# Test di valutazione – Modulo 6 – Microsoft Azure

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Angela |
|  |  | Cognome | Colella |
|  |  | Data | 15/12/2020 |

Leggete attentamente ogni domanda e argomentare quanto più possibile fornendo anche degli esempi.  
ATTENZIONE: Le domande a risposta multipla possono contenere più risposte corrette.

1. *Qual è il motivo principale per cui le aziende dovrebbero pensare di portare le loro risorse in cloud?*

Perché potrebbero risparmiare sui costi di gestione dell’hardware

Perché il livello di sicurezza è “by design” superiore a quello medio che si può trovare in situazioni “on premise”

Perché le applicazioni sviluppate per il cloud sono più veloci

Perché è possibile scalare in maniera semplice e adeguare le risorse alle richieste del business

1. *Quali tipologie di servizi sono disponibili nell’ambiente Cloud Microsoft Azure?*

IAAS

PAAS

SAAS

Serverless

1. *Fornire un elenco delle tipologie di servizio erogabile in un ambiente cloud (IAAS, PAAS, ecc) e fornire una descrizione di ciascuna di esse, spiegando quando è conveniente utilizzare una tipologia o l’altra*

***SaaS = Software as a Service****. Il software viene fornito come servizio da parte di un provider. L’utente finale può anche non avere alcuna competenza tecnica al fine di utilizzare il servizio. Egli può servirsi del software tramite internet (non si tratta di una applicazione installata in locale), anche ignorando totalmente come è stata fatta l’applicazione. Il vantaggio più evidente consiste nel fatto che gli utenti non si devono preoccupare di intallare e gestire l’applicazione perchè questo compito è svolto per loro dal provider sel SaaS. Inoltre, essendo accessibile da internet, può essere utilizzata da più dispositivi, a differenza di un’applicazione installata in locale.*

***Paas = Platform ad a Service****. In questo caso il consumer finale è lo sviluppatore, il quale potrà sviluppare applicazioni senza preoccuparsi della gestione dell’infrastruttura server sottostante.*

*In pratica, quando l’utente ha bisogno di determinate risorse, le va a creare appositamente. Ad esempio, se durante lo sviluppo di una applicazione si ha bisogno di un database, se ne può creare uno a seconda delle specifiche esigenze (si crea un web server per installarci sopra l’applicazione). In questo modo, lo sviluppatore non ha bisogno del sistemista o di competenze sistemistiche per svolgere il suo lavoro in quanto l’infrasrtuttura sottostante alla piattaforma è gestita dal provider. In sostanza, lo sviluppatore può svolgere il proprio lavoro senza doversi preoccupare di cose che non lo competono. È agevolato anche il lavoro in team e il lavoro da remoto.*

***Iaas = Infrastucture as a Service.***  *Il target finale è in questo caso il sistemista (o un team di lavoro composto da sviluppatori e sistemisti). I vantaggi di utilizzare questo tipo di servizio, consistono in nella possibilità di pagare un harware a consumo, invece di acquistarlo; nel risparmio del costo di manutenzione di tale harware e nelle scalabilità dell’infrastruttura a seconda delle necessità.*

***Serverless****. Anche in questo caso il target finale è lo sviluppatore, il quale si può concentrare sulla creazione di applicazioni senza dover gestire l’infrastruttura. Dal punto di vista dell’utente, è, quindi, piuttosto simile al PaaS, ma con alcune differenze. Una di esse consiste nel fatto che il costo delle risorse non è fisso, un’altra consiste nella effettiva possibilità di auto-scaling. Infatti, in questo caso non si scala una applicazione intera ma si può scalare solo una funzione dell’applicazione. È utilissima nei casi in cui non serva una elevata potenza di calcolo per tutta l’applicazione ma solo per una parte di essa. È una sorta di “servizio nel servizio”.*

1. *Quali sono le differenze principali tra un database SQL Azure e Azure CosmosDb e, in generale, tra un database relazionale e un database documentale NoSQL?*

