



Il Progetto IncluDO

di Carla Manavella

CODEPEN: [HTTPS://CODEPEN.IO/CARLAISABELLE/PEN/WBRBWZZ](https://codepen.io/carlaisabelle/pen/wrbbwzz)
GITHUB: [HTTPS://GITHUB.COM/CARLAISABELLE/INCLUDE_APP](https://github.com/carlaisabelle/includo_app)

INDICE

P. 01

Il brief

P. 05

La
preparazione

P. 07

L'interfaccia
IPartecipante

P. 09

L'interfaccia
ICorso

P. 11

L'interfaccia
IAzienda

P. 14

La classe
Partecipante

P. 16

La classe
Corso

P. 19

La classe
Azienda

P. 22

app.ts

Appendice 1

I partecipanti

Appendice 2

I corsi

Appendice 3

Le aziende

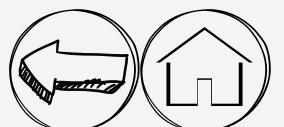


IL BRIEF

“IncluDO” è un’azienda ipotetica (ma verosimile) che si occupa di offrire percorsi di formazione professionale a migranti e persone svantaggiate contribuendo, al contempo, alla salvaguardia dei mestieri artigianali tradizionali.

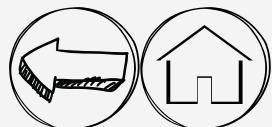
L’idea è di una coppia, marito e moglie, che col tempo ha visto svuotarsi il proprio Paese e la conseguente chiusura di molte botteghe storiche.

Il loro obiettivo è quello di aiutare persone svantaggiate a inserirsi nel mercato del lavoro e, al contempo, preservare i mestieri artigianali tradizionali che stanno scomparendo.



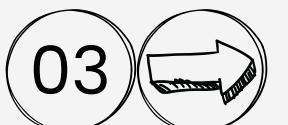
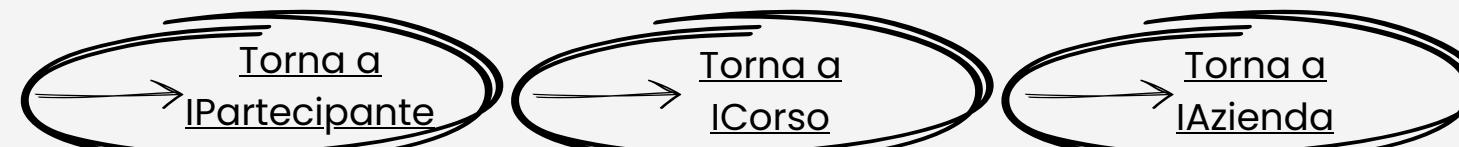


Funziona così: “IncluDO” si occupa di organizzare percorsi formativi gratuiti, sviluppati in collaborazione con professionisti e artigiani locali che si occupano di insegnare il proprio mestiere ai partecipanti. I percorsi formativi, che sono gratuiti, vengono finanziati sia con fondi Europei e regionali, sia grazie alle sponsorizzazioni locali; inoltre, i partecipanti stessi contribuiscono al finanziamento della loro formazione organizzando workshop per turisti (dove possono anche mettere in pratica quanto appreso in classe).



Il mio obiettivo è quello di creare una applicazione per IncluDO. Le richieste:

- Definire un'**interfaccia IPartecipante** (che rappresenta i migranti che partecipano ai programmi formativi) con
 - proprietà per “nome”, “cognome”, “Paese di origine”, “livello di istruzione”, “competenze linguistiche” e “ambito di formazione di interesse”,
 - un metodo “iscrivitiCorso(corso: ICors): void” per iscrivere il partecipante al corso di formazione.
- Definire un'**interfaccia ICORSO** (che rappresenta i corsi di formazione offerti da IncluDO) con
 - proprietà per “titolo del corso”, “descrizione”, “durata” e un array per l’elenco degli iscritti (“IPartecipante[]”),
 - un metodo “aggiungiPartecipante(partecipante: IPartecipante): void” per aggiungere un partecipante all’elenco degli iscritti.
- Definire un'**interfaccia IAzienda** (che rappresenta le aziende partner che offrono opportunità di tirocinio o impiego ai partecipanti formati) con
 - proprietà per “nome azienda”, “settore di attività”, “descrizione”, “posizioni aperte (string[])”;
 - un metodo “offriPosizione(partecipante: IPartecipante, posizione: string): void” per offrire una posizione lavorativa a un partecipante specifico.

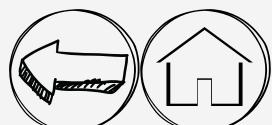


Le interfacce servono, poi, per implementare le relative classi:

- la **Classe Partecipante**, che gestisce le informazioni di ciascun partecipante e l'iscrizione ai corsi di formazione;
- la **Classe Corso**, che gestisce le informazioni sui corsi di formazione (inclusi i dettagli e gli iscritti);
- la **Classe Azienda**, che rappresenta un'azienda partner e ne gestisce non solo le informazioni, ma anche le offerte di lavoro disponibili.

