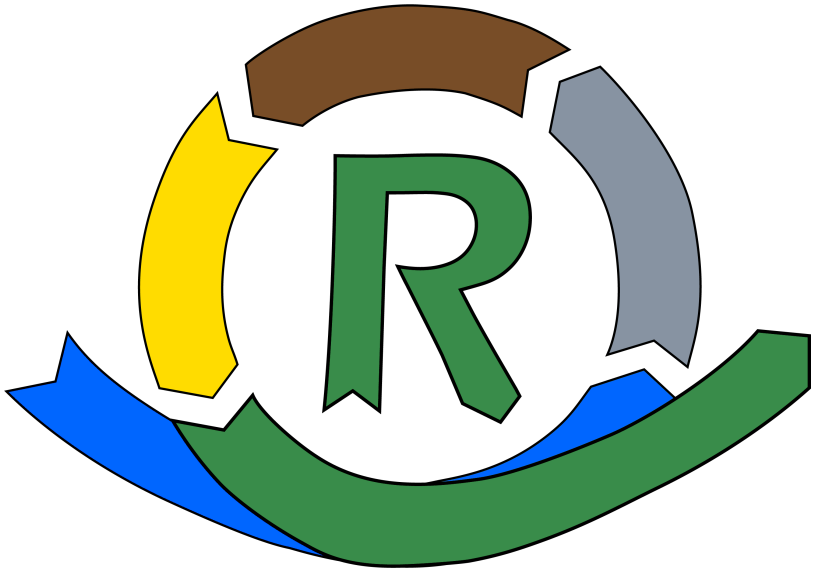
Rius.Co.

L'e-commerce per il RIUSo COllaborativo



Mauro Pellonara

Elaborato di Informatica e Sistemi e Reti

I.I.S. Volterra Elia

31/05/2021

Indice

1 Introduzione

L'accelerazione del progresso tecnologico, la progressiva liberalizzazione degli scambi commerciali e il grande sviluppo delle comunicazioni hanno portato alla nascita del mercato globale. Le aziende realizzano nuovi prodotti a ritmi sempre più elevati con prezzi sempre più competitivi ed economici; questo spinge i consumatori a credere che gli oggetti in loro possesso diventino obsoleti nell'arco di pochi mesi e li invita quindi ad acquistare sempre di più. Questo continuo ciclo d'acquisto è alla base del consumismo moderno che negli ultimi anni sta producendo sempre più spazzatura mettendo in pericolo il pianeta intero. Numerosi studi [2] rivelano che il problema spesso risiede nel fatto che i consumatori e le aziende, non sapendo che fare dei prodotti obsoleti, li buttano via anche se funzionanti.

Rius.Co. cerca di risolvere questa problematica attraverso una piattaforma per il riuso online che permette a chiunque di poter acquisire oggetti di seconda mano funzionanti in modo del tutto gratuito e facilmente accessibile. Rius.Co. si ispira ai già esistenti centri del riuso presenti nella Regione Marche, ma innova portando questa iniziativa solidale nelle mani di tutti attraverso il sito web sulla quale si appoggerà la piattaforma. Il sito web consiste in un e-commerce customer-to-customer in cui tutti possono offrire e richiedere oggetti usando la moneta di scambio virtuale coniata a questo scopo: il Green Coin. Spendendo un Green Coin gli utenti potranno richiedere ed ottenere un prodotto da un altro utente appartenente alla piattaforma. Ogni prodotto, qualsiasi esso sia, se funzionante e in buone condizioni potrà essere inserito nel Marketplace dove gli verrà assegnato il "prezzo" di un Green Coin a prescindere dal suo valore economico che è trascurato in quanto si tratta di oggetti che altrimenti sarebbero stati buttati in discarica.

(a) Logo Rius.Co. (b) Logo esteso Rius.Co. (c) Green Coin

Figura 1: Loghi ed icone dell'azienda

Un importante punto di forza della piattaforma è la possibilità d'integrazione con i già esistenti centri del riuso locali facilitandone il flusso degli oggetti e aiutandoli quindi a costruire un inventario completo o ad alleviare la mole di lavoro e la quantità di oggetti che questi centri, spesso di modeste dimensioni, gestiscono con difficoltà. I centri del riuso locali possono quindi beneficiare in modo considerevole da una collaborazione con la piattaforma Rius.Co. che ne guadagnerebbe l'opportunità di ingrandire il proprio bacino d'utenza. La piattaforma può anche servire come rete di collegamento dei centri del riuso già esistenti, aiutandoli nella comunicazione e negli scambi di oggetti in modo efficiente. Il progetto risulta quindi semplice dal punto di vista operativo, economico dal punto di vista finanziario e sostenibile dal punto di vista ambientale.

