Immagine che contiene testo, Carattere, logo, simbolo

Descrizione generata automaticamente

**TESI DI LAUREA**

**Gestione file JSON per interoperabilità tra sistemi eterogenei**

Relatore:

**Chiar.mo Prof. Paolo BUONO**

Laureando:

**Domenico Romanazzi**

Sommario

[1. Interazione tra sistemi informativi 4](#_Toc168159716)

[1.1. Sistemi Informativi odierni 5](#_Toc168159717)

[1.2. Scambio di dati nei sistemi informativi 9](#_Toc168159718)

[1.3. Sistema informativo orientato ai servizi 11](#_Toc168159719)

[1.4. Esempi di sistemi informativi 13](#_Toc168159720)

[2. Parsing 16](#_Toc168159721)

[2.1. JSON 22](#_Toc168159722)

[2.2. Html 28](#_Toc168159723)

[2.3. Javascript 29](#_Toc168159724)

[2.4. Dart 32](#_Toc168159725)

[2.5. Flutter 34](#_Toc168159726)

[2.6. Low code e No code 37](#_Toc168159727)

[3. Visualizzazione 42](#_Toc168159728)

[3.1. Knime 43](#_Toc168159729)

[3.2. Visual Studio Code 48](#_Toc168159730)

[3.3. Firebase 50](#_Toc168159731)

[3.4. Git e GitHub 54](#_Toc168159732)

[3.5. D3 e SVG 57](#_Toc168159733)

[4. Lavoro svolto 59](#_Toc168159734)

[4.1. Web App 59](#_Toc168159735)

[4.2. Modello di Knime 68](#_Toc168159736)

[Conclusioni e sviluppi futuri 75](#_Toc168159737)

[Ringraziamenti 76](#_Toc168159738)

# Interazione tra sistemi informativi

Per secoli, l'informazione è stata un fondamento essenziale della società umana, trasmettendo conoscenza tramite tradizioni, cultura e moderni sistemi di comunicazione. Questa importanza dell'informazione si estende ai contesti aziendali, dove diventa cruciale.

Nelle imprese, l'informazione è al centro delle attività quotidiane e strategiche, consentendo di adattarsi ai cambiamenti del mercato, prendere decisioni informate e mantenere un vantaggio competitivo in un mondo globalizzato.

Investire nella gestione dell'informazione è divenuto fondamentale per il successo aziendale. Le metodologie di pianificazione e progettazione delle tecnologie dell'informazione si basano su ricerche in settori non tecnologici, spesso considerando variabili non tecniche per migliorare l'efficienza e la competitività. Questo sottolinea l'importanza delle interazioni tra sistemi informativi e discipline gestionali.

Le tecnologie informatiche costituiscono la base del sistema informatico aziendale, comprendendo hardware, software e risorse umane.

Questo sistema gestisce la raccolta, l'elaborazione, l'archiviazione e la distribuzione delle informazioni, supportando le decisioni aziendali e garantendo la sicurezza dei dati.

Tra le tecnologie, i software di gestione dei database sono cruciali per la raccolta e l'organizzazione dei dati aziendali, mentre i sistemi operativi assicurano la stabilità e la compatibilità delle applicazioni aziendali.

Gli strumenti di comunicazione e sicurezza informatica proteggono le informazioni e automatizzano i processi, migliorando la produttività e facilitando la collaborazione tra dipendenti.

Le tecnologie informatiche continuano a evolversi e a innovarsi, offrendo alle aziende strumenti sempre più potenti e versatili per affrontare le sfide del mondo di oggi. Questo permette loro di adattarsi rapidamente alle varie esigenze del mercato e di rimanere competitive.

Inoltre, investire nell'aggiornamento e nell'ottimizzazione di queste tecnologie è fondamentale per garantire la sostenibilità e il successo a lungo termine dell'azienda nel panorama aziendale in continua evoluzione.

Per comprendere meglio il funzionamento di queste tecnologie nel contesto aziendale, è utile considerare il concetto di sistema informativo. Un sistema informativo non è solo una raccolta di hardware e software, ma piuttosto una complessa rete di risorse umane, materiali e procedurali, creato appositamente per produrre e scambiare le informazioni necessarie al funzionamento e alla gestione di un'organizzazione.

Questo sistema può essere paragonato a una sorta di struttura fondamentale per la gestione dell’informazione, che consente all'organizzazione di operare in modo efficace ed efficiente.

Ogni organizzazione, consapevolmente o meno, dispone di un proprio sistema informativo che svolge un ruolo cruciale nel facilitare le operazioni quotidiane, supportare le decisioni strategiche e promuovere l'innovazione.

All'interno di un'azienda, è importante individuare esplicitamente il sistema informativo aziendale, composto da un patrimonio di dati, procedure, strumenti, risorse umane e regole organizzative e gestionali. Questo sistema supporta le operazioni e le decisioni aziendali.

Si può distinguere tra sistema informativo operativo, che gestisce le operazioni all'interno dell'azienda, e sistema informativo decisionale, che fornisce informazioni per guidare i processi operativi.

Un sistema informatico è la parte automatizzata di un sistema informativo, dove le informazioni vengono raccolte e trattate digitalmente con strumenti informatici. Un sistema informatico costituisce quindi un sottoinsieme di un sistema informativo.

## Sistemi Informativi odierni

Con l'innovazione tecnologica in costante evoluzione, i mezzi informatici diventano sempre più accessibili, economici e potenti.

Questo ha portato ad una crescente integrazione tra sistema informatico e sistema informativo, riducendo le aree in cui le informazioni vengono prodotte e gestite senza l'uso di tecnologie informatiche.

Questo fenomeno evidenzia l'importanza di adottare soluzioni informatiche avanzate per ottimizzare la gestione dell'informazione aziendale e migliorare le performance operative.

La sinergia tra sistema informatico e sistema informativo rappresenta quindi una tendenza imprescindibile nel panorama aziendale contemporaneo, con l'obiettivo di ottenere un vantaggio competitivo attraverso una gestione più efficiente e una migliore fruizione delle informazioni disponibili.

Nei sistemi informativi aziendali moderni, i database costituiscono un pilastro fondamentale per la gestione delle informazioni. Essi rappresentano un deposito centrale in cui vengono raccolti, organizzati e gestiti i dati aziendali provenienti da varie fonti e processi.

Le diverse funzioni aziendali utilizzano il database centrale in modi specifici per supportare le proprie operazioni e obiettivi.

Ad esempio, il reparto vendite può accedere al database per ottenere informazioni dettagliate sui clienti esistenti e potenziali, comprese le loro preferenze, i loro comportamenti di acquisto passati e le comunicazioni precedenti.

Questo accesso ai dati consente al gruppo di vendita di personalizzare le proprie offerte e strategie di vendita, migliorando così le relazioni con i clienti e aumentando le opportunità di conversione.

Il reparto marketing può utilizzare il database per analizzare i dati demografici dei clienti, le tendenze di mercato e l'efficacia delle campagne pubblicitarie.

Con queste informazioni a disposizione, la squadra di marketing può creare campagne più mirate e strategie promozionali, ottimizzando così il ritorno sull'investimento e l'engagement del cliente.

Il reparto conformità si affida al database per monitorare e garantire la conformità alle normative e agli standard del settore.

Attraverso l'accesso ai dati sulle regolamentazioni in materia di servizi finanziari e sui requisiti di conformità, il gruppo di conformità può valutare e mitigare i rischi legali e normativi, garantendo che l'azienda operi in modo etico e conforme alle leggi vigenti.

Il reparto finanziario utilizza il database per monitorare le transazioni finanziarie, analizzare i flussi di cassa e generare report finanziari.

Questi dati sono fondamentali per la gestione della liquidità, la pianificazione finanziaria e il reporting agli stakeholder interni ed esterni.

Inoltre, l'accesso alle informazioni finanziarie aggiornate consente al reparto finanziario di prendere decisioni informate e strategiche per ottimizzare le risorse e massimizzare il rendimento aziendale.

Complessivamente, l'uso del database centrale consente all'azienda di consolidare e gestire in modo efficiente una vasta gamma di dati critici per le proprie operazioni.

La sua affidabilità e sicurezza avanzata garantiscono che le informazioni sensibili siano protette e accessibili solo alle persone autorizzate, contribuendo così a promuovere la collaborazione tra i dipartimenti e a supportare una presa di decisioni più informata e tempestiva.

Ad esempio, la squadra di marketing può estrarre dati demografici e comportamentali dei clienti per sviluppare strategie di targeting più efficaci, mentre la squadra di conformità può consultare normative e regolamenti aggiornati per assicurare la conformità legale delle attività aziendali.

Il sistema informativo aziendale assicura che i dati all'interno del database siano aggiornati in tempo reale e che le informazioni rilevanti vengano condivise tra i vari dipartimenti in modo sicuro e protetto.

Ciò favorisce una comunicazione efficiente e una collaborazione sinergica tra le diverse funzioni aziendali, consentendo una migliore coordinazione e un'implementazione più rapida delle strategie aziendali.

Inoltre, i database sono progettati per supportare una vasta gamma di operazioni e analisi. I manager possono generare report dettagliati, dashboard interattive e analisi predittive basate sui dati del database per monitorare le performance aziendali, identificare tendenze di mercato e prendere decisioni strategiche informate.

I database sono essenziali per il funzionamento efficace dei sistemi informativi aziendali, fornendo un fondamento solido per la gestione delle informazioni aziendali.

Grazie alla loro capacità di raccogliere, organizzare e distribuire dati in modo efficiente e sicuro, i database consentono alle aziende di ottenere un vantaggio competitivo nel mercato odierno sempre più informato e dinamico.

Il formato CSV (Comma-Separated Values) ha un ruolo significativo all'interno del contesto delineato. Pur essendo meno sofisticato rispetto ai database tradizionali, il formato CSV è un'opzione popolare per l'archiviazione e la gestione di dati strutturati, specialmente quando si tratta di condividere informazioni tra diverse applicazioni e sistemi.

Nell'azienda descritta nel testo, i file CSV potrebbero essere utilizzati per una serie di scopi. Ad esempio, potrebbero essere impiegati per esportare e importare dati da e verso il database centrale.

Questo potrebbe essere utile quando si devono condividere informazioni con partner esterni, integrare dati provenienti da diverse fonti o lavorare con applicazioni che richiedono un formato di dati semplice come CSV.

Inoltre, i file CSV potrebbero essere utilizzati come formato di scambio per la trasmissione di dati tra diverse funzioni aziendali.

Ad esempio, il reparto vendite potrebbe generare rapporti di vendita in formato CSV per condividere con il reparto marketing, consentendo loro di analizzare le tendenze delle vendite e adattare le proprie strategie di marketing di conseguenza.

I file CSV potrebbero anche essere utilizzati come parte del processo di conformità, consentendo al reparto conformità di analizzare rapidamente grandi quantità di dati per individuare eventuali discrepanze o violazioni delle normative.

Inoltre, i file CSV potrebbero essere utilizzati come base per l'analisi dei dati da parte del reparto finanziario.

Ad esempio, potrebbero essere estratti dati finanziari dal database centrale e formattati come CSV per l'importazione in fogli di calcolo per l'analisi finanziaria dettagliata.

## Scambio di dati nei sistemi informativi

Anche se i file CSV possono essere utili per la gestione dei dati, soprattutto quando si tratta di dati tabellari semplici, il passaggio a JSON può offrire vantaggi significativi, specialmente in contesti in cui è necessaria una struttura dati più complessa e flessibile.

Nel settore dei servizi finanziari, ad esempio, l'utilizzo di file JSON anziché CSV potrebbe consentire una migliore rappresentazione dei dati clienti, delle transazioni finanziarie e di altre informazioni aziendali.

JSON è in grado di gestire strutture dati più complesse rispetto ai file CSV, consentendo di rappresentare dati annidati e gerarchici in modo più efficace.

Ciò significa che è possibile rappresentare informazioni dettagliate sui clienti, comprese le loro preferenze, le transazioni passate e altre informazioni pertinenti, in un formato più strutturato e organizzato.

Inoltre, il formato JSON è più flessibile rispetto ai CSV e può essere facilmente integrato con moderni sistemi e applicazioni aziendali che utilizzano API REST per la comunicazione dei dati.

Le API REST, o Interfacce di Programmazione delle Applicazioni basate su Rest, sono un insieme di regole e convenzioni che definiscono come le applicazioni comunicano tra loro attraverso Internet.

Queste api offrono un modo standardizzato per le applicazioni di richiedere e scambiare dati con i server attraverso il protocollo HTTP (Hypertext Transfer Protocol), che è il protocollo principale utilizzato per la comunicazione su Internet. Utilizzando metodi HTTP standard come GET, POST, PUT e DELETE, le API REST consentono agli utenti di richiedere e manipolare risorse su server remoti.

Le API REST sono caratterizzate da alcune caratteristiche chiave:

* Stateless (senza stato): Ogni richiesta da un client al server contiene tutte le informazioni necessarie per comprendere e soddisfare la richiesta. Il server non mantiene alcuno stato tra le richieste, il che semplifica il mantenimento e la scalabilità del sistema.
* Interfacce uniformi: Le risorse sono identificate da URI(Uniform Resource Identifier) e sono accessibili tramite metodi standard di http. Questo fornisce un’interfaccia uniforma e prevedibile per l’accesso alle risorse.
* Rappresentazione dei dati: I dati sono rappresentati in un formato ben noto e standard, come JSON o XML. Questo consente una facile comprensione e interoperabilità tra diverse applicazioni.
* Cacheable (cache): Le risposte alle richieste possono essere contrassegnate come cacheable o non-cacheable, consentendo un miglioramento delle prestazioni attraverso la memorizzazione temporanea delle risorse.

Le API REST sono ampiamente utilizzate per consentire a diverse applicazioni di comunicare tra loro in modo efficiente e sicuro su Internet. Consentono una comunicazione efficiente e sicura tra diversi sistemi e applicazioni, consentendo loro di scambiare dati in tempo reale.

Quando si passa da file CSV a JSON, i dati possono essere elaborati e trasformati in formato JSON per essere utilizzati nelle richieste e nelle risposte delle API REST.

Questo permette una maggiore interoperabilità e integrazione tra i vari componenti del sistema informativo aziendale, migliorando l'efficienza operativa e la capacità di rispondere alle esigenze del mercato in tempo reale.

Il passaggio da file CSV a JSON e l'integrazione con API REST possono migliorare significativamente la gestione e lo scambio di dati all'interno di un'azienda nel settore dei servizi finanziari, consentendo una rappresentazione più ricca e flessibile dei dati e una comunicazione più efficace tra i diversi sistemi e applicazioni aziendali.

L'azienda potrebbe avere un'applicazione web per i clienti che consente loro di visualizzare il saldo del conto, effettuare trasferimenti di denaro e monitorare le transazioni. Questa applicazione web interagisce con il sistema informativo aziendale tramite API REST.

Quando un cliente accede al proprio conto tramite l'applicazione web, l'applicazione fa una richiesta API REST al server dell'azienda per ottenere i dati del conto del cliente, come saldo attuale e lista delle transazioni recenti.

Il server risponde alla richiesta API REST inviando i dati richiesti sotto forma di risposta JSON.

Inoltre, l'azienda potrebbe utilizzare un sistema interno per la gestione delle transazioni finanziarie, che è integrato con il sistema di conformità per garantire la conformità alle normative del settore. Questi due sistemi comunicano tra loro tramite API REST.

Quando viene effettuata una nuova transazione, il sistema di gestione delle transazioni invia una richiesta API REST al sistema di conformità per verificarne la conformità alle normative. Il sistema di conformità risponde alla richiesta API REST con un'approvazione o un rifiuto basato sulla conformità della transazione.

Inoltre, l'azienda potrebbe utilizzare API REST per integrare il proprio sistema informativo con fornitori di servizi esterni, come piattaforme di pagamento online o servizi di analisi finanziaria. Ad esempio, potrebbero utilizzare un'API REST fornita da un fornitore di servizi di pagamento per consentire ai clienti di effettuare pagamenti tramite l'applicazione web dell'azienda.

Quando un cliente effettua un pagamento, l'applicazione web invia una richiesta API REST al servizio di pagamento del fornitore per elaborare il pagamento.

Una volta elaborato il pagamento, il servizio di pagamento risponde alla richiesta API REST con una conferma di pagamento, che viene quindi elaborata dall'applicazione web dell'azienda.

L'utilizzo di API REST consente all'azienda nel settore dei servizi finanziari di integrare e coordinare diversi sistemi e applicazioni all'interno della propria infrastruttura tecnologica, facilitando un flusso efficiente di dati e operazioni tra i vari componenti del sistema informativo aziendale.

## Sistema informativo orientato ai servizi

L'utilizzo delle API REST favorisce il trasferimento diretto delle informazioni dall'azienda agli utenti attraverso un portale web, consolidando così i sistemi informativi orientati ai servizi. Questo permette agli utenti di interagire con i servizi offerti, trasformandoli in clienti o acquirenti dell'azienda.

L’informazione, che era il perno centrale del sistema aziendale, diventa ora la colonna portante del sistema informativo e dei servizi che l'azienda offre attraverso il sistema informatico. Il fattore tecnologico catalizza questo passaggio, accelerandone la trasformazione.

