Università Politecnica delle Marche


Facoltà di Ingegneria
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione

# "Ticket Management", un'applicazione Web per la gestione di un help desk aziendale

# "Ticket Management", a Web-application to manage a help desk for a company

Relatore:

Prof. Alessandro Cucchiarelli

Candidato:

Davide Massa

Anno accademico 2021/2022

# Indice

# Capitolo 1 - Introduzione

## Introduzione

Negli ultimi anni l'interesse per il settore delle applicazioni basate su architetture distribuite è cresciuto notevolmente, le Web-application ed i Web-services stanno riscuotendo un enorme successo in ambito aziendale, nel privato e nel pubblico, con progetti di smart factory e industria 4.0 che prevedono l'integrazione di tutte le tecnologie al fine di incrementare efficienza produttiva, performances, analisi dei dati ed altri innumerevoli aspetti.

Questo spinge molte grandi aziende che dispongono di considerevoli capitali, ad investire nella ricerca di tecnologie disponibili sul mercato o nello sviluppo e nell'impiego di nuove applicazioni.

Lo scopo del lavoro presentato in questa tesi è quello di illustrare una Web application per la gestione ed il tracciamento di problemi aziendali di vario genere sviluppata per una grande azienda italiana, attiva nel mondo della produzione e distribuzione di beni alimentari in tutto il mondo, dopo che questa ha valutato e scartato una soluzione software già esistente e consolidata.

L'elaborato presenterà una prima parte completamente dedicata all'esposizione di concetti teorici, che serviranno come introduzione sull'argomento al lettore. Nella seconda parte, la più corposa, verranno presentate le tecnologie impiegate ed il software, nella sua interezza, evidenziando caratteristiche generali e sue differenze rispetto ad i competitor.

# Capitolo 2 – Contesto applicativo

In questo capitolo, si andrà ad illustrare una panoramica sull'Industria 4.0 al fine di fornire indicazioni su quali aspetti procedurali e della vita aziendale ad essa possano essere riferiti, oltre che a dare indicazioni su cosa significa beneficiare dell'utilizzo di applicazioni Web.

Più nel dettaglio parlerà degli ERP (Enterprise Resource Planning), software di gestione che integra tutti i processi di business rilevanti di un'azienda, delle loro caratteristiche principali, degli ambiti applicativi e di come la fusione tra concetti di "cloud computing" e "gestionale aziendale" possa costituire un concetto cardine di questo elaborato, e per estensione per le aziende che intendano proiettarsi al futuro.

Infine, il focus si sposterà ancor più nel dettaglio (effettuando anche una comparazione con il principale prodotto offerto dal mercato), sul progetto presentato da questo elaborato, ove l'applicativo dovrebbe esser pensato come componente di un più esteso gestionale aziendale, ed in generale di un sistema di cloud computing che ospiti un ERP.

# Industria 4.0 – Impiego delle Web application

Nel Corso del XIX e del XX secolo si sono succedute rispettivamente tre rivoluzioni industriali che hanno cambiato radicalmente il modo di fare azienda in ambito produttivo/industriale. L'introduzione della meccanica del motore a vapore, l'innovazione della catena di montaggio e l'avvento dei calcolatori elettronici hanno portato innovazioni che hanno rimodellato e cambiato sostanzialmente le modalità di svolgimento dell'attività produttiva, incrementando efficienza, efficacia e volume produttivo. [1]

Quella che stiamo attraversando negli ultimi anni rappresenta la quarta rivoluzione industriale (da qui il termine di industria 4.0), che si pone come obiettivo quello di connettere attraverso la rete globale Internet ed attraverso le reti locali aziendali, i sistemi di automazione, il monitoraggio dei macchinari, il reporting, il cloud computing ed in ultimo ma non per grado di importanza, il mondo degli ERP (Enterprise Resources Planning), come rappresentato graficamente in Figura 1, che include la gestione di molteplici aspetti aziendali e dei quali si parlerà in dettaglio più avanti.

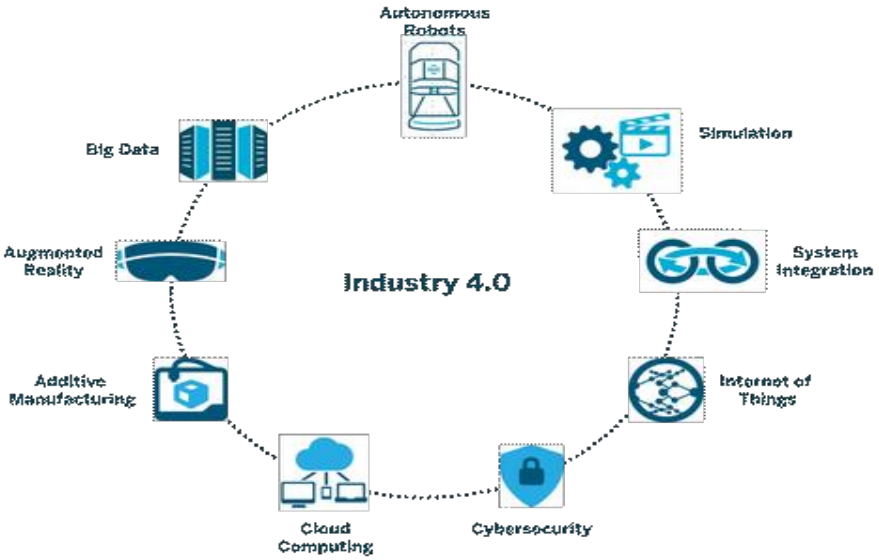



*Figura 1 - Rappresentazione grafica del concetto di Industria 4.0*

L'industria 4.0 passa per il concetto di Smart Factory che a sua volta si compone delle seguenti 3 parti *[2]*:

- **Smart production:** Tecnologie produttive che creano collaborazione tra operatore, macchine e strumenti (PLC, pannelli di interfaccia uomo/macchina)
- **Smart energy:** Impiego preponderante di energia pulita e sostenibile ed utilizzo di tecnologie per riduzione degli sprechi
- **Smart service:** Infrastrutture informatiche che permettono agli operatori, agli impiegati, ai manager ed ai clienti di gestire in prospettiva innumerevoli aspetti ed attività aziendali.

In questo elaborato si parlerà principalmente del mondo degli "smart service", nell'ambito dei quali le "Web application" rivestono certamente un ruolo di primissima importanza. Questo genere di prodotto, infatti, introduce un concetto nuovo nel mondo dell'IT e cioè che il software non viene più inteso come servizio desktop per il quale viene pagata una licenza di utilizzo per un periodo limitato nel tempo ed in cui le risorse hardware impiegate sono principalmente quelle messe a disposizione dall'utilizzatore, ma inteso come servizio erogato dall'azienda produttrice (SAAS – software as a service), utilizzabile su qualsiasi dispositivo, in cui è semplicemente richiesta una autenticazione.

Questa tipologia di architettura rappresentata in figura 2 è la forma più completa di servizi erogati tramite cloud computing che consiste nella fornitura di una intera applicazione gestita da un provider attraverso un browser Web. Il provider si occupa di effettuare gli aggiornamenti al software, della correzione di eventuali bug e della generica manutenzione del prodotto, mentre l'utente non deve fare altro che connettersi alla app mediante una dashboard. Non è, quindi, prevista nessuna installazione sulle singole macchine e l'accesso e l'utilizzo sono in via generale più affidabili sia in termini di sicurezza che per quanto riguarda le performances *[3]*.

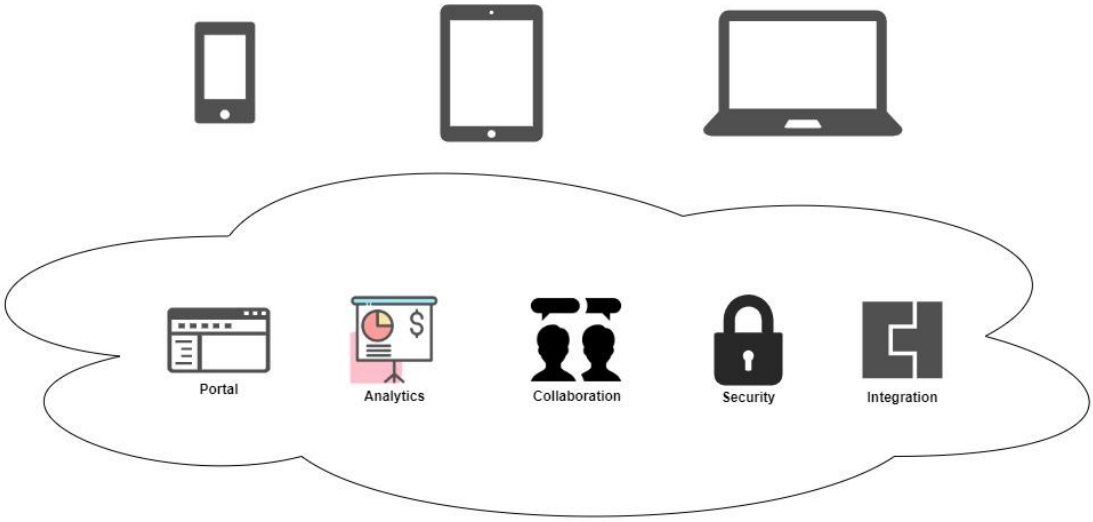



*Figura 2 - Software erogato come servizio disponibile in cloud, e possibilità di accesso da qualsiasi genere di dispositivo. Vengono anche illustrate le principali caratteristiche del SAAS*

Fattorizzando quanto detto in precedenza, si potrebbe riassumere brevemente quali sono i principali vantaggi derivati dall'uso delle Web-application [4]:

- Più utenti possono accedere alla stessa versione del software
- Non necessitano di alcuna installazione
- Si possono utilizzare da qualsiasi genere di dispositivo (es. smartphone, tablet, desktop), poiché possiedono la caratteristica che viene definita "responsiveness", che consentono ad esse di adattarsi a differenti tipologie di dispositivi riorganizzando il contenuto in modo dinamico
- Accesso possibile tramite l'utilizzo di differenti browser Web

Nel prossimo paragrafo verrà introdotto più nel dettaglio il concetto di ERP, che va a coprire tutti quegli ambiti legati alle attività manifatturiere (MES), al monitoring ed al reporting delle stesse, fin alla gestione dei flussi di lavoro aziendali e alla risoluzione di eventuali problemi collegati.

# ERP – Caratteristiche ed ambiti applicativi

In un contesto aziendale, vi sono principalmente due aree funzionali di importanza strategica che oggi possono essere notevolmente espanse, semplificate ed altresì migliorate integrandole ed interfacciandole con i sistemi informativi.

SCM (Supply Chain Management), cioè l'insieme di metodologie gestionali e soluzioni software che consentono all'azienda di gestire in modo efficiente l'intero flusso per l'approvvigionamento, la produzione e la consegna di prodotti e servizi ai clienti; include movimenti di materie prime e semilavorati da un punto di origine ad un punto finale, in una rete di business globale denominata "catena della fornitura" o Supply Chain. L'integrazione di tecnologie del mondo dell'IT permette così la creazione di piattaforme collaborative tra più aziende.

CRM (Customer Relationship Management) è un processo con il quale un'azienda o una generica organizzazione amministra le sue interazioni con i clienti utilizzando in genere le basi di dati per immagazzinare ed analizzare grandi quantità di informazioni. Non si tratta di una tecnologia da acquistare, ma nella sua accezione più generale il riferimento alle strategie che l'azienda mette in atto per l'acquisizione e la fidelizzazione dei propri clienti. [5]

L'impiego di tecnologie del mondo dell'IT permette quindi di compiere analisi per:

- Automatizzare le vendite
- Automatizzare le attività di marketing
- Svolgere attività di data mining per analizzare le tendenze e le abitudini della clientela
- Instaurare meccanismi di collaborazione tra fornitori, distributori ed altre imprese

Una considerazione ed un'evidenza messa in risalto da quanto detto in precedenza è rappresentata dall'esigenza, sempre più importante per le imprese, di raccogliere ed utilizzare i dati di diversa natura e diversa provenienza al fine di conoscere le necessità e le abitudini della propria clientela. Se, infatti, per incrementare il volume produttivo, aumentare l'efficienza, la qualità del prodotto o servizio fornito, occorre semplicemente raccogliere ed utilizzare dati già disponibili per finalità tecniche (setting migliore delle macchine, organizzazione più efficiente dei turni di lavoro, monitoraggio della filiera produttiva), non si può dire lo stesso per la raccolta ed utilizzo delle informazioni utili per finalità di vendita, di marketing e di fidelizzazione del cliente. Questi dati ed informazioni devono, infatti, esser prima reperiti, con la consapevolezza che non sempre questa attività risulta semplice se non si ricorre a tecnologie di cui dispongono i sistemi informativi.

Un ERP (Enterprise Resource Planning) tenta di colmare tutte le lacune dei precedenti sistemi utilizzati nella gestione aziendale. Si tratta di un sistema di gestione che integra tutti i processi aziendali rilevanti quali ad esempio contabilità, acquisti, vendite, gestione del personale e del magazzino, attività di analisi dei dati e molte altre ancora come evidenziato in figura 3. Tutte queste attività risiedono in un unico sistema centrale, il quale risulta essere indispensabile per il supporto alle decisioni del management team. Anche i dati vengono memorizzati in un unico livello protetto di persistenza a cui possono accedere tutte le figure ed applicazioni aziendali che dispongono di autorizzazione. [6]

Un ERP possiede, quindi, tre caratteristiche fondamentali:

- È un sistema standardizzato
- È pensato per le più comuni esigenze, ma comunque completamente configurabile
- Incorpora al suo interno SCM e CRM e loro principali peculiarità


*Figura 3 - [7] Struttura ed aree di utilizzo di un sistema Enterprise Resource Planning*

La Web application "Ticket Management", presentata in questo elaborato, sarà una delle molte applicazioni che compongono un ERP ed avrà il preciso scopo di aiutare e supportare le varie figure professionali dell'impresa nella risoluzione di problematiche che possono riguardare diversi ambiti, da problemi legati all'utilizzo di software aziendali alla gestione di problemi di varia natura. Problematiche ed eventi interni in relazione ai quali, a seguito di espressa segnalazione, occorre intervenire con determinate azioni correttive/risolutive.

Analizzando nuovamente la figura 3 [7], la Web application sviluppata, si pone come obiettivo finale (e lo si evidenzierà anche negli sviluppi futuri) quello di gestire aspetti legati alle risorse umane, alla supply chain management, al customer management, alla gestione finanziaria e, più in generale, al project management.

# Ticket management - Contesto e problemi affrontati

Il Contesto di mercato in cui si pone il software presentato in questo elaborato, è ricco di possibilità e di alternative, dove il prodotto che di gran lunga risulta essere il più utilizzato è "Service Now" [8]. Si tratta di una piattaforma di digitalizzazione e gestione dei flussi di lavoro, nata con l'idea di semplificare l'erogazione, alle aziende, di alcuni servizi caratteristici dell'IT, tramite proposte di soluzioni di Information Technology Service Management (ITSM).

Oggi "Service Now" è molto di più di questo, in quanto avendo ampliato la propria piattaforma "cloud based", mette a disposizione del management aziendale tutta una serie di altri dati e di info necessarie per la gestione di flussi di lavoro, quali: gestione delle risorse umane, dei clienti, filiera produttiva e operation excellence[1].

Service Now semplifica ed agevola le comunicazioni, evitando scambi di e-mail spesso confusionarie, o telefonate che, invece, per loro natura escludono altre figure aziendali che dovrebbero essere informate circa lo stato delle attività. Il tutto grazie all'utilizzo di un singolo portale, accentratore di tutto il flusso di informazioni, chiaro ed esente da errori derivati da attività svolte manualmente. [9]

I servizi offerti da questo programma cloud sono molteplici e possono essere riassunti in modo breve ed intuitivo dalla figura 4:

*1 L'Operation Excellence è un sistema di continuo miglioramento che integra strumenti, approcci e programmi per dare il massimo beneficio e ottenere la massima efficacia nelle operazioni in cui è implicato*

- Apertura/gestione/risoluzione/chiusura di “Incident” (o “request for change” nel caso in cui si renda necessario fare delle modifiche radicali al tool/software/applicazione per il quale si è segnalato il problema)
- Chat dedicata per interloquire in qualsiasi momento con il tecnico che sta risolvendo il problema, o con il team leader/manager di riferimento.
- Sondaggi per la valutazione del servizio offerto
- Knowledge base con tutorial per la risoluzione dei problemi più frequenti
- Dashboards con statistiche e grafici per l’analisi del servizio offerto

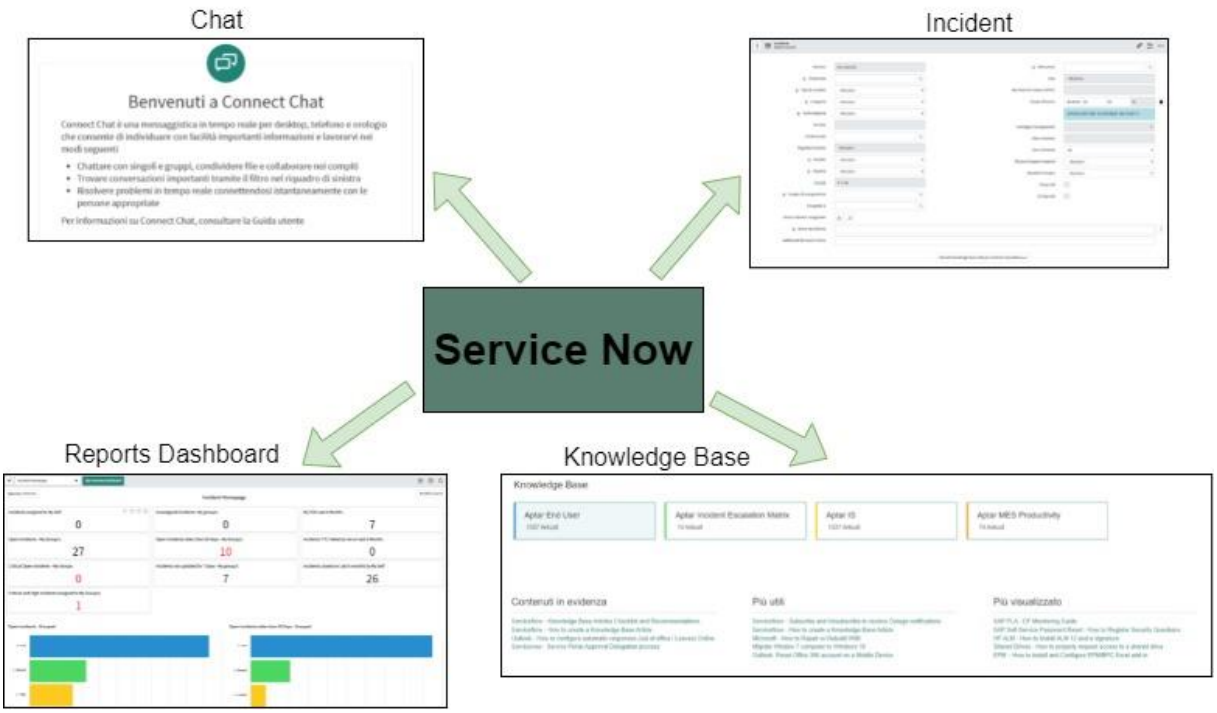



*Figura 4 - Alcune delle più importanti funzionalità offerte da Service Now*

Service Now è, quindi, parte integrante di un ERP aziendale che si occupa di gestire tutti gli aspetti legati al mondo della business intelligence di una impresa che intende proiettarsi al futuro.

Ticket management, l'applicazione sviluppata e presentata in questa relazione, è una Web application che vuol avere l'ambizione di esser competitor di Service Now, seppur ancora acerba ed in fase di ulteriore sviluppo, ma che si vedrà nei capitoli successivi possedere alcuni vantaggi strategici nei confronti di questo e di altri prodotti o servizi della stessa tipologia.

Sviluppata per essere integrata all'interno di un ERP SAP 4/Hana e pubblicata nella suite SAP Fiori (si parlerà più avanti di queste tecnologie SAP), si occuperà principalmente di gestire tutti i flussi di lavoro digitali che saranno categorizzati sotto il nome di "incident" (incidente/problema/bug/investigazione post mortem).

La mission finale della app. Ticket Management è quella di accentrare tutte le informazioni, rendendole altamente disponibili per un futuro riutilizzo.

# Capitolo 3 – Obiettivi del progetto

In questo capitolo verranno illustrati gli obiettivi del progetto "Ticket Management" non prima, però, di aver focalizzato l'attenzione sull'esigenza esposta da più aziende, le quali possono essere riepilogate con le seguenti semplici domande:

- *Possiamo internalizzare l'attività di gestione di incident, senza utilizzare software già presenti sul mercato e non customizzabili?*
- *L'azienda potrà avere la possibilità di customizzare il software?*
- *È possibile abbattere i costi sia in termini di licenze d'uso acquistate che in termini di ulteriori sistemi hardware necessari?*
- *Sarà possibile utilizzare questa nuova applicazione all'interno del nostro ecosistema SAP costruendo così una banca dati comune?*

Queste sono solo alcune delle domande che sono state poste dagli stakeholders[2] e che verranno approfondite nel capitolo 5.

Si parlerà del concetto cardine di tutto il progetto, il ticket, si parlerà dei requisiti funzionali e tecnici che il software dovrà possedere, nonché della rappresentazione dell'organigramma aziendale in funzione dell'utilizzo che se ne farà attraverso l'impiego dei diagrammi dei casi d'uso come strumento per la rappresentazione di queste figure e delle loro azioni sul software.

*2 Gli stakeholders sono quei soggetti influenti nei confronti di una particolare iniziativa economica ed operativa, che hanno l'onere nei confronti dell'azienda di farsi portavoce delle esigenze della stessa.*

# Project scoping form

Il Project scoping form è un documento che raccoglie tutta una serie di informazioni riguardanti il progetto svolto ed offre al contempo una panoramica di quelle che sono le attività da svolgere, gli obiettivi da raggiungere *(hard/soft goals)* e le risorse che all'interno del team sono impegnate nelle fasi di realizzazione del prodotto finale. [10]

Nella tabella sono presenti le seguenti informazioni:

- **Project Name:** Il nome assegnato al progetto
- **Project Managers:** I responsabili del progetto
- **Problem Opportunity:** Quali sono le esigenze che hanno determinato la realizzazione del progetto
- **Project GOAL:** Obiettivo finale da perseguire
- **Objectives, durations, costs:** Vengono definite delle attività, con durate temporali e impegno finanziario per ogni attività. Ad intervalli regolari di tempo e carichi di lavoro si determinano delle milestones[3] di progetto.
- **Success Criteria:** Indica i criteri con i quali verrà giudicata la bontà di un progetto.
- **Assumptions, risks, obstacles:** Si evidenziano quali sono gli eventuali ostacoli ed impedimenti alla realizzazione del progetto.

*3 Nel project management una milestone indica un importante traguardo, normalmente intermedio, che dovrebbe avvenire entro una determinata data, fissata durante l'arco temporale di svolgimento di un progetto. All'interno del progetto possono essere fissate più milestones.*

| PROJECT SCOPING FORM | | |
|---|---|---|
| **Project name:** | **Project manager:** | **Team members:** |
| Ticket Management | Davide Massa | Davide Massa |

**Problem/Opportunity (Why do this project?):**

La mancanza di una vera e propria gestione delle criticità all'interno di una azienda di medio/grandi dimensioni, crea un'opportunità per il team che, intende realizzare un software di help desk, che possa raccogliere tutte le segnalazioni di problematiche di diverso genere. Le segnalazioni verranno gestite sotto forma di ticket, che saranno assegnati dai manager di riferimento, direttamente ad utenti esperti/tecnici, per la loro risoluzione. L'utente, al termine di questa fase, dovrà espressamente o tacitamente confermare la avvenuta risoluzione della problematica.

Questo software permetterebbe di velocizzare tutti i processi interni e di accentrarne le informazioni aumentando, conseguentemente, efficienza e produttività delle risorse umane aziendali.

**Project goal:**

Gli obiettivi di progetto sono essenzialmente i seguenti:

- Permettere ad ogni addetto dell'azienda una facile via per la segnalazione di problemi direttamente dal proprio smartphone/tablet/PC.
- Fornire agli utenti esperti, uno strumento per la gestione efficace delle criticità.
- Velocizzare ed ottimizzare i processi aziendali interni.

**Objectives, duration, costs:**

Il prodotto finale si compone essenzialmente di due parti:

1. La parte di Frontend legata principalmente alla realizzazione della GUI (graphic user interface), alla gestione della lista dei ticket (filtri custom, filtri standard, ordinamenti, raggruppamenti, ricerca) e alla presentazione del dettaglio ticket.
2. La parte di Backend a cui sarà demandato il compito di gestire il Database (dati Tickets, dati utente), implementare la logica di base e gestire tutte le operazioni con lo strato superiore rappresentato dal Frontend.

L'obiettivo finale di questo progetto è nello specifico quello di analizzare, progettare e realizzare il frontend e la logica applicativa del backend.

La persistenza, quindi il database, sarà realizzato da altro team, e la comunicazione tra le componenti attraverso il protocollo ODATA.

I tempi di realizzazione del progetto sono stimabili nell'ordine dei tre mesi.

**Success criteria:**

Il corretto funzionamento del software sarà determinato da una buona efficienza ed efficacia dello stesso, da una facilità di configurazione iniziale, da una interfaccia intuitiva per tutti gli utenti, ma soprattutto da un effettivo incremento della velocità di risoluzione dei problemi interni all'ambiente aziendale.

In ultimo ma non per ordine di importanza, verrà valutata la possibilità di sviluppo e customizzazione sulla base delle richieste future degli stakeholders.

**Assumptions, risks, Obstacles:**

Si presenteranno quasi certamente degli ostacoli nella fase operativa post realizzazione, cioè quella di binding (connessione) del frontend con il backend; la difficoltà principale è infatti legata soprattutto all'eterogeneità delle tecnologie e linguaggi utilizzati per la realizzazione delle due componenti.

# Caso di studio - Il Ticket

Il ticket rappresenta il fulcro dell'intero progetto esposto in questo elaborato: è un processo che nasce nel momento in cui l'utente (un dipendente o dirigente dell'impresa, utilizzatore della Web application) nota un problema all'interno o all'esterno del proprio ambito di lavoro ed apre quindi una segnalazione, e termina nel momento in cui il ticket viene chiuso. In questo lasso di tempo esistono diversi stati possibili, ed ognuno di essi ha un significato ben preciso. Di seguito verrà data collocazione in termini temporali e operativi ad ognuno di essi e si cercherà altresì di rappresentare, attraverso un diagramma a stati finiti[4] mostrato in figura 5, i possibili flussi tra gli stati:

1. **Creato:** Il ticket è stato creato mediante form apposita dall'utente, il quale è tenuto a specificare alcune informazioni molto importanti tra cui: la severity (gravità) e la priority (urgenza) del ticket, una breve descrizione, un titolo e se possibile, immagini/screenshots o video che mostrino il problema.
2. **Approvato:** Il manager del team di primo supporto, dopo aver constatato l'effettiva veridicità della segnalazione, prende in carico il ticket, lo approva e lo assegna alla squadra di tecnici del primo supporto per la sua lavorazione. Il ticket può essere approvato ed assegnato automaticamente dal sistema per problemi ordinari (es. utente con password smarrita) o approvato manualmente per categorie di incidents che richiedono esplicita analisi da parte del manager. Nei prossimi capitoli capiremo quali.
3. **Rigettato:** Allo stesso modo dell'approvazione, un ticket può essere rigettato automaticamente dal sistema o manualmente dal manager o dal tecnico.

