Strumenti Software e Tecnologie Utilizzate

L'obiettivo principale di questo documento è fornire un'ampia panoramica dei software e delle tecnologie utilizzate nel corso del nostro progetto di sviluppo del software. Questo capitolo non solo evidenzierà gli strumenti fondamentali che abbiamo impiegato, ma offrirà anche consigli per affrontare la modifica di questo progetto o simili, specialmente per coloro che intraprendono un cammino senza profonde conoscenze pregresse, proprio come nel nostro caso.

Abbiamo imparato che intraprendere un progetto senza precedenti esperienze può essere una sfida significativa. Tuttavia, abbiamo abbracciato una filosofia basata sulla condivisione e sulla regola d'oro di "trattare gli altri come vorresti essere trattato tu". Questi principi hanno guidato il nostro approccio al lavoro in team, aiutandoci a superare ostacoli e a facilitare il percorso per chi verrà dopo di noi, accelerando il processo di apprendimento e aiutare a creare un ambiente di lavoro collaborativo, sereno e stimolante.

Nel corso di questo capitolo, esploreremo in dettaglio i software e le tecnologie fondamentali utilizzate, fornendo anche consigli e informazioni per comprenderne l'importanza all'interno del contesto del progetto. Abbiamo riconosciuto la necessità di questa documentazione esaustiva, poiché spesso mancano spiegazioni dettagliate sulle risorse software in progetti e documentazioni tecnologiche, nonché una chiara descrizione delle loro funzioni e utilità nell'ambito del progetto. Riteniamo che tale approccio semplifichi l'accesso a risorse cruciali, eliminando le barriere di ingresso per chi si approccia a questo progetto, essendo alle prime armi.

Ciò non toglie che, per comprendere appieno l'ecosistema di un progetto e sviluppare competenze durature, è essenziale 'toccare con mano' queste risorse e tecnologie. L'esperienza pratica rimane fondamentale per acquisire una comprensione approfondita, poiché teoria e pratica si completano a vicenda.

Microsoft Teams

Microsoft Teams è stato uno strumento essenziale per facilitare la comunicazione in tempo reale all'interno del nostro team sfruttando le funzionalità di messaggistica istantanea e videoconferenza per agevolare la collaborazione a distanza. Nonostante la comodità di questo strumento, si è sempre preferito e incentivato l’incontro del team di persona, poiché crediamo fermamente che lo sviluppo di software tragga enormi benefici dall'interazione diretta. In un ambiente universitario, le discussioni faccia a faccia stimolano il problem solving e l'apprendimento condiviso. L'interazione fisica consente scambi più spontanei di idee, la risoluzione dei problemi, rafforzando al contempo il senso di appartenenza e il coinvolgimento, promuovendo una maggiore sinergia e motivazione nel raggiungimento degli obiettivi del progetto.

Microsoft OneDrive

Microsoft OneDrive è stato utilizzato per la condivisione di file e documenti in modo rapido ed efficiente. Grazie alla licenza fornita dall'università, abbiamo beneficiato di un ampio spazio di archiviazione fino a 5TB, consentendoci di accedere ai file da qualsiasi dispositivo. Inoltre, abbiamo sfruttato le funzioni di condivisione e commento di Microsoft 365 (Word) per migliorare la collaborazione tra i membri del team. Si ritiene che il lavoro di un membro del team debba sempre essere a disposizione di tutti gli altri, facilitando così l'accesso alle risorse del progetto in modo trasparente e promuovendo una maggiore coesione e condivisione delle conoscenze.

Visual Paradigm

Abbiamo fatto un ampio uso di Visual Paradigm come strumento di modellazione dei diagrammi per creare e modificare vari tipi di diagrammi UML. Questi ci hanno permesso di visualizzare chiaramente la struttura del software e di documentare in modo dettagliato la struttura e le funzionalità dell'applicazione.

Riguardo questo software va notato come anche piccoli dettagli come caricare una cartella contenente i file .vpp creati da un gruppo, possa far risparmiare ore di tempo a un team che deve riutilizzare un diagramma apportandogli delle modifiche, ma che non ha a disposizione il file sorgente che il team precedente ha generato.