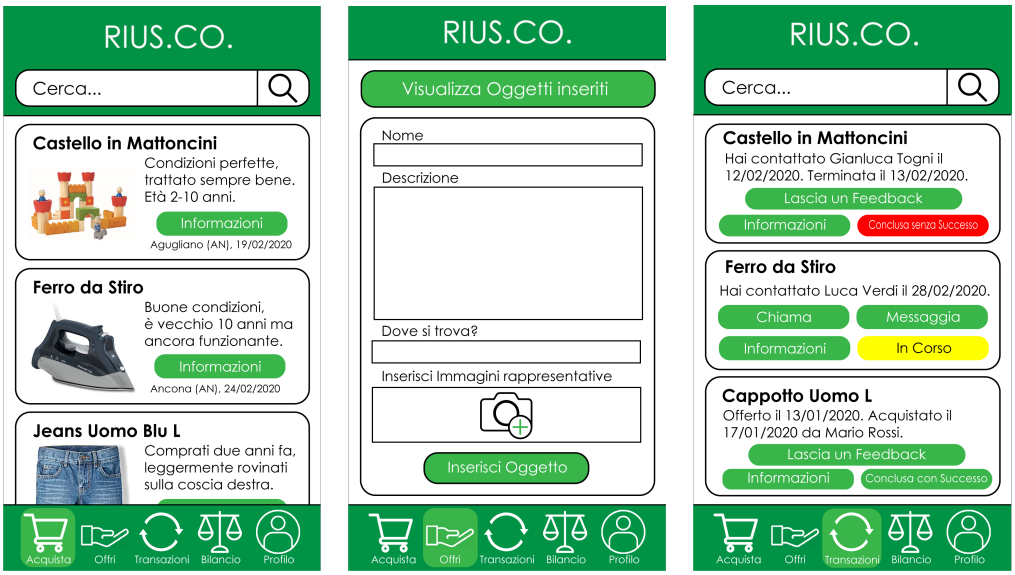
Rius.Co. ha in comune numerosi obiettivi con l'importante Agenda 2030, un programma d'azione per le persone, il pianeta e la prosperità sottoscritto nel 2015 dai Paesi membri dell'ONU [5], questi obiettivi sono:

- 1° Obiettivo – "Porre fine alla povertà in tutte le sue forme": chiunque può commerciare gratuitamente sulla piattaforma a prescindere dal proprio reddito o situazione economica;
- 11° Obiettivo – "Rendere le città e gli insediamenti umani inclusivi, sicuri, duraturi e sostenibili": le città possono diventare più sostenibili e sicure grazie alla riduzione dei rifiuti;
- 12° Obiettivo – "Garantire modelli sostenibili di produzione e di consumo": la piattaforma offre un modello sostenibile di consumo congruo alla vita d'oggi;
- 13° Obiettivo – "Promuovere azioni, a tutti i livelli, per combattere il cambiamento climatico": promuovendo il riutilizzo degli oggetti e, di conseguenza, scoraggiando l'accumulo di rifiuti nelle discariche, Rius.co. supporta la lotta contro il cambiamento climatico.

Figura 2: Obiettivi 1, 11, 12 e 13 dell'Agenda 2030

2 Analisi dei requisiti

La registrazione alla piattaforma Rius.Co. sarà obbligatoria per poter accedere al Marketplace, gli utenti dovranno inserire la propria e-mail identificativa, il nome dell'account, la password, un'immagine di profilo e la città in cui vivono. La piattaforma inizialmente sarà accessibile esclusivamente tramite il sito web, verrà successivamente introdotta un'applicazione mobile della quale è già presente il prototipo visualizzabile qui [12].



(a) Acquista (b) Offri (c) Transazioni

Figura 3: Prototipo dell'applicazione mobile

Gli utenti potranno ottenere dei Green Coin per commerciare sul Marketplace attraverso le seguenti modalità:

- **Registrazione sulla piattaforma**: verrà fornito un Green Coin alla registrazione di un nuovo account;
- **Condivisione della piattaforma sui propri canali Social**: gli utenti che sceglieranno di condividere e pubblicizzare online l'applicazione otterranno un Green Coin. Questa modalità sarà rinnovabile mensilmente, perciò gli utenti potranno ottenere un Green Coin ogni mese semplicemente pubblicizzando la piattaforma che ne guadagnerà in popolarità ed utenti, fidelizzando anche i propri clienti;

- **Inserimento di oggetti in offerta nel Marketplace**: l'utente potrà inserire oggetti a patto che siano funzionanti e in buone condizioni, otterrà un Green Coin per ogni prodotto inserito nel Marketplace e ceduto ad un altro utente;

- **Dimostrare di essere in una situazione di difficoltà economica** attraverso la dichiarazione dei redditi o l'attestazione ISEE, a tali utenti verranno forniti mensilmente dai 2 ai 5 Green Coin.

Il Green Coin è stato pensato per risolvere una problematica importante comune a tutti gli attuali centri del riuso fisici: la presenza di utenti che abusano della piattaforma richiedendo una grande mole di oggetti, senza mai contribuire al sistema offrendone altrettanti.

