

Istituto d'Istruzione Superiore "L.da Vinci"



Alunno: **Minaudo Marco**

Classe: **5C**

Anno scolastico: **2020/2021**

Indice

1 Introduzione

2 Informatica

2.1	Progettazione e realizzazione sito.	3
2.1.1	Idea.	3
2.1.2	Struttura.	3
2.1.3	Funzionalità nascoste	8
2.1.4	Zoom sul codice.	9
2.1.5	Diritti del consumatore.	15
2.2	Progettazione e realizzazione Database.	17
2.2.1	Database.	17
2.2.2	Modello concettuale	17
2.2.3	Modello logico.	19
2.2.4	Creazione database.	21
2.3	Le interrogazioni annidate.	23
2.3.1	Riassunto generale del linguaggio SQL.	23
2.3.2	Utilizzo e utilità delle interrogazioni annidate.	23
2.3.3	Uso delle clausole exist, all, any.	23
2.3.4	Utilizzo delle interrogazioni annidate all'interno del sito	24
2.3.5	Alternative alle interrogazioni utilizzate nel sito.	26

3 Sistemi

3.1	Tipologia hosting e descrizione rete del sito.	28
3.2	Le reti wireless	31
3.3	La trasmissione nelle reti wireless.	31

Introduzione

La consegna data per il mio elaborato è la seguente:

“Realizza un **sito web** a piacere con collegamento a un database Mysql tramite linguaggio PHP e descrivi il lavoro svolto soffermandoti sul linguaggio SQL e **le interrogazioni annidate**. Descrivi inoltre lo schema generale di una architettura Client-Server con l'utilizzo di un firewall, scegli un piano di indirizzamento per la rete Intranet e motiva la scelta di un piano Hosting o Housing per la pubblicazione del sito web. Infine soffermati a descrivere **la trasmissione nelle reti wireless**”

Per la creazione del sito ho dato sfogo alla mia creatività, ho inserito tutta la conoscenza che ho acquisito negli anni.

Informatica

Progettazione sito

Idea

Il sito si basa sull'idea di creare un **e-commerce** che vende principalmente scarpe, il sito è stato ideato con un'interfaccia semplice e leggera e permette all'utente di acquistare dei prodotti con pochi e semplici click. L'utente deve necessariamente registrarsi o autenticarsi per poter acquistare, questo avviene tramite un'apposita pagina contenente i relativi form.

Per imprimere agli utenti il colore del brand, ho utilizzato lo stesso colore sul logo e sulle varie pagine del sito.

Struttura

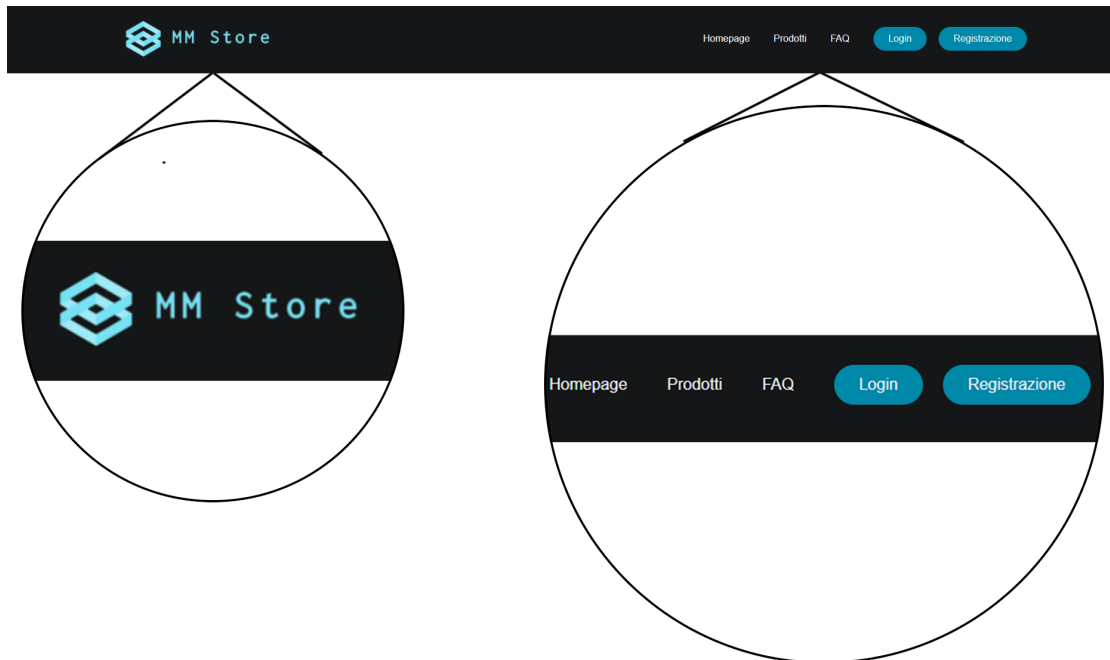
L'intero sito è stato creato con diversi linguaggi:

- **Html** (HyperText Markup Language): è un linguaggio di markup
- **PHP**: è un linguaggio di programmazione lato server.
- **Javascript**: è un linguaggio di programmazione lato client.
- **Css**: è un linguaggio usato per definire la formattazione di documenti HTML

MENU

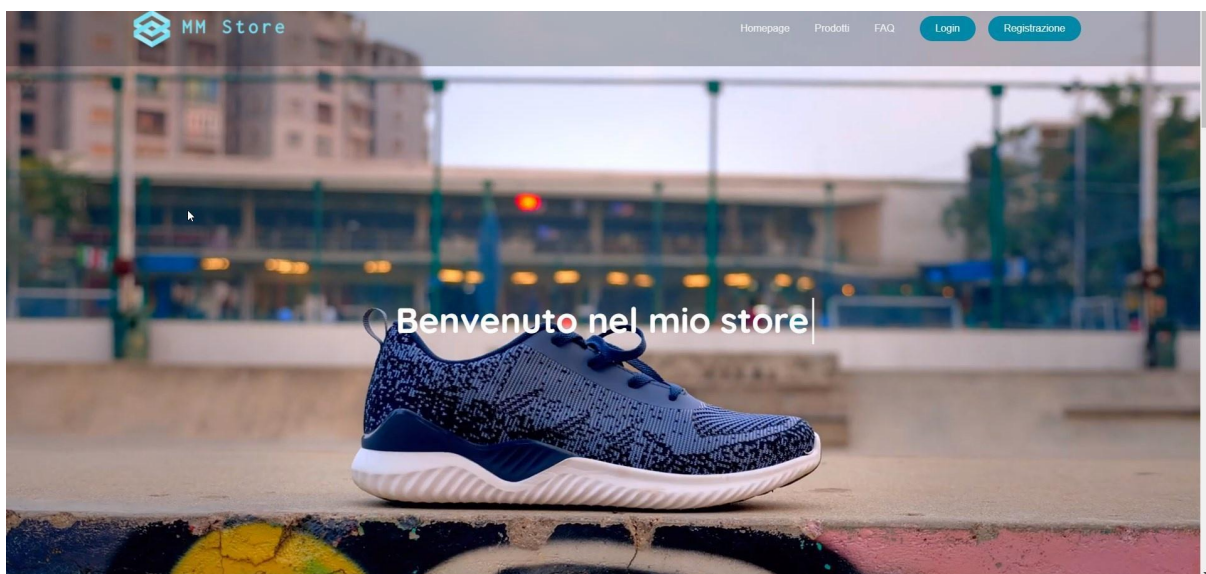
Il menu ha un design semplice, ha un colore di background tendente al nero, a sinistra è presente il logo con il nome dello store mentre a destra sono presenti tutti

link delle varie pagine, i link per il login e la registrazione hanno un design diverso per distinguerli dagli altri link.



HOME

La pagina **home** è stata ideata per attirare i nuovi utenti, infatti la pagina si apre con un video in background e una scritta a comparsa che dà il benvenuto all'utente. Inizialmente la barra del menu è trasparente per rendere più visibile il video, ma quando si scorre verso il basso essa si scurisce, in questo modo si crea un contrasto tra il bianco del background e il colore scuro del menu.