Miro

Abbiamo sfruttato i template Miro di tabella, planning e multistick notes per organizzare il diario, pianificare le attività e per quanto concerne le storie utente.

Abbiamo imparato che uniformare gli stili per user stories, piani, casi d'uso e altri documenti rende la comunicazione più chiara e riduce il rischio di confusione.

Inoltre, tenere un diario ci ha aiutato a monitorare il nostro progresso e a comprenderne i tempi. I futuri team possono beneficiare del tempo richiesto dal nostro team per effettuare alcune attività per valutare quanto tempo impiegheranno per attività simili.

Visual Studio Code

Visual Studio Code è stato il nostro ambiente di sviluppo principale per scrivere, gestire il codice del progetto e gestire le dipendenze. È stata utilizzata anche la funzione di debugging per identificare e risolvere i problemi di codice. Questo editor è stato arricchito con diverse estensioni che ci hanno aiutato a ottenere i risultati desiderati. L'integrazione con GitHub è stata particolarmente utile, poiché ci ha permesso di coordinare i vari branch direttamente dall'ambiente di editing del codice.

Abbiamo imparato che commentare il codice man mano che lo scriviamo semplifica la comprensione per noi stessi e per gli altri. Non bisogna attendere di avere una montagna di codice da commentare; piuttosto, è meglio inserire commenti passo dopo passo.

Il riuso del codice è fondamentale quando si lavora tra più team, è responsabilità di chi sviluppa una porzione di codice commentarla adeguatamente. Un team che inizia il proprio lavoro sulla base di materiale precedentemente creato deve poter avere una visione chiara del codice attraverso commenti mirati. Nella pratica, infatti, in un ambiente lavorativo sarà raro creare funzionalità ex novo anzi spesso si parte da un progetto preesistente che qualcun altro ha iniziato a sviluppare.

diffchecker.com è stato impiegato in modo particolare per i file di codice particolarmente estesi, con l'obiettivo di garantire che nessuna modifica fosse sfuggita all'attenzione. Il processo di segnalare e confrontare le modifiche tra la versione originale e quella finale è stato essenziale per assicurare l'integrità e l'accuratezza del codice. Questa pratica meticolosa di verifica ha contribuito a mantenere la coerenza e la qualità del prodotto finale.

GitHub

Git è stato uno strumento chiave per tenere traccia delle modifiche al codice, collaborare in modo efficiente e garantire la gestione delle versioni del software in modo coerente. Con Git, ogni membro del team poteva lavorare sui propri rami (branches) in modo indipendente, senza intaccare il codice principale, consentendo a ciascun membro del team di caricare, condividere e sincronizzare le modifiche in modo coordinato. Questo approccio ha favorito lo sviluppo parallelo, la sperimentazione e il mantenimento di una cronologia dettagliata delle modifiche.

È stato possibile condividere in maniera efficiente il codice e realizzare uno sviluppo incrementale suddiviso in più step è stato impiegato il servizio di hosting GitHub, il quale ha permesso al team di eseguire un lavoro di gruppo sul codice e di ottimizzare l'impiego della tecnica di pair programming.

Questo strumento ci ha permesso di mantenere un flusso di lavoro efficiente, facilitando la collaborazione e garantendo che il nostro codice fosse sempre allineato e affidabile, facilitando la gestione dei problemi e la revisione del codice.

Si è notato che un progetto dovrebbe avere una struttura di cartelle ben organizzata, unita a nomi chiari e uniformi per i container, in modo da non essere disorientati e utilizzando tempo e risorse a cercare file o cartelle.

Maven

Maven è uno strumento di gestione dei progetti Java. Grazie a Maven, la gestione delle dipendenze è stata semplificata e automatizzata, garantendo la coerenza e l'affidabilità del processo di sviluppo. Inoltre, Maven ci ha consentito di integrare agevolmente il progetto con altri strumenti di sviluppo, facilitando ulteriormente il processo di sviluppo e distribuzione dell'applicazione. La sua integrazione nel progetto è utile in quanto Maven permette di specificare le librerie necessarie nel file di configurazione "pom.xml" e si occupa di scaricarle automaticamente.

