ESAME DI STATO DI ISTRUZIONE SECONDARIA SUPERIORE
ISTITUTO TECNICO INDUSTRIALE STATALE C.ZUCCANTE
Indirizzo: ITIA –INFORMATICA E TELECOMUNICAZIONI
ARTICOLAZIONE INFORMATICA

Candidato: Enrik Rucaj
Classe: 5IC
Anno scolastico: 2019/2020
Titolo: Applicazione per il monitoraggio dei tassi di cambio

Abstract:

Nel seguente elaborato si provvederà alla fase di definizione di un'applicazione mobile. Dovremo infatti proporre, per via dell'agenzia di intermediazione mobiliare Macchiavelli un'app che permetta ai suoi clienti di poter monitorare il tasso di cambio delle cripto monete (aggiunte ultimamente nel suo portafoglio di investimenti) in rapporto al dollaro statunitense e all'euro. Per la fase di definizione dell'applicazione si procederà nel seguente modo:

- *Fornire una analisi di massima per il progetto.*
- *Proporre un progetto di massima del servizio REST, e perciò pensare anche al modello concettuale del database e all'API REST necessario alla consultazione.*
- *Mostrare una parte significativa del codice necessario allo sviluppo dell'applicazione mobile (il linguaggio di programmazione da noi utilizzato sarà DART).*

Una volta analizzato con attenzione e svolto i 3 punti di sopra, bisognerà pensare anche alle varie modalità di protezione della nostra rete, in modo da garantire confidenzialità, integrità e autenticazione dei dati sensibili in caso di un possibile attacco al sistema di comunicazione.

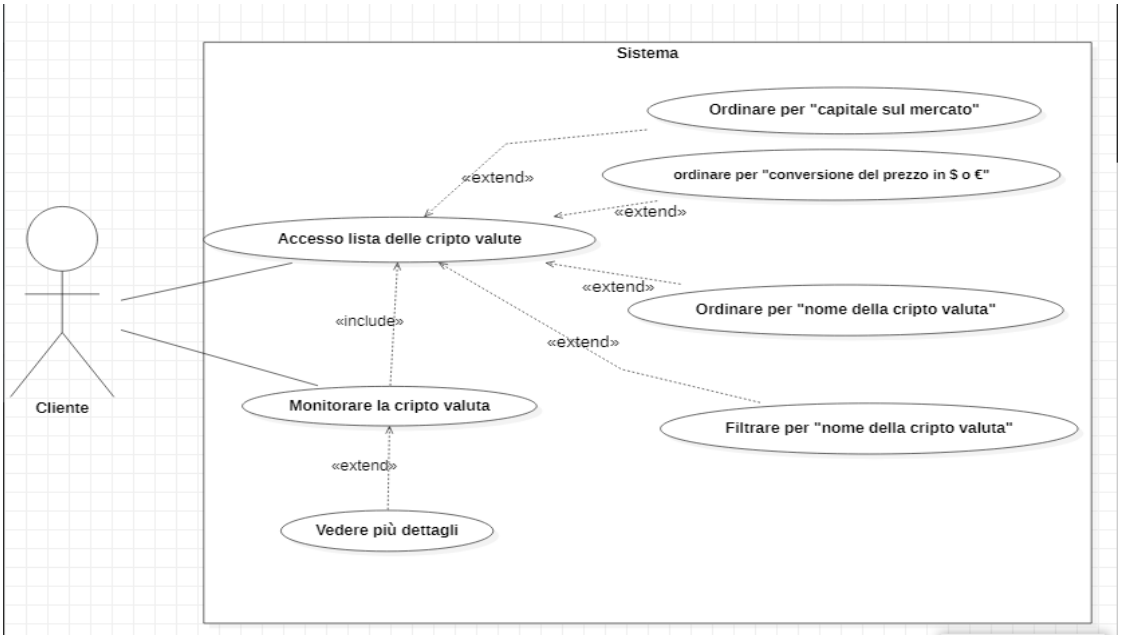
Per la riuscita della nostra applicazione la quale servirà soltanto a monitorare il tasso di cambio delle cripto valute in rapporto al dollaro e all'euro, il server dell'azienda (al quale l'applicazione farà richiesta per l'ottenimento delle informazioni) dovrà essere previsto di un nuovo servizio che aggiornerà in tempo reale le informazioni riguardanti il tasso di cambio.
Le informazioni precedentemente nominate dovranno quindi essere salvate nel database dell'azienda e perciò sarà necessaria anche un aggiornamento del database, aggiungendo quindi un nuovo spazio nel quale esse verranno salvate.
Si suppone poi che il sito web dell'azienda tramite il protocollo HTTP metterà a disposizione delle API (riguardanti le informazioni sul tasso di cambio) in formato JSON le quali saranno disponibili a tutti gli host che cercheranno di farne richiesta, perciò anche alle applicazioni da noi sviluppate.