Scorrendo nella home è possibile vedere un **countdown** che indica per quanto tempo le offerte saranno valide e i prodotti più venduti con lo sconto più alto. Ogni scheda contiene la recensione del prodotto rappresentata da stelle (scala che va da 1 a 5 stelle), l'immagine delle scarpe da acquistare, il prezzo scontato con lo sconto evidenziato in alto a sinistra delle schede

Le offerte termineranno tra:

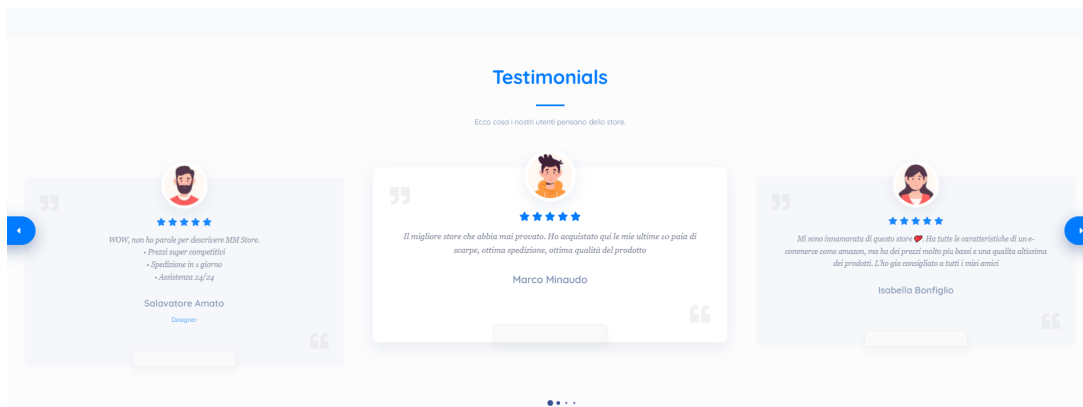
82d 0h 39m 48s

I Più Venduti

Benvenuto nuovo cliente. Ecco i nostri nuovi prodotti in offerta, approfittane subito!!

<p>SCONTO 50%</p>  <p>★★★★☆ NIKE AIR MAX PLUS 3 €179 €89.5</p>	<p>SCONTO 50%</p>  <p>★★★★☆ NIKE AIR PRESTO €119 €59.5</p>	<p>SCONTO 50%</p>  <p>★★★★☆ NIKE AIR MAX PLUS III €179 €89.5</p>	<p>SCONTO 50%</p>  <p>★★★★★ NIKE AIR MAX PLUS €169 €84.5</p>
<p>SCONTO 50%</p>  <p>★★★★☆ NIKE AIR MAX PLUS EOI €179 €89.5</p>	<p>SCONTO 50%</p>  <p>★★★★☆ NIKE AIR MAX 97 €179 €89.5</p>	<p>SCONTO 50%</p>  <p>★★★★☆ NIKE WEARALLDAY €64 €32</p>	<p>SCONTO 50%</p>  <p>★★★★★ NIKE GO FLYEASE €119 €59.5</p>

Continuando sempre nella home abbiamo i **testimonials**, dove utenti che hanno già provato ad acquistare dei prodotti esprimono la loro opinione sul prodotto in sé e sul servizio offerto dal sito.

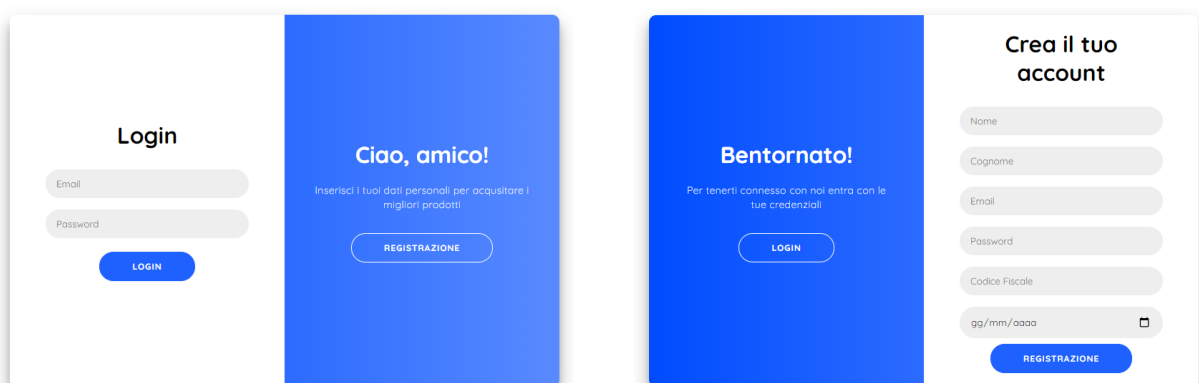


Verso la fine della pagina è presente il **footer** dove sono presenti i miei link agli account github, instagram e facebook e sono anche presenti informazioni riguardo l'e-commerce come la storia o la privacy. Il footer ha un colore più scuro per evidenziare il contrasto tra la fine della pagina e ciò che è presente sopra.



LOGIN E REGISTRAZIONE

La pagina di login e registrazione sono state unite in un'unica pagina, infatti l'utente può switchare da una modalità all'altra mediante un pulsante presente nella schermata. In caso di problemi con la registrazione o con il login l'errore sarà stampato all'interno del riquadro di login con un colore rosso sgargiante, per far comprendere all'utente che qualcosa nel processo di login o di registrazione è andato male.



CHECKOUT

La pagina di **checkout**, accessibile solamente dopo aver effettuato il login, permette di acquistare i prodotti inseriti nel carrello, il pagamento avviene con la carta di credito dell'utente, il tutto viene gestito con la massima sicurezza, infatti il numero della carta di credito viene criptato e salvato nel database

The screenshot shows a checkout form titled "Acquisto". At the top, a progress bar has four steps: "Carrello", "Prodotti", "Pagamento", and "Fine". The "Pagamento" step is currently active. Below the progress bar, there are input fields for "Nome", "Numero carta", "Data", and "CVV / CVC *". A small note states: "*Il CVV è il codice di sicurezza posto dietro la carta". There is also a field for "Indirizzo di spedizione". At the bottom, there are two buttons: "Indietro" (Back) and "Avanti" (Next).

The screenshot shows a completion page titled "Acquisto completato". It features the same four-step progress bar as the previous page, but now all steps ("Carrello", "Prodotti", "Pagamento", "Fine") are completed. The text "Acquisto effettuato" (Purchase completed) is displayed in the center. At the bottom right, there is a button labeled "Torna allo store" (Return to store).

FAQ

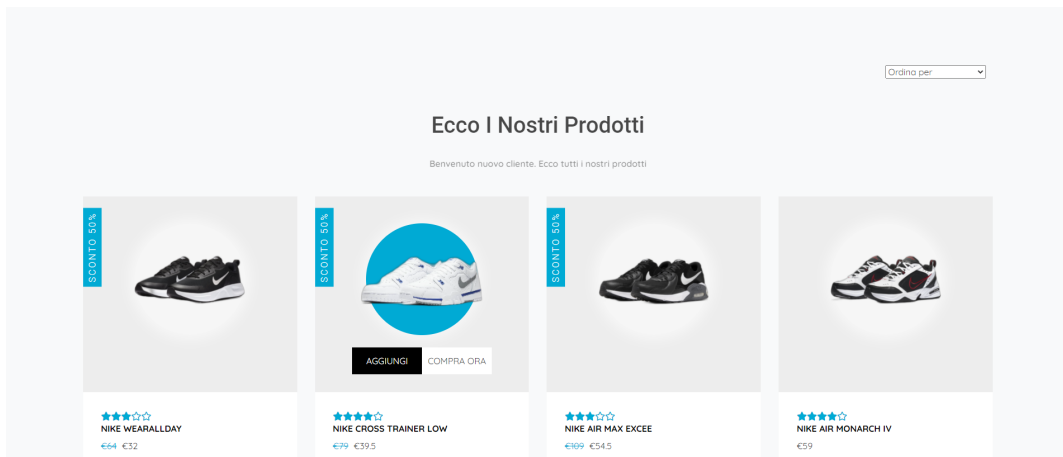
La pagina **FAQ** è stata ideata per aiutare e chiarire i vari dubbi che gli utenti possono avere. Questa pagina serve principalmente a rassicurare il cliente, infatti le domande presenti sono quelle che un utente potrebbe pensare prima di acquistare un prodotto.

