Elaborato

Table of content

1. Struttura Rete

Introduzione

La richiesta e' quella di progettare una rete e una applicazione web per una catena di negozi che vuole gestire tutti gli ordini, gli articoli presenti, gli utenti e i dipendenti presenti all'interno dei singoli negozi. La possibilita' di acquistare prodotti online per ora non esiste per i seguenti motivi:

1. i negozi non sono molti e il personale e' ristretto quindi non riuscirebbe a stare dietro a tutti gli ordini in caso di molta affluenza
2. la pandemia di Covid si e' appena conclusa e la gente puo' tornare, finalmente, alla vita normale, ormai quasi dimenticata, e il proprietario della catena di negozi ha deciso di stimolare questo ritorno non volendo un sito web attuo all'acquisto, quindi un modo piu' distaccato di fare acquisti. Facendo cosi' il cliente e' obbligato ad andare in negozio, parlare con qualcuno del reparto clienti e quindi conversare. Inoltre ci sono delle console libere all'uso con dei videogiochi in prova e anche altri dispositvi che si possono provare, il fine e' quello di creare una cerchia di clienti che andando li' si diverte in compagnia e puo' giocare liberamente senza costi.

La catena di negozi e' di modeste dimensioni. Inizialmente era composta solo da un negozio centrale, e i dati per ;a gestione di articoli e personale erano salvati su fogli excel, poi per motivi di necessita ad espandersi sono stati aperti altri negozi "periferici" (2-3) e si vuole migliorare la gestione usando una applicazione e un database su cui salvare i dati. L'applicazione e' di tipo web e il server per il deploy e' situato nel negozio principale, il database, anch'esso presente solo nel negozio principale, contiene tutti i dati e i negozi secondari andranno a collegarsi al server centrale per accedere al database e creare ordini, controllare gli articoli presenti o aggiungere personale.

Architettura rete

- Per salvare i dati si utilizza un server con un database, usando il DBMS MariaDB che e' un fork di mysql completamente gratuito e open source in modo da evitare costi aggiuntivi.

- Per la pubblicazione del sito web si usa Apache2 installato su un computer con sistema operativo Linux Debian e per il deploy al pubblico il server verra' messo in DMZ.
- Per le mail viene utilizzato un mail server
- Tutti gli altri dispostivi della sottorete sono collegati via cavo ethernet (almeno Cat 6) per una connesione piu' veloce e stabile
- La struttura della rete e' la seguente:
  - Sezione server: Web Server e Mail Server, entrambi in DMZ
  - Database: database per salvare tutti i dati per la gestione dei negozi
  - Sezione assistenza: comprende i computer dei dipendenti adibiti all'assistenza, i loro computer devono essere configurati in modo tale che possano leggere le mail in entrata dal mail server
  - Sezione vendite: questa sezione deve poter fare ricerche rapide per controllare se in magazzino ci sono articoli rimasti e creare nuovi ordini, i computer si interfacciano col database tramite una applicazione adibita alla gestione interna
  - Sezione manager: comprende i dispostivi degli amministratori del negozio

Questa soluzione prevede la divisione della rete in molteplici sottoreti, ogni sottorete svolge un servizio differente. Il mail server e il web server sono in DMZ, dietro a un firewall che li protegge dall'esterno, in modo tale che possano interfacciarsi con l'esterno senza mettere in pericolo le altre parti della rete mentre il database e le altre sottoreti sono fuori dalla DMZ e dietro a un'altro firewall.

Dato che la rete non e' di grandi dimensioni e non ci sono molti computer viene usata una rete di tipologia C:

Router Interno

Rete	Inditizzo rete
Router	192.168.0.1/24
Sezione Vendite	192.168.2.0/24
Sezione Assistenza	192.168.3.0/24
Sezione Manager	192.168.4.0/24
Database	192.168.30.0/24
Router esterno	192.168.0.2

Router Esterno

Name	Tags
Router	192.168.0.2/24
Router interno	192.168.0.1
DMZ	192.168.50.0/24
Server	192.168.50.10/24
Mail server	192.168.50.20/24

I negozi periferici, quindi non quello centrale che ha anche il server e il database, occuperanno gli indirizzi IP in modo tale che, in reti diverse, non ci siano due IP uguale, in modo da non creare conflitti, cosicche' le reti di altri negozi possano connettersi tramite una VPN alla rete interna di quello principale per accedere al database.

Un esempio e' il seguente:

Router Interno Negozio 2

Rete	Inditizzo rete
Router	192.168.0.3/24
Sezione Vendite	192.168.12.0/24
Sezione Assistenza	192.168.13.0/24
Sezione Manager	192.168.14.0/24

Immagine della rete:

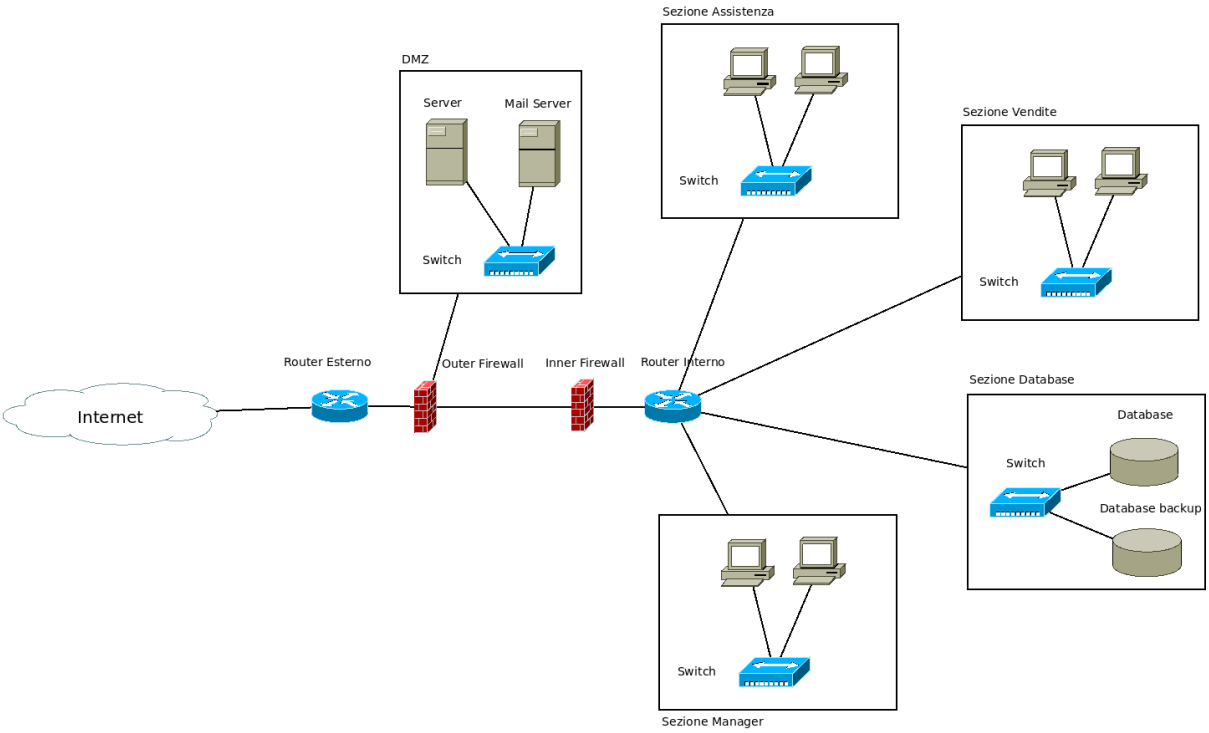
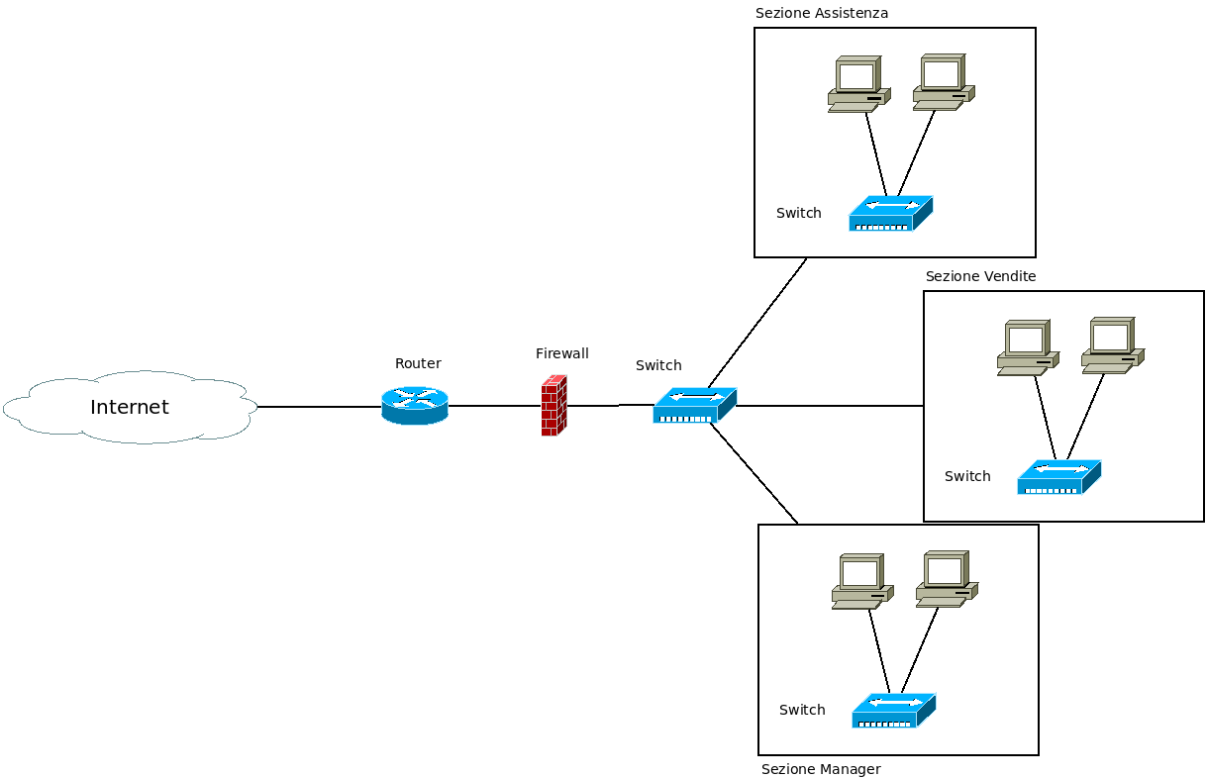


