# Specifica Tecnica

## Indice generale

# Introduzione

## Scopo del documento

Il documento ha lo scopo di definire la struttura architetturale e il design di dettaglio dell'applicazione *ADeQA* descrivendo le tecnologie utilizzate, le API utilizzate e l'architettura.

## Riferimenti

Molti riferimenti all'interno del documento riguardano il documento *Analisi dei Requisiti*, in particolar modo alla sezione "*Architettura*" relativamente alla sotto-sezione "*requisiti soddisfatti*".

# Tecnologie

## Introduzione

La seguente sezione elenca in modo esaustivo le tecnologie scelte e concordate per l'implementazione di ADeQA.

## Elenco delle tecnologie

| Nome | Versione | Descrizione |
|---|---|---|
| Linguaggi | | |
| CSS | 3 | Il linguaggio di definizione di stili standard del web, mantenuto dal consorzio W3C |
| HTML | 5 | Il linguaggio di markup standard per la creazione di pagine web, mantenuto dal consorzio W3C |
| Typescript | 5.1.6 | Linguaggio di programmazione orientato agli oggetti, è fortemente tipizzato per agevolare la scrittura di codice Javascript (di cui è sovra-linguaggio). Compila nello standard ECMAScript 6, producendo codice adatto all'uso su pagine web |
| Framework | | |
| Angular | 16.2.9 | Framework per la creazione di single-page application per browser / Progressive Web App |
| Jasmine | 4.6.0 | Framework open-source per il testing di codice Javascript |
| Karma | 6.4.0 | Strumento che genera un server web che esegue il codice di test Javascript per ogni browser connesso |
| Strumenti di utilità | | |
| Compodoc | 1.1.22 | Strumento open-source per la generazione di documentazione per web app Angular a partire da commenti scritti nel codice sorgente |
| ng-openapi-gen | 0.50.2 | Modulo NPM che genera servizi, modelli e funzioni Angular a partire da una specifica OpenAPI 3 |
| Librerie | | |
| Ngx-translate | 15.0.0 | Libreria di internazionalizzazione per Angular |
| Rxjs | 7.8.0 | Libreria per programmazione reattiva e asincrona |
| Angular Material | 16.2.8 | Libreria che mette a disposizione componenti grafiche per web app Angular |
| Strumenti di sviluppo | | |
| VSCode | 1.84.1 | Editor di testo con la possibilità di integrare plugin |
| Figma | 9.0 | Web-app collaborativa per la creazione di design di interfaccia grafica |
| StarUML | 6.0.1 | Applicazione per la modellazione di diagrammi UML, utilizzata in particolare per i diagrammi dei casi d'uso ed i diagrammi delle classi |
| Git | 2.42.0.2 | Sistema di controllo di versione distribuito |
| Runtime | | |
| Node.js | 18.17.1 | Runtime asincrono ed event-driven per linguaggio JavaScript |

## Internazionalizzazione e localizzazione

Per poter capire le scelte progettuali occorre prima dare delle definizioni ai seguenti termini:

- **Internazionalizzazione**: è il processo di design e sviluppo di un prodotto in modo tale da ridurre la difficoltà di adattarlo per renderlo fruibile da persone appartenenti a culture diverse;
- **Localizzazione**: è il processo di adattamento di un prodotto in base alle esigenze culturali di una particolare area / di un particolare mercato.

Per quanto riguarda l'internazionalizzazione e la localizzazione del contenuto statico dell'applicazione, si è scelto di usare la libreria **ngx-translate** al posto del pacchetto **@angular/localize**, nativamente supportato da Angular; di seguito, una breve analisi sulle potenzialità e sulle carenze di entrambe le tecnologie.

**@angular/localize**

- Vantaggi:
  - Supportato nativamente dal framework Angular;
  - Velocità di esecuzione delle applicazioni che la usano;
  - Supporto concreto per progetti di grandi dimensioni data la scalabilità offerta.
- Svantaggi
  - Necessita di una compilazione manuale per aggiornare le traduzioni;
  - Necessita di un passaggio manuale di dati per aggiornare ogni traduzione;
  - Presenza di multiple versioni dell'applicazione (una per ogni traduzione);
  - Ogni versione necessita del proprio processo di build;
  - Ogni cambio di lingua si traduce nel caricamento di una versione diversa della stessa applicazione (con conseguente caricamento di una nuova pagina).

**ngx-translate**

- Vantaggi:
  - Di facile utilizzo e apprendimento;
  - Utilizzo di file JSON, facilmente gestibili e modificabili;
  - Consente di gestire la mancanza di traduzione;
  - Consente un cambio di linguaggio di traduzione senza necessità di ricaricare la pagina.
- Svantaggi:
  - E' una libreria nata come soluzione temporanea ai problemi di internazionalizzazione del framework Angular (https://github.com/ngx-translate/core/issues/495) e quindi non è frequentemente aggiornata;
  - Appesantisce l'esecuzione dell'applicazione.

La scelta di **ngx-translate** è stata dettata dalle ridotte dimensioni del prodotto e dalle sue esigenze, in particolare:

1. **Integrazione in un *iframe***: dato che la web-app deve essere in grado di eseguire all'interno di un *iframe* e l'applicazione "contenitore" utilizza tale libreria, si vuole garantire una traduzione uniforme all'interno della schermata visualizzata, evitando di ricaricare l'applicazione all'interno della porzione di schermo delimitata dall'*iframe*;
2. **Semplicità di sviluppo**: l'applicazione non ha dimensioni tali da prendere in considerazione misure di traduzione scalabili; questo consente di prediligere un approccio orientato alla semplicità di utilizzo durante lo sviluppo e la manutenzione del prodotto;
3. **Semplicità di test**: il tempo dedicato al testing delle traduzioni deve essere proporzionale all'apporto di tale funzionalità ed alle dimensioni dell'applicazione.

# API

L'applicativo ADeQA utilizza delle API messe a disposizione da un backend comune a tutta la suite di prodotti TriZeta; il corpo delle richieste è in formato JSON, così come il corpo delle risposte.

Di seguito sono riportate le API utilizzate da ADeQA:

## Autenticazione

### Login

**Descrizione:** questo endpoint permette l'autenticazione al sistema.

- **Endpoint:** /login
- **Metodo HTTP:** POST
- **Corpo richiesta:**
  - *username*: nome utente identificativo per l'accesso;
  - *password*: stringa per garantire l'identità del soggetto all'accesso.

| Esito | Codice HTTP | Body | Descrizione |
|---|---|---|---|
| Positivo | 200 | {<br>"token": "string"<br>} | Ritorna un token (stringa alfanumerica) da usare nelle API che lo richiedono (per autenticazione) |
| Negativo | 500 | {<br>"code": "string",<br>"description": "string"<br>} | Ritorna un JSON di errore avente codice e descrizione |

### Logout

**Descrizione:** questo endpoint permette la disconnessione dal sistema.

- **Endpoint:** /logout
- **Metodo HTTP:** POST

| Esito | Codice HTTP | Body | Descrizione |
|---|---|---|---|
| Positivo | 200 | { | Ritorna lo stato della richiesta |

| | | | |
|---|---|---|---|
| | | "status": "string"<br>} | |
| Negativo | 500 | {<br>"code": "string",<br>"description": "string"<br>} | Ritorna un JSON di errore avente codice e descrizione |

## Fasi

### QualityPhase - Fetch

**Descrizione:** questo endpoint permette di ottenere le fasi di cui eseguire i controlli.

- **Endpoint:** /9000005/qualityphase/fetch
- **Metodo HTTP:** POST
- **Header**: token di autenticazione
- **Corpo richiesta:**
  - *startRow*: riga di partenza da cui si inizia ad acquisire le fasi, tra le righe ottenute dal filtraggio (indicato dall'array *criteria*);
  - *criteria*: array di criteri di filtraggio, aventi:
    - *fieldname*: nome della colonna per la quale filtrare;
    - *value*: valore di confronto per il filtraggio;
    - *operator*: operatore di confronto da applicare.
  - *orderby*: array di criteri di ordinamento, aventi:
    - *columnname*: nome della colonna per la quale ordinare;
    - *direction*: direzione dell'ordinamento.
  - *endRow*: riga da cui si finisce di acquisire dati.

| Esito | Codice HTTP | Body | Descrizione |
|---|---|---|---|
| Positivo | 200 | {<br>"data": [<br>{<br>"phasename": "string",<br>"color": "string",<br>"ad_org_id": 0,<br>"projectplan_timeline_id": 0,<br>"c_projectphase_id": 0,<br>“phasetitle”: string<br>"end_plan": "Unknown Type: date",<br>"linename": "string",<br>"c_projectline_id": 0,<br>"ad_client_id": 0,<br>"isglobal": "Y",<br>"c_bpartner_id": 0,<br>"status": "Y",<br>"start_plan": "Unknown Type: date",<br>"customer": "string"<br>}<br>],<br>"startRow": 0, | Ritorna una lista di fasi filtrata e ordinata come da richiesta, oltre all'indicazione sulle righe ottenute ed il numero di record oggetto della ricerca |

| | | | |
|---|---|---|---|
| | | "endRow": 0,<br>"totalRows": 0<br>} | |
| Negativo | 401 | | L'utente non è autorizzato |
| Negativo | 500 | {<br>"code": "string",<br>"description": "string"<br>} | Ritorna un JSON di errore avente codice e descrizione |

## Operatori

### Operators - Fetch

**Descrizione:** questo endpoint permette di ottenere gli operatori per configurare l'interfaccia grafica in base alle preferenze salvate.

- **Endpoint:** /9000006/operators/fetch
- **Metodo HTTP:** POST
- **Header**: token di autenticazione
- **Corpo richiesta:**
  - *startRow*: riga di partenza da cui si inizia ad acquisire gli operatori, tra le righe ottenute dal filtraggio (indicato dall'array *criteria*);
  - *criteria*: array di criteri di filtraggio, aventi:
    - *fieldname*: nome della colonna per la quale filtrare;
    - *value*: valore di confronto per il filtraggio;
    - *operator*: operatore di confronto da applicare.
  - *orderby*: array di criteri di ordinamento, aventi:
    - *columnname*: nome della colonna per la quale ordinare;
    - *direction*: direzione dell'ordinamento.
  - endRow: riga da cui si finisce di acquisire dati.

| Esito | Codice HTTP | Body | Descrizione |
|---|---|---|---|
| Positivo | 200 | {<br>"data": [<br>{<br>"note": "string",<br>"userpin": "string",<br>"mes_theme": "string",<br>"mes_theme_display": "string",<br>"foto": 0,<br>"numero_matricola": "string",<br>"isactive": "Y",<br>"name": "string",<br>"ismobileuser": "Y",<br>"ad_user_id": 0<br>}<br>],<br>"startRow": 0,<br>"endRow": 0,<br>"totalRows": 0 | Ritorna una lista di operatori filtrata e ordinata come da richiesta, oltre all'indicazione sulle righe ottenute ed il numero di record oggetto della ricerca |

| | | | |
|---|---|---|---|
| | | } | |
| Negativo | 401 | | L'utente non è autorizzato |
| Negativo | 500 | {<br>"code": "string",<br>"description": "string"<br>} | Ritorna un JSON di errore avente codice e descrizione |

## Attributi di qualità

### QualityAttribute - Fetch

**Descrizione:** questo endpoint permette di ottenere i controlli di qualità per il prodotto selezionato.

- **Endpoint:** /9000003/qualityattribute/fetch
- **Metodo HTTP:** POST
- **Header**: token di autenticazione
- **Corpo richiesta:**
  - *startRow*: riga di partenza da cui si inizia ad acquisire gli attributi (controlli) di qualità, tra le righe ottenute dal filtraggio (indicato dall'array *criteria*);
  - *criteria*: array di criteri di filtraggio, aventi:
    - *fieldname*: nome della colonna per la quale filtrare;
    - *value*: valore di confronto per il filtraggio;
    - *operator*: operatore di confronto da applicare.
  - *orderby*: array di criteri di ordinamento, aventi:
    - *columnname*: nome della colonna per la quale ordinare;
    - *direction*: direzione dell'ordinamento.
  - endRow: riga da cui si finisce di acquisire dati.

| Esito | Codice HTTP | Body | Descrizione |
|---|---|---|---|
| Positivo | 200 | {<br>"data": [<br>{<br>"attributedescription": "string",<br>"optionvalue": "string",<br>"groupdescription": "string",<br>"m_product_id": 0,<br>"attributevalue": "string",<br>"m_product_category_id": 0,<br>"ad_reference_id": 0,<br>"attributename": "string",<br>"attributevaluetype": "Y",<br>"groupname": "string",<br>"c_project_attribute_group_id": 0,<br>"attributeseqno": 0<br>}<br>],<br>"startRow": 0,<br>"endRow": 0,<br>"totalRows": 0<br>} | Ritorna una lista di attributi di qualità filtrata e ordinata come da richiesta (ad esempio, per prodotto o per categoria di prodotto) oltre all'indicazione sulle righe ottenute ed il numero di record oggetto della ricerca |

| | | | |
|---|---|---|---|
| Negativo | 401 | | L'utente non è autorizzato |
| Negativo | 500 | {<br>"code": "string",<br>"description": "string"<br>} | Ritorna un JSON di errore avente codice e descrizione |

## Log di qualità

### QualitySaveLog - Add

**Descrizione:** questo endpoint permette di salvare un log contenente i dati di un controllo qualità.

- **Endpoint:** /9000004/qualitysavelog/add
- **Metodo HTTP:** PUT
- **Header**: token di autenticazione
- **Corpo richiesta:**
  - *c_projectphase_id*: identificativo della fase su cui si stanno inserendo valori per il controllo qualità;
  - *qualitystatus*: stato del log di qualità;
  - *qualityvalue*: valore del log di qualità, è una stringa in formato JSON;
  - *c_projectphase_quality_log_id*: identificativo del log di qualità;
  - *isactive*: indica se il log di qualità è attivo o meno.

