Differenza tra back-end e front-end

- Il **front-end**, o "client-side", fa riferimento alla parte dell'applicazione che **l'utente finale vede e interagisce con**. Questa parte è solitamente costituita da HTML, CSS e JavaScript e viene eseguita sul browser dell'utente. Il front-end è responsabile dell'**interfaccia utente** e della **presentazione dei dati**.
- Il **back-end**, o "server-side", invece, si occupa delle attività che non sono visibili all'utente finale. Questa parte include le applicazioni, i database e i server che gestiscono le richieste del front-end, elaborano i dati e restituiscono risposte. Il back-end è spesso scritto in linguaggi di programmazione come Python, Ruby o PHP.

Back-end: API, JSON, PHP, DATABASE

- **API** (Application Programming Interface) è un **insieme di regole e protocolli che governano la comunicazione tra il back-end e il front-end**. Le API consentono al front-end di accedere ai dati e alle funzionalità fornite dal back-end.
  Ad esempio, se hai un'app che mostra le previsioni meteorologiche, l'API potrebbe essere utilizzata per recuperare i dati sulle previsioni dal provider di previsioni meteorologiche.
- **JSON** (JavaScript Object Notation) è un **formato di dati leggero e facilmente leggibile** che viene spesso utilizzato per la comunicazione tra il front-end e il back-end tramite API.
  JSON è **basato sulla sintassi JavaScript** e viene utilizzato per rappresentare i dati in un formato che possa essere facilmente analizzato e utilizzato da molte piattaforme e linguaggi di programmazione.
- **PHP** è un **linguaggio di programmazione server-side** molto utilizzato per lo sviluppo di applicazioni web. Può essere utilizzato per scrivere il codice del back-end.
- **Database** è un **sistema che memorizza e organizza i dati**. Il back-end di solito si connette a un database per accedere e gestire i dati utilizzati dall'applicazione. MySQL, PostgreSQL e MongoDB sono alcuni esempi di database comunemente utilizzati.

Utilizzo di “Postman” per testare le API (https://www.postman.com/)

- **Postman** è una piattaforma che **facilita il test delle API**.
  Con Postman, puoi **inviare richieste HTTP** (come GET, POST, PUT, DELETE, ecc.) a un'API, rappresentata da un URL, e **visualizzare la risposta dell'API** in un'interfaccia intuitiva. Puoi anche creare e gestire test automatizzati per verificare che l'API stia funzionando come previsto.

Front-end: GUI, FRAMEWORK

- La **GUI** (Graphical User Interface) si riferisce all'**interfaccia grafica** che un'applicazione offre all'utente. In altre parole, è ciò che **l'utente vede e interagisce con** quando utilizza un'applicazione. Il front-end è responsabile della creazione e della presentazione della GUI.
- Un **framework** è un insieme di librerie, tool e linee guida che forniscono una struttura per lo sviluppo di un'applicazione software. Un framework è **progettato per semplificare la creazione di un'applicazione**, fornendo una serie di componenti già implementati e una guida per l'utilizzo di questi componenti in un modo coerente.
  Alcuni aspetti di un framework includono:
    1. **Architettura:** Un framework fornisce una struttura ben definita per l'architettura di un'applicazione, definendo il modo in cui i componenti devono essere organizzati e interagire tra loro.
    2. **Funzionalità predefinite**: Un framework fornisce un insieme di funzionalità già implementate, come la gestione della persistenza dei dati, la gestione degli errori e la gestione delle richieste HTTP, che possono essere utilizzate direttamente senza doverle scrivere da zero.

3. **Convenzioni di codifica**: Un framework definisce un insieme di convenzioni di codifica che gli sviluppatori devono seguire per garantire la coerenza del codice e facilitare la manutenzione del codice a lungo termine.
4. **Modelli di sviluppo**: Un framework fornisce un modello di sviluppo che definisce il modo in cui le funzionalità devono essere implementate, migliorando la produttività e la qualità del codice.
5. **Documentazione**: Un framework fornisce una documentazione dettagliata che descrive come utilizzare il framework e come implementare le diverse funzionalità.