Punto 1: Analisi per il progetto dell'App

Facendo un analisi per il progetto della nostra applicazione ora cerchiamo di capire quali sono le funzionalità principali che un cliente qualunque potrà far eseguire per interagire con l'app.
Se ne individuano 2 funzionalità principali e 5 secondarie:

- **Accesso alla lista delle cripto valute**
- **Monitorare la singola cripto valuta**
- Ordinare la lista delle cripto valute in base al capitale disponibile sul mercato
- Ordinare la lista delle cripto valute in base alla conversone del prezzo in rapporto al Dollaro o all'Euro
- Ordinare la lista delle cripto valute in base al nome delle cripto valute
- Filtrare la lista delle cripto valute in base al nome della cripto valuta cercata
- Visualizzare maggiori dettagli riguardanti la cripto valuta selezionata

Si vuole infatti che il cliente possa visualizzare innanzitutto la lista di tutte le cripto valute e poi selezionando una di queste possa monitorare il loro tasso di cambio (non ci sarà la possibilità di scegliere se monitorare il tasso di cambio in rapporto o al Dollaro o all'Euro, bensì verranno visualizzate entrambe insieme).
Una volta che si "sorveglia" la cripto valuta selezionata (ad esempio il bitcoin) in essa ci sarà la possibilità di poter visualizzare maggiori dettagli, come ad esempio le variazioni rispetto al giorno precedente, la quantità di moneta in circolazione ecc.
Il cliente avrà poi anche la scelta di poter visualizzare la lista di tutte le cripto valute secondo un certo ordine (quelli elencati di sopra) e anche di filtrarle facendo una query in base al loro nome.



Punto 2: Progetto del servizio REST

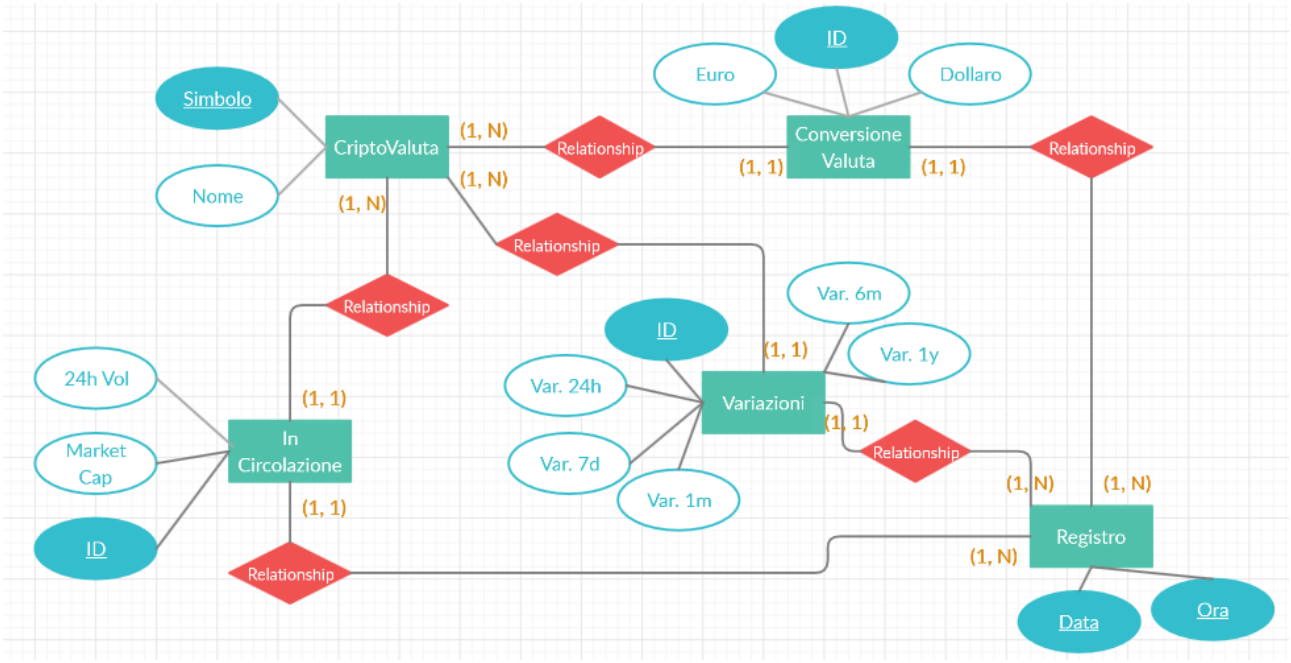
Avendo un'idea più chiara su come vogliamo procedere con la programmazione della nostra applicazione, bisogna pensare poi a quali dati di preciso vogliamo trasmettere alla nostra app e come.
Dobbiamo innanzitutto aggiornare il vecchio database presente nel server, creando degli spazi appostiti nei quali salvare le informazioni sulle nostre cripto valute.
Partendo da un modello concettuale del nostro database si potrebbe pensare di aver bisogno di 5 entità in totale:

