**RIASSUNTO DOMANDE NANNINI**

*Domande chiuse (risposte corrette da lui, sperem ben!)*

* Una web application si può testare attraverso il **browser** SOLO per la parte frontend.
* I **browser** mettono a disposizione strumenti per testare le prestazioni delle pagine
* Attraverso gli **strumenti diagnostici della rete** è possibile verificare principalmente lo scambio di messaggi tra client e server
* **TDD**: test-driven development cioè lo sviluppo del codice guidato dai test
* durante le **fasi di test** è necessario effettuare il collaudo su vari browser e/o sistemi operativi
* durante le **fasi di test,** usare il **trace log all’interno di un'applicazione** serve a verificare il funzionamento dell'applicazione run time(in esecuzione)
* nello **sviluppo software TDD** la parte di codice che viene scritta per prima è il codice dei test automatici
* lo **sviluppo TDD** è parte integrante della metodologia di programmazione Agile
* nello **sviluppo di un progetto software** bisogna definire i test automatici nella fase iniziale
* i **breakpoint** sono i **punti di interruzione nell’esecuzione dell'applicazione** utili a verificare (attraverso i valori assunti dalle variabili) il **corretto percorso** nell’esecuzione dell'applicazione stessa
* il **trace** e il **debug** di un’applicazione NON sono la stessa cosa (quali sarebbero le differenze? trace verifica applicazione run time mentre debug? secondo ChatGPT, il tracing è un'attività di monitoraggio generale che fornisce una visione ad alto livello del flusso di esecuzione, mentre il debugging è un processo specifico per individuare e correggere errori o problemi specifici nel codice)
* **il file di log** serve a tenere traccia del flusso di esecuzione scrivendo runtime il valore delle variabili da controllare
* i **DevTools** sono in grado di testare il corretto funzionamento del backend di una web app solo in parte rispettivamente alle direttive https
* grazie all’impostazione di un **trace log** si può monitorare l’applicazione in background una volta rilasciata (il background è una modalità di esecuzione riferita ad un [processo](https://it.wikipedia.org/wiki/Processo_(informatica)?wprov=wppw2) funzionante dietro le quinte senza l’intervento da parte dell’utente)
* le **metodologie di aggiornamento** impattano sulla definizione di un progetto software
* la **metodologia RAD** (rapid application development) prevede indicazioni sull’aggiornamento del prodotto software nella fase di rilascio
* il **load-balancing** è la distribuzione del carico di lavoro insistente su determinate parti del sistema
* il **tempo di latenza di una pagina web** misura il tempo che intercorre tra la richiesta del server e la risposta
* l’utilizzo di **node js** migliora le prestazioni in caso di molti accessi contemporanei rispetto ad un normale web server
* una **funzione di callback** è una funzionalità associata alla gestione di un evento generato
* la differenza tra un **IP pubblico e un IP privato** è il dominio di univocità
* in **POST** i parametri fanno parte del body, in **GET** i parametri sono parte della stringa URL
* una **web application può lavorare in modalità disconnessa** solo se è previsto in fase di sviluppo
* la **scheda di rete** permette la connessione di un dispositivo di elaborazione ad una rete
* la **pagina di avvio di una web application** può avere un nome qualunque ed è impostabile tra le configurazioni del web server
* se una **homepage non è raggiungibile attraverso il nome di dominio, ma solo attraverso l’IP** dopo aver registrato il dominio, il problema potrebbe essere che i server DNS non sono aggiornati
* il codice di errore http **404** indica una risorsa non trovata

*Domande aperte (la tipologia domande aperte non ci sarà all'esame, ma eventualmente questi sono gli argomenti trattati nel caso li metta in delle domande a risposta chiusa).*

