REST (Representational State Transfer):

1. **Stateless**: Ogni richiesta tra client e server deve contenere tutte le informazioni necessarie per comprendere e processare la richiesta. Il server non deve memorizzare informazioni sullo stato delle sessioni tra richieste multiple.
2. **Cacheable**: Le risposte del server possono essere marcate come cacheable o non-cacheable. Se una risposta è cacheable, il client può memorizzarla e riutilizzarla per richieste future, riducendo così la latenza e migliorando le prestazioni.
3. **Client-Server**: L'architettura REST si basa sul modello client-server, in cui il client (solitamente un'applicazione web o mobile) comunica con il server attraverso richieste e risposte HTTP. Questo modello separa le responsabilità tra client e server, permettendo a entrambi di evolversi indipendentemente.
4. **Layered System**: L'architettura REST può essere suddivisa in diversi livelli, ognuno dei quali ha una responsabilità specifica. Questo permette una maggiore modularità e flessibilità nel sistema.
5. **Code on Demand** (opzionale): Il server può estendere temporaneamente o personalizzare la funzionalità del client inviando codice eseguibile, come script JavaScript, che il client può eseguire.
6. **Risorse**: Le REST API si basano sul concetto di risorse, che sono entità o oggetti identificati da un URI (Uniform Resource Identifier). Le risorse possono essere dati, servizi o collezioni di oggetti.
7. **Verbi HTTP**: Le REST API utilizzano i verbi standard del protocollo HTTP per eseguire operazioni sulle risorse, come **GET, POST, PUT, DELETE e PATCH**.
8. **URI e URL**: Le risorse nelle REST API sono identificate tramite URI (Uniform Resource Identifier), che sono stringhe univoche che permettono di localizzare e accedere alle risorse. Gli URL (Uniform Resource Locators) sono un tipo di URI utilizzato per identificare risorse web.
9. Formati di scambio dati: Le REST API utilizzano formati di scambio dati standard, come **JSON o XML**, per trasmettere dati tra client e server.
10. Autenticazione e autorizzazione: Le REST API possono implementare meccanismi di autenticazione e autorizzazione per proteggere l'accesso alle risorse e garantire che solo gli utenti autorizzati possano eseguire determinate operazioni.