https://datascience-project.eu/

Scheda didattica

Titolo:	Software
Parole chiave	SQL GitHub
Lingua e lingua	Italiano
Obiettivi/Traguardi/Esiti di apprendimento	1. **Capire quando usare SQL** 2. **Importanti campi di applicazione di SQL** 3. **I comandi SQL di base** 4. **Come sviluppare il codice insieme in un team** 5. **Funzionalità di base di GitHub**

Corso di formazione:	
Alfabetizzazione della scienza dei dati	
Visualizzazione dei dati e modulo di analisi visiva	
Software	X
Introduzione alla scienza dei dati per le scienze umane e sociali	
Scienza dei dati per il bene	
Giornalismo e narrazione dei dati	

• **Descrizione**	• Questo corso introdurrà brevemente i più importanti linguaggi di programmazione e strumenti che gli scienziati dei dati utilizzano quotidianamente. • Il contesto e lo scopo in cui vengono tipicamente utilizzati saranno delineati e verranno presentati i comandi più preziosi per i principianti: ○ **SQL** è diventata una pietra miliare della moderna gestione dei dati. In questo corso, esploreremo diversi

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

	modi in cui SQL può essere utilizzato per recuperare i dati dai database. ○ Discuteremo di cosa è **GitHub**, quali caratteristiche offre e come gli sviluppatori di software possono trarne beneficio. ● Al termine del corso, gli studenti conosceranno il campo di attività e i comandi più comuni.
Contenuti disposti in 3 livelli	**1. Introduzione a SQL** **1.1 Informazioni di fondo** **2.1.1 SQL: il linguaggio standard globale per la gestione e l'anali dei database** Structured Query Language (SQL) è un linguaggio di database ampiamente utilizzato che è progettato per gestire e manipolare database relazionali. È uno strumento potente che consente agli utenti di analizzare, manipolare ed elaborare i dati in vari modi, rendendolo una scelta popolare in molti settori e aree applicative, tra cui finanza, e-commerce, assistenza sanitaria e governo. Una delle caratteristiche principali di SQL è la sua flessibilità. Offre un metodo standard per la conservazione e l'elaborazione di grandi quantità di dati, rendendolo una prima scelta per le aziende che hanno bisogno di gestire grandi quantità di informazioni. La sua flessibilità consente inoltre di generare query e report personalizzati, fornendo agli utenti preziose informazioni sui dati con cui stanno lavorando. Oltre alla sua flessibilità, SQL è anche molto efficace nella sua capacità di elaborare e manipolare i dati in modo rapido ed efficiente. Con l'uso di istruzioni SQL complesse e tecniche di indicizzazione, SQL può individuare e recuperare rapidamente i

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

dati, il che è particolarmente importante quando si lavora con database di grandi dimensioni.

Un altro vantaggio di SQL è la sua natura facile da usare. Per la maggior parte degli utenti, è relativamente facile da imparare e da usare, il che lo rende una scelta popolare per le aziende e le organizzazioni di tutte le dimensioni. Ci sono anche numerose fonti disponibili per imparare la lingua, dai tutorial online e corsi a libri di testo e manuali.

Nonostante i suoi numerosi vantaggi, SQL ha alcune limitazioni. Ad esempio, non è sempre la scelta migliore per lavorare con dati non strutturati, come immagini, video o file audio. Inoltre, potrebbe non essere la scelta più efficace per alcuni tipi di analisi, come quelli che richiedono tecniche statistiche avanzate.

1.1.2 Finalità di SQL

Uno degli usi principali di SQL è quello di recuperare i dati da un database. Ciò può includere la selezione di colonne specifiche di dati, il filtraggio dei dati in base a criteri specifici o la combinazione di dati da più tabelle. Ad esempio, supponiamo che tu abbia un database di clienti e dei loro ordini. Con SQL, è possibile recuperare facilmente un elenco di tutti gli ordini per un particolare cliente o tutti gli ordini per un determinato periodo di tempo.

