Università degli studi di Salerno

Dipartimento di Informatica



Corso di Laurea in Informatica

Tesi di Laurea in

Informatica

**Progettazione e sviluppo di una Web App per vendere servizi e interazione con un sistema di pagamento**

Candidato:

Manlio Santonastaso **Matr:** 0512105404

Relatore:

**Prof.** Roberto De Prisco

Anno Accademico 2019/2020

**Indice**

[**Introduzione** 9](#_Toc57041445)

[**1.Sito Web** 11](#_Toc57041446)

[**1.1 Cos’è un sito web?** 11](#_Toc57041447)

[**1.2 Evoluzione del web** 11](#_Toc57041448)

[**1.2.1 Il web 1.0** 11](#_Toc57041449)

[**1.2.2 Il web 2.0** 12](#_Toc57041450)

[**1.2.3 Il web 3.0** 12](#_Toc57041451)

[**1.2.4 Il web 4.0** 13](#_Toc57041452)

[**1.3 Importanza di avere un sito web** 14](#_Toc57041453)

[**2. Supporto tecnologico alla realizzazione del sito** 15](#_Toc57041454)

[**2.1 Web Server** 15](#_Toc57041455)

[**2.1.1 Cos'è un server e cos'è un client?** 15](#_Toc57041456)

[**2.1.2 Funzionamento di un Web Server** 16](#_Toc57041457)

[**2.1.3 La scelta del Web Server** 16](#_Toc57041458)

[**2.2 DBMS** 17](#_Toc57041459)

[**2.2.1 MySQL** 17](#_Toc57041460)

[**2.2.2 Compiti, funzioni e requisiti di un sistema di gestione di base di dati** 18](#_Toc57041461)

[**2.2.3 Tipi di DBMS** 19](#_Toc57041462)

[**2.2.4 Alternative** 21](#_Toc57041463)

[**2.3 I Framework** 21](#_Toc57041464)

[**2.3.1 Bootstrap** 21](#_Toc57041465)

[**2.3.2 I vantaggi dei framework** 22](#_Toc57041466)

[**2.3.3 Gli svantaggi dei framework** 23](#_Toc57041467)

[**2.3.4 Framework front-end più utilizzati** 23](#_Toc57041468)

[**2.4 Linguaggi di programmazione** 24](#_Toc57041469)

[**2.4.1 HTML** 24](#_Toc57041470)

[**2.4.2 CSS** 25](#_Toc57041471)

[**2.4.3 JavaScript, JQuery e AJAX** 26](#_Toc57041472)

[**2.4.4 Java Enterprise Edition** 27](#_Toc57041473)

[**2.5 API Application Programming Interface** 28](#_Toc57041474)

[**2.5.1 API PayPal** 28](#_Toc57041475)

[**2.5.2 Perché utilizzare l’API** 28](#_Toc57041476)

[**2.5.2 Esempi di API** 29](#_Toc57041477)

[**3. Realizzazione del Progetto** 31](#_Toc57041478)

[**3.1 DBMS** 31](#_Toc57041479)

[**3.1.1 Schema Entità-Relazione** 31](#_Toc57041480)

[**3.1.2 Modello Relazionale** 32](#_Toc57041481)

[**3.1.3 Script Tabelle** 33](#_Toc57041482)

[**3.2 Back-End** 36](#_Toc57041483)

[**3.2.1 le funzioni del back-end** 36](#_Toc57041484)

[**3.2.2 Struttura: Architettura a 3 livelli** 36](#_Toc57041485)

[**3.2.3 Packages** 38](#_Toc57041486)

[**3.3 Front-End** 38](#_Toc57041487)

[**3.3.1 Home Page** 38](#_Toc57041488)

[**3.3.2 Multiple Step Form** 43](#_Toc57041489)

[**3.3.3 Carousel-Card** 47](#_Toc57041490)

[**4. Integrare il pagamento PayPal nella web app** 50](#_Toc57041491)

[**4.1 Tipico Flusso di lavoro Checkout-Payment** 50](#_Toc57041492)

[**4.2 Passi iniziale per usare l’API** 51](#_Toc57041493)

[**4.3 Come Integrare L’API di PayPal** 52](#_Toc57041494)

[**4.4 Passare dalla modalità sandbox alla modalità live** 59](#_Toc57041495)

[**5. Un piccolo esempio della web app** 61](#_Toc57041496)

[**5.1 Creazione Servizio** 61](#_Toc57041497)

[**5.2 Abbonarsi al servizio** 63](#_Toc57041498)

[**Conclusioni** 68](#_Toc57041499)

[**Bibliografia** 69](#_Toc57041500)

**Elenco delle figure**

[**Figura 1. Esempio di funzionamento di un Web Server 16**](#_Toc55988641)

[**Figura 2. modello gerarchico 20**](#_Toc55988642)

[**Figura 3. modello reticolare 20**](#_Toc55988643)

[**Figura 4. modello relazionale 20**](#_Toc55988644)

[**Figura 5. modello a oggetti 20**](#_Toc55988645)

[**Figura 6. Schema E-R 32**](#_Toc55988646)

[**Figura 7. Modello Relazionale 33**](#_Toc55988647)

[**Figura 8. Script DDL tabella abbonato 34**](#_Toc55988648)

[**Figura 9. Script DDL tabella abbonamento 34**](#_Toc55988649)

[**Figura 10. Script DDL tabella provider 34**](#_Toc55988650)

[**Figura 11. Script DDL tabella notizia 35**](#_Toc55988651)

[**Figura 12. Script DDL tabella servizio 35**](#_Toc55988652)

[**Figura 13. Suddivisione del Sistema in sottosistemi 37**](#_Toc55988653)

[**Figura 14. Object diagram del sito web 38**](#_Toc55988654)

[**Figura 15. Navbar 39**](#_Toc55988655)

[**Figura 16. Codice per realizzare la Navbar 39**](#_Toc55988656)

[**Figura 17. Visualizzazione Navbar su dispositivi mobile 39**](#_Toc55988657)

[**Figura 18. Carousel 40**](#_Toc55988658)

[**Figura 20. Codice di realizzazione del carousel 40**](#_Toc55988659)

[**Figura 21. Form di accesso 41**](#_Toc55988660)

[**Figura 22. Codice per realizzare la sezione di login 41**](#_Toc55988661)

[**Figura 23. Sezione chi siamo 42**](#_Toc55988662)

[**Figura 24. Codice di realizzazione della sezione chi siamo 42**](#_Toc55988663)

[**Figura 25. Footer 42**](#_Toc55988664)

[**Figura 26. Codice di realizzazione della sezione footer 43**](#_Toc55988665)

[**Figura 27. Step Form 1 43**](#_Toc55988666)

[**Figura 28. Step Form 2 44**](#_Toc55988667)

[**Figura 29. Step Form 3 44**](#_Toc55988668)

[**Figura 30. Codice HTML per realizzare il multiple step form 45**](#_Toc55988669)

[**Figura 31. In caso di errore nel form 45**](#_Toc55988670)

[**Figura 32. Codice CSS 45**](#_Toc55988671)

[**Figura 33. Codice JavaScript 46**](#_Toc55988672)

[**Figura 34. Carousel-Card 47**](#_Toc55988673)

[**Figura 35. Codice HTML 48**](#_Toc55988674)

[**Figura 36. Codice JQuery 48**](#_Toc55988675)

[**Figura 37. Flusso di lavoro Checkout-Pagamento 50**](#_Toc55988676)

[**Figura 38. Client ID e Secret code 51**](#_Toc55988677)

[**Figura 39. Libreria Rest-api-sdk 52**](#_Toc55988678)

[**Figura 40. Pagina Register-Abbonamento 52**](#_Toc55988679)

[**Figura 41. Servlet AutorizzaPagamento 53**](#_Toc55988680)

[**Figura 42. Pagina di errore 53**](#_Toc55988681)

[**Figura 43. Classe PaymentServices 54**](#_Toc55988682)

[**Figura 44. Metodo getRedirectURLs 54**](#_Toc55988683)

[**Figura 45. Check-out di PayPal 55**](#_Toc55988684)

[**Figura 46. Metodo getTransactionInformation 55**](#_Toc55988685)

[**Figura 47. Pagina pagamento PayPal 56**](#_Toc55988686)

[**Figura 48. Metodo getApprovalLink 56**](#_Toc55988687)

[**Figura 49. Servlet ReviewPaymentServlet 57**](#_Toc55988688)

[**Figura 50. Metodo getPaymentDetails 57**](#_Toc55988689)

[**Figura 51. Pagina Review-Payment 58**](#_Toc55988690)

[**Figura 52. Servlet EsecuzionePagamento 58**](#_Toc55988691)

[**Figura 53. Metodo executePayment 59**](#_Toc55988692)

[**Figura 54. Passare dalla modalità sanbox alla modalità live 59**](#_Toc55988693)

[**Figura 55. Navbar 61**](#_Toc55988694)

[**Figura 56. Sezione registrazione 61**](#_Toc55988695)

[**Figura 57. Sezione Crea Servizio 62**](#_Toc55988696)

[**Figura 58. Sezione Carica Immagine 62**](#_Toc55988697)

[**Figura 59. Homepage Provider 63**](#_Toc55988698)

[**Figura 60. Sezione Servizi 63**](#_Toc55988699)

[**Figura 61. Sezione abbonati 64**](#_Toc55988700)

[**Figura 62. Portale d’accesso di PayPal 64**](#_Toc55988701)

[**Figura 63. Check-out PayPal 65**](#_Toc55988702)

[**Figura 64. Pagina web Revisione 65**](#_Toc55988703)

[**Figura 65. Homepage Abbonato 66**](#_Toc55988704)

# **Introduzione**

L’evoluzione tecnologica ha modificato i comportamenti degli individui e delle aziende nel processo di comunicazione informativo e d’acquisto. Tali cambiamenti hanno creato nuove opportunità e un nuovo mercato con caratteristiche e bisogni differenti rispetto a quello tradizionale. Le nuove dinamiche hanno fatto sì che le imprese rivedessero le loro procedure, considerando Internet come un vantaggio sia in termini di efficienza che di efficacia. Infatti, fino a qualche anno fa, la mancanza di contatto e poca familiarità con il mondo del web oppure l’impossibilità di possedere un pc, tablet, smartphone e altri dispositivi connessi alla rete, limitavano l’utilizzo dei siti web o qualsiasi altra applicazione web. Oggi la situazione è completamente diversa in quanto la fruizione delle applicazioni web è entrata a far parte della vita quotidiana di milioni di persone, complice anche una maggiore fiducia nel trattamento dei dati personali e soprattutto nei metodi di pagamento.

Il progetto realizzato ha diversi vantaggi sia per gli utenti sia per le aziende di provider[[1]](#footnote-1). Gli utenti, infatti, beneficiano della comodità di informarsi direttamente a casa propria, della possibilità di abbonarsi rapidamente ai vari servizi, inclusi quelli difficilmente reperibili nelle proprie città. Le aziende di provider, d’altro canto, attraverso la piattaforma online, godono di una maggiore visibilità, con conseguente incremento sia nel numero di clienti raggiungibili, sia nel fatturato. Pertanto, l’obiettivo principale del progetto è la realizzazione di una piattaforma web che permetta a un utente che voglia offrire un qualunque servizio di creare un account nel quale inserire le notizie inerenti tale servizio e renderle visibili ai clienti abbonati.

Nel dettaglio, l’elaborato si struttura come segue: il primo capitolo spiegherà cosa è un sito web, ripercorrendo la sua evoluzione nel tempo e soffermandosi sulla sua importanza in questo periodo. Nel secondo capitolo si affronterà la tematica degli strumenti tecnologici attualmente disponibili per lo sviluppo di un sito web. I principali argomenti saranno i Web Server[[2]](#footnote-2), i linguaggi mark-up[[3]](#footnote-3), di programmazione, framework[[4]](#footnote-4) e infine le API[[5]](#footnote-5). Il terzo capitolo riguarda lo sviluppo del progetto nella pratica e le procedure attuate nella progettazione e nella realizzazione della soluzione software. Si partirà definendo lo schema entità-relazioni, poi si passerà al modello relazionale e, a quel punto, sarà possibile implementare la struttura del database nel sito. La fase di realizzazione del progetto continua con la creazione del back-end[[6]](#footnote-6), utilizzando come riferimento le richieste del cliente in termini di funzionalità. Si procederà in seguito allo sviluppo del front-end[[7]](#footnote-7) del sito, per il quale sarà utilizzato un framework Bootstrap al fine di rendere il sito responsive e realizzare strutture grafiche, come carousel-card e multiple step forms, per facilitare l’interazione con l’utente. Il quarto capitolo riguarderà l’integrazione del sistema di pagamento PayPal all’interno del progetto: inizialmente, sarà spiegato un tipico flusso di lavoro dal processo di check-out al processo di pagamento, per poi approfondire l’implementazione di tali processi all’interno del sistema e infine descrivere come passare dalla modalità di test alla modalità reale del sistema di pagamento. Nel quinto capitolo verrà presentato, infine, un piccolo esempio del progetto, mostrando attraverso delle schermate le funzionalità implementate.

