



SYA

Share Your Art

Esame di Stato 2020/2021

Massimo Piedimonte
ITI A . Volta

Prototipo online disponibile su: sya.altervista.org

Codice Open Source disponibile su: github.com/superserio/SYA

Indice

Introduzione	3
Lo schema relazionale	4
Ipotesi aggiuntive	4
Descrizione delle entità	5
Analisi dei dati	5
Analisi delle associazioni.....	6
Modello concettuale: Diagramma E/R	6
Modello logico	6
Implementazione in linguaggio SQL (DDL):	7
Alcuni esempi di query significative (QL):	7
E se “Disegno” e “Tipo” fossero due entità distinte?	9
Modello concettuale: Diagramma E/R	9
Modello logico	9
L’architettura di rete	10
Le piattaforme di <i>web hosting</i>	10
Differenza tra application server e web server: Perché usarli entrambi?	10
Il protocollo HTTP (con esempio pratico)	11
Il servizio DNS	12
Il protocollo DHCP	12
Il firewall ASA-1 e il protocollo NAT.....	13
Il protocollo HTTPS	13
La piattaforma web	14
I controlli lato server per garantire l’integrità degli attributi	14
Prevenzione alla SQL Injection (SQLI).....	16
Hashing delle password nel database	17
Gestione degli errori sui vincoli d’integrità	18
Caricare le immagini nel database	19
Gestione delle estensioni dei file caricati.....	19
Le pagine utente	20
I fogli di stile (CSS)	21
Le custom properties di CSS3	21
Responsive web design.....	21
Mediaqueries.....	22
Block Element Modifier (BEM)	23
Studio di fattibilità	24

Scelta del servizio di web hosting.....	24
Privacy Policy	25
Individuazione delle risorse umane.....	26
Profilo economico di sintesi	26
L'analisi SWOT	27
Gli obiettivi SMART	27
Bibliografia.....	28

Introduzione

SYA – Share Your Art è una piattaforma web che consente agli utenti registrati al sito di caricare le *proprie* opere (tra foto amatoriali e disegni digitali) dopo averne specificato la licenza. L'obiettivo della piattaforma è quello di fornire ai visitatori del sito la possibilità di scaricare ed utilizzare il materiale caricato in maniera totalmente gratuita ed è questo il motivo per il quale tutte le opere caricate sul sito sono coperte da licenze appartenenti alla famiglia *Creative Commons*.

È possibile interagire con un prototipo della piattaforma attraverso il seguente link:


sya.altervista.org

Inoltre, l'intero codice del sito, i comandi SQL DDL necessari alla definizione delle entità e questo stesso elaborato scritto sono disponibili sotto licenza MIT Open Source sulla piattaforma Github al seguente link: github.com/superserio/SYA


NB. Il file di configurazione e di connessione con il database (*connessione_db.php*) è stato modificato e chiaramente richiede una configurazione manuale (*host*, *username* e *password*) che varia a seconda del *web server* utilizzato e delle impostazioni di configurazione dello stesso.

SYA
Login
Registrati


Lavori più recenti




Ruri
Pubblicato da @max99
Licenza: CC BY-NC-ND



Momo
Pubblicato da @luigi99
Licenza: CC0



Asuna
Pubblicato da @max99
Licenza: CC0



okoko
Pubblicato da @luigi99
Licenza: CC0

Lo schema relazionale

In questa sezione mi occuperò di fornire, fatte le dovute ipotesi aggiuntive, il modello concettuale e logico con analisi dei dati dell'architettura informatica del progetto. Dedicherò inoltre una sezione di questo capitolo allo sviluppo progettuale di un'architettura informatica alternativa.

Ipotesi aggiuntive

Per questo progetto ho ritenuto fondamentale concentrarmi sull'aspetto implementativo delle funzionalità web che la piattaforma offre per cui ad una più complessa (ed "elegante") progettazione ho preferito prevalessse la praticità e la semplicità che una struttura informatica minimale offriva.

L'intera piattaforma è dunque basata sul solo uso di due entità: "Utenti" e "Opera". Non ho ritenuto necessario differenziare le opere per "Foto" e "Disegno" attraverso la dichiarazione di ulteriori entità per il semplice motivo che non era necessario avere altre entità che descrivessero le caratteristiche proprie di una foto o di un disegno.

Per completezza, come avrete modo di vedere, ho comunque inserito un ulteriore fase di progettazione concettuale e logica nella quale ho considerato "Foto" e "Disegno" come due entità separate.

Specifico inoltre che la [CHECK CONSTRAINT nelle versioni del DBMS MySQL superiori alla 8.0.15 non è supportata](#) motivo per il quale ho optato per controlli *server-side* con PHP sui vincoli degli attributi "**Tipo**" e "**Licenza**" dell'entità "**Opera**" e quelli dell'entità "**Utente**": "**Password**" e "**Username**".

Il progetto richiedeva di sviluppare una piattaforma nella quale gli utenti avrebbero potuto caricare le proprie foto e/o i propri disegni. Il caricamento di immagini nel database SQL è possibile in due modi:

- Si sfrutta il tipo di dato **BLOB** e si carica "direttamente" l'immagine nel database in formato binario.
- Si salva l'immagine in locale nel **web server** e nel database si registra il percorso in cui l'immagine è salvata.

Come spiegato in un articolo di Jim Gray (sviluppatore Microsoft) [To BLOB or Not To BLOB](#) il tipo di dato BLOB è preferibile utilizzarlo se le immagini da salvare non superano i 256Kilobyte ed è questa la ragione per il quale ho deciso di optare per la seconda possibilità.

L'**autenticazione** è possibile attraverso il nome utente e la password in quanto in primis non era nemmeno previsto che implementassi un meccanismo di autenticazione ed ho preferito concentrarmi sugli aspetti previsti dalla traccia ma, cosa forse più importante, per un progetto "simulato" come questo credo che un'autenticazione semplice (ma nemmeno troppo, come vedremo a breve) rende bene l'idea del funzionamento complessivo della piattaforma.

Molti degli attributi delle entità specificate (come la data di pubblicazione o l'anagrafica dell'autore) non sono utilizzati praticamente nella piattaforma ma sono registrati per rendere la stessa scalabile a futuri, eventuali, aggiornamenti.

Dulcis in fundo, come già specificato in precedenza, ho scelto (un po' per attenermi a quella che era la consegna della traccia e un po' per ragioni etiche) di consentire all'utente di caricare del materiale solo se specificata la licenza d'uso (come da traccia) la quale, tuttavia, deve necessariamente appartenere alla famiglia di licenze Creative Commons utilizzate [quando un autore vuole concedere ad altri il diritto di usare o modificare un'opera che lui stesso \(l'autore\) ha creato.](#)

Descrizione delle entità

L'entità **"Utente"** descrive gli utenti della piattaforma, contiene una breve anagrafica dell'utente e le due informazioni necessarie al *login* in piattaforma: *username* e *password*.

L'entità **"Opera"** descrive le opere contenute nella piattaforma e mostrate nella *homepage* del sito web. L'entità contiene tutta una serie di attributi relativi alle informazioni dell'opera pubblicata e, attraverso il tipo di dato [ENUM](#) di MySQL e un controllo *server-side* con PHP, valori predefiniti per gli attributi **"Tipo"** (che può assumere due valori: foto o disegno) e **"Licenza"** (l'utente può scegliere tra una serie di licenze della categoria *Creative Commons*).

Entrambe le tabelle hanno come chiave primaria un codice identificativo che si auto-incrementa all'aggiunta del record nella tabella.

Analisi dei dati

<i>Utente</i>	<ul style="list-style-type: none"> • ID: Numero intero che si auto-incrementa all'aggiunta del record nella tabella. (INTERO, CHIAVE PRIMARIA E DUNQUE NOT NULL). • Nome: Il nome dell'utente registrato. (STRINGA, NOT NULL). • Cognome: Il cognome dell'utente registrato. (STRINGA, NOT NULL). • Username: Il nome utente dell'utente registrato che è univoco per ogni utente della tabella ma non costituisce la chiave primaria. (STRINGA, NOT NULL e UNIQUE). • Password: La password dell'utente registrato memorizzata come stringa. Le password sono salvate nella loro forma crittata secondo il principio di sicurezza che né eventuali malintenzionati né lo stesso autore del sito possono risalire alla password originale pur avendo accesso al database. (STRINGA, NOT NULL).
---------------	--