*4 Il diagramma a stati finiti è utilizzato per descrivere il comportamento dei sistemi, rappresentato tramite una serie di stati ed eventi che ovviamente devono essere in numero finito.*

Se si ritiene che sia stato già risolto o privo di fondamento valido, lo si rigetta definitivamente (tutti i ticket in stato chiuso costituiscono una knowledge base a disposizione dell'utente). Per qualsiasi altro motivo che renda impossibile momentaneamente la lavorazione, viene rigettato temporaneamente in attesa di altro feedback da parte dell'utente (Es. mancanza di alcune informazioni potenzialmente utili).

4. **Assegnato:** Il ticket viene assegnato dal manager ad un tecnico esperto del team di primo supporto. Il tecnico aprendo la dashboard della Web-app visualizzerà tutti i ticket assegnati ed in attesa di risoluzione. In questa fase l'utente potrà interloquire direttamente con il tecnico attraverso la chat dedicata, al fine di condividere altre informazioni e seguire lo stato di avanzamento. Qualora il tecnico non possieda le necessarie competenze tecniche per portare a compimento il proprio lavoro, può cambiare lo stato da "assegnato" ad "approvato", in tal caso il manager si farà carico di riassegnarlo ad altro tecnico.
5. **Da Verificare:** Il lavoro sul ticket del tecnico esperto è ultimato, si attende l'approvazione finale dell'utente che confermerà l'avvenuta risoluzione del problema.
6. **Riaperto:** Il problema non è stato risolto, il test dell'utente finale ha esito negativo, il ticket viene riaperto e si cerca una nuova soluzione.
7. **Chiuso:** Test/verifica con esito positivo e conseguente chiusura del ticket. Questa azione è un onere dell'utente che ha un tempo limite, Al di fuori di questa deadline (solitamente cinque giorni lavorativi), il sistema chiude in automatico i ticket in stato "da verificare" utilizzando la regola del "silenzio/assenso".

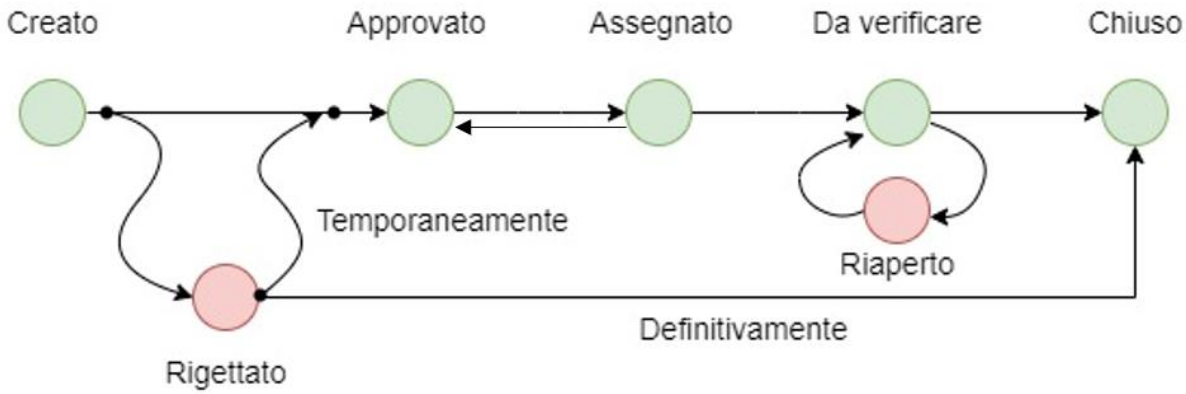



*Figura 5 - Rappresentazione degli stati del ticket in ordine temporale da sinistra verso destra. Lo stato rigettato può essere definitivo o temporaneo, e gli stati 'da verificare' e 'riaperto' si alternano fintanto che il tecnico non ha risolto il problema.*

# Caso di studio - Organigramma Aziendale

Le figure professionali coinvolte nel progetto sono:

- **Utente esperto/tecnico:** L'utente esperto è quella figura appartenente al team di supporto che, in base alla propria area tecnica di competenza, ha l'onere di prendere in carico e lavorare i ticket. Ne consegue che gli utenti esperti sono molteplici, almeno uno per ogni area tecnica (infrastructure, databases, CRM, ERP, MES, productivity software, etc..). Tra tutti gli utenti esperti, spicca il SuperUser, una figura professionale senior e di grande esperienza che conosce perfettamente tutti i flussi interni e riesce a trovare una soluzione laddove si dovesse essere in presenza di problematiche che coinvolgono più di una singola area tecnica, e con una severity particolarmente elevata (problemi critici che impattano talvolta anche un intero stabilimento).

- **Utente Finale:** L'utente è un qualsiasi addetto dell'azienda che in un dato momento della sua attività lavorativa trova un problema nella gestione dei propri flussi digitali. Apre un ticket che deve contenere delle informazioni e può opzionalmente contenerne delle altre, come evidenziato in figura 6:

<table>
<tr><td>Titolo*</td><td>Categoria*<br>Database, ERP, reporting, monitoring, productivity software</td><td>Descrizione problema</td><td colspan="2">Priorità/Gravità*<br>Con una scala di valori da 1 a 4 si attribuisce una priorità ed una gravità</td></tr>
<tr><td colspan="3">Immagini/Video<br>Eventualmente disponibili, può essere fatto un upload di screenshots, immagini o video dimostrativi</td><td>E-mail utente*</td><td>Owner<br>Nel caso in cui un utente esperto apra il ticket per conto di altro utente</td></tr>
</table>

*Figura 6 - In questa tabella vengono mostrate le informazioni per l'apertura di un ticket. Con l'asterisco vengono contrassegnati i campi obbligatori.*

- **Manager:** Ogni area tecnica ha il suo manager di riferimento. Ognuno di essi può decidere in autonomia quali sono le categorie e le caratteristiche per le quali un ticket viene approvato automaticamente dal sistema e viceversa quali sono le categorie per le quali i ticket richiedono una procedura di approvazione manuale. Assegna inoltre i ticket agli utenti esperti del proprio settore, o di altre aree tecniche se ritiene che l'assegnazione alla sua area tecnica sia errata. Può infine decidere di rigettare temporaneamente o definitivamente il ticket se ritiene che ve ne siano ragioni di farlo.
- **Supermanager:** È la figura più importante nell'organigramma, svolge una funzione di garanzia e controllo nei confronti dell'operato dei manager di settore, può controllare a campione il lavoro svolto dagli utenti esperti, può fare le veci di un qualsiasi manager in caso di estrema necessità o urgenza.

Si fa notare come sulla figura del Manager ricadano gran parte delle responsabilità. Tutto

ciò inevitabilmente si ripercuote sulla percezione, giusta o sbagliata, che l'utente potrà avere dell'intera applicazione. Pertanto, si è deciso, in fase di analisi dei requisiti, che l'operato del manager dovesse essere posto sotto il controllo/supervisione del Supermanager. L'organigramma presenterà una struttura piramidale rappresentata in figura 7:

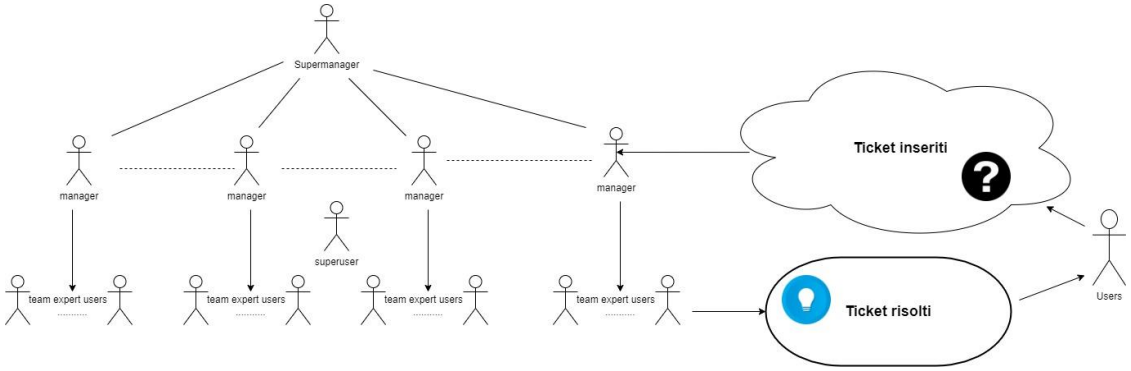



*Figura 7 - Organigramma aziendale. Utenti e Super Utenti qui rappresentati all'esterno della piramide, potrebbero anche farne parte.*

# Casi d'uso

In UML[5] il diagramma di un caso d'uso è un grafico che cita brevemente una storia riguardante il modo in cui l'utente finale (o altre figure), interagiscono con il sistema in un determinato insieme di circostanze. La descrizione può essere narrativa, una sequenza di compiti o interazioni, una descrizione basata su modelli, o una rappresentazione schematica [9].

Il primo passo nella realizzazione di un caso d'uso consiste nel fornire un insieme di "attori" che saranno coinvolti nell'azione. Gli attori sono le varie persone o dispositivi che utilizzeranno l'applicazione nel contesto della funzione e il comportamento che deve essere descritto [10].

*5 - UML acronimo di Unified modeling language è un linguaggio di modellazione basato sul paradigma orientato agli oggetti.*

L'associazione tra attore e suo caso d'uso può essere di due differenti tipologie:

1. **<<include/use>>:** Indica che un caso d'uso, per la sua esecuzione, necessità di un altro caso d'uso (es. per la verifica del dettaglio ticket c'è bisogno di autenticarsi nel sistema)
2. **<<extend>>:** Questa relazione lega due casi d'uso quando il secondo (da cui la freccia della relazione parte), aggiunge una funzionalità al primo al verificarsi di specifiche condizioni.

Verranno di seguito presentati i casi d'uso per i 4 attori, evidenziando per ognuno le principali interazioni con Ticket Management.

***Utente finale – figura 8:*** Svolge tre differenti azioni sulla piattaforma: apre un ticket inserendo le informazioni obbligatorie ed aggiungendo eventuali file a supporto, controlla lo stato di avanzamento del ticket attraverso la lista ticket o comunicando direttamente con l'utente esperto attraverso la chat dedicata, effettua il test per confermare o rigettare la soluzione proposta ed applicata dal tecnico (riapre il ticket).

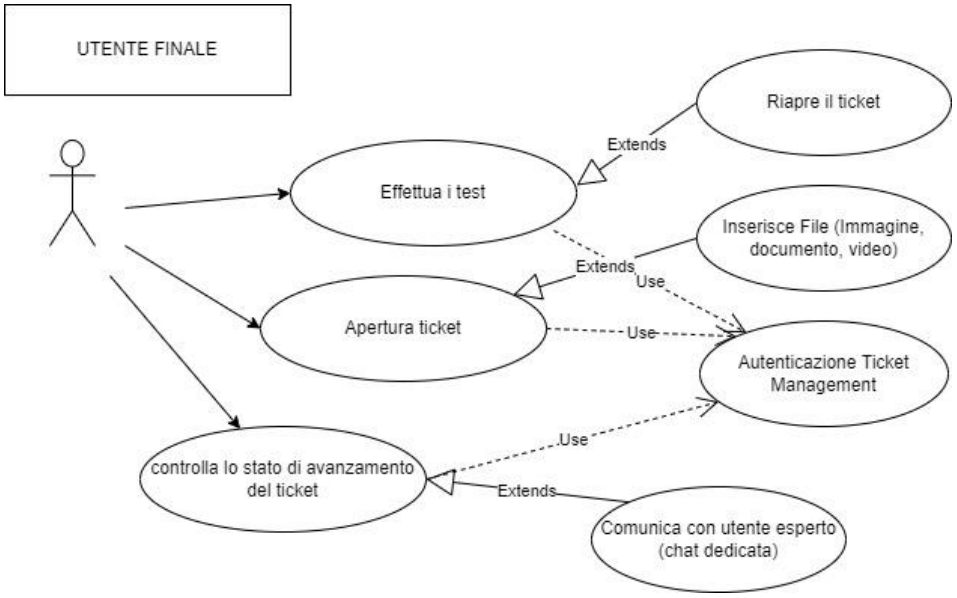



*Figura 8 - Caso d'uso utente finale*

***Utente esperto – figura 9:*** Il tecnico/utente esperto svolge 3 azioni:

- Accetta/rifiuta l'assegnazione di un determinato ticket e, di conseguenza, lavora per la sua risoluzione o chiede al proprio manager di riassegnare il task ad un altro utente esperto
- Apre un ticket con diverso owner
- Apre un ticket per conto proprio

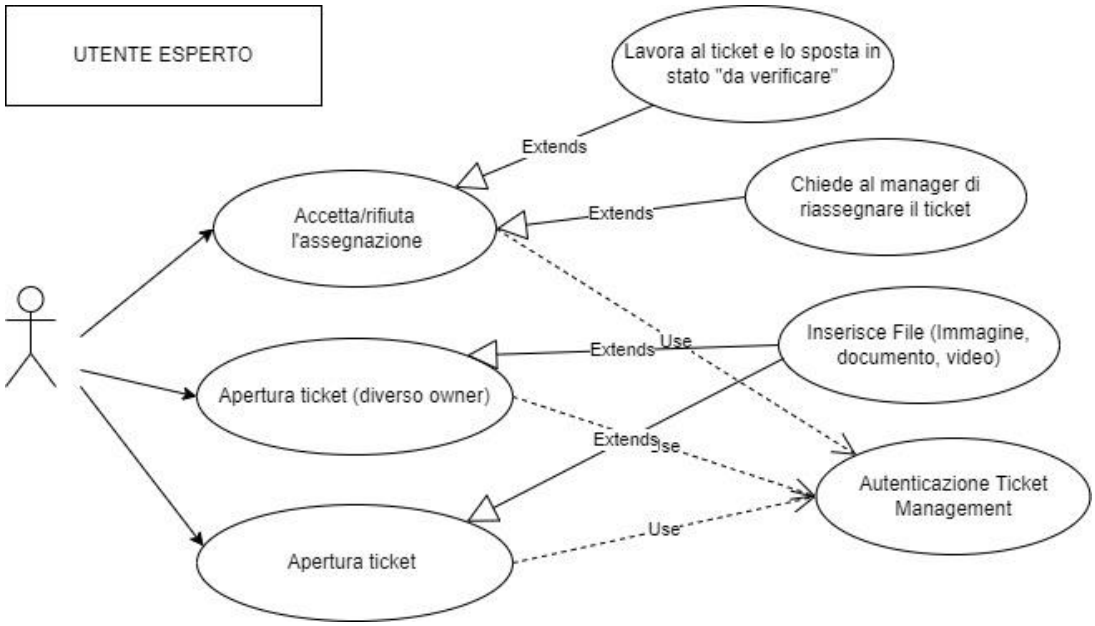



*Figura 9 - Caso d'uso Utente esperto/tecnico*

***Manager – figura 10:*** Il manager approva ed assegna manualmente i ticket, ed in caso non siano stati assegnati al corretto utente esperto, li riassegna. Monitora lo stato dei ticket assegnati agli utenti esperti del proprio team. La Web-application gestisce le approvazioni automatiche sulla base delle regole definite dal manager (es. i ticket di reset/recupero

passwords sono solitamente approvati automaticamente dal sistema).

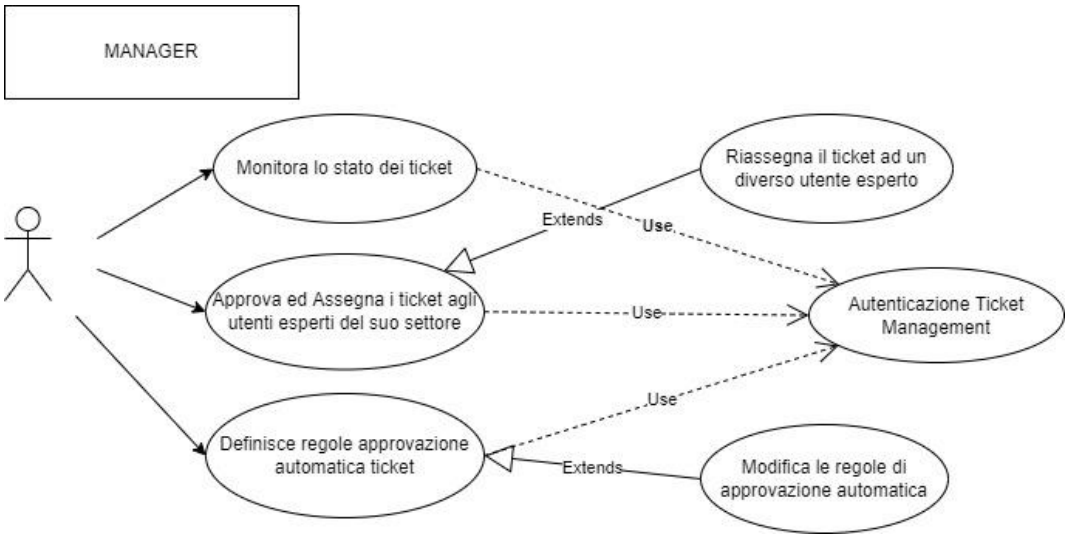



*Figura 10 - Caso d'uso manager*

***Supermanager – figura 11:*** Il supermanager supervisiona il lavoro di tutti i manager, sostituisce uno o più manager nelle decisioni in caso di escalation o di assenza del manager, controlla il lavoro degli utenti esperti attraverso la dashboard della app.

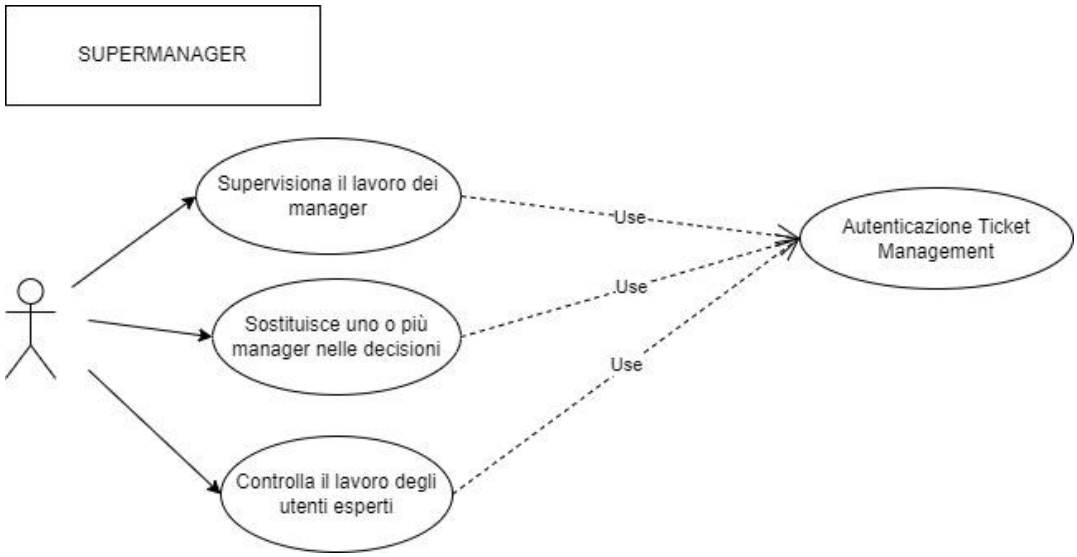



*Figura 11 - Caso d'uso supermanager*

# Requisiti funzionali e applicativi

Dopo una fase di raccolta e definizione dei requisiti e dopo una programmazione del lavoro da svolgere per la realizzazione dell'applicazione, sono state identificate le caratteristiche e le peculiarità che di seguito vengono elencate:

- Il ticket deve poter essere inserito da un qualsiasi utente nella rete aziendale (anche attraverso VPN) ed anche da utenti terzi o tecnici che per conto dei primi effettuano questa operazione.
- Il flusso di lavoro deve essere ben definito per il processo di approvazione, risoluzione e test in condizioni ordinarie e straordinarie.
- Lo stato "da verificare" deve consentire all'utente di poter testare l'effettiva risoluzione del problema in tempi ragionevoli (al massimo una settimana).
- L'applicazione deve consentire la ricerca di un particolare ticket attraverso il suo titolo, o attraverso qualsiasi altra informazione che possa costituire una chiave univoca di ricerca.
- Il software deve permettere di utilizzare filtri, ordinamenti e raggruppamenti standard.
- Il software deve permettere di utilizzare filtri e ordinamenti personalizzati definiti dall'utente.
- Le tabelle mostrate presenteranno attributi (colonne) che potranno sempre essere configurate dall'utente (rimosse/aggiunte).
- Nella pagina di dettaglio verranno mostrate le principali informazioni e sarà presente

una chat per dialogare direttamente con l'utente esperto.

# Capitolo 4 – Strumenti utilizzati

## SAP – System Analysis Program

La società SAP, che mosse i primi passi alla fine degli anni 70 con la denominazione di System Analysis Program, è un produttore di software per la gestione dei processi aziendali tra i più importanti al mondo. Sviluppa soluzioni che facilitano l'efficace elaborazione dei dati ed il flusso di informazioni tra le organizzazioni.

Con l'introduzione del software originale R/2 ed R/3, SAP ha fissato quelli che oggi vengono definiti, gli standard per i software ERP. Oggi, con SAP 4/HANA, l'ERP si compie un ulteriore salto di qualità, sfruttando la potenza del cloud computing per elaborare enormi quantità di dati e supportare delle tecnologie avanzate come l'intelligenza artificiale e il machine learning, che consentono alle imprese di trasformare in realtà il concetto di industria 4.0 [*11*].

Le applicazioni integrate di SAP, chiamate moduli, connettono tutti i settori aziendali in una suite intelligente basata su una piattaforma completamente digitale. Ogni processo può essere studiato, mappato e digitalizzato. Il software raccoglie ed elabora i dati in un'unica piattaforma, dall'acquisto di materie prime sino alla produzione ed alla customer satisfaction.

Le soluzioni SAP possono essere installate "on premise", su server di proprietà dall'azienda, oppure fruite in soluzione "cloud based" in cloud servers SAP (SaaS).

Come si vedrà nei paragrafi successivi, le tecnologie di SAP sono state fondamentali per la realizzazione del progetto Ticket Management permettendo di realizzare una app. che

possieda, al suo interno, concetti di “on premise”. Verrà, infatti, impiegato un server gestito dai sistemisti aziendali per ospitare la “soluzione”.

La possibilità di trasformare, in maniera quasi immediata ed automatica, il software in applicazione per dispositivi mobile, data anche l’elevata attitudine alla responsiveness delle interfacce e la divisione tra le componenti backend e frontend, costituisce un fondamentale vantaggio rispetto alla concorrenza e permette, in tempi brevissimi, se non addirittura trascurabili, di evolvere il prodotto ed esporlo nel mercato delle Web-app di SAP.

Ultimo aspetto da citare, ma non per grado di importanza, è l’eventuale possibilità di customizzazione della Web-application sulla base delle esigenze proprie del cliente, che ne costituisce un ulteriore punto di forza e, probabilmente, un vantaggio determinante per la sua scelta.

# Tecnologie SAP per le Web-app

Di seguito verranno mostrati tutti gli strumenti e le tecnologie impiegate per lo sviluppo della soluzione proposta. Verranno presentati i tools utilizzati per la scrittura della applicazione, le librerie, i protocolli per la comunicazione ed altro.

## *SAP GUI vs SAP Fiori*

Per parlare di cosa è SAP Fiori bisogna necessariamente prima analizzare quali sono le esigenze che hanno spinto l’Azienda SAP ad investire su questa nuova tecnologia.

Una delle particolarità ben note di SAP è che l’interfaccia, che utilizziamo per lavorare con il programma (la GUI), è davvero molto complicata da utilizzare e poco intuitiva.

L'aspetto generale appare molto datato, usabile solo da PC e, pur riuscendo soddisfare tutte quelle che sono le esigenze in termini di gestione dei flussi digitali, non trova largo utilizzo dato che, nella quasi totalità delle aziende, si rende necessario formare del personale all'interno dell'IT altamente competente e che abbia, inoltre, una grande conoscenza funzionale e procedurale dei flussi interni. L'applicazione, infatti, lavora per transazioni, alle quali vengono attribuiti dei codici univoci (esempio in figura/tabella 12) per l'accesso, senza possibilità quindi di navigare liberamente all'interno dell'interfaccia. Questo non consente a coloro che non posseggono determinate competenze tecniche, di trovare quel che cercano con facilità *[12]*.

In figura 12 sono presenti rispettivamente 3 tabelle con esempi di transazioni SAP tra le più comuni ed utilizzate, in vari ambiti, e per disparate esigenze.

| PREFISSO | MODULO DI RIFERIMENTO |
|---|---|
| F | Financial |
| M | Material Management |
| MB | Inventory management |
| MM | Material masterdata |
| ME | Purchasing |
| VA | Sales order |
| CO | Internal Order management |
| PA | Personal management |
| LO | Logistics |
| AP | Application Platform |
| QM | Quality management |
| TM | Transportation |
| KB | Knowledge Base |
| CRM | Customer relationship |
| Z- | Custom Transactions |

1

| SUFFISSO | SIGNIFICATO |
|---|---|
| 01 | Creazione |
| 02 | Modifica |
| 03 | Visualizzazione |

2

| CODICE | SIGNIFICATO |
|---|---|
| MIGO | Material movements |
| SE16N | General table display |

3

*Figura 12 – Prefissi delle più importanti transazioni (tabella 1), suffissi utilizzati per inserimento, modifica e visualizzazione (tabella 2), transazioni che non rispettano le regole (tabella 3)*

Nella tabella 1 vengono elencati i prefissi delle più importanti ed utilizzate transazioni. SAP mette a disposizione il prefisso Z, con il quale devono essere contrassegnate tutte le

transazioni non standard, create ad hoc per una specifica azienda.

Nella tabella 2 vengono elencati i 3 suffissi utilizzati rispettivamente per creazione, modifica e visualizzazione.

Nella tabella 3 sono presenti due transazioni che non rispettano le regole precedenti. La SE16N è una transazione che permette di avere accesso diretto a tutte le tabelle che compongono il database di SAP, un descrittore dell'intero sistema.

Proprio per contrastare questa scarsa amichevolezza dell'interfaccia, il mancato utilizzo che l'utente medio fa della vecchia SAP GUI, l'inconsistenza della user experience e la complessità derivante da un approccio di tipo funzionale come mostrato in figura 13 e figura 14, l'Azienda ha deciso di investire nel progetto SAP FIORI.

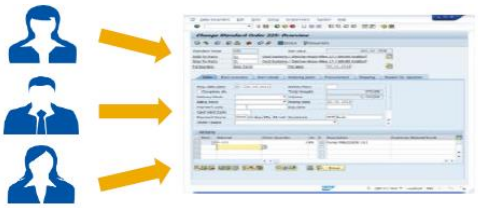
| SAP GUI | SAP FIORI |
|---|---|
| Una sola transazione per più ruoli, funzionalità complesse  | Applicazioni più atomiche e dedicate al singolo ruolo  |
| Diversi punti di ingresso per singola transazione, GUI inconsistente  | Unico punto di ingresso per singola funzione, indicazioni di progettazione condivise tra tutte le applicazioni  |

*Figura 13 – Principali differenze tra SAP GUI e SAP FIORI*

SAP FIORI è un sistema di progettazione che consente di creare applicazioni aziendali con una esperienza utente di tipo consumer trasformando, così, gli utilizzatori occasionali in veri esperti attraverso semplici schermate eseguibili su qualsiasi dispositivo. Adottando le linee guida di progettazione SAP e gli strumenti a disposizione, è possibile creare e personalizzare le Web-app secondo le proprie esigenze *[13]*.