**Capitolo 1**

# **1.Sito Web**

## **1.1 Cos’è un sito web?**

Un sito web è un insieme di pagine caricate su un server, che raccontano o descrivono una particolare attività, sono collegate tra loro attraverso dei link e raggiungibili attraverso un URL[[8]](#footnote-8) chiamato dominio. [1] Un sito web è uno strumento di comunicazione che può essere sfruttato dalle aziende o da qualsiasi persona che voglia raccontare, descrivere o spiegare un determinato argomento su cui si ritiene di essere preparati o a cui si è interessati. Per poter essere visibile da un dispositivo come un tablet o un pc, il sito web ha bisogno di alcuni servizi essenziali, quali un nome a dominio, un servizio di *hosting* dove caricare tutti i file e un database per raccogliere i dati in tabelle ordinate. Tutti questi dati e questi file devono essere ordinati e schematizzati attraverso l’uso di linguaggi di programmazione web, come HTML, PHP e JavaScript. Infine, necessita di un motore di ricerca, come Google Chrome o Safari, ovvero software creati per raggiungere un sito web.

## **1.2 Evoluzione del web**

Tutto è partito nel 1989, quando Tim Berners Lee inventò il *world wide web,* un sistema che permetteva la consultazione collettiva via Internet di pagine web raggiungibili con *iperlink[[9]](#footnote-9).* Era il 1991, quando Berners Lee pubblicò il primo sito web al mondo, presso il CERN e da quel momento in poi il mondo non è stato più lo stesso. Egli sviluppò il primo programma in grado di leggere il codice HTML e i protocolli http e FTP nonché il primo browser al quale seguirono Safari ed Explorer. [2]

### **1.2.1 Il web 1.0**

Gli anni dal 1990 al 2000 sono stati l’era del Web 1.0 ovvero, secondo la definizione dello stesso Berners, dell’*only read web*. Era l’internet dei contenuti, contrassegnata da siti web statici, realizzati in semplice HTML con una frequenza di aggiornamento ridotta. [2] Solo i *webmaster[[10]](#footnote-10)* avevano le competenze tecniche necessarie e gli strumenti per poter aggiornare le pagine di un sito internet. L’utenza poteva solamente usufruire dei contenuti senza creare interazione e le pagine offrivano la possibilità di essere semplicemente consultate.

### **1.2.2 Il web 2.0**

Negli anni 2000 – 2006, si entra nella fase del *read-write web*. Con l’introduzione dei linguaggi di programmazione dinamici, gli sviluppatori hanno permesso all’utenza non tecnica di interagire con i contenuti dei siti internet. [2] Per la prima volta, si è data grande importanza all’usabilità e al modo di condividere i contenuti. Questa fase è caratterizzata da una partecipazione attiva degli utenti alla costruzione dei contenuti, alla loro classificazione e distribuzione; gli elementi principali sono i Blog, i Wiki, i Social Network ed i Forum.

### **1.2.3 Il web 3.0**

Il web non ha smesso di evolversi: nel 2006, si è cominciato a parlare di web 3.0, l’epoca del *read-write-execute* web. Le parole d’ordine, in questo tempo, sono dati e semantica. La differenza in questo caso non è l’introduzione di nuove tecnologie, ma è data da diversi fattori, quali:

* La rete come enorme database: l’introduzione, prima degli RSS[[11]](#footnote-11) e dei file XML[[12]](#footnote-12), poi dei *rich snippet[[13]](#footnote-13)* e metadata offre la possibilità di utilizzare Internet come un enorme database, da impiegare in diverse applicazioni per recuperare dati da fornire all’utenza. Si parla quindi di Data Web.
* Le intelligenze artificiali, ovvero software capaci di interagire con l’utenza, ad esempio: gli algoritmi di Google, che analizzano la rete per comprendere come posizionare i contenuti a seconda della qualità e della pertinenza con determinate parole chiave; l’algoritmo di Facebook, che mette in evidenza le notizie dei nostri amici che sono maggiormente in linea con il nostro pensiero; gli algoritmi che regolano i software anti-spam per filtrare i messaggi realizzati da utenti reali e quelli creati artificiosamente da software.
* Il web semantico, ossia una declinazione del concetto di rete come database con contenuti correlati a determinate parole chiave, che rendono più performante la ricerca di informazioni.
* Un web adattabile ai diversi dispositivi: il Responsive Web Design e la grafica vettoriale correlata permettono di poter visualizzare il contenuto presente sulla rete in maniera congeniale ai diversi dispositivi.
* Un web potenziato, capace di modificare la società: grazie ai Social Network i contenuti di internet escono dallo schermo e modificano il mondo che ci circonda. La potenza del mezzo di comunicazione digitale permette uno scambio di informazioni tra azienda/consumatore, politico/elettore, artista/fan e proprietario/visitatore che prima era impensabile.
* Il web in 3 dimensioni: le nuove tecnologie e l’elevato accesso alla rete internet hanno permesso di replicare la realtà in formato digitale. Possiamo quindi accedere alla rete e effettuare buona parte delle attività del nostro quotidiano. Il primo esempio è stato second life, ma ad oggi, Facebook, pur se graficamente in una sola dimensione, è il nuovo modello di web 3D. [3]

### **1.2.4 Il web 4.0**

Il tempo digitale sembra correre a velocità elevatissime, infatti già si parla di Web 4.0. Per adesso, non abbiamo una definizione certa, ma possiamo dire che le parole d’ordine sono “spazio” e “big data”, mentre i fattori che porteranno a questa evoluzione potrebbero essere:

* La realtà aumentata dei Google Glasses o gli smartwatch, interfaccia veloce di comunicazione con il proprio microcomputer portatile (lo smartphone), i quali ci permetteranno in futuro (e in alcuni casi già al giorno d’oggi) di interagire in tempo reale con il web, sovrapponendo il mondo che ci circonda con la rete. Ad esempio, una persona con al viso i Google Glasses, dopo aver visionato la mappa per andare in un bar, potrebbe scegliere di cambiare meta perché in quel momento è sovraffollato o chiuso a causa di un’emergenza.
* La formazione di un alter ego digitale: pian piano che i nostri documenti si aggiornano e collegano fra loro, inglobando chip con a supporto una infrastruttura tecnica e man mano che popoliamo la rete con i nostri contenuti personali, andremo a creare un vero e proprio alter ego virtuale che ci permetterà, nel bene e nel male, di far interagire in real-time le due identità, quella reale e quella digitale.
* Le nuove interfacce: la domotica, che pian piano si diffonde nei nostri elettrodomestici e nelle nuove automobili intelligenti, ci permetterà di scambiare i dati relativi al mondo reale con il nostro alter ego digitale. [3]

## **1.3 Importanza di avere un sito web**

Avere un sito web ormai non è solo utile, ma addirittura necessario. Facebook, Twitter, Instagram, YouTube e altri social network hanno favorito tantissime realtà professionali e imprenditoriali, offrendo la possibilità di apparire su Internet. In Italia, il numero di persone che utilizzano Internet abitualmente, nella fascia di età compresa fra i 16-74 anni, si aggira intorno al 56% e questo dato percentuale, per quanto ancora basso rispetto ad altre realtà Europee, aumenta al decrescere della fascia di età presa in considerazione. Con il passare degli anni, sulla base delle attuali tendenze, avremo sicuramente una popolazione sempre più connessa e votata alla rete per la soddisfazione di qualsiasi richiesta o bisogno. [4]

I vantaggi dell’avere un sito web sono innumerevoli. Innanzitutto, un sito Internet consente di essere sempre online perché è accessibile da ogni parte del mondo, ad ogni ora della giornata, permettendo di ricevere prenotazioni, acquisti e visualizzazioni anche dopo l’orario di chiusura. Un altro vantaggio consiste nella possibilità di effettuare aggiornamenti continui ed in tempo reale. Un esempio lampante di questa opportunità ci è offerto dai volantini, i cui contenuti diventano rapidamente obsoleti, ma grazie ad un sito web essi possono essere aggiornati in modo quasi immediato, evitando per di più i relativi costi di stampa e distribuzione. Non trascurabile è anche il vantaggio di semplificare alcuni processi aziendali, come la consultazione diretta di un magazzino da parte di un cliente, che può verificare la disponibilità di un prodotto senza la mediazione di un operatore fisico appositamente impiegato dall’azienda. Per questi ed altri motivi, il sito web risulta essere un elemento cruciale per il successo di molte realtà aziendali e non.

**Capitolo 2**

# **2. Supporto tecnologico alla realizzazione del sito**

Prima di parlare degli strumenti necessari allo sviluppo di un sito web, è necessario chiarire cosa serva per creare un generico sito Web dinamico, ovvero un sito che permetta all’utente di interagire con lo stesso. Innanzitutto, il primo strumento utile è un database che contenga tutte le informazioni da presentare agli utenti in base alla richiesta effettuata. A causa di questa caratteristica fondamentale, il sito dinamico non può essere realizzato solamente con il linguaggio di mark-up HTML, ma deve essere implementato con un linguaggio di programmazione che ne permetta l’interazione con il database.

Oggigiorno non è necessario avere delle competenze di programmazione elevate, infatti, grazie ad alcuni software detti Content Management System, che creano un ambiente preconfezionato e standardizzato senza effettuare alcuna installazione, è possibile gestire i contenuti, come ad esempio i post di un blog.

I CMS attualmente più utilizzati sono *WordPress, Joomla* e *Drupal,* i quali tramite l’implementazione di alcuni plug-in, permettono di realizzare servizi difficili da effettuare da zero, come la creazione di un sito di e-commerce, che può essere facilmente costruito grazie al plug-in WooCommerce di WordPress.

## **2.1 Web Server**

Recentemente, con lo sviluppo della tecnologia e di Internet, si sente spesso parlare di server e in particolare di Web Server anche da parte di “non addetti” al settore, quindi è doveroso darne una definizione: un Web Server è un’applicazione software che, in esecuzione su un server, è in grado di gestire le richieste di trasferimento di pagine web di un client ovvero di un *browser*.

### **2.1.1 Cos'è un server e cos'è un client?**

Per approcciarsi al mondo dei Web server, occorre definire il paradigma Client-Server, su cui si basa la comunicazione Web. Secondo questo paradigma, la comunicazione è distribuita su almeno due nodi: uno, definito client, richiede un servizio a un altro nodo, definito server, utilizzando lo stesso linguaggio comune, detto protocollo. Il server risponde indicando l’esito della richiesta e allegando eventuali risorse fornite come risultato.

Il protocollo usato dal paradigma Client-Server per il Web è http ed i servizi consistono principalmente in richieste di pagine Web da parte del client, che si identifica nel *browser Web* (il software usato per la navigazione su Internet).

### **2.1.2 Funzionamento di un Web Server**

Prendiamo come esempio una classica visita ad un sito internet da un *browser Web.* L’utente inserisce l’url nella barra degli indirizzi e il *browser* invia questa richiesta, attraverso il protocollo HTTP o HTTPS, verso il server. Nel server risiedono i file che compongono il sito web oggetto della richiesta, per cui quest’ultima viene presa in carico proprio dal Web Server. Questo andrà a recuperare sul server il file necessario e lo restituirà al *browser* che ne ha fatto richiesta e, a quel punto, il *browser* tradurrà questa risposta visualizzando sullo schermo la pagina web desiderata.

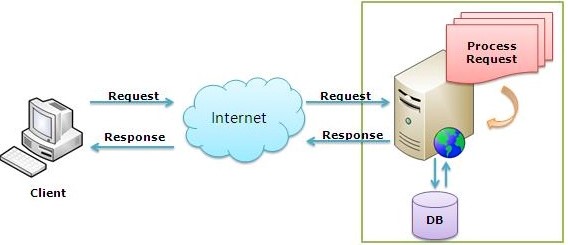


Figura 1. Esempio di funzionamento di un Web Server

### **2.1.3 La scelta del Web Server**

La scelta del Web server per gestire le comunicazioni HTTP e HTTPS non è difficile, poiché di solito si ricade su *Apache[[14]](#footnote-14),* un web server di tipo open source e compatibile con la maggior parte dei sistemi operativi in commercio, grazie al quale possiamo costruire e provare applicativi web senza il bisogno di caricarli su hosting web, perché è possibile installare in locale, cioè sul nostro computer. Altri Web server meno utilizzati, ma pur sempre molto affidabili, sono *lighttpd[[15]](#footnote-15)* (server open source e molto leggero sviluppato dalla Lightppd Developers) e *nginx[[16]](#footnote-16),* scritto in C e C++.

## **2.2 DBMS**

Il database è un insieme di dati omogeneo per contenuti e per formato, memorizzati in un computer. Risulta essere il cuore dei moderni siti Web dinamici e si identifica con il DBMS[[17]](#footnote-17), che rappresenta il sistema software per creare, manipolare e interrogare database.

Il linguaggio standard per la creazione, l’interrogazione e la manipolazione nei DBMS relazionali è SQL[[18]](#footnote-18) e vi è una vasta gamma di prodotti software che implementano tale linguaggio. Inoltre, è necessario decidere se utilizzare un DBMS open source o acquistarne uno con licenza proprietaria.

### **2.2.1 MySQL**

MySQL[[19]](#footnote-19) è un DBMS open source di tipo relazionale. La peculiarità dei DBMS relazionali è la struttura del modello di memorizzazione dei dati. Questi, nei database sono memorizzati in tabelle aventi uno o più attributi o campi e un valore per ciascun attributo all’interno del dominio o tipo di dato consentito.

Un concetto importante è la tupla o riga, che rappresenta l’insieme non ordinato di valori assunti dagli attributi. L’esempio classico è la tabella agenda telefonica, dove gli attributi sono nome, cognome e numero di telefono e i valori assunti da una delle righe potrebbero essere “Mario”, “Rossi” e “+39376351591”.