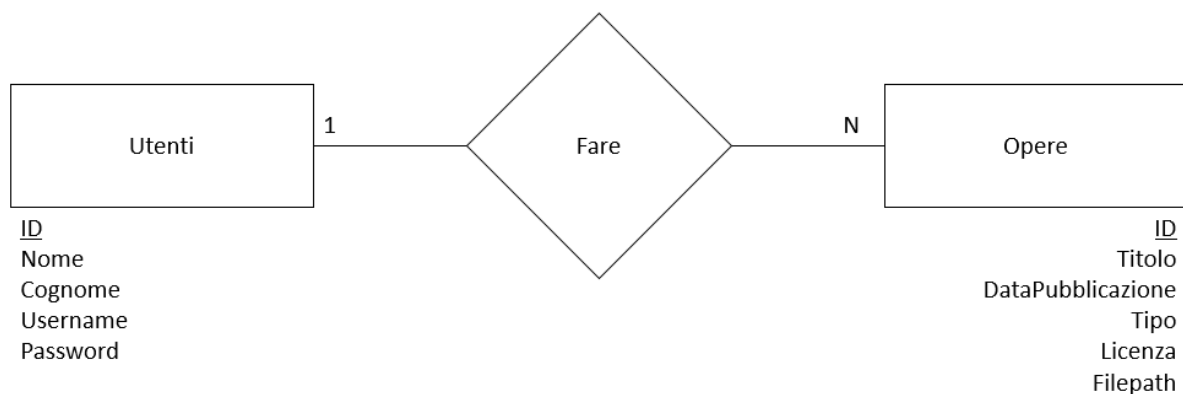
<i>Opera</i>	<ul style="list-style-type: none"> • ID: Numero intero che si auto-incrementa all'aggiunta del record nella tabella. (INTERO, CHIAVE PRIMARIA E DUNQUE NOT NULL). • Titolo: Il titolo dell'opera caricata. (STRINGA, NOT NULL). • DataPubblicazione: La data di pubblicazione in formato yyyy-mm-dd. (DATA, NOT NULL). • Tipo: Attributo che identifica se l'opera caricata è una foto o un disegno. (ENUM - STRINGA, NOT NULL). • Licenza: Attributo che identifica il tipo di licenza <i>Creative Commons</i> dell'opera caricata. (ENUM - STRINGA, NOT NULL). • Filepath: Percorso dell'immagine caricata nel file system del web server (STRINGA, NOT NULL).
--------------	--

Analisi delle associazioni

L'associazione **FARE** che intercorre tra le due entità è un'associazione di tipo 1 a N. In particolare ogni utente può caricare da 0 (*partecipazione minima*) a N (*partecipazione massima*) opere mentre ogni opera dev'essere caricata da uno ed un solo utente (*partecipazione minima/massima*).

NB. Nel database finale non ho chiamato l'attributo "Fare" ma "Autore" in quanto mi sembrava più consono al contesto. Chiaramente "Fare" descrive meglio l'associazione ed è per questo che ho deciso di lasciarlo in questa parte progettuale.

Modello concettuale: Diagramma E/R



Modello logico

- **Utenti**(ID, Nome, Cognome, Username, Password);
- **Opere**(ID, Titolo, DataPubblicazione, Tipo, Licenza, Filepath, [Fare]);

Implementazione in linguaggio SQL (DDL):

```

1. # creazione del database
2. CREATE DATABASE sya;
3. USE sya;
4.
5. # creazione delle tabelle
6. CREATE TABLE utenti(
7.     ID INT AUTO_INCREMENT,
8.     Nome VARCHAR(255) NOT NULL,
9.     Cognome VARCHAR(255) NOT NULL,
10.    Username Varchar(20) NOT NULL,
11.    Password Varchar(50) NOT NULL,
12.    PRIMARY KEY(ID)
13.);
14.
15. CREATE TABLE opera(
16.     ID INT AUTO_INCREMENT,
17.     Titolo VARCHAR(255) NOT NULL,
18.     DataPubblicazione DATE NOT NULL,
19.     Autore INT NOT NULL,
20.     Tipo ENUM('Foto', 'Disegno') NOT NULL,
21.     Licenza ENUM('CC0', 'CC BY-NC-ND', 'CC BY-NC-SA', 'CC BY-SA', 'CC BY-ND', 'CC
    BY-NC', 'CC BY') NOT NULL,
22.     PRIMARY KEY(ID),
23.     FOREIGN KEY(Autore) REFERENCES utenti(ID)
24.);
25. ALTER TABLE opera ADD COLUMN Filepath VARCHAR(255) NOT NULL;
26. ALTER TABLE utenti ADD UNIQUE (Username);
27. ALTER TABLE utenti CHANGE Password Pass VARCHAR(255) NOT NULL;

```

Alcuni esempi di query significative (QL):

```

1. # seleziona gli utenti che non hanno caricato niente
2. SELECT u.Username
3. FROM utenti u LEFT JOIN opera o ON u.ID=o.Autore
4. WHERE o.Titolo IS NULL;
5.
6. # seleziona per ogni autore il numero di opere caricate
7. SELECT Username, COUNT(*) AS 'Disegni caricati'
8. FROM opera O INNER JOIN utenti U ON O.Autore=U.ID
9. GROUP BY Autore;
10.
11. # visualizza il numero di disegni caricati per ogni autore
12. SELECT Username, COUNT(*) AS 'Disegni caricati'
13. FROM opera O INNER JOIN utenti U ON O.Autore=U.ID
14. WHERE Tipo='Disegno'
15. GROUP BY Autore;
16.
17. # visualizza il numero di opere caricate per ogni autore
18. SELECT Username, COUNT(Tipo)
19. FROM utenti u LEFT JOIN opera o ON u.ID=o.Autore
20. GROUP BY Username;
21.
22. # crea una vista per il numero di opere caricate per ogni autore
23. CREATE VIEW opere_caricate AS

```



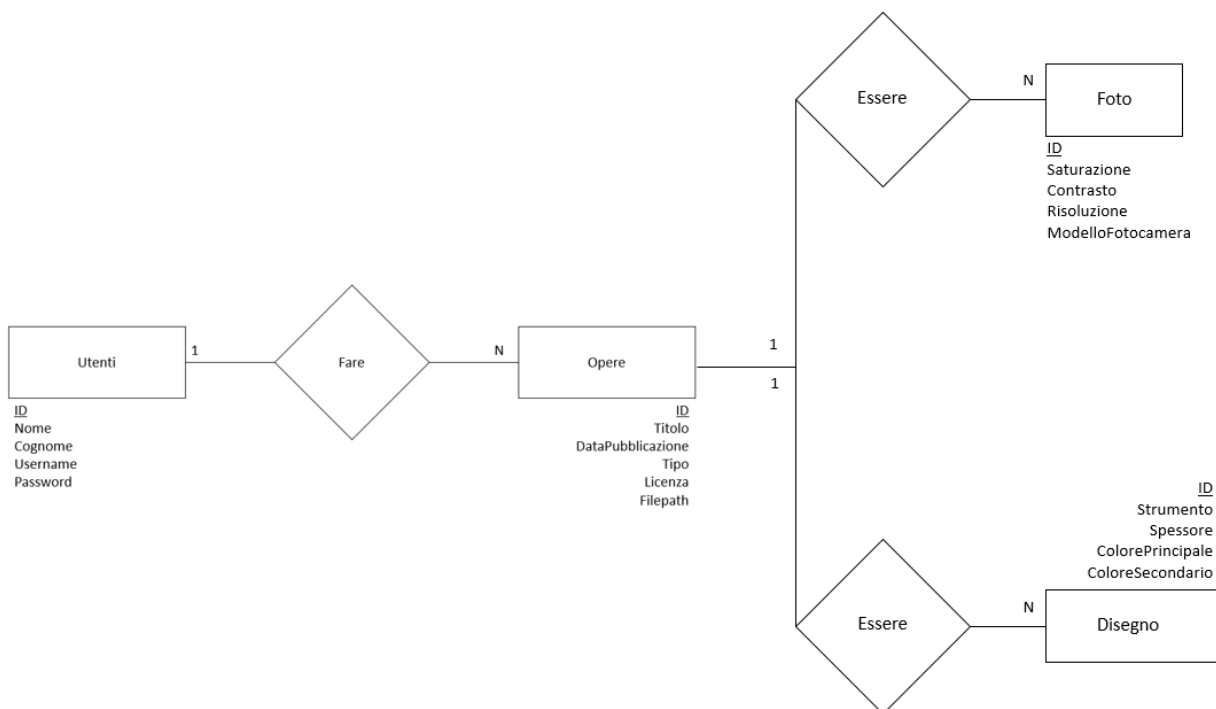
```
24. SELECT Username, COUNT(Tipo)
25. FROM utenti u LEFT JOIN opera o ON u.ID=o.Autore
26. GROUP BY Username;
27.
28. # seleziona gli utenti che non hanno caricato nessun disegno
29. SELECT Username
30. FROM utenti
31. WHERE Username NOT IN (SELECT Username
32.                          FROM opera O INNER JOIN utenti U ON O.Autore=U.ID
33.                          WHERE Tipo='Disegno'
34.                          GROUP BY Autore);
35.
36.
37.
38. # seleziona gli utenti che non hanno caricato nessun disegno
39. # ma che hanno caricato almeno una foto
40. SELECT Username
41. FROM utenti
42. WHERE Username NOT IN (SELECT Username
43.                          FROM opera O INNER JOIN utenti U ON O.Autore=U.ID
44.                          WHERE Tipo='Disegno'
45.                          GROUP BY Autore)
46. AND Username NOT IN (SELECT u.Username
47.                        FROM utenti u LEFT JOIN opera o ON u.ID=o.Autore
48.                        WHERE o.Titolo IS NULL);
```

E se “Disegno” e “Tipo” fossero due entità distinte?