Le logiche di collegamento:

- nella classe Partecipante, il metodo “*iscriviCorso()*” dovrà consentire al migrante di iscriversi a uno dei corsi di formazione disponibili;
- nella classe Corso, il metodo “*aggiungiPartecipante()*” dovrà incrementare l’elenco degli iscritti con i migranti interessati;
- nella classe Azienda, il metodo “*offriPosizione()*” dovrà permettere alle aziende di offrire un’opportunità lavorativa direttamente ai migranti formati.

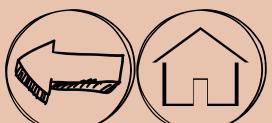


LA PREPARAZIONE

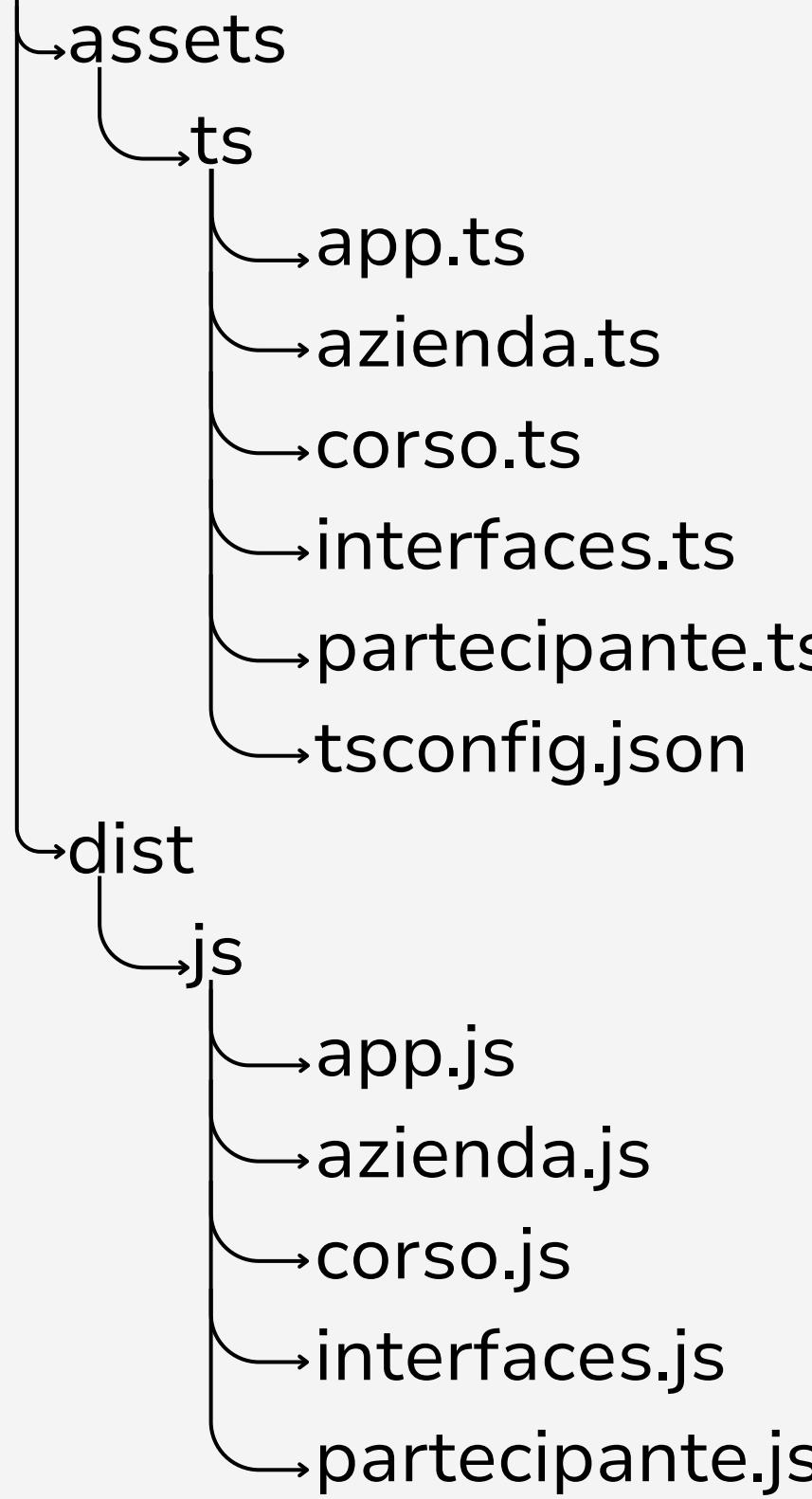
Prima di iniziare, avevo bisogno di un elenco con i partecipanti, i corsi e le aziende su cui lavorare. Ho quindi sottoposto il brief a un'intelligenza artificiale (Claude) e le ho chiesto di:

- generare un elenco di 10 partecipanti, con la loro storia;
- generare un elenco con 8 corsi che un'entità come IncluDO (che ho ipotizzato avesse sede in Puglia) potesse verosimilmente offrire, a cui ho aggiunto un corso di italiano per stranieri;
- generare un elenco con 7 aziende ipotetiche ma verosimili (anch'esse con sede in Puglia) che avrebbero potuto collaborare con un'entità come IncluDO.

