



2020/2021



PCHECKER

SOMMARIO

PChecker - Specifiche	2
Navbar	2
Login	3
Registrazione	4
Home	5
Prodotti	5
Contatti	6
Carrello	7
Pagina Personale	7
Pagina Personale – Admin	8
Aggiornamento dei Dati di Spedizione	8
Funzioni	8
PChecker - Analisi	10
Descrizione	10
Spiegazione Entità e Associazioni	10
Attributi	10
Descrizione di Entità e Associazioni	11
Forme Normali	14
Modello Tabellare	15
Modello E/R	15
Variabili dei Form	16
Stili CSS	17
Funzioni SQL	18
PChecker - Rete	22
Tabelle IP	23
Tabella PAT	23
Protocolli di Sicurezza	24
PChecker – Analisi di Mercato	26

Link Utili

Indirizzo Web del Sito: <https://mattiascotellaro.altervista.org>

Repository GitHub per il Source Code: <https://github.com/sorcho/PChecker.git>

PChecker - Specifiche

PChecker è il sito che desidero portare come base del mio elaborato di Informatica.

Nasce dalla mia passione riguardante il mondo della componentistica hardware dei computer, ambito che mi ha formato e che è stata la scintilla che ha fatto scattare la mia attrazione per l'informatica e per tutto quello che la riguarda.

Il sito presenta un'interfaccia molto minimale, intuitiva e semplice da capire, di conseguenza anche un nuovo utente può ambientarsi facilmente.

Navbar

Il sito sarà composto da una navbar (abbreviazione per Navigation Bar, ovvero la Barra di Navigazione), la quale si presenta in due versioni.

La prima versione la troviamo quando l'utente **non** ha ancora effettuato il login:



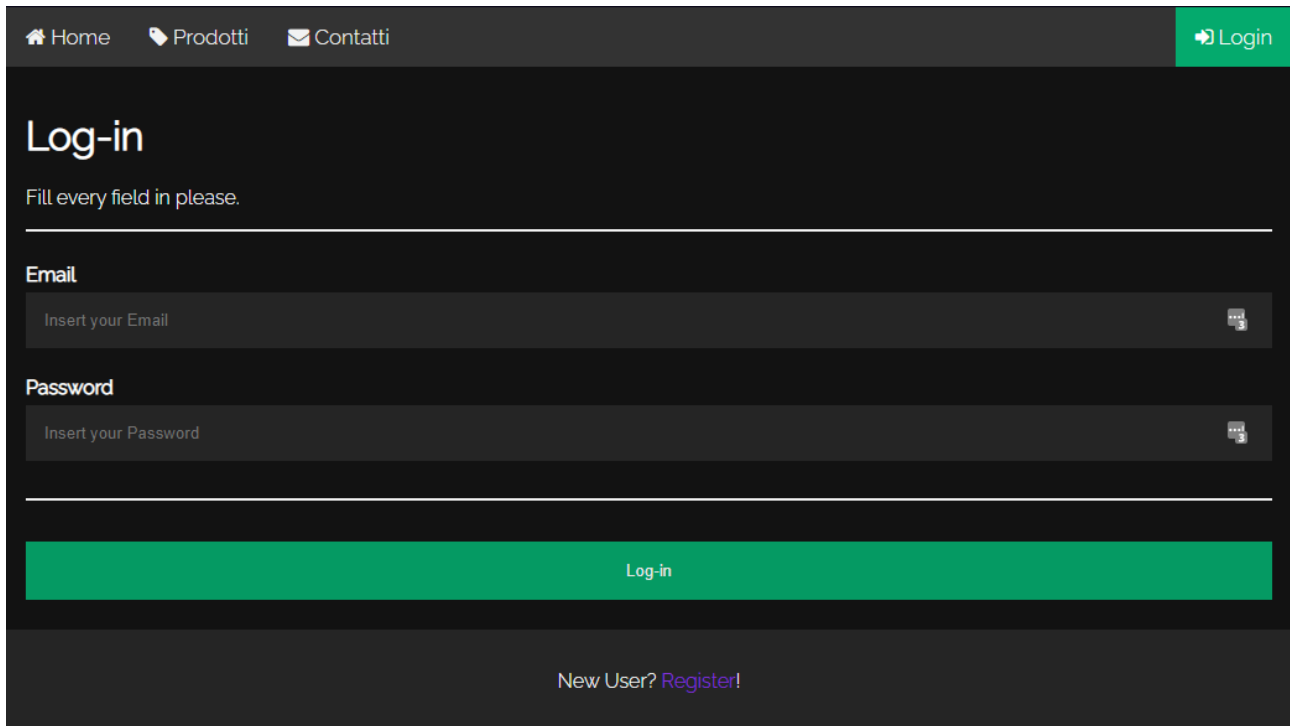
La seconda versione si presenta quando l'utente effettua finalmente il **login**:



Come possiamo vedere, una volta effettuato l'accesso si visualizzerà il nome dell'utente all'estrema destra, oltre che la possibilità di **visualizzare** il carrello personale.

Login

La pagina di Login è **sempre** presente all'interno della navbar, permette all'utente di inserire le proprie credenziali e di conseguenza effettuare l'**accesso** all'interno del sito web, e si presenta nel seguente modo:



Home Prodotti Contatti Login

Log-in

Fill every field in please.

Email

Insert your Email

Password

Insert your Password

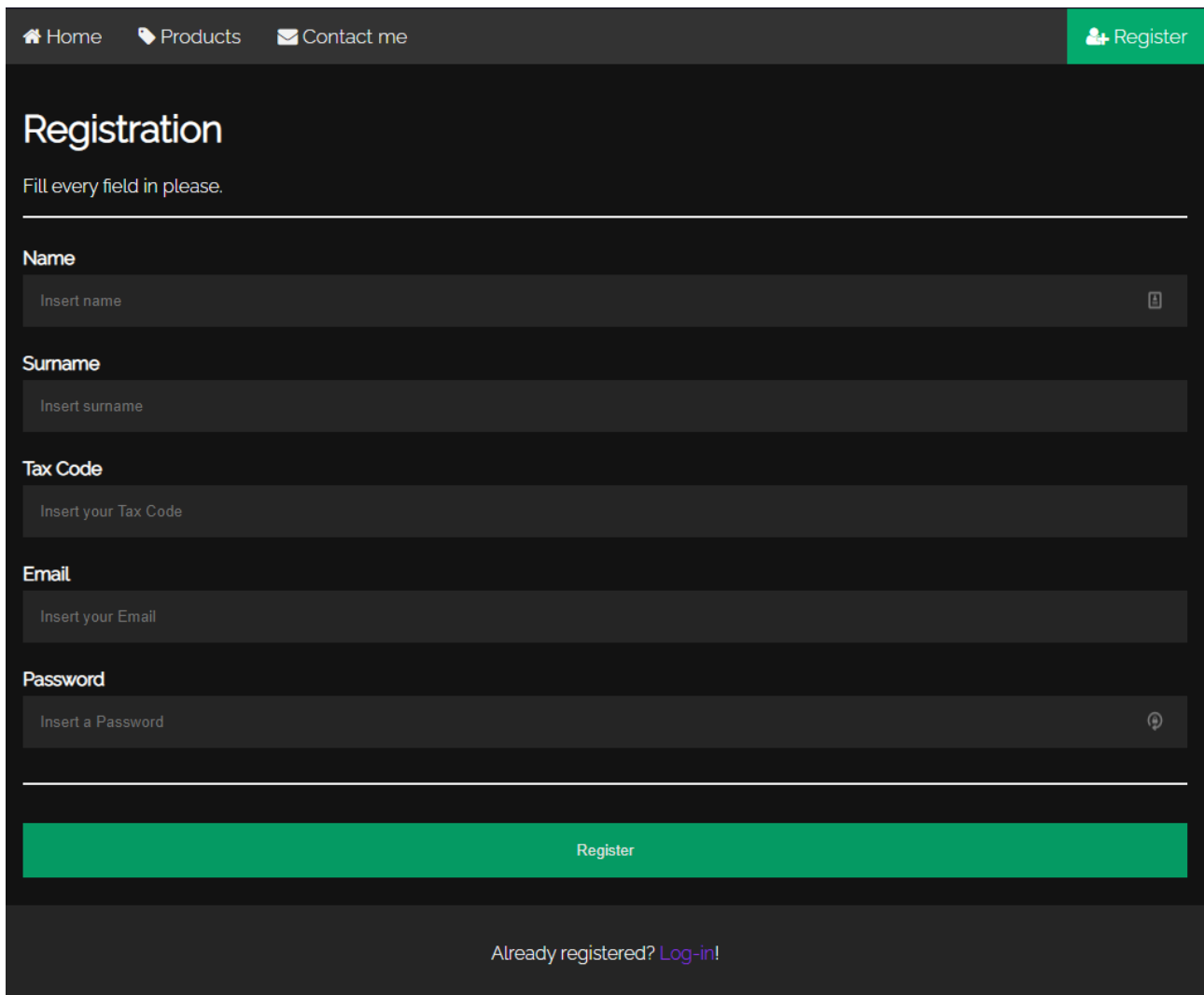
Log-in

New User? [Register!](#)

Il form richiede **Email** e **Password**, dati salvati all'interno di un Database al momento della registrazione.