Un altro uso di SQL è quello di aggiungere nuovi dati a un database, modificare i dati esistenti o eliminare i dati che non sono più necessari. Questo può essere particolarmente utile quando è necessario aggiornare grandi quantità di dati contemporaneamente. Ad esempio, se è necessario aggiornare l'indirizzo di spedizione per tutti i clienti che vivono in un particolare codice postale, è possibile effettuare questa modifica rapidamente e facilmente con SQL.

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

Oltre alla gestione dei dati, SQL può essere utilizzato anche per creare e gestire intere tabelle nei database. Ciò include la creazione di nuove tabelle, la modifica delle tabelle esistenti e l'eliminazione di tabelle che non sono più necessarie. Ad esempio, se si desidera creare una nuova tabella per registrare i dati di vendita dell'azienda, è possibile utilizzare SQL per definire la struttura della tabella e specificare i tipi di dati per ciascuna colonna.

Infine, SQL è particolarmente adatto per l'elaborazione di grandi quantità di dati. Questo perché è progettato per un'elevata efficienza e può gestire query e operazioni complesse senza sforzo. Questo lo rende la scelta migliore per le aziende e le organizzazioni che hanno bisogno di gestire e analizzare grandi quantità di dati su base regolare.

1.1.3 Ci sono solo 4 cose essenziali di cui hai bisogno per fare un uso prezioso di SQL

Se vuoi iniziare a usare SQL, potresti chiederti quali strumenti e conoscenze hai bisogno per iniziare. Fortunatamente, hai solo bisogno di 4 cose essenziali.

In primo luogo, è necessario un sistema di gestione del database (DBMS). Un DBMS è un sistema software che consente di creare, gestire e manipolare database. Ci sono molti DBMS diversi, ma tra i più popolari sono MySQL, Oracle, PostgreSQL e Microsoft SQL Server. Questi sistemi offrono la possibilità di organizzare e archiviare i dati, nonché di accedere e manipolare tali dati utilizzando SQL.

La **seconda** cosa di cui hai bisogno è un database. Un database è una raccolta di dati organizzata in modo specifico per facilitare l'accesso e la modifica. Ci sono molti diversi tipi di database, ma i più popolari sono Oracle, PostgreSQL, MySQL e

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

SQL Server. È possibile scaricare il database open source PostgreSQL dal loro sito web: https://www.postgresql.org/

La **terza** cosa di cui hai bisogno è un client SQL. Un client SQL è uno strumento che consente di connettersi a un database ed eseguire istruzioni SQL. Ci sono molti client SQL diversi, ma i più popolari sono MySQL Workbench, SQL Developer e SQL Server Management Studio. In alternativa, è possibile utilizzare un linguaggio di programmazione come Java, Python o C# per eseguire istruzioni SQL per un database.

Quarto, e infine, hai bisogno di una conoscenza di base della sintassi SQL e dei concetti. Ciò include sapere come creare e manipolare tabelle, come utilizzare le istruzioni SELECT, INSERT, UPDATE e DELETE per interagire con i dati e come utilizzare le clausole DOVE per filtrare i dati. Una volta capito questi concetti, è possibile utilizzare SQL per estrarre, manipolare e gestire i dati in una varietà di situazioni.

2.2 Le banche dati relazionali sono fondamentali

I database relazionali sono una parte fondamentale della moderna gestione dei dati in organizzazioni di tutte le dimensioni. Vengono utilizzati per archiviare, gestire e analizzare i dati in una varietà di ambienti, dalle piccole imprese alle grandi aziende, alle agenzie governative e alle organizzazioni senza scopo di lucro.

Uno dei principali vantaggi di un database relazionale è che grandi quantità di dati vengono archiviati in modo strutturato e organizzato, il che consente di recuperare i dati (trovati) molto rapidamente. Ad esempio, puoi utilizzare un database relazionale per archiviare e gestire tutti i dati dei dipendenti della tua azienda.