A un'altra intelligenza artificiale (Gemini) ho chiesto, invece, la generazione delle immagini che ho usato in questa presentazione.

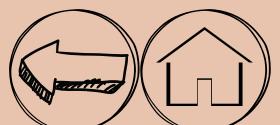


IncluDO_v1



Ora che avevo un'idea più chiara del materiale con cui avrei lavorato, ho suddiviso il progetto in file dedicati per rispettare il principio di separazione delle responsabilità e rendere il codice più facilmente gestibile, testabile e scalabile.

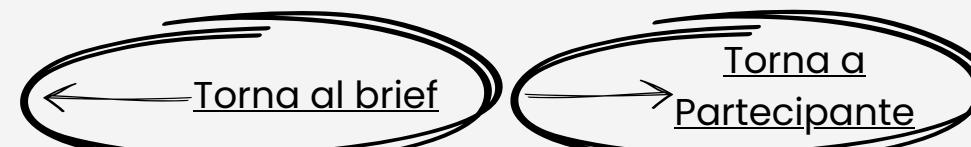
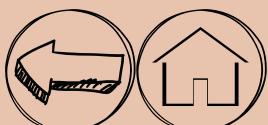
- In “interfaces.ts” si trova la definizione delle interfacce (“IPartecipante”, “ICorso” e “IAzienda”), che verranno esportate dove serve e usate come “contratto” per la costruzione delle classi;
- In “partecipante.ts” si trova la classe “Partecipante” (che implementa IPartecipante) e la definizione del suo metodo per iscrivere un partecipante a un corso;
- In “corso.ts” si trova la classe “Corso” (che implementa ICORSO) e la definizione dei metodi che gestiscono gli iscritti ai corsi;
- In “azienda.ts” si trova la classe “Azienda” (che implementa IAzienda) e la definizione del suo metodo per offrire una posizione lavorativa;
- Il file “app.ts”, invece, è il punto d’ingresso dell’intera app e ne contiene l’istanziazione, la configurazione e la logica di test;
- Tutti i file JavaScript transpilati a partire dai file TypeScript sono contenuti nella cartella “dist > js”.



L'INTERFACCIA IPARTECIPANTE

L'interfaccia si trova nel file *interfaces.ts* e rappresenta i partecipanti al progetto IncluDO. È composta da:

- “*nome: string*” → una stringa per il nome del partecipante;
- “*cognome: string*” → una stringa per il cognome del partecipante;
- “*paeseOrigine: string*” → una stringa per il Paese di origine del partecipante;
- “*livelloIstruzione: string*” → una stringa per il livello di istruzione del partecipante;
- “*competenzeLingue: string []*” → un array di stringhe per tutte le lingue parlate dal partecipante;
- “*ambitoInteresse: string[]*” → un array di stringhe che racchiude gli interessi formativi del partecipante;
- “*iscrivitiCorso(corso: ICORSO): void*” → il metodo che permette al partecipante di iscriversi a uno dei corsi proposti da IncluDO.



Oltre a quelle richieste dal brief, ho aggiunto due proprietà che ho ritenuto utili:

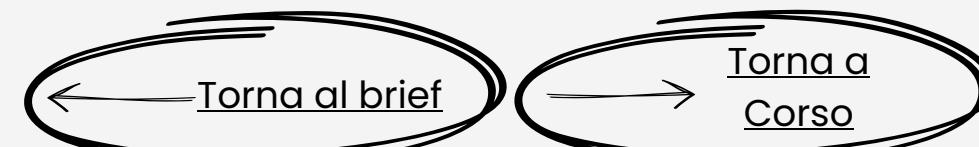
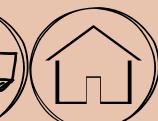
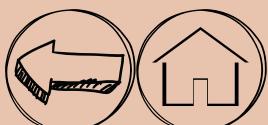
- “*storia: string*” → una stringa in cui inserire una breve descrizione della storia del partecipante, in modo da averla sempre a disposizione;
- “*contatto: string*” → una stringa in cui inserire l’email del partecipante a cui, più avanti, un’azienda lo/a potrà contattare.



L'INTERFACCIA ICORSO

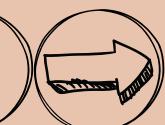
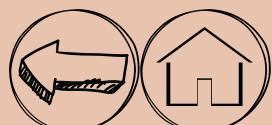
Anche questa interfaccia si trova nel file *interfaces.ts* e rappresenta i corsi offerti da IncluDO. È composta da:

- “*titoloCorso: string*” → una stringa per il titolo del corso;
- “*descrizioneCorso: string*” → una stringa con una breve descrizione del corso;
- “*settoreProfessionale: string*” → una stringa per descrivere il settore professionale a cui fa riferimento il corso;
- “*durataMesi: number*” → un “*number*” che descrive la durata (in mesi) del corso;
- “*elencolscritti: IPartecipante []*” → un array che contiene degli elementi tipizzati come “*IPartecipante*” (come richiesto dal brief; in questo modo, qui è come se si conservasse la “scheda completa” del partecipante che si iscrive);
- “*aggiungiPartecipante (partecipante: IPartecipante): void*” → un metodo che aggiunga un partecipante all’elenco degli iscritti al corso (“*elencolscritti: IPartecipante []*”).