Immagine esempio rete di un negozio secondario:



Regole ACL dei firewall:

Tabella ACL firewall esterno

Numero	Azione	IP sorgente	Porta sorgente	IP destinatario	Porta destinatario
1	allow	any	any/TCP	192.168.50.10	80/TCP
2	allow	any	any/TCP	192.168.50.10	443/TCP
3	allow	any	any/TCP	192.168.50.20	993/TCP
4	allow	any	any/TCP	192.168.50.20	993/TCP
5	allow	192.168.4.0	any	any	any
6	deny	any	any	any	any

Tabella ACL firewall interno

Numero	Azione	IP sorgente	Porta sorgente	IP destinatario	Porta destinatario
1	allow	any	any/TCP	192.168.50.10	80/TCP
2	allow	any	any/TCP	192.168.50.10	443/TCP
3	allow	192.168.50.10	any/TCP	192.168.30.0	3306/TCP
5	allow	192.168.4.0	any/TCP	any	any/TCP
6	allow	192.168.3.0	any/TCP	192.168.50.20	993/TCP
7	deny	any	any/TCP	any	any/TCP

Caratteristiche Server

Il sito verra' erogato tramite una web application il cui deploy verra' affidato a un web server con sistema operativo Debian e tramite Apache, un HTTP server completamente open source, in modo che l'applicazione web possa essere vista tramite interner, ovviamente il server sara' in DMZ cosicche' ci si possa accedere dall'esterno. Il sito web sara' codificato in php per il backend e javascript, HTML, CSS per il frontend. Le informazioni verranno memorizzate in

un'altro computer Linux con un RDBMS MariaDb (fork di MySql) (a cui si potra' accedere tramite la porta 3306). Per quanto riguarda il Mail Server si e' deciso di usare il protocollo IMAP che permette di effettuare la sincronizzazione e inoltre si puo' usare in parallelo da piu' dispositivi, cosa utile dato che l'assistenza ha una sola mail ma viene utilizzata da piu' assistenti. Dato che privacy e sicurezza sono molto importanti e' stato acquistato un protocollo SSL da una Autorita' Certificante ed e' poi stato installato nel server Apache, le richieste ora potranno essere ricevute sulla porta 443 tramite il protocollo HTTPS.

Modalita' di comunicazione tra server e dispositivi

- Per la comunicazione tra i dispositvi degli utenti e il server si usa il protocollo HTTPS (sulla porta 443) e a livello di trasporto il protocollo TCP . Per usare HTTPS si deve prima acquistare una certificazione SSL o TLS da una Autorita' Certificante che esegue dei controlli e poi, se tutto va bene, rilascia la certificazione. Grazie al protocollo HTTPS e' garantita la sicurezza della comunicazione ei pacchetti vengono cifrati e garantisce la protezione dagli attacchi MiM (Man in the Middle) grazie allo scambio di chiavi e autenticazione.
- Per il mail server si utilizza il protocollo IMAP SSL (porta 993), anch'esso usa la certificazione SSL quindi garantisce la sicurezza cifrando i dati e l'autenticazione tra i due host comunicanti.
- Per permettere a sedi esterne di accedere al database o server in caso di problemi o bisogno di effettuare della manutenzione, quindi senza accedere al sito, si usa una VPN Site-to-Site configurata per collegare tramite un tunnel la rete di un negozio secondario al router esterno del negozio principale, quindi si garantisce l'accesso sia al server che al database. Una VPN e' una rete privata virtuale che garantisce privacy, anonimato e sicurezza e crea un canale di comunicazione logicamente riservato chiamato tunnel VPN, il tunnel collega un router di una rete ad uno di un'altra distaccata dalla prima, grazie a questo tunnel le due reti collegate si vedono come fossero una rete unica.

Gestione della sicurezza

- Firewall: sono stati usati due firewall per incrementare la sicurezza
  - Firewall esterno: posizionato esternamente quindi si affaccia a internet e filtra i pacchetti dall'esterno verso la DMZ e in uscita. E' firewall di livello 4, quindi stateful firewall, che memorizza lo stato delle connessioni in delle tabelle e le controlla, e' un po' piu' esigente a livello di prestazioni ma garantisce una sicurezza maggiore.
  - Firewall interno: posizionato dopo il firewall esterno, filtra i pacchetti che provengono dalla DMZ e in uscita dalle sottoreti interne. E' un firewall di tipo packet filter quindi piu' veloce dello stateful firewall, si limita pero' a garantire o negare l'accesso dei pacchetti da e verso determinati IP, non controlla lo stato della connessione. Questo firewall, secondo le regole definite nella tabella ACL permette solo ai pacchetti provenienti dal mail server e dal web server di entrare e solo ai pacchetti provenienti dalla sezione manager di uscire, bloccando tutti gli altri
- Database:
  - Le password di utenti e dipendenti vengono salvate nel database non in chiaro ma si esegue prima l'hashing di esse. Grazie a questa tecnica, se per caso, a seguito di un attacco informatico, un utente malevolo ottenga l'accesso al database, o a una parte di esso, non puo' vedere le password in chiaro ma trova solo delle stringhe di caratteri senza apparente senso. La riga di codice che permette di eseguire l'hasing di password (in php) e' la seguente:

    ```
    password_hash("password", PASSWORD_DEFAULT);
    ```

  - Il database è stato posizionato dopo il firewall interno e nella tabella ACL gli è stato consentito l'accesso solo dal il server, oltre che dagli altri dispostivi presenti nella rete interna. Ciò non lo rende immune dalle sql injection tramite il sito ma almeno se un utente prova ad accedere solo al database viene bloccato dal firewall.
  - Per tenere al sicuro i dati ogni fine settimana viene eseguito un backup nel database di backup, in modo tale che, in caso di guasto al database, i dati possano essere recuperati. Il backup viene eseguito in automatico grazie ad un cron job: uno script bash eseguito automaticamente ogni domenica grazie a crontab (su linux).
- L'autenticazione dei dipendenti avviene tramite email, password e un codice riportato dal badge, questo codice cambia ogni t minuti e lo stesso codice lo conosce anche il sevrer. Quando i dipendenti si loggano il codice inserito viene controllato e se e' giusto viene eseguito l'accesso. Grazie a questo metodo un utente esterno, che vuole entrare come admin nel sito, anche se ha ottenuto in precedenza email e password di un dipendente si trova

davanti un ostacolo in più e, dato che il codice cambia di continuo, fa certamente fatica a trovarlo tramite un attacco brute force.

- Il server usa la connessione tramite protocollo HTTPS, grazie ad esso viene garantita l'integrita' dei pacchetti, la certificazione del server, la cifratura dei pacchetti e l'autenticita' grazie allo scambio di chiavi.

2. Database

Analisi dello scenario

Si deve creare un database che gestisca una catena di negozi. Ogni negozio ha un indirizzo, un numero di telefono e una mail, dei dipendenti lavorano all'interno del negozio ed essi sono individuati attraverso alcune caratteristiche che sono: codice fiscale, nome, cognome, indirizzo, email, password, un manager che li coordina e un ruolo. Il negozio deve gestire degli ordini, ogni ordine si riferisce ad un articolo (che ha un nome, un prezzo e appartiene a una categoria), e dal suo acquirente, quindi un utente individuato dalle seguenti caratteristiche: nome, congnome email e password, ogni ordine deve avere una data e una garanzia di 2 anni.

Entita' estratte dall'analisi dello scenario

Negozi (id, indirizzo, numero di telefono, email)

PK = id

Dipendenti (codice_fiscale, nome, cognome, indirizzo, email, password, stipendio, manager, id_negozio, ruolo)

PK = codice_fiscale

FK = id_negozio REFERENCES Negozi(id)

FK = manager REFERENCES Dipendenti(codice_fiscale) ← chiave ricorsiva

Articoli (nome, prezzo, categoria, marca)

PK = nome

ArticoloImmagazzinatoNegozio (id, nome_articolo, id_negozio, quantita)

PK = id

FK = nome_articolo REFERENCES Articoli(nome)

FK = id_negozio REFERENCES Negozi(id)

Utente (id, nome, cognome, email, password)

PK = id

Ordine (id, nome_articolo, id_utente, data, garanzia, id_negozio)

PK = id

FK = nome_articolo REFERENCES Articoli(nome)

FK = id_utente REFERENCES Utenti(id)

FK = id_negozio REFERENCES Negozi(id)

