Verifica Ilaria Bonelli 28/05/2021

1. Un servizio web permette lo scambio di informazioni tra elaboratori diversi all’interno di una stessa rete, sia essa locale (lan), sia essa distribuita. È quindi un’interfaccia che ci permette di mettere due dispositivi o applicazioni in comunicazione. Vengono realizzati tramite protocolli legati alla connettività web (http) e sono multipiattaforma anche a livello di sistema operativo.Tutte le comunicazioni che usano un servizio web hanno un client (chi fa la richiesta) e un server(quello che risponde) tramite protocollo http, specializzato in collegamenti ipertestuali.
2. Il Response del server mi ridà la versione del protocollo HTTP presente nella request, il codice dello status (404 not found, 500 internal server error, 401 non autorizzato, 200 ok, ecc); un header che mi dice cosa contiene il body, quanti caratteri è lunga la mia risposta,ecc. e infine il body contenente le informazioni restituite al client.
3. In un servizio WCF, oltre ad address e binding, un ruolo fondamentale è rivestito dal contract: esso rappresenta i metodi che un servizio espone all’esterno; il Service Contract mi dice quali sono effettivamente le operazioni disponibili sul servizio (Operation Contract), mentre il Data Contract (che sta all’interno del service contract) è un accordo tra un servizio e un cliente che descrive in modo astratto i dati da scambiare, cioè in cui finiscono le informazioni che io voglio ricevere dal mio client.
4. Un servizio REST si basa esclusivamente sulle info che sono in possesso di chi usa il protocollo http per comunicare. Cosa ci serve quindi? L’url per recuperare le informazioni di quella specifica risorsa (l’endpoint di wcf) e gli http verbs o methods: get, post, put, delete sono i principali e sono in sostanza il read, create, update and delete. Per fare un esempio di chiamata: GET (lista di risorse), url, returns array JSON con le risorse che ho chiesto; GET per singola risorsa richiede di specificare l’ID della risorsa alla fine dell’url; POST, url, body:resource as JSON. PUT,, come il GET per singola risorsa richiede di esplicitare l’ID e di mettere nel body la risorsa richiesta e infine DELETE che richiede semplicemente l’ID della risorsa da cancellare alla fine dell’url.
5. La configurazione del mio progetto per ASP.NET core WebAPI è spalmata in due parti dell’applicazione, la prima sta in appsetting.json (che fa in sostanza le veci dell’app config per wcf), l’altra sta in development.json perchè questo tipo di progetto supporta diversi ambienti di sviluppo (production, staging e development) e se sto facendo development userà le impostazioni del file di development (va a cercare ciò che gli serve nei file preposti in automatico). Nella classe Startup ci sono le informazioni di configurazione del mio applicativo. La classe ha due metodi: configure services e configure: il primo dice alla mia applicazione quali servizi utilizza con la loro configurazione (tipo EF, dependency injection, JSON, Identity,ecc). Nel configure andiamo a definire la pipeline di gestione delle richieste: la richiesta arriva alla mia API dal client e una sequenza di codici (step con “use”) che la manipolano e la modificano è la pipeline di gestione. Nella cartella controllers andiamo a scrivere i metodi con cui la mia API va a funzionare (metodi che gestiscono la richiesta in get, in post, in put, in delete).