| Esito | Codice HTTP | Body | Descrizione |
|---|---|---|---|
| Positivo | 200 | {<br>"c_projectphase_id": 0,<br>"qualitystatus": "string",<br>"qualityvalue": "string",<br>"c_projectphase_quality_log_id": 0,<br>"isactive": "Y"<br>} | Ritorna il corpo della richiesta per dare riscontro dei valori inseriti |
| Negativo | 401 | | L'utente non è autorizzato |
| Negativo | 500 | {<br>"code": "string",<br>"description": "string"<br>} | Ritorna un JSON di errore avente codice e descrizione. |

### QualitySaveLog - Remove

**Descrizione:** questo endpoint permette di eliminare un log contenente i dati di un controllo qualità.

- **Endpoint:** /9000004/qualitysavelog/remove
- **Metodo HTTP:** DELETE
- **Header**: token di autenticazione
- **Corpo richiesta:**
  - *c_projectphase_quality_log_id*: identificativo del log di controllo qualità da rimuovere.

| Esito | Codice HTTP | Body | Descrizione |
|---|---|---|---|
| Positivo | 200 | { | Ritorna il log di qualità eliminato per |

| | | | |
|---|---|---|---|
| | | "c_projectphase_id": 0,<br>"qualitystatus": "string",<br>"qualityvalue": "string",<br>"c_projectphase_quality_log_id": 0,<br>"isactive": "Y"<br>} | dare riscontro dell'eliminazione |
| Negativo | 401 | | L'utente non è autorizzato |
| Negativo | 500 | {<br>"code": "string",<br>"description": "string"<br>} | Ritorna un JSON di errore avente codice e descrizione |

### QualitySaveLog - Fetch

**Descrizione:** questo endpoint permette di ottenere tutti i log relativi a una fase di produzione.

- **Endpoint:** /9000004/qualitysavelog/fetch
- **Metodo HTTP:** POST
- **Header**: token di autenticazione
- **Corpo richiesta:**
  - *startRow*: riga di partenza da cui si inizia ad acquisire i log dei controlli di qualità, tra le righe ottenute dal filtraggio (indicato dall'array *criteria*);
  - *criteria*: array di criteri di filtraggio, aventi:
    - *fieldname*: nome della colonna per la quale filtrare;
    - *value*: valore di confronto per il filtraggio;
    - *operator*: operatore di confronto da applicare.
  - *orderby*: array di criteri di ordinamento, aventi:
    - *columnname*: nome della colonna per la quale ordinare;
    - *direction*: direzione dell'ordinamento.
  - endRow: riga da cui si finisce di acquisire dati.

| Esito | Codice HTTP | Body | Descrizione |
|---|---|---|---|
| Positivo | 200 | {<br>"c_projectphase_id": 0,<br>"qualitystatus": "string",<br>"qualityvalue": "string",<br>"c_projectphase_quality_log_id": 0,<br>"isactive": "Y"<br>} | Ritorna il log dei dati di qualità inseriti per la fase richiesta |
| Negativo | 401 | | L'utente non è autorizzato |
| Negativo | 500 | {<br>"code": "string",<br>"description": "string"<br>} | Ritorna un JSON di errore avente codice e descrizione |

### QualitySaveLog - Update

**Descrizione:** questo endpoint permette di aggiornare un log contenente i dati di un controllo qualità.

- **Endpoint:** /9000004/qualitysavelog/update
- **Metodo HTTP:** POST
- **Header**: token di autenticazione
- **Corpo richiesta:**
    - *c_projectphase_id*: identificativo della fase della quale si stanno modificando i valori per il controllo qualità;
    - *qualitystatus*: stato del log di qualità;
    - *qualityvalue*: valore del log di qualità, è una stringa in formato JSON;
    - *c_projectphase_quality_log_id*: identificativo del log di qualità;
    - isactive: indica se il log di qualità è attivo o meno.

| Esito | Codice HTTP | Body | Descrizione |
|---|---|---|---|
| Positivo | 200 | {<br>"c_projectphase_id": 0,<br>"qualitystatus": "string",<br>"qualityvalue": "string",<br>"c_projectphase_quality_log_id": 0,<br>"isactive": "Y"<br>} | Ritorna il corpo della richiesta per dare riscontro dei valori aggiornati |
| Negativo | 401 | | L'utente non è autorizzato |
| Negativo | 500 | {<br>"code": "string",<br>"description": "string"<br>} | Ritorna un JSON di errore avente codice e descrizione. |

# Mockup

Data la natura del progetto (prevalentemente incentrato su frontend  e middleware) è fondamentale eseguire delle scelte di progettazione (design) per quanto riguarda l'interfaccia grafica: tali scelte sono riportate di seguito mediante degli screenshot relativi alle diverse viste che si possono avere utilizzando l'applicazione.

## Login

Il login utente avviene attraverso due passaggi:

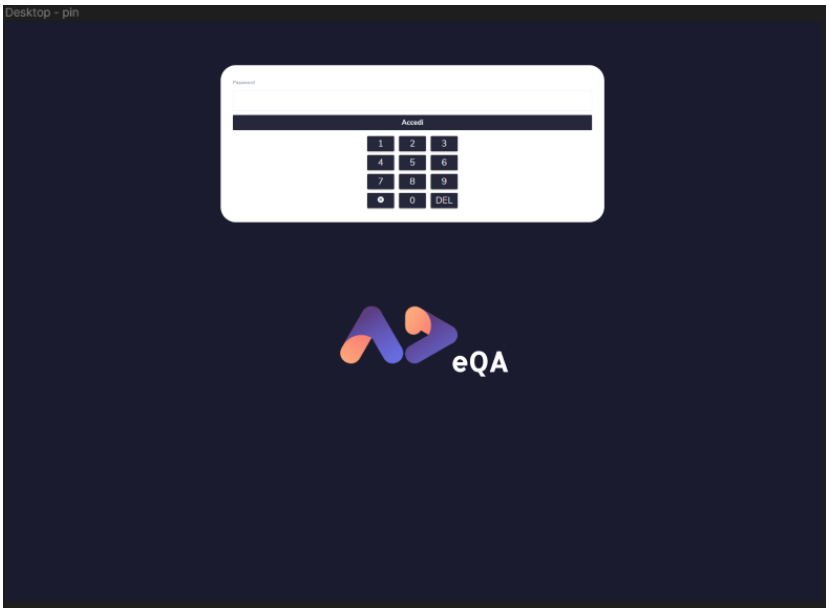
1. Accesso tramite nome utente e password;
2. Accesso tramite PIN.

Occorre aver eseguito con successo l'accesso tramite credenziali standard (nome utente e password) per poter accedere alla schermata di accesso tramite PIN.

### Login – passo 1

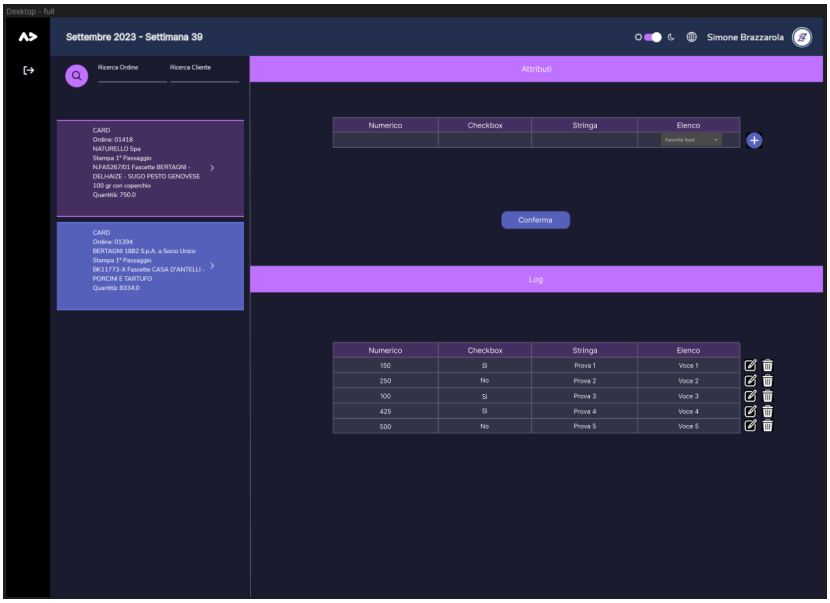

L'interfaccia per il primo passo di accesso è minimale, essendo costituita dal logo dell'applicazione ed un form contenente due caselle di testo ed un bottone.

## Login – passo 2




L'interfaccia per il secondo passo di accesso è molto simile all'interfaccia per il primo passo, con l'unica differenza di avere una "pulsantiera" per l'inserimento del pin; tale scelta è stata effettuata per compatibilità con l'interfaccia grafica del software ADeMES, nel quale ADeQA dovrà integrarsi.

## Visualizzazione principale




Questa pagina rappresenta l'applicazione dopo il login ed è contraddistinta dai seguenti elementi:

- **Menù centrale**: contiene l'indicazione della data attuale (mese, anno e settimana corrente), un widget per cambiare il tema dell'applicazione (chiaro / scuro), un widget per cambiare lingua all'applicazione, il nome utente dell'utente che ha fatto l'accesso ed il logo aziendale;
- **Menù laterale**: contiene il logo della suite di prodotti aziendali (AdeSuite) ed un pulsante per eseguire il logout;
- **Barra laterale**: contiene una lista delle fasi di lavorazione attive per l'organizzazione di cui fa parte l'utente che ha fatto l'accesso all'applicazione (con scorrimento verticale in caso eccedessero lo spazio verticale a disposizione); in alto si trova uno strumento di ricerca/filtraggio atto a ridurre il numero di fasi visualizzate;
- **Frame principale**:
  - *Attributi*: questa sezione di schermo consente l'aggiunta e la modifica di informazioni relative al controllo qualità tramite una tabella avente numero di colonne variabile (in base al numero degli attributi);
  - *Log*: questa sezione di schermo consente la visualizzazione delle informazioni di controllo qualità già inserite per la fase selezionata (oltre alla possibilità di eliminare e indicare la volontà di modificare tali informazioni).

## Visualizzazione responsive

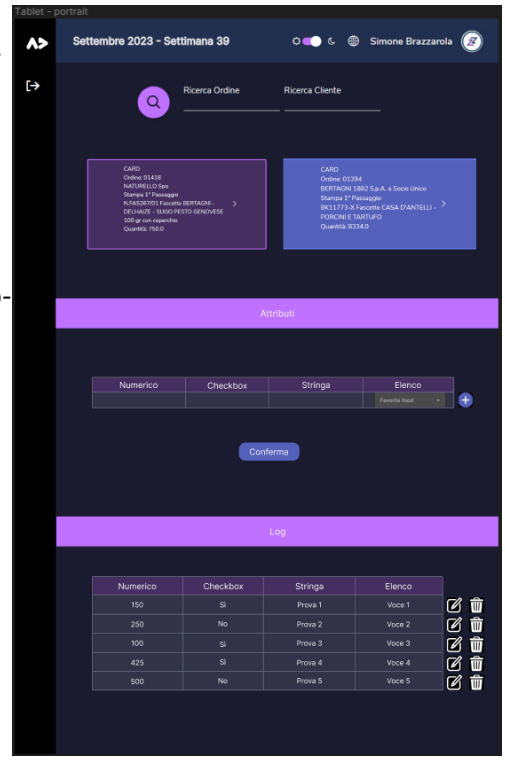
Tra i requisiti obbligatori del prodotto vi è la responsività dell'applicazione, ovvero la capacità del software di poter essere utilizzabile da dispositivi aventi dimensioni e risoluzioni eterogenee.

## Visualizzazione tablet - 1

La visualizzazione per tablet in modalità ritratto (avente la dimensione più piccola sul lato superiore) prevede che la barra laterale venga portata superiormente rispetto alle sezioni “Attributi” e “Log”.

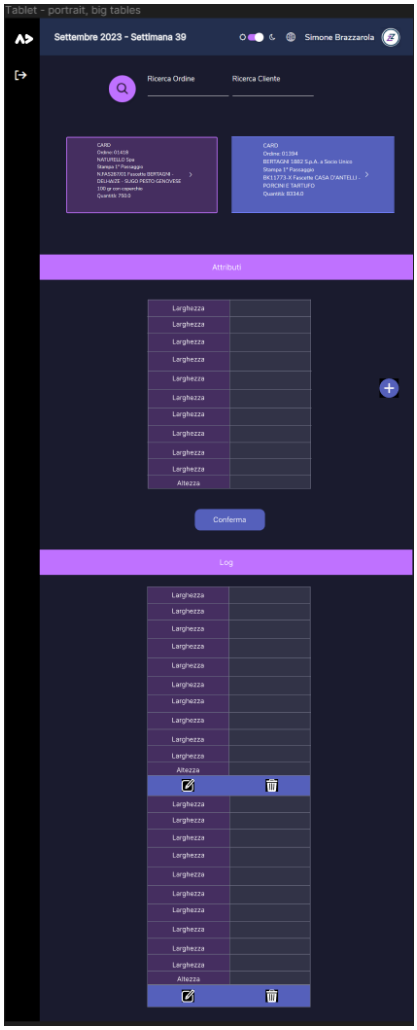
Le fasi vengono visualizzate sempre in riquadri colorati ma il loro scorrimento diventa orizzontale.