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

I database relazionali si basano su un'idea semplice ma potente: i dati possono essere memorizzati in tabelle, che consistono di righe e colonne. Ogni **tabella** rappresenta un particolare tipo di dati e una **riga** nella tabella rappresenta un singolo elemento. Ad esempio, nella nostra tabella dei dipendenti, ogni riga contiene un individuo diverso e ogni **colonna** rappresenta un **attributo** diverso del dipendente (ad esempio nome, indirizzo e numero di telefono). Una riga è anche chiamata **record** e i singoli attributi sono memorizzati in **campi**. La riga di intestazione della tabella è anche chiamata **schema**, perché qui c'è il riferimento, a quali dati sono memorizzati in quale colonna.

Un altro vantaggio di un database relazionale è la capacità di garantire l'integrità e la coerenza dei dati. Ciò significa che è possibile impostare regole e restrizioni su come i dati vengono inseriti nel database e quindi assicurarsi che tali regole siano seguite. Ad esempio, puoi richiedere che un indirizzo sia memorizzato per tutti i dipendenti o un numero di telefono univoco. Ciò impedisce errori di dati e incongruenze che possono portare a problemi.

Esistono altri tipi di database. Ad esempio, i database NoSQL sono progettati per gestire dati non strutturati e semi-strutturati, come documenti, grafici e coppie chiave-valore. Sono spesso utilizzati per l'analisi dei big data e le applicazioni web in tempo reale.

Altri tipi sono database orientati agli oggetti, database grafici o database In-memoria. La scelta del tipo di database dipende dalle esigenze e dai requisiti specifici dell'applicazione utilizzata.

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

1.3 Istruzioni SQL

2.3.1 Struttura e sintassi delle istruzioni SQL

In questa scrittura, impareremo a conoscere le **parole chiave** comuni (noto anche come clausola) utilizzate nelle istruzioni SQL e la **struttura generale e la sintassi** delle istruzioni SQL.

Le parole chiave identificano il tipo di operazione da eseguire. Di seguito sono riportate le parole chiave più comuni utilizzate nelle istruzioni SQL:

SELEZIONA: Utilizzato per recuperare i dati da una o più tabelle

INSERISCI: Utilizzato per inserire nuovi dati in una tabella

AGGIORNAMENTO: Utilizzato per aggiornare i dati esistenti in una tabella

CANCELLA: Usato per eliminare i dati da una tabella

MODIFICA: Usato per modificare la struttura di una tabella

DROP: Usato per eliminare una tabella o un oggetto di database

CREA: Utilizzato per creare una nuova tabella o un oggetto di database

USO: Utilizzato per selezionare un database con cui lavorare

MOSTRA: Utilizzato per visualizzare informazioni su un oggetto di database

La **struttura e la sintassi** delle istruzioni SQL possono variare a seconda dello specifico sistema di gestione del database (DBMS) utilizzato. Tuttavia, ci sono alcune linee guida generali che si applicano alla maggior parte delle istruzioni SQL.

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

Un'istruzione SQL di base consiste tipicamente dei seguenti elementi:

Parola chiave: La parola chiave o clausola che identifica il tipo di operazione da eseguire

Argomenti: Uno o più argomenti o parametri che forniscono informazioni aggiuntive sull'operazione da eseguire

Punto e virgola: Tutte le istruzioni SQL terminano con un punto e virgola (;)

Ad esempio, un'istruzione SELECT di base assomiglierebbe a questa:

SELEZIONA colonna1, colonna2 DA table_name;

In questa affermazione, "SELEZIONA" è la parola chiave, "colonna1, colonna2" sono gli argomenti e "table_name" è la tabella da cui vengono recuperati i dati. Il punto e virgola alla fine dell'istruzione indica la fine dell'istruzione SQL.

