Gestione salone

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

2 Analisi 4

2.1 Analisi del dominio 4

2.2 Analisi e specifica dei requisiti 4

2.3 Analisi dei mezzi 6

2.3.1 Software 6

2.3.2 Hardware 6

3 Progettazione 7

3.1 Design dell’architettura del sistema 7

3.1.1 Model View Controller 7

7

3.2 Design del database 8

4 Implementazione 9

4.1 Implementazione database 9

4.2 Implementazione applicativo web 10

5 Test 15

5.1 Protocollo di test 15

5.2 Risultati test 18

6 Conclusioni 24

6.1 Sviluppi futuri 24

6.2 Considerazioni personali 24

7 Glossario 25

8 Sitografia 26

# Introduzione

## Informazioni sul progetto

**Titolo**: Gestione salone

**Allievi coinvolti nel progetto**:

* Andrea Curti, [andrea.curti@samtrevano.ch](mailto:andrea.curti@samtrevano.ch)
* Mattia Pasquini, [mattia.pasquini@samtrevano.ch](mailto:mattia.pasquini@samtrevano.ch)

**Classe**: I4AA Scuola Arti e Mestieri Trevano, Informatica

**Formatore**: Massimo Sartori, [massimo.sartori@edu.ti.ch](mailto:massimo.sartori@edu.ti.ch)

**Data** **inizio**: 05.10.2022

**Data** **fine**: 21.12.2022

## Abstract

All’inizio di questo progetto c’è stato chiesto di realizzare un applicativo web che permettesse di facilitare la gestione dei clienti di un salone di parrucchiere. Con questo prodotto è possibile gestire i servizi effettuati da un cliente con i loro relativi acquisti dei prodotti. I pagamenti sono suddivisi in tipologie, come contanti, bonifico…

# Analisi

## Analisi del dominio

Questo prodotto verrà utilizzato dalle commesse del salone.

Serve per gestire e visualizzare i propri clienti con i relativi servizi effettuati e i prodotti comprati.

L’applicativo è semplice da utilizzare con un’interfaccia grafica intuitiva.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| ID: REQ-001 | |
| Nome | Creazione database |
| Priorità | 1 |
| Versione | 1.0 |
| Note | Si necessitano i permessi di root. |
| Sotto requisiti | |
| 001 | Installazione MySQL. |

|  |  |
| --- | --- |
| ID: REQ-002 | |
| Nome | Pagina di login |
| Priorità | 1 |
| Versione | 1.0 |
| Note | Si necessita di una pagina dove effettuare il login. |
| Sotto requisiti | |
| 001 | Si necessita una maschera di login. |
| 002 | Si necessita un bottone di conferma. |

|  |  |
| --- | --- |
| ID: REQ-003 | |
| Nome | Pagine gestione clienti |
| Priorità | 1 |
| Versione | 1.0 |
| Note | Si necessitano delle pagine per la gestione dei clienti. |
| Sotto requisiti | |
| 001 | Si necessita una maschera di login. |
| 002 | Si necessita una pagina per l’aggiunta di un cliente. |
| 003 | Si necessita una pagina per l’eliminazione di un cliente. |
| 004 | Si necessita una pagina per la modifica di un cliente. |

|  |  |
| --- | --- |
| ID: REQ-004 | |
| Nome | Pagine gestione servizi |
| Priorità | 1 |
| Versione | 1.0 |
| Note | Si necessitano delle pagine per la gestione dei servizi. |
| Sotto requisiti | |
| 001 | Si necessita una maschera di login. |
| 002 | Si necessita una pagina per l’aggiunta di un servizio. |
| 003 | Si necessita una pagina per l’eliminazione di un servizio. |
| 004 | Si necessita una pagina per la modifica di un servizio. |

|  |  |
| --- | --- |
| ID: REQ-005 | |
| Nome | Pagine gestione prodotti |
| Priorità | 1 |
| Versione | 1.0 |
| Note | Si necessitano delle pagine per la gestione dei prodotti. |
| Sotto requisiti | |
| 001 | Si necessita una maschera di login. |
| 002 | Si necessita una pagina per l’aggiunta di un prodotto. |
| 003 | Si necessita una pagina per l’eliminazione di un prodotto. |
| 004 | Si necessita una pagina per la modifica di un prodotto. |

