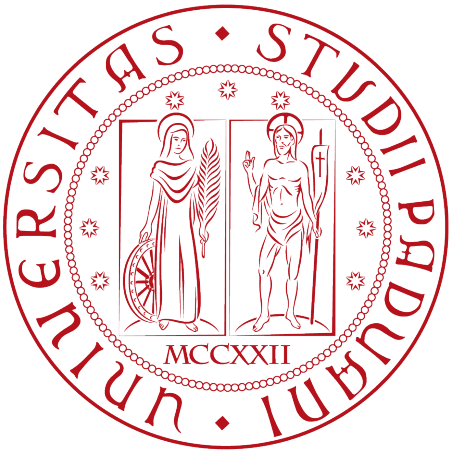
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Progettazione e sviluppo di un'interfaccia grafica per un sistema di raccomandazione di istanze di processi aziendali

Tesi di laurea

Relatore

Prof. Nicolò Navarin

Laureando

Michele Gatto

Anno Accademico 2022-2023

Sommario

Il presente documento descrive il lavoro svolto da Michele Gatto durante lo stage interno all'Università di Padova. L'obiettivo era la progettazione e lo sviluppo di un'interfaccia grafica per un sistema di raccomandazione di istanze di processi di business. Il sistema implementa l'analisi prescrittiva e ha lo scopo di suggerire le migliori opzioni di decisione nell'ambito dei processi aziendali. Per fare questo vengono sfruttati, in particolare, i risultati dell'analisi predittiva, algoritmi di ottimizzazione e intelligenza artificiale. L'interfaccia grafica, sviluppata attraverso il framework Python Dash, si occupa di tutte le fasi necessarie per il funzionamento del sistema: dal caricamento dei file dati da analizzare, al caricamento del file dati attuali su cui applicare l'analisi prescrittiva fino alla visualizzazione dei risultati finali.
All'interno del documento viene descritta l'organizzazione del suddetto stage e vengono ripercorse le fasi di analisi, progettazione e sviluppo dell'interfaccia, evidenziando e motivando le scelte stilistiche ed architetturali effettuate.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 Concetti chiave

1.1.1 Processi di business e process mining

"Il **processo aziendale** (o business process) è un insieme di attività interrelate, svolte all'interno dell'azienda nell'ambito della gestione operativa delle sue funzioni aziendali, che creano valore trasformando delle risorse (input del processo) in un prodotto finale (output del processo) a valore aggiunto, destinato ad un soggetto interno o esterno all'azienda (cliente)" [21].
Un processo è quindi caratterizzato da una sequenza di eventi, ad ognuno dei quali è associata un'attività, le sequenze di eventi sono denominate "tracce". É importante notare come le tracce siano spesso standardizzate, di conseguenza si avranno, per la maggior parte, esecuzioni simili.
A loro volta i processi di business sono la fonte di informazione per il process mining.
"Il **process mining** è una tecnica di process management, che permette l'analisi dei processi di business basati sui log degli eventi." [22]
Il process mining quindi offre varie tecniche per estrarre informazioni utili dai log degli eventi.

Log degli eventi

In generale un sistema informativo (che può essere un database o un ERP[G]) genera dati relativi agli eventi avvenuti in forma di log, tipicamente per registrare e tracciare i cambiamenti avvenuti allo stato del sistema. Rappresenta infatti l'impronta digitale elettronica delle varie operazioni aziendali. Spesso questi dati sono organizzati in forma tabulare e prendono la forma di un log di eventi. Un log di eventi (o *event log*) quindi, è una collezione di dati relativi ad eventi registrata e tracciata da un sistema informativo e conservata in file strutturati (per esempio CSV, XES, ecc.). Si parla di **traccia** quando ci si riferisce ad un insieme di eventi con lo stesso case id, quindi un event log può essere visto anche come una collezione di tracce.
Va osservato che, in generale, un log non è mai direttamente utilizzabile, può infatti contenere gap, inconsistenze e ripetizioni. Si ritiene quindi necessario delle operazioni di preparazione, che permettono di filtrare e pulire i dati.

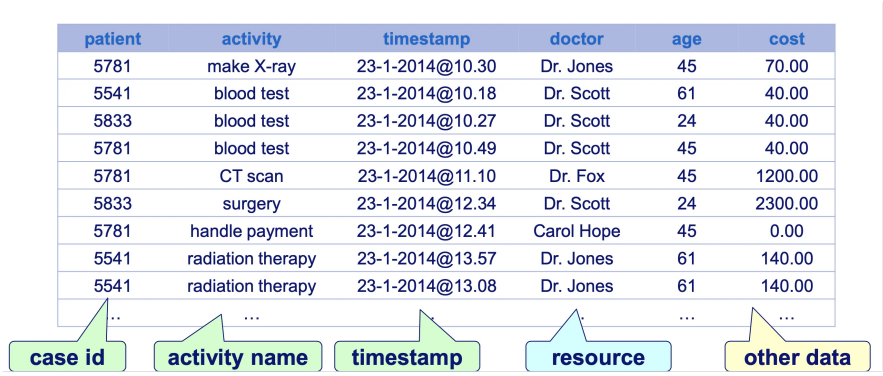


Figura 1.1: Esempio di *event log* [11]

Un evento (osservando la figura 1.1) è rappresentato da una singola "riga" dell'event log, e contiene informazioni relative a:

- il caso trattato (identificato dal *case id*);
- l'attività eseguita;
- il riferimento temporale (*timestamp*).

Importante notare come, i campi appena elencati, siano essenziali per descrivere un processo e necessari per rendere un event log utilizzabile. Un evento può contenere anche altre informazioni aggiuntive (ad esempio risorse, costi, ecc.), queste non sono strettamente necessarie per modellare un processo ma sono comunque utili per effettuare analisi su prospettive diverse.

Tecniche di process mining

Le tecniche di process mining si occupano principalmente di:

- *Process discovery*, con lo scopo di generare un *process model*, ovvero una rappresentazione grafica di un processo, permettendone per la visualizzazione, la documentazione e l'estrazione di informazioni;
- *Conformance checking*, che confronta il process model con un event log per individuare le differenze e deviazioni, può anche essere utilizzato per valutare la qualità del process model generato durante l'attività di discovery, oppure individuare colli di bottiglia;
- *Enhancement*, ovvero il miglioramento del process model attraverso l'introduzione di nuove informazioni (per esempio misure relative alle performance).

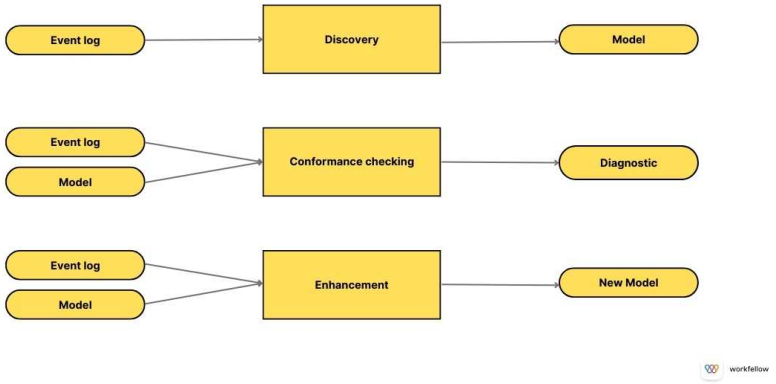


Figura 1.2: 3 tipi di attività nel process mining, con relativi input e output [13]

La creazione di un modello che descrive i processi rende l'individuazione di processi ripetitivi e inefficienti più semplice, permettendo di agire su di essi. Il modello a sua volta può continuamente essere affinato con informazioni aggiuntive che permettono di analizzare le performance e costruire workflow più efficienti. Lo stesso modello può poi essere comparato con event log successivi per individuare deviazioni, mostrando dove, quando e perché accadono.
Ovviamente, oltre a individuare i problemi e i punti deboli ottimizzabili, il process mining permette anche di scovarli in maniera precoce.
Riassumendo, l'applicazione delle tecniche offerte dal process mining offre vantaggi come: l'ottimizzazione del workflow, la riduzione dei costi, incremento della qualità di beni o servizi offerti e la crescita della soddisfazione del cliente.

1.1.2 Predictive Process Monitoring

Il *Predictive Process Monitoring* rientra tra le attività di Enhancement, ed è una branca del process mining che ha lo scopo di predire il futuro di una esecuzione in corso (e quindi ancora incompleta) di un processo [9].
Generalmente si è interessati a fare predizioni sul tempo totale di esecuzione, sulle attività successive o sui risultato finale del processo.
Questo permette di ottimizzare le operazioni di business, riducendo i costi e i rischi. Inoltre rende la pianificazione in materia di allocazione delle risorse disponibili più semplice ed efficiente. Importante è anche i benefici relativi alla soddisfazione del cliente: si possono fare predizioni relative alle aspettative dei clienti o prevedere il risultato finale di un processo, offrendo tempi di attesa più certi.
Come osservato in [9], in generale il Predictive Process Monitoring è svolto in due fasi:

- la fase di *training*: dove viene allenato un modello predittivo (tramite tecniche di Machine Learning) su dati storici di esecuzione, dove le tracce sono complete;
- la fase di *runtime*: dove il modello predittivo creato viene interrogato con lo scopo di ottenere predizioni relative a tracce incomplete e/o in corso di esecuzione.

1.1.3 Prescriptive Process Monitoring

Va osservata la differenze tra predizione, ovvero un'affermazione che spiega ciò che, probabilmente, avverrà (o meno) in futuro e una raccomandazione, ovvero un'affermazione che spiega quale sia la miglior azione da scegliere. Lo scopo finale del Predictive Process Monitoring è infatti, quello di offrire una predizione relativa all'ottenimento di un risultato positivo, ma non fornisce informazioni in relazione a quale sia l'azione successiva migliore da effettuare.
Il Prescriptive Process Monitoring si pone questo obiettivo, supportando e offrendo raccomandazioni agli stakeholder per prevenire risultati indesiderati o per determinare quali sono le migliori attività da eseguire per ottimizzare determinati aspetti di interesse [9].

1.1.4 Sistema di raccomandazione per istanze di processi aziendali

Un *process-aware recommender system*[14], o PAR, è una tipologia di sistema informativo che ha lo scopo di prevedere come le istanze di processi andranno ad evolversi e raccomandare le azioni necessarie per correggere le istanze con il rischio più alto di non raggiungere il risultato desiderato, definito in termini di *Key Performance Indicators* o KPI, ovvero un'insieme di misure quantificabili che un'azienda utilizza per valutare le sue prestazioni nel tempo (per esempio ricavi, soddisfazione del cliente, tempi di esecuzioni dei processi ecc.).
PAR (come descritto in [14]) è composto da due sotto-sistemi: il Predictive Process Monitoring che si occupa di generare le predizioni e il Prescriptive Process Monitoring che genera le raccomandazioni relative alle azioni correttive da applicare.
Sempre in [14] viene fatta un'importante osservazione: le raccomandazioni, se non correttamente spiegate, spingono l'utente a diffidare di esse, aumentando il rischio che queste non vengano applicate, dato che i responsabili preferiranno prendere decisioni soggettive non fidandosi del sistema.
Quindi la sola raccomandazione non basta, è importante che le ragioni che la determinano siano spiegate e rese comprensibili, così da far sentire l'utente partecipe nelle decisioni prese dal sistema e di conseguenza aumentare la fiducia in esso.
Le spiegazioni del sistema proposto in [14] si basano su caratteristiche relative al processo come il valore delle sue variabili, le attività effettuate e le risorse sfruttate per svolgere le attività. Il sistema utilizza i valori di Shapley (in inglese *Shapley Values*) per fornire le spiegazioni, concentrandosi solamente sugli aspetti il cui impatto negativo viene maggiormente mitigato della raccomandazione.

Shapley Values

Gli Shapley Values nascono nell'ambito della teoria dei giochi, e rappresentano un approccio per suddividere equamente la ricompensa tra i giocatori che hanno collaborato in un gioco cooperativo [14]. Servono quindi per quantificare la contribuzione che ogni giocatore ha portato al gioco stesso. In [14] viene spiegato come le caratteristiche di un istanza di processo corrispondano ai giocatori, e la differenza tra la predizione fatta dal modello predittivo e la predizione media è la ricompensa. Quindi il valore di Shapley di una caratteristica di una predizione rappresenta quanto essa contribuisce a variare la predizione del modello da quella media.

1.2 Struttura del resto della relazione

Il secondo capitolo descrive lo stage, gli scopi, l'organizzazione e gli obiettivi.

Il terzo capitolo descrive le varie tecnologie utilizzate, in particolare il framework, proponendo un'analisi comparativa tra i due framework considerati.

Il quarto capitolo approfondisce l'analisi dei requisiti del progetto.

Il quinto capitolo descrive le fasi di progettazione e sviluppo.

Il sesto capitolo viene discussa la fase di verifica e la validazione.

Il settimo capitolo trae le conclusioni dell'esperienza di stage.

Convenzioni tipografiche

Tutti i termini giudicati non di uso comune verranno approfonditi in un glossario situato alla fine del documento. Per ogni occorrenza di uno di questi termini verrà applicata la nomenclatura: parola^{G}.

Capitolo 2

Descrizione dello stage

2.1 Scopo dello stage

L'idea dello stage nasce dalla necessità di sviluppare una nuova interfaccia grafica front-end per un sistema di raccomandazione per le istanze di processi aziendali.
Il back-end del sistema è stato precedentemente sviluppato dal dottorando Alessandro Padella, come parte del suo lavoro di tesi di dottorato ([14]).
Il back-end è un PAR (vedasi sottosezione 1.1.4) che implementa l'analisi prescrittiva, e ha lo scopo di raccomandare le migliori opzioni di decisione nell'ambito dei processi aziendali. Per fare questo vengono sfruttati, in particolare, i risultati dell'analisi predittiva, algoritmi di ottimizzazione e intelligenza artificiale. Le opzioni vengono individuate in termini di KPI (tempo totale di esecuzione e numero di volte in cui una specifica attività si verifica). Infine il sistema genera anche le spiegazioni (sotto forma di Shapley Values) per le raccomandazioni individuate.
Lo scopo sarà quindi di sviluppare un'interfaccia grafica per il sistema, che si occupi di tutte le fasi necessarie:

- caricamento dell'event log storico su cui allenare il modello;
- selezione delle opzioni relative all'allenamento del modello desiderate dell'utente;
- caricamento dell'event log incompleto su cui fare le predizioni;
- visualizzazione delle predizioni, delle raccomandazioni e delle relative spiegazioni.

2.2 Contenuti formativi

Durante il progetto di stage sono state approfondite le conoscenze nei seguenti ambiti:

- Process mining;
- Progettazione e sviluppo di interfacce grafiche;
- Framework per la *data visualization*;

2.3 Obiettivi

Si farà riferimento agli obiettivi secondo le seguenti notazioni:

- O per i requisiti obbligatori, vincolanti in quanto obiettivo primario richiesto dal committente;
- D per i requisiti desiderabili, non vincolanti o strettamente necessari, ma dal riconoscibile valore aggiunto;
- F per i requisiti facoltativi, rappresentanti valore aggiunto non strettamente necessario.

Le sigle precedentemente indicate saranno seguite da un numero sequenziale, identificativo dell'obiettivo.
Lo stage prevedeva inizialmente i seguenti obiettivi:

- Obbligatori
  - *O01*: Produzione di un mock-up che illustra le diverse schermate che costituiscono l'interfaccia grafica;
  - *O02*: Produzione di diagrammi per i casi di uso considerati;
  - *O03*: Documento di almeno quattro pagine che riassume i diversi framework analizzati con relativi pro e contro di ognuno e discute il framework scelto, motivando la scelta;
  - *O04*: Definizione e utilizzo delle API[G] di back-end, con il supporto di diagrammi di classe e diagrammi di sequenza;
  - *O05*: Sviluppo della prima versione del codice per lo sviluppo dell'interfaccia grafica e per la comunicazione con le API[G] di back-end;
  - *O06*: Definizione ed esecuzione dei casi di test (di unità, di interfaccia, ecc.);
  - *O07*: Sviluppo della versione finale del codice per lo sviluppo dell'interfaccia grafica e per la comunicazione con le API[G] di back-end;
  - *O08*: Rilascio e deployment del codice finale dopo ultima validazione.
- Desiderabili
  - *D01*: Validazione delle API[G] di back-end.
- Facoltativi
  - *F01*: Stesura documentazione API[G].

2.4 Pianificazione del lavoro

Il periodo di svolgimento dello stage è stato dal 10 ottobre 2022 al 10 febbraio 2023, svolto in part-time, per la durata totale di 320 ore. Il part-time consisteva nello svolgere 4 ore di lavoro al giorno, per 5 giorni alla settimana, per un totale di 20 ore di lavoro settimanali. Infine sono state considerate le 2 settimane di ferie natalizie.
La pianificazione delle attività settimanali era la seguente:

Dalla prima alla quarta settimana (80 ore)

- Studio del problema;
- Sviluppo di mock-up e casi d'uso;

Dalla quinta alla settima settimana (60 ore)

- Ricerca e studio di framework grafici e visualizzazioni dati;

Dall'ottava alla nona settimana (40 ore)

- Studio e definizione di API[G] per il back-end;

Dalla decima alla dodicesima settimana (60 ore)

- Sviluppo di un primo prototipo;

Tredicesima Settimana (20 ore)

- Validazione del prototipo;

Dalla quattordicesima alla quindicesima settimana (40 ore)

- Sviluppo del prodotto finale (che elimina le criticità osservate nei test effettuati nella tredicesima settimana);

Sedicesima Settimana (20 ore)

- Validazione e collaudo finale;

2.5 Variazioni alla pianificazione originale

Durante lo svolgimento dello stage sono state applicate delle modifiche alla pianificazione e agli obiettivi da raggiungere. Tutto ciò in relazione al framework che è stato scelto per svolgere il progetto, che sarà approfondito nella sezione 3.2. Grazie ad esso l'API[G] per la comunicazione tra front-end e back-end si è resa non necessaria. In particolare l'attività pianificata nell'ottava e nella nona settimana è stata sostituita con la seguente attività: sviluppo di un primo prototipo;
Per quanto riguarda gli obiettivi sono stati modificati nel seguente modo:

- *O04* è stato sostituito con: Definizione architettura generale con il supporto di diagrammi di classe;
- *O05* e *O07* sono stati aggiornati, rimuovendo la parte relativa allo sviluppo delle API[G];
- *D01* è stato sostituito con: Implementazione funzionalità multiutente;
- *F01* è stato rimosso.

2.6 Organizzazione dello stage

Una volta ogni due settimane sono stati organizzati incontri diretti con il tutor esterno, il Prof. Massimiliano de Leoni, per verificare lo stato di avanzamento del progetto, chiarire eventualmente gli obiettivi o i dubbi sorti, affinare la ricerca e aggiornare il piano di lavoro. Il tutor esterno è stato coadiuvato dal Dott. Alessandro Padella, che si è reso disponibile con frequenza anche settimanale per dare supporto nello sviluppo del progetto.

Capitolo 3

Tecnologie utilizzate

3.1 Panoramica generale

Durante la realizzazione del progetto sono state utilizzate le seguenti tecnologie e strumenti:

Git

Git è un sistema di controllo di versione open source per la gestione di progetto di qualsiasi dimensione in maniera rapida ed efficiente.

Github

GitHub è un servizi di hosting basato sul cloud che permette di gestire i progetti basati sul sistema di controllo versione Git. Oltre alle funzionalità di versionamento, offre funzionalità come issue tracking e continuous integration.

Python

Python è un linguaggio di alto livello, general-purpose, dinamicamente tipizzato e garbage-collected. Nasce intorno al 1980 da Guido van Rossum, come sostituto del linguaggio di programmazione ABC. Per lo sviluppo del progetto è stata utilizzata la versione 3.9 del linguaggio.