1.3.2 SELEZIONARE informazioni

Un semplice esempio di un'affermazione **SELEZIONARE** rispetto al nostro database relazionale potrebbe assomigliare a questo:

SELEZIONARE * **DA PARTE** dei dipendenti;

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

"**SELEZIONARE**" è la parola chiave e
"*****" e "**DAI dipendenti**" sono gli argomenti

Questa dichiarazione SQL sta recuperando tutti i dati dalla nostra tabella chiamata "dipendenti". Il * rappresenta tutti e in questo caso tutte le colonne.

Nella maggior parte dei casi, tuttavia, non vogliamo tutti i dati memorizzati, ma solo una parte di essi. Ad esempio, il nome e il numero di telefono associato di tutti i dipendenti:

SELEZIONARE Nome, TelNr **DAI** dipendenti;

Ora diamo un'occhiata agli argomenti speciali **DA** e **DOVE:**

DA determina la tabella o le tabelle per questa query

DOVE viene utilizzato per aggiungere una condizione alla query
se la condizione è un testo, allegare il testo in singole virgolette

Ad esempio, se vogliamo solo il numero di telefono dei nostri dipendenti il cui cognome è Mouse, allora usiamo:

SELEZIONA Nome, TelNr **DAI** dipendenti **DOVE** Nome = 'Mouse';

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

Le informazioni SELEZIONATE sono molto potenti e supportano vari approfondimenti per gli scienziati dei dati. Per argomenti più particolari, vedere l'allegato di questa scrittura.

1.3.3 Altre informazioni SQL utili

Il DML — Data Manipulation Language — i comandi di SQL sono SELECT, INSERT, UPDATE e DELETE e sono essenziali per la gestione e la manipolazione dei dati all'interno di un database relazionale. Imparando questi 4 comandi e come utilizzarli in modo efficace, gli utenti possono diventare abili in SQL e utilizzarli per gestire una vasta gamma di attività di database.

SELEZIONA: Il comando SELECT viene utilizzato per recuperare i dati da una o più tabelle in un database. Consente agli utenti di specificare le colonne che desiderano recuperare e filtrare i dati in base a criteri specifici. Abbiamo lavorato con loro nell'ultimo capitolo.

INSERIRE: Il comando INSERT viene utilizzato per aggiungere nuovi dati a una tabella in un database. Consente agli utenti di specificare i valori da inserire nella tabella e le colonne in cui tali valori devono essere inseriti.

INSERT INTO table_name (colonna1, colonna2...)
VALORI (valore1, valore2... valoreX);

INSERT INTO è seguito dal nome della tabella in cui i dati devono essere inseriti, nell'esempio "table_name"
La parte successiva dell'istruzione specifica le colonne della

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

tabella in cui devono essere inseriti i dati, che è rappresentata da "(colonna1, colonna2...)". Successivamente, la parola chiave VALORI viene utilizzata per specificare i dati effettivi che verranno inseriti nel campo. I valori devono corrispondere all'ordine delle colonne specificate nella sezione precedente e sono rappresentati da "(value1, value2... valueX)".

Ad esempio, se inseriamo i dati nella nostra tabella chiamata "dipendenti" con colonne "Nome", "Adresse" e "TelNr", l'istruzione SQL per inserire un nuovo record potrebbe apparire come questa:

INSERT INTO dipendenti (Nome, Adresse, TelNr) VALUES ("John Smith", "High Street", 50573);

Si noti che non tutte le colonne di una tabella devono essere specificate nell'istruzione INSERT INTO. Se una colonna viene lasciata fuori, il database assegnerà un valore predefinito o inserirà un valore NULL in quella colonna, a seconda di come è stata creata la tabella.

Inoltre, l'ordine in cui i valori sono specificati deve corrispondere all'ordine delle colonne. Se l'ordine è diverso o se manca un valore, il database riporterà un errore.