Oltre alle proprietà e ai metodi previsti dal brief, ne ho aggiunti alcuni che possono tornare utili:

- “*idCorso: string*” → una stringa “*id*” che identifica ogni corso in maniera univoca (da usare, ad esempio, in un menù a tendina nel caso di un form nel frontend);
- “*durataOre?: number*” → un numero che indica le ore totali necessarie per completare il corso (la durata in ore è un’indicazione molto diffusa nei corsi di formazione proposti da agenzie formative come IncluDO);
- “*oreTeoria?: number*” → un numero che indica quante ore sono quelle di teoria;
- “*orePratica?: number*” → un numero che indica quante sono, invece, le ore di pratica;
 - ↳ Queste tre proprietà sono opzionali, perché modellano degli scenari reali che ho immaginato e che, se si verificassero, potrebbero causare dei problemi con l’implementazione di questa interfaccia (quello, ad esempio, di un corso in cui le ore di teoria e quelle di pratica sono una cosa sola, come un corso di lingua, oppure quello di un professionista che propone un corso a IncluDO, che sa quantificarne la durata in mesi ma che non è in grado di stimarne le ore necessarie né tantomeno una loro suddivisione tra teoria e pratica). In questo modo, si possono inserire i dati in modo meno rigido garantendo, però che l’interfaccia possa sempre essere implementata correttamente.



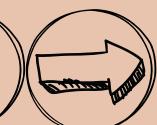
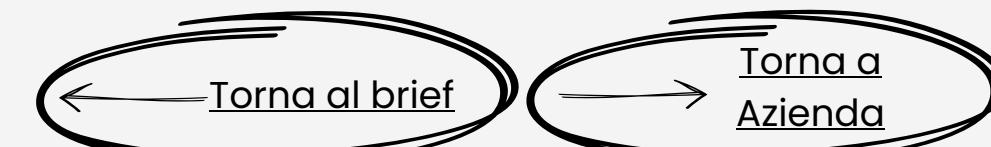
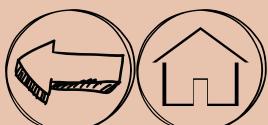


- “mostralscritti (): void” → siccome nell’array “elencolscritti[]” previsto dal brief viene salvato l’intero partecipante, consultare l’elenco di chi è iscritto a un corso potrebbe essere complicato (ad esempio: se a una persona del team IncluDO servisse l’elenco dei partecipanti di un corso molto frequentato, dovrebbe scorrere righe e righe di informazioni e segnarsi manualmente il loro nome e cognome - un’operazione prona all’errore umano e molto *time consuming*); per questo, ho previsto il metodo “mostralscritti()” che permette di visualizzare solamente un elenco con nome e cognome di chi è iscritto a un corso.

L'INTERFACCIA AZIENDA

Anche questa interfaccia si trova nel file *interfaces.ts* e rappresenta le aziende che collaborano con IncluDO. Come da brief, è composta da:

- “*nomeAzienda: string*” → una stringa per il nome dell’azienda;
- “*descrizioneAzienda: string*” → una stringa in cui inserire una breve descrizione dell’azienda;
- “*posizioniAperte?: string[]*” → un array di stringhe in cui sono raccolte le posizioni aperte disponibili presso l’azienda; ho reso questa proprietà opzionale per fare in modo che sia possibile mantenere “valida” l’interfaccia (e la relativa classe) anche nel caso in cui un’azienda non avesse, in un determinato momento, delle posizioni lavorative aperte;
- “*offriPosizione(partecipante: IPartecipante, posizione: string): void*” → il metodo che permette all’azienda di offrire una posizione a un partecipante.



Oltre alle proprietà richieste dal brief, ne ho aggiunta una chiamata “*corsoRichiesto: string*” con cui l’azienda indica l’*idCorso* (cfr. interfaccia “*ICorso*”) che il partecipante deve aver seguito per lavorare in quella azienda; se l’azienda non prevede corsi specifici (es. [“Agriturismo Masseria Tramonti”](#)), questa proprietà è impostata su “varie”. Questo per assicurarsi che un partecipante abbia effettivamente le competenze che l’azienda sta cercando.



LA CLASSE PARTECIPANTE

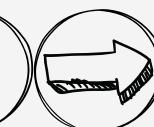


Nel file “`partecipante.ts`” si trova la classe `Partecipante` che, come detto, implementa l’interfaccia `IPartecipante` (importata dal file `interfaces.ts` insieme all’interfaccia `ICorso`, che serve per il metodo “`iscrivitiCorso`”).

Nel costruttore della classe, le proprietà hanno visibilità “`public`” per assicurarne una lettura corretta da qualunque punto di questo codice.