PyCharm Community Edition

PyCharm Community Edition è un ambiente di sviluppo integrato open-source, specializzato nello sviluppo di progetti Python. Sviluppato da JetBrains, offre syntax highlighting, auto indentazione, code completion, controllo ortografia e un debugger Python built-in.

LaTeX

LaTeX è un linguaggio dichiarativo per la stesura di documenti. É stato usato per la stesura dei vari documenti richiesti per il progetto di stage.

StarUML

StarUML è un software utilizzato per modellare diagrammi UML[G]. É stato sviluppato da MKLabs ed è disponibile per Windows, Linux and MacOS.

3.2 Analisi framework proposti

3.2.1 Dash

Figura 3.1: Logo Dash

Dash è un microframework open source sviluppato in Python annunciato nel 2017. Viene utilizzato per la visualizzazione dati e in generale per la creazione di applicazioni web per l'analisi dati. Dash utilizza internamente le tecnologie Flask, Plotly.js e React.js.
Un'app Dash è essenzialmente un'istanza di server Flask che comunica con i client tramite HTTP con pacchetti JSON, usando React.js per il front-end e in particolare sfrutta Plotly.js per la gestione dei grafici.

Vantaggi e punti di forza

Alcuni tra i punti di forza e vantaggi individuati sono:

- Non richiede l'utilizzo di HTML/CSS o Javascript aggiuntivi, in quanto ogni componente in Dash codifica tutte le proprietà di uno specifico componente di React;
  Sono infatti presenti classi Python che codificano la maggior parte dei tag HTML (tramite *dash_html_component*) e tutti i componenti interattivi di base offerti da React (Slider, Dropdown, Graph, ...), tramite *dash_core_components* [5];
- Dash supporta l'aggiunta di nuovi componenti, creando un oggetto sopra a componenti preesistenti in React o definendo un proprio componente personalizzato tramite HTML/CSS e Javascript;
- Pur facendo uso di fogli di stile di default, supporta l'aggiunta di fogli di stile CSS esterni, quindi ogni elemento è personalizzabile usando le API[G] offerte da Dash, oppure introducendo i propri fogli di stile, garantendo il massimo livello di personalizzazione possibile;
- Il codice è dichiarativo e reattivo, e questo semplifica molto lo sviluppo di applicazioni (anche complesse) con numerosi componenti interattivi;
- L'istanza di Flask alla base del server, resta totalmente accessibile, permettendo la modifica di tutte le sue proprietà configurabili. Inoltre può essere estesa (a seconda dei bisogni) con tutta una serie di plugin Flask;

- Ottima visualizzazione degli errori per il debugging; in particolare non vengono nascosti nella console Javascript, ma evidenziati nelle grafica dell'applicazione, quando è in modalità di debug. Vengono anche isolati dalle eccezioni generate da Flask internamente, permettendo una netta distinzione tra un errore causato da Flask o da Dash;
- Tracciamento efficace delle callback; Dash crea una visualizzazione grafica dell'albero delle callback, indicando quando sono state eseguite, per quanto hanno operato e i dati che sono stati passati. Questo si prova molto utile, soprattutto in applicazioni medio-grandi, dove il numero di callback può diventare notevole rapidamente;
- Dash salva lo stato dell'applicazione nel client, cioè nel browser, questo permette di avere dashboard che supportano più utenti in simultanea, permettendo di avere interazioni indipendenti con la stessa applicazione e riducendo le risorse necessarie per un nuovo utente dal lato server;
- La presenza di più community attive e molto materiale informatico e formativo, all'infuori della documentazione;
- Molto utilizzato e riconosciuto in ambito aziendale, tanto da offrire un'opzione premium a pagamento, Dash Enterprise.

Svantaggi e punti di debolezza

Alcuni tra i punti di debolezza individuati sono:

- L'aggiornamento costante di elementi grafici può diventare rapidamente oneroso vista l'architettura sopra la quale è sviluppato, dato che non avendo una cache lato server vanno comunque rifatti tutti i calcoli necessari anche se i dati non sono cambiati;
- Dash rimane molto chiuso ad integrazioni con altre librerie grafiche, dato che per design l'unica supportata rimane Plotly.js, questo può essere limitante a seconda delle applicazioni;
- La creazione di applicazioni multi pagina è stata resa disponibile di recente (prima non era esplicitamente supportata e c'era il bisogno di sviluppare soluzioni non standard e poco stabili), e rimane limitata. Ad esempio lo scambio di dati tra le pagine dell'applicazione deve essere sviluppato esplicitamente (visto che Dash è stateless) e può diventare complesso in poco tempo e provocare una riduzione delle performance;
- L'interattività offerta da Dash, seppur funzionale e concisa, rimane limitata, sia per opzioni (solo callback collegate a cambi di proprietà dei componenti) che per funzionalità, per esempio non è possibile che due o più callback aggiornino lo stesso elemento.
  Questo può portare a soluzioni macchinose e poco eleganti (passaggio di argomenti aggiuntivi come identificativi e definire una grande callback che accetta tutti gli input e output e con una logica complessa per ritornare la risposta corretta, mescolando comportamenti e funzionalità logicamente separate tra di loro);

- Può richiedere conoscenze rudimentali di HTML (e CSS per un design "grafico" migliore e personalizzazione maggiore). Non sono strettamente necessarie, finché la dimensione dell'applicazione che si vuole sviluppare rimane ridotta, però con il crescere di quest'ultima possono diventarlo. A queste si aggiunge la necessità di conoscere Javascript se si vuole deviare dai comportamenti standard dei componenti;
- In generale, anche dovuto a quanto detto nel punto sopra, lo sviluppo in Dash può diventare complesso, soprattutto appena ci si sposta su layout più complicati;
- Difficoltà nel deployment, dato che per un'applicazione Dash il deploy avviene solo nelle piattaforme che supportano Flask, inoltre necessita un setup non triviale dell'ambiente di esecuzione, e su questi aspetti la documentazione è scarsa;
- Dash non supporta molte funzionalità relative all'accessibilità per la maggior parte dei componenti di default offerti e per i grafici renderizzati con Plotly.js.

3.2.2 Panel

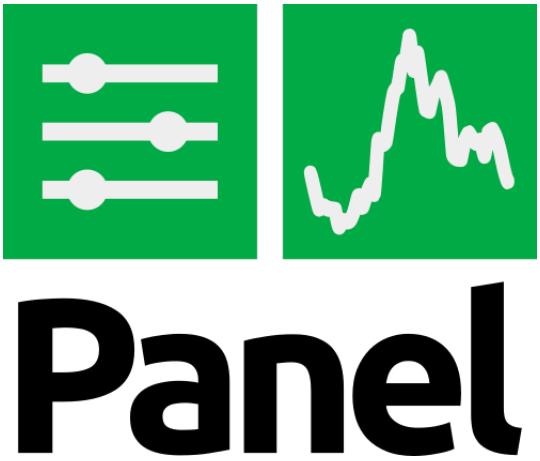


Figura 3.2: Logo Panel

Panel è una libreria Python open source, annunciata nel maggio del 2019, che permette di creare applicazioni web interattive e dashboard personalizzate. Panel è costruito sopra due librerie principali:

- Bokeh, un framework che implementa il pattern MVC su cui si basa Panel; offre anche alcuni componenti base, un layout manager e un server Tornado per la comunicazione tra Python e browser (usando Websocket);
- Param, un framework per definire parametri reattivi, usati per definire tutti i componenti di Panel.

Vantaggi e punti di forza

Alcuni tra i punti di forza e vantaggi individuati sono:

- Panel è stato pensato per essere utilizzato puramente in Python, rendendo non necessaria neanche una minima conoscenza di HTML/CSS e Javascript;
- Panel può interfacciarsi con praticamente tutte le librerie di plotting più comuni (es. Plotly.js, matplotlib, ...);
- Panel si interfaccia nativamente anche con molti degli strumenti connessi all'analisi dati (es. Pandas, Dusk, Datashader, ...);
- Ottimo supporto per la creazione di applicazioni multi-pagina, tramite l'utilizzo di Pipelines, che permettono la facile definizione di un workflow, dove lo stadio precedente passa i dati allo stadio successivo;
- Offre potenti API[G] per i bisogni di ogni sviluppatore [15]:
  - Interact functions: genera un'interfaccia grafica in automatico, data una funzione (la più semplice ma meno personalizzabile);
  - Reactive functions: simile ad Interact functions, ma richiede che i componenti siano dichiarati in maniera esplicita e che i collegamento tra componenti e funzioni sia esplicito;
  - Parameterized classes: tramite Param, permette di definire classi parametrizzate (dove vengono dichiarati i parametri e i loro intervalli), indipendenti dall'interfaccia grafica, che sarà poi generata da Panel. Questo determina molte ottimizzazioni e soprattutto una funzionalità di validazione dei parametri già pronta all'uso (tramite Param);
  - Callbacks: l'interfaccia grafica viene generata dichiarando tutti i componenti necessari e ogni callback necessaria per interazione e l'aggiornamento (il metodo più "vicino" alla macchina, quindi più complesso ma più personalizzabile).

  Questo determina un elevato grado di flessibilità di sviluppo;
- L'architettura su cui Panel è costruito salva lo stato per ogni utente e sessione sia nel server che nel client (browser) e se necessario è possibile sincronizzarli. Questo ha importanti implicazioni, permette infatti di salvare in una cache lato server le computazioni intermedie per ogni utente e rende l'applicazione molto più responsive, permettendo un'esplorazione continua di una visualizzazione con rallentamenti anche impercettibili, perché vengono ricalcolati solo i passaggi necessari;
- Panel renderizza l'applicazione tramite l'utilizzo di un template di default ma offre la possibilità di inserire nuovi template personalizzati e sviluppati dall'utente, permettendo il controllo sul layout di ogni singolo componente.

Svantaggi e punti di debolezza

Alcuni tra i punti di debolezza individuati sono:

- L'architettura sulla quale è sviluppato non scala molto bene se l'applicazione richiede di supportare una molteplicità di utenti simultanei, in quanto può velocemente saturare le risorse del server;

- Similmente a Dash, la difficoltà di utilizzo aumenta di molto se le applicazioni e i layout diventano più complessi. In generale Panel ha una curva di apprendimento più complessa rispetto a Dash;
- La quantità di materiale informativo e formativo disponibile (non considerando la documentazione) è considerevolmente minore rispetto a Dash. Alcuni dei motivi sono il fatto che Dash è uscito 2 anni prima di Panel, e vista la presenza di Dash Enterprise, si è venuta a una comunità più numerosa;
- L'attività di debugging viene resa più complessa (rispetto a Dash) per la mancanza di un interfaccia chiara per la visualizzazione degli errori che sono mostrati nella console Javascript come errori o eccezioni delle librerie interne al framework. Questo rende più difficile, rispetto a Dash, la comprensione di cosa causa l'errore in caso essi siano non triviali. Di recente è stata introdotta un Debugging widget, che pur mitigando il problema resta limitata;
- Difficoltà nel deployment, dove Panel ha comunque un numero più vasto di piattaforme sulla quale si può effettuare il deploy, resta il problema della scarsa documentazione, che manca di profondità (le varie opzioni sono discusse solo brevemente);
- La presenza di molteplici API[G] di sviluppo può portare (soprattutto degli sviluppatori alle prime armi con il framework) ad effettuare delle scelte sbagliate, richiedendo in un tempo successivo un grande sforzo di refactoring.

3.2.3 Scelta framework e motivazioni

In accordo con il proponente, è stato scelto di utilizzare il framework Dash. In un primo momento si era deciso di usare Panel, ma lo sviluppo di un piccolo PoC[G] ha reindirizzato la scelta.
In particolare le motivazioni (alcune personali) sono:

- Difficoltà nel debugging di Panel rispetto a Dash: come spiegato sopra, il fatto che Panel visualizzi principalmente gli errori e le eccezioni delle librerie interne, rendeva il processo di individuazione dei bug inevitabilmente lungo e tedioso. Ho invece trovato l'interfaccia offerta da Dash per la visualizzazione delle eccezioni più chiara e diretta, senza contare l'utilità delle mappa grafica dell'esecuzione delle callback generata da Dash, che invece in Panel era completamente assente;
- Alcuni punti di debolezza di Dash sono stati mitigati, tramite le Dash Extensions [6]. In particolare il `MultiplexerTransform`, che permette di definire più callback con lo stesso output, migliorando la definizione delle callback e permettendo la suddivisione delle funzionalità;
- La scarsa presenza di materiare formativo e informativo per Panel e la ridotta documentazione rispetto a Dash, è stata forse la motivazione che ha avuto più impatto nella scelta del framework. Infatti la presenza di un ottimo forum [16], di numerose guide online e di componenti aggiuntivi non standard già sviluppati si sono rivelati di fondamentale importanza e utilità.

Capitolo 4

Analisi dei requisiti

4.1 Analisi dei casi d'uso

Per lo studio dei casi d'uso sono stati creati dei diagrammi. I diagrammi dei casi d'uso (in inglese *Use Case Diagram*) sono diagrammi di tipo UML[G] dedicati alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso.

Attori individuati

L'unico attore individuato è l'utente generico. Il sistema infatti non include nessun meccanismo di registrazione e login. L'idea iniziale, in accordo con il committente, era che il sistema fosse operato da un solo utente.il quale si occupa anche di installare ed eseguire il back-end in una propria macchina. Solo successivamente è stato introdotto l'obiettivo (non obbligatorio ma desiderabile) di rendere il sistema capace di supportare più utenti contemporaneamente.

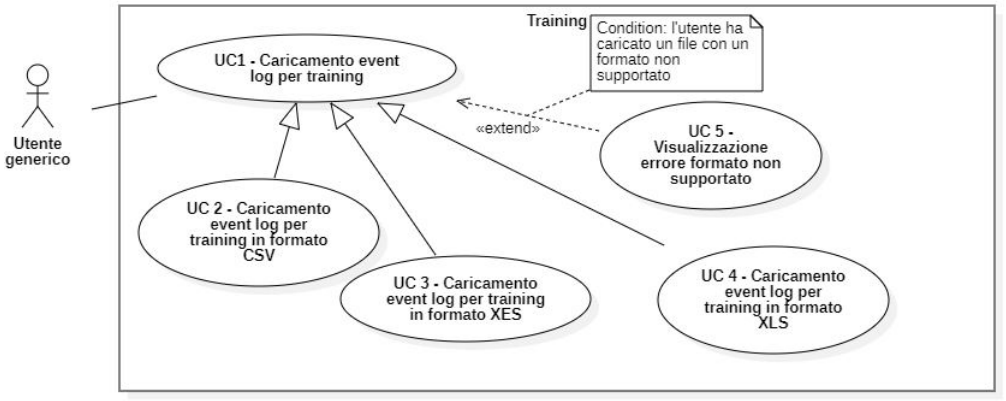


Figura 4.1: Casi d'uso relativi al caricamento file di training

UC 1 - Caricamento dell'event log per il training

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter caricare l'event log per la fase di training;
- **Scenario principale:**
  1. L'utente seleziona l'event log su cui eseguire il training, in formato CSV (**UC 2**), XES (**UC 3**) o XLS (**UC 4**);
  2. L'utente carica l'event log.
- **Estensioni:** L'utente ha tentato di caricare un event log con formato non supportato e viene mostrato un errore (**UC 4**);
- **Precondizioni:** Non è stato caricato nessun event log su cui effettuare training;
- **Postcondizioni:** L'event log per il training è stato correttamente caricato nel sistema.

UC 2 - Caricamento dell'event log per il training in formato CSV

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter caricare l'event log per il training in formato CSV;
- **Scenario principale:** L'utente carica l'event log per il training in formato CSV;
- **Precondizioni:** Non è stato caricato nessun event log su cui effettuare training;
- **Postcondizioni:** L'event log per il training in formato CSV è stato correttamente caricato nel sistema.

UC 3 - Caricamento dell'event log per il training in formato XES

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter caricare l'event log per il training in formato XES;
- **Scenario principale:** L'utente carica l'event log per il training in formato XES;
- **Precondizioni:** Non è stato caricato nessun event log su cui effettuare training;
- **Postcondizioni:** L'event log per il training in formato XES è stato correttamente caricato nel sistema.

UC 4 - Caricamento dell'event log per il training in formato XLS

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter caricare l'event log per il training in formato XLS;
- **Scenario principale:** L'utente carica l'event log per il training in formato XLS;
- **Precondizioni:** Non è stato caricato nessun event log su cui effettuare training;
- **Postcondizioni:** L'event log per il training in formato XLS è stato correttamente caricato nel sistema.

UC 5 - Visualizzazione errore formato non supportato

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve ricevere un errore in caso venga caricato un event log con formato non supportato
- **Scenario principale:**
  1. L'utente seleziona un event log da caricare con un formato non supportato;
  2. L'utente carica l'event log;
  3. Viene mostrato un messaggio d'errore esplicativo.
- **Precondizioni:** L'utente ha caricato un event log con un formato non supportato;
- **Postcondizioni:** Viene mostrato il messaggio d'errore e l'event log non viene caricato.

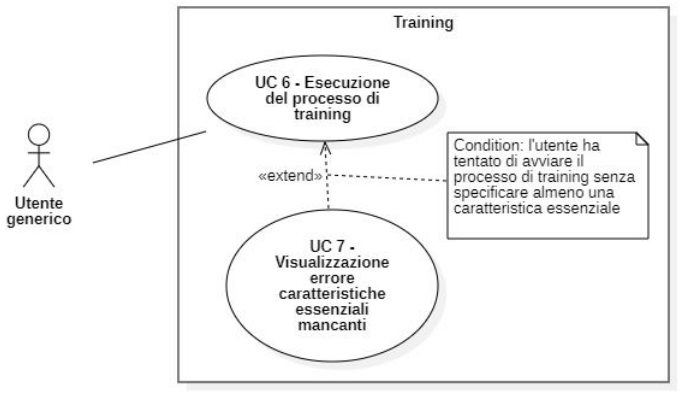


Figura 4.2: Casi d'uso relativi al processo di training

UC 6 - Esecuzione del processo di training

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter avviare il processo di training;
- **Scenario principale:**
  1. L'utente seleziona le colonne relative alle features dell'event log (**UC 6.1**);
  2. L'utente seleziona il KPI che desidera (**UC 6.2**);
  3. L'utente seleziona la soglia di frequenza degli outliers che desidera (**UC 6.3**);
  4. L'utente clicca sul pulsante "Start training";
  5. Il processo di training sull'event log caricato e con le caratteristiche selezionate viene avviato:
  6. L'utente visualizza il progresso del processo di training (**UC 6.4**);
  7. Al termine del processo l'utente viene notificato;
- **Estensioni:** Almeno una delle caratteristiche necessarie non è stata specificata e viene mostrato un errore all'utente (**UC 7**);
- **Precondizioni:** L'event log per il training è stato caricato correttamente (**UC 1**);
- **Postcondizioni:** Il processo di training è stato completato con successo.

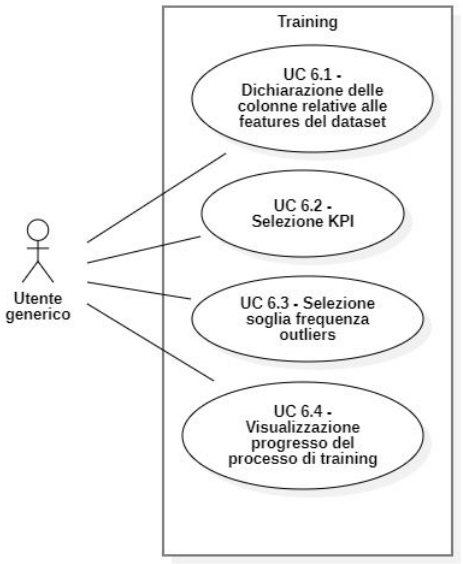


Figura 4.3: Casi d'uso relativi al processo di training

UC 6.1 - Dichiarazione delle colonne relative alle features del dataset

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter dichiarare quali colonne sono relative a quali features del dataset caricato.
- **Scenario principale:**
  1. L'utente dichiara la colonna relativa alla feature "id" (**UC 6.1.1**);
  2. L'utente dichiara la colonna relativa alla feature "activity" (**UC 6.1.2**);
  3. L'utente dichiara la colonna relativa alla feature "timestamp" (**UC 6.1.3**);
  4. L'utente dichiara la colonna relativa alla feature "resurce" (**UC 6.1.4**).
- **Precondizioni:** L'event log in formato CSV o XLS è stato caricato correttamente (**UC 2**, **UC 4**);
- **Postcondizioni:** Il sistema conosce a quale colonna corrisponde ogni feature relative al dataset CSV o XLS caricato.

