Documento specifica requisiti progetto applicazione "TodoList":

In questo documento si vogliono identificare le funzionalità che la nostra applicazione dovrà avere. Per prima cosa bisogna definire le <u>user story</u> le quali aiutano a far capire al cliente l'utilità della funzione stessa e la sua priorità.

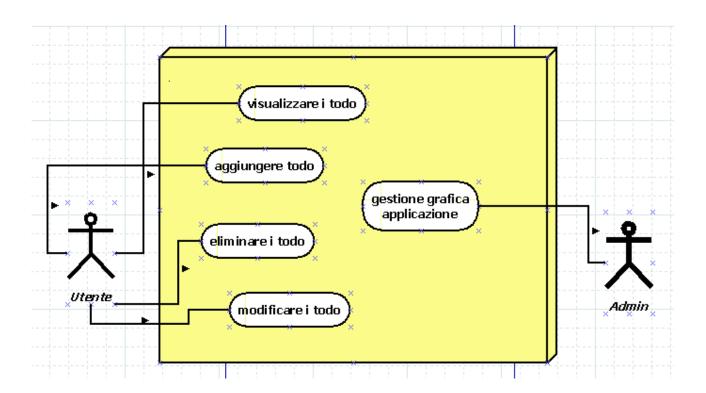
Successivamente potremo passare alla realizzazione del *use case diagram* che ci permetterà di:

- 1- Modellare sequenze di azioni del sistema che forniscono un risultato osservabile all'esterno:
- 2- Fornire una base per definire l'interazione uomo/macchina del sistema;
- 3- Dare una notazione visuale adatta agli utenti finali;
- 4- Dare una base per le specifiche dei test.

User story:

- Come utente voglio visualizzare i todo
- Come utente voglio aggiungere nuovi todo
- Come utente voglio eliminare i todo
- Come utente modificare i todo
- Come administrator voglio poter gestire l'interfaccia grafica dell'applicazione

Use case diagram:



Class diagram:

Il digramma delle classi ci consentirà di rappresentare una vista del sistema software che pone l'accento sulla struttura degli oggetti (classi di appartenenza, relazioni, attributi, operazioni) e descrive il tipo degli oggetti che compongono il sistema e le relazioni statiche esistenti tra lo.

Todolist	Todoltems	Арр
Classe che gestisce il form per l'inserimento di un nuovo todo - render () Rirona il form composto da un input di testo ed un bottone add per aggiungere un item alla lista di item	Classe che gestisce la visualizzazione dei todo - render () Ritorna la lista di todo che abbiamo aggiunto composta da bottoni per eliminare o modificare uno specifico todo	Classe che gestisce la logica dell'applicazione - deleteltem = key => { } Elimina un item dalla lista di item - handleInput = e => { } Aggiorna lo stato di un item dall'input dell'utente - addItem = e => { } Aggiunge un item alla lista di item