Le API REST, insieme a concetti come CRUD, diventano il fondamento di questa trasformazione. CRUD rappresenta le quattro operazioni fondamentali per la gestione dei dati in un sistema informatico: Create, Read, Update, Delete. Tali operazioni sono descritte come segue:

* Create: aggiungere nuovi dati al sistema. Questa operazione consente di inserire nuove informazioni in un database. Ad esempio: inserire un nuovo utente all’interno di una tabella degli utenti.
* Read: recuperare e visualizzare i dati esistenti. Questa operazione permette di leggere i dati salvati nel sistema. Ad esempio: cercare e visualizzare le informazioni di un utente in particolare.
* Update: aggiornare o modificare i dati già presenti all’interno del sistema. Questa operazione consente di cambiare i dati già presenti nel sistema.
* Delete: eliminare i dati esistenti dal sistema. Questa operazione consente di rimuovere dati non più necessari o non corretti dal sistema.

Le API REST consentono di eseguire queste operazioni in modo efficiente e standardizzato, fornendo agli utenti un accesso sicuro e intuitivo ai servizi e ai dati aziendali.

Creano un cambiamento nelle informazioni che non riguarda solo il flusso delle informazioni, ma anche il ruolo centrale che l'informazione assume nel sistema informativo aziendale. Prima considerata un aspetto fondamentale e prioritario da tenere internamente, l'informazione diventa ora il fulcro del sistema informatico, alimentando i servizi offerti e guidando le decisioni aziendali. Il progresso tecnologico continua ad avere un ruolo cruciale in questa trasformazione.

Le API REST e i concetti CRUD, supportati da innovazioni emergenti, accelerano l'adozione di sistemi informativi orientati ai servizi, fornendo alle aziende la flessibilità, la scalabilità e l'adattabilità necessarie per rimanere competitive all’interno di un mercato in perenne evoluzione. L’interazione con le API REST avviene mediante un sistema o tecnologia che ne permette l’uso. Il mezzo più efficace e più utilizzato è tramite un portale web. Il portale web funzionerebbe come un intermediario tra l'utente e il sistema informativo, consentendo all'utente di accedere e utilizzare i servizi disponibili in modo più semplice e più intuitivo.

Questo portale web sarebbe sviluppato utilizzando tecnologie o framework che supportano sia il lato utente (front-end), garantendo un'interfaccia user-friendly e reattiva, sia il lato dei sistemi informativi (back-end), assicurando una gestione efficiente dei dati e delle operazioni nel sistema. La ricezione e l’invio dei dati avrebbero attraverso l’inserimento degli stessi in file che avrebbero un formato e una loro struttura.

Durante la fase di ricezione, i dati sarebbero strutturati secondo un formato o uno standard, hanno bisogno un processo di analisi o lettura delle informazioni in esso contenute. Viceversa, l’invio delle informazioni comporterebbe il processo inverso della ricezione con la conversione dei dati in un file strutturati.

Questo passaggio, di estrapolazione e di conversione dei dati in informazioni, è una parte importante se non critica dei sistemi informativi. Se l’informazione è il perno centrale dei sistemi informativi, la sua comprensione ne è l’elemento cruciale.

Ecco come i sistemi informativi orientati ai servizi, grazie alle API REST e ai concetti CRUD, stanno rimodernando il panorama del settore aziendale.

## Esempi di sistemi informativi

Tutto ciò che è stato definito precedentemente rappresenta la descrizione di ciò che avviene quotidianamente nel mondo. Questo permette di definire degli esempi validi che danno applicazione a quanto descritto.

Ad esempio, ipoteticamente esiste una impresa chiamata "DigitalTech", un fornitore leader di soluzioni software per aziende, il cui sistema informativo svolge un ruolo fondamentale nel fornire servizi efficienti e personalizzati ai clienti. Utilizzando le API REST, DigitalTech gestisce la distribuzione delle sue soluzioni software, consentendo ai clienti di accedere e utilizzare facilmente le applicazioni tramite un portale web dedicato. Grazie alla sua architettura scalabile, il sistema è in grado di gestire un elevato volume di utenti e transazioni senza compromettere le prestazioni. Il reparto sviluppo di DigitalTech è responsabile dello sviluppo e della manutenzione del sistema informativo, lavorando a stretto contatto con altri reparti per garantire che le esigenze aziendali siano soddisfatte in modo efficiente e tempestivo. La sicurezza dei dati è una priorità assoluta per DigitalTech, che implementa robuste misure di protezione per proteggere le informazioni dei clienti e dell'azienda stessa da accessi non autorizzati e minacce esterne. Grazie alla sua affidabilità e flessibilità, il sistema informativo di DigitalTech continua a essere un pilastro fondamentale del successo dell'azienda, consentendo di fornire soluzioni innovative e di alta qualità ai clienti in tutto il mondo.

L’esempio appena descritto rappresenta un sistema informativo creato su un livello di competenze tecnologiche e umane più ampie. Non tutte le aziende sono come quella appena descritta.

Nell'azienda artigianale chiamata "WoodLAb", specializzata nella produzione di mobili su misura. Il loro sistema informativo, basato su API REST, consente ai clienti di interagire direttamente durante la fase di progettazione, apportando modifiche e monitorando lo stato degli ordini attraverso un'interfaccia web intuitiva. Questo non solo aumenta l'engagement dei clienti, ma facilita anche la comprensione reciproca tra artigiani e committenti. Integrando strumenti analitici, WoodLAb raccoglie feedback dettagliati sui progetti, anticipando le esigenze dei clienti e rimanendo all'avanguardia nelle tendenze di mercato. La sicurezza dei dati è una priorità, garantendo la riservatezza delle informazioni dei clienti e la sicurezza dei progetti in fase di sviluppo. Grazie a un sistema informativo all'avanguardia, WoodLAb offre prodotti di qualità superiore e un'esperienza clienti senza pari nel settore dell'arredamento su misura.

Nell'esempio appena descritto, l'azienda WoodLab ha adattato il suo sistema informativo orientandolo verso l'utente. Questo ha portato vantaggi sia all'azienda che al cliente.

Allo stesso modo, la Pubblica Amministrazione affronta sfide simili come sistema informativo. La Pubblica Amministrazione sta sempre più utilizzando sistemi software basati su portali web per interagire con i cittadini. Prendiamo ad esempio il Comune "Porto Grigio", dove il sistema informativo costituisce il nucleo delle attività quotidiane.

Attraverso l'uso di API REST, i cittadini possono accedere online ai servizi comunali, come la pianificazione di appuntamenti per le pratiche municipali, la segnalazione di emergenze ambientali e l'accesso a documenti pubblici.

Con un'interfaccia web più intuitiva, i residenti possono interagire con l'amministrazione in modo rapido e agevole. La gestione dei dati è garantita attraverso misure di sicurezza per salvaguardare la riservatezza dei cittadini e assicurare la conformità normativa. Il sistema informativo di "Porto Grigio" si distingue come esempio di trasparenza e accessibilità, promuovendo una partecipazione più ampia dei cittadini alla vita pubblica e semplificando le pratiche amministrative.

Gli esempi appena descritti rappresentano ipotetici casi dove i sistemi informativi hanno usato al meglio le informazioni e di come si sono addatati ad essa.

Negli ultimi anni, l'informazione è diventata cruciale per le aziende, facilitando adattamenti al mercato e decisioni informate. La gestione dell'informazione è ora fondamentale per il successo aziendale, con i sistemi informativi che giocano un ruolo centrale. L'integrazione tra sistemi informatici e informativi è cresciuta, ottimizzando la gestione dei dati aziendali attraverso database centralizzati e API REST. Queste API consentono un flusso efficiente di dati tra sistemi, facilitando operazioni come vendite personalizzate, marketing mirato e monitoraggio finanziario. Il passaggio da file CSV a JSON e l'uso di API REST stanno trasformando i sistemi informativi aziendali, migliorando l'efficienza operativa e l'esperienza dell'utente.

Nel prossimo capitolo si parlera dell’importanza della analisi sui dati ricevuti e sui formati.

# Parsing

Il parsing (dall’inglese analisi grammaticale) è un processo informatico fondamentale che consiste nell'analisi di un input testuale, come una stringa di caratteri, al fine di comprendere la sua struttura e il suo significato seguendo regole specifiche.

Questo compito è svolto da un modulo software chiamato parser.

Un parser è un componente essenziale del processo di compilazione. Un compilatore traduce il codice sorgente in istruzioni comprensibile dal computer. L'analisi risultante, anch'essa parte integrante del processo di compilazione, avviene durante la fase di analisi della compilazione.

Durante l'analisi, il codice viene preso dal preprocessore, suddiviso in parti più piccole e analizzato in modo che altri software possano comprenderlo.

Il parser svolge questa operazione costruendo una struttura dati a partire dai pezzi di input.

Più specificamente, una persona scrive codice in un linguaggio leggibile dall'uomo, come C++ o Java, e lo salva come una serie di file di testo. Il parser prende questi file di testo come input e li scompone in modo che possano essere tradotti sulla piattaforma di destinazione.

Il parser è costituito da tre componenti, ciascuno dei quali gestisce una fase diversa del processo di analisi.

Le tre fasi sono: Analisi lessicale, Analisi sintattica e Analisi semantica.

Durante l'Analisi lessicale, o scanner, il codice prelevato dal preprocessore viene suddiviso in parti più piccole chiamate lessemi. Ogni lessema corrisponde a un token, che rappresenta un'unità grammaticale del linguaggio di programmazione compresa dal compilatore.

In questa fase, vengono anche eliminati spazi bianchi, commenti ed eventuali errori dall'input.

L'Analisi sintattica controlla la struttura sintattica dell'input utilizzando un albero di derivazione o di analisi come struttura dati. Un analizzatore sintattico utilizza i token per creare l'albero di parsing, combinando la grammatica predefinita del linguaggio di programmazione con i token estratti dalla stringa di input. Durante questa fase, viene segnalata la presenza di eventuali errori di sintassi.

L'ultima fase dell'analisi è l'Analisi semantica, durante la quale l'albero di parsing viene verificato utilizzando una tabella dei simboli per determinare la sua correttezza dal punto di vista semantico. Questa fase è nota come analisi sensibile al contesto e include controlli sul tipo di dati, sulle variabili e sul flusso di esecuzione.

Ad esempio, se consideriamo un comando di assegnazione come float a = 10; float b = a\*3; durante l'analisi, il valore 3 sarà trattato come 3.0 prima di eseguire l'operazione di moltiplicazioni.

Quando viene creato un formato di file, di markup o linguaggio di programmazione, si definiscono delle regole. Tali regole costituiscono la base per creare affermazioni valide per quel formato o linguaggio di programmazione. Un esempio di regole di grammatica molto semplici potrebbe essere il seguente:

<frase> ::=<soggetto> <verbo> <object>

<soggetto> ::= <articolo> <sostantivo>

<articolo> ::= il |un |una

<sostantivo> ::= cane | gatto | persona |animale

<verbo> ::= animali |nutrito | è

<oggetto> ::= <articolo> <sostantivo>

Dalle seguenti regole si comprende che una frase può contenere solo soggetto, verbo e oggetto; a loro volta, il soggetto e l'oggetto possono contenere solo articolo e sostantivo, in quest'ordine. L'articolo può essere 'il', 'un' o 'una', il verbo può essere solo 'animali', 'nutrito' o ‘è’, e il sostantivo può essere solo 'cane', 'gatto', 'persona' o ‘animale’.

Un esempio di tali regole sarebbe: “un cane è un animale”.

L’analisi determina se l’input fornito dall’utente rispetti le regole di quel linguaggio o formato per determinare se è valido. Ciò avviene utilizzando algoritmi che effettuano l’analisi in ordine differente. Ci sono due approcci di parser: Top-Down e Bottom-Up.

Top-Down, significa dall’alto verso in basso, è parte iniziando con la regola in alto. Ad esempio <frase> ::=<soggetto> <verbo> <object>. Se usiamo come input la stringa “un cane è un animale”, il parser analizzerà la prima regola e controllerà tutte le regole sottostanti alla prima. Inizierà dalla prima parola sottostante, cioè, <soggetto>, controllerà la regola <soggetto> e passerà alla successiva <verbo>.

Bottom-Up, significa dal basso verso l’alto, è parte dalla regola in basso. Nell’esempio precedente è <oggetto>

Top-Down inizia dalla prima regola, mentre Bottom-up dall’ultima.

Oltre a queste, ci sono anche le derivazioni di questi approcci. Tali derivazioni differiscono nell’ordine con cui la grammatica processa la stringa di input. Esse sono: LL e LR.

LL analizza l’input da sinistra a destra, utilizzando la derivazione più a sinistra per combinare le regole della grammatica sull’input. Sostanzialmente, deriva la stringa che convalida l’input espandendo l’elemento più a sinistra dell’albero di analisi.

LR, simmetricamente al LL, espande le regole della grammatica partendo da quella più a destra.

Oltre a quelli, ci sono altri tipi come: parser di discesa ricorsiva, parser iniziali e parser con riduzione di turno.

I parser di discesa ricorsiva tornano indietro dopo ogni punto decisionale per controllare l'accuratezza. Utilizzano l'analisi Top-Down.

I parser iniziali analizzano tutte le grammatiche libere dal contesto. Queste grammatiche descrivono le strutture di un linguaggio utilizzando regole di produzione che consentono di generare stringhe di simboli.

Queste regole non dipendono dal contesto circostante e possono essere utilizzate per analizzare e creare stringhe di un linguaggio. La maggior parte dei linguaggi di programmazione non utilizza grammatiche libere dal contesto.

I parser con riduzione del turno utilizzano un approccio diverso, spostando e riducendo una stringa di input finché non è completamente controllata.

I parser sono necessari quando si deve rappresentare i dati di input dal codice sorgente in modo astratto, come una struttura dati, per determinare se la sintassi sia corretta.

L'analisi del codice è una parte fondamentale di diverse tecnologie per il controllo del codice: linguaggi di programmazione, formato di file, linguaggi di database, protocolli e generatori di parser.

Nei linguaggi di programmazione, i parser sono utilizzati nei linguaggi di programmazione come C++, PHP, Java, JavaScript e Python.

I formati di file hanno anche loro delle regole di grammatica a cui attenersi come CSV, JSON, XML.

Anche nei linguaggi per le basi di dati come l’SQL si utilizzano i parser, così come nei protocolli, come HTTP e nelle chiamate di funzioni remote, come le operazioni CRUD.

I generatori di parser prendono una grammatica come input e ne generano il codice sorgente. Rispetto alle operazioni di parsing utilizzate anche per validare la stringa di input, essi effettuano l'analisi al contrario, costruendo parser da espressioni regolari.

Le espressioni regolari sono stringhe speciali utilizzate per gestire e abbinare modelli nel testo. Funzionano definendo un pattern che può includere caratteri letterali (come lettere o numeri) e caratteri speciali (come metacaratteri) per identificare sequenze specifiche di caratteri all’interno di una stringa.

Ad esempio, se si desidera trovare tutte le occorrenze specifica in un testo, è possibile utilizzare un’espressione regolare per individuare tutte le occorrenze di quel pattern nel testo di input.

Per esempio, se abbiamo la stringa di partenza “il gatto nero è veloce e il gatto bianco è lento”, possiamo utilizzare un’espressione regolare come “gatto” per individuare tutte le occorrenze della parola “gatto” nel testo.

In questo caso, l’espressione regolare “gatto” abbinerebbe entrambe le occorrenze di “gatto” nella stringa di partenza.

Le espressioni regolari offrono anche la possibilità di effettuare più sofisticati, come identificare pattern di testo che eseguono una struttura specifica o che soddisfano determinate condizioni.

Ad esempio, possiamo utilizzare un’espressione regolare come “\b[gG]atto\b” per individuare solo le occorrenze della parola “gatto” che sono precedute o seguite da un confine essere scritto sia in maiuscolo che in minuscolo.

Inoltre, le espressioni regolari consentono di effettuare ricerche avanzate e manipolazioni di testo, come la sostituzione di parti di una stringa che corrispondono a un certo pattern con un’altra stringa. Ad esempio, possiamo utilizzare un’espressione regolare come “gatto” insieme alla funzione di sostituzione per sostituire tutte le occorrenze della parola “gatto” con “cane”, ottenendo così la stringa “il cane nero è veloce il cane bianco è lento”.

Altri esempi sono:

L’individuazione di indirizzi e-mail: utilizzando un pattern come “[a-zA-Z0-9.\_%+-][+@[a-zA-Z0-9.-]+[a-zA-Z]{2,}](mailto:+@[a-zA-Z0-9.-]+[a-zA-Z]%7b2,%7d)” è possibile individuare gli indirizzi e-mail all’interno di un testo.

La validazione di numeri di telefono: un’espressione regolare come “\d{3}-\d{3}-\d{4}” può essere utilizzata per identificare numeri di telefono nel formato comune “xxx-xxx-xxxx”.

La ricerca di Url: con pattern come “https?://(?:www.)?[\w-]+(?:.[\w-]+)+\S\*”, è possibile individuare gli URL nelle stringhe di testo.

La ricerca di date: un’espressione regolare come “\b\d{1,2}/\d{1,2}/\d{4}\b” può essere utilizzare per individuare date nel formato “gg/mm/aaaa”.

La ricerca di codici postali: utilizzando un pattern come “\b\d{5}(?:-\d{4})?\b”, è possibile individuare codici postali negli Stati Uniti nel formato “xxxxx” o “xxxxx-xxxx”.

