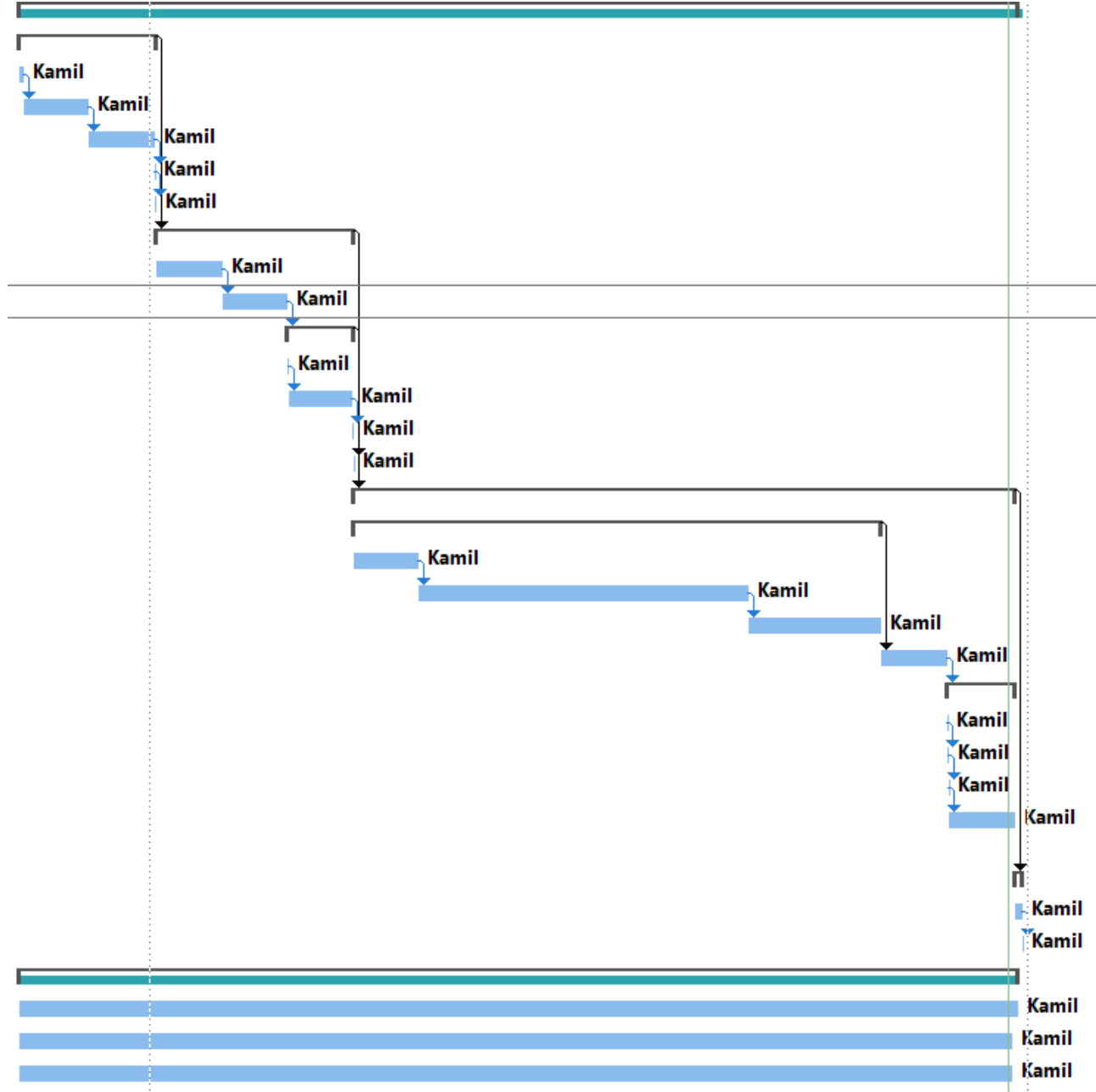


Documentazione Progetto Astronomic Picture of the Day

Titolo del progetto: Astronomic Picture of the Day
Alunno/a: Kamil Siddiqui
Classe: Info 3AC
Anno scolastico: 2024/2025
Docente responsabile: Guido Montalbetti

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

1 Introduzione

1.1 Informazioni sul progetto

1.2 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?

1.3 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.

2 Analisi

2.1 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.

2.2 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'utente 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

2.2.1 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.