The FAQ page features a blue background. On the left, there is an illustration of a person standing next to a large computer monitor displaying a chat interface. To the right of the illustration, the heading "FAQ" is displayed in bold. Below the heading, there is a list of five questions, each followed by a downward-pointing arrow indicating a dropdown menu:

- Quali metodi di pagamento sono accettati?
- Quali sono i tempi di spedizione?
- Quanto dura la garanzia dei prodotti?
- Quanto costa la spedizione dei prodotti che ho acquistato?
- Ho inserito i miei dati per acquistare un prodotto. Chi ne avrà accesso?

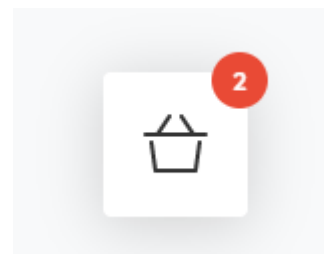
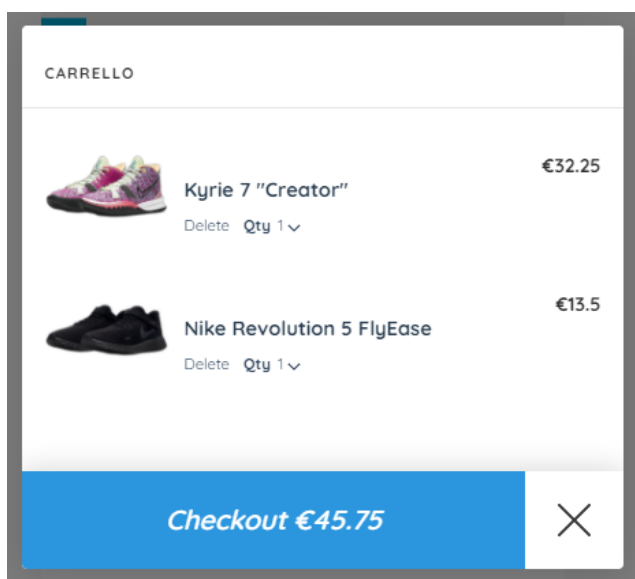
Prodotti

Nella pagina prodotti, abbiamo tutte le scarpe che l'e-commerce vende, i risultati possono essere filtrati per prezzo, categoria, sconto e per i più acquistati dai giovani.



Funzionalità nascoste

Il sito utilizza un carrello pop-up che compare appena l'utente aggiunge un prodotto al carrello, se l'utente decide di non voler più un prodotto già inserito nel carrello, c'è un pulsante dedicato per eliminarlo. Un'altra funzione, pensata soprattutto per gli utenti più distratti, è il *torna indietro*, cioè se l'utente per sbaglio elimina un articolo dal carrello, per 8 secondi compare un piccolo pulsante che permette di re-inserire il prodotto. Il carrello rimane salvato anche spostandosi da una pagina all'altra o chiudendo e riaprendo il browser visto che il tutto rimane salvato nei cookie. Per ogni singolo prodotto possiamo specificare la quantità desiderata.



Zoom sul codice

In questa sezione sarà analizzato una parte del codice dell'e-commerce:

Questo è il codice della **barra di navigazione**, è presente sia del codice html che php, se l'utente non è loggato mostra i pulsanti di login e registrazione, mentre se l'utente è già loggato mostra un pulsante di logout. Per verificare se l'utente è loggato viene controllata se la variabile 'start_time' della sessione è settata.

```
<!-- Menu -->
<header id = "navbar">
    <a class="logo" href="index.php"></a>
    <div>
        <div class="nav__links">
            <li><a href="Index.php">Homepage</a></li>
            <li><a href="Products.php">Prodotti</a></li>
            <li><a href="faq.php">FAQ</a></li>
        </div>
    </div>
<?php
    session_start();
    if (!isset($_SESSION['start_time'])){
        ?>
        <a class="cta" href="LoginRegister.php?x=1">Login</a>
        <a class="cta" href="LoginRegister.php?x=2">Registrazione</a>
        <?php
        }
        else{
            ?>
            <a class="cta" href="Logout.php">Logout</a>
            <?php } ?>
    </header>
```

Questo codice è presente sia nell'**index** che nella pagina **prodotti** e permette di creare tutte le schede dei prodotti da acquistare. Nel codice php c'è un ciclo while che viene ripetuto per ogni prodotto che verrà visualizzato e in base al risultato dato dalla query possiamo visualizzare diversi prodotti.

Ogni prodotto contiene delle informazioni, come immagini, prezzo o nome, che saranno salvate nei cookie dopo l'aggiunta al carrello.

```
<?php
// Connessione al DBMS
$conection = new mysqli("localhost", "root", "", "elaborato");
```

```

$query = "";
$resultato = $connection->query($query);
if($resultato->num_rows == 0)
    echo "Nessun prodotto in sconto";
else{
    $n = 0;
    while (($dataDBMS = $resultato->fetch_array()) && $n < 12) {
        echo "<div class = \"product\">";
        echo "<div class = \"product-content\">";
        echo "<div class = \"product-img\">";
        echo "<img src = ".$dataDBMS['PercorsoImmagine']."' alt = 
\"product image\">";
        echo "</div>";
        echo "<div class = \"product-btns\">";
        echo "<button type = \"button\" class = \"btn-cart 
js-cd-add-to-cart\" data-price=\"".$dataDBMS['Prezzo']."' 
price-off=\".($dataDBMS['Prezzo']*(100-$dataDBMS['Sconto'])/100).\" href=\"#0\" 
img=\"".$dataDBMS['PercorsoImmagine']."' name=\"\"".$dataDBMS['Nome']."' idProduct 
=\"".$dataDBMS['Id']."'> Aggiungi";
        echo "</button>";
        echo "<form action=\"Checkout.php\" method= \"POST\">";
        echo "<button type = \"submit\" class = \"btn-buy\" name=\"IdP\" 
value=\"".$dataDBMS['Id']."'> Compra ora";
        echo "</button>";
        echo "</form>";
        echo "</div>";
        echo "</div>";

        echo "<div class = \"product-info\">";
        echo "<div class = \"product-info-top\">";
        echo "<div class = \"rating\">";
        for ($i=0; $i < $dataDBMS['Recensione']; $i++)
            echo "<span><i class = \"fas fa-star\"></i></span>";
        for ($i=0; $i < 5 - $dataDBMS['Recensione']; $i++)
            echo "<span><i class = \"far fa-star\"></i></span>";
        echo "</div>";
        echo "</div>";
        echo "<a href = \"#\" class = 
\"product-name\">".$dataDBMS['Nome']."'</a>";
        echo "<p class = \"product-price\">€".$dataDBMS['Prezzo']."'</p>";
        echo "<p class = 
\"product-price\">€.($dataDBMS['Prezzo']*(100-$dataDBMS['Sconto'])/100).\"</p>";
        echo "</div>";

        echo "<div class = \"off-info\">";
        echo "<h2 class = \"sm-title\">Sconto 
".$dataDBMS['Sconto']."'<%</h2>";
        echo "</div>";
        echo "</div>";
        $n = $n + 1;
    }
}
?>