Individuare hashtags: con un pattern come “#\w+”, è possibile individuare gli hashtag all’interno di un testo, dove “\w+” rappresenta una sequenza di caratteri alfanumerici.

Oltre ad essere incorporati nei compilatori, è possibile emulare il parsing attraverso programmi dedicati sviluppati utilizzando linguaggi di programmazione o database. Ciò offre una maggiore flessibilità per soddisfare esigenze specifiche, come l'analisi dei dati presenti all'interno di file codificati in diversi formati.

Il CSV (Comma-Separated Values) è stato il primo formato ad essere ampiamente utilizzato per il parsing dell'input, grazie alla sua ampia interoperabilità su diversi sistemi. Questo formato di file è stato standardizzato nell'RFC 4180, il quale definisce le specifiche per il CSV. RFC sta per "Request For Comments" ed è un documento che contiene standard. La creazione degli RFC è gestita dall'IETF, un organismo internazionale e aperto che si occupa dell'evoluzione delle tecnologie legate a Internet.

Lo standard RFC 4180 documenta il formato CSV nel seguente modo:

* Ogni record è posizionato su una linea separata, delimitata da una interruzione di riga, CLRF.
* L’ultimo record nel file può avere o non avere una interruzione finale.
* È possibile che sia presente una riga intestazione opzionale che appare come prima riga del file, con lo stesso formato delle righe di record normale. Questa intestazione conterrà nome corrispondenti ai campi nel file e dovrebbe contenere lo stesso numero di campi dei record nel resto del file.
* All’interno dell’intestazione e di ogni record, possono esserci uno o più campi, separati da virgole. Ogni riga dovrebbe contenere lo stesso numero di campi in tutto il file. Gli spazi sono considerati parte di un campo e non dovrebbero essere ignorati. L’ultimo campo nel record non deve essere eseguito da una virgola.
* Ciascun campo può essere o meno racchiuso tra virgolette doppie (alcuni programmi lo fanno, come Microsoft Excel, le omettono). Se i campi non sono racchiusi tra virgolette doppie, allora le virgolette doppie non possono apparire all’interno dei campi.
* I campi contenenti interruzioni di riga, CLRF, virgolette doppie e virgole dovrebbero essere racchiusi tra virgolette doppie.
* Se vengono utilizzate virgole doppie per racchiudere i campi, allora una virgoletta doppia che appare all’interno di un campo deve essere preceduta con un’altra virgoletta doppia per dire al parser che si tratta di una virgoletta doppia e non dell’inizio o la fine di una stringa.

Come formato, consente l'inserimento di dati al suo interno, fungendo da ponte o connessione di informazioni tra due dispositivi. I programmi non possono leggere i file reciprocamente; è necessario un sistema che consenta lo scambio di dati, proprio come un formato di file.

Il file CSV è accessibile a praticamente tutti i linguaggi di programmazione, attraverso una serie di passaggi, di fasi:

* Aprire il file CSV utilizzando il percorso completo del file, includendo la lettera dell'unità e i nomi di tutte le directory che portano alla posizione del file.
* Leggere il primo record nel file e caricare il record nelle variabili appropriate.
* Avviare un ciclo che dovrà terminare quando il file arriva alla fine.
* All'interno del ciclo, inserire codice che elabora le informazioni presenti nel record.
* Chiudere il file dopo il ciclo.

## JSON

Lo scambio di informazione avviene tramite l’incapsulamento dei dati all’interno dei file. All’interno dei file, i dati sono strutturati in formati. Ogni formato rappresenta un modo di organizzare i dati al loro interno utilizzando strutture e/o caratteri diversi. Alcuni di essi sono più usati rispetto ad altri e di conseguenza sono più conosciuti e apprezzati. Uno di essi è il formato JSON.

JSON (JavaScript Object Notation) è un formato molto utilizzato oggi per lo scambio di informazioni. Questo formato è stato standardizzato per la prima volta nel 2013 all'interno dello standard ECMA-404, e successivamente è stato aggiornato nella sua seconda versione nel 2017 come parte dello standard ISO/IEC 21778.

È stato definito anche all’interno del RFC 8259. Dalla prima alla seconda versione dell’RFC 8259, sono state fatte alcune modifiche per rendere più facile capire e usare il formato JSON. Queste modifiche includono spiegazioni extra per chiarire il significato di alcune idee, consigli nuovi su come usare JSON correttamente e nuovi esempi per aiutare a capire meglio come funziona. Inoltre, sono state aggiunte informazioni sulla sicurezza e su come gestire eventuali errori, per aiutare gli sviluppatori a fare applicazioni più sicure e solide che usano JSON per scambiare i dati.

ECMA (European Computer Manufacturers Association) è un'associazione fondata nel 1961 che si occupa di standardizzazione nel settore informatico e dei sistemi di telecomunicazione. ISO/IEC 21778 è uno standard definito dall'ISO (Organizzazione internazionale per la normazione) e dall'IEC (Commissione elettrotecnica internazionale), due entità che collaborano nella definizione degli standard.

Il formato JSON è composto da una sequenza di token ed include sei caratteri strutturali ([ - { - ] - } - : - ,), stringhe, numeri e tre nomi letterali (true, false, null) ed è un valore serializzato. Quest’ultimo aspetto permette di rappresentare dati strutturati in una sequenza di caratteri secondo le specifiche di JSON.

La sintassi di JSON è definita come segue.

JSON è composto da:

Spazi bianchi (ws: white spaces), seguiti da un valore (value) nel seguente modo:

JSON Text = ws value ws

Sei caratteri strutturali:

* Inizio-array = ws %x5B ws; ‘[‘ parentesi quadra aperta per l’inizio di un array
* Inizio-oggetto = ws %x7B ws; ‘{‘ parantesi graffa aperta per l’inizio di un oggetto
* Fine-array = ws %x5D ws; ‘]’ parentesi quadra chiusa per la fine di un array
* Fine-oggetto = ws %7Dws; ‘}’ parentesi graffa chiusa per la fina di un oggetto
* separatore-nome = ws %x3A ws; ‘:’ due punti
* separatore-valore = ws %x2C ws; ‘,’ virgola

I valori sono rappresentati usando la notazione ‘%xhh’ per rappresentare i valori esadecimali. Ad esempio %x5B indica la parentesi quadra aperta ‘[‘.

È consentito inserire spazi vuoti prima o dopo uno qualsiasi dei sei caratteri strutturali:

* %x20; spazio
* %x09; tabulazione orizzontale
* %x0A; avanzamento di riga o Nuova riga
* %x0D; ritorno a capo

Un valore JSON deve essere un oggetto, array, un numero o una stringa, oppure uno dei letterali: true, false, null.

I nomi dei letterali dei letterali devono essere in minuscolo. Non sono ammessi altri letterali secondo le seguenti regole:

* valori = false / null / true / oggetto / array / stringa / numero
* false = %x66.61.6c.73.65 : false
* null = %x6e.75.6c.6c : null
* true = %x74.72.75.65 : true

Una struttura di oggetto è rappresentata da una coppia di parentesi graffe che circondano zero o più coppie di nome/valore. Un nome è una stringa. Dopo ogni nome viene inserito un solo due punti, separando il nome dal valore. Una singola virgola separa un valore da un nome seguente.

La sintassi risulta:

oggetto = inizio-oggetto [ membro \* (separatore-valore membro ) ] fine-oggetto

membro = stringa separatore-nome valore

I nomi all’interno di un oggetto devono assolutamente essere unici. Quando in nomi di un oggetto non sono unici, il comportamento delle diverse implicazioni del software può variare.

Alcune implementazioni potrebbero ignorare le coppie duplicate e considerare solo l’ultima coppia di un nome specifico, mentre altre potrebbero restituire un errore o comportarsi in modo imprevedibile. Le librerie JSON possono differire sulla questione di mantenere l’ordine dei membri dell’oggetto.

Le implementazioni che non dipendono dall’ordine dei membri saranno compatibili tra loro, mentre quelle che si basano sull’ordine dei membri potrebbero mostrare comportamenti diversi.

Un array è rappresentato da una serie di elementi racchiusi tra parentesi quadre.

La sintassi risulta:

array = inizio-array [ valore \*( separatore-valore valore ) ] fine-array

Non è richiesto che gli elementi all’interno di un array siano dello stesso tipo.

Per i numeri la rappresentazione è molto simile a quella utilizzata nella maggior parte dei linguaggi di programmazione comuni. Un numero può contenere una componente intera, una parte frazionarie e/o una parte esponenziale.

La parte intera può essere preceduta da un segno meno opzionale (per rappresentare numeri negativi) e consiste in una sequenza di cifre decimali. Non sono ammessi zeri iniziali.

La parte frazionaria è costituita da un punto decimale seguito da un a o più cifre decimali.

La parte esponenziale inizia con la lettera “E” (in maiuscolo o minuscolo) seguita da un segno più o meno opzionale e da una o più cifre decimali che rappresentano l’esponente.

La specifica consente alle implementazioni di impostare limiti sul range e sula precisione dei numeri accettati, la precisione e il range dei numeri JSON possono variare a seconda dell’implementazione, ma per garantire l’interoperabilità, le implementazioni dovrebbero approssimare i numeri JSON entro la cifra prevista.

Ad esempio, i numeri molto grandi come 1E400 o con molte cifre come il pi greco possono causare problemi di interoperabilità se il software che riceve questi numeri non ha la capacità di gestirli correttamente.

Numeri speciali come infinito e NaN (Not a Number) non sono ammessi in JSON.

La sintassi sarebbe:

numero = [meno] int [frazione] [esponente]

punto decimale = %x2E ; . punto decimale

cifra da 1 a 9 = %x31-39 ; 1-9

e = %x65 / %x45 ; e / E

esponente = e [meno / più] 1\*cifra

intero = zero / (cifra da 1 a 9 \*cifra)

meno = %x2D ; -

più = %x2V ; +

zero = %x30 ; 0

Le stringhe in JSON sono delimitate da virgolette e possono contenere qualsiasi carattere Unicode, ad eccezione di alcuni caratteri che devono essere sequenze di escape, ovvero il carattere virgolette doppie ("), la barra rovesciata (), e i caratteri di controllo nell’intervallo U+0000 a U+001F.

Ogni carattere può essere una sequenza di escape. Se il carattere si trova nel Basic Multilingual Plane (U+0000 a U+FFFF), può essere rappresentato come una sequenza di escape di sei caratteri. Questa sequenza inizia con una barra rovesciata (), seguita dalla lettera 'u' minuscola e quattro cifre esadecimali che codificano il punto di codice del carattere. Le lettere esadecimali da A ad F possono essere sia maiuscole che minuscole.

In alternativa, alcuni caratteri comuni possono essere rappresentati in modo più compatto utilizzando una sequenza di escape di due caratteri.

Per i caratteri estesi che non rientrano nel Basic Multilingual Plane, la loro rappresentazione è una sequenza di 12 caratteri, che codifica la coppia surrogata UTF-16 del carattere.

La sintassi è:

stringa = virgolette \*carattere virgolette

carattere = senza caratteri di sequenza di escape / sequenza di escape (

%x22 / ; “ virgolette U+0022

%x5C / ; \ barra rovesciata U+005C

%x2F / ; / barra U+002F

%x62 / ; b retrospazio U+0008

%x66 / ; f alimentazione a capo U+000C

%x6E / ; n avanzamento riga U+000A

%x72 / ; r ritorno a capo U+000D

%x74 / ; t tabulazione U+0009

%x75 4HEXDIG ) ; uXXXX U +XXXX

Sequenza di escape = %x5C ; \

virgolette = %x22 ; “

senza sequenza di escape = %x20-21 / %x23-5B / %x5D-10FFFF

Tutte le stringhe in un testo JSON sono composte interamente da caratteri Unicode, anche se alcuni di essi sono sequenze di escape (cioè, rappresentati con la notazione “\u” seguita da un codice esadecimale), allora quel teso JSON è considerato interoperabile. Ciò significa che tutte le implementazioni software che analizzano quel testo JSON dovrebbero essere d’accordo sul significato dei nomi e die valori delle stringhe all’interno degli oggetti e degli array.

Comunque, la specifica JSON permette teoricamente la presenza di sequenza di bit che non possono rappresentare correttamente caratteri Unicode validi. Questo può causare problemi, come quando una libreria tronca una stringa senza verificare se tale troncamento divide una coppia di surrogati UTF-16. In tali casi, il comportamento del software che riceve questi testi JSON contenenti tali valori imprevedibili, potendo restituire valori diversi per la lunghezza di una stringa o subire eccezioni fatali durante l’esecuzione.

{

"nome": "Paolo",

"cognome": "Rossi",

"età": 42,

"sesso": "maschio",

"sposato": true,

"hobby": [

"escursionismo",

"giardinaggio",

"cucina"

],

"indirizzo": {

"via": "Via Roma",

"numero": 10,

"città": "Milano",

"cap": "20121"

}

}

L'esempio fornito illustra un documento JSON che contiene una serie di informazioni relative a persone. Ogni persona è rappresentata da un oggetto all'interno di un array. Ogni oggetto contiene diversi attributi, come il nome, il cognome, l'età e il sesso.

Si possono notare vari tipi di valori: le stringhe per i nomi e i cognomi, i valori numerici per le età e le stringhe per i sessi, che possono essere "maschio" o "femmina". Inoltre, è presente un array associato all'attributo "hobby", che potrebbe contenere una lista di interessi o attività preferite, e un oggetto associato all'attributo "indirizzo", che potrebbe rappresentare dettagli relativi al luogo di residenza di ciascuna persona.

[

{

"nome": "Paolo",

"cognome": "Rossi",

"età": 42,

"sesso": "maschio"

},

{

"nome": "Anna",

"cognome": "Bianchi",

"età": 35,

"sesso": "femmina"

}

]

Questo ulteriore esempio mostra come i dati possano essere organizzati in un formato strutturato e interoperabile come JSON, facilitando lo scambio e l'interpretazione delle informazioni tra diversi sistemi e applicazioni.

Un parser JSON è un componente software che converte un testo scritto in formato JSON in un’altra forma di rappresentazione, solitamente una struttura dati che il programma possa elaborare in modo più efficiente o significato . Il parser deve essere in grado di accettare tutti i testi che rispettano la grammatica definita per il JSON. Per le derivazioni o le estensioni del formato JSON bisogna fare una ulteriore considerazione. Le estensioni possono introdurre regole aggiuntive o modificare il comportamento standard del parser.

Le estensioni possono rendere il codice meno interoperabile e più complesso, è importante usarle con cautela. Le estensioni dovrebbero essere implementate solo se necessario e se vi è una chiara comprensione degli effetti che potrebbero avere sulle altre parti del sistema.

Quando si utilizzano estensioni o derivazioni è importante documentarle in maniera chiara e ordinata per evitare fraintendimenti e problemi di compatibilità.

## Html

Come i formati per i file esiste una tipologia di linguaggi di programmazione che permette di dare una struttura similmente al formato JSON. Questi linguaggi di programmazione vengono definiti linguaggi di marcatura. Uno dei linguaggi di marcatura più utilizzati e diffusi in Informatica è Html.

Html (HyperText Markup Language) è un linguaggio di marcatura usato nella creazione di siti internet o di singoli documenti web.

In una web app di Dart all’interno del cartella web è presente un file html, index.html. Qui si costruirà la parte grafica della web app.

Un linguaggio di marcatura o linguaggio di markup è in insieme di regole che permettono di descrivere dei sistemi di rappresentazione o d’impaginazione di un testo.

È composto da comandi avente sintassi: <”comando”> “contenuto” </”comando”>; Tale struttura può annidata e contenere all’interno del corpo del comando altri comandi di natura diversa gli uni sugli altri.

Il comando rappresenta una proprietà o funzione che viene usata sul contenuto tra i tag. Se vi saranno tag annidati all’interno, il contenuto più interno sarà soggetto delle variazioni o proprietà dei comandi.

Ad esempio, il tag <p> sarà strutturato nel seguente modo: “<p>Esempio paragrafo</p>” visualizza un paragrafo nel documento web la scritta “Esempio paragrafo”.

<!DOCTYPE html>

<html>

<head>

<title>Titolo del documento</title>

</head>

<body>

Contenuto del documento

</body>

</html>

Nell’esempio sopra indicato è possibile vedere una struttura di base di un documento html. Il contenuto della struttura viene rappresentato come un albero di nodi attraverso il DOM.

DOM (Document Object Model) è una forma per rappresentare il contenuto di documenti strutturati come l’html. Quando un browser carica una pagina HTML, crea internamente una rappresentazione del documento sotto forma di DOM.

Il DOM consente agli sviluppatori web di accedere, manipolare e modificare dinamicamente il contenuto, la struttura e lo stile del documento HTML utilizzando JavaScript o altri linguaggi di scripting.

<!DOCTYPE html> indica che il documento web è scritto in html 5. Il “5” indica la versione a cui fare riferimento di html.

<html> </html> all’interno del tag si costruisce quello che è il documento html vero è proprio.

<head></head> si inseriscono informazioni che permettono il corretto funzionamento del documento web come il titolo o i metadati. L’utente non sa cosa c’è all’interno, ne è all’oscuro.

<body></body> contiene il corpo del documento del documento vero e proprio. Qui si costruisce la parte grafica dell’utente.