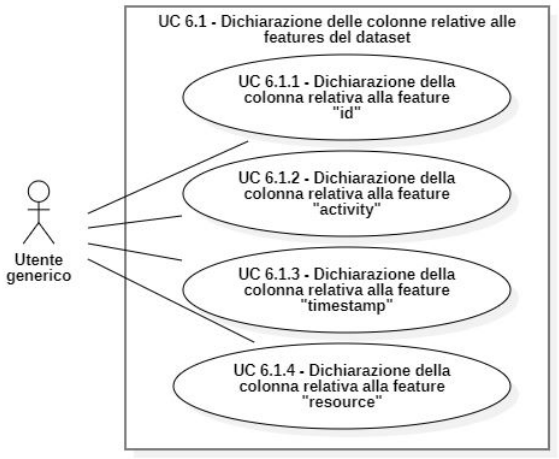


Figura 4.4: Casi d'uso relativi alla dichiarazione delle colonne

UC 6.1.1 - Dichiarazione della colonna relativa alla feature "id"

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter selezionare la colonna corrispondente alla feature "id".
- **Scenario principale:**
  1. L'utente seleziona il campo relativo alla feature "id";
  2. L'utente dichiara la colonna relativa alla feature "id".
- **Precondizioni:** L'event log in formato CSV o XLS è stato caricato correttamente (**UC 2**, **UC 4**);
- **Postcondizioni:** La colonna relativa alla feature "id" è stata dichiarata.

UC 6.1.2 - Dichiarazione della colonna relativa alla feature "activity"

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter selezionare la colonna corrispondente alla feature "activity".
- **Scenario principale:**
  1. L'utente seleziona il campo relativo alla feature "activity";
  2. L'utente dichiara la colonna relativa alla feature "activity".
- **Precondizioni:** L'event log in formato CSV o XLS è stato caricato correttamente (**UC 2**, **UC 4**);
- **Postcondizioni:** La colonna relativa alla feature "activity" è stata dichiarata.

UC 6.1.3 - Dichiarazione della colonna relativa alla feature "timestamp"

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter selezionare la colonna corrispondente alla feature "timestamp".
- **Scenario principale:**
  1. L'utente seleziona il campo relativo alla feature "timestamp";
  2. L'utente dichiara la colonna relativa alla feature "timestamp".
- **Precondizioni:** L'event log in formato CSV o XLS è stato caricato correttamente (**UC 2**, **UC 4**);
- **Postcondizioni:** La colonna relativa alla feature "timestamp" è stata dichiarata.

UC 6.1.4 - Dichiarazione della colonna relativa alla feature "resource"

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter selezionare la colonna corrispondente alla feature "resource".
- **Scenario principale:**
  1. L'utente seleziona il campo relativo alla feature "resource";
  2. L'utente dichiara la colonna relativa alla feature "resource".
- **Precondizioni:** L'event log in formato CSV o XLS è stato caricato correttamente (**UC 2**, **UC 4**);
- **Postcondizioni:** La colonna relativa alla feature "resource" è stata dichiarata.

UC 6.2 - Selezione KPI

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter selezionare il KPI che desidera tra quelli disponibili;
- **Scenario principale:** L'utente seleziona il KPI tra:
  - "Total time" (**UC 6.2.1**);
  - "Minimize activity occurence" (**UC 6.2.2**);
- **Precondizioni:** Il file di training è stato caricato correttamente (**UC 1**);
- **Postcondizioni:** Il KPI è stato selezionato.

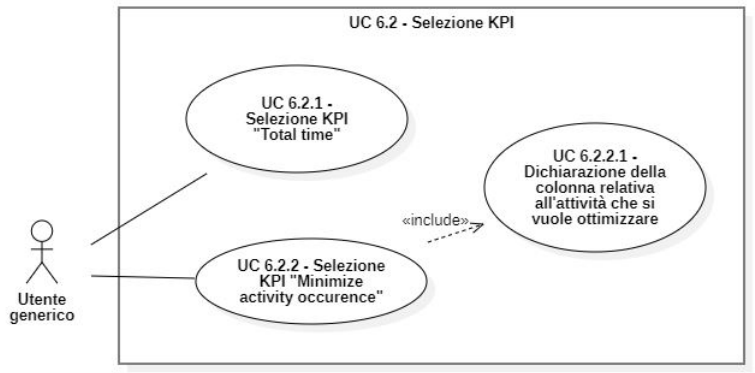


Figura 4.5: Casi d'uso relativi alla selezione del KPI

UC 6.2.1 - Selezione KPI "Total time"

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter selezionare il KPI "Total time";
- **Scenario principale:** L'utente seleziona il KPI "Total time";
- **Precondizioni:** Il file di training è stato caricato correttamente (**UC 1**);
- **Postcondizioni:** Il KPI "Total time" è stato selezionato.

UC 6.2.2 - Selezione KPI "Minimize activity occurence"

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter selezionare il KPI "Minimize activity occurence";
- **Scenario principale:**
  1. L'utente seleziona il KPI "Minimize activity occurence";
  2. L'utente dichiara la colonna relativa all'attività di cui vuole minimizzare le occorrenze (**UC 6.2.2.1**).
- **Precondizioni:** Il file di training è stato caricato correttamente (**UC 1**);
- **Postcondizioni:** Il KPI "Minimize activity occurence" è stato selezionato.

UC 6.2.2.1 - Dichiarazione della colonna relativa all'attività che si vuole ottimizzare

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter selezionare la colonna corrispondente all'attività che vuole ottimizzare;
- **Scenario principale:**
  1. L'utente seleziona il campo relativo all'attività che vuole ottimizzare;

2. L'utente dichiara la colonna relativa all'attività che vuole ottimizzare.

- **Precondizioni:**
  - Il KPI "Minimize activity occurence" è stato selezionato (**UC 6.1.3**);
  - La feature "activity" è stata selezionata (**UC 6.1.2**);
- **Postcondizioni:** L'attività che si vuole ottimizzare è stata dichiarata.

UC 6.3 - Selezione soglia frequenza outliers

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve avere la possibilità di selezionare la soglia della frequenza degli outliers che desidera;
- **Scenario principale:** L'utente seleziona la soglia che desidera;
- **Estensioni:** Se l'utente non seleziona nessuna soglia viene considerata una soglia di default;
- **Precondizioni:** L'utente ha caricato correttamente il file di training (UC 1);
- **Postcondizioni:** La soglia immessa è stata registrata dal sistema.

UC 6.4 - Visualizzazione progresso del processo di training

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter visualizzazione il progresso del processo di training
- **Scenario principale:** L'utente visualizza un indicatore relativo all'attuale progresso del processo di training;
- **Precondizioni:** Il processo di training è stato avviato correttamente (**UC 6**);
- **Postcondizioni:** L'utente è in grado di stimare quanto manca al termine del processo di training.

UC 7 - Visualizzazione errore caratteristiche essenziali mancanti

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente ha tentanto di avviare il training non avendo inserito una delle caratteristiche essenziali;
- **Scenario principale:**
  1. L'utente non ha specificato almeno una tra le seguenti caratteristiche essenziali:
     - KPI (**UC 6.2**);
     - Colonna relativa ad una feature (**UC 6.1.x**);
     - Attività da ottimizzare (**UC 6.2.2.1**).
  2. L'utente ha cliccato il pulsante "Start training";

- **Precondizioni:** L'utente ha tentato di avviare il processo di training senza specificare almeno una caratteristica essenziale;
- **Postcondizioni:** Il processo di training non viene avviato e all'utente viene visualizzato un errore esplicativo.

UC 8 - Download del process model generato dal processo di training

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter fare il download dei file del process model generato del processo di training;
- **Scenario principale:**
  1. L'utente clicca il pulsante "Download training files";
  2. I file che compongono il process model vengono scaricati.
- **Precondizioni:** L'utente ha terminato con successo il processo di training (**UC 6**);
- **Postcondizioni:** L'utente ha scaricato i file del process model.

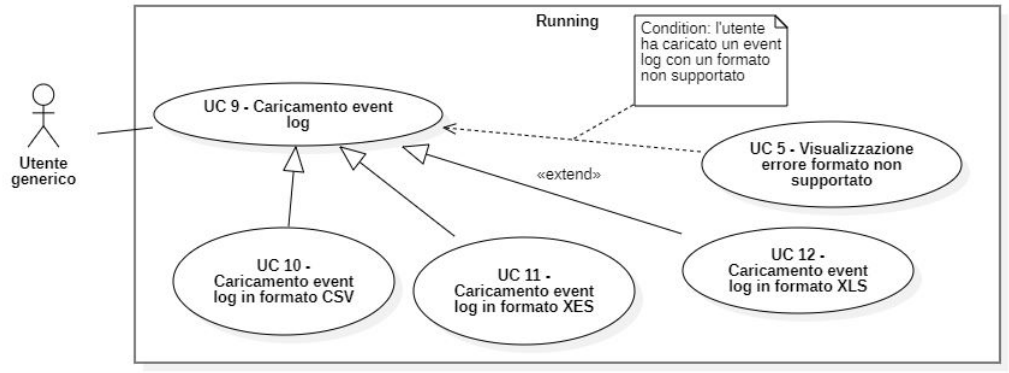


Figura 4.6: Casi d'uso relativi al caricamento del file di log

UC 9 - Caricamento dell'event log per il calcolo delle raccomandazioni

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter caricare l'event log per il calcolo delle raccomandazioni;
- **Scenario principale:**
  1. L'utente seleziona l'event log per il calcolo delle raccomandazioni in formato CSV (**UC 10**), XES (**UC 11**) o XLS (**UC 12**);
  2. L'utente carica l'event log selezionato.
- **Estensioni:** L'utente ha tentato di caricare un event log con formato non supportato e viene mostrato un errore (**UC 5**);

- **Precondizioni:** La fase di training è stata completata (**UC 6**) oppure l'utente ha caricato un modello già allenato (**UC 19**);
- **Postcondizioni:** L'event log per la running phase è stato correttamente caricato nel sistema.

UC 10 - Caricamento dell'event log per il calcolo delle raccomandazioni in formato CSV

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter caricare l'event log per il calcolo delle raccomandazioni in formato CSV;
- **Scenario principale:** L'utente carica l'event log per il calcolo delle raccomandazioni in formato CSV;
- **Precondizioni:** Non è stato caricato nessun event log per il calcolo delle raccomandazioni;
- **Postcondizioni:** L'event log per il calcolo delle raccomandazioni in formato XES è stato correttamente caricato nel sistema.

UC 11 - Caricamento dell'event log per il calcolo delle raccomandazioni in formato XES

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter caricare l'event log per il calcolo delle raccomandazioni in formato XES;
- **Scenario principale:** L'utente carica l'event log per il calcolo delle raccomandazioni in formato XES;
- **Precondizioni:** Non è stato caricato nessun event log per il calcolo delle raccomandazioni;
- **Postcondizioni:** L'event log per il calcolo delle raccomandazioni in formato XES è stato correttamente caricato nel sistema.

UC 12 - Caricamento dell'event log per il calcolo delle raccomandazioni in formato XLS

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter caricare l'event log per il calcolo delle raccomandazioni in formato XLS;
- **Scenario principale:** L'utente carica l'event log per il calcolo delle raccomandazioni in formato XLS;
- **Precondizioni:** Non è stato caricato nessun event log per il calcolo delle raccomandazioni;
- **Postcondizioni:** L'event log per il calcolo delle raccomandazioni in formato XLS è stato correttamente caricato nel sistema.

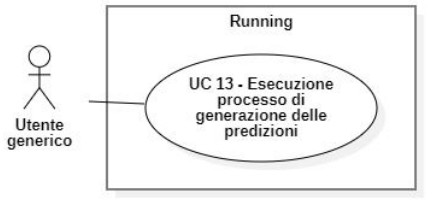


Figura 4.7: Caso d'uso relativo al processo di generazione delle raccomandazioni

UC 13 - Esecuzione processo di generazione delle raccomandazioni

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter avviare il processo di generazione delle raccomandazioni;
- **Scenario principale:**
  1. L'utente clicca sul pulsante "Generate recommendations";
  2. Il processo di generazione delle predizioni sull'event log caricato si avvia;
  3. L'utente visualizza il progresso del processo di generazione delle raccomandazioni (**UC 13.1**);
  4. Al termine del processo l'utente viene notificato.
- **Precondizioni:** L'event log per il calcolo delle raccomandazioni è stato caricato correttamente (**UC 9**);
- **Postcondizioni:** Il processo di generazione delle raccomandazioni è stato completato con successo.

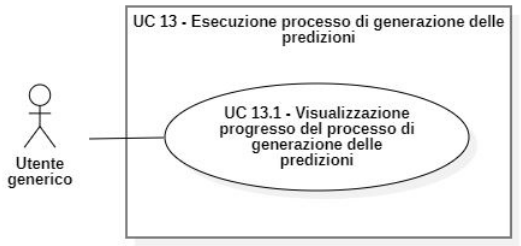


Figura 4.8: Caso d'uso relativo alla visualizzazione del progresso relativo al processo di generazione delle raccomandazioni

UC 13.1 - Visualizzazione progresso del processo di generazione delle raccomandazioni

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter visualizzazione il progresso del processo di generazione delle raccomandazioni

- **Scenario principale:** L'utente visualizza un indicatore relativo all'attuale progresso del processo di generazione delle raccomandazioni;
- **Precondizioni:** Il processo di generazione delle raccomandazioni è stato avviato correttamente (**UC 13**);
- **Postcondizioni:** L'utente è in grado di stimare quanto manca al termine del processo di generazione delle raccomandazioni.

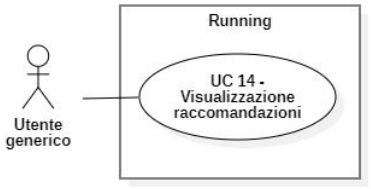


Figura 4.9: Caso d'uso relativo alla visualizzazione delle raccomandazioni

UC 14 - Visualizzazione raccomandazioni

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve essere in grado di visualizzare le raccomandazioni generate;
- **Scenario principale:** L'utente visualizza l'insieme delle raccomandazioni generate.
- **Sottocasi:**
  - L'utente visualizza la raccomandazione relativa alla singola traccia (**UC 14.1**);
- **Estensioni:** L'utente può riordinare le raccomandazioni secondo alcuni ordinamenti predefiniti (**UC 14.2**)
- **Precondizioni:** Il processo di generazione delle predizioni è completato (**UC 13**);
- **Postcondizioni:** L'utente ha visualizzato le predizioni.

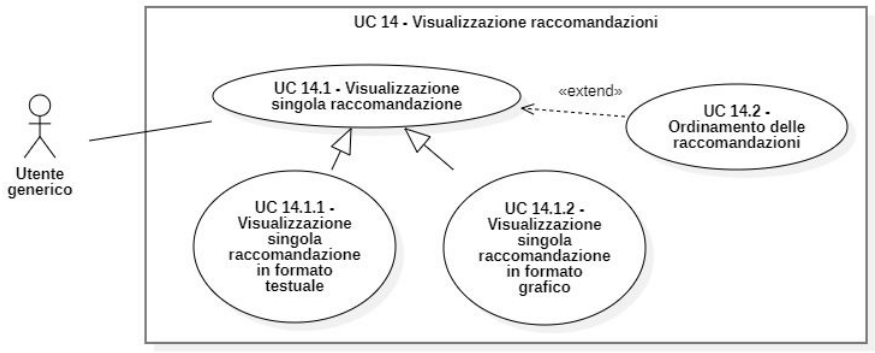


Figura 4.10: Casi d'uso relativi alla visualizzazione della singola raccomandazione

UC 14.1 - Visualizzazione raccomandazioni per una singola traccia

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter visualizzare le raccomandazioni per una singola traccia;
- **Scenario principale:**
  1. L'utente seleziona la traccia a cui è interessato;
  2. L'utente visualizza le raccomandazioni (per la singola traccia) in formato testuale (**UC 14.1.1**) e grafico (**UC 14.1.2**).
- **Precondizioni:** L'utente sta svolgendo l'attività di visualizzazione;
- **Postcondizioni:** L'utente ha visualizzato le raccomandazioni di suo interesse.

UC 14.1.1 - Visualizzazione singola raccomandazione in formato testuale

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter visualizzare le migliori raccomandazioni (fino ad un massimo di 3) e la predizione dell'attività in corso, con i corrispettivi KPI espressi in maniera testuale;
- **Scenario principale:**
  1. L'utente visualizza le 3 migliori raccomandazioni e la predizione dell'attività in corso;
  2. L'utente visualizza i KPI predetti delle attività raccomandate e visualizza il KPI della predizione sulla base dell'attività in corso.
- **Precondizioni:** L'utente sta svolgendo l'attività di visualizzazione;
- **Postcondizioni:** L'utente ha visualizzato la singola raccomandazione di suo interesse.

UC 14.1.2 - Visualizzazione singola raccomandazione in formato grafico

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter visualizzare i KPI della migliore raccomandazione in forma grafica;
- **Scenario principale:** L'utente visualizza un grafico che rappresenta i KPI della migliore attività raccomandata;
- **Precondizioni:** L'utente sta svolgendo l'attività di visualizzazione;
- **Postcondizioni:** L'utente ha visualizzato la singola predizione di suo interesse tramite un grafico.

UC 14.2 - Ordinamento delle raccomandazioni

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente può riordinare l'insieme di tutte le raccomandazioni secondo alcuni ordinamenti predefiniti;
- **Scenario principale:**
  - L'utente sceglie in tipo di ordinamento tra:
    - "Delta from maximum": Le raccomandazioni sono riordinate in ordine decrescente di variazione;
    - "Delta from minimum": Le raccomandazioni sono riordinate in ordine crescente di variazione;
    - "Maximize": Le raccomandazioni sono riordinate in ordine decrescente;
    - "Minimize": Le raccomandazioni sono riordinate in ordine crescente;
  - L'utente visualizza le raccomandazioni con il nuovo ordinamento;
- **Precondizioni:** L'utente sta visualizzando tutte le raccomandazioni generate;
- **Postcondizioni:** L'utente visualizza le raccomandazioni riordinate secondo l'ordinamento scelto;

UC 15 - Ricerca traccia per id

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter ricercare le tracce per id
- **Scenario principale:**
  1. L'utente seleziona il campo di ricerca;
  2. L'utente digita l'id della traccia a cui è interessato;
  3. L'utente può selezionare la traccia ricercata;
- **Precondizioni:** L'utente sta svolgendo l'attività di ricerca;
- **Postcondizioni:** L'utente ha trovato la traccia corrispondente all'id a cui era interessato.

UC 16 - Ricerca traccia tramite grafico

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter ricercare le tracce selezionandole in un grafico che le rappresenta;
- **Scenario principale:**
  1. L'utente visualizza un grafico che mostra tutte le tracce;
  2. L'utente clicca sulla traccia a cui è interessato;
  3. La traccia viene selezionata;
- **Precondizioni:** L'utente sta svolgendo l'attività di ricerca;
- **Postcondizioni:** L'utente ha selezionato una traccia a cui è interessato.