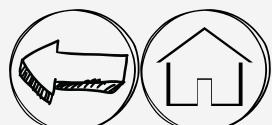


[Torna al brief](#)



Qui si trova la definizione del metodo “*iscrivitiCorso*”, che permette a un partecipante di iscriversi a un corso di sua scelta. Il metodo, come da brief, tipizza *ICorso* per garantire che tutti gli oggetti passati rispettino il contratto di quell’interfaccia, assicurandosi così che l’interazione tra i moduli sia sempre coerente e tipizzata. Il metodo, che interagisce con “*aggiungiPartecipante*” della classe *Corso*, funziona così:

- quando *iscrivitiCorso* viene invocato, questo riceve un “corso” tipizzato come *ICorso* (*corso: ICorso*);
- il metodo non restituisce nulla se non dei console log e degli alert (motivo per cui, nella firma, il return è *void*):
- siccome “corso” è di “tipo” *ICorso*, al suo interno contiene il metodo “*aggiungiPartecipante*” - quello, cioè, responsabile di aggiornare l’elenco degli iscritti di suddetto corso;
- viene quindi invocato il metodo “*aggiungiPartecipante*” del corso richiesto, e “questo” partecipante (quello da cui è partito “*iscrivitiCorso*”) viene aggiunto all’elenco dei suoi iscritti (*corso.aggiungiPartecipante(this)*);
- la comparsa di un log in console (*console.log*) e di un ipotetico pop-up (*alert*) confermano che l’operazione è avvenuta con successo.

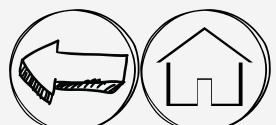


LA CLASSE CORSO

Nel file “corso.ts” si trova la classe Corso, che implementa l'interfaccia ICorso (importata dal file *interfaces.ts* insieme all’interfaccia *IPartecipante*, che serve per il metodo “*iscrivitiCorso*”).

Anche nel costruttore di questa classe hanno visibilità “*public*” per fare in modo che vengano lette correttamente da qualunque punto del codice.

Sempre nel costruttore viene dichiarato l’array vuoto “*elencoPartecipanti: IPartecipanti[]*”: viene dichiarato in questo modo per assicurarsi che la proprietà esista sempre.

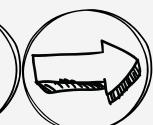


[Torna al brief](#)



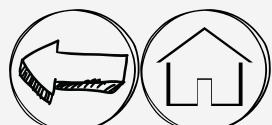
Il primo dei due metodi in questa classe, “aggiungiPartecipante”, è responsabile dell’aggiornamento dell’elenco dei partecipanti iscritti a un corso. Funziona così:

- il metodo riceve un “partecipante”, tipizzato come *IPartecipante* (*partecipante: IPartecipante*), e non restituisce valori se non un console log e un (ipotetico) alert (*void*);
- quando il metodo viene invocato, questo accede all’array “elencolscritti[]” del corso da cui il metodo viene invocato e vi aggiunge un oggetto “partecipante” (*this.elencolscritti.push(partecipante)*);
- a operazione avvenuta, viene visualizzato un messaggio di conferma in console (*console.log*) e dovrebbe comparire pop-up (*alert*) di conferma.



Il secondo dei due metodi, invece, è il metodo “extra” “mostraPartecipanti”. Il suo scopo è quello di mostrare un elenco facilmente consultabile dei partecipanti iscritti a un corso e funziona così:

- il metodo accede all’elenco degli iscritti del corso da cui il metodo è stato invocato e ne “passa in rassegna” tutti gli elementi (“iscritto”) che contiene (`this.elencolscritti.map((iscritto) =>`);
- per ogni elemento, il metodo accede all’oggetto, estrae il contenuto delle proprietà “nome” e “cognome” e li concatena in un’unica stringa, separandoli con uno spazio per migliorarne la lettura (`=> iscritto.nome + " " + iscritto.cognome`);
- il risultato del mapping viene “archiviato” dentro la costrante “`iscritti`”, che quindi contiene il nuovo array; l’array “`iscritti`” viene poi mostrato in console con i risultati separati da un punto e virgola (`${iscritti.join("; ")}`).



LA CLASSE AZIENDA

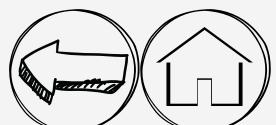


Nel file “azienda.ts” si trova la classe *Azienda*, che implementa l’interfaccia *IAzienda* (importata dal file *interfaces.ts*) insieme all’interfaccia *IPartecipante*, che serve per il metodo “*offriPosizione*”).

Anche nel costruttore di questa classe, le proprietà hanno visibilità “public” per fare in modo che siano visibili da ogni punto del codice.

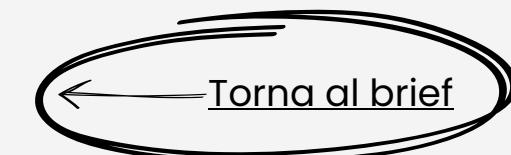
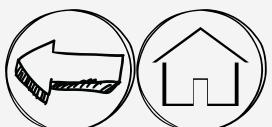
Anche questa classe prevede un metodo: “*offriPosizione*”, che permette a un’azienda di offrire una posizione (lavorativa o di tirocinio) a un/a partecipante. Questo metodo fa 3 cose:

1. invia un'email a un partecipante in cui gli/le viene comunicato che un’azienda (quella da cui è stato invocato il metodo) gli/le sta offrendo un’opportunità lavorativa (l'email è da parte della segreteria di IncluDO e invita chi la riceve a mettersi in contatto con loro, che faranno da ponte tra lui/lei e l’azienda);
2. invia un'email alla segreteria IncluDO che li avverte che è stata proposta una posizione a un/a partecipante (e che quindi potrebbero essere contattati per mettersi e, eventualmente, dover fare da supporto se il/la partecipante ne avesse bisogno);
3. invia un'email all’azienda da cui è stato invocato il metodo per confermare che l’operazione è avvenuta con successo.