Fullstack (SvelteKIT + Vite), utilizzo di NPM

- SvelteKit (https://kit.svelte.dev/docs/introduction)

- SvelteKit è un **framework** per lo **sviluppo del front-end** basato sul linguaggio Svelte. SvelteKit è progettato per semplificare la creazione di app web complesse che funzionano sia sul lato client che sul lato server.
- SvelteKit fornisce una serie di strumenti e librerie per la creazione di interfacce utente e la gestione dello stato dell'app, il rendering del lato server e molto altro ancora.
- Uno dei punti di forza di SvelteKit è la sua **architettura di rendering lato server**. SvelteKit effettua il rendering dell'interfaccia utente sul lato server, inviando al client solo il codice HTML reso. Questo rende l'app più rapida e accessibile, soprattutto su connessioni lente o dispositivi meno potenti.
- SvelteKit fornisce anche una serie di strumenti per la gestione dello stato dell'app, l'accesso ai dati sul lato server. Con SvelteKit, è possibile creare app web che possono gestire molte richieste e dati in modo efficiente, offrendo un'esperienza utente fluida e reattiva.
- Inoltre, SvelteKit è molto facile da apprendere per chi ha già esperienza con Svelte o con altri framework per lo sviluppo del front-end. La sua sintassi e la sua filosofia di programmazione sono **familiari** e **semplici** da comprendere, il che rende più facile per i team creare app web di qualità.
- La **compilazione** di Svelte avviene in fase di build, quando l'app viene preparata per la distribuzione. Durante questa fase, Svelte esegue una serie di passaggi per convertire il codice sorgente in codice eseguibile dal browser.
- Le **route** in SvelteKit sono semplicemente un modo per associare un URL specifico a un componente Svelte. Questo significa che quando un utente accede a un URL, viene visualizzato il componente associato alla route corrispondente.
- I **components** in SvelteKIT sono elementi modulari che rappresentano una porzione di interfaccia utente o di funzionalità. In Svelte, i componenti sono scritti in file separati che contengono sia il codice HTML che quello JavaScript e CSS necessario per renderli funzionali. Questi componenti possono quindi essere riutilizzati in molte parti diverse del tuo sito o della tua applicazione, aiutandoti a mantenere il codice organizzato e facile da gestire. (Ad esempio, puoi creare un componente che rappresenti un pulsante e utilizzarlo ovunque sul tuo sito in cui hai bisogno di un pulsante. In questo modo, se devi apportare modifiche al pulsante, puoi farlo in un unico luogo e vedere gli effetti in tutte le parti del sito in cui il componente è stato utilizzato.)
- Libreria di components di esempio: https://carbon-components-svelte.onrender.com/

- Vite (https://vitejs.dev/)

- **Vite** è un **ambiente di sviluppo** (development environment) per la creazione di app web basate su Vue.js. Vite mira a fornire un'esperienza di sviluppo più veloce e reattiva rispetto ai tradizionali ambienti di sviluppo basati su build.
- Uno dei punti di forza di Vite è il suo **approccio al-volo alla compilazione**. Invece di creare una build completa prima di eseguire il codice, **Vite importa solo i file necessari quando sono richiesti**. Ciò significa che **le modifiche apportate al codice possono essere visualizzate istantaneamente nel browser**, senza la necessità di una build intermedia.
- Questo rende lo sviluppo più rapido e reattivo.Vite fornisce anche una **serie di strumenti per la gestione dei pacchetti e delle dipendenze**, il **debugging** e la **distribuzione**.

- NPM (https://www.npmjs.com/)

- NPM (Node Package Manager) è un gestore di pacchetti per il linguaggio di programmazione JavaScript. Serve come repository centrale per la pubblicazione e la distribuzione di pacchetti di codice JavaScript che possono essere utilizzati da sviluppatori per implementare funzionalità specifiche in progetti di sviluppo web.
- Gli sviluppatori possono facilmente installare questi pacchetti in un progetto utilizzando il comando **npm install** e gestire dipendenze e versioni di pacchetti utilizzando il file **package.json** del progetti.