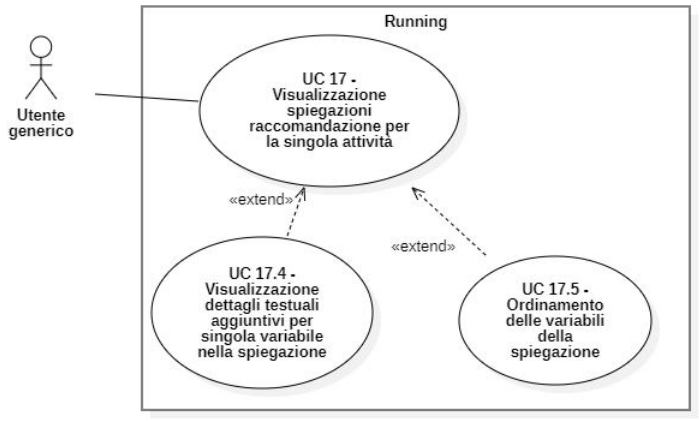


Figura 4.11: Caso d'uso relativo alla visualizzazione delle spiegazioni

UC 17 - Visualizzazione spiegazioni singola raccomandazione

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter visualizzare le spiegazioni per ogni raccomandazione di una singola traccia
- **Scenario principale:**
  1. L'utente seleziona la traccia a cui è interessato (**UC 15**, **UC 16**);
  2. L'utente seleziona la raccomandazione di cui vuole visualizzare la spiegazione (**UC 17.1**);
  3. L'utente seleziona la quantità di variabili che vuole visualizzare nella spiegazione(**UC 17.2**);
  4. L'utente visualizza la spiegazione in formato grafico (**UC 17.3**);
- **Estensioni**:

  - Una volta visualizzate le spiegazioni l'utente può selezionare una singola variabile spiegata per poter ottenere dettagli testuali aggiuntivi (**UC 17.4**);
  - L'utente può riordinare le variabili delle spiegazioni secondo ordinamenti predefiniti (**17.5**);

- **Precondizioni:** L'utente sta svolgendo l'attività di visualizzazione e il processo di generazione delle raccomandazioni è completato (**UC 13**);
- **Postcondizioni:** L'utente ha visualizzato le spiegazioni relative ad una attività tra quelle raccomandate per una traccia di suo interesse.

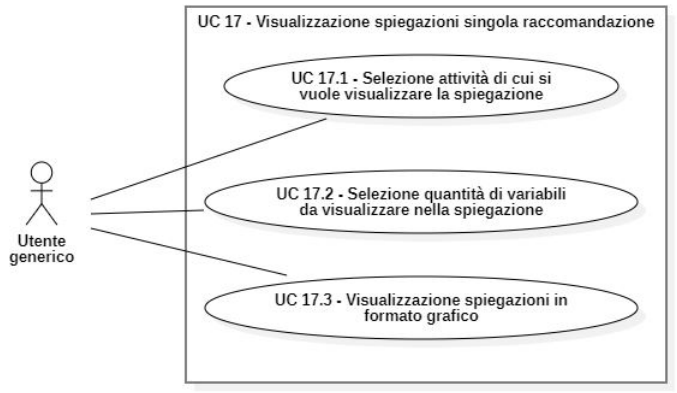


Figura 4.12: Casi d'uso relativi alla visualizzazione di una singola raccomandazione

UC 17.1 - Selezione attività di cui si vuole visualizzare la spiegazione

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter selezionare l'attività raccomandata di cui vuole visualizzare la spiegazione;
- **Scenario principale:** L'utente seleziona l'attività raccomandata che desidera;
- **Precondizioni:** L'utente sta svolgendo l'attività di visualizzazione delle raccomandazioni;
- **Postcondizioni:** Il sistema ha registrato l'attività a cui l'utente è interessato di vedere le spiegazioni.

UC 17.2 - Selezione quantità di variabili da visualizzare nella spiegazione

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter visualizzare quante variabili visualizzare per la spiegazione;
- **Scenario principale:**
  1. L'utente seleziona il campo di selezione quantità;

2. L'utente sceglie la quantità desiderata.

- **Precondizioni:** L'utente sta svolgendo l'attività di visualizzazione delle raccomandazioni;
- **Postcondizioni:** Il sistema ha registrato la quantità di spiegazione che l'utente è interessato a visualizzare

UC 17.3 - Visualizzazione spiegazione in formato grafico

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter visualizzare un grafico che rappresenta le singole variazioni relative alle variabili da spiegare;
- **Scenario principale:** L'utente visualizza un grafico in cui sono rappresentati le singole variazioni alle variabili prese in considerazione;
- **Precondizioni:**
  - l'utente ha selezionato l'attività di cui vuole visualizzare la spiegazione (**UC 17.1**)
  - l'utente ha selezionato la quantità di variabili che desidera visualizzare nella spiegazione (**UC 17.2**)
- **Postcondizioni:** L'utente ha visualizzato le spiegazione in formato grafico;

UC 17.4 - Visualizzazione dettagli testuali aggiuntivi per singola variabile nella spiegazione

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente può selezionare una singola variabile per ottenere una spiegazione testuale più dettagliata;
- **Scenario principale:**
  - L'utente seleziona la variabile della spiegazione sui cui vuole avere dei dettagli aggiuntivi;
  - L'utente visualizza i dettagli aggiuntivi relativi alla variabile selezionata in forma testuale;
- **Precondizioni:** L'utente sta visualizzando la spiegazione di una raccomandazione;
- **Postcondizioni:** L'utente ha ottenuto dettaglia aggiuntivi su una singola variabile della spiegazione;

UC 17.5 - Ordinamento delle variabili della spiegazione

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente può riordinare le variabili della spiegazione secondo alcuni ordinamenti predefiniti;

- **Scenario principale:**
  - L'utente sceglie in tipo di ordinamento tra:
    - "Delta from maximum": Le variabili della spiegazione sono riordinate in ordine decrescente di variazione;
    - "Delta from minimum": Le variabili della spiegazione sono riordinate in ordine crescente di variazione;
  - L'utente visualizza le spiegazioni con il nuovo ordinamento;
- **Precondizioni:** L'utente sta visualizzando la spiegazione di una raccomandazione;
- **Postcondizioni:** L'utente visualizza le variabili della spiegazione riordinate secondo l'ordinamento scelto;

UC 18 - Inserimento nome esperimento

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter nominare l'esperimento effettuato;
- **Scenario principale:** L'utente inserisce il nome che preferisce per l'esperimento;
- **Estensioni:** Se l'utente non inserisce nessun nome viene inserito un nome di default;
- **Precondizioni:** L'utente non ha ancora iniziato il processo di training (**UC 6**)
- **Postcondizioni:** Il sistema ha registrato la preferenza dell'utente.

UC 19 - Caricamento modello già precedentemente allenato

- **Attore primario:** Utente generico;
- **Descrizione:** L'utente deve poter caricare un modello già allenato (permettendo di saltare la fase di training)
- **Scenario principale:**
  1. L'utente seleziona il campo relativo al caricamento di modelli già allenati;
  2. L'utente seleziona un modello da caricare;
  3. L'utente clicca su "Load model".
- **Precondizioni:** L'utente deve aver a disposizione dei modelli già allenati;
- **Postcondizioni:** L'utente può procedere all'inserimento dell'event log per la generazione delle raccomandazioni con il modello caricato.

4.2 Tracciamento dei requisiti

Da un'attenta analisi dei requisiti e degli use case effettuata sul progetto è stata stilata la tabella che traccia i requisiti in rapporto agli use case.
Sono stati individuati diversi tipi di requisiti e si è quindi fatto utilizzo di un codice identificativo per distinguerli.
Il codice dei requisiti è così strutturato:

R[Tipologia][Classificazione]-[Identificativo]

- **Tipologia**:
  - **F**: Funzionale;
  - **NF**; Non funzionale;
- **Classificazione**:
  - **O**: Obbligatorio;
  - **F**: Facoltativo;
- **Identificativo**: codice univoco incrementale per ogni requisito con valore iniziale pari ad 1.

Requisito	Descrizione	Rilevanza	Fonte
RF1	L'interfaccia permette all'utente di caricare un event log per il processo di training	Obbligatorio	UC1
RF2	L'interfaccia permette all'utente di caricare un event log per il processo di training in formato CSV	Obbligatorio	UC2
RF3	L'interfaccia permette all'utente di caricare un event log per il processo di training in formato XES	Obbligatorio	UC3
RF4	L'interfaccia permette all'utente di caricare un event log per il processo di training in formato XLS	Obbligatorio	UC4
RF5	L'interfaccia visualizza un'errore se l'utente carica un event log con formato sono supportato	Obbligatorio	UC5
RF6	L'interfaccia permette all'utente di avviare il processo di training	Obbligatorio	UC6
RF6.1	Se il formato dell'event log per il training caricato è CSV o XLS, L'interfaccia permette all'utente di dichiarare le colonne relative alle features necessarie	Obbligatorio	UC6.1
RF6.1.1	L'interfaccia permette all'utente di dichiarare la colonna relativa alla feature "ID"	Obbligatorio	UC6.1.1
RF6.1.2	L'interfaccia permette all'utente di dichiarare la colonna relativa alla feature "Activity"	Obbligatorio	UC6.1.2
RF6.1.3	L'interfaccia permette all'utente di dichiarare la colonna relativa alla feature "Timestamp"	Obbligatorio	UC6.1.3
RF6.1.4	L'interfaccia permette all'utente di dichiarare la colonna relativa alla feature "Resource"	Obbligatorio	UC6.1.4
RF6.2	L'interfaccia permette all'utente di selezionare il KPI a cui è interessato	Obbligatorio	UC6.2

RF6.2.1	L'interfaccia permette all'utente di selezionare il KPI "Total time"	Obbligatorio	UC6.2.1
RF6.2.2	L'interfaccia permette all'utente di selezionare il KPI "Minimize activity occurrence"	Obbligatorio	UC6.2.2
RF6.2.2.1	Se il KPI "Minimize activity occurrence" è stato selezionato, l'interfaccia permette all'utente di dichiarare l'attività che vuole ottimizzare	Obbligatorio	UC6.2.2.1
RF6.3	L'interfaccia permette all'utente di selezionare la soglia di frequenza degli outliers	Obbligatorio	UC6.3
RF6.4	L'interfaccia permette all'utente di visualizzare il progresso del processo di training	Obbligatorio	UC6.4
RF7	L'interfaccia visualizza un errore se l'utente non inserisce almeno una delle caratteristiche essenziali per il processo di training	Obbligatorio	UC7
RF8	L'interfaccia permette all'utente di effettuare il download dei file del process model generato dal processo di training	Obbligatorio	UC8
RF9	L'interfaccia permette all'utente di caricare l'event log per il processo di generazione delle raccomandazioni	Obbligatorio	UC9
RF10	L'interfaccia permette all'utente di caricare l'event log per il processo di generazione delle raccomandazioni in formato CSV	Obbligatorio	UC10
RF11	L'interfaccia permette all'utente di caricare l'event log per il processo di generazione delle raccomandazioni in formato XES	Obbligatorio	UC11
RF12	L'interfaccia permette all'utente di caricare l'event log per il processo di generazione delle raccomandazioni in formato XLS	Obbligatorio	UC12
RF13	L'interfaccia permette all'utente di avviare il processo di generazione delle raccomandazioni	Obbligatorio	UC13
RF13	L'interfaccia permette all'utente di avviare il processo di generazione delle raccomandazioni	Obbligatorio	UC13
RF13.1	L'interfaccia permette all'utente di visualizzare il progresso del processo di generazione delle raccomandazioni	Obbligatorio	UC13.1
RF14	L'interfaccia permette all'utente di visualizzare le raccomandazioni generate	Obbligatorio	UC14
RF14.1	L'interfaccia permette all'utente di visualizzare le raccomandazioni relative ad una singola traccia	Obbligatorio	UC14.1
RF14.1.1	L'interfaccia permette all'utente di visualizzare le raccomandazioni relative ad una singola traccia in formato testuale	Obbligatorio	UC14.1.1
RF14.1.2	L'interfaccia permette all'utente di visualizzare le raccomandazioni relative ad una singola traccia in formato grafico	Obbligatorio	UC14.1.2
RF14.2	L'interfaccia permette all'utente di riordinare l'insieme delle raccomandazioni generate	Obbligatorio	UC14.2

RF15	L'interfaccia permette all'utente di ricercare una traccia tramite id	Obbligatorio	UC15
RF16	L'interfaccia permette all'utente di ricercare una traccia tramite grafico	Obbligatorio	UC16
RF17	L'interfaccia permette all'utente di visualizzare le spiegazioni di una singola raccomandazione	Obbligatorio	UC17
RF17.1	L'interfaccia permette all'utente di selezionare l'attività raccomandata di cui vuole visualizzare la spiegazione	Obbligatorio	UC17.1
RF17.2	L'interfaccia permette all'utente di selezionare la qualtità di variabili che vuole visualizzare nella spiegazione	Obbligatorio	UC17.2
RF17.3	L'interfaccia permette all'utente di visualizzare la spiegazione in formato grafico	Obbligatorio	UC17.3
RF17.4	L'interfaccia permette all'utente di visualizzare dettagli testuali aggiuntivi per ogni singola variabile della spiegazione	Obbligatorio	UC17.4
RF17.5	L'interfaccia permette all'utente di riordinare le variabili della spiegazione	Obbligatorio	UC17.5
RF18	L'interfaccia permette all'utente di definire il nome da dare all'esperimento effettuato	Obbligatorio	UC18
RF19	L'interfaccia permette all'utente caricare nel sistema un modello già allenato in precedenza	Obbligatorio	UC19

Tabella 4.1: Tabella del tracciamento dei requisti funzionali

Requisito	**Descrizione**	**Rilevanza**	**Fonte**
RNF1	L'interfaccia deve funzionare correttamente nelle ultime attuali versioni dei sistemi operativi Windows, MacOS e Ubuntu	Obbligatorio	Committente
RNF2	L'interfaccia deve essere sviluppata usando il framework Dash	Obbligatorio	Scelta interna
RNF3	L'interfaccia deve essere sviluppata in modalità wizard (cioè una procedura step by step)	Obbligatorio	Committente
RNF4	L'interfaccia deve poter essere utilizzata da più utenti indipendenti contemporaneamente	Desiderabile	D01
RNF5	Devono essere presenti test di unità per l'interfaccia grafica	Obbligatorio	O06
RNF6	Il linguaggio dell'interfaccia deve essere l'inglese	Obbligatorio	Committente
RNF7	Tutti i dati relativi al process model, alle raccomandazioni e alle spiegazioni generate vanno salvati su disco	Obbligatorio	Committente

Tabella 4.2: Tabella del tracciamento dei requisti non funzionali

Capitolo 5

Progettazione e codifica

5.1 Design dell'interfaccia grafica

Per definire un primo design dell'interfaccia grafica sono stati realizzati alcuni mock up[G], utili anche per semplificare la progettazione e l'analisi dei vari aspetti relativi all'esperienza dell'utente. Come da requisiti, l'interfaccia è stata sviluppata in modalità "wizard", ovvero una procedura a fasi dove l'utente deve completare l'azione richiesta da una fase per poter passare alla successiva.

Sono state individuate 3 fasi differenti (a cui corrispondono 3 pagine nell'interfaccia grafica), di seguito definite:

- La fase di *training*: corrisponde alla prima fase, in cui viene creato il process model tramite tecniche di Machine Learning. Viene utilizzato l'event log caricato dall'utente contenente tracce complete (processi già terminati). L'utente potrà procedere alla fase successiva solo se il process model viene generato con successo;
- La fase di *runtime*: corrisponde alla seconda fase, in cui vengono generate le raccomandazioni. Viene utilizzato l'event log caricato dall'utente contenente tracce incomplete (processi ancora da terminare). L'utente potrà procedere solo se il processo di generazione delle raccomandazioni termina con successo;
- La fase di visualizzazione: corrisponde alla terza ed ultima fase, in cui l'utente può visualizzare le raccomandazioni generate e le corrispettive spiegazioni. Questa fase rappresenta l'obiettivo finale dell'interfaccia, e la più "interessante" dal punto di vista dell'utente, che ha eseguito le fasi precedenti con lo scopo di visualizzare le raccomandazione e spiegazioni a cui era interessato.

Con il termine **esperimento** ci si riferirà, nel resto del testo, ad una singola esecuzione completa di almeno la prima fase.

5.1.1 Fase di training

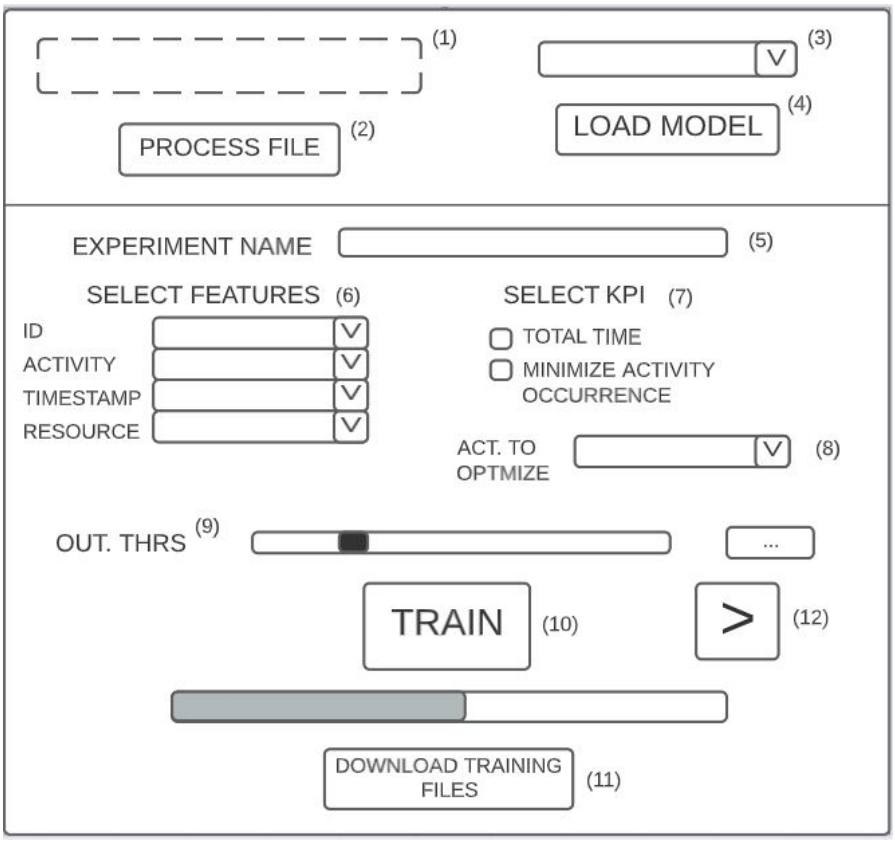


Figura 5.1: Design iniziale della pagina relativa alla fase di training

In Figura 5.1 il mock up su cui si è basato l'aspetto finale della pagina relativa alla fase di training.
Seguendo i requisiti, il flusso principale di utilizzo della pagina è il seguente (i riferimenti numerici successivi, in forma (x), sono relativi alla Figura 5.1):

1. L'utente carica l'event log che vuole utilizzare per il processo di training, usando l'area di drag-and-drop[G] (1) e clicca il pulsante "Process file" (2) che permette all'interfaccia di analizzare ed elaborare il file caricato;

2. L'utente dichiara i nomi delle colonne dell'event log caricato, corrispondenti alle features richieste, usando gli elementi dropdown[G] (6), le opzioni disponibili sono l'insieme delle colonne dell'event log caricato (grazie alla fase di elaborazione). Questo semplifica e velocizza il processo di compilazione, in quanto l'utente deve solo selezionare la colonna giusta al posto di scriverla, riducendo gli errori. Da notare come le features sono tutte obbligatorie tranne la feature "resource";

3. L'utente seleziona il KPI che desidera (7), se sceglie "Minimize activity occurrence" dovrà anche dichiarare il nome dell'attività da ottimizzare, tramite l'apposito dropdown[G] (8), che anche qui semplifica e velocizza la compilazione in quanto contiene già l'insieme di tutte le attività presenti nell'event log;

4. L'utente inserisce il nome dell'esperimento nella casella di testo (5) apposita e seleziona, tramite uno slider[G] (9), la soglia degli outliers che desidera (essa ha comunque un valore di default iniziale);

5. L'utente avvia il processo di training cliccando sul pulsante "Train" (10);

6. Al termine del processo di training l'utente può, se lo desidera, scaricare i file che compongono il process model cliccando il pulsante "Download training files" (11);

7. Infine l'utente può passare alla pagina successiva cliccando sul pulsante a freccia (12).

Va evidenziato che lo step 4. può essere effettuato in qualsiasi momento, basta che sia prima del passo 5. e dopo il passo 1., mentre il passo 6. è opzionale.