## Javascript

Un documento HTML definisce gli elementi che verranno visualizzati tramite il browser una volta caricato. La pagina risultante avrà elementi definiti a livello di codice, risultando quindi immutabile. Questo tipo di pagina è descritta come statica. Per rendere le pagine più dinamiche e interattive, viene inserito del codice all’interno della pagina. Attualmente, uno dei linguaggi di programmazione più utilizzati nelle pagine web è JavaScript.

Javascript è un linguaggio di programmazione versatile e potente, nato originariamente come un linguaggio agli oggetti con base in Java e altri linguaggi. La sua flessibilità gli consente di essere utilizzato sia sul lato client che sul lato server, rendendolo una componente fondamentale nello sviluppo web moderno.

Attualmente, JavaScript è standardizzato nella sua tredicesima versione, dimostrando un costante sviluppo e miglioramento nel corso degli anni. Una delle sue caratteristiche principali è la capacità di manipolare il Document Object Model (DOM) in modo dinamico. Questo significa che può essere utilizzato per aggiungere, modificare o eleminare elementi all’interno di una pagina web mentre viene visualizzata dal browser.

Un esempio di utilizzo di JavaScript all’interno di una pagina HTML è l’inserimento di codice all’interno del tag ‘<script></script>’, come illustrato di seguito:

<script>

var messaggio = "Ciao, mondo!";

console.log(messaggio);

</script>

Questo codice visualizza il contenuto della variabile “messaggio” nella console del browser.

Un aspetto crucciale di JavaScript è la sua natura orientata agli oggetti basata su prototipi. Questo significa che la programmazione orientata agli oggetti in JavaScript si basa sull’uso di prototipi anziché sul concetto tradizionale di classi. Sebbene non sia possibile ereditare direttamente da un oggetto a un altro, è possibile simulare l’ereditarietà tra oggetti mediante tecniche come la delega dei prototipi.

function Veicolo(marca, modello) {

this.marca = marca;

this.modello = modello;

}

Veicolo.prototype.descrizione = function() {

return "Marca: " + this.marca + ", Modello: " + this.modello;

};

function Auto(marca, modello, colore) {

Veicolo.call(this, marca, modello);

this.colore = colore;

}

Auto.prototype = Object.create(Veicolo.prototype);

Auto.prototype.constructor = Auto;

Auto.prototype.info = function() {

return this.descrizione() + ", Colore: " + this.colore;

};

var miaAuto = new Auto("Ford", "Focus", "Blu");

console.log(miaAuto.info()); // Output: Marca: Ford, Modello: Focus, Colore: Blu

JavaScript offre una vasta gamma di oggetti predefiniti, come Array, Boolean, Date, Function, Math e altri. Inoltre, il browser fornisce anche oggetti ospiti che consentono di interagire con il DOM e gestire eventi, formulari e altre funzionalità della pagina web.

Oltre alle variabili primitive e alle strutture di controllo del flusso comuni a molti linguaggi di programmazione, JavaScript supporta anche funzionalità avanzate come le espressioni regolari, la gestione degli errori tramite blocchi try-catch, e la programmazione asincrona tramite Promises e async/awit. Di seguito sono riportati degli estratti di codice che mostrano tali funzionalità.

// Uso di espressioni regolari per validare una stringa che rappresenta una e-mail

const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

function validaEmail(e-mail) {

if (emailPattern.test(e-mail)) {

console.log("L'email è valida.");

} else {

throw new Error("L'email non è valida.");

}

}

// Gestione degli errori tramite blocchi try-catch

try {

validaEmail("example@example.com");

} catch (errore) {

console.error(errore.message);

}

// Esempio di programmazione async/await con Promise

function operazioneAsincrona() {

return new Promise((resolve, reject) => {

// Simuliamo un ritardo di 1 secondo prima di risolvere la promessa

setTimeout(() => {

// Risolviamo la promessa con un messaggio

resolve("Operazione completata con successo!");

}, 1000);

});

}

async function eseguiOperazioneAsincrona() {

try {

const messaggio = await operazioneAsincrona();

console.log(messaggio); // Output: Operazione completata con successo!

} catch (errore) {

console.error(errore); // Gestione degli errori

}

}

eseguiOperazioneAsincrona();

In sintesi, JavaScript è un linguaggio di programmazione essenziale per lo sviluppo web moderno, con una vasta gamma di funzionalità e una continua evoluzione che lo rende uno strumento fondamentale per la creazione di esperienze utente dinamiche e interattive.

## Dart

Attraverso JavaScript si è reso possibile creare pagine web dinamiche attraverso la manipolazione del DOM. Le pagine web dinamiche hanno acquisito notorietà e diffusione rendendo JavaScript molto diffuso e conosciuto. Successivamente sono nati linguaggi di programmazione ispirati a JavaScript. Uno di questi è Dart.

Dart è un linguaggio di programmazione open-source sviluppato da Google, progettato per offrire un ambiente di sviluppo moderno e per performante per la creazione di applicazione web e mobile. La sua sintassi chiara e intuitiva, insieme a una serie di strumenti e framework, rendono Dart una scelta popolare tra gli sviluppatori che cercano ci creare applicazioni robuste e scalabili.

Una delle caratteristiche distintive di Dart è la sua flessibilità nell’ambito dello sviluppo cross-platform. Con Dart, gli sviluppatori possono scrivere codice una sola volta e distribuirlo su diverse piatteforme, tra cui Android, iOS, web e persino applicazioni desktop. Questo è reso possibile grazie a kit di sviluppo come Flutter e AngularDart.

AngularDart è un kit di sviluppo di applicazioni web che sfrutta la potenza di Dart per creare interfacce utente dinamiche e complesse. Basato sul framework JavaScript Angular, AngularDart offre un’esperienza di sviluppo moderna e scalabile per la creazione di applicazioni web complesse.

Oltre alle sue capacità cross-platform, Dart offre una serie di funzionalità avanzate ch lo rendono un linguaggio potente per lo sviluppo di applicazioni moderne. Tra queste vi sono:

* Tipizzazione Statica: Dart supporta la tipizzazione statica, che consente agli sviluppatori di individuare errori nel codice durante la compilazione anziché a tempo di esecuzione. Questo aiuta a migliorare la robustezza e la stabilità delle applicazioni.

int somma(int a, int b) {

return a + b;

}

void main() {

print(somma(5, 10)); // Output: 15

}

* Garbage collection: Dart gestisce automaticamente la memoria allocata per gli oggetti non utilizzati mediante il garbage collection, semplificando il processo di gestione della memoria per gli sviluppatori.
* Asincronia: Dart offre un supporto nativo per la programmazione asincrona, consentendo agli sviluppatori di scrivere codice che gestisce operazioni non bloccanti come chiamate di rete e operazioni di I/O in modo efficiente e conciso.

Future<void> main() async {

print(‘Inizio’);

await Future.delayed(Duration(seconds: 2));

print(‘Fine’);

}

* Librerie standard: Dart fornisce una ricca libreria standard che include una vasta gamma di funzionalità per la gestione di dati, la manipolazione di stringhe, l’elaborazione di file e molto altro ancora. Queste librerie aiutano gli sviluppatori a scrivere codice più efficiente e riducono la necessità di scrivere codice personalizzato per compiti comuni.

Per iniziare a sviluppare applicazioni con Dart, è necessario scaricare e installare l’SDK di Dart. Una volta installato, è possibile utilizzare strumenti come Webdev per semplificare lo sviluppo di applicazioni web in Dart.

Webdev gestisce le dipendenze del progetto e avvia un server locale per la visualizzazione dell’applicazione durante lo sviluppo.

La creazione di una nuova web app Dart è semplice grazie al comando ‘dart create-t web nomeprogetto’.

Questo comando crea una struttura di base per la web app, inclusi i file main.dart per la logica dell’applicazione, un file HTML per la struttura della pagina web e un file CSS per lo stile.

Una delle caratteristiche più potenti di Dart è la possibilità di ricarica automaticamente le modifiche apportate al codice durante lo sviluppo.

Ogni volta che vengono apportate modifiche ai file della web app, Webdev rileva automaticamente queste modifiche e le ricarica nel browser in tempo reale, consentendo un flusso di lavoro efficiente e produttivo.

Inoltre, Dart offre una serie di librerie e framework per lo sviluppo di applicazioni web e mobile. AngularDart e Flutter sono due esempi principali.

AngularDart è un framework per lo sviluppo di applicazioni mobile cross-platform. Entrambi offrono un’esperienze di sviluppo moderno e scalabile per la creazione di applicazioni complesse.

In sintesi, Dart è un linguaggio di programmazione potente e flessibile che offre molte funzionalità per lo sviluppo web e mobile.

Con il supporto cross-platform, la sua sintassi intuitiva e le sue caratteristiche avanzate, Dart è una scelta eccellente per gli sviluppatori che desiderano creare applicazione moderne e performanti per una varietà di piattaforme.

La comunità di sviluppatori attorno a Dart è attiva e in costante crescita, offrendo supporto e risorse aggiuntive per colo che vogliono approfondire le proprie conoscenza e competenze nel linguaggio.

## Flutter

Dart, oltre ad essere utilizzato come alternativa a JavaScript, è un linguaggio di programmazione completo con numerose caratteristiche simili a quelle presenti in altri linguaggi di programmazione moderni. È il linguaggio principale utilizzato all'interno di Flutter.

Flutter è un kit di sviluppo open-source introdotto da Google nel 2017, creato per consentire agli sviluppatori lo sviluppo di interfacce utente per dispositivi mobile, web e desktop utilizzando un singolo codice sorgente.

Basato sul linguaggio di programmazione Dart, Flutter ha rapidamente guadagnato popolarità per la sua capacità di offrire esperienze utente più fluide e native su una vasta gamma di piattaforma.

Uno degli aspretti più distintivi di Flutter è il meccanismo di “Hot Reload”, una funzione che consente agli sviluppatori di fare modifiche al codice e visualizzarle istantaneamente nell’app durante l’esecuzione.

L’Hot Reload riduce notevolmente il tempo necessario per il ciclo di sviluppo e la fase di testing. Usando un approccio iterativo, gli sviluppatori possono rapidamente sperimentare e iterare sul design e sulle funzionalità dell’app, migliorando l’efficienza, nel suo complesso, dello sviluppo.

Flutter offre una vasta gamma di widget personalizzabili che consentono agli sviluppatori di creare facilmente interfacce utente più ricche e più coinvolgenti.

I widget di Flutter sono costruiti su un sistema di rendering proprietario che permette prestazioni migliori e una resa precisa su diverse piattaforme.

Con l’utilizzo dei widget per tutto, dalle caselle di testo ai complessi layout multipli, Flutter fornisce agli sviluppatori gli strumenti necessari per creare esperienze utente sodisfacente e di alta qualità.

Le prestazioni sono un altro aspetto in cui Flutter garantisce ottimi risultati. Grazie alla sua architettura ottimizzata e alla compilazione just-in-time (JIT) o ahead-of-time (AOT), Flutter offre un’esperienza utente molto fluida e reattiva, anche su dispositivi di fascia economica.

Just-in-time e ahead-of-time sono due approcci di compilazione utilizzati nel contesto dello sviluppo del software. Entrambi hanno un impatto significativo sulle prestazioni e sulle modalità di esecuzione delle applicazioni.

Just-in-time o JIT è un processo di compilazione in cui il codice sorgente viene tradotto in codice eseguibile durante l’esecuzione dell’applicazione stessa. Durante l’esecuzione, il codice sorgete viene analizzato e tradotto il codice macchina ad alta velocità da parte di un compilatore JIT.

Questo approccio consente un’ottimizzazione dinamica del codice, in cui le parte di codice frequentemente eseguite possono essere ottimizzate in tempo reale, adattandosi al comportamento effettivo dell’applicazione durante l’esecuzione.

L’uso di JIT facilita lo sviluppo e il debug dell’applicazione, in quanto consente di vedere immediatamente i risultati delle modifiche al codice senza dover ricompilare l’intero programma.

Il secondo, contrariamente al primo, in AOT il codice sorgente viene compilato in codice eseguibile (spesso codice nativo della piattaforma di destinazione) prima dell’esecuzione dell’applicazione. Durante la compilazione AOT, il codice sorgente viene tradotto in linguaggio macchine e vengono applicate varie ottimizzazioni.

L’esecuzione dell’applicazione avviene quindi utilizzando direttamente il codice compilato, senza ulteriori fasi di traduzione o ottimizzazione durante l’esecuzione.

Questo approccio di compilazione è comunemente utilizzato per migliorare le prestazioni e ridurre le dimensioni delle applicazioni, specialmente quando si tratta di distribuire l’applicazione al pubblico.

Nel complesso, questo approccio consente a Flutter di rappresentare una scelta ideale per lo sviluppo di applicazioni che richiedono elevate prestazioni e una risposta immediate agli input dell’utente.

Si integra perfettamente con altre con altre tecnologie e piattaforme. Ad esempio, l’integrazione con Firebase, la suite di sviluppo di app di Google, offre agli sviluppatori accesso a una vasta gamma di strumenti e servizi, tra cui l’autenticazione degli utenti, il database in tempo reale, la gestione delle notifiche e molto altro ancora.

Questa integrazione semplifica notevolmente lo sviluppo di funzionalità avanzate e consente agli sviluppatori di concentrarsi sulla creazione di esperienze utente uniche e coinvolgenti.

In più, Flutter è ottimo per aiutare gruppo di sviluppatori a lavorare insieme. Grazie al suo modo di usare lo stesso codice per più piattaforme, gli sviluppatori possono lavorare meglio insieme, senza dover fare lo stesso lavoro più volte, e assicurandosi che le varie versioni dell’app siano coerenti tra di loro.

L’ecosistema di Flutter creare costantemente, con molte funzionalità e aggiornamenti che migliorano l’esperienza degli sviluppatori.

Google e la community di Flutter si concentrano sulla qualità e sull’innovazione, garantendo che il framework rimanga aggiornato per soddisfare le esigenze dello sviluppo di app.

Un altro punto forte di Flutter è la sua flessibilità nell’integrarsi con altre tecnologie. Gli sviluppatori possono facilmente combinare Flutter con codice nativo esistente o utilizzare librerie di terze parte per aggiungere funzionalità specifiche all’app.

Questo consente di sfruttare le risorse già disponibili e di amplificare le capacità dell’applicazione senza dover ricominciare da zero.

In sintesi, Flutter è un framework versatile, potente e adatto a una vasta gamma di progetti, grazie alle sue caratteristiche avanzate, alle prestazioni elevate e al supporto attivo della community, rimane una scelta di primo piano per gli

sviluppatori che desiderano creare app moderne e innovative su diverse piatteforme.

## Low code e No code

In genere, i linguaggi di programmazione sono complessi e richiedono una profonda conoscenza tecnica. La creazione di programmi e software richiede competenze e conoscenze specifiche, spesso possedute solo da sviluppatori qualificati. I programmi realizzati tendono ad essere complessi e lunghi, rendendoli difficili da comprendere per chi è esterno allo sviluppo o ha poche o nulle competenze di programmazione, almeno nella programmazione tradizionale.

Lo sviluppo con poco codice o “Low-code” offre un’alternativa, rendendo più facile e accessibile per chi non è un programmatore professionista creare programmi e software. Questo approccio si distingue dalla programmazione tradizionale per alcune peculiarità:

* Le piattaforme low code offrono un’interfaccia utente molto più intuitiva che consente agli sviluppatori di progettare applicazioni attraverso l’uso di modelli grafici, controlli e componenti visivi.
* Tali piattaforme forniscono una vasta gamma di componenti predefiniti, come moduli, tabelle, grafici e funzioni di integrazione, che possono essere utilizzato per creare rapidamente funzionalità di base dell’applicazione.
* Le piattaforme low-code spesso includono strumenti per l’automazione dei processi aziendali, consentono agli sviluppatori di creare flussi di lavoro automatizzati senza scrivere codice.
* Facilitano l’integrazione come sistemi esterni, come database, servizi web e applicazioni di terze parti.

Di contro risultano svantaggiate se prendiamo in considerazione diverse caratteristiche:

* A volte le piattaforme low-code possono non offrire la flessibilità necessaria per soddisfare requisiti di personalizzazione complessi o specifici.
* Le applicazioni create con piattaforme low-code potrebbero non avere le stesse prestazioni ottimali delle applicazioni sviluppate manualmente, specialmente per progetti più complessi o ad alte prestazioni.
* La sicurezza delle applicazioni create con piattaforme low-code può essere una preoccupazione, specialmente se le piattaforme non seguono le migliori pratiche di sicurezza o se non vengono applicati aggiornamenti regolari per affrontare le vulnerabilità.

Mentre il low-code permette l’utilizzo del basso livello di codice, nel “no-code” non si fa usa di codice all’interno di un programma. La filosofia del no-code si basa sull’idea di consentire a chiunque, anche a coloro che non hanno competenze tecniche o esperienze di programmazione, di creare applicazioni software senza la necessità di scrivere alcun codice. Le piattaforme no-code offrono strumenti visivi e interfacce grafiche che consentono agli utenti di progettare, sviluppare e distribuire applicazioni utilizzando principalmente trascinamento e rilascio di elementi già predefiniti e configurati dalle piattaforme. Questo tipo di approccio serve a consentire a un numero più ampio di utenti di partecipare allo sviluppo software, permettendo a una gamma più ampia di persone di creare applicazioni senza la necessità di imparare linguaggi di programmazione complessi.