## Analisi dei mezzi

### Software

Programmi e software:

* Apache 2.4
* MySQL 8
* PHP 8.1.6
* HTML 5
* CSS 3
* PhpStorm 2022.1.1

### Hardware

Macchina di sviluppo:

* Intel Core I7-7700 3.60 GHz
* 16GB RAM

Macchina di produzione:

* Host Infomaniak fornito dalla scuola

# Progettazione

## Design dell’architettura del sistema

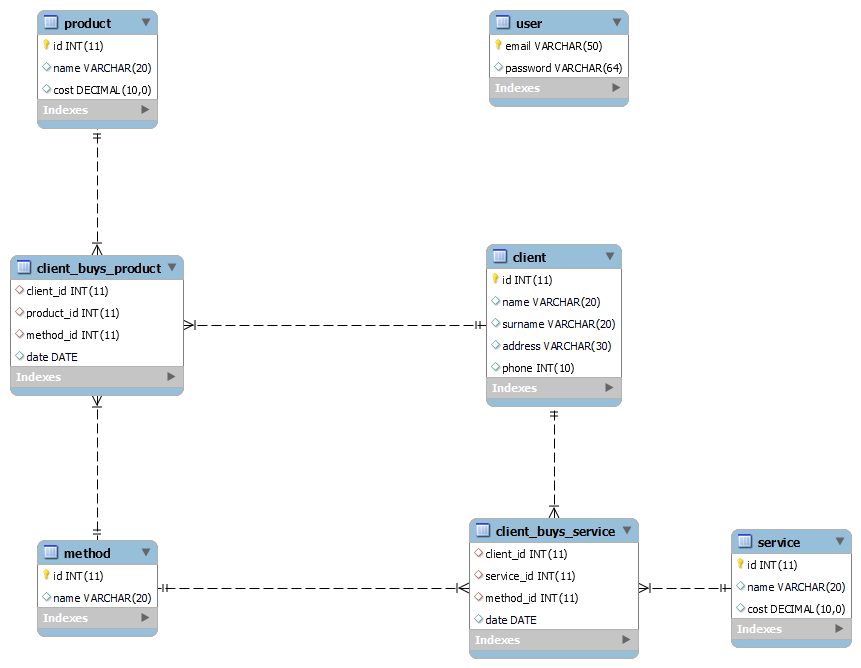
### Model View Controller

Model View Controller (MVC) è una struttura che permette di gestire al meglio tutti i file php. Viene utilizzato nell’ambito web con il linguaggio di scripting PHP. In ogni cartella vengono salvati dei file con compiti differenti.

* **Model:** in questa cartella vengono messe tutte le classi che elaborano i dati e che comunicano con il database
* **View:** in questa cartella vengono messi tutti i file contenenti script html. Non viene messo codice php. Si può usare php in questi file solo per stampare file, quindi è concesso l’utilizzo di “for” e semplici “if”.
* **Controller:** in questa cartella vengono messi tutte le classi che ricevono le richieste da parte del browser. Queste classi elaborano le richieste, prendono i dati necessari dal database grazie ai file nella cartella “model”. Dopo passano gli eventuali dati ai file nella cartella “view” e mandando al browser il file giusto dalla cartella “view”.

### Divisione dei ruoli in MVC

## Design del database



# Implementazione

## Implementazione database

I file per la creazione e popolazione del database sono su GitHub nella cartella 4\_Database.

C’è il file di configurazione del database “script.sql” che contiene tutte le istruzioni per creare il database correttamente. Ci pensa lo script ad eliminare il database se è già esistente e poi ricrearlo.

Poi c’è lo script “dati.sql” che contiene tutte le istruzioni per popolare il database.