Registrazione

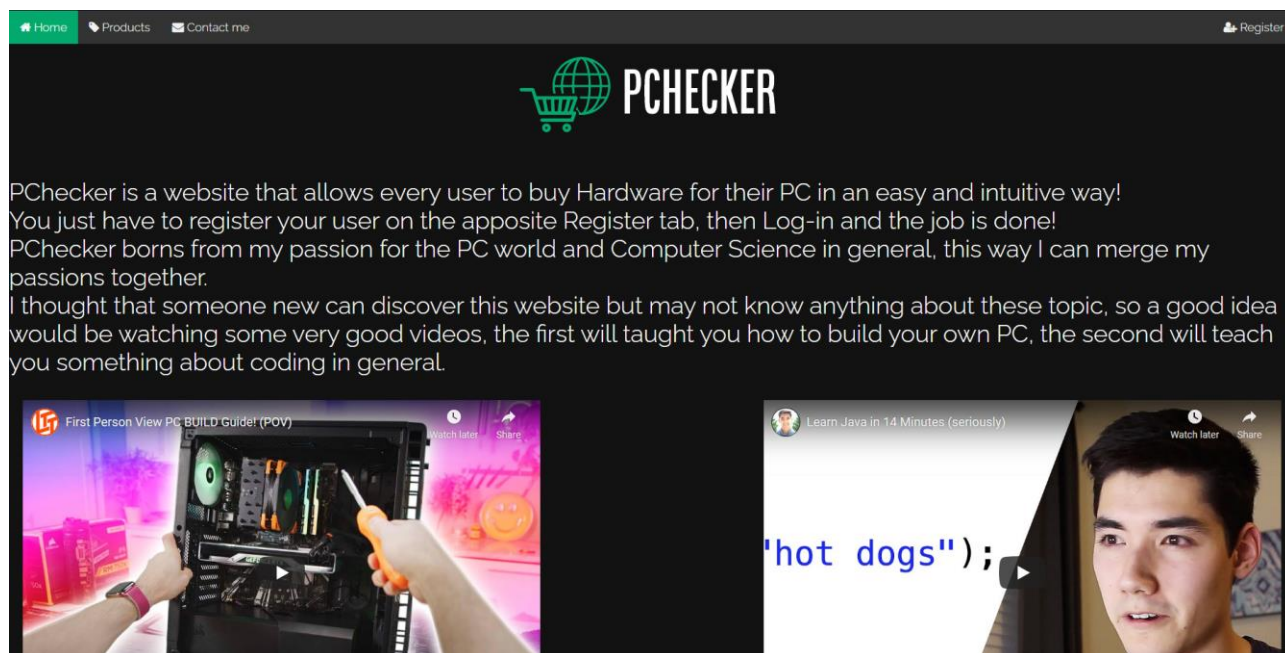
È inoltre presente l'opzione per **registrarsi** in caso si fosse dei nuovi utenti:

A screenshot of a web registration form. At the top, there is a dark navigation bar with links for 'Home', 'Products', and 'Contact me', and a green 'Register' button with a user icon. Below the navigation bar, the page has a dark background. The title 'Registration' is displayed in white. A subtitle 'Fill every field in please.' is shown. The form consists of several input fields: 'Name' with placeholder 'Insert name', 'Surname' with placeholder 'Insert surname', 'Tax Code' with placeholder 'Insert your Tax Code', 'Email' with placeholder 'Insert your Email', and 'Password' with placeholder 'Insert a Password'. Each field has a small icon on the right side. Below the fields is a large green 'Register' button. At the bottom, there is a link 'Already registered? Log-in!'.

Ovviamente questo form è composto da **più** campi, infatti sono presenti anche **Nome**, **Cognome** e **Codice Fiscale**. Il campo più importante è l'**ultimo** e deve essere composto da **16** caratteri, non di più né di meno.

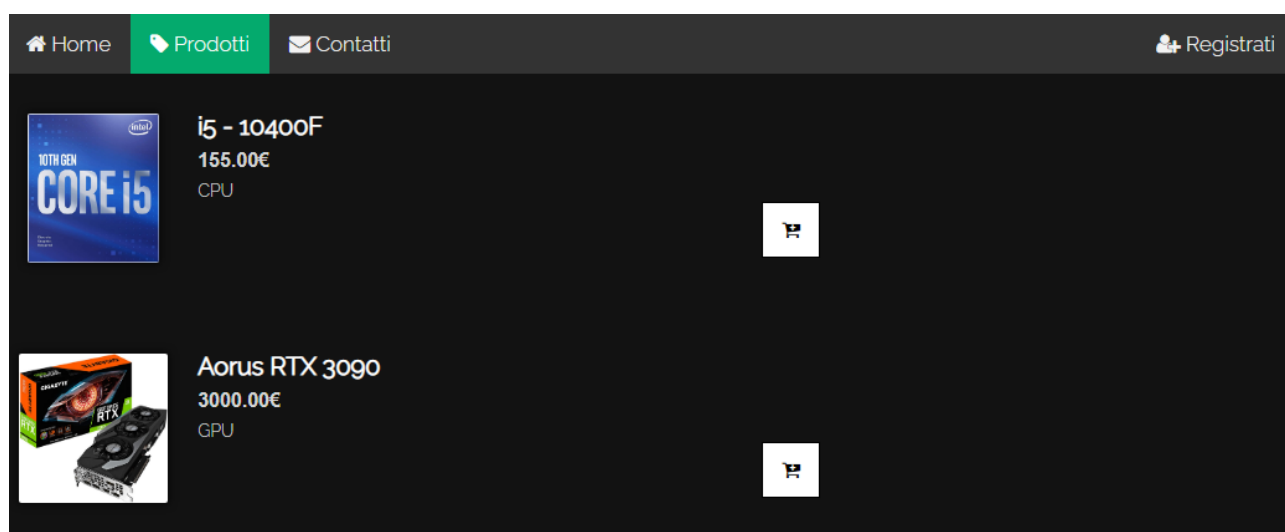
Home

Presenta una **breve** descrizione del sito per renderlo chiaro ai nuovi utenti, insieme a due video la cui presenza è spiegata nel paragrafo soprastante:



Prodotti

La Zona **Prodotti** consente di visualizzare i prodotti disponibili all'interno di questo e-shop:

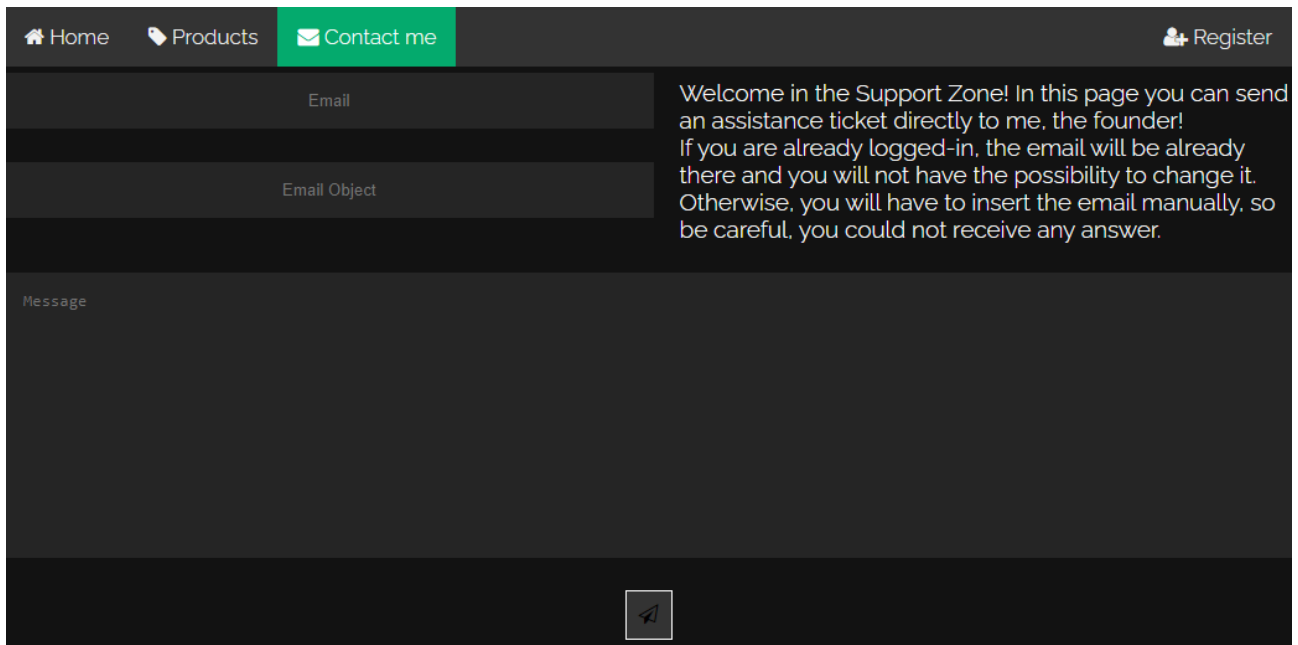


In alternativa è possibile inserire una pagina intermedia che permetta di selezionare i prodotti per tipologie piuttosto che scorrere tra tutti quelli disponibili.

Contatti

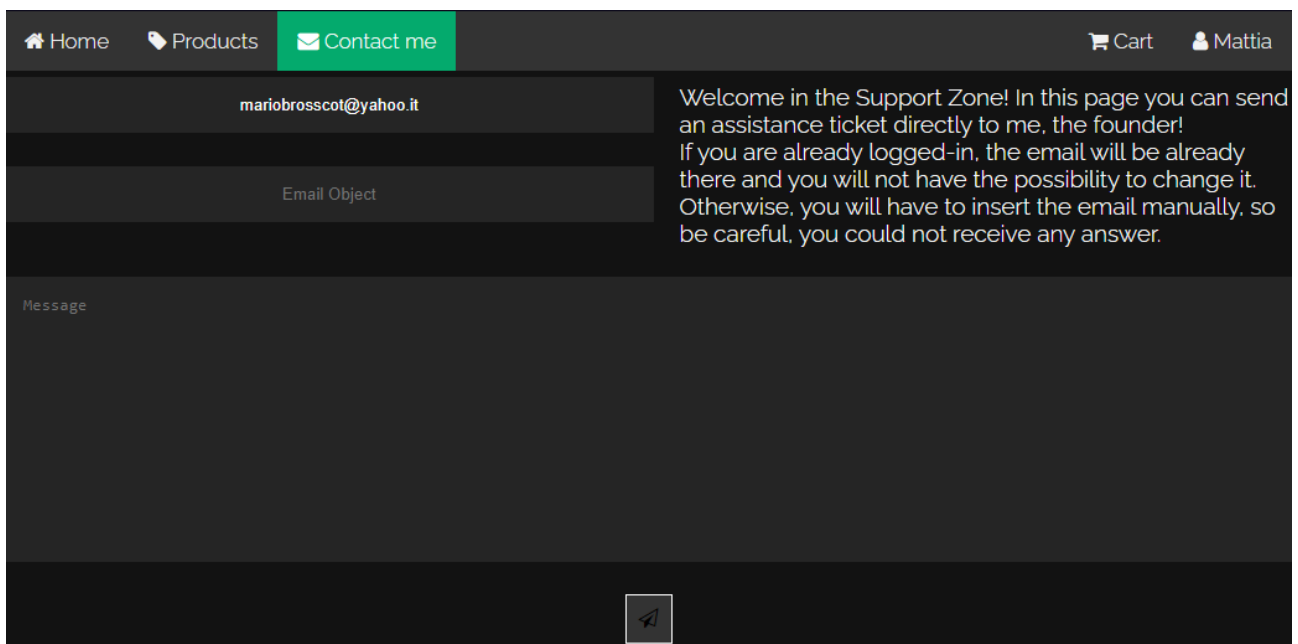
La pagina dei contatti permette di mandare una email per richiedere **supporto**, e anche questa varia in base a sé l'utente ha effettuato o meno l'accesso al sito.