*SQL Azure è un db relazionale, mentre Azure CosmoDb è un db documentale (è l’analogo microsoft di MongoDb).*

|  |  |
| --- | --- |
| ***RRELAZIONALE*** | ***DOCUMENTALE*** |
| *Costituito da tabelle, ciascuna delle quali ha vari record* | *Costituto da oggetti (con eventualmente innestati degli oggetti)*  *non strutturati, come ad esempio documenti.*  *Ad esempio, in formato json.* |
| *È possibile impostare relazioni tra tabelle. La struttura del db si basa proprio sulle tabelle e le relazioni tra esse* | *Non si impostano relazioni* |
| *Si possono fare join* | *Non si possono fare join* |
| *C’è logica di business in quanto si impostano relazioni* | *Non c’è logica di business* |
| *Possono essere impostati constraints* | *Non c’è controllo sui dati (no costraint) a parte sull’id, il quale deve essere univoco. Va denominato \_id e, se non viene fornito, viene aggiunto automaticamente* |
| *Più lento* | *Più veloce (ottimo per processare big data)* |

1. *Descrivere alcune delle funzionalità dell’Azure Management Portal, per come è stato utilizzato durante le lezioni*

*I servizi offerti dal Portale di Azure sono svariati: si possono creare risorse con caratteristiche (e costi, ovviamente) differenti a seconda delle esigenze. Si possono creare database SQL, db documentali (cosmo db), App Services, Macchine virtuali e tanto altro.*

1. *Dare una descrizione delle principali differenze che ci sono tra un deploy (una pubblicazione) di una applicazione su Azure Virtual Machine e una su Azure AppService. Classificare i due servizi tra IaaS, PaaS o SaaS.*
2. *Quali sono le funzioni principali di Azure DevOps? Dare una descrizione delle funzionalità più importanti che sono state mostrate in azione durante le lezioni, e descrivere nel dettaglio per quale motivo un’azienda dovrebbe iniziare a pensare di dotarsi di sistemi di ALM strutturati.*

*Su Azure DevOps, dopo aver creato un progetto, si può accedere ad una gamma di funzionalità che racchiudono tutto il processo di ALM (Apllication Lifecycle Management). Questo approccio consente di avere visibilità su tutto il processo di sviluppo: si può vedere ad esempio quale membro del teams si sta occupando di cosa, si può vedere quali processi/fasi sono state portate a termine, qual è il tempo necessario per concludere le varie attività rimanenti, quali test sono stati passati con successo e quali no, ecc..*

*In Azure DevOps c’è un vasto menù, che comprende le seguenti sezioni:*

***Overview***

*• Wiki: in questa sezione si mette la documentazione architetturale e evenutale altra documentazione*

***Boards****: contine tutte le iterazioni (sprint). Consente la gestione dei task.*

***Repository****: consente di visualizzare e gestire i repository del progetto (si possono avere più repository in un solo progetto).*

***Pipelines****: contiene appunto le pipeline, utilizzate per automatizzare processi di build, test e deploy.*

***Artifacts****: condivisione di codice tra più progetti.*

*Generalmente per impostare il lavoro si parte da Backlogs (in Boards):*

*• Si stabilisce un’epica (mileston target cioè obiettivo comune a cleinte e sviluppatore. Un’epica corrisponde ad un obiettivo di business)*

*• Nelle features (sottocategorie dell’epica) si mettono tutte le richieste del cliente.*

*• In ciascuna feature si può aggiungere un backlog (user story), che descrive cosa si vuole fare. Sostituisce in qualche modo la documentazione.*

*Ogni user story, convoglia in tanti task, ciascuno dei quali verrà assegnato ad una persona. A questo punto si stimerà anche quanto tempo servirà per implementare una funzionalità (sezione effort hours). A questo punto si sta facendo una stima iniziale.*

*Invece delle ore si possono assegnare degli story points da 1 a 10 che indicano la difficoltà del task. È una metrica non assoluta, perchè dipende anche dalla velocità, dalla capacità del team e dal modello che si prende di paragone come task difficilissimo (da 3 o più giorni) o come task semplicissimo (10 min).*

*La valutazine delle ore/della difficoltà dei tasks viene poi fatta da tutto il team in un secondo momento (stima condivisa).*

*A questo punto si imposta la prima iterazione (sprint): si determina una data di inizio e si prevede una data di fine. Di solito gli sprint sono di 2 settimane, cioè 10 gg lavorativi, che è il tempo minimo per:*

*• assorbire dei requisiti (1 gg)*

*• fare il design dell’applicazione (1 gg)*

*• sviluppo ad alto livello (creazione delle strutture) (1 gg)*

*• sviluppo (4gg)*

*• fixing (1 gg)*

*• test (1 gg)*

*• deploy (1 gg)*

*I gg di sviluppo di fatto sono 4 di hard core + 1 di alto livello + 1 di testing.*