Se è già presente un process model (creato in un esperimento precedente) è possibile caricarlo nel sistema e saltare tutto il processo di training (che può richiedere una quantità di tempo considerevole). Il flusso di esecuzione alternativo è il seguente (i riferimenti numerici successivi, in forma (x), sono relativi alla Figura 5.1):

1. L'utente seleziona un process model già creato tra quelli disponibili, usando l'apposito dropdown[G] (3) e clicca il pulsante "Load model" (4) per caricare il process model nel sistema;

2. L'utente può, se lo desidera, scaricare i file che compongono il process model cliccando il pulsante "Download training files" (11);

3. Infine l'utente può passare alla pagina successiva cliccando sul pulsante a freccia (12).

In questo caso invece, il passo 2. è da considerarsi opzionale.

5.1.2 Fase di runtime

In Figura 5.2 il mock up su cui si è basato l'aspetto finale della pagina relativa alla fase di runtime.
Seguendo i requisiti, il flusso principale di utilizzo della pagina è il seguente (i riferimenti numerici successivi, in forma (x), sono relativi alla Figura 5.2):

1. L'utente carica l'event log che vuole utilizzare per la generazione delle raccomandazioni, usando l'area di drag-and-drop[G] (1) e clicca il pulsante "Process file" (2) che permette all'interfaccia di analizzare ed elaborare il file caricato;

2. L'utente clicca sul pulsante "Generate recommendations" (3) per avviare il processo di generazione delle raccomandazioni;

3. Al termine del processo di generazione delle raccomandazioni l'utente può passare alla pagina successiva cliccando sul pulsante a freccia (4).

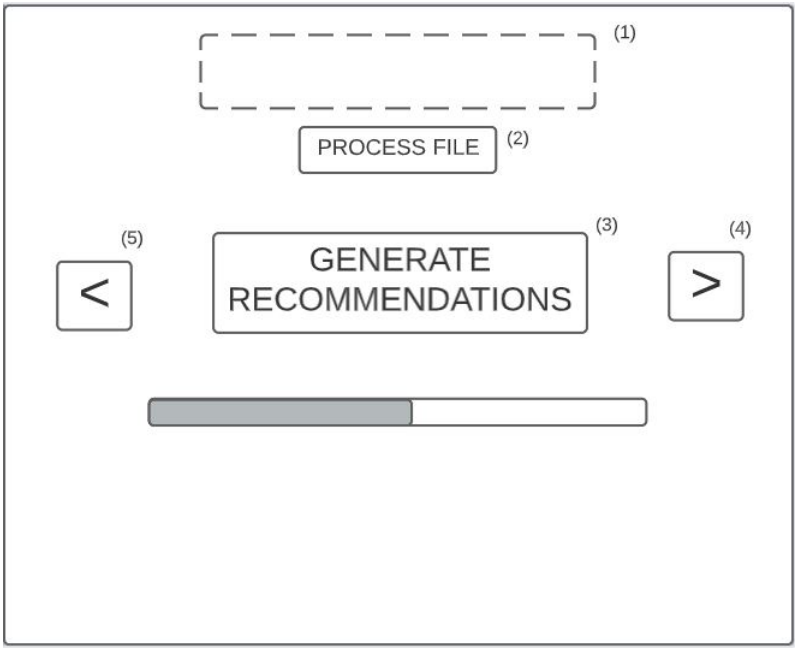


Figura 5.2: Design iniziale della pagina relativa alla fase di runtime

5.1.3 Fase di visualizzazione

In Figura 5.3 il mock up su cui si è basato l'aspetto finale della pagina relativa alla fase di visualizzazione.
Seguendo i requisiti il flusso principale di utilizzo della pagina è il seguente (i riferimenti numerici successivi, in forma (x), sono relativi alla Figura 5.3):

1. L'utente visualizza, tramite il grafico (1), la migliore delle 3 raccomandazioni generate, e ha la possibilità di selezionare una traccia cliccando sulla rispettiva raccomandazione nel grafico oppure inserendo la traccia nella casella di testo apposita (2);

2. L'utente, se desidera, può riordinare le raccomandazioni generate usando uno dei 4 pulsanti in (3);

3. Una volta selezionata la traccia l'utente può visualizzare il resto delle raccomandazioni generate, con relativo KPI predetto nella tabella (4);

4. L'utente può selezionare una delle raccomandazioni di cui vuole visualizzare la spiegazione, tramite un click sulla corrispondente riga della tabella (4);

5. L'utente può scegliere la quantità di variabili da visualizzare nella spiegazione tramite lo slider[G] apposito (5), in ogni caso viene definita una quantità di default;

6. L'utente avvia il processo di generazione della spiegazione cliccando sul pulsante "Generate explanazions" (6);

7. Al termine del processo di generazione della spiegazione l'utente può visualizzarla nel grafico apposito (7);

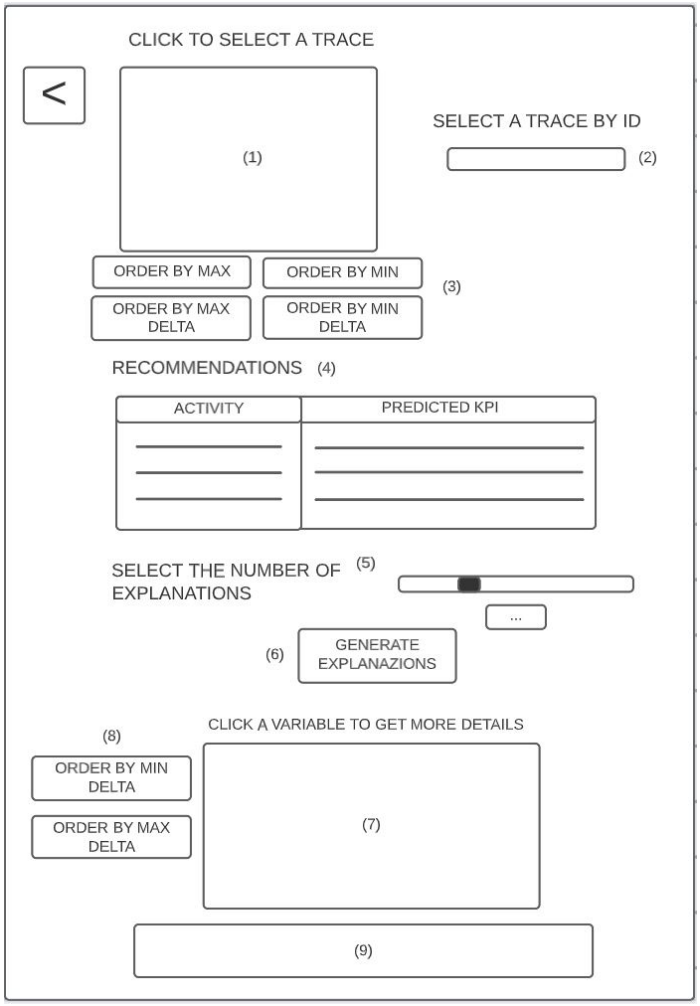


Figura 5.3: Design iniziale della pagina relativa alla fase di visualizzazione

8. L'utente, se desidera, può riordinare le variabili della spiegazione generata usando uno dei 2 pulsanti in (8);

9. L'utente può cliccare su una specifica variabile della spiegazione generata per ottenere maggiori dettagli in formato testuale nello spazio designato (9).

Alcune osservazioni sui punti 1. e 3.: il sistema genera fino ad un massimo di 3 raccomandazioni per traccia, la scelta di visualizzare sul grafico solo la migliore delle 3, come descritto nel punto 1., era per evitare di saturare il grafico. Va considerato infatti, che l'utente sarà interessato maggiormente alla migliore delle raccomandazioni. Poi è libero di consultare il resto delle raccomandazioni generate ma in forma solo testuale, come descritto nel punto 3., un grafico troppo saturo avrebbe sono effetti negativi sulla comprensione dell'utente.

5.2 Architettura del sistema

L'architettura del sistema segue il design pattern[G] **MVP**, molto comune per lo sviluppo di interfacce grafiche.

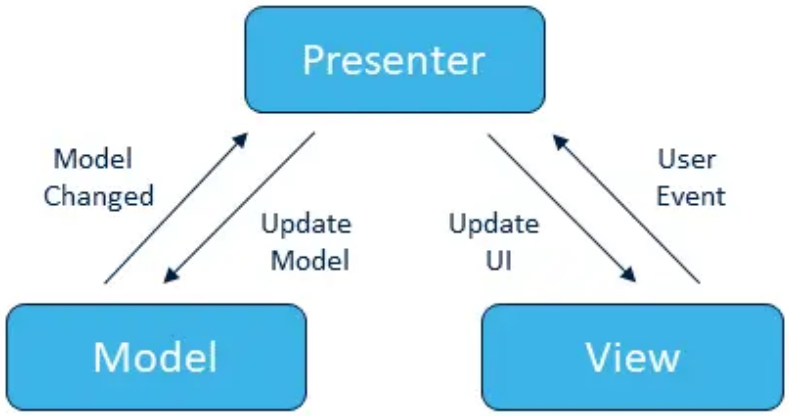


Figura 5.4: Struttura design pattern MVP [12]

Esso è composto da 3 componenti principali:

- *Model*: Contiene la logica di business, ovvero i dati a cui siamo interessati, ed è responsabile della loro gestione (lettura, scrittura, modifica);
- *View*: Rappresenta l'interfaccia più a stretto contatto con l'utente e si occupa di mostrare i dati e di raccogliere l'input dell'utente. Nel caso particolare del MVP, la vista è passiva, quindi non dovrebbe contenere nessuna logica applicativa ma solo visuale;
- *Presenter*: Rappresenta il ponte do comunicazione tra View e Model, riceve gli input dalla View, si occupa di recuperare i dati e dell'esecuzione della logica di business da parte del Model, per poi formattare e inviare alla View i dati da visualizzare.

Un'aspetto importante è il fatto che la View non ha conoscenze (e nessuna dipendenza) dal Model e viceversa.
Questo porta ai principali motivi che hanno spinto a scegliere il design pattern[G]:

- Testabilità: le singole parti possono essere testate in isolamento più facilmente dato che le logiche applicative e visuali sono separate;
- Separazione dei compiti: la separazione logiche e la ridotta dipendenza tra le parti (in particolare Model e View sono totalmente separati) permette di avere codice più modulare, quindi semplice e mantenibile;
- Compatibilità con il framework: vista la forma dichiarativa delle viste costruite con il framework Dash si adatta bene alla View passiva caratteristica del MVP.

5.3 Progettazione classi dell'interfaccia grafica

Di seguito saranno utilizzati i digrammi UML[G] per la rappresentazione delle classi che compongono l'interfaccia.

5.3.1 Progettazione View

Per la progettazione della componente View è stata sviluppata una struttura a views multiple, una per ogni pagina descritta nella sezione 5.1.

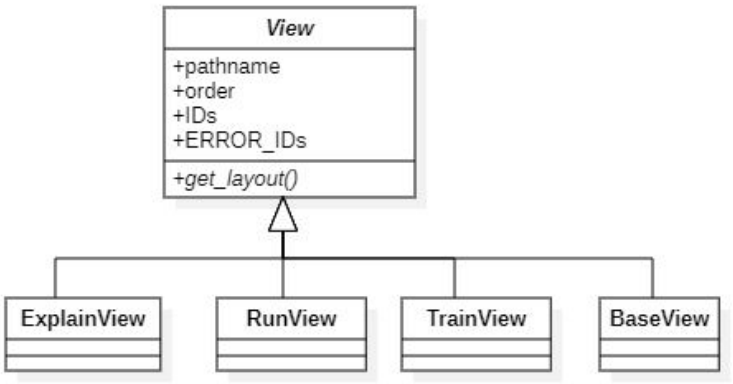


Figura 5.5: Diagramma UML per la classi della View

Viene definita una classe astratta, che rappresenta una view generica, sulla quale si basano le altre views. Questo è utile per esporre le funzionalità comuni che deve avere una view. In particolare ogni view deve definire:

- Un attributo `pathname` che rappresenta il nome della vista, usato per sviluppare l'interfaccia in modalità wizard;
- Un attributo `order` che rappresenta l'ordimento utilizzato per mostrare le pagine, serve a definire i predecessori e/o successori di ogni pagina, usato per sviluppare l'interfaccia in modalità wizard;
- Un attributo `IDs` che definisce l'insieme di tutti gli identificativi univoci dei componenti usati dal framework Dash per la costruzione della parte grafica dell'interfaccia;
- Un attributo `ERROR_IDs` che definisce l'insieme di tutti gli identificativi univoci dei componenti usati dal framework Dash per visualizzare gli eventuali messaggi d'errore;
- L'implementazione del metodo astratto `get_layout` che definisce il layout dell'applicazione web composto dall'insieme dei componenti Dash e la loro struttura, per la creazione della parte grafica della specifica pagina.

Sono state definite 4 classi concrete che implementano la classe `View`:

- La classe `TrainView` relativa alla pagina per la fase di training,
- La classe `RunView` relativa alla pagina per la fase di runtime,

- La classe `ExplainView` relativa alla pagina per la fase di visualizzazione;
- La classe `BaseView` che si occupa della gestione dei componenti comuni a tutte le view (ad esempio i pulsanti per navigare tra le pagine).

5.3.2 Progettazione Model

La progettazione del Model ha tenuto conto del back-end sottostante, cercando di suddividere la struttura in classi specializzate, promuovendo l'incapsulamento e la modularità.

Del diagramma di classi in Figura 5.6 si possono individuare le 3 classi principali, le cui responsabilità sono correlate alle 3 fasi descritte nella sezione 5.1:

- La classe `Trainer` che gestisce il processo di training e la memorizzazione dei dati generati su disco, in particolare sfrutta il metodo `prepare_dataset` che si occupa di processare l'event log caricato per prepararlo al processo di training ed il metodo `train` che esegue il processo di training stesso per generare il process model;
- La classe `Recommender` che gestisce il processo di generazione e memorizzazione delle raccomandazioni generate su disco. Anche per questa classe è necessario il metodo `prepare_dataset` per processare l'event log caricato dall'utente e renderlo utilizzabile per il processo di generazione delle raccomandazioni che, a sua volta, viene eseguito dal metodo `generate_recommendations`;
- La classe `Explainer` che si occupa della generazione, gestione e memorizzazione su disco delle spiegazioni per le raccomandazioni. Per far questo utilizza i dati prodotti dalle classi sopra descritte (ad esempio l'attributo `model` che rappresenta il process model generato) ed il metodo `calculate_explanations` che genera le spiegazioni per una singola raccomandazione. Inoltre si occupa della visualizzazione delle raccomandazioni, generando le struttura dati che userà poi il componente View, usando i metodi `calculate_best_scores` e `get_best_n_scores`;

Le classi principali sopra descritte sfruttano a loro volta delle classi accessorie che si occupano di quelle funzionalità che, pur rimanendo necessarie, possono essere separate da quello che è il compito della classe principale. Esse sono:

- La classe `TrainDataSource` che viene usata dalla classe `Trainer` per la gestione dell'event log, infatti rappresenta la sorgente dei dati per il processo di training. Deriva dalla classe astratta `DataSource` di cui implementa il metodo `read_data` che si occupa di leggere il file dell'event log dalla memoria. Alcuni attributi di nota sono, ad esempio, `data` che contiene i dati dell'event log in forma grezza e `columns_list` che contiene tutti i nomi delle colonne presenti nell'event log. Offre i metodi `get_activity_list` per ottenere la lista di attività presenti nell'event log (necessita di sapere a quale colonna corrisponde la feature "activity") ed il metodo `convert_datetime_to_seconds` per convertire i dati temporali della colonna relativa alla feature "timestamp" in secondi;
- La classe `RunDataSource` che rappresenta la sorgente di dati relativi all'event log, per la classe `Recommender`. Essa ha la sua implementazione del metodo `read_data` per la lettura del file dell'event log in memoria. Gli scopi sono paralleli a quelli della classe `TrainDataSource` e gli attributi hanno funzioni

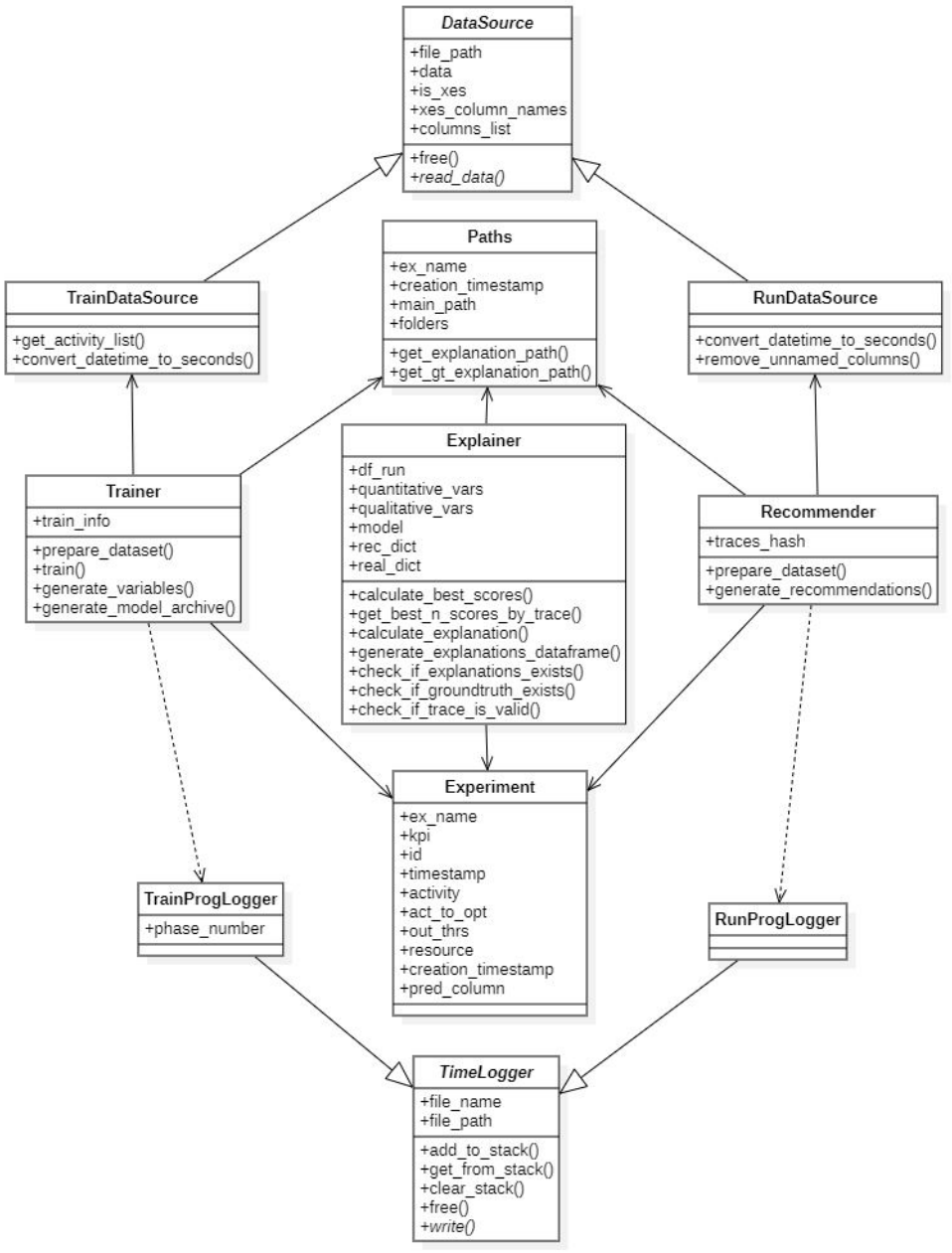


Figura 5.6: Diagramma UML per la classi del Model

simili. Una differenza è la presenza del metodo `remove_unnamed_columns` che ripulisce l'event log di eventuali colonne vuote o senza identificativo che possono essere generate dopo aver letto i dati con il metodo `read_data`;

- La classe `Experiment` è una classe senza metodi che contiene tutti i dati relativi all'esperimento corrente, ed esempio `ex_name` identifica il nome dell'esperimento, `creation_timestamp` il timestamp della creazione dell'esperimento mentre

`id`, `timestamp`, `activity` e `resource` rappresentano colonne relative alle varie features richieste per gli event log. La classe viene usata da tutte e 3 le classi principali;

- La classe `Paths` viene utilizzata da tutte e 3 le classi principali per la gestione di ogni percorso nel file system del server per i vari file usati per la memorizzazione dei dati generati, così da avere una sorgente univoca e facilitarne l'accesso e la modifica;
- La classe `TrainProgLogger` che si occupa di gestire la visualizzazione del progresso del processo di training. Viene utilizzata dal metodo `train` della classe `Trainer`;
- La classe `RunProgLogger` che si occupa di gestire la visualizzazione del progresso del processo di generazione delle raccomandazioni. Viene utilizzata dal metodo `generate_recommendations` della classe `Recommender`.