GitHub (https://github.com/) (https://desktop.github.com/)

GitHub è una piattaforma web per il **controllo di versione e la collaborazione sul codice sorgente**. È stato creato nel 2008 ed è diventato uno dei principali luoghi di ritrovo per sviluppatori di tutto il mondo. Ecco alcune delle sue principali caratteristiche e funzionalità:

1. **Repository:** GitHub consente di creare repository per memorizzare e organizzare il codice sorgente. Ciascun repository può contenere file, immagini, documenti e altri tipi di contenuti.
2. **Controllo di versione:** GitHub utilizza Git, un sistema di controllo di versione open source, per tracciare le modifiche al codice e mantenere una cronologia delle versioni.
3. **Collaborazione:** GitHub consente ai team di sviluppo di lavorare insieme sul codice in modo efficiente. Gli sviluppatori possono sfruttare la funzionalità di fork per creare una copia di un repository e iniziare a lavorare su di esso, oppure possono richiedere pull request per integrare le proprie modifiche nel codice principale.
4. **Issues e bug tracking:** GitHub offre un sistema integrato per la gestione delle problematiche e del bug tracking. Gli sviluppatori possono utilizzare questa funzionalità per segnalare bug, richiedere funzionalità o discutere questioni relative al codice.
5. **Documentazione:** GitHub offre la possibilità di creare documentazione in linea e di utilizzare Wiki per creare guide e documentazione dettagliate.
6. **Integrazione con altri strumenti:** GitHub integra numerosi strumenti di sviluppo, tra cui editor di codice, servizi di continuous integration e deployment, e altri strumenti di gestione del progetto.

VS Code (https://code.visualstudio.com/)

- **Visual Studio Code** (spesso abbreviato in VS Code) è un **editor di codice sorgente gratuito** e open-source sviluppato da Microsoft. VS Code è progettato per essere leggero e veloce, ma al tempo stesso offre molte funzionalità avanzate per sviluppatori di tutti i livelli.
- Alcune delle caratteristiche di VS Code includono la possibilità di **eseguire il debug del codice, la funzionalità di completamento automatico del codice, la visualizzazione del codice in formato "side-by-side", la visualizzazione di diff e la funzionalità di esplorazione del codice**.
- Le **estensioni** sono plug-in che **estendono le funzionalità dell'editor di codice**. Possono essere installati direttamente dal Visual Studio Code Marketplace e possono essere utilizzati per personalizzare l'ambiente di sviluppo, migliorare la produttività e integrare ulteriori funzionalità.

Estensioni consigliate su VS Code per il progetto:

- HTML CSS Support
- HTML Snippets
- Javascript code snippets
- Live Server
- PHP Debug
- PHP Extension Pack
- PHP Intelephense
- PHP IntelliSense
- PHP Server
- Svelte for VS Code

Regole di Workflow:

1. Effettuare sempre il fetch Principale su GitHub desktop prima di iniziare a lavorare;
2. Se si sta lavorando sul back-end, una volta aperto VS Code, utilizzare l'estensione **PHP Server** e **Postman** per testare le API;
3. Se si sta lavorando sul front-end, una volta entrati nella directory, eseguire il comando ***npm install*** per scaricare eventuali nuove librerie; per far partire il server SvelteKIT eseguire il comando ***npm run dev -- --open***
4. Effettuare Fetch e Pull Principale su GitHub desktop per aggiornare la propria repository locale;
5. Una volta creato o modificato un file, aggiungere un titolo e una descrizione dettagliata al push;
6. Il push va effettuato sul branch a cui si è assegnati;
7. Dopo il push aggiornare i "compiti" sulla sezione projects (se si ha appena iniziato un compito spostarlo sulla colonna "In progress"; una volta finito, spostarlo sulla colonna "Done");

Suddivisione lavoro/branch:

1. Back-end (Reperti): Di Cataldo, Scala, Tomaiuolo
2. Back-end (Users): Gurgoglione, Mangano, Russo
3. Front-end (Reperti): Conti, D'Antuono, Pompilio
4. Front-end (Users): Di Giacomo, Doko, Puzzolante
5. Back-End/Front-End: Palladino

Convenzioni di codice:

- Commentare il più possibile e in maniera chiare il codice in modo che tutti possano comprenderlo e quindi, debbuggarlo;
- nomi_file.*
- nomi_variabili
- nomiFunzioni()