**ChatGPT ha detto:**

Perfetto! Iniziamo con la **relazione tecnica del componente app-custom-alert**, basata sui file che hai caricato (custom-alert.component.ts, .html, .scss).

**Componente: app-custom-alert**

Il componente **app-custom-alert** fornisce una finestra di dialogo (modal) personalizzata per mostrare messaggi di conferma o avviso all’utente. È progettato per essere riutilizzabile all’interno di qualsiasi pagina del progetto.

Il codice si trova nella cartella src/app/components/custom-alert/.

**🔧 Funzionalità**

Il componente espone le seguenti proprietà @Input:

* alertTitle: titolo visualizzato nella parte superiore della modale.
* alertMessage: messaggio centrale del corpo.
* cancelText: etichetta del pulsante di annullamento.
* confirmText: etichetta del pulsante di conferma.
* isVisible: determina se l’alert è attualmente visibile.

Espone inoltre due @Output:

* confirmed: evento emesso quando l’utente conferma.
* cancelled: evento emesso quando l’utente annulla o chiude.

**🧩 Struttura HTML**

Nel file custom-alert.component.html, la modale è composta da:

* Un backdrop semitrasparente che chiude l’alert al clic ((click)="closeAlert()").
* Una finestra floating-modal con:
  + modal-title: titolo dinamico.
  + modal-body: messaggio principale.
  + modal-footer: due pulsanti: annulla e conferma.

**🎨 Stile**

Il file SCSS (custom-alert.component.scss) gestisce:

* Posizionamento fisso sopra il contenuto della pagina.
* Transizioni fluide in apertura/chiusura.
* Differenziazione visiva tra pulsante di conferma e annullamento.
* Colori coerenti con la palette principale dell’app (bordeaux, senape, bianco).

**💡 Utilizzo tipico**

html

CopiaModifica

<app-custom-alert

[alertTitle]="'Attenzione'"

[alertMessage]="'Vuoi davvero eliminare questo itinerario?'"

[cancelText]="'Annulla'"

[confirmText]="'Conferma'"

[isVisible]="showDeleteAlert"

(confirmed)="deleteConfirmed()"

(cancelled)="deleteCancelled()"

></app-custom-alert>

**✅ Vantaggi**

* Componente completamente **standalone e riutilizzabile**.
* Facilmente tematizzabile.
* Nessuna dipendenza esterna, può essere richiamato da qualsiasi pagina con logica personalizzata.

**Componente: app-custom-sidebar**

Il componente **app-custom-sidebar** implementa un menu laterale (drawer) utilizzando il sistema ion-menu di Ionic. È progettato per fornire un accesso rapido alle sezioni principali dell'applicazione, ed è integrabile in qualsiasi layout di pagina con identificatore content-id="main-content".

Il componente si trova in src/app/components/custom-sidebar/.

**🧩 Struttura HTML**

Nel file custom-sidebar.component.html troviamo:

* <ion-menu>: struttura standard del menu laterale con:
  + side="start": comparsa da sinistra
  + menu-id="main-menu": identificativo logico
  + content-id="main-content": collega il menu alla pagina principale
* <ion-list> con voci di menu:
  + **Profilo** → routerLink="/tabs/profilo"
  + **Impostazioni** → routerLink="/settings"
  + **Storico** → routerLink="/storico"
  + **Logout** → bottone con chiamata alla funzione logout()

Ogni ion-item include:

* Icona (ion-icon) coerente con la funzione
* Etichetta (ion-label) descrittiva

**🔧 Funzionalità**

* Le voci di menu navigano tra le sezioni principali dell'app.
* La voce "Esci" esegue una funzione logout(), che molto probabilmente cancella il token di sessione, resetta i dati locali e reindirizza alla pagina di login.

**🎨 Stile**

Il file SCSS (custom-sidebar.component.scss) gestisce:

* Colori coerenti con il tema dell’app (ion-toolbar color="primary")
* Spaziature e separazione tra le voci
* Eventuale ombreggiatura laterale o background del menu

**📦 Integrazione**

Il componente può essere incluso globalmente nel layout root, o solo in alcune pagine (ad esempio nel wrapper tabs), tramite:

html

CopiaModifica

<app-custom-sidebar></app-custom-sidebar>

<ion-router-outlet id="main-content"></ion-router-outlet>

**✅ Vantaggi**

* Navigazione semplice, sempre accessibile con swipe o bottone menù.
* Nessuna logica rigida: facilmente estendibile con nuove voci.
* Compatibile con tutti i dispositivi mobili grazie all’uso del componente ion-menu.

**omponente: app-generation-overlay**

Il componente **app-generation-overlay** fornisce una **schermata di caricamento full-screen** sovrapposta all’interfaccia utente, con messaggio opzionale.  
È pensato per fasi critiche dell'app come la generazione dell’itinerario o il caricamento da API, in modo da informare visivamente l’utente e prevenire interazioni.

Il codice si trova nella cartella src/app/components/generation-overlay/.

**🧩 Struttura HTML**

Nel file generation-overlay.component.html:

* Un contenitore principale .overlay-container copre tutta la viewport.
* All’interno:
  + Un elemento .loader-animation per visualizzare un’animazione (spinner o effetto custom).
  + Un header <h2> con messaggio dinamico, nascosto se message è falsy ([hidden]="!message").

**🔧 Funzionalità**

* Il componente riceve un input @Input() message: string, che viene mostrato come messaggio principale (es. “Generazione in corso…”).
* La presenza del componente nella pagina **oscura lo sfondo** e **blocca l’interazione** con altri elementi.