A scopo di completezza in questa sezione mi occuperò di fornire brevemente la progettazione concettuale attraverso il **diagramma E/R** e il relativo modello logico di un’architettura informatica alternativa nella quale “Disegno” e “Tipo” vengono considerate come due entità distinte.

Modello concettuale: Diagramma E/R

Il seguente schema mostra l’aggiunta delle due entità “**Foto**” e “**Disegno**” attraverso l’associazione “**ESSERE**” con l’entità “**Opera**”.



PS. In maniera ancora alternativa si sarebbe potuto sostituire l’entità “**Opera**” completamente e lasciare solamente le due entità “**Foto**” e “**Disegno**”. In questo caso, tuttavia, avremmo avuto tutta una serie di attributi duplicati tra le due tabelle.

Modello logico

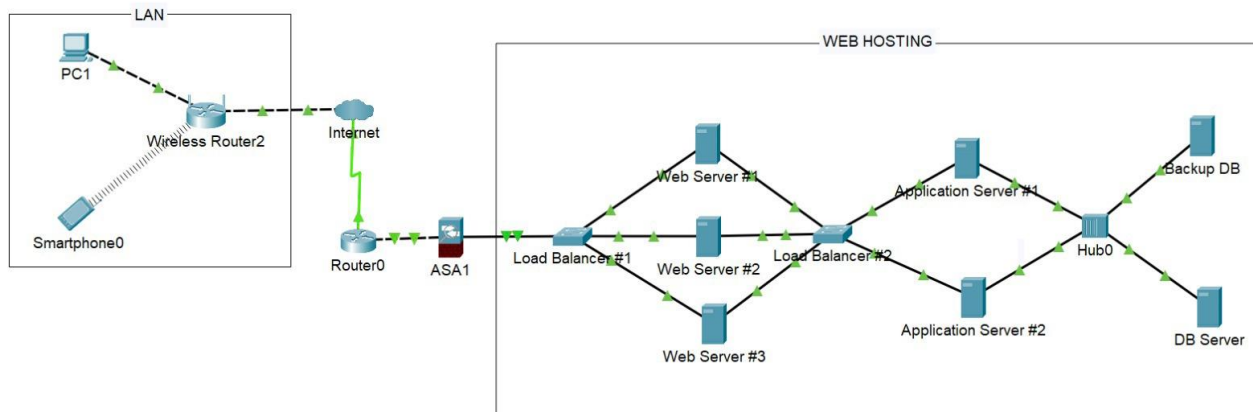
- **Utenti**(ID, Nome, Cognome, Username, Password);
- **Opere**(ID, Titolo, DataPubblicazione, Tipo, Licenza, Filepath, [Fare]);
- **Foto**(ID, Saturazione, Contrasto, Risoluzione, ModelloFotocamera, [Essere]);
- **Disegni**(ID, Saturazione, Contrasto, Risoluzione, ModelloFotocamera, [Essere]);

PS. L’attributo “**Essere**” è la chiave di riferimento esterna verso la tabella “**Opera**” che, pur avendo lo stesso nome in entrambe le entità “**Foto**” e “**Disegno**”, conterrà (ovviamente) valori diversi della chiave primaria dell’entità a cui sono associati.

L'architettura di rete

Per questioni di coerenza con lo studio di fattibilità e considerando che il progetto nasce come *startup* mi è sembrato il caso di non investire in costosi server personali per la pubblicazione del sito internet ma ho invece preferito optare per una soluzione professionale ma che riduceva gli investimenti iniziali: il *web hosting*.

Vediamo adesso lo schema di rete che simula una connessione tra un cliente remoto e il servizio di web hosting nella quale si trova la nostra piattaforma web, dopodiché mi occuperò di descriverne i protocolli:



Le piattaforme di *web hosting*

L'architettura tipica di un'applicazione web si compone di tre elementi che comunicano tra loro:

- Il **database**
- Un **application server**
- Il **web server**

Le piattaforme di *web hosting* si compongono dei medesimi elementi seppur (dovendo *hostare* un gran numero di siti web) in quantità maggiore.

Vediamo però di descrivere quelle che sono caratteristiche peculiari di una tipica architettura di rete per una piattaforma di *web hosting* e, in particolare, del diagramma di rete di cui sopra.

In primis, partendo da destra, abbiamo aggiunto un *database di backup* il cui ruolo è quello di offrire ridondanza dei dati in caso di guasti al database principale.

Vi sono inoltre due *switch* che simulano il funzionamento dei **Load Balancer** ovvero dei dispositivi in grado di bilanciare il carico di informazioni e reindirizzare le richieste dell'utente alle risorse designate. Questi dispositivi, com'è facile immaginare, sono di estrema importanza per ogni piattaforma di *web hosting*.

Differenza tra application server e web server: Perché usarli entrambi?

La principale differenza che vi è tra un **server web** e un **application server** consiste nel fatto che il primo è destinato a servire pagine statiche (ovvero ciò che l'utente finale visualizzerà nel suo browser web) mentre il secondo è responsabile di generare contenuto dinamico per mezzo di appositi linguaggi di scripting *server-side* (es. PHP, Node.js).

È importante notare che l'introduzione di un *application server* (specialmente nel caso di una piattaforma di web hosting come quella in cui è ospitato il nostro sito) migliora notevolmente le performance generali del sito in quanto delega a un'entità specifica tutti i processi computazionali che avvengono per mezzo di linguaggi lato server lasciando al *web server* il solo compito di eseguire il *rendering* delle pagine statiche HTML, CSS e (eventualmente) Javascript.

Il protocollo HTTP (con esempio pratico)

Le trasmissioni tra l'utente remoto nella LAN e la nostra piattaforma avvengono per mezzo di richieste (e di risposte) HTTP.

Ad ogni richiesta HTTP che il browser dell'utente invia al nostro sito sono associate tutta una serie di informazioni (*la versione del web server, l'indirizzo IP da cui arriva la richiesta, etc.*) e in particolare nella *start-line* della richiesta vi è il **metodo**.

Ad esempio, quando l'utente accede alla pagina di registrazione (vedi figura in basso) per creare un nuovo profilo invierà una richiesta di tipo **POST** al server che, in questo caso, elaborerà i dati inviati con un linguaggio di scripting (PHP) per registrare l'utente nel database e fornirgli un interfaccia grafica personalizzata.

NB. In tutto questo processo sono stati coinvolti tutti e tre i dispositivi essenziali di una piattaforma web descritti nel paragrafo 2.0.1



Nome
Massimo

Cognome
Piedimonte

Username
massimo1999
Lo username non può superare i 20 caratteri.

Password
.....
La lunghezza della password dev'essere compresa tra 8 caratteri e 50 caratteri.

Invia

Form di registrazione della piattaforma



SYA Profilo Esci

Bentornato massimo1999

Hai qualcosa da condividere con la community di SYA? Comincia subito a caricare i tuoi lavori!

Carica una foto o un disegno

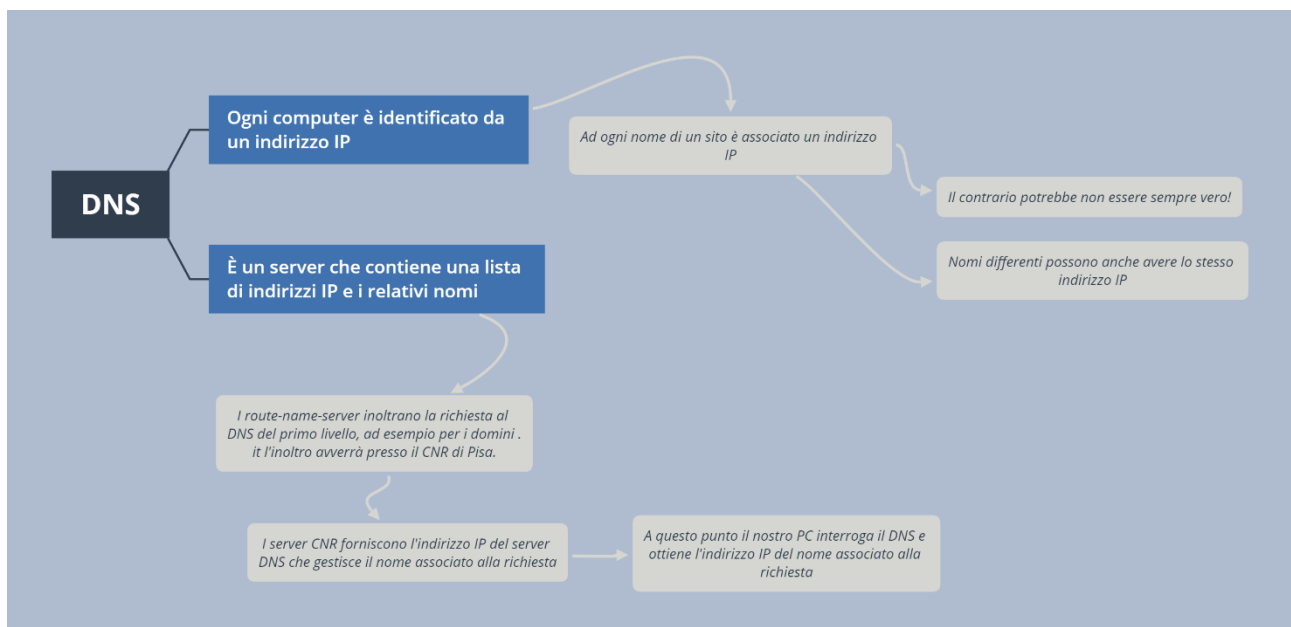
Pagina per utenti registrati al sito

Il servizio DNS

Seppure non presente nello schema di rete disegnato è chiaro che ogni utente che avrà accesso alla nostra piattaforma usufruirà del servizio DNS con il quale i nomi di dominio vengono convertiti in indirizzi IP.

