ANALISI DEI REQUISITI

• Requisiti funzionali

Per lo sviluppo di tale progetto,denominato "**Banking Service**", si è partiti dall'idea di realizzare una Web-app per un sistema bancario, la quale consente di gestire la lista dei clienti registrati presso una banca.

L'idea nasce dall'esigenza di ogni istituto bancario di tenere traccia dei propri clienti ai quali offre servizi. I clienti possono essere distinti in privati o aziende. Ogni cliente è caratterizzato dai seguenti dati:

1. ID
2. Nome (o ragione sociale nel caso di aziende)
3. Codice fiscale (o partita iva)
4. Email
5. Saldo del conto

Sulla base di quanto è stato ipotizzato sopra, sono state previste le seguenti funzionalità:

- Visualizzazione della lista dei clienti, con possibilità di visualizzare in maniera separata i privati, le aziende o entrambe le tipologie di clienti;
- Inserimento di nuovi clienti;
- Eliminazione o modifica dei dati relativi ai clienti già esistenti;
- Ordinamento dell'elenco per tipologia di cliente o per saldo, in ordine crescente o decrescente;
- Barra di ricerca per il filtraggio dei clienti;

• Requisiti non funzionali

- ***Corretto utilizzo delle best practice;***
  L'utilizzo di un set di regole informali diventa essenziale nello sviluppo del software in quanto facilita lo sviluppo iniziale e la successiva fase di manutenzione che possono essere realizzate anche da personale differente da quello iniziale;

- ***Qualità del design della UI;***
  L' interfaccia deve essere intuitiva e di facile comprensione; l'utilizzo deve essere veloce e le operazioni possibili devono essere effettuate con pochi e semplici comandi.

- ***Design Responsive;***
  La web-app deve essere progettata in modo da adattare i contenuti in base al dispositivo in cui viene visualizzata;

- ***SPA (Single Page Application);***
  L'applicazione deve essere eseguita all'interno di una singola pagina; all'interno di tale pagina vengono caricate viste diverse in base all'interazione dell'utente con la pagina stessa,dando l'illusione all'utente di navigare su pagine differenti.

- ***Qualità del codice sorgente;***
  Strutturare l'applicazione secondo il principio LIFT,in modo da ottenere una struttura modulare,aumentando l'efficienza degli sviluppatori in quanto consente di trovare rapidamente il codice desiderato attraverso una struttura di file e cartelle "flat", ovvero di bassa profondità.

CASI D'USO

Andiamo a definire adesso nel dettaglio i vari casi d'uso per fornire una descrizione delle interazioni tipiche tra l'utente e la web-app.

- **INSERIMENTO NUOVO CLIENTE**

  Utente
  1) Attraverso un pulsante viene visualizzata una finestra di dialogo contenente il form con gli input text;
  2) L'utente inserisce i dati relativi al nuovo cliente;
  3) Se i dati sono stati correttamente inseriti, il cliente viene aggiunto alla lista dei clienti;
  4) Viene visualizzata un alert dialog per la conferma dell'avvenuta registrazione.

  Web-app
  1) Alla pressione del pulsante apre la finestra di dialogo;
  2) Durante l'inserimento dei dati relativi al cliente effettua un controllo sul form per verificare la correttezza dei dati;
  3) Se i dati sono stati correttamente inseriti, abilita il pulsante "Submit" per consentire l'inserimento del cliente nella lista.
  4) Visualizza un alert dialog.

- **ELIMINAZIONE CLIENTE**

  Utente
  1) L'utente visualizza l'elenco dei clienti registrati nel sistema bancario;
  2) Per ogni cliente vengono visualizzate le informazioni relative al cliente stesso, e 3 pulsanti per la selezione, la modifica o l'eliminazione del cliente;
  3) L'utente clicca sull'icona relativa all'eliminazione del cliente;
  4) Conferma dell'eliminazione;
  5) Il cliente viene cancellato dalla lista dei clienti.

  Web-app
  1) L'applicazione fornisce una view con l'elenco dei clienti registrati presso la banca;
  2) Alla pressione del pulsante viene visualizzata una finestra di dialogo dove è possibile confermare l'eliminazione del cliente;
  3) Se il cliente conferma l'eliminazione esegue una ricerca del cliente attraverso l'indice e una volta trovato viene cancellato dalla lista.
  4) Restituisce l'elenco aggiornato dei clienti.

