Pianificare lo sviluppo

da dove iniziare

Top Down

Si parte da una visione d'insieme e poco per volta si *specializza*.

Divide et impera:

- Identificazione delle macro funzionalità
- Suddivisione in sotto funzionalità
- Implementazione
- Ricostruzione della piramide

valorizza il **perché**, da cui il **come** emerge

Bottom Up

Si parte da un'idea e si va iterativamente verso una visione, tramite *composizione*.

- Soluzioni già esistenti
- Soluzioni facili
- Creazione di unità funzionali

parte da un **come**, cercando di fare emergere un perchè idale per *POC develpment* (proof of concept «*dimostrare la fattibilità*») e validazione di soluzioni tecniche *quick'n'dirty*

Core Approach

Si fa un design *top down* ma si sviluppa *bottom up*.
Ordine di sviluppo

- le aree di maggiore rapporto complessità / valore
- le aree di alto valore e minore complessità
- le aree di basso valore e bassa complessità
- esclusione delle aree di basso valore e alta complessità

Mock

Componenti software (oggetti / api / database) che **fingono** una funzionalità ritornando staticamente dei dati. Utili per sviluppare e testare componenti senza attendere altri sviluppi.

```
const getBookByID = async function(bookId) {
  return {
    id: bookId,
    title: "mocked book from db",
    isbn: "fakeisbn0123",
    ...{ otherStaticData },
  };
};
```

Raccolta Requisiti

- requisiti aziendali / stakeholder / funzionali / di soluzione
- surveys
- indagini di mercato
- analisi dei competitor
- analisi del legacy
- restrizioni di qualità

=> output: lista di requisiti

Analisi

Si valutano i requisiti e si immagina un'esperienza utente, scoprendo le caratteristiche funzionali che il software dovrà soddisfare.
Per ogni punto si fa una valutazione della priorità e del rischio.
Solitamente è quello che viene firmato con il cliente prima di iniziare a *fare*. Definisce lo *scope* del progetto.

=> output: analisi funzionale

Design

Si valuta come implementare tecnicamente, architetturalmente e implementativamente la lista di richieste dell'analisi funzionale.
Si creano le **strutture dati** (data model), **i flussi**, i diagrammi delle **classi** (UML), le integrazioni con **sistemi terzi** e le interfacce per le **API**, e si documentano a fondo le decisioni prese.
Una buona analisi tecnica permette a *posteriori* di essere una guida per i tecnici che dovranno mettere mano alla piattaforma software.
=> output: analisi tecnica

Sviluppo

Nella fase di sviluppo si passa dall'analisi funzionale al codice, seguendo la guida dell'analisi tecnica.
Vengono applicate qui la più grande quantità di metodologie per la qualità del codice: test unitari, test end2end, test di integrazione, linting, code review, static code analysis, eccetera...

=> output: codice sorgente + **deploy** (test / demo / collaudo / integration)

Test

Il sofware viene testato dai membri del team di sviluppo seguendo l'analisi funzionale.
Ogni richiesta non soddisfatta o anomalia nel funzionamento porta alla creazione di un *bug* e fa ripartire un flusso di **sviluppo**.
In questa fase viene anche fatto il dummy user testing, per verificare la solidità della soluzione creata.
=> output: testbook (e correzione dei bug)

Validazione – Collaudo

Il testbook viene validato dagli stakeholder, e eseguito da utenti estranei al team di sviluppo.
In questa fase si decide se è possibile *andare in produzione* oppure è necessario un nuovo flusso di sviluppo.

=> output: via libera alla produzione

Deploy (GoLive!)

Operazione di rilascio in produzione del sofware.
Potrebbe essere necessario bloccare temporaneamente altri attori e inibire l'accesso a determinati sistemi per la durata del rilascio.
Alla fine del rilascio *tecnico* viene eseguito un testbook per validare che il software sia operativo e senza alterazioni.
Parallalemente viene redatto il manuale utente e i documenti tecnici necessari al mantenimento.
=> output: software **live** + documentazione

Mantenimento

Questa fase è di duranta uguale alla vita del software.
Comprende tutte le attività manuali di routine:

- pulizia spazi e tabelle temporanee
- aggiornamento dei sistemi operativi ospitanti
- aggiornamenti delle librerie del sistema
- correzione di bug e anomalie
- indicizzazione e storicizzazione dei dati

Evolutive (CR)

Sono le richieste aggiuntive ad un progetto esistente.
Per un blocco di evolutive si segue tutto il processo, dai requisiti al mantenimento.

Ruoli
UX Designer

Aiuta l'analista nella fase di definizione di *come sarà il software*.
Utilizza varie metodologie per *pensare* il software più efficace possibile e indirizzare eventuali requisiti non emersi in prima fase.
Tramite la creazione di **wireframes** fornisce al cliente un'anteprima dell'esperienza utente e al team di sviluppo un'idea del desiderata

Ruoli
Dev

Mettiamo sotto il gruppo dev tutti quelli che si occupano di sviluppo, ossia dell'attività di scrivere del codice

Ruoli
Frontend developer

Il programmatore di frontend si occupa del livello di presentazione del progetto, solitamente le interfacce grafiche. Non esiste una regola che definisce dove finisce il compito di un FE dev e dove inizia quello di un BE dev, ma sono i team / le organizzazioni a stabilirlo.