AGGIORNAMENTO: Il comando UPDATE viene utilizzato per modificare i dati esistenti in una tabella in un database. Consente agli utenti di specificare i nuovi valori che dovrebbero sostituire i valori esistenti e di filtrare i dati in base a criteri specifici.

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

AGGIORNARE table_name
IMPOSTA colonna1 = valore1, colonna2 = valore2... colonnaN = valoreN
[**SE** condizione];

AGGIORNARE table_name specifica il nome della tabella da aggiornare.

IMPOSTA colonna1 = valore1, colonna2 = valore2... colonnaN = valueN imposta i valori di una o più colonne nella tabella a nuovi valori. Ogni colonna e il suo valore corrispondente sono separati da un segno uguale (=), e più colonne sono separate da virgole.

[SE condizione] è una clausola facoltativa che specifica le condizioni che devono essere soddisfatte affinché l'aggiornamento abbia luogo. Se non viene specificata alcuna condizione, tutte le righe della tabella verranno aggiornate.

Esempi di condizioni valide:

DOVE nome = "Mouse";
DOVE età > 50;
DOVE categoria = "elettronica" E stock_quantità > 0;
DOVE order_date lt; '2022-01-01';

ELIMINA: Il comando DELETE viene utilizzato per rimuovere i dati da una tabella in un database. Consente agli utenti di filtrare i dati in base a criteri specifici e di rimuovere tutti o un sottoinsieme dei dati che corrispondono a tali criteri.

ELIMINARE DA table_name **DOVE** {condizione};

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

ELIMINARE DA table_name specifica il nome della tabella da cui le righe devono essere cancellate.

DOVE {condizione} è una clausola facoltativa che specifica le condizioni che devono essere soddisfatte per eliminare le righe. Se non viene specificata alcuna condizione, tutte le righe della tabella verranno cancellate!

Esempi di condizioni valide:

DOVE NameID = 123456;

Questa affermazione cancellerebbe la riga con un NameID di 12345

DOVE NameID **IN** (12345, 23456, 34567);

Questa istruzione eliminerebbe tutte le righe in cui il "NomeID" è 12345 o 23456 o 34567.

È importante prestare attenzione quando si utilizza l'istruzione DELETE, in quanto rimuove in modo permanente i dati da una tabella. Si consiglia di effettuare un backup dei dati o di utilizzare una transazione per eseguire il rollback delle modifiche, se necessario.

In conclusione, SQL è un linguaggio di database potente e flessibile che è ampiamente utilizzato in molti settori e aree applicative diverse. La facilità d'uso, la flessibilità e l'efficienza lo rendono una scelta eccellente per le aziende e le organizzazioni che devono elaborare e mantenere grandi quantità di dati. Con i suoi numerosi vantaggi e risorse disponibili per l'apprendimento, SQL è uno strumento prezioso per chiunque lavori con database relazionali.

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

Autovalutazione (domande a scelta multipla e risposte)	1. Chi usa SQL? A) Gli esseri umani per lavorare con i dati memorizzati in un database relazionale B) Sviluppatori Web per codificare siti web C) Amministratori di database per la gestione dei dati 2. Per che cosa si usa SQL? A) Creare ed eliminare i dati in un database B) Per i pagamenti del negozio online C) Per recuperare i dati da un ambiente non-database 3. Quali parole chiave possono avviare un'istruzione SQL? A) DA B) DOVE C) SELEZIONA
Risorse (video, link di riferimento)	https://www.postgresql.org/ https://www.youtube.com/watch?v=HXV3zeQKqGY
Materiale correlato	
Contenuti disposti in 3 livelli	**2. Capire GitHub** **2.1 Scopo di GitHub** **2.1.1 Gestione del codice software di grandi dimensioni** Per tutti gli sviluppatori di software, la gestione del codice è una parte essenziale della programmazione. I sistemi di controllo delle versioni consentono agli sviluppatori di

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

controllare i loro cambiamenti di codice, collaborare con gli altri e gestire i loro progetti in modo efficiente. Un sistema di controllo delle versioni popolare è Git, e GitHub è un servizio di controllo delle versioni basato su cloud per progetti di sviluppo software costruiti su Git.