Rilasciato per la prima volta nel 23 maggio 1995, MySQL è diventato proprietà della Sun Microsystems che in seguito è stata acquisita dalla Oracle Corporation. È supportato da numerosissimi linguaggi di programmazione, come Java, .NET, PHP e Python, ed è composto da un client a riga di comando e un server.

Supportato dai sistemi operativi *Unix, Unix-like* e *Windows*, ne esistono versioni modificate, come *MariaDB[[20]](#footnote-20), Drizzle[[21]](#footnote-21)* e *Percona server[[22]](#footnote-22),* tutte, però, meno utilizzate rispetto a MySQL.

### **2.2.2 Compiti, funzioni e requisiti di un sistema di gestione di base di dati**

Il DBMS è la componente più importante di un sistema di database. In assenza di tale sistema, il database non potrebbe essere gestito, controllato o monitorato. Il software è responsabile anche della gestione di tutti gli accessi in lettura e scrittura al database. Un termine ampiamente usato per descrivere le funzioni e i requisiti delle transazioni di un sistema di gestione di base di dati è ACID[[23]](#footnote-23), acronimo che racchiude i requisiti più importanti di un DBMS:

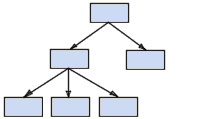
* L’atomicità, ossia la proprietà del tutto o niente del DBMS, sta ad indicare che la transazione deve essere eseguita per intera, non sono ammesse esecuzioni parziali.
* La coerenza implica che le transazioni di successo lascino un database stabile, il che richiede un controllo costante di tutte le transazioni.
* L’isolamento è il requisito secondo il quale le transazioni non devono ostacolarsi a vicenda e di solito è garantito da alcune funzioni di blocco.
* La durabilità indica che tutti i dati vengono memorizzati in modo permanente nel DBMS, anche dopo che una transazione è stata completata con successo. Ciò vale anche o soprattutto in caso di errori di sistema o di problemi del DBMS. I messaggi di errore delle transazioni, che registrano tutti i processi nel DBMS, sono essenziali per la durabilità. [5]

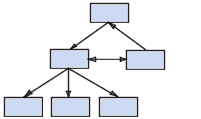
Il compito principale dei database è di memorizzare testi, documenti, password e altre informazioni a cui si può accedere tramite *query*. La maggior parte dei database consente, a seconda dei permessi di accesso, di modificare direttamente le informazioni memorizzate, e anche di cancellarle completamente. In alcuni casi, i dati cancellati possono essere recuperati, in altri, le informazioni sono perse per sempre. Un altro compito principale dei database riguarda la sicurezza, in quanto le persone non autorizzate non possano accedere ai dati memorizzati. Essenziale per la sicurezza dei dati è, oltre a un efficace metodo di crittografia, un’attenta amministrazione, soprattutto da parte dell’amministratore. Altri compiti importanti sono ottimizzazione delle *query*, funzionamento multiutente e *stored procedures[[24]](#footnote-24).*

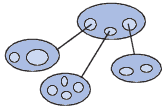
### **2.2.3 Tipi di DBMS**

Nel corso degli anni sono stati adottati numerosi modelli per i dati, a fronte dei quali esistono vari tipi di database. I tipi più comuni sono:

* Database gerarchici, in cui i dati sono classificati gerarchicamente, secondo una direzione discendente. Questo modello prevede che i dati siano organizzati secondo strutture ad albero, il che significa che esiste una relazione padre-figlio tra i dati.
* Database reticolari: modello molto simile a quello gerarchico, che nasce come estensione di quest’ultimo, poiché, anche in questo caso, gli insiemi di dati sono legati dalla relazione padre-figlio. Tuttavia, nei database reticolari ogni insieme di dati può appartenere a uno o più insiemi, per cui la struttura risultante è una rete di insieme di dati.
* Database relazionali: i database appartenenti a questa categoria si basano sul modello relazionale, la cui struttura principale è la relazione, cioè una tabella composta da righe e colonne. Ciascuna riga, che in gergo viene chiamata tupla, rappresenta un’entità che vogliamo memorizzare nel database. Le caratteristiche di ciascuna entità sono definite invece dalle colonne delle relazioni, che vengono chiamate attributi. Entità con caratteristiche comuni, cioè descritte dallo stesso insieme di attributi, faranno parte della stessa relazione.
* Database ad oggetti: lo schema di un database ad oggetti è rappresentato da un insieme di classi che definiscono le caratteristiche ed il comportamento degli oggetti. La principale differenza con i modelli esaminati finora è la non passività dei dati. Infatti, con un database tradizionale, le operazioni che devono essere effettuate sui dati vengono gestite dalle applicazioni che li utilizzano. Con un database ad oggetti, al contrario, gli oggetti memorizzati nel database contengono sia i dati e sia le operazioni possibili su tali dati, per cui, quando vengono chiamati in causa, gli oggetti sanno come comportarsi, senza doversi appoggiare ad applicazione esterne.



* Figura 2. modello gerarchico
* 
* Figura 3. modello reticolare
* Immagine che contiene schermo, edificio, finestra, treno

  Descrizione generata automaticamente
* Figura 4. modello relazionale
* 
* Figura 5. modello a oggetti

### **2.2.4 Alternative**

Come detto nei paragrafi precedenti, la scelta del DBMS può essere effettuata tra una grande varietà di prodotti e produttori. Esistono, ad esempio, DBMS non relazionali, ovvero che non utilizzano la struttura classica a tabelle dei database relazionali. Tra questi, spicca MongoDB[[25]](#footnote-25), software libero che utilizza documenti in stile JSON[[26]](#footnote-26) ed è utilizzato prevalentemente per la memorizzazione di documenti. Per altre applicazioni particolari è possibile anche ricorrere a database basati sullo storage key-value, come Redis[[27]](#footnote-27), open source con persistenza facoltativa.

## **2.3 I Framework**

Nello sviluppo di applicazioni spesso si ricorre all’utilizzo di strutture di supporto su cui un software può essere organizzato: i framework. Alla base di un framework c’è sempre una serie di librerie software che possono essere utilizzate da uno o più linguaggi di programmazione. Lo scopo di un framework è di far risparmiare tempo allo sviluppatore nella riscrittura di codice già steso per compiti simili, caratteristica particolarmente utile adesso che le interfacce utente sono diventate sempre più complesse. L’esempio più classico è il menu a tendina che, favorendo un’esperienza utente ottimale, dovrebbe rimanere sempre uguale, proprio grazie ad un framework, che può essere implementato con poche righe di codice in qualsiasi pagina web.

### **2.3.1 Bootstrap**

Nato come un progetto all’interno di Twitter ad opera degli sviluppatori Mark Otto e Jacob Thornton ed oggi indipendente, Bootstrap può essere considerato il re dei framework per lo sviluppo interfacce web. [6] La caratteristica più importante di Bootstrap è il *responsive web design:* il layout delle pagine web si regola dinamicamente, tenendo conto delle caratteristiche del dispositivo utilizzato, sia esso desktop, tablet o smartphone. Un altro punto di forza di Bootstrap è la documentazione, caratteristica importante nel settore informatico. I membri della comunità hanno tradotto la documentazione in varie lingue tra cui cinese, spagnolo e russo, arricchendo il tutto con esempi d’uso pratico.

Un componente importante di Bootstrap è il Grid System, ovvero un insieme di fogli che considerano il contenitore generale disposto su una griglia con una larghezza base di 960 px, nella quale distribuire i contenuti in varie righe e colonne che definiranno il layout, ovvero la struttura base del nostro template. Altro fondamentale componente è il CSS Base, lo strumento che contiene una serie di stili predefiniti riguardanti tutta la parte tipografica, ovvero la parte inerente i titoli e la gestione di tabelle, paragrafi, form nonché pulsanti per richiamare stili e icone pronti all’uso. Infine, abbiamo gli altri componenti JavaScript, ossia elementi quali menu *dropdown[[28]](#footnote-28)*, interfacce a tab, *tooltip[[29]](#footnote-29)*, alert, menu *adaccordion[[30]](#footnote-30)*, slider e altri componenti.

### **2.3.2 I vantaggi dei framework**

Utilizzando un framework, si risparmiano tempo e costi nello sviluppo software. L’idea di base è il riutilizzo del codice, cosa che si riferisce soprattutto alle funzionalità essenziali, come connessioni database, template delle pagine, processi di caching o funzioni di sicurezza, messi a disposizione dai framework come elementi già preimpostati del codice. In questo modo, il lavoro di sviluppo si limita ad uno specifico codice del programma di una nuova applicazione web.

Poiché la maggior parte dei framework vengono offerti gratuitamente, di solito non sussistono costi per la realizzazione. Inoltre, i framework supportano la generazione di un codice sorgente corretto, dato che gli sviluppatori possono ricorrere a componenti già verificati per quanto riguarda le funzioni standard. Le parti dei programmi messe a disposizione dai framework, infatti, sono sottoposte il più delle volte a numerosi cicli di sviluppo e vengono ottimizzate continuamente, facendo sì che la community svolga la funzione di tester e co-sviluppatore e permettendo di scoprire e risolvere velocemente le falle di sicurezza dei nuovi componenti anche in caso di progetti grandi. Uno scambio tra utenti e sviluppatori avviene generalmente nel forum del progetto, che sono in parte moderati dai team di supporto

### **2.3.3 Gli svantaggi dei framework**

Tra gli svantaggi nell’utilizzo dei framework troviamo sicuramente la presenza di codice superfluo, poiché essendo essi progettati come soluzioni universali, è raro che gli sviluppatori utilizzino tutte le funzioni della struttura preimpostata, per cui un’applicazione comprenderà più codice di quanto sia necessario. Altro svantaggio è la raccomandazione a rivolgersi ad uno specifico produttore o ad una community di sviluppatori per utilizzare al meglio un framework, incappando spesso in precise condizioni di licenza che possono causare problemi in caso, ad esempio, di ulteriore sviluppo del framework. Molto sfavorevole è, inoltre, il ricorso eccessivo ai framework, poiché si corre il rischio che le conoscenze di base vadano perdute. Infatti, gli utenti che programmano esclusivamente sulla base di framework, si confrontano in maniera meno intensiva con i linguaggi di programmazione utilizzati. Infine, il rovescio della medaglia nell’avere un codice sorgente quasi sempre disponibile liberamente, è rappresentato dalla pressoché totale esposizione ad eventuali attacchi da parte di avversari sulla rete.

### **2.3.4 Framework front-end più utilizzati**

I Framework Front-End sono veri e propri kit di progettazione dedicati al Web, che permettono di avviare un progetto in modo rapido ed efficiente, in quanto dotati di tutte le basi necessarie alla fase di realizzazione. Tra i più importanti, oggi troviamo:

* Pure Css: è simile all’ offerta di Bootstrap, con la promessa di estensibilità e personalizzazione. Dal sito ufficiale[[31]](#footnote-31) possiamo ammirare alcuni design accattivanti che mostrano segni di individualità. Pure Css taglia fuori tutto il JavaScript, offrendo un kit CSS puro (come suggerisce il nome) che è certamente un incentivo per coloro che desiderano cominciare con leggerezza.
* UI Kit: Un framework front-end leggero e modulare per lo sviluppo di interfacce web veloci e potenti. Esso offre una raccolta completa di HTML, CSS, JS e componenti semplici da usare, facile da personalizzare ed estendibili. È probabilmente il più vicino al bootstrap, ma al contempo abbastanza diverso da fornire un minimo di indipendenza nel design. Esso è, inoltre, un framework open source, creato per dare al web un bell’aspetto.
* React**:** è una libreria JavaScript dedicata alla creazione di interfacce utente. Non è un Framework MVC e non fa uso di Modelli, ma utilizza JavaScript per generare Markup. In un’applicazione JavaScript tradizionale, è necessario guardare ai dati modificati e apportare modifiche al DOM per tenerlo up-to-date. Anche AngularJS, che fornisce un’interfaccia dichiarativa tramite direttive e l’associazione dati, richiede una funzione di collegamento per aggiornare manualmente i nodi DOM. React, invece, offre un diverso approccio: quando il componente viene inizializzato, viene chiamato il metodo render, generando una rappresentazione leggera della vista. Da tale rappresentazione, una stringa di markup viene prodotta e iniettata nel documento. Quando i dati subiscono delle modifiche, il metodo render viene chiamato di nuovo. Per poter eseguire gli aggiornamenti nel modo più efficiente possibile, il valore restituito dalla precedente chiamata si differenzia rispetto a quello nuovo, generando un set minimo di modifiche da applicare al DOM. [7]

Oltre a questi framework front-end più utilizzati, meritano una menzione anche GroundworkCSS, Cardinal, Workless, TopCoat e Centurion.

## **2.4 Linguaggi di programmazione**

### **2.4.1 HTML**

Quando si parla di programmazione web, è bene partire dall’HTML, sebbene esso non sia un linguaggio di programmazione. Come si intuisce dal nome, l’HTML è un linguaggio di markup, vale a dire un insieme di regole che descrivono i meccanismi di rappresentazioni del testo. Quindi, tramite una sequenza di caratteri, si marcano gli elementi di un file di testo marcatori detti tag per assegnare loro determinate caratteristiche o funzioni. Le regole sono stabilite dal World Wide Web Consortium, associazione che definisce gli standard del World Wide Web, e la versione attuale HTML5 è stata rilasciata dal W3C nell’ottobre 2014. Tale rilascio è avvenuto a causa della necessità di fornire direttamente le funzionalità che in precedenza erano fruibili tramite estensioni proprietarie all’esterno dei browser, come Adobe Flash. [8]

I documenti HTML vengono immagazzinati di solito sui dischi rigidi di computer-server costantemente collegati alla rete Internet. Su queste macchine è installato un software specifico, detto web server, che si occupa di produrre e inviare i documenti ai browser degli utenti che ne fanno richiesta usando il protocollo HTTP per il trasferimento dati. Spesso il documento HTML viene generato del tutto o parzialmente tramite un codice eseguibile residente sul server Internet in grado di interagire con le altre applicazioni presenti sul server stesso, come un database, e inviare poi al browser il risultato finale, realizzando le cosiddette pagine dinamiche con cui un utente può compiere operazioni interattive avanzate, ad esempio, filtrare gli articoli all’interno di un catalogo on-line.