Riassumendo brevemente le caratteristiche tecniche di SAP FIORI:

- La dashboard SAP FIORI, in esecuzione su SAP HANA cloud Platform è un software fruibile come servizio (SaaS).
- Può funzionare come una soluzione SaaS sul cloud, o in uno scenario ibrido, con una installazione on premise su server all'interno della rete aziendale (soluzione adottata nel progetto).
- Usa strategie di design e paradigma di programmazione comune a tutte le applicazioni sviluppate con la medesima tecnologia e per la medesima piattaforma.
- Ha la possibilità di adattare, aggiornare, modificare e pubblicare applicazioni in tempi brevi sul launchpad/dashboard utente.
- Consente la condivisione delle informazioni di autenticazione e autorizzazione tra le app ospitate nella medesima dashboard.
- È possibile creare applicazioni basate su utilizzo diviso per ruoli (principi di autenticazione, autorizzazione e settorializzazione dei contenuti mostrati).
- Permette di realizzare interfacce responsive e focalizzate sulle informazioni importanti (che si adattano al contenuto ed al dispositivo sul quale vengono visualizzate).
- Possiede una suite di tools SAP per un design e realizzazione veloce ed efficace.

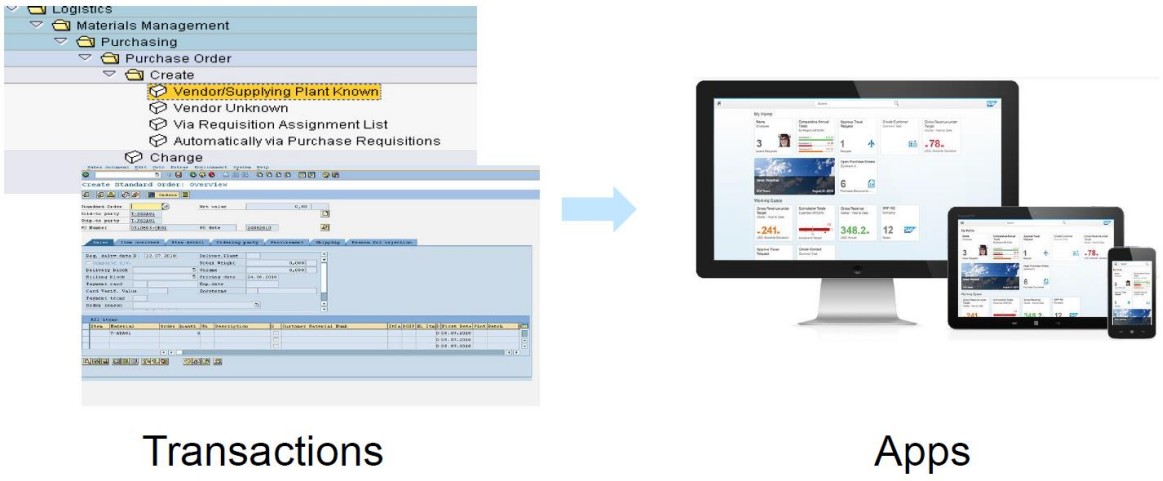



*Figura 14 – A sinistra la vecchia SAP GUI con un modello per transazioni, a destra il concetto di Web-app introdotto con SAP FIORI con chiaro riferimento alla responsiveness*

## *SapUI5/OpenUI5*

SAPUI5 è un framework che consente di creare facilmente SAP o non SAP Web-application. Come strumento lato client, utilizza CSS, HTML5 e JavaScript e consente di disaccoppiare completamente lo sviluppo del front-end dal back-end.

Sebbene sia stato progettato per la creazione di una nuova generazione di applicazioni per ambienti SAP, attualmente è diventato uno strumento per lo sviluppo di generiche applicazioni Web.

Può essere utilizzato principalmente in due modalità differenti:

1. Effettuando il download dalla SAP Hana Cloud Platform e utilizzandolo nella propria cartella di progetto nell'IDE SAP Business Application Developer.
2. Usufruendone attraverso una rete di distribuzione di contenuti (CDN) [6]

*6 - https://sapui5.hana.ondemand.com/resources/sap-ui-core.js*

L'URL, che contiene al suo interno uno script eseguibile, viene richiesto durante la configurazione e definizione del bootstrap e sarà richiamato ad ogni avvio dell'applicazione SAPUI5 (nel prossimo capitolo verrà mostrato in dettaglio).

Il framework segue il paradigma Model-View-Controller (MVC) dove l'applicazione viene divisa in tre parti; nella view sono contenute tutte le interfacce utente, nel model è contenuta la persistenza applicativa ed il mapping delle variabili, nel controller risiede la logica applicativa, quindi le funzionalità del software. Lo sviluppatore ha a disposizione diversi linguaggi per la scrittura delle viste (JSON, HTML, XML, XHTML), anche se i linguaggi da preferire sono quelli che permettono l'estensione attraverso la definizione di nuovi TAGs ed una facile leggibilità e manutenibilità, come XML e XHTML.

SAPUI5 è un framework indipendente dal dispositivo sul quale viene eseguito, questo significa che possiede dei meccanismi per la determinazione di hardware, sistema operativo e browser (tipologia e versione) usato dall'utente. Determina tutte queste informazioni attraverso una API[7] opportunamente interrogata dallo sviluppatore (con possibilità di definire dei comportamenti di default cablati nel codice e attivati da un determinato evento, per i quali si presenta un estratto in figura 15), e dispone di funzionalità che permettono all'utente di riorganizzare il contenuto visualizzato in modo dinamico in base alle proprie esigenze *[14]*.

Ricordando quanto esposto nella prima parte del paragrafo, SAPUI5 consente la creazione di applicazioni inserite nel contesto SAP, ma anche di software che non perdano la loro identità se separate dal proprio contenitore. Questo consente anche a piccole aziende, che non possono sostenere economicamente l'installazione di SAP, di beneficiare di queste valide soluzioni.

*7 - API che sta per Application Programming Interface costituisce un insieme di procedure che hanno un determinato compito e possono essere interrogate attraverso chiamate HTTP o HTTPS.*

```
// Da agganciare all'evento (es. loginMobile)

sap.ui.Device.media.attachHandler(fnSizeChanged, null,
sap.ui.Device.media.RANGESETS.SAP_STANDARD);

// eventHandler:
function fnSizeChanged(mParams) {
    switch(mParams.name) {
        case "Phone":
            // Resize del contenuto per schermi smartphone
            break;
        case "Tablet":
            // Resize del contenuto per schermi tablet
            break;
        case "Desktop":
            // Resize del contenuto per desktop devices
    }
}
```

*Figura 15 – Funzione utilizzata nel controller per il resize del contenuto dinamico e statico della pagina nei diversi dispositivi.*

Proprio la precedente considerazione ha portato all'impiego di OPENUI5, libreria open-source derivata da SAPUI5, con la quale è consentito sviluppare Web-app senza essere necessariamente vincolati all'infrastruttura SAP. Si vogliono far notare al lettore due concetti fondamentali estrapolati dalla figura 16, un grafico comparativo tra SAPUI5 e OPENUI5 *[15]*:

1. Le componenti di integrazione con Back-end e persistenza (DB), sono fornite da SAPUI5, l'utilizzo di questa soluzione obbliga inevitabilmente il progettista a costruire applicazioni back-end in ABAP[8] e utilizzare DBMS SAP (SAP Hana DB Management). Al contrario OpenUI5 consente l'uso di altri linguaggi di programmazione per lo sviluppo del back-end (Java) e l'impiego di altre tipologie di DBMS[9] (Microsoft SSMS, PostgreSQL).
2. Il supporto SAPUI5 è fornito dall'azienda e i contributi da parte della community non sono previsti.

*8 – Advanced Business Application Programming è un linguaggio di programmazione proprietario SAP.*

*9 – Database Management System, software per la creazione, manipolazione e interrogazione di database.*

Al contrario in OpenUI5, il supporto è fornito dalla community presente su StackOverflow[10] e si può contribuire al progetto attraverso il repository GitHub[11]

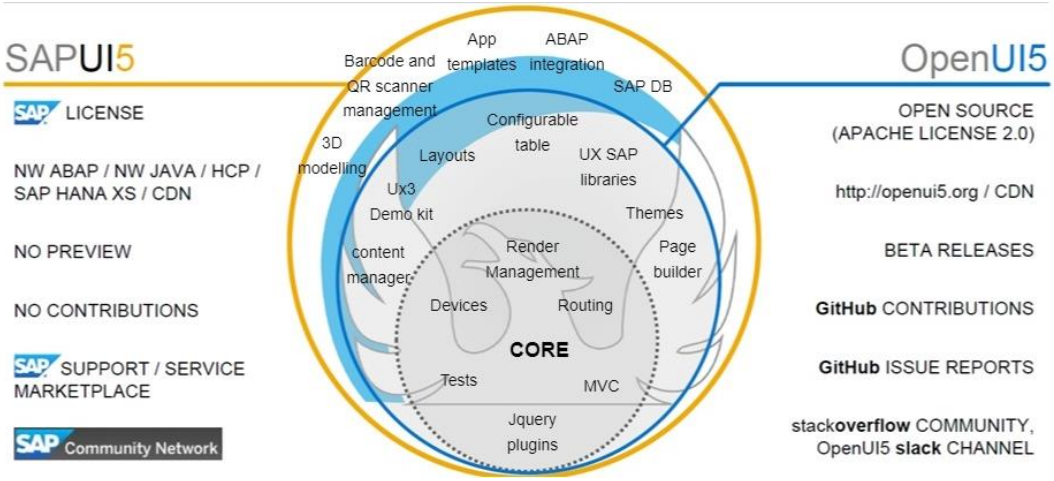



*Figura 16 – benchmarking tra SapUi5 e OpenUUI5 rispettivamente framework proprietario SAP e framework open source derivato da quest'ultimo.*

## *SAP Build*

SAP Build è uno strumento di progettazione collaborativa delle interfacce in cloud (disponibile all'indirizzo https://www.build.me/) che consente ad utenti tecnici e non, di creare prototipi interattivi (mockup) rendendo più semplice l'acquisizione di feedback accurati da parte degli stakeholders in fase di prototyping *[16]*.

Nel prossimo capitolo si esporrà nel dettaglio tutto il processo di design delle interfacce dell'applicazione Ticket Management mentre, per il momento, si vuole soltanto mostrare con un grafico in figura 17, tutto il processo realizzativo del software inserendo Build all'interno del contesto.

*10 - Stack Overflow è un sito Web che fa parte della rete Stack Exchange in cui si possono porre domande riguardo vaste tematiche nel mondo della programmazione software.*

*11 - https://github.com/SAP/openui5*

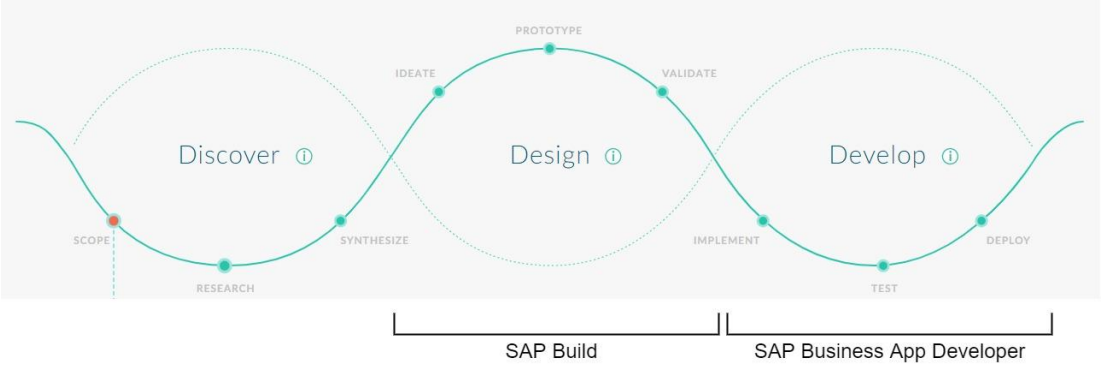



*Figura 17 -Fase di Design e prototyping con SAP Build e fase di sviluppo realizzata attraverso SAP Business Application Developer*

## *SAP Business Application Studio (ex. SAP Web IDE)*

SAP Business Application Studio (precedentemente SAP Web IDE), è un ambiente di sviluppo basato sul cloud per applicazioni SAPUI5 e OpenUI5. Tutto quello di cui lo sviluppatore ha bisogno è un Web browser di ultima generazione per accedervi. L'IDE fa parte della piattaforma SAP Hana Cloud, pertanto, per poter utilizzare la configurazione IDE + OpenUI5 è necessario un account gratuito (trial) sottoscrivibile al seguente indirizzo.[12]

L'IDE possiede al suo interno interprete e compilatore per i linguaggi e le librerie più diffuse nello sviluppo di applicazioni Web, quali ad esempio CSS, HTML, XML, JSON, JavaScript, PHP, JQuery, AJAX e molti altri. Si basa sull'IDE open source "Eclipse Theia", ed abbraccia l'esperienza grafica offerta da Microsoft Visual Studio Code *[14]*.

Per accelerare i tempi di formazione e ridurre la curva di apprendimento, ogni spazio di sviluppo è ottimizzato per uno specifico scenario di business, ciò significa che:

- L'IDE è precaricato con un set di estensioni e tool necessari per quello specifico scenario.

*12 - https://account.hanatrial.ondemand.com/*

- Lo spazio di sviluppo è dotato di strumenti ad alta produttività per lo scenario aziendale pertinente (es. procedure guidate, modelli, editor grafici)
- Nello spazio di sviluppo sono inclusi strumenti generici aggiuntivi necessari per lo sviluppo, il test immediato delle modifiche e l'esecuzione della soluzione (es. mock server SAP, mock data generator SAP utilissimi per costruire un set di dati e testare il comportamento della applicazione in un contesto ristretto e simulato).

Nello scenario di business del progetto Ticket Management, si vuole realizzare un software che l'utente potrà avviare direttamente dal proprio browser (mobile o desktop) all'interno della dashboard. Per far questo occorre fare delle considerazioni e adottare delle best practices suggerite dall'azienda SAP che possono essere riassunte nei seguenti punti:

1. Creazione ed utilizzo di un repository per la gestione e versioning del codice sorgente (GIT, GitLab, GitHub).
2. Installazione di estensioni consigliate quali Layout editor, servizi per la gestione sintattica del codice SAPUI5/OpenUI5 (completamento codice, intellisense[13]).
3. Creazione di un set di dati (MockData) per il test delle interfacce e delle procedure
4. Test del codice sorgente su ambiente runtime simulato (Mock server SAP)
5. Distribuzione della app nella dashboard FIORI/SAP hana Cloud.

*13 - È una forma di completamento automatico introdotta da Microsoft sull'IDE Visual Studio code, serve inoltre come documentazione per i nomi delle variabili, delle funzioni e dei metodi usando metadati e reflection.*

## *UI5 Inspector Tool & Icon Explorer*

**UI5 Inspector Tool:** Estensione per il browser Google Chrome, installabile all'interno del pannello per gli sviluppatori come in figura 17. Con questo strumento si possono facilmente correggere, ispezionare e modificare a runtime oggetti UI5 all'interno dell'applicazione e vederne gli effetti.

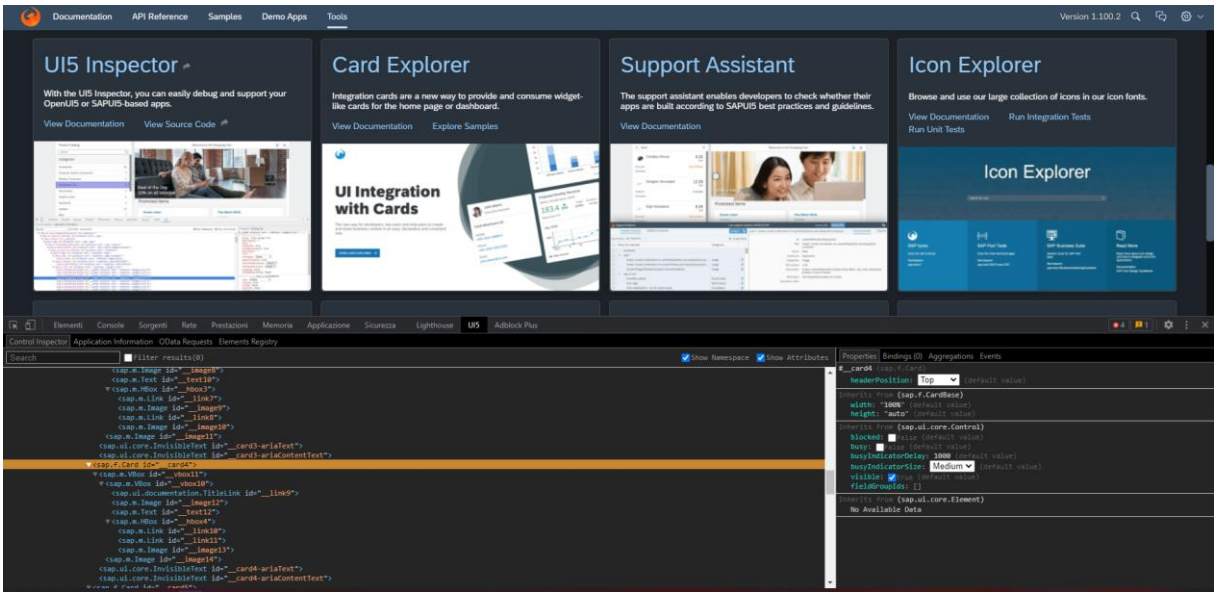



*Figura 17 – UI5 Inspector tool nella barra strumenti per gli sviluppatori disponibile su Google Chrome*

**Icon Explorer:** È un tool in cloud web che contiene una collection molto vasta di icone usabili all'interno di applicazioni SAP FIORI. Aiuta lo sviluppatore nella realizzazione di un prodotto standardizzato e di facile assimilazione per l'utente finale che ha l'impressione, seppur cambiando contesto, di lavorare sempre sulla stessa piattaforma con una user experience estremamente omogenea ed un tempo di adattamento alle nuove interfacce trascurabile.

## *Protocollo Odata*

L'Open data protocol (abbreviato Odata) consente la creazione di servizi basati su architettura REST che consentono alle risorse, identificate tramite URL e definite nel modello dei dati, di essere pubblicate e modificate dai client WEB tramite semplici messaggi HTTP/HTTPS. La semantica di base e gli aspetti comportamentali del protocollo sono i seguenti *[17]*:

- [Odata-URL] definisce un insieme di regole per la creazione di URL e identificare dati e metadati esposti da un servizio Odata nonché un insieme di opzioni per l'interrogazione (queries)
- [Odata-CSDLJSON] definisce una rappresentazione JSON del modello di dati dell'entità esposto da un servizio Odata.
- [OData-CSDLXML] definisce una rappresentazione XML del modello di dati entità esposto da un servizio OData.
- [OData-JSON] specifica il formato JSON delle rappresentazioni delle risorse scambiate tramite OData.

Tralasciando una esposizione teorica del protocollo che rischierebbe di essere prolissa e notevolmente lunga, in questa sede si vuol cercare di mostrarne il funzionamento, attraverso esempi reali e porzioni di codice che meglio riescono a suscitare curiosità ed interesse.

Nella seguente porzione di codice viene rappresentato un prodotto con un prezzo ed una descrizione e sono stati evidenziati *[15]*:

- In verde la tipologia di dato (nell'esempio Edm.String dal nome, rappresenta un tipo di dato Stringa)

- In giallo le annotation cioè le caratteristiche di ogni oggetto (es. un campo di tipo stringa nell'interfaccia può essere filtrato se sap:filterable="true")
- In rosso il nome univoco di una entità
- In blu il nome attribuito al campo chiave di quella entità (rappresenta un elemento biunivoco coincidente al campo chiave in una tabella del database)

```
<EntityType Name="Product" sap:content-version="1">
  <Key>
     <PropertyRef Name="ProductID"/>
  </Key>
  <Property Name="Name" Type="Edm.String" Nullable="false" MaxLength="80"
      sap:label="Product Name" sap:creatable="false"
      sap:filterable="false"/>
  <Property Name="Description" Type="Edm.String" Nullable="false"
      MaxLength="12"
      sap:label="Prod.Descrip." sap:creatable="false"   b
      sap:filterable="false"/>
  <Property Name="Price" Type="Edm.String" Nullable="false" MaxLength="40"
      sap:label="Price" sap:creatable="false" sap:filterable="false"/>
  <Property Name="WeightMeasure" Type="Edm.String" Nullable="false"
      MaxLength="40"
      sap:label="WeightMeasure name" sap:creatable="false"
      sap:filterable="false"/>
  <NavigationProperty Name="ToSupplier"
      Relationship="/IWBEP/GWSAMPLE_BASIC.
      Assoc_BusinessPartner_Products"
      FromRole="ToRole_Assoc_BusinessPartner_Products"
      ToRole="FromRole_Assoc_BusinessPartner_Products"/>
</EntityType>
```

Volendo strutturare a livello grafico la composizione dell'URL, come in *figura 18:*

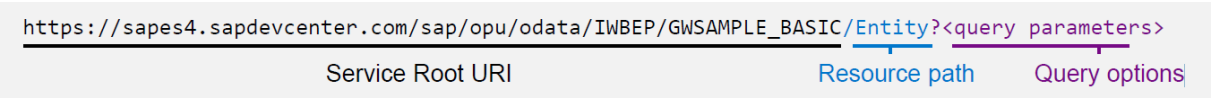



*Figura 18 – Decomposizione dell'URL*

Il **Service Root URI** rappresenta il path radice dell'applicazione, a cui viene aggiunto il relative path di navigazione.

Il **Resource path** rappresenta l'entità sulla quale si vuole agire e viene identificato attraverso la sintassi: `/<entity>(<key>=<value>,<key>=<value>).`

Se si vuol accedere ad una determinata proprietà di quella entità lo si può fare attraverso la sintassi: `/<entity>(<key>=<value>,<key>=<value>)/<property>`

**Le Query Options** rappresentano dei parametri, traducibili in comandi SQL equivalenti per l'interrogazione della persistenza applicativa o Database. Se ne mostrano alcuni esempi, con le relative corrispondenze in SQL, nella figura 19 in basso.

Di seguito un esempio di chiamata HTTPS con protocollo Odata e relativa risposta del server *[18]*:

| Operation | Odata Query Option | SQL Equivalent |
|---|---|---|
| Filtering | $filter | WHERE column1='value' |
| Projecting | $select | SELECT * or column1, column2, … |
| Sorting | $orderby | ORDER BY column2 desc/asc |
| Paging | $top | TOP(number_records) |
| Inlining | $expand | Table1 JOIN Table2 |
| Count | $count | COUNT(column_name) |

*Figura 19 – Corrispondenze Odata Queries e SQL queries*

Un esempio di applicazione del protocollo è l'estrazione di tutti gli impiegati che siano nati nel 1948.

L'URL della richiesta sarà:

https://services.odata.org/Northwind/Northwind.svc/Employees?$filter=year(BirthDate) eq 1948

Il server risponderà con il seguente costrutto XML:

```
<content type="application/xml">
 <m:properties>
       <d:EmployeeID m:type="Edm.Int32">1</d:EmployeeID>
       <d:LastName>Davolio</d:LastName>
       <d:FirstName>Nancy</d:FirstName>
```

```
        <d:Title>Sales Representative</d:Title>
        <d:TitleOfCourtesy>Ms.</d:TitleOfCourtesy>
        <d:BirthDate m:type="Edm.DateTime">1948-12-
08T00:00:00</d:BirthDate>
        <d:HireDate m:type="Edm.DateTime">1992-05-01T00:00:00</d:HireDate>
        <d:Address>507 - 20th Ave. E.&#xD;Apt. 2A</d:Address>
        <d:City>Seattle</d:City>
        <d:Region>WA</d:Region>
        <d:PostalCode>98122</d:PostalCode>
        <d:Country>USA</d:Country>
        <d:HomePhone>(206) 555-9857</d:HomePhone>
        <d:Extension>5467</d:Extension>
        <d:Notes>Education includes a BA in psychology from Colorado State
              University in 1970.  She also completed "The Art of the Cold
              Call."  Nancy is a member of Toastmasters
              International.</d:Notes>
        <d:ReportsTo m:type="Edm.Int32">2</d:ReportsTo>
        <d:PhotoPath>http://accweb/emmployees/davolio.bmp</d:PhotoPath>
 </m:properties>
</content>
```

# Altre tecnologie utilizzate

## *GitHub*

GitHub è un servizio di hosting per progetti software. Il nome contiene in sé volutamente il prefisso git che indica il più famoso sistema di controllo e versioning del codice sorgente open source attualmente disponibile nel panorama internazionale *[19]*. Lo Strumento git, che viene tecnicamente definito un DVCS che sta per “Distributed Version Control System”, rappresenta una metodologia di controllo di versione che permette di tenere traccia delle modifiche e delle versioni apportate al codice sorgente in modo distribuito, cioè senza la necessità di utilizzare un cluster centrale.

In un DVCS, i client non solo controllano lo snapshot più recente dei file, ma piuttosto fanno una copia identica dell’archivio (repository) completa di tutta la propria storia. In questo modo se un server smettesse di funzionare e se i sistemi interagissero tramite

questo server, il repository di un qualsiasi client potrebbe essere copiato sul server per ripristinarlo. Inoltre, molti di questi sistemi consentono di avere più repository remoti su cui lavorare (ridondanza). Lo sviluppatore può così collaborare con gruppi differenti di persone che apportano modifiche contemporaneamente allo stesso progetto, avendo con una certa approssimazione, idea di quali file gli altri sviluppatori stanno modificando. Questo permette di impostare diversi tipi di flussi di lavoro che non sono possibili in sistemi centralizzati, come i modelli gerarchici *[20]*.

Di seguito vengono elencate (e mostrate graficamente in figura 20) alcune delle operazioni più importanti su git:

- `git init` comando per la creazione di un nuovo progetto
- `git clone [url]` comando per il download dell'intero progetto in locale
- `git commit -m "messaggio per la commit"` comando per proporre la modifica/cancellazione/creazione di uno o più file nel repository. L'utilizzo del verbo proporre (commit) non è una scelta a caso: ogni commit, deve essere revisionata ed approvata da almeno uno degli altri contributors di progetto.
- `git push origin [master/branch]` una volta ricevuta approvazione, lo sviluppatore, attraverso questo comando, fa confluire il codice nel master o in alternativa in un branch specifico (figura 19). Questa attività può essere schedulata attraverso un job automatico, subordinato all'approvazione della commit.
- `git pull` comando utile per aggiornare la repository locale alla commit più recente disponibile sul server.
- `git merge [branch]` comando per incorporare il codice di un branch all'interno del master.
- `git checkout [branch]` comando per muoversi su uno specifico branch/master.
- `git diff <sorgente> <target>` commando che mostra le differenze tra due commit o due branch o tra master e un branch.