*Un tempo più lungo non andrebbe bene perchè nel caso peggiore viene buttato via lo sprint interamente, quindi se lo sprint è lungo significa perdere tantissimo tempo.*

*Generalmente nel primo sprint gli sviluppatori non vengono coinvolti.*

*Alla fine di ogni sprint viene rilasciata una versione funzionante (deploy), che quindi deve aver superato con successo i test richiesti.*

*Ad ogni sprint andrebbero ritestate tutte le funzionalità nuove ma anche tutte quelle vecchie. Servirà quindi fare automation: per questo si utilizzano le pipeline, che in automatico ad ogni push sul repository si occupano di fare build, test ed eventualmente deploy dell’applicaizone.*

1. *Descrivere un processo di ALM, partendo dalla raccolta dei requisiti, per arrivare alla fase Deploy del sistema. Definire quali figure professionali sono coinvolte nelle varie fasi, e quali di queste figure possono essere aiutate (o rimpiazzate) con sistemi di “automation”.*

*Se si adotta una metodologia iterativa, le fasi descritte in seguito non vengono svolte una di seguito all’altra senza più possibilità di tornare indeitro, ma vengono portate avanti in parallelo.*

* *il product owner (il proprietario del prodotto finale) spiega cosa desidera che faccia l’applicazione*
* *i “desideri” del product owner vengono tradotti in requisiti da parte dei Business Analyst e dall’architteto e vengono concordati con tutto il team*
* *disegno della struttura dell’applicazione da parte dell’architetto, in modo tale che successivamente gli sviluppatori possano avere delle basi solide su cui iniziare lo sviluppo*
* *gli sviluppatori iniziano ad implementare l’applicazione a partire dalla struttura impostata precedentemente (implementation)*
* *testing*
* *deploy*

1. Dare una descrizione del concetto di Container; poi fornire la motivazione per cui la “containerizzazione” di una applicazione può essere più efficiente del deploy della stessa applicazione in uno scenario IaaS o PaaS. Se le si conoscono, elencare le maggiori differenze che esistono tra una macchina virtuale e un container.

I container consentono di astrarre le applicazione software. In pratica, a differenza delle Macchine Viruali, che permettono di virtualizzare solo l’infrastruttura informatica, i container sono in grado di ottimizzare le risorse hardware a disposizione utilizzando il sistema operativo della macchina host, invece di averne uno proprio. In questo modo non si ha spreco di risorse, si ottiene una scalabilità automatica delle risorse (vengono usate solo se necessarie), si ha una maggiore velocità di funzionamento in quando non è necessario ogni volta inizializzare un ambiente per ciascun container.

1. *Realizzare una applicazione console utilizzando, .NET Core 3.1, che sia in grado di adempiere alle seguenti funzionalità:* 
   * *Chiedere il nome di un database server, il nome del database, uno username e la password da linea di comando*
   * *Comporre la stringa di connessione del database usando i dati precedenti e mostrarla a video*
   * *Collegarsi ad un database SQL Azure (da creare sulla propria sottoscrizione, in caso di mancanza di sottoscrizione chiedere ad una collega di creare un secondo database, aggiungendo l’abilitazione dell’indirizzo IP del computer locale usando “Set firewall rules”).*
   * *Creare una tabella “Products\_[nome]” con i seguenti campi: “Code” (string), “Name” (string) e “Description” (string), Price (int). (Suggerimento: Lanciare il comando “CREATE TABLE…” usando il comando “ExecuteNonQuery di ADO.NET sulla connessione aperta, direttamente da applicazione)*
   * *Inserire un oggetto Product nella tabella su database SQL Azure, eseguendo una “INSERT…” da ADO.NET. Le informazioni del prodotto devono essere chieste all’utente*
   * *Inserire un secondo oggetto “Product”, con dati differenti dal precedente (sempre richiedendo i dati all’utente)*
   * *Estrarre le lista dei prodotti contenuti nel database*
   * *Visualizzare a video (nella console) la lista dei prodotti*
   * *Se volete usare Entity Framework sarà conteggiato un punto Extra!*

*La soluzione deve essere consegnata compressa in un file ZIP insieme alla prova di valutazione.*

[Tot: \_\_/10]