2.3 Use case

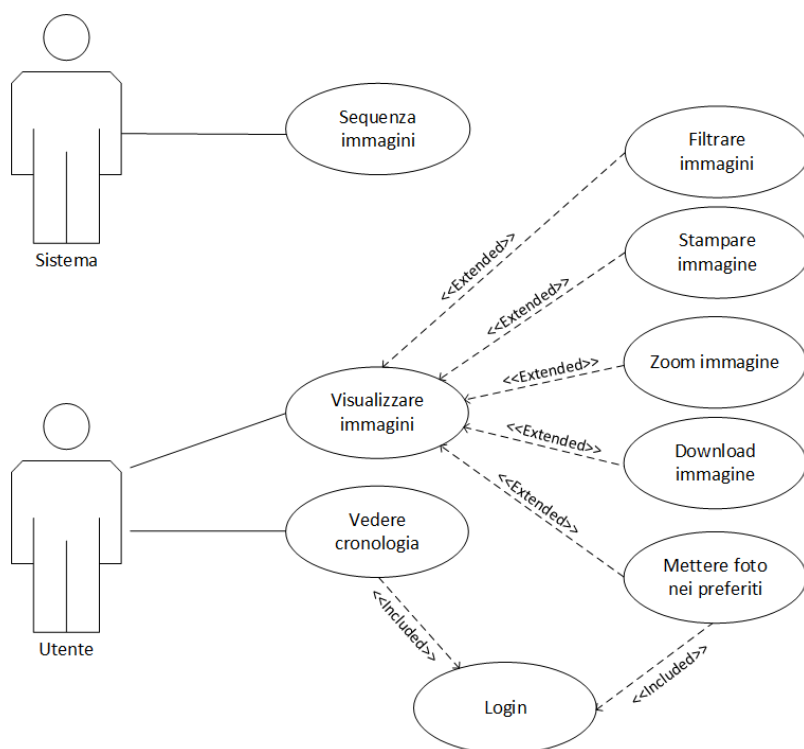


Figura 1: Use case

2.4 Pianificazione

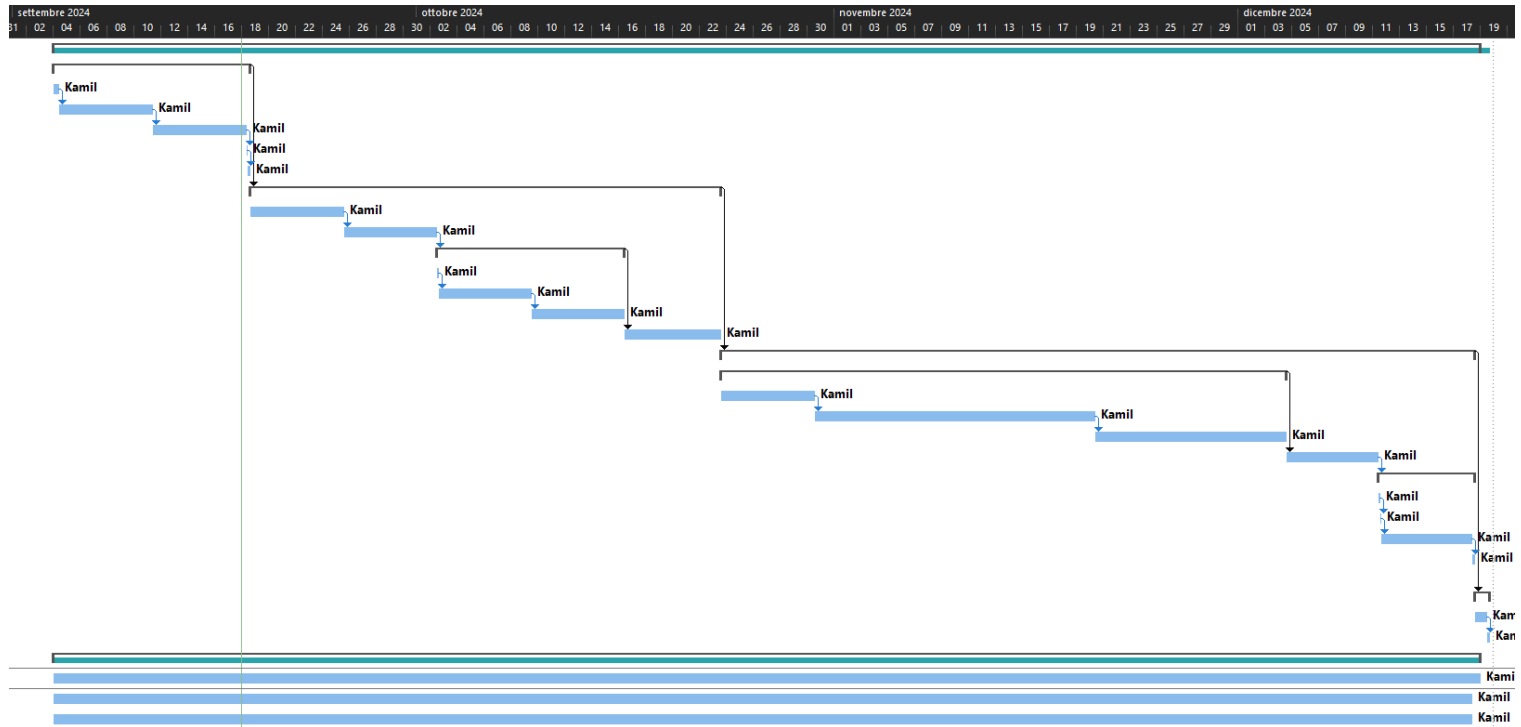


Figura 2: Gantt preventivo

2.5 Analisi dei mezzi

2.5.1 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.

2.5.2 Hardware

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

3 Progettazione

Questo capitolo descrive esaurientemente 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.

3.1 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

3.2 Design dei dati e database

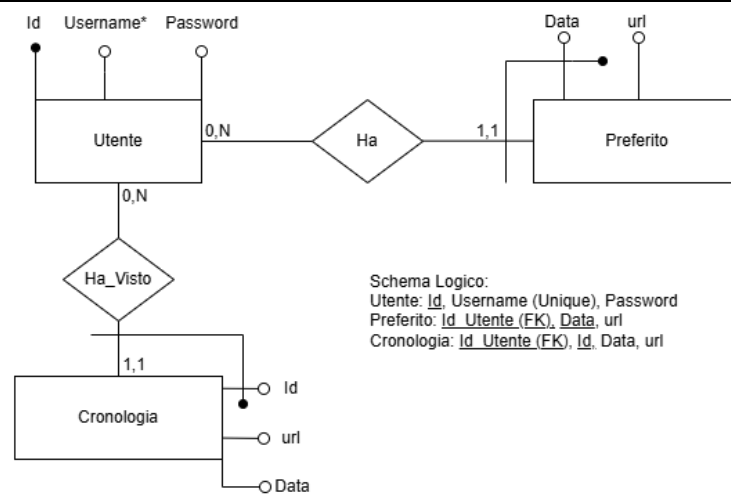


Figura 3: Schema E-R

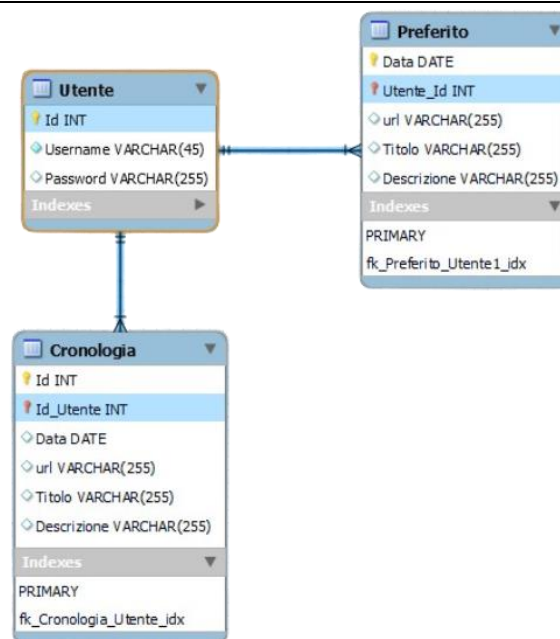
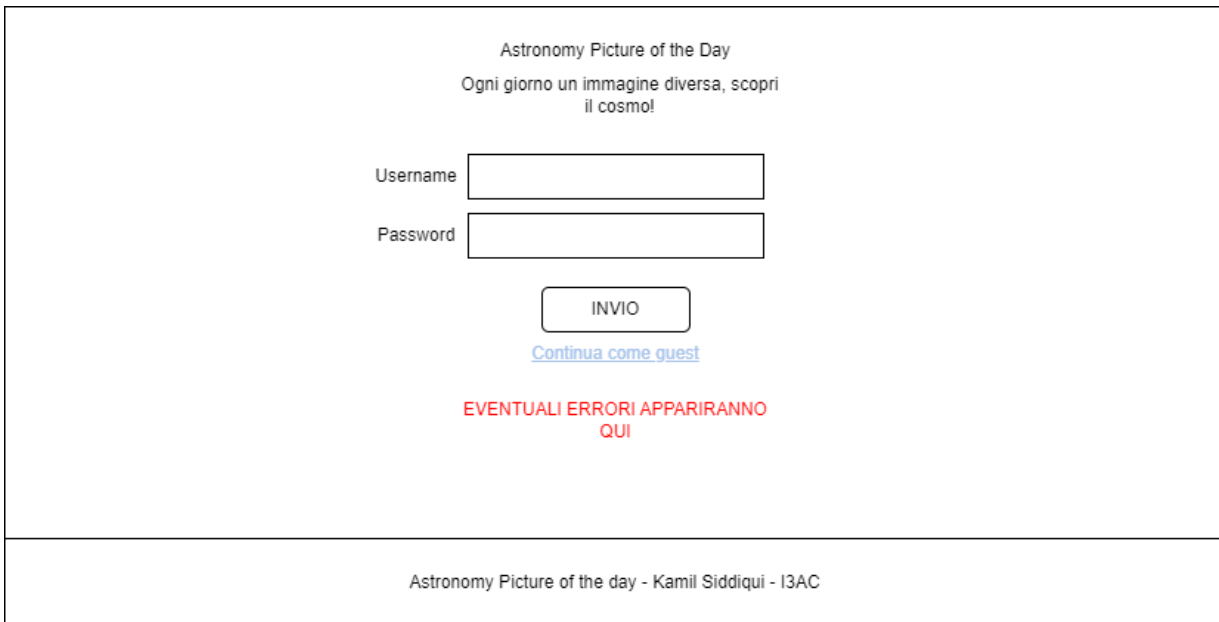


Figura 4: Schema del Database

3.3 Design delle interfacce

3.3.1 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.