5.3.3 Progettazione Presenter

Per la progettazione del componente Presenter è stato tenuto conto della relazione tra singole view e singolo presenter, è stata definita una struttura a presenter multipli, che va a riflettere la struttura della componente View (sottosezione 5.3.1).

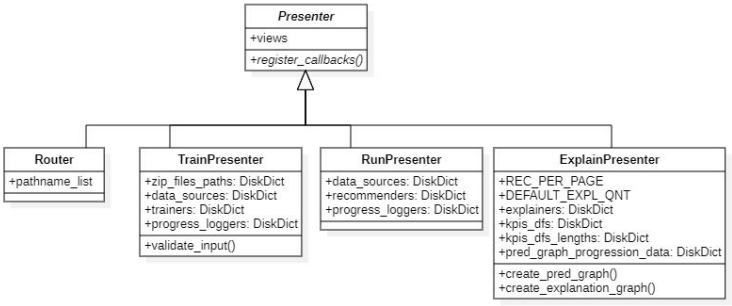


Figura 5.7: Diagramma UML per la classi del Presenter

Lo scopo fondamentale del componente Presenter è la definizione delle callback per il framework Dash.
Le callback rappresentano lo strumento offerto da Dash per sviluppare la componente interattiva di un'interfaccia grafica.
In una applicazione Dash gli input e output non sono altro che le proprietà dei vari componenti che formano il layout. Le callback, sono delle funzioni Python che accettano una o più proprietà di specifici componenti come input e ritornano una o più proprietà di specifici componenti come output. Quando almeno una delle proprietà tracciate negli input viene modificata, la callback si avvia, esegue la sua logica interna e ritorna le modifiche alle proprietà di output, che vengono applicate al layout.

Un esempio di callback:

```
        @app.callback(
    Output(component_id='textout', component_property='value'),
    Input(component_id='textin', component_property='value')
)
def update_output(input_value):
    return input_value
```

In questo caso la callback `update_output` verrà eseguita quando la proprietà `value` del componente con id `textin` verrà modificata. Al termine dell'esecuzione il valore in input verrà semplicemente ritornato e Dash si occuperà di aggiornare la proprietà `value` del componente con id `textout` con il nuovo testo.

Anche per progettazione del Presenter è stata sfruttata una classe astratta, la classe `Presenter`, che espone le funzionalità comuni per un singolo presenter, ovvero:

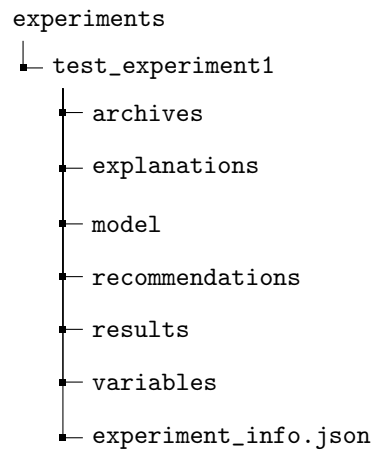
- L'attributo `views` che contiene i vari riferimenti all'insieme delle view che il presenter può gestire;
- Il metodo `register_callbacks`, un metodo astratto che ogni istanza della classe `Presenter` deve implementare. Esso definisce tutte le callback che codificano il comportamento dell'interfaccia in risposta all'input dell'utente.

Sono state definite 4 classi concrete che implementano la classe `Presenter`:

- La classe `Router` che gestisce principalmente la classe `BaseView` della componente View, quindi si occupa di tutti gli aspetti relativi al cambio di pagina. L'attributo `pathname_list` è definito in automatico e contiene gli identificativi di tutte le altre view, ordinate secondo il loro attributo `order`.
- La classe `TrainPresenter` che gestisce la classe `TrainView` della componente View. Si interfaccia con il componente Model per l'esecuzione del processo di training. Per fare questo vengono utilizzati gli attributi `trainers`, `data_sources` e `progress_loggers` che rappresentano le istanze delle classi `Trainer`, `TrainDataSource` e `TrainProgLogger` della componente Model. Poiché la pagina della fase di training accetta l'input dell'utente è stato definito il metodo `validate_input` che ne gestisce la validazione;
- La classe `RunPresenter` che gestisce la classe `RunView` della componente View. Anche questa classe necessita di interfacciarsi con il componente Model per gestire l'esecuzione del processo di generazione delle raccomandazioni. Sfrutta gli attributi `recommenders`, `data_sources` e `progress_loggers` che rappresentano le istanze delle classi `Recommender`, `RunDataSource` e `RunProgLogger` della componente Model.
- La classe `ExplainPresenter` che gestisce la classe `RunView` della componente View. Quindi gestisce la generazione delle spiegazioni, tramite la componente Model, sfruttando, in particolare l'attributo `explainers` che rappresenta le istanze della classe `Explainer`. Si occupa anche della visualizzazione delle raccomandazioni e delle spiegazioni, sfruttando il metodo `create_pred_graph` per generare il grafico delle raccomandazioni ed il metodo `create_explanation_graph` per generare il grafico delle spiegazioni. Il resto degli attributi è usato per gestire la visualizzazione dei grafici.

5.3.4 Struttura cartelle per il salvataggio dei file su disco

Come da requisiti, era necessario il salvataggio di tutti i dati relativi al process model generato, alle raccomandazioni e alle spiegazioni, per ogni esperimento eseguito. La scelta più semplice era appunto salvarli su disco, ovvero sulla memoria permanente del server.
Di seguito l'organizzazione delle cartelle che è stata progettata:



La struttura sopra rappresenta un ipotetico esperimento nominato "test_experiment1". In particolare sono definiti i seguenti elementi:

- `archives` è una cartella che contiene l'archivio compresso di tutti i dati del process model. Questo può venir scaricato dall'utente al termine del processo di training, se lo desidera;
- `explanations` è una cartella che contiene tutte le spiegazioni generate;
- `model` è una cartella che contiene tutti i dati relativi al process model;
- `recommendations` è una cartella che contiene tutti i dati relativi alle raccomandazioni generate;
- `results` è una cartella che contiene alcune delle metriche calcolate durante il processo di training;
- `variables` è una cartella che contiene i dati relativi alle variabili qualitative e quantitative generate al termine del processo di training;
- `experiment_info.json` è un file che contiene tutti i dati relativi all'esperimento (nome, timestamp di creazione, KPI scelto e features).

5.4 Codifica

5.4.1 Sviluppo della modalità wizard

Prima di tutto è stata sviluppato il sistema per attuare la modalità wizard, che definisce l'aspetto base dell'interfaccia: una procedura a fasi ordinate, dove ad ogni singola fase corrisponde una pagina, con la possibilità di navigare tra le pagine tramite un comando

che permette di andare alla pagina successiva ed un comando che permette di andare alla pagina precedente. Questo meccanismo rappresenta lo scheletro dell'interfaccia.

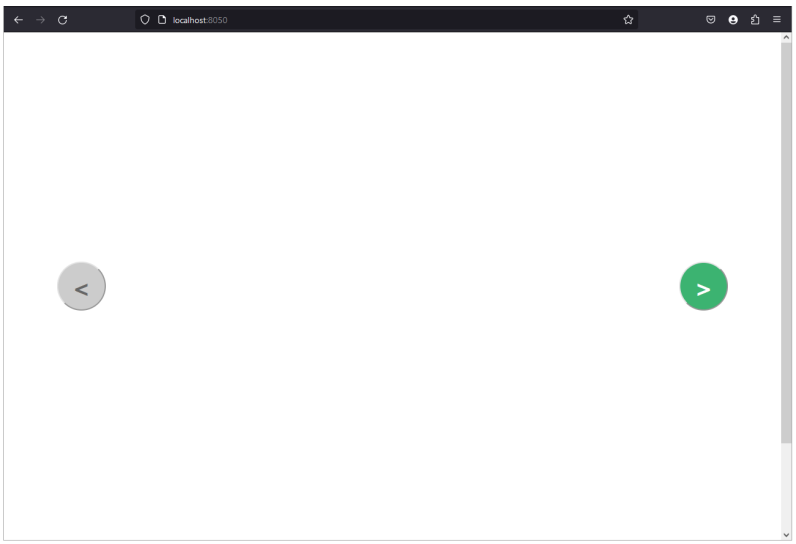


Figura 5.8: Comandi wizard

L'aspetto del sistema viene raffigurato in Figura 5.8, in particolare si possono notare i due bottoni per navigare le pagine come descritto sopra.
Il funzionamento è semplice: viene monitorato l'url, e ad ogni sua modifica, una callback estrae il percorso e verifica se esso combacia con l'attributo `pathname` di una delle view, in caso positivo il layout della view viene mostrato in un elemento contenitore. I pulsanti di navigazione servono a modificano l'url con il percorso corretto. Infine il sistema supporta la funzionalità di disabilitazione dei comandi di navigazione, per esempio se ci sono delle condizioni da superare per andare alla pagina successiva. Tutto ciò è gestito dal presenter `Router`.

Il layout invece è definito nella view `BaseView`. Esso è stata partizionato in 3 sezioni: due laterali per i comandi di navigazione e una sezione centrale che ospita l'elemento contenitore di cui si è parlato in precedenza.

5.4.2 Sviluppo della pagina per la fase di training

Per lo sviluppo della pagina per la fase di training sono state apportate alcune modifiche grafiche e funzionali alla versione progettata inizialmente.

Componente per il caricamento degli event log

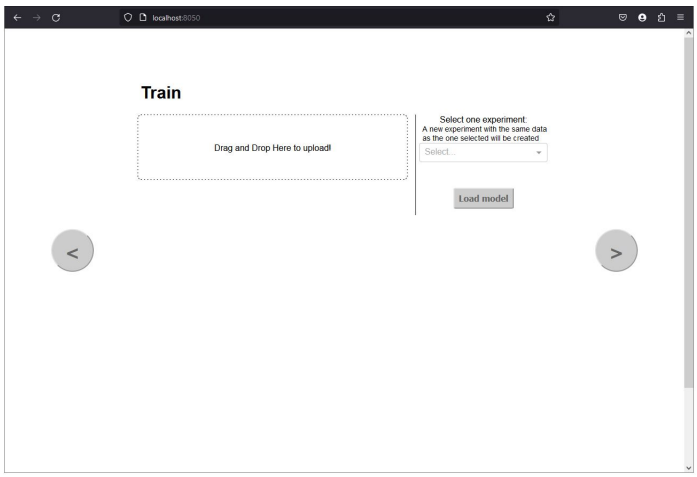


Figura 5.9: Pagina fase di training senza nessun event log caricato

Un'importante modifica riguarda il componente drag-and-drop[G] per caricare gli event log. Inizialmente si era pensato di utilizzare il componente `Upload` offerto da Dash nei suoi componenti base. Ma questo componente aveva una grossa restrizione: pur essendo in teoria capace di gestire file di dimensioni arbitrariamente grandi, nella pratica la dimensione massima dei file gestibili era limitata a circa 150MB mentre la dimensione consigliata era intorno ai 10MB [20].
Questo è dovuto al funzionamento del componente che prima converte il file in una stringa base64[G] caricandolo nella memoria locale del browser e, successivamente, carica la stringa generata nella memoria permanente del server. Se il file da caricare è troppo grande la saturazione della memoria locale porta in browser a stalli e/o terminazioni improvvise.
Si è quindi deciso di utilizzare il componente `dash-uploader` [8], sviluppato e mantenuto dalla community. Esso infatti risolve la restrizione riscontrata poiché esegue il caricamento del file a blocchi di dimensioni ridotte direttamente sulla memoria permanente del server. L'unica limitazione di dimensioni diventa quindi la dimensione del disco rigido del server.

Inserimento delle opzioni di training

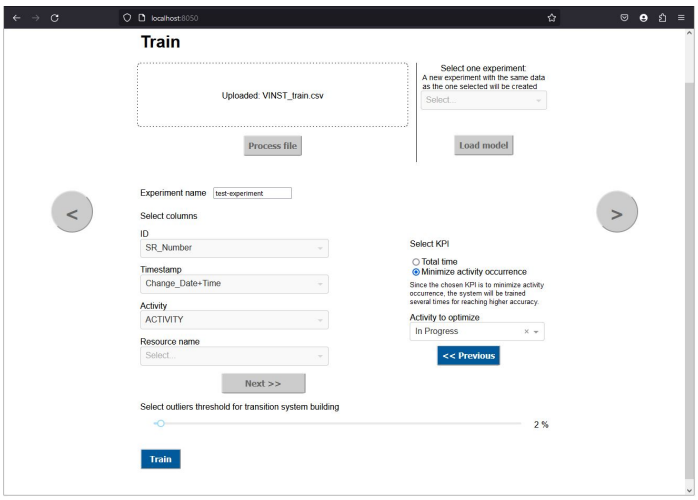


Figura 5.10: Pagina fase di training con un event log caricato e tutte le opzioni di training inserite

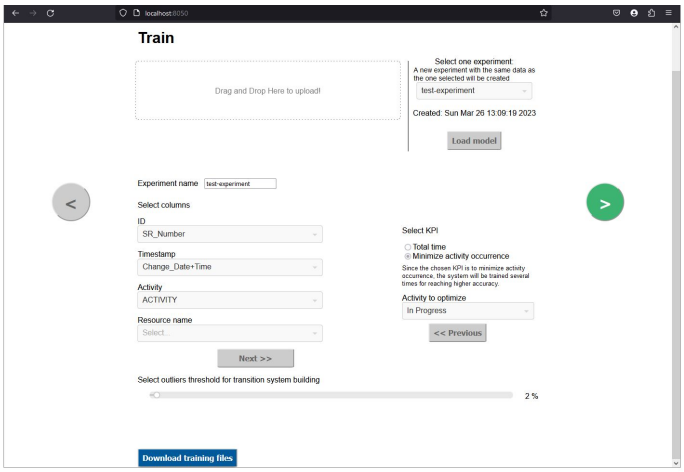


Figura 5.11: Pagina fase di training con caricato un process model già allenato

L'interfaccia responsabile per l'inserimento delle opzioni di training ha subito alcuni cambiamenti. Vista la necessità di seguire il flusso di esecuzione descritto nella sottosezione 5.1.1 (in particolare i passi 2 e 3 del flusso principale), sono stati inseriti dei bottoni aggiuntivi che permettono di cambiare "fase" di compilazione solo dopo aver completato quella corrente. Fintanto che una fase non è accessibile tutti i componenti che la costituiscono rimangono disabilitati, come si può notare in Figura 5.10.
Questo si è reso necessario per facilitare la compilazione, per ridurne gli errori e per permettere all'interfaccia di conoscere la colonna corrispondente alla feature "activity" così da poter elencare le attività corrette nel dropdown[G] "Activity to optimize" nel caso in cui si sia scelto il KPI "Minimize activity occurrence".

Un altro cambiamento da considerare riguarda quei casi di utilizzo in cui i controlli vengono usati per la sola visualizzazione delle opzioni. Questo accade quando l'utente usa un event log in formato XES, dove le feature sono già definite implicitamente nella codifica del file, e quando l'utente carica un process model già allenato (come nel caso in Figura 5.11), per il quale tutte le opzioni sono già state specificate in un momento precedente. In questi casi, i controlli le cui le opzioni sono già determinate rimangono *read-only*, cioè non modificabili.

Visualizzazione del progresso del processo di training

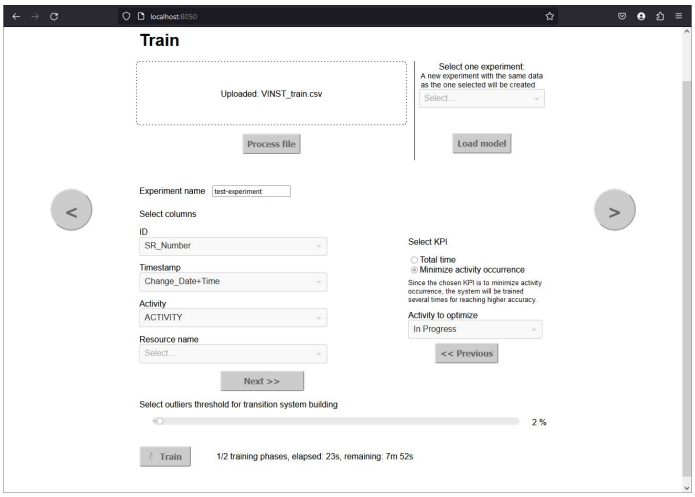


Figura 5.12: Pagina fase di training con il processo di training avviato

Innanzitutto va precisato che il processo di training è suddiviso in 3 fasi:

- Preparazione dell'event log per la fase di training;
- Processo di training effettivo;
- Generazione di variabili qualitative e quantitative.

L'interesse principale viene posto nella seconda fase, che è quella che richiede più tempo. Per questo, il tempo impiegato dalla prima e dalla terza fase può essere trascurato.
La seconda fase, il training effettivo, è basato sulla funzione `fit` di della libreria Catboost [2]. L'unica possibilità per avere una corretta indicazione del progresso del processo di training era l'utilizzo del parametro `logging_level="Verbose"` offerto dalla funzione.
Il parametro stampa nell'output standard varie informazioni relative al training. Quelle utilizzate per tracciare il progresso del training sono: l'indicazione di quanto tempo è passato dall'avvio delle funzione e l'indicazione, stimata, del tempo rimanente al termine della funzione.
L'idea iniziale per la visualizzazione del progresso del processo di training era l'utilizzo di una barra di caricamento. Però è chiaro come queste informazioni temporali, stimate, avrebbero reso un'eventuale barra di caricamento molto instabile, dato che fluttuazioni dei tempi indicati avrebbero portato ad improvvisi spostamenti in avanti o indietro

della barra stessa. Si è quindi optato per un'indicazione testuale delle informazioni temporali che si aggiorna ad intervalli regolari (come si pò vedere in Figura 5.12).
Per poter utilizzare queste informazioni era necessario reindirizzare l'output su un file temporaneo. Per fare questo la funzione `fit` offriva già un altro parametro apposito, `log_cout`. Specificando il nome del file temporaneo la funzione avrebbe stampato le informazioni al suo interno e questo avrebbe permesso all'interfaccia di accedervi in maniera più comoda.
L'accesso al suddetto file è mediato dalla classe `TrainProgLogger`: ad intervalli regolari il sistema preleva l'informazione grezza più recente dal file (interfacciandosi con `TrainProgLogger`), questa poi viene analizzata e ripulita e infine viene mostrata nell'interfaccia grafica.