In generale quando un utente inserisce nella barra degli indirizzi il nome di dominio del nostro sito web la richiesta viene instradata a un dominio di DNS di primo livello (la locazione fisica di questi server varia a seconda del dominio di primo livello, ad esempio per *.it* l'inoltro avverrà presso il CNR di Pisa) i quali forniscono l'indirizzo IP dei server DNS che gestiscono il nome di dominio associato alla richiesta. A questo punto il PC interroga il DNS e ottiene l'indirizzo IP associato alla richiesta.

Nel seguente diagramma, realizzato nel corso di quest'anno, ho provato a sintetizzare quanto appena descritto:



Il protocollo DHCP

Il servizio DHCP è configurato direttamente sul router in maniera tale da assegnare automaticamente gli indirizzi IP ai dispositivi della rete di *web hosting*.

Di seguito i comandi utilizzati per configurare il servizio DHCP sul *Router0* che comunica con la rete *Web Hosting* attraverso un'interfaccia Gigabit Ethernet:

```
1. Router(config-if)# ip dhcp pool gig0-dhcp
2. Router(dhcp-config)# network 192.168.1.5 255.255.255.0
3. Router(dhcp-config)# default-router 192.168.1.254
4. Router(dhcp-config)# dns-server 192.168.1.253
```

NB. La configurazione del servizio DHCP sul router va fatta in modalità di **configurazione interfaccia** (*config-if*).

Il firewall ASA-1 e il protocollo NAT

L'architettura di rete implementa una **zona demilitarizzata informatica (DMZ)** attraverso l'uso di un firewall CISCO ASA-1 che si occupi di filtrare il traffico in entrata e in uscita dalla rete *Web Hosting*. Per quanto riguarda le reti *LAN* il firewall è implementato a livello software.

Dati i limiti del protocollo Ipv4 è inoltre previsto (ma non implementato concretamente) il servizio offerto dal protocollo **NAT** situato al terzo livello (*rete*) del modello ISO/OSI. Con il protocollo NAT gli indirizzi IP locali della rete *Web Hosting* e delle diverse reti *LAN* vengono convertiti (in gergo: **nattare**) in un unico indirizzo IP pubblico (univoco per ogni rete e assegnato, nel caso degli utenti domestici, dal loro provider di servizi Internet) che comunica con l'esterno. Tale protocollo consente inoltre di fornire una simulazione più accurata della rete nella quale sia l'utente appartenente alla rete *LAN* che uno qualsiasi dei dispositivi appartenente alla rete *Web Hosting* possono comunicare liberamente anche nel caso (molto probabile) in cui posseggano lo stesso indirizzo IP locale.

Il protocollo HTTPS

Al protocollo HTTP che abbiamo definito in maniera pragmatica nel paragrafo 2.0.3 si aggiunge un ulteriore strato volto a garantire la **confidenzialità** delle informazioni scambiate tra l'utente e la piattaforma web. Tale strato è anch'esso un protocollo situato tra il livello di trasporto (nel quale abbiamo il **protocollo TCP**) e il livello di sessione (quinto livello) nel modello ISO/OSI e prende il nome di **SSL** o **TLS** (quest'ultima è in effetti la versione più recente del protocollo **SSL**).

La *confidenzialità* delle informazioni è garantita attraverso la crittografia a chiave asimmetrica (che si occupa anche di autenticare il mittente e il destinatario del messaggio attraverso l'uso di due chiavi distinte) dei dati cosicché, anche nel caso la nostra piattaforma dovesse subire attacchi informatici di tipo *Man in The Middle (MITM)* le informazioni conservino un aspetto illeggibile e intraducibile da parte di terzi.

La piattaforma web

Entriamo adesso nel dettaglio in merito agli aspetti più interessanti dello sviluppo della piattaforma web. Come già anticipato, l'intero codice sorgente è disponibile su Github per cui, anche per una questione di praticità, non andrò a copiare ed incollare il codice di ogni file qui nell'elaborato scritto ma mi limiterò ad inserire dei piccoli *snippet* di codice che mi siano d'aiuto nella descrizione di alcune caratteristiche (a mio avviso) particolarmente significative relative agli aspetti implementativi della piattaforma web.

I controlli lato server per garantire l'integrità degli attributi

Nella capitolo "*Lo schema relazionale*" abbiamo appurato l'impossibilità di sfruttare la **CHECK CONSTRAINT** per definire alcuni vincoli d'integrità degli attributi delle due entità "**Utente**" e "**Opera**".

Per elencare sinteticamente i controlli lato server che sono stati necessari a sostituzione della *CHECK* abbiamo:

1. La lunghezza dello *Username* dell'utente doveva essere inferiore a 20 caratteri.
2. La lunghezza della *Password* dell'utente doveva essere compresa tra gli 8 e i 50 caratteri.
3. I valori assunti dall'attributo *Licenza* dell'entità "*Opera*" potevano essere:
 - *CC0* : Licenza *Creative Commons* senza vincoli;
 - *CC BY*: Licenza *Creative Commons* con diritti di attribuzione;
 - *CC BY-NC*: Licenza *Creative Commons* con diritti di attribuzione e per uso non commerciale;
 - *CC BY-ND*: Licenza *Creative Commons* con diritti di attribuzione e con divieto di opere derivate dalla stessa;
 - *CC BY-SA*: Licenza *Creative Commons* con diritti di attribuzione e con obbligo di condividere l'opera allo stesso modo;
 - *CC BY-NC-SA*: Licenza *Creative Commons* con diritti di attribuzione e con obbligo di condividere l'opera allo stesso modo per uso non commerciale;
 - *CC BY-NC-ND*: Licenza *Creative Commons* con diritti di attribuzione e con divieto di opere derivate dalla stessa il tutto per uso non commerciale.
4. I valori assunti dall'attributo *Tipo* dell'entità "*Opera*" potevano essere:
 - *Foto*
 - *Disegno*

Tali controlli vengono individuati dai seguenti *snippet* di codice PHP:

1. Ci assicuriamo che la lunghezza di *Username* e *Password* siano quelli identificati:

```
1. if((strlen($password) >= 8 && strlen($password) < 50) && strlen($username) < 20)
   {
2.     // ...
3. }
```

2. Per questioni di praticità ho pensato di sfruttare un form HTML per consentire all'utente di inviare al server solo i valori consentiti per gli attributi *Tipo* e *Licenza* dell'entità "**Opera**" come mostrato nella figura sottostante:

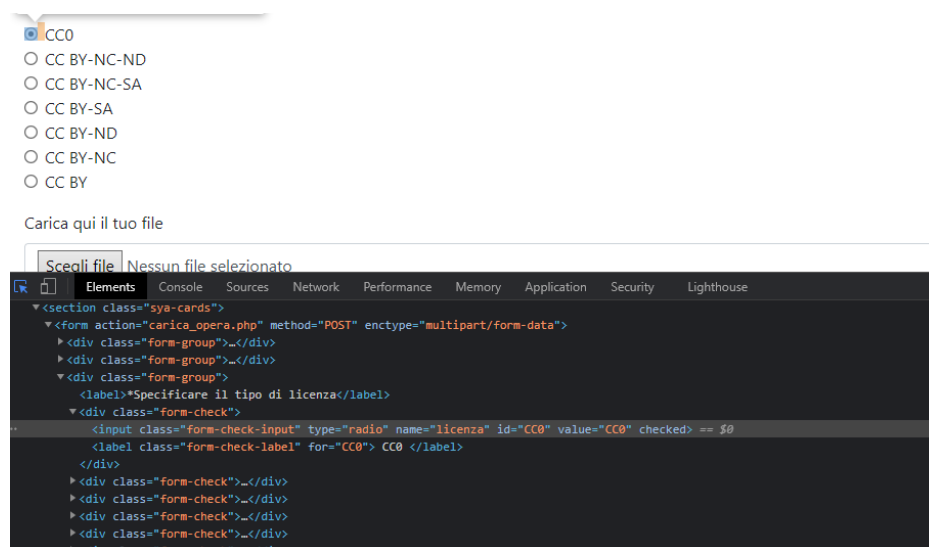
*Specifica se si tratta di una foto o di un disegno

