HTML Mobile App: perchè utilizzare l'MVC

In un precedente articolo avevo parlato delle view e della loro importanza per quanto riguarda l'incremento delle prestazioni di **rendering** e della **fluidità,** nel caso dell'interfaccia di un'applicazione mobile sviluppata in **HTML, CSS3** e **Javascript.**

La view è uno dei componenti principali dello **sviluppo MVC,** una logica molto diffusa nella realizzazione di applicazioni a pagina singola (**single page application),** ovvero quelle applicazioni in cui la pagina non viene mai ricaricata, come nel caso di **applicazioni Mobile Ibride.**

Tuttavia molti sviluppatori javascript che si occupano di font-end fanno ancora fatica a prendere in considerazione l'idea di utilizzare **l'MVC** per i propri progetti, scoraggiati in particolar modo dallo "scoglio iniziale" in cui si imbattono quando si affacciano a questo tipo di programmazione.

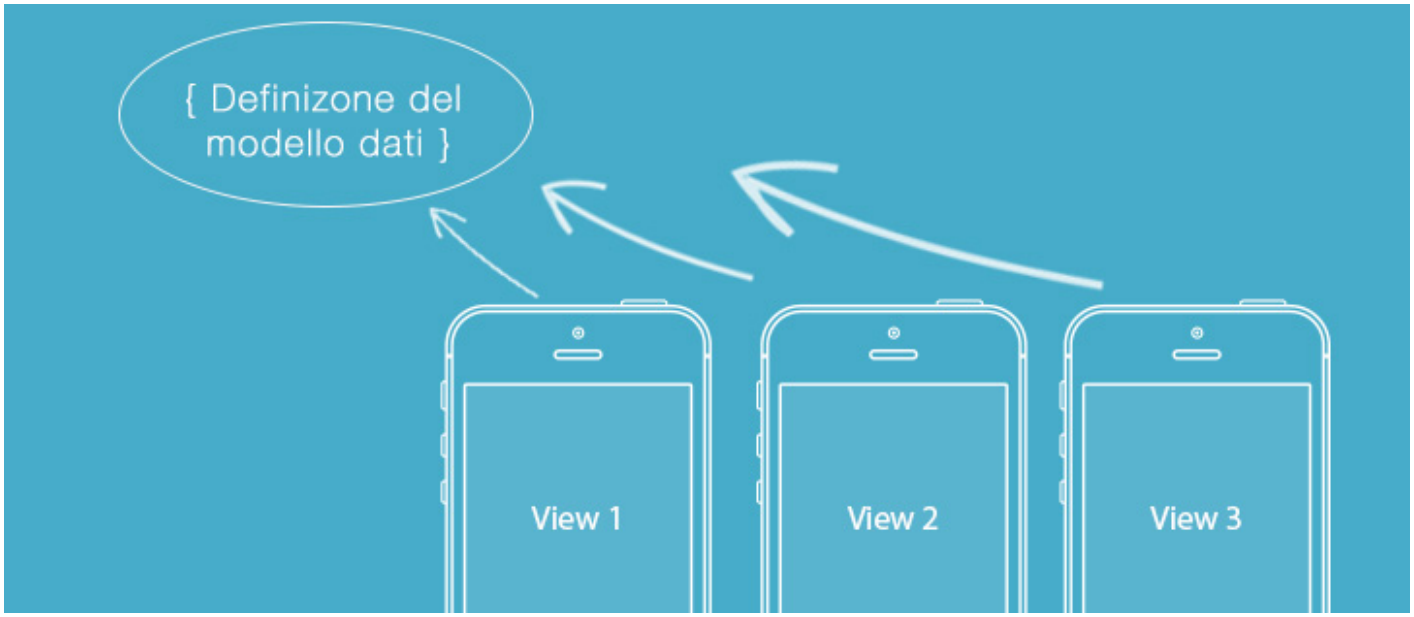
Tuttavia, se all'inizio può essere di difficile comprensione, soprattutto per coloro che sono abituati a lavorare principalmente con **normali siti web,** una volta che ne abbiamo compreso il funzionamento diventa difficile farne a meno (in questo articolo ho affrontato tale argomento in modo più approfondito).

Posso affermarlo per esperienza personale: dopo aver iniziato ad utilizzare l'MVC per le mie applicazioni mobile e single page app remote, ora, in qualsiasi progetto web (che si tratti di un sito web statico, un sito realizzato con un mio CMS personale, un tema Wordpress realizzato ad-hoc per il cliente ecc.) non posso fare a meno di includere il framework MVC Backbone.js.

Ma vediamo quali sono i tre benefici principali che derivano dall'utilizzo di un framework MVC.

Facilità a livello di manutenzione generale (manetnibilità)

Nel momento in cui si ha la necessità di apportare degli aggiornamenti all'applicazione, diventa molto più semplice capire dove e come intervenire, sia se gli aggiornamenti prevedono delle modifiche alla struttura dei dati, sia quando gli aggiornamenti riguardano interfaccia ed elementi visivi, caso in cui dovremmo modificare componenti come le view.



Credimi, sulla base delle oltre 10 applicazioni che ho sviluppato in HTML, CSS3 e Javascript, posso affermare con certezza che questo è un grandissimo vantaggio.

