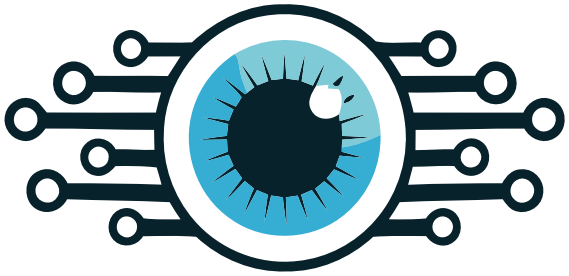
PREMI



Pragma

Verbale esterno 2015-03-17

Informazioni sul documento

Versione	0.1.0
Redazione	Stefano Munari
Verifica	Daniele Marin
Approvazione	Stefano Munari
Uso	Esterno
Distribuzione	Pragma
Destinato a	Prof. Vardanega Tullio
	Prof. Cardin Riccardo
	Zucchetti S.p.A.

Sommario

Verbale dell'incontro avvenuto martedì 17 marzo 2015 tra il gruppo Pragma e il committente Prof. Cardin Riccardo, relativamente all'attività di Progettazione.

A.A. 2014-15

pragma.swe@gmail.com

Diario delle modifiche

Data	Descrizione modifica	Autore	Ruolo	Versione
2015-03-25	*Verifica del documento*	Daniele Marin	Progettista	0.1.1
2015-03-18	*Stesura completa delle sezioni del documento*	Stefano Munari	*Responsabile di Progetto*	0.1.0
2015-03-17	*Creazione scheletro del documento*	Stefano Munari	*Responsabile di Progetto*	0.0.0

Tabella 1: Diario delle modifiche.

Indice

1 Informazioni generali

- **Data incontro:** 2015-03-17
- **Ora incontro:** 13:10
- **Luogo incontro:** Aula 1C150 complesso Torre di Archimede;
- **Durata:** 20 minuti
- **Partecipanti:**

Nominativo	Ruolo
Prof. Cardin Riccardo	Committente
Massimiliano Baruffato Daniele Marin Stefano Munari Andrea Ongaro Fabio Vedovato	Membri Pragma

2 Problemi e decisioni

2.1 Modellazione pagine HTML

Problema La modellazione tramite *UML*$_G$ della parte *front-end*$_G$, composta da pagine *HTML*$_G$, rappresenta un problema in quanto non si riescono ad individuare delle entità rappresentabili.

Decisione La parte *front-end*$_G$ modellata non farà riferimento alle pagine *HTML*$_G$ ma bensì al *framework*$_G$ che adotteremo. Quest'ultimo si occuperà di generare le pagine attraverso delle direttive inserite nelle pagine *HTML*$_G$ stesse.

2.2 Descrizione framework

Problema Sono sorti dei dubbi riguardo a cosa descrivere precisamente del *framework*$_G$ che andremo ad adottare per costruire la *View*$_G$.

Decisione Andranno modellate (quindi descritte nel documento *Specifica Tecnica*) solo le parti del *framework*$_G$ che useremo, ovvero quelle parti che interagiscono con l'application logic. Le *View*$_G$ vanno quindi modellate secondo l'interfaccia che offriranno ai propri *Controller*$_G$ o *Presenter*$_G$.

2.3 Package

Problema Decidere se mantenere la suddivisione in package imposta da *AngularJS*$_G$ oppure se adottarne una propria, su misura per il tipo di soluzione pensata. Inoltre non è chiaro quali siano i criteri da seguire per comporre un package.

Decisione Adottando un *framework*$_G$ ci si dovrà adeguare a ciò che il *framework*$_G$ fornisce.
Sicuramente si dovrà mantenere la divisione view-controller-services perchè *AngularJS*$_G$ lo impone.
Nel caso di *AngularJS*$_G$, ad esempio, i *controller*$_G$ sono degli oggetti le cui primitive per la costruzione sono fornite dal *framework*$_G$ stesso e quindi non si dovrà definire una propria versione del *controller*$_G$ perché questa è già predisposta.
I package devono racchiudere componenti o entità che siano tra loro fortemente correlate.
I package devono essere consistenti.
Un package per la *mappa mentale*$_G$ è un buon esempio di package.

2.4 Linguaggio Javascript

Problema Avendo iniziato la modellazione approcciandoci ad essa pensando ad un linguaggio stile Java, perché più comodo in quanto utilizza oggetti e divisione in package, ora ci troviamo in difficoltà dovendoci rapportare con un linguaggio profondamente diverso.

Decisione Non è possibile modellare una soluzione senza fare riferimento al linguaggio che si andrà ad utilizzare. Il fatto di utilizzare un *framework*$_G$ o delle *librerie*$_G$ impone delle scelte che legano profondamente la progettazione al linguaggio di programmazione che si andrà poi ad utilizzare nel momento della codifica del prodotto, quindi non è possibile essere troppo astratti in questa fase.

2.5 Uso dei Design Pattern

Problema É stata valutata l'idea di utilizzare l'implementazione di *Observer*$_G$ tramite l'uso degli aspetti per rendere più efficiente il controllo delle *View*$_G$, ma non si è certi che questa sia la soluzione ideale per il problema. A tal proposito sono emersi dubbi su come utilizzare i desing pattern per la parte *front-end*$_G$ e *back-end*$_G$ del sistema.

Decisione Il *framework*$_G$ per la parte *front-end*$_G$ offre già i pattern di cui abbiamo bisogno.
Il modo in cui *AngularJS*$_G$ esegue il binding fra *View*$_G$ e Scope è certamente un *Observer*$_G$ che viene implementato come un ciclo infinito che controlla costantemente le azioni delle diverse *View*$_G$.
Nella parte *front-end*$_G$ dell'applicativo saranno utilizzati i design pattern resi disponibili da *AngularJS*$_G$.
Mentre per la parte *back-end*$_G$ saranno definiti ed implementare i desing pattern che riterremo più adatti al nostro sistema.

2.6 Utilizzo di Facade

Problema Abbiamo avuto dei problemi nell'utilizzo del *design pattern*$_G$ *facade*$_G$ in quanto ha aumentato notevolmente la complessità del Model progettato, quindi sono emersi dubbi riguardo a quando sia necessario usare un tale pattern.

Decisione Il *design pattern*$_G$ *facade*$_G$ verrà utilizzato quando sarà necessario fornire un interfaccia semplice ad un sottosistema complesso.
Nel caso in cui si debbano organizzare i sottosistemi in una struttura a diversi livelli il *facade*$_G$ potrà essere una buona soluzione per definire un unico punto d'ingresso ad ogni livello.
Quando sono presenti molte dipendenze tra i vari *client*$_G$ e le classi che implementano un'astrazione è utile utilizzare *facade*$_G$ per disaccoppiare il sistema dai *client*$_G$ e dagli altri sottosistemi.

2.7 Inserire i pattern nella *Specifica Tecnica*

Problema Non è chiaro come debbano essere inseriti i design pattern all'interno del documento *Specifica Tecnica*.

Decisione Nel documento *Specifica Tecnica* dovranno essere indicati i design pattern utilizzati e descritto il loro funzionamento. Inoltre dev'essere spiegato come i pattern vengono integrati nel sistema e perché.

2.8 Contenuto *Definizione di Prodotto*

Problema Sono emersi dei dubbi su cosa vada effettivamente specificato nel documento *Definizione di Prodotto*.

Decisione Nel documento *Definizione di Prodotto* vanno indicate le componenti che sono state individuate nella *Specifica Tecnica*.
Queste devono essere portate ad un livello di dettaglio tale che il programmatore riesca a tradurre le informazioni del documento in codice.
Per scrivere questo documento è necessario conoscere il linguaggio che si utilizzerà per la successiva codifica della soluzione.