi correttamente nella pagina home mostrando la foto voluta,    Figura : Filtro  Ma nella pagina history i dati non vengono salvati correttamente, solo in parte vengono salvati    Figura : tabella cronologia aggiornata | 18.12.2024 |
| 009 | Passato | **Risultato atteso:** Rimozione con successo dell’immagine nella pagina apposita  **Risultato effettivo:**  Premendo il pulsante “Rimuovi dalla cronologia” dalla pagina ‘history.php’ l’immagine viene rimossa correttamente  Prima    Figura : History.php e la tabella cronologia prima della rimozione di un valore  Dopo:    Figura : History.php e la tabella cronologia dopo la rimozione di un valore | 18.12.2024 |
| 010 | Passato | **Risultato atteso:** Aggiunta con successo dell’immagine e corretta visualizzazione nella pagina apposita **Risultato effettivo:**  Dopo aver effettuato il login, nella home page premo il bottone “Metti nei preferiti”.  Ora vado alla pagina dei preferiti e la foto è stata aggiunta con successo:    Figura : Pagina dei preferiti | 18.12.2024 |
| 011 | Passato | **Risultato atteso:** Rimozione con successo dell’immagine nella pagina apposita **Risultato effettivo:**  Premendo il pulsante “” la foto viene eliminata dalla tabella preferito con successo    Figura : Pagina dei preferiti e tabella preferito dopo aver rimosso un valore | 18.12.2024 |
| 012 | Passato | **Risultato atteso:** Venir reindirizzati alla home page e visualizzare la foto della data ricercata **Risultato effettivo:**  Dalla pagina per le ricerche inserendo una data valida a mia scelta e premendo il button “cerca”, vengo reindirizzato con successo alla homepage con la foto corretta    Figura : Ricerca di una data valida    Figura : Ricerca della data valida finita con successo | 18.12.2024 |
| 013 | Superato parzialmente | **Risultato atteso:** rimanere nella stessa pagina e visualizzare l’errore “Data invalida” **Risultato effettivo:**  Esce l’errore “Data non valida”, ma veniamo comunque reindirizzati alla home page | 18.12.2024 |
| 014 | Superato parzialmente | **Risultato atteso:** L’immagine centrale cambia e diventa quella premuta, di conseguenza anche quelle ai lati **Risultato effettivo:**  premendo sulle foto ai lati si riesce a cambiare foto con successo, ma se si preme su un video no, essendo che quest’ultimo parte. Per risolvere questo problema ho aggiunto delle frecce nel footer per spostarsi di un giorno in avanti o indietro | 18.12.2024 |
| 015 | Passato | **Risultato atteso:** Restituizione dell’errore “Data non valida” e nessun cambiamento **Risultato effettivo:**  Se si prova a premere sulla foto del giorno seguente se inesistente, ci viene rstituito l’errore “data non valida”    Figura : Errore restituito dopo aver provato a premere su una foto con data non valida | 18.12.2024 |

## Mancanze/limitazioni conosciute

I file del progetto sono tutti nella stessa cartella senza un ordine, anche i nomi sono complicati e spesso ripeto pezzi di codice che avrei potuto fare una volta e richiamare più volte, come l’header e il menu per le varie pagine.

Sarebbe stato utile utilizzare un framework MVC, per avere i file ordinati, ma purtroppo lo abbiamo visto a scuola nelle ultime settimane del progetto, e sarebbe stato troppo lungo implementare un framework così nel progetto.

