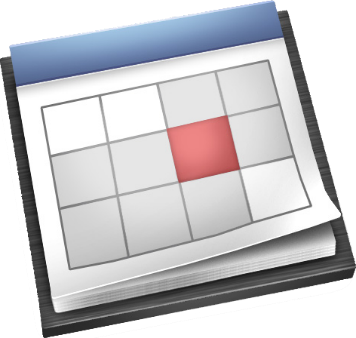
**Cemin** stefano

**madonia** tommaso

**marogna** luca

DynamiCal

The smartest way to be organized



# Descrizione del Problema

Si vuole realizzare un'agenda elettronica con la quale tenere organizzati i propri calendari e impegni. Deve essere possibile gestire più calendari (locali o condivisi con i propri amici, identificati tramite email) identificati univocamente da un nome e ai quali è possibile aggiungere o rimuovere eventi. Ogni evento appartiene ad uno e un solo calendario ed è caratterizzato da un nome, una data, una durata (in minuti) e una periodicità, inoltre può avere un luogo e una descrizione. La periodicità di un evento descrive in che modo un evento si ripete nel tempo (mai, ogni tot giorni/settimane/mesi/anni). Durante la creazione di un evento è necessario assegnare un modello all'evento. Un modello di evento è personalizzabile dall'utente e contiene una lista ordinata di voci (nome e tipo) che descrivono campi aggiuntivi dell'evento; ogni modello è identificato univocamente da un nome. Sarà presente un modello di evento di default. Più eventi possono essere descritti da uno stesso modello e un modello non può più essere modificato una volta creato. Un modello può essere eliminato solo se non ci sono eventi descritti da questo. Infine deve essere possibile visualizzare gli eventi filtrandoli con diverse strategie (per calendario/arco temporale/modello di evento/ricerca). L'inserimento, la modifica o la cancellazione di un evento o di un calendario provoca un eventuale aggiornamento automatico del risultato di un filtraggio. Il criterio di filtraggio può includere più di un filtro.

Tipologie di filtro:

* Un filtro per calendario filtra gli eventi appartenenti ad un sottoinsieme dei calendari di un'agenda
* Un filtro per arco temporale filtra gli eventi la cui data di inizio appartiene ad un determinato periodo di tempo (data inizio/fine)
* Un filtro per modello di evento filtra gli eventi che hanno un determinato modello
* Un filtro per ricerca filtra gli eventi che contengono nel titolo la parola o le parole da ricercare

# Glossario

# Casi d’uso e scenari

# Descrizione dei casi d’uso

# Diagramma delle Classi d’Analisi

# Diagramma di sequenza

# Diagramma delle classi di progettazione

# Design patterns e design principles

Si è deciso di usare il pattern **Singleton** per l’implementazione di *Agenda* in modo che i *Calendari* e i *Modelli* di evento siano facilmente accessibili globalmente; inoltre il progetto non prevende la possibilità di gestire più *Agende* quindi l’instanza di *Agenda* dev’essere unica. Sono stati implementati eventi per notificare l’aggiunta o rimozione di un calendario e di un modello di evento.

Si è scelto di usare il pattern **Factory** per gestire la creazione dei *Calendari*, dei *Filtri* e delle *Voci<T>*. Questa scelta è stata possibile grazie all’applicazione del **Dependecy Inversion Principle** (DIP): ogni calendario estende la superclasse astratta *Calendario*, ogni filtro estende la superclasse astratta *Filtro* mentre è stata introdotta un’interfaccia *IVoce* che viene implementata dalla classe generica *Voce<T>*.

Per la gestione dei filtri abbiamo usato il pattern **Decorator** grazie al quale è possibile aggiungere facilmente nuovi filtri al sistema ed è possibile eseguire filtraggi di qualsiasi complessità. È possibile decorare un *CriterioDiFiltraggio* usando un qualsiasi numero di filtri che estendono *Filtro*. L’utilizzo del pattern Decorator è conforme al DIP: *CriterioDiFiltraggio* e *Filtro* implementano un’interfaccia comune *IFiltro*. Questo pattern segue anche l’**Open/Close Principle** (OCP) in quanto è facilmente possibile aggiungere nuovi filtri al sistema creando una sottoclasse di *Filtro* e senza dover fare altre modifiche.

Infine abbiamo usato il pattern **Template** per l’implementazione dei *Filtri*, il metodo *FiltraEventi()* della classe astratta *Filtro* fornisce un’implementazione comune a tutti i filtri e viene specializzato nelle sottoclassi di *Filtro* attraverso l’ovveride del metodo protetto e astratto *StrategiaFiltro()*. Anche questo pattern è conforme al DIP.