Complessivamente un approccio no-code consentono una serie di vantaggi e di svantaggi rispetto al low-code e alla programmazione classica:

* Il no code è ancora più accessibile del low-code, poiché non richiede alcuna competenza di programmazione. Un’ampia gamma di utenti, inclusi coloro che non hanno alcuna esperienza tecnica, può utilizzare queste piattaforme per creare applicazioni.
* Siccome non è richiesta alcuna conoscenza di codifica, il processo di sviluppo con no-code può essere anche più veloce rispetto al low-code e alla programmazione classica che richiede tempi di programmazione decisamente lunghi. Gli utenti possono creare rapidamente prototipi e applicazioni funzionali senza dover scrivere alcun codice.
* Gli utenti possono iterare e aggiornare i programmi in modo più semplice, poiché le modifiche possono essere apportate direttamente attraverso l’interfaccia visiva senza la necessità di modifiche complesse nel codice.

L’uso del no-code rispetto al low-code incide negativamente se si considerano:

* Spesso offrono una gamma più limitata di opzioni di personalizzazione rispetto al low-code. Non è possibile scrivere codice personalizzato, funzionalità avanzate o complesse potrebbero essere difficili da implementare.
* Possono avare un rendimento prestazionale inferiore rispetto alle soluzioni create con low-code e alla programmazione classica. Rappresenta un elemento non secondario per programmi ad alte prestazioni o di una certa complessità.
* Uso di una piattaforma no-code significa una forte dipendenza dal fornitore della piattaforma. Non avendo accesso al codice sorgente, potrebbero essere limitati dalle funzionalità e dagli aggiornamenti offerti dalla piattaforma stessa.

Sia il low-code che il no-code trovano la loro utilità nello sviluppo di programmi, siti web e mobile ma anche nell’analisi dei dati. In questo contesto, si possono trovare esempi di parsing dei dati, che consiste nell’analizzare ed estrarre informazioni utili da diverse fonti di dati.

Attraverso l’automazione dei processi, è possibile creare sistemi all’interno di piattaforme low-code o no-code che facilitano il parsing dei dati. Questi sistemi consentono di estrarre, manipolare e utilizzare le informazioni contenute nei dati senza richiedere la scrittura di codice da zero.

Ad esempio, utilizzando strumenti low-code, è possibile creare flussi di lavoro visivi che comprendono operazioni di parsing dei dati da fonti strutturate come database o API. Questi flussi possono essere configurati per eseguire azioni specifiche, come l’estrazione di informazioni da documenti o la manipolazione di dati da file Excel.

D’altra parte, utilizzando approcci no-code, è possibile utilizzando un’interfaccia intuitiva per creare flussi di lavoro che coinvolgono il parsing dei dati da fonti come e-mail o pagine web. Questi strumenti semplificano il processo di analisi dei dati, consentono agli utenti di definire regole per estrarre informazioni specifiche senza la necessità di scrivere codice.

Anche se sistemi no-code o low-code permettono ad un numero elevato di utenti la possibilità di creare soluzioni software con maggiore autonomia e una bassa o nulla conoscenza dei linguaggi di programmazione vi sono delle controindicazioni. Tali controindicazioni vanno tenuti in considerazione prima di iniziare qualsiasi progetto in queste piattaforme.

In conclusione, il parsing riveste un ruolo fondamentale nell'ambito dell'elaborazione e dell'interpretazione dei dati provenienti da una vasta gamma di fonti informative, che possono comprendere file, database, API e documenti web, sia in formati comuni come CSV, JSON e HTML, che in altri formati più specifici.

Allo stesso tempo, il parsing trova applicazione in linguaggi di programmazione ampiamente utilizzati come C++, Python, Java, JavaScript e i loro derivati, come Il linguaggio Dart e il framework Flutter, consentono la creazione di applicazioni web e mobile. Il parsing è un concetto essenziale in questi linguaggi poiché permette di interpretare e manipolare dati strutturati o testuali in modo da renderli utilizzabili all'interno del programma. Inoltre, è impiegato anche in piattaforme di sviluppo low-code o no-code, che offrono strumenti intuitivi per la creazione di applicazioni senza la necessità di scrivere codice da zero. Anche se in queste piattaforme l'utente può non essere direttamente coinvolto nella scrittura del codice, il parsing sottostante aiuta a interpretare e processare le azioni e le logiche definite attraverso l’uso di interfacce visive o diagrammi di flusso, trasformandole in codice eseguibile.

In questo contesto, il parsing assume un ruolo molto importante nell'analisi e nell'estrarre informazioni rilevanti dai dati per alimentare le applicazioni, consentendo loro di funzionare in modo più efficace e di offrire un'esperienza utente più ottimale. Grazie al parsing, è possibile interpretare la struttura dei dati e manipolarli in modo da adattarli meglio alle esigenze specifiche delle applicazioni, contribuendo così, in maggior misura, al successo e alla versatilità di progetti software in vari settori e ambiti di utilizzo.

Dopo il completamento del processo di parsing, i dati estrapolati diventano utili solo quando vengono presentati all'utente. Questa fase, risultante dal parsing, è fondamentale per il processo di visualizzazione. La visualizzazione dei dati consente agli utenti di comprendere e interagire con le informazioni in modo più significativo, facilitandone la comprensione, l'analisi e la presa di decisioni basate su tali informazioni. Attraverso varie tecniche di presentazione, come tabelle, grafici, mappe o altri elementi visivi, i dati elaborati possono essere resi accessibili e comprensibili agli utenti finali, fornendo un valore aggiunto alle applicazioni e ai sistemi che li utilizzano.

Nel Capitolo 3 è presentato Knime, uno strumento di low code con cui si può creare un processo di estrazione, elaborazione e visualizzazione di dati senza scrivere linee di codice.

# Visualizzazione

Il parsing dei dati implica l'analisi e l'interpretazione delle informazioni in un formato specifico, come testo strutturato o dati tabellari. Dopo il completamento del parsing, i dati estratti devono essere resi comprensibili all'utente attraverso la fase di visualizzazione.

Quest'ultima consiste nell'organizzare e presentare i dati in modo chiaro e significativo, utilizzando grafici, tabelle, diagrammi o altri strumenti visivi per rendere più intuitiva la comprensione delle informazioni.

Tra il processo di parsing e di visualizzazione vi sono diverse fasi da effettuare:

* Caricamento dati: è essenziale per raccogliere i dati da una fonte esterna e predisporli per l'elaborazione. Questa fase comporta il trasferimento delle informazioni nel sistema e la loro adeguata preparazione per l'analisi successiva. Dopo il caricamento, i dati sono pronti per essere elaborati attraverso il parsing e, alla fine in ultima istanza, per essere visualizzati in modo chiaro e comprensibile all'utente.
* Preparazione dati: è un passaggio cruciale nel processo di acquisizione delle informazioni da fonti esterne. Questa fase include la trasformazione dei dati grezzi in dati aggregati (vedere il capitolo 2) e il loro adeguamento per l'analisi successiva. Una volta completata questa fase, i dati sono pronti per essere sottoposti al parsing, dove vengono analizzati e strutturati in modo appropriato.
* Realizza grafica: i dati ottenuti nel parsing vanno trasformati. In questa fase avviene la generazione dei grafici per visualizzare le informazioni in modo chiaro e intuitivo.
* Genera grafica: la generazione dei grafici riveste un ruolo centrale nel rendere le informazioni più chiare e più intuitive. Questo processo implica l'utilizzo di strumenti e tecniche grafiche per rappresentare i dati in svariati formati, come grafici a barre, a torta, a dispersione, e altri ancora. La capacità di creare grafici efficaci è fondamentale per comunicare i risultati dell'analisi in modo comprensibile e consentire agli utenti di interpretare rapidamente i dati.

La visualizzazione dei dati può essere automatizzata attraverso sistemi predefiniti, che possono essere attivati al ricevimento dei dati di input e operare sia sul client che sul server.

Questi sistemi possono essere attivati al ricevimento dei dati di input e operare automaticamente sia sul client che sul server. L’attivazione può essere innescata da un evento nei database, sistemi temporizzati basati su cadenze giornaliere, settimanali o ecc.

La creazione di sistemi di visualizzazione che operino attraverso i passaggi precedentemente descritti o sistemi automatici richiede ambienti di sviluppo dedicati.

Un esempio rilevante in questo contesto è rappresentato da Knime, che si configura come uno strumento efficace per la realizzazione di applicazioni autonome e automatiche.

## Knime

Knime (Konstanz Information Miner) è un progetto open-source per l’analisi dei dati e l’integrazione di flussi di lavoro. Creato nel 2004 all’Università di Konstanz e rilasciato pubblicamente nella metà del 2006.

Offre un ambiente visuale drag-and-drop per la creazione, l'esecuzione, l'integrazione e l'automazione di analisi dei dati e processi di machine learning. Consente agli utenti di combinare in modo molto semplice dati provenienti da diverse fonti, eseguire analisi statistiche, applicare algoritmi di machine learning e visualizzare i risultati in modo chiaro e comprensibile.

Knime, grazie a queste sue caratteristiche, rientra nella categoria no-code (vedere paragrafo 2.6).

Knime utilizza due concetti importanti: il nodo e il flusso di lavoro.

L’interfaccia utente di Knime è composta da sei finestre principali ognuna con un proprio scopo. Tali finestre sono visibili una volta avviata l’applicazione di Knime.

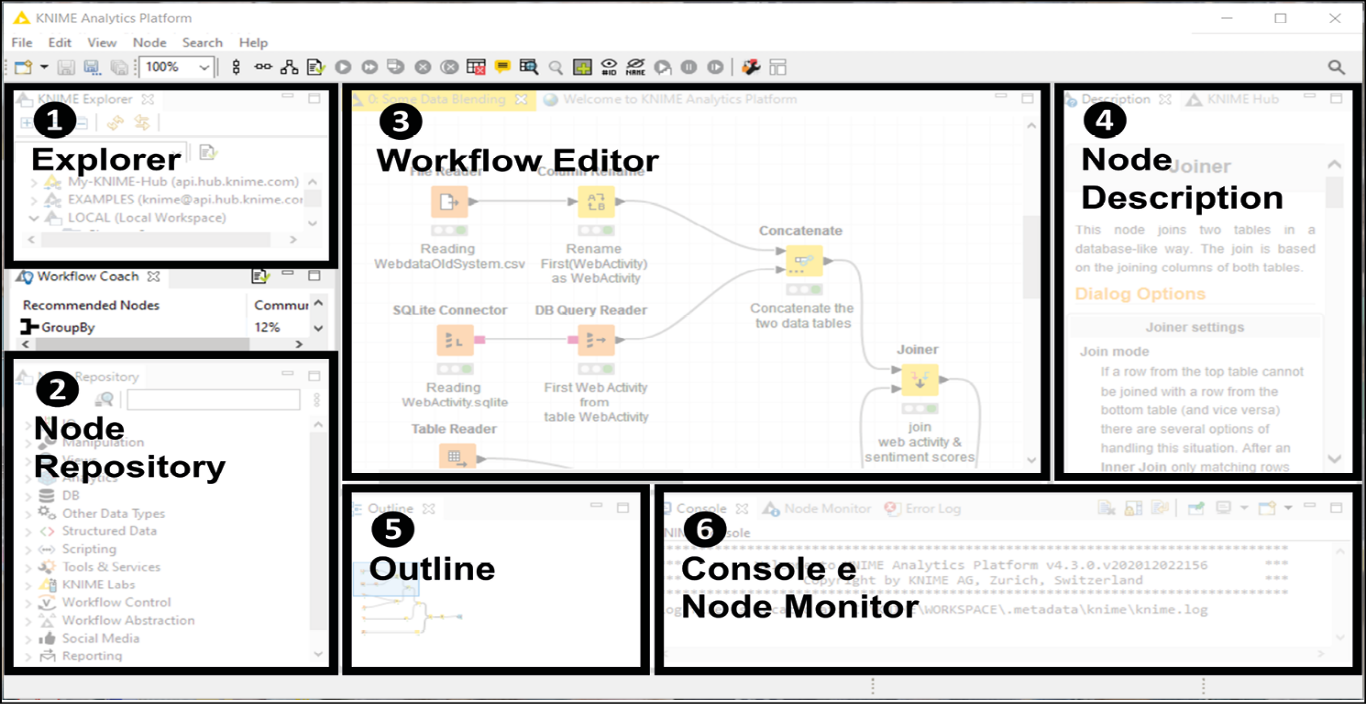


Figura 1 Interfaccia dell'ambiente di lavoro di Knime

1. Explorer: è dove i diversi flussi di lavoro sono conservati e organizzato in una struttura gerarchica di cartelle. Tra le varie cartelle che sono presenti nell’Explorer vi sono: il server pubblico Knime con diversi flussi di lavoro di esempio strutturati per argomento da trarre per ispirazione e riadattare alle nostre esigenze; lo spazio My-Knime-hub, collegato al nostro spazio utente sulla piattaforma cloud chiamate Knime Hub, dove possiamo condividere o pubblicamente a tutti o renderli privati e visibili solo all’utente.
2. Node Repository: in questo spazio si trova il menu completo dei nodi Knime disponibili e pronti per essere usati nel flusso di lavoro. Sono organizzati per categorie e a volte per sottocategorie.
3. Workflow Editor: qui i nodi vengono trascinati e usati per creare un flusso di lavoro. Rappresenta uno spazio bianco dove è possibile creare ogni tipo di flusso di lavoro, sulla base delle proprie esigenze o sulla base dei propri obbiettivi.
4. Node Description: è una finestra molto utile perche offre una guida pratica per l’utilizzo di ciascun nodo all’interno di Knime. Cliccando su un nodo qualsiasi, che sia presente nel Workflow Editor o che sia presente nel repository dei nodi, apparirà tutto quello che c’è da sapere per usare al meglio il nodo. La descrizione sarà composta da tre parti: un riassunto di che cosa fa e come funziona, elenco dei vari passi necessari alla configurazione e, infine, descrizione di quali dati ci si aspetta di avere alle porte di ingresso e di uscita del nodo, le porte.
5. Outline: flusso di lavoro può raggiunge ampi volumi ed estendersi oltra il Workflow Editor. La finestra di Outline permetta una vista a volo d’uccello del flusso di lavoro e mostra quale sezione è attualmente visualizzata nell’editor. Trascinando con il mouse il riquadro blu è possibile muoversi sulla parte del workflow che è di interesse.
6. Console e Node Monitor: in questa sezione ci trovano due strumenti utili per capire se ci sono dei problemi nella fase di esecuzione del flusso di lavoro ed eventualmente correggerli attraverso attività di debug.

In Figura 2 è rappresentato un esempio di flusso di lavoro (o workflow), all’interno del Workflow Editor di Knime che inizia leggendo un file Excel e successivamente procede con un processo di analisi e manipolazione dei dati.

L’esempio permette di dare una migliore prospettiva di come funziona Knime e del flusso di lavoro che in questo caso va da sinistra a destra.

Il flusso di lavoro è costituito da nodi, rappresentati da un’icona quadrata con del testo e delle forme intorno.

In cima al nodo vi è il suo nome in grassetto. Il nome rappresenta la funzione di quel nodo. Se per esempio, si dovesse leggere in file Json usiamo il nodo Json Reader.

Nella parte inferiore è possibile scrivere un commento. Il commento permette di dare una spiegazione o contesto al nodo all’interno del flusso di lavoro. Di default Knime crea un comento in ogni nodo aggiunto al flusso di lavoro usando un semplice contatore. Ad esempio: Nod1, Node2 e così via. Con un doppio clic si può modificare facilmente.

I nodi sono collegati attraverso connettori o delle porte, presenti a sinistra e a destra del quadrato del nodo. Per convenzione la parte sinistra del nodo indica l’input e la parte destra indica l’output.

Tali porte possono avere forma e colore diverso a seconda del tipo di contenuto che le attraversa: di forma triangolare indica il trasporto di tabelle di dati o dataset e rappresentano la maggior parte di esse ricadono in questa categoria, di forma quadrata trasportano modelli statistici, connessioni remote o immagini o di forma triangolare per le variabili.

Immagine che contiene schermata, diagramma, linea, testo

Descrizione generata automaticamente

Figura 2 Esempio workflow editor

Nella parte inferiore di ogni nodo, si trova un semaforo che segnala lo stato del nodo. Se la luce rossa del semaforo è accesa indica che il nodo non è ancora pronto per fare il suo lavoro, potrebbe essere che alcuni dati richiesti in input non siano disponibili o che sia necessario qualche ulteriore fase di configurazione. Luce gialla, il nodo ha tutto ciò che gli serve ed è pronto per essere pronto per essere eseguito, non appena gli viene dato l’ordine di esecuzione. Luce verde, indica che il nodo è stato eseguito con successo e i risultati sono disponibili per i connettori di uscita. Possono apparire anche altre icone sul semaforo:

* Triangolo giallo con un punto esclamativo, segnala un warming, un potenziale errore. Rappresenta un insolito e potenziale errore che richiede la nostra attenzione.
* Cerchio rosso con una croce annuncia invece un errore che bloccante. Non permette quindi al nodo di essere, in alcun modo, eseguito. In questi casi, con il puntatore del mouse per un secondo, appare un’etichetta con le informazioni o vedendo nei messaggi sulla console di Knime.