5.4.3 Sviluppo della pagina per la fase di runtime

Lo sviluppo della pagina di runtime non ha richiesto particolari modifiche di carattere grafico alla versione progettata inizialmente.
Ad ogni modo, visto che la pagina necessitava del caricamento di un'event log tramite un componente drag-and-drop[G], è stato usato di nuovo il componente `dash-uploader`, facendo le stesse considerazioni fatte in precedenza (vedi paragrafo Componente per il caricamento degli event log).

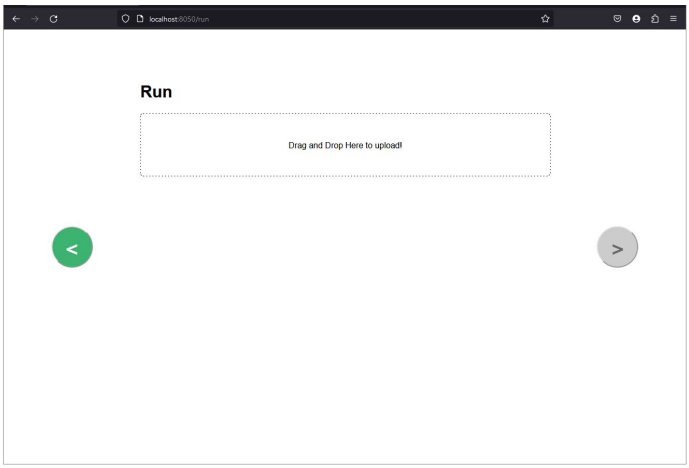


Figura 5.13: Pagina fase di runtime senza nessun event log caricato

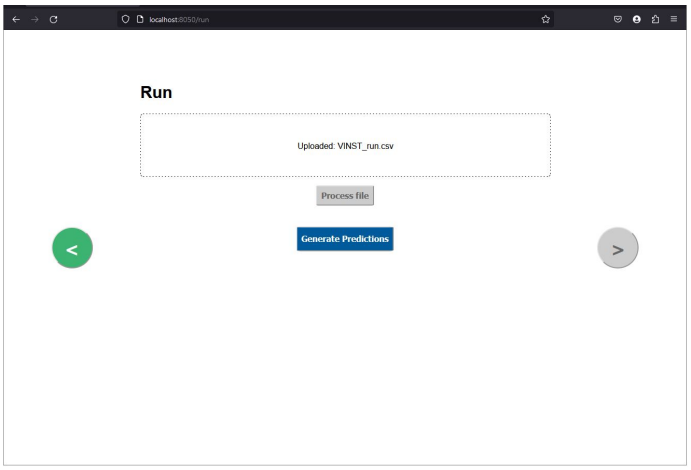


Figura 5.14: Pagina fase di runtime con un event log caricato

Visualizzazione del progresso del processo di generazione delle raccomandazioni

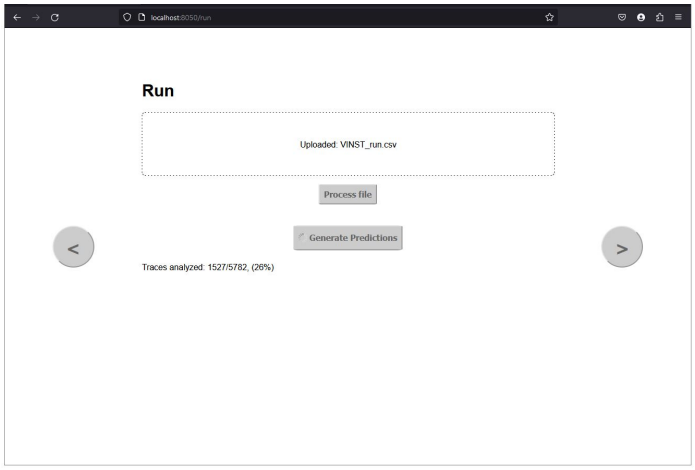


Figura 5.15: Pagina fase di runtime con il processo di generazione delle raccomandazioni avviato

In questo caso il processo di generazione delle raccomandazioni itera sulle tracce dell'event log caricato ed è implementato direttamente nel back-end del sistema, quindi una barra di progresso era di facile implementazione.
Però, per evitare di cambiare modalità di visualizzazione rispetto alla pagina precedente, si è deciso di utilizzare ancora l'indicazione testuale del progresso.
Per far questo è stato utilizzato un principio simile a quanto descritto nel paragrafo Visualizzazione del progresso del processo di training. Viene sfruttato l'output della libreria `tqdm` [19] applicata all'iterazione principale, che è poi reindirizzato su un file temporaneo. Quest'ultimo viene letto ad intervalli periodici acquisendo le informazioni grezze più recenti che vengono filtrate. Vengono estratti i dati relativi al progresso in percentuale, alle tracce analizzate e alle tracce totali. Infine le informazioni vengono

riscritte in un formato più leggibile e mostrate nell'interfaccia grafica (come in Figura 5.15).
La classe `RunProgLogger` si occupa dell'accesso al file temporaneo e della manipolazione dell'informazione grezza.

5.4.4 Sviluppo della pagina per la fase di visualizzazione

Anche per lo sviluppo della pagina di visualizzazione sono state necessarie alcune modifiche, sia grafiche che funzionali, rispetto a quanto progettato.

Paginazione per il grafico di visualizzazione delle raccomandazioni

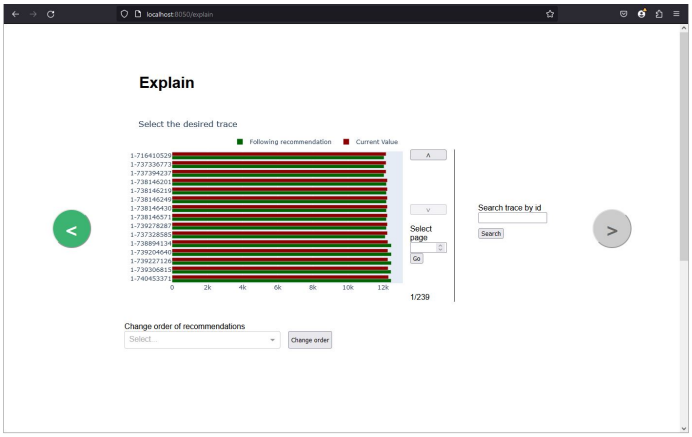


Figura 5.16: Pagina fase di visualizzazione senza nessuna raccomandazione selezionata

Durante la progettazione non si è considerato il problema che riguardava la visualizzazione di molte raccomandazioni nel grafico. Infatti esso diventava illeggibile anche andando oltre le 20 raccomandazioni visualizzate.
Considerando quindi che il numero tipico di raccomandazioni sarebbe stato di gran lunga maggiore e che le dimensioni del grafico dovevano rimanere quelle decise in sede di progettazione, la soluzione pensata richiedeva la necessità di paginare il grafico.
Il dataset contenente l'insieme delle raccomandazioni viene suddiviso in porzioni di dimensione fissata a 15 e queste porzioni, o pagine, vengono visualizzate nel grafico una alla volta. Per la navigazione delle pagine sono stati sviluppati alcuni comandi aggiuntivi (possono essere visualizzati nella Figura 5.16):

- Un indicatore della pagina attuale e del numero totale di pagine;
- Due pulsanti per passare alla pagine successiva o precedente;
- Un pulsante per andare ad una specifica pagina dopo averne specificato il numero.

In questo modo è stata resa possibile la visualizzazione di un numero di raccomandazioni arbitrariamente grande.

Miglioramento della visualizzazione testuale delle raccomandazioni

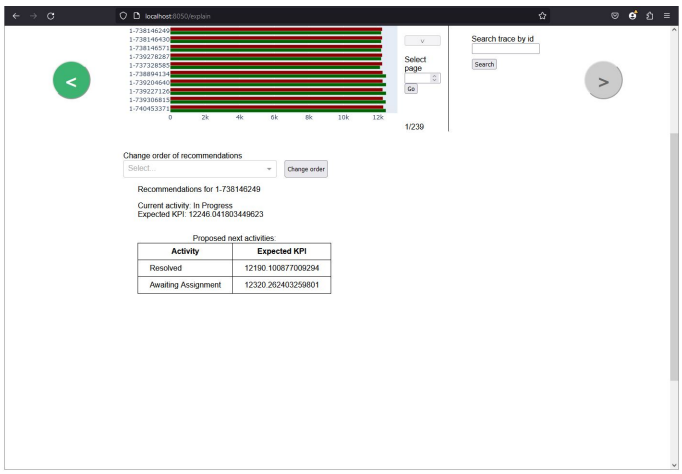


Figura 5.17: Pagina fase di visualizzazione con raccomandazione selezionata

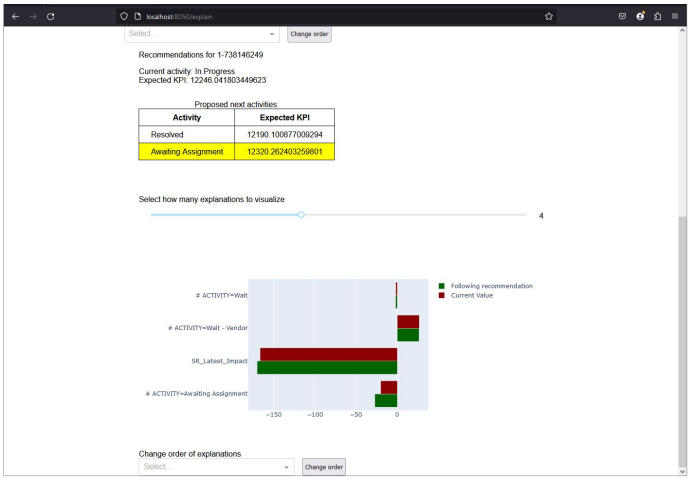


Figura 5.18: Pagina fase di visualizzazione con spiegazione generate

Come visto nella progettazione il sistema genera un massimo di 3 raccomandazioni per traccia, la migliore viene mostrata nel grafico e il resto può essere visionato, in forma testuale, selezionando la traccia. Si è deciso di aggiungere altri dati, per migliorare la comprensione da parte dell'utente. Essi sono:

- L'identificativo della traccia selezionata;
- L'attività attuale, ovvero l'attività a cui è arrivata la traccia nell'event log caricato nella fase di runtime;
- Il KPI medio predetto, inteso come il KPI che ci si aspetta al completamento della traccia senza considerare le raccomandazioni, quindi il KPI da migliorare.

5.4.5 Utilizzo delle background callback di Dash

Dash offre un'interessante funzionalità per le callback le cui computazioni mediamente superano i 10/20 secondi. Esse infatti possono causare errori improvvisi e comportamenti non definiti determinati dal periodo di timeout imposto alle richieste HTTP. Può succedere infatti che, alla scadenza del timeout, la callback fallisca silenziosamente l'esecuzione.
Da notare come questo periodo di timeout dipenda dalla macchina su cui si esegue il back-end, quindi non c'è un modo sicuro e indipendente dalla piattaforma di gestire questo parametro. Inoltre è sconsigliato aumentarlo, poiché potrebbe portare ad un esaurimento delle risorse disponibili e mandare in stallo l'applicazione.
Per rimediare a questo Dash offre il meccanismo delle background callback [3]. Il funzionamento è semplice: la richiesta di esecuzione della callback viene inserita in una coda. Quando arriva il suo turno, viene fatta eseguire su un processo secondario e periodimente viene inviata una richiesta per controllare se ha terminato. Questo permette di evitare il problema del timeout.

I processi computazionali, che possono richiedere una considerevole quantità di tempo, e che quindi necessitano di queste particolari callback sono:

- L'analisi e l'elaborazione al caricamento dell'event log per la fase di training;
- Il processo di training;
- L'analisi e l'elaborazione al caricamento dell'event log per la fase di runtime;
- Il processo di generazione delle raccomandazioni;
- Il processo di generazione delle spiegazioni.

Le background callback possono richiedere una breve attesa prima di poter essere eseguite, visto che esse sono poste in una coda. Ad esempio, che se l'utente preme un pulsante che avvia una background callback, ci possono volere alcuni secondi perché l'interfaccia risponda. Questo può causare confusione all'utente, spingendolo a premere multiple volte il pulsante.
Per evitare questa situazione è stata aggiunta un'ulteriore funzionalità, per ognuno di questi bottoni che avviano una background callback: la disabilitazione del comando stesso e la comparsa di un "indicatore di attesa". Un esempio è il pulsante "Train" in Figura 5.15.

5.4.6 Sviluppo delle funzionalità multiutente

L'idea iniziale per l'interfaccia era che essa fosse utilizzabile solo da un singolo utente, che avrebbe installato ed utilizzato il sistema in maniera autonoma. Per questa ragione il sistema è stato progettato e sviluppato interamente secondo una prospettiva a singolo utente.
Solo in un momento successivo, si è considerata la possibilità di estendere il sistema per renderlo multiutente, pur considerando l'aggiunta di questa nuova funzionalità desiderabile e non obbligatoria (vedi sezione 2.3).
L'obiettivo era quindi andare a trasformare un sistema a singolo utente, già funzionante, in multiutente, riducendo al minimo le modifiche al codice già scritto. Questo si è reso necessario per evitare di dover riprogettare e re-implementare il sistema per intero, dato che non era compatibile con i tempi prefissati di durata dello stage.

Il problema principale

Il framework Dash offre nativamente la capacità di supportare utenti multipli, ad una condizione: la componente server deve essere *stateless*. Questo significa che l'applicazione non deve salvare dati generati dall'utente, dati di sessioni precedenti o dati di computazioni precedenti.
Come spiegato nella documentazione [18], Dash scala in maniera orizzontale, quindi, per poter gestire più utenti, vengono eseguite più istanze della stessa applicazione in più processi paralleli. Da notare come non ci sia una associazione uno a uno tra una sessione utente e un processo, perciò ogni richiesta di esecuzione di una callback può essere gestita da uno qualsiasi dei processi disponibili.

Consideriamo la situazione in cui due utenti diversi eseguono la stessa operazione. Per far questo vengono avviate due esecuzioni diverse della stessa callback. Assumiamo che esse opereranno su due processi diversi che condividono la stessa memoria.
Assumiamo anche che l'applicazione non sia stateless e mantenga uno stato globale. Se questa callback va a modificare un dato mantenuto nello stato globale, si verrebbe a causare una *race conditions*, ovvero una situazione in cui il dato modificato è stato corrotto. Infatti, poiché il dato è globale, esso conterrà il risultato della callback che per ultima ha terminato di eseguire, sovrascrivendo tutti i risultati delle callback precedenti.
Riprendendo la situazione considerata prima, ora l'utente la cui callback ha finito per prima avrà un dato sbagliato, che renderà le successive computazioni che lo riguardano errate a loro volta.
Chiaramente questa situazione deve essere evitata, tanto che, generalmente, i server che eseguono il back-end di un'applicazione Dash usano processi che non condividono la memoria, così da evitare il problema delle race conditions. L'utilizzo di processi con memoria condivisa per un'applicazione Dash è fortemente sconsigliato, sopratutto se l'applicazione deve gestire utenti multipli.
Il sistema sviluppato però non era stateless, in quanto c'era la necessità di persistere i dati del Model. Questo fatto inizialmente non aveva posto nessun problema, visto che l'applicazione era a singolo utente.

Le soluzioni applicate

Di seguito le modifiche principali che hanno permesso all'applicazione di gestire multipli utenti, evitando di riprogettare il sistema.

Salvataggio dello stato condiviso su disco Per il corretto funzionamento del sistema è necessario mantenere, per tutta la durata dell'utilizzo dell'applicazione da parte di un utente, i dati relativi alle istanze delle classi del Model. Basti pensare ad esempio alle informazioni relative al process model, generate dalla classe `Trainer` ma usato anche dalle classi `Recommender` e `Explainer` (vedi sottosezione 5.3.2).
Similmente a come opera Dash, è stato deciso di scalare alcune parti del sistema in orizzontale.
Infatti è presente un'associazione uno a uno tra le istanze delle classi del Model e un singolo utente, quindi idealmente basterebbe modificare il sistema per permettere la gestione di una molteplicità di istanze delle classi del Model per rendere possibile la gestione di utente multipli. Va assicurato però che sia possibile determinare univocamente a quale istanza è associato un utente.
Per fare ciò si è deciso di assegnare, ad ogni nuovo utente che si connette al sistema,

un identificativo alfanumerico univoco, generato casualmente. Successivamente tutti i riferimenti alle classi del Model presenti nei presenters (vedi sottosezione 5.3.3) sono stati convertiti da singoli riferimenti a collezioni di riferimenti indicizzate tramite identificativo utente, in modo da associare in maniera univoca un particolare utente al riferimento dell'istanza di classe corretto, e permettere di gestire istanze multiple.
Infine è bastato aggiungere, nelle callback che lo necessitavano, un parametro aggiuntivo per l'identificativo utente, così da permettere di utilizzare, ad ogni esecuzione, l'istanza di classe corretta, indipendentemente dal processo che la esegue.

Il meccanismo sopra descritto però funziona solo se la memoria tra processi è condivisa, cosa che, nell'ambito delle applicazione Dash, è assolutamente sconsigliata. Utilizzando processi con memoria separata infatti, le modifiche effettuate da un processo allo stato globale, rimangono nello spazio di memoria del processo e quindi processi differenti che dovranno usare la stessa informazione, la troveranno non aggiornata.
La soluzione a questo problema è stato rendere le classi del Model JSON-serializzabili, cioè permettere ad un'istanza di classe di essere salvata in un supporto di memoria, come file JSON, per poi poter essere caricata nel programma al momento del bisogno. Quindi ognuna di queste classi è stata dotata di un metodo `to_dict` che converte la classe in un dizionario Python [10] che a sua volta viene scritto su disco come file in formato JSON. Essendo la memoria disco una risorsa condivisa tra i processi, ora è possibile mantenere stato condiviso tra più processi. Inoltre la relazione uno a uno tra utente e istanza di classe del Model risolve il problema delle race conditions tra utenti diversi.
Per la gestione della serializzazione e deserializzazione è stata sviluppata una classe aggiuntiva nel Model, chiamata `DiskDict`. Essa viene utilizzata nei presenter e gestisce le collezioni di riferimenti delle classi alle classi del Model. Questo spiega il tipo `DiskDict` negli attributi in Figura 5.7.

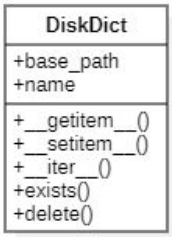


Figura 5.19: Diagramma classe DiskDict

La classe `DiskDict` viene sfruttata per poter accedere e modificare le istanze delle classi, utilizzando la stessa notazione usata per i dizionari in Python. Attraverso l'identificativo utente infatti, la callback interessata può accedere al file in memoria e `DiskDict` si occuperà del recupero e della ricostruzione dell'oggetto, che viene poi restituito per poter essere utilizzato.

Pulizia dei file inutilizzati Il salvataggio dello stato condiviso su disco portava alla creazione di parecchi file. Quando l'utente si disconnette dal sistema, tutti questi file diventano inutili e vanno eliminati, per evitare lo speco di memoria.
Per capire quando un utente si disconnette, il server mantiene una mappa, salvata su disco, che contiene tutti gli identificativi utente attuali con i relativi timestamp.