## Implementazione applicativo web

**config.php**

|  |
| --- |
| $actual\_link = (isset($\_SERVER['HTTPS']) && $\_SERVER['HTTPS'] === 'on' ? "https" : "http") . "://$\_SERVER[HTTP\_HOST]";  $documentRoot = $\_SERVER['DOCUMENT\_ROOT'];  $dir = str\_replace('\\','/',getcwd().'/');  $final = $actual\_link.str\_replace($documentRoot,'',$dir);  define('URL', $final); |

Questo codice serve per trovare il percorso host del sito, così da poter salvare il tutto in una costante “URL” e utilizzarla in tutte le classi per caricare o richiamarne un’altra.

Come prima cosa viene controllato se è https o http, poi viene tornata una stringa $final che contiene il protocollo dell’applicativo con la directory del progetto.

Il codice è stato scritto da Massimo Sartori.

**antiCsScript.php**

|  |
| --- |
| public static function check($data) {      $data = trim($data);      $data = stripslashes($data);      $data = htmlspecialchars($data);      return $data;  } |

Il metodo “check” server per “sanificare” una stringa, rimuovendo gli spazi vuoti all’inizio e alla fine, rimuove caratteri speciali e richiama il metodo per controllare che non contenga parti di script. Questo metodo serve per evitare anche Cross Side Scripts.

**controller.php**

|  |
| --- |
| public function locate($path){      header("Location: " . URL . $path);  } |

Il metodo “locate” serve per caricare la pagina index del controller passato come argomento.

**database.php**

|  |
| --- |
| public static function getConnection(){      if(self::$connection != ""){          return self::$connection;      }      self::$connection = new mysqli(self::$hostname, self::$username, self::$password, self::$dbname);      if(self::$connection->connect\_error){          die("Connection failed: ".self::$connection->connect\_error);      }      return self::$connection;  } |

Questa classe viene richiamata da ogni metodo che deve interrogare o comunicare col database;

infatti, stabilisce una connessione con il database, effettuando l’accesso.

**email.php**

|  |
| --- |
| public static function isValidClient($email){      if(self::checkFilters($email)){          if(self::checkEmailsInDBClient($email)){              return true;          }else{              throw new Exception("Email già esistente");          }      }else{          throw new Exception("Email non valida");      }  } |

Questo metodo serve per controllare se la nuova email di un cliente che verrà modificata è identica a quella precedente oppure è diversa.

**password.php**

|  |
| --- |
| public static function isValid($pass){      $uppercase = preg\_match('@[A-Z]@', $pass);      $lowercase = preg\_match('@[a-z]@', $pass);      $number = preg\_match('@[0-9]@', $pass);      $specialChars = preg\_match('@[^\w]@', $pass);      if(!$uppercase || !$lowercase || !$number          || !$specialChars || strlen($pass) < 8) {              return false;      }      return true;  } |

Questo metodo serve per controllare che la password abbia tutti i criteri di sicurezza.

|  |
| --- |
| public static function doHash($plain){      return hash('sha256', $plain);  } |

Serve per calcolare l’hash in sha256 della password

**view.php**

|  |
| --- |
| public function render($name, $onlyIncludeBody = false, $data = array()){      require 'application/views/links.php';      if($onlyIncludeBody){          require "application/views/" . $name;      }else{          if(isset($\_SESSION['email'])){              require 'application/views/header.php';          }          require "application/views/" . $name;          require 'application/views/footer.php';      }  } |

La funzione “render” serve per caricare una view col nome passato come argomento. C’è la possibilità di scegliere se includere l’header ed il footer, in più anche la possibilità di inserire dei dati tramite l’array “data”.