```

Questo è una parte del codice per il funzionamento del carrello, qui possiamo vedere la funzione `setCount()` che setta il prezzo e il counter dei prodotti inseriti nel carrello, mentre la funzione `setCart()` prendi tutti i prodotti salvati nei cookie e li inserisce nel carrello

```
<!-- Carrello -->
<div class="cd-cart cd-cart--empty js-cd-cart">
  <a href="#0" class="cd-cart__trigger text-replace">
    <ul class="cd-cart__count"> <!--Contatore carrello -->
      <li><script type="text/javascript">
        setCount();
      </script></li>
      <li>0</li>
    </ul>
  </a>

  <div class="cd-cart__content">
    <div class="cd-cart__layout">
      <header class="cd-cart__header">
        <h2>Carrello</h2>
        <span class="cd-cart__undo">Articolo rimosso. <a
href="#0">Torna indietro</a></span>
      </header>

      <div class="cd-cart__body">
        <ul id="CartAll">
          <!-- I prodotti saranno aggiunti qui -->
          <script type="text/javascript">setCart();</script>
        </ul>
      </div>

      <div class="cd-cart__footer">
        <a href="Checkout.php" class="cd-cart__checkout">
          <em>Checkout €<span><script
type="text/javascript">document.write(price)</script></span>
          <svg class="icon icon--sm" viewBox="0 0 24 24"><g
fill="none" stroke="currentColor"><line stroke-width="2" stroke-linecap="round"
stroke-linejoin="round" x1="3" y1="12" x2="21" y2="12"/><polyline
stroke-width="2" stroke-linecap="round" stroke-linejoin="round" points="15,6
21,12 15,18 "/></g>
          </svg>
        </em>
      </a>
    </div>
  </div>
</div>
<script type="text/javascript">
  function setCart() {
    price = 0;
```

```

        for (var i = document.cookie.split(';').length - 1; i >= 0 &&
document.cookie; i--) {
            //Prendo il cookie fra tanti    Divido il singolo cookie e prendo
il nome

            a = document.cookie.split(';')[i].split('=')[0];
            a = a.replace(/ /g, '');
            if (a == "PHPSESSID")
                continue;
            data = JSON.parse(getCookie(a));
            var productAdded = '<li idProduct = ' + a + '
class="cd-cart__product"><div class="cd-cart__image"><a href="#0"></a></div><div class="cd-cart__details"><h3
class="truncate"><a href="#0">' + data[1] + '</a></h3><span
class="cd-cart__price">€' + data[2] + '</span><div class="cd-cart__actions"><a
href="#0" class="cd-cart__delete-item">Delete</a><div
class="cd-cart__quantity"><label for="cd-product-' + data[3] + '">Qty</label><span
class="cd-cart__select"><select class="reset" id="cd-product-' + data[4] + '"
name="quantity"><option value="1">1</option><option value="2">2</option><option
value="3">3</option><option value="4">4</option><option
value="5">5</option><option value="6">6</option><option
value="7">7</option><option value="8">8</option><option
value="9">9</option></select><svg class="icon" viewBox="0 0 12 12"><polyline
fill="none" stroke="currentColor" points="2,4 6,8 10,4
"/></svg></span></div></div></div></li>';
            document.write(productAdded);
            price = Number(price) + (Number(data[2]) * data[5]);

            // Select quantity
            document.getElementById('cd-product-' + data[4]).value = data[5];
        }
    }

    function setCount(){
        count = 0;

        for (var i = document.cookie.split(';').length - 1; i >= 0 &&
document.cookie; i--) {
            //Prendo il cookie fra tanti    Divido il singolo cookie e prendo
il nome

            a = document.cookie.split(';')[i].split('=')[0];
            a = a.replace(/ /g, '');
            if (a == "PHPSESSID")
                continue;
            data = JSON.parse(getCookie(a));
            count = count + data[5];
        }
        document.write(count);
    }
</script>

```

Questo codice php non permette di accedere alla pagina se non è stato effettuato il login, infatti, se la variabile *start_time* non è settata riporta l'utente alla pagina di login, mentre se è settata ma è attiva da più di 1h riporta l'utente alla pagina di logout.

```
<?php
    if(!isset($_SESSION)) {
        session_start();
    }
    if (!isset($_SESSION['start_time'])) {
        header('Location: LoginRegister.php');
        die; // per essere sicuro di chiudere la connessione
con la pagina
    } else {
        $now = time();
        $time = $now - $_SESSION['start_time'];
        if ($time > 3600) {
            header('Location: logout.php');
            die;
        }
    }
?>
```

Questo codice è presente all'interno della pagina checkout e permette di aggiungere al database i prodotti acquistati.

Nella prima parte viene **criptato** il numero della carta di credito, successivamente avviene la connessione al database e viene inserito l'ordine, una volta inserito l'ordine, bisogna inserire nel database il singolo prodotto che il cliente vuole acquistare o tutti i prodotti del carrello; in caso di acquisto di un singolo articolo, mediante il tasto "*compra subito*" presente nella scheda del prodotto, verrà preso l'id dell'articolo salvato nella sessione e verrà inserito nel database, mentre nel caso di più articoli presenti nel carrello, verranno presi tutti gli id dei prodotti dai cookie e saranno inseriti nel database.

```
/*ORDINE*/
$card = crypt(str_replace(" ", "", $_POST['cardnumber1']), 10);
$ship = $_POST['shipment'];
// Connetto al DMBS
$conconnection = new mysqli("localhost", "root", "", "elaborato");
$email = $_SESSION["email"];
// Carico in tabella ordine
```

```

$query = "INSERT INTO `ordine` (`Id`, `NumeroCarta`,
`IndirizzoSpedizione`, `Corriere`, `CodiceFiscaleUtente`) VALUES (NULL,
'$card', '$ship', 'Brt', (SELECT CodiceFiscale FROM `utente` WHERE email
= '$email'))";
$connection->query($query);

/*DETTAGLI ORDINE*/
// Get Id ordine
$query = "SELECT LAST_INSERT_ID()";
$risultati = $connection->query($query);
$tmp = $risultati->fetch_array();
$idOrdine = $tmp[0];

// Compra subito
if(isset($_SESSION["IdP"])){
    $idProduct = $_SESSION["IdP"];
    $query = "INSERT INTO `dettaglioordine` (`Id`, `Quantita`,
`IdProdotti`, `IdOrdine`) VALUES (NULL, '1', '$idProduct',
'$idOrdine')";
    $connection->query($query);
    unset($_SESSION['IdP']);
}
// Acquisto piu prodotti
else{
    $n = count($_COOKIE);
    for ($i=0; $i < $n - 1; $i++) {
        $array = json_decode(array_shift($_COOKIE));
        $idProduct = $array[4];
        $quantity = $array[5];
        // Fix problem NaN cookie
        if ($idProduct != "") {
            $query = "INSERT INTO `dettaglioordine` (`Id`, `Quantita`,
`IdProdotti`, `IdOrdine`) VALUES (NULL, '$quantity', '$idProduct',
'$idOrdine')";
            $connection->query($query);
            echo "<script>document.cookie = $idProduct +
';;expires=Thu, 01 Jan 1970 00:00:01 GMT;'</script>";
        }
    }
}
}

```

Nella pagina di logout viene eliminata la sessione corrente e viene creata una sessione vuota.

Il ciclo for serve per eliminare le informazioni presenti nel carrello, perché lo stesso utente potrebbe entrare con un altro account.

```
<?php
// distruzione sessione corrente
session_start();
session_unset();
session_destroy();
$_SESSION = array();
$n = count($_COOKIE);
for ($i=0; $i < $n; $i++) {
    $array = json_decode(array_shift($_COOKIE));
    $idProduct = $array[4];
    // Fix problem NaN cookie
    if ($idProduct != "") {
        echo $i."\n";
        echo "<script>document.cookie = $idProduct +
';;expires=Thu, 01 Jan 1970 00:00:01 GMT;'</script>";
    }
}
echo "<script>window.location.href = \"index.php\";</script>";
?>
```

Diritti del consumatore

Introdotta con il D.lgs. n. 206 del 6 settembre 2005, a norma dell'art. 7 della l. delega n. 229/2003, in attuazione di una serie di direttive dell'Unione Europea, il Codice del Consumo rappresenta un fondamentale traguardo nella tutela dei consumatori.

Esso sancisce:

- **Stop a spese e costi nascosti su internet:** I consumatori saranno protetti dalle "trappole dei costi" su internet, incluse le situazioni in cui i truffatori si fanno pagare con l'inganno per servizi cosiddetti "gratuiti", quali oroscopi o ricette.
- **Maggiore trasparenza dei prezzi:** I venditori dovranno indicare chiaramente il costo totale del prodotto o servizio, incluso qualunque addebito supplementare. Gli acquirenti online non dovranno pagare spese o altri costi se non ne sono stati adeguatamente informati prima di effettuare l'ordine.