Gli utenti dopo la registrazione potranno accedere alla piattaforma ed usufruire dei servizi che essa offre, sarà quindi possibile:

- **Acquistare un prodotto di cui si necessita attraverso la pagina "Acquista".** Nella pagina saranno presenti tutti gli oggetti offerti da utenti membri della piattaforma nella vicinanza dell'utente. Verrà data anche la possibilità di cercare un determinato prodotto all'interno del Marketplace. L'utente interessato ad un prodotto potrà spendere un Green Coin per contattare l'utente offerente e accordarsi per lo scambio del prodotto. Nel caso in cui lo scambio non si sia concluso con successo l'utente richiedente otterrà indietro il Green Coin che aveva speso;

- **Offrire un prodotto attraverso la pagina "Offri".** Nella pagina sarà possibile inserire un prodotto da offrire nel Marketplace, saranno richieste una breve descrizione e delle foto rappresentative del prodotto. Verrà resa pubblica anche la posizione del prodotto. Nel caso un utente sia interessato al prodotto il venditore verrà contattato da quest'ultimo e otterrà un Green Coin nel caso lo scambio si concluda con successo;

- **Controllare le proprie transazioni attraverso la pagina "Transazioni".** L'utente potrà visualizzare tutte le sue transazioni, i dettagli relativi ad esse e lo stato attuale: "In Corso", "Conclusa senza Successo" e "Conclusa con Successo". Saranno anche visualizzabili i movimenti di Green Coin in entrata ed uscita;

- **Visualizzare e modificare il proprio profilo attraverso la pagina "Profilo".** L'utente potrà visualizzare il proprio profilo e modificare i dati relativi ad esso mantenendo aggiornati dati importanti come la città di residenza.

3 Analisi funzionale

framework
- mvc
- ajax
- sqlite3(tipi)
- ef core
- trasferimento dati

4 Database

4.1 Descrizione

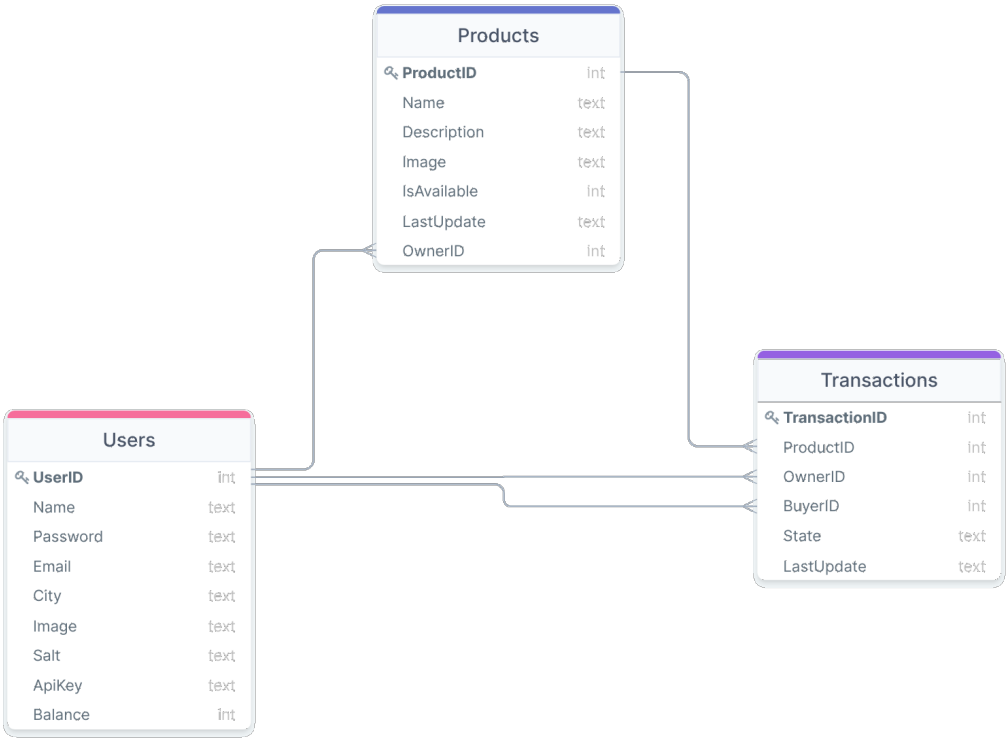
Per garantire la semplicità di gestione e la velocità di esecuzione delle query nel database sono presenti solo 3 tabelle: Users, Products e Transactions. Queste sono il minimo indispensabile per la creazione di un e-commerce funzionante. Il database sarà collocato nel data server che conterrà anche le immagini di profilo degli utenti e quelle legate ai prodotti. Quella riportata nel modello E/R è la prima versione del database che permette di caricare un'unica immagine per prodotto, nelle versioni successive verranno introdotti aggiornamenti graduali paralleli allo sviluppo completo del sito web. Il database è visualizzabile insieme a tutto il progetto scaricando la repository di GitHub disponibile qui [13].