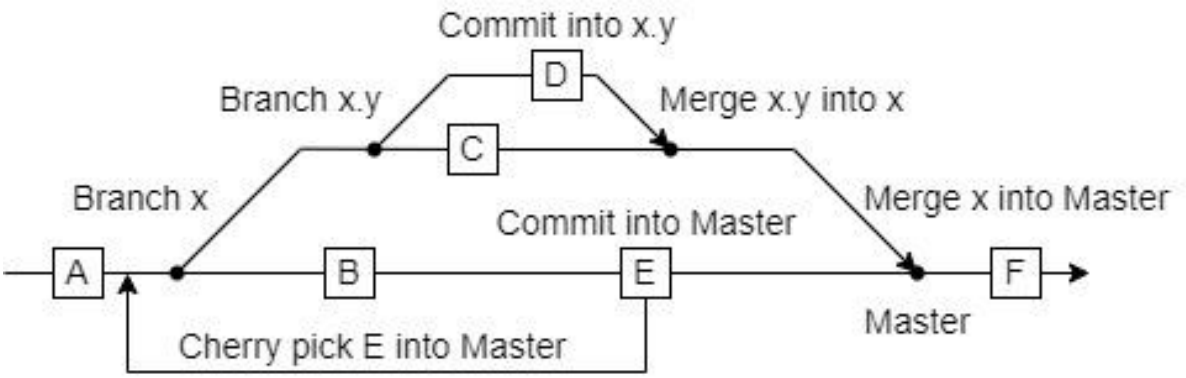



*Figura 20 – Rappresentazione grafica di alcune delle operazioni git*

In figura 20, la linea principale (master) è quella di default quando si crea un repository. Si possono utilizzare dei branch, per isolare il codice committato dal master e creare quindi delle altre versioni dell'applicativo. Il merge è una operazione atta a far confluire nuovamente il codice nel master, o in branch di livello più alto, il cherry pick serve a riportare una singola commit in un punto diverso del master.

Nel progetto Ticket Management, il contribitor è unicamente il candidato ma si è deciso, nonostante questo, di utilizzare uno strumento di versioning, come GitHub, per permettere una più agevole gestione del codice sorgente e per dar modo ad eventuali futuri membri del team di lavorare su un codice già disponibile.

## *PostgreSQL & PgAdmin*

**PostgreSQL** è un DBMS ad oggetti openSource, rappresenta una reale alternativa a prodotti analoghi con codice chiuso quali Microsoft SQL Azure, Oracle DB e IBM Informix ed offre caratteristiche uniche nel suo genere che lo pongono per alcuni aspetti all'avanguardia nel settore delle basi di dati *[21]*.

**PgAdmin** è una applicazione libera con una interfaccia grafica che consente di

amministrare in modo semplice un database PostgreSQL. Permette di creare un database, creare tabelle ed eseguire operazioni di ottimizzazione su di esse *[21]*.

## *Microfocus UFT One & Run Result Viewer*

Microfocus UFT One precedentemente noto come HP UFT, è un software che consente di automatizzare test funzionali di non regressione per applicazioni ed ambienti software in ambiti mobile, desktop e Web. Supporta metodi di scripting e dispone di una interfaccia utente di veloce comprensione. Si avvale del linguaggio Visual Basic, scripting edition (VBScript), per implementare una procedura di test e manipolare gli oggetti, i controlli e le funzionalità del software sottoposto a test [*22*].

UFT consente agli sviluppatori di testare tutti e tre i livelli applicativi in un'unica console:

- GUI ed interfaccia utente
- Logica applicativa
- Persistenza applicativa o Database

Nella Web application Ticket Management, sono stati sviluppati degli script in grado di replicare le azioni utente per specifici casi d'uso reali, come quelli già analizzati nel capitolo 3.

Le motivazioni che hanno spinto all'uso di un simile strumento di test sono molteplici; si vogliono elencare solo le più evidenti e rilevanti:

1. Creazione di pacchetti di script con obiettivi puntuali (es. collection di scripts minimali in grado di garantire il funzionamento del core dell'applicazione)
2. Incremento rapido e costante della test coverage grazie alla possibilità di creare librerie di oggetti grafici riusabili e con sviluppo di nuovi scripts a mano a mano che

il software si aggiornerà con nuove features.

3. Rilascio di versioni sempre testate ed esenti da eventuali bug e regressioni di codice.
4. Automatizzazione e schedulazione di script runs, riducendo l'intervento umano al solo check dei reports di test automaticamente generati dal sistema a seguito di ciascun script run.
5. Incremento frequenza di test all'interno delle release. Utilizzando l'UFT test scheduler è possibile lanciare una collection di test con una frequenza di una o più volte al giorno, compatibilmente con la disponibilità da parte di un tester di controllare i reports dei test falliti per evidenziarne le cause.

# Capitolo 5 – Realizzazione software

Il seguente capitolo sarà il cuore di tutto l'elaborato ed avrà come obiettivo quello di descrivere come l'applicazione è stata pensata, progettata e realizzata, seguendo tutti i passaggi esposti dal "Software lifecycle management", ed una metodologia di sviluppo del codice di tipo "Agile".

Del modello di processo di tipo "Agile" si parlerà ampiamente nei prossimi paragrafi, mentre sin da ora si vuol rappresentare il concetto di processo di sviluppo di un software.

Il Software Lifecycle management è costituito dai seguenti steps *[8]*:

1. *Pianificazione:* Quali attività devono essere svolte per portare il progetto a compimento finale.
2. *Raccolta ed analisi dei requisiti:* Vengono sostenuti dei colloqui con gli stakeholder di progetto ed in generale raccolte informazioni attingendo a diverse fonti, questo perché il progettista e gli sviluppatori possano comprendere appieno quali sono le esigenze degli utenti finali. Fattorizzando quanto raccolto, si cercherà di esporre i concetti estrapolati in maniera piuttosto utilizzando documenti che verranno mostrati e fatti firmare dai committenti dell'applicazione.
3. *Progettazione e definizione del modello dei dati:* Si prendono delle decisioni fondamentali per il progetto come l'architettura hardware/rete da utilizzare, il modello dei dati, la struttura degli stessi per la persistenza applicativa, il paradigma di programmazione, il modello di processo adottato (che in questo caso sarà agile), gli strumenti software impiegati (IDE, framework, DBMS, ecc…).
4. *Realizzazione applicazione:* Viene implementata la soluzione finale, sia nella sua componente frontend, con la creazione prima di prototipi di navigazione (mokup

interattuvi) e in seguito delle interfacce utente definitive chiamate GUI, sia nella sua componente backend, con la logica applicativa, le funzionalità caratterizzanti ed il mapping delle informazioni estratte dallo strato di persistenza (database).

5. *Validazione e testing:* Si valida il software nella sua versione definitiva, lo si fornisce ad i beta testers per l'utilizzo in situazioni che riproducano casi reali ed allo stesso tempo si creano degli ecosistemi per il testing funzionale automatico e programmato. Al termine di queste attività che procederanno per un tempo imprecisato (fino a quando saranno risolti tutti gli eventuali errori), si procederà con la validazione del prodotto ed il conseguente rilascio ed installazione in ambienti di produzione.

Nell'ambito della progettazione software e più in generale nell'ingegneria informatica, il project manager è la figura cardine che, alla genesi dell'intero progetto, è chiamato a prendere una decisione fondamentale per lo stesso circa il modello di processo da adottare.

Il modello di processo/sviluppo software è il principio teorico che indica quale metodo seguire nel progettare e sviluppare un programma. Il modello indica una determinata linea di sviluppo.

Nell'ambito della pianificazione del lavoro da svolgere, i modelli di processo si dividono principalmente in due metodologie *[23]*:

1. Processi guidati dal piano *(Plan Driven):* Tutte le attività del processo sono pianificate in anticipo e lo stato di avanzamento è misurato in funzione di quanto previsto dal piano.
2. Processi Agili *(Agile method):* Lo sviluppo del piano è incrementale e può essere facilmente modificato in accordo ad eventuali cambiamenti dei requisiti da parte del committente del software.

# Pianificazione

Seguendo fedelmente gli steps del software lifecycle management, di seguito verranno illustrate tutte le attività necessarie per portare a compimento il lavoro. Un progetto infatti è definito come una sequenza complessa di attività interconnesse tra loro che presenta determinate caratteristiche quali:

- Possiede un principale obiettivo *(hard goal),* ed altri di minore importanza ma comunque di rilievo *(soft goals)*.
- Deve essere completato entro e non oltre una scadenza specifica e predeterminata, con alcuni vincoli quali il budget e le risorse umane da impiegare.

Con una efficace attività di pianificazione, stabilendo le risorse da impiegare *(umane e materiali)*, si possono facilmente fare delle previsioni sui costi e tempi necessari alla produzione del software *[21]*.

Nell'ambito del progetto ticket management, l'azienda che ha ospitato il tirocinio e che si è fatta promotrice del progetto anche nei confronti dell'ateneo, non ha dovuto sostenere costi per le risorse umane, in quanto il candidato stava svolgendo attività di tirocinio formativo esterno previsto dal proprio piano di studi accademico; pertanto, gli unici oneri sono quelli relativi ai beni materiali ed immateriali forniti al candidato per lo svolgimento del proprio lavoro, nonché rimborsi spese riconosciuti al candidato stesso.

## *Metodologia Agile*

Come si è già specificato più volte nell'elaborato, la metodologia prescelta dal project manager di "Ticket Management" è stata quella Agile.

Le caratteristiche più importanti di questo modello di processo sono le seguenti:

- Modello di processo orientato alle persone: gli individui e le interazioni hanno, quindi, la precedenza sui processi e gli strumenti.
- Fatto per rispondere efficientemente ed efficacemente ai cambiamenti, talvolta senza pianificazione e definizione di un piano di lavoro.
- Lo sviluppo software ha sempre la precedenza su di una documentazione esaustiva ed ogni minima modifica, cambiamento, correzione, deve essere subito consegnata al cliente in ambiente di test per un tempestivo feedback.
- La collaborazione a stretto contatto con il cliente ha la precedenza sul seguire il piano di lavoro, i key users sono sempre in contatto con gli sviluppatori.

## *Work Breakdown Structure*

L'Individuazione dei task di lavoro va sotto il nome di Work Breakdown Structure, che fornisce una divisione gerarchica del lavoro tramite una serie di macro-attività e sotto attività. Il progetto viene infatti suddiviso in tasks a diversi livelli di astrazione ottenendo, così, una struttura gerarchica ad albero rappresentata in figura 21.

Si fa notare come l'utilizzo di alcuni termini quali "struttura gerarchica" ed "albero", caratterizzino e determinino direttamente, senza l'ausilio di ulteriori chiavi di lettura, la

metodologia con la quale questo grafico deve essere letto ed interpretato, e l'ordine con il quale i tasks al suo interno devono essere svolti.

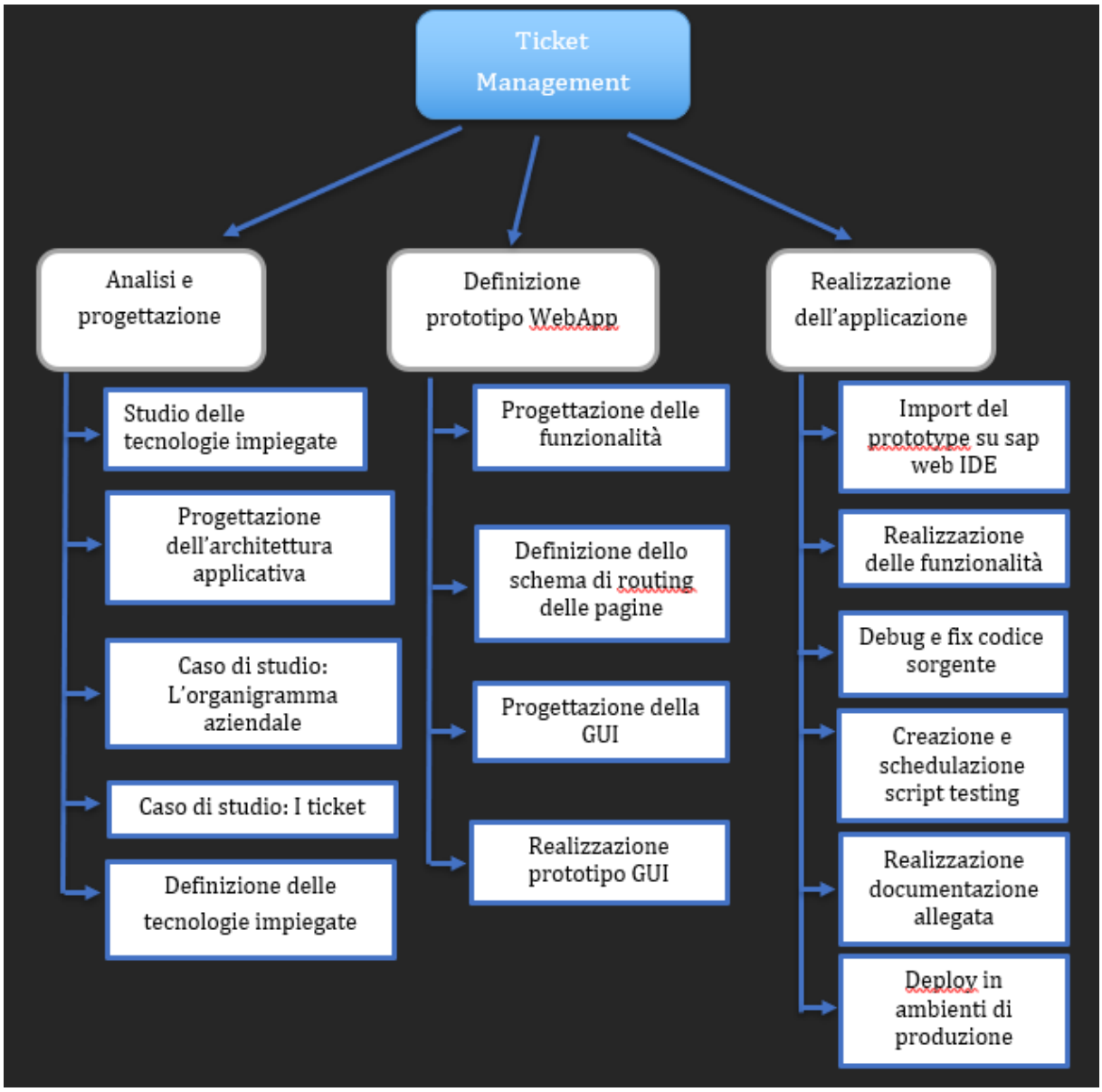



*Figura 21 – Work Breakdown Structure del Progetto Ticket Management.*

## *Matrice delle responsabilità*

La Matrice delle responsabilità è uno strumento nel quale si indicano le risorse umane associate alle attività/tasks che compongono l'intero progetto. Nel caso in esame, il software si compone di tre parti: Frontend, Backend e persistenza applicativa.

Il candidato ha progettato e sviluppato per intero la parte Frontend dell'applicazione e partecipato allo sviluppo del Backend. La progettazione della struttura della persistenza, la creazione di queries per l'accesso ai dati e le stored functions sono state onere del Database Team interno all'azienda.

| TASK | Frontend | Backend | Persistenza |
|---|---|---|---|
| Candidato | X | X | |
| Backend Team | | X | |
| Database Team | | | X |

*Tabella 22 – Matrice delle responsabilità del progetto Ticket Management*

# Raccolta ed analisi dei requisiti

## *Raccolta dei requisiti*

Quando si adotta un modello di processo Agile, la redazione di un documento di specifica dei requisiti viene considerata un'inutile perdita di tempo; I requisiti mutano velocemente e di conseguenza il documento risulta essere sempre non aggiornato. Si adotta quindi un'ingegneria dei requisiti di tipo evolutivo, le caratteristiche sono quindi espresse mediante User Stories o Technical Features *[21]*:

- Le User Stories rappresentano delle specifiche peculiarità funzionali o procedurali desiderate dagli stakeholders o dai committenti del software (es. la possibilità di avere un filtro completamente configurabile che permetta all'utente di cercare i ticket con velocità).
- Le Technical Features rappresentano delle caratteristiche tecniche di specifica scelta dei progettisti (es. possibilità di accedere alla app. connettendosi alla rete interna aziendale o dall'esterno, mediante utilizzo di una VPN, impiego di un protocollo di comunicazione HTTP e non HTTPS).

Quello che si fa solitamente nel modello di processo prescelto, è definire uno o più casi di studio. Nella fattispecie "Il Ticket" e "L'organigramma aziendale", dei quali si è già ampiamente parlato nel capitolo 3, come preludio alla successiva definizione dei casi d'uso per ognuna delle figure fruitrici della Web-app.

Pertanto, preso atto dei casi di studio descritti nel capitolo 3 e dei casi d'uso, nel seguente paragrafo andremo ad esporre quelli che sono i requisiti utente, divisi per ogni specifica categoria, i requisiti funzionali (user stories), ed infine i requisiti tecnici (Technical Features) dopo un'opportuna fase di analisi dei requisiti.

## *Requisiti Utente*

Ogni tipologia di utente, deve svolgere delle particolari operazioni, ha delle particolari esigenze e deve raggiungere determinati obiettivi.

- **Utente Finale:**
  - Obiettivo 1: Apertura Ticket
  - Obiettivo 2: Controllare lo stato di avanzamento di uno o più ticket mediante l'uso di filtri di ricerca, barra di ricerca libera, ordinamenti, raggruppamenti e chat con l'utente esperto/tecnico
  - Obiettivo 3: Effettuare il test per confermare l'avvenuta risoluzione
  - Ambiente 1: Postazione di lavoro interna alla rete aziendale
  - Ambiente 2: Laptop/Smartphone personale mediante VPN
- **Utente Esperto/Tecnico:**
  - Obiettivo 1: Apertura ticket per conto di un utente
  - Obiettivo 2: Accettazione/rifiuto assegnazione ticket *(il rifiuto può avvenire solo in mancanza di competenze tecniche per la risoluzione, in questo caso il ticket torna in stato Approvato)*
  - Obiettivo 3: Lavorare alla risoluzione del problema *(questo comprende anche la comunicazione costante con l'utente per la condivisione ed il reperimento di ulteriori informazioni)*
  - Obiettivo 4: Ultimare il lavoro sul ticket e cambiare lo stato in *"da verificare"*

- Ambiente 1: Postazione di lavoro interna alla rete aziendale
- Ambiente 2: Laptop/Smartphone personale mediante VPN

- **Manager:**
  - Obiettivo 1: Approvazione manuale dei ticket *(si parla di approvazione manuale, perché per specifiche situazioni il sistema sarà in grado di operare una approvazione automatica)*
  - Obiettivo 2: Assegnazione del ticket all'utente esperto *(nel caso di rigetto, il ticket verrà riassegnato ad un diverso utente esperto)*
  - Obiettivo 3: Definizione regole per l'approvazione automatica del sistema
  - Obiettivo 4: Controllare lo stato di avanzamento dei ticket in lavorazione
  - Ambiente 1: Postazione di lavoro interna alla rete aziendale
  - Ambiente 2: Laptop/Smartphone personale mediante VPN
- **Supermanager:**
  - Obiettivo 1: Controlla il lavoro degli utenti esperti e dei manager
  - Obiettivo 2: Sostituisce il manager nelle decisioni più critiche ed urgenti
  - Ambiente 1: Postazione di lavoro interna alla rete aziendale
  - Ambiente 2: Laptop/Smartphone personale mediante VPN

# *Analisi dei requisiti*

L'analisi dei requisiti viene fatta attraverso il release planning game, una attività nella quale vengono estrapolate per ogni singola iterazione, chiamata sprint, tutte le funzionalità che si implementeranno e rilasceranno agli utenti. Il "Game", quindi, rappresenta una sorta di contrattazione che gli sviluppatori, ingegneri e testers sostengono con il Business team (key users, clienti, manager e stakeholders in genere), che ha come obiettivo quello di determinare il contenuto della successiva versione.

Alla base di questo "gioco" troviamo le User Stories. Ognuna di esse ha un valore ed un costo (sviluppatori coinvolti, tempistiche, tecnologie da utilizzare, licenze), difficile talvolta da determinare con esattezza dato che, per intrinseca natura di questi task e della progettazione e programmazione Agile, i valori possono mutare nel tempo, le analisi talvolta diventano rapidamente inadeguate ed alcune User stories sono dipendenti da altre *[24]*.

In ticket management, l'applicazione è completamente nuova e viene proposta come nuova installazione, con una serie di funzionalità e peculiarità già presenti e determinate da anni di esperienza e lavoro del project manager come consulente esterno nel reparto IT della azienda che dovrà acquistare la soluzione.

Pertanto, nel successivo paragrafo verranno elencati una serie di requisiti funzionali e tecnici utilizzando un linguaggio naturale e facilmente comprensibile e solo successivamente, attraverso la già precitata attività di "Release Planning Game", si esporranno delle Task Cards e Test Cards contenenti rispettivamente User Stories e loro relativi Test Cases[14].

*14 - Un Test Case, come facilmente intuibile, rappresenta un caso di test standard e probante per una determinata funzionalità del software che si vuol implementare nella successiva release.*

## *Requisiti Funzionali e Tecnici*

- L'applicazione deve essere utilizzabile in due contesti ambientali differenti, attraverso la rete interna aziendale, mediante la rete globale utilizzando una VPN per una connessione sicura.
- Flusso di lavoro ben definito per il processo di approvazione, risoluzione e test in condizioni ordinarie e straordinarie.
- Lo stato "da verificare" deve rimanere invariato per un periodo di tempo utile a testare l'effettiva risoluzione.
- È onere dell'utente rigettare la soluzione implementata dal tecnico qualora i risultati non fossero soddisfacenti.
- Il sistema automaticamente chiuderà tutti quei ticket in stato "da verificare" più vecchi di una settimana.
- L'applicazione deve permettere la ricerca di un particolare ticket attraverso la barra di ricerca libera.
- Il software deve permettere di utilizzare filtri, ordinamenti e raggruppamenti standard.
- Il software deve permettere di utilizzare filtri e ordinamenti custom definiti dall'utente mediante espressioni logiche.
- Le tabelle mostrate, presenteranno attributi (colonne) che potranno sempre essere configurate dall'utente (rimosse/aggiunte).
- Nella pagina di dettaglio verranno sempre mostrate le principali informazioni e sarà presente una chat per dialogare direttamente con l'utente esperto.

- Nella dashboard sarà presente una reportistica veloce e di facile lettura.
- Seguendo le linee guida di SAP Fiori, le interfacce saranno divise per ruoli, per permessi di accesso e cambieranno da utente ad utente (GUI contestualizzata e personalizzata).
- L'utente che intende utilizzare la app "Ticket Management" deve prima effettuare il login all'interno del sistema SAP Hana Cloud e successivamente accedere alla app cliccando sull'icona presente nella dashboard.
- Access rights e access roles vengono ereditati da SAP Hana Cloud
- La piena compatibilità del software è garantita unicamente con i più recenti browser (Google Chrome, Mozilla Firefox, Microsoft Edge)
- Possibilità di inserimento screenshots/immagini o brevi video per descrivere meglio il problema.
- Possibilità di modifica del template con uno ad alto contrasto per favorire l'utilizzo nei soggetti a ridotta capacità visiva e/o affetti da daltonismo (Legge Stanca, legge n.4 del 9 gennaio 2004).

## *Release Planning Game*

Non volendo oltremodo appesantire la lettura presentando ogni singola User Story, nel seguente paragrafo saranno rappresentate mediante Story Cards e Test Cards unicamente le funzionalità principali e caratterizzanti della app Ticket Management, con l'obiettivo di mostrare al lettore la strategia di comunicazione e contrattazione adottata con i committenti, ed in modo da lasciare ampio spazio all'esposizione delle fasi di progettazione

e realizzazione del software in esame.

Di seguito, nelle figure 22-25, vengono descritte le principali user stories:

**TASK CARD**

**Release:** 1.0 **Story ID:** 1 **Task ID:** 1.1 **Developer:** Davide Massa

**Story TAG:** Login SAP Hana Cloud e accesso alla app. Ticket Management

**Description:** Creazione bottone di accesso app, pubblicazione della app all'interno della SAP Hana Cloud platform, gestione delle versioni e degli upgrade in produzione della app.

**TEST CARD**

**Input:** L'utente simula l'accesso alla APP Ticket Management

**Test:** Test login in SAP Hana Cloud - Test accesso Ticket Management dalla Hana Cloud dashboard - Test ricerca Ticket Management con barra di ricerca libera - Test condivisione informazioni di login Hana Cloud/Ticket Management

**Output:** Autorizzazioni informazioni di login condivise correttamente - Ricerca di Ticket Management possibile mediante barra di ricerca - Accesso a Ticket Management senza errori

*Figura 22 – Login SAP Hana Cloud e accesso a Ticket Management*

**TASK CARD**

**Release:** 1.0 **Story ID:** 2 **Task ID:** 2.1 **Developer:** Davide Massa

**Story TAG:** Inserimento nuovo ticket e upload Media

**Description:** Creazione delle interfacce di inserimento ticket e procedure di salvataggio media e salvataggio ticket, sviluppo della logica di backend

**TEST CARD**

**Input:** Il tester simula l'inserimento di un nuovo ticket, l'upload di una immagine o di un video, simula errori come non riempire campi obbligatori o inserire informazioni sbagliate

**Test:** Test workflow standard inserimento ticket - Test omissione informazioni obbligatorie nel form - Test inserimento informazioni errate nel form - Test upload media non riconosciuto nella procedura guidata.

**Output:** Il sistema accetta e crea nuovo ticket - il sistema blocca l'utente che non ha inserito le informazioni obbligatorie - la app mostra un errore se vengono inserite info errate - la procedura guidata mostra un errore per file media non riconosciuto

*Figura 23 - Inserimento nuovo Ticket - Procedura di upload Media*

**TASK CARD**

**Release:** 1.0 **Story ID:** 3 **Task ID:** 3.1 **Developer:** Davide Massa

**Story TAG:** Dettaglio ticket e comunicazione con l'utente esperto

**Description:** Creazione delle interfacce di Dettaglio Ticket e Chat, sviluppo della logica backend

**TEST CARD**

**Input:** Simulazione consultazione della pagina dettaglio Ticket - simulazione invio di messaggi nella chat dedicata

**Test:** Test sulla estrazione delle informazioni ticket - test sul corretto invio/ricezione dei messaggi nella chat

**Output:** Il sistema mostra le informazioni corrette all'utente - il software mostra lo storico della chat con l'utente esperto

*Figura 24 - Visualizzazione dettaglio ticket e interazione con utente esperto mediante chat dedicata*

**TASK CARD**

**Release:** 1.0 **Story ID:** 3 **Task ID:** 4.1 **Developer:** Davide Massa

**Story TAG:** Ricerca ticket, filtri standard, filtri custom, ordinamenti, raggruppamenti e organizzazione vista tabella ticket

**Description:** Creazione della interfaccia principale di Lista Ticket e di tutte le interfacce collegate (one page app), sviluppo della logica di backend per la gestione di filtri, ordinamenti e raggruppamenti.

**TEST CARD**

**Input:** Creazione e salvataggio di filtri custom - applicazione filtri standard - applicazione raggruppamenti standard - riorganizzazione contenuto tabella Lista ticket

**Test:** Test funzionamento definizione e salvataggio di filtri personalizzati - test sul funzionamento di filtri standard, raggruppamenti e ordinamenti - test sulla riorganizzazione contenuto tabella Lista Ticket

**Output:** Il sistema salva correttamente il filtro personalizzato definito - il sistema mostra con espressioni matematico logiche corrette ed in tempo reale il filtro definito dall'utente - i raggruppamenti filtri e ordinamenti standard vengono applicati correttamente sulla tabella lista ticket - la riorganizzazione del contenuto della tabella funziona senza errori

*Figura 13 - Utilizzo della barra di ricerca libera, filtri standard, filtri personalizzati, ordinamenti, raggruppamenti e visualizzazione customizzata*

# Progettazione

Per poter esporre correttamente la fase di progettazione dell'applicazione Ticket Management, bisogna prima definire alcuni aspetti chiave che meglio aiuteranno il lettore nella comprensione dei contenuti che seguiranno.