- **Divieto delle caselle preselezionate sui siti web:** Quando si fanno acquisti online, è possibile che vengano offerte opzioni supplementari, quali assicurazioni viaggio o noleggi auto. Questi servizi supplementari possono essere offerti mediante le cosiddette "caselle preselezionate"
- **14 giorni per cambiare idea su un acquisto fatto anche on line:** Il periodo durante il quale i consumatori possono recedere dal contratto di acquisto è portato a 14 giorni di calendario, qualora un venditore non informi chiaramente il cliente circa il diritto di recesso, la durata del periodo di ripensamento è estesa a un anno. Il periodo di recesso decorrerà dal momento in cui il consumatore riceve le merci.
- **Maggiori diritti di rimborso:** I commercianti sono tenuti a rimborsare i consumatori per il prodotto entro 14 giorni dal recesso. Il rimborso deve coprire anche le spese di consegna. In generale, il commerciante assume su di sé il rischio di eventuali danni alle merci che si verificano durante il trasporto fino al momento in cui l'acquirente ne prende possesso.
- **Eliminazione di sovrattasse per l'uso di carte di credito e di servizi di assistenza telefonica:** i commercianti non possono più addebitare ai consumatori costi supplementari per i pagamenti con carta di credito, se non i costi effettivamente sostenuti per offrire tale opzione di pagamento.
- **Più tutele per i consumatori negli acquisti digitali:** Anche le informazioni sui contenuti digitali devono essere più chiare, comprese quelle relative alla compatibilità con hardware e software e all'applicazione di eventuali sistemi tecnici di protezione - che ad esempio limitino il diritto del consumatore di fare copie del contenuto

Progettazione e realizzazione Database

Database


Lo scopo di un database è quello di raccogliere un certo insieme di dati per metterlo poi a disposizione di chi ne faccia richiesta. I database si progettano attraverso dei modelli, come il modello concettuale e il modello logico, che ne semplifica la visione dell'insieme.

Modello concettuale

Il modello concettuale rappresenta la realtà dei dati e le relazioni tra essi attraverso uno schema, il modello che ho utilizzato è quello E/R.

Le diverse entità che compongono lo schema concettuale per la gestione dei dati del mio e-commerce sono:

L'entità **Categoria** è composta da soli 2 campi, questa entità, come suggerisce il nome, serve per suddividere i prodotti in categorie differenti, come: Lifestyle, Jordan, Running. Essa è formata da 2 attributi: l'id che è chiave primaria e il nome categoria che è una varchar.

Categoria	
 Id	int
NomeCategoria	varchar

L'entità **Prodotto** composto da 7 attributi, questa entità rappresenta tutti i prodotti che l'e-commerce vende. Conserva anche informazioni utili per l'utente per esempio la recensione del prodotto che va da una scala da 1 a 5 stelle o lo sconto sul prodotto

Prodotto	
 Id	int
Nome	varchar
Prezzo	int
Giacenza	varchar
Sconto	int
PercorsoImmagine	varchar
Recensione	int

L'entità **Dettaglio Ordine** composta da 2 attributi e rappresenta ogni singolo prodotto inserito nel carrello.

Dettaglio Ordine	
🔑 Id	int
Quantita	int

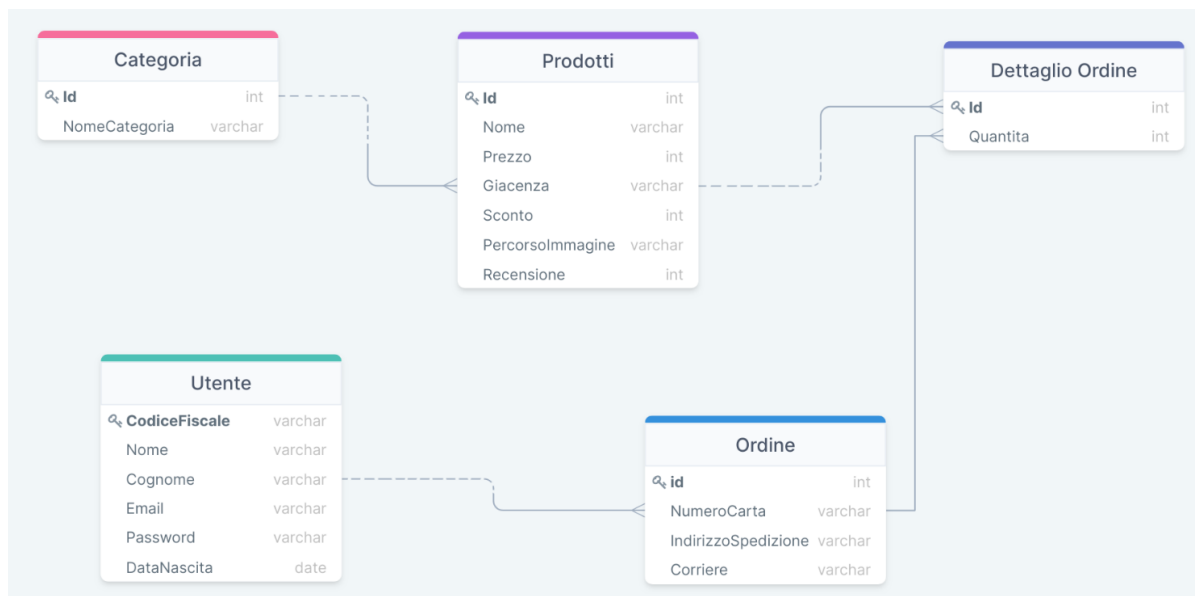
L'entità **Ordine** composta da 4 attributi e rappresenta l'acquisto effettuato da un utente, l'acquisto può essere di 1 o più prodotti. Il numero della carta è criptato per avere una maggiore sicurezza in caso di furto dei dati.

Ordine	
🔑 id	int
NumeroCarta	varchar
IndirizzoSpedizione	varchar
Corriere	varchar

L'entità **Utente** è composta da 6 attributi. Gli utenti si registrano nella pagina dedicata inserendo i loro dati personali come codice fiscale e email. Questa permette anche il funzionamento del sistema di login.

Utente	
🔑 CodiceFiscale	varchar
Nome	varchar
Cognome	varchar
Email	varchar
Password	varchar
DataNascita	date

Lo schema concettuale completo è il seguente:



Tra le entità **categoria** e **prodotto** abbiamo un'associazione 1 a n, dove l'entità categoria può avere uno o più prodotti, mentre il un prodotto deve avere una sola categoria.

Tra le entità **prodotto** e **dettaglio ordine** c'è un associazione 1 a n, dove l'entità prodotto può essere contenuto da uno o più dettagli ordine e ogni dettaglio ordine deve contenere un solo prodotto

Tra le entità **dettaglio ordine** e **ordine** abbiamo un associazione n a 1, dove l'entità dettaglio ordine deve essere contenuta in un ordine, mentre un ordine deve contenere uno o più dettaglio ordine.

Tra le entità **ordine** e **utente** abbiamo un'associazione n a 1, dove l'entità ordine deve essere collegata ad un utente, mentre l'entità utente può essere collegata a uno o più ordini

Modello logico

Dopo il modello concettuale si passa al modello logico che può essere di 3 tipi :

- Gerarchico
- Reticolare
- Relazione

Io ho utilizzato quello relazione che rappresenta il database come un insieme di tabelle.