# Consuntivo

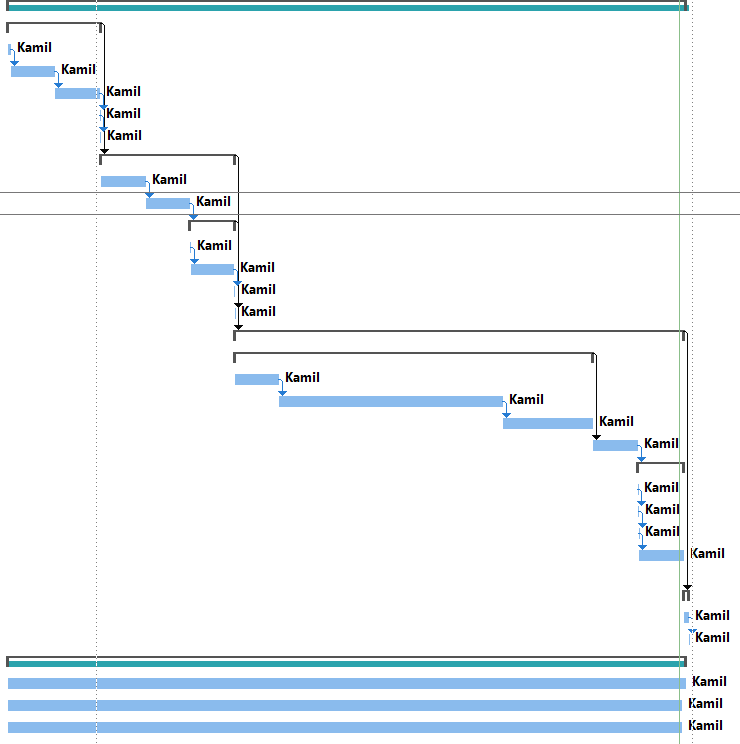


Figura : Gantt consuntivo

# 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 potrebbero essere una pagina dedicata all’admin per la gestione degli utenti, la cronologia funzionante, la possibilità di cercare foto anche per il titolo e non solo per la data.

## Considerazioni personali

Ho imparato a lavorare con un database e comunicarci con php, a cercare informazioni online e a leggere le documentazioni.

# 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. |
| Source | Indirizzo per raggiungere la foto |
| HTML | **Hyper Text Markup Language**: Linguaggio per creare pagine web |
| PHP | **Hypertext Preprocessor:** linguaggio lato server per comunicare con un Database |

# Indice delle figure

[Figura 1: Use case 7](#_Toc185427858)

[Figura 2: Gantt preventivo 8](file:///E:\I3AC\Professionale\APOD_API-2024-2025\3_Documentazione%20(word%20e%20pdf)\Documentazione_AstronomicPictureOfTheDay.docx#_Toc185427859)

[Figura 3: Schema E-R 10](#_Toc185427860)

[Figura 4: Schema del Database 10](#_Toc185427861)

[Figura 5: Pianificazione della GUI del Login 11](#_Toc185427862)

[Figura 6: Pianificazione della GUI della Home page 11](#_Toc185427863)

[Figura 7: Pianificazione della page dei Preferiti e cronologia 12](#_Toc185427864)

[Figura 8: Design procedurale 12](file:///E:\I3AC\Professionale\APOD_API-2024-2025\3_Documentazione%20(word%20e%20pdf)\Documentazione_AstronomicPictureOfTheDay.docx#_Toc185427865)

[Figura 9: Test1, login corretto 33](#_Toc185427866)

[Figura 10: Login effettuato con successo 33](#_Toc185427867)

[Figura 11: Login con dati non validi fallito 34](#_Toc185427868)

[Figura 12: Form Login compilato con tentata SQL Injection 34](#_Toc185427869)

[Figura 13: SQL Injection non riuscita 34](#_Toc185427870)

[Figura 14: SQL Injection nel form di registrazione 35](#_Toc185427871)

[Figura 15: Creazione dell'utente con username la SQL injection 35](#_Toc185427872)

[Figura 16: Creazione di un nuovo utente 35](#_Toc185427873)

[Figura 17: Creazione e login del nuovo user effettuato con successo 35](#_Toc185427874)

[Figura 18: Provare a crear un account con un username già esistente 36](#_Toc185427875)

[Figura 19: Home con foto default 36](#_Toc185427876)

[Figura 20: Filtro 36](#_Toc185427877)

[Figura 21: tabella cronologia aggiornata 36](#_Toc185427878)

[Figura 22: History.php e la tabella cronologia prima della rimozione di un valore 37](#_Toc185427879)

[Figura 23: History.php e la tabella cronologia dopo la rimozione di un valore 37](#_Toc185427880)

[Figura 24: Pagina dei preferiti 37](#_Toc185427881)

[Figura 25: Pagina dei preferiti e tabella preferito dopo aver rimosso un valore 38](#_Toc185427882)

[Figura 26: Ricerca di una data valida 38](#_Toc185427883)

[Figura 27: Ricerca della data valida finita con successo 39](#_Toc185427884)

[Figura 28: Errore restituito dopo aver provato a premere su una foto con data non valida 39](#_Toc185427885)

[Figura 29: Gantt consuntivo 40](#_Toc185427886)

# Bibliografia

## Sitografia

* <https://www.php.net/>, *php.net,* durante tutto il progetto
* <https://www.w3schools.com/php/>, *w3school,* durante tutto il progetto
* <https://www.youtube.com/@SimplilearnOfficial>, *youtube.com,* 09.10.2024 / 16.10.2024