Whitepaper

Modernizzazione delle applicazioni: guida strategica per l'innovazione digitale

Nel panorama tecnologico odierno, una delle principali sfide che le aziende devono fronteggiare è mantenere il proprio ecosistema software costantemente aggiornato e al passo coi tempi.

La presenza di sistemi antiquati e di diverse forme di obsolescenza accumulatesi col passare del tempo all'interno delle stesse organizzazioni rende il tutto ancor più complesso: come si può, dunque, affrontare l'esigenza di innovazione e competitività richiesta per la propria infrastruttura IT?

Insight

Cos'è una Modern Application?

Un'applicazione moderna, o "modern application", è un sistema software che presenta architettura, tecnologie, processi di sviluppo e modelli operativi allineati con le più recenti best practices, metodologie e tendenze tecniche e di mercato. Questi sistemi sono progettati con una concezione cloud-native, ossia con una progettazione realizzata con il cloud come motore principale, in grado di facilitare la realizzazione di sistemi distribuiti e a microservizi, integrabili modularmente in altri ecosistemi, utilizzabili da molteplici piattaforme e che possano garantire affidabilità, scalabilità, mantenibilità e agilità nella gestione del proprio ciclo di vita, nonché un'esperienza di utilizzo di qualità per tutti gli utenti.

Questi sistemi sono solitamente sviluppati seguendo metodologie agili, supportate da discipline DevOps e da tool che consentono di automatizzare i processi di build, deploy e testing, ottenendo cicli rapidi di rilascio e favorendo un continuo miglioramento del sistema attraverso un'analisi regolare di feedback.

Perché è importante parlare di Modern Application?

Porsi l'obiettivo di rendere "moderne" le proprie applicazioni non è un vezzo o frutto della volontà di seguire una tendenza, ma è piuttosto una scelta che porta a benefici tangibili: basti pensare alle difficoltà nel mantenere sistemi complessi, ed aggiornarli in tempi brevi minimizzando i disservizi e gli effetti indesiderati, e agli oneri, economici e non, che ne derivano. Questa difficoltà può essere efficacemente affrontata ricorrendo ad esempio ad architetture a microservizi, nei quali il rischio di propagazione di malfunzionamenti si riduce drasticamente in quanto ogni "modulo" che compone l'ecosistema software può vivere di vita propria, con specifici processi di aggiornamento e controllo qualità.

Anche la necessità di rendere il proprio business più agile richiede un approccio moderno. Le condizioni di mercato e le esigenze dei clienti cambiano ora più rapidamente che mai. Le aziende devono fornire nuove funzionalità, integrarsi con gli ecosistemi dei partner e sperimentare con tecnologie emergenti (come AI e IoT) rapidamente. I sistemi legacy spesso bloccano tali iniziative perché qualsiasi cambiamento è lento e rischioso.

Inoltre, aderire ai più elevati standard di sicurezza e conformità è più semplice con piattaforme moderne. I fornitori di cloud investono massicciamente in sicurezza fisica e informatica: ad esempio, i principali cloud mantengono data center fisici con rigorosi controlli di accesso e offrono crittografia integrata e gestione dell'identità. Sotto il modello di responsabilità condivisa, il fornitore garantisce la sicurezza del cloud (infrastruttura, hardware, virtualizzazione), mentre il cliente è responsabile della sicurezza di quanto è contenuto nel cloud (dati, politiche di accesso, ecc.). In pratica, ciò libera le aziende da gran parte dello sforzo routinario di garantire la sicurezza dell'hardware e dei servizi di base, permettendo loro di concentrarsi sulla sicurezza del proprio codice e delle configurazioni. Questo miglioramento della sicurezza – combinato con l'elasticità e l'affidabilità dei servizi cloud – si traduce direttamente in SLA di uptime più elevati (spesso 99.9% o più) e una migliore protezione e tecniche di disaster recovery, che sono cruciali per i servizi digitali.

