# Verifica di
# Tecnologie e progettazione di sistemi informatici e di telecomunicazioni

Nome Cognome: _____ Classe: _____ Data: _____

**Per ogni domanda scegli le opzioni corrette (una o più, a seconda della domanda).**
**+1 ogni opzione corretta scelta; 0 ogni opzione corretta non scelta; -0.5 ogni opzione errata scelta.**

**1.** Le API fanno astrazione nel senso che:

- forniscono dati astratti, svincolati dal contesto aziendale;
- forniscono dati astratti, impossibili da comprendere per un umano;
- usano formati di rappresentazione standardizzati dal W3C;
- nascondono i meccanismi dietro all'elaborazione della richiesta;

**2.** Esempi di API:

- DOM;
- CSS;
- HTTP e SMTP;
- HTTP;

**3.** Le API possono essere descritte come:

- Application Protocols for Interaction;
- interfacce machine-to-machine;
- Application Programming Interface;
- strumenti utili per l'interfaccia utente;

**4.** L'uso di API nello sviluppo di un'applicazione:

- comporta sempre un elevato costo iniziale;
- si inserisce in un'ottica di riutilizzo del codice;
- richiede sempre accesso alla rete;
- può tornare utile per scrivere meno codice;

**5.** Le API:

- sono un tipo di Web Services;
- usano obbligatoriamente HTTP per il trasferimento di dati;
- possono usare HTTP per il trasferimento di dati, se in contesto web;

**6.** I Web Services:

- sono Application Protocols for Interaction in contesto web;
- sono interfacce machine-to-machine;
- forniscono risorse in formato machine-readable;
- forniscono pagine web destinate all'utente finale;

**7.** I Web Services formattano i dati principalmente con:

- XML;
- Java;
- YAML;
- PHP;

**8.** SOAP:

- è un protocollo per lo sviluppo di API hardware-SO;
- è un'architettura orientata ai servizi protocollari;
- è un protocollo per Web Services;
- fu il primo modo di realizzare Web Services;

**9.** SOAP, inizialmente acronimo di:

- Services-Oriented Architecture Protocol;
- Services-Oriented Architecture Program;
- Simple Object Access Protocol;
- Solo Object Access Protocol;

**10.** SOAP:

- segue l'architettura REST;
- prevede l'uso di linguaggi comuni per la descrizione delle interfacce e dei dati;
- prevede "buste" da trasferire solo mediante HTTP;
- prevede "buste" da trasferire solo mediante WSDL;

**11.** SOAP ha i seguenti vantaggi:

- dipendenza dalla piattaforma;
- semplicità d'uso;
- permette di pubblicare descrizioni della propria interfaccia;
- velocità, perché riduce al minimo l'incapsulamento;

**12.** REST:

- è un protocollo web basato su HTTP;
- prevede lo scambio di dati in formato XML;
- prevede l'uso di WSDL, come SOAP;
- è oggi l'unico modo per implementare Web Services;

**13.** I RESTful Web Services:

- rispettano i 4 pilastri dell'architettura REST;
- espongono URI gerarchiche che necessitano di estesa documentazione;
- prevedono l'invio di HTTP Response usando solo GET, POST, PUT o DELETE;

**14.** REST:

- disincentiva l'uso di HTTP, perché insicuro;
- disincentiva l'uso delle query string nelle URI;
- è più lento e più semplice se confrontato con SOAP;
- è l'architettura di base necessaria per la realizzazione di SOA;

**15.** Tra i seguenti, non sono pilastri di REST:

- l'uso esplicito dei metodi HTTP;
- l'uso di verbi nelle URI per autodocumentare le interfacce;
- lo scambio di dati in formato XML/JSON;
- l'uso dei metodi CREATE, READ, UPDATE e DELETE di HTTP;

**16.** In ottica RESTful, la HTTP Request con metodo PUT alla URI `/users/123`:

- aggiunge la risorsa utente con id 123 alla lista di utenti;
- aggiorna la risorsa utente con id 123;
- tenta di aggiornare la risorsa utente con id 123 ma fallisce perché non passa in query string il nuovo id;
- non è una richiesta RESTful, poiché l'URI è troppo corta;

**17.** In ottica RESTful, per ottenere la risorsa utente con id 123:

- si deve produrre una HTTP Request con metodo GET alla URI `/users/list.php?id=123`;
- si deve produrre una HTTP Request con metodo GET alla URI `/users/123`;
- si deve produrre una HTTP Request con metodo POST alla URI `/users/list.php?id=123`;
- non si può fare nulla, perché serve per forza ricorrere all'uso delle query string;

**18.** REST è acronimo di REpresentational State Transfer e:

- ciò permette di optare per design stateful dei servizi;
- ciò permette di non mantenere lo stato lato server;
- ciò suggerisce la capacità di rappresentare il trasferimento di una richiesta HTTP;
- tutte le precedenti sono false;

**19.** L'architettura client-server:

- è un primo modello di applicazione distribuita;
- non è un primo modello di applicazione distribuita, perché logica applicativa e di dati sono spesso centralizzate su un dispositivo;
- produce un'architettura peer-to-peer quando divido logica applicativa e di dati su dispositivi diversi;
- prevede che sia il server a esporre le interfacce dei Web Services;

**20.** Potrei optare per un'applicazione distribuita se il mio obiettivo fosse:

- la sicurezza in termini di confidenzialità;
- il parallelismo;
- la semplicità di progettazione;
- la sicurezza in termini di disponibilità;

**21.** Esempi di applicazioni distribuite sono:

- i servizi in Cloud;
- Trivago;
- solo le applicazioni che usano API;
- anche le applicazioni che dividono su calcolatori diversi i tre aspetti UI, logica applicativa e dati;

**22.** Una SOA:

- poggia obbligatoriamente sull'architettura REST, aggiungendo un registro dei servizi;
- è un'architettura che poggia solo su SOAP;
- prevede l'esistenza di un ESB, in grado di fare da singolo punto d'accesso alle interfacce dei servizi;
- prevede l'esistenza di un ESB, gestito da un fornitore master;

**Punteggio:** _____/30
**Alcune domande valgono più di altre!**