La tabella Users contiene le informazioni base necessarie per l'utilizzo della piattaforma: ID utente, nome utente, password, email identificativa, città di residenza, immagine di profilo, salt per la generazione dell'hash della password, api key per eseguire richieste tramite l'API e quantità di green coin posseduti.

La tabella Products contiene invece i seguenti campi: ID prodotto, nome prodotto, descrizione prodotto, immagine del prodotto, disponibilità (per tenere traccia dei prodotti anche dopo la loro vendita o la rimozione dal Marketplace), data dell'ultimo aggiornamento apportato al prodotto e ID utente proprietario del prodotto.

La tabella Transactions contiene le transazioni effettuate tra gli utenti e ha i seguenti campi: ID transazione, ID prodotto scambiato, ID utente proprietario del prodotto al momento dello scambio (campo necessario per tenere traccia dei cambi di proprietà e delle transazioni anche dopo che queste si sono concluse), ID utente acquirente, stato della transazione (può essere "In Corso", "Completata Con Successo" o "Completata Senza Successo") e data dell'ultimo aggiornamento relativo alla transazione.

4.2 Modello E/R



4.3 Modello logico

Users		
Nome Campo	**Tipo**	**Note**
UserID	INTEGER	NOT NULL, PK, AUTOINCREMENT
Name	TEXT	NOT NULL
Password	TEXT	NOT NULL
Email	TEXT	NOT NULL
City	TEXT	NOT NULL
Image	TEXT	NOT NULL
Salt	TEXT	NOT NULL
ApiKey	TEXT	NOT NULL
Balance	INTEGER	NOT NULL

Products		
Nome Campo	**Tipo**	**Note**
ProductID	INTEGER	NOT NULL, PK, AUTOINCREMENT
Name	TEXT	NOT NULL
Description	TEXT	NOT NULL
Image	TEXT	NOT NULL
IsAvailable	INTEGER	NOT NULL
LastUpdate	TEXT	NOT NULL
OwnerID	INTEGER	NOT NULL, FK(Users)

Transactions		
Nome Campo	**Tipo**	**Note**
TransactionID	INTEGER	NOT NULL, PK, AUTOINCREMENT
ProductID	INTEGER	NOT NULL, FK(Products)
OwnerID	INTEGER	NOT NULL, FK(Users)
BuyerID	INTEGER	NOT NULL, FK(Users)
State	TEXT	NOT NULL
LastUpdate	TEXT	NOT NULL

4.4 Entity Framework Core

Il database è stato generato automaticamente da Entity Framework Core a partire dai modelli scritti in C# e inseriti nel progetto (generazione code-first) [7]. I modelli inseriti devono essere riportati nel DbContext indicato nel file Startup.cs, il database context si occupa quindi di descrive il database e le tabelle che lo compongono, quest'ultime vengono invece descritte dai modelli indicati nel database context. La nomenclatura dei modelli comprende l'espressione DTO (Data Transfer Object [11]) in quanto queste classi vengono usate per il trasferimento dei dati dagli utenti al database all'interno del sistema distribuito sul quale è basata la piattaforma.

MainDbContext.cs

```
using riusco_mvc.Models;
using Microsoft.EntityFrameworkCore;

namespace riusco_mvc.Data
{
    public class MainDbContext : DbContext
    {
        public DbSet<UserDTO> Users { get; set; }
        public DbSet<ProductDTO> Products { get; set; }
```

```
        public DbSet<TransactionDTO> Transactions { get; set; }
        public MainDbContext(DbContextOptions<MainDbContext>
            options) : base(options) { }
    }
}
```

UserDTO.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace riusco_mvc.Models
{
    public class UserDTO
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        // chiave primaria con autoincrement
        public int UserID { get; set; }
        public string Name { get; set; }
        public string Password { get; set; }
        public string Email { get; set; }
        public string City { get; set; }
        public string Image { get; set; }
        public string Salt { get; set; }
        public string ApiKey { get; set; }
        public int Balance { get; set; }

        // metodi costruttori scritti con overloading

        public UserDTO(string name, string password, string
            email, string image, string salt, string apiKey,
            int balance, string city)
        {
            Name = name;
            Password = password;
            Email = email;
            Image = image;
            Salt = salt;
            ApiKey = apiKey;
            Balance = balance;
```

```
            City = city;
        }

        public UserDTO(int userId, string name, string
            password, string email, string image, string salt,
            string apiKey, int balance, string city)
        {
            UserID = userId;
            Name = name;
            Password = password;
            Email = email;
            Image = image;
            Salt = salt;
            ApiKey = apiKey;
            Balance = balance;
            City = city;
        }

        public UserDTO() { }
    }
}
```

ProductDTO.cs

```
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace riusco_mvc.Models
{
    public class ProductDTO
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        // chiave primaria con autoincrement
        public int ProductID { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public string Image { get; set; }
        public bool IsAvailable { get; set; }
        public DateTime LastUpdate { get; set; }
```