- **MODIFICA DEL CLIENTE**

  Utente
  1) L'utente visualizza l'elenco dei clienti registrati nel sistema bancario;
  2) Per ogni cliente vengono visualizzate le informazioni relative al cliente stesso, e 3 pulsanti per la selezione, la modifica o l'eliminazione del cliente;
  3) L'utente clicca sull'icona relativa alla modifica del cliente;
  4) Viene aperta una finestra di dialogo contenente il form con gli input text già compilati con i dati attuali;
  5) Modificare i vari campi;
  6) Se i dati sono stati correttamente inseriti, viene effettuata la modifica del cliente;
  7) Viene visualizzata un alert dialog per la conferma dell'avvenuta modifica.

Web-app

1) L'applicazione fornisce una view con l'elenco dei clienti registrati presso la banca;
2) Durante l'inserimento dei dati relativi al cliente effettua un controllo sul form per verificare la correttezza dei dati;Se i dati sono stati correttamente inseriti, abilita il pulsante "Submit" per consentire l'inserimento del cliente nella lista.
3) Viene eliminato dalla lista il cliente con i dati precedentemente inseriti e viene sostituito con il nuovo cliente.
4) Viene fornita la lista aggiornata dei clienti.

- **FILTRAGGIO DEI CLIENTI**

Utente

1) Sulla toolbar l'utente può usufruire di una barra di ricerca;
2) Inserisce i caratteri per la ricerca di determinati clienti;
3) Viene visualizzato l'elenco filtrato dei clienti.

Web-app

1) Prende i caratteri forniti dall'utente;
2) Effettua una ricerca dei clienti che contengono nel proprio nome i caratteri passati dall'utente;
3) Restituisce la lista dei clienti filtrata.

- **ORDINAMENTO DELL'ELENCO PER NOME CLIENTE O PER SALDO, IN ORDINE CRESCENTE O DECRESCENTE;**

Utente

1) Visualizza l'elenco dei clienti con un ordinamento basato sull'ordine di inserimento del cliente nella lista;
2) Ha a disposizione due selection field, uno in cui può decidere se ordinare in base al saldo o in base al nome cliente, e uno in cui può scegliere l'ordine crescente o decrescente;
3) Visualizzato l'elenco ordinato dei clienti secondo le proprie preferenze.

Web-app

1) L'applicazione fornisce una view con l'elenco dei clienti registrati presso la banca;
2) Riceve i parametri passati dall'utente ed applica un filtro alla lista dei clienti;
3) Restituisce la lista dei clienti filtrata secondo le preferenze dell'utente.

- **VISUALIZZAZIONE PERSONALIZZATA DELLA LISTA DEI CLIENTI**

Utente

1) Ha a disposizione tre tab,ognuna con una visualizzazione dei clienti differenziata per tipo(Privato, Azienda, Entrambi);
2) Seleziona il tab desiderato;
3) Visualizza l'elenco dei clienti del tipo scelto.

Web-app

1) Alla selezione del tab da parte dell'utente viene passata la tipologia del cliente da visualizzare;
2) Viene applicato un filtro alla vista in base alla tipologia del cliente;
3) Viene restituita la lista filtrata dei clienti.

IMPLEMENTAZIONE

In questa fase si procede nella descrizione delle scelte implementative che soddisfano i requisiti richiesti.

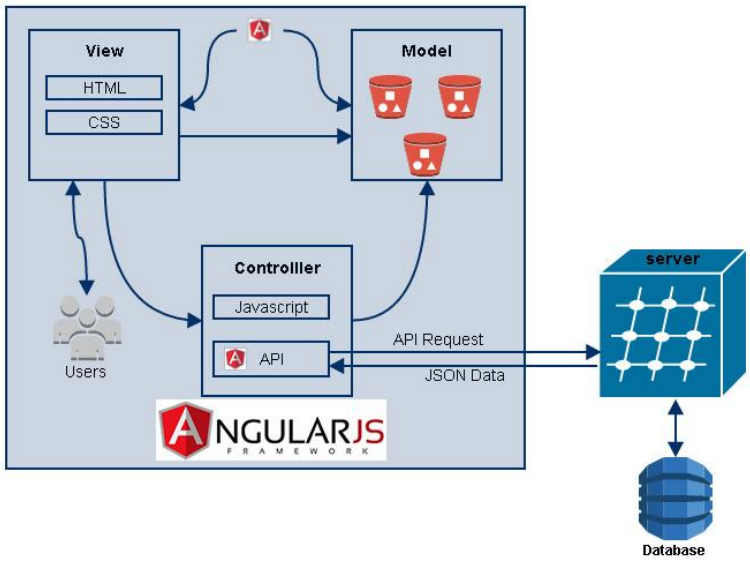
La web-app da realizzare è una **SPA(Single page application),** ovvero un'applicazione eseguita in un'unica pagina html, all'interno della quale vengono caricate viste diverse in base all'interazione dell'utente con la pagina stessa. Il meccanismo che consente di mappare ad ogni vista il proprio URL prende il nome di "**routing**",applicabile in Angular attraverso il modulo **ngRoute**.