Si definisce Unified Process l'attività guidata dalla definizione dei requisiti tecnici e funzionali specificati dal committente dell'applicazione, ed espressi attraverso i casi d'uso, già ampiamente esposti nei paragrafi precedenti. L'analisi dei casi d'uso permette di definire le caratteristiche dell'architettura software che li realizza in modo integrato.

Si dice quindi che lo Unified Process è centrato sull'architettura ed è composto di diverse fasi [25]:

- Inception o inizio nel quale vengono definiti tutti i casi d'uso.
- Elaboration o elaborazione nella quale è definita l'architettura di base.
- Construction o costruzione nella quale sono realizzate le diverse funzionalità.
- Transition o transizione mirata al testing/collaudo della applicazione ed alla sua installazione in ambienti di produzione.

La definizione dei casi d'uso, l'estrazione dei concetti di specifiche funzionali (user stories) e specifiche tecniche (technical features), la redazione di task e test cards, rappresentano attività da svolgere nella fase di Inception o inizio.

La definizione dell'architettura di base (software e hardware), la scelta del paradigma di programmazione, le soluzioni tecniche da adottare e le tecnologie da utilizzare, rappresentano attività da svolgere in ambito di Elaboration o progettazione e saranno presentate nei successivi paragrafi.

## *Architettura hardware*

La progettazione dell'architettura applicativa rappresenta il secondo step nell'iter rappresentato dallo Unified Process, è utilizzata per mostrare ai committenti del progetto da quali componenti è formato il software e come sono fisicamente interconnessi tra loro. In UML si utilizza il diagramma di dislocamento per la rappresentazione di queste informazioni.

I componenti del software Ticket Management sono connessi mediante una architettura di tipo three tier[15] e sono mostrati in dettaglio nel diagramma di dislocamento della figura 24:

- **Strato di Presentazione o GUI**: Il browser si occupa dell'interpretazione dei linguaggi XHTML, CSS e JavaScript e della presentazione delle interfacce all'utente. Svolge, inoltre, alcune altre importanti attività che consentono al Frontend e Backend application servers di sgravarsi di alcuni compiti.
- **Logica applicativa**: Nell'application server risiede tutta la logica applicativa di Ticket Management, il cosiddetto controller dell'applicazione.
- **Backend e persistenza DB**: Nel backend o DB Server risiede la logica applicativa di basso livello, ovvero tutte quelle primitive utili per l'accesso e la manipolazione dei dati nel database relazionale. Vengono, inoltre, esposti i servizi (API), che l'applicazione interroga per leggere o manipolare dati sul DB.

*15 - L'architettura three tier o a tre strati indica una particolare architettura software a più strati per l'esecuzione di una applicazione Web che prevede la suddivisione dell'applicazione in tre diversi moduli dedicati rispettivamente alla interfaccia utente, alla logica funzionale e alla persistenza e logica di basso livello.*

Le connessioni tra i vari strati applicativi sono rappresentate dai protocolli di comunicazione utilizzati per lo scambio di informazioni. Tra la Graphic user interface ed il frontend application server si utilizzano generiche chiamate HTTP o HTTPS con sintassi Odata, mentre per la comunicazione tra application server e persistenza, si utilizzano solamente chiamate HTTP innescate sempre mediante sintassi definita dal protocollo Odata, di cui abbiamo già ampiamente mostrato le peculiarità nel capitolo 4.

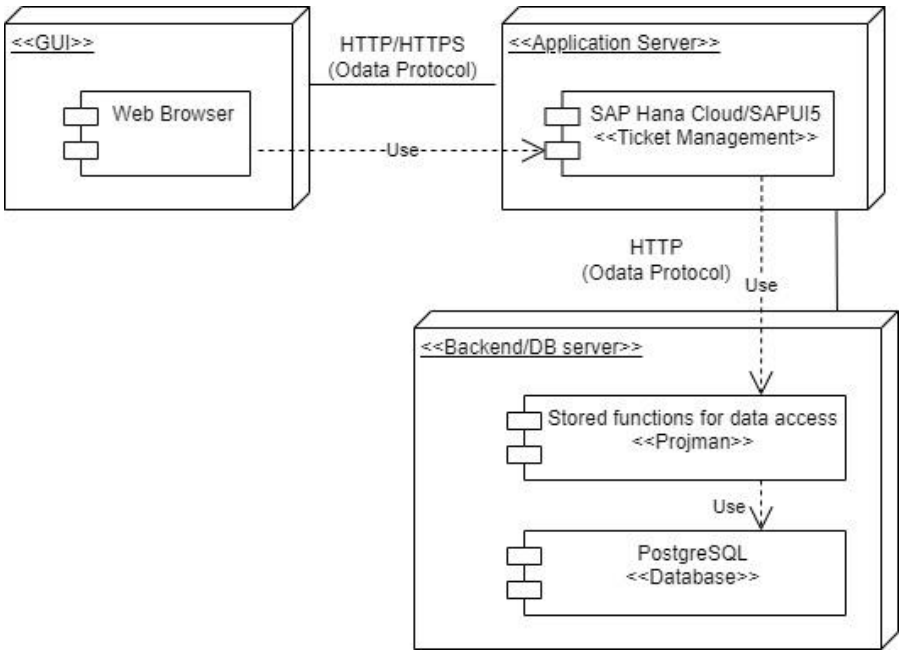



*Figura 24 – Diagramma di dislocamento per la rappresentazione dell'architettura di Ticket Management.*

La prima impressione è che la quasi totalità della logica applicativa di alto livello venga demandata all'application server. Questo accade proprio perché le nuove tecnologie, principalmente l'utilizzo della piattaforma SAP Hana Cloud installata su datacenter di proprietà dell'azienda committente, permettono di avere una capacità elaborativa considerevole, e di utilizzare il backend come gestore di accessi e manipolatore di dati, mediante l'utilizzo di primitive di basso livello che agiscono direttamente sul database.

## *Architettura software*

La crescente capacità elaborativa e di calcolo dei dispositivi mobile e desktop degli utenti, nonché quella offerta dalla piattaforma di esecuzione SAP Hana Cloud sulla SAP Fiori Launchpad, permette di trasferire parte della logica applicativa dal backend al frontend, rendendo così di fatto più veloci alcune funzionalità inserite in un set di base per l'utente. Ovviamente, per tutte le operazioni che necessitano di un accesso al database in lettura e scrittura, c'è bisogno di comunicare con il backend server.

Questo ha consentito si progettisti della app Ticket Management di sviluppare un software con una architettura come quella rappresentata in figura 25 e che presenta le seguenti caratteristiche tecniche:

1. Le viste possono essere sviluppate utilizzando un linguaggio XML puro, o in alternativa XHTML dove l'interprete del linguaggio riconosce i classici costrutti di base dell'HTML 5 con aggiunta di nuovi tag dichiarati dallo sviluppatore.
2. La Root view (HomeTicketManagement.view.xml) è il fulcro di tutta l'app, tutte le altre viste devono essere connesse ed accessibili (direttamente o indirettamente) da quest'ultima.
3. Ogni vista, inclusa la root view, ha il proprio controller che per naming convention definita dal framework SAPUI5, deve aver lo stesso prefisso della vista associata (HomeTicketManagement.controller.js)
4. Il Component.js è un file dove sono definiti tutti i metadati di progetto quali ad esempio, il relative path della root view, l'URI del model, le componenti di base che SAPUI5 si occuperà di avviare attraverso la funzione di init, ed altro ancora.
5. Il Manifest.json è il file descrittore di tutte le componenti e librerie utilizzate, dove

viene specificata anche la versione del framework SAPUI5, se utilizzarlo in CDN (content distribution network) ovvero con download e copia delle cartelle del framework all'interno della root principale di progetto, il routing delle pagine ed altro ancora.

Il default model ed il modello i18n sono i medesimi utilizzati in una qualsiasi altra applicazione Web non necessariamente sviluppata con l'utilizzo del framework SAPUI5. Non contengono, pertanto, caratteristiche tecniche di rilievo da presentare agli occhi del lettore come "particolari".

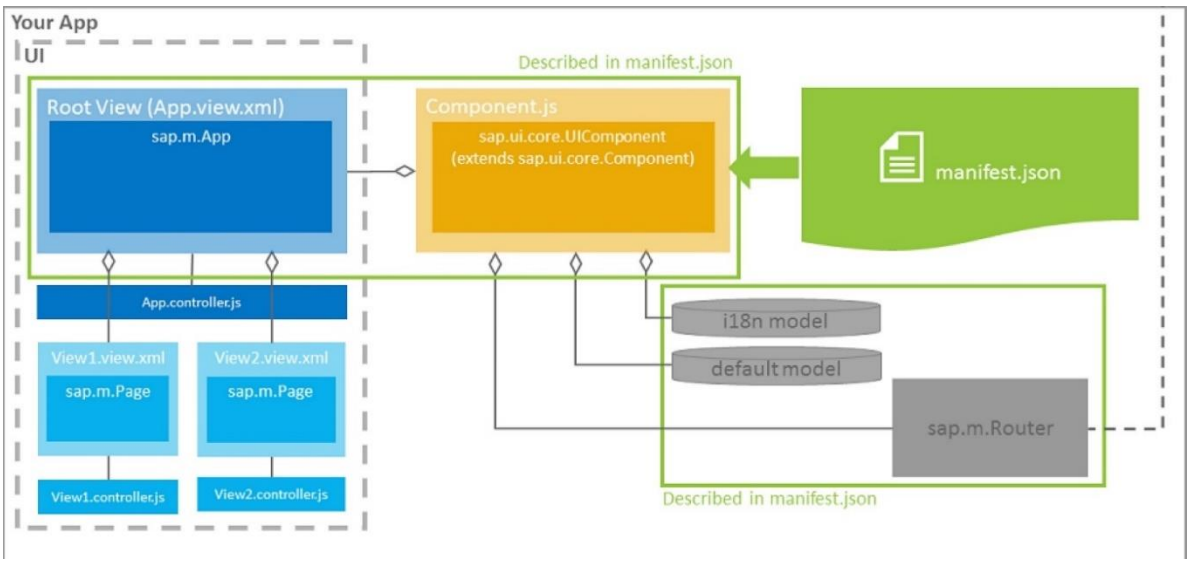



*Figura 25 – Architettura Web application FIORI con framework SAPUI5/OpenUI5.*

## *Pattern di progettazione*

Come già si potrebbe intuire leggendo tra le righe del precedente paragrafo, il pattern di programmazione utilizzato per il progetto Ticket Management è l'MVC (Model View Controller).

MVC prevede un'architettura composta da tre parti, rappresentate in figura 26:

- **Model:** Contiene i metodi di accesso ai dati
- **View:** Realizza la visualizzazione e gestione delle informazioni e delle iterazioni dell'utente con l'infrastruttura sottostante
- **Controller:** Implementa la logica applicativa

È un pattern architetturale molto diffuso nello sviluppo di sistemi software, in particolare nell'ambito della programmazione orientata agli oggetti e nelle applicazioni Web in grado di separare la logica di presentazione dalla logica di business. Si pone all'interno di un contesto hardware di tipo Multi-tier *[26] (multistrato)*.

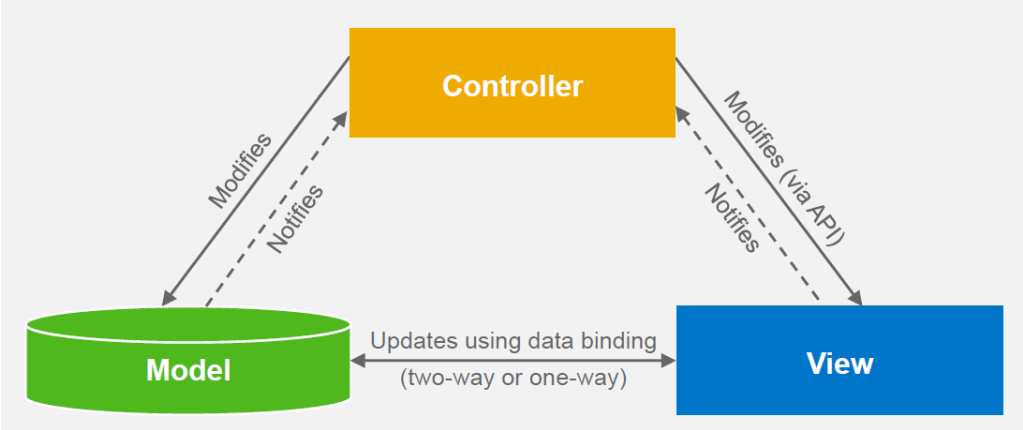



*Figura 26 – Rappresentazione grafica del pattern MVC estratta dal corso "Developing Webapp with SAPUI5"*

## *Linguaggi di Programmazione*

Nella seguente tabella sono elencati i linguaggi di programmazione utilizzati per ognuna delle componenti che formano l'intero progetto Ticket Management.

| COMPONENTE | LINGUAGGIO UTILIZZATO |
|---|---|
| VIEW | XML e CSS |
| CONTROLLER | JavaScript (librerie SAPUI5, OpenUi5 e JQuery) |
| MODEL | JSON |
| BACKEND | Java |
| PERSISTENZA | SQL (dialetto PostgreSQL) |

# Realizzazione & Testing

Nei seguenti paragrafi, che occuperanno lo spazio centrale e finale di questo elaborato, verranno esposte tutte le soluzioni tecniche e funzionali adottate e come hanno permesso di soddisfare gli obiettivi di progetto, intesi come esigenze dell'azienda committente.

Verrà mostrato il software nella sua interezza e nella sua versione finale, così come presentata agli stakeholder, come opera la sua interfaccia utente, ed alcune porzioni di codice ritenute significative per la soluzione dei problemi affrontati.

Volendo dare una rapida panoramica, gli argomenti trattati riguarderanno:

- Prototipo della GUI (Build.me)
- Export GUI (in GitHub)
- Import GUI in SAP Business Application studio (da GitHub)
- Web-Application Ticket Management (SAP Business application studio)
- Backend TicketManagement (SAP Business application studio)
- Scripts di test automatici e reportistica di lancio (test card da analisi dei requisiti, scripts Microfocus UFT, reports Run Result Viewer)
- Benchmarking app (con ServiceNow il più famoso ed affermato competitor)

# *Prototipo Graphic User Interface*

Il primo processo per la creazione della Web application Ticket Management passa per la definizione del prototipo dell'interfaccia mediante l'utilizzo del designer di GUI Build.me. L'impiego di questo designer è stato di grande aiuto perché ha permesso ai committenti del progetto di esprimere feedback determinanti per la definizione delle caratteristiche di ciascuna interfaccia e per la definizione del routing delle pagine.

Di seguito verranno mostrate tutte le principali viste della app, e per ognuna, verranno evidenziate peculiarità tecniche e grafiche.

### ***HomeUser.view.xml***

La vista HomeUser è la root view di Ticket Management, è formata da 3 sezioni di tipo sap.ui.layout.BlockLayout, il layout a blocchi impiegato in applicazioni FIORI per contestualizzare il contenuto informativo (es. finanza, HR, reportistica, ecc.), come è possibile vedere in figura 27.

All'interno della sezione 1 è presente una BlockLayoutRow e due BlockLayoutCell nei quali ambiti sono definite rispettivamente, logo dell'azienda (che per ragioni di privacy è stata omessa dalla seguente figura), e breve descrizione dell'app.

All'interno della sezione 2 è presente una BlockLayoutRow e tre BlockLayoutCell nei quali ambiti sono definiti 3 bottoni del tipo sap.m.standardTile, ognuno dei quali rimanda ad una vista dedicata (o alla stessa vista con diversi filtri).

All'interno della sezione 3 è presente una BlockLayoutRow e tre BlockLayoutCell nei quali sono definiti 3 grafici aggiornati in tempo reale del tipo sap.suite.ui.MicroChart ognuno dei quali riporta i risultati di un'analisi differente.

1. Estrae i ticket in stato “assegnato” o “riaperto” e li divide per priorità/gravità
2. Estrae tutti i ticket in stato “creato” e li divide per priorità/gravità
3. Estrae i ticket in stato “chiuso” e li divide per priorità/gravità (analisi post mortem)

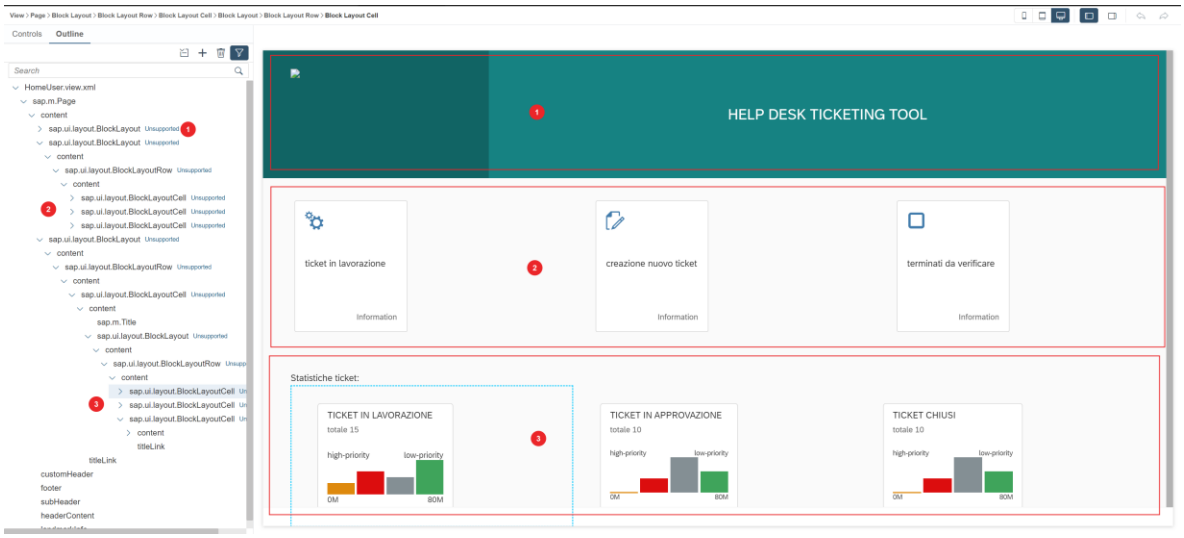



*Figura 27 – Home page della app Ticket management*

## ***ListaTicket.view.xml***

La vista in figura 28 è pensata per essere una single page application[16] ove è possibile svolgere la maggior parte delle operazioni. Nella parte superiore della pagina è presente una BlockLayoutRow che contiene il titolo della pagina e un sap.m.button di ritorno alla home page; direttamente al di sotto di essa una sap.m.Toolbar con all’interno una serie di sap.m.Button ed una barra di ricerca libera sap.m.SearhField.

*16 – Per Single-page application si intende un'applicazione Web che può essere usata su una singola pagina Web con l'obiettivo di fornire una esperienza utente più fluida e simile alle applicazioni desktop.*

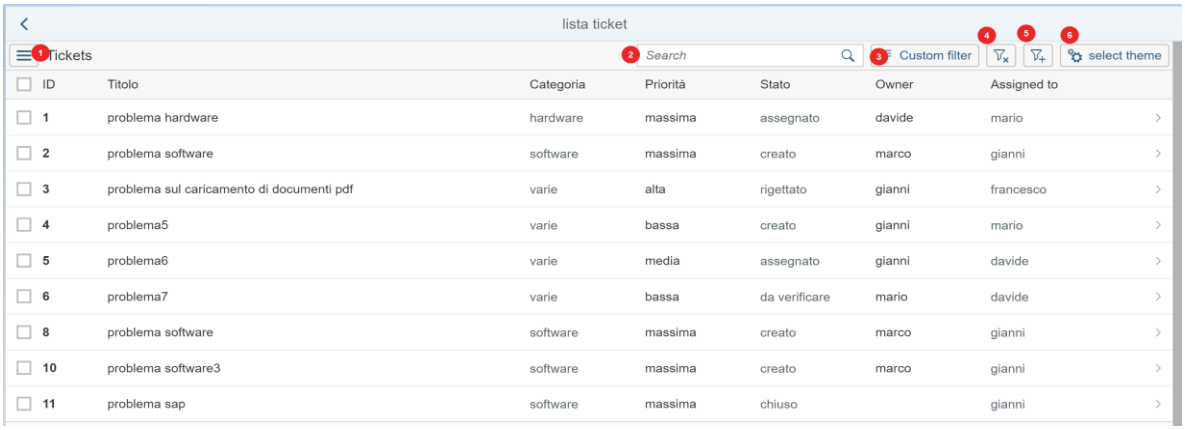



*Figura 28 – Vista Lista Ticket*

Le funzioni sono qui di seguito elencate nel dettaglio:

1. Bottone mostra/nascondi barra dei filtri/ordinamenti preferiti
2. Barra di ricerca libera con suggerimenti (ricerca senza bisogno di submit)
3. Bottone mostra/nascondi pannello di definizione filtri e ordinamenti personalizzati – figura 29
4. Bottone per la rimozione di filtri–ordinamenti–raggruppamenti

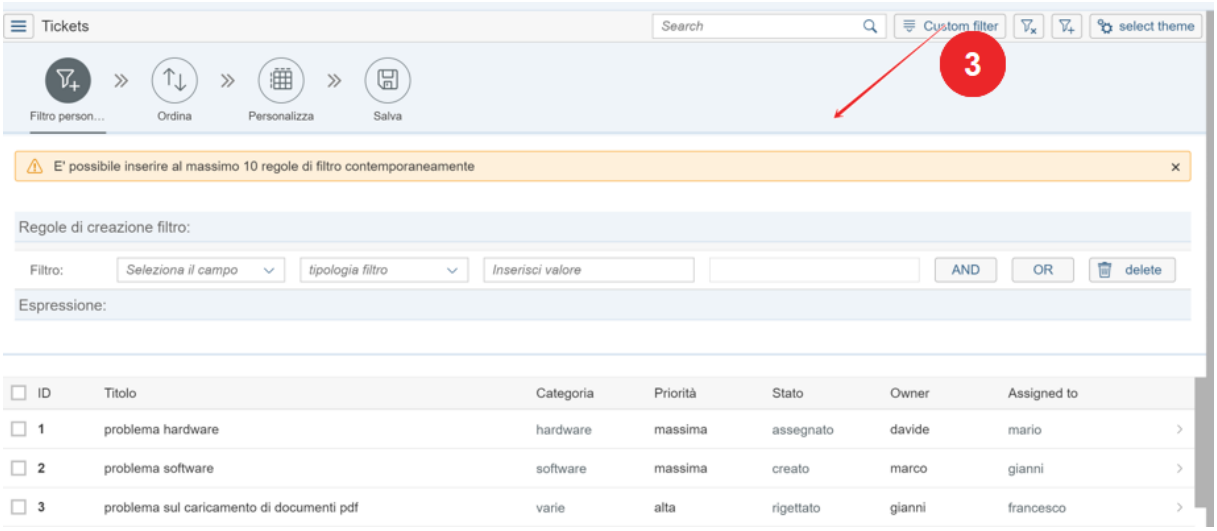



*Figura 29 – Interactive panel per la definizione di filtri-ordinamenti, personalizzazione tabella e salvataggio tra i filtri preferiti*

5. Bottone per la aggiunta di filtri-ordinamenti-raggruppamenti standard – figura 30
6. Bottone per la selezione del tema (alto contrasto/tema default)

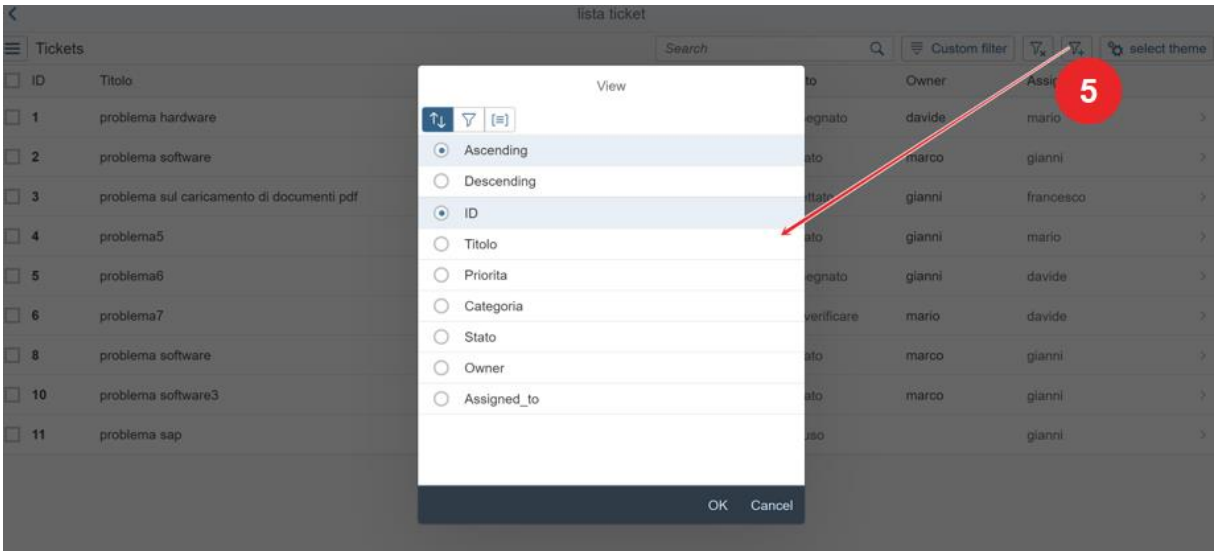



*Figura 30 – Popup view per la definizione di filtri standard*

## ***NewTicket.view.xml***

La vista NewTicket è usata per la apertura di nuovi ticket e si compone di tre parti figura 31:

1. sap.ui.layout.Form con la form che l'utente deve corredare di tutte le informazioni (alcune di esse obbligatorie)
2. sap.m.UploadCollection per l'upload di immagini-video-file
3. sap.m.Bar nel footer con bottoni InviaTicket e annulla inserimento

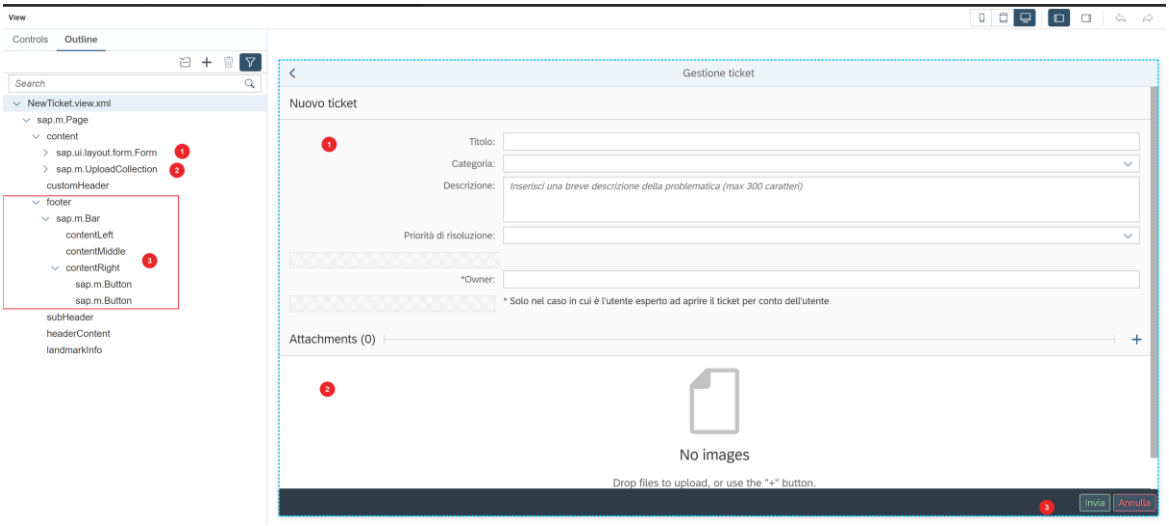



*Figura 31 – Pagina di inserimento nuovo ticket*

## ***TicketDetail.view.xml***

La vista TicketDetail che, come si evince senz'altro dal nome, è la vista per la presentazione delle informazioni di dettaglio di ogni ticket e da cui si può accedere cliccando direttamente sul singolo ticket dalla schermata ListaTicket mostrata in precedenza. È divisa in due sap.m.IconTabFilter all'interno di un'unica sap.m.IconTabBar.