Il debito tecnico

Le architetture moderne e i servizi cloud offrono infine potenti modi per gestire e ridurre il debito tecnico. Questo termine si riferisce ai costi nascosti e al lavoro futuro accumulati quando vengono utilizzate pratiche software non ottimali ma convenienti per raggiungere obiettivi a breve termine. Così come il debito finanziario accumula interessi se non viene estinto, il debito tecnico si accumula nel tempo: scorciatoie a breve termine nella

codifica, nel design o nella documentazione possono far risparmiare tempo in un primo momento, ma creano sforzo extra, bug e inflessibilità nel futuro. Fonti comuni di debito tecnico includono codice obsoleto o mal strutturato, test automatizzati insufficienti, architetture monolitiche difficili da modificare e persino hardware o infrastrutture legacy che non possono essere facilmente gestite e potenziate.

Le piattaforme cloud incoraggiano l'applicazione di best practices, come l'uso di managed services e un approccio infrastructure-as-code, che sostituiscono operazioni manuali soggette a errori con l'automazione. Ad esempio, l'automazione di processi di build tramite pipeline consente di rilevare regressioni o problematiche sin dai primi istanti. Inoltre, concetti cloud-native come l'infrastruttura immutabile e funzioni serverless consentono di distribuire unità di codice "nuove e pulite" piuttosto che rilasciare patch o aggiornare incrementalmente codice su vecchi server.

Cosa si intende per "modernizzazione"?

La modernizzazione delle applicazioni è il processo di aggiornamento delle applicazioni esistenti di un'organizzazione per renderle più allineate con le esigenze aziendali attuali e le migliori pratiche tecnologiche. In un contesto cloud, generalmente significa passare da distribuzioni on-premises o obsolete a un modello cloud-first. Questo potrebbe comportare il trasferimento delle applicazioni su infrastrutture cloud pubbliche o private, il refactoring in microservizi, la riprogettazione delle architetture dei dati o l'adozione di nuove metodologie di sviluppo. La definizione di Microsoft è emblematica: "La modernizzazione delle applicazioni è il processo di aggiornamento delle app e dei dati attuali a un modello cloud-first per allinearsi alle esigenze aziendali".[1]

Un'importante sfumatura è che la modernizzazione delle applicazioni non deve necessariamente coincidere con una riscrittura completa del software o con stravolgimenti estremamente impattanti. Molte aziende seguono infatti un approccio graduale: possono iniziare migrando un componente critico a un servizio gestito (ad esempio, spostando un database on-prem a un'istanza di database cloud), quindi gradualmente riadattare il codice dei sistemi annessi per utilizzare le API di quel servizio. Oppure possono containerizzare parti di un'applicazione (senza modificarne pesantemente il codice) per aumentarne la portabilità e successivamente evolvere quei container in microservizi. Esistono strumenti e framework che possono supportare in questo processo.

La modernizzazione delle applicazioni mira, dunque, a migliorare le prestazioni organizzative e tecniche, accelerare il time to market e migliorare le esperienze degli utenti. La modernizzazione implica l'adozione dei processi moderni correlati richiesti dalle applicazioni moderne, implicando necessariamente anche un cambio culturale all'interno dell'organizzazione che decide di optare per questo percorso. Spesso si va infatti di pari passo con una strategia di trasformazione digitale più ampia: passando a un'organizzazione più agile e scalabile, le aziende possono rispondere più rapidamente al feedback dei clienti e ai cambiamenti del mercato. Una storia di successo potrebbe coinvolgere un'azienda che sceglie di ristrutturare un proprio servizio online fondamentale in microservizi, permettendo a più team di distribuire funzionalità indipendentemente e riducendo i cicli di rilascio da mesi a giorni.

In breve, la modernizzazione delle applicazioni riguarda l'allineamento delle applicazioni con modelli architetturali moderni e capacità cloud in modo che possano meglio servire l'azienda. Si riconosce che ciò che funzionava in passato (sistemi strettamente accoppiati, rilasci a cascata, server on-prem) potrebbe non essere più sufficiente in un mondo centrato sul cloud e guidato dai clienti. Per questo la modernizzazione è importante: perché prepara l'azienda al futuro.