```
        [ForeignKey("Owner")]
        public int OwnerID { get; set; }
        public UserDTO Owner { get; set; }

        // metodi costruttori scritti con overloading

        public ProductDTO(string name, string description,
            string image, DateTime lastUpdate, bool
            isAvailable, int ownerId)
        {
            Name = name;
            Description = description;
            Image = image;
            LastUpdate = lastUpdate;
            IsAvailable = isAvailable;
            OwnerID = ownerId;
        }

        public ProductDTO(int productId, string name, string
            description, string image, DateTime lastUpdate,
            bool isAvailable, int ownerId)
        {
            ProductID = productId;
            Name = name;
            Description = description;
            Image = image;
            LastUpdate = lastUpdate;
            IsAvailable = isAvailable;
            OwnerID = ownerId;
        }

        public ProductDTO() { }
    }
}
```

TransactionDTO.cs

```
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Text.Json;
```

```
namespace riusco_mvc.Models
{
    public class TransactionDTO
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        // chiave primaria con autoincrement
        public int TransactionID { get; set; }
        [ForeignKey("Product")]
        public int ProductID { get; set; }
        public ProductDTO Product { get; set; }
        [ForeignKey("Owner")]
        public int OwnerID { get; set; }
        public UserDTO Owner { get; set; }
        [ForeignKey("Buyer")]
        public int BuyerID { get; set; }
        public UserDTO Buyer { get; set; }
        public DateTime LastUpdate { get; set; }
        public string State { get; set; }

        // metodi costruttori scritti con overloading

        public TransactionDTO(int productId, int ownerID, int
            buyerId, DateTime lastUpdate, string state)
        {
            ProductID = productId;
            OwnerID = ownerID;
            BuyerID = buyerId;
            LastUpdate = lastUpdate;
            State = state;
        }

        public TransactionDTO() { }

        // metodo override ToString()
        public override string ToString()
        {
            return JsonSerializer.Serialize(this);
        }
    }
```

```
}
```

4.5 Query

Usando Entity Framework Core per la creazione e la gestione del database le query possono essere scritte con la sintassi sviluppata da Microsoft chiamata LINQ [6], alternativamente possono essere anche usati i metodi appositi inclusi nella libreria Microsoft dedicata (Microsoft.EntityFrameworkCore) [8]. Nel progetto ho scelto di usare la libreria Microsoft dedicata in quanto è la più veloce e quella che richiede meno l'intervento del programmatore, evitando quindi errori semplici di distrazione.

Query LINQ

```
using Microsoft.Linq;
using Microsoft.EntityFrameworkCore;

// query che ottiene il prodotto con chiave primaria uguale a 2
List<ProductDTO> products = _context.Products.ToList();
ProductDTO product = (from product in products
                      where product.ProductID == 2
                      select product).First();

// equivalente SQL
// SELECT * FROM Products WHERE Products.ProductID = 2 LIMIT 1;
```

Query con metodi

```
using Microsoft.EntityFrameworkCore;

// query che ottiene l'utente con email uguale
   mauro.pellonara@gmail.com, nel caso piu' utenti abbiano la
   stessa email ritorna un errore
UserDTO user = _context.Users.Single(x => x.Email ==
   "mauro.pellonara@gmail.com");