Le sezioni “Attributi” e “Log” rimangono nella posizione originale, potendo usufruire così di tutta la larghezza dello schermo.



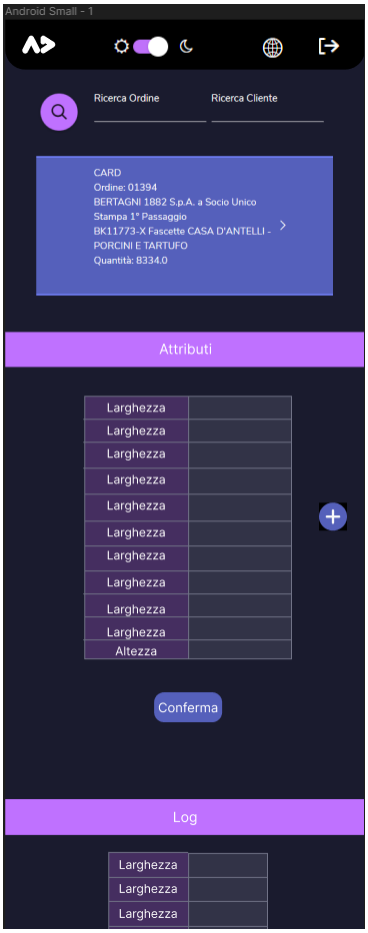

## Visualizzazione tablet - 2

La visualizzazione per tablet deve essere in grado di gestire le tabelle delle sezioni “Attributi” e “Log” indipendentemente dal numero di attributi di qualità: se la larghezza di tali tabelle dovessero eccedere lo spazio orizzontale a disposizione, le intestazioni delle colonne diventerebbero le intestazioni delle righe creando una tabella di sole due colonne come in figura.

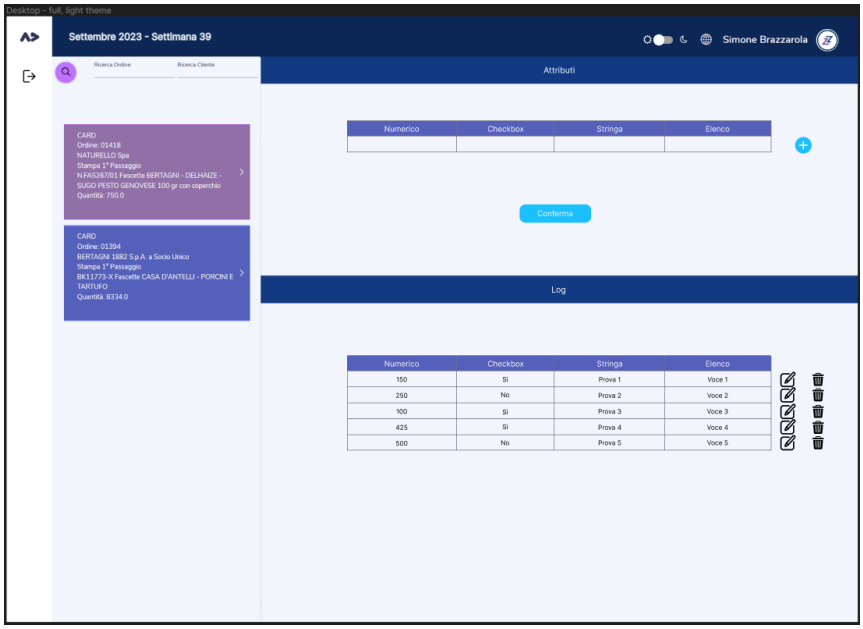


## Visualizzazione smartphone

La visualizzazione per smartphone deve essere in grado di gestire anche la mancanza di spazio per il menù di intestazione: in questo caso, esso si fonde con la barra laterale rimuovendo il nome dell'utente, il logo aziendale e l'indicazione della data.



### Visualizzazione con tema chiaro




Per garantire compatibilità rispetto all'interfaccia grafica dell'applicazione contenitrice ADeMES, occorre fornire la possibilità di cambiare tema alla visualizzazione.

# Architettura

## Componenti

Un componente Angular è un insieme di 4 files che descrivono un elemento HTML personalizzato al quale sono associati stile ed eventi: gli eventi sono contenuti all'interno di una classe in un file TypeScript e possono essere richiamati direttamente dal template HTML.

Si è deciso di usare fogli di stile in formato SCSS per poter assegnare temi e colori ai componenti del framework Angular Material come da manuale (Theming Angular Material | Angular Material ).

### Login – passo 1

La schermata di login iniziale consente l'autenticazione tramite la coppia di stringhe "nome utente" e "password": questa serve per l'ottenimento di un token (codice alfanumerico) per poter garantire l'identità di chi effettua una richiesta al server.

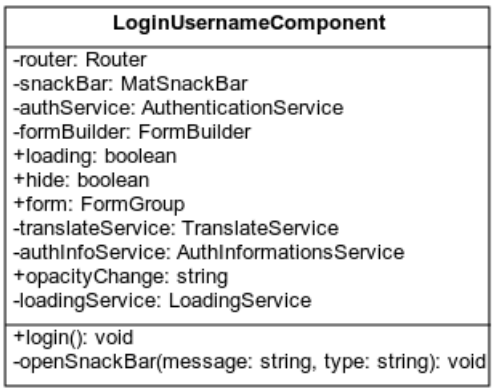
Questo passo di login viene eseguito solamente alla prima installazione del prodotto presso le infrastrutture di aziende clienti: serve per autenticare l'azienda (o la filiale) e non verrà più effettuato fino alla scadenza del token (la scadenza non è determinata a priori, può dipendere anche da fattori esterni quale può essere, per esempio, un guasto all'infrastruttura).

- **Percorso**: */login/username*
- **Elementi**
  - Logo del prodotto;
  - Riquadro che delimita l'area del form di inserimento dati;
  - Casella di testo per inserire il nome utente;
  - Casella di testo per inserire la password;

        - Pulsante per nascondere / mostrare la password;
    - Pulsante per confermare i dati inseriti e avviare la procedura di autenticazione;
    - Barra di notifica in caso di errori.

- **Requisiti soddisfatti**
    - **RF-1**: l'utente deve riuscire ad inserire i propri dati per effettuare il login;
    - **RF-1.1**: l'utente deve riuscire ad inserire la coppia "nome utente, password" per effettuare il login;
    - **RF-E1**: l'utente deve poter visualizzare un messaggio di errore in caso le credenziali inserite all'autenticazione siano errate;
    - **RF-E1.1**: l'utente deve poter visualizzare un errore in caso la coppia "nome utente, password" inserita all'autenticazione sia errata;
    - **RF-E2**: l'utente deve poter visualizzare un messaggio di errore in caso avvenga un errore lato server all'autenticazione.

- **Architettura**



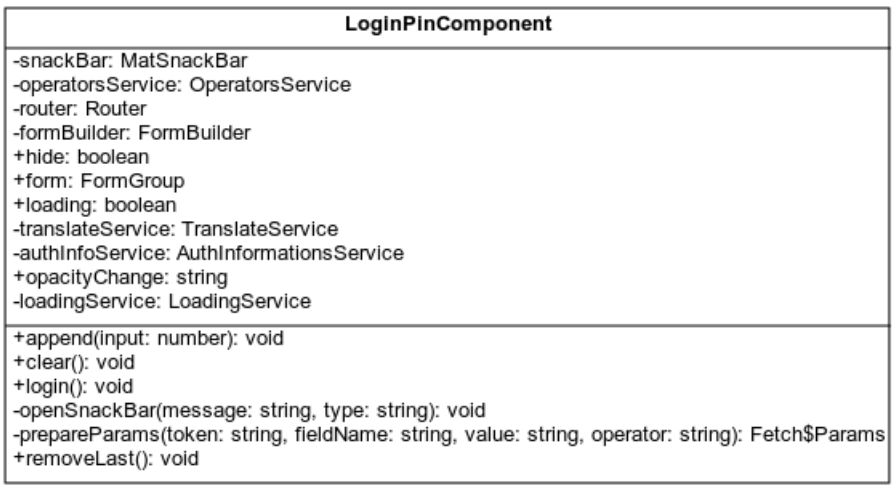

    - **LoginUsernameComponent**: classe che definisce il comportamento della pagina del primo login utente (tramite nome utente e password).
        - **Attributi:**
            - *router:* servizio utile per eseguire dei reindirizzamenti su browser;
            - *snackbar*: barra di visualizzazione di messaggi di stato (ad esempio, l'errore generato al fallimento del login);
            - *authService*: servizio che gestisce l'autenticazione, si occupa di richiamare il server per ottenere il token di accesso;
            - *formBuilder*: attributo atto a costruire il form di login;
            - *loading*: attributo booleano che indica se la richiesta HTTP è ancora in attesa di risposta;
            - *hide*: attributo booleano per cambiare l'icona che nasconde il contenuto del campo "password";
            - *form*: attributo che consente di accedere ai valori inseriti dall'utente nel form di login;
            - *translateService*: servizio di gestione delle traduzioni;
            - *authInfoService*: servizio di gestione delle informazioni di login;
            - *opacityChange*: attributo di gestione del cambiamento di opacità all'avvio;
            - *loadingService*: servizio di gestione del caricamento dei widget della visualizzazione principale.

- **Metodi**
  - *openSnackBar(message: string, type: string)*: metodo per gestire l'apertura della barra di stato;
  - *login()*: metodo per eseguire il login con nome utente e password.

### Login – passo 2

La schermata di login tramite codice numerico a 4 cifre consente di ottenere informazioni (quali nome e tema dell'applicazione predefinito) sull'operatore che vuole accedere alle funzionalità offerte.

- **Percorso**: /login/pin
- **Elementi**:
  - Logo del prodotto;
  - Riquadro che delimita l'area del form di inserimento dati;
  - Casella di testo per visualizzare il codice inserito
    - Pulsante per nascondere / mostrare il codice;
  - Pulsanti per inserire il codice numerico;
  - Pulsante per confermare i dati inseriti e avviare la procedura di autenticazione;
  - Barra di notifica in caso di errori.
- **Requisiti soddisfatti**
  - **RF-1**: l'utente deve riuscire ad inserire i propri dati per effettuare il login;
  - **RF-1.2**: l'utente deve riuscire ad inserire il pin per effettuare il login;
  - **RF-E1**: l'utente deve poter visualizzare un messaggio di errore in caso le credenziali inserite all'autenticazione siano errate;
  - **RF-E1.2**: l'utente deve poter visualizzare un errore in caso il pin inserito all'autenticazione sia errato;
  - **RF-E2**: l'utente deve poter visualizzare un messaggio di errore in caso avvenga un errore lato server all'autenticazione.
- **Architettura**



- **LoginPinComponent**: classe che definisce il comportamento della pagina del secondo login utente (tramite pin).
  - **Attributi**
    - *snackbar*: barra di visualizzazione di messaggi di stato (ad esempio, l'errore generato al fallimento del login);
    - *operatorsService*: servizio per ottenere gli operatori attivi;
    - *router:* servizio utile per eseguire dei reindirizzamenti su browser;
    - *formBuilder*: attributo per costruire il form di login;
    - *hide*: attributo booleano per cambiare l'icona che nasconde il contenuto del campo "pin";
    - *form*: attributo che consente di accedere ai valori inseriti dall'utente nel form di login;
    - *loading*: attributo booleano che indica se la richiesta HTTP è ancora in attesa di risposta;
    - *translateService*: servizio di gestione delle traduzioni;
    - *authInfoService*: servizio di gestione delle informazioni di login;
    - *opacityChange*: attributo di gestione del cambiamento di opacità all'avvio;
    - *loadingService*: servizio di gestione del caricamento dei widget della visualizzazione principale.

  - **Metodi**
    - *append(input: number)*: metodo per inserire un numero nella casella di input per il pin;
    - *clear()*: metodo per rimuovere totalmente il pin dalla relativa casella di input;
    - *login()*: metodo per eseguire il login tramite pin;
    - *openSnackBar(message: string, type: string)*: metodo per gestire l'apertura della barra di stato;
    - *prepareParams(token: string, fieldName: string, value: string, operator: string)*: metodo che prepara i parametri per il login tramite pin;
    - *removeLast()*: metodo per cancellare l'ultima cifra inserita nella casella di input per il pin.