### **2.4.2 CSS**

Il CSS è un linguaggio di programmazione utilizzato per definire la formattazione dei documenti HTML. La nascita del CSS è diventata un’esigenza quando ci si è resi conto che i produttori di browser sviluppavano tag propri per la formattazione del codice, creando un problema per i webmaster, poiché la resa grafica era differente utilizzando client differenti e l’assenza di una standardizzazione non permetteva di poter sviluppare lo stesso codice per due browser differenti. Con la nascita dei palmari e degli smartphone, aventi basse risoluzioni e differente rapporto d’aspetto dell’immagine, non era possibile visualizzare pagine create per essere mostrate su un computer. Inoltre, si sentiva la necessità di rendere il codice HTML più leggibile. Per questa ragione, W3C alla fine del 1996 rilascia le direttive per comporre il CSS e separare il contenuto dalla sua formattazione. [9]

Usando i fogli di stile è possibile controllare bordi, colori, margini, allineamenti, font e tante altre proprietà per ottenere l’effetto visivo desiderato. La pratica migliore è posizionare i fogli di stile in un file esterno all’HTML che sarà richiamato dall’HTML stesso all’avvio del file, o tramite JavaScript (discusso in seguito) al verificarsi di un evento come il click del mouse. Inserendo i fogli di stile in file separati sarà possibile riutilizzare lo stesso CSS in più pagine. Il foglio di stile può essere inserito anche all’interno della pagina HTML, ma è necessario copiare ed incollare il codice da una pagina all’altra per riutilizzarlo. Diventa quindi difficile la sincronizzazione e l’aggiornamento. Per questa ragione, il suddetto metodo è sconsigliato, a meno che non si tratti di modifiche grafiche da applicare solo alla pagina considerata. Il terzo modo per inserire i fogli di stile all’interno dell’HTML è detto inline. L’inserimento di fogli di stile inline è un modo per inserire il linguaggio dei fogli di stile direttamente all’interno del tag di apertura di un elemento HTML. Avendo affermato che sarebbe sempre bene separare il contenuto dalla formattazione, può risultare strano osservare che il linguaggio CSS venga integrato nel contenuto, ma, a volte, risulta molto semplice e veloce l’inserimento del CSS inline per piccole modifiche che non saranno utilizzate in altre pagine.

Gli elementi della pagina vengono individuati tramite l’utilizzo di Selettori che specificano uno o più elementi dell’HTML e che si differenziano in 3 tipologie:

* Selettori universali: utilizzati per gli elementi generici dell’HTML espressi nel codice html utilizzando tag, come <h1> o <input>.
* Selettori di classe: identificano gli elementi HTML aventi il nome di classe specificato all’interno dell’attributo class.
* Selettori ID: permettono di identificare in modo univoco un singolo elemento tramite una stringa testuale <input id =” nome\_id”>.

I colori all’interno della sintassi CSS possono essere espressi o in formato esadecimale o sotto forma RGB. Un terzo modo, ovvero usare il nome dei colori in inglese, funziona soltanto con 16 colori predefiniti.

### **2.4.3 JavaScript, JQuery e AJAX**

JavaScript è un linguaggio di scripting lato client, orientato agli oggetti e agli eventi, comunemente utilizzato all’interno del codice HTML di siti Web e applicazioni Web per ottenere effetti dinamici interattivi tramite funzioni invocate da eventi innescati dall'utente sulla pagina Web in uso. L’integrazione avviene direttamente nel file HTML racchiusa dal tag <script> o tramite appositi file separati con estensione .js, richiamati inserendo il tag <script> l’attributo src che abbia come valore l’URL del file.

JQuery è una libreria JavaScript per Web Application, nata con l’obiettivo di semplificare la selezione, la manipolazione, la gestione degli eventi e l'animazione di elementi. Il framework fornisce metodi e funzioni per gestire al meglio gli aspetti grafici e strutturali, come il posizionamento degli elementi, il click sulle immagini, la manipolazione del documento, la validazione lato client dei campi, l’esecuzione di animazioni e la modifica dei documenti di stile. Tra le API, è presente anche un gestore delle richieste AJAX.

AJAX è una tecnica di sviluppo software che permette di effettuare lo scambio di dati in background tra client e server. Tramite AJAX è quindi possibile ottenere siti più interattivi, poiché lo scambio di dati avviene senza ricaricare la pagina. Inoltre, le richieste asincrone permettono di ottenere pagine più veloci, grazie alla possibilità di utilizzare meno banda, evitando di ricaricare tutta la pagina e aggiornando solo la parte interessata.

### **2.4.4 Java Enterprise Edition**

Java Enterprise Edition prevede, oltre alle caratteristiche di base del linguaggio incluse nella Standard Edition, una serie di tecnologie volte a creare applicazioni complesse Client-Server, non necessariamente basate sull’uso del web. Diversamente dai linguaggi di scripting, Enterprise Edition si basa su concetti orientati agli oggetti.

Gli elementi costituitivi sono quattro:

* servlet: classe Java con metodi per la gestione delle richieste e delle risposte HTTP, in particolare i metodi GET e POST.
* Java Servlet Pages (JSP): documenti in cui può essere presente codice Java mescolato all’HTML che viene compilato sul server. Si utilizza lo pseudo tag per racchiudere il codice che è compilato alla prima richiesta in una servlet.
* Componenti software: *Java Beans* ed Enterprise Java Beans gestiscono funzioni specifiche e sono progettati per rendere semplice il riutilizzo. Nella versione Enterprise fruiscono di opzioni avanzate quali la persistenza degli oggetti e la distribuzione del carico.
* Contenitore: il contenitore di servlet è un elemento software che lavora in cooperazione con un server Web oppure lo incorpora. Le funzioni di un contenitore sono: la decodifica delle richieste dei client, la gestione del ciclo di vita delle servlet, gestione delle sessioni, supporto alla creazione della risposta http, distribuzione del carico, sicurezza e persistenza.

Un’applicazione Web in Java è un insieme di file e directory disposti secondo uno schema standard descritto della specifica delle servlet. L’insieme dei file contiene codice HTML, immagini, pagine JSP, classi Java (servlet e Java Beans) ecc., sempre sotto una directory radice ospitata nella directory webapps.

## **2.5 API Application Programming Interface**

Si tratta di applicazioni che, mediante modalità standard, espongono le funzionalità di altre applicazioni, ovvero favoriscono il riutilizzo dei servizi noti che possono a loro volta comporsi e scomporsi in base alle necessità. Le API sono indispensabili per rendere disponibili i servizi in risposta alle esigenze di business, poiché grazie ad esse, si può rendere fruibile via mobile un’applicazione enterprise. Il significato di API è dunque quello di semplificare la possibilità di dialogo tra un’applicazione e un’altra evitando ridondanze e inutili replicazioni di codice.

### **2.5.1 API PayPal**

Le API di PayPal sono interfacce di programmazione che offrono strumenti per gestire le funzionalità messe a disposizione da PayPal. Per utilizzare le API di PayPal è necessario disporre di un conto PayPal come pure dei permessi API, per un accesso sicuro dei dati. PayPal dispone di diverse API per la procedura di pagamento, per la ricerca e l’elaborazione delle transazioni ed altre funzioni. Una volta ottenuti e installati i permessi API, è possibile cominciare ad effettuare subito le corrispondenti chiamate. Sono disponibili due interfacce:

* interfaccia Name-Value-Pair (NVP): vengono scambiati parametri fissi e le variabili corrispondenti tramite HTTP. Questo tipo di interfaccia si adatta particolarmente allo sviluppo semplice con linguaggi server tradizionali.
* Interfaccia SOAP: i dati vengono scambiati su protocollo SOAP. Questo tipo di interfaccia è particolarmente indicata per lo sviluppo orientato agli oggetti. [10]

In caso di problemi con una chiamata API, PayPal restituisce dei messaggi di errori dettagliati che aiutano a comprendere e risolvere eventualmente gli errori.

### **2.5.2 Perché utilizzare l’API**

I tre maggiori vantaggi nell’uso delle API sono:

* Indipendenza: le API sono indipendenti dai linguaggi o da precise piattaforme e sono completamente adattabili, garantendo quindi la massima libertà. È possibile, infatti, avere un server Java, PHP, Python senza riscontrare alcun problema.
* Separazione Client-Server: non è solo uno dei principi fondamentali dell’architettura software, ma rappresenta un vero e proprio vantaggio. Tale separazione, infatti, consente di trattare indipendentemente l’evoluzione delle diverse componenti e modificare solo una parte progettuale senza, per questo, essere obbligati a mettere mano sia al server che al client. L’interfaccia utilizzata, inoltre, è utilizzabile su diverse tipologie di piattaforme.
* Scalabilità: la separazione Client-Server si traduce in una miglior scalabilità del sistema stesso.

### **2.5.2 Esempi di API**

Tra le API più famose, troviamo sicuramente quelle di Facebook. Esse hanno permesso a sviluppatori e terze parti di creare migliaia di applicazioni e servizi che accedono ai dati offerti dal social network: basta condividere il proprio account di Facebook per comprare, ad esempio, i biglietti di un’agenzia di viaggi o completare l’acquisto da un sito di e-commerce.

Microsoft e Sony, al contrario, custodiscono molto gelosamente le proprie API di sviluppo per Xbox e PlayStation. Entrambe, infatti, hanno interesse a distribuire le API a un numero ristretto di programmatori e case sviluppatrici, così da tenere sotto controllo coloro che sviluppano giochi per queste piattaforme.

Altre software house, invece, hanno un atteggiamento diverso: una maggiore diffusione delle API garantisce una più ampia diffusione del loro software e piattaforma, per cui garantiscono una distribuzione capillare di questi strumenti. La stessa Microsoft distribuisce liberamente le API di Windows, in virtù del fatto che maggiore è il numero di software esistenti per il suo sistema operativo, maggiore sarà la diffusione del sistema operativo stesso.

**Capitolo 3**

# **3. Realizzazione del Progetto**

In questo capitolo si parlerà di come è stato progettato e implementato il sito web. Indipendentemente dal ciclo delle attività di progetto, è necessario suddividere lo sviluppo in due sezioni distinte: il front-end e il back-end. Nell’ambito dello sviluppo di un sito web, viene definita *front-end* la parte del sito visibile, con cui l’utente interagisce direttamente, mentre si definisce *back-end* la parte invisibile all’utente, dove si implementano le funzionalità ed è accessibile solo dall’amministratore. Per creare un sito web, prima ancora di creare il front-end e il back-end, è tuttavia necessario strutturare il DBMS avendo una chiara conoscenza dei dati da memorizzare e le relazioni che ci sono tra di essi.

## **3.1 DBMS**

Per impostare la struttura del DBMS è necessario conoscere i dati, le informazioni e avere una visione generale delle funzioni che dovrà offrire il sito. Per poter implementare il database bisogna partire dallo schema E-R[[32]](#footnote-32), un modello concettuale comprensibile anche ad utenti non tecnici. Si passa poi al modello relazionale, meno comprensibile ad utenti non tecnici, e successivamente alla costruzione dello schema fisico dei dati contenente le definizioni delle tabelle, dei relativi vincoli di integrità e delle viste[[33]](#footnote-33) espresse in SQL. Di seguito, verranno mostrate nel dettaglio le varie fasi.

### **3.1.1 Schema Entità-Relazione**

Il modello Entità-Relazione è un modello diffusissimo di dati di alto livello, principalmente utilizzato per definire lo schema concettuale di un database. Il modello E-R ha avuto una grandissima diffusione soprattutto per i formalismi grafici semplici e chiari che incorpora. [11] Nella figura seguente viene mostrato lo schema E-R del sito web.