Senza Accesso:



The screenshot shows a web interface with a dark theme. At the top, there is a navigation bar with links for 'Home', 'Products', and 'Contact me' (which is highlighted in green). On the right side of the navigation bar, there is a 'Register' link with a user icon. Below the navigation bar, the page is divided into two main sections. On the left, there are two input fields: 'Email' and 'Email Object'. Below these fields is a large text area labeled 'Message'. On the right side, there is a welcome message: 'Welcome in the Support Zone! In this page you can send an assistance ticket directly to me, the founder! If you are already logged-in, the email will be already there and you will not have the possibility to change it. Otherwise, you will have to insert the email manually, so be careful, you could not receive any answer.' At the bottom right of the page, there is a small icon of a paper plane inside a square box.

Con Accesso:

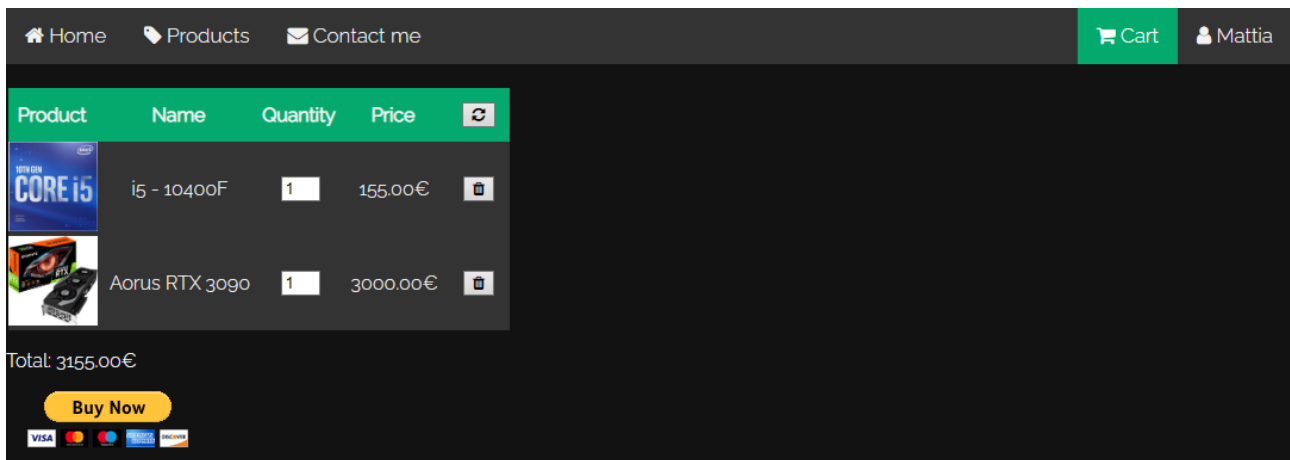


The screenshot shows the same web interface as the previous one, but with some differences. In the navigation bar, there is a 'Cart' link and a user profile link labeled 'Mattia' with a user icon. The 'Email' input field now contains the text 'mariobrosscot@yahoo.it'. The 'Email Object' input field is empty. The 'Message' text area is also empty. The welcome message on the right is the same as in the previous screenshot. At the bottom right, there is a small icon of a paper plane inside a square box.

La differenza sostanziale consiste nel fatto che se l'utente effettua l'accesso, la mail sarà **preinserita** e non ci sarà modo di cambiarla, al contrario l'utente dovrà inserirla **manualmente**.

Carrello

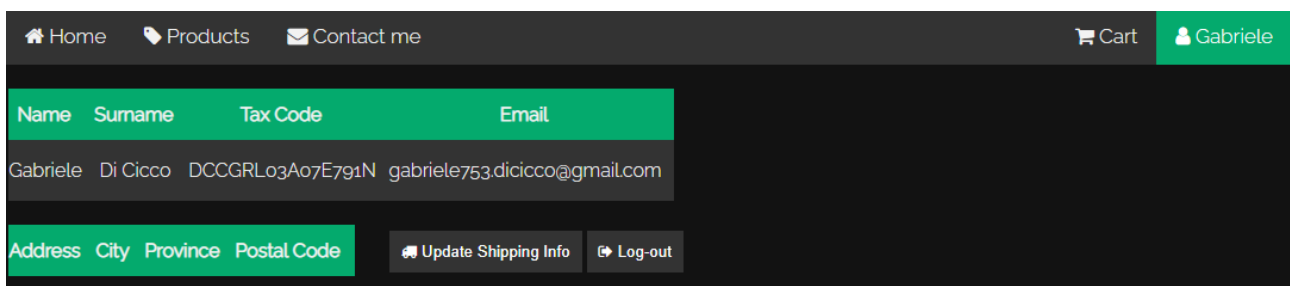
All'interno di questa sezione possiamo notare i prodotti che il cliente desidera acquistare, inoltre è possibile effettuare il pagamento tramite **PayPal**:



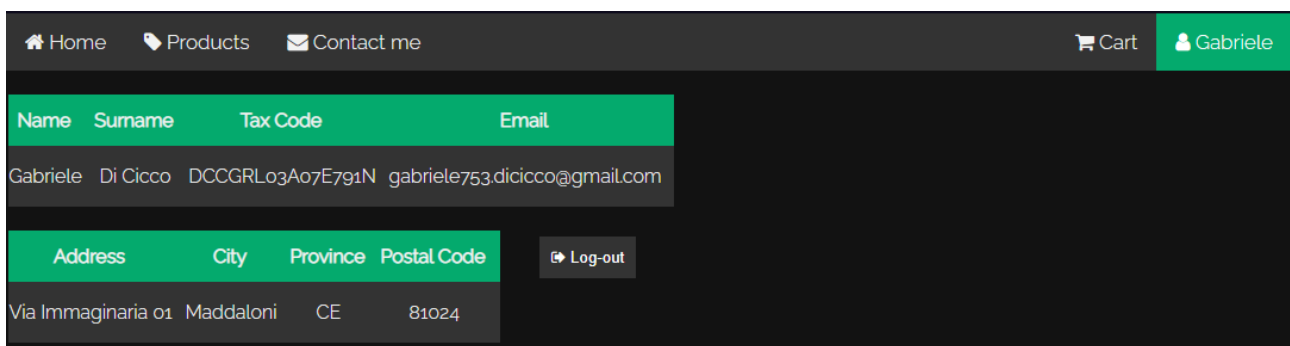
La Pagina Personale

Anche per questa pagina è disponibile in **due** versioni.

L'utente **non** ha immesso le sue informazioni di spedizione:

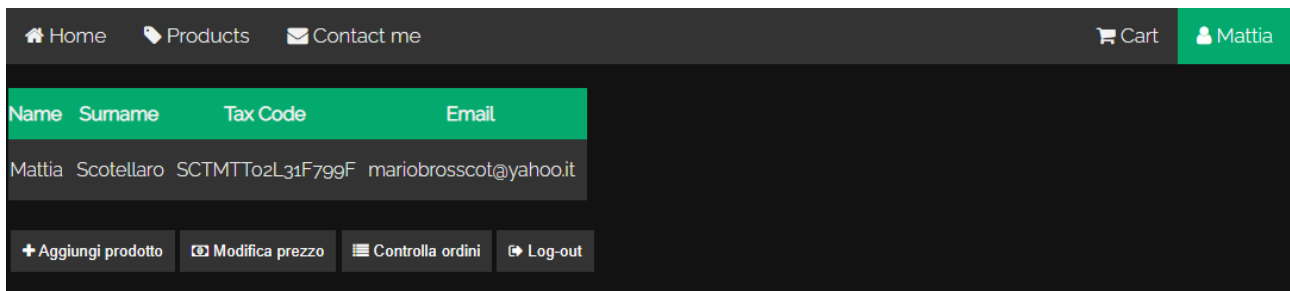


L'utente **ha** immesso le informazioni di spedizione:



La Pagina Personale – Admin

Solo l'Amministratore sarà capace di visualizzare la seguente pagina, in fatti al posto dei dati di spedizione, vedremo diverse funzioni **aggiuntive**, come:



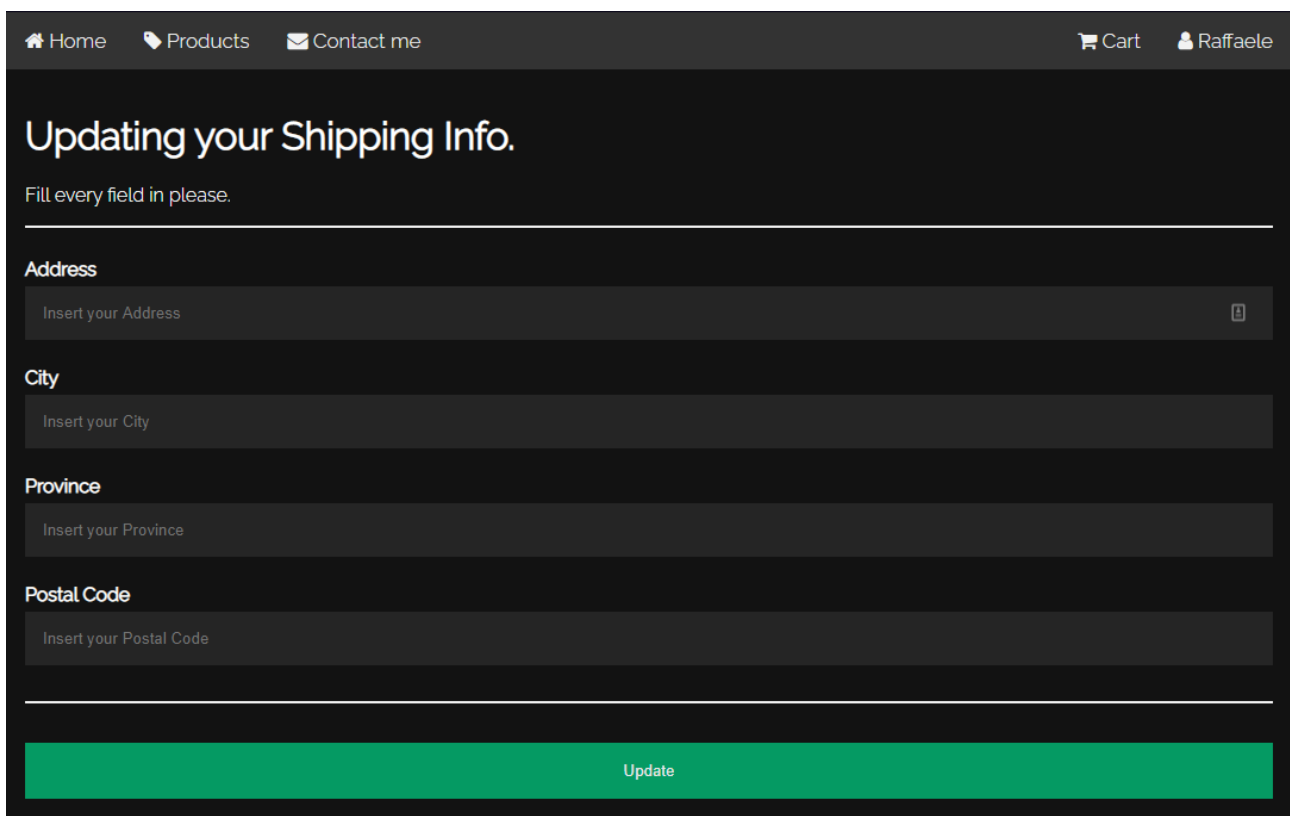
The screenshot shows the Admin user profile page. At the top, there is a navigation bar with links for Home, Products, and Contact me, and a user profile section for Mattia. Below the navigation bar, there is a table with the following data:

Name	Surname	Tax Code	Email
Mattia	Scotellaro	SCTMTTo2L31F799F	mariobrosscot@yahoo.it

Below the table, there are four buttons: + Aggiungi prodotto, Modifica prezzo, Controlla ordini, and Log-out.

Aggiornamento dati di Spedizione

Nel caso l'utente non avesse inserito ancora dei dati di spedizione li può aggiungere tramite l'apposito **tasto**, una volta premuto si aprirà la pagina **seguente**:



The screenshot shows the 'Updating your Shipping Info.' form. The form has a title 'Updating your Shipping Info.' and a subtitle 'Fill every field in please.' Below the subtitle, there are four input fields: Address, City, Province, and Postal Code. Each field has a placeholder text 'Insert your [field name]'. At the bottom of the form, there is a green button labeled 'Update'.

PChecker avrà **due** tipologie di utilizzi, uno per l'utente **base** e uno per l'**amministratore**.

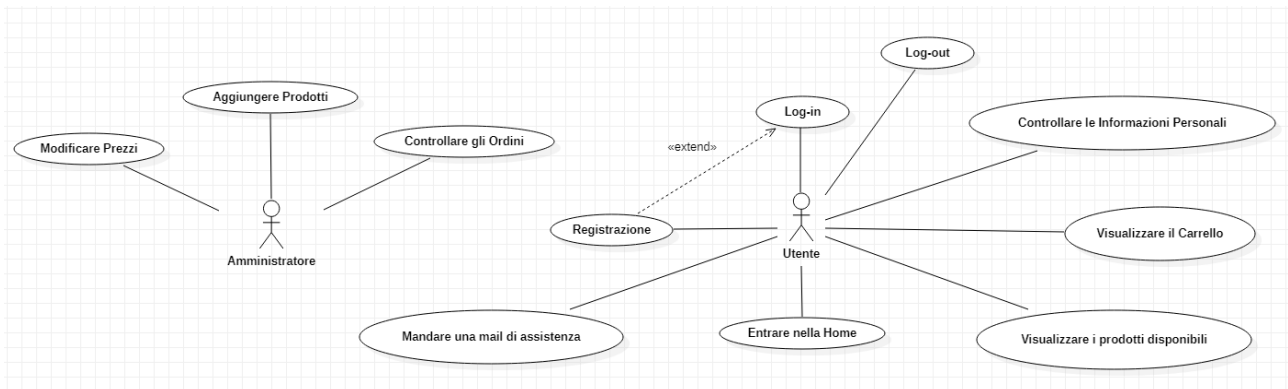
Funzioni per l'utente:

- Effettuare la registrazione;
- Effettuare il Log-in;
 - o Effettuare il Log-out;
- Visualizzare i prodotti disponibili nel negozio;
- Contattare l'amministratore tramite la pagina Contatti;
- Controllare i prodotti presenti nel Carrello;
- Controllare le informazioni personali all'interno della pagina personale.

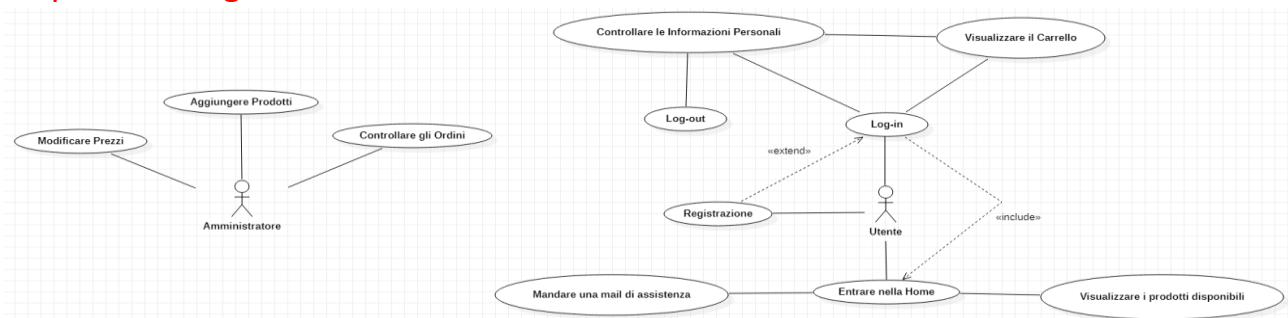
Funzioni per l'amministratore:

- Aggiungere nella pagina Prodotti nuovi oggetti;
- Controllare gli ordini di un utente;
- Modificare i prezzi di tutti i prodotti.

Use Case Diagram:



Sequential Diagram:



PChecker – Analisi

Descrizione:

PChecker sfrutterà diversi linguaggi, tra programmazione e formattazione, ovvero:

- **HTML**: il linguaggio che permette di creare le pagine web che l'utente vede;
- **CSS**: è il linguaggio utilizzato per la formattazione dei documenti HTML, tramite i cosiddetti fogli di stile a cascata;
- **MySQL**: permette la creazione del database e delle diverse tabelle che servono a PChecker per immagazzinare informazioni e dati importanti;
- **PHP**: probabilmente il linguaggio più usato, permette di salvare i dati presi dall'HTML e inserirli successivamente all'interno del database MySQL, o viceversa, ovvero prendere i dati dal Database e mostrarli sulla pagina web.

Spiegazione delle entità e delle associazioni:

Molto importante per la realizzazione del sito **PChecker** è stata l'identificazione delle entità. Abbiamo a che fare con tre entità principali, seguite da due associazioni. La prima entità trovata è stata **Utente**, consiste nelle informazioni che l'utente utilizzerà per effettuare il Log-in al sito; di seguito troviamo **Prodotti**, entità che specifica i dati riguardanti i diversi componenti messi in vendita sul sito; come ultima entità troviamo **Spedizione**, che contiene i dati per la spedizione di ogni specifico utente. Come abbiamo detto oltre alle entità sono presenti due tabelle associative, la prima è **Compra**, tabella molto semplice, specifica quale utente ha comprato quale prodotto, quindi verrà utilizzata al momento dell'ordine chiuso, la seconda tabella associativa è **Carrello**, ogni utente ha a disposizione il proprio carrello personale e non si avrà modo di visualizzare quello degli altri utenti.

Attributi delle tabelle:

Utente -> nome, cognome, codice_fiscale, email, password, root;

Prodotti -> ID, modello, img_dir, prezzo, tipologia;

Spedizione -> id, ID_Utente, indirizzo, città, provincia, CAP;

Compra -> ID, ID_Utente, ID_Componente, Data_Acquisto, Importo;

Carrello -> ID, ID_Utente, ID_Prodotto, quantità;

Descrizione di entità e associazioni:

Legenda [Tabella, Chiave Primaria, Chiave Esterna, Attributo].