**clients.php**

|  |
| --- |
| public function index()  {      if(isset($\_SESSION['email'])){          $this->view->render('clients/index.php');      }else{          $this->view->render('login/index.php');      }  } |

La funzione “index” è presente in tutte le classi controller, e serve per caricare la pagina index del relativo controller. Ovviamente vengono effettuati i controlli sui permessi dell’utente, che deve aver effettuato prima l’accesso per poter accedere a questa pagina.

|  |
| --- |
| public function getSpecificClientInfo($id){      require\_once 'application/models/client\_model.php';      try {          $shop = ClientClass::getSingleClientInfos(intval($id));      } catch (Exception $e) {          $shop = $e->getMessage();      }      echo json\_encode($shop);  } |

Questa funzione viene richiamata attraverso Javascript dalle pagine HTML. Ritorna in formato JSON le informazioni di un cliente specifico, trovato tramite il su id.

**products.php**

|  |
| --- |
| public function addProduct(){      if(isset($\_SESSION['email'])){          require\_once 'application/models/product\_model.php';          if($\_SERVER["REQUEST\_METHOD"] == "POST"){              try{                  ProductClass::addProduct();                  $this->locate('products/index');              }catch(Exception $e){                  $this->view->render('products/addProduct.php',                      false, array('error' => $e->getMessage(),                          'lastName' => $\_POST["name"],                          'lastCost' => $\_POST["cost"]));              }          }      }else{          $this->view->render('login/index.php');      }  } |

Quando l’utente preme il bottone per aggiungere un nuovo prodotto, viene richiamata questa funzione. Passati i controlli di sicurezza, richiama il metodo del model, il quale lavora direttamente sul database. Se l’aggiunta va a buon fine, viene caricata la pagina index, altrimenti viene ricaricata la pagina di aggiunta con l’errore.

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-001 | **Nome:** | Database funzionante |
| **Descrizione:** | Si controlla se è possibile il collegamento al database | | |
| **Prerequisiti:** | * Avere apache funzionante sulla porta 80 * Avere php funzionante * Avere mysql installato | | |
| **Procedura:** | 1. Visitare localhost 2. Premere invio | | |
| **Risultati attesi:** | Si apre una pagina con scritto “Connected successfully” | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-002  REQ-002 | **Nome:** | Pagina di login funzionante |
| **Descrizione:** | Si controlla se il login viene effettuato correttamente per un amministratore | | |
| **Prerequisiti:** | * Avere il database funzionante * Avere un utente amministratore | | |
| **Procedura:** | 1. Aprire la pagina per il login 2. Inserire il nome utente e le credenziali dell’amministratore 3. Premere il tasto login | | |
| **Risultati attesi:** | Viene mostrata la pagina home dell’amministratore | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-003  REQ-003 | **Nome:** | Creare un cliente |
| **Descrizione:** | Si controlla che vengano creati clienti correttamente | | |
| **Prerequisiti:** | * Fare login con un utente | | |
| **Procedura:** | 1. Navigare fino alla pagina per aggiungere un cliente 2. Inserire i vari dati 3. Premere il bottone per aggiungere il cliente | | |
| **Risultati attesi:** | Si riesce a vedere il nuovo cliente nella lista di clienti | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-004  REQ-003 | **Nome:** | Modificare un cliente |
| **Descrizione:** | Si controlla che vengano modificati clienti correttamente | | |
| **Prerequisiti:** | * Fare login con un utente | | |
| **Procedura:** | 1. Navigare fino alla pagina per modificare un cliente 2. Inserire i vari dati 3. Premere il bottone per modificare il cliente | | |
| **Risultati attesi:** | Si riesce a vedere il cliente con le nuove modifiche apportate | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-005  REQ-003 | **Nome:** | Eliminare un cliente |
| **Descrizione:** | Si controlla che vengano eliminati clienti correttamente | | |
| **Prerequisiti:** | * Fare login con un utente | | |
| **Procedura:** | 1. Navigare fino alla pagina per eliminare un cliente 2. Scegliere il cliente da eliminare 3. Premere il bottone per eliminare il cliente | | |
| **Risultati attesi:** | Il cliente non è più presente tra la lista di clienti disponibili | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-006  REQ-004 | **Nome:** | Aggiungere un servizio |
| **Descrizione:** | Si controlla che vengano aggiunti servizi correttamente | | |
| **Prerequisiti:** | * Fare login con un utente | | |
| **Procedura:** | 1. Navigare fino alla pagina per aggiungere un servizio 2. Inserire i vari dati 3. Premere il bottone per aggiungere il servizio | | |
| **Risultati attesi:** | Il nuovo servizio è presente nella lista | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-007  REQ-004 | **Nome:** | Modificare un servizio |
| **Descrizione:** | Si controlla che vengano modificati servizi correttamente | | |
| **Prerequisiti:** | * Fare login con un utente | | |
| **Procedura:** | 1. Navigare fino alla pagina per modificare un servizio 2. Inserire i vari dati 3. Premere il bottone per modificare il servizio | | |
| **Risultati attesi:** | Il servizio ha le informazioni appena modificare nella lista | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-008  REQ-004 | **Nome:** | Eliminare un servizio |
| **Descrizione:** | Si controlla che vengano eliminati servizi correttamente | | |
| **Prerequisiti:** | * Fare login con un utente | | |
| **Procedura:** | 1. Navigare fino alla pagina per eliminare un servizio 2. Inserire i vari dati 3. Premere il bottone per eliminare il servizio | | |
| **Risultati attesi:** | Il servizio non è più presente nella lista | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-009  REQ-005 | **Nome:** | Aggiungere l’acquisto del prodotto |
| **Descrizione:** | Acquistare un prodotto effettuato da un cliente | | |
| **Prerequisiti:** | * Fare login con un utente | | |
| **Procedura:** | 1. Navigare fino all’acquisto del prodotto 2. Selezionare il prodotto e il metodo di pagamento 3. Aggiungere il pagamento effettuato | | |
| **Risultati attesi:** | Acquisto visibile nella lista dei prodotto acquistati dal cliente | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-010  REQ-004 | **Nome:** | Aggiungere l’acquisto di un servizio |
| **Descrizione:** | Acquistare un servizio effettuato da un cliente | | |
| **Prerequisiti:** | * Fare login con un utente | | |
| **Procedura:** | 1. Navigare fino all’acquisto del servizio 2. Selezionare il prodotto e il metodo di pagamento 3. Aggiungere il pagamento effettuato | | |
| **Risultati attesi:** | Acquisto visibile nella lista dei servizi acquistati dal cliente | | |