Nel primo TAB "Dettaglio", sono presenti due differenti sezioni: la prima sap.m.Carousel, dove vengono mostrate immagini, video o generici file che l'utente ha caricato, la seconda sezione contiene una sap.m.List con tutte le principali info del ticket, come mostrato in figura 32.

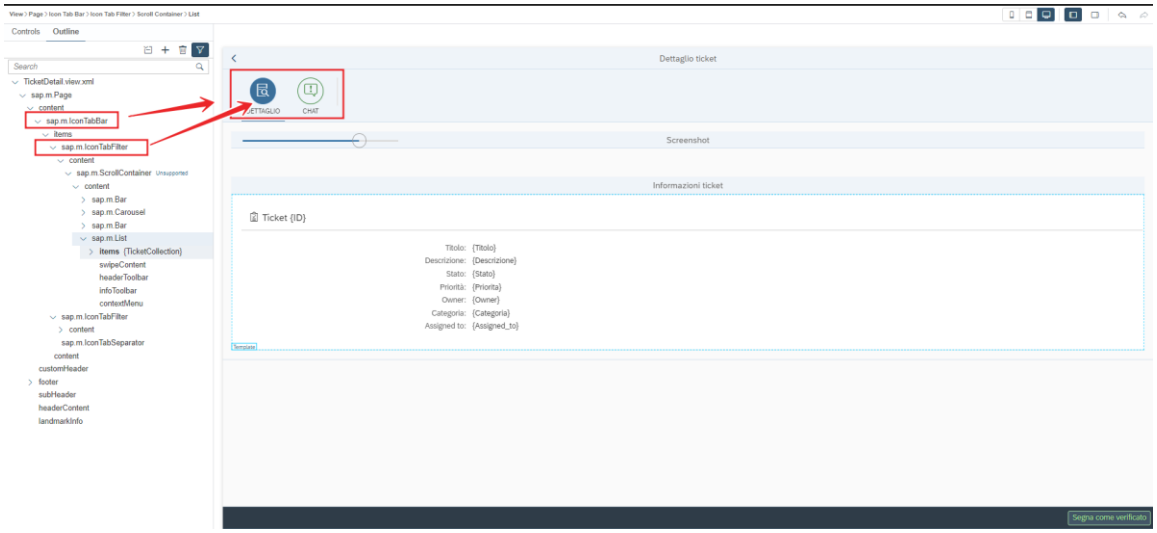



*Figura 32 – Dettaglio ticket*

Nel secondo TAB "chat" è sempre presente una prima sezione con sap.m.Carousel per la presentazione delle immagini caricate dall'utente ed una seconda sezione con elementi sap.m.FeedInput per l'inserimento di un messaggio e sap.m.List per la presentazione dello storico chat con l'utente esperto come in figura 33.

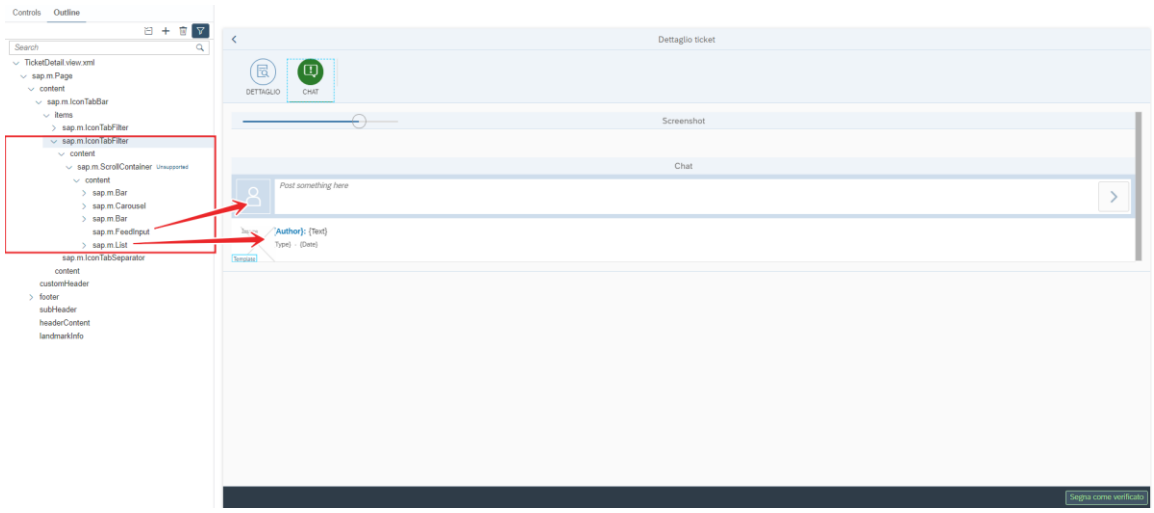



*Figura 33 – Chat con l'utente esparto*

# *Export GUI & Import SAP app Business Studio*

Una volta terminato il design delle interfacce, anche con l'aiuto dei feedbacks degli stakeholders di progetto, è giunto il momento di esportare il prototipo all'interno di un repository GitHub, che porta lo stesso nome del progetto, con il successivo obiettivo di re-importare il prototipo all'interno dell'IDE SAP Application Business Studio.

Riassumendo brevemente tutti gli steps mostrati graficamente in figura 34 si ha:

1. Export del GUI prototype
2. Git Push del prototipo sulla repository Ticket_Management
3. Git Clone del prototipo in Sap App Business Studio

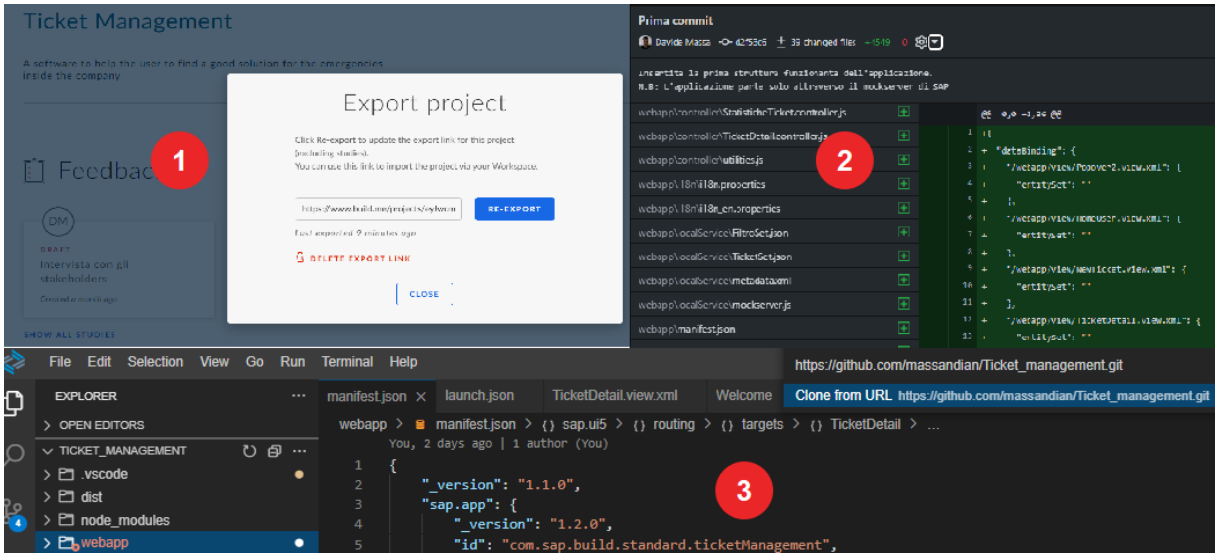



*Figura 34 – le tre fasi: export progetto Build.me, git push del prototipo su repository GitHub e Git Clone del prototipo in SAP Business Application Studio*

## *Web-application Ticket Management*

Ripartendo dai concetti e dai documenti prodotti nella fase di raccolta ed analisi dei requisiti (task card), l'obiettivo del paragrafo è quello di presentare Ticket Management nella sua interezza, mostrando ai lettori alcune funzionalità caratterizzanti ed interessanti dal punto di vista tecnico con l'ausilio, talvolta, di spezzoni di codice sorgente e di commenti (in verde).

È interessante, inoltre partire da una esposizione delle caratteristiche generali quali: binding dei dati (interazione model-view) anche in relazione ai campi di applicabilità di filtri e ordinamenti sulle informazioni presentate, librerie utilizzate nel framework OpenUI5, struttura folders di progetto e informazioni su come l'autenticazione e l'autorizzazione vengono gestite dal sistema.

### ***File di progetto***

Come già introdotto nel paragrafo in cui si è parlato dell'architettura software, il Ticket Management è corredato di alcuni file di progetto che hanno il compito di svolgere delle operazioni preliminari ad ogni avvio dell'applicazione.

Il **Component.js** è il file dove sono definiti tutti i metadati di progetto quali ad esempio: il relative path della root view, l'URI del model, le componenti di base che SAPUI5 si occuperà di avviare attraverso la funzione di init. Di seguito verrà mostrato il codice sorgente e si evidenzieranno con commenti le principali informazioni:

```
sap.ui.define([
    /**
* Si richiamano tutte le componenti del framework necessarie per   l'avvio
     * dell'applicazione, ed il localModel e le si passano alla funzione.
     */
    "sap/ui/core/UIComponent",
    "sap/ui/Device",
    "com/sap/build/standard/ticketManagement/model/models",
    "sap/ui/model/json/JSONModel"
], function(UIComponent, Device, models, JSONModel) {
    "use strict";
```

```
     var navigationWithContext = {

    };
    return UIComponent.extend("com.sap.build.standard.ticketManagement.Component",
{
        metadata: { //si definisce la root view, la start page
            manifest: "json",
            rootView: "com.sap.build.standard.ticketManagement.view.HomeUser",
            routing: {
                routes: [
                    {
                        name: "master",
                        pattern: "",
                        target: ["master"]
                    },
                    {
                        name: "ticketDetails",
                        pattern: "TicketSet/:ID:",
                        target: ["master", "ticketDetails"]
                    }
                ]
            }
        },

        /**
         * Le seguenti componenti sono avviate da UI5 durante l'avvio della
         * applicazione
         * @public
         * @override
         */
        init: function() {
            // set the device model
            this.setModel(models.createDeviceModel(), "device");
            // set the FLP model
            this.setModel(models.createFLPModel(), "FLP");
            // set the dataSource model
            this.setModel(new sap.ui.model.json.JSONModel({
                "uri": "/model/model.js"
            }), "dataSource");

            // set application model
            var oApplicationModel = new sap.ui.model.json.JSONModel({});
            this.setModel(oApplicationModel, "applicationModel");

            // call the base component's init function
            UIComponent.prototype.init.apply(this, arguments);

            // create the views based on the url/hash
            this.getRouter().initialize();
        },

        createContent: function() { //la app viene containerizzata in un SAP
context
            var app = new sap.m.App({
                id: "App"
            });
            var appType = "App";
```

```
            var appBackgroundColor = "#FFFFFF";
            if (appType === "App" && appBackgroundColor) {
                app.setBackgroundColor(appBackgroundColor);
            }
            return app;
        },

        getNavigationPropertyForNavigationWithContext: function(sEntityNameSet, ta
rgetPageName) {
            var entityNavigations = navigationWithContext[sEntityNameSet];
            return entityNavigations == null ? null :
entityNavigations[targetPageName];
        }
    });
});
```

Il **Manifest.json** è il descrittore di tutte le librerie utilizzate, definisce il routing delle pagine, la versione della app, del protocollo Odata usato, del framework UI5 e contiene l'URL relativo dei file CSS per lo stile delle viste. In Ticket Management, sono state poche le classi di stile CSS su cui si è dovuto intervenire, dato che la costruzione delle interfacce è stata fatta con BUILD.ME seguendo i feedbacks degli stakeholders:

```
{
    "_version": "1.1.0",
    "sap.app": {
        "_version": "1.2.0", //versione applicazione Ticket Management
        "id": "com.sap.build.standard.ticketManagement",
        "type": "application",
        "i18n": "i18n/i18n_it.properties",
        "applicationVersion": {
            "version": "1.0.0"
        },
        "dataSources": {
            "local": {
                "uri": "localService/metadata.xml",
                "type": "OData",
                "settings": {
                    "odataVersion": "2.0", //versione protocollo Odata
                    "localUri": "localService/metadata.xml"
                }
            }
        },
        "title": "{{appTitle}}",
        "description": "{{appDescription}}",
        "ach": "ach",
        "resources": "resources.json",
        "sourceTemplate": {
            "id": "ui5template.basicSAPUI5ApplicationProject",
            "version": "1.96.9",
            "toolsId": "eeebe379-e802-462c-9940-cd3735cf432b"
        }
    },
    "sap.ui": {
```

```
        ..................
    },
    "sap.ui5": {
        "_version": "1.96.9", //versione UI5 utilizzata per il run
        "rootView": {
            "viewName": "generated.app.view.HomeUser",
            "type": "XML"
        },
        "dependencies": {
            "minUI5Version": "1.32.0", //versione minima UI5 supportata
            "libs": {

"sap.ui.core": "https://openui5.hana.ondemand.com/resources/sap-ui-core.js", //URI
per CDN (content distribution network) delle librerie OpenUi5
                "sap.m": {},
                "sap.ui.layout": {}
            }
        },
        "contentDensities": {
            "compact": true,
            "cozy": true
        },
        "models": {
            "i18n": {
                "type": "sap.ui.model.resource.ResourceModel",
                "uri": "i18n/i18n.properties"
            },
            "OdataModel": {
                "dataSource": "local",
                "type": "sap.ui.model.odata.v2.ODataModel",
                "settings": {
                    "loadMetadataAsync": false,
                    "json": true,
                    "bJSON": true,
                    "defaultBindingMode": "TwoWay",
                    "useBatch": true,
                    "refreshAfterChange": false,
                    "disableHeadRequestForToken": true
                }
            }
        },
        "resources": {
            "css": [
                {
                    "uri": "css/style.css"
                }
            ]
        },
        "routing": { //routing delle pagine
            "config": {
                "routerClass": "sap.m.routing.Router",
                "viewType": "XML",
                "viewPath": "com.sap.build.standard.ticketManagement.view",
                "controlId": "App",
                "clearTarget": false,
                "controlAggregation": "pages",
                "bypassed": {
```

```
                    "target": [
                        "HomeUser"
                    ]
                }
            },
            "targets": { //si espone la struttura JSON solo per la root view, si
omettono le restanti
                "HomeUser": {
                    "controlAggregation": "pages",
                    "viewName": "HomeUser",
                    "viewId": "HomeUser",
                    "viewLevel": 1,
                    "transition": "slide"
                },
                "NewTicket": {
                   ..................
                },
                "TicketDetail": {
                   ..................
                },
                "StatisticheTicket": {
                   ..................
                },
                "ListaTicket": {
                   ..................
                }
            },
            "routes": [
                {..................},
                {
                    "pattern": "",
                    "name": "default",
                    "target": [
                        "HomeUser"
                    ]
                },
                {..................},
                {..................},
                {..................},
                {
                    "pattern": "ListaTicket/:context:",
                    "name": "ListaTicket",
                    "target": [
                        "ListaTicket"
                    ]
                }
            ]
        }
    },
  ..................
}
```

### ***Login Web Application***

SAP Single Sign-on (SSO) fornisce agli utenti un accesso controllato alle applicazioni del mondo SAP e non SAP con un'unica password. Migliora l'esperienza e la facilità d'uso di tutti i prodotti, supporta tutti i protocolli di sicurezza e semplifica l'amministrazione delle utenze [27]. Ticket Management risulterà usabile dall'utente, previo accesso nella SAP Fiori dashboard mediante procedura di SSO, come mostrato in figura 35:

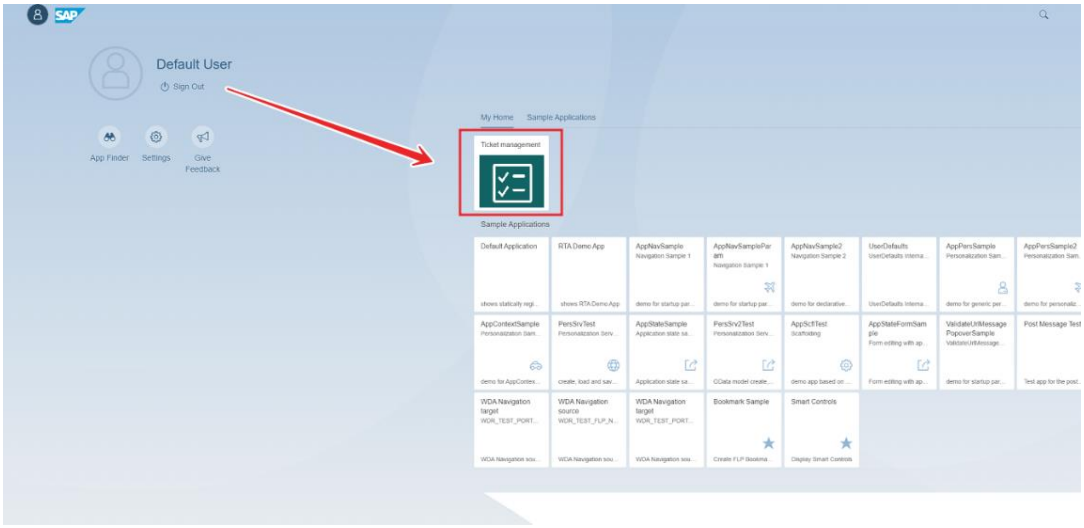



*Figura 35 – Ticket Management disponibile nella Fiori dashboard dell'utente loggato mediante SSO*

### ***Gestione dei ruoli e delle Autorizzazioni***

Per la gestione delle utenze, dei ruoli e delle autorizzazioni, si utilizzerà la SAP BTP Cockpit (Business Technology Platform), una interfaccia grafica di gestione delle utenze, basata sui moduli SAP HR (human resources) *[28]*.

Quando la Web application sarà consegnata all'azienda committente il team, che si occupa di gestire i profili digitali dei dipendenti, avrà l'onere di creare le utenze SAP SSO ed

assegnare i ruoli per Ticket Management come mostrato in figura 36.

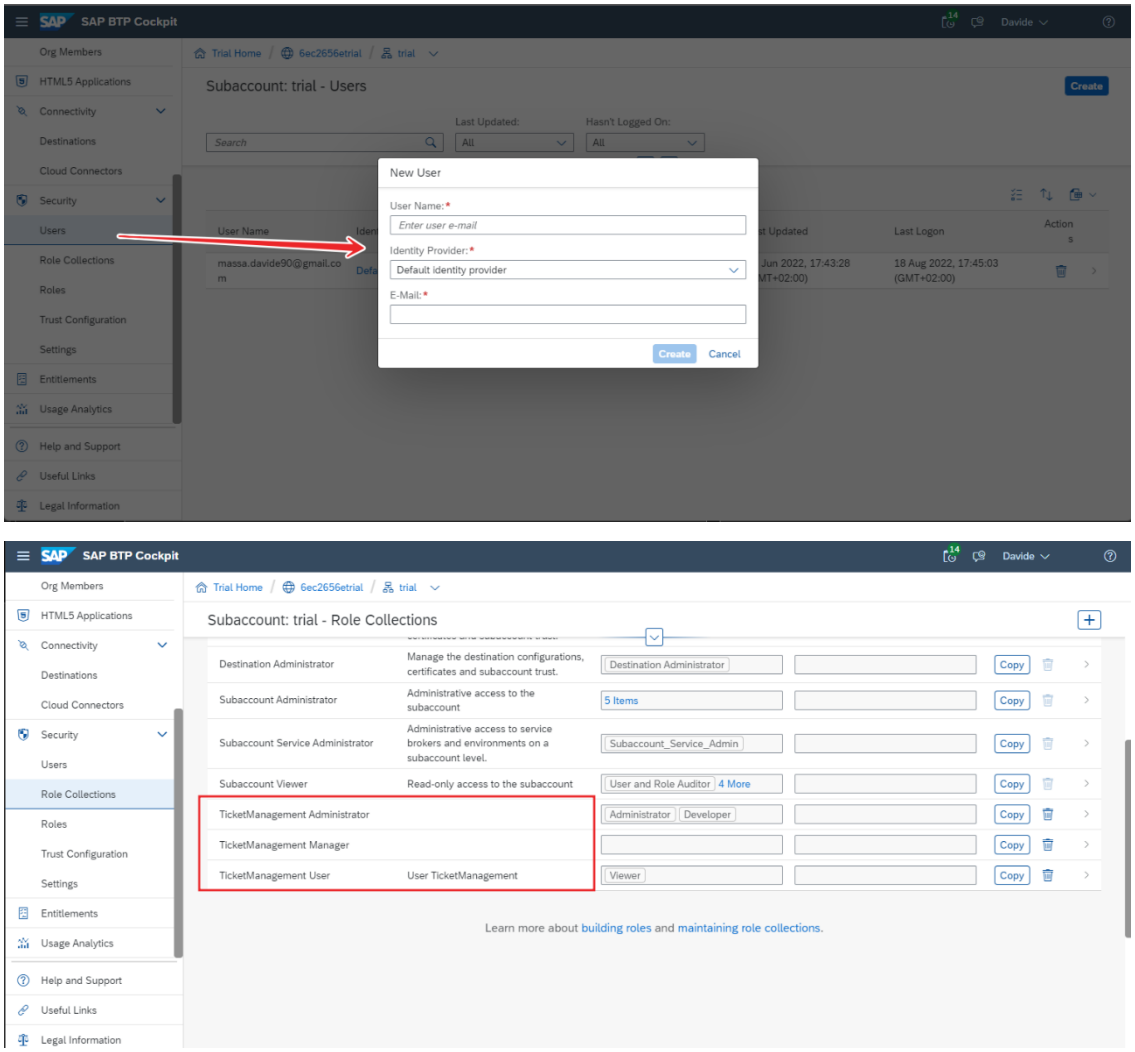



*Figura 36 – Procedura inserimento nuovo utente Ticket Management e assegnazione ruoli*

Le utenze di tipo administrator sono state create proprio a tal proposito, con l'unico scopo di gestire autenticazioni ed autorizzazioni, e la seguente funzione Java riassume quanto appena detto.

```
public User addUser(
        @EdmFunctionImportParameter(name = "uname"    , facets = @EdmFacets(nullable
= false, maxLength = 100)) final String uname,
        @EdmFunctionImportParameter(name = "password", facets = @EdmFacets(nullable
= false, maxLength = 20 )) final String password,
```

```
        @EdmFunctionImportParameter(name = "email"    , facets = @EdmFacets(nullable
= false, maxLength = 100)) final String email,
        @EdmFunctionImportParameter(name = "fullname", facets = @EdmFacets(nullable
= false, maxLength = 100)) final String fullname
        ) throws ODataException, InvocationTargetException {

    PersistenceFactory uupf = UurlaServiceKit.getPersistenceFactoryImpl();
    EntityManager em = uupf.getEmf().createEntityManager();

    User us= em.find(User.class,uupf.getUser());
    List<Role> lr=us.getRoles();
    boolean flag=false;

    //Ciclo sulla lista dei ruoli dell'user
    for(Role r:lr){
        //Se l'user è admin settiamo il flag su true ed usciamo dal ciclo
        if(r.getRolename().equals("admin")){
            flag = true;
            break;
        }
    }

    //Nel caso non si è admin (e quindi il flag è false) mostra il      messaggio
di negata autorizzazione ad effettuare l'operazione
    if(!flag){
        throw new ODataException("no authorized");
    }

    try {

        em.getTransaction().begin();

        UserDao userDao = UserDao.getInstance();
        us=userDao.addNewUser(em, uname, password, email, fullname);

        em.getTransaction().commit();

    } catch (Exception e) {
        em.getTransaction().rollback();
        throw
createHandlerException(this.getLogger(),"\"GeneralODataFunctions.addUser()",e);
    } finally {
        if (em.getTransaction().isActive()) {
            em.getTransaction().rollback();
        }
        em.close();
    }
    return us;
}
```

Quanto appena mostrato per la AddUser, vale anche per la EditUser e per la DeleteUser. Si vuol inoltre far notare che l'utenza Amministratore non è stata menzionata in fase di analisi dei requisiti, questo perché non rappresenta un attore con un business role rilevante dal punto di vista funzionale. La mission è rilevante da un punto di vista

prettamente tecnico e non rappresenta una innovazione rispetto ad altri prodotti software presenti nel mercato.

### ***Binding dei dati***

Il binding dei dati dal prototype model (utilizzato per testare l'applicazione in locale) e dal model, viene gestito nella medesima maniera. Tutte le informazioni da leggere e manipolare risiedono nel database relazionale e vengono pubblicate dal backend mediante local services, API asincrone[17] che si occupano di esporre tutti i dati a runtime.

Avremo così una serie di endpoint da interrogare mediante il protocollo Odata ed attraverso chiamate:

- HTTP GET per la lettura delle informazioni dal database *(es. binding delle informazioni ticket nella vista ListaTicket.view.xml)* figura 37

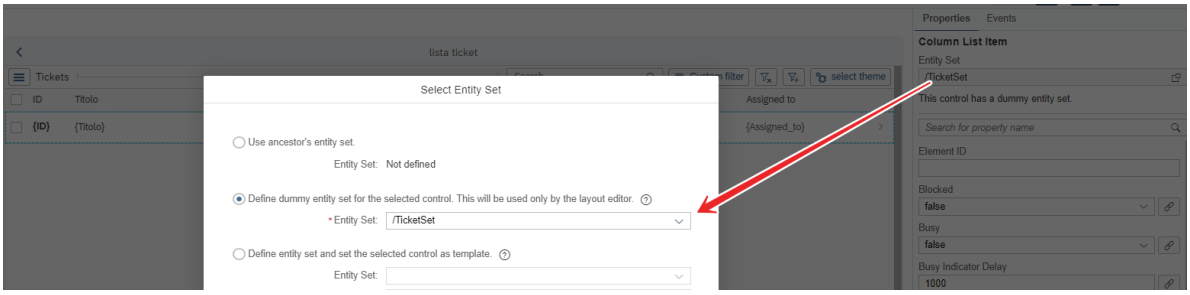



*Figura 37 –Binding lista ticket aperti attraverso la collection /TicketSet estratta a partire da una chiamata Odata http GET*

- HTTP POST per la manipolazione delle informazioni sul database *(es. all'inserimento di un nuovo ticket l'evento click sul bottone invia scatena una http post ed una DB insert)*

---

*17 - Le API asincrone sono interfacce di programmazione dell'applicazione che restituiscono i dati per le richieste in un secondo momento. Le API asincrone forniscono un modo per effettuare richieste pianificate di risorse, dati o servizi quando disponibili. Sono utili per mantenere la funzionalità in un'applicazione piuttosto che vincolare le risorse dell'applicazione in attesa di una richiesta.*

### ***Inserimento nuovo ticket e Upload Media***

In questa procedura l'utente, per mezzo della vista NewTicket.view.xml, crea un nuovo ticket inserendo alcune informazioni obbligatorie nella form e con la possibilità di caricare screenshots, immagini scattate con altri dispositivi, video, file di testo, in generale qualsiasi file che possa essere di aiuto al tecnico per comprendere il problema ed aiutarlo nella ricerca di una soluzione.