*Ho usato chat gpt per avere delle risposte tendenzialmente complete.*

* **procedura per installare node JS**
* Scarica e installa Node.js dal sito ufficiale.
* Apri il terminale o la riga di comando e verifica l'installazione eseguendo node -v e npm -v.
* Crea una nuova directory per il progetto React.js.
* Accedi alla directory del progetto tramite il terminale.
* Inizializza il progetto Node.js eseguendo npm init e rispondi alle domande per generare il file package.json.
* Installa React.js eseguendo npm install react react-dom.
* Installa Create React App eseguendo npm install -g create-react-app.
* Crea un'app React eseguendo npx create-react-app nome-app.
* Accedi alla directory dell'app eseguendo cd nome-app.
* Avvia l'app React eseguendo npm start.
* **come valutare le performance di un'applicazione?**
* Misurare i tempi di caricamento: Utilizza strumenti come Lighthouse, PageSpeed Insights o WebPageTest per analizzare i tempi di caricamento della pagina, inclusi i tempi di risposta del server, il rendering del contenuto e il completamento delle risorse.
* Analizzare il tempo di risposta del server: Puoi utilizzare strumenti come curl, Postman o l'analisi delle richieste di rete nei DevTools di Chrome per misurare il tempo di risposta delle richieste HTTP effettuate dall'applicazione. Verifica se ci sono richieste lente o ritardi nelle risposte del server.
* Profilazione del codice: Utilizza gli strumenti di profilazione nei DevTools di Chrome o altri profiler per analizzare il tempo di esecuzione delle diverse parti del codice, identificando eventuali funzioni o sezioni che richiedono più tempo. Questo può aiutarti a individuare aree di ottimizzazione e miglioramento delle prestazioni.
* Monitoraggio delle risorse: Utilizza gli strumenti di monitoraggio delle risorse per identificare risorse (come immagini, script, fogli di stile) che possono essere eccessivamente grandi o che richiedono molto tempo per il download. Ottimizza queste risorse riducendo le dimensioni dei file o utilizzando tecniche di caching.
* Testing su dispositivi reali: Valuta le performance dell'applicazione su dispositivi reali, come smartphone, tablet o computer con diverse specifiche hardware. Ciò ti aiuterà a identificare eventuali differenze nelle prestazioni in base al dispositivo utilizzato e ottimizzare l'applicazione per offrire una buona esperienza utente su tutte le piattaforme.
* A/B testing: Confronta le performance di diverse versioni dell'applicazione utilizzando tecniche di A/B testing. Questo ti permetterà di valutare le modifiche apportate e confrontarle con una versione di controllo per determinare l'impatto sulle prestazioni.
* **Qual è un piano di test da realizzare su una web app?**
* Test funzionali: Verifica che tutte le funzionalità dell'applicazione funzionino correttamente. Testa le azioni degli utenti, le operazioni di input/output e la corrispondenza ai requisiti funzionali.
* Test di compatibilità del browser: Verifica che l'applicazione funzioni correttamente su diversi browser e versioni. Testa la compatibilità su browser comuni come Chrome, Firefox, Safari, Edge, ecc.
* Test di risposta al carico: Verifica come l'applicazione si comporta sotto carichi di lavoro elevati. Testa la scalabilità e la gestione delle richieste simultanee, monitorando i tempi di risposta e le prestazioni generali.
* Test di sicurezza: Verifica la sicurezza dell'applicazione identificando e mitigando possibili vulnerabilità. Testa l'iniezione di codice, la gestione delle sessioni, l'autenticazione, ecc.
* Test di usabilità: Verifica l'usabilità dell'applicazione attraverso test con utenti reali. Raccoglie feedback sulla navigazione, il flusso dell'interfaccia utente e la facilità d'uso complessiva.
* Test di integrazione: Verifica che i componenti dell'applicazione si integrino correttamente. Testa l'interazione tra le diverse parti dell'applicazione, come il back-end, il front-end, i database, i servizi esterni, ecc.
* Test di prestazioni: Verifica le prestazioni dell'applicazione, come i tempi di caricamento delle pagine, i tempi di risposta delle richieste di rete, l'efficienza dell'algoritmo, ecc. Utilizza strumenti di profiling e misurazione per identificare e risolvere eventuali problemi di prestazioni.
* Test di recupero di emergenza: Verifica la capacità dell'applicazione di gestire situazioni di emergenza, come guasti del server, errori del database, interruzioni di rete, ecc. Testa il ripristino del sistema e la continuità del servizio.