The login interface wireframe includes the following elements:

- Header:** "Astronomy Picture of the Day" and "Ogni giorno un immagine diversa, scopri il cosmo!"
- Form Fields:** "Username" and "Password" input boxes.
- Buttons:** "INVIO" (Submit) and a link "Continua come guest" (Continue as guest).
- Error Message:** "EVENTUALI ERRORI APPARIRANNO QUI" (Possible errors will appear here).
- Footer:** "Astronomy Picture of the day - Kamil Siddiqui - I3AC"

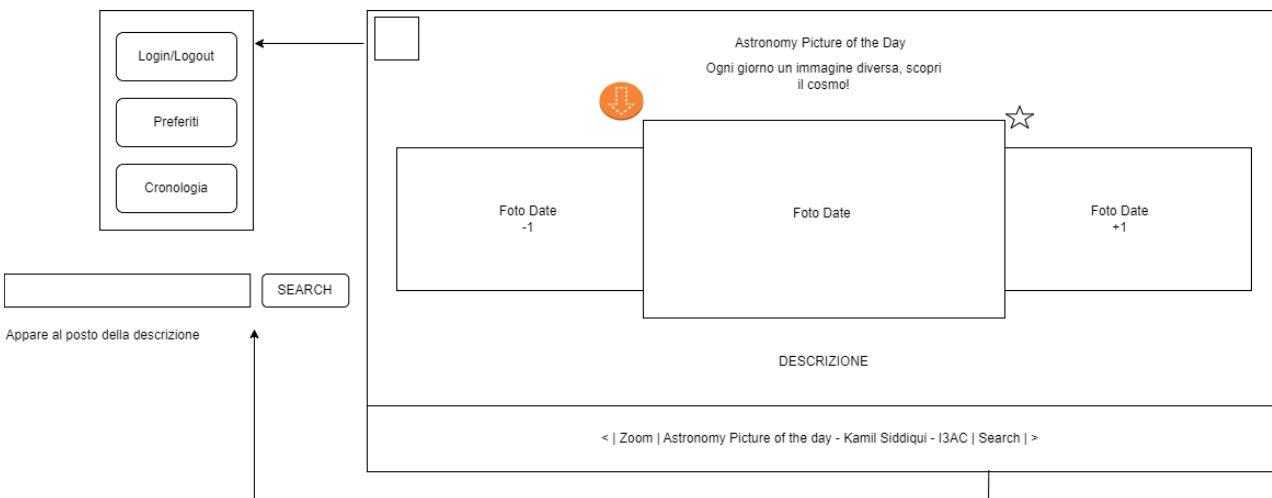
Figura 5: Pianificazione della GUI del Login

3.3.2 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



The home page interface wireframe includes the following elements:

- Header:** "Astronomy Picture of the Day" and "Ogni giorno un immagine diversa, scopri il cosmo!".
- Navigation:** A dropdown menu (indicated by an orange circle) in the top left corner.
- Image Carousel:** Three image boxes labeled "Foto Date -1", "Foto Date", and "Foto Date +1".
- Description:** A large box labeled "DESCRIZIONE" below the image carousel.
- Footer:** "< | Zoom | Astronomy Picture of the day - Kamil Siddiqui - I3AC | Search | >".
- Side Menu:** A vertical menu on the left with options: "Login/Logout", "Preferiti", and "Cronologia".
- Search:** A search bar with a "SEARCH" button.
- Placeholder:** A note "Appare al posto della descrizione" (Appears in place of the description) pointing to the description area.

Figura 6: Pianificazione della GUI della Home page

3.3.3 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.