- ☒ Foto
☐ Disegno

*Specificare il tipo di licenza

- ☒ CC0
☐ CC BY-NC-ND
☐ CC BY-NC-SA
☐ CC BY-SA
☐ CC BY-ND
☐ CC BY-NC
☐ CC BY

Tutto ciò crea un vantaggio per lo sviluppatore che può risparmiarsi di scrivere righe e righe di controlli sulla *Licenza* ma uno svantaggio per l'utente in quanto i controlli fatti esclusivamente lato client con HTML possono facilmente essere "evasi" per mezzo dello strumento *Ispeziona elemento* di cui ogni browser è dotato:



Il form prima di subire modifiche

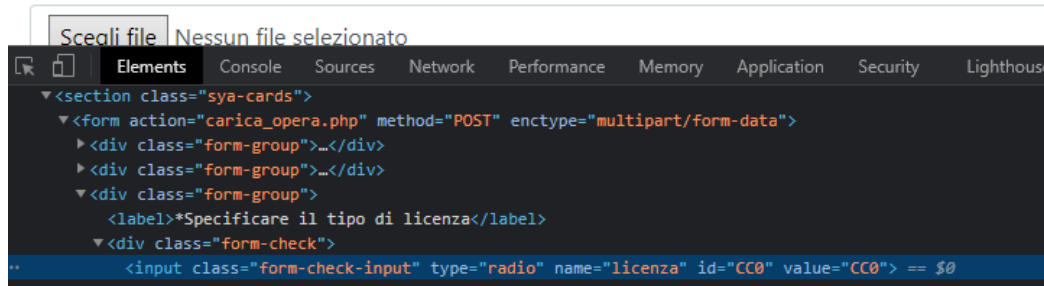
Notare in particolare la presenza, all'interno dell'input HTML, della parola chiave **checked** grazie al quale è possibile selezionare automaticamente una licenza di partenza. E se l'utente provasse a rimuovere questo vincolo e inviare il form? Nella seguente immagine vediamo come sarebbe semplice rimuovere la parola chiave attraverso lo strumento

Ispeziona Elemento e che soluzioni abbiamo implementato server-side per ovviare al problema:

*Specificare il tipo di licenza

- ☐ CC0
- ☐ CC BY-NC-ND
- ☐ CC BY-NC-SA
- ☐ CC BY-SA
- ☐ CC BY-ND
- ☐ CC BY-NC
- ☐ CC BY

Carica qui il tuo file



Il form dopo aver subito modifiche

Dunque inviando il form l'attributo *Licenza* sarebbe risultato non impostato (**undefined**). Per ovviare al problema sono stati necessari un paio di controlli PHP per i due attributi *Licenza* e *Tipo* (i quali funzionano esattamente allo stesso modo):

```
1. $opera = isset($_POST['opera']) ? mysqli_real_escape_string($conn, $_POST['opera']) : "";
2. $licenza = isset($_POST['licenza']) ? mysqli_real_escape_string($conn, $_POST['licenza']) : "";
```

Dunque le due variabili “*\$opera*” e “*\$licenza*” sono impostate come stringhe vuote (“”) se l’utente non imposta i valori nel form come mostrato precedentemente. A questo punto non resta che verificare se i valori delle due variabili sono o meno delle stringhe vuote e agire di conseguenza nel caso lo siano avvertendo l’utente che c’è stato un errore nel caricamento della sua immagine (vedi “Gestione degli errori sui vincoli d’integrità”):

```
1. if($titolo != "" && $opera != "" && $licenza != "") {
2.     // ...
3. }
```

Prevenzione alla SQL Injection (SQLI)

La SQL Injection (**SQLI**) è un attacco informatico nel quale un malintenzionato può inserire nei campi di un form semplici stringhe di codice PHP e SQL e inviarle al server che, se non configurato correttamente, elaborerà il codice. Questo può essere fatto all’interno di un form o direttamente nell’URL di una richiesta HTTP di tipo GET.

Non è difficile capire che dare a chiunque la possibilità di *iniettare* codice nell'*application server* mette a rischio l'integrità del nostro database e la confidenzialità dei dati ivi presenti.

PHP mette a disposizione una funzione volta ad effettuare un'operazione di *escaping* dei caratteri in modo tale da evitare scenari di questo tipo.

Per lo sviluppo della piattaforma ho utilizzato la funzione `mysqli_real_escape_string()` per ogni singolo valore inviato dall'utente attraverso un form (a prescindere dal metodo).

Vediamo un estratto del file *registrazione.php* nella quale ho fatto uso della suddetta funzione:

```
1. $nome = mysqli_real_escape_string($conn, $_POST['nome']);
2. $cognome = mysqli_real_escape_string($conn, $_POST['cognome']);
3. $username = mysqli_real_escape_string($conn, $_POST['username']);
4. $password = mysqli_real_escape_string($conn, $_POST['password']);
```

NB. Essendo una versione “demo” e semplificata la nostra piattaforma non prevede anche il *binding dei parametri* che avrebbe garantito un ulteriore livello di sicurezza dei dati.

Hashing delle password nel database

Nel fortuito e malaugurato caso in cui un malintenzionato ha accesso alle informazioni contenute nel nostro database siamo comunque obbligati, in quanto sviluppatori (e per legge), a garantire la confidenzialità delle informazioni sensibili dell'utente ed è qui che entra in gioco un altro aspetto relativo alla sicurezza dei dati: l'**hashing delle password**.

In generale, una *funzione hash* ha lo scopo di prendere in input una stringa di N caratteri variabili e restituire in output una stringa univoca di M caratteri fissi. Quest'operazione è particolarmente utile in svariate situazioni anche nel campo delle reti in quanto permette di verificare l'**integrità** di un messaggio scambiato tra due nodi.

Il tutto, fortunatamente, si rivela essere relativamente semplice da implementare per mezzo di sole due funzioni PHP:

- `password_hash()` che prende come parametro la password a cui è necessario applicare la funzione hash e il *metodo di hashing*.
- `password_verify()` che consente di verificare se la password inserita dall'utente corrisponde a quella presente nel database.

È importante notare come quest'ultima funzione non si limiti a riapplicare la funzione hash sulla stringa inserita dall'utente in quanto ogni volta che si applica tale funzione l'output generato è sempre diverso (e prende in considerazione l'esatto momento in cui la funzione è stata applicata a una stringa). Il processo è dunque **irreversibile** e ciò consente di garantire in maniera abbastanza *affidabile* che non si possa risalire facilmente alla password memorizzata nel database per mezzo di operazioni crittoanalitiche basate sull'algoritmo (l'unico modo sarebbe andare di **forza bruta** ma ci vorrebbero anni se non secoli a seconda della complessità dell'output generato dalla funzione hash per ottenere risultati apprezzabili).

Di seguito vediamo un estratto del codice PHP in cui viene applicata la funzione hash alla password inserita dall'utente che desidera registrarsi al sito:

```
1. $hashed_password = password_hash($password, PASSWORD_DEFAULT);
```

E ancora una volta di come si sfrutta la funzione `password_verify()` per autenticare l'utente che desidera accedere alla piattaforma:

```
1. if(password_verify($password, $row['Pass']))
```

NB. Se è vero che le funzioni hash generano sempre un output univoco ciascuna stringa che esse prendono in input c'è da sottolineare (più per curiosità che perché sia realmente utile saperlo) che c'è una minuscola probabilità (praticamente "impossibile") che una funzione hash possa generare per due stringhe input diverse la stessa **impronta** (così è chiamato l'output del messaggio nel campo delle reti).

Gestione degli errori sui vincoli d'integrità

In caso di campi non compilati nel form o in cui i valori inseriti non rispettano i vincoli d'integrità definiti allora è necessario avvertire l'utente con un messaggio d'errore. In generale sarebbe preferibile personalizzare il messaggio d'errore in base alle circostanze. Ad esempio se si vuole accedere a una piattaforma potrebbe non essere il caso dare all'utente l'informazione che ha inserito lo username errato e limitarsi a dire che *"le credenziali inserite sono errate"* mentre in altri casi un messaggio d'errore personalizzato può realmente essere d'aiuto all'utente per capire dove ha "sbagliato" nella compilazione del form.

Nel caso della piattaforma realizzata ho predisposto una gestione generica degli errori per mezzo di un parametro GET.