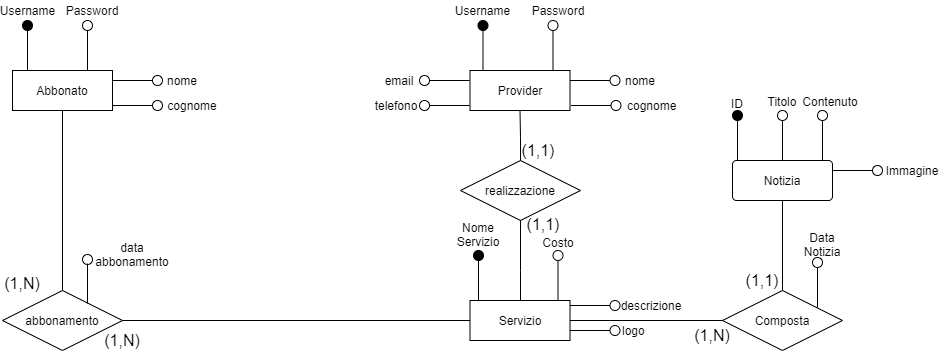


Figura 6. Schema E-R

In questo modello possiamo notare 3 concetti fondamentali: entità, attributi e relazioni. Le entità corrispondono a classi di oggetti del mondo reale che hanno proprietà comuni. Nel progetto realizzato, le entità di interesse sono abbonato, provider, servizio e notizia. Ognuna di queste entità ha delle proprietà dette attributi, ad esempio, l’entità abbonato ha gli attributi username, password, nome e cognome. Tra questi c’è un attributo designato con un pallino nero, cioè username, che è una chiave primaria che serve a identificare univocamente le righe della tabella: in altre parole, due righe distinte non possono avere lo stesso valore sui campi scelti come chiave primaria. L’ultimo concetto è rappresentato dalle relazioni, che corrispondono a legami logici tra entità, come “abbonamento” che esprime un legame tra abbonato e servizio (*Fig.6*). Su ogni relazione sono espresse delle cardinalità, che specificano il numero di istanze in cui l’entità può partecipare alla relazione. Ad esempio, prendendo in considerazione la relazione “realizzazione”, l’istanza “provider” partecipa alla relazione una sola volta, ovvero un provider può creare un solo servizio. In questo modello non sono state realizzate tutte le funzionalità dello schema perché alcuni concetti, come entità deboli, relazioni ricorsive e attributi composti, non sono serviti ai fini dell’applicazione web.

### **3.1.2 Modello Relazionale**

Una volta terminata la fase di analisi concettuale del database e creato un modello concettuale che descrive gli oggetti dell’applicazione web, possiamo passare alla definizione di uno schema logico più vicino al DBMS, ma meno comprensibile agli utenti non informatici. Il modello relazionale si basa sul concetto di relazione, che ha una rappresentazione naturale per mezzo di tabelle, dove ciascuna riga rappresenta una collezione di valori di dati relati. [12]

Nella figura seguente viene mostrato il modello relazionale del sito web.

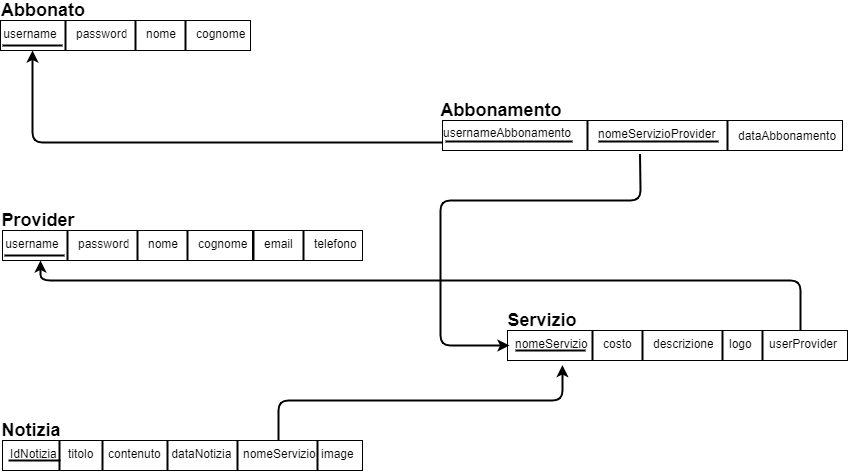


Figura 7. Modello Relazionale

Questo modello è realizzato partendo dallo schema E-R.

Una relazione molti a molti tra due entità si traduce con una relazione con lo stesso nome avente come attributi le chiavi delle entità coinvolte, che formano la chiave della relazione, più eventuali attributi della relazione. Nel nostro caso, ad esempio, la relazione abbonamento si trasforma in tre tabelle: Abbonato, Abbonamento e Servizio, come si può vedere in figura 7. Una relazione uno a molti tra le entità A e B si considera la relazione B (sul lato N) e si include in essa, come chiave esterna, la chiave primaria di A oltre ad eventuali attributi dell’associazione. Nel nostro caso la relazione composta si trasforma in due tabelle: Notizia e Servizio, come si può vedere in figura 7. Infine, le frecce indicano le chiavi esterne e la sottolineatura indica le chiavi.

### **3.1.3 Script Tabelle**

Una volta stabilito il modello relazionale, possiamo definire la struttura del database, attraverso il linguaggio DDL[[34]](#footnote-34). Il linguaggio DDL è un insieme di istruzioni utilizzate per modificare la struttura della base di dati, tra cui: istruzioni di inserimento, cancellazione e modifica di tabelle. Il DDL, oltre a ciò, permette di definire vincoli sia a livello di tupla che a livello di tabella, permette di definire nuovi domini oltre a quelli predefiniti e consente di definire le viste, ovvero tabelle virtuali. [13]

Nelle figure seguenti viene mostrata la struttura del database del sito web.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 8. Script DDL tabella abbonato

Immagine che contiene testo

Descrizione generata automaticamente

Figura 9. Script DDL tabella abbonamento

Immagine che contiene testo

Descrizione generata automaticamente

Figura 10. Script DDL tabella provider

Immagine che contiene testo

Descrizione generata automaticamente

Figura 11. Script DDL tabella notizia

Immagine che contiene testo

Descrizione generata automaticamente

Figura 12. Script DDL tabella servizio

La creazione di una tabella avviene attraverso l’enumerazione delle colonne che la compongono. Per ogni attributo va specificato il dominio, un eventuale valore di default ed eventuali vincoli. Mediante l’istruzione CREATE TABLE si definisce lo schema di una tabella e se ne crea un’istanza vuota. I tipi di dati associati a ciascun attributo possono essere scelti tra i tipi di base già definiti in SQL oppure tra nuovi tipi definiti dall’utente. Durante le creazioni delle tabelle, sono stati definiti i vincoli d’integrità, principalmente è stato usato il vincolo NOT NULL, che vieta la presenza di valori nulli in quella colonna. È stato inoltre usato il vincolo PRIMARY KEY, che identifica la chiave primaria della tabella ed è stato usato il vincolo FOREIGN KEY, in cui la definizione avviene specificando un vincolo e indicando quale chiave viene referenziata, solitamente seguita dalla parola chiave REFERENCES, che permette di specificare a quale colonna della tabella referenzia.

Una volta definita la chiave esterna per garantire che, a fronte di cancellazioni e modifiche, i vincoli di integrità referenziale siano rispettati, sono state inserite le parole chiave ON DELETE e ON UPDATE, sintassi che permette di definire l’azione che il sistema intraprende quando vi è una richiesta di cancellazione e modifica. In tutte le tabelle è stato utilizzato l’attributo CASCADE, di modo che, in caso di richiesta di modifica di un valore di un attributo di una riga che faccia riferimento a chiavi esterne in righe di altre tabelle, tali valori verranno anch’essi aggiornati al nuovo valore, mentre in caso di richiesta di cancellazione questi valori saranno cancellati.

## **3.2 Back-End**

Dopo aver strutturato il database è stato necessario creare il back-end per poter gestire il database tramite interfaccia utente e permettere di realizzare le funzioni richieste dal cliente. Di seguito, sono mostrate le funzioni implementate, la suddivisione del sistema in sottosistemi più piccoli e le classi che costituiscono il sistema.

### **3.2.1 le funzioni del back-end**

Tramite il back-end, il sito supporta due tipi di utenti: abbonato e provider, ognuno dei quali può effettuare operazioni ben precise. L’abbonato supporta diverse operazioni tra cui: abbonarsi a uno o più servizi, visualizzare servizi abbonati, visualizzare servizi non abbonati e infine visualizzare notizie. Il provider, invece, supporta operazioni, quali: creare un servizio, inserire, modificare e cancellare una notizia nonché eliminare definitivamente il servizio. Nei prossimi paragrafi verrà mostrato da quale sottosistema e classi sono state realizzate le operazioni descritte.

### **3.2.2 Struttura: Architettura a 3 livelli**

L’architettura a 3 livelli è un’architettura Client-Server in cui l’interfaccia utente, la logica applicativa e la persistenza dei dati sono sviluppate e mantenute come moduli indipendenti, la maggior parte delle volte su piattaforme separate. Questo tipo di struttura oltre ai vantaggi abituali di software modulare con interfacce ben definite, l’architettura a 3 livelli consente a qualsiasi dei tre livelli di essere aggiornati o sostituiti indipendentemente dal cambiamento di requisiti o tecnologia. L’architettura a 3 livelli ha i seguenti livelli:

Livello di Presentazione: è il livello più alto dell’applicazione ed è visibile all’utente attraverso le pagine web.

Livello di applicazione: è il livello intermedio dell’applicazione e si occupa di controllare le funzionalità di un’applicazione eseguendo elaborazioni dettagliate.

Livello Dati: è il livello più basso dell’applicazione e si occupa di accedere al database per recuperare i dati richiesti dal client. [14]

Nella figura seguente, viene mostrato come è stato suddiviso il sistema in sottosistemi più piccoli.

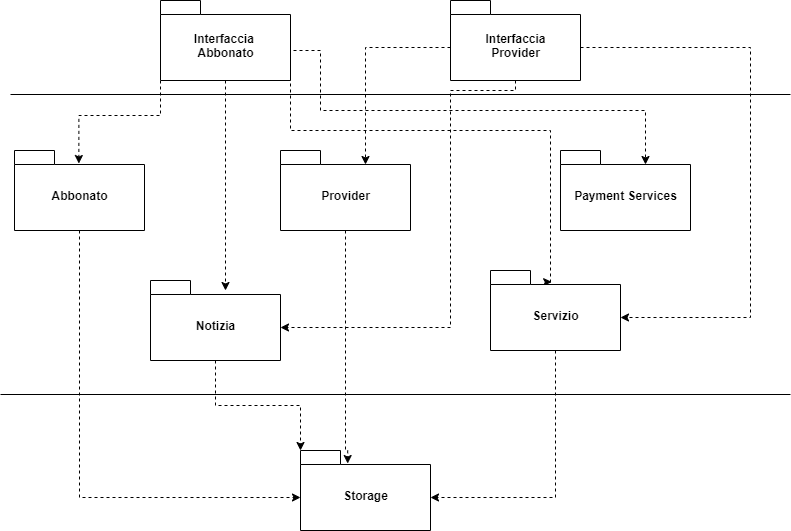


Figura 13. Suddivisione del Sistema in sottosistemi

Il sistema viene diviso in tre livelli. Il primo livello viene diviso per utente, cioè abbonato e provider. All’interno di ogni modulo del primo livello troviamo tutte le pagine web visibili all’utente. Il secondo livello include la logica applicativa, costituita dalle classi che contengono metodi per realizzare le funzioni richieste dal cliente. L’ultimo livello è costituito solo dal gestore dei dati, che contiene le classi DAO[[35]](#footnote-35), le quali permettono di accedere al DBMS per effettuare operazioni su di esso. Infine, le linee tratteggiate indicano le dipendenze tra i vari moduli, ad esempio, le classi della logica applicativa dipendono da storage perché, per ogni operazione che effettuano, chiamano in causa il DBMS.

### **3.2.3 Packages**

Di seguito, viene mostrato il diagramma delle classi, per descrivere il sistema in termini di oggetti e relativi relazioni.

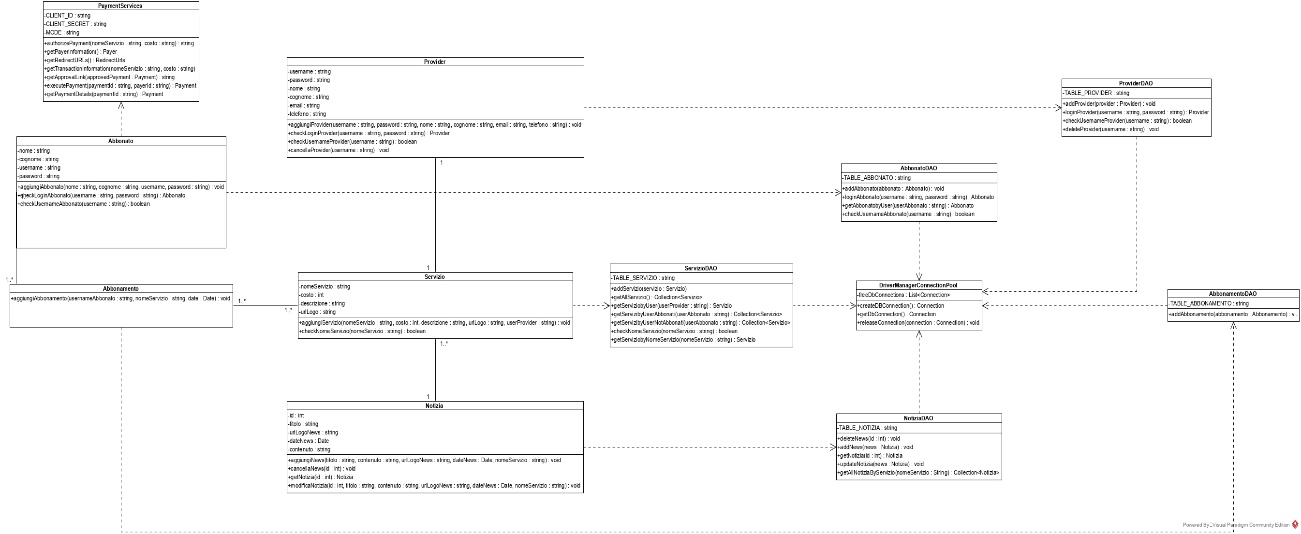


Figura 14. Object diagram del sito web

La classe più importante è DriverManagerConnectionPool perché permette di interfacciarsi con il database sottostante, infatti tutte le classi DAO, da come si può vedere dalla figura 14, dipendono dalla classe DriverManagerConnectionPool. Le classi DAO sono responsabili di effettuare operazioni sul database, infatti contengono metodi che si occupano di implementare le query. Le classi Abbonato e Provider si occupano di gestire la parte di accounting, cioè l’autenticazione dell’utente abbonato e dell’utente provider, mentre le classi Servizio e Notizia permettono di gestire le operazioni possibili da effettuare da parte dell’utente provider, cioè creare un servizio e inserire, cancellare e modificare una notizia. Infine, la classe PaymentServices permette di dialogare con l’API di PayPal per gestire la transazione di pagamento.