Per l'implementazione dell'interfaccia utente è stata sfruttata la libreria **Angular Material** la quale mette a disposizione delle specifiche di **Google Material Design**, permette allo sviluppatore di realizzare UI in maniera semplice e dal design moderno.
Una delle caratteristiche principali di questa libreria è la semplicità nel rendere il design della web-app di tipo **responsive**, ovvero adattabile a qualsiasi dispositivo (pc, tablet,smartphone).
Ciò è possibile attraverso le delle semplici direttive disponibili nella libreria.

Per consentire la persistenza dei dati inseriti durante l'utilizzo della web-app, è stata aggiunta la possibilità di esportare l'elenco dei clienti in un file CSV.

Lo schema della web-app segue i prinicipi del design pattern **MVC (Model – View – Controller)** dove:

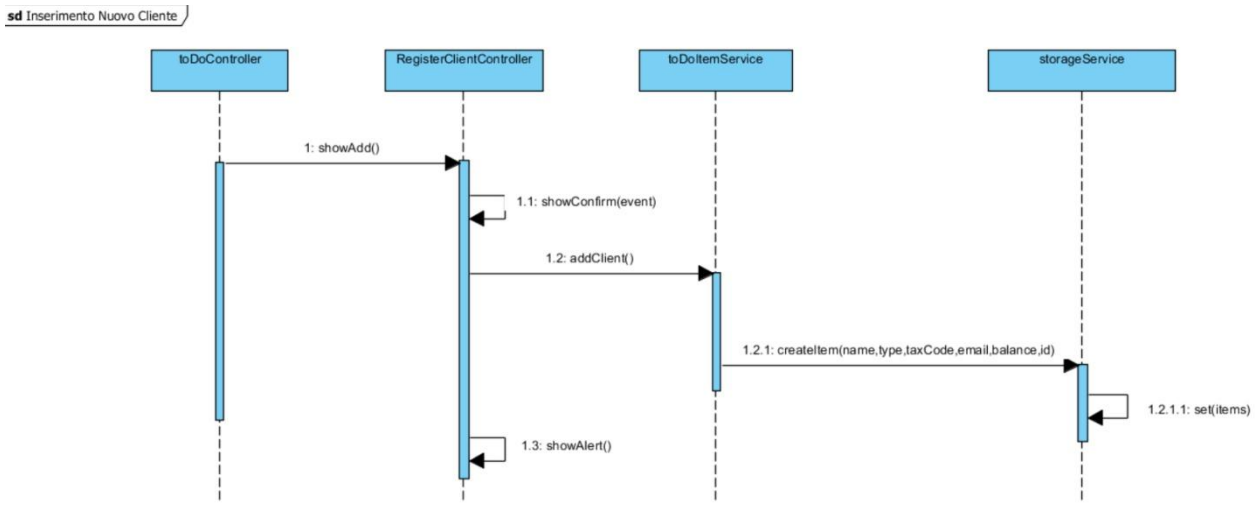
- Model rappresenta il modello dei dati della nostra applicazione;
- View, rappresenta il modo in cui i dati vengono visualizzati all'utente;
- Controller, mette in relazione il modello dei dati con le varie viste.