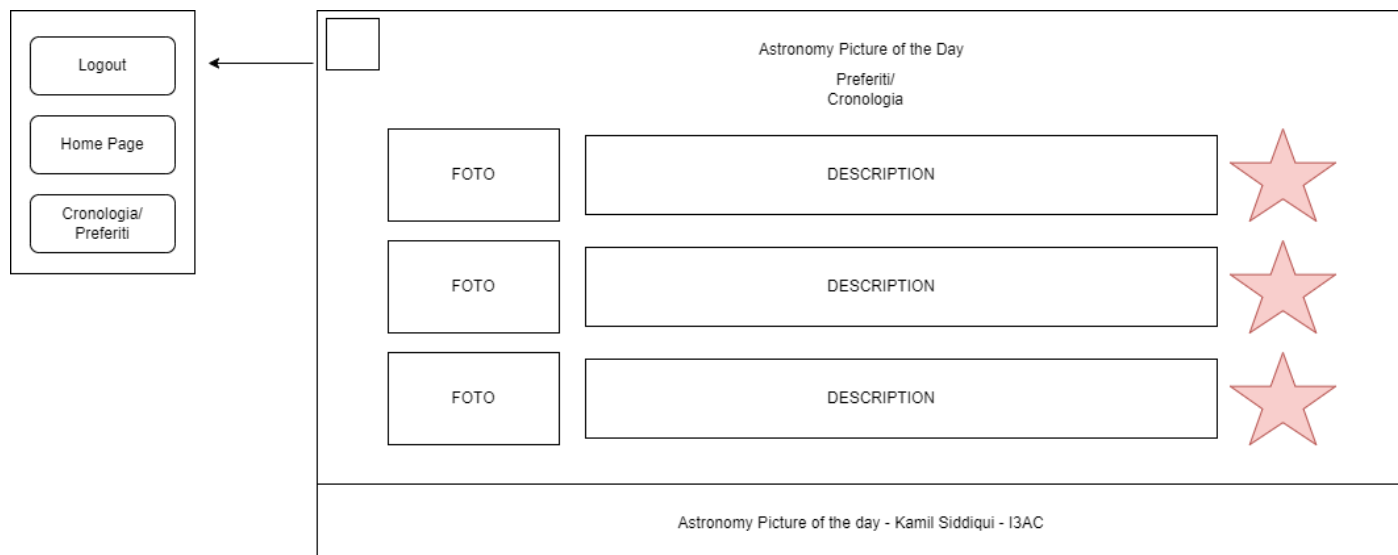


Figura 7: Pianificazione della page dei Preferiti e cronologia

3.4 Design procedurale

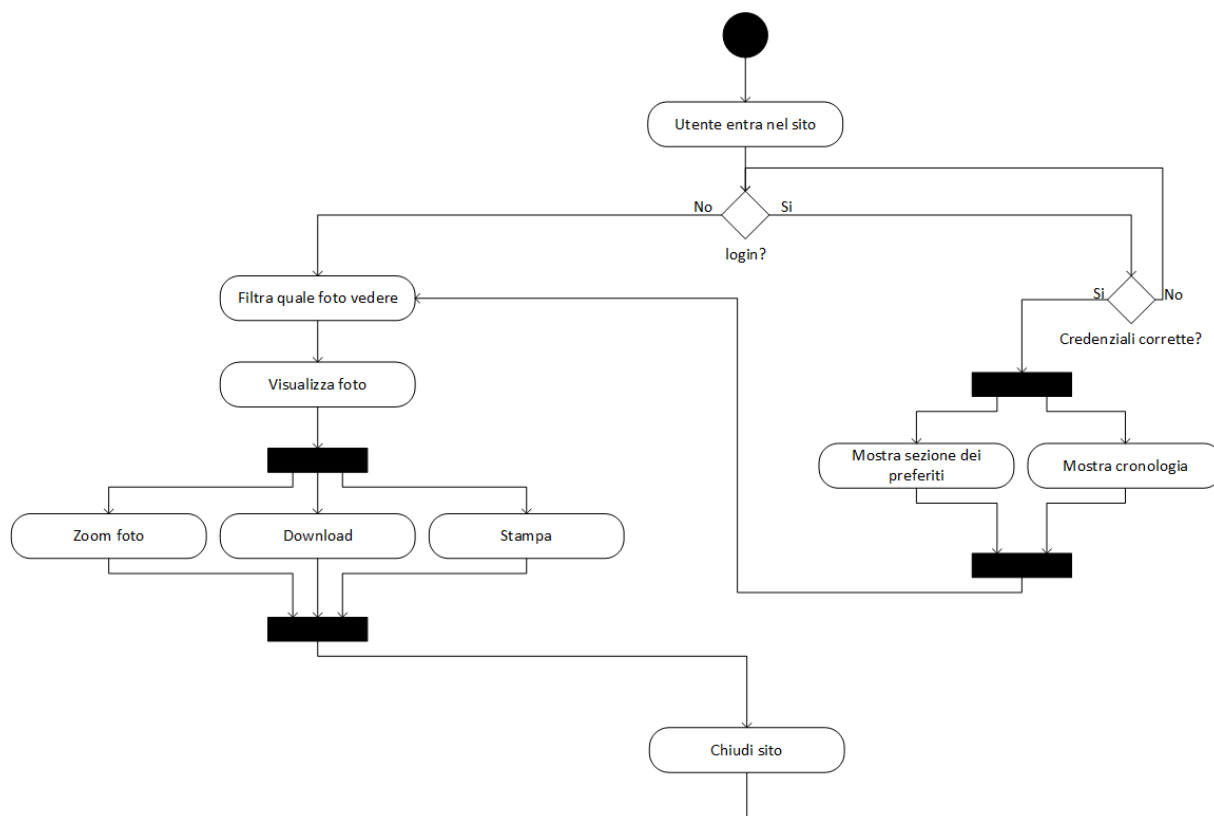


Figura 8: Design procedurale

4 Implementazione

4.1 Database

4.1.1 Utente

```
CREATE TABLE utente(
    Id INT AUTO_INCREMENT,
    Username VARCHAR(45),
    Password VARCHAR(255),
    PRIMARY KEY (Id)
);
```

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 creato 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.

4.1.2 Preferiti

```
CREATE TABLE preferito (
    Data DATE,
    Utente_Id INT,
    url VARCHAR(255),
    Titolo VARCHAR(255) ,
    Descrizione VARCHAR(255) ,
    PRIMARY KEY (Data, Utente_Id),
    FOREIGN KEY (Utente_Id)
    REFERENCES Utente(Id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

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 preferita 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.

4.1.3 Cronologia

```
CREATE TABLE Cronologia(
    Id INT AUTO_INCREMENT,
    Utente_Id INT NOT NULL,
    Data Date NOT NULL,
    url VARCHAR(255),
    PRIMARY KEY(Id,Utente_Id ),
    FOREIGN KEY (Utente_Id)
    REFERENCES Utente(Id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

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.

4.2 UML

4.2.1 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 gestisce il layout della pagina html e uno che gestisce le richieste all'api.

4.3 Codice

4.3.1 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;
}
?>
```

4.3.2 Connessione al Database

```
<?php

$name = "sam-labo-web";
$user = "dbapodk";
$password = "kaR4oo_Jah";
$dbName = "dbapodk";

$conn = mysqli_connect($name, $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.

4.3.3 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']);
$password = convalida($_POST['password']);

//CONTROLO 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($password)){
    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='$_password'";
//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 effettuato 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();
}
}

```

4.3.4 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'utente non sbaglia 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']);

//CONTROLLA 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 effettuato 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.

4.3.5 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 distruggerà 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");
```

4.3.6 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

```
<?php
if(isset($_SESSION['Id']) && isset($_SESSION['Username'])) {
?>

<form action="addPreferiti.php" method="get">
    <div id="inputDaNascondere">
        <input type="text" id="url" name="url" value=" " style="visibility:hidden;">
        <input type="text" id="dataInput" name="dataInput" value=" " style="visibility:hidden;">
        <input type="text" id="titoloInput" name="titoloInput" value=" " style="visibility:hidden;">
        <input type="text" id="descInput" name="descInput" value=" " style="visibility:hidden;">
    </div>
    <button id="button_preferito" type="submit">Metti nei preferiti</button><br><br>
</form>
<?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

```
function change(){
    document.getElementById("url").value = " ";
    document.getElementById("dataInput").value = " ";
    document.getElementById("titoloInput").value = " ";
    document.getElementById("descInput").value = " ";
    //Gli input si svuotano così in caso di assenza di url non salvo quello sbagliato
    var data = formatDataUSA();
    var url = document.getElementById("immagine").src;
    if(url==""){
        url = document.getElementById("iframe").src;
    }
    var titolo = document.getElementById("titolo_img").innerHTML;
    var desc = document.getElementById("descrizione_immagine").innerHTML;
    document.getElementById("url").value = url;
    document.getElementById("dataInput").value = data;
    document.getElementById("titoloInput").value = titolo;
    document.getElementById("descInput").value = desc;
}
```

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

```
<?php
    session_start();
    include "db_conn.php";
    include "utils.php";

    $data = convalida($_POST['dataInput']);
    $url = convalida($_POST['url']);
    $id = convalida($_POST['Id']);
    $titolo = convalida($_POST['titoloInput']);
    $desc = convalida($_POST['descInput']);

    if($stmt = $conn->prepare('SELECT Utente_Id, Data FROM preferito WHERE Utente_Id = ? AND Data = ?')){
        $stmt->bind_param('ss', $_SESSION['Id'], $data);
        $stmt->execute();
        $stmt->store_result();

        if($stmt->num_rows>0){
            setcookie('date', $data, time() + 5, '/');
            $_COOKIE['date'] = $data;
            header("Location: home.php?error=Foto già messa nei preferiti");
            exit();
        }
        else{
            if($stmt = $conn->prepare('INSERT INTO preferito VALUE(?, ?, ?, ?, ?)')){
                $stmt->bind_param('sssss', $data, $_SESSION['Id'], $url, $titolo, $desc);
                $stmt->execute();

                setcookie('date', $data, time() + 5, '/');
                $_COOKIE['date'] = $data;

                header("Location: home.php?");
                exit();
            }
        }
        $stmt->close();
    }
    else{
        header("Location: home.php?error= ERRORE");
        exit();
    }
}
?>
```

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 altro statement dove aggiungo nei preferiti questa 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.

4.3.7 Rimuovere una foto dalla cronologia e dai preferiti

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

```
<?php
session_start();
include "db_conn.php";