// equivalente SQL
// SELECT * FROM Users WHERE Users.Email =
   "mauro.pellonara@gmail.com" LIMIT 1;
```

5 Infrastruttura IT

5.1 Descrizione

5.2 Apparati Utilizzati

5.2.1 Descrizione

5.2.2 Data server

5.2.3 Application server

5.2.4 API server

5.2.5 Web server

5.3 Progetto di rete

Ho progettato la rete per la sede centrale dell'azienda usando Cisco Packet Tracer 8.0, nelle etichette da me inserite sono riportati gli indirizzi IP degni di nota e le informazioni relative alle 4 sottoreti create. Il file è stato caricato insieme a tutto il progetto nell'apposita repository creata su GitHub e raggiungibile qui [13].

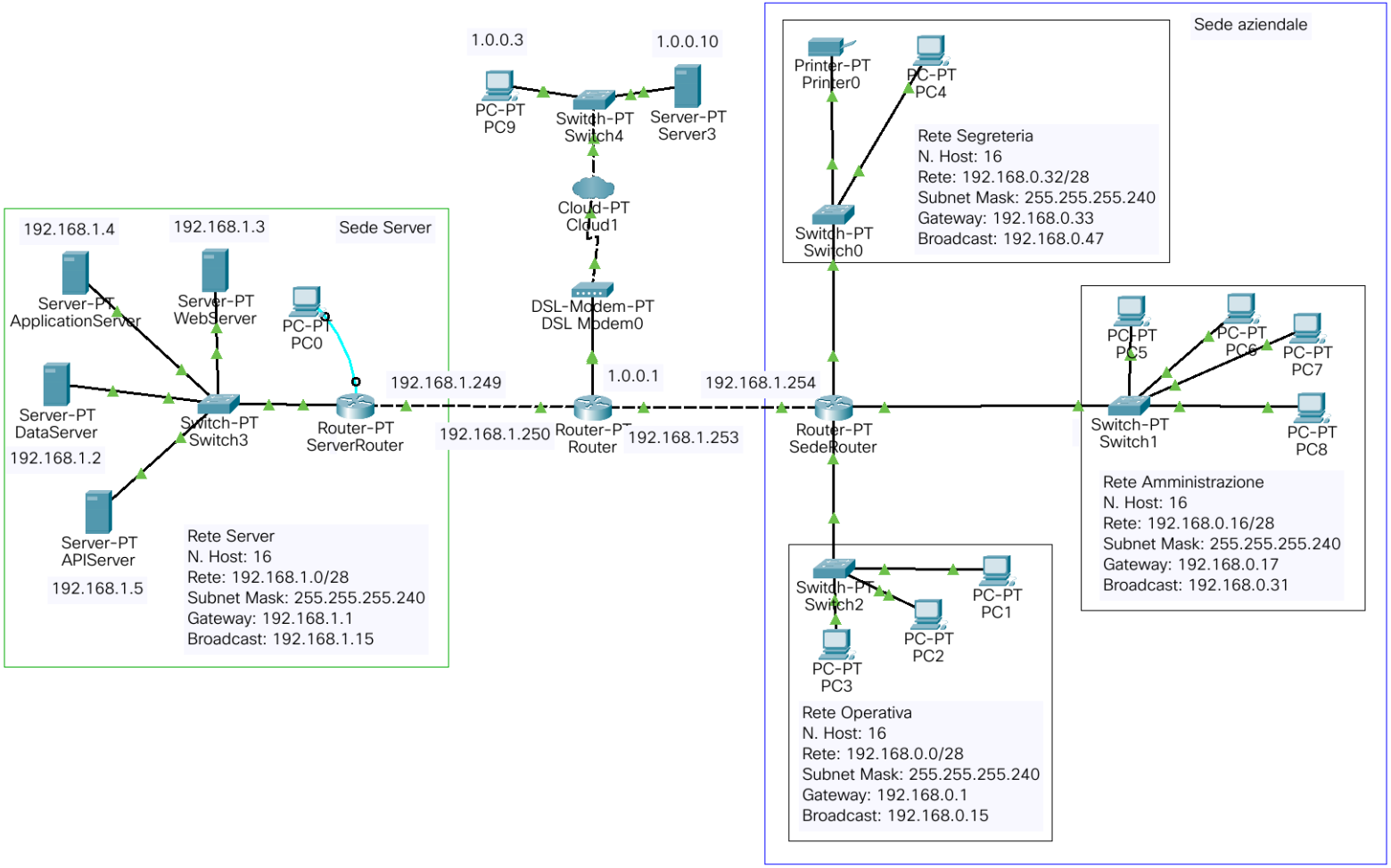


Figura 4: Progetto della rete scritto con Cisco Packet Tracer.

6 Politiche di sicurezza

6.1 Pratiche comuni

Gli e-commerce, più di molti altri siti web, devono garantire un livello di sicurezza più che eccellente in quanto le conseguenze causate da eventuali bug o vulnerabilità sono potenzialmente catastrofiche per gli utenti che usufruiscono della piattaforma e gli amministratori che ne sono responsabili. Ho quindi adottato numerose pratiche comuni in ambito web per garantire un livello di sicurezza consono alla piattaforma, queste sono:

- **Rinominare i file caricati dagli utenti**, degli eventuali attaccanti possono caricare file con nomi particolari volti ad accedere a cartelle contenenti file sensibili che quindi risulterebbero compromessi;
- **Controllare il contenuto dei file caricati dagli utenti**, gli unici file che gli utenti possono caricare sono immagini, viene quindi fatto un doppio controllo (lato client e lato server) per assicurarsi che i file caricati rispettino il corretto formato;
- **Uso del pattern Post/Redirect/Get**, questo pattern particolare adottato in maniera estensiva nel web permette di ovviare al problema della ricarica o navigazione temporale delle pagine e richieste già inviate al server [10]. Seguendo questo pattern tutte le richieste POST verranno prima elaborate e poi reindirizzate su la stessa o un'altra pagina che verrà ottenuta tramite richiesta GET, in questo modo l'utente nella sua navigazione non incontrerà mai messaggi relativi alla celebre "Conferma reinvio modulo";
- **HTTPS** [1], il sito usa il protocollo TLS/SSL insieme al protocollo HTTP per la navigazione sicura all'interno del sito, i dati trasmessi da e verso il sito attraversano un tunnel logico end-to-end dentro il quale sono crittografati e non leggibili da possibili attaccanti;
- **Autenticazione degli utenti**, gli utenti si devono autenticare per poter compiere azioni sul sito come inserire un prodotto;
- **Password hashing**, le password usate dagli utenti non vengono salvate in chiaro dal database che invece le memorizza già crittografate, in questo modo in caso di data breach le password degli utenti anche se diffuse non sono leggibili.

6.2 Autenticazione

La piattaforma ha due modalità di accesso: il sito web e l'API. Entrambe le modalità devono effettuare l'autenticazione dell'utente che s'interfaccia con la piattaforma Rius.Co. Il sito web si occupa dell'autenticazione richiedendo agli utenti di impostare una password di lunghezza variabile tra 8 e 24 caratteri. I clienti possono anche richiedere la modifica della password che è comunque una pratica da fare periodicamente per evitare problemi relativi alla sicurezza del profilo.

L'API invece esegue l'autenticazione dell'utente in tutt'altro modo, gli utenti alla registrazione ricevono una chiave univoca privata (API key) di 32 byte che viene generata in modo casuale e non è modificabile [16]. Questa chiave dovrà quindi essere inserita nel form di tutte le richieste fatte all'API per permettere l'identificazione dell'utente associato alla richiesta. Alcune richieste come ottenere un file in formato JSON contenente tutti i prodotti presenti sul Marketplace non necessitano dell'autenticazione e vengono eseguite anche senza l'API key in quanto sono informazioni pubbliche visualizzabili da chiunque.

Esempio di richiesta all'API tramite curl