Utente:

- **Nome**: questo attributo contiene il nome dell'utente, inserito dallo stesso al momento della registrazione, lo ritroviamo all'interno delle variabili di sessione disponibili in tutte le pagine, in modo tale da visualizzare il nome del suddetto (solo se ha effettuato l'accesso) al posto dell'opzione di Registrazione/Log-in. Nome non è stato scelto come Chiave Primaria siccome più utenti possono avere lo stesso nome, di conseguenza sarebbe stata una decisione poco sensata;
- **Cognome**: questo attributo contiene il cognome dell'utente, inserito dallo stesso al momento della registrazione, il cognome al contrario del Nome non viene utilizzato nelle variabili di sessione, ma condivide con l'attributo di cui sopra la scelta di non essere utilizzato come Chiave Primaria per motivi ovvi, ad esempio, se un'intera famiglia si iscrive, ci saranno sicuramente dei problemi di gestione;
- **Codice_Fiscale**: il Codice_Fiscale è inserito manualmente dall'utente, deve essere di 16 caratteri, non uno in meno né uno in più. Questo attributo è stato scelto come Chiave Primaria essendo ovviamente univoco per ogni persona, ottimizzando così la gestione degli utenti escludendo poi possibili errori;
- **Email**: la mail viene inserita all'interno del database nel momento in cui l'utente si registra, la mail viene utilizzata per effettuare la registrazione, l'accesso e per mandare una mail di assistenza, di conseguenza bisogna controllare più volte di aver inserito la mail corretta. Email è stato il campo che avevo intenzione di usare come Chiave Primaria, scelta poi evitata siccome è nato l'attributo Codice_Fiscale, quindi ho preferito usare quello per una maggiore sicurezza;
- **Password**: la password è una delle cose più importanti per l'utente in quanto permette l'accesso al proprio account, infatti si è optato per una criptazione hash della stessa, in modo da renderla sicura e unica. Per motivi ovvi la password non è stata scelta come chiave primaria, l'amministratore non saprà mai la tua password, ma allo stesso tempo più utenti possono aver immesso la stessa password;
- **Root**: è il campo che specifica se l'utente che effettua l'accesso è un admin o un utente normale, al momento del login si fa un controllo, se risulta che l'attributo root equivale a 1 si effettuerà l'accesso come admin, altrimenti come utente.

Prodotti:

- **ID**: ID è la chiave primaria di questa tabella, è un INT auto-incrementante e ho scelto questa chiave dopo aver usato “Modello” come PK, ho notato alcuni problemi nella stampa del modello all’interno del Carrello, di conseguenza ho optato per quest’altra opzione;
- **Modello**: modello è l’attributo principale di questa tabella, contiene il nome del prodotto, la PK è stata trasferita al campo ID per problemi di gestione;
- **Img_dir**: contiene la directory dell’immagine del prodotto, l’immagine è unica per ogni prodotto siccome sono tutti diversi, non è stata scelta come PK per ovvi problemi di gestione siccome contiene caratteri speciali;
- **Prezzo**: Il prezzo è un double, in modo da contenere anche valori decimali. È evidente che non è stata scelta come Primary Key in quanto diversi prodotti possono avere lo stesso prezzo;
- **Tipologia**: Si specifica la tipologia del prodotto, ovvero a che categoria appartiene, come GPU, CPU, MB, e così via. Non è stato scelto come Chiave Primaria siccome sono tranquillamente presenti più prodotti con la stessa tipologia.

Spedizione:

- **ID**: ID è la chiave primaria di questa tabella, è un INT auto-incrementante e ho scelto questa chiave siccome è il modo più semplice per gestire le PK;
- **ID_Utente**: questa è una chiave secondaria, prendo questo dato dalla tabella Utente e mi serve per collegare i dati di queste ultime;
- **Indirizzo**: l’indirizzo viene immesso dall’utente al momento dell’aggiornamento dei dati di spedizione, sezione alla quale si può effettuare l’accesso solo se Loggati e dalla pagina dell’utente, non è una chiave primaria siccome più utenti possono vivere nella stessa via;
- **Città**: anche la città viene immessa quando l’utente vuole aggiornare i propri dati di spedizione, sempre alle stesse condizioni, ovvero aver effettuato l’accesso ed entrare nella pagina utente, per lo stesso motivo di cui sopra, questo attributo non è stato scelto come PK;
- **Provincia**: la provincia è uguale agli altri due campi appena esposti, e la motivazione per la Chiave Primaria è sempre la stessa;
- **CAP**: il CAP infine viene immesso dagli utenti nell’ultimo passaggio di aggiornamento dei dati di spedizione, anche qui questo campo non è stato scelto come PK per evidenti problemi di gestione al momento in cui più utenti dello stesso CAP si siano registrati.

Compra:

- **ID**: ID è la chiave primaria di questa tabella, è un INT auto-incrementante e ho scelto questa chiave siccome è il modo più semplice per gestire le PK;
- **ID_Utente**: questa è una chiave secondaria, prendo questo dato dalla tabella Utente e mi serve per collegare i dati di queste ultime, in questo caso specifico però mi serve insieme al prossimo campo per effettuare un'associazione con i prodotti;
- **ID_Componente**: questa è una chiave secondaria, prendo questo dato dalla tabella Prodotti e mi serve per collegare i dati di queste ultime, in questo caso specifico però mi serve insieme al prossimo campo per effettuare un'associazione con gli utenti;
- **Data_Acquisto**: questo attributo carica nel database la data precisa in cui l'utente effettua un ordine, è un campo datetime quindi specifica sia data che ora dell'acquisto;
- **Importo**: consiste nell'importo che l'utente deve pagare siccome ha concluso un ordine, è un double in modo tale da consentire la presenza anche di cifre decimali.

Carrello:

- **ID**: ID è la chiave primaria di questa tabella, è un INT auto-incrementante e ho scelto questa chiave siccome è il modo più semplice per gestire le PK;
- **ID_Utente**: questa è una chiave secondaria, prendo questo dato dalla tabella Utente e mi serve per collegare i dati di queste ultime, in questo caso specifico però mi serve insieme al prossimo campo per effettuare un'associazione con i prodotti;
- **ID_Prodotto**: questa è una chiave secondaria, prendo questo dato dalla tabella Prodotti e mi serve per collegare i dati di queste ultime, in questo caso specifico però mi serve insieme al prossimo campo per effettuare un'associazione con gli utenti;
- **Quantità**: questo INT consiste nella quantità dei prodotti inseriti all'interno del carrello, non è una PK per chiari motivi.

Definizione delle Forme Normali

Dopo un'accurata **analisi** che vediamo precedentemente, possiamo identificare le seguenti **forme normali**:

- **1^a** Forma Normale -> **Utente, Prodotti**;
- **2^a** Forma Normale -> **Compra, Spedizione, Carrello**;
- **3^a** Forma Normale -> Non è presente la Terza Forma Normale in quanto non sono presenti dipendenze funzionali tra le colonne di una tabella con le varie Chiavi Primarie.

Una tabella è in **1NF** se:

- **Non** contiene attributi complessi (tranne la **data**), cioè attributi che contengono più informazioni, come ad esempio:
 - Nome;
 - Cognome;
 - Email;
 - Password.
- Non contiene attributi multipli (tipo **array**);
- Ogni riga è **diversa** da tutte le altre;
- Ogni attributo deve essere presente in **ogni** riga.

Una tabella è in **2NF** se:

- È in **1NF**;
- Ciascun attributo non chiave dipende da **TUTTA** la chiave, e non solo da una **parte**.

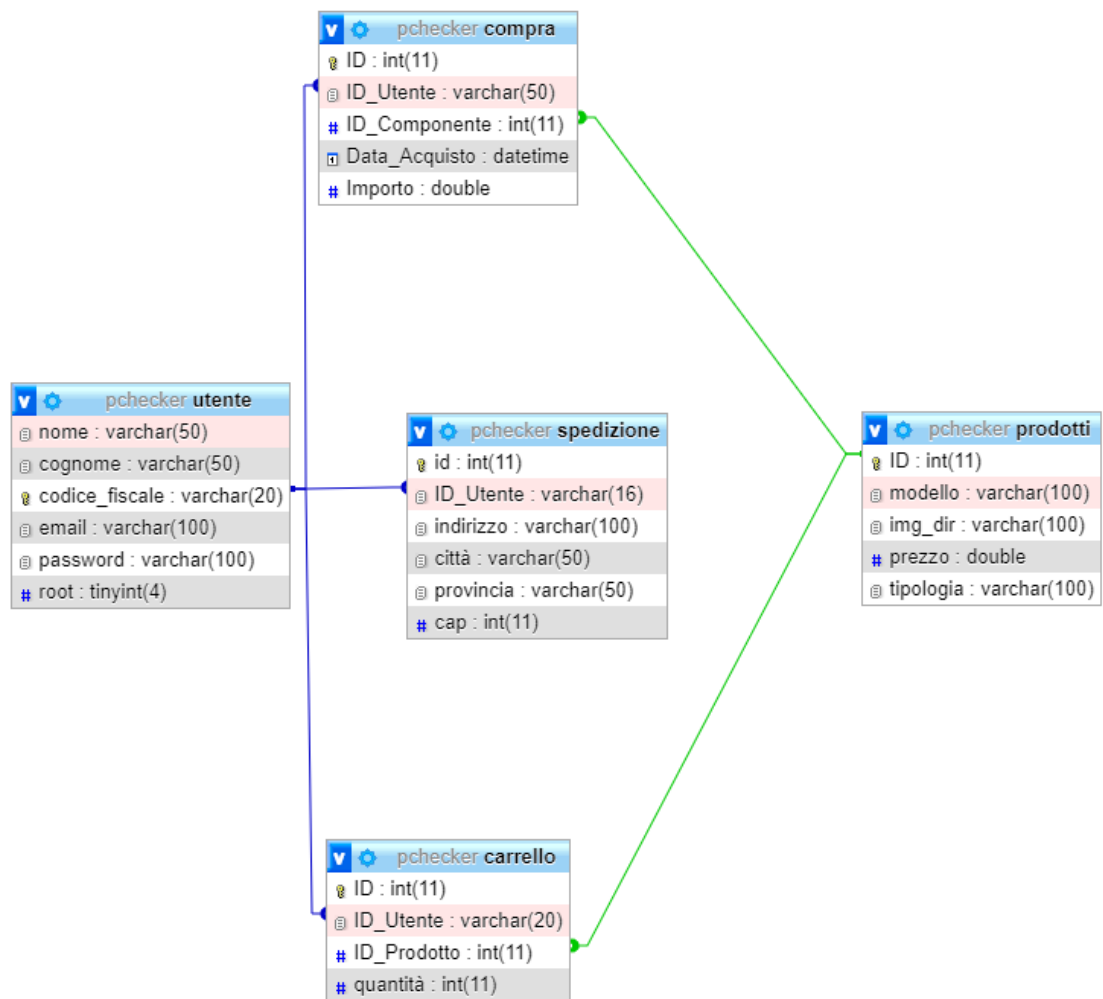
Nel caso in cui una tabella non sia in 2NF perché alcuni attributi dipendono solo dal primo campo della chiave ed altri solo dal secondo, è necessario **scomporre** la tabella in due tabelle **separate**, ognuno contenente una chiave e i campi che dipendono solo da essa