## **3.3 Front-End**

Durante la fase finale di sviluppo del back-end si è portato avanti parallelamente lo sviluppo del front-end con l’ausilio del framework Bootstrap.

### **3.3.1 Home Page**

In questa sezione sarà discussa la struttura principale della homepage per comprendere al meglio l’utilizzo di Bootstrap e tratteremo il modo di modificare o sovrascrivere codice CSS per adattare al meglio l’interfaccia grafica al nostro sito. La homepage è divisa in vari contenitori. Il primo contenitore contiene la navbar, un elemento strutturale che include link che consentono di muoversi tra le varie sezioni del sito web. Nella figura seguente viene mostrato la navbar del sito.



Figura 15. Navbar

In seguito, viene mostrato il codice di realizzazione.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 16. Codice per realizzare la Navbar

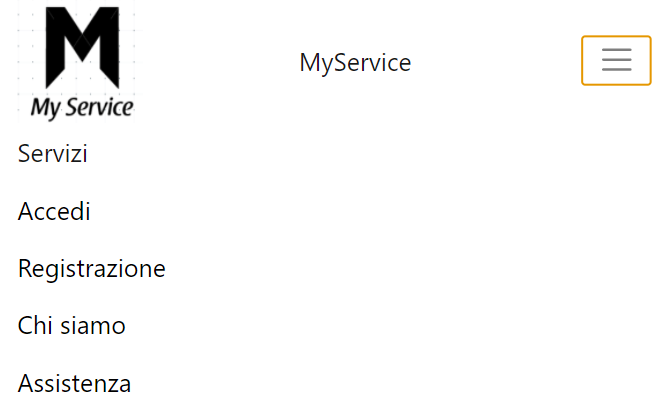
Per realizzare la navbar abbiamo effettuato un cambiamento rispetto alla versione predefinita di Bootstrap, infatti nel tag <nav> abbiamo inserito una classe aggiuntiva, cioè container-fluid, che consente alla navbar di occupare l’intera larghezza dello schermo. Pertanto, la navbar si adatta automaticamente alle dimensioni dello schermo, risultando visibile su qualsiasi tipo di dispositivo usato, come mostrato nella figura sottostante: 

Figura 17. Visualizzazione Navbar su dispositivi mobile

Il secondo contenitore contiene il carousel, una componente di Bootstrap che permette di scorrere una serie di contenuti, nel nostro caso immagini, come si può vedere in figura.

Immagine che contiene testo

Descrizione generata automaticamente Immagine che contiene palla, interni, fotografia, calcio

Descrizione generata automaticamente

Figura 18. Carousel 1 Figura 19. Carousel 2

Di seguito, viene mostrato il codice per realizzare il carousel.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 20. Codice di realizzazione del carousel

Bootstrap mette a disposizione una serie di carousel con indicatori, didascalie, dissolvenza incrociata e intervallo individuale. Nel nostro caso, si è deciso di utilizzare quella con intervallo individuale e per modificare il tempo di ritardo tra il passaggio automatico all’elemento successivo, si è aggiunto a data-interval il ritardo, espresso in millisecondi. Infine, nel div abbiamo anche specificato delle classi; la prima classe è carousel che prende specifiche dello slide show (larghezza, altezza e posizione dell’immagine), mentre la seconda classe è slide che permette di aggiungere una transizione CSS e un effetto di animazione.

Il terzo contenitore include un altro carousel, ma in questo caso riguarda le card e verrà spiegato nel dettaglio nella sezione 3.3.3.

Il quarto contenitore contiene la sezione di login per gli utenti abbonati e provider, nella quale gli utenti possano inserire le loro credenziali, username e password, per accedere alla loro area personale.

Nella figura seguente viene mostrato la sezione di login.

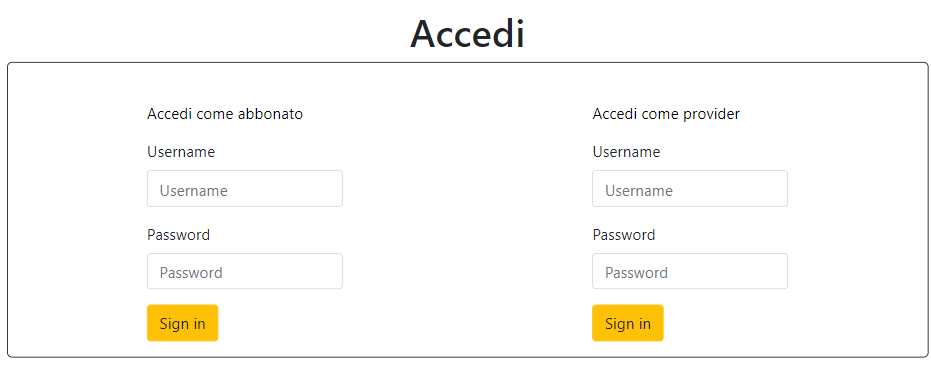


Figura 21. Form di accesso

Di seguito, viene mostrato il codice di realizzazione.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 22. Codice per realizzare la sezione di login

Per realizzare l’interfaccia grafica della figura 21, abbiamo modificato lo stile predefinito di Bootstrap, aggiungendo alla classe container l’attributo justify-content con valore space-around in style, permettendo così di posizionare le sezioni di login l’una accanto all’altra. Un altro elemento aggiunto è l’attributo display con valore flex, che permette alle sezioni di login di allargarsi o restringersi per evitare sovrapposizioni o eccedenze, sempre consentendo all’interfaccia di adattarsi ai vari dispositivi.

Il quinto contenitore contiene la sezione di registrazione, anche questa spiegata nel dettaglio successivamente.

Il sesto contenitore e l’ultimo contengono rispettivamente la sezione Chi siamo e la sezione di footer. Per esse non è stato usato codice predefinito da Bootstrap, ma è stato sufficiente utilizzare i tag html. Nella figura seguente viene mostrata la sezione chi siamo.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 23. Sezione chi siamo

Di seguito, viene mostrato il codice di realizzazione.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 24. Codice di realizzazione della sezione chi siamo

Mentre nella figura seguente viene mostrato il footer.



Figura 25. Footer

In basso, viene mostrato il codice per la realizzazione del footer.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 26. Codice di realizzazione della sezione footer

### **3.3.2 Multiple Step Form**

Il multiple step form, oggi sempre più usato, è un form lungo che viene diviso in più pezzi per permettere agli utenti di completare le loro informazioni in step più piccoli, così da creare un’esperienza utente positiva e meno scoraggiante. Nelle figure seguenti mostreremo i vari pezzi del form.

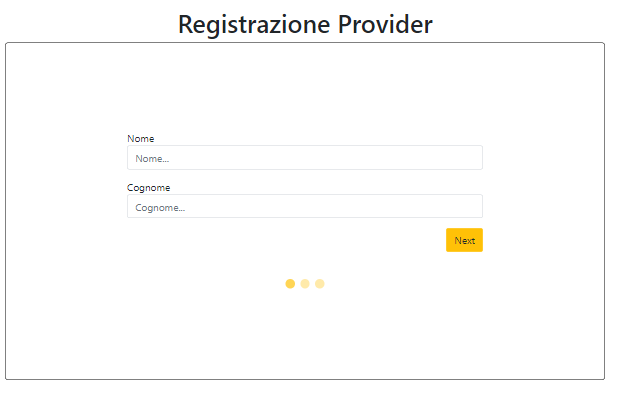


Figura 27. Step Form 1

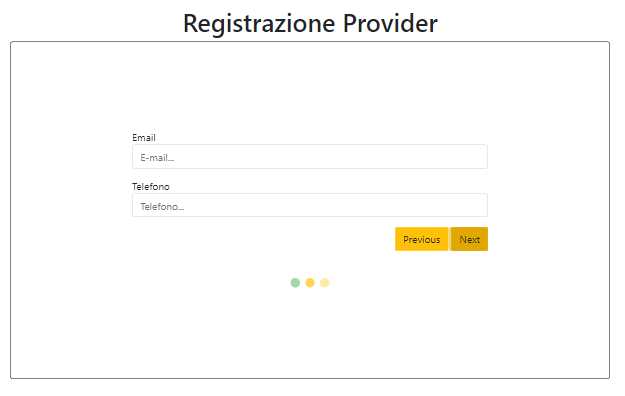


Figura 28. Step Form 2

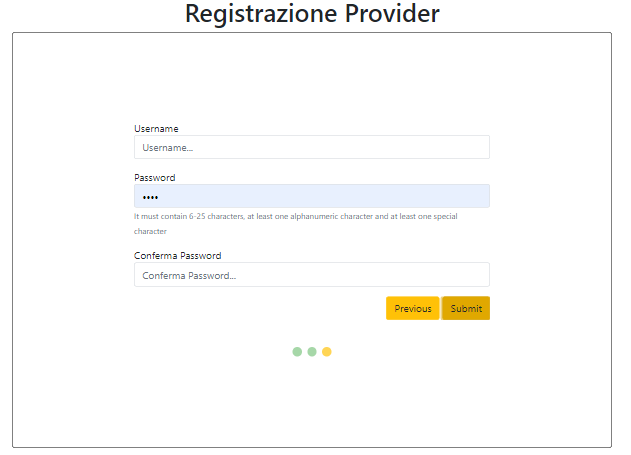


Figura 29. Step Form 3

Per realizzare il multiple step form ci sono 3 passi da compiere. Il primo è dare una struttura ai vari pezzi del form.

Qui di seguito viene mostrato il codice HTML.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 30. Codice HTML per realizzare il multiple step form

Analizzando il codice HTML, si può notare che sono state aggiunte delle funzioni all’attributo onblur[[36]](#footnote-36), le quali permettono di controllare il formato delle parole che vengono inserite, di modo che, in caso di errore, l’utente venga informato attraverso il cambiamento dell’interfaccia grafica, come accade ad esempio in figura 30.

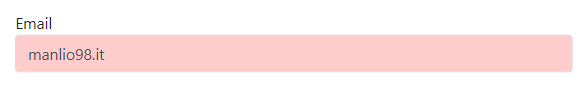


Figura 31. In caso di errore nel form

Il secondo passo per realizzare il multiple step form è quello di inserire il codice CSS per dare uno stile ai vari pezzi del form. Di seguito, viene mostrato il codice CSS.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 32. Codice CSS

Un fattore da notare nella figura 31 è il selettore tab, perché al suo interno ha come attributo display con valore none, che permette ai vari pezzi del form di nascondersi senza essere eliminati o ricreati ed è normalmente usato con JavaScript per realizzare tutto ciò, come vedremo al prossimo passo.

L’ultimo passo appunto è JavaScript, che permette di cambiare in modo dinamico una pagina web.

Di seguito viene mostrato il codice JavaScript.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 33. Codice JavaScript

La funzione showTab ha il compito di visualizzare il form specificato come parametro, infatti, l’elemento in questione passa da display none a display block, cioè visibile. La funzione nextPrev, invece, ha il compito di decidere quale parte del form visualizzare, infatti, una volta effettuato il controllo dei campi, essa mette a display none l’attuale pezzo di form e aumenta o decrementa di una posizione, cioè passa al precedente pezzo di form o al successivo in base al pulsante premuto dall’utente. Infine, nell’ultimo pezzo di form vengono inviate le informazioni alla servlet in questione.

### **3.3.3 Carousel-Card**

Il carousel-Card è una giostra di immagini che si susseguono, l’una dopo l’altra. È stata realizzata nel nostro sito perché permette di risparmiare spazio in una pagina web, e in più dà la possibilità all’utente di scorrere tra le varie immagini, senza un uso eccessivo del mouse. Nella figura seguente viene mostrato il Carousel-Card realizzato.



Figura 34. Carousel-Card

Per realizzare il carousel-card è stato usato il plug-in jQuery[[37]](#footnote-37), per implementarlo bisogna effettuare alcuni passi. Il primo passo è quello di avvolgere i nostri articoli in un contenitore con classe owl-carousel, come è possibile vedere in figura 34.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 35. Codice HTML

Una volta effettuato ciò, è possibile utilizzare le funzioni messe a disposizione dal plug-in. Nella figura seguente viene mostrato il codice JQuery.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 36. Codice JQuery

Con il plug-in è possibile determinare alcuni comportamenti da dare alle immagini. Ad esempio, l’attributo autoplay con valore true consente alle immagini di scorrere automaticamente, mentre l’attributo responsive consente di adattare il carousel a qualsiasi schermo, offrendo la possibilità di scegliere, in base alla dimensione dello schermo, quanti elementi vogliamo visualizzare.