Esempio di utilizzo:

html

CopiaModifica

<app-generation-overlay \*ngIf="isLoading" [message]="'Generazione in corso…'"></app-generation-overlay>

**🎨 Stile**

Nel file generation-overlay.component.scss troviamo:

* Uno sfondo semitrasparente per coprire l’intera pagina (position: fixed, z-index alto).
* Centramento verticale/orizzontale dell’animazione e del testo.
* L’animazione può essere uno spinner, un effetto pulsante o una gif SVG/CSS.

**✅ Vantaggi**

* Separazione della logica di loading dal contenuto.
* Utilizzo semplice e riusabile in qualsiasi punto dell’app.
* Utile per evitare comportamenti errati dell’utente durante operazioni asincrone lunghe.

**Componente: app-google-autocomplete**

Il componente **app-google-autocomplete** fornisce un campo input avanzato con suggerimenti dinamici basati sulle **Google Places API**, filtrabili per tipo di luogo, bounds geografici, e con possibilità di prefetch dei dettagli.

È uno dei componenti più usati nel progetto, presente in:

* Creazione itinerario (città, alloggio)
* Itinerario (aggiunta luoghi)
* Personalizzazione
* Ricerca dinamica in altre pagine

Il codice è collocato in src/app/components/google-autocomplete/.

**⚙️ Funzionalità principali**

Il componente implementa:

* **Autocompletamento dinamico** con Google Places API (autocompleteService.getPlacePredictions(...))
* Emissione di eventi:
  + @Output() placeSelected → restituisce l’intero PlaceResult
  + @Output() valueChange → usato per azzerare lo stato o disabilitare pulsanti esterni
* Supporta restrizioni:
  + bounds: limitazione geografica dei risultati
  + types: filtro tipologie (es. ['(cities)'], ['restaurant'], ['lodging'])
  + strictBounds opzionale

**🧠 Logica interna**

Nel .ts vengono gestite:

* Inizializzazione AutocompleteService e PlacesService
* Visualizzazione delle suggestions[] tramite una lista a scomparsa
* Metodo fetchPredictions() che invia la richiesta con:

ts

CopiaModifica

const req: AutocompletionRequest = {

input,

sessionToken,

types,

bounds,

strictBounds: restrictToBounds

}

* Funzione prefetchDetails() per caricare in background i dettagli dei primi 5 risultati
* Metodo selectSuggestion() che salva e trasmette il risultato selezionato

**🎨 Stile (SCSS)**

Il file google-autocomplete.component.scss gestisce:

* Aspetto dell’input (.autocomplete-input)
* Lista risultati (.suggestions-list)
* Hover sugli item (li:hover)
* Layering corretto per evitare conflitti (z-index alto)

**💡 Esempio d’uso**

html

CopiaModifica

<app-google-autocomplete

[placeholder]="'Seleziona città'"

[types]="['(cities)']"

[(value)]="cityInput"

[restrictToBounds]="false"

(placeSelected)="handleCityPlace($event)">

</app-google-autocomplete>

**✅ Vantaggi**

* Componente riutilizzabile e altamente configurabile
* Perfettamente integrato con Google Maps e Places
* Interfaccia pulita, completamente adattata a mobile
* Gestione interna di selezione, chiusura lista, fallback

**Componente: app-header / app-navigation-bar**

Il componente **app-header** rappresenta una **barra di navigazione superiore** riutilizzabile in tutte le pagine dell'app. Semplifica l'inclusione di un titolo coerente, eventuali pulsanti “indietro”, o elementi personalizzati nei ion-buttons.

Si trova nella cartella src/app/components/app-header/.

**🧩 Struttura HTML**

Il file app-header.component.html contiene:

* **Includo Google Font**: Dancing Script per un tocco personalizzato al titolo.
* **<ion-header>**: componente nativo Ionic con classe condizionale no-box-shadow basata su @Input() box\_shadow.
* **<ion-toolbar>**: contiene il title al centro e pulsanti opzionali agli slot start o end.

Nel codice sono **commentati** alcuni pulsanti utili:

* Tasto back: <ion-back-button \*ngIf="showBackButton">
* Icona profilo o impostazioni
* Icona home (in alternativa al back button)

**⚙️ Funzionalità**

Il componente espone:

* @Input() title: string → titolo centrale della toolbar
* @Input() showBackButton: boolean = true → mostra/nasconde il pulsante indietro
* @Input() box\_shadow: boolean = false → abilita/disabilita l’ombra sotto l’header

Nel file .ts, è presente una funzione NavHome() che può essere legata al click sul titolo per tornare alla home.

**🎨 Stile**

Il file app-header.component.scss:

* Personalizza l’aspetto della ion-toolbar
* Usa il font Dancing Script
* Gestisce l’ombra con la classe no-box-shadow e altri effetti visivi

**💡 Esempio d’uso**

html

CopiaModifica

<app-header [title]="'SoTourist'" [showBackButton]="false" [box\_shadow]="true"></app-header>

Oppure nella variante navigation-bar:

html

CopiaModifica

<app-navigation-bar title="Panoramica" [backUrl]="'/tabs/viaggi'"></app-navigation-bar>

**✅ Vantaggi**

* Componente **riutilizzabile** con titolo dinamico.
* Estensibile con pulsanti personalizzati.
* Visualmente coerente con lo stile dell’applicazione.
* Facilmente tematizzabile e compatibile con tutte le pagine Ionic.