Categoria (Id, NomeCategoria)

Prodotto (Id, Nome, Prezzo, Giacenza, Sconto, PercorsoImmagine, Recensione, *Id categoria*)

Dettaglio ordine (Id, Quantità, *IdProdotti*, *IdOrdine*)

Ordine (Id, NumeroCarta, IndirizzoSpedizione, Corriere, *CodiceFiscaleUtente*)

Utente (Nome, Cognome , Email, Password, CodiceFiscale, DataNascita)

RELAZIONE	ATTRIBUTO	CHIAVE	FORMATO	DIMENSIONE
Categoria	Id	pk	Numerico	11
	NomeCategoria		Carattere	60
Prodotto	Id	pk	Numerico	11
	Nome		Carattere	50
	Prezzo		Numerico	11
	Giacenza		Carattere	70
	Sconto		Numerico	11
	PercorsoImmagine		Carattere	70
	Recensione		Numerico	11
	Id categoria	fk	Numerico	11
Dettaglio ordine	Id	pk	Numerico	11
	Quantità		Numerico	11
	IdProdotti	fk	Numerico	11
	IdOrdine	fk	Numerico	11
Ordine	Id	pk	Numerico	11
	NumeroCarta		Carattere	60
	IndirizzoSpedizione		Carattere	80
	Corriere		Carattere	50
	CodiceFiscaleUtente	fk	Carattere	16

Utente	CodiceFiscale	pk	Carattere	16
	Nome		Carattere	50
	Cognome		Carattere	50
	Email		Carattere	80
	Password		Carattere	126
	DataNascita		Data	10

Creazione database

Il database è stato creato mediante linguaggio sql.

```
-- Crea il contenitore
CREATE DATABASE Elaborato;
-- Usa il contenitore
USE Elaborato;

CREATE TABLE `categoria`(
    `Id` INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    `NomeCategoria` VARCHAR(60) NOT NULL
);

CREATE TABLE `prodotto`(
    `Id` INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    `Nome` VARCHAR(50) NOT NULL,
    `Prezzo` INT NOT NULL,
    `Giacenza` VARCHAR(70) NOT NULL,
    `Sconto` INT NOT NULL,
    `PercorsoImmagine` VARCHAR(70) NOT NULL,
    `Recensione` INT NOT NULL,
    `IdCategoria` int(11) NOT NULL,
    Foreign Key(`IdCategoria`) references categoria(Id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

CREATE TABLE `utente`(
    `CodiceFiscale` VARCHAR(16) PRIMARY KEY NOT NULL,
    `Nome` VARCHAR(50) NOT NULL,
    `Cognome` VARCHAR(50) NOT NULL,
    `Email` VARCHAR(80) NOT NULL,
    `Password` VARCHAR(126) NOT NULL,
```

```

    `DataNascita` DATE NOT NULL
);
CREATE TABLE `ordine`(
    `Id` INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    `NumeroCarta` VARCHAR(60) NOT NULL,
    `IndirizzoSpedizione` VARCHAR(80) NOT NULL,
    `Corriere` VARCHAR(50) NOT NULL,
    `CodiceFiscaleUtente` varchar(16) NOT NULL,
    Foreign Key(`CodiceFiscaleUtente`) references
utente(`CodiceFiscale`)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
CREATE TABLE `dettaglioordine`(
    `Id` INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    `Quantita` INT NOT NULL,
    `IdProdotti` int(11) NOT NULL,
    `IdOrdine` int(11) NOT NULL,
    Foreign Key(`IdProdotti`) references prodotto(`Id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
    Foreign Key(`IdOrdine`) references ordine(`Id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

```

Le interrogazioni annidate

Linguaggio SQL

SQL (acronimo di Structured Query Language) è il linguaggio di interrogazione per Database Management Systems (DBMS) più diffuso, viene definito linguaggio per DBMS universale, è un linguaggio standardizzato, infatti se un utente cambia DBMS non ha bisogno di apprendere un nuovo linguaggio

Con il linguaggio SQL si possono svolgere vari tipi di operazioni: come la creazione o rimozione del database stesso, permette anche la creazione, modifica e eliminazione di tabelle e degli elementi presenti nelle tabelle ma l'utilità maggiore consiste nell'elaborazione dei dati.

I costrutti più utilizzati sono:

- **SELECT** per selezionare le colonne di nostro interesse.
- **FROM** per indicare le tabelle su cui eseguire la query.
- **WHERE** per indicare condizioni di filtro e prelevare i soli dati che soddisfano tali condizioni.
- **ORDER BY** per impostare un ordine di visualizzazione dei dati.

Interrogazioni annidate

Un'**interrogazione annidata** è una query che sta all'interno di un'altra query: la query interna chiamata anche **subquery** passa i risultati alla query esterna che li verifica nella condizione che segue la clausola Where. Una query situata a un livello più basso rispetto ad un'altra viene definita "**child**", al contrario la query situata ad un livello più alto rispetto alla child viene definita "**parent**". Le subquery possono essere suddivise in: subquery che restituiscono un solo valore e subquery che restituiscono un insieme di valori. La subquery più semplice è quella che restituisce un singolo valore e possiamo utilizzare i normali operatori logici come = > <.

Quando una subquery restituisce più righe, possiamo confrontare i risultati attraverso gli operatori ANY, SOME, IN e ALL.

Clausole exist, all, any

Il predicato **ANY** è vero se il confronto è vero per almeno uno dei valori dell'elenco. La condizione di ricerca è falsa se la sottoquery restituisce un insieme vuoto oppure se il confronto è falso per ciascuno dei valori restituiti dalla sottoquery.

Il predicato **ALL** restituisce vero se il confronto è vero per ciascuno dei valori in Elenco. La condizione di ricerca è falsa se il confronto è falso per almeno uno tra i valori dell'elenco restituito dalla sottoquery.

Il predicato **EXISTS** controlla se vengono restituite righe dall'esecuzione della sottoquery: la condizione di ricerca è vera se la Select nidificata produce una o più righe come risultato, è falsa se la subquery restituisce un insieme vuoto. Il predicato Exists può essere negato nella costruzione della condizione di ricerca inserendo la parola **Not** prima di Exists

Le interrogazioni annidate all'interno del sito

All'interno del sito, nella pagina index possiamo trovare una query annidata, che **visualizza** i prodotti **più venduti** con lo sconto più alto.

```
SELECT prodotto.percorsoimmagine,
       prodotto.prezzo,
       prodotto.nome,
       prodotto.id,
       prodotto.sconto,
       prodotto.percorsoimmagine,
       prodotto.recensione,
       Sum(dettaglioordine.quantita) AS SommaQuantita
FROM   dettaglioordine,
       prodotto
WHERE  dettaglioordine.idprodotti = prodotto.id
       AND prodotto.id = ANY (SELECT prodotto.id
                              FROM   prodotto
                              WHERE  prodotto.sconto >= 75)
GROUP BY prodotto.id
ORDER BY Sum(dettaglioordine.quantita) DESC
```

Analizziamo la query:

Nel from ho importato le tabelle **dettaglio ordine** e **prodotto**, nel where ho effettuato il join tra le 2 tabelle, infatti importando 2 o più tabelle il risultato è il prodotto logico, cioè tutte le possibili combinazioni fra le righe delle tabelle. Il join permette di associare ogni riga di prodotto a ogni riga di dettaglio ordine. La sottoquery restituisce ogni prodotto con uno sconto maggiore del 75%, quindi la condizione presente nel where dice che l'id del prodotto deve essere uguale ad almeno un id prodotto che abbia almeno uno sconto del 75%.

Visto che più prodotti si ripetono ho raggruppato i risultati per l'id di ogni prodotto e li ho ordinati in base alla somma di tutte le quantità acquistate.

Un'altra query annidata presente nel sito è quella presente nella pagina prodotti che visualizza i prodotti più acquistati dai giovani.

```
SELECT *
FROM   utente,
       prodotto,
       dettaglioordine,
       ordine
WHERE  utente.codicefiscale = ordine.codicefiscaleutente
      AND ordine.id = dettaglioordine.idordine
      AND dettaglioordine.idprodotti = prodotto.id
      AND utente.codicefiscale = ANY (SELECT utente.codicefiscale
                                       FROM   utente
                                       WHERE
                                           Datediff(CURRENT_DATE, utente.datanascita) /
                                           365.25 <= 18)
GROUP BY prodotto.id
ORDER BY Sum(dettaglioordine.quantita) DESC
```

Analizziamo la query:

Nel from importo 4 tabelle: utente, prodotto, dettaglioordine, ordine.

Nel where effettuo i diversi join e prendo tutti gli utenti che hanno un codice fiscale uguale ad almeno uno presente nella sottoquery.

La sottoquery mi restituisce tutti gli utenti giovani cioè tutti gli utenti che hanno un'età minore o uguale a 18 anni.

Visto che più prodotti si ripetono raggruppo i risultati per ogni prodotto e li ordino in base alla somma di tutte le quantità acquistate di ogni prodotto.