Nel repository di Knime sono disponibili diverse famiglie di nodi disponibili e ognuna delle quali risponde a una diversa classe di esigenza. I più popolari sono:

* Input e Output: questa categoria di nodi portano i dati all’interno e all’esterno di Knime comportandosi da sistemi di Input e sistemi di Output. Tipicamente i nodi di input sono all’inizio del flusso di lavoro e servono ad aprire file in diversi formati(CSV, Excel, immagini, pagine web, pdf e altro) o a connettersi a database remoti o locali ed estrarre i dati di cui si ha bisogno. I nodi di questa categoria hanno la funzione di iniziare il flusso di lavoro che termina con quelli di output.
* Manipolazione: i nodi appartenenti a questa categoria permettono la manipolazione di tabelle di dati e trasformarle secondo le esigenze dell’utente. Si possono aggreggiare, combinare, ordinare, filtrare e rimodellare le tabelle, ma anche gestirne i valori mancanti normalizzando le scale e convertire i tipi di dato, come xml a Json, ma anche gestire i valori mancanti, normalizzare le scale e convertire i tipi di dato. I nodi appartenenti a questo gruppo e a quelli di Input e Output sono sempre presenti all’interno di un flusso di lavoro. I nodi della Manipolazione possono avere più di una porta di ingresso o più di una porta di uscita, essendo in grado di fondere più tabelle insieme o di suddividerle in più parti.
* Analytics: sono nodi con una componente più complessa in quanto sono in grado di generare modelli statistici e di implementare algoritmi di apprendimento basati sui sistemi di intelligenza artificiale. Alcune porte dei nodi di questa categoria sono quadrate.
* Flow Control: i flussi di lavoro avranno bisogno di andare oltre il semplice andamento lineare da sinistra a destra scorrendo in sequenza da un nodo a un altro. I nodi appartenenti a questa categoria sono in grado di controllare il flusso di esecuzione attraverso l’implementazione dei cicli, dei loop controllati, in modo da poter ripetere dei passi più volter similmente alle strutture di controllo dei linguaggi di programmazione, come gli “if”, “for” e i “do/while”. Si può controllare dinamicamente il comportamento dei nodi gestendone la configurazione con l’uso di variabili. Con i nodi di controlli si può controllare i cicli e le variabili e implementare logiche più complesse anche se le esigenze di progetto consento di farne a meno.
* Altre tipologie: oltre alle tipologie viste fino ad ora, vi sono molteplici tipi di nodi che permettono di aiutare l’utente nelle esigenze più specifiche. Alcuni nodi ci permettono per esempio di interagire con applicazione terze attraverso Api. Altri nodi permettono di integrare i linguaggi di programmazione come il Javascript nel flusso di lavoro o al contrario di innestare il flusso di lavoro preparato in Knime all’interno di un programma scritto in un altro linguaggio. Ci sono anche nodi per test statistici e per costruire visualizzazioni o generare report più complessi.

## Visual Studio Code

Visual Studio Code (VS Code) è un editor di codice sorgente creato e sviluppato da Microsoft per sistemi come Windows, Linux e macOS e per browser web. Rappresenta un ecosistema integrato di sviluppo e mette a disposizione un numero alto di strumenti e funzionalità per il processo di creazione dei software.

Come editor, VS Code è rivolto a sviluppatori aventi diverse competenze e background e di conseguenza si presta a diversi linguaggi di programmazione, tra cui Html, javascript, Dart, Java, Python e molti altri (vedere Capitolo 2). Si presta anche piattaforme diverse. Il che permette lo sviluppo di applicazioni per desktop, web e mobile. Permette anche la gestione di database e la creazione di sistemi basati su apprendimento come sistemi di Intelligenza Artificiale.

In Figura 3 è rappresentata l’interfaccia di avvio di VS Code.

L’interfaccia utente di è progettata per massimizzare la produttività degli sviluppatori, offrendo a loro una esperienza più intuitiva e maggiormente personalizzabile . l’editor di codice è dotato di avanzate funzionalità, come l’auto completamento del codice, la formattazione automatica e la segnalazione degli errori in tempo reale. Questo permette agli sviluppatori di scrivere codice in modo più efficiente e accurato.

Tra gli strumenti di VS Code vi è anche un efficiente sistema di controllo del codice sorgente integrato che supporta i principali sistemi di versionamento come GIT, SVN e TFS. Questo è un importante aiuto ai gruppi di sviluppo dandogli maggiore efficienza, tiene traccia delle modifiche del codice e facilitando la gestione dei rami di sviluppo.

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Figura 3 l'interfaccia di avvio di Visual Studio Code

Per quanto riguarda il testing, Visual Studio offre una vasta gamma di opzioni per il supporto integrato per i test di unità con framework. Questo dà l’opportunità agli sviluppatori di eseguire test automatici per verificare la correttezza del codice e garantire una maggior integrità delle applicazioni.

VS Code include anche strumenti avanzati per il debugging consentendo agli sviluppatori di individuare e risolvere in modo più rapido eventuali errori nel codice. Tali strumenti offrono funzionalità come il tracciamento dei processi, il monitoraggio delle variabili e l’analisi dei crash, facilitando il processo di debugging e ottimizzando il software.

L’interfaccia utente di VS Code offre una serie di strumenti per la creazione di interfacce grafiche intuitive e accattivanti. Di questi strumenti includono designer visivi per la creazione di form e controllo, nonché strumenti per la progettazione di layout e stili.

Infine, VS Code è dotato di un’ampia gamma di estensioni e plug-in disponibili attraverso il Visual Studio Marketplace. L’insieme delle estensioni e dei plug-in costituisce un aspetto molto importante per lo sviluppatore, quasi come un vero e proprio tesoro. Oltre a offrire le funzionalità di base, queste estensioni consentono di personalizzare ulteriormente l’ambiente di sviluppo secondo le esigenze specifiche dei progetti e degli sviluppatori stessi.

In figura 4 è possibile vedere il marketplace di VS Code a sinistra dell’immagine e le informazioni di un plug-in selezione sulla destra.

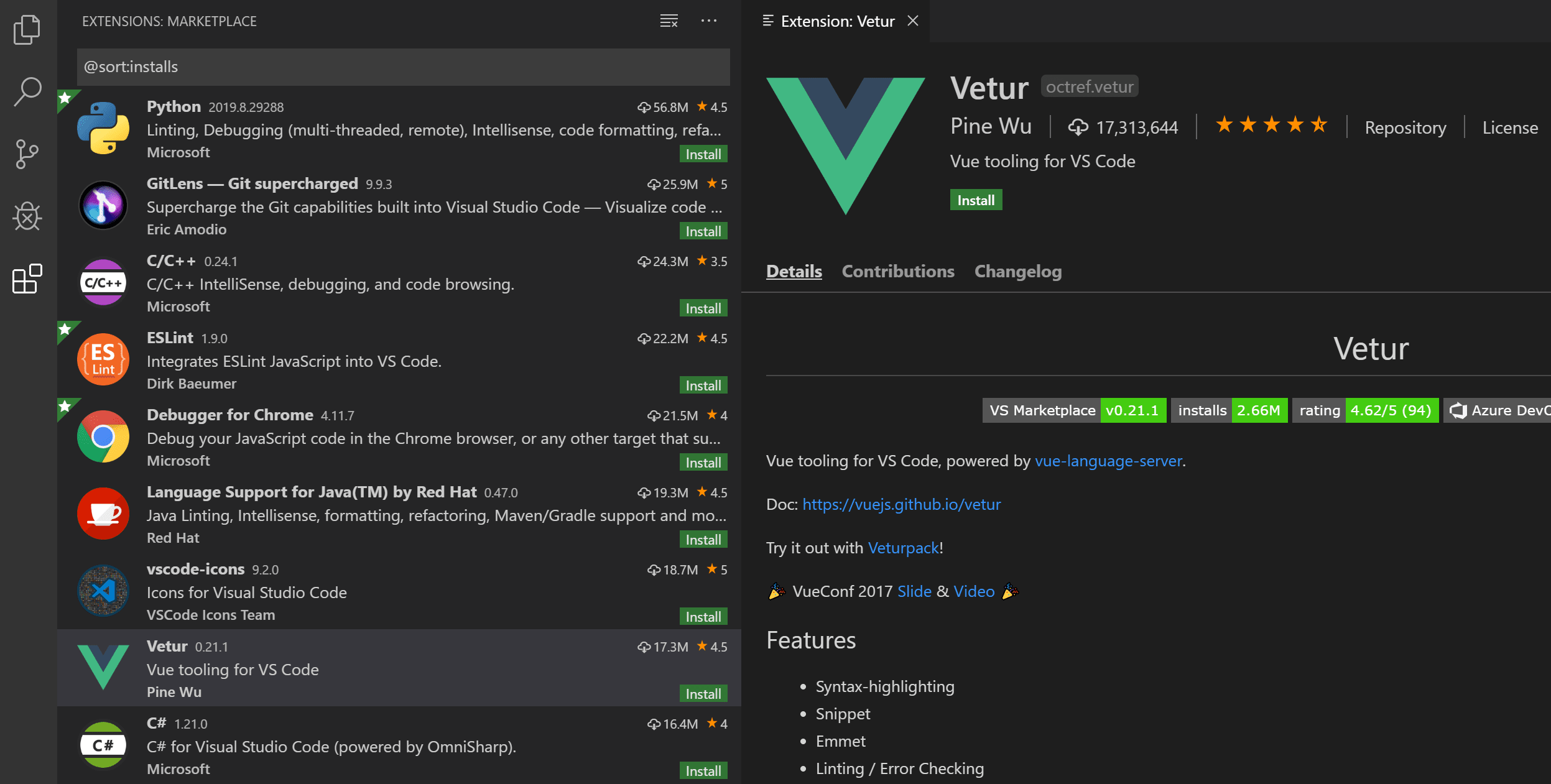


Figura 4 esempio di usa del marketplace di Visual Studio Code

Tra le stensioni più popolari, si trovano quelle che migliorano la produttività, come estensioni per l’auto completamento del codice o per la gestione delle dipendenze del progetto. Queste estensioni aiutano gli sviluppatori a risparmiare tempo ea scrivere codice più pulito ed efficiente.

Oltre tutto questo, VS Code integra una interfaccia a riga di comando, un terminale, che permette di interfacciarsi con i gestori di pacchetti, di usare compilatori come Dart (vedere capitolo 2.4) e l’uso di strumenti come Flutter (vedere capitolo 2.6) o di ulteriori pacchetti software che possono essere usati a riga di comando.

## Firebase

Firebase è una piattaforma software per la gestione di applicazioni web e mobile attraverso un completo set di servizi cloud in backend.

Firebase è stata fondata nel 2010 ed in seguito è stata acquisito da Google. Successivamente è stata integrata successivamente nei servizi di cloud di Google mantenendosi autonoma. In questo modo fornisce servizi che sono presenti sia in Firebase che in GCP, il cloud di Google.

In Figura 5 è visibile la schermata principale che permette l’accesso alle funzioni di Firebase. Tale accesso è subordinato ad una registrazione/accesso dell’utente con un proprio account.

Immagine che contiene testo, schermata, software, Pubblicità online

Descrizione generata automaticamente

Figura 5 schermata di ingresso di Firebase

Come molte piattaforme cloud, Firebase offre servizi sia in formato gratuito sia a pagamento attraverso un sistema di pagamento al consumo.

Immagine che contiene testo, schermata, software

Descrizione generata automaticamente

Figura 6 dashboard di un progetto di Firebase

In Figura 6 è visibile una dashboard di un progetto di un utente all’interno di Firebase. Alla sinistra è possibile vedere i servizi di Firebase raggruppati in quattro categorie.

Oltre alla piattaforma web, Firebase ha sviluppato un’interfaccia a linea di comando, Firebase Command Line Interface (Firebase CLI) e si può installare anche attraverso un gestore di pacchetto sul proprio pc.

Firebase CLI è un imporporante strumento che semplifica il lavoro degli sviluppatori interagendo con i loro progetti all’interno di Firebase tramite comando di riga di comando.

Con Firebase CLI è possibile automatizzare una serie di compiti, riducendo il carico di lavoro manuale e migliorando l’efficienza nello sviluppo e nel deployment delle applicazioni, attraverso una interazione diretta dei progetti presenti su Firebase .

Dalla configurazione iniziale dei progetti su Firebase alle distribuzioni di siti web su Firebase Hosting, alla gestione delle regole di accesso per i database real-time e del Cloud Firestore, la Firebase CLI copre una vasta gamma di funzionalità inerenti Firebase.

Una dei comandi più usati di Firebase CLI è il comando ‘firebase init’. Tale comando consente di inizializzare un nuovo progetto Firebase o di aggiungere nuove funzionalità legate ai servizi di Firebase ad un progetto esistente. In questa fase si può selezionare il servizio Firebase che si vuole usare e di conseguenza configurando l’ambiante di sviluppo in modo più rapido e maggiormente personalizzato.

Può semplificare il processo di deployment delle applicazioni all’interno di Firebase Hosting. Dopo aver completato los viluppo della tua applicazione o sito web, si effetua il processo di deployment attraverso il comando ‘firebase deploy’ per distribuirla su Firebase Hosting in modo sicuro e rapido.

Firebase Hosting è una delle funzionalità maggiormente usate su Firebase. È un servizio di hosting di Firebase. Consente agli sviluppatori di distribuire facilmente siti web e applicazioni web statiche e dinamiche con funzionalità più avanzate. Firebase Hosting offre un’esperienza utente più intuitiva integrando altri servizi Firebase come l’autenticazione degli utenti, il database in tempo reale e le funzioni cloud, anche le funzionalità avanzate tra cui certificati SSL automatici. Firebase Hosting offre una scalabilità automatica per la gestione del traffico in modo più efficiente e affidabile.

Firebase CLI ha funzionalità più complesse che vanno oltre la semplice configurazione e il deployment della app Firebase. Tali funzionalità consentono agli sviluppatori di migliorare, testare e ottimizzare le proprie applicazioni in modo più completo ed efficace.

Tra di esse vi è la gestione delle regole di sicurezza per il database in tempo reale di Firebase. Da linea di comando si possono definire ed aggiornare successivamente le regole di acceso al database su cloud dando accesso solo agli utenti autorizzati la possibilità di leggere e scrivere dati aventi informazioni sensibili. Questo aspetto è importante per mantenere la sicurezza e l’integrità dei dati.

Firebase CLI supporta anche l’esecuzione di test di unità e l’integrazione per le funzioni di Cloud Functions. Le Cloud Functions consentono agli sviluppatori agli sviluppatori di creare ed eseguire codice, di un linguaggio di programmazione, in risposta agli eventi generati dai servizi usati in Firebase o da altri servizi cloud. Attraverso Firebase CLI si può scrivere con maggior facilità le funzioni dando una maggior certezza che si comportino correttamente in diverse situazioni.

Il monitoraggio delle prestazioni e delle analisi app è un’altra funzionalità chiave offerta dalla Firebase CLI attraverso strumenti integrati come Firebase Performance Monitoring e Firebase Analytics. Quest’ultimi consentono di identificare e risolvere eventuali problemi di prostrazioni, nonché di avere una maggior comprensione del comportamento degli utenti, delle funzionalità più usate e in quale contesto, e delle loro iterazioni.

Firebase CLI come strumento rappresenta un importante alleato nelle mani degli sviluppatori poiché le sue funzioni rispecchiano in toto i servizi di Firebase. Se i servizi di Firebase sono posizionati in backend, le funzioni di Firebase CLI equivalgono a frontend. In generale possiamo dire che Firebase CLI sia una estensione su lato frontend di Firebase e dei suoi servizi.

Tra i servizi più usati di Firebase, oltre a quelli citati, vi sono:

* Firebase Authentication: fornisce l’autenticazione degli utenti attraverso diverse modalità di accesso, tra cui e-mail/password e/o l’autenticazione tramite social media come Google, Facebook, Twitter, etc., e autenticazione tramite telefono.
* Cloud Firestore: è un database NoSQL in tempo reale che offre una struttura di dati più flessibile e maggiormente scalabile. Memorizza e sincronizza i dati dell’app in tempo reale su dispositivi client.
* Realtime Database: è un database cloud NoSQL in tempo reale che consente di sincronizzare i dati dell’app in tempo reale tra gli utenti e il server. Utile per applicazioni avente funzioni in tempo reale come chat, giochi e app con funzioni di gruppo.
* Firebase Storage: Un servizio di Hosting di file che consente di caricare e scaricare file come immagini, documenti e video direttamente da app Firebase, offrendo maggior scalabilità e maggior affidabilità.
* Firebase Cloud Messaging: servizio di messagistica push che consente di inviare notifiche push a utenti iOS, Android e Web in modo rapido ed efficiente.
* Firebase Remote Config: consente di personalizzare il comportamento e l’aspetto senza dover rilasciare una nuova versione, consentendo di testare e ottimizzare le funzionalità dell’app in modo dinamico.
* Firebase Test Lab per Android/iOS: un ambiente di test che consente di eseguire test automatizzati su app Android e iOS su dispositivi reali e virtuali, identificando problemi e garantendo la qualità dell’app.
* Firebase App Distribution: è un servizio che distribuisce più facilmente le versioni beta della app iOS e Android ai tester al fine di raccogliere feedback e migliorare la qualità dell’app prima del suo rilascio.

L’insieme di questi servizi permette a Firebase, insieme a Firebase CLI, di essere uno strumento valido e fondamentale nelle mani dello sviluppatore.