Diagramma ER

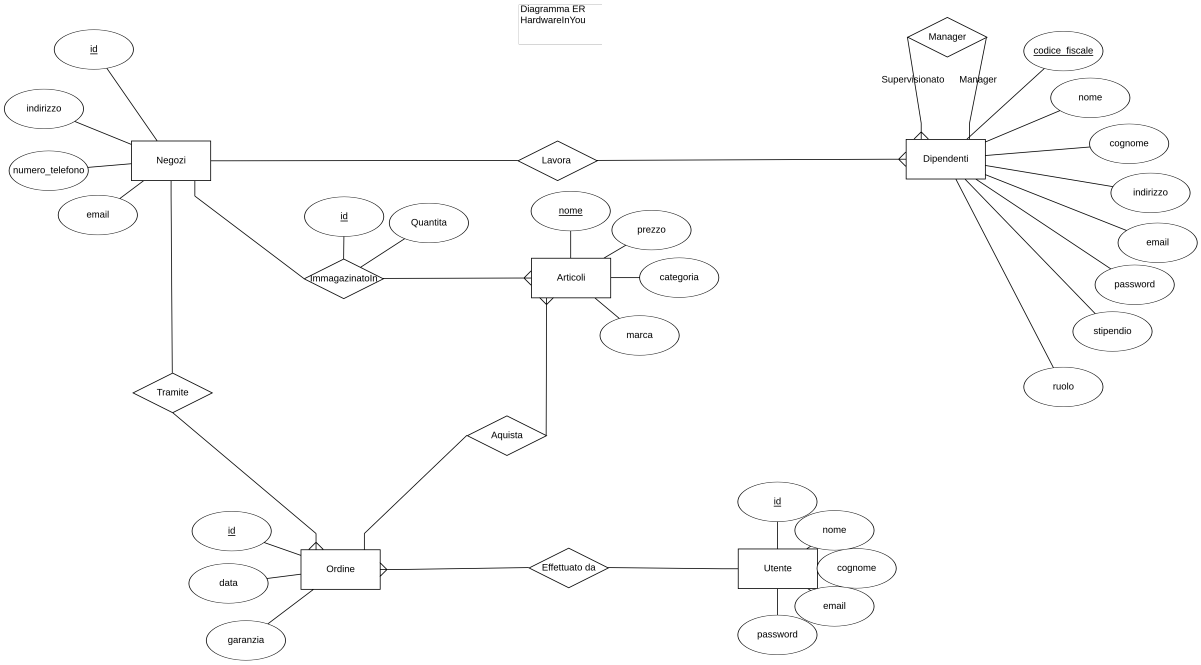


Tabelle estratte dopo la Normalizzazione

Negozio

Name	Type	Attribute
id	int	auto inc, primary key
indirizzo	varchar	not null
id_comune	int	foreign key
telefono	varchar	
email	varchar	

Dipendente

Name	Type	Attribute
codice_fiscale	varchar	primary key
nome	varchar	not null
cognome	varchar	not null
indirizzo	varchar	not null
id_comune	int	foreign key
email	varchar	not null, unique
password	varchar	not null
stipendio	double	
manager	varchar	foreign key
id_negozio	int	foreign key
id_ruolo	int	foreign key

Ruoli

Name	Type	Attribute

Name	Type	Attribute
id	int	primary key
ruolo	varchar	

Comuni

Name	Type	Attribute
id	int	primary key
nome	varchar	
id_provincia	int	foreign key

Province

Name	Type	Attribute
id	int	primary key
nome	varchar	
sigla	varchar	
id_regione	int	foreign key

Regioni

Name	Type	Attribute
id	int	primary key
nome	varchar	
id_stato	int	

Stati

Name	Type	Attribute
id	int	primary key
nome	varchar	

Ordine

Name	Type	Attribute	
id	int	primary key	
nome_pezzo	varchar	foreign key	not null
id_utente	int	foreign key	not null
data	date		
garanzia	int		
id_negozio	int	foreign key	not null

Articolo

Name	Type	Attribute
nome	varchar	primary key
prezzo	double	
id_marca	int	foreign key

Name	Type	Attribute
id_categoria	int	foreign key

ImmagazzinatoIn

Name	Type	Attribute
id	int	primary key
nome_pezzo	varchar	foreign key
id_negozio	int	foreign key
quantita	int	

Utente

Name	Type	Attribute
id	int	primary key
nome	varchar	
cognome	varchar	
email	varchar	
password	varchar	