GitHub fornisce un'interfaccia facile da usare per la creazione e la gestione di repository Git, nonché strumenti per collaborare con altri sviluppatori su un progetto. GitHub può essere utilizzato sia per progetti personali che commerciali ed è gratuito nella sua edizione base.

GitHub può essere caratterizzato come una sorta di social network per gli sviluppatori di software. I membri possono seguirsi, valutare il lavoro dell'altro, ricevere aggiornamenti su progetti specifici e comunicare pubblicamente o privatamente. Consente agli sviluppatori di condividere il proprio lavoro, imparare dagli altri e costruire una comunità.

GitHub fornisce anche un set completo di strumenti di project management che rendono facile per gli sviluppatori gestire i loro progetti. Include strumenti per il monitoraggio dei problemi, l'organizzazione delle attività e la collaborazione con gli altri.

GitHub si integra con molti altri strumenti e servizi, tra cui l'integrazione continua e i servizi di distribuzione. Ciò consente agli sviluppatori di automatizzare il processo di test, costruzione e distribuzione dei loro cambiamenti di codice.

Dalla fine del 2018, GitHub Inc è di proprietà di Microsoft.

2.1.2 Che cos'è il controllo di versione?

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

Quando si lavora su progetti software complessi, tenere traccia delle modifiche e gestire più versioni di codice può essere un compito impegnativo. I sistemi di controllo delle versioni come Git forniscono una soluzione a questo problema monitorando e registrando le modifiche.

Git fornisce funzionalità per recuperare vecchie versioni di un progetto, confrontare, analizzare, unire le modifiche e molto altro ancora. Questo processo è chiamato **controllo di versione** e aiuta gli sviluppatori a tenere traccia della cronologia del loro codice. Git non è l'unico sistema di controllo della versione disponibile. Sono disponibili anche altri sistemi di controllo delle versioni come Perforce, Mercurial, CVS e SVN. Tuttavia, Git è diventato il sistema di controllo delle versioni più popolare e ampiamente utilizzato tra gli sviluppatori grazie alla sua flessibilità, velocità e facilità d'uso.

Una delle caratteristiche uniche di Git è che si tratta di un sistema di controllo della versione decentralizzato. A differenza di altri, Git non dipende da un server centrale per mantenere vecchie versioni dei file. Invece, funziona completamente localmente, memorizzando i dati come cartelle sul disco rigido dell'utente. Questo si chiama **repository**. Ciò consente agli sviluppatori di lavorare sul loro codice offline e rende facile tracciare le modifiche senza fare affidamento su un server centrale.

Se un utente vuole lavorare con altri sullo stesso codice, può fornire una copia del proprio repository online per l'accesso a tutto il team. Ciò consente a più sviluppatori di lavorare sulla stessa base di codice senza sovrascrivere le modifiche reciproche.

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

2.2 Modalità d'uso

2.2.1 Installalo sul tuo dispositivo

Per utilizzare Github in modo produttivo, hai bisogno di Git sul computer locale. Ci sono molti programmi Git che possono essere utilizzati gratuitamente o a pagamento:

Windows - "Git for Windows" fornisce un client GUI e un emulatore a riga di comando BASH.
https://git-scm.com/downloads/win o
https://gitforwindows.org/

Linux — basta aprire un nuovo terminale e installare Git tramite il gestore dei pacchetti della vostra distribuzione Linux. Per Ubuntu, il comando è ad esempio sudo apt-get install git

Mac OS — Il modo più semplice è installare homebrew e quindi eseguire brew install git dal terminale.
https://brew.sh/ o https://git-scm.com/downloads/mac

2.2.2 Creare un account e configurare GitHub

Prima di poterlo utilizzare, crea un account Github su www.github.com

Ci sono diverse configurazioni per l'aspetto e la funzionalità del client. Decidi in base alle tue preferenze di lavorare in modo efficiente.