Problemi risolti dalla modernizzazione e benefici per il business

Molte applicazioni legacy, come abbiamo visto, soffrono di una serie di problemi che la modernizzazione mira specificamente a risolvere:

- **Complessità:** le applicazioni monolitiche spesso cercano di gestire tutte le funzionalità e i casi d'uso in un unico codice, diventando estremamente grandi e complicate. Gli sviluppatori faticano a comprenderne i componenti, il che porta a uno sviluppo lento e frequenti bug. Le moderne architetture a microservizi affrontano questo problema dividendo un'app in servizi più piccoli e focalizzati su specifici domini.
- **Fragilità:** in un sistema monolitico o con componenti strettamente dipendenti l'uno dall'altro, un singolo bug può far crollare l'intera applicazione. Le app moderne enfatizzano l'isolamento dei guasti: se un microservizio fallisce, gli altri possono continuare a funzionare, spesso reindirizzando le richieste o gestendo performance degradate.
- **Mancanza di specializzazione:** i sistemi più vecchi erano spesso costruiti da team generalisti che li gestivano in toto. Questo poteva portare alla realizzazione di software con troppi compiti, mescolando interfaccia utente, logica aziendale e gestione dei dati. Il design delle app moderne

favorisce la specializzazione: team o servizi distinti possiedono ciascuno una specifica capacità.

- **Lentezza e scarse prestazioni:** il codice legacy spesso presuppone un utilizzo delle risorse uniforme, o manca di tecniche quali caching e elaborazione parallela. Un approccio moderno identifica le esigenze uniche di risorse di ciascun componente (intenso utilizzo di CPU, intenso ricorso ad operazioni di I/O, elevato uso di memoria ecc...) e li esegue su istanze cloud con dimensioni e configurazioni appropriate.
- **Scarsa qualità dei test e feedback lento:** i processi legacy spesso si basavano su test manuali o cicli di QA lenti e infrequenti. Ogni modifica poteva innescare un test di regressione lunghi giorni, il che significava che anche le piccole correzioni potevano richiedere tempi lunghi per essere rilasciate. Le metodologie moderne applicano tecniche di testing continuo: unit test, integration test e performance test vengono eseguiti su ogni modifica del codice.

La modernizzazione delle applicazioni offre anche una serie di benefici per le organizzazioni. Anche se ogni situazione è diversa, l'esperienza del settore rivela diversi vantaggi comuni:

- **Scalabilità su richiesta:** uno dei benefici maggiormente riconosciuti è la capacità di scalare per soddisfare la domanda digitale. Le piattaforme cloud forniscono scalabilità automatica orizzontale e verticale: ad esempio, un servizio web può avviare altre istanze di container man mano che il traffico aumenta.
- **Ottimizzazione dei costi:** le architetture moderne consentono un utilizzo delle risorse molto più granulare. Nell'IT tradizionale, le aziende spesso acquistano server in eccesso per gestire carichi di picco rari; nel resto del tempo, quella capacità rimane inutilizzata ma continua a rappresentare un costo. Con il cloud, le risorse sono elastiche: si "dimensionano automaticamente".
- **Miglioramento della sicurezza:** la modernizzazione migliora la sicurezza su più fronti. Innanzitutto, come notato in precedenza, i provider cloud si occupano di mantenere datacenter altamente sicuri e gestiscono la sicurezza dell'infrastruttura. Inoltre, un approccio cloud-first tipicamente prevede l'integrazione di sistemi di sicurezza direttamente nelle pipeline che implementano gli automatismi, abbracciando un paradigma DevSecOps: test di sicurezza

automatizzati, scansione del codice e verifiche sulla gestione dei segreti applicativi assicurano che le vulnerabilità siano rilevate precocemente e continuamente, piuttosto che dopo il rilascio.