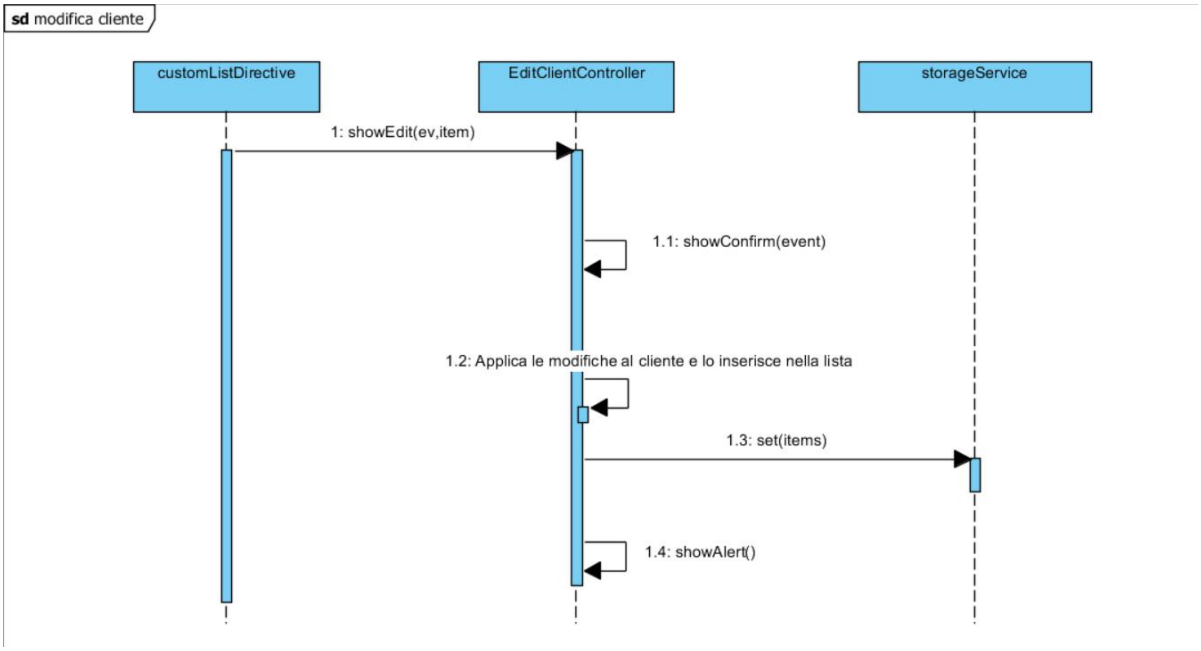
DIAGRAMMI DELLE ATTIVITA’

Attraverso i diagrammi delle attività andiamo a definire nel dettaglio come avvengono le interazioni tra i vari componenti software della nostra web-app (controller, service ecc…) per descrivere le dinamiche con cui si sviluppano alcuni dei casi d’uso.