* Test di localizzazione: Verifica che l'applicazione funzioni correttamente in diverse lingue e regioni. Testa l'adattabilità alle lingue, i formati delle date/orari e le convenzioni culturali.
* Test di manutenzione e aggiornamento: Verifica la stabilità dell'applicazione durante le attività di manutenzione e gli aggiornamenti del software. Testa l'installazione di nuove versioni, la migrazione dei dati, la compatibilità con le versioni precedenti, ecc.
* **quali funzionalità utilizzare tra quelle sui devtools di chrome per verificare incompatibilità tra il software realizzato e l’infrastruttura di supporto (sistemi operativi, librerie, ecc)?**
* User Agent Switcher: Puoi utilizzare questa funzionalità per simulare diverse stringhe User Agent che corrispondono a diversi browser, sistemi operativi o dispositivi. Ciò ti consente di testare come l'applicazione si comporta su diverse configurazioni di infrastruttura.
* Device Toolbar: La Device Toolbar dei DevTools consente di simulare diversi dispositivi mobili e risoluzioni dello schermo. Puoi testare l'aspetto e il comportamento dell'applicazione su dispositivi specifici e verificare eventuali problemi di layout o di reattività.
* Emulazione di rete: La funzionalità di emulazione di rete dei DevTools consente di simulare diverse condizioni di rete, come la latenza, la velocità di connessione e le limitazioni della larghezza di banda. Puoi testare come l'applicazione si comporta in diverse condizioni di rete e verificare eventuali problemi di prestazioni o di caricamento dei dati.
* Console: Controlla la console dei DevTools per individuare eventuali errori o avvertimenti legati all'infrastruttura di supporto. Ad esempio, potrebbero essere visualizzati avvisi relativi alla mancanza di librerie o alla compatibilità con il sistema operativo.
* Network: Utilizza la scheda Network per monitorare le richieste di rete e le risposte dell'applicazione. Puoi verificare se ci sono errori di connessione o problemi di caricamento delle risorse che potrebbero essere causati da incompatibilità nell'infrastruttura di supporto.
* Audits: Esegui un'auditing delle prestazioni e della compatibilità dell'applicazione utilizzando la scheda Audits dei DevTools. Puoi ottenere suggerimenti e informazioni su eventuali incompatibilità con l'infrastruttura di supporto e su come risolverle.
* **come si tracciano e correggono le anomalie riscontrate durante la scrittura del codice?**
* IDE (Integrated Development Environment): Gli ambienti di sviluppo integrati, come Visual Studio Code, JetBrains WebStorm o Eclipse, forniscono funzionalità di debugging, evidenziazione degli errori, suggerimenti di codice e refactoring che facilitano l'individuazione e la correzione delle anomalie nel codice.
* Strumenti di linting: I linter, come ESLint o TSLint, analizzano il codice sorgente per individuare violazioni delle regole di stile, errori di sintassi o problemi potenziali. Forniscono segnalazioni durante la scrittura del codice, aiutando a mantenere una buona qualità e rilevare anomalie.
* Strumenti di debugging del browser: I DevTools di Chrome, Firefox Developer Tools o Safari Web Inspector offrono strumenti di debugging per analizzare il codice JavaScript, ispezionare il DOM, monitorare le richieste di rete e verificare l'esecuzione del codice. Questi strumenti consentono di individuare e correggere anomalie durante l'esecuzione dell'applicazione web.
* Strumenti di testing automatizzato: I framework di testing come Jest, Mocha, Jasmine o Selenium consentono di creare e automatizzare test per verificare il corretto funzionamento dell'applicazione. Questi strumenti aiutano ad individuare e correggere anomalie durante lo sviluppo e a garantire una maggiore stabilità del codice.
* Sistemi di monitoraggio degli errori: Strumenti come Sentry, Bugsnag o Rollbar permettono di tracciare gli errori che si verificano in produzione, fornendo informazioni dettagliate sulle anomalie riscontrate. Questi strumenti consentono di individuare e correggere le anomalie dopo il rilascio dell'applicazione.
* Sistemi di controllo delle versioni: L'utilizzo di un sistema di controllo delle versioni come Git permette di tenere traccia delle modifiche apportate al codice, facilitando la collaborazione tra i membri del team e consentendo di tornare a versioni precedenti se necessario.
* Code review: Le revisioni del codice da parte dei colleghi o dei membri del team aiutano a individuare e correggere anomalie, errori o problemi di logica nel codice. L'uso di piattaforme di code review come GitHub o Bitbucket semplifica il processo di revisione e collaborazione.