Questa subquery è nella pagina di checkout e permette di inserire un nuovo ordine

```
INSERT INTO ordine (Id, NumeroCarta, IndirizzoSpedizione,
Corriere, CodiceFiscaleUtente)
VALUES (NULL, 1, 1, 1, (
    SELECT CodiceFiscale
    FROM utente
    WHERE email = '$email'
))
```

Analizziamo la query:

Questa query è stata pensata perché durante il login nella sessione viene salvata solamente l'email, quindi per evitare di creare una query che prendesse il codice fiscale mediante l'email e un'altra query che inserisse l'ordine nel database, è stata creata una subquery che prende il codice fiscale con l'email, inserita direttamente nella query per l'inserimento dell'ordine in modo da creare una singola query.

Alternative alle interrogazioni annidate

Le interrogazioni viste precedentemente possono essere costruite in maniera differente, per esempio sappiamo che '**= any**' è uguale a '**IN**', applicando questo ragionamento alla prima query che visualizza i prodotti più venduti con lo sconto più alto possiamo ottenere:

```
SELECT prodotto.percorsoimmagine,
       prodotto.prezzo,
       prodotto.nome,
       prodotto.id,
       prodotto.sconto,
       prodotto.percorsoimmagine,
       prodotto.recensione,
       Sum(dettaglioordine.quantita) AS quantita
FROM   dettaglioordine,
       prodotto
WHERE  dettaglioordine.idprodotti = prodotto.id
       AND prodotto.id IN (SELECT prodotto.id
                           FROM   prodotto
                           WHERE  prodotto.sconto >= 75)
GROUP BY prodotto.id
ORDER BY Sum(dettaglioordine.quantita) DESC
```

Possiamo riscrivere la stessa query in un altro modo, visto che l'id del prodotto deve essere **uguale** ad almeno un prodotto con **sconto >= 75%**, possiamo dire che l'id del prodotto deve essere **diverso** da tutti i prodotti che hanno uno **sconto minore** del 75%, quindi la query diventa:

```
SELECT prodotto.percorsoimmagine,
       prodotto.prezzo,
       prodotto.nome,
```

```

        prodotto.id,
        prodotto.sconto,
        prodotto.percorsoimmagine,
        prodotto.recensione,
        Sum(dettaglioordine.quantita) AS quantita
FROM    dettaglioordine,
        prodotto
WHERE   dettaglioordine.idprodotti = prodotto.id
        AND prodotto.id != ALL (SELECT prodotto.id
                                FROM    prodotto
                                WHERE   prodotto.sconto < 75)
GROUP  BY prodotto.id
ORDER  BY Sum(dettaglioordine.quantita) DESC

```

Lo stesso ragionamento lo possiamo applicare con la query che visualizza i prodotti più acquistati dai giovani. Infatti si può dire che gli utenti devono essere diversi da tutti le persone che hanno più di 18 anni.

```

SELECT  prodotto.id,
        Sum(dettaglioordine.quantita)
FROM    utente,
        prodotto,
        dettaglioordine,
        ordine
WHERE   utente.codicefiscale = ordine.codicefiscaleutente
        AND ordine.id = dettaglioordine.idordine
        AND dettaglioordine.idprodotti = prodotto.id
        AND utente.codicefiscale != ALL (SELECT utente.codicefiscale
                                          FROM    utente
                                          WHERE
                                          Datediff(CURRENT_DATE, utente.datanascita) / 365.25 > 18)
GROUP  BY prodotto.id
ORDER  BY Sum(dettaglioordine.quantita) DESC

```

Sistemi

Piano hosting e rete del sito

Per il mio e-commerce dovevo scegliere se usare un piano **hosting** o un piano **housing**. Nell'housing i server risiedono nella propria rete, bisogna avere schede di rete di qualità, una rete veloce, backup automatico e servizio di restore e accesso da terminale. Il costo è maggiore rispetto al piano hosting ma i vantaggi sono:

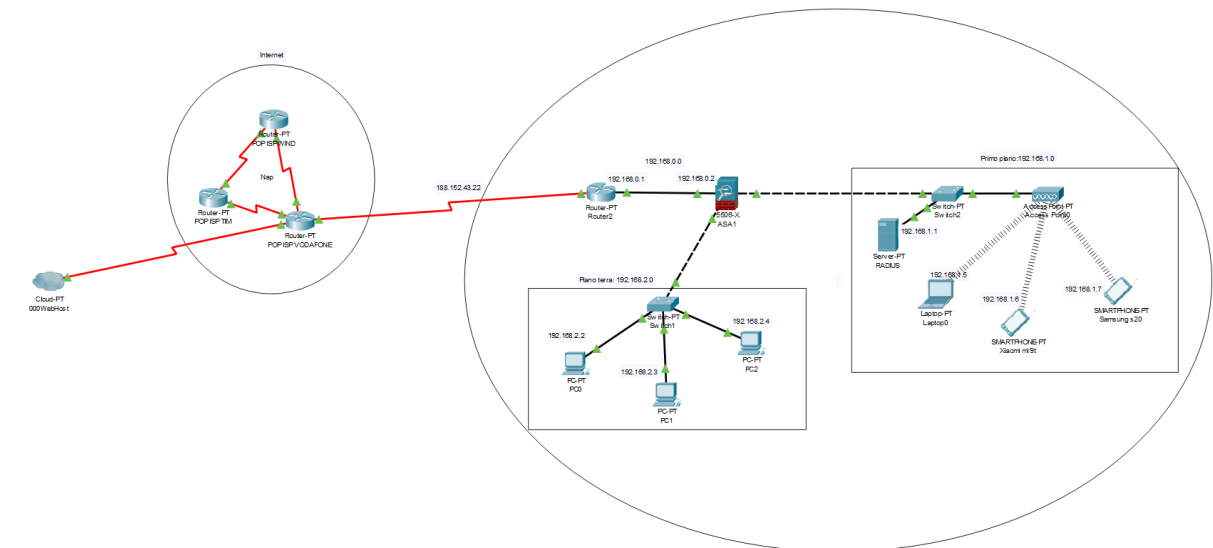
- Le pagine si caricano più **velocemente**
- Possibilità di **installare** applicativi personalizzati
- Accesso più diretto alle funzione di **amministrazione** del server
- Possibilità di installare **add-on**.

Nell'hosting invece il server appartiene al provider e permette di avere uno staff tecnico sempre pronto a intervenire in caso di problemi.

L'e-commerce usa un piano **hosting** del provider 000webhost, soprattutto per l'elevato numero di utenza, quindi era necessario un server prestante e una banda molto elevata. Ho scelto il piano hosting anche per gli altri vantaggi come:

- Manutenzione a carico del gestore,
- La presenza di un **firewall**
- La presenza di un pannello di controllo per gestire i permessi sui file o gli utenti **FTP**.
- Sono presenti strumenti per **monitorare** l'attività dei siti web
- Ha un piano di **assistenza online**

Il mio provider offre anche una protezione contro i **DDOS**, che è fondamentale per e-commerce, in quanto se il sito va offline, gli utenti non possono fare più acquisti e la credibilità/affidabilità del e-commerce potrebbe scendere.



Questa è la mia rete, a sinistra c'è 000webhost che è il mio provider, alla destra del provider è rappresentato internet, mediante il collegamento dei vari isp, sulla destra possiamo vedere una rete casalinga di un possibile **cliente** dell'e-commerce.

La rete casalinga ha un router con un firewall, nel piano terra sono presenti 3 computer fissi collegati con cavi fisici, mentre nel secondo piano è presente un server RADIUS collegato allo switch mentre gli host sono collegati ad un access point, sono stati utilizzati degli indirizzi di classe C, in quanto questa tipologia di indirizzi è quella solitamente utilizzata nelle reti domestiche.

Tutti i componenti utilizzati nella rete sono:

- **Router:** è un dispositivo che collega tra loro più reti scegliendo il percorso migliore per i dati
- **Switch:** Permette di collegare alla rete vari nodi, ogni dispositivo collegato con uno switch può essere identificato dal suo indirizzo MAC, infatti può inoltrare dei pacchetti ai nodi che hanno un determinato MAC address
- **Access Point:** è un dispositivo di rete che collegato ad una LAN permette all'utente di accedervi con la connessione wireless
- **Firewall:** è un dispositivo per la sicurezza della rete che permette di monitorare il traffico in entrata e in uscita.
- **Server RADIUS:** Permette l'autenticazione ai dispositivi che si connettono via wireless

Il provider 000webhost gestisce vari server come HTTP e FTP.