Vediamo in PHP come si traduce il tutto prendendo ad esempio il file *login.php*:

```
1. if($username != "" && $password != "") {
2.     // ...
3.     if(mysqli_num_rows($rows) == 1) {
4.         // ...
5.         if(password_verify($password, $row['Pass'])) {
6.             // ...
7.         } else header("Location: login.php?err=true");
8.     } else header("Location: login.php?err=true");
9. } else header("Location: login.php?err=true");
```

Com'è possibile notare ogni qualvolta una delle condizioni non è verificata l'utente viene rimandato alla pagina di login e viene aggiunto un parametro GET di nome *err* impostato a *"true"* (come *stringa di testo*, non come valore *booleano*) che identifica che c'è stato un errore.

Non resta che mostrare a schermo il generico messaggio d'errore (generalmente è mostrato immediatamente sopra o sotto al form):

```
1. <form action="login.php" method="POST">
```

```

2.     <?php
3.         if(isset($_GET['err'])) {
4.             $err = mysqli_real_escape_string($conn, $_GET['err']);
5.             if($err == "true") echo "<p class='error'>Credenziali errate!</p>";
6.         }
7.     ?>
8. <!-- ... -->

```

Caricare le immagini nel database

Uno dei problemi che ho già affrontato nelle “Ipotesi aggiuntive” è quello relativo al caricamento delle immagini nel database. Alla fine, come specificato in precedenza, abbiamo scelto di optare per una soluzione che ci consentisse di salvare le immagini caricate all’interno del *web server* e il *path* delle stesse nel database.

PHP mette a disposizione il vettore `$_FILES[]` il quale contiene tutta una serie di informazioni sul file caricato dall’utente e tra queste vi è la locazione temporanea del file (***tmp_location***). Il gioco consiste nell’ottenere la locazione temporanea e quella di destinazione (ad esempio in una cartella “*immagini*”) in cui sarà memorizzato il nostro file e passare questi due parametri alla funzione PHP `move_uploaded_file()` come mostrato nel seguente *snippet* di codice:

```

1. $fileTmpName = $_FILES['file']['tmp_name'];
2. // ...
3. $fileDest = "uploads/$fn";
4. move_uploaded_file($fileTmpName, $fileDest);

```

La variabile `$fn` è una variabile che, attraverso la funzione `uniqid()` e l’estensione del file genera un nome univoco per ogni singolo file caricato:

```

1. $fn = uniqid('', true).".$extension;

```

NB. Specifico che l’unico modo per caricare file in un form HTML è attraverso l’aggiunta dell’attributo *enctype* impostato come *multipart/form-data*.

Gestione delle estensioni dei file caricati

La nostra piattaforma mette a disposizione degli utenti la possibilità di caricare file immagine di qualunque tipo, siano essi disegni o fotografie, siano essi in formato *.jpg* o in formato *.png* insomma: va bene tutto purché si parli di immagini.

È chiaro dunque che un ulteriore controllo andrebbe fatto sull’estensione dei file caricati dall’utente.

Nel seguente *snippet* di codice definiamo un vettore `$allowed_extensions` con le estensioni immagine consentite e, successivamente, verifichiamo che l’estensione del file caricata dall’utente corrisponda a una delle estensioni definite nel vettore (in caso contrario, all’utente verrà mostrato un messaggio d’errore, per maggiori informazioni si rimanda al paragrafo “Gestione degli errori sui vincoli d’integrità”).

```

1. $allowed_extensions = array("jpg", "jpeg", "png");
2. if(in_array($extension, $allowed_extensions) && !$fileErr) {
3.     // ...
4. } else header("Location: carica_opera.php?err=true");

```

Le pagine utente

Ogni volta che un'opera viene caricata l'*index.php* della piattaforma web si aggiorna con le nuove immagini, con le informazioni di licenza e con lo *username* dell'autore dell'opera:



Cliccando sullo username l'utente viene indirizzato a una pagina *utente.php* e si definisce il parametro GET *username* grazie al quale è possibile interrogare il database per ottenere tutte le opere pubblicate dall'utente interessato.

Il link al quale si clicca è il seguente ed è evidenziata la definizione del parametro GET *username*:

```

1. <p class="card-text">Pubblicato da
2.     <a href="utente.php?username=".$row['Username']."'>@".$row['Username']."'</a>
3. </p>

```

Non resta che interrogare il database nella pagina *utente.php* sulla base del valore contenuto nella variabile *\$username* definita come mostrato nella figura sottostante:

```

1. if(isset($_GET['username'])) {
2.     $username = mysqli_real_escape_string($conn, $_GET['username']);
3.     $query = "SELECT o.Filepath, o.Titolo, o.Licenza, u.Username
4.              FROM opera AS o INNER JOIN utenti AS u ON o.Autore=u.ID
5.              WHERE u.Username='$username'";
6.     // ...

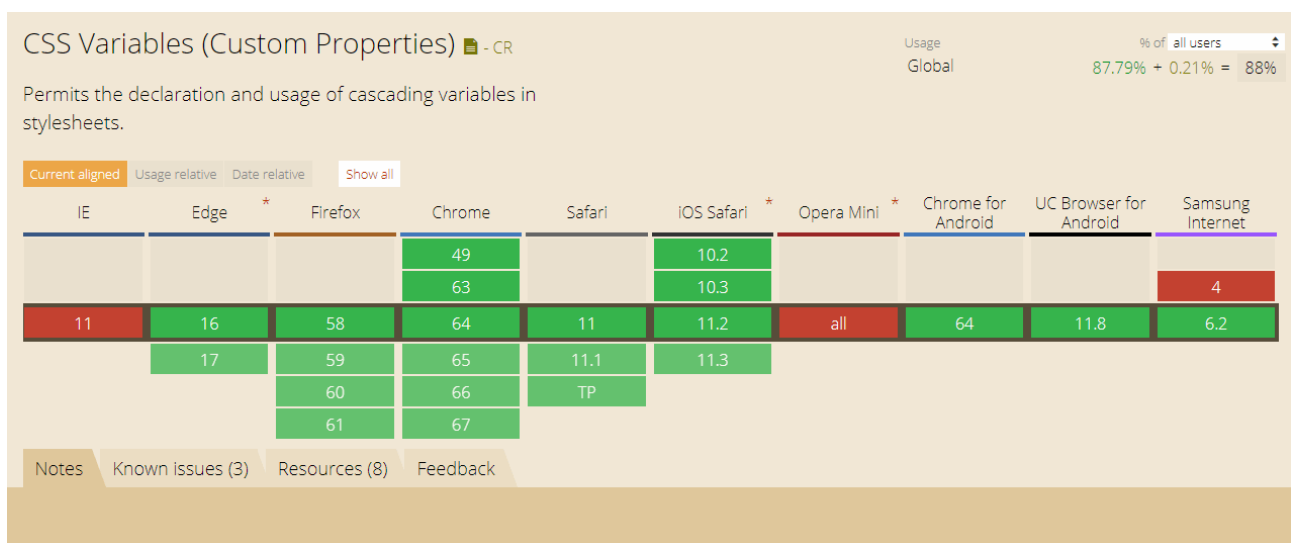
```

I fogli di stile (CSS)

In questo capitolo, come fatto per il precedente, mi concentrerò sugli aspetti più interessanti questa volta in merito alle scelte di stile e accessibilità della piattaforma web.

Le custom properties di CSS3

CSS3 mette a disposizione la possibilità di definire delle *custom properties* ovvero delle vere e proprie variabili. Nella nostra piattaforma la scelta dei colori della *navbar*, del *footer* oltre ad altri parametri come la “rotondezza” del bordo nelle immagini sono impostati per mezzo di questa funzionalità. Il vantaggio è chiaro, meno codice duplicato (e in CSS la duplicazione del codice è piuttosto comune) secondo il principio *Don't Repeat Yourself (DRY)* e miglioramento della manutenibilità del codice stesso.



Supporto delle variabili CSS

Vediamo le *custom properties* definite nel file *style.css*:

```

1. :root {
2.   --primary-color: #533B4D;
3.   --btn-primary: #C1ABA6;
4.   --btn-primary-hover: #775f59;
5.   --btn-secondary: #C1ABA6;
6.   --btn-secondary-hover: #775f59;
7.   --border-radius-size: 10px;
8. }
```

Responsive web design

Nel 2018, per la prima volta, i dati che arrivavano da dispositivi mobili superarono quelli arrivati da PC e laptop. È un dato storico che ci obbliga a considerare quella larga porzione di utenza che visiterà la nostra piattaforma con il proprio smartphone o tablet.

Di fronte a un dato del genere un web designer ha fondamentalmente due opzioni:

1. Il sito deve potersi adattare a qualunque risoluzione schermo (almeno a quelle più comuni), questa caratteristica viene definita *responsive web design*.
2. Si sviluppano più siti sfruttando i sottodomini a seconda della risoluzione schermo con la quale si sta visitando la piattaforma web (è il caso, ad esempio, di *facebook.com* e della sua versione “mobile” *m.facebook.com*).

Chiaramente, la nostra piattaforma è una piccola startup che sfrutta un servizio di web hosting (vedi “Le piattaforme di *web hosting*”) per cui ho deciso di optare per la prima soluzione.