Il metodo è molto semplice e funziona così:

- riceve un argomento “partecipante” (tipizzato come *IPartecipante*) e una “posizione” (di tipo *string*) e non restituisce altro che un “invio di email”, qui simulato con una serie di console log (*(partecipante: IPartecipante, posizione: string): void*);
- per prima cosa, viene inizializzata una costante (*messaggioOfferta*) che contiene un template literal con il messaggio pre-compilato che viene inviato al/la partecipante IncluDO con l'offerta lavorativa;
- viene, poi, inizializzata una seconda costante (*messaggioSegreteria*) che contiene un template literal con il messaggio che comunica alla segreteria IncluDO che è stata proposta una posizione a un/a partecipante;
- quindi, viene inizializzata una terza costante (*messaggioConfermaAzienda*) che contiene un template literal con il messaggio che conferma all'azienda che l'email è stata inviata correttamente;
- infine, vengono lanciati 3 console log (uno per messaggio) che simulano l'invio delle email; a questo punto, è previsto anche un pop-up di conferma.

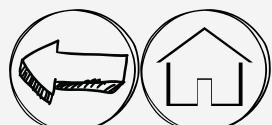


IL FILE APP.TS

Il file `app.ts`, come detto all'inizio, è il “punto d'ingresso” della app. Nel file non ci sono solo gli import delle 3 classi (`Partecipante`, `Corso` e `Azienda`), ma anche l'istanziazione, la configurazione e la logica di test dell'intera applicazione: è qui, infatti, che

- sono state istanziati tutti gli oggetti delle classi `Partecipante`, `Corso` e `Azienda`;
- è stata creata una “testing zone” in cui provare l'interazione tra gli elementi dell'applicazione;
 - in particolare, è stato lanciato il “`console.log`” di due partecipanti, due corsi e due aziende e sono stati invocati i metodi `iscrivitiCorso`, `aggiungiPartecipante`, `mostralscritti` e `offriPosizione`.

Il file è stato transpilato nel suo corrispettivo “`app.js`”, che è poi stato inserito in un generico “`index.html`” per poter visualizzare il funzionamento dello script attraverso la console di una pagina web. Per la migliore compatibilità browser, ho scelto “ES2018” come versione del linguaggio per il JavaScript emesso.

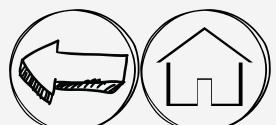


Una nota: negli import presenti in questo file (app.ts) si può notare l'estensione “.js”

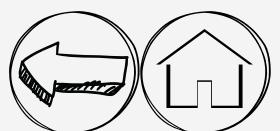
```
import { Parecipante } from "./partecipante.js";  
import { Corso } from ".corso.js";  
import { Azienda } from "azienda.js"
```

Non è una svista, ma una scelta voluta: serve a forzare la generazione di link relativi con la giusta estensione (.js) anche dentro il file transpilato app.js, in modo che il browser possa trovare i file corretti senza dover ricorrere a un bundler. Non ho toccato in alcun modo, invece, nessuno dei file “.js”.

Ho preso questa decisione per un semplice motivo: trattandosi di un progetto di dimensioni molto ridotte, non ho ritenuto valesse la pena “appesantire” il programma aggiungendovi un bundler per risolvere questo problema. Se, però, in futuro avessi bisogno di ingrandire il progetto, mi basterà semplicemente rimuovere queste tre estensioni (che sono solo 3 e sono tutte concentrate in un unico file), aggiornare il file di configurazione di TypeScript (tsconfig.json) per far sì che rifletta l'adozione di un bundler (“moduleResolution”: “bundler”) e procedere, poi, di conseguenza.



Grazie per l'attenzione!



APPENDICE 1: I PARTECIPANTI

AMARA DIALLO

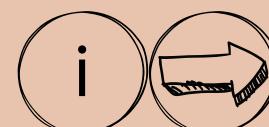
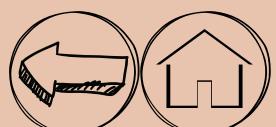
- *Paese d'origine:* Senegal
- *Livello di istruzione:* Diploma di scuola superiore
- *Competenze linguistiche:* Francese (madrelingua), wolof (madrelingua) (italiano: A2)
- *Ambito di interesse:* Artigianato della ceramica
- *La sua storia:* Fuggita dalla siccità in Senegal, sogna di aprire un laboratorio di ceramica.

HASSAN AL RASHID

- *Paese d'origine:* Siria
- *Livello di istruzione:* Laurea in ingegneria
- *Competenze linguistiche:* Arabo (madrelingua), inglese (B2) e italiano (B1)
- *Ambito di interesse:* Artigianato del ferro
- *La sua storia:* Ingegnere siriano, vuole ricostruire la sua vita lavorando con le mani.