FTP è un protocollo utile al trasferimento dei file. L'FTP utilizza 2 canali TCP separati:

- **Connessione di controllo:** Utilizza la porta 21 e viene usato per lo scambio di informazioni di controllo tra client e server come utente e password.
- **Connessione dati :** Utilizza la porta 20 e si occupa del trasferimento dei file.

Il protocollo **HTTP** permette la comunicazione tra un client e un web server.

Un esempio di comunicazione con il protocollo HTTP:

- Si apre una connessione TCP tra il client e il server
- Il browser richiede una risorsa al **web server**
- Il server risponde
- Si chiude la connessione

La connessione creata da **HTTP 1.0** è di tipo non permanente e viene trasmessa una singola pagina html alla volta, mentre la connessione creata da **HTTP 1.1** è di tipo permanente, infatti la connessione si chiude solamente dopo il time-out.

I messaggi HTTP sono composti da una **start-line** , un **header** e da un **body**.

Lo **start-line** contiene:

- Il metodo: GET, POST, HEAD, PUT, DELETE
- Il percorso del server
- La versione del protocollo HTTP

L'Header: Sono le informazioni sul tipo di documento che il client riceve dal server

Body: Contiene i dati che il client e il server si scambiano

Il protocollo HTTP è **stateless** e quindi le richieste del client non lasciano nessuna informazione sul server, per ovviare a questo problema si usano i cookie. I **cookie** è un file memorizzato nel pc che contiene le informazioni sulle pagine visitate dall'utente, quando l'utente si collega al web server esso estrae le informazioni presenti nei cookie e li analizza.

Le reti wireless

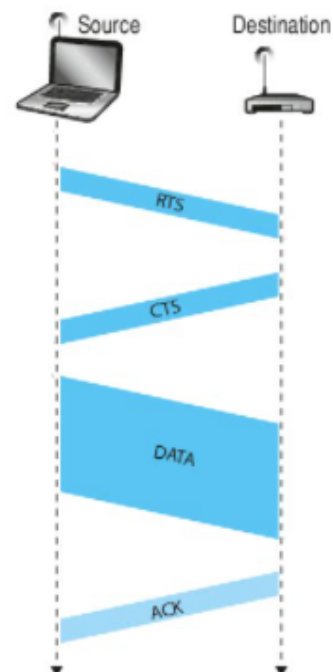
Le reti wireless si suddividono in base alla copertura del segnale e si dividono in:

- **BAN:** Hanno un raggio di copertura massimo di 2 metri.
- **PAN:** Hanno un raggio di copertura di 10 metri, un esempio è il bluetooth che usa la frequenza 2.45Ghz.
- **WLAN:** Hanno un raggio di copertura che va dai 100 ai 500 metri, ed è formata da **Station (STA)**, e **Access Point (AP)**.
- **WWAN:** Sono reti che possono estendersi per chilometri, sono state create diverse generazioni:
 - **1G:** Utilizza la frequenza 800 MHz.
 - **2G:** Utilizza la frequenza 900 MHz.
 - **2.5G:** Utilizza la frequenza 1800 MHz, ed ha una velocità massima di 348 Kbps.
 - **3G:** Utilizza la frequenza 1900 MHz, ed ha una velocità massima di 2 Mbps.

La trasmissione nelle reti wireless

Il **CDMA** è il protocollo di accesso multiplo a canale condiviso più diffuso nelle reti wireless e ad ogni utente viene assegnato un codice univoco, l'utente trasmette sulla stessa frequenza degli altri utenti ma ogni utente ha una propria sequenza per codificare i dati.

Le reti **WLAN** sono soggette a interferenze, a differenza delle reti lan è difficile rilevare le collisioni, quindi per evitarle il mittente può prenotare il canale mediante il meccanismo **RTS/CTS**. In questo meccanismo il mittente manda un piccolo pacchetto all'AP, l'AP manda in broadcast il pacchetto **CTS** in modo che gli altri dispositivi smettano di trasmettere, successivamente il dispositivo invia tutti i dati all'AP. Nella modalità **DCF** i frame vengono scomposti in più frammenti e ogni frammento avrà il proprio ack, questo perchè in caso di collisioni non si dovrà trasmettere il frame intero ma solo una porzione.



Problemi connessioni wireless

I problemi di un sistema wireless sono:

- **L'attenuazione del segnale:** Le onde elettromagnetiche hanno problemi nel superare gli ostacoli, in base al materiale il segnale potrebbe essere più o meno attenuato, nella seguente tabella abbiamo un esempio di materiali disposti in ordine di attenuazione del segnale.

Debole	Medio	Alto	Molto alto
Plastica Vetro Legno	Vetro colorato Mattoni Gesso	Ceramica Cemento Vetro blindato	Metallo Cemento Armato

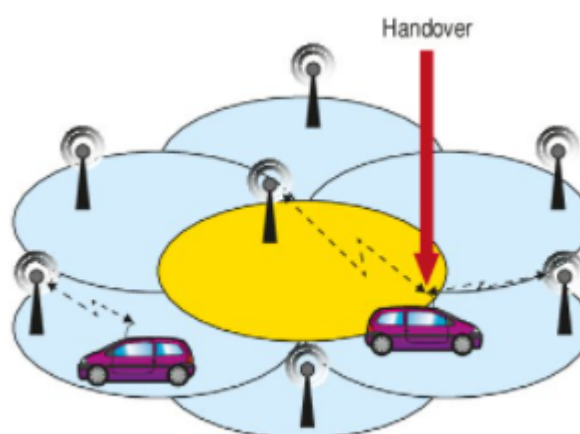
- **Interferenza da parte di altre sorgenti:** La presenza di più sorgenti che trasmettono sulla stessa frequenza può portare a collisioni
- **Propagazione su più cammini:** Il segnale si riflette su oggetti e sul terreno compiendo più cammini al ricevitore arrivano perciò più repliche del segnale trasmesso, sfasate e ritardate in modo diverso a seconda del tragitto percorso.
- **Shadowing:** sono zone d'ombra causate dalla presenza di ostacoli come edifici
- **Effetto Doppler**

Posizionamento degli host

Handoff consiste nella situazione in cui l'host si sposta dall'area di copertura di una stazione base all'altra cambiando il suo punto di collegamento.

Gli handoff possono essere di 2 tipi:

- **Handoff Orizzontale:** un terminale passa da un'AP ad un altro avente la stessa tecnologia
- **Handoff Verticale:** un terminale passa da un'AP ad un altro avente una diversa tecnologia

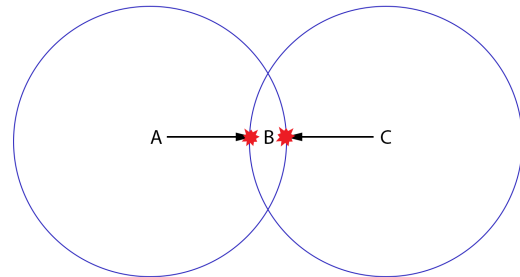


Problema della stazione nascosta (hidden terminal)

Avendo il caso di 3 stazioni radio A B C dove B è nel mezzo, A e C non si vedono ma possono comunicare con B, in questo caso A e C pensano che B sia **libero** e trasmettono entrambi, ma trasmettendo **contemporaneamente** la connessione verrà disturbata e il messaggio dovrà essere rimandato.

Quindi:

- B e A possono comunicare
- B e C possono comunicare
- A e C non possono sentirsi ma possono causare interferenza



Problema della stazione esposta (Exposed terminal)

Consiste che abbiamo 4 stazioni radio. La stazione A comunica con la stazione B e la stazione C vuole comunicare con D, ma C ascoltando l'etere sente che è **disturbato**, quindi non trasmette a D, anche se in questo caso D era libero quindi la comunicazione poteva avvenire senza il rischio di collisioni.