Il linguaggio CSS mette a disposizione uno strumento potentissimo per adattare il layout delle nostre pagine web a seconda della risoluzione dello schermo: le **mediaqueries**.

Tuttavia sviluppare l’intero design da zero e definire le *mediaqueries* per ogni risoluzione avrebbe portato via troppo tempo ed è qui che entra in gioco **Bootstrap 4**, un framework CSS e Javascript (ormai già alla sua quinta versione) con una serie di classi pre-definite grazie al quale è possibile sviluppare siti interamente *responsive*.

Nello sviluppo della piattaforma ho adottato entrambe le soluzioni, cercando di utilizzare Bootstrap 4 per definire il layout degli elementi della pagina web e scrivendo alcune *mediaqueries* laddove ho ritenuto necessario agire “manualmente”. Nel prossimo capitolo vediamo la definizione delle suddette *mediaqueries* e le motivazioni che mi hanno portato ad implementarle.

Mediaqueries

Le *mediaqueries* definite nel file *style.css* sono volte a centrare ed inserire in due colonne i contenuti del *footer* e a perfezionare i pulsanti di navigazione nella *navbar*. Notare come sono state definite due *mediqueries* per le risoluzioni **992px** e **768px**, tali risoluzioni non sono casuali ma sono definite da *Bootstrap 4* com’è possibile visualizzare al seguente link:

<https://getbootstrap.com/docs/4.1/layout/overview/#responsive-breakpoints>

Di seguito l’implementazione delle *mediaqueries* nel file *style.css*:

```

1. @media (max-width: 992px) {
2.   .navbar-toggler { border: none; }
3.   .navbar-toggler:focus { outline: none !important; }
4.   .fa-bars { color: var(--btn-primary); }
5.   footer .footer-col-lf, footer .footer-col-rt { text-align: center; margin: 10px 0; }
6.   footer .footer-col-rt { justify-content: center; }
7. }
8.
9. @media (max-width: 768px) {
10.  footer { bottom: -100px; }
11. }
```

Block Element Modifier (BEM)

Dando un'occhiata al codice HTML dei diversi file PHP si può notare come la gerarchia degli elementi del linguaggio è identificata, oltre che dalle indentazioni nell'editor di testo, da una particolare scelta di nomenclatura degli elementi stessi.

Ad esempio, nel file *index.php*, è possibile notare come l'elemento *.sya-cards-card-wrapper* si trovi all'interno dell'elemento padre *.sya-cards-card* che a sua volta si trova all'interno dell'elemento *.sya-cards*. Tale approccio prende il nome di *Block Element Modifier* (o, semplicemente, approccio **BEM**) e consente con estrema facilità di stabilire una gerarchia di classi volte a migliorare la manutenibilità e la flessibilità del codice.

```
1. <div class="sya-cards">
2.     <div class="sya-cards-card">
3.         <!-- ... -->
4.     </div>
5. </div>
```

NB. C'è da specificare che l'approccio **BEM** prevede molte più regole di quelle descritte in questo paragrafo ed invito il letto, la mia è una rudimentale rivisitazione dell'approccio per una piattaforma web relativamente semplice dal punto di vista implementativo. Invito il lettore ad approfondire, qualora interessato, sul sito ufficiale: <http://getbem.com>

Studio di fattibilità

SYA – Share Your Art nasce come startup il cui modello di guadagno è basato su abbonamenti mensili dell'utenza i quali verranno implementati al crescere della stessa. Parte del compenso ottenuto dai piani mensili sarà condiviso con un artista che l'utente ritiene particolarmente meritevole. L'infografica seguente mostra sono elencati i diversi piani previsti con i relativi vantaggi:



Scelta del servizio di web hosting

Sono a carico dello sviluppatore, ovvero del sottoscritto, tutte le spese relative al caricamento e al mantenimento della piattaforma web attraverso il servizio di *web hosting* scelto.

I requisiti necessari che mi hanno consentito di scegliere il servizio di *web hosting* sono:

- **Spazio su disco illimitato:** È necessario, per questioni di scalabilità e per essere in linea con i piani previsti dal modello di guadagno, che lo spazio offerto dal servizio di *web hosting* sia illimitato in modo da poter consentire all'utente di poter caricare le sue opere senza vincoli di "spazio".
- **Inclusione del dominio di primo livello:** Per ammortizzare i costi sarebbe preferibile che il dominio di primo livello (.it) venga fornito con il piano previsto dal servizio di *web hosting*. *NB.* Dal momento in cui desideriamo che il servizio possa raggiungere il maggior numero di persona possibile è assolutamente necessario che il servizio offra anche la possibilità di avere un dominio internazionale (.com) di primo livello.
- **Servizio di backup e anti-malware:** Per garantire l'integrità dei dati è necessario che vi sia ridondanza attraverso servizi di backup. È preferibile che nel piano sia inoltre previsto un anti-malware.
- **Assistenza continua 24H:** Dal momento in cui si tratta di una *startup* prevedo che vi saranno non pochi problemi tecnici relativi al mantenimento della piattaforma; ho bisogno

di poter contare su un servizio di assistenza stabile, veloce e disponibile in qualsiasi momento.

- **Certificato SSL:** La piattaforma deve disporre di certificato SSL e il dominio dev'essere protetto dal protocollo HTTPS per prevenire attacchi di tipo MITM (vedi "Il protocollo HTTPS").

Fatte queste considerazioni, si è scelto di utilizzare la piattaforma di *web hosting* offerta da [aruba.it](https://www.aruba.it) attraverso il servizio *Hosting Easy* in quanto risponde in maniera soddisfacente ai suddetti requisiti e non richiede enormi investimenti iniziali.

La tabella seguente mostra i costi relativi al caricamento e al mantenimento della piattaforma web attraverso il servizio di *web hosting* scelto:

COSTO MENSILE	TOTALE
€9,99 + IVA 22%	€12.19

Al totale va poi sommato, annualmente, il prezzo di rinnovo che ammonta a €50.

Privacy Policy

Dal momento in cui la piattaforma è priva di pubblicità e non condivide i dati dei suoi utenti con terzi per fini pubblicitari (e non) verrà implementata una politica di Privacy Policy relativamente semplice ma è necessario che il servizio a cui facciamo affidamento sia professionale e, soprattutto, aggiorni in automatico la Policy in base alle nuove leggi in materia di protezione dei dati personali.

La piattaforma a cui facciamo riferimento è iubenda.com che prevede un piano mensile di €9 con il quale è possibile generare fino a 5 licenze di Privacy Policy il che è ottimo in quanto si è detto di volere due domini di primo livello, uno nazionale (.it) ed uno internazionale (.com).



OLTRE 1500 CLAUSOLE AGGIORNATE AUTOMATICAMENTE

Al centro della nostra soluzione ci sono centinaia di clausole redatte in dettaglio e aggiornate automaticamente dal nostro team legale internazionale. Con la nostra soluzione avanzata puoi generare in pochi minuti una privacy policy accurata e completa. Non è richiesta alcuna competenza legale.



RISPETTA LE PRINCIPALI NORMATIVE INTERNAZIONALI

Le nostre soluzioni rispettano i requisiti più stringenti al mondo come il GDPR e la Direttiva ePrivacy, dandoti comunque la possibilità di personalizzare secondo necessità.



PIÙ OPZIONI DI INTEGRAZIONE

Hai bisogno di un link per gli app store? Di un pulsante per la tua newsletter? Preferisci includere la tua privacy policy direttamente sul tuo sito o app? Nessun problema: puoi integrare la tua policy tramite codice di inserimento, link diretto o inclusione diretta del testo.

Aggiorniamo dunque la tabella dei costi con le suddette informazioni:

	COSTO ANNUALE	TOTALE
WEB HOSTING	€9,99 + IVA 22% al mese + €50 al rinnovo	€62.19
PRIVACY POLICY	€9 al mese (IVA compresa)	€9

Individuazione delle risorse umane

Nella fase iniziale del progetto le tre figure necessarie ad ogni attività imprenditoriale (direttore generale, responsabile delle finanze, responsabile tecnico) sono ricoperte dal sottoscritto. Al crescere dell'utenza e con l'introduzione dei piani di assumere del personale specifico al fine di garantire un servizio di qualità ed espandere il bacino d'utenza.

Il lavoro, almeno in un primo momento, sarà svolto interamente in modalità *smart working* al fine di ammortizzare i costi relativi all'acquisto e al mantenimento dell'infrastruttura aziendale.