```
$ curl -X GET --insecure
   "https://0.0.0.0:6066/Transactions/GetTransactionsByUserID/9" -F
   "apiKey=aBQKfxy175kL1v5EdOwCkHWtgwAj9Kbzu339OkDIgxA="
```

Risposta del server contenente tutte le transazioni in cui ha partecipato l'utente con l'API key indicata


```
[
    {
        "transactionID": 8,
        "productID": 12,
        "ownerID": 9,
        "buyerID": 10,
        "lastUpdate": "2021-05-26T21:53:52.5569413",
        "state": "Completed"
    },
    {
        "transactionID": 7,
        "productID": 12,
        "ownerID": 9,
        "buyerID": 10,
```

```
        "lastUpdate": "2021-05-18T11:51:42.1961266",
        "state": "Pending"
    },
    {
        "transactionID": 4,
        "productID": 13,
        "ownerID": 9,
        "buyerID": 10,
        "lastUpdate": "2021-05-17T12:42:48.5219505",
        "state": "Closed"
    }
]
```


6.3 Password hashing

Per garantire la privacy degli utenti ed evitare che i profili vengano violati in caso di data breach o altri tipi di fughe di informazioni è necessario rendere praticamente impossibile ottenere le password impostate dagli utenti. Le password devono essere quindi crittografate con algoritmi che rendano impraticabile la decifrazione, resistendo ad attacchi come il brute-force. Per la piattaforma Rius.Co. ho deciso di usare gli standard più recenti e robusti dal punto di vista informatico per eseguire l'hashing di tutte le password. L'hashing consiste nel produrre una stringa o digest correlata ad una serie di dati che sono in questo caso la password, una buona funzione di hashing rende praticamente impossibile risalire alla password partendo dal digest ma restituisce sempre lo stesso digest nel caso venga usata la stessa password [15]. In questo modo nel database è riportata l'elaborazione della password dalla quale nessuno può risalire all'originale.

Per eseguire l'hashing ho implementato la funzione di derivazione della chiave consigliata sia da Microsoft [9] che dal RFC 8018 [4], questa è definita nel RFC 2898 [3] e prende il nome di PBKDF2 (Password-Based Key Derivation Function 2) [14]. Questa funzione è ritenuta la migliore dal punto di vista di semplicità, prestazioni e sicurezza; restituisce una chiave derivata (DK) e richiede i seguenti parametri:

- PRF, la funzione pseudocasuale che si occuperà di eseguire l'hashing dei dati con la generazione del Message Authentication Code (HMAC) ad ogni iterazione;
- Password, la password della quale eseguire l'hashing e ottenere la chiave derivata;

- Salt, una sequenza di byte assegnata automaticamente all'utente e salvata in chiaro nel database necessaria per non avere hash uguali nel caso più utenti impostino la stessa password;
- c, il numero di iterazioni da svolgere per ottenere la chiave derivata, più iterazioni equivalgono a un maggiore livello di sicurezza ma contemporaneamente aumenta anche il tempo di esecuzione della funzione;
- dkLen, la lunghezza in bytes della chiave derivata.

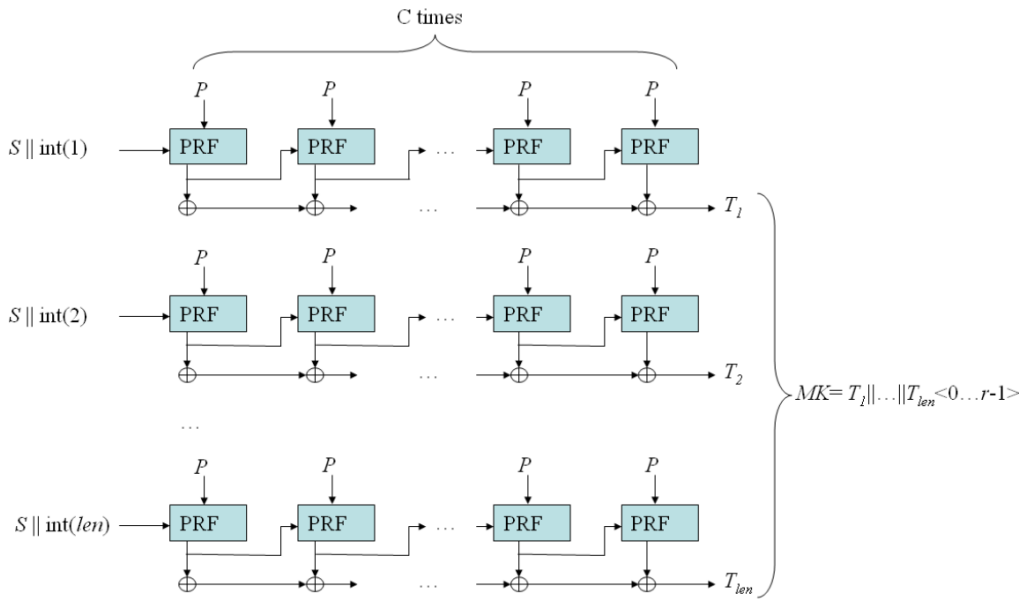


Figura 5: Rappresentazione grafica dell'algoritmo che compone un'iterazione

Nel mio caso PBKDF2 esegue 10000 iterazioni con l'algoritmo di hashing SHA256, usa un Salt di 128 byte e produce un'elaborazione della password lunga 32 byte, è implementato all'interno della seguente funzione.