**Capitolo 4**

# **4. Integrare il pagamento PayPal nella web app**

Il pagamento è una parte cruciale di un sito di e-commerce e PayPal è il portale di pagamento online più popolare e affidabile al mondo. PayPal consente ai clienti di acquistare beni o, come nel nostro progetto, servizi utilizzando le carte di credito o il saldo PayPal. Esso fornisce anche un SDK maturo che aiuta i programmatori a integrare la soluzione di pagamento PayPal in un sito web.

In questo capitolo spiegheremo i passi che sono stati effettuati per integrare il pagamento PayPal in un’applicazione web.

## **4.1 Tipico Flusso di lavoro Checkout-Payment**

Prima di passare alla parte implementativa, diamo un’occhiata alla seguente immagine per comprendere il flusso di lavoro standard del check-out al processo di pagamento su un sito di e-commerce.

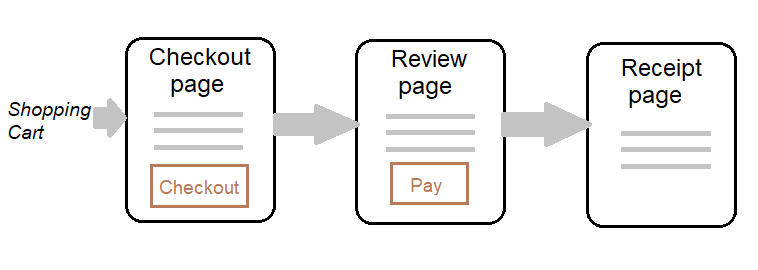


Figura 37. Flusso di lavoro Checkout-Pagamento

Dal carrello un cliente procede al pagamento. La pagina Checkout mostra i dettagli dell’ordine, comprese le tasse, le spese di spedizione e l’importo totale. Quando il cliente fa clic sul pulsante Checkout nella pagina web, verrà reindirizzato a una pagina PayPal che consente al cliente di scegliere il pagamento utilizzando il suo saldo PayPal o la sua carta di credito. Una volta approvata, la pagina di revisione consente al cliente di rivedere il costo totale, l’indirizzo di spedizione e altre dati inseriti. Quando il cliente clicca il pulsante Paga nella pagina Revisione, il pagamento verrà eseguito e verrà visualizzata la pagina Ricevuta.

## **4.2 Passi iniziale per usare l’API**

PayPal fornisce un Software Development Kit che supporta molti linguaggi di programmazione e aiuta i programmatori a scrivere codice per integrare il pagamento PayPal in un’applicazione esistente. L’API di PayPal comunica con il server PayPal tramite i servizi Restful, ma è un’API di alto livello, il che significa che non bisogna occuparsi direttamente dei servizi web.

Per poter usare i servizi web di PayPal bisogna seguire una specifica procedura. Il primo passo è creare un account Sandbox PayPal, che viene utilizzato per testare il pagamento durante lo sviluppo. Per creare quest’account bisogna accedere al portale per sviluppatori PayPal con il conto business PayPal e cliccare sul pulsante Crea app per ricevere ID cliente e il codice segreto, che verranno utilizzati nel codice Java per comunicare con il server PayPal. Nella figura seguente viene mostrato ID cliente e il codice segreto usato. [15]

Immagine che contiene testo

Descrizione generata automaticamente

Figura 38. Client ID e Secret code

Un altro passo da effettuare è aggiungere la dipendenza REST SDK di PayPal all’interno del progetto, per utilizzare le classi messe a disposizioni da PayPal. Nella figura seguente viene mostrata la libreria inserita. [15]

Immagine che contiene testo

Descrizione generata automaticamente

Figura 39. Libreria Rest-api-sdk

## **4.3 Come Integrare L’API di PayPal**

In questa sezione mostreremo in dettaglio le classi e le servlet utilizzate per integrare il pagamento PayPal all’interno del progetto.

Seguendo il flusso di lavoro mostrato nella sezione 4.1, creiamo una pagina web che presenta i dettagli (nome servizio e costo) del servizio all’utente. Quando l’utente clicca sul pulsante abbonati, una servlet Java gestirà la richiesta, descritta di seguito. Nella figura seguente viene mostrata la pagina web Register-Abbonamento, in cui si può visualizzare il nome e il costo del servizio che verranno inviati ai server di PayPal.

Immagine che contiene tavolo

Descrizione generata automaticamente

Figura 40. Pagina Register-Abbonamento

Prima che il pagamento venga eseguito, PayPal verifica i dettagli della transazione, per cui inviamo le informazioni del servizio (nome servizio e costo) ai server PayPal al fine di ottenere un link di approvazione per la validità della transazione. La servlet che si occupa di inviare le informazioni è la seguente.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 41. Servlet AutorizzaPagamento

Analizzando il codice della figura 41, la servlet legge i dettagli del servizio dalla pagina Register-Abbonamento e chiama il metodo authorizePayment della classe PaymentServices per la verifica. Se le informazioni sui dettagli del servizio sono valide, PayPal restituisce un URL di approvazione e l’utente viene reindirizzato a questo URL, che è l’interfaccia di accesso di PayPal. Nel caso in cui le informazioni fornite non siano valide o si siano verificati errori, viene generata un’eccezione PayPalRestException e l’utente visualizzerà una pagina di errore con il codice seguente:

Immagine che contiene testo

Descrizione generata automaticamente

Figura 42. Pagina di errore

La servlet AutorizzaPagamento utilizza la classe PaymentServices, che funge da livello intermedio tra le servlet Java e il server PayPal e incapsula le chiamate API all’SDK PayPal. Ecco il codice della classe PaymentServices.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 43. Classe PaymentServices

Come possiamo notare, abbiamo inserito il Client ID e il Client Secret estrapolati dall’account sandbox PayPal creato.

Oltre al metodo pubblico authorizePayment, la classe PaymentServices contiene altri metodi privati che vengono invocati internamente alla classe. Il primo metodo privato è getRedirectURLs. [16] Ecco il codice:

Immagine che contiene testo

Descrizione generata automaticamente

Figura 44. Metodo getRedirectURLs

Qui bisogna specificare due URL a cui PayPal reindirizzerà l’utente durante la procedura di pagamento. L’utente verrà reindirizzato all’URL di annullamento se sceglie di annullare il pagamento oppure, se l’utente accetta di continuare, verrà reindirizzato all’URL di restituzione che, tramite la servlet ReviewPaymentServlet lo porterà alla pagina di revisione.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 45. Check-out di PayPal

Il secondo metodo privato è getTransactionInformation, [16] che presenta il codice seguente:

Immagine che contiene testo

Descrizione generata automaticamente

Figura 46. Metodo getTransactionInformation

Come possiamo vedere, questo metodo serve a inviare i dettagli del servizio a PayPal in termini di API, come il costo di spedizione, l’importo parziale, la tassa, il nome del servizio e il codice di valuta. Tutti questi dettagli verranno poi visualizzati nella pagina di pagamento PayPal, come mostrato nella seguente figura.

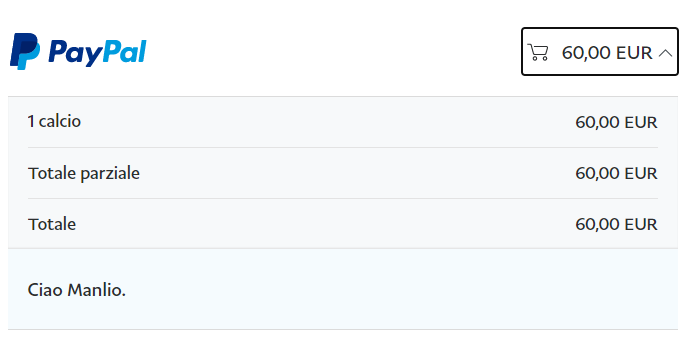


Figura 47. Pagina pagamento PayPal

Il terzo metodo privato è getApprovalLink, [16] di cui leggiamo di seguito il codice.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 48. Metodo getApprovalLink

Questo metodo analizza l’oggetto Payment restituito da PayPal per trovare l’URL di approvazione nella risposta JSON, ma grazie a PayPal Rest SDK, non dobbiamo trattare direttamente con JSON.

In seguito, ci soffermeremo su alcuni metodi pubblici della classe PaymentServices.

Ritornando alla figura 45, quando l’utente clicca su Continua nella pagina di PayPal, viene visualizzata la pagina di revisione, che consente all’utente di rivedere le informazioni di pagamento prima di effettuare l’acquisto o, nel nostro caso, abbonarsi a un servizio. La servlet ReviewPaymentServlet in figura, ha il compito di inviare le informazioni di pagamento alla pagina web revisione.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 49. Servlet ReviewPaymentServlet

Come possiamo vedere dalla figura 49, la servlet cattura i valori di PayerID e paymentID inviati da PayPal, dopodiché richiama il metodo getPaymentDetails della classe PaymentServices per ottenere i dettagli di pagamento memorizzati da PayPal e salva in sessione due oggetti: PayerInfo e Transaction. Queste informazioni verranno visualizzate nella pagina revisione, alla quale la servlet reindirizzerà l’utente. Ecco il metodo getPaymentDetails. [17]

Immagine che contiene testo

Descrizione generata automaticamente

Figura 50. Metodo getPaymentDetails

Questo metodo si collega semplicemente al server PayPal per ottenere un oggetto Payment basato sull’ID inviato come parametro. Nella figura seguente viene mostrata la pagina revisione.

Immagine che contiene tavolo

Descrizione generata automaticamente

Figura 51. Pagina Review-Payment

L’utente reindirizzato alla pagina revisione (*Fig.51*) potrà decidere se effettuare l’abbonamento oppure tornare indietro. Se l’utente clicca sul pulsante Paga ora, viene richiamata la servlet EsecuzionePagamento, ecco il codice.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 52. Servlet EsecuzionePagamento

Questa servlet acquisisce paymentID e PayerID passati dalla pagina revisione e richiama il metodo executePayment della classe PaymentServices, [16] che è codificato come di seguito.

Immagine che contiene interni, screenshot, tavolo, uccello

Descrizione generata automaticamente

Figura 53. Metodo executePayment

Infine, se il pagamento viene eseguito correttamente, la servlet EsecuzionePagamento inoltra la richiesta alla homepage e l’utente potrà inserire le credenziali per usufruire del servizio.

## **4.4 Passare dalla modalità sandbox alla modalità live**

Dopo aver testato il processo di pagamento, è necessario passare dalla modalità sandbox alla modalità live, in modo che gli utenti possano pagare con denaro reale. Per fare questo dobbiamo accedere al portale di PayPal dedicato agli sviluppatori e creare un app live, [18] per poi sostituire ClientID e Client Secret nella classe PaymentServices, con i valori ottenuti dall’app creata.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 54. Passare dalla modalità sanbox alla modalità live

**Capitolo 5**

# **5. Un piccolo esempio della web app**

In questo capitolo, viene fornito al lettore un piccolo esempio del sito web realizzato per facilitare la comprensione anche agli utenti meno esperti in ambito informatico.

## **5.1 Creazione Servizio**

Per creare un servizio, l’utente deve prima registrarsi come provider.



Figura 55. Navbar

Per farlo, l’utente clicca su registrazione e viene reindirizzato alla pagina mostrata in figura 56 per completare la registrazione.

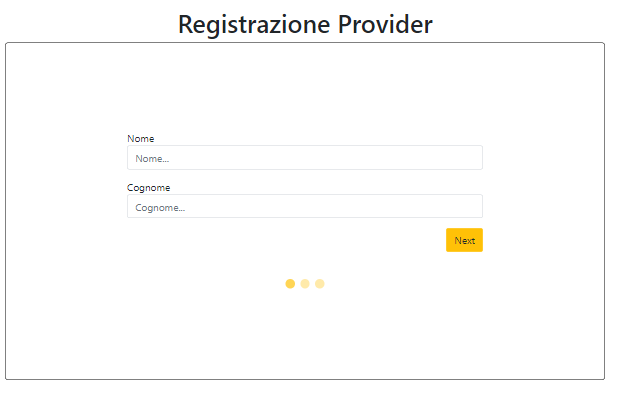


Figura 56. Sezione registrazione

Una volta inserite le informazioni necessarie (nome, cognome, e-mail, telefono, username, password) e cliccato il pulsante submit, l’utente potrà passare alla fase successiva della registrazione, cioè alla creazione vera e propria del servizio, come mostrato nella figura 57.

Immagine che contiene tavolo

Descrizione generata automaticamente

Figura 57. Sezione Crea Servizio

L’utente inserisce le informazioni (nome servizio, costo, descrizione) riguardanti il servizio e, cliccando sul pulsante Completa Registrazione, passa al caricamento dell’immagine.



Figura 58. Sezione Carica Immagine

Caricata l’immagine, potrà vedere l’anteprima cliccando sul pulsante carica immagine oppure, cliccando su registrazione completata, l’utente verrà reindirizzato alla homepage per poter accedere alla sua area riservata e iniziare le operazioni di inserimento, modifica ed eliminazione.

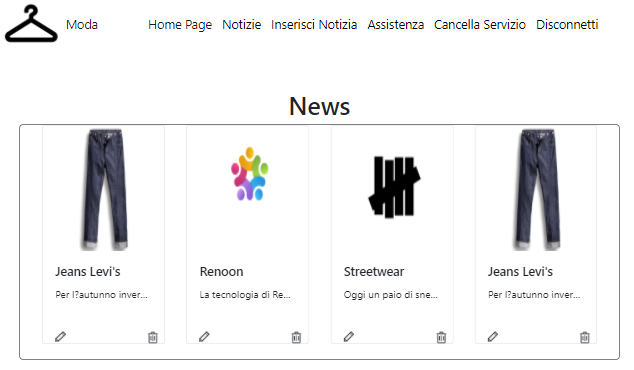
****

Figura 59. Homepage Provider

## **5.2 Abbonarsi al servizio**

Per abbonarsi a un servizio, l’utente deve cliccare il pulsante Abbonati sul servizio interessato, come mostrato in figura 60.