Una tabella è in **3NF** se:

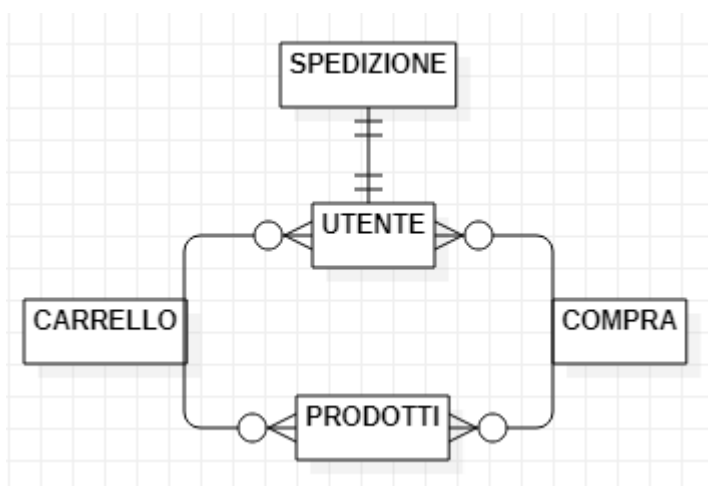
- È in **2NF**
- Ogni attributo non chiave dipende **SOLO** dalla chiave (cioè non deve esistere alcun'altra chiave candidata)

Anche in questo caso, la tabella andrebbe **scomposta** in tante tabelle quante sono le chiavi candidate, ognuna con chiave primaria **coincidente** con la chiave primaria scelta per la tabella **originaria**.

Modello Tabellare:



Modello E/R:



Variabili dei Form

Per passare i dati dai form **HTML** ai file **PHP** è ovviamente necessario l'utilizzo dei **FORM**, i quali hanno due parametri principali, il primo è il **method**, ovvero in che modo vengono trasmessi i dati al PHP. Sono presenti due metodi principali, il metodo **POST** e il metodo **GET**, il primo scrive i parametri **URL** nella richiesta **HTTP** indirizzata al server, celandoli però alla vista dell'utente. Le richieste POST non prevedono un limite massimo di grandezza. Con il metodo GET, invece, i dati che devono essere inviati al server sono scritti direttamente all'interno dell'**URL**. Tutte le informazioni fornite dall'utente, quelli che sono definiti parametri URL, sono trasmesse tanto apertamente quanto l'URL stesso. L'altro parametro è l'**action**, ovvero a quale file vengono inviati i dati, successivamente nel file PHP bisogna recuperare i dati tramite la variabile **\$_POST[""]**, la quale all'interno delle parentesi conterrà il **name** degli **input** presenti nell'HTML.

Avendo parlato degli input mi sembra corretto spiegare di cosa si tratta. Gli input sono di diversi tipi, in base al loro scopo, troviamo il **type="email"**, che verifica automaticamente la corretta formattazione della mail inserita, oppure troviamo il **type="password"**, che in automatico cela il contenuto alla vista, in modo da proteggere informazioni sensibili da occhi indiscreti. Il **name** è un attributo che applichiamo a tutti gli input in modo tale da renderli poi riconoscibili dal metodo POST nel momento in cui i dati vengono successivamente inviati al PHP.

Stili CSS

Il **CSS** è un linguaggio che personalmente ho utilizzato molto all'interno del mio sito in modo tale da rendere esteticamente più apprezzabile le diverse pagine web.

Il CSS (sigla di **Cascading Style Sheets**, in italiano fogli di stile a cascata) permette di **modificare** parti specifiche di codice utilizzando gli attributi **style**, in questo modo possiamo modificare dalla radice l'aspetto di interi tag **HTML**, oppure creare **classi** personalizzate in modo tale da affidarle a uno specifico elemento, di seguito alcuni esempi pratici:

```
body {  
  margin: 0;  
  font-family: 'Raleway', sans-serif;  
  background-color: #121212;  
}
```

Grazie a questo stile si va a modificare per intero il contenuto del tag **body** dell'HTML, in questo caso vediamo che il parametro "**margin**" è stato impostato a zero, questo vuol dire che ora all'interno delle diverse pagine non ci saranno margini dai vari bordi presenti sul sito.

Font-family invece specifica il font da utilizzare, come vediamo utilizziamo il font "**Raleway**" con formattazione **sans-serif**.

Infine cambiamo il colore dello sfondo grazie a **background-color**, impostando tramite codice **esadecimale** il colore che più preferiamo.

```
.logo{  
  width: 205px;  
  height: 75px;  
  vertical-align: middle;  
}
```

Questa invece non è una modifica a un tag HTML, bensì in questo modo **creiamo** una classe, che poi affideremo alla variabile che più preferiamo tramite l'attributo **class="nomedellaclasse"**.

```
<a class='active' style='float: right;' href='carrello.php'></a>
```

Questo è un altro modo per impostare degli stili, infatti nel caso non ci fossero **importanti** modifiche da fare, come in questo caso, è preferibile affidare un **attributo** style e inserire all'interno ciò che avremmo messo nel CSS, in questo modo non **inquineremo** maggiormente gli stili più pesanti con una cosa talmente **banale** e allo stesso tempo renderemo il codice del sito più **leggibile** e **pulito**.

Operazioni MySQL

MySQL è la piattaforma tramite la quale gestisco il mio **Database** e tutto quello che lo riguarda, quindi i dati che vengono inseriti tramite i **FORM** di cui abbiamo parlato prima e anche i dati che vediamo stampati all'interno delle varie pagine.

Come esempio di codice **PHP** con chiamata a funzione **SQL** ho pensato di presentare il codice che utilizzo per stampare i prodotti che un utente ha all'interno del proprio **carrello**, il codice è il seguente:

```
<?php
$selectCart = "select * from carrello where ID_Utente = '$fiscale'";
$resultCart = mysqli_query($conn, $selectCart);
$rowCart = mysqli_fetch_assoc($resultCart);
$countCar = mysqli_num_rows($resultCart);

for ($i = 0; $i < $countCar; $i++) {
    $idp = $rowCart['ID_Prodotto'];

    $selectProduct = "select * from prodotti where ID = '$idp'";
    $resultProduct = mysqli_query($conn, $selectProduct);
    $rowProduct = mysqli_fetch_assoc($resultProduct);

    $danaro += $rowProduct['prezzo'] * $rowCart['quantità'];

    echo "<tr>
        <td><img class='img' src='\" . $rowProduct['img_dir'] . \"'></td>
        <td>\" . $rowProduct['modello'] . \"</td>
        <td><input type='number' value='\" . $rowCart['quantità'] . \"' min='1' max='5' style='width: 25px' /></td>
        <td><p>\" . $rowProduct['prezzo'] . \",00€</p></td>
        <td><form action='rimuoviarticolo.php' method='post'><input style='width: 0; visibility: hidden;' type='text' name='id' value='\" . $rowCart['ID'] . \"'>
            <button title='Elimina Prodotto'><i class='fa fa-trash' aria-hidden='true'></i></button></form></td>
    </tr>";

    $rowCart = mysqli_fetch_assoc($resultCart);
}
?>
```

Per prima cosa apro il tag **PHP** che viene chiuso alla fine del codice come si può evincere dalla foto, successivamente decido di immagazzinare la **query** da eseguire all'interno di una variabile chiamata **\$selectCart**, questa query ha il compito di selezionare qualsiasi informazione presente all'interno della tabella **carrello** con una **condizione** però, ovvero che **ID_Cliente** deve essere uguale alla variabile **\$fiscale**, al cui interno è immagazzinato il Codice Fiscale dell'utente attualmente attivo. Successivamente eseguo la query grazie alla funzione **mysqli_query()**, alla quale passo come parametri la connessione **\$conn** e la query **\$selectCart**, in questo modo questa funzione restituirà **0** se la query contiene errori o **1** se invece è andata a buon fine. Una volta eseguita questa parte di codice inserisco all'interno di un'ulteriore variabile un **array associativo**, tramite la funzione **mysqli_fetch_assoc()** alla quale passo come parametro il risultato della query di cui sopra, il compito di questa funzione è quello di **creare**, come specificato sopra, un array associativo il quale ha come indice il campo della tabella specificata nella query, in questo modo possiamo trarne i dati in modo molto semplice. Come ultimo passaggio di questa operazione troviamo il **\$countCar**, con all'interno il risultato di **mysqli_num_rows()** che ha come parametro sempre il risultato della query, all'interno di questa variabile sarà inserito il numero di **righe** della tabella che **soddisfano** la query iniziale.

A questo punto inizia un ciclo **for** che ha all'interno un **indice** che parte da **0** e **incrementerà** finché la variabile **\$countCar** glielo consentirà.