*Il codice di questa procedura viene mostrato in Appendice A*

### ***Dettaglio ticket e Chat con utente esperto***

Nella schermata TicketDetails.view.xml l'utente può monitorare lo stato del ticket, leggere le informazioni, interagire con l'utente esperto che ha preso in carico il task attraverso la chat dedicata e "segnare come verificati" i ticket che sono in stato "da verificare" aperti da lui o per suo conto da un utente esperto *(ticket con diverso owner)*, ed effettivamente risolti.

*Il codice di questa procedura viene mostrato in Appendice B*

### ***Utilizzo di filtri-ordinamenti-raggruppamenti standard***

Nella pagina ListaTicket.view.xml è possibile definire una serie di filtri, ordinamenti e raggruppamenti standard in quanto messi a disposizione del framework OpenUI5. L'evento onClick sul bottone contrassegnato da questo simbolo  apre una vista popup con tre diversi TAB dove l'utente può definire rispettivamente filtri, ordinamenti e raggruppamenti, o parte di essi, per la lista ticket da mostrare.

```
//Funzione che apre la dialog per la definizione delle regole di filtro standard
        handleViewSettingsDialogButtonPressed: function(oEvent) {

            if (!this._oDialog) {
                this._oDialog =
sap.ui.xmlfragment("com.sap.build.standard.ticketManagement.view.ViewSettingsDialog
3", this);
            }
```

```
            // toggle compact style
            jQuery.sap.syncStyleClass("sapUiSizeCompact", this.getView(),
this._oDialog);
            this._oDialog.open();
        },

        handleConfirm: function(oEvent) {

            var oView = this.getView();
            var oTable = oView.byId("sap_Responsive_Page_0-content-
sap_ui_layout_BlockLayout-1515407526987-content-sap_ui_layout_BlockLayoutRow-2-
content-sap_ui_layout_BlockLayoutCell-1-content-build_simple_Table-1515407548335");

            var mParams = oEvent.getParameters();
            var oBinding = oTable.getBinding("items");

            // Applica ordinamenti al binding tra model e view
            // (grouping comes before sorting)
            var sPath;
            var bDescending;
            var vGroup;
            var aSorters = [];
            if (mParams.groupItem) {
                sPath = mParams.groupItem.getKey();
                bDescending = mParams.groupDescending;
                vGroup = this.mGroupFunctions[sPath];
                aSorters.push(new Sorter(sPath, bDescending, vGroup));
            }
            sPath = mParams.sortItem.getKey();
            bDescending = mParams.sortDescending;
            aSorters.push(new Sorter(sPath, bDescending));
            oBinding.sort(aSorters);

            // Applica i filtri al binding tra model e view
            var aFilters = [];
            jQuery.each(mParams.filterItems, function (i, oItem) {
                var aSplit = oItem.getKey().split("___");
                var sPath = aSplit[0];
                var sOperator = aSplit[1];
                var sValue1 = aSplit[2];
                var oFilter = new Filter(sPath, sOperator, sValue1);
                aFilters.push(oFilter);
            });
            oBinding.filter(aFilters);

        },


        filterCountFormatter: function(sValue1, sValue2) {
            return sValue1 !== "" || sValue2 !== "" ? 1 : 0;

        },
```

Leggendo questa parte di codice si nota come il filtro agisce sul binding delle informazioni tra model e viste, come mostrato graficamente in figura 38

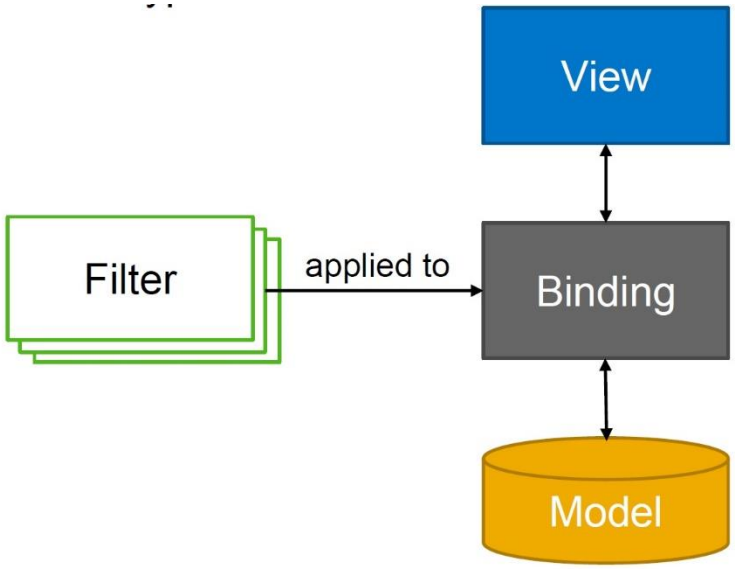



*Figura 38 – I filtri agiscono sul binding delle informazioni tra model e viste*

***Definizione di filtri-ordinamenti-raggruppamenti personalizzati***

È possibile definire e salvare filtri, ordinamenti e raggruppamenti completamente personalizzabili, attraverso un menu a comparsa dall'alto, cliccando sul bottone con il simbolo "custom filter". Ticket management rappresenterà in tempo reale, a schermo, l'espressione logico matematica definita.

*Il codice di questa procedura viene mostrato in Appendice C*

***Ricerca ticket e cancellazione filtri***

È possibile cercare un ticket attraverso la barra di ricerca libera e cancellare filtri e ricerche precedentemente definiti mediante il bottone dedicato.

*Il codice di questa procedura viene mostrato in Appendice* D

## *Testing scripts & Testing reports*

Prima di consegnare il software Ticket Management all'azienda committente, bisogna effettuare una serie di test manuali e sviluppare degli script di test automatici che possano supportare il team degli sviluppatori anche in futuro, per il rilascio di nuove versioni della Web application che siano stabili e funzionanti al 100%.

Si vuol mostrare uno script automatico, sviluppato a partire da un test case/test card già esposto precedentemente. I test manuali non rappresentano una particolare innovazione tecnologica o procedurale, pertanto, non verranno mostrati al lettore.

- Script Inserimento nuovo ticket e upload media (Vbscript)

```
'Inserimento titolo ticket
Browser("appTitle").Page("appTitle").SAPUITextEdit("Titolo").Set "Ticket di test"

'Selezione categoria
Browser("appTitle_2").Page("appTitle").SAPUIList("SAPUIList_2").Select "Software"


'Descrizione Problema
Browser("appTitle").Page("appTitle").SAPUITextEdit("Descrizione").Set "La
applicazione per l'inserimento dei turni degli operai di produzione sembra non
funzionare. Viene mostrato un errore 404"

'Severity/priority ticket
Browser("appTitle_2").Page("appTitle").SAPUIList("SAPUIList_3").Select "Massima"

'Owner Ticket
Browser("appTitle").Page("appTitle").SAPUITextEdit("*Owner").Set "Marco di Febo"
Browser("appTitle").Page("appTitle").WebElement("__container5-Grid").Click

'Upload file media non supportato
Window("Google Chrome").WinObject("Chrome Legacy Window").Click 1580,447
wait(1)
Window("Google Chrome").Dialog("Apri").WinComboBox("ComboBox").Select "Tutti i file
(*.*)"
Window("Google Chrome").Dialog("Apri").WinObject("Visualizzazione
elementi").WinList("Visualizzazione elementi").Select ".project.json"
Window("Google Chrome").Dialog("Apri").WinButton("Apri").Click

'Checkpoint - si verifica che effettivamente il sistema mostri un errore all'utente
'in caso contrario il test fallisce e questo malfunzionamento viene evidenziato nel
report
Browser("appTitle").Page("appTitle").WebElement("WebElement").Check
CheckPoint("media_non_supportato")
```

```
'Upload file media supportato
Window("Google Chrome").WinObject("Chrome Legacy Window").Click 1580,447
wait(1)
Window("Google Chrome").Dialog("Apri").WinObject("Visualizzazione
elementi").WinList("Visualizzazione elementi").Select "LOGO_UNIVPM_390x154px.gif"
Window("Google Chrome").Dialog("Apri").WinButton("Apri").Click

'Checkpoint - si verifica che l'upload è andato a buon fine
Browser("appTitle").Page("appTitle").WebElement("WebElement_2").Check
CheckPoint("media_supportato")

'Upload file media supportato
Window("Google Chrome").WinObject("Chrome Legacy Window").Click 1580,447
wait(1)
Window("Google Chrome").Dialog("Apri").WinObject("Visualizzazione
elementi").WinList("Visualizzazione elementi").Select "LOGO_UNIVPM_390x154px.gif"
Window("Google Chrome").Dialog("Apri").WinButton("Apri").Click

'Cancellazione media duplicato
Browser("appTitle_2").Page("appTitle").SAPUIButton("Delete").Click
Browser("appTitle").Page("appTitle").SAPUIButton("OK").Click

'Uplodad ticket
Browser("appTitle").Page("appTitle").SAPUIButton("Invia").Click

'Checkpoint - ticket creato e salvato correttamente
Browser("appTitle").Page("appTitle").WebElement("WebElement_3").Check
CheckPoint("UploadSuccess")
```

- Report generato dallo script (figura 39)

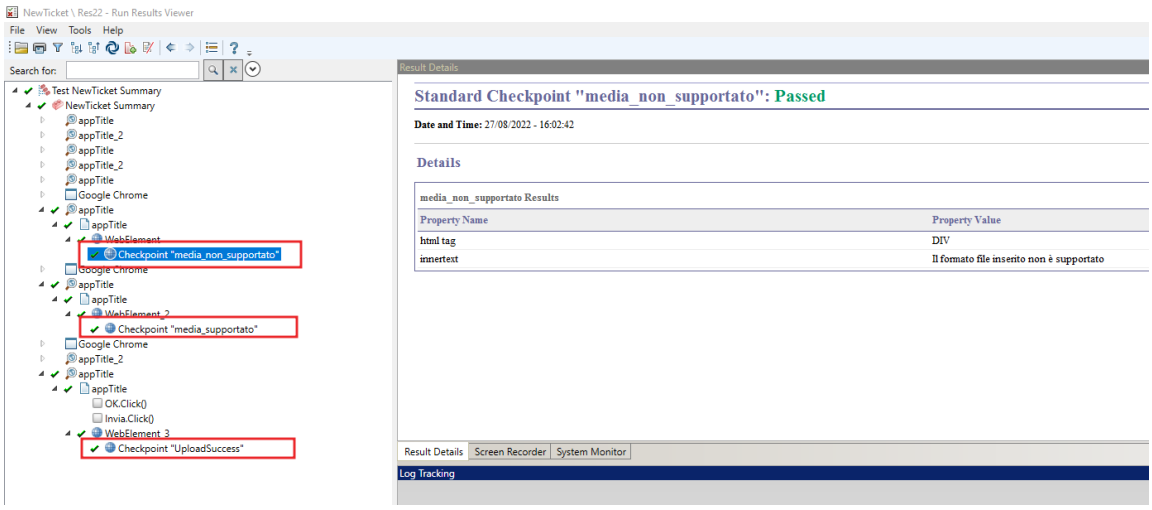



*Figure 39 – Evidenziati in rosso i tre checkpoint dello script che servono per determinare il successo o il fallimento dello stesso.*

**NOTA:** È possibile anche esportare il report in formato pdf corredato di tutti gli screenshots e le informazioni, ed allegarlo al progetto come prova di quali e quanti test sono stati eseguiti sull'applicazione consegnata all'azienda committente.

## *Ticket Management vs ServiceNow*

ServiceNow come già enunciato nel capitolo 4, è una piattaforma di cloud computing che aiuta le aziende nella gestione dei flussi di lavoro digitali per le operazioni aziendali.

Ticket Management non ha attualmente la capacità di poter competere con un simile colosso, né dal punto di vista delle infrastrutture, né tantomeno dal punto di vista del servizio offerto. Risulta però quantomeno stimolante ed interessante poter fare una analisi comparativa tra le due Web app. per comprenderne differenze.

La seguente tabella mostra alcuni punti di confronto tra le due web application:

| | **Ticket Management** | **ServiceNow** |
|---|---|---|
| **Infrastruttura hardware** | Three tier con possibilità di hosting su datacenter interni o esterni all'azienda | Hosting fornito da cloud servers dell'azienda per rispettare il service level agreement (SLA) |
| **Framework** | SAPUI5 & OpenUI5 | AngularJS |
| **Supporto** | Supporto 24/7 per la app e 16/5 per l'infrastruttura hardware | Supporto via E-Mail certificata |
| **Costi** | Costo di sviluppo, customizzazione e installazione della soluzione + 15% annuale per assistenza e supporto | Costo di installazione e configurazione della soluzione consolidata e acquisizione delle licenze. |

| | | |
|---|---|---|
| **Knowledge base** | In via di sviluppo | Presente |
| **Licenza** | Open Source | Licenza da acquistare in base al numero di server necessari |

# Capitolo 6 – Risultati e Sviluppi futuri

## Considerazioni finali

L'utilizzo di un framework consolidato come SAPUI5, seppur nella sua versione open source OpenUI5, consente di creare un'applicazione ben strutturata dal punto di vista del frontend ed altamente consistente per quanto riguarda il backend e la persistenza. Questi rappresentano senz'altro dei buoni prerequisiti per un software con la minor presenza possibile di bugs e che consente, al testing team, di disporre di una base di lavoro immediatamente stabile per lo sviluppo di script automatici di testing.

Avere una strategia di design e implementazione comune, rende la app molto intuitiva per l'utente sin dal primo utilizzo e simile a tutte le altre sviluppate con la stessa strategia. I temi proposti infatti, conferiscono alla app il medesimo aspetto delle altre presenti sulla dashboard FIORI, ma lascia poco spazio alla personalizzazione ove si tenti di uscire dai binari e dai paradigmi imposti dall'Azienda SAP.

Tra tutte le tecnologie sperimentate per lo sviluppo di ticket management questa, comunque, è quella che lascia più libertà per lo sviluppo, dato che le componenti autogenerate dal designer delle interfacce sono poche e, comunque, sempre modificabili dall'IDE. È necessaria, quindi, una maggior competenza nella programmazione.

# Sviluppi futuri

L'applicazione da ticket management utilizzata unicamente per la gestione di incidents aziendali di vario genere, potrebbe evolvere e diventare una applicazione di gestione di flussi aziendali digitali con i seguenti ulteriori sviluppi:

1. Estensione della portata dell'applicazione. Non più unicamente gestione di incidents, ma anche richieste di cambiamenti di procedure interne *(changes)* o nuove technical features e investigazioni su problemi già risolti *(post mortem analysis)*. Insomma, il software come più in generale tutto l'ERP, deve essere autoreferenziale, cioè deve avere la possibilità sotto precisi feedback dei singoli utenti, di essere migliorata nel tempo.
2. Gestione task di lavoro per consulenti esterni ed interni all'azienda
    - Aggiornamento task (consulenti)
    - Stima del lavoro/creazione/chiusura task (project manager/dirigente IT)
3. Gestione fogli presenze *(timesheets)* per consulenti esterni ed interni all'azienda
    - inserimento/consultazione (consulenti)
    - consultazione globale (supervisor/dirigenti human resources)
4. Creazione e consultazione di report riguardanti l'efficienza delle singole risorse da parte del team delle risorse umane, abbinando timesheet e quindi capacità della singola risorsa, alle tempistiche stimate per un singolo task di lavoro.
    - Creazione di report (tecnici IT)
    - Consultazione (project manager/dirigenti IT)

5. Creazione/gestione di una knowledge base per velocizzare le attività di risoluzione incident e sensibilizzare gli utenti alla self resolution, cioè formarli alla risoluzione indipendente dei problemi mediante l'utilizzo intelligente dello strumento. Questo contribuirà senz'altro alla creazione di quelle che si definiscono competenze trasversali della risorsa *(T-SHAPE)*, che possono aiutare l'utente anche per problemi esterni al proprio ambito di lavoro.
    - Creazione di documentazione per il knowledge base (personale tecnico, IT, contabilità, amministrazione, human resources, etc)
    - Approvazione della documentazione (project managers)
    - Consultazione della documentazione (utente/super utente/dirigente)
6. Creazione/gestione/analisi di sondaggi a cui i rispettivi dirigenti delle singole aree aziendali sottopongono tutte le risorse del proprio team e dove le stesse possono eventualmente fornire feedback.
    - Creazione sondaggi (project managers, tecnici)
    - Completamento surveys (utenti, tecnici)
    - Analisi surveys (dirigenti)

# Bibliografia/Sitografia


[1] - *Klaus Schwab – "The fourth industrial revolution" – World economic forum, 2015*

[2] - *https://itsolutionsrl.it/2017/07/industria-4-0/ - Marzo 2022*

[3] - *Peter Mell, Timothy Grance, "The NIST Definition of Cloud Computing" NIST, Special Publication 800-145, Settembre 2011.*

[4] - *Giampio Bracchi, Chiara Francalanci, Gianmario Motta, "Sistemi informativi d'impresa", MC-Graw Hill, 2010*

[5] - *Gabriele Levy, "La logistica nei sistemi ERP", Milano, Franco Angeli Editore, 2006*

[6] - *https://www.1c-erp.it/supporto/faq/cosa-significa-erp/ - Marzo 2022*

[7] - *https://news.beta80group.it/servicenow-cos-e-come-funziona-e-perche-serve-alle-aziende – marzo 2022*

[8] - *https://www.servicenow.com/it/products/itsm.html - Aprile 2022*

[9] - *Roger S. Pressman, "Principi di ingegneria del software", McGraw-Hill 2008*

[10] - *Jim Arlow, Ila Neustadt, "UML 2 e Unified Process", McGraw-Hill, 2006*

[11] - *https://www.sap.com/italy/about/company/what-is-sap.html- Aprile 2022*

[12] - *https://www.tcodesearch.com/sap-tcodes – Aprile 2022*

[13] - *"Understand SAP Fiori Launchpad and Applications", https://open.sap.com/ - OpenSAP Courses, 2018*

[14] - *C. Guebels, D. Nepraunig, T. Seidel, "SAPUI5–The comprehensive guide", Rheinwerk Publishing, 2016*

[15] - *"Developing Web-App using SAPUI5", https://open.sap.com/ - OpenSAP Courses, 2016*

[16] - *"Design your first App with Build", https://open.sap.com/ - OpenSAP Courses, 2017*

[17] - *https://www.odata.org/documentation/ - Aprile 2022*

[18] - *https://www.odata.org/documentation/odata-version-2-0/uri-conventions/ - Aprile 2022*

[19] - https://github.com/ - Aprile 2022

[20] - S.Chacon, B. Straub, "Pro Git", Apress 2021

[21] - https://www.postgresql.org/ - Aprile 2022

[22] - C. Ghezzi, M. Jazayeri, D. Mandrioli, "Ingegneria del Software, fondamenti e principi", seconda edizione Pearson Prentice Hall, 2004

[23] - https://www.microfocus.com/it-it/products/uft-one/overview - Maggio 2022

[24] - K. Beck, "Test-Driven Development by Example", Addison Wesley, 2003

[25] - ExtremeProgramming.org, http://wiki.c2.com/?PlanningGame – Maggio 2022

[26] - Kruchten, Philippe. The Rational Unified Process: An Introduction (3rd Ed.) 2004

[27] - https://martinfowler.com/eaaCatalog/modelViewController.html - Luglio 2022

[28] - https://community.sap.com/topics/single-sign-on - Luglio 2022

[29] - https://account.hana.ondemand.com/#/home/welcome - Luglio 2022

# Appendice A

## ***Inserimento nuovo ticket e Upload Media***

```
sap.ui.define(["sap/ui/core/mvc/Controller",
    "sap/m/MessageBox",
    "./utilities",
    "sap/ui/core/routing/History",
    "sap/m/UploadCollectionParameter", //Componente libreria upload media
    "sap/ui/model/json/JSONModel",
    "sap/m/MessageToast",
    "jquery.sap.global"],

    function(BaseController, MessageBox, Utilities, History,
UploadCollectionParameter,JSONModel, MessageToast) {
    "use strict"; //function binding dati model-view iniziale, con contenuto delle
combo box, etichette e altri elementi grafici.

    return
BaseController.extend("com.sap.build.standard.ticketManagement.controller.NewTicket
", {
        handleRouteMatched: function(oEvent) {

            var oParams = {};

            if (oEvent.mParameters.data.context) {
                this.sContext = oEvent.mParameters.data.context;
                var oPath;
                if (this.sContext) {
                    oPath = {
                        path: "/" + this.sContext,
                        parameters: oParams
                    };
                    this.getView().bindObject(oPath);
                }
            }

        },
        _onPageNavButtonPress: function(oEvent) {

            var oBindingContext = oEvent.getSource().getBindingContext();

            return new Promise(function(fnResolve) {

                this.doNavigate("HomeUser", oBindingContext, fnResolve,"");
            }.bind(this)).catch(function(err) {
                if (err !== undefined) {
                    MessageBox.error(err.message);
                }
            });
        },
        doNavigate: function(sRouteName, oBindingContext, fnPromiseResolve,
sViaRelation) {
```

```
            var sPath = (oBindingContext) ? oBindingContext.getPath() : null;
            var oModel = (oBindingContext) ? oBindingContext.getModel() : null;

            var sEntityNameSet;
            if (sPath !== null && sPath !== "") {
                if (sPath.substring(0, 1) === "/") {
                    sPath = sPath.substring(1);
                }
                sEntityNameSet = sPath.split("(")[0];
            }
            var sNavigationPropertyName;
            var sMasterContext = this.sMasterContext ? this.sMasterContext : sPath;

            if (sEntityNameSet !== null) {
                sNavigationPropertyName = sViaRelation ||
this.getOwnerComponent().getNavigationPropertyForNavigationWithContext(sEntityNameS
et,
                    sRouteName);
            }
            if (sNavigationPropertyName !== null && sNavigationPropertyName !==
undefined) {
                if (sNavigationPropertyName === "") {
                    this.oRouter.navTo(sRouteName, {
                        context: sPath,
                        masterContext: sMasterContext
                    }, false);
                } else {
                    oModel.createBindingContext(sNavigationPropertyName,
oBindingContext, null, function(bindingContext) {
                        if (bindingContext) {
                            sPath = bindingContext.getPath();
                            if (sPath.substring(0, 1) === "/") {
                                sPath = sPath.substring(1);
                            }
                        } else {
                            sPath = "undefined";
                        }
                        if (sPath === "undefined") {
                            this.oRouter.navTo(sRouteName);
                        } else {
                            this.oRouter.navTo(sRouteName, {
                                context: sPath,
                                masterContext: sMasterContext
                            }, false);
                        }
                    }.bind(this));
                }
            } else {
                this.oRouter.navTo(sRouteName);
            }

            if (typeof fnPromiseResolve === "function") {
                fnPromiseResolve();
            }
        },
```

```
        onChange : function(oEvent) { //function per upload media scatenata
dall'evento onChange sul bottone Invia
            var oUploadCollection = oEvent.getSource();
            var oCustomerHeaderToken = new UploadCollectionParameter({
                name : "x-csrf-token",
                value : "securityTokenFromModel"
            });
            oUploadCollection.addHeaderParameter(oCustomerHeaderToken);
            MessageToast.show("File aggiunto correttamente");
        },

        onFilenameLengthExceed : function(oEvent) {
            MessageToast.show("Il nome attribuito al file è troppo lungo (max 60
caratteri)");
        },

        onTypeMissmatch : function(oEvent) {
            MessageToast.show("Il formato file inserito non è supportato");
        },

// Upload del ticket appena inserito dall'utente
        onStartUpload : function(oEvent) {

            var data = this.getView().getModel().getData();

                var oUploadCollection = this.getView().byId("UploadCollection");
                var cFiles = oUploadCollection.getItems().length;
                var titolo = sap.ui.getCore().byId("__input3");
                var owner = sap.ui.getCore().byId("__input4");
                var categoria = sap.ui.getCore().byId("__box2");
                var priorita = sap.ui.getCore().byId("__box3");
                var descrizione = sap.ui.getCore().byId("__area0");
                data.TicketCollection.push({"ID" : "",
                                            "Titolo": titolo,
                                            "Categoria": categoria,
                                            "Descrizione": descrizione,
                                            "Priorita": priorita,
                                            "Owner": owner
                });
                oUploadCollection.upload();
        },

        onBeforeUploadStarts : function(oEvent) {
//Azioni preliminari upload del media innescate dall'evento onBeforeUploadStarts
            var oCustomerHeaderSlug = new sap.m.UploadCollectionParameter({
                name : "slug",
                value : oEvent.getParameter("fileName")
            });
            oEvent.getParameters().addHeaderParameter(oCustomerHeaderSlug);
            setTimeout(function() {
                MessageToast.show("Upload ticket in corso");
            }, 4000);
        },
        onUploadComplete : function(oEvent) {
//Azioni successive all'upload scatenate dall'evento onUploadComplete
            var sUploadedFileName = oEvent.getParameter("files")[0].fileName;
            setTimeout(function() {
```

```
                var oUploadCollection = this.getView().byId("UploadCollection");

                for (var i = 0; i < oUploadCollection.getItems().length; i++) {
                    if (oUploadCollection.getItems()[i].getFileName() ===
sUploadedFileName) {

oUploadCollection.removeItem(oUploadCollection.getItems()[i]);
                        break;
                    }
                }

                MessageToast.show("Upload ticket completato");
            }.bind(this), 8000);

            var oBindingContext = oEvent.getSource().getBindingContext();
            return new Promise(function(fnResolve) {
            this.doNavigate("HomeUser", oBindingContext, fnResolve, "");
            }.bind(this)).catch(function(err) {
                if (err !== undefined) {
                    MessageBox.error(err.message);
                }
            });
        },
/......................./
        _onButtonPress6: function(oEvent) {
//Funzione binding view-model finale, dopo l'inserimento, il model locale
TicketSet.json viene aggiornato con l'aggiunta di un costrutto contenente le info
del nuovo ticket, la API asincrona si occuperà in un secondo momento di leggere il
JSON e scatenare una DB insert.
            oEvent = jQuery.extend(true, {}, oEvent);
            return new Promise(function(fnResolve) {
                    fnResolve(true);
                })
                .then(function(result) {
                    return new Promise(function(fnResolve) {
                        var aChangedEntitiesPath, oChangedBindingContext;
                        var oModel = this.oModel;
                        if (oModel && oModel.hasPendingChanges()) {
                            aChangedEntitiesPath =
Object.keys(oModel.mChangedEntities);

                            for (var j = 0; j < aChangedEntitiesPath.length; j++) {
                                oChangedBindingContext = oModel.getContext("/" +
aChangedEntitiesPath[j]);
                                if (oChangedBindingContext &&
oChangedBindingContext.bCreated) {

oModel.deleteCreatedEntry(oChangedBindingContext);
                                }
                            }
                            oModel.resetChanges();
                        }
                        fnResolve();
                    }.bind(this));

                }.bind(this))
                .then(function(result) {
```

```
                    if (result === false) {
                        return false;
                    } else {

                        var oHistory = History.getInstance();
                        var sPreviousHash = oHistory.getPreviousHash();
                        var oQueryParams =
this.getQueryParameters(window.location);

                        if (sPreviousHash !== undefined ||
oQueryParams.navBackToLaunchpad) {
                            window.history.go(-1);
                        } else {
                            var oRouter =
sap.ui.core.UIComponent.getRouterFor(this);
                            oRouter.navTo("default", true);
                        }
                    }
                }.bind(this)).catch(function(err) {
                    if (err !== undefined) {
                        MessageBox.error(err.message);
                    }
                });
        },
        onInit: function() {
            this.mBindingOptions = {};
            this.oRouter = sap.ui.core.UIComponent.getRouterFor(this);

this.oRouter.getTarget("NewTicket").attachDisplay(jQuery.proxy(this.handleRouteMatc
hed, this));
            this.oModel = this.getOwnerComponent().getModel();
            var sPath =
jQuery.sap.getModulePath("com.sap.build.standard.ticketManagement.localService",
"/TicketSet.json");
            var oModel = new JSONModel(sPath);
            this.getView().setModel(oModel);
        }
    });
}, /* bExport= */ true);
```