Tuttavia, importante è la configurazione del **nome utente** e l' **indirizzo e-mail** corrispondente

Apri "Git Bash" ed esegui questi comandi sulla riga di comando:

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

Git config --global user.name "Mary"

Git config --utente globale.email "mary.smith@email.eu"

Tutte le attività in Git sono collegate al rispettivo nome utente e indirizzo e-mail specificati nella configurazione! Ciò rende le modifiche tracciabili perchè gli altri utenti sanno sempre chi ha apportato le modifiche specifiche e fornisce una panoramica nei progetti in cui molti sviluppatori stanno lavorando.

2.3 Come usare GitHub

2.3.1 Termini comuni in GitHub

Quando si lavora con Git, è essenziale capire la sua terminologia per essere in grado di utilizzarla in modo efficace.

Un **repository**, o repo in breve, è una cartella in cui vengono memorizzati tutti i file e le loro storie di versione. È la sede centrale per la gestione e l'organizzazione del codice. Un repository può essere ospitato su un server remoto, come GitHub o Bitbucket, oppure può essere memorizzato localmente sul computer.

Un **ramo** è uno spazio di lavoro in cui è possibile apportare modifiche isolate che non influenzeranno gli altri e ha una propria storia. È come una linea temporale separata in cui è possibile sperimentare cambiamenti senza influenzare la base di codice principale. Gli sviluppatori possono lavorare su diversi rami contemporaneamente, rendendo più facile collaborare ed evitare conflitti.

Un **commit** è un record salvato delle modifiche apportate a un file all'interno del repo. È lo stato del nostro repository in un momento. Un'istantanea alla quale il codice può essere ripristinato. I commit sono essenziali in Git perchè consentono di tenere traccia delle modifiche nel tempo e di tornare a una versione precedente del codice, se necessario.

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

Dopo che gli aggiornamenti sono stati apportati a un repository, altri sviluppatori possono scaricare tutte le modifiche con una **richiesta pull** o PR. Una richiesta pull è una richiesta per unire le modifiche da un ramo all'altro. Ciò consente agli sviluppatori di rivedere e approvare le modifiche prima di essere fuse nella base del codice principale.

Push è il processo di aggiunta di una modifica locale al repository remoto. Quando spingi i tuoi cambiamenti, diventano disponibili per gli altri per vedere e scaricare. Questo è un passo fondamentale nella collaborazione con altri sviluppatori.

Una volta approvata una richiesta pull, il commit verrà **unito** da una filiale all'altra. La fusione combina i cambiamenti da un ramo all'altro, in genere il ramo principale. Questo processo garantisce che tutte le modifiche siano incorporate nella base di codice principale.

Un **clone** è una copia completamente funzionale di un progetto. Copiare un repository da un server remoto è clonazione. Questo ti permette di avere una copia completa sulla macchina locale, rendendo più facile lavorare sul tuo codice senza fare affidamento su una connessione Internet.

2.3.2 I primi passi in Github

Quando si utilizza Git, prima deve essere creata una cartella mkdir learning-git (=**m**a**k**e **dir**ectory)

in cui il repository può essere creato per ogni programma o progetto

CD learning-git (= **C**hange **d**irectory)

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

Git init

A seguito di questo, è possibile collegare un locale e un repo online con
Git remote aggiunge origine
https://github.com/[username]/[projectname].git

Git lavora con il concetto di "area di sosta". All'inizio, l'area di sosta è vuota. I file possono aggiungere (o anche singole righe e parti di file) con il comando git add e infine commettere tutto (creare un'istantanea) con git commit:

Git aggiungere learning_script.txt

Git commit —m "firstsnapshot"