$data = $_POST['dataInput'];
$id = $_SESSION['Id'];

if($stmt = $conn->prepare('SELECT Utente_Id, Data FROM preferito WHERE Utente_Id = ? AND Data = ?')){
    $stmt->bind_param('ss', $_SESSION['Id'], $data);
    $stmt->execute();
    $stmt->store_result();

    if($stmt->num_rows>0){
        if($stmt = $conn->prepare('DELETE FROM preferito WHERE Utente_Id = ? AND Data = ?')){
            $stmt->bind_param('ss', $_SESSION['Id'], $data);
            $stmt->execute();

            header("Location: favorite.php?error=Foto Rimossa con successo!");
            exit();
        }
    }
    $stmt->close();
}else{
    header("Location: favorite.php?error= ERRORE");
    exit();
}

?>
```

DESCRIZIONE QUA.

4.3.8 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 mettere 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 = "";
        }
        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.

4.3.9 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 = "CH0wsKM4d6Y0pvI7WI5tsul03snZ5ybxueUIyb71";
    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:

```
req1.open("GET", url + api_key + dataUtenite);
req1.send();
req1.addEventListener("load", function(){
    if(req1.status == 200 && req1.readyState == 4){ //Se non restituisce un codice di errore. 200 richiesta con successo e 4 operazione completata
        var response1 = JSON.parse(req1.responseText); //JSON con la risposta
        document.getElementById("titolo_img").innerHTML = response1.title; //modifica il titolo

        document.getElementById("data_immagine").innerHTML = formatDataEU(response1.date);
        document.getElementById("descrizione_immagine").innerHTML = formattaStringa(response1.explanation);

        scegliMedia("immagine", "iframe", response1);
        change();
    }
    else if(req1.status == 404){
        window.alert("Immagine inesistente per questa data");
        document.getElementById("titolo_img").innerHTML = "Immagine inesistente";
        document.getElementById("descrizione_immagine").innerHTML = "Per favore cercare la foto di un'altra data, grazie!";
        defaultSRC("immagine", "iframe");
    }
    else{
        defaultSRC("immagine", "iframe");
    }
})
```

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:

```
req2.open("GET", url + api_key + dataIeri);
req2.send();
req2.addEventListener("load", function(){
    if(req2.status == 200 && req2.readyState == 4){ //Se n
        var response2 = JSON.parse(req2.responseText); //J
        scegliMedia("immagine-1", "iframe-1", response2);
    }
    else{
        defaultSRC("immagine-1", "iframe-1");
    }
})
```

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

Foto 3, destra:

```
req3.open("GET", url + api_key + dataDomani);
req3.send();
req3.addEventListener("load", function(){
    if(req3.status == 200 && req3.readyState == 4){ //Se 1
        var response3 = JSON.parse(req3.responseText); //J
        scegliMedia("immagine1", "iframe1", response3);
    }
    else{
        defaultSRC("immagine1", "iframe1");
    }
})
```

Identico al codice per la foto 2.

5 Test

5.1 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:	TC-001	Nome:	Login
Riferimento:	REQ-1		
Descrizione:	Effettuare il login con un nome utente e password corretti		
Prerequisiti:	Nella tabella degli utenti deve essere presente l'utente		
Procedura:	<ol style="list-style-type: none"> 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:	TC-002	Nome:	Login fallito
Riferimento:	REQ-1		
Descrizione:	Effettuare il login con un nome utente o password sbagliati		
Prerequisiti:	Nella tabella degli utenti non deve essere presente l'utente		
Procedura:	<ol style="list-style-type: none"> 4. Aprire il sito http://apod-kamil.labosam.cpt.local e compilare il form 5. Inserire come username e password 'sbagliato' 6. Inviare il form 		
Risultati attesi:	Rimanere nella stessa pagina e ricevere l'errore "Username o password sbagliati"		

Test Case:	TC-003	Nome:	Protezione da un SQL Injection nel form di login
Riferimento:	REQ-1		
Descrizione:	Verificare se il form è a prova di SQL Injection		
Prerequisiti:	Tabella degli utenti		
Procedura:	<ol style="list-style-type: none"> 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:	TC-004	Nome:	Protezione da un SQL Injection nel form di registrazione
Riferimento:	REQ-2		
Descrizione:	Verificare se il form è a prova di SQL Injection		
Prerequisiti:	Tabella degli utenti		
Procedura:	<ol style="list-style-type: none"> Andare nella pagina di registrazione e compilare il form provando ad eseguire un SQL Injection Inserire come username e le due password: ' or 'x'='x' Inviare il form 		
Risultati attesi:	Errore e nessuna restituzione dei dati delle tabelle del database		

Test Case:	TC-005	Nome:	Registrarsi con un nuovo account
Riferimento:	REQ-2		
Descrizione:	Crear un nuovo utente ed effettuare il login		
Prerequisiti:	Tabella degli utenti		
Procedura:	<ol style="list-style-type: none"> Dalla pagina di login premere il pulsante "Crea un account" Inserire come username "user" e le due password "user" Inviare il form 		
Risultati attesi:	Creazione dell'utente nella tabella utente ed arrivare alla pagina home con il login effettuato con successo		

Test Case:	TC-006	Nome:	Creare account con lo stesso username
Riferimento:	REQ-2		
Descrizione:	Verificare che ci siano adeguati controlli e sia impossibile creare due utenti con lo stesso username		
Prerequisiti:	Tabella degli utenti		
Procedura:	<ol style="list-style-type: none"> Andare nella pagina di register e compilare il form con il nome di un utente già esistente Inserire le due password identiche a scelta Inviare il form 		
Risultati attesi:	Errore e nessuna creazione di un nuovo utente		

Test Case:	TC-007	Nome:	Restituzione immagine di default se API restituisce un errore
Riferimento:	REQ-3		
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:	<ol style="list-style-type: none"> Effettuare il login o continuare come guest 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:	TC-008	Nome:	Aggiungere foto alla cronologia
Riferimento:	REQ-4		
Descrizione:	Aggiungere una foto alla cronologia		
Prerequisiti:	Effettuare il login Tabella cronologia		
Procedura:	<ol style="list-style-type: none"> 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:	TC-009	Nome:	Rimuovere foto dalla cronologia
Riferimento:	REQ-4		
Descrizione:	Rimuovere una foto dalla cronologia		
Prerequisiti:	Effettuare il login Tabella cronologia		
Procedura:	<ol style="list-style-type: none"> 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:	TC-010	Nome:	Aggiungere foto ai preferiti
Riferimento:	REQ-5		
Descrizione:	Aggiungere una foto nei preferiti		
Prerequisiti:	Effettuare il login Tabella preferiti		
Procedura:	<ol style="list-style-type: none"> 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:	TC-011	Nome:	Rimuovere foto dai preferiti
Riferimento:	REQ-5		
Descrizione:	Rimuovere una foto dai preferiti		
Prerequisiti:	Effettuare il login Tabella preferiti		
Procedura:	<ol style="list-style-type: none"> 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		

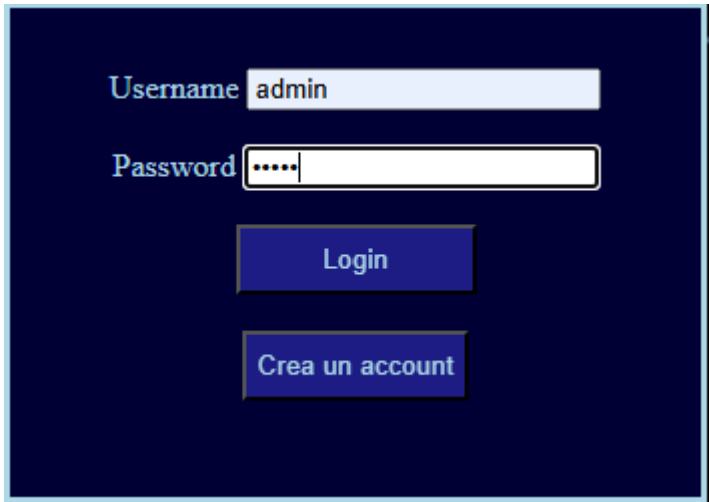

Test Case:	TC-012	Nome:	Effettuare una ricerca valida
Riferimento:	REQ-6		
Descrizione:	Effettuare una ricerca valida tramite la pagina http://apod-kamil.labosam.cpt.local/filtro.php		
Prerequisiti:	-		
Procedura:	<ol style="list-style-type: none"> 1. Dal menu in alto sinistra andare nella pagina per 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		

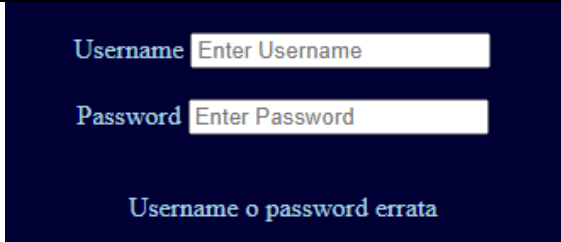
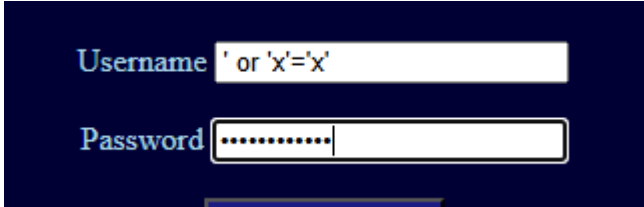
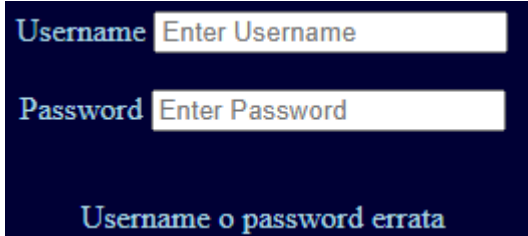
Test Case:	TC-013	Nome:	Effettuare una ricerca invalida
Riferimento:	REQ-6		
Descrizione:	Effettuare una ricerca invalida tramite la pagina http://apod-kamil.labosam.cpt.local/filtro.php		
Prerequisiti:	-		
Procedura:	<ol style="list-style-type: none"> 1. Dal menu in alto sinistra andare nella pagina 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:	TC-014	Nome:	Navigare grazie alla sequenza delle immagini