Il "Project Object Model" (POM) è un file XML che contiene le informazioni fondamentali per la gestione di un progetto in Maven. Nel file POM di Maven che definisce la struttura del progetto, gli artefatti sono specificati tramite i tag "groupId", "artifactId" e "version", identificano univocamente l'artefatto generato dal progetto. Inoltre, il POM specifica le dipendenze del progetto, ovvero le librerie esterne necessarie per la compilazione, insieme alle configurazioni dei plugin di Maven, che influenzano il comportamento dei plugin durante la compilazione, il testing e la distribuzione del progetto.

È buona norma tenere in grande considerazione la compatibilità: studiare preventivamente le compatibilità tra software e versioni utilizzate dai diversi team è cruciale. Usare una versione senza considerare cosa stiano usando gli altri può causare problemi di integrazione e debug difficili. Comunicare e sincronizzare le scelte software è essenziale.

Per scaricare Maven bisogna andare su https://maven.apache.org/download.cgi, scaricare il file binario e seguire le istruzioni per l'installazione.

Per utilizzare le istruzioni di Maven è necessaria l'installazione di JavaJDK, pacchetto software essenziale per lo sviluppo di applicazioni Java (https://www.oracle.com/java/technologies/downloads/); va inoltre effettuata l'aggiunta delle variabili di ambiente di Java e di Maven.

Per fare questo segue un esempio (**nomi e path possono variare**): bisogna andare su Pannello di controllo -> Sistema e sicurezza -> Sistema -> Impostazioni di sistema avanzate (sulla destra) -> Variabili d'ambiente. Su variabili di sistema Nuova … -> Nome: JAVA_HOME Valore: C:\Program Files\Java\jdk-21 -> OK. Nuova … -> Nome: MVN_HOME Valore: C:\Users\apache-maven-3.9.4 -> OK.

A questo punto modificare la variabile di sistema Path tramite doppio click -> Nuovo -> %JAVA_HOME%\bin -> Nuovo -> %MVN_HOME%\bin. Per verificare la corretta installazione aprire il prompt dei comandi e digitare "mvn -v". Potrebbe essere necessario il riavvio della macchina.

Nel nostro caso quando veniva modificato un file .java (nella cartella src) si è notato che il relativo file target (nella cartella target) non veniva aggiornato correttamente. Questo poiché era necessario nella main folder avviare, tramite il prompt, il comando "mvn clean install", in questo modo le modifiche effettuate saranno effettivamente visibili una volta montata l'applicazione su docker. (Si noti che nel caso il comando mvn non venga riconosciuto bisogna controllare, o eventualmente riscrivere, le variabili d'ambiente).

Un ultima nota riguarda il fatto che Maven può essere integrato su Visual Studio Code come plug in.

WSL

WSL (Windows Subsystem for Linux) per l'esecuzione di Docker in ambiente Windows, garantendo un'ottima integrazione tra i componenti del sistema.

L'installazione di WSL e Docker è trattata nel file ReadMe del repository GitHub.

Docker

Docker è la piattaforma di containerizzazione per la distribuzione dell'applicazione su server e dispositivi remoti. L'utilizzo di Docker agevola la creazione di un ambiente distribuito dedicato per ciascun componente dell'applicazione e ha semplificato la gestione delle risorse, contribuendo così all'efficienza del processo di deployment, garantendo la massima portabilità dell'applicazione indipendentemente dall'ambiente di esecuzione sottostante per creare e gestire container isolati.

All'interno del sistema operativo viene eseguita l'applicazione Docker Desktop, che si occupa della gestione delle istanze di tipo Docker.

Dal lato client, l'utente si serve di un qualsiasi browser come ambiente di esecuzione, e interagisce con l'applicativo sfruttando un'interfaccia implementata a partire da artefatti di tipo HTML, JavaScript e CSS. Dal lato server, l'applicazione impiega Docker come ambiente per l'esecuzione, in modo tale da evitare problemi di integrabilità. Lo scopo è quello di creare un container con un ambiente dedicato per ogni artefatto. Inoltre, dal lato server è necessario realizzare delle interconnessioni tra i vari container per permettere agli artefatti di comunicare tra di loro.