Le tre figure professionali che sarà necessario individuare una volta raggiunto un bacino d'utenza notevole sono riassunte nella seguente tabella in cui viene anche riportato lo stipendio medio lordo annuale per ciascuna risorsa:

PROGRAMMATORE WEB JUNIOR	€31.000
SPECIALISTA SEO	€33.000
ECONOMISTA AZIENDALE	€35.000

Di conseguenza aggiorniamo la tabella dei costi annuali con le suddette informazioni:

	COSTO ANNUALE	TOTALE
WEB HOSTING	€9,99 + IVA 22% al mese + €50 al rinnovo	€62.19
PRIVACY POLICY	€9 al mese (IVA compresa)	€108
RISORSE UMANE	€99.000	€99.000

Profilo economico di sintesi

In definitiva l'investimento iniziale può essere sostenuto anche dal sottoscritto in quanto comprende le spese relative al caricamento e al mantenimento della piattaforma nel servizio di *web hosting* previsto, quello relativo al servizio di *privacy policy* e le irrisorie spese di piccole campagne pubblicitarie su *social networks* come Instagram (generalmente intorno a €1/€2) per iniziare a far conoscere la piattaforma all'utenza finale.

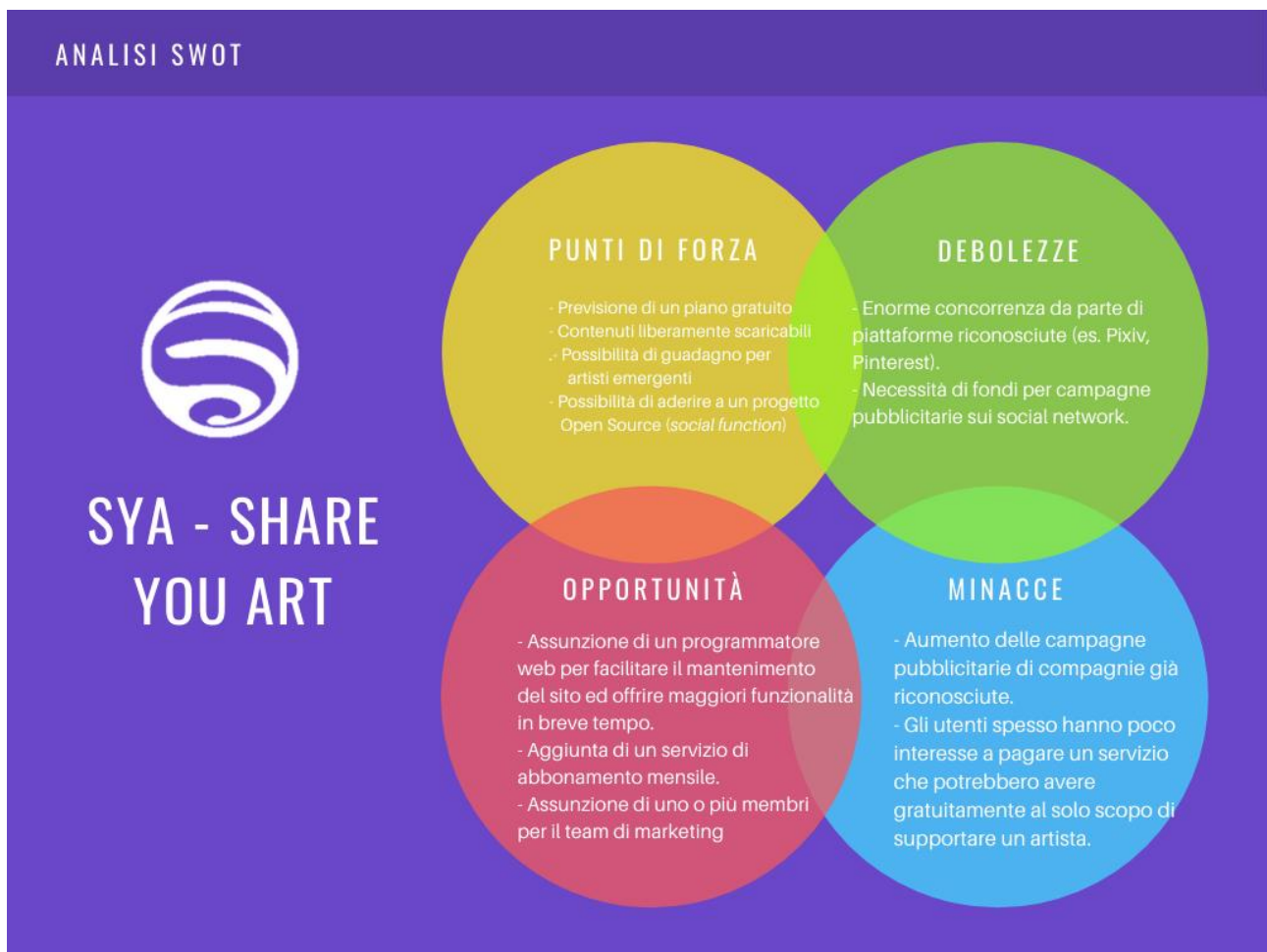
Al crescere dell'utenza e con l'introduzione dei piani d'abbonamento (a livello tecnico realizzabili sempre da una sola persona) si prevede di assumere le figure evidenziate nel paragrafo precedente.

È importante specificare che solo al crescere dell'utenza sarà necessario introdurre nuove figure professionali e che queste non devono essere necessariamente inserite tutte allo stesso momento nel team. In ordine di priorità l'impresa avrà bisogno di almeno un altro *programmatore web junior* come risorsa fondamentale al mantenimento del sito e all'implementazione di nuove *feature* e di un economista aziendale (un commercialista o un ragioniere) che segua l'impresa aiutandola per quanto concerne gli aspetti inerenti alla contabilità aziendale nel momento in cui si comincia a *fatturare*. Infine si prevede l'introduzione di uno specialista SEO per il *piazzamento online* sui motori di ricerca più comuni.

In definitiva, essendo un progetto relativamente a "basso rischio" in quanto prevede investimenti iniziali minimi e investimenti importanti solo al crescere dell'utenza e quindi già basati su risorse contabili interne provenienti dal modello di guadagno (vedi "Studio di fattibilità"), varrebbe la pena

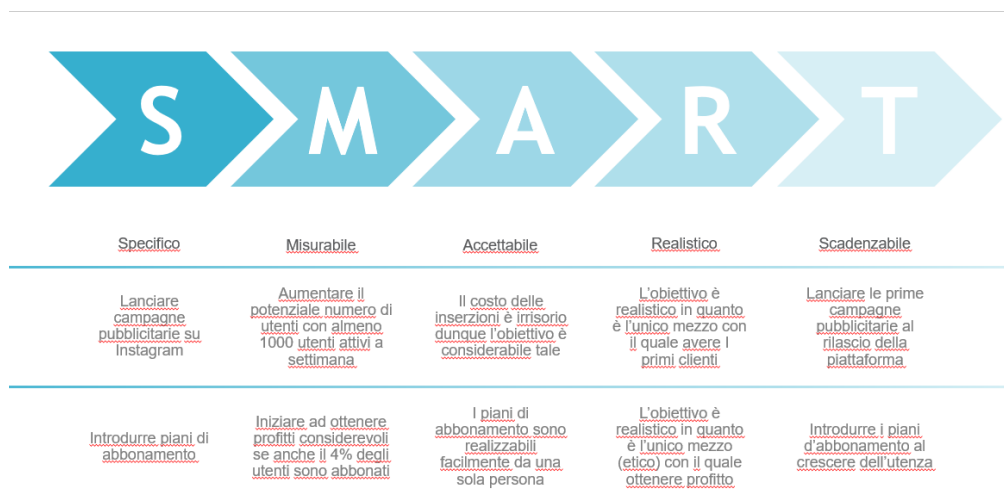
partire con l'idea imprenditoriale una volta individuati i punti di forza e di debolezza del servizio che stiamo offrendo attraverso un'opportuna analisi SWOT (prossimo paragrafo).

L'analisi SWOT



Gli obiettivi SMART

Vediamo in questo diagramma gli obiettivi SMART (*Specific, Measurable, Accepted, Realistic, Timely*) che mi sono posto in merito al progetto imprenditoriale sviluppato:



Bibliografia

- <https://templates.office.com/en-us/s-m-a-r-t-goals-tm16401938>
- <https://slidesgo.com/theme/pricing-table-infographics>
- <https://www.nodopiano.it/utenti-navigano-piu-da-mobile/>
- <https://dash.e.jimdo.com/websites/logos?grw414ToDashboard>
- https://owasp.org/www-pdf-archive/Parata_SMAU06.pdf
- <https://www.youtube.com/watch?v=TAK-sgF2gAs>
- <https://www.pinterest.it/pin/52002570675282958/>
- <https://www.ovh.com/blog/web-hosting-how-to-host-3-million-websites/>
- <https://swhco.files.wordpress.com/2011/03/diag11.jpg>
- <https://www.educative.io/edpresso/web-server-vs-application-server>
- <https://www.infoworld.com/article/2077354/app-server-web-server-what-s-the-difference.html>
- <https://www.uml-diagrams.org/examples/web-application-network-diagram-example.html>
- <https://www.educative.io/api/edpresso/shot/5043667712606208/image/5659644338896896>
- http://www.maurodeberardis.it/index.php?option=com_jdownloads&Itemid=338&view=finish&cid=560&catid=18