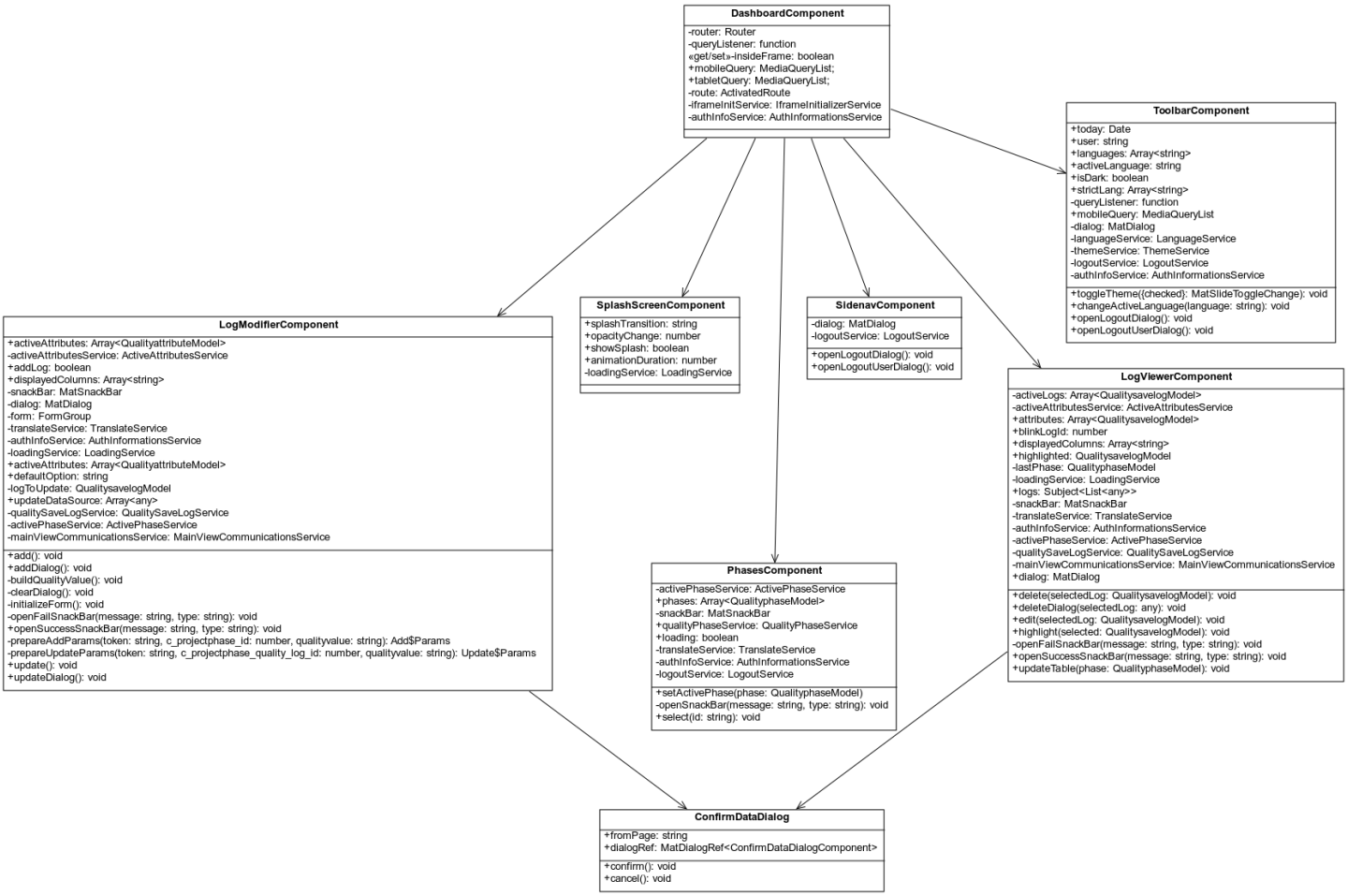
### Dashboard

La schermata di dashboard racchiude le funzionalità fondamentali del prodotto: visualizzazione fasi, attributi e log di qualità.

- **Percorso**: /dashboard
- **Elementi**:
  - Barra laterale;
  - Barra di intestazione;

- Widget di visualizzazione  delle fasi di qualità;
- Widget di visualizzazione dei log di qualità;
- Widget di inserimento / modifica dei log di qualità;
- Barra di notifica in caso di errori.

- **Architettura**



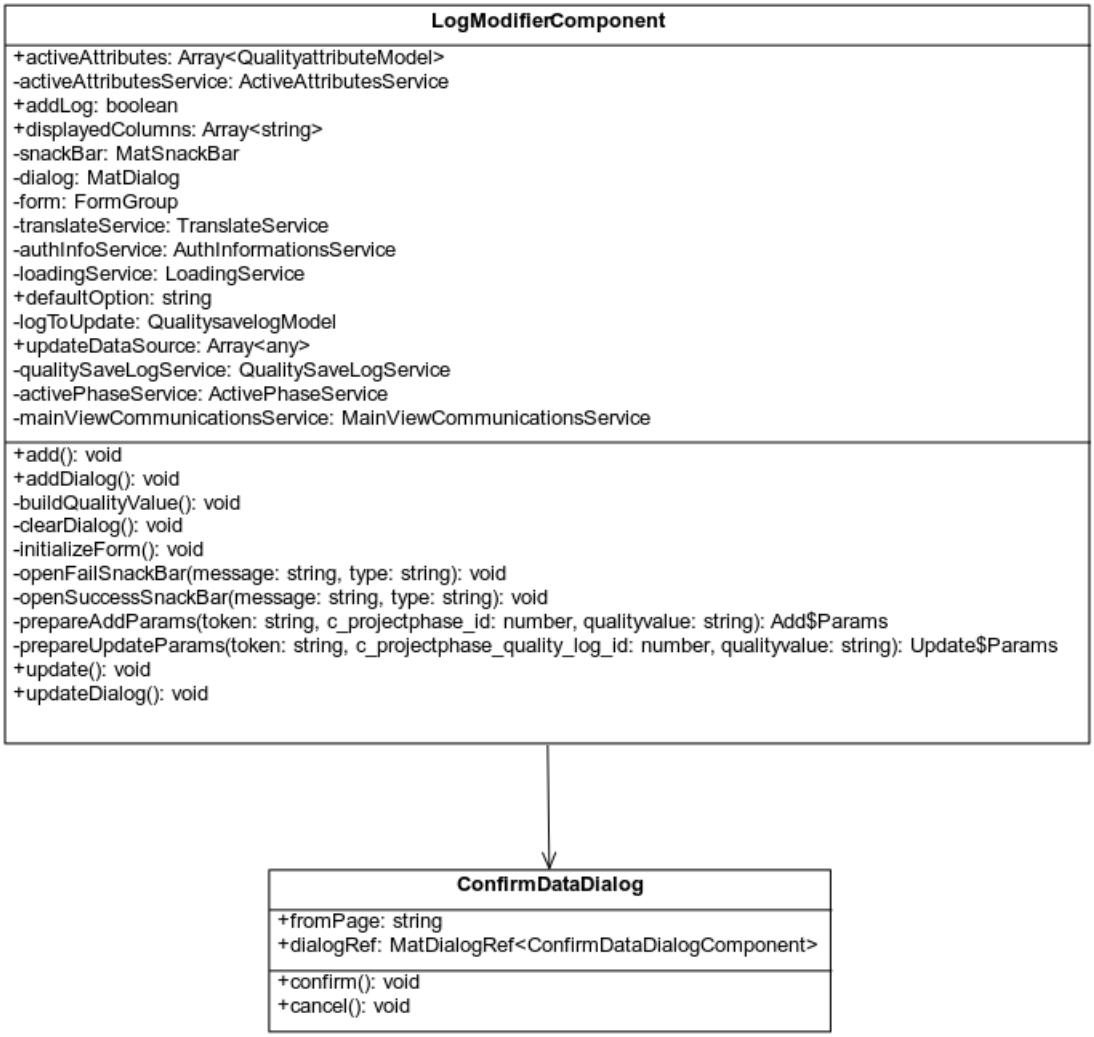

- **DashboardComponent**: classe che definisce la logica della visualizzazione principale.
  - **Attributi**
    - *router:* servizio utile per eseguire dei reindirizzamenti su browser;
    - *queryListener*: callback che consente di impostare lo stile del widget delle fasi in modo da gestire la visualizzazione su schermi di dimensioni ridotte;
    - *insideFrame*: indica se l'applicazione è in esecuzione all'interno di un <iframe>, serve per gestire questa casistica (e visualizzare il minimo numero di componenti);
    - *mobileQuery*: query che stabilisce il limite oltre il quale si deve adattare l'interfaccia per dispositivi mobili;

- *tabletQuery*: query che stabilisce il limite oltre il quale si deve adattare l'interfaccia per dispositivi con area di schermo simile a quella di un tablet;
- *route*: indica l'URL utilizzato per accedere a questa pagina, serve per capire se l'applicazione sta eseguendo all'interno di un <iframe> (l'applicazione contenitrice usa il parametro 'inside=true' all'interno dell'URL di chiamata dell'applicazione, vedi sezione *Integrazione all'interno di un frame HTML* );
- *iframeInitService*: servizio di inizializzazione dei servizi minimi per far eseguire l'applicazione in un <iframe>;
- *authInfoService*: servizio di gestione delle informazioni di login.

### Modificatore dei log

Il widget di modifica dei log consente di inserire ed aggiornare i log di qualità relativi alla fase di produzione attiva.

- **Elementi**
  - Tabella per inserire / aggiornare i dati dei log di qualità: questa tabella non ha una struttura fissa in quanto le sue colonne sono gli attributi della fase attiva (che può cambiare durante l'esecuzione dell'applicativo);
  - Pulsante di conferma dei dati inseriti;
  - Helper associato al pulsante quando esso è disattivato (quando non tutti gli attributi sono stati valorizzati);
  - Barra di notifica in caso di errori.
- **Requisiti soddisfatti**
  - **RF-4**: l'utente deve poter visualizzare gli attributi della fase di produzione selezionata;
  - **RF-5**: l'utente deve poter assegnare un valore agli attributi della fase di produzione selezionata;
  - **RF-5.1**: l'utente deve poter visualizzare un messaggio positivo in caso l'assegnazione di valori agli attributi avvenga correttamente;
  - **RF-8**: l'utente deve poter modificare i log di controllo qualità relativi alla fase selezionata;
  - **RF-E4**: l'utente deve poter visualizzare un messaggio di errore in caso vi sia un errore lato backend all'acquisizione degli attributi della fase di lavorazione selezionata;
  - **RF-E5**: l'utente deve poter visualizzare un messaggio di errore in caso vi sia un errore lato backend all'inserimento dei valori per gli attributi della fase di lavorazione selezionata;
  - **RF-E8**: l'utente deve poter visualizzare un messaggio di errore in caso vi sia un errore lato backend all'aggiornamento dei log di controllo qualità della fase di lavorazione selezionata.
- **Architettura**




- **LogModifierComponent**

  - **Attributi**
    - *activeAttributes*: attributi di qualità per la fase attiva;
    - *activeAttributesService*: servizio utile all'ottenimento automatico (tramite flussi di dati asincroni) degli attributi di qualità per la fase attiva;
    - *addLog*: attributo booleano che indica se il form da visualizzare deve gestire l'aggiunta o la modifica di un log;
    - *displayedColumns*: nomi degli attributi, sono le intestazioni delle colonne della tabella;
    - *snackbar*: barra di visualizzazione di messaggi di stato (ad esempio, l'errore generato al fallimento del login);
    - *dialog*: dialog di conferma dei dati inseriti;
    - *form*: attributo che consente di accedere ai valori inseriti dall'utente nel form di aggiunta / modifica dei log;
    - *translateService*: servizio di gestione delle traduzioni;
    - *authInfoService*: servizio di gestione delle informazioni di login;
    - *loadingService*: servizio di gestione del caricamento dei widget della visualizzazione principale;
    - *defaultOption*: opzioni attive per gli attributi di tipo "lista" per il log da modificare;
    - *logToUpdate*: log da aggiornare in un dato istante;

- *updateDataSource*: riferimento ai dati da modificare;
- *qualitySaveLogService*: servizio per eseguire operazioni CRUD su log di qualità;
- *activePhaseService*: servizio di gestione della fase attiva;
- *mainViewCommunicationsService*: servizio che consente la comunicazione tra *LogViewerComponent e LogModifierComponent*.

▪ **Metodi**

- *add()*: metodo per aggiungere un log di qualità per la fase attiva;
- *addDialog()*: metodo per aprire il dialog di conferma per aggiungere un nuovo log di qualità per la fase attiva;
- *buildQualityValue()*: metodo per costruire una stringa in formato JSON che deve essere usata come parametro "*qualityvalue*" all'inserimento / modifica di un log di qualità;
- *clearDialog()*: metodo per riportare il dialog di inserimento / modifica di un log di qualità allo stato iniziale;
- *initializeForm()*: metodo per inizializzare il form di inserimento / aggiornamento di un log per la fase attiva;
- *openFailSnackBar(message: string, type: string)*: metodo per gestire l'apertura della barra di stato dopo un fallimento;
- *openSuccessSnackBar(message: string, type: string)*: metodo per gestire l'apertura della barra di stato dopo un successo;
- *prepareAddParams(token: string, c_projectphase_id: number, qualityvalue: string):* metodo per costruire i parametri per eseguire una richiesta HTTP che aggiunga un log di qualità;
- *prepareUpdateParams(token: string, c_projectphase_quality_log_id: number, qualityvalue: string)*: metodo per costruire i parametri per eseguire una richiesta HTTP che modifichi un log di qualità;
- *update()*: metodo per modificare un log di qualità per la fase attiva;
- *updateDialog()*: metodo per aprire il dialog di conferma per modificare un log di qualità per la fase attiva.

◦ **ConfirmDataDialog**

▪ **Attributi**

- *fromPage*: indica il testo da inserire nel dialog di conferma dei dati (per inserimento / modifica log di qualità);
- *dialogRef*: riferimento al dialog stesso.