## Risultati test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case** | **Esito** | **Risultati** | **Data** |
| TC-001 | **PASSATO** | La pagina mostra che il database è connesso | 02.02.2023 |
| TC-002 | **PASSATO** | Dopo aver effettuato il login come amministratore, viene mostrata la pagina home | 02.02.2023 |
| TC-003 | **PASSATO** | Si riesce ad aggiungere correttamente il cliente | 02.02.2023 |
| TC-004 | **PASSATO** | Quando si naviga nuovamente alla pagina di modifica cliente, si vede che le informazioni sono state modificate correttamente. | 02.02.2023 |
| TC-005 | **PASSATO** | Viene eliminato correttamente l’utente selezionato | 02.02.2023 |
| TC-006 | **PASSATO** | Un nuovo servizio viene creato correttamente | 02.02.2023 |
| TC-007 | **PASSATO** | Vengono salvate le informazioni modificate correttamente | 02.02.2023 |
| TC-008 | **PASSATO** | Viene eliminato correttamente il servizio selezionato. | 02.02.2023 |
| TC-009 | **PASSATO** | Immettere il nome del prodotto e il metodo di pagamento    Viene salvato l’aggiunta di un prodotto acquistato di un cliente selezionato. | 02.02.2023 |
| TC-010 | **PASSATO** | Immettere il nome del servizio e il metodo di pagamento    Viene salvato l’aggiunta di un servizio effettuato di un cliente selezionato. | 02.02.2023 |

# Conclusioni

## Sviluppi futuri

Uno sviluppo futuro potrebbe essere sicuramente l’aggiunta delle statistiche, per conoscere quali prodotti o servizi sono acquistati maggiormente.