MARIA SANTOS

- *Paese d'origine:* Brasile
- *Livello di istruzione:* Licenza di scuola media
- *Competenze linguistiche:* Portoghese (madrelingua) e spagnolo (C1) (italiano: A2)
- *Ambito di interesse:* Sartoria
- *La sua storia:* Madre single brasiliana, cerca stabilità per sé e per sua figlia.



FATOU KONÉ

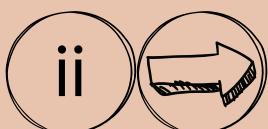
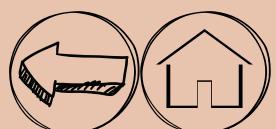
- *Paese d'origine:* Costa d'Avorio
- *Livello di istruzione:* Scuola primaria
- *Competenze linguistiche:* Baoule (madrelingua) e francese (B2) (italiano: A1)
- *Ambito di interesse:* Alimentare, panificazione e ristorazione
- *La sua storia:* Vedova ivoriana, ama cucinare e vuole tramandare le ricette della nonna.

NICOLAE POPESCU

- *Paese d'origine:* Romania
- *Livello di istruzione:* Formazione carpentiere
- *Competenze linguistiche:* Romeno (madrelingua) e italiano (B2)
- *Ambito di interesse:* Artigianato del ferro e restauro
- *La sua storia:* Ex carpentiere in Italia da più di trent'anni, dopo un incidente cerca una nuova opportunità lavorativa.

AISHA OSMAN

- *Paese d'origine:* Somalia
- *Livello di istruzione:* Diploma di scuola superiore
- *Competenze linguistiche:* Somalo (madrelingua), arabo (B2), inglese (B1) (italiano: A2)
- *Ambito di interesse:* Sartoria, alimentare e ristorazione
- *La sua storia:* Arrivata dalla Somalia, ha sempre amato cucinare e creare vestiti.



CARLOS MENDOZA

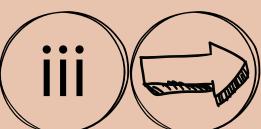
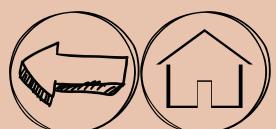
- *Paese d'origine:* Perù
- *Livello di istruzione:* Laurea in agronomia
- *Competenze linguistiche:* Spagnolo (madrelingua), quechua (C2) e italiano (B1)
- *Ambito di interesse:* Agricoltura, ulivocoltura
- *La sua storia:* Agronomo peruviano, vuole unire scienza e tradizione agricola locale.

IVANA PETKOVIC

- *Paese d'origine:* Serbia
- *Livello di istruzione:* Licenza scuola media
- *Competenze linguistiche:* Serbo (madrelingua) (italiano: A2)
- *Ambito di interesse:* Artigianato della ceramica, artigianato della pietra e artigianato del ferro
- *La sua storia:* Artista frustrata, cerca di esprimere la sua creatività attraverso la materia.

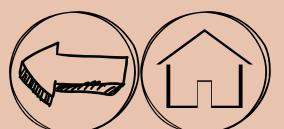
MAMADOU TRAORE

- *Paese d'origine:* Mali
- *Livello di istruzione:* Scuola primaria
- *Competenze linguistiche:* Bambara (madrelingua) e francese (B2) (italiano: A1)
- *Ambito di interesse:* Falegnameria e restauro
- *La sua storia:* Giovane maliano che sogna di diventare un maestro falegname come il nonno.



ELENA MORAU

- *Paese d'origine:* Moldavia
- *Livello di istruzione:* Diploma di scuola superiore
- *Competenze linguistiche:* Moldavo (madrelingua) e romeno (C2) (italiano: A2)
- *Ambito di interesse:* Alimentare e panificazione
- *La sua storia:* Nonna moldava, vuole preservare le antiche ricette di conserve familiari.



APPENDICE 2: I CORSI

ITALIANO COME LINGUA STRANIERA

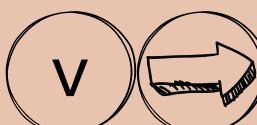
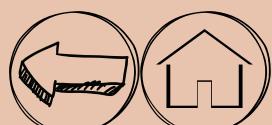
- *Descrizione:* Corso di italiano come seconda lingua per comunicare in maniera efficace e destreggiarsi in un mondo totalmente nuovo; consigliato in particolare per chi ha un livello di italiano A2 o inferiore.
- *Durata mesi:* 4
- *Durata ore:* 320

PANETTERIA TRADIZIONALE

- *Descrizione:* Corso completo di panificazione tradizionale pugliese, dalle farine tipiche locali all'utilizzo del forno a legna; include corso per la sicurezza alimentare e HCCP.
- *Durata mesi:* 4
- *Durata ore:* 320
- *Ore teoria:* 70
- *Ore pratica:* 250

TESSITURA, SARTORIA E RICAMO TRADIZIONALE

- *Descrizione:* Apprendimento delle tecniche sartoriali tradizionali e delle tecniche per realizzare i ricami tipici pugliesi; include taglio, cucito (sia a mano che a macchina) e creazione di decorazioni folkloristiche.
- *Durata mesi:* 6
- *Durata ore:* 480
- *Ore teoria:* 90
- *Ore pratica:* 390