- **CriptoValuta** → Entità nella quale verranno salvate le informazioni sulle cripto valute che noi andremmo a vedere; ad esempio il loro nome e la simbologia con la quale vengono presentati nel mercato.
  Dato che la simbologia per ciascuna cripto valuta è sempre unica, il nostro attributo "simbolo" sarà una chiave primaria.
- **ConversioneValuta** → Entità nella quale verranno salvate la quantità di 1 unità della cripto valuta in questione, in dollari statunitensi e in euro.
  Dato che i dati presenti in questa entità si suppone che verranno aggiornati ogni 30 minuti si sente il bisogno di aggiungere un attributo "ID" come chiave primaria.
- **InCircolazione** → Entità nella quale verranno salvate la quantità di cripto valuta messa in circolazione nelle ultime ventiquattro ore e quella totale nel mercato da quando la cripto valuta è stata creata.
  Anche qua dato che, i dati presenti in questa entità si suppone che vengano aggiornati ogni 30 minuti, bisogna identificarli in base ad un attributo "ID" che sarà una chiave primaria.
- **Variazioni** → Entità nella quale verranno inserite tutte le variazioni in percentuale di quella cripto valuta in base ad un certo periodo di tempo; noi ne inseriamo 5 in base alla: variazione rispetto al giorno precedente – variazione rispetto alla settimana precedente – variazione rispetto al mese precedente – variazione rispetto al semestre precedente – variazione rispetto all'anno precedente. E come sempre aggiungiamo anche un attributo "ID" che identifica l'insieme delle variazioni dato che esse cambiano ogni 30 minuti.
- **Registro** → Una delle entità più utili nel database, essa identifica tramite due attributi "Data" e "Ora" (che saranno due chiavi primarie) il momento temporale in cui i dati presenti nelle entità precedenti sono accaduti. Permettendo di distinguere quali sono gli ultimi dati inseriti nel database rispetto a quelli vecchi e di fare una possibile query in futuro nel caso si volesse costruire un grafico sull'andamento delle cripto valute ad esempio.

Ciascuna di queste entità dovrebbero essere collegate in qualche modo tra di loro tramite delle relazioni, quelle individuate sono:

- Relazione uno a molti tra l'entità "CriptoValuta" e le altre tre entità: "ConversioneValuta", "InCircolazione" e "Variazioni". Questo perché ad una singola cripto valuta ci possono essere più informazioni dello stesso tipo (ad esempio la conversione in euro), dato che di esse (conversioni in euro in questo caso) ce ne saranno di nuove ogni 30 minuti.
- Relazione uno a molti tra l'entità "Registro" e le altre tre entità: "ConversioneValuta", "InCircolazione" e "Variazioni". Questo perché ciascun insieme di informazioni (ad esempio l'insieme delle varie variazioni) accadranno in un unico preciso momento temporale che sarà identificato nel registro tramite la data e l'ora, d'altro canto però in un

unico momento temporale (in quella precisa data e ora) potranno essere registrati più informazioni che fanno riferimento alle diverse cripto valute.