All'interno del **for** inizializzo ancora un'altra variabile chiamata **\$idp**, al cui interno immagazzino l'**ID_Prodotto**, dato che viene prelevato come detto dalla tabella **carrello**.

Ora viene inizializzata una **seconda** query in **contemporanea** con la prima, ma in questa query si seleziona qualsiasi campo dalla tabella prodotti alla condizione che l'**ID** sia uguale alla variabile **\$idp**, in questo modo siamo sicuri di star stampando il prodotto giusto.

Come possiamo vedere ora troviamo le stesse funzioni utilizzate per la prima query, ad **eccezione** di **mysqli_num_rows()** siccome in questo momento non ci interessa.

All'interno della variabile **\$danaro** viene eseguito un calcolo molto semplice, si moltiplica il **prezzo** del prodotto, che viene prelevato dalla tabella corrispondente grazie a **\$rowProduct**, con la **quantità** presente nel carrello, la quale viene ottenuta sempre tramite l'array associativo creato precedentemente, questa variabile verrà poi stampata a fine pagina per visualizzare il totale del carrello.

A questo punto troviamo un **echo**, che serve per **stampare** del testo, in questo caso però viene utilizzato per stampare del codice **HTML**, in modo tale da visualizzarlo poi sulla pagina web. Come si evince dal codice creiamo una **table row** grazie al tag **<tr>** e creiamo diversi campi grazie ai vari **<td>**, in ordine troveremo stampati i seguenti dati:

- L'**immagine** del prodotto, tramite il campo **img_dir** della tabella prodotti, grazie alla **concatenazione** del **\$rowProduct['img_dir']**;
- Il **modello** del prodotto, ovvero il **nome**, dato che otteniamo sempre dalla stessa tabella sempre grazie all'**array associativo**;
- Il terzo campo è la **quantità**, questo valore viene preso dalla tabella **carrello** piuttosto che da quella dei prodotti siccome il **database** è stato strutturato in modo tale da contenere in questo modo le **informazioni**;
- Per ultimo valore stampato troviamo il **prezzo** del prodotto, prelevato sempre dalla tabella prodotti;
- Infine creiamo un **FORM** al cui interno inseriamo:
 - o Un campo **<input type="text">** **nascosto** dagli occhi dell'utente al cui interno inseriamo l'**id** del prodotto presente nel carrello tramite la concatenazione con **\$rowCart['ID']**;
 - o Un **bottone** di **submit** che, nel caso venisse premuto, **invierà** l'**ID** del prodotto in **POST** al file **PHP** predisposto per l'eliminazione dell'oggetto in questione e che verrà cancellato per l'appunto dal carrello dell'utente.
- Come ultimo punto vediamo la **nuova** creazione di un array associativo, questo perché serve allo **script** per andare avanti nelle righe da stampare nel momento in cui il ciclo **for** andrà avanti.

```

<?php
$conn = mysqli_connect("localhost", "mattiascotellaro", "", "my_mattiascotellaro");

function controllo($conn_info, $query)
{
    if (mysqli_query($conn_info, $query)) {
        echo "Comando eseguito con successo<br>";
    } else {
        echo mysqli_error($conn_info);
    }
}

$id = $_POST['id'];

$query = "select * FROM `carrello` WHERE `carrello`.`ID` = $id";
$result = mysqli_query($conn, $query);
$count = mysqli_num_rows($result);

if ($count == 1) {
    $query = "DELETE FROM `carrello` WHERE `carrello`.`ID` = $id";

    controllo($conn, $query);

    header("Location: carrello.php");
}

```

Questa **query** serve per l'**eliminazione** del prodotto dal **carrello**, prodotto specificato tramite la **query** che abbiamo visto **precedentemente**.

In primo luogo apro il tag **PHP** per iniziare a programmare e creo una variabile di **connessione** con il server chiamata **\$conn**, al cui interno uso la funzione **mysqli_connect()** e come **parametri** passo:

- Il nome dell'**host**;
- Il nome dell'**utente**;
- La **password**;
- Il nome del **database** al quale collegarsi.

Successivamente creo una **funzione** che serve per eseguire le query e che utilizzerò nel momento in cui il **controllo** vada a buon fine.

Ora prendo tramite il **\$_POST[""]** l'**ID** del prodotto da eliminare, lo inserisco all'interno della variabile **\$id** e la uso all'interno della query immagazzinata in **\$query**, successivamente vengono effettuate le operazioni che abbiamo già visto:

- Controlliamo se la query viene eseguita **correttamente**;
- Controlliamo in quante **righe** questa operazione si avvererà.

Viene poi aperto un **if** la cui condizione consiste nel **controllare** se il valore presente in **\$count** è **uguale** a **uno**, ovvero se ci sono elementi con quell'ID. Si effettua un'altra query, ovvero **eliminare** dal carrello l'elemento con **ID = \$id**, eseguo la funzione spiegata precedentemente che serve semplicemente per eseguire la suddetta e **rimando** immediatamente l'utente alla pagina del **carrello**, in questo modo si vedrà che l'elemento scelto è stato **eliminato** e il totale da pagare è stato **aggiornato**.

PChecker – Rete

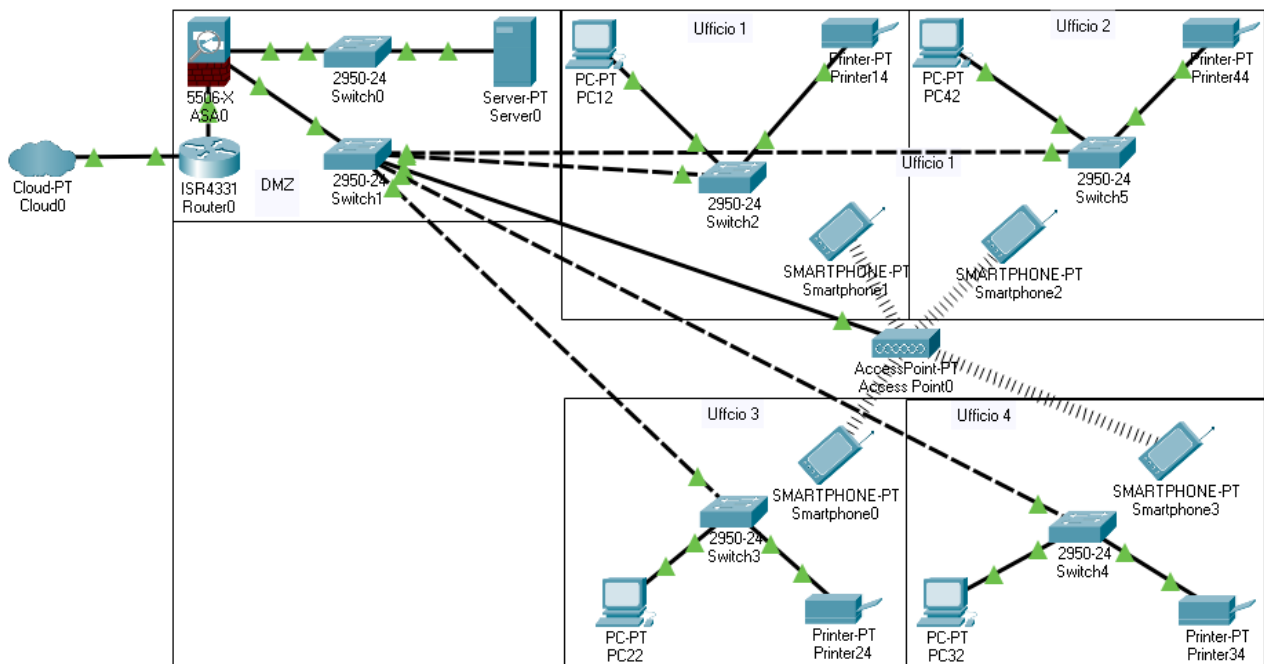


Tabella IP – Switch2

Nome	IP	Subnet Mask	Porta
PC12	192.168.0.12	255.255.255.0	0/2
Printer14	192.168.0.14	255.255.255.0	0/4

Tabella IP – Switch3

Nome	IP	Subnet Mask	Porta
PC22	192.168.0.22	255.255.255.0	0/2
Printer24	192.168.0.24	255.255.255.0	0/4

Tabella IP – Switch4

Nome	IP	Subnet Mask	Porta
PC32	192.168.0.32	255.255.255.0	0/2
Printer34	192.168.0.34	255.255.255.0	0/4

Tabella IP – Switch5

Nome	IP	Subnet Mask	Porta
PC42	192.168.0.42	255.255.255.0	0/2
Printer44	192.168.0.44	255.255.255.0	0/4

Tabella PAT

Porta di Ingresso	Porta di Uscita
80	192.168.0.69:80
21	192.168.0.69:21
3306	192.168.0.69:3306

Protocolli di Sicurezza

Per garantire la protezione del sito **PChecker** ho deciso di applicare il famoso protocollo **HTTPS** (**HyperText Transfer Protocol over Secure Socket Layer**). Non è altro che un protocollo per la comunicazione sicura attraverso una rete di computer utilizzato su **Internet**. La porta utilizzata generalmente (ma non necessariamente) è la **443**. Consiste nella comunicazione tramite il protocollo **HTTP** (**HyperText Transfer Protocol**) all'interno di una connessione criptata, tramite crittografia asimmetrica, dal Transport Layer Security (**TLS**) o dal suo predecessore, Secure Sockets Layer (**SSL**) fornendo come requisiti chiave:

- un'**autenticazione** del sito web visitato;
- protezione della **privacy** (riservatezza o confidenzialità);
- integrità dei dati scambiati tra le parti comunicanti.

Per capire come funziona **HTTPS** è necessario introdurre alcuni concetti di crittografia quali **chiavi** pubbliche e private, **firme digitali**, **certificati digitali** e **autorità di certificazione**.