- **Distribuzione delle funzionalità più rapida:** una volta che un'app viene modernizzata, i team possono consegnare nuove funzionalità e aggiornamenti molto più rapidamente. Questo è in parte dovuto a fattori tecnici (ad esempio, il design modulare implica meno dipendenze da coordinare) e in parte culturali (le pratiche Agile/DevOps accelerano la collaborazione). L'automazione è un moltiplicatore di forza: il deployment continuo significa che una modifica testata può andare in produzione in qualsiasi momento, e può persino essere ripristinata in sicurezza se sorgono problemi.
- **Miglioramento dell'esperienza utente:** le app moderne, per design, tendono a fornire un'esperienza più fluida e affidabile, mentre le architetture ad alta disponibilità assicurano minor downtime. I meccanismi di monitoraggio automatico e "auto-guarigione" (come il riavvio dei container su cui si verificano errori) mantengono il servizio attivo e disponibile, ed i framework moderni di front-end offrono interfacce utente più reattive e interattive. Il supporto mobile e multipiattaforma permette agli utenti di interagire con i servizi da qualsiasi dispositivo.
- **Resilienza:** La ridondanza incorporata del cloud (grazie a caratteristiche come le zone di disponibilità multiple, i backup automatizzati e molte altre funzionalità) migliora notevolmente la resilienza di un'app. Questo si traduce in accordi di livello di servizio (SLA) più forti e un incremento del rapporto di fiducia con utenti e\o partner.
- **Produttività e umore degli sviluppatori:** un beneficio meno ovvio ma altrettanto importante è l'impatto sul team di sviluppo e sull'annessa organizzazione. Le piattaforme moderne permettono di rimuovere attività ripetitive, permettendo agli sviluppatori di concentrarsi sulla logica aziendale e sul valore aggiunto che sono in grado di fornire, di innovare e sperimentare. Nel tempo, questo crea un ciclo virtuoso: sviluppatori più felici possono realizzare prodotti migliori e più velocemente.

L'utente al centro

Un tratto distintivo delle applicazioni moderne è il loro focus sull'utente finale. Mentre il software aziendale legacy spesso era costruito attorno ai processi interni e alla convenienza sul puro fronte IT, le app moderne sono concepite come "assistenti digitali personali" che si adattano perfettamente ai flussi di lavoro degli utenti. Questo approccio centrato sull'utente guida ogni decisione nel design e nello sviluppo:

- **Interazione intuitiva:** le applicazioni moderne prioritizzano la facilità d'uso e l'accessibilità. Le interfacce sono progettate con i principi dell'esperienza utente (UX) al centro. Ciò si traduce in flussi di lavoro semplificati, clic minimi per completare le attività e design visivo coerente.
- **Funzionalità contestuali e proattive:** invece di attendere passivamente l'input degli utenti, le app moderne avanzate spesso includono elementi proattivi. Potrebbero mostrare informazioni rilevanti in base al contesto (ad esempio, posizione, ora del giorno, ruolo dell'utente) o automatizzare attività ripetitive.
- **Accessibilità e personalizzazione:** le app moderne sono generalmente costruite con standard di accessibilità in mente (supportando lettori di schermo, navigazione tramite tastiera, ecc.), così da servire tutti gli utenti. Possono anche adattarsi alle preferenze dell'utente – modalità scura, impostazioni linguistiche o canali di notifica preferiti.
- **Funzionalità di collaborazione e social:** riconoscendo che molte attività sono collaborative, le app moderne spesso includono funzionalità di collaborazione ed integrazioni social.
- **Prestazioni e feedback:** nulla frustra gli utenti più di app lente o di errori poco chiari. Il design moderno tenta di risolvere questo aspetto enfatizzando la reattività dell'esperienza. Tecniche come il caching lato client, il caricamento asincrono e il monitoraggio delle prestazioni aiutano a garantire che l'app restituisca un feedback di operatività più veloce.
- **Mobile-first e multipiattaforma:** oggi gli utenti desiderano un'esperienza coerente su tutti i dispositivi. Un'app aziendale moderna potrebbe essere accessibile tramite browser desktop, app mobile e persino assistente vocale o smartwatch. Progettare un percorso utente multipiattaforma è quindi cruciale.