# Appendice B

### ***Dettaglio ticket e Chat con utente esperto***

```
sap.ui.define(["sap/ui/core/mvc/Controller",
    "sap/m/MessageBox",
    "./utilities",
    "sap/ui/core/routing/History",
    "sap/ui/model/json/JSONModel",
    "sap/m/MessageToast",
    "sap/ui/core/format/DateFormat"
], function(BaseController, MessageBox, Utilities, History, JSONModel,
MessageToast, DateFormat) {
    "use strict";

    return
BaseController.extend("com.sap.build.standard.ticketManagement.controller.TicketDet
ail", {
        onInit: function() {
            this.mBindingOptions = {};
            this.oRouter = sap.ui.core.UIComponent.getRouterFor(this);

this.oRouter.getTarget("TicketDetail").attachDisplay(jQuery.proxy(this.handleRouteM
atched, this));
            this.oModel = this.getOwnerComponent().getModel();
            // set model
            var sPath =
jQuery.sap.getModulePath("com.sap.build.standard.ticketManagement.localService",
"/TicketSet.json");
            var oModel = new JSONModel(sPath);
            this.getView().setModel(oModel);

            // set the possible screen sizes
            var oCarouselContainer = {
                screenSizes : [
                    "350px",
                    "420px",
                    "555px",
                    "743px",
                    "908px"
                ]
            };
            var oScreenSizesModel = new JSONModel(oCarouselContainer);
            this.getView().setModel(oScreenSizesModel, "ScreenSizesModel");
        },

        onPost: function (oEvent) {
            var oFormat = DateFormat.getDateTimeInstance({style: "medium"});
            var oDate = new Date();
            var sDate = oFormat.format(oDate);
            var sUser = getUser();
            // create new entry
            var sValue = oEvent.getParameter("value");
```

```
            var oEntry = {
                Author : sUser,
                AuthorPicUrl : "",
                Type : "Reply",
                Date : "" + sDate,
                Text : sValue,
                ID : sap.n.oTicketId
            };
            // update model
            var oModel = this.getView().getModel();
            var aEntries = oModel.getData().ChatCollection;
            var eEntries = oModel.getData().TicketCollection;
            aEntries.unshift(oEntry);
            oModel.setData({
                ChatCollection : aEntries,
                TicketCollection : eEntries
            });
            this.oModel.refresh();

            //Toast message invio
            MessageToast.show("Messaggio inviato");
        },

        /.............................../

        handleRouteMatched: function(oEvent) {
            var oParams = {};
            if (oEvent.mParameters.data.context) {
                this.sContext = oEvent.mParameters.data.context;
                var oPath;
                if (this.sContext) {
                    oPath = {
                        path: "/" + this.sContext,
                        parameters: oParams
                    };
                    this.getView().bindObject(oPath);
                }
            }
            this.onFilterTicketDetail();
            this.onFilterTicketScreenshot();
            this.onFilterTicketScreenshot1();
            this.onFilterTicketChat();
        },

//Filtraggio dei dati del ticket nella pagina di dettaglio
        onFilterTicketDetail: function() {
            if (sap.n.oTicketId) {
//Filtra il ticket selezionato nella tabella e produce i risultati nella pagina di
dettaglio (media TAB dettaglio)
                var oTicketId = sap.n.oTicketId;
                var listDetailTicket = this.getView().byId("__component0---
TicketDetail--listDetailTicket");
                var oFilterByTicketId = new sap.ui.model.Filter("ID",
sap.ui.model.FilterOperator.EQ, oTicketId);
                listDetailTicket.getBinding("items").filter(oFilterByTicketId);
            }
        },
```

```
//slider per lo zoom degli screenshot nella pagina di dettaglio
        onSliderMoved: function (oEvent) {
            var origingalHeight = 650;
            var screenSizesJSON =
this.getView().getModel("ScreenSizesModel").getData();
            var iValue = oEvent.getParameter("value");
            var screenWidth = screenSizesJSON.screenSizes[Number(iValue) - 1];
            var oCarouselContainer = this.getView().byId("screenshot1");
            oCarouselContainer.setWidth(screenWidth);
            var screenHeight = origingalHeight * parseFloat(screenWidth) / 1000;
            oCarouselContainer.setHeight(screenHeight + 'px');
        },

        /........................../

//Filtra il ticket selezionato nella tabella e produce i risultati nella pagina di
dettaglio (media TAB chat)
        onFilterTicketScreenshot1: function () {
            if (sap.n.oTicketId) {
                var oTicketId = sap.n.oTicketId;
                var listDetailTicket = this.getView().byId("__component0---
TicketDetail--screenshot1");
                var oFilterByTicketId = new sap.ui.model.Filter("ID",
sap.ui.model.FilterOperator.EQ, oTicketId);
                listDetailTicket.getBinding("pages").filter(oFilterByTicketId);
            }
        },

//Return back nella pagina ListaTicket
        _onPageNavButtonPress: function() {
            var oHistory = History.getInstance();
            var sPreviousHash = oHistory.getPreviousHash();
            var oQueryParams = this.getQueryParameters(window.location);

            if (sPreviousHash !== undefined || oQueryParams.navBackToLaunchpad) {
                window.history.go(-1);
            } else {
                var oRouter = sap.ui.core.UIComponent.getRouterFor(this);
                oRouter.navTo("default", true);
            }
        },

        getQueryParameters: function(oLocation) {
            var oQuery = {};
            var aParams = oLocation.search.substring(1).split("&");
            for (var i = 0; i < aParams.length; i++) {
                var aPair = aParams[i].split("=");
                oQuery[aPair[0]] = decodeURIComponent(aPair[1]);
            }
            return oQuery;

        },
//Funzione invio messaggio nella chat dedicata
        _onFeedInputPost: function() {
            return new Promise(function(fnResolve) {
                var sTargetPos = "default";
```

```
                sTargetPos = (sTargetPos === "default") ? undefined : sTargetPos;
                sap.m.MessageToast.show("Messaggio inviato", {
                    onClose: fnResolve,
                    duration: 3000 || 3000,
                    at: sTargetPos,
                    my: sTargetPos
                });
            }).catch(function(err) {
                if (err !== undefined) {
                    MessageBox.error(err.message);
                }
            });

        },
//Funzione per il feedback dell'utente attraverso un rating idicator
//vautazione del servizio offerto dall'utente esperto
        _onRatingIndicatorChange: function() {
            return new Promise(function(fnResolve) {
                var sTargetPos = "default";
                sTargetPos = (sTargetPos === "default") ? undefined : sTargetPos;
                sap.m.MessageToast.show("Hai valutato il servizio", {
                    onClose: fnResolve,
                    duration: 3000 || 3000,
                    at: sTargetPos,
                    my: sTargetPos
                });
            }).catch(function(err) {
                if (err !== undefined) {
                    MessageBox.error(err.message);
                }
            });

        },
//Funzione di binding model-view simile a quella presentata nel controller
precedente
        _onButtonPress3: function(oEvent) {
            oEvent = jQuery.extend(true, {}, oEvent);
            return new Promise(function(fnResolve) {
                    fnResolve(true);
                })
                .then(function(result) {

                var oBindingContext = oEvent.getSource().getBindingContext();

                    return new Promise(function(fnResolve) {

                        this.doNavigate("ListaTicket", oBindingContext, fnResolve,
"");
        /......................./

        },

        /......................./

//Funzione scatenata dall'evento onclick del bottone "segnato come verificato"
//Controlla che il ticket sia nello stato "da verificare"
        _onButtonPress16: function() {
```

```
            var data = this.getView().getModel().getData();
            for (var i=0; i<=data.TicketCollection.length; i++) {
                var Record =
this.getView().getModel().getData("ID").TicketCollection[i];
                if (data.TicketCollection[i] === Record){
                    if (status!=="da verificare") {
                        var oCompact =
!!this.getView().$().closest(".sapUiSizeCompact").length;
                            MessageBox.warning(
                                "Il seguente ticket non è in stato: 'da
verificare'!!",
                                {
                                    styleClass: oCompact ? "sapUiSizeCompact" : ""
                                }
                            );
                        break;
                    } else if (status==="da verificare") {

                        var cCompact =
!!this.getView().$().closest(".sapUiSizeCompact").length;
                            MessageBox.confirm(
                                "Il ticket sarà contrassegnato come verificato",
                                {
                                    styleClass: cCompact ? "sapUiSizeCompact" : ""
                                }
                            );
                        break;
                    }
                }
            }
        });
}, /* bExport= */ true);
```

# Appendice C

## ***Definizione di filtri-ordinamenti-raggruppamenti personalizzati***

```
//Dichiarato oggetto che contiene le regole per la definizione dei filtri custom, e
gli oggetti per l'ordinamento custom
            this._data = {
                    CustomFilterCollection : [
                                            {"Nome" : ['ID', 'Titolo', 'Descrizione',
'Stato', 'Priorita', 'Owner', 'Categoria', 'Assigned_to'],
                                             "Tipo": ["COMPRESO TRA", "UGUALE",
"DIVERSO DA", "CONTIENE"]}
                                        ],

                    OrdinaCollection : [{ID:"1", Nome:"ID"},
                                        {Nome:"Titolo", ID:"2"},
                                        {Nome:"Descrizione", ID:"3"},
                                        {ID:"4", Nome:"Stato"},
                                        {ID:"5", Nome:"Priorita"},
                                        {Nome:"Owner", ID:"6"},
                                        {Nome:"Assigned_to", ID:"7"},
                                        {Nome:"Categoria", ID:"8"}
                                        ],
            this.jModel = new sap.ui.model.json.JSONModel();
            this.jModel.setData(this._data);
        },

        onBeforeRendering: function() {
            this.byId('CustomFilter').setModel(this.jModel);
            this.byId('OrdinaCampi').setModel(this.jModel);
            this.byId('ordinatiCampi').setModel(this.jModel);
            this.byId('FiltriSalvati').setModel(this.jModel);
        },

        fetchRecords : function(){
            //data will be in this._data.Custom filter
            console.log(this._data.CustomFilterCollection);
            console.log(this._data.OrdinaCollection);
            console.log(this._data.OrdinatiCollection);
            console.log(this._data.FiltriSalvatiCollection);
        },

//Funzione che edita in tempo reale l'espressione matematica del filtro custom
definito
        handleLiveChange: function() {

            for (var i=0; i<=this._data.CustomFilterCollection.length ;i++) { //for
che cicla finchè l'utente continua a definire nuove righe di filtro custom

                var Combo1 = sap.ui.getCore().byId("__component0---ListaTicket--
combo1-__component0---ListaTicket--CustomFilter-" + i).getValue();
                var Combo2 = sap.ui.getCore().byId("__component0---ListaTicket--
combo2-__component0---ListaTicket--CustomFilter-" + i).getValue();
```

```
                var input = sap.ui.getCore().byId("__input0-__component0---
ListaTicket--CustomFilter-" + i).getValue();
                var input2 = sap.ui.getCore().byId("__input1-__component0---
ListaTicket--CustomFilter-" + i).getValue();
                this.getView().byId('getValue' + i).setText(Combo1);

                switch (Combo2) {
                    //4 possibilità di definizione di un operatore di confronto
                    case "COMPRESO TRA":
                        sap.ui.getCore().byId("__input0-__component0---ListaTicket-
-CustomFilter-" + i).setPlaceholder("Da");
                        sap.ui.getCore().byId("__input1-__component0---ListaTicket-
-CustomFilter-" + i).setEnabled(true).setPlaceholder("A");
                        this.getView().byId('getValue' + i).setText("[(" + Combo1 +
" > " + input + ")" + " AND " + "(" + Combo1 + " < " + input2 + ")]");
                        break;

                    case "UGUALE":
                        sap.ui.getCore().byId("__input0-__component0---ListaTicket-
-CustomFilter-" + i).setPlaceholder("Uguale a");
                        sap.ui.getCore().byId("__input1-__component0---ListaTicket-
-CustomFilter-" + i).setEnabled(false);
                        this.getView().byId('getValue' + i).setText("(" + Combo1 +
" = " + input + ")");
                        break;

                    case "DIVERSO DA":
                        sap.ui.getCore().byId("__input0-__component0---ListaTicket-
-CustomFilter-" + i).setPlaceholder("Diverso da");
                        sap.ui.getCore().byId("__input1-__component0---ListaTicket-
-CustomFilter-" + i).setEnabled(false);
                        this.getView().byId('getValue' + i).setText("(" + Combo1 +
" /= " + input + ")");
                        break;

                    case "CONTIENE":
                        sap.ui.getCore().byId("__input0-__component0---ListaTicket-
-CustomFilter-" + i).setPlaceholder("Contiene");
                        sap.ui.getCore().byId("__input1-__component0---ListaTicket-
-CustomFilter-" + i).setEnabled(false);
                        this.getView().byId('getValue' + i).setText("(" + input + "
∈ " + Combo1 + ")");
                        break;

                    default:
                        sap.ui.getCore().byId("__input0-__component0---ListaTicket-
-CustomFilter-" + i).setPlaceholder("Inserisci valore");
                        sap.ui.getCore().byId("__input1-__component0---ListaTicket-
-CustomFilter-" + i).setEnabled(false);
                        break;
                }
            }
        },

//Inserisce una nuova riga all'interno dei filtri custom per stabilire una nuova
regola
        addRowAND : function(oArg){
```

```
            var andRecord = oArg.getSource().getBindingContext().getObject();
            this._data.CustomFilterCollection.push({"Nome" : ['ID', 'Titolo',
'Descrizione', 'Stato', 'Priorita', 'Owner', 'Categoria', 'Assigned_to'],
                                                    "Tipo": ["COMPRESO TRA",
"UGUALE", "DIVERSO DA", "CONTIENE"]});

            this.jModel.refresh();  //Al refresh del model  viene visualizzato il
nuovo record in tabella

            for(var i=0 ; i<this._data.CustomFilterCollection.length ; i++){ //
Viene inserito in append all'espressione matematica di filtro, il connettivo logico
AND
                if (this._data.CustomFilterCollection[i] === andRecord){
                    this.getView().byId('conn' + i).setText(" AND ");
                    break;
                }
            }
        },

                deleteRow : function(oArg, oEvent){

//Al press del tasto di cancellazione, elimina una riga per la dichiarazione di un
filtro custom
            var deleteRecord = oArg.getSource().getBindingContext().getObject();

            for (var i=0; i<=this._data.CustomFilterCollection.length; i++){

                if (this._data.CustomFilterCollection[i] === deleteRecord && i>0) {

                    this._data.CustomFilterCollection.splice(i, 1);
                    this.getView().byId('getValue' + i).setText(); //cancella
l'elemento eliminato dall'espressione matematica
                    this.getView().byId('conn' + i).setText();  //cancella il
connettivo logico corrispondente
                    this.jModel.refresh();
                    break; //esce dal loop

                } else if (this._data.CustomFilterCollection[i] === deleteRecord &&
i===0) { //al press del pulsante delete sulla prima riga vengono azzerati i campi

//non viene effettuato lo splice della riga in questo caso
                    sap.ui.getCore().byId("__component0---ListaTicket--combo1-
__component0---ListaTicket--CustomFilter-0").setValue();
                    sap.ui.getCore().byId("__component0---ListaTicket--combo2-
__component0---ListaTicket--CustomFilter-0").setValue();
                    sap.ui.getCore().byId("__input0-__component0---ListaTicket--
CustomFilter-0").setValue();
                    sap.ui.getCore().byId("__input1-__component0---ListaTicket--
CustomFilter-0").setValue();
                    this.getView().byId('getValue0').setText();
                    this.getView().byId('conn' + i).setText();
                    this.jModel.refresh();
                    break;
                }
            }
        },
```

```
//Questa funzione, associata al tasto refresh, elimina tutti i campi che si erano
selezionati per l'ordinamento custom
        onRefresh: function () {

            for (var i=0; i<=this._data.OrdinatiCollection.length; i++) {

                this._data.OrdinatiCollection.splice(i, 10);
                this.jModel.refresh();
                break;
            }
            for (var j=0; j<=this._data.OrdinaCollection.length; j++) {

                sap.ui.getCore().byId("__component0---ListaTicket--OrdinaCampi-
sa").setEnabled(true);
                sap.ui.getCore().byId("__item6-__clone" + j + "-
selectMulti").setSelected(false).setEnabled(true);
                sap.ui.getCore().byId("__item6-__clone" + j).setSelected(false);
                this.jModel.refresh();
                MessageToast.show("Ordinamento azzerato");
            }

        },

//Questa funzione muove nella tabella di sinistra i campi selezionati dall'utente
per la definizione dell'ordinamento custom
        onMoveLeft: function () {


            for (var i=0; i<=this._data.OrdinaCollection.length; i++) {

                var checked = sap.ui.getCore().byId("__item6-__clone" + i + "-
selectMulti").getSelected();
                var campo = sap.ui.getCore().byId("__identifier1-__clone" +
i).getText();
                var selId = sap.ui.getCore().byId("__identifier1-__clone" +
i).getId();

                while (checked === true) {

                        this._data.OrdinatiCollection.push({ID: selId, Nome:
campo});
                        sap.ui.getCore().byId("__item6-__clone" + i + "-
selectMulti").setEnabled(false);
                        sap.ui.getCore().byId("__item6-__clone" +
i).setSelected(false);
                        sap.ui.getCore().byId("__component0---ListaTicket--
OrdinaCampi-sa").setEnabled(false);
                        this.jModel.refresh();
                        break;
                }
                MessageToast.show("Definisci il tipo di ordinamento per i campi
selezionati");

            }
        },

//Apertura confirm box e salvataggio del custom filter appena creato
```

```
        _onButtonPress5: function(oEvent) {

            for (var i=0; i<=this._data.FiltriSalvatiCollection.length; i++) {
                var preferito = sap.ui.getCore().byId("__box1").getSelected();
                var NomeFiltro = sap.ui.getCore().byId("__input2").getValue();

                if (preferito === true && NomeFiltro!=="") {

                    this._data.FiltriSalvatiCollection.push({Nome: NomeFiltro,
ID:i});
                    this.jModel.refresh();
                    break;
                }
                if (NomeFiltro=== "") {
                    var bCompact =
!!this.getView().$().closest(".sapUiSizeCompact").length;
                    MessageBox.warning(
                        "Non hai specificato nessun nome per il tuo filtro custom",
                        {
                            styleClass: bCompact ? "sapUiSizeCompact" : ""
                        }
                    );
                return;
                }
            }

            this.ToggleSecondaryContent();
            oEvent = jQuery.extend(true, {}, oEvent);
            return new Promise(function(fnResolve) {
                    fnResolve(true);
                })
                .then(function(result) {
                    var oView = this.getView();
                    var oController = this;

                    return new Promise(function(fnResolve, fnReject) {
                        var oModel = oController.oModel;

                        var fnResetChangesAndReject = function(sMessage) {
                            oModel.resetChanges();
                            fnReject(new Error(sMessage));
                        };
                        if (oModel && oModel.hasPendingChanges()) {
                            oModel.submitChanges({
                                success: function(oResponse) {
                                    var oBatchResponse =
oResponse.__batchResponses[0];
                                    var oChangeResponse =
oBatchResponse.__changeResponses && oBatchResponse.__changeResponses[0];
                                    if (oChangeResponse && oChangeResponse.data) {
                                        var sNewContext =
oModel.getKey(oChangeResponse.data);
                                        oView.unbindObject();
                                        oView.bindObject({
                                            path: "/" + sNewContext
                                        });
```

```
                                    if (window.history &&
window.history.replaceState) {
                                        window.history.replaceState(undefined,
undefined, window.location.hash.replace(encodeURIComponent(oController.sContext),
encodeURIComponent(sNewContext)));
                                    }
                                    oModel.refresh();
                                    fnResolve();
                                } else if (oChangeResponse &&
oChangeResponse.response) {

fnResetChangesAndReject(oChangeResponse.message);
                                } else if (!oChangeResponse &&
oBatchResponse.response) {

fnResetChangesAndReject(oBatchResponse.message);
                                } else {
                                    oModel.refresh();
                                    fnResolve();
                                }
                            },
                            error: function(oError) {
                                fnReject(new Error(oError.message));
                            }
                        });
                    } else {
                        fnResolve();
                    }
                });

            }.bind(this))
            .then(function(result) {
                if (result === false) {
                    return false;
                } else {
                    return new Promise(function(fnResolve) {
                        sap.m.MessageBox.confirm("Le regole custom definite
saranno applicate alla lista ticket", {
                            title: "Filtro Custom",
                            actions: ["OK", "Annulla"],
                            onClose: function(sActionClicked) {
                                fnResolve(sActionClicked === "OK");
                            }
                        });
                    });

                }
            }.bind(this)).catch(function(err) {
                if (err !== undefined) {
                    MessageBox.error(err.message);
                }
            });

        },
```

# Appendice D

### ***Ricerca ticket e cancellazione filtri***

```
//Funzione che gestisce la barra di ricerca dei ticket
        _onSearchFieldLiveChange2: function(oEvent) {
            var sControlId =
                "sap_Responsive_Page_0-content-sap_ui_layout_BlockLayout-
1515407526987-content-sap_ui_layout_BlockLayoutRow-2-content-
sap_ui_layout_BlockLayoutCell-1-content-build_simple_Table-1515407548335";
            var oControl = this.getView().byId(sControlId);

            // Get the search query, regardless of the triggered event ('query' for
the search event, 'newValue' for the liveChange one).
            Var sQuery = oEvent.getParameter("query") ||
oEvent.getParameter("newValue");
            var sSourceId = oEvent.getSource().getId();

            return new Promise(function(fnResolve) {
                var aFinalFilters = [];

                var aFilters = [];
                // 1) Search filters (with OR)
                if (sQuery && sQuery.length > 0) {

                    aFilters.push(new sap.ui.model.Filter("ID",
sap.ui.model.FilterOperator.Contains, sQuery));

                    aFilters.push(new sap.ui.model.Filter("Titolo",
sap.ui.model.FilterOperator.Contains, sQuery));

                    aFilters.push(new sap.ui.model.Filter("Priorita",
sap.ui.model.FilterOperator.Contains, sQuery));

                    aFilters.push(new sap.ui.model.Filter("Stato",
sap.ui.model.FilterOperator.Contains, sQuery));

                    aFilters.push(new sap.ui.model.Filter("Owner",
sap.ui.model.FilterOperator.Contains, sQuery));

                    aFilters.push(new sap.ui.model.Filter("Assigned_to",
sap.ui.model.FilterOperator.Contains, sQuery));

                }

                var aFinalFilters = aFilters.length > 0 ? [new
sap.ui.model.Filter(aFilters, false)] : [];
                var oBindingOptions = this.updateBindingOptions(sControlId, {
                    filters: aFinalFilters
                }, sSourceId);
                var oBindingInfo = oControl.getBindingInfo("items");
                oControl.bindAggregation("items", {
                    model: oBindingInfo.model,
```

```
                    path: oBindingInfo.path,
                    parameters: oBindingInfo.parameters,
                    template: oBindingInfo.template,
                    sorter: oBindingOptions.sorters,
                    filters: oBindingOptions.filters
                });
            }.bind(this)).catch(function(err) {
                if (err !== undefined) {
                    MessageBox.error(err.message);
                }
            });

        },
        updateBindingOptions: function(sCollectionId, oBindingData, sSourceId) {
            this.mBindingOptions[sCollectionId] =
this.mBindingOptions[sCollectionId] || {};

            var aSorters = oBindingData.sorters === undefined ?
this.mBindingOptions[sCollectionId].sorters : oBindingData.sorters;
            var oGroupby = oBindingData.groupby === undefined ?
this.mBindingOptions[sCollectionId].groupby : oBindingData.groupby;

            // 1) Update the filters map for the given collection and source
            this.mBindingOptions[sCollectionId].sorters = aSorters;
            this.mBindingOptions[sCollectionId].groupby = oGroupby;
            this.mBindingOptions[sCollectionId].filters =
this.mBindingOptions[sCollectionId].filters || {};
            this.mBindingOptions[sCollectionId].filters[sSourceId] =
oBindingData.filters || [];

            // 2) Reapply all the filters and sorters
            var aFilters = [];
            for (var key in this.mBindingOptions[sCollectionId].filters) {
                aFilters =
aFilters.concat(this.mBindingOptions[sCollectionId].filters[key]);
            }

            // Add the groupby first in the sorters array
            if (oGroupby) {
                aSorters = aSorters ? [oGroupby].concat(aSorters) : [oGroupby];
            }

            var aFinalFilters = aFilters.length > 0 ? [new
sap.ui.model.Filter(aFilters, true)] : undefined;
            return {
                filters: aFinalFilters,
                sorters: aSorters
            };

        },

//Funzione associata al bottone clear (pulisce tutti i filtri, gli ordinamenti e i
raggruppamenti inseriti)
        _onButtonPress13: function(oEvent) {

            //Clear all sortings and grouping rules
```

```
            var oTable = this.getView().byId("sap_Responsive_Page_0-content-
sap_ui_layout_BlockLayout-1515407526987-content-sap_ui_layout_BlockLayoutRow-2-
content-sap_ui_layout_BlockLayoutCell-1-content-build_simple_Table-1515407548335");
            oTable.getBinding("items").sort(null);
            oTable.getBinding("items").filter(null);


        },
```

# Ringraziamenti

Non è facile per me scrivere in poche righe quel che credo di dimostrare tutti i giorni con i comportamenti. Non sono mai stato molto bravo con le parole, però, credo fortemente di dover ringraziare tutti coloro che mi sono stati vicino in questi anni, nel mio percorso universitario, e al di fuori, perché d'altronde lo studente non smette di essere tale nel momento in cui esce dal luogo in cui studia, come un lavoratore che tiene al suo impiego, non smette di essere tale quando esce dal luogo di lavoro.

Lo studio è oramai parte integrante della mia vita, quando iniziai questo percorso qualcuno mi disse che l'ingegnere non smette mai di aggiornare le proprie conoscenze, perché questa professione insegna non una specifica nozione, ma un metodo, un modo di approcciarsi alle difficoltà ed alle sfide che il mondo lavorativo ci presenta e ci presenterà ogni giorno.

Pertanto, voglio ringraziare tutti i professori, ognuno di loro mi ha dato dei diversi punti di vista con i quali osservare le cose ed ognuno di loro mi ha trasmesso delle competenze per risolvere problemi.

Vorrei ringraziare la mia famiglia, che non ha mai dubitato di me, nemmeno quando l'attività di studio si è fatta improvvisamente così pesante, perché ho tentato, riuscendoci, di intraprendere al contempo, una attività lavorativa che potesse darmi soddisfazioni e che potesse attribuire un senso a tutto quel che sto studiando sui libri.

*"Last but not the least"* la persona più importante della mia vita, il mio faro, mia Moglie, che mi ha supportato e sopportato in tutte le scelte, e che contribuisce a far divenire ogni giorno migliore di come sarebbe senza di lei.