```
private static string GetHash(string password, string salt)
{
    return Convert.ToBase64String(KeyDerivation.Pbkdf2(
        password: password,
        salt: Encoding.UTF8.GetBytes(salt),
        prf: KeyDerivationPrf.HMACSHA256,
        iterationCount: 10000,
        numBytesRequested: 32));
}
```

Riferimenti bibliografici

[1] Cloudflare. Descrizione del protocollo https. URL: https://www.cloudflare.com/learning/ssl/what-is-https/.

[2] Il fatto quotidiano. Abiti invenduti e la grande quantità di vestiti buttati. URL: https://www.ilfattoquotidiano.it/2020/02/01/i-marchi-di-alta-moda-bruciano-i-capi-invenduti-e-la-francia-prova-a-fermar 5691528/.

[3] IETF. Descrizione della funzione pbkdf2. URL: https://datatracker.ietf.org/doc/html/rfc2898#section-5.2.

[4] IETF. Indicazioni sul password hashing, versione aggiornata del rfc 2898. URL: https://datatracker.ietf.org/doc/html/rfc8018.

[5] ONU Italia. L'agenda 2030 presentata direttamente dall'onu. URL: https://unric.org/it/agenda-2030/.

[6] Documentazione Microsoft. Descrizione della sintassi linq. URL: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/query-syntax-and-method-syntax-in-linq.

[7] Documentazione Microsoft. Descrizione della tecnologia entity framework core. URL: https://docs.microsoft.com/en-us/ef/core/.

[8] Documentazione Microsoft. Documentazione della libreria microsoft.entityframeworkcore. URL: https://docs.microsoft.com/en-us/dotnet/api/microsoft.entityframeworkcore?view=efcore-5.0.

[9] Documentazione Microsoft. Eseguire l'hashing delle password in asp.net core. URL: https://docs.microsoft.com/en-us/aspnet/core/security/data-protection/consumer-apis/password-hashing?view=aspnetcore-5.0.

[10] Stack Overflow. Descrizione del pattern post/redirect/-get. URL: https://stackoverflow.com/questions/4327236/stop-browsers-asking-to-resend-form-data-on-refresh.

[11] Stack Overflow. Documentazione della libreria microsoft.entityframeworkcore. URL: https://stackoverflow.com/questions/1051182/what-is-a-data-transfer-object-dto.

[12] Mauro Pellonara. Prototipo navigabile dell'app mobile. URL: https://mauro886267.invisionapp.com/console/share/7Z10U19EHJ/476334736.

[13] Mauro Pellonara. Repository dedicata al progetto caricata su github. URL: https://github.com/MauroPello/elaborato.

[14] Wikipedia. Funzione di derivazione della chiave pbkdf2. URL: https://en.wikipedia.org/wiki/PBKDF2.

[15] Wikipedia. Funzione di hashing. URL: https://it.wikipedia.org/wiki/Funzione_di_hash.

[16] Wikipedia. Uso delle api key per l'autenticazione. URL: https://en.wikipedia.org/wiki/Application_programming_interface_key.