Adottando un approccio centrato sull'utente, le applicazioni moderne diventano più che semplici strumenti: diventano compagni di lavoro che semplificano la complessità.

Approcci alla modernizzazione

Quando si pianifica un'iniziativa di modernizzazione, le organizzazioni considerano generalmente un insieme di strategie e tecniche. Piuttosto che definire un'unica soluzione per tutte le casistiche analizzate, è bene riconoscere che diverse applicazioni e sistemi richiedono trattamenti differenti. I framework di settore spesso definiscono tre principali approcci alla modernizzazione:

- **Rehosting (noto anche come "Lift-and-shift"):** questo approccio prevede la migrazione di un sistema verso il cloud con modifiche minime o nulle, e si concretizza molto spesso nell'ottenere immagini di macchine virtuali da server on-premise e distribuirle in un ambiente IaaS (Infrastructure as a Service) cloud. Esistono tool a supporto che possono automatizzare gran parte di questo processo. Il rehosting è popolare perché è rapido, generalmente meno costoso e può portare ad immediati risparmi sui costi dell'infrastruttura: le uniche attività necessarie possono essere limitate a task relativi all'infrastruttura, come la configurazione della macchina virtuale che ospiterà il sistema e l'applicazione di strategie di networking volte a garantire il necessario livello di sicurezza e gestione del traffico.
- **Replatforming:** si tratta di una strategia più profonda rispetto al Rehosting, e mira a compiere ottimizzazioni specifiche durante la migrazione. Essa pone le sue basi nell'utilizzo delle offerte PaaS (Platform as a Service) cloud, apportando modifiche minime all'applicazione, al database e al sistema in generale. Un esempio tipico consiste nella distribuzione delle applicazioni in servizi come Azure App Services e nello spostamento del database verso soluzioni gestite come Amazon RDS MySQL o Azure SQL. Può essere visto come un compromesso molto comune tra i vari approcci di modernizzazione, poiché offre vantaggi importanti come la rimozione dell'overhead dovuto alla gestione di un server on-premise o di una macchina virtuale, facilita la configurazione delle varie componenti software e non richiede una completa revisione del sistema coinvolto.
- **Full modernization:** consiste nello spostare l'architettura del sistema sorgente verso un'architettura full-cloud. Questo è l'approccio più costoso, sia dal punto di vista del budget che del tempo necessario per portare a termine il processo, ma offre una trasformazione profonda e conferisce al sistema un assetto "future-proof". Questo approccio è a sua volta suddiviso in tre modalità diverse, in base a quanti elementi del sistema sorgente vengono impattati o riscritti per meglio adattarsi alla nuova architettura cloud-native:
  - **Refactor:** il codice viene ristrutturato e ottimizzato, senza apportare cambiamenti comportamentali, per rimuovere il debito tecnico e migliorare i requisiti non funzionali, meglio conformandosi agli elementi cloud in cui viene distribuito.
  - **Rearchitect:** consiste nell'alterare profondamente il codice per supportare un'architettura molto diversa rispetto a quella sorgente, sfruttando appieno le nuove capacità cloud-native.
  - **Rebuild:** il sistema viene riprogettato e reimplementato da zero mantenendo il suo scopo e le specifiche. Questo si rende necessario quando, rispetto alla sua prima progettazione, il contesto o i requisiti sono cambiati in modo radicale e non risulta ragionevole, sia per scelte di business, sia per scelte economiche, sia per scelte tecniche, riadattare quanto già realizzato. Quando si opta per una riscrittura completa, si parla di sviluppo cloud-native, ed in tale contesto si ricorre sin dall'inizio delle attività a tecnologie e scelte di design tipiche del cloud, come architetture a microservizi, containerizzazione, implementazione di funzioni serverless e strategie di distribuzione di risorse cloud basate su una definizione dei componenti in modalità Infrastructure-as-Code (IaC).

