https://github.com/hariton-andrei-marius/esercizi

Configurazione computer

Installazione Git

Andare al indirizzo: https://git-scm.com/downloads

Git è un software per il controllo di versione distribuito, utilizzabile da interfaccia a riga di comando oppure attraverso i vari client utilizzabili attraverso la Graphical User Interfare (ad esempio GitExtensions).

Comandi principali sono:

- git clone url-repository >>> per clonare un repository sul proprio PC

- git add . >>> per mettere in stage tutti i file modificati

- git add nome_file >>> per mettere in stage un file specifico

- git commit -m “commit-message” >>> per fare il commit dei file messi in stage

- git push >>> per mandare sul server remoto i commit (upload)

- git pull >>> per fare il download delle modifiche dal server remoto (download ultimi commit)

- git checkout nome-branch >>> per spostarsi sul branch

- git init >>> per creare in locale un repository non ancora collegato ad un reposity remoto

- git status >>> per vedere lo stato delle modifiche non ancora committate sul branch del repository

- git branch >>> lista dei rami

- git stash >>> permette di accantonare le modifiche senza buttarle via definitivamente

- git stash pop >>> permette di recuperare le modifiche accantonate in precedenza

Un repository Git è strutturato in 1 o più branch che contengono file diversi o differenze nei file tra un branch e un’altro. Se vogliamo allineare le modifiche di 2 branch ci si sposta con CHECKOUT su uno dei 2 e poi si fa il MERGE del secondo sul primo: se sono presenti modifiche allo stesso file sui 2 branch, è possibile che si verifichino MERGE-CONFLICTS, ovvero conflitti che Git non è riuscito a risolvere in automatico: in questo caso bisognerà andare nel proprio IDE o EDITOR a scegliere quale modifica mantenere e quale cancellare (è possibile mantenere anche parzialmente le 2 modifiche o riscrivere tutta quella parte di codice).

Di solito si lavora con 2 branch principali, ed altri branch temporanei che verranno mantenuti o cancellati una volta fatto il MERGE sui rami/branch principali; quelli principali sono:

- master oppure main: questo deve essere il branch principale / di default del repository, tutte le modifiche prima o poi dovranno essere riportate su questo branch, dal quale si procederà con il deploy (rilascio) del applicazione sul ambiente di DEV (sviluppo) poi TEST poi PROD (produzione).

- develop oppure development: questo è secondario rispetto a master, tutte le modifiche dovrebbero essere “MERGIATE” (merge delle modifiche) dal branch/ramo del specifico TASK/COMPITO oppure dal branch del specifico REFACTOR (una volta testate), al branch di DEVELOP, poi una volta testate anche su DEVELOP, possono essere portare su MASTER, una volta lì, possono essere rilasciate.

Installazione Node.js

Andare al indirizzo: https://nodejs.org/it/download/

Node.js è un runtime system open source multipiattaforma orientato agli eventi per l’esecuzione di codice JavaScript. Orientato agli eventi ovvero il flusso del programma è determinato da eventi esterni.

Una volta installato, Node.js contiene NPM (Node Package Manager) che viene utilizzato per installare dipendenze/applicazioni globalmente sulla macchina oppure localmente nella cartella del repository. Nel secondo caso bisogna avere un file package.json dal quale leggere le dipendenze locali da installare nella cartella node_modules. Questa cartella deve essere presente nel file .gitignore in modo da non venir salvata sul repository; invece il file package.json deve essere salvato.

Comandi principali sono:

- npm install >>> installa le dipendenze presenti nel file package.json dentro la cartella node_modules

- npm install nomedipendenza --save >>> installa la dipendenza e ne salva il nome nel package.json

- npm list -g --depth=0 >>> mostra i pacchetti installati globalmente limitando la visualizzazione al primo livello di nodi

- npm update -g nomepacchetto >>> aggiorna un pacchetto globale

- npm run start >>> lancia lo script di start presente nel package.json

- npm run build >>> lancia lo script di build presente nel package.json

Installazione Visual Studio Code

Andare al indirizzo: https://code.visualstudio.com/download

Microsoft Visual Studio Code è un editor gratuito con integrazione Git. Una volta installato sarà molto utile installare delle estensioni per Angular e Angular Snippet in modo da avere un integrazione migliore con il progetto Angular e delle facilitazioni per la creazione di componenti Angular.

Installazione Git Extensions

Andare al indirizzo: https://github.com/gitextensions/gitextensions/releases/

Git Extensions è un’applicazione per la gestione di repositories Git. Permette di gestire molteplici repository in modo più semplice rispetto all’interfaccia testuale. E’ anche un facilitatore per le operazioni più complesse. Permette di fare merge, rebase dei branch. Permette di fare revert o cherry pick dei commit.