## Git e GitHub

Durante lo sviluppo di un programma, durante la fase di programmazione e di testing, il testo del codice viene più volte riscritto. All’interno di un team di sviluppo, alcuni programmatori possono scrivere o riscrivere lo stesso porzione di codice o eventualmente modifica codice connesso a quella porzione di codice. Per evitare il sovrapporsi di variazioni conflittuali di codice, si utilizzano metodologie che evitano tale fenomeno dando un controllo sugli aggiornamenti del codice. Queste categorie di software si chiamano ‘software di versione distribuito’ ed uno dei più noti è Git.

Git è un software open-source creato da Linus Torvalds nel 2005. Inizialmente sviluppato all’interno dell’ambiente open-source per superare molte delle criticità di altre alternative, è cresciuto fino ad acquisire notorietà il tutto il mondo. È ampiamente utilizzato per tenere traccia delle modifiche al codice sorgente, permettendo a più sviluppatori di lavorare contemporaneamente senza sovrapporsi.

Il repository (repo) è il cuore di Git. Si tratta di un archivio che contiene tutte le versioni di un progetto, sia in locale sul proprio computer che su un server remoto accessibile via internet.

Clonare un repository significa creare una copia esatta del repository remoto sul proprio computer. Una volta clonata, è possibile creare, modificare e cancellare file nel repository locale. Ogni modifica viene registrata tramite un commit, che cattura lo stato del progetto in un determinato momento. Ogni commit è identificato univocamente da un hash SHA-1.

Il concetto di branch (ramo) è essenziale in Git. Permette di lavorare su diverse versioni del progetto in parallelo. Il ramo principale è solitamente chiamato "main" o "master", ma si possono creare altri rami per sviluppare nuove funzionalità, risolvere bug o sperimentare senza influire sul ramo principale.

Per condividere le modifiche tra repository locali e remoti, Git utilizza comandi come push e pull. Il comando push invia le modifiche dal repository locale a quello remoto, mentre il comando pull aggiorna il repository locale con le modifiche presenti nel repository remoto.

Il merge è un processo di fusione delle modifiche da un ramo all'altro, risolvendo eventuali conflitti che possono sorgere quando due versioni dello stesso file vengono modificate in modi diversi.

GitHub è una piattaforma che utilizza Git per fornire servizi di hosting (veder capitolo 3.3) per repository Git, arricchita con strumenti che facilitano la collaborazione e la gestione dei progetti software. Fondata nel 2008 e acquisita da Microsoft nel 2018, GitHub è diventata indispensabile per sviluppatori e team di sviluppo in tutto il mondo.

Una delle caratteristiche più importanti di GitHub è la capacità di ospitare repository Git all’interno di server remoti. Questo semplifica la collaborazione tra sviluppatori in diverse località sparse nel mondo. L'interfaccia utente del sito web di GitHub è facile da usare e consente di visualizzare il codice, monitorare le modifiche effettuate, eseguire dei merge e gestire i branch, direttamente dal browser e senza dover utilizzare la riga di comando.

La Pull Request è uno degli strumenti più utili e più importanti offerti da GitHub. Consente agli sviluppatori di proporre modifiche al progetto principale e di richiedere una revisione da parte dei collaboratori prima che le modifiche vengano integrate nel progetto.

Questo processo migliora di molto la qualità del codice e facilita la collaborazione e la discussione sulle modifiche proposte.

GitHub offre anche funzioni di issue tracking, che permettono di segnalare bug, richiedere nuove funzionalità e monitorare i progressi del progetto.

Inoltre, GitHub Actions consente di automatizzare i flussi di lavoro di sviluppo, come la compilazione del codice, l'esecuzione di test e la distribuzione del software, integrandosi perfettamente nel ciclo di vita del progetto.

GitHub Pages permette di ospitare siti web statici direttamente da un repository, rendendolo uno strumento versatile anche per la documentazione e la presentazione dei progetti.

La comunità open source è particolarmente attiva su GitHub, con migliaia di progetti disponibili pubblicamente, favorendo la condivisione delle conoscenze e la collaborazione a livello globale. Oltre a comunità open-source e le imprese anche i programmatori amatoriali, con un background tecnico e professionisti trovano in Git e GitHub uno strumento molto versatile e utile.

## D3 e SVG

D3.js (Data-Driven Documents) è una libreria JavaScript che offre un ampio set di strumenti per la creazione di visualizzazioni dati dinamiche e interattive sul web. Usa come base standard web come HTML, SVG e CSS.

D3 consente agli sviluppatori di collegare facilmente i dati a elementi DOM (vedere 2.2) e di manipolare il documento in modo efficiente per generare grafici, mappe, diagrammi e altre visualizzazioni personalizzate.

SVG, ovvero Scalable Vector Graphics, è un formato di grafica vettoriale basato su XML utilizzato per descrivere grafici bidimensionali e interattivi.

SVG è supportato nativamente dai browser moderni ed è particolarmente adatto per la creazione di visualizzazioni dati a causa della sua scalabilità, della sua flessibilità e della sua capacità di interazione.

Con SVG, è possibile creare grafici di alta qualità che mantengono la loro nitidezza e chiarezza anche quando ingranditi o ridimensionati, il che lo rende un'opzione popolare per la creazione di visualizzazioni dati su web.

D3 sfrutta SVG per la creazione di molte delle sue visualizzazioni, poiché SVG offre una serie di vantaggi chiave, tra cui la capacità di disegnare forme personalizzate, tracciare linee e curve, applicare trasformazioni e aggiungere interattività agli elementi grafici.

Grazie alla combinazione di D3 e SVG, gli sviluppatori hanno la facoltà di creare visualizzazioni dati estremamente sofisticate e coinvolgenti che si adattano dinamicamente ai dati e alle azioni degli utenti.

La visualizzazione dei dati è importante perché aiuta a capire meglio le informazioni estratte dai dati durante il parsing. Questo processo coinvolge diverse fasi: prima si caricano i dati, poi li si prepara per l'analisi, e infine si creano grafici e altre immagini per mostrare i risultati in modo facile da capire.

Knime è uno strumento utile perché permette di fare tutto questo con molta facilità andando incontro alle esigenze dell’utente e migliorando la user-experience nel suo complesso. Tutto questo grazie alla sua interfaccia facile da usare e molti strumenti che sono a disposizione. Con Knime, è possibile unire dati da varie fonti, fare calcoli statistici, mostrare i risultati in grafici chiari e molto altro ancora.

Costituisce un supporto fondamentale sia per coloro che sono esperti nel campo, sia per chi si avvicina per la prima volta all'analisi dei dati e oltre aspetti. Esplora diverse aree, quali l'intelligenza artificiale, i sistemi automatici, la creazione di API e il supporto alla programmazione, offrendo un'ampia gamma di strumenti e conoscenze per affrontare queste sfide in modo efficace.

Ma nonostante sia uno strumento utile per chi non ha un background tecnico essa non è priva di difetti e compromessi.

Mentre strumenti come Visual Studio Code vanno intesi come soluzioni a diversi problemi che coloro che hanno un background possono usare a pieno. Oltre a questo, Visual Studio Code, Firebase e Git ( e GitHub) vanno intesi come strumenti importanti nelle mani degli sviluppatori. Non vanno intesi come strumenti indipendenti e isolati ma come un connubio vincente non solo per quanto riguarda il processo di parsing ma anche il processo di visualizzazione.

D3 e SVG permettono una migliore visualizzazione dei dati analizzati producendo rappresentazioni infografiche dinamiche tramite la manipolazione del DOM (vedere 2.2)

I temi che introdotti fino ad ora, sono rappresentati attraverso una spiegazione introduttiva e, per certi aspetti, teorica del funzionamento del parsing, dei linguaggi di programmazione e dell’ambiente in cui vengono eseguiti. Ora invece si parlerà dell’atto pratico di questi temi attraverso spiegazione passo-passo dei passaggi e di eventuali scelte progettuali che sono state prese e il perché.

Nel capitolo 4 viene introdotto il lavoro di progetto e indicate le scelte che ne sono state derivate.

# Lavoro svolto

Nei capitoli precedenti sono stati introdotti diversi argomenti aventi come punto focale i Sistemi Informativi e della loro capacità/necessità di comunicare tra di loro scambiandosi informazioni. La scambio di informazioni avviene per mezzo di diversi formati, come il formato JSON. Quando un Sistema Informativo riceve un file JSON, come input da un altro Sistema informativo, si avvia un processo di parsing. Tale processo permette di estrarre le informazioni dal formato JSON. Il processo di parsing può avvenire indipendentemente dal linguaggio di programmazione o dalla piattaforma nel quale avviene il processo.

Qui di seguito si introdurrà un sistema di parsing che simulerà in toto un processo di parsing utilizzando un'applicazione web (Web App).

## Web App

Per il processo di parsing e la successiva visualizzazione, si è optato per la realizzazione di un'applicazione web (Web App).

La Web App usa come framework Dart (vedere capitolo 2.4) con delle parti di Javascript (vedere capitolo 2.3).

L’applicazione usa come hosting il servizio offerta da Firebase Hosting (vedere capitolo 3.3). In questo modo è possibile connettersi ad essa da ogni dispositivo avente connessione alla rete. L’indirizzo al quale connettersi è ‘https://parsingjson.web.app’.

Il progetto della applicazione web utilizza come strumento di gestione di gestione degli aggiornamenti Git e tiene un repository su GitHub (vedere capitolo 3.4) all’indirizzo: <https://github.com/domenico1995/JsonDart.git> per il momento presente su una cartella pubblica a tutti gli utenti.

La scelta è derivata dalla necessità di testare l’applicazione al di fuori del dispositivo di localhost, dove essa viene sviluppata e dalla necessita di valutare il comportamento dell’applicazione sia su PC (laptop e desktop) sia su dispositivi mobili con dimensioni varie, per individuare eventuali difetti di progettazione.

Il compito dell’applicazione è quello di:

1. Ricevere dati in input, come testo JSON, e di effettuare un processo di parsing e di visualizzare sotto forma di testo di output.
2. Ricevere dati in input, come testo JSON non ben formato, e visualizzare in output il testo secondo il formato JSON aderente alla RFC 8259.
3. Ricevere dati in input, come testo JSON, e di rappresentarlo, mediante usa di un elemento grafico, come albero che si sviluppa ad orientamento orizzontale.

Immagine che contiene testo, schermata

Descrizione generata automaticamente

Figura 7 Immagine Web App

In Figura 7 è possibile vedere l’immagine l’applicazione a schermo su un desktop.

Il progetto della Web App è composto da file di configurazione e directory con sottodirectory, divise per funzionalità come segue:

* .dart\_tool: è una directory creata automaticamente dal compilatore di Dart. All’interno vi sono gli strumenti di Dart: il SDK di Dart e il framework Flutter. La directory contiene i file e le informazioni utilizzati per lo sviluppo, la compilazione e l’esecuzione di app scritti in Dart.
* .firebase: viene usata nei progetti che integrano i servizi del cloud di Firebase di Google (). La directory contiene configurazioni e informazioni che sono necessarie per interagire con il cloud di Firebase durante lo sviluppo e la distribuzione dell’applicazione.
* .vscode: è utilizzato da Visual Studio Code (), per memorizzare le configurazioni delle specifiche di progetto. Queste informazioni sono presenti all’interno del file ‘launch.json’.
* build: una volta che la Web App è testate testata nelle sue componenti, è possibile creare la ‘build’. La build contiene una versione della Web App da distribuire all’interno di un hosting. Oltre al codice, vi è anche le informazioni contenute nei file di configurazione e le immagini.
* images: è una directory contenete le immagini utilizzate dalla Web App.
* file di configurazione: alcuni file contengono le informazioni sulle dipendenze usate nel progetto, per documentare il progetto e le informazioni di Firebase e di Git. L’insieme di questi file sono vitali per realizzare il progetto e della sua configurazione che vengono poi racchiusi, in parte nella build, del progetto.
* web: la directory ‘web’ contiene il cuore e le funzioni vere e proprie della Web App. Le funzioni aventi compiti specifici sono presenti all’interno di una sottodirectory, ‘utils’. All’interno della cartella web, oltre ad utils, vi sono:
  + index.html: il file html contiene gli elementi che saranno visualizzati sullo schermo una volta effettuata la connessione all’indirizzo della Web App. All’interno comandi nel linguaggio di html e collegamenti ad altri file, come ad esempio, per i fogli di stile in css e i file in javascript (script.js e i file presenti nella sottocartella utils: tab.js e albero.js) e dart (main.dart).
  + main.dart: il file funge da main principale per il controllo e la gestione della maggior parte delle funzioni che sono utilizzate nella Web App e del flusso di controllo. Rappresenta il centro nevralgico della maggior parte delle funzioni associate alla Web App.
  + script.js: il file contiene una funzione che modifica il DOM spostando il diveder tra le due parti della Web App, quella che funge da input e quella che funge da view delle funzioni centrali della Web App, cioè da output.
  + styles.css: il file contiene il foglio di stile usato dal Browser durante l’esecuzione del file html. All’interno non vi sono le informazioni usate dalla funzione albero.js presente invece in tree.css.
  + tree.css: il file contiene il foglio di stile usato per funzione albero.js. tale funzione permette di visualizzare ad albero i dati di input. La parte grafica dell’albero visualizzato è definita in questo albero.

Oltre ai file descritti c’è una cartella, ‘utils’ con all’interno i file contenenti le funzioni usati per compiti specifici. Uno per ogni file. È sono composti nel seguente modo:

* + albero.js: contiene la funzione javascript per generare un grafico ad albero con le informazioni fornite in input. Per la creazione del grafico viene utilizzato la libreria D3 e l’uso di SVG (vedere capitolo 3.5) per creare un grafico più fluido e iterativo. La funzione utilizza come foglio di stile tree.css. In Figura 7 è possibile vedere l’esito di un testo dato in input a sinistra e a destra il grafico creato dalla funzione.
  + aprifile.dart: contiene il codice in Dart che permette di caricare un file da sistema locale e inserire nella Web App come dati di input.
  + conversion.dart: una volta forniti un testo JSON di input avviene il processo di estrapolazione dei dati, il parsing, che vengono successivamente mostrati in output sulla Web App.
  + format.dart: contiene la funzione che permetta a un testo JSON in input di formattare bene il testo di input (vedere capitolo 2.1). il risultato è visibile in output
  + tab.js: la Web App ha tre funzioni: ‘formatta’, ‘conversione’ e ‘albero’. Ognuna delle funzioni è all’interno di un proprio spazio. In Figura 7 è possibile vedere la pagina principale della Web App, ad esempio la funzione ‘Albero’, come precedente descritto. Il file si occupa della gestione e della visualizzazione a schermo dello spazio di output di esse. È possibile vedere una solo funzione alla volta

La descrizione dei file usati nel progetto, delinea solo una descrizione del loro compito, il loro obbiettivo finale o le loro informazioni. Ma descrive solo file per file solo il loro funzionamento. Mentre, nel loro insieme risulta più articolato e più complesso.

L’applicazione web è composta da due insiemi di elementi grafici e di funzioni. Tali insiemi si possono raggruppare in: Input e Output.

Per l’insieme di Input è composto:

* Textarea di ingresso: la textarea permette di inserire il testo JSON al suo interno. All’avio dell’applicazione è presente un testo JSON per testare il funzionamento dell’applicazione stessa. Tale testo è presente all’interno del codice stesso di index.html. La textarea ha come identificativo: id=”input” nel file Html.
* Bottone per i file: il bottone attiva il file aprifile.dart presenta nella sottocartella utils di web. In questo modo è possibile caricare un testo all’interno della textarea che funge in seguito da input. All’interno del file Html, bottone ha come identificativo: id=”apriFile” e testo: “Apri File”.
* Bottone per pulire: il bottone permette di pulire sia la textarea “input” sia le textarea usate come output. Ha come identificativo: id=”pulireTutto” e testo: “Pulire Tutto”.

Mentre per quanto riguarda l’insieme di Output, gli elementi sono definiti come segue:

* Bottone per la formattazione: il bottone è connesso dal file main.dart al file format.dart. Il bottone attiva la funzione di formattazione del testo Json preso dalla textarea “input”. All’interno del file Html, bottone ha come identificativo: id=”conversione” e testo: “Conversione”.
* Bottone per la conversione: il bottone è connesso dal file main.dart al file conversion.dart. Il bottone attiva la funzione che effettua il processo di parsing del testo fornito in input alla textarea “input”. La funzione di conversione crea un testo simile del testo di input anche se il testo di ingresso non è ben formato.
* Bottone per l’albero: il bottone è connesso direttamente con il file albero.js. Una volta premuto, il codice presente all’interno nel file javascript procede, in sequenza, a prendere il testo di “input”, formattare, estrarne il contenuto, crearne un albero di nodi e collegamenti, ed infine a mostrarlo. ”. All’interno del file Html, il bottone ha come identificativo: id=”albero” e testo: “Albero”.
* Textarea per la formattazione: la textarea permette di visualizzare il testo formato dal codice presente in format.dart. La textarea non permette di modificare il testo che risulta solo come output. All’interno del file Html, la textarea ha come identificativo: id=”outputFormatta”.
* Textarea per la conversione: la textarea permette di visualizzare il testo convertito dal codice presente in conversion.dart. La textarea non permette di modificare il testo che risulta solo come output. All’interno del file Html, la textarea ha come identificativo: id=”output”.
* <div> per grafico albero: per visualizzare il grafico dell’albero si è scelto usare come riferimento un <div>. La scelta è derivata dalla volontà di creare con maggiore libertà di manovra rispetto a quella espressa da un textarea. Il grafico risultante è rappresentato come un albero che si sviluppa in orizzontale. Ogni nodo dell’albero può essere ridotto ed eventualmente ritornare alla sua precedente forma. I nodi-foglia non possono essere ridotti. L’espansione di un nodo porta l’albero allo stato precedente e di conseguenze se ci sono dei nodi ridotti in quella porzione di albero, rimangono. L’elemento <div> in questione è chiamato

A gestire la visualizzazione delle due textarea e dell’elemento div per l’albero è una funzione presente nel file tab.js, è si chiama openTab.