Per caricare il codice in un repo remoto, prima connettersi ad esso

Git remoto aggiungere origine https://github.com/learning.git

Quindi, i commit locali possono essere trasferiti al server, eseguiti ogni volta che si desidera aggiornare il repository remoto

Git push server_origin local_master

Una volta fatto questo, altri sviluppatori possono scaricare le modifiche dal repository remoto con un singolo comando

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

Git pull origin master

Per clonare un intero programma o progetto utilizzare

clone di Git https://github.com/tutorial.git

2.3.3 Comandi relativi al ramo

Quando si sviluppa una nuova funzionalità nello sviluppo del software, è meglio lavorare su una copia del progetto originale, chiamato ramo. Un ramo è una copia separata della base di codice che consente agli sviluppatori di apportare modifiche senza influire sulla versione live del codice.

Ogni ramo ha una propria storia e isola le modifiche apportate ad esso da altri rami fino a quando non si decide di fonderli. I vantaggi di questo approccio sono:

1. Una versione stabile (live) del codice non è influenzata da bug indesiderati: Quando si lavora su un ramo separato, eventuali bug o errori che si verificano durante il processo di sviluppo non influenzano la versione live del codice.

2. Un team di sviluppatori può lavorare su molte funzionalità contemporaneamente: Le filiali consentono a più sviluppatori di lavorare contemporaneamente su funzionalità diverse senza interferire con il lavoro dell'altro.

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

3. Ogni sviluppatore può lavorare sul proprio ramo senza il rischio di cambiare la propria base di codice dal lavoro di un altro sviluppatore: Lavorare su filiali separate assicura che gli sviluppatori possano lavorare in modo indipendente ed evitare conflitti tra le loro modifiche di codice.

4. Più versioni della stessa funzionalità possono essere sviluppate su diverse filiali e poi confrontate per scoprire la versione migliore: Le filiali consentono agli sviluppatori di sperimentare approcci diversi a una funzionalità e confrontare i risultati prima di fondere le modifiche nella base di codice principale.

In sintesi, l'utilizzo di filiali è una migliore pratica nello sviluppo del software in quanto aiuta ad evitare conflitti ed errori, consente il lavoro di squadra e consente la sperimentazione e il confronto di approcci diversi a una funzionalità.

Il ramo predefinito di ogni repository è chiamato master. Per creare più rami, utilizzare il comando
Git ramo & nome>

Passare al ramo appena creato utilizzando
Git checkout & nome>

Per unire due rami passare a uno e utilizzare
Git unire lt;name>

Per eliminare un ramo usare
ramo Git —d &name>

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

	In conclusione, molte aziende utilizzano GitHub. Quindi, se stai cercando un lavoro, farai bene se hai già familiarità con GitHub. GitHub è anche una piattaforma di apprendimento e collaborazione. Esploralo ed espandi la tua conoscenza e la tua comunità.
Autovalutazione (domande a scelta multipla e risposte)	1. Perché il controllo di versione è importante? A) Codice di sviluppo rapido B) Sviluppare un programma con un team in un processo tracciabile C) Necessario per la vendita del progetto 2. Chi può usare GitHub? A) Chiunue abbia un account Git B) Chi l'ha comprato C) Solo sviluppatore qualificato 3. Cos'è un ramo? A) Una società autorizzata B) Un progetto specifico C) Una copia del progetto originale
Risorse (video, link di riferimento)	https://www.github.com https://git-scm.com/downloads/win https://gitforwindows.org/

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.

	https://brew.sh/ https://git-scm.com/downloads/mac
Materiale correlato	
PPT correlato	
Bibliografia	
Sviluppato da da	WOMEN IN AI Austria

Co-funded by the Erasmus+ Programme of the European Union

Con il sostegno del programma Erasmus+ dell'Unione europea. Questo documento e il suo contenuto riflettono la
opinioni solo degli autori, e la Commissione non può essere ritenuta responsabile per qualsiasi uso che possa essere fatto delle informazioni ivi contenute.