- Cherry Pick commit: ovvero prendere uno specifico commit da un branch su un altro branch senza riportarsi dietro tutta la cronologia di commit.

- Revert commit: ovvero cancella le modifiche di quel commit, le ripristina.

- Merge branch: ovvero unisce la cronologia di un branch con quella di un'altro, riportandosi tutte le modifiche e unendole alle altre.

- Rebase branch: invece di fare un merge che mantiene tutta la cronologia e crea un commit di merge, il rebase crea una cronologia più pulita senza ulteriori commit, praticamente mette in ordine i vari commit ma sarà necessario un force push perché riscrive la cronologia dei commit.

Angular

Angular2+ o Angular è un framework open source per lo sviluppo di applicazioni web. Da non confondere con il suo predecessore AngularJS o Angular1. Esso è sviluppato da Google principalmente, dal 2016.

https://angular.io/guide/setup-local

Prima installare globalmente Angular Cli:

```
npm install -g @angular/cli
```

Poi impostare la policy di esecuzione dei script:

```
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
```

Poi creare il progetto Angular:

```
ng new my-platform
```

- Vuoi aggiungere Angular routing? Y

- Quale formato per fogli di stile? SCSS

Ora si può eseguire l'applicazione generata:

```
cd my-platform
ng serve --open
```

Prima cosa, aggiungiamo Angular Material, se vogliamo utilizzarlo:

```
ng add @angular/material
```

Comandi utili:

```
ng generate component xyz >>> generazione automatica di un nuovo componente
ng generate service xyz >>> generazione automatica di un nuovo servizio
ng test >>> lancio dei test nel progetto
ng build >>> esecuzione rilascio del progetto
```

Link utili:

https://angular.io/tutorial

https://material.angular.io/

https://angular.io/cli

Esercizio 1

Creare una piattaforma di applicazioni senza un backend dietro, soltanto frontend. Bisognerà dividerla in più rotte:

- /dashboard >>> che sarà la homepage, bisognerà mettere una wildcard ** per reindirizzare verso questa pagina.

- /app/:name >>> sarà la rotta per visualizzare l'applicazione, raggiungibile dalla lista in homepage.

- /about >>> pagina contenente del semplice HTML e SCSS per dettagli sulla piattaforma e dettagli sui autori.

Per quanto riguarda le applicazioni, si potrebbe cominciare con alcune semplici app:

- /app/maps >>> utilizzerà openlayers https://www.npmjs.com/package/ol per visualizzare openstreetmaps con alcuni comandi di base.

- /app/notes >>> permetterà di scrivere delle note in una textarea di una form con la possibilità di salvare su file txt tramite https://www.npmjs.com/package/file-saver

La configurazione delle applicazioni dovrà essere dinamica e caricata da un file di configurazione TS contenente l'array di applicazioni con i relativi dati.

Creare 2 componenti principali nel layout, la navigation bar ed il body della pagina.

Creare un service che fornisce la configurazione delle applicazioni.

Creare una struttura delle cartelle per possibili espansioni (utilizzare file .gitkeep):

- app
    - configuration: files di configurazione.
    - modules: moduli angular, come moduli per il routing.
    - shared
        - components: classici componenti formati da classe TS, html, SCSS.
        - decorators: esistono dec. standard come @Component, @Pipe, @Input.
        - directives: per direttive custom, modificano la struttura DOM come ngIf, ngFor.
        - interfaces: strutture di dati da usare per definire variabili.
        - models: modelli di dati da usare prima di mandare i dati al backend.
        - services: servizi che forniscono dati, usati più che altro per interagire con backend.
        - interceptors: intercettano le chiamate API al backend ed applicano modifiche.

- utils: altri componenti standard.
- guards: guardie delle rotte come CanActivate (eager) oppure CanLoad (lazy load).
- pipes: modificatori di dati nel HTML, esistono pipes di base e custom.

Creare 2 componenti body e navbar (TS, HTML, SCSS) ed aggiungerli al modulo di base.

In app component aggiungere in HTML i 2 componenti appena creati, in navbar creare una lista menu, in body aggiungere router outlet.

Applicare un reset al SCSS: https://meyerweb.com/eric/tools/css/reset/

Applicare un po' di stile alla navbar con la lista UL.

Creare componenti dashboard, app-page, about e aggiungerli al modulo delle rotte ed al modulo base.

Applicare un po' di stile e creare un wrapper in body component.

Creare i componenti wrapper delle 2 applicazioni maps e notes in utils.

Creare file di configurazione e creare servizio di cui fare provide in modulo base.

Nel servizio fare return of(APPLICATIONS). Notare che è Observable.

In dashboard creare i 2 modi possibili per ricavare i dati.

Lavorare su App Page per la gestione della singola applicazione.