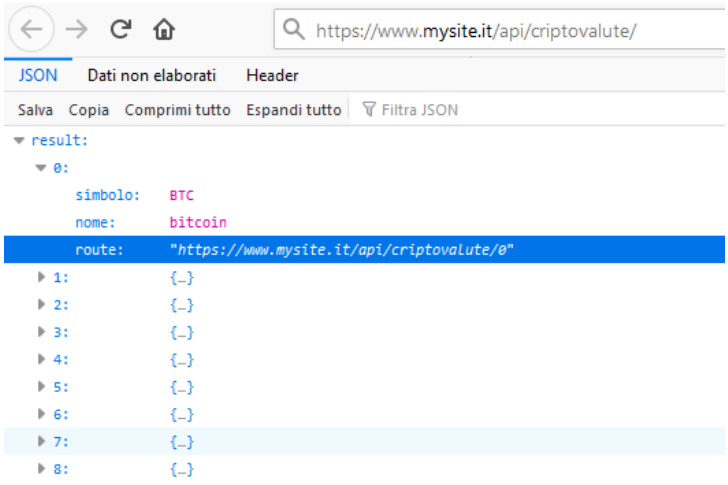
Una volta completato il nostro modello concettuale non ci rimane che convertirlo in un possibile modello logico e poi scriverlo sotto forma di codice tramite un linguaggio di programmazione a noi noto, utilizzando ad esempio il frame work Django.

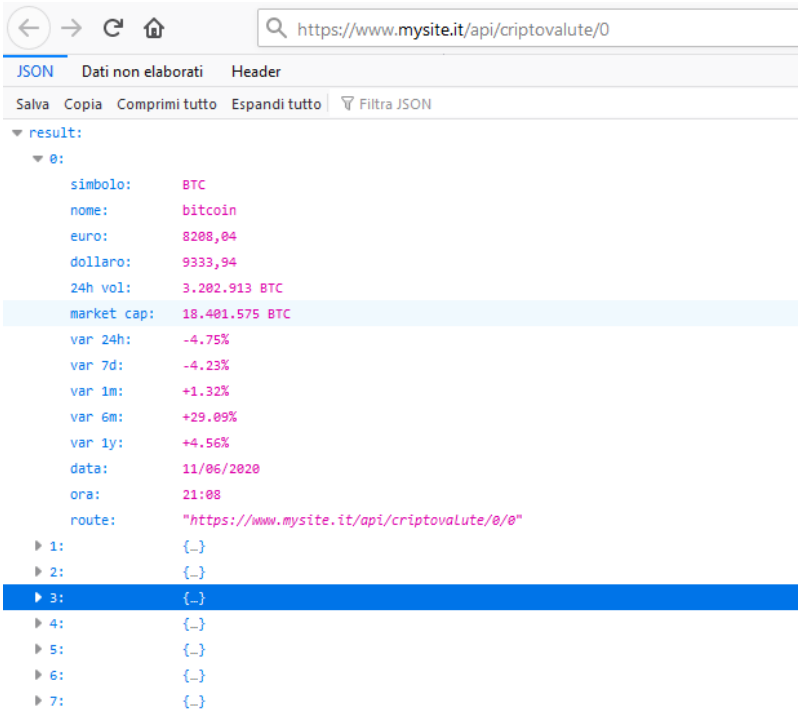
A questo punto non ci rimane altro che elaborare i dati del database e trasformarle in delle API Rest che verranno poi usate dalla nostra applicazione per poter visualizzare nei dispositivi mobile le dovute informazioni.
Prima di procedere con la creazione delle API bisogna fare in modo che il client e il server siano in grado di dialogare tra di loro, e per fare questo bisogna prima concordarsi sull'utilizzo di uno stesso protocollo di comunicazione e di uno stesso formato dei dati. Come abbiamo detto all'inizio, il protocollo di comunicazione da noi usato è quello HTTP mentre il formato di comunicazione sarà un JSON.
Nel protocollo HTTP ci dovrà sempre esserci una richiesta del client al quale poi seguirà una risposta dal server. Solitamente il servizio REST permette l'utilizzo di quattro metodi principali (GET-POST-PUT-DELETE) che indicano il tipo di operazioni che il client richiederà al server; siccome, nel nostro caso, a noi serve soltanto leggere i dati dal server, useremo un unico metodo che sarà il GET.
Ogni richiesta verrà inviata ad un particolare URL che sarà univoco per la risorsa sulla quale richiederemo l'azione. Nel nostro caso i possibili URL che ci serviranno saranno soltanto 3:

- www.mysite.it/api/criptovalute
- www.mysite.it/api/criptovalute/PK
- www.mysite.it/api/criptovalute/PK/PK2

Nel primo link metteremo in formato JSON una lista (e perciò un array) di tutte le cripto valute che si trovano salvate/registrate nel nostro database.
Nel secondo link metteremo in formato JSON una lista (e perciò un array) di una cripto valuta rappresentata da un numero intero (PK), e la suddetta cripto valuta verrà ripetuta tot volte, con informazioni ad essa associata diverse però, in base al numero di aggiornamenti che ha subito il

database da quando è stato creato; con aggiornamenti si intende l'inserimento ogni 30 minuti di nuovi dati.
Nel terzo link metteremo sempre in formato JSON un insieme di dati (rappresentati da un numero intero PK2) di una certa cripto valuta (rappresentato da un numero intero PK) in un certo periodo (rappresentati dalla data e dall'ora contenuti all'interno del JSON).





La risposta del server (una volta ricevuta la richiesta del client) conterrà un codice di stato che ci dirà qual è stato l'esito della richiesta. I possibili codici di stato che noi restituiremo al client saranno:

- **200 "OK"** per indicare un'operazione eseguita con successo.
- **400 "BAD REQUEST"** per indicare che una richiesta non è stata formulata correttamente.
- **404 "NOT FOUND"** per indicare che una risorsa dal client cercato non è stata trovata.
- **500 "INTERNAL SERVER ERROR"** per esprimere un generico errore interno al server.

Nel caso l'esito della richiesta sia corretto, verranno poi inviati i dati in formato JSON al client che provvederà ad elaborarli per mostrarli graficamente nel dispositivo mobile.

Punto 3: Codice significativo dell'App mobile

```dart
import 'dart:async';
import 'package:http/http.dart' show Client;
import 'dart:convert';
import '../models/item_model_lista_criptovalute.dart';
import '../models/item_model_storico_criptovaluta.dart';
import '../models/item_model_criptovaluta.dart';


class ApiProvider {
  Client client = Client();
  const ultimiDati = 0;
  final int criptoValuta;

  ApiProvider(this.criptoValuta);

  Future<ItemModelListaCriptovalute> fetchCriptoValuteList() async {
    final response = await client.get("https://www.mysite.it/api/criptovalute/");
    if (response.statusCode == 200)
      return ItemModelListaCriptovalute.fromJson(json.decode(response.body));
    else
      throw Exception('Failed to load post');
  }

  Future<ItmeModelCriptovaluta> fetchCriptoValuta() async {
    final response = await client.get("https://www.mysite.it/api/criptovalute/$criptoValuta/$ultimiDati");
    if (response.statusCode == 200)
      return ItmeModelCriptovaluta.fromJson(json.decode(response.body));
    else
      throw Exception('Failed to load post');
  }
}
```

L'immagine di sopra mostra una parte di codice che potrebbe essere benissimo una parte molto significativa per lo sviluppo dell'applicazione. Infatti ciò che viene mostrato è semplicemente l'operazione di richiesta delle API e la risposta del server, il tutto dal punto di vista del client. Consideriamo questo codice molto importante perché se esso venisse a mancare, tutti i nostri sforzi fatti fin'ora nel programmare l'applicazione andrebbero persi, per l'appunto proprio perché l'applicazione non avrebbe poi nessun dato da mostrare al cliente che lo utilizza.

Nel codice la prima cosa che facciamo è quella di importare delle librerie che permetteranno la comunicazione col server tramite il protocollo HTTP e di convertire i dati dal formatto JSON in variabili utilizzabili dalla applicazione, in seguito importiamo anche altre classi precedentemente codificate che faranno da supporto al convertitore.