Un bravo FE dev è in grado di scrivere sia test unitari sia test automatici.

In fase di lavorazione il FE dev lavora usando un mock backend

Ruoli
Backend developer

Lo sviluppatore di backend si occupa del livello di business logic del progetto, scrive e implementa il cuore delle applicazioni, le interazioni con i database, le API.

Un bravo BE dev è in grado di scrivere sia test unitari che test di integrazione end to end

Ruoli
Fullstack developer

Lo sviluppatore fullstack si muove agilmente sia nel BE che nel FE.

Non è necessario che sia eccellente in entrambi gli ambiti, ma sapersi orientare su tutto il codice gli permette di avere un visione d'insieme maggiore.

Ragionare fullstack è fondamentale quando si usano quei design pattern che non dividono il sofware tra presentazione e business logic.

Ruoli
Web developer

Il web developer si occupa di sviluppo di piccoli siti web.

La differenza con un Frontend developer o un Fullstack developer è che questi ultimi utilizzano le tecnologie web per creare applicazioni web, mentre invece il web developer si occupa di siti di medio-piccola entità e bassa complessità.

Tipicamente il web developer non lavora in team, ma è un freelance solitario..

Ruoli
DB developer

Il db-dev è colui che si occupa dello sviluppo (e test + manutenzione) di *stored procedure*, *triggers*, *db functions* et simile.

La difficoltà del versionamento di tali soluzioni e la poca praticità di mantenimento stanno spostando le logiche fuori dal database e questo ruolo sta scomparendo, comunque capita ancora di trovare annunci di lavoro per db-dev.

Ruoli
Tester

Il tester si occupa di redare, eseguire e mantenere il test-case. Si occupa di verificare che le implementazioni siano completate e identificare eventuali regressioni.

Lavora a stretto contatto con il team di sviluppo, sovente ne è parte. Ha una formazione tecnica ma non una spiccata abilità nello sviluppo.

Un buon tester è una figura molto difficile da trovare e può fare la differenza in un team.

Ruoli
Technical Leader

Si occupa della guida tecnica del team. Redige l'analisi tecnica del progetto ed è il referente tecnologico degli stakeholder.

Prende le decisioni di tipo tecnico e infrastrutturale e aiuta il team nel raggiungimento del traguardo, anche sviluppando lui stesso.

Ruoli
Project Manager (PM)

E' responsabile della riuscita del progetto.
Si occupa di tempi e costi ed è colui che decide la composizione del team.

Si fa aiutare per le questioni funzionali dall'analista, per le questioni tecniche dal technical leader

Ruoli
IT Operations (OPS)

E' responsabile dei livelli più bassi del progetto, anche se solitamente il suo lavoro è di *servizio*, ma non prende parte alle dinamiche del team.

Si occupa di networking, server fisici, sistemi operativi, backup, configurazioni degli application server / web server.

Con l'avvento del *cloud computing* si occupa di configurare le macchine virtuali, le reti e le infrastrutture più o meno complesse, nei servizi cloud (*aws*, *azure*, *digitalocean*, *heroku*, eccetera).

Ruoli
Dev-OPS

E' il punto di congiunzione tra un dev e un ops. Si occupa di gestire le infrastrutture cloud tramite strumenti automatizzati di deploy, provisioning, integrazione, mantenimento e orchestrazione.

La direzione è quella del infrastructure as code (IaC), dove le operazioni di infrastruttura vengono definite da definizioni di operazioni tramite codice, standardizzato e versionabile, invece di utilizzare strumenti di configurazione e *conf* file

Ruoli
DBA

L'amministratore del database si occupa di installare, amministrare e ottimizzare le performance di grossi database relazionali.

Può essere assimilato a un ops ma richiede conoscenza specifica profonda del database amministrato.

Si occupa anche di backup, ruoli, accessi, sicurezza

Ruoli
Security Engineer

L'esperto di sicurezza, solitamente esterno al team, si occupa di fare *assessment* di sicurezza sul progetto maturo.

E' un ruolo molto quotato e il numero di esperti di sicurezza bravi è estremamente basso

Ruoli
Stakeholders

Sono tutti i portatori di interesse legati al progetto software.

Ognuno ha la propria necessità di valore dal software e le proprie priorità.

Distinguere tra *utente finale e cliente e finanziatore*

Documentazione Ufficiale

- Analisi Funzionale
- Analisi Tecnica
- Manuale Utente
- Mappe infrastrutturali
- Diagramma delle classi
- Data Model
- Manuale di manutenzione

Documentazione
In project

- README
- CONTRIBUTING
- CHANGELOG
- git log
- codice auto-documentante
- /_ commenti _/ (solo se super necessari)
- configurazioni del linter / IDE

Grazie