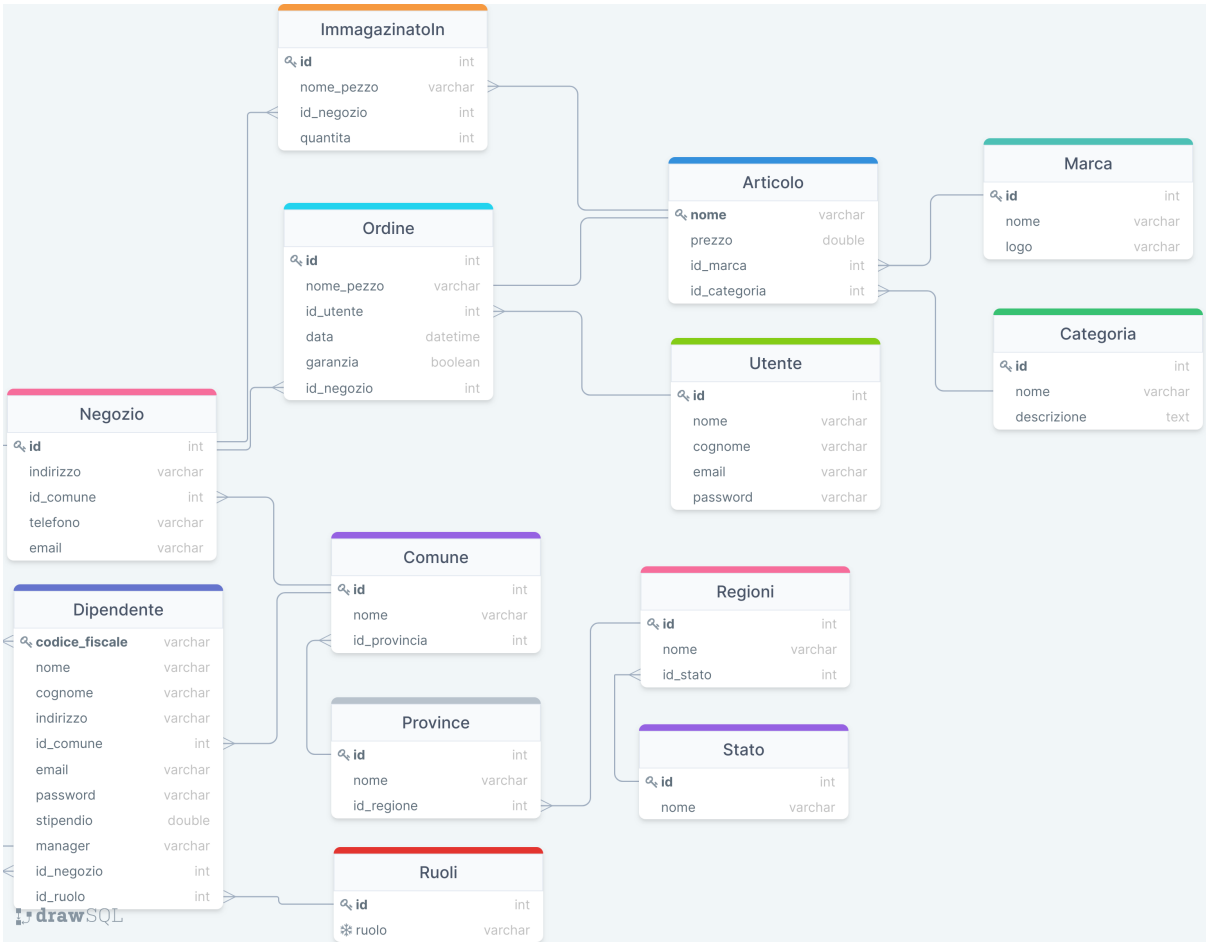
Marca

Name	Type	Attribute
id	int	primary key
nome	varchar	
logo	varchar	

Categoria

Name	Type	Attribute
id	int	primary key
nome	varchar	
descrizione	text	

Diagramma logico



3. Linguaggi di programmazione utilizzati

Per questo progetto sono stati usati linguaggi orientati alla programmazione web quindi HTML, CSS, Javascript, PhP e nel dettaglio:

Front end

Il front end e' la parte del sito visibile all'utente, quindi l'interfaccia utente, ed e' quella con cui l'utente interagisce per accedere alle funzionalita' del sito. Il front end e' responsabile dell'acquisizione di dati da parte dell'utente.

HTML

HTML (HyperText Markup Language) non e' un linguaggio di programmazione ma un linguaggio di markup, cioe' si usano i cosidetti tag (es. per i paragrafi) per definire la struttura della pagina web.

Esempio di codice HTML:

- Form di login:

```
<form action="login.php" method="POST">
        <img class="mb-4 login-img" id="img" src="../svg/user-solid.svg" alt=""/>

        <div class="form-floating">
            <input
                    type="email"
                    class="form-control"
                    id="email"
                    placeholder="Email address"
                    name="email"
                    required
            />
            <label for="email">Email address</label>
        </div>
        <div class="form-floating">
            <input
```

```
                    type="password"
                    class="form-control"
                    id="password"
                    placeholder="Password"
                    name="password"
                    required
                />
                <label for="password">Password</label>
            </div>

            <div class="badge-div form-floating" id="badge-div">


            </div>


            <label for="user_type">Come vuoi loggarti?</label>
            <select class="form-select form-select-sm" name="user_type" id="user_type" aria-label=".form-select-sm example"
                    onchange="onSelectChange(this.value)">
                <option value="user" selected>Utente</option>
                <option value="employee">Dipendente</option>
            </select>

            <p class="mt-5 mb-3 text-black-50">Non sei ancora registrato? <a href="register.html">registrati</a></p>
            <input type="submit" class="w-100 btn btn-lg btn-primary" id="submit" name="submit" value="Accedi">
            <p class="mt-5 mb-3 text-muted">&copy; 2021-today</p>
        </form>
```

CSS

CSS (Cascade Style Sheets) e' un linguaggio che, tramite delle regole, permette di definire la formattazione e l'aspetto degli elementi HTML, e' detto Cascade perche' ogni regola che viene definita per l'elemento padre viene applicata anche agli elementi figli. Definendo regole che si applicano agli elementi HTML il file che definisce le regole CSS deve essere linkato alla pagina HTML tramite il tag <link rel="stylesheet" href="style.css"> oppure queste regole possono anche essere definite direttamente nel file HTML all'interno del tag <style></style>.

Esempio di regola CSS:

```
html,
body {
  height: 100%;
}

.form {
  display: flex;
  align-items: center;
  padding-top: 40px;
  padding-bottom: 40px;
  background-color: #f5f5f5;
  margin: auto;
  width: 30%;
}
```

Nel progetto e' stato usato principalmente boostrap 5, un framework CSS (ma anche js) che contiene regole CSS gia' definite e pronte all'uso, quindi noi definendo l'attributo class nei tag HTML possiamo attribuire delle determinate regole gia' definite da bootstrap alla nostra pagina.

Javascript

Javascript e' un linguaggio di programmazione utilizzato principalmente nell'ambito web, viene interpretato lato client quindi e' il browser che lo esegue e definisce il comportamento delle pagine web. Puo' essere usato anche lato server con l'utilizzo di librerie come node.js. Nel progetto javascript e' stato utilizzato principalmente assieme ad un framework chiamato Vue.js. Vue.js permette di creare componenti a cui viene associato un nome, i componenti possono essere utilizzati come tag HTML, questi componenti sono reattivi cioe' tengono traccia delle variabili da cui dipendono e quando queste variabili subiscono delle modifiche il motore di Vue sa' quale componente renderizzare e quando. Oltre a Vue.js e' stato anche usato jQuery per le chiamate Ajax.