Riferimento:	REQ-7		
Descrizione:	Premere sulle foto ai lati per navigare tra i giorni		
Prerequisiti:	-		
Procedura:	<ol style="list-style-type: none"> 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:	TC-015	Nome:	Provare a navigare premendo un immagine invalida
Riferimento:	REQ-7		
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		

5.2 Risultati test

Test Case	Risultato	Descrizione	Data di test
001	Passato	<p>Risultato atteso: Arrivare nella home page http://apod-kamil.labosam.cpt.local/home.php con il login effettuato con successo</p> <p>Risultato effettivo: Dopo aver compilato il form inserendo 'admin' come username e password e premuto il pulsante 'Login', il login viene effettuato con successo.</p>  <p>Figura 9: Test1, login corretto</p> <p>Dopo venir reindirizzati nella home page, in alto in centro vediamo 'ciao admin', segno che siamo riusciti ad accedere con successo.</p>  <p>Figura 10: Login effettuato con successo</p>	18.12.2024
002	Passato	<p>Risultato atteso: Rimanere nella stessa pagina e ricevere l'errore "Username o password sbagliati"</p> <p>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"</p>	18.12.2024

		 <p>Username <input type="text" value="Enter Username"/></p> <p>Password <input type="password" value="Enter Password"/></p> <p>Username o password errata</p>	
		<p>Figura 11: Login con dati non validi fallito</p>	
		<p>Risultato atteso: Errore e nessuna restituzione dei dati delle tabelle del database</p> <p>Risultato effettivo: Compilando il form con ' or 'x'='x', così che la query sia <code>SELECT * FROM utente WHERE Username= "" or 'x'='x'</code> <code>AND Password= "" or 'x'='x'</code></p>  <p>Username <input type="text" value="' or 'x'='x'"/></p> <p>Password <input type="password" value="....."/></p> <p>Username o password errata</p> <p>Figura 12: Form Login compilato con tentata SQL Injection</p> <p>A una prima prova, mi restituiva un errore, ma dopo aver aggiunto un controllo aggiuntivo, mi restituisce un semplice errore "Username o password errata"</p> <p>Prima: Fatal error: Uncaught TypeError: mysqli_num_rows(): Argument #1 (\$result) must be of type mysqli_result, bool given</p> <p>Dopo: Ho aggiunto un controllo al tipo di \$result, se è diverso da mysqli_result ritorna errore</p>  <p>Username <input type="text" value="Enter Username"/></p> <p>Password <input type="password" value="Enter Password"/></p> <p>Username o password errata</p> <p>Figura 13: SQL Injection non riuscita</p>	
003	Passato		18.12.2024
004	Superato parzialmente	<p>Risultato atteso: Errore e nessuna restituzione dei dati delle tabelle del database</p> <p>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</p>	18.12.2024

		<div><div>Username<input type="text" value="' or '3'='3'"/></div><div>Password<input type="password" value=""/></div><div>Ripeti password<input type="password" value=""/></div></div> <p>Figura 14: SQL Injection nel form di registrazione</p> <table><tr><td>9</td><td>' or '1'='1'</td><td>c4ca4238a0b923820dcc509a6f75849b</td></tr><tr><td>10</td><td>' or '2'='2'</td><td>c81e728d9d4c2f636f067f89cc14862c</td></tr><tr><td>11</td><td>' or '3'='3'</td><td>eccbc87e4b5ce2fe28308fd9f2a7baf3</td></tr></table> <p>Figura 15: Creazione dell'utente con username la SQL injection</p>	9	' or '1'='1'	c4ca4238a0b923820dcc509a6f75849b	10	' or '2'='2'	c81e728d9d4c2f636f067f89cc14862c	11	' or '3'='3'	eccbc87e4b5ce2fe28308fd9f2a7baf3	
9	' or '1'='1'	c4ca4238a0b923820dcc509a6f75849b										
10	' or '2'='2'	c81e728d9d4c2f636f067f89cc14862c										
11	' or '3'='3'	eccbc87e4b5ce2fe28308fd9f2a7baf3										
		<p>Risultato atteso: Creazione dell'utente nella tabella utente ed arrivare alla pagina home con il login effettuato con successo</p> <p>Risultato effettivo: Riempio il form inserendo come username “Jenny” e password “abcd”</p> <div><div>Username<input type="text" value="Jenny"/></div><div>Password<input type="password" value="...."/></div><div>Ripeti password<input type="password" value="...."/></div><div>Registrati</div></div> <p>Figura 16: Creazione di un nuovo utente</p> <p>L'account viene creato con successo e veniamo automaticamente loggati con il nuovo utente</p> <div><div>Ciao Jenny, scopri il cosmo!</div><div>XAMPP for Windows - mysql -u root -p</div><table><tr><td>12</td><td>Jenny</td><td>e2fc714c4727ee9395f324cd2e7f331f</td></tr></table></div> <p>Figura 17: Creazione e login del nuovo user effettuato con successo</p>	12	Jenny	e2fc714c4727ee9395f324cd2e7f331f							
12	Jenny	e2fc714c4727ee9395f324cd2e7f331f										
005	Passato		18.12.2024									
006	Passato	<p>Risultato atteso: Errore e nessuna creazione di un nuovo utente</p> <p>Risultato effettivo: Provando a creare un utente con username ‘admin’ restituisce l'errore ‘Username già esistente’</p>	18.12.2024									

		<div><div>Username <input type="text" value="admin"/></div><div>Password <input type="password"/></div><div>Ripeti password <input type="password"/></div><div>Username già esistente</div></div> <p>Figura 18: Provare a crear un account con un username già esistente</p>													