Vale la pena sottolineare che la modernizzazione non è una decisione che comporta un cambio immediato, ma un percorso. Ad esempio, un'organizzazione potrebbe prima optare per il Rehosting di un'applicazione per compiere rapidamente le prime ottimizzazioni, poi in una fase successiva, una volta visti i benefici del cloud, operare un Refactoring dei componenti critici. Ogni passo incrementale è volta a mitigare il debito tecnico e ad offrire una maggiore efficienza operativa, anche prima che la trasformazione completa sia ultimata.

Soluzioni Low-Code\No-Code

Quando i requisiti del sistema da implementare (o re-implementare) sono relativamente semplici e non sussistono esigenze di business o tecniche specifiche e complesse, alcune organizzazioni decidono di virare verso piattaforme che consentono di realizzare soluzioni "Low-Code" o "No-Code" (LCNC).

Queste piattaforme (come Microsoft Power Platform) consentono di creare rapidamente applicazioni attraverso interfacce visive, componenti drag-and-drop e modelli precostruiti, e accelerano notevolmente il time to market perché astraggono gran parte della complessità del codice sottostante. Anche i non-sviluppatori possono creare app semplici, flussi di lavoro e moduli, mentre gli sviluppatori professionisti possono avvalersi degli strumenti LCNC per accelerare attività di routine (come l'implementazione di pagine CRUD o

flussi di approvazione), liberando tempo per lavori più complessi.

I vantaggi di queste soluzioni includono facilità d'uso, presenza di best practices by-design e disponibilità di connettori spesso integrati (a fonti di dati cloud, provider di identità, ecc.). Anche la manutenzione e gli aggiornamenti possono risultare più semplici, poiché il fornitore della piattaforma si occupa di molte problematiche infrastrutturali. Microsoft, ad esempio, sottolinea che Power Apps può generare applicazioni con un codice minimo, abbassando le barriere di ingresso e migliorando la produttività.

Tuttavia, ci sono compromessi. Le soluzioni LCNC di solito conferiscono limitate possibilità di personalizzazione, e potrebbero non gestire logiche molto complesse o carichi di transazione elevati. Come sottolinea Microsoft, il low-code può essere limitante per i progetti più complessi e potrebbe non sempre offrire la capacità di scalare in concomitanza con la crescita aziendale. Inoltre, integrare un'app low-code con sistemi legacy o hardware specializzati può essere impegnativo.

In realtà, molte organizzazioni adottano una strategia ibrida. Ricorrono alle piattaforme low-code per applicazioni non critiche o specifiche (strumenti di nota spese, semplici dashboard, strumenti di pianificazione, ecc.) e prediligono implementazioni custom per le applicazioni più importanti per il business. Alcune piattaforme LCNC sono estensibili tramite componenti di codice personalizzati, permettendo ai team, quando necessario, di colmare il divario tra soluzioni LCNC e sviluppo custom.

Conclusioni

Disporre di applicazioni moderne, tramite implementazioni ex-novo o dopo un processo di modernizzazione, non è semplicemente uno dei tanti elementi dell'agenda IT, bensì una scelta strategica che si basa sull'introduzione di asset fondamentali per l'organizzazione. Abbracciando completamente i principi delle app moderne – dallo sviluppo agile ad una concezione cloud-native dei sistemi – le organizzazioni possono costruire soluzioni robuste, scalabili e soddisfacenti per l'utente e per l'azienda stessa. Questo, a sua volta, favorisce l'innovazione, la soddisfazione del cliente e degli stakeholder e il vantaggio competitivo.

Infine, è bene ricordare che la modernizzazione è spesso uno sforzo collaborativo. Partner tecnologici, come Insight, offrono competenze nella valutazione dell'architettura, nell'implementazione delle soluzioni e nel change management. I servizi di Insight come la consulenza sulla modernizzazione delle app, la consulenza DevOps, la progettazione di soluzioni cloud e lo sviluppo low-code dimostrano come consulenti esperti possano offrire supporto per disegnare ed accelerare questo percorso. Coinvolgere specialisti e apprendere dalle migliori pratiche del settore può massimizzare i benefici dei processi di trasformazione e modernizzazione.

1] What is Application Modernization? | Microsoft Azure