Immagine che contiene testo, schermata, schermo, software

Descrizione generata automaticamente

Figura 8 codice per la gestione dei tab

In Figura 8 è possibile vedere il codice in javascript della gestione della sezione di Output attraverso l’uso dei tab. In questo modo è possibile vedere il risultato delle funzioni connesse ai bottoni uno alla volta.

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Figura 9 Variabili di Input di main.dart

Immagine che contiene testo, schermata, software, Sistema operativo

Descrizione generata automaticamente

Figura 10 Corpo della funzione main.dart

I compiti, funzioni e gli elementi grafici della applicazione web definiti in precedenza sono assolti come segue:

* Per la conversione, l’1): tale compito viene svolto dal codice presente in conversion.dart. Per inserire il testo JSON di input da convertire è possibile scrivere direttamente sulla textarea ‘input’ oppure caricare il testo da un file presente nel dispositivo attivando il file aprifile.dart una volta premuto il pulsante ‘Apri File’. Una volta fatto questo è possibile effettuare la conversione del testo secondo standard JSON. Per la gestione dell’evento generato dal bottone, se ne occupa main.dart. In Figura 9 è possibile vedere i collegamenti tra le variabili presenti in main.dart e gli elementi grafici del file index.html. In Figura 10 è possibile il corpo principale della funzione main del file main.dart. Si intravede l’azione che viene attivata dal bottone ‘conversione’ collegata alla variabile locale ‘conversione’ di main.dart. questo evento prende il testo di input, svuota la textarea dove visualizzare il risultato e passa il testo di input alla funzione leggi del file conversion.dart con il comando: leggi(contenuto, “”);

Il risultato sarà visualizzato direttamente sulla textarea ‘output’.

Immagine che contiene testo, schermata, schermo, software

Descrizione generata automaticamente

Figura 11 funzione di formattazione del file format.dart

* Per la formattazione, il 2): è uguale al 1) per quanto riguarda la ricezione del testo di input e l’attivazione del file per la formattazione. Si differenze, di conseguenza, nell’attivazione diretta del codice presente in format.dart. In Figura 11 è possibile vedere il funzionamento della formattazione.
* Per la visualizzazione ad albero, il 3): la ricezione del testo di input è uguale al 1) e al 2). Come definito all’interno dello spazio utilizzato per visualizzare il grafico, questo compito risulta il più complesso e articolato perché il processo di visualizzazione contiene anche il processo usato per la conversino e la formattazione. L’albero che viene generato utilizza D3 come librerie e SVG come area per la creazione della stessa (vedere capitolo 3.5).

Per la funzione di conversione presente nel file conversion.dart si è scelto di creare delle funzioni specifiche che permettono l’analisi del testo di input e stampare nella textarea il risultato. Tali funzioni presenti nel file hanno compiti specifici descritti come segue:

* “leggi”: accetta una stringa JSON e una spaziatura come parametri. Prima corregge la stringa JSON utilizzando la funzione fixJson, poi la decodifica in un oggetto dinamico. Se il risultato della decodifica è una lista (List), invoca la funzione processJsonArray per elaborarla; se invece è una mappa (Map<String, dynamic>), chiama la funzione processJsonObject per gestirla. Se il contenuto non è né una lista né una mappa, aggiorna il valore di output per indicare che il JSON non è valido. In caso di errore durante la decodifica, stampa un messaggio di errore con il contenuto JSON che ha causato il problema.
* processJsonArray: itera attraverso un array JSON e processa ciascun elemento. Se l'elemento è una mappa (Map<String, dynamic>), chiama un'altra funzione chiamata processoJsonObject, passando l'elemento e una chiave aggiornata. Se l'elemento non è una mappa, converte il valore in una stringa e controlla se deve essere modificato o semplicemente aggiunto all'output. Se necessario, modifica e legge il testo con formattazione aggiornata, altrimenti aggiunge il valore formattato direttamente all'output.
* processoJsonObject: accetta una mappa JSON, una chiave genitore e uno spazio di formattazione. Itera attraverso le coppie chiave-valore della mappa. Se il valore è una stringa, lo formatta e lo aggiunge all'output con una nuova spaziatura; altrimenti, aggiunge direttamente il valore all'output.
* controlloTesto: verifica se una stringa contiene parentesi graffe o quadre. Se sì, restituisce vero, altrimenti falso.
* modificaTesto: esegue una sequenza di operazioni sulla stringa di testo per adattarla al formato JSON. Trasforma i marcatori dei valori nelle virgolette appropriate, corregge gli spazi e i caratteri di apertura e chiusura delle parentesi.
* fixJson: corregge una stringa JSON. Sostituisce le virgolette non precedute da un backslash con sequenze di escape, e risolve i casi in cui sono presenti duplicati di chiavi con due punti.

Queste funzioni hanno un applicazione ricorsiva che termina una volta esaminato completamente il testo JSON fornito.

## Modello di Knime

Similmente a quanto realizzato con la Web App, è stata realizzato un progetto di Knime con un flusso di lavoro che permette l’estrazione dei dati da un file JSON (vedere capitolo 2).

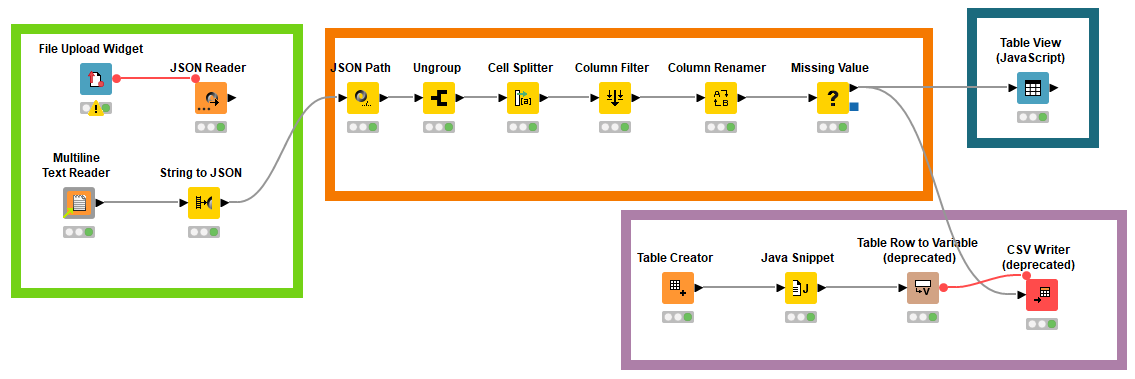


Figura 12 Immagine del flusso su Knime

Il progetto permette, la lettura e la serializzazione, di un testo JSON. I dati estratti dal testo JSON vengono filtrati ed elaborati nel flusso di lavoro. Una volta terminato, i dati vengono salvati in un file csv o visualizzati in forma di tabella.

In Figura 12 è possibile vedere il flusso di lavoro del progetto di Knime e dei nodi che sono stati utilizzati.

Il flusso di lavoro di Knime però è parametrato secondo una struttura di file JSON.

Di conseguenza, qualsiasi altro file JSON in input avrebbe problemi nell’esecuzione del flusso.

Il contenuto e l’organizzazione del file JSON è descritto come segue:

* "numero\_ordine" ( numero di ordine)
* "cliente" (i valori dell'utente): "nome", "email", "telefono", "indirizzo".
* "articolo" (i valori dell'articolo): "id\_prodotto" (id del prodotto), "nome", "prezzo\_unitario" (prezzo totale).
* e altri valori correlati: "metodo spedizione" (metodo di spedizione), "costo spedizione" (costo di spedizione), "costo totale" (costo totale),

"metodo\_pagamento" (metodo di pagamento), "stato\_pagamento" (stato del pagamento), "stato\_ordine", (stato dell’ordine), "data\_ordine" (data dell’ordine), "data\_consegna\_prevista" (data della consegna) e le "note".

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Figura 13 esempio di formato Json accettabile.

In Figura 13 è possibile vedere un formato di testo JSON compatibile per l’estrazione ed elaborazione del progetto di Knime.

Sono presenti dei file di esempio nella cartella di test del repository del progetto di Knime: “https://github.com/domenico1995/JsonDart/tree/main/Knime/Test”.

I nomi dei valori citati in precedenza vanno scritti senza errori all’interno del file o del testo JSON. Il flusso di lavoro o modello di Knime, effettua un matching con i nomi descritti precedentemente.

Qualora non dovesse individuare, almeno, uno dei nomi dei valori, cercherà la prima corrispondenza di quel nome all’interno del file o del testo. In caso di mancanza di matching, di un nome del valore, darà come esito valore “null”. Se l’errore del nome di un valore, ad esempio, riguarda “cliente” anche i valori all’interno come “nome”, “email”, “telefono” e “indirizzo” daranno valore “null” poiché lo stesso effetto ricadrà sui valori all’interno.

Come descritto nel capitolo 3.1, Knime utilizza dei nodi per poter eseguire dei compiti o funzioni. Tali compiti o funzioni si raggruppano in categorie e sottocategorie. I nodi appartenenti ad una categoria possono essere ricondotti ad essa o ad una sottocategoria, per via del colore di sfondo che hanno. I nodi di colore arancione appartengono alla categoria del Input e Output (IO) e a diverse sottocategorie come quelle che permettono la lettura dei file (Read). I nodi dei colori gialli

In Figura 12 è possibile vedere il flusso di lavoro del progetto nella sua interezza.

Il progetto è suddiviso in diverse parti e ognuna è delimitata da un riquadro di colore diverso come bordo. Ogni parte ad un suo compito e funzionalità inerente al progetto: Input, Elaborazione, Visualizzazione e Salvataggio.

La sezione di Input ha il compito di svolgere il compito di inserire i dati di ingresso.

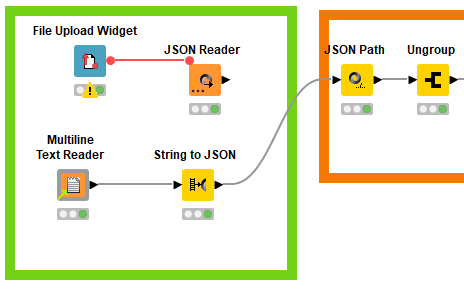


Figura 14 sezione di Input del modello.

All’interno del riquadro verde presente nella Figura 13 vi sono i nodi che permettono di dare il testo o file di JSON. Si ha, di conseguenza, la possibilità di selezionare il file di testo o di fornire un testo JSON come input principale del progetto.

Immagine che contiene testo, schermata, linea, diagramma

Descrizione generata automaticamente

Figura 15 sezione di Elaborazione del modello.

La sezione centrale con il riquadro arancione, presente nella Figura 14, si occupa della fase di estrazione ed elaborazione dei dati dal file/testo di input. Questa sezione è di Elaborazione.

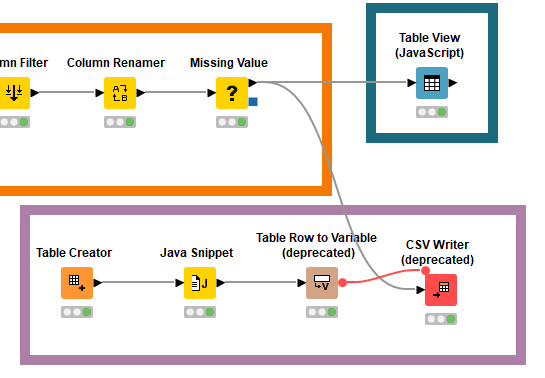


Figura 16 sezione di Visualizzazione e Salvataggio del progetto.

Alla fine del processo di estrazione ed elaborazione dei dati, la sezione di Elaborazione, le informazioni vengono fornite all’utente.

Questo passaggio viene effettuato con due sezioni distinti ed in contemporanea. Tali sezioni sono: Visualizzazione e Salvataggio.

In Figura 16 sono presenti due riquadri: uno viola per la sezione di Salvataggio ed uno blu per la sezione di Salvataggio.

La sezione di Salvataggio è composta da uno nodo, “Table View Javascript”, che ha la funzione di mostrare attraverso una view i dati elaborati in forma tabellare.

La Sezione di Output permette il salvataggio dei dati elaborati inserendoli all’interno di file csv. Il nome del file ha al suo interno la data e l’orario dell’esecuzione del flusso di lavoro.

Qualora più dati sono stati analizzati nello stesso orario, non verranno sovrascritti ma messi in coda all’interno del file.

Come definito in precedenza, ci sono due modi per fornire l’input al modello: fornire il testo o selezionando il file JSON.

Ognuno delle quali necessito di operazioni preliminari descritti come segue prima dell’esecuzione:

1. Fornire il testo: collegare il nodo “String to JSON” al nodo “JSON Path”. In Figura 14 è presente tale collegamento. Configurare il nodo “Multiline Text Reader” inserendo il testo JSON. Il flusso di lavoro del progetto di Knime tiene di default questa opzione e quindi è necessario solo inserire il testo nel nodo “Multiline Text Reader”.
2. Selezionare il file: collegare il nodo “JSON Reader” al nodo “JSON Reader”, premere il tasto destro sul nodo “File Upload” e selezionare "Interactive View: File Upload". Si aprirà una finestra con un bottone giallo con scritto "Select File", premendolo vi permetterà di scegliere il file. Va selezionato il file da esaminare. Una volta fatta bisogna chiudere scegliendo l'opzione "Aplly settings as new default" come illustrato nella Figura 17. In questo modo il file sarà selezionato e pronto per l'esecuzione.

Il modello indipendentemente dalla scelta di selezionare il file o fornire il testo va configurato in base al dispositivo usato per eseguire il programma di Knime e il progetto appena descritto.

Immagine che contiene testo, elettronica, schermata, software

Descrizione generata automaticamente

Figura 17 Immagine finestra di chiusura per il salvataggio del nodo File Upload Widget

Queste configurazioni si basano sull’inserimento di file di default per il nodo “File Upload Widget” e la cartella di output nel nodo “Table Creator”, dove salvare i file elaborati. Tali configurazioni vanno effettuati come segue:

* Per il nodo “File Upload Widget”: nella dicitura di configurazione “Default File” va inserito il file o selezionato un file, che sarà di default per l’esecuzione del progetto.
* Per il nodo “Table Creator”: va inserito il percorso della cartella dove verranno salvati i file. Nella Figura 18 è possibile vedere un esempio con un percorso verso una cartella. Una volta selezionato il percorso presente in Row0 basta modificare il testo presente nella cella o eventualmente nella casella Input line quando la cella di Row0 viene selezionata.

Immagine che contiene testo, software, Software multimediale, Icona del computer

Descrizione generata automaticamente

Figura 18 immagine configurazione nodo Table Creator.

La sezione di Elaborazione effettua la parte di processo di parsing e filtraggio delle informazioni. Tale processo costituito da una serie di nodi avente ognuno un compito diverso. I nodi sono descritti come segue:

* JSON Path: il nodo ha il compito di filtrare i campi del testo JSON tramite matching dei nomi. In questa fase, eventuali nomi dei valori, vengono riscontrati.
* Ungroup: i campi individuati vengono divisi i due gruppi. L’assegnazione di un campo verso un gruppo avviene manualmente sulla base delle necessita del progettista.
* Cell Splitter: divide il contenuto delle celle in una tabella. Per file JSON, usa "," come delimitatore per separare i dati all'interno delle celle in più colonne.
* Column Filter: per compito simile al Ungroup, divide i campi in due parti. Include alcuni ed esclude altri. Ungroup è nodo che esplode le colonne contenenti dati aggregati, creando nuove righe per ciascun elemento all'interno di una colonna aggregata (come liste o colonne JSON). Column Filter è un nodo seleziona quali colonne mantenere o rimuovere da una tabella, permettendo di filtrare i dati in base alle colonne.
* Column Rename: rinomina i campi con altri nomi. I nuovi nomi saranno usati nei nodi successivi del modello.
* Missing value: individua i valori mancanti nei valori associati ai nomi. Una volta individuato il valore mancante, verrà inserito la stringa “null”.

Il processo di parsing e di visualizzazione trova piena applicazione all’interno del progetto della Web App e del modello su Knime. Anche se effettuato con metodiche diverse. Nel caso della Web App il processo utilizza funzioni e testo in codice per l’analisi e la visualizzazione del testo fornito in input.

Mentre per Knime, non c’è codice o funzioni. Il processo è composto da nodi che agiscono in maniera simili. Ogni nodo ha un suo compito e in maniera inversa sulla base della funzione/compito troviamo il nodo che serve.

Questo approccio diverso sta alla base della differenza tra Web App e modello di Knime.

# Conclusioni e sviluppi futuri

Possibile visualizzazione:   
sito: https://bvjebin.github.io/JSONTree/  
git: <https://github.com/bvjebin/JSONTree>

# Ringraziamenti