L'utilizzo di Docker permette di creare immagini e container che operano in ambienti isolati, consentendo l'esecuzione di servizi separati e la comunicazione tra di essi attraverso il mapping dei porti specifici. Questo approccio favorisce la modularità, la scalabilità e semplifica la distribuzione delle applicazioni, ottimizzando l'efficienza e la gestione delle risorse.

In Docker, ci sono tre componenti chiave: immagini, container e volumi. Le immagini Docker sono snapshot di applicazioni con tutte le loro dipendenze e configurazioni. I container Docker sono istanze in esecuzione di queste immagini, isolati e autosufficienti. I volumi Docker sono usati per memorizzare dati in modo persistente, anche dopo la chiusura di un container.

Nel contesto di Docker Desktop, è possibile gestire immagini, container e volumi direttamente sul sistema operativo locale, semplificando lo sviluppo di applicazioni containerizzate. Le immagini sono usate per creare container, che possono archiviare dati nei volumi. Questo sistema facilita la gestione delle applicazioni e assicura la persistenza dei dati tra le esecuzioni dei container.

Una volta effettuata una modifica a livello di file, e aggiornato il relativo target può essere necessario eliminare, sulla base della modifica, i container, volumi e immagini di uno o più container. Dopodiché le immagini dei container vengono scaricate e create con il comando “docker-compose build”, mentre “docker-compose up” avvia i container basati su queste immagini.

Bisogna infatti capire che essendo un container un’istanza indipendente, qualsiasi modifica all’esterno di Docker sarà trasparente fintanto che la stessa immagine di Docker non verrà aggiornata.

docker-compose.yml è un file di configurazione che definisce servizi, container e configurazioni per applicazioni Docker multi-container usando Docker Compose.

Dockerfile è un file che contiene istruzioni per creare un'immagine Docker personalizzata, definendo cosa includere nell'immagine e come configurarla.

Bisogna notare che è inoltre possibile eseguire il prompt di comandi interno ad un container (l’exec). Tuttavia, lavorare con l’exec di Docker e il prompt dei comandi del sistema operativo può comportare differenze significative dovute alle dipendenze software. Docker consente di creare ambienti isolati con le proprie dipendenze. Ad esempio, un container potrebbe avere Maven installato, permettendoti di eseguire comandi Maven al suo interno, ma se Maven non è installato sul proprio sistema host, non potrà essere usato nel prompt dei comandi del tuo computer. La scelta dipende dall'ambiente necessario per il progetto, quindi è essenziale essere consapevoli delle differenze.

Come ultima nota si fa notare come sia possibile modificare un file dell’istanza del container stesso dentro docker, dopo averlo individuato nella sezione file. Tuttavia, è una pratica sconsigliata poiché, oltre a non avere una UI chiara, le modifiche saranno relative alla sola istanza.

Servizi RESTful

I servizi RESTful sono un tipo di servizio web basato sul protocollo HTTP, progettato per consentire una comunicazione efficiente tra client e server. Questi servizi definiscono operazioni che possono essere eseguite sulle risorse del server utilizzando i verbi HTTP, come GET, POST, PUT e DELETE.

Ad esempio, l'annotazione `@PostMapping` è stata utilizzata per definire i metodi che gestiscono le richieste HTTP POST, mentre `@GetMapping` è stata utilizzata per definire i metodi che gestiscono le richieste HTTP GET.

Questi servizi RESTful hanno consentito al nostro sistema di comunicare in modo efficace utilizzando il protocollo HTTP standard. Grazie a questa architettura, il nostro sistema è stato in grado di esporre funzionalità specifiche attraverso API REST, facilitando l'interazione e lo scambio di dati tra client e server.

Menzioni finali

OpenCSV: una libreria utilizzata per gestire file CSV.

Dos2Unix: strumento di conversione utile quando si lavora con file di script shell su sistemi Unix o Linux (file .sh ad esempio), per garantire che i file ".sh" siano formattati correttamente per l'esecuzione su tali sistemi, soprattutto se sono stati scritti o preparati su sistemi Windows.

Junit 4: la versione di Junit compatibile con la versione 1.0.6 di Evosuite utilizzata.

Alberto Aimone M63/1508

Valerio Di DomenicoM63/1465

Genny Fedele M63/1422