007	Passato	<p>Risultato atteso: Tutte le foto devono essere quella di default</p> <p>Risultato effettivo: Dopo aver rimosso la richiesta all’API dal codice, nella home page vediamo solo le foto di default:</p> <div><div>DIAMO SCUSA PER</div><div>APOD NON DISPONIBILE, CHIEDIAMO SCUSA PER IL DISAGIO</div></div> <p>Figura 19: Home con foto default</p>	18.12.2024												
008	Superato parzialmente	<p>Risultato atteso: Aggiunta con successo dell’immagine e corretta visualizzazione nella pagina apposit</p> <p>Risultato effettivo: La ricerca tramite la pagina filtro va a buon fine, portandomi correttamente nella pagina home mostrando la foto voluta,</p> <div><div>04. 12. 2024</div><div>Cerca</div></div> <p>Figura 20: Filtro</p> <p>Ma nella pagina history i dati non vengono salvati correttamente, solo in parte vengono salvati</p> <table><tr><th>Id</th><th>Id_Utente</th><th>Data</th><th>url</th><th>Titolo</th><th>Descrizione</th></tr><tr><td>17</td><td>2</td><td>2024-12-12</td><td>NULL</td><td>NULL</td><td>NULL</td></tr></table> <p>Figura 21: tabella cronologia aggiornata</p>	Id	Id_Utente	Data	url	Titolo	Descrizione	17	2	2024-12-12	NULL	NULL	NULL	18.12.2024
Id	Id_Utente	Data	url	Titolo	Descrizione										
17	2	2024-12-12	NULL	NULL	NULL										

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

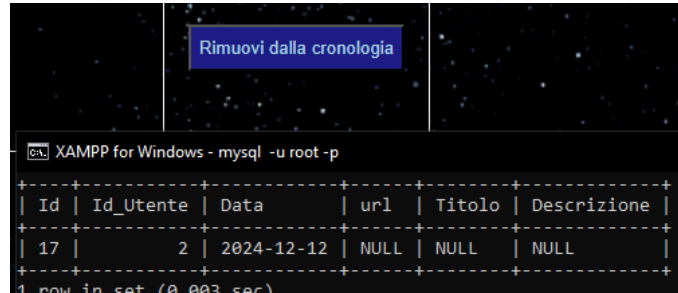


Figura 22: History.php e la tabella cronologia prima della rimozione di un valore

Dopo:



Figura 23: History.php e la tabella cronologia dopo la rimozione di un valore

009

Passato

18.12.2024

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 “Mettili nei preferiti”.

Ora vado alla pagina dei preferiti e la foto è stata aggiunta con successo:

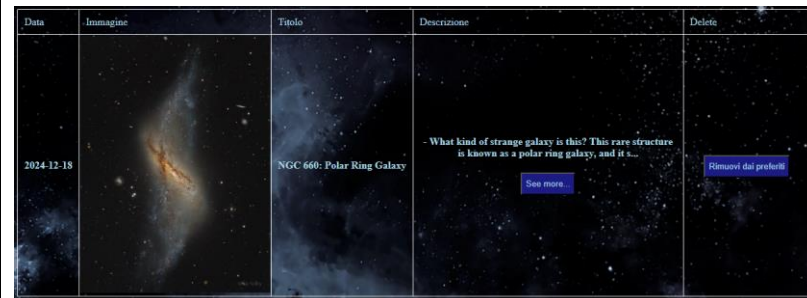

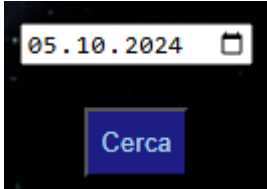


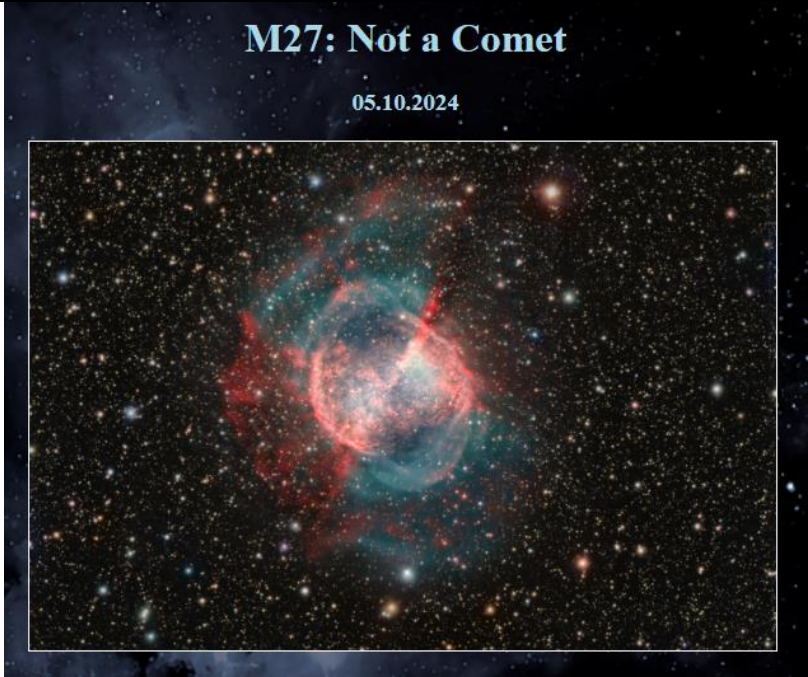
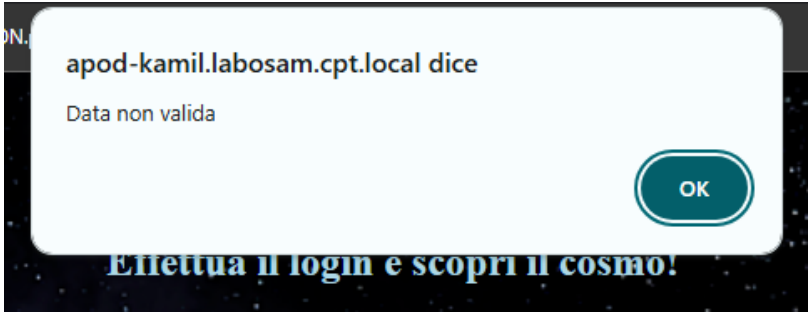
Figura 24: Pagina dei preferiti

010

Passato

18.12.2024

011	Passato	<p>Risultato atteso: Rimozione con successo dell'immagine nella pagina apposita</p> <p>Risultato effettivo: Premendo il pulsante "" la foto viene eliminata dalla tabella preferito con successo</p>  <p>Figura 25: Pagina dei preferiti e tabella preferito dopo aver rimosso un valore</p>	18.12.2024
012	Passato	<p>Risultato atteso: Venir reindirizzati alla home page e visualizzare la foto della data ricercata</p> <p>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</p>  <p>Figura 26: Ricerca di una data valida</p>	18.12.2024

		 <p>Figura 27: Ricerca della data valida finita con successo</p>	
013	Superato parzialmente	<p>Risultato atteso: rimanere nella stessa pagina e visualizzare l'errore "Data invalida"</p> <p>Risultato effettivo: Esce l'errore "Data non valida", ma veniamo comunque reindirizzati alla home page</p>	18.12.2024
014	Superato parzialmente	<p>Risultato atteso: L'immagine centrale cambia e diventa quella premuta, di conseguenza anche quelle ai lati</p> <p>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</p>	18.12.2024
015	Passato	<p>Risultato atteso: Restituizione dell'errore "Data non valida" e nessun cambiamento</p> <p>Risultato effettivo: Se si prova a premere sulla foto del giorno seguente se inesistente, ci viene rstituito l'errore "data non valida"</p>  <p>Figura 28: Errore restituito dopo aver provato a premere su una foto con data non valida</p>	18.12.2024

5.3 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.

6 Consuntivo

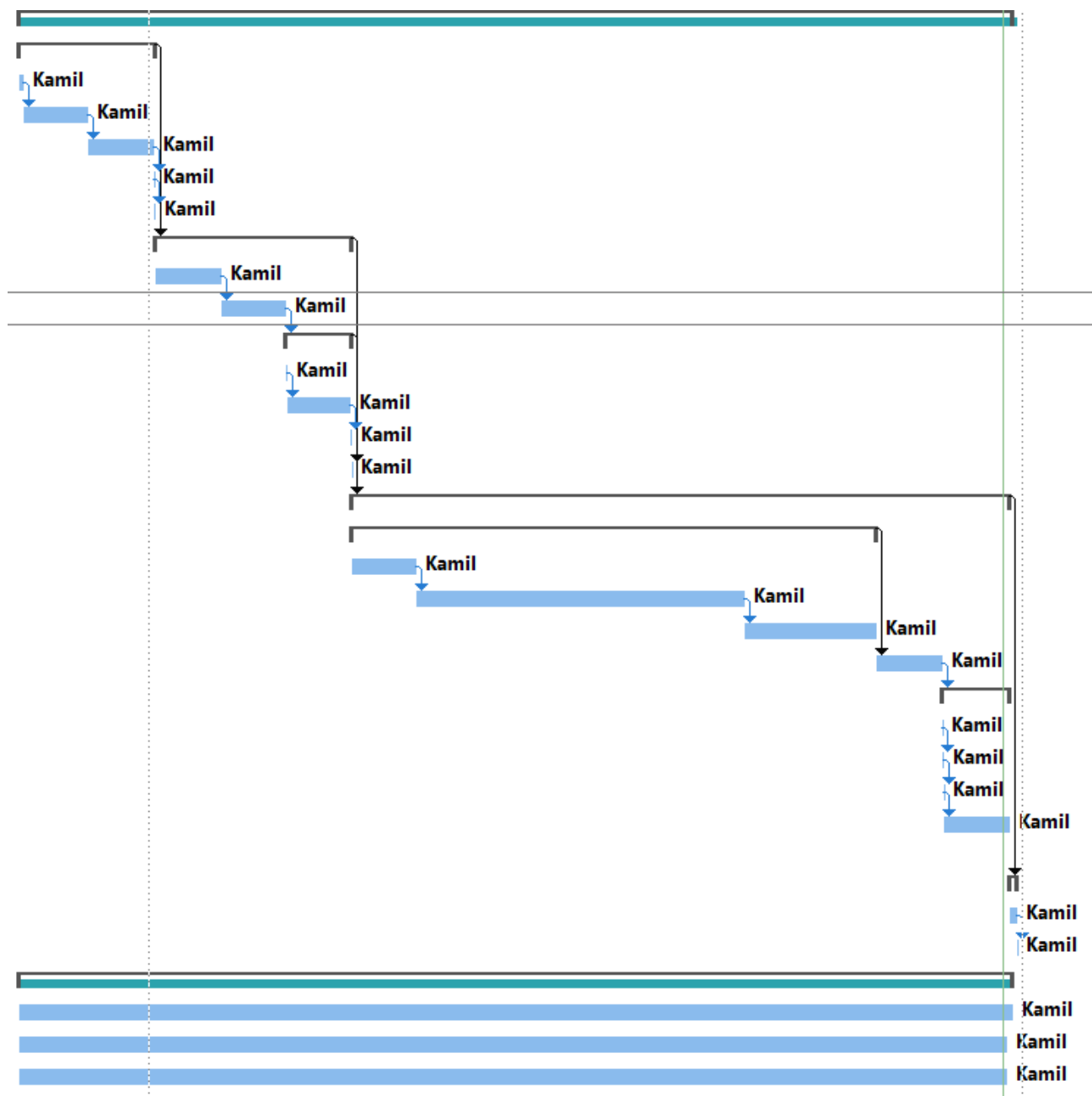


Figura 29: Gantt consuntivo

7 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.

7.1 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.

7.2 Considerazioni personali

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

8 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

9 Indice delle figure

Figura 1: Use case.....	7
Figura 2: Gantt preventivo	8
Figura 3: Schema E-R	10
Figura 4: Schema del Database	10
Figura 5: Pianificazione della GUI del Login.....	11
Figura 6: Pianificazione della GUI della Home page	11
Figura 7: Pianificazione della page dei Preferiti e cronologia.....	12
Figura 8: Design procedurale	12
Figura 9: Test1, login corretto.....	33
Figura 10: Login effettuato con successo	33
Figura 11: Login con dati non validi fallito	34
Figura 12: Form Login compilato con tentata SQL Injection	34
Figura 13: SQL Injection non riuscita	34
Figura 14: SQL Injection nel form di registrazione	35
Figura 15: Creazione dell'utente con username la SQL injection	35
Figura 16: Creazione di un nuovo utente	35
Figura 17: Creazione e login del nuovo user effettuato con successo.....	35
Figura 18: Provare a crear un account con un username già esistente.....	36
Figura 19: Home con foto default	36
Figura 20: Filtro.....	36
Figura 21: tabella cronologia aggiornata	36
Figura 22: History.php e la tabella cronologia prima della rimozione di un valore	37
Figura 23: History.php e la tabella cronologia dopo la rimozione di un valore	37
Figura 24: Pagina dei preferiti	37
Figura 25: Pagina dei preferiti e tabella preferito dopo aver rimosso un valore.....	38
Figura 26: Ricerca di una data valida	38
Figura 27: Ricerca della data valida finita con successo.....	39
Figura 28: Errore restituito dopo aver provato a premere su una foto con data non valida	39
Figura 29: Gantt consuntivo.....	40

10 Bibliografia

10.1 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