Ad ogni nuova connessione, il nuovo identificativo utente viene aggiunto alla mappa. Questi timestamp sono poi aggiornati periodicamente, a brevi intervalli, da un timer gestito dal client di ogni utente.
Dopo un intervallo prefissato di tempo, il server avvia un processo secondario, che ha il compito di comparare i timestamp di tutti gli utenti con il timestamp attuale: se la differenza supera una certa soglia predefinita, l'utente viene considerato disconnesso e tutti i file a lui collegati vengono cancellati. Infine l'identificativo viene rimosso dalla mappa gestita dal server.
Questo semplice sistema funziona perché quando l'utente si disconnette, il timer gestito dal client termina di aggiornare la mappa utenti del server. Ovviamente è necessario aggiustare correttamente i vari intervalli e soglie, e questo è stato fatto tramite osservazione empiriche.

Gestire l'unicità dei nomi degli esperimenti e del caricamento dei process models Per rendere univoci i nomi degli esperimenti, che sono salvati dal sistema su disco insieme a tutti i dati generati (process model, raccomandazioni e spiegazioni) si è deciso di aggiungere al nome stesso il timestamp di creazione. Questo risolve il problema e rende univoci i nomi degli esperimenti salvati sul file system del server.
Il nome dell'esperimento però è utilizzato anche per visualizzare i vari process models già allenati, disponibili per essere caricati (vedasi il flusso si esecuzione alternativo descritto nella sottosezione 5.1.2) e, aggiungere il timestamp al nome, può rendere la lettura di quest'ultimo difficile. Per evitare questo, quando il nome dell'esperimento deve essere mostrato all'utente, questo viene diviso dal timestamp e i due dati vengono visualizzati separatamente (esempio in Figura 5.11)

Un ultimo problema individuato riguarda il caricamento dei process models già allenati per saltare la fase di training. Infatti se due utenti selezionano lo stesso process model, questi andranno ad operare sugli stessi file su disco, e l'utente con l'esecuzione che termina per ultima andrà a sovrascrivere i file generati dal primo utente.
La soluzione decisa per questo problema è quella di duplicare i file del process model. Quindi, quando un'utente carica un process model già allenato, i file che compongono quest'ultimo vengono duplicati e il nome dell'esperimento viene aggiornato con il nuovo timestamp. Così facendo viene effettivamente creato un nuovo esperimento, l'unicità dei nomi è mantenuta e ogni utente può operare su una copia separata e personale del process model, evitando qualsiasi rischio di sovrascrittura.

Capitolo 6

Verifica e validazione

6.1 Attività di verifica

6.1.1 Test di unità

Durante l'attività di codifica sono stati sviluppati vari test di unità per verificare il funzionamento del sistema sviluppato. Con test di unità si intende l'attività di collaudo effettuata su una singola unità del software. La definizione di unità è variabile a seconda dei bisogni di progetto, ad esempio: una singola funzione, un singolo componente, una singola classe. Nel caso del progetto in questione, l'unità viene intesa come una singola funzione.
In accordo con il committente si è deciso di testare l'intero componente Presenter, quindi tutte le callback, e le sole parti del componente Model strettamente relative al funzionamento dell'interfaccia grafica. Infatti, il testing dei metodi relativi al sistema di raccomandazione sottostante (i metodi che rappresentano il porting[G] del sistema di raccomandazione originale) è stato considerato al di fuori della portata dello stage. Similmente, è stato deciso di non scrivere test di unità per il componente View. Per questo, essendo un componente passivo e definito in maniera puramente dichiarativa, è bastata la conferma visuale che la componente grafica seguisse quanto progettato nei mock up[G].
La libreria utilizzata per effettuare i test di unità è `dash.testing` [7].

6.1.2 Test di integrazione

I test di integrazione sono quelle attività di collaudo che verificano il funzionamento di più unità, moduli o componenti nel loro insieme. In relazione ai test di unità essi vengono effettuati successivamente e impiegano più risorse (tempo di sviluppo e tempo di esecuzione) per essere svolti, per questo sono meno numerosi.
Per il progetto in questione, i testi di integrazione sono stati sfruttati per verificare l'interazione tra pagine. In particolare sono state verificate le condizioni che permettono, o impediscono, il passaggio da una pagina alla successiva o precedente. Inoltre sono state verificate le condizioni che determinano la disabilitazione o abilitazione dei controlli dell'interfaccia.
In questo caso, per lo sviluppo dei test di integrazione, sono stati usati due strumenti: la libreria `dash.testing` e Selenium WebDriver [17] per l'automazione del browser.

6.1.3 Test di sistema

Infine sono stati sviluppati test di sistema. Essi sono quelle attività di verifica e collaudo effettuate nel sistema completo e hanno lo scopo di valutare e dimostrare la conformità del sistema in relazione ai requisisti definiti.
Nel caso del progetto di stage, essi sono stati effettuati, insieme al committente, facendo due esecuzioni complete su due event log differenti:

- **Bank Account Closure** (BAC): rappresenta un log relativo alla gestione del processo di chiusura di conti bancari di un sistema bancario italiano;
- **VINST**: rappresenta un log usato nella BPI challenge del 2013, fornito dalla Volvo Belgium e contiene eventi relativi ad un sistema di gestione incidenti chiamato VINST.

Gli event log necessari sono stati forniti dal committente. Per una definizione più dettagliata e per la spiegazione di come sono stati generati gli event log per la fase runtime ci si riferisce a [14].

6.2 Validazione

Al termine dell'esperienza di stage è stata verificata la copertura dei requisiti definiti nel Capitolo 4. Essa si è rivelata totale ed il prodotto è stato considerato soddisfacente da parte del committente.

Tipo requisito	Coperti	Totali	Percentuale copertura
Funzionali	41	41	100%
Non Funzionali	7	7	100%

Tabella 6.1: Tabella validazione dei requisiti

Capitolo 7

Conclusioni

7.1 Raggiungimento degli obiettivi

Di seguito viene riportato lo stato di raggiungimento e soddisfazione degli obiettivi definiti nella sezione 2.3.

- Obbligatori
  - *O01*: Soddisfatto;
  - *O02*: Soddisfatto;
  - *O03*: Soddisfatto;
  - *O04*: Soddisfatto;
  - *O05*: Soddisfatto;
  - *O06*: Soddisfatto;
  - *O07*: Soddisfatto;
  - *O08*: Soddisfatto.
- Desiderabili
  - *D01*: Soddisfatto.

7.2 Consuntivo finale

Di seguito viene riportato il consuntivo orario finale del progetto di stage. Verranno evidenziate le differenze rispetto al piano orario preventivato nella sezione 2.4, comparandole al totale di ore che sono state effettivamente svolte. Verranno inoltre espresse le motivazioni per ogni eventuale variazione oraria.

Dalla prima alla quarta settimana		
Attività svolte	**Ore preventivate**	**Ore effettive**
Studio del problema	20	20
Sviluppo mock up[G] e casi d'uso	60	60
Totale	**80**	**80**
Motivazioni variazione	Nessuna variazione	

Dalla quinta alla sesta settimana		
Attività svolte	**Ore preventivate**	**Ore effettive**
Studio del framework per la visualizzazione dati	48	32
Sviluppo PoC[G]	12	8
Totale	**60**	**40**
Motivazioni variazione	Lo studio del framework si è rivelato più semplice del previsto, in quanto, dall'insieme di framework considerati inizialmente (circa 5-6 framework), solo due sono stati considerati utilizzabili, scartando il resto. Similmente lo sviluppo di un PoC[G] ha richiesto meno tempo del previsto data la presenza di numerose fonti informative sul framework scelto	

Settima settimana		
Attività svolte	**Ore preventivate**	**Ore effettive**
Progettazione dell'architettura generale per il sistema	40	20
Totale	**40**	**20**
Motivazioni variazione	Poiché la necessità di sviluppo delle API[G] è venuta meno, la progettazione è stata semplificata	

Dall'ottava all'undicesima settimana		
Attività svolte	**Ore preventivate**	**Ore effettive**
Sviluppo del primo prototipo	60	80
Totale	**60**	**80**
Motivazioni variazione	L'aumento di ore si è ritenuto necessario per lo sviluppo di parte dei cambiamenti descritti nella sezione 5.4. Ad esempio la ricerca di un nuovo componente per il caricamento degli event log o l'uso di background callback	

Dodicesima settimana		
Attività svolte	**Ore preventivate**	**Ore effettive**
Verifica e validazione del primo prototipo	20	20
Totale	**20**	**20**
Motivazioni variazione	Nessuna variazione	

Dalla tredicesima alla quindicesima settimana		
Attività svolte	**Ore preventivate**	**Ore effettive**
Sviluppo del prodotto finale	40	60
Totale	**40**	**60**
Motivazioni variazione	L'aumento di ore rispetto a quanto previsto si è ritenuto necessario per lo sviluppo delle funzionalità multiutente	

Sedicesima settimana		
Attività svolte	**Ore preventivate**	**Ore effettive**
Verifica, validazione e collaudo del prodotto finale	20	20
Totale	**20**	**20**
Motivazioni variazione	Nessuna variazione	

L'attività di stage ha richiesto un totale di 320 ore di lavoro effettive, che è uguale al totale preventivato e rappresenta il limite massimo posto dall'Università di Padova per i tirocini del corso di Informatica.

7.3 Criticità individuate e possibili miglioramenti

Di seguito sono descritte le criticità di maggiore rilevanza individuate nel sistema sviluppato. Saranno anche delineate alcune proposte di soluzione applicabili.

Utilizzo della memoria su disco

Come già descritto nei capitoli precedenti, il sistema fa ampio uso di memoria su disco, sia per mantenere lo stato condiviso che per salvare i dati generati dai processi di training, generazione delle raccomandazioni e spiegazioni.
Mentre i dati dello stato condiviso permangono per un periodo limitato di tempo (sono eliminati secondo la procedura descritta in Pulizia dei file inutilizzati), i dati generati dai processi sono permanenti.
Essi possono, col tempo, saturare la memoria (considerando anche il processo di duplicazione descritto in Gestire l'unicità dei nomi degli esperimenti e del caricamento dei process models). Questo può accadere sopratutto se sono utilizzati event log di grandi dimensioni. Il sistema sviluppato inoltre non sfrutta nessun metodo di compressione per ridurre le dimensioni di questi file.
La soluzione più ovvia sarebbe quindi di introdurre un sistema di compressione e decompressione dei file utilizzati per memorizzare su disco i dati generati.

Gestione dello stato condiviso per file di grandi dimensioni

Parte dello stato condiviso è rappresentata dai dati degli event log già pre-processati. Questo si ritiene necessario, infatti sarebbe un inutile spreco risorse computazionali mantenere solo l'event log ed effettuare la sua elaborazione ad ogni occorrenza.
I file generati per mantenere questi dati tendono ad essere di grandi dimensioni. Il problema sorge durante la scrittura e lettura di questi file, quando le loro informazioni sono richieste per lo svolgimento di determinate azioni nell'interfaccia grafica. Tanto è vero queste operazioni posso richiedere una considerevole quantità di tempo, introducendo stalli e rallentamenti che peggiorano l'esperienza dell'utente.
Una possibile soluzione potrebbe essere quella di sfruttare formati binari al posto dei formati testuali che sono attualmente utilizzati. Andrebbe quindi investito del tempo per ricercare quali possano essere i migliori formati binari da utilizzare, bilanciando, nella scelta, l'effettiva riduzione dei tempi di lettura e scrittura con le dimensioni finali dei file generati.

Metodo per la generazione di nomi di esperimenti univoci

L'ultima criticità riguarda il metodo di generazione dei nomi di esperimento univoci spiegato in Gestire l'unicità dei nomi degli esperimenti e del caricamento dei process models. In particolare non viene controllata l'unicità nel nome, dato che ad esso viene poi aggiunto un timestamp. Questo può portare alla generazione di una grande quantità di esperimenti che hanno tutti lo stesso nome, il che rende più difficile la selezione di uno specifico process model se non se ne conosce il timestamp.
Ci sono vari metodi per introdurre un controllo sull'unicità dei nomi degli esperimenti, ad esempio mantenere una tabella hash in memoria con tutti i nomi attuali. Alla generazione di un nuovo esperimento la tabella viene controllata e, in caso di collisione, viene chiesto all'utente di selezionare un nuovo nome per l'esperimento.

7.4 Sviluppi futuri

Una nuova e interessante funzionalità potrebbe essere l'implementazione un sistema di login. In questo modo si possono collegare gli esperimenti all'utente che gli ha creati. Questo permetterebbe anche di definire un sistema di privilegi e permessi, che si può evolvere, ad esempio, in un sistema di condivisione di process model già allenati (per saltare la fase di training) oppure nella possibilità di limitare la visibilità di determinati esperimenti.
Successivamente il sistema potrebbe essere spostato nel cloud, usando servizi di archiviazione come Amazon S3 [1]. Così facendo è possibile incrementare lo spazio di memorizzazione in maniera significativa, mantenendo dei costi ridotti. Infine si può sviluppare un sistema di *client-side encryption*, ovvero un sistema di cifratura da parte del client. Questo permetterebbe di caricare dati sensibili o privati (ad esempio event log aziendali) in un servizio di archiviazione cloud come Amazon S3, assicurandosi che questi restino inaccessibili anche per il provider.

7.5 Valutazione personale

Con il termine dello stage, ritengo di essere soddisfatto del lavoro svolto. Questa esperienza mi ha permesso di apprendere nuovi concetti relativi al Process Mining. Inoltre, ho avuto la possibilità di applicare molti dei concetti imparati nel corso di Ingegneria del Software.
Ho avuto anche la possibilità di lavorare a stretto contatto con professori e studenti di dottorato, e questo mi ha fornito una nuova prospettiva sull'ambiente accademico che altrimenti non avrei potuto ottenere.
Valuto quindi l'esperienza di stage in maniera molto positiva, dato che è stata soddisfacente e formativa.

Glossario

API In informatica con il termine *Application Programming Interface API* (ing. interfaccia di programmazione di un'applicazione) si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'hardware e il programmatore o tra software a basso e quello ad alto livello semplificando così il lavoro di programmazione. 8, 9, 12, 15, 16, 66

base64 Base64 è un sistema di codifica-decodifica che permette di rappresentare file binari (o anche file di testo) in un formato che usa una base di soli 64 caratteri scelti tra i 128 caratteri dell'ASCII standard 7 bit. 52

design pattern Un Design Pattern rappresenta una soluzione progettuale generale ad un problema ricorrente incontrato durante la fase di progettazione o sviluppo di un sistema software. Esso astrae e identifica gli aspetti principali della struttura utilizzata per la soluzione del problema. 44

drag-and-drop In informatica, il termine drag-and-drop si riferisce a 3 azione successive: la selezione di un oggetto virtuale, il trascinamento di questo in un altra posizione e il suo rilascio. Rappresenta un modo alternativo e grafico di selezionare elementi come icone o file. 40, 41, 52, 55

dropdown Un dropdown, o lista a cascata, è un controllo grafico che permette la selezione di un valore da una lista predefinita. 40, 41, 53

ERP ERP è l'acronimo di Enterprise Resource Planning e ci si riferisce a un tipo di software che le organizzazioni utilizzano per gestire le attività quotidiane di business, come ad esempio contabilità, project management, gestione del rischio e operazioni relative all'approvvigionamento. I sistemi ERP mettono in relazione tra loro un insieme di processi di business e ne consentono lo scambio di dati. Grazie alla raccolta di dati transazionali condivisi provenienti da diverse fonti dell'organizzazione, i sistemi ERP eliminano la duplicazione dei dati e ne garantiscono l'integrità tramite un'unica fonte di informazioni. 1

mock up Un mock up è uno strumento utilizzato per lo sviluppo del design di un nuovo prodotto. Rappresenta un modello, una realizzazione a scopo illustrativo di un oggetto o un sistema, senza le complete funzioni dell'originale. 39, 63, 65

PoC Un PoC, acronimo di *Proof of Concept* ("prova di concetto") non è altro che un test o prova (ad esempio in ambiente informatico può essere un piccolo programma) che ha lo scopo di determinare la fattibilità di un'idea o di verificare che l'idea funzionerà come previsto. 16, 66

porting Con porting si intende l'attività di trasposizione, con o senza modifiche, di un componente software o parti di esso per poterlo usare in una piattaforma diversa da quella originale. 63

slider Uno slider è un controllo grafico che permette all'utente di selezionare un valore scorrendo un cursore, in una barra orizzontale o verticale, su di un intervallo prefissato. 41, 42

UML In ingegneria del software *UML, Unified Modeling Language* (ing. linguaggio di modellazione unificato) è un linguaggio di modellazione e specifica basato sul paradigma object-oriented. L'*UML* svolge un'importantissima funzione di "lingua franca" nella comunità della progettazione e programmazione a oggetti. Gran parte della letteratura di settore usa tale linguaggio per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico. 12, 17, 45

Bibliografia

[1] *Amazon S3*. URL: https://aws.amazon.com/it/pm/serv-s3/ (cit. a p. 68).

[2] *CatBoost fit*. URL: https://catboost.ai/en/docs/concepts/python-reference_catboost_fit (cit. a p. 54).

[3] *Dash Background Callbacks*. URL: https://dash.plotly.com/background-callbacks (cit. a p. 59).

[4] *Dash Basic Callbacks*. URL: https://dash.plotly.com/basic-callbacks.

[5] *Dash Core Components*. URL: https://dash.plotly.com/dash-core-components (cit. a p. 12).

[6] *Dash Extensions*. URL: https://www.dash-extensions.com/getting_started/enrich (cit. a p. 16).

[7] *Dash Testing*. URL: https://dash.plotly.com/testing (cit. a p. 63).

[8] *dash-uploader*. URL: https://github.com/np-8/dash-uploader (cit. a p. 52).

[9] Chiara Di Francescomarino e Chiara Ghidini. *Predictive Process Monitoring*. A cura di Wil M. P. van der Aalst e Josep Carmona. Cham: Springer International Publishing, 2022, pp. 320–346. ISBN: 978-3-031-08848-3. DOI: 10.1007/978-3-031-08848-3_10. URL: https://doi.org/10.1007/978-3-031-08848-3_10 (cit. alle pp. 3, 4).

[10] *Dizionari Python*. URL: https://docs.python.org/3/tutorial/datastructures.html#dictionaries (cit. a p. 61).

[11] *Immagine Event log*. URL: https://www.processmining.org/event-data.html#:~:text=Process%20mining%20assumes%20the%20existence,a%20trace%2Fsequence%20of%20events. (cit. a p. 2).

[12] *Immagine MVP*. URL: https://support.touchgfx.com/4.19/docs/development/ui-development/software-architecture/model-view-presenter-design-pattern (cit. a p. 44).

[13] *Immagine tecniche P.M.* URL: https://www.workfellow.ai/guides/process-mining-101 (cit. a p. 3).

[14] Alessandro Padella et al. «Explainable Process Prescriptive Analytics». In: *2022 4th International Conference on Process Mining (ICPM)*. 2022, pp. 16–23. DOI: 10.1109/ICPM57379.2022.9980535 (cit. alle pp. 4, 7, 64).

[15] *Panel APIs*. URL: https://panel.holoviz.org/user_guide/APIs.html (cit. a p. 15).

[16] *Plotly Community Forum*. URL: https://community.plotly.com/ (cit. a p. 16).

[17] *Selenium WebDriver*. URL: https://www.selenium.dev/documentation/webdriver/ (cit. a p. 63).

[18] *Sharing Data Between Callbacks - Dash*. URL: https://dash.plotly.com/sharing-data-between-callbacks (cit. a p. 60).

[19] *tqdm*. URL: https://github.com/tqdm/tqdm (cit. a p. 56).

[20] *Upload limitazioni dimensioni*. URL: https://github.com/plotly/dash-docs/pull/1135 (cit. a p. 52).

[21] *Wikipedia*. URL: https://it.wikipedia.org/wiki/Processo_aziendale (cit. a p. 1).

[22] *Wikipedia*. URL: https://it.wikipedia.org/wiki/Process_mining (cit. a p. 1).