MAESTRO DEL FERRO BATTUTO

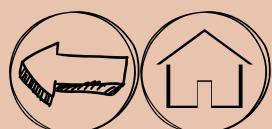
- *Descrizione:* Formazione nella lavorazione artistica del ferro, dalle tecniche di fagiola tradizionale alla creazione di cancelli, ringhiere e oggetti decorativi; include corsi specifici sulla sicurezza e sull'uso degli strumenti.
- *Durata mesi:* 6
- *Durata ore:* 480
- *Ore teoria:* 100
- *Ore pratica:* 380

RESTAURATORE MOBILI ANTICHI

- *Descrizione:* Corso per apprendere le tecniche di restauro conservativo di mobili d'epoca e del trattamento di legno e decorazioni tradizionali; include panoramica per il riconoscimento degli stili decorativi e il loro trattamento.
- *Durata mesi:* 7
- *Durata ore:* 560
- *Ore teoria:* 100
- *Ore pratica:* 460

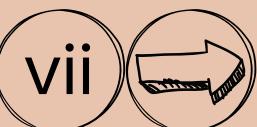
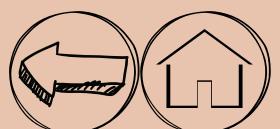
OLIVICOLTORE E FRANTOIANO

- *Descrizione:* Corso completo sulla coltivazione degli ulivi e sulla produzione dell'olio extra-verGINE, dalle tecniche di coltivazione tradizionali alle innovazioni tecnologiche più recenti; include l'insegnamento della gestione del frantoio.
ATTENZIONE: il corso è stagionale.
- *Durata mesi:* 9
- *Durata ore:* 720
- *Ore teoria:* 120
- *Ore pratica:* 600



CERAMICA ARTISTICA E DECORATIVA

- *Descrizione:* Corso intensivo per apprendere le tecniche tradizionali della ceramica pugliese, dalla lavorazione dell'argilla alla decorazione con motivi locali; oltre alla decorazione, il corso comprende l'uso del tornio, la cottura della ceramica e la smaltatura.
- *Durata mesi:* 6
- *Durata ore:* 480
- *Ore teoria:* 100
- *Ore pratica:* 380



APPENDICE 3: LE AZIENDE PARTNER

OLEIFICIO SAN GIUSEPPE

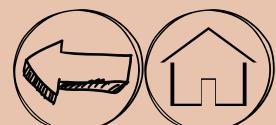
- *Descrizione:* Frantoio familiare attivo dal 1952 e specializzato nella produzione di olio extravergine di oliva biologico; gestisce 200 ettari di uliveti e un moderno frantoio che combina tecniche tradizionali e innovative.
- *Posizioni aperte:* Operatore frantoiano; addetto coltivazione uliveti.

MEDITERRA CERAMICHE D'ARTE

- *Descrizione:* Laboratorio artigianale specializzato in ceramiche decorative tradizionali pugliesi. Produce oggetti per l'arredamento, stoviglie artistiche e pezzi su commissione per hotel e ristoranti.
- *Posizioni aperte:* Ceramista junior; decoratore ceramico.

PANIFICIO TRADIZIONI DI GRANO

- *Descrizione:* Panificio artigianale che utilizza farine locali e lievito madre per produrre pane, focacce e altri prodotti tipici pugliesi. Fornisce ristoranti, hotel e mercati locali.
- *Posizioni aperte:* Panettiere; addetto vendite specializzato.



FERRO BATTUTO MERIDIONALE

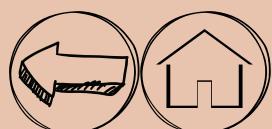
- *Descrizione:* Officina specializzata in cancelli, ringhiere e complementi d'arredo in ferro battuto. Lavora per privati, comuni e strutture turistiche per la realizzazione di opere su misura.
- *Posizioni aperte:* Fabbro specializzato; apprendista saldatore.

FALEGNAMERIA RADICI

- *Descrizione:* Laboratorio di falegnameria tradizionale specializzato in mobili su misura, restauri e oggetti in legno massello. Utilizza essenze locali e tecniche di incastro tradizionali.
- *Posizioni aperte:* Falegname qualificato; Addetto al restauro mobili.

ATELIER FILI D'ORO

- *Descrizione:* Laboratorio sartoriale specializzato in abiti tradizionali pugliesi, costumi folkloristici e ricami artistici. Collabora con gruppi locali e produce per il mercato turistico.
- *Posizioni aperte:* Sarta specializzata; ricamatore o ricamatrice.



TESSUTI DEL SUD

- *Descrizione:* Laboratorio di tessitura artigianale che produce tessuti per l'arredamento e l'abbigliamento utilizzando fibre naturali locali.
- *Posizioni aperte:* Tessitore esperto; addetto preparazione filati.

AGRITURISMO MASSERIA DEI TRAMONTI

- *Descrizione:* Struttura turistica di 15 camere con ristorante tipico, fattoria didattica e laboratori artigianali. Organizza workshop turistici sui mestieri tradizionali da proporre ai suoi ospiti.
- *Posizioni aperte:* Addetto ai laboratori didattici; animatore workshop.