- Un esempio di Vue.js e' il seguente:

```
//creo il componente Vue e gli assegno il nome card-article, nel mio codice html se uso il tag <card-article> </card-article>
//uso questo componente
Vue.component('card-article', {
    //il template definisce come viene renderizzato il componente ed e' composto da codice HTML con l'aggiunta che
    //posso usare delle variabili al suo interno tramite {{ variable }}
```

```
    template: `
      <div class="col">
      <div class="card shadow-sm">
        <svg class="bd-placeholder-img card-img-top" width="100%" height="225"
             xmlns="http://www.w3.org/2000/svg" role="img" aria-label="Placeholder: Thumbnail"
             preserveAspectRatio="xMidYMid slice" focusable="false"><title>{{ prodname }}</title>
          <image :href="imglink"
                height="100%" width="100%"></image>
        </svg>

        <div class="card-body">
          <p class="card-text">{{ prodname }}</p>
          <div class="d-flex justify-content-between align-items-center">
            <div class="btn-group">
              <button type="button" class="btn btn-sm btn-outline-secondary" @click="showDetails">Dettagli</button>

            </div>
            <button type="button" class="price btn btn-warning">
              {{ price }} €
            </button>
          </div>
        </div>
      </div>

      <div style="position: fixed; width: 100%; height: 100%; z-index: 100; top: 0; left: 0" v-if="showdetails"
           @click="handleClick" ref="detailsMask">
        <div class="details" v-html="prodname"
             style="background-color: red; margin: 3rem; top: 0; left: 0">
        </div>
      </div>

      </div>`,


    data() {
        return {
            showdetails: false
        }
    },
    //queste variabili le definisco nel'HTML in store.php e vengono usate dal template
    props: {
        prodname: String,
        price: Number,
        imglink: String
    },
    //definisco i metodi
    methods: {
        showDetails() {
            this.showdetails = true
        },
        handleClick(evt) {
            if (evt.target === this.$refs.detailsMask) {
                this.showdetails = false;
            }
        }
    }
})


new Vue({
    el: '#app',     //elemento di aggancio
    data() {
        return {
            articles: {},
            inputstring: ''
        };
    },
    methods: {
        //ajax usando jquery
        //evt = event -> prende la key dall'user input
        search_articles(evt) {
            //eseguo una chiamata ajax usando jquery con il metodo post
            $.ajax(
                'search.php',
                {
                    success: (data) => {    //quando ricevo risposta e non si e' verificato nessun errore
                        // console.log(data)
                        if (data !== 'not_found') {
                            this.articles = JSON.parse(data);           //cambio la variabile articles e il componente vue
                                                                        //grazie alla sua rattivita' si aggiorna mostrando
                                                                        //le nuove informazioni
                        }
                        // console.log(this.articles)
                    },
                    error: function () {                                //in caso da errore
                        //da errore se la ricerca non trova risultati ma
                        //non e' da tenere conto
```

```
                },
                method: "POST",                                     //definisco il metodo POST per ajax
                data: {search: this.inputstring}                    //invio l'oggetto tramite post
            }
        );
    }
},
mounted() {
    this.search_articles()
},
computed: {
    get_articles() {
        // console.log(this.articles)
        return this.articles
    }
},
});
```

In questo caso uso Vue per renderizzare gli articoli che ricerco tramite ajax (jquery), grazie a Vue ho definito un template per ogni articolo e quando scrivo sulla barra di ricerca nella pagina dello store la funzione search_articles() viene richiamata, lei modifica l'array di articoli (articles) e Vue accorgendosi che l'array e' cambiato renderizza nuovamente il componente modificando i dati. Questo e' possibile anche grazie alla codifica nella parte HTML:

```
<div id="app">
            <div>
                <div class="bg-light form-control">
                    <label for="search">Cerca:</label>
                    <input type="text" class="form-control" name="search" id="search"
                           v-model="inputstring"
                           @input="search_articles">
                </div>

                <div class="album py-5 bg-light">
                    <div class="container">
                        <div class="row row-cols-1 row-cols-sm-2 row-cols-md-3 g-3">

                            <card-article
                                    v-for="item in articles"
                                    :prodname="item.product_name"
                                    :price="item.price"
                                    :imglink="item.linkImage"
                                    :key="item.product_name"
                            >
                            </card-article>

                        </div>
                    </div>
                </div>
            </div>
        </div>
```

- Funzione che usa la libreria jQuery per effettuare una chiamata ajax quindi asincrona (non blocca l'esecuzione in attesa della risposta)

```
search_articles(evt) {
            //eseguo una chiamata ajax usando jquery con il metodo post
            $.ajax(
                'search.php',
                {
                    success: (data) => {    //quando ricevo risposta e non si e' verificato nessun errore
                        // console.log(data)
                        if (data !== 'not_found') {
                            this.articles = JSON.parse(data);           //cambio la variabile articles e il componente vue
                                                                        //grazie alla sua rattivita' si aggiorna mostrando
                                                                        //le nuove informazioni
                        }
                        // console.log(this.articles)
                    },
                    error: function () {                                //in caso da errore
                        //da errore se la ricerca non trova risultati ma
                        //non e' da tenere conto
                    },
                    method: "POST",                                     //definisco il metodo POST per ajax
                    data: {search: this.inputstring}                    //invio l'oggetto tramite post
                }
            );
        }
```

Back end

Il back end e' la parte del sito invisibile all'utente poiche' viene processata nel server, questa parte ha il compito di processare ed elaborare i dati, interfacciarsi col server e gestire le richieste (login, form di input ecc) degli utenti.

PHP

Php e' il linguaggio scelto per il backend del progetto, e' un linguaggio di scripting interpretato e viene usato per elaborare i dati sul server e inviare l'output al client sotto forma di codice HTML. Grazie a php si possono creare pagine web dinamiche.