Inoltre sarebbe utile aggiungere un sistema di email automatico, che spedisce ai clienti un’email di notifica ogni qualvolta viene acquistato un prodotto o servizio, come ringraziamento.

Poi si potrebbe aggiornare la grafica, magari renderla ancora più pulita e carina, ma non è una cosa relativamente importante.

## Considerazioni personali

**Andrea**:

Questo progetto mi ha aiutato a migliorare il metodo di lavorare in coppia, poiché è stato necessario organizzarsi per suddividere i vari compiti, e inoltre per migliorare l’abilità nell’utilizzare GitHub in un repository in comune dove entrambi potevamo effettuare modifiche.

Siamo riusciti a rispettare le varie specifiche del mandante. Non ho avuto molte difficoltà nello sviluppo dell’applicativo, dato che ho già realizzato in passato vari progetto di applicativi web.

**Mattia**:

Questo progetto mi è stato molto utile per capire come si lavora in gruppo: abbiamo suddiviso i vari compiti, e per far ciò è stata necessaria una buona comunicazione tra entrambi.

Fortunatamente avevamo già implementato in passato degli applicativi web, così abbiamo potuto riciclare una piccola parte di codice che ci ha permesso di risparmiare tempo.

Sono riuscito a migliorare le mie abilità organizzative: dividere i vari compiti da svolgere così da evitare perdite di tempo.

# Glossario

|  |  |
| --- | --- |
| **Nome** | **Definizione** |
| **Abstract** | Breve riassunto che spiega il progetto |
| **Apache** | Un web server open source utilizzato per fornire contenuti al browser |
| **Applicativo WEB** | Il sito web che verrà visualizzato a progetto terminato |
| **Classe** | Per i linguaggi orientati agli oggetti, è un record a cui sono associati attributi e/o metodi |
| **Cliente** | La persona o azienda che finanzia il progetto e che detta le specifiche dei requisiti |
| **Controller** | È una classe che collega la parte grafica con la parte di modifica dei dati di un database. |
| **Database** | Struttura preparata su tabelle che permette lo storage di informazioni riguardanti un applicativo o progetto e rende possibile il reperimento dei dati voluti. |
| **Date** | Tipo di dato che salva unicamente la data in SQL |
| **Gantt** | Diagramma per pianificare un progetto dividendolo in varie attività, per avere un lavoro più organizzato |
| **Interfaccia** | L'hardware e il software che permettono a un utente di interagire con il computer. |
| **Libreria** | Insieme di funzioni o strutture dati predefinite per essere collegate ad un software. Questo evita allo sviluppatore di dover riscrivere ogni volta le stesse funzioni o strutture di dati. |
| **Model** | Le classi che interagiscono col database |
| **MVC** | Pattern utilizzato per dividere le funzionalità ben distinte di un applicativo |
| **MySQL** | Sistema open source per la gestione dei database |
| **Open source** | Un software open source è libero per tutti, ovvero può essere utilizzato, modificato o distribuito da chiunque |
| **Pagina home** | Pagina iniziale di un sito Web, nella quale sono contenuti i collegamenti alle altre pagine del sito |
| **PHP** | Linguaggio di programmazione lato Server per la gestione degli applicativi  WEB |
| **Sito internet** | Sistema informatico che mette a disposizione informazioni e servizi mediante la rete Internet |
| **Specifiche dei requisiti** | Descrizione completa del comportamento di un sistema software da sviluppare |
| **Use case** | È un metodo per raffigurare i requisiti raccolti, mostrando le possibili interazioni tra gli utenti ed il sistema |
| **View** | Pagine html che comunicano con i Controller |

# Sitografia

* <https://mdbootstrap.com/> , 15.12.2022
* <https://www.daterangepicker.com/#google_vignette>, 26.12.2022
* <https://www.php.net/>, 01.01.2023
* <https://stackoverflow.com/>, 15.12.2022
* <https://github.com/> 05.12.2022
* <https://guidaphp.it/glossario-php>, 02.01.2023