▪ **Metodi**

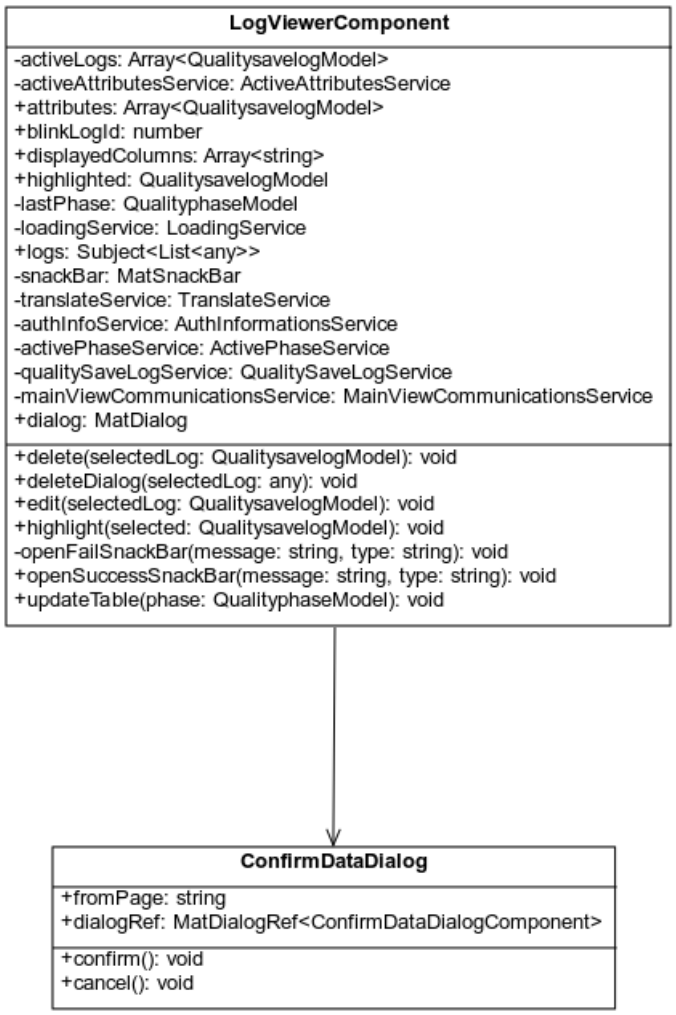
- *confirm()*: metodo per confermare che i dati inseriti (e visualizzati nel dialog) sono corretti;
- *cancel()*: metodo per indicare di uscire dal dialog senza aggiungere / modificare il log attualmente evidenziato.

## Visualizzatore dei log

Il widget di visualizzazione dei log consente di visualizzare, eliminare e far partire la procedura di modifica dei log registrati per la fase attiva.

- **Elementi**
    - Tabella per visualizzare i dati dei log di qualità: questa tabella non ha una struttura fissa in quanto le sue colonne sono gli attributi della fase attiva (che può cambiare durante l'esecuzione dell'applicativo);
    - Pulsante di modifica di un log di qualità;
    - Pulsante di eliminazione di un log di qualità;
    - Barra di notifica in caso di errori.
- **Requisiti soddisfatti**
    - **RF-4**: l'utente deve poter visualizzare gli attributi della fase di produzione selezionata;
    - **RF-7**: l'utente deve poter visualizzare i log di controllo qualità relativi alla fase selezionata;
    - **RF-8**: l'utente deve poter modificare i log di controllo qualità relativi alla fase selezionata;
    - **RF-9**: l'utente deve poter eliminare i log di controllo qualità relativi alla fase selezionata;
    - **RF-E4**: l'utente deve poter visualizzare un messaggio di errore in caso vi sia un errore lato backend all'acquisizione degli attributi della fase di lavorazione selezionata;
    - **RF-E7**: l'utente deve poter visualizzare un messaggio di errore in caso vi sia un errore lato backend all'acquisizione dei log di controllo qualità della fase di lavorazione selezionata;
    - **RF-E8**: l'utente deve poter visualizzare un messaggio di errore in caso vi sia un errore lato backend all'aggiornamento dei log di controllo qualità della fase di lavorazione selezionata;
    - **RF-E9**: l'utente deve poter visualizzare un messaggio di errore in caso vi sia un errore lato backend all'eliminazione dei log di controllo qualità della fase di lavorazione selezionata.

- **Architettura**




- **LogViewerComponent**

  - **Attributi**
    - *activeLogs*: log di qualità salvati per la fase di produzione attiva;
    - *activeAttributesService*: servizio di gestione degli attributi attivi;
    - *attributes*: attributi attivi per la fase di produzione attiva;
    - *blinkLogId*: identificativo di un log di qualità appena inserito / modificato (deve "lampeggiare" e questo attributo consente di ottenere questo effetto grafico);
    - *displayedColumns*: colonne che devono essere mostrate nella tabella dei log;
    - *highlighted*: log di qualità selezionato per l'aggiornamento, è un attributo che serve per ottenere questo effetto a livello grafico;
    - *lastPhase*: ultima fase selezionata, questo attributo si rende necessario alla costruzione dei parametri per l'eliminazione di un log di qualità;
    - *loadingService*: servizio di gestione del caricamento dei widget della visualizzazione principale;
    - *logs*: attributo osservabile che emette i log di qualità da visualizzare nella relativa tabella;

- *snackbar*: barra di visualizzazione di messaggi di stato (ad esempio, l'errore generato al fallimento dell'eliminazione di un log);
- *translateService*: servizio di gestione delle traduzioni;
- *authInfoService*: servizio di gestione delle informazioni di login;
- *activePhaseService*: servizio per l'ottenimento automatico (tramite flussi di dati asincroni) della fase attiva;
- *qualitySaveLogService*: servizio per eseguire operazioni CRUD su log di qualità;
- *mainViewCommunicationsService*: servizio che consente la comunicazione tra *LogViewerComponent* e *LogModifierComponent*;
- *dialog*: dialog di conferma dei dati inseriti.

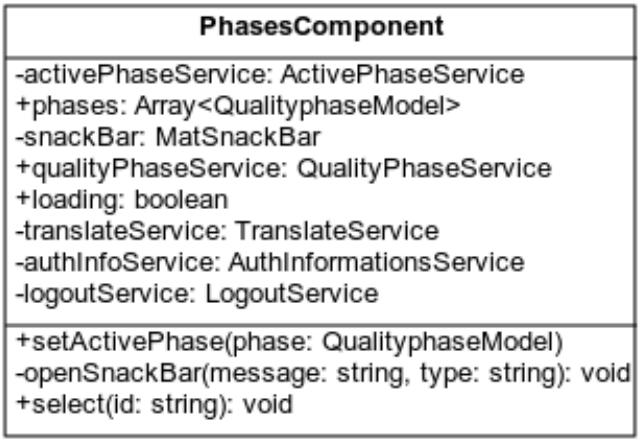
- **Metodi**
  - *delete(selectedLog: QualitysavelogModel)*: metodo per eliminare il log selezionato;
  - *deleteDialog(selectedLog: any)*: metodo per visualizzare un dialog di conferma per l'eliminazione del log selezionato;
  - *edit(selectedLog: QualitysavelogModel)*: metodo per modificare un log di qualità (inviare il suo stato al componente *LogModifier*);
  - *highlight(selected: QualitysavelogModel)*: metodo per evidenziare graficamente un log quando viene selezionato per essere modificato;
  - *openFailSnackBar(message: string, type: string)*: metodo per l'apertura della barra di visualizzazione di messaggi di stato in caso di fallimento;
  - *openSuccessSnackBar(message: string, type: string)*: metodo per l'apertura della barra di visualizzazione di messaggi di stato in caso di successo;
  - *updateTable(phase: QualityphaseModel)*: metodo per aggiornare la tabella di visualizzazione dei log.

### Visualizzatore delle fasi

Il widget di visualizzazione e selezione delle fasi di lavorazione.

- **Elementi**
  - Carte contenenti le fasi: sono dei riquadri colorati contenenti una breve descrizione delle singole fasi di lavorazione;
  - Widget di scorrimento delle fasi: è un widget invisibile (la barra di scorrimento è stata nascosta) che consente di scorrere verticalmente tutta la lista di fasi.
- **Requisiti soddisfatti**
  - **RF-2**: l'utente deve poter visualizzare le fasi di produzione della propria filiera;
  - **RF-E3**: l'utente deve poter visualizzare un messaggio di errore in caso vi sia un errore lato backend all'acquisizione delle fasi.

- **Architettura**




  - **PhasesComponent**
    - **Attributi**
      - *activePhaseService*: servizio per l'ottenimento automatico della fase attiva;
      - *phases*: fasi ottenute per l'operatore autenticato, aventi le seguenti caratteristiche:
        - "start_plan": valore minimo per la data di inizio fase;
        - "end_plan": valore massimo per la data di fine fase;
        - "status_value": valore di stato della fase;
      - *snackbar*: barra di visualizzazione di messaggi di stato (ad esempio, l'errore generato al fallimento dell'ottenimento delle fasi);
      - *qualityPhaseService*: servizio per l'ottenimento delle fasi di lavorazione;
      - *loading*: attributo booleano che indica se la richiesta HTTP è ancora in attesa di risposta;
      - *translateService*: servizio di gestione delle traduzioni;
      - authInfoService: servizio di gestione delle informazioni di login;
      - *logoutService*: servizio di gestione del logout.
    - **Metodi**
      - *setActivePhase(phase: QualityphaseModel)*: metodo per impostare la fase attiva;
      - *openSnackBar(message: string, type: string)*: metodo per gestire l'apertura della barra di stato;
      - *select(id: string)*: metodo per gestire la visualizzazione grafica della fase selezionata.

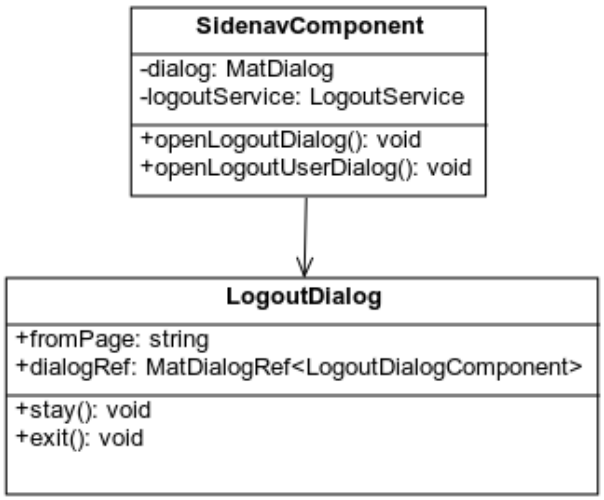
### Barra laterale

La barra laterale della visualizzazione principale, essa consente di eseguire i due tipi di logout.

- **Elementi**

- Logo dei prodotti aziendali;
- Bottone per il logout parziale (logout relativo all'operatore): si ritorna alla pagina di login con la stringa alfanumerica contenente il pin;
- Bottone per il logout totale: si ritorna alla pagina di login con la coppia di stringhe "nome utente, password".

- **Requisiti soddisfatti**
  - **RF-6**: l'utente deve potersi disconnettere, uscendo dalla propria area riservata;
  - **RF-6.1**: l'utente deve potersi disconnettere relativamente all'operatore a cui ha eseguito l'autenticazione;
  - **RF-6.2**: utente deve potersi disconnettere da ogni accesso eseguito: operatore e azienda;
  - **RF-E6**: l'utente deve poter visualizzare un messaggio di errore in caso vi sia un errore lato backend alla disconnessione.
- **Architettura**




  - **SidenavComponent**
    - **Attributi**
      - *dialog*: dialog di conferma dei dati inseriti;
      - *logoutService*: servizio di gestione del logout.
    - **Metodi**
      - *openLogoutDialog()*: metodo per l'apertura del dialog di logout generale;
      - *openLogoutUserDialog()*: metodo per l'apertura del dialog di logout parziale.
  - **LogoutDialog**
    - **Attributi**
      - *fromPage*: indica il testo da inserire nel dialog di conferma del logout

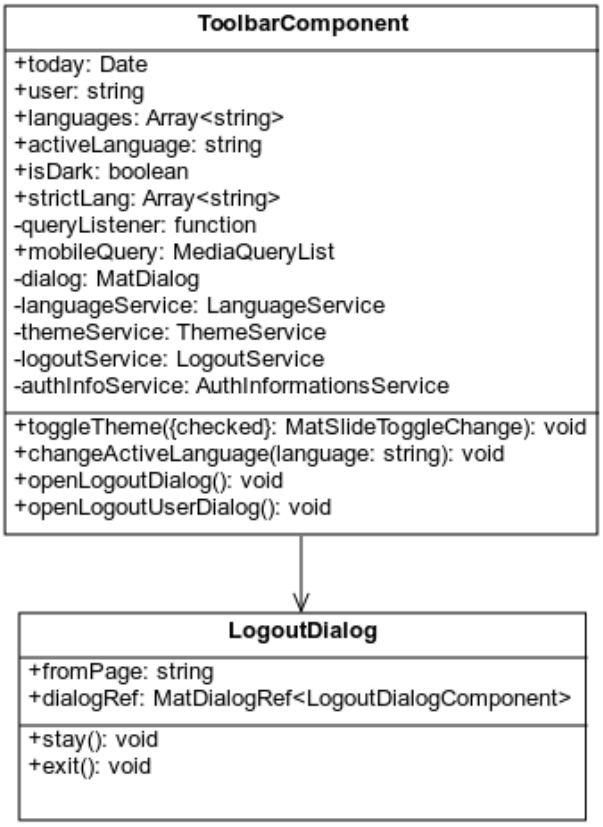
- *dialogRef*: riferimento al dialog stesso.

- **Metodi**
  - *confirm()*: metodo per confermare la volontà di eseguire il logout
  - cancel(): metodo per indicare di uscire dal dialog senza eseguire il logout.

### Barra di intestazione

La barra di intestazione della visualizzazione principale, gestisce le traduzioni, il cambio di tema e l'apertura del widget laterale che contiene le fasi.