Chiavi pubbliche e private

Le chiavi **pubbliche** e **private** sono strumenti utilizzati nella cosiddetta **crittografia asimmetrica** o a doppia chiave. Sono progettati per mantenere i dati **privati**, e proteggere le comunicazioni su **Internet**.

È possibile **generare** una coppia di chiavi composta da una chiave **privata** e una chiave **pubblica**.

La chiave **privata**, come suggerisce il nome, deve rimanere **segreta** e custodita in un luogo **sicuro** (nel caso di un sito web verrà custodita sul server), mentre la chiave **pubblica** è destinata a essere distribuita **pubblicamente**.

Firme digitali

Usando la chiave privata è possibile **firmare** digitalmente i dati. Funziona come nel mondo reale in cui una firma viene apposta tramite una **penna** su un foglio di **carta**, con la differenza che una firma **digitale** non può essere **falsificata**.

Dopo aver firmato i dati con la chiave privata, **chiunque** può verificare la firma usando la chiave pubblica **associata**. Se il destinatario dei dati firmati verifica correttamente la firma utilizzando la chiave pubblica ottiene due **garanzie**:

- **Autenticazione**: il destinatario è sicuro che i dati provengono dal proprietario della chiave privata associata.
- **Integrità**: il destinatario è sicuro che i dati non siano stati modificati da terzi durante il percorso.

Certificati digitali e autorità di certificazione

Lo **scambio** di chiavi pubbliche e la firma digitale introducono un nuovo **problema** da risolvere.

Se ad esempio vogliamo comunicare in modo **sicuro** con Google, abbiamo bisogno della sua chiave **pubblica** e lo stesso vale per **altri** siti che vogliamo **visitare**.

Dato che esistono miliardi di siti web su Internet, come possiamo ottenere una chiave pubblica per **ogni** sito web che vogliamo visitare?

Ed è qui che entrano in scena le autorità di certificazione o **CA**.

Una CA è un'organizzazione di terze parti con 3 **obiettivi** principali:

- Rilasciare **certificati digitali**.
- **Confermare** l'identità del proprietario del certificato.
- Fornire la prova che il certificato è **valido**.

Nello scenario di **HTTPS** la chiave pubblica è rappresentata da un certificato digitale, più noto come certificato **SSL/TLS**.

Ora vediamo in che modo è possibile ottenere un certificato SSL/TLS firmato da una CA:

- Il proprietario del sito web **genera** una chiave **pubblica** e una chiave **privata** ed invia un file di richiesta di firma del certificato (**CSR**) e la sua chiave pubblica alla CA.
- La CA crea quindi un certificato personale basato sulla **CSR**, dove sono indicati nome di dominio, nome del proprietario, data di scadenza e altre informazioni, appone la firma digitale, infine crittografa l'intero certificato con la chiave pubblica del server e lo rimanda al proprietario del sito web.
- Il certificato viene quindi **decifrato** con la chiave privata del proprietario del sito web e, infine, viene **installato** sul server.

N.B. La firma digitale della CA è **crittografata** dalla chiave **privata** della CA e può essere decifrata **solamente** con la chiave **pubblica** della CA, chiamiamo questa chiave Certificato Radice (**Root Certificate**).

Ogni dispositivo (pc, smartphone) ha, installato nel browser, un elenco di certificati radice di molte CA fidate.

PChecker – Analisi di Mercato

PChecker sarà un'azienda la cui sede verrà stabilita nel comune di **Napoli**, in **Campania**, di conseguenza è giusto effettuare una precisa **analisi di mercato** che seguirà i seguenti **punti**:

- Analisi Demografica;
- Dimensioni del Mercato:
 - Analisi Top Down;
 - Analisi Bottom Up;
- Mercato di Riferimento;
- Bisogni del target di mercato;
- Concorrenza;
- Barriere all'entrata;
- Regolamentazione.

Iniziamo quindi con l'**Analisi Demografica**:

Nel comune di Napoli la popolazione residente, costituita dalle persone aventi dimora abituale, ammonta a **962.003** unità. Il peso della popolazione maschile risulta essere **minore** rispetto a quello femminile: il rapporto di **mascolinità** (rapporto percentuale avente a numeratore la popolazione maschile residente e a denominatore la popolazione femminile residente) è pari a **90,15** uomini ogni **100** donne, inferiore al dato nazionale (93,67 uomini ogni 100 donne).

Si rileva una presenza relativa di bambini **superiore** alla media nazionale. In particolare la percentuale dei bambini con **meno** di cinque anni è pari a 4,86%, superiore al **4,63%** registrato a livello nazionale.

A **differenza** della situazione nazionale, dai dati definitivi del Censimento 2001 sulla struttura demografica della popolazione, emerge un Comune demograficamente **meno** anziano. Un ulteriore indicatore, con rilevanza economica e sociale, è l'**indice di dipendenza**, o anche detto **indice demografico di dipendenza**, con il quale le persone che in via presuntiva non sono autonome per ragioni demografiche (**l'età**) - e cioè gli anziani e i giovanissimi - e che perciò sono **dipendenti**, sono poste in rapporto alle persone che si presume debbano sostenerli con la loro **attività**. Nel comune di Napoli l'indice, pari al 50,70%, è **inferiore** a quello nazionale (in Italia 53,49%).

Un notevole **interesse** demografico, soprattutto per i suoi riflessi sul movimento della popolazione, ha la composizione della popolazione secondo lo **stato civile**.

Per i cittadini stranieri la composizione per genere mostra uno sbilanciamento a favore delle **donne** (59,77% di donne). Il 41,09% degli stranieri censiti proviene dai paesi **asiatici**, ed il 40,35% dall'**Europa**.

Ora bisogna definire le Dimensioni del Mercato, però prima di effettuare questa operazione c'è bisogno di parlare dei due **fattori** più importanti:

- Volume;
- Valore.

Il **Volume** consiste nel numero di potenziali clienti, nel mio caso sono i privati, quindi una qualsiasi persona può entrare all'interno del negozio ed effettuare l'acquisto del prodotto tanto desiderato.

Il **Valore** invece è possibile stimarlo tramite due diversi approcci:

- **Top Down**: ovvero si parte da una visione globale, scomponendola e dettagliando via via nel particolare;
- **Bottom Up**: si inizia specificando nel dettaglio ogni singola componente di un sistema, connettendole tra loro in **macro-componenti**, che interconnesse a loro volta daranno una visione del sistema completo.

Per il mio progetto è stata scelta la **seconda** modalità, ovvero la Bottom Up. Partendo per esempio dall'ambito dei processori, possiamo calcolare un prezzo **medio** di 200€. Conosciamo due dati molto importanti, ovvero che nella sola zona di Napoli sono presenti, approssimando per **difetto**, circa 962 mila di persone e che di solito un processore si cambia una volta ogni **anno**. Per effettuare una stima più precisa però, è giusto specificare che solitamente è necessario un solo processore, quindi computer, per **famiglia**, di conseguenza dividiamo la popolazione (962.000) per il numero **medio** di persone che compongono una famiglia (3) e otteniamo all'incirca 320.000 **famiglie**. Possiamo quindi dire che avremo come volume annuale delle transazioni circa 320 mila di esse, moltiplicando questo valore poi per il costo medio di una **CPU** otteniamo come valore di **mercato** circa 64 milioni di euro.

Per quanto riguarda il **mercato di riferimento** ho pensato di optare principalmente per tutte quelle persone che tendono ad utilizzare il computer per un motivo professionale, quale lavorare o videogiocare. Di conseguenza come **driver** di domanda ho scelto la completezza del catalogo, andandomi a posizionare così nella fascia **alta** di questo settore.

Parlando dei **bisogni del target di mercato** troviamo la particolare esigenza che ogni singolo utente può richiedere, ad esempio un videogiocatore avrà bisogno di una GPU molto **potente**, mentre un grafico richiederà un monitor dai colori **perfetti**, di

conseguenza PChecker metterà a disposizione **qualsiasi** prodotto anche tramite **richiesta** del singolo.

Come per tutte le attività la **concorrenza** esiste e si fa sentire, soprattutto nell'ambito hardware in questo periodo a causa della grave carenza di componentistica, principalmente per quanto riguarda le **GPU**. PChecker però si accerterà di stabilire contatti con **aziende** produttrici, quali **NVIDIA**, **AMD** e **intel**, in modo da assicurare ai propri utenti la **certezza** di poter acquistare ciò che desiderano. Un'altra caratteristica presente è la possibilità di richiedere assistenza **post-acquisto**, quindi per qualsiasi problema o richiesta di assistenza tecnica si potrà contattare l'**Amministratore** specificando il numero d'ordine in modo da richiedere tutto ciò che si vuole.

A seguito di un'approfondita ricerca si evince che per aprire un negozio di **informatica**, nella migliore delle ipotesi, richiederà un investimento iniziale di almeno **30.000/40.000** euro.

Con 30.000 euro di investimento iniziale è di fatto possibile far fronte ai costi iniziali **minimi** per un locale di piccole dimensioni (anche soli 50mq, purché allestito in modo ordinato ed efficiente), per l'arredamento e l'allestimento, per la creazione di un sito web con possibilità di acquistare anche **online** o prenotare riparazioni ed assistenza anche a domicilio, per l'advertising online e offline, per far fronte al primo **rifornimento** di merce nonché per adempiere agli oneri ed all'iter **burocratico**.

Ovviamente, **maggiore** sarà la dimensione del locale e l'ampiezza della gamma di prodotti venduti, e dei servizi offerti, maggiore sarà l'**investimento** necessario per aprire un negozio di **informatica** ed **elettronica** efficiente.

Per quanto riguarda i **ricavi**, il margine di guadagno su computer ed elettronica è davvero basso, attestandosi sul **10%**. Di conseguenza è intuibile che per aprire un negozio di informatica redditizio sia necessario puntare su **servizi** quali riparazioni ed assistenza, anche a domicilio, vendita di ricambi ed accessori nonché di prodotti complementari su cui il margine di guadagno sia maggiore, fattori sui quali PChecker conta **molto**.