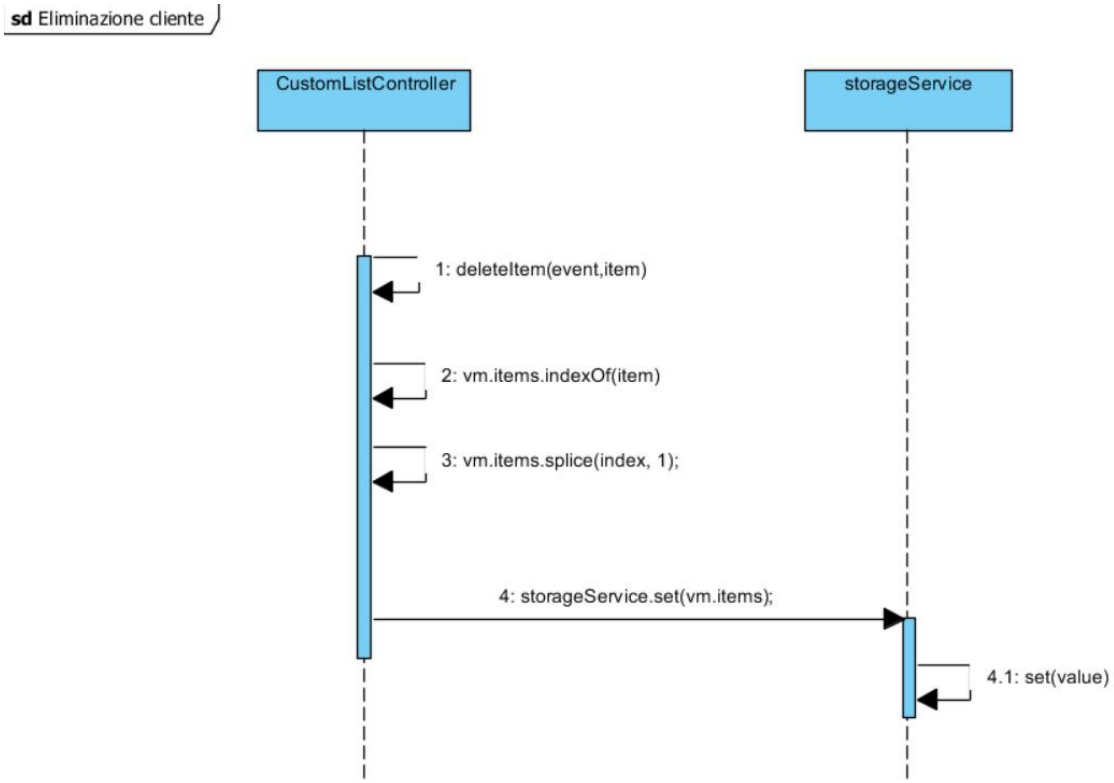
- ***INSERIMENTO NUOVO CLIENTE***



- ***MODIFICA CLIENTE***



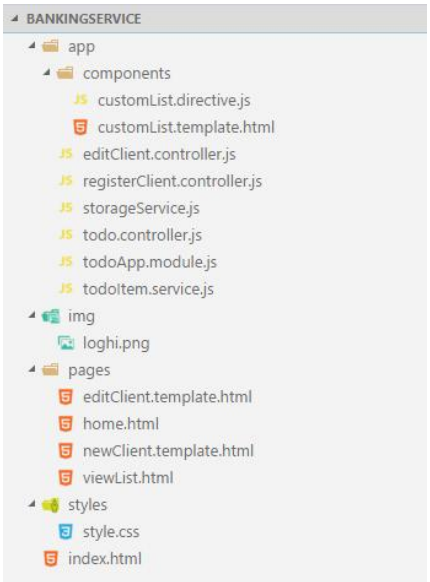
- ***ELIMINAZIONE CLIENTE***



STRUTTURA DELLA WEB-APP

Per quanto riguarda la strutturazione della web-app si è cercato di seguire il principio di strutturazione delle cartelle per features,in modo da velocizzare la ricerca dei file e migliorarne l'organizzazione.

La web-app risulta dunque formata



dai seguenti file e cartelle:

Andiamo a definire adesso nel dettaglio l'utilizzo dei vari file all'interno della nostra web-app:

MODULI

- **todoAppModule.js**
  Questo file contiene la definizione del modulo principale della web-app, all'interno del quale viene configurato anche il meccanismo di routing per la SPA(Single page application).

CONTROLLORI

- **Todo.controller.js**
  Questo file contiene la definizione del controller principale che si occuperà della gestione delle funzionalità di base della web-app.
  Contiene metodi per l'eliminazione dei clienti dalla lista, per la gestione della sidenav laterale e della toolbar ecc…

- **registerClient.controller.js**
  Questo file contiene la definizione del controller che si occuperà di gestire la registrazione di un nuovo cliente.

- **editClient.controller.js**
  Questo file contiene la definizione del controller che si occuperà di gestire le modifiche di un cliente già esistente.

SERVIZI

- **todoItem.service.js**
  In questo file definiamo un servizio personalizzato per gestire in maniera adeguata l'inserimento di ogni nuovo cliente all'interno della lista.

- **storageService.js**
  In questo file definiamo un servizio personalizzato per gestire le operazioni di lettura e scrittura della lista dei clienti nella session storage.

DIRETTIVE

- **customList.directive.js**
  Questo file contiene la direttiva personalizzata che si occupa di manipolare e gestire l'elenco dei clienti nella pagina html.
  Nella funzione della direttiva, vengono passate le impostazioni della direttiva, tra le quali il templateUrl, ovvero il template di come verrà visualizzata la lista dei clienti,e il controllore che si occupera di gestire la view relativa alla lista dei clienti.

HTML

- **index.html**
  Rappresenta la pagina principale della web-app,all'interno del quale viene utilizzato il meccanismo di routing per la gestione delle view da inserire di volta in volta a seconda delle esigenze.
  Al suo interno viene definita la toolbar, la sidenav e vengono definite tutte le librerie utili per il funzionamento della web-app.

- **home.html**
  All'interno di tale pagina è stata creata una semplice presentazione del progetto e dei relativi partner.
  Tale pagina sarà la prima view che verrà visualizzata al caricamento della index.

- **viewList.html**
  All'interno di tale pagina viene definita la visualizzazione dell'elenco dei clienti.
  Essa sarà formata da tre tabs, e all'interno di ogni tab viene utilizzata la direttiva custom-list per personalizzare l'elenco da visualizzare a seconda del tab scelto.

- **customList.template.html**
  Rappresenta il template per la visualizzazione dei clienti, all'interno del quale vengono inseriti una serie di filtri e una serie di funzionalità per ogni singolo cliente, come ad esempio la possibilità di selezionare, modificare o cancellare ogni singolo cliente.
  Questo template viene gestito dalla direttiva customList la quale contiene i vari metodi utili a sfruttare correttamente le varie funzionalità previste.

- **newClient.template.html**
  Tale pagina contiene il form per la registrazione di un nuovo cliente.
  In tale form vengono applicati una serie di controlli sui vari input in modo tale che l'utente possa inserire i dati in maniera corretta.
  Tale pagina viene gestita dal controllore registerClient.

- **editClient.template.html**
  Tale pagina contiene il form per la modifica del cliente selezionato.
  Il form è identico a quello definito nella pagina newClient.template.html con la differenza che in tal caso i vari campi di input sono già pre-compilati con i dati relativi al cliente che sono stati inseriti al momento della registrazione.
  Tale pagina è gestita dal controllore editClient.