Entrando ora nel cuore del codice, ovvero dentro alla nostra classe "ApiProvider", la prima cosa che viene fatta è quella di inizializzare il client per renderlo pronto alla comunicazione tramite il protocollo HTTP, in seguito viene creata una costante "ultimiDati" che ci permetterà di visualizzare soltanto **i dati più recenti** inseriti nel database, riguardanti una certa cripto valuta; e poi viene creata una variabile "criptoValuta" che verrà inizializzata soltanto quando la classe ApiProvider verrà chiamata da un'altra classe. Quest'ultima variabile servirà ad indicarci **di quale cripto valuta vogliamo sapere i dati più recenti** (quelli disponibili al momento nel mercato).

Inizializzate le variabili, ci troviamo di fronte due metodi:

- **fetchCriptoValuteList()** che ci permetterà di ottenere le API riguardanti la lista di tutte le cripto valute disponibili.

- **fetchCriptoValuta()** che ci permetterà di ottenere le API riguardanti le informazioni su una singola cripto valuta una volta che essa è stata selezionata dalla lista.

I metodi funzionano tutti e due nello stesso identico modo. All'inizio il client fa la richiesta al server tramite un preciso URL e poi si aspetta la risposta del server. Se la risposta del server avesse un esito positivo (ovvero con il codice di stato 200) allora procediamo a convertire il JSON consegnatoci in un agglomerato di variabili, questo agglomerato poi verrà smaltito da un'altra classe (nel nostro caso da **ItemModelListaCriptovalute** e da **ItemModelCriptovaluta**) che salverà in se stessa quelle variabili, da utilizzare più tardi per l'elaborazione grafica.
Nel caso l'esito fosse negativo invece (400-404-500), verrà eseguita un'eccezione.

Punto 4: Analisi delle modalità di protezione della rete

Una volta avute ben chiare le modalità con cui procedere per passare alla programmazione dell'applicazione vera e propria, non ci rimane altro che pensare a come proteggere la rete aziendale da un possibile attacco al sistema. Si parte sempre dai tipi di problemi più frequenti, spostandosi man mano verso quelli più complessi.
Nel nostro caso la prima ipotesi che potrebbe accadere potrebbe essere quella di avere qualche dipendente che involontariamente scarica dei file pericolosi dal web; per prevenire tale problema è sempre buona norma provvedere i vari dispositivi della rete aziendale con un antivirus e se possibile si dovrebbero istruire con un corso pure i propri dipendenti.
Pensando invece a come poter proteggere la rete da veri e propri malintenzionati che cercano di attaccare il sistema aziendale, le prime opzioni sono sempre le stesse, ovvero rifornire la rete locale con un firewall e proteggere la comunicazione tra due host con un sistema di crittografia end to end.
Si potrebbe poi usare la tecnica della segmentazione, ovvero frammentare la propria rete interna in modo da assicurare alle varie parti frammentate di rimanere attive anche nel caso una parte della rete venisse attaccata.
Una volta segmentata la nostra rete, poi, si dovrebbe fare una analisi di essa, cercando di individuare "i singoli punti di errore", ovvero quei singoli punti della rete che in caso di errore, potrebbero creare un malfunzionamento dell'intera parte segmentata della rete.
Dato che molti attacchi dai malintenzionati partono quasi sempre dal tracciamento dell'ip dei server (in alcuni casi anche dei client) è consigliato l'ottenimento anche di una vpn, in modo da impedire una facile tracciabilità.
E infine, ultima ma non meno importante, si consiglia di assumere periodicamente un team specializzato nel campo del "penetration test", ovvero un team che si occupa di attaccare volontariamente il sistema aziendale in modo da individuarne le debolezze.

Sitografia

Devacademy (2019), "REST API: cosa sono, come funzionano e come progettarle", devacademy.it/rest-api-cosa-sono-come-funzionano-e-come-progettarle/

Wikipedia (2020), "Sicurezza e vulnerabilità delle reti", it.wikipedia.org/wiki/Sicurezza_e_vulnerabilità_delle_reti

Mattiello Sergio (2020), "Introduzione a UML", Formato PDF