Esempi codice PhP:

- Funzione per connettersi al database:

```
<?php
//funzione da usare globalmente per connettersi al database
function connect_db($user, $psw, $database): mysqli
{
    $hostname = "127.0.0.1";

    $conn = new mysqli($hostname, $user, $psw, $database);

    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }
//    echo "<script>console.log('Connected successfully')</script>";
    return $conn;
}
?>
```

- Registrazione di un nuovo utente:

```
<?php
session_start();
require '../connection_db.php';
$conn = connect_db("root", "", "db_catena_negozi_2");

if (isset($_REQUEST['submit'])) {
    //insert con prepared statement
    $stmt = $conn->prepare("INSERT INTO db_catena_negozi_2.utente (nome, cognome, email, password)
                                VALUES (?,?,?,?)");
    $stmt->bind_param("ssss", $name, $surname, $email, $psw_hash);    //s = string


    $name = $_POST['name'];
    $surname = $_POST['surname'];
    $email = $_POST['email'];
    $password = $_POST['password_1'];
    $psw_hash = password_hash($password, PASSWORD_DEFAULT);

    if ($stmt->execute()) {
        $_SESSION['name'] = $name;
        $_SESSION['surname'] = $surname;
        $_SESSION['email'] = $email;
        header("Location: ../app/homepage.php");
    } else {
        echo "Error: ".$stmt->error;

        header("Location: ../generic_error.html");
    }
} else {
    header("Location: register.html");
    exit();
}
?>
```

- Oggetto Product (Articolo in vendita nel sito)

```
<?php

class Product implements JsonSerializable
{

    private string $product_name;
    private string $description;
    private string $category;
```

```
    private string $brand;
    private float $price;
    private string $linkImage;

    /**
     * Product constructor.
     * @param string $product_name
     * @param string $description
     * @param string $category
     * @param string $brand
     * @param float $price
     * @param string $linkImage
     */
    public function __construct(string $product_name, string $description, string $category, string $brand, float $price, stri
    {
        $this->product_name = $product_name;
        $this->description = $description;
        $this->category = $category;
        $this->brand = $brand;
        $this->price = $price;
        $this->linkImage=$linkImage;
//        $this->articleCard = $this->getCard();
    }

    /**
     * @return string
     */
    public function getLinkImage(): string {
        return $this->linkImage;
    }

    /**
     * @param string $linkImage
     */
    public function setLinkImage(string $linkImage): void {
        $this->linkImage = $linkImage;
    }


    /**
     * @return string
     */
    public function getProductName(): string
    {
        return $this->product_name;
    }

    /**
     * @param string $product_name
     */
    public function setProductName(string $product_name): void
    {
        $this->product_name = $product_name;
    }

    /**
     * @return string
     */
    public function getDescription(): string
    {
        return $this->description;
    }

    /**
     * @param string $description
     */
    public function setDescription(string $description): void
    {
        $this->description = $description;
    }

    /**
     * @return string
     */
    public function getCategory(): string
    {
        return $this->category;
    }

    /**
     * @param string $category
     */
    public function setCategory(string $category): void
    {
        $this->category = $category;
    }
```

```
    /**
     * @return string
     */
    public function getBrand(): string
    {
        return $this->brand;
    }

    /**
     * @param string $brand
     */
    public function setBrand(string $brand): void
    {
        $this->brand = $brand;
    }

    /**
     * @return float
     */
    public function getPrice(): float
    {
        return $this->price;
    }

    /**
     * @param float $price
     */
    public function setPrice(float $price): void
    {
        $this->price = $price;
    }

    public function jsonSerialize()
    {
        // TODO: Implement jsonSerialize() method.
        return get_object_vars($this);
    }
}
?>
```

Sql

Sql (Structured query language) e' un linguaggio per database che ci permette di interrogare i dati memorizzati, inserire e modificare dati, creare e modificare le tabelle. Sql viene utilizzato per interagire con il database, inserire dati e eseguire interrogazioni.

Ecco alcuni comandi sql:

- Ricerca fulltext: ricerca una stringa all'interno di piu' tables, si deve prima fare un indice fulltext con quelle tables

```
SELECT * FROM db_catena_negozi_2.articolo
WHERE MATCH(nome, nome_marca, descrizione)
AGAINST("RTX 2070" in boolean mode);
```

- Registrazione di un utente e inserimento dei suoi dati nel database:

```
INSERT INTO db_catena_negozi_2.utente (nome, cognome, email, password)
                                VALUES ("Cristian",
                                        "Scapin",
                                        "crissca@gmail.com",
                                        "$2y$10$YSvJ.krXidvJPYEfNoV.beljv7JyI3MF2N7AcvDuVXXA1Qzgumr4q");
```

- Ricevere informazioni su tutti gli ordini:

```
SELECT O.id,
       O.id_negozio,
       O.id_utente,
       U.email,
       A.nome,
       A.nome_marca,
       A.categoria,
       A.prezzo,
       O.garanzia,
       O.data
FROM db_catena_negozi_2.ordine O
```

```
        JOIN db_catena_negozi_2.articolo A ON O.nome_pezzo = A.nome
        JOIN db_catena_negozi_2.utente U ON O.id_utente = U.id;
```