Immagine che contiene tavolo

Descrizione generata automaticamente

Figura 60. Sezione Servizi

Una volta cliccato, l’utente viene reindirizzato in una pagina web per completare l’abbonamento, come mostrato in figura 61.

Immagine che contiene tavolo

Descrizione generata automaticamente

Figura 61. Sezione abbonati

L’utente inserisce le informazioni necessarie (nome, cognome, username, password) per abbonarsi al servizio e clicca sul pulsante abbonati per iniziare la fase di pagamento. Qui l’utente viene reindirizzato alla pagina di PayPal per inserire i dati d’accesso, come mostrato in figura 62.

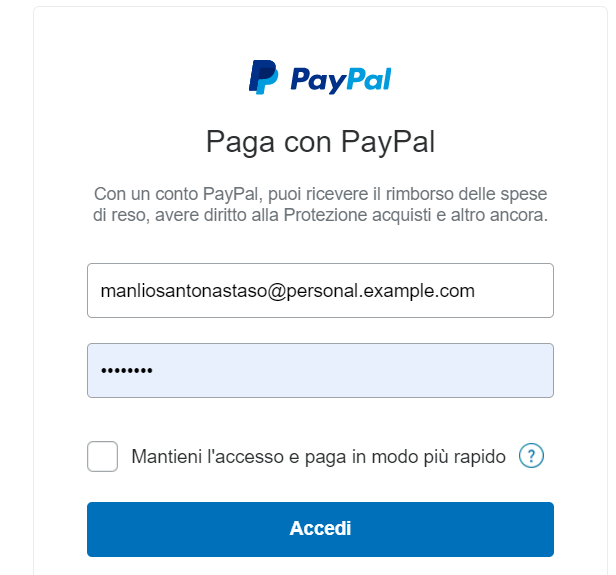


Figura 62. Portale d’accesso di PayPal

Dopo aver effettuato l’accesso, l’utente dovrà decidere se continuare con il pagamento oppure annullarlo, tramite i due pulsanti continua o annulla, come mostrato in figura 63.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 63. Check-out PayPal

Cliccando sul pulsante continua, l’utente viene reindirizzato alla pagina web mostrata in figura 64 e decide anche in questo caso se effettuare il pagamento.

Immagine che contiene tavolo

Descrizione generata automaticamente

Figura 64. Pagina web Revisione

Cliccato il pulsante Paga ora, l’utente effettuerà il pagamento e il sistema lo reindirizzerà alla homepage per permettergli di accedere alla sua area riservata e iniziare ad usufruire del servizio.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 65. Homepage Abbonato

# **Conclusioni**

Il sito web porta alle aziende provider numerosi benefici, come la riduzione dei costi, l’internazionalizzazione del brand e un conseguente incremento del fatturato. La progettazione e la creazione del sito web, oggetto di questo elaborato, è iniziata con la realizzazione del database, per proseguire con la creazione del back-end e le relative funzioni predisposte e, infine, del front-end che permette l’utilizzo del sito da parte dell’utente finale. Al fine di ottimizzare la competitività del sito web, ci si è focalizzati su diversi fattori. Innanzitutto, si è pensato ad attrarre l’utente con un front-end semplice e visivamente accattivante; successivamente, si è ampliata da un lato la visibilità sul Web, attraverso l’integrazione dei social, dall’altro la compatibilità con diversi dispositivi mobili, tramite un template con design responsive realizzato grazie al framework Bootstrap. Per incrementare i volumi degli abbonamenti, inoltre, si è pensato al modello di vendita online detto *guest-checkout*, grazie al quale l’utente può effettuare l’abbonamento senza dover prima eseguire la registrazione. Infine, per garantire la massima affidabilità durante la fase finale di acquisto, si è scelto di offrire all’utente la modalità di pagamento più diffusa e utilizzata online, ovvero PayPal.

Raggiunti questi obiettivi minimi, si intravedono margini di miglioramento su diversi fronti. Oltre a rendere il sito ancora più semplice dal punto di vista grafico nella parte di front-end, è possibile aggiungere ulteriori funzionalità per l’utente abbonato e l’utente provider nella parte di back-end. Per l’utente abbonato è, infatti, possibile aggiungere l’operazione di “disdetta”, in modo da poter annullare l’abbonamento in qualsiasi momento, mentre per l’utente provider, apportando alcune modifiche al database, è possibile inserire l’operazione “ulteriore servizio”, per aggiungere un nuovo servizio.

Date le ormai consolidate tendenze, che premiano l’approccio *Mobile First* nella creazione di siti web e confermano il sempre più massiccio utilizzo di app dedicate al posto degli stessi siti web, risulta chiaro che il presente progetto potrà essere sviluppato in futuro con la creazione di un’app per dispositivi Android e iOS, proponendo magari un metodo di creazione ibrido che consiste nell’impacchettare l’applicazione web attraverso dei framework PhoneGap o Cordova e renderla disponibile nei vari Store delle diverse piattaforme.

# **Bibliografia**

**Che cos’è un sito web**

|  |  |
| --- | --- |
| [1] | [https://www.ionos.it/digitalguide/siti-web/creare-siti/che-cose-davvero-una-pagina-web/.](https://www.ionos.it/digitalguide/siti-web/creare-siti/che-cose-davvero-una-pagina-web/.%20)  **Storia HTML** |
| [2] | [http://elearning.informatica.unisa.it/el-platform/pluginfile.php/37713/mod\_folder/content/0/L00a%20-%20Introduzione.pdf?forcedownload=1.](http://elearning.informatica.unisa.it/el-platform/pluginfile.php/37713/mod_folder/content/0/L00a%20-%20Introduzione.pdf?forcedownload=1.%20)  **Evoluzione del Web** |
| [3] | [https://www.manthone.edu.it/wp-content/uploads/2016/04/Evoluzione-del-Web-Dal-1.0-al-4.0.pdf.](https://www.manthone.edu.it/wp-content/uploads/2016/04/Evoluzione-del-Web-Dal-1.0-al-4.0.pdf.%20)  **Importanza di avere un sito web** |
| [4] | [https://www.comunicarekairos.it/blog/social-web-marketing/32-importanza-avere-sito-web.html#:~:text=In%20Italia%20il%20numero%20di,di%20et%C3%A0%20presa%20in%20considerazione..](https://www.comunicarekairos.it/blog/social-web-marketing/32-importanza-avere-sito-web.html%23:~:text=In%20Italia%20il%20numero%20di,di%20et%C3%A0%20presa%20in%20considerazione..%20)  **Requisiti di un sistema di gestione di base di dati** |
| [5] | [https://www.ionos.it/digitalguide/hosting/tecniche-hosting/sistema-di-gestione-di-base-di-dati-dbms/.](https://www.ionos.it/digitalguide/hosting/tecniche-hosting/sistema-di-gestione-di-base-di-dati-dbms/.%20)  **Bootstrap** |
| [6] | [https://www.html.it/pag/44806/introduzione-a-bootstrap/.](https://www.html.it/pag/44806/introduzione-a-bootstrap/.%20)  **Framework** |
| [7] | [https://www.idearia.it/blog/10-migliori-framework-front-end/.](https://www.idearia.it/blog/10-migliori-framework-front-end/.%20)  **HTML** |
| [8] | [http://elearning.informatica.unisa.it/el-platform/pluginfile.php/38298/mod\_folder/content/0/L03%20HTML%20intro.pdf?forcedownload=1.](http://elearning.informatica.unisa.it/el-platform/pluginfile.php/38298/mod_folder/content/0/L03%20HTML%20intro.pdf?forcedownload=1.%20)  **CSS** |
| [9] | <http://elearning.informatica.unisa.it/el-platform/pluginfile.php/39015/mod_folder/content/0/L15%20css.pdf?forcedownload=1.>  **API PayPal** |
| [10] | <https://www.paypalobjects.com/it_IT/html/IntegrationCentre/api.htm.>  **Modello Concettuale** |
| [11] | [http://elearning.informatica.unisa.it/el-platform/pluginfile.php/45332/mod\_resource/content/1/Capitolo-7-Progettazione-Concettuale.pdf.](http://elearning.informatica.unisa.it/el-platform/pluginfile.php/45332/mod_resource/content/1/Capitolo-7-Progettazione-Concettuale.pdf.%20)  **Modello Relazionale** |
| [12] | [http://elearning.informatica.unisa.it/el-platform/mod/resource/view.php?id=8123.](http://elearning.informatica.unisa.it/el-platform/mod/resource/view.php?id=8123.%20)  **DDL** |
| [13] | <http://www2.ing.unipi.it/~a080066/didattica/BD/7%20-%20SQL%20Costrutti%20DDL.pdf.>  **Architettura a 3 livelli** |
| [14] | [http://elearning.informatica.unisa.it/el-platform/pluginfile.php/45398/mod\_folder/content/0/01.System%20Design%20-%20System%20Decomposition.pdf?forcedownload=1.](http://elearning.informatica.unisa.it/el-platform/pluginfile.php/45398/mod_folder/content/0/01.System%20Design%20-%20System%20Decomposition.pdf?forcedownload=1.%20)  ***Riferimenti***  **Integrare PayPal** |
| [15] | [https://developer.paypal.com/docs/business/get-started/.](https://developer.paypal.com/docs/business/get-started/)  **Elabora pagamenti PayPal** |
| [16] | [https://developer.paypal.com/docs/api/quickstart/payments/.](https://developer.paypal.com/docs/api/quickstart/payments/)  **Classe Payment** |
| [17] | [https://paypal.github.io/PayPal-PHP-SDK/docs/class-PayPal.Api.Payment.html](https://paypal.github.io/PayPal-PHP-SDK/docs/class-PayPal.Api.Payment.html.)  **Test and go live** |
| [18] | [https://developer.paypal.com/docs/business/test-and-go-live/.](https://developer.paypal.com/docs/business/test-and-go-live/) |

1. Un fornitore di servizi Internet [↑](#footnote-ref-1)
2. Un’applicazione software in grado di gestire le richieste da parte dei clienti. [↑](#footnote-ref-2)
3. Il linguaggio di marcatura è un insieme di regole che descrivono i meccanismi di rappresentazione o layout di un testo. [↑](#footnote-ref-3)
4. È un'architettura logica di supporto sul quale un software può essere progettato e realizzato. [↑](#footnote-ref-4)
5. Application Programming Interface, sono set di definizioni e protocolli con i quali vengono realizzati e integrati software applicativi. [↑](#footnote-ref-5)
6. È un programma con il quale l'utente interagisce indirettamente. [↑](#footnote-ref-6)
7. La parte di un programma visibile all'utente con cui egli può interagire [↑](#footnote-ref-7)
8. Uniform Resource Locator  [↑](#footnote-ref-8)
9. è un collegamento ipertestuale tra due pagine web, che serve a far spostare l’utente da un punto A a un punto B. [↑](#footnote-ref-9)
10. Responsabile del progetto e della gestione di un sito web [↑](#footnote-ref-10)
11. Really Simple Syndication [↑](#footnote-ref-11)
12. eXtensible Markup Language [↑](#footnote-ref-12)
13. È un risultato di una ricerca che non si limita a mostrare solamente il titolo e il contenuto, ma mostra altre informazioni più dettagliate. [↑](#footnote-ref-13)
14. https://httpd.apache.org/ [↑](#footnote-ref-14)
15. https://www.lighttpd.net/ [↑](#footnote-ref-15)
16. https://nginx.org/en/ [↑](#footnote-ref-16)
17. DataBase Management System [↑](#footnote-ref-17)
18. Structured Query Language [↑](#footnote-ref-18)
19. https://www.mysql.com/it/ [↑](#footnote-ref-19)
20. https://mariadb.org/ [↑](#footnote-ref-20)
21. https://launchpad.net/drizzle [↑](#footnote-ref-21)
22. https://www.percona.com/software/mysql-database/percona-server [↑](#footnote-ref-22)
23. Atomicità Coerenza Isolamento Durabilità [↑](#footnote-ref-23)
24. è un programma scritto in SQL o in altri linguaggi, mantenuto nel database stesso. [↑](#footnote-ref-24)
25. https://www.mongodb.com/ [↑](#footnote-ref-25)
26. JavaScript Object Notation [↑](#footnote-ref-26)
27. https://redis.io/ [↑](#footnote-ref-27)
28. Menu a tendina [↑](#footnote-ref-28)
29. Finestra di aiuto [↑](#footnote-ref-29)
30. Menu a fisarmonica, ogni elemento può essere cliccato per rilevare il contenuto. [↑](#footnote-ref-30)
31. https://purecss.io/ [↑](#footnote-ref-31)
32. Entità-relazione. [↑](#footnote-ref-32)
33. Query memorizzate con un proprio nome che possono essere considerate simili a tabelle virtuali. [↑](#footnote-ref-33)
34. Data Definition Language [↑](#footnote-ref-34)
35. Data Access Object [↑](#footnote-ref-35)
36. si verifica quando un oggetto perde lo stato attivo. Ad esempio, quando l’utente lascia un campo del modulo [↑](#footnote-ref-36)
37. https://github.com/OwlCarousel2/OwlCarousel2 [↑](#footnote-ref-37)