- **Elementi**
  - Bottone di apertura / chiusura del widget contenente le fasi produttive;
  - Indicazione del mese, anno e settimana della data corrente;
  - Widget di selezione del tema grafico;
  - Bottone e menù per il cambio di linguaggio di traduzione;
  - Nome dell'operatore autenticato;
  - Logo aziendale.
- **Architettura**




  - **ToolbarComponent**

- **Attributi**
  - *today*: data odierna;
  - *user*: nome dell'operatore;
  - *languages*: linguaggi disponibili;
  - *activeLanguage*: linguaggio attivo;
  - *isDark*: attributo booleano che indica se c'è il tema scuro attivo;
  - *strictLang*: linguaggi disponibili, in codice ISO 639-1;
  - *queryListener*: callback che consente di impostare lo stile del widget stesso in modo da gestire la visualizzazione su schermi di dimensioni ridotte;
  - *mobileQuery*: query che stabilisce il limite oltre il quale si deve adattare l'interfaccia per dispositivi mobili;
  - *dialog*: dialog di logout;
  - *languageService*: servizio di gestione delle traduzioni supportate;
  - *themeService*: servizio di gestione del tema di interfaccia attivo;
  - *logoutService*: servizio di gestione del logout;
  - *authInfoService*: servizio di gestione delle informazioni di login.

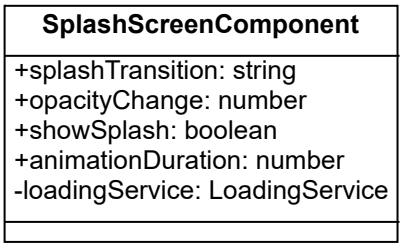
- **Metodi**
  - *toggleTheme({checked}: MatSlideToggleChange})*: metodo per cambiare il tema grafico dell'applicazione;
  - *changeActiveLanguage(language: string)*: metodo per cambiare il linguaggio di traduzione dell'applicazione;
  - *openLogoutDialog()*: metodo per aprire il dialog di logout generale;
  - *openLogoutUserDialog()*: metodo per aprire il dialog di logout parziale.

### Schermata di caricamento

La schermata di caricamento consente di nascondere l'interfaccia principale all'avvio: evita di far vedere all'utente il caricamento dei widget di visualizzazione e modifica dei log.

- **Elementi**
  - Logo aziendale di colore bianco;
  - Scritta con il nome del prodotto;
  - Widget di caricamento.

- **Architettura**




  - **SplashScreenComponent**

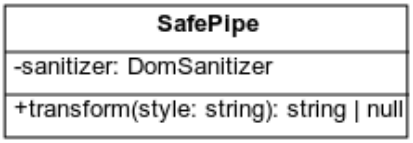
- **Attributi**
  - *splashTransition*: transizione con la quale la schermata viene nascosta (è stata scelta una dissolvenza in 0.4 secondi);
  - *opacityChange*: attributo che indica l'opacità della schermata in un dato momento;
  - *showSplash*: attributo booleano che indica se visualizzare (o meno) la schermata;
  - *animationDuration*: durata della transizione;
  - *loadingService*: servizio di gestione del caricamento dei widget della visualizzazione principale.

## Pipes

Una pipe è una classe TypeScript che implementa l'interfaccia *PipeTransform* (presente in ***@angular/core***): deve implementare il metodo *"transform"*, metodo che può avere parametri in numero e tipo indefinito e serve per eseguire la trasformazione degli input dati.

### SafePipe

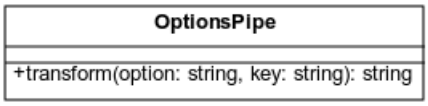
- **Architettura**




- **Attributi**
  - *sanitizer*: attributo usato per sanificare una stringa in input; nel prodotto, viene usato per sanificare HTML ritornato dal server per essere usato nelle carte.
- **Metodi**
  - *transform(style: string)*: metodo per trasformare l'input, sanificandolo.

### Options

- **Architettura**




- **Metodi**
  - transform(option: string, key: string): metodo per trasformare l'input, rimuovendo dal parametro '*option*' la stringa contenuta in *'key'*.

## Servizi

Un servizio è una classe TypeScript che può avere più di uno scopo:

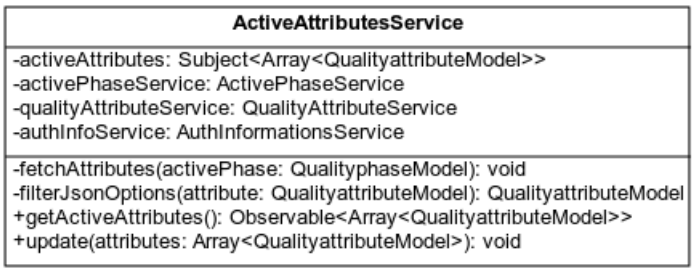
- Consentire la comunicazione tra componenti diversi;

- Eseguire operazioni di basso livello che non sono dipendenti da singoli widget (ad esempio, l'esecuzione di chiamate HTTP).

### ActiveAttributes

E' un servizio di gestione degli attributi attivi in un determinato momento dell'esecuzione del prodotto.

- **Architettura**

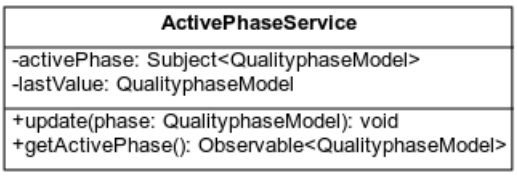



- **Attributi**
  - *activeAttributes*: attributi di qualità per la fase attiva;
  - *activePhaseService*: servizio per l'ottenimento automatico della fase attiva;
  - *qualityAttributeService*: servizio per l'ottenimento degli attributi di qualità;
  - *authInfoService*: servizio di gestione delle informazioni di login.
- **Metodi**
  - *fetchAttributes(activePhase: QualityphaseModel)*: metodo per acquisire gli attributi relativi alla fase attiva;
  - *filterJsonOptions(attribute: QualityattributeModel)*: metodo per ottenere le opzioni per un attributo di tipo "lista";
  - *getActiveAttributes()*: metodo per ottenere un oggetto osservabile degli attributi attivi (serve per consentire l'aggiornamento automatico delle tabelle contenute in *LogViewerComponent* e *LogModifierComponent*);
  - *update(attributes: Array<QualityattributeModel>)*: metodo per aggiornare gli attributi attivi.

### ActivePhase

E' un servizio di gestione della fase attiva in un determinato momento dell'esecuzione del prodotto.

- **Architettura**




- **Attributi**

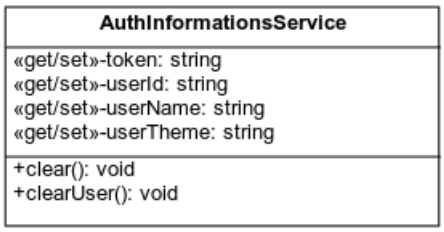
- *activePhase*: flusso di fasi di qualità attive;
- *lastValue*: ultima fase selezionata, serve per evitare di fare richieste HTTP per ottenere gli attributi per la fase già attiva.

- **Metodi**
  - *update(phase: <QualityphaseModel>)*: metodo per aggiornare la fase attiva;
  - *getActivePhase()*: metodo per ottenere un oggetto osservabile della fase attiva (serve per iniziare l'aggiornamento automatico delle tabelle contenute in *LogViewerComponent* e *LogModifierComponent*).

### AuthInformationService

E' un servizio di gestione e condivisione delle informazioni utente.

- **Architettura**

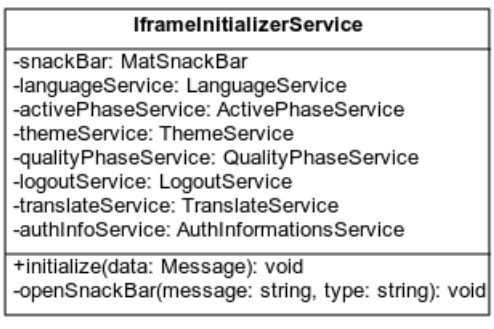



- **Attributi**
  - *token*: stringa alfanumerica per l'autenticazione utente durante l'utilizzo delle API;
  - *userId: identificativo dell'operatore;*
  - *userName*: nome dell'operatore, da visualizzare all'interno della barra di intestazione;
  - userTheme: tema predefinito per l'operatore, l'applicazione si avvierà usando questo tema.
- **Metodi**
  - *clear()*: metodo per cancellare tutte le informazioni relative all'autenticazione;
  - *clearUser()*: metodo per cancellare le informazioni relative al login tramite pin.

### IframeInitializerService

E' un servizio di gestione dell'inizializzazione dei servizi minimi dell'applicazione quando essa viene utilizzata all'interno di un <iframe>.

- **Architettura**



- **Attributi**
  - *snackbar*: barra di visualizzazione di messaggi di stato (ad esempio, l'errore generato all'acquisizione delle fasi);
  - *languageService*: servizio di gestione delle traduzioni supportate;
  - *activePhaseService*: servizio per l'ottenimento automatico della fase attiva;
  - *themeService*: servizio di gestione del tema di interfaccia attivo;
  - *qualityPhaseService*: servizio per l'ottenimento delle fasi di lavorazione;
  - *logoutService*: servizio di gestione del logout;
  - *translateService*: servizio per gestire le traduzioni;
  - *authInfoService*: servizio di gestione delle informazioni di login.

- **Metodi**
  - *initialize(data: Message)*: metodo che inizializza i servizi minimi per il funzionamento del prodotto se in esecuzione all'interno di un <iframe>;
  - *openSnackBar(message: string, type: string)*: metodo per gestire l'apertura della barra di stato.

## LanguageService

E' un servizio di gestione delle traduzioni e della fruizione del contenuto statico del prodotto.

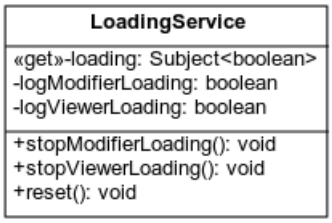
- **Architettura**


- **Attributi**
  - *translateService*: servizio di traduzione fornito da ***@ngx-translate/core***;
  - *activeLanguage*: linguaggio attivo.
- **Metodi**
  - *changeLanguage(language: string)*: metodo per cambiare il linguaggio attivo.

## LoadingService

E' un servizio di gestione del rendering grafico dei widget di visualizzazione principale.

- **Architettura**



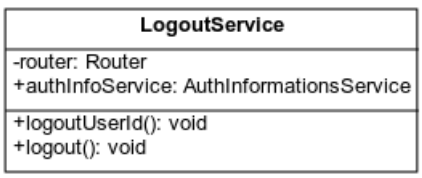

- **Attributi**
  - *loading*: attributo osservabile che emette valore **true** quando è stato completato il caricamento da parte di tutti i widget della visualizzazione primaria;
  - *logModifierLoading*: attributo booleano che indica se il componente addetto alla modifica dei log deve completare il rendering;
  - *logViewerLoading*: attributo booleano che indica se il componente addetto alla visualizzazione dei log deve completare il rendering.

- **Metodi**
  - *stopModifierLoading()*: metodo per cambiare valore all'attributo *logModifierLoading*, per segnalare il completamento del rendering;
  - *stopViewerLoading()*: metodo per cambiare l'attributo *logViewerLoading*, per segnalare il completamento del rendering;
  - *reset()*: metodo per riportare il servizio allo stato iniziale (le variabili booleane vengono impostate a **true**).

## LogoutService

E' il servizio di gestione dei logout.

- **Architettura**




- **Attributi**
  - *router:* servizio utile per eseguire dei reindirizzamenti su browser;
  - *authInfoService*: servizio di gestione delle informazioni di login.

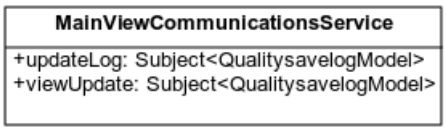
- **Metodi**

- *logoutUserId()*: metodo per eseguire il logout parziale (occorre ri-eseguire il login tramite pin);
- *logout()*: metodo per eseguire il logout totale (occorre ri-eseguire entrambi i passi di autenticazione).

### MainViewCommunicationsService

E' un servizio che consente la comunicazione tra *LogViewerComponent e LogModifierComponent*.

- **Architettura**

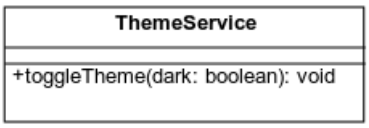



- **Attributi**
  - *updateLog*: oggetto osservabile che funge da tramite per il passaggio del log da modificare tra *LogViewerComponent* e *LogModifierComponent*;
  - *viewUpdate*: oggetto osservabile che indica a *LogViewerComponent* di fare un refresh della tabella dei log attivi.

### ThemeService

E' un servizio di gestione del tema attivo.

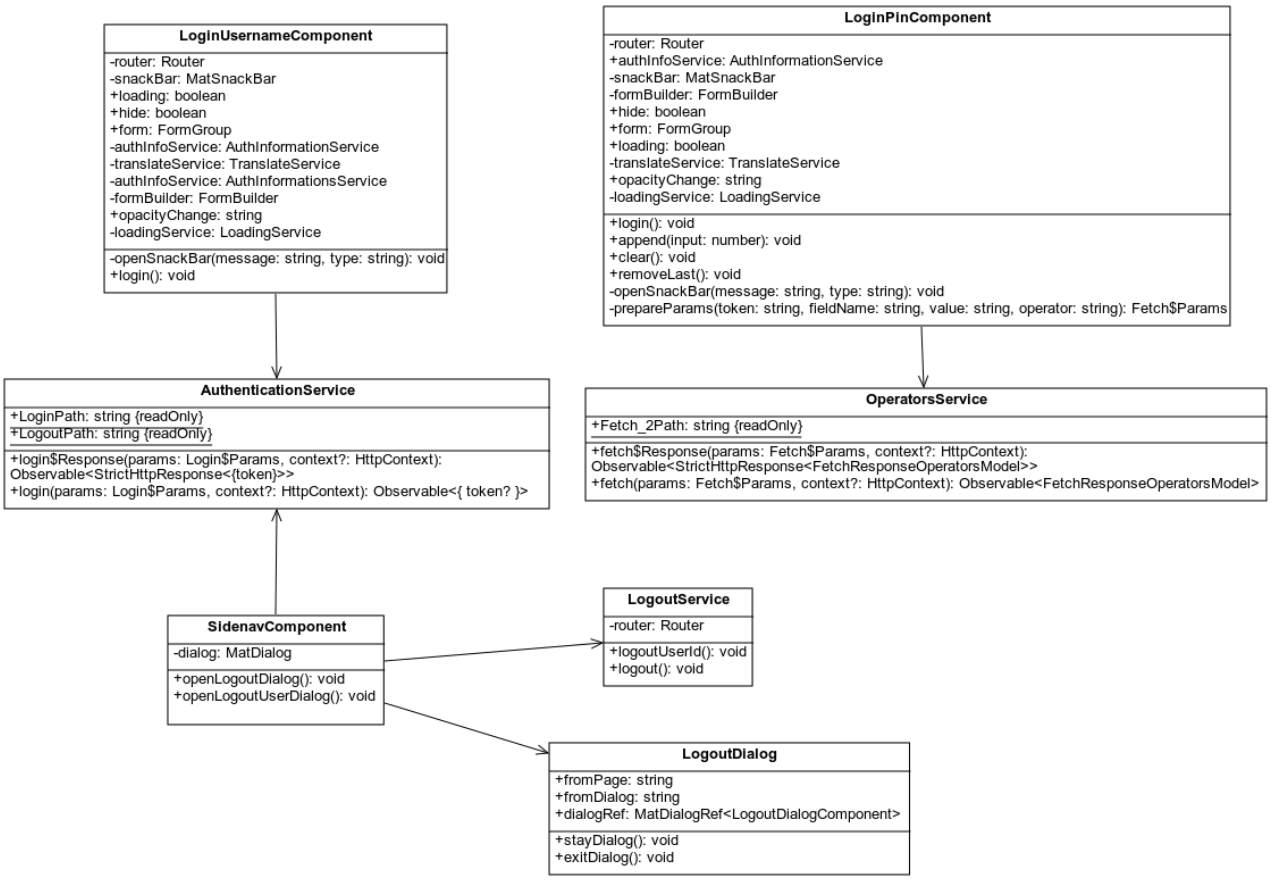
- **Architettura**



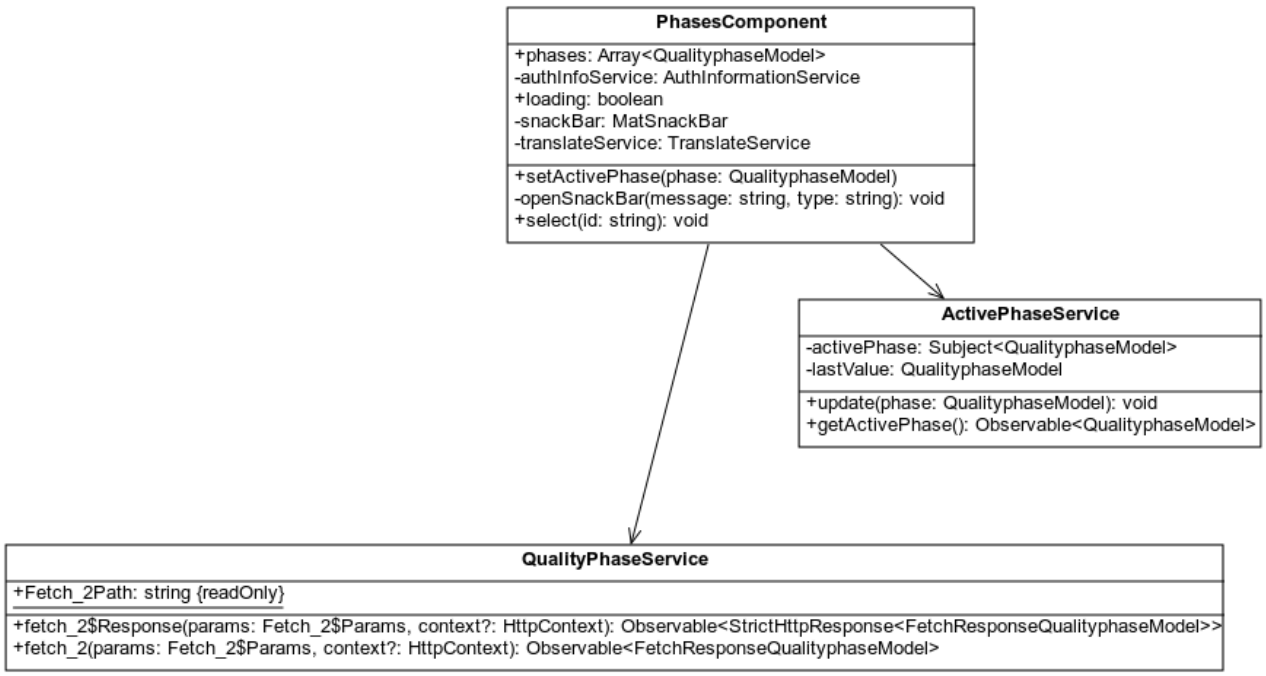

- **Metodi**
  - *toggleTheme(dark: boolean)*: metodo per cambiare il tema dell'applicazione.

# Componenti e collegamenti con servizi

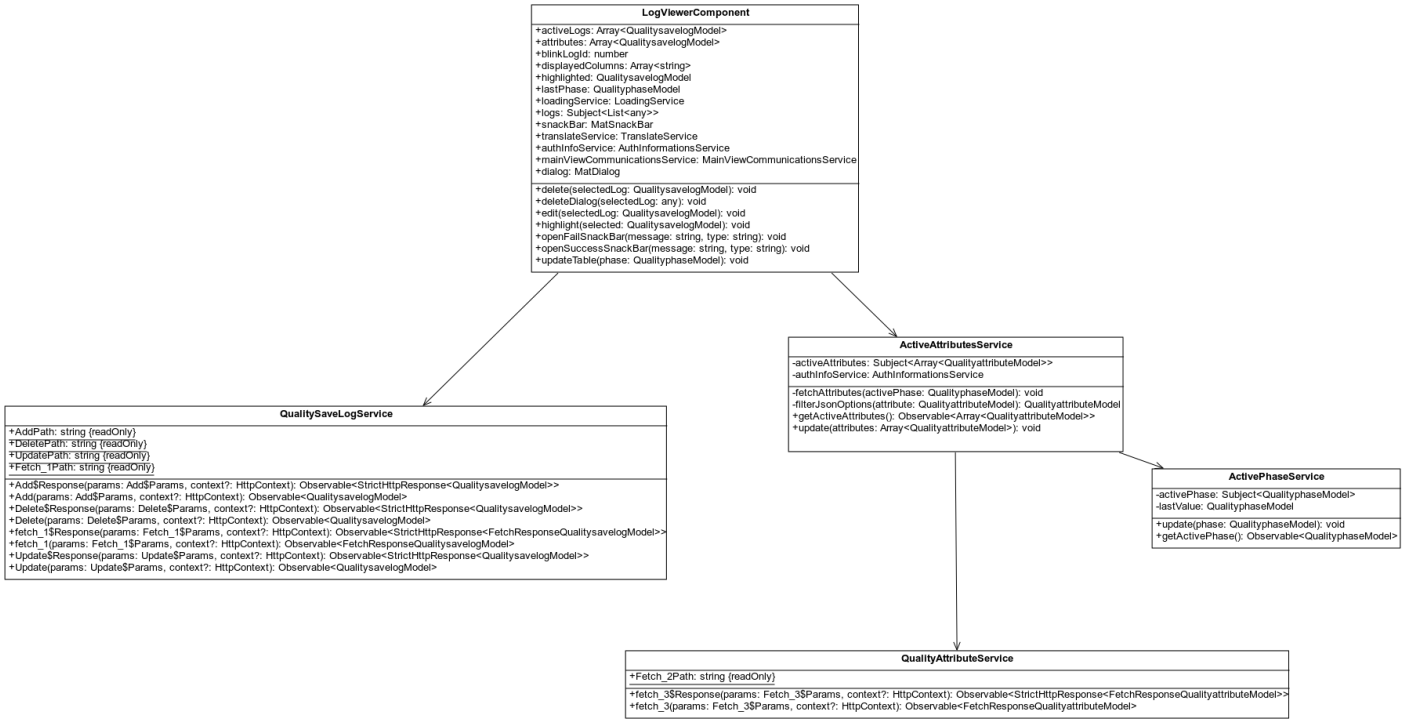
## Login/logout



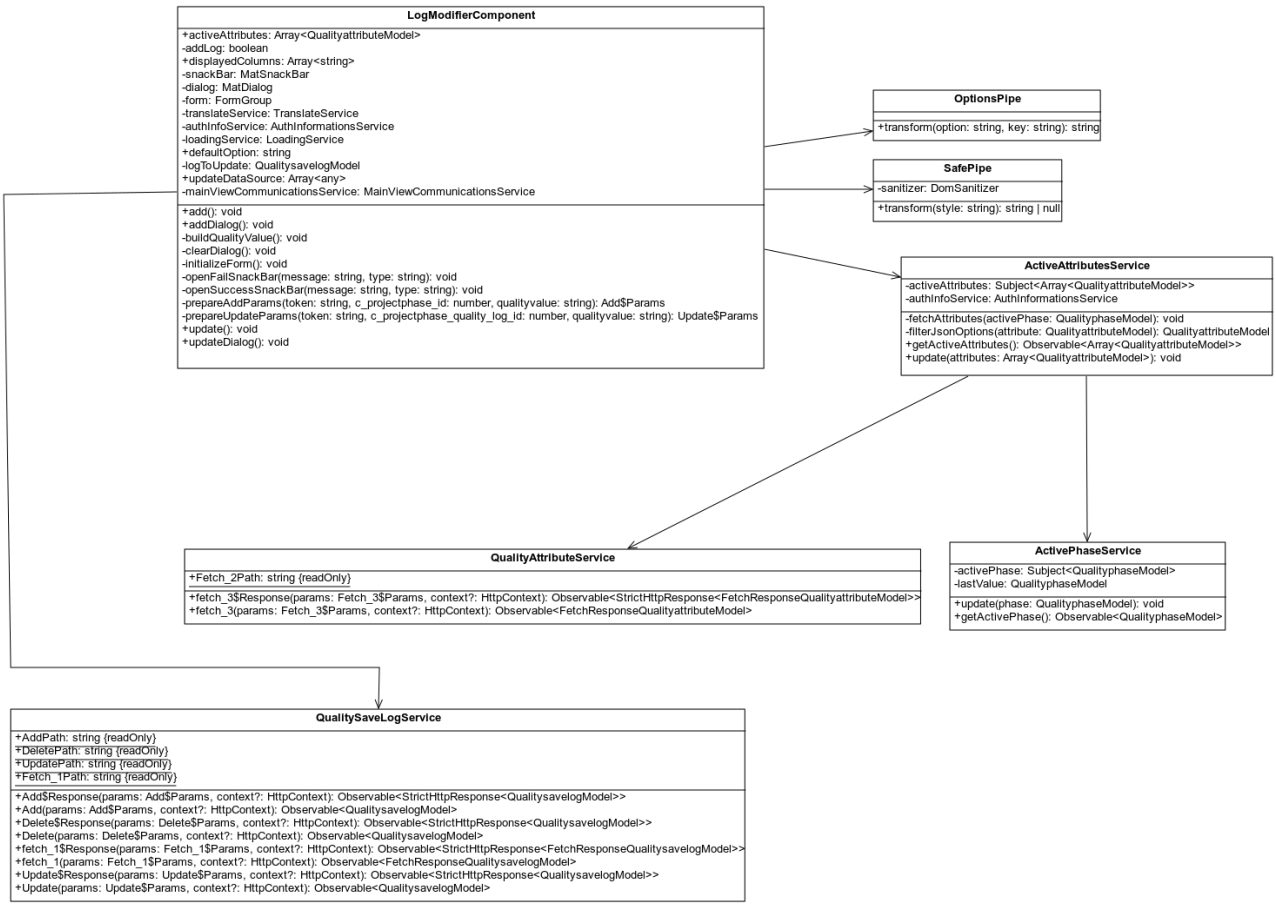

## Fasi



## Visualizzatore dei log




## Modificatore dei log



## Codice generato automaticamente

A partire dalla descrizione delle API in specifica OpenAPI 3.0, sono stati generati (tramite il pacchetto npm *ng-openapi-gen*) servizi, modelli e funzioni per la comunicazione al backend tramite protocollo HTTP.

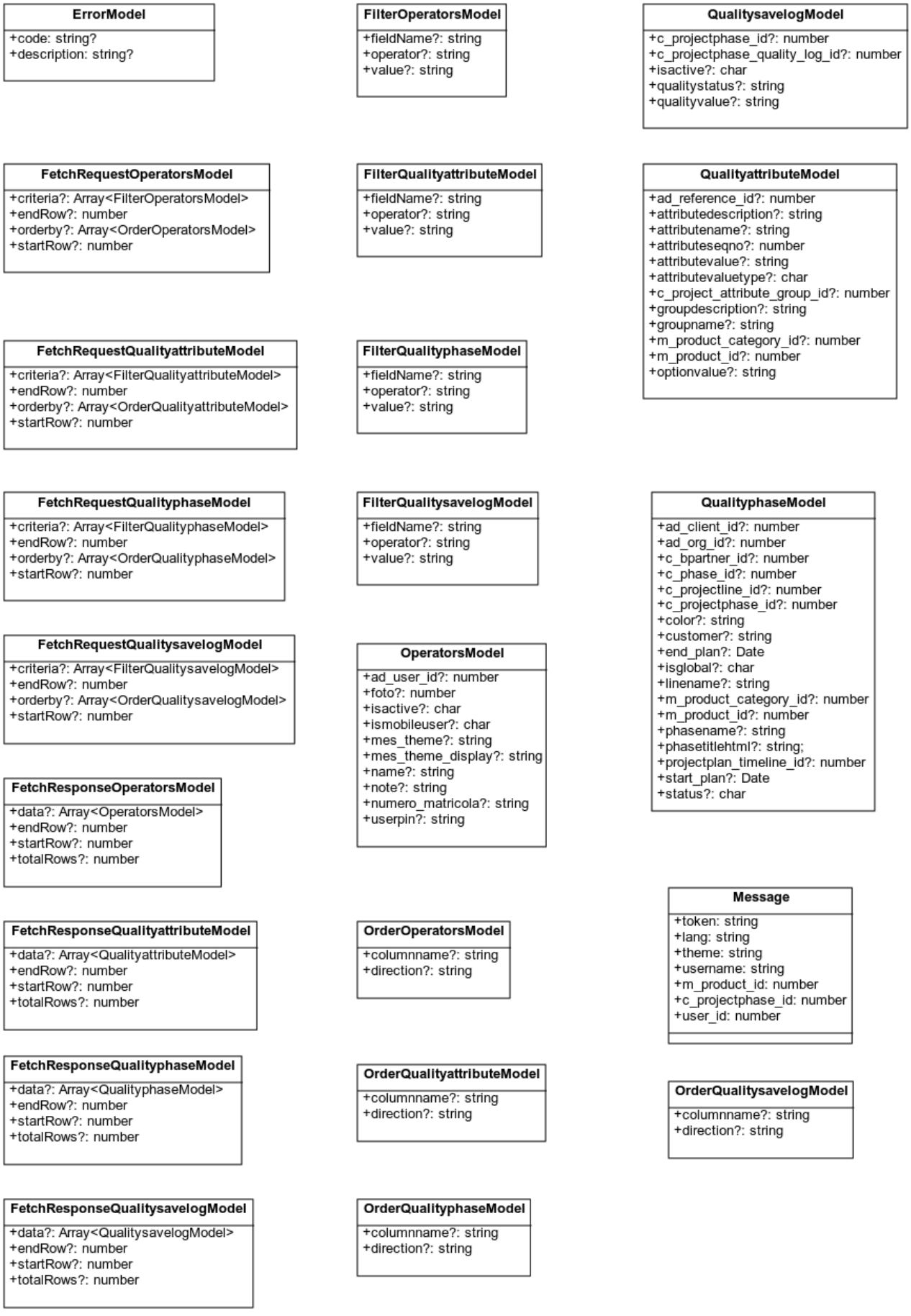
La generazione è avvenuta tramite il seguente comando:

*ng-openapi-gen --input .\src\assets\api-config.json --output .\src\app\api*

### Modelli

I modelli generati sono classi TypeScript che modellano il corpo delle richieste e delle risposte HTTP; fa eccezione la classe **Message** che modella la struttura del messaggio che deve essere passato in ingresso quando il prodotto viene eseguito all'interno di un <iframe>.

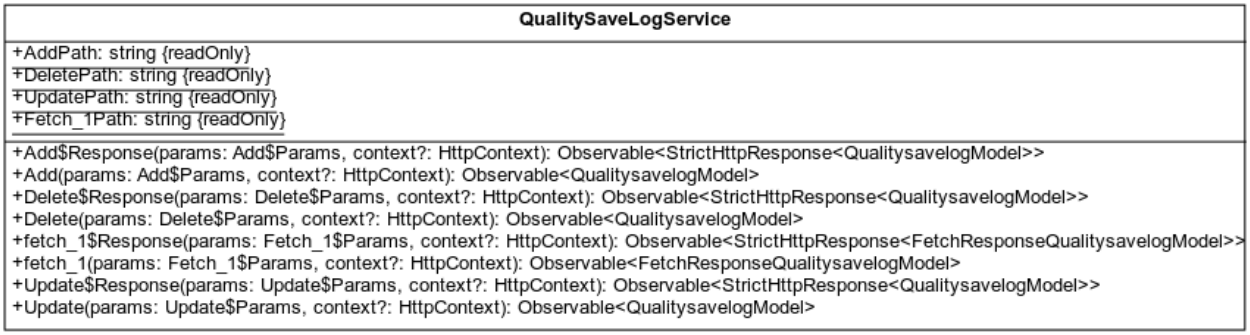
Di seguito un'immagine di tutti i modelli generati automaticamente:

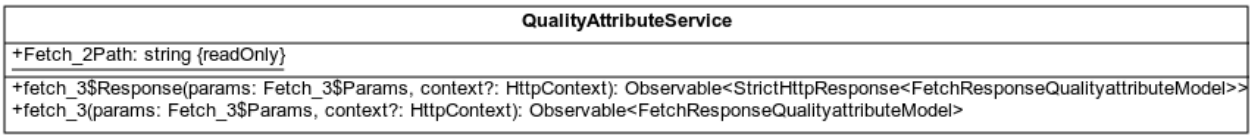

ErrorModel
+code: string?
+description: string?
FetchRequestOperatorsModel
+criteria?: Array<FilterOperatorsModel>
+endRow?: number
+orderby?: Array<OrderOperatorsModel>
+startRow?: number
FetchRequestQualityattributeModel
+criteria?: Array<FilterQualityattributeModel>
+endRow?: number
+orderby?: Array<OrderQualityattributeModel>
+startRow?: number
FetchRequestQualityphaseModel
+criteria?: Array<FilterQualityphaseModel>
+endRow?: number
+orderby?: Array<OrderQualityphaseModel>
+startRow?: number
FetchRequestQualitysavelogModel
+criteria?: Array<FilterQualitysavelogModel>
+endRow?: number
+orderby?: Array<OrderQualitysavelogModel>
+startRow?: number
FetchResponseOperatorsModel
+data?: Array<OperatorsModel>
+endRow?: number
+startRow?: number
+totalRows?: number
FetchResponseQualityattributeModel
+data?: Array<QualityattributeModel>
+endRow?: number
+startRow?: number
+totalRows?: number
FetchResponseQualityphaseModel
+data?: Array<QualityphaseModel>
+endRow?: number
+startRow?: number
+totalRows?: number
FetchResponseQualitysavelogModel
+data?: Array<QualitysavelogModel>
+endRow?: number
+startRow?: number
+totalRows?: number
FilterOperatorsModel
+fieldName?: string
+operator?: string
+value?: string
FilterQualityattributeModel
+fieldName?: string
+operator?: string
+value?: string
FilterQualityphaseModel
+fieldName?: string
+operator?: string
+value?: string
FilterQualitysavelogModel
+fieldName?: string
+operator?: string
+value?: string
OperatorsModel
+ad_user_id?: number
+foto?: number
+isactive?: char
+ismobileuser?: char
+mes_theme?: string
+mes_theme_display?: string
+name?: string
+note?: string
+numero_matricola?: string
+userpin?: string
OrderOperatorsModel
+columnname?: string
+direction?: string
OrderQualityattributeModel
+columnname?: string
+direction?: string
OrderQualityphaseModel
+columnname?: string
+direction?: string
QualitysavelogModel
+c_projectphase_id?: number
+c_projectphase_quality_log_id?: number
+isactive?: char
+qualitystatus?: string
+qualityvalue?: string
QualityattributeModel
+ad_reference_id?: number
+attributedescription?: string
+attributename?: string
+attributeseqno?: number
+attributevalue?: string
+attributevaluetype?: char
+c_project_attribute_group_id?: number
+groupdescription?: string
+groupname?: string
+m_product_category_id?: number
+m_product_id?: number
+optionvalue?: string
QualityphaseModel
+ad_client_id?: number
+ad_org_id?: number
+c_bpartner_id?: number
+c_phase_id?: number
+c_projectline_id?: number
+c_projectphase_id?: number
+color?: string
+customer?: string
+end_plan?: Date
+isglobal?: char
+linename?: string
+m_product_category_id?: number
+m_product_id?: number
+phasename?: string
+phasetitlehtml?: string;
+projectplan_timeline_id?: number
+start_plan?: Date
+status?: char
Message
+token: string
+lang: string
+theme: string
+username: string
+m_product_id: number
+c_projectphase_id: number
+user_id: number
OrderQualitysavelogModel
+columnname?: string
+direction?: string

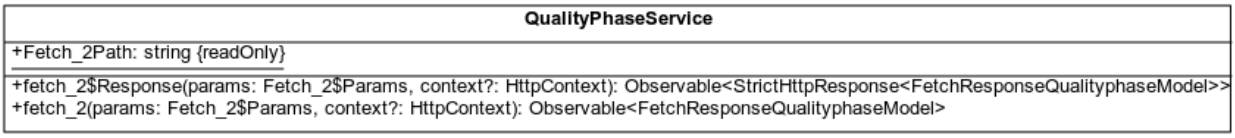
## Servizi

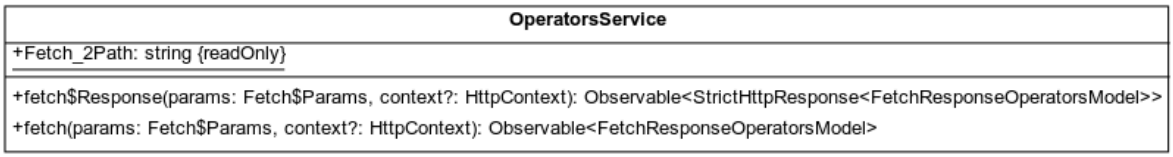
I servizi generati automaticamente sono relativi alla comunicazione tramite API con il backend aziendale; i servizi sono i seguenti:

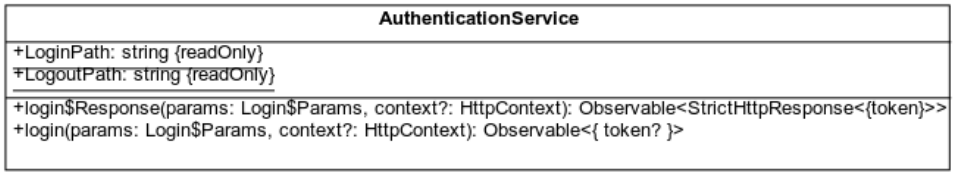
- **QualitySaveLogService**: servizio per eseguire operazioni CRUD su log di qualità;
- **QualityAttributeService**: servizio per ottenere gli attributi di qualità per la fase attiva;
- **QualityPhaseService**: servizio per ottenere le fasi di lavorazione;
- **OperatorsService**: servizio per ottenere gli operatori;
- **AuthenticationService**: servizio per gestire l'autenticazione tramite nome utente e password.



## Interfaccia grafica

Per quanto riguarda l'interfaccia grafica, si è deciso di seguire lo schema di colori dato dal prodotto ADeMES (prodotto nel quale il software dovrà essere integrato) per compatibilità.

Di seguito, una tabella che indica le corrispondenze tra i colori (in formato esadecimale) relativi ai temi di interfaccia per i vari elementi:

| Componente | Scuro | Chiaro |
|---|---|---|
| Barra di intestazione | #232F4E | #0C2656 |
| Barra laterale | #000000 | #FFFFFF |
| Intestazioni delle sezioni principali (attributi e log), bordi delle fasi selezionate e bottone per il cambio tema | #BF71FF | #123A83 |
| Intestazione delle tabelle | #452E60 | #5560BB |
| Pulsante di inserimento / aggiornamento log | #5560BB | #1CBFFF |
| Sfondo delle tabelle | #323347 | #F3F5FD |
| Linee divisorie della visualizzazione principale | #707286 | #CED0DB |
| Sfondo della carta contenente la fase selezionata | #452E60 | #9070A6 |
| Sfondo delle carte non contenenti la fase selezionata | #5560BB | #5560BB |
| Bordi delle carte non contenenti la fase selezionata | #7280F9 | #6673DB |
| Sfondo della visualizzazione principale | #1A1B2E | #F3F5FD |

## Integrazione all'interno di un frame HTML

Per quanto riguarda l'integrazione del prodotto all'interno di un *<iframe>* del software ADeMES (e, in generale, della suite aziendale) occorre seguire i passi successivi:

1. Passare come parametro ***inside=true*** nel link presente nell'attributo *src* del tag *iframe*; serve per indicare al prodotto contenuto che è eseguito all'interno di un'altra applicazione web;
2. Passare tramite *postMessage* un oggetto corrispondente alla struttura definita nella classe *Message* (vedi sezione **Codice generato automaticamente** > **Modelli**); tale oggetto contiene:
    1. *token*: stringa alfanumerica che indica il token di autenticazione ai servizi REST TriZeta;
    2. *lang*: linguaggio della traduzione attiva, in stringa codificata secondo lo standard ISO 639-1 (ad esempio, "italiano" diventa "it");

3. *theme*: stringa che indica il tema grafico attivo, può avere valore **DM** (tema scuro) o **WM** (tema chiaro);
4. *username*: nome dell'operatore autenticato;
5. *m_product_id*: identificativo del prodotto della fase attiva, serve per ottenere gli attributi e poter inserire i log di qualità;
6. *c_projectphase_id*: identificativo della fase attiva;
7. *user_id*: identificativo dell'operatore attivo.