Nelle prime 3 o 4 applicazioni realizzate e distribuite al cliente non avevo considerato minimamente la possibilità di un pattern MVC per la loro architettura.

Quelle applicazioni mi danno grandissimi problemi ancora oggi, a distanza di 4-5 anni.

Pur avendo scritto ogni riga di codice personalmente, nei progetti in cui non ho utilizzato una logica di tipo MVC impiego il doppio del tempo di quello necessario per effettuare le stesse modifiche in progetti MVC-based.

Ogni volta bisogna passare al setaccio infinite righe di codice javascript o markup HTML, prima di sapere con esattezza il punto nel quale intervenire per apportare la modifica di cui abbiamo bisogno.

In certi casi può essere un fattore veramente frustrante.

Evitare la duplicazione di controlli e componenti.

Grazie al pattern MVC possiamo **evitare di riscrivere ogni volta lo stesso codice** per realizzare un controllo già utilizzato in altri casi: esso infatti rende più facile la riusabilità dei componenti.

In un progetto realizzato senza tenere conto di una logica ben definita (come quella fornita dall’MVC), pensare di riutilizzare un modello dati o il codice di una schermata in più parti

dell’applicazione risulterebbe quasi impossibile:

In tal caso dovremmo ricorrere a degli "adattamenti personalizzati”, utilizzabili solo e solamente per quella situazione.

Se poi volessimo utilizzare quel componente in un’ulteriore parte dell’applicazione, allora la cosa si complicherebbe maggiormente: oltre che i parametri impostati per la prima situazione e quelli per la seconda, dovremmo andare ad aggiungere anche quelli per la terza.

In una parola? Caos.

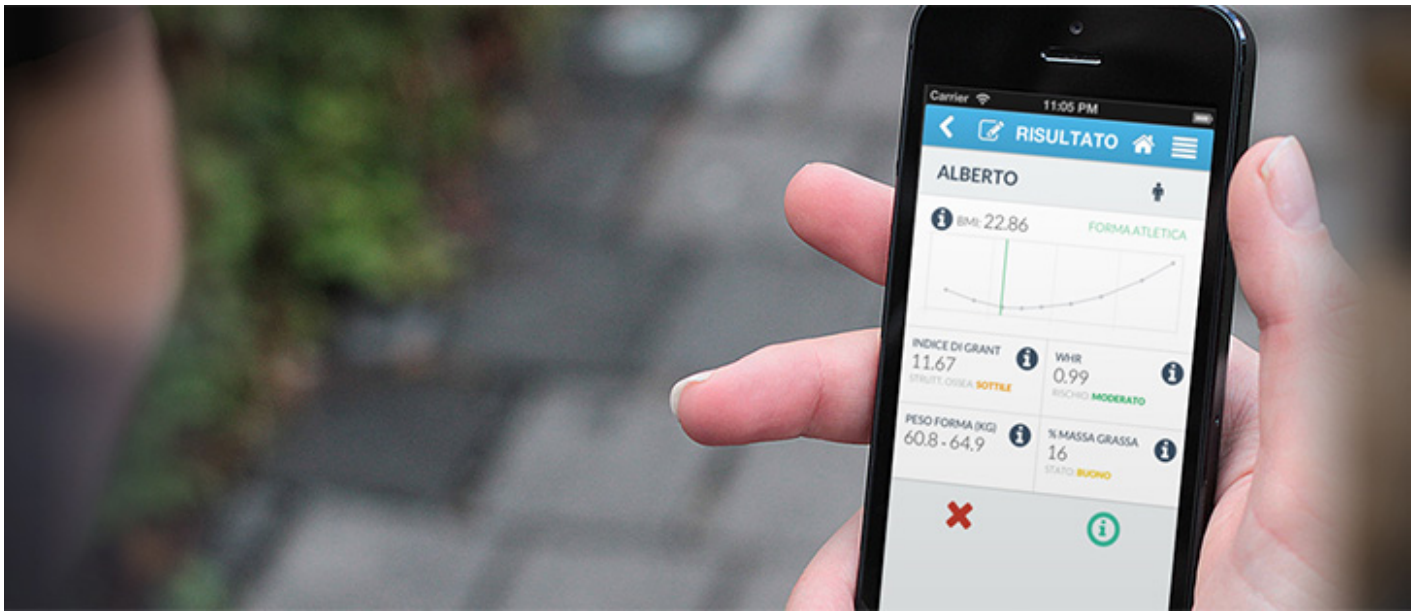
Fortunatamente la soluzione ci è fornita dai pattern MVC.

A cominciare dalla struttura dei dati.

Possiamo per esempio definire una volta soltanto, attraverso componenti MVC come *model* e *collection,* come dovrebbero comportarsi e quali caratteristiche dovrebbero avere determinati tipi di dati.

Un caso reale

Sulla mia applicazione per la dieta (Dieta SI o NO?), dove permetto di aggiungere, modificare o eliminare dei test sulla massa corporea per ogni individuo, ho definito comportamento e caratteristiche del tipo di dati “Test” in un model MVC, specificando che ogni volta che l’utente crea un nuovo test oppure inserisce determinate informazioni aggiuntive, vengano effettuati dei calcoli automatici per i risulati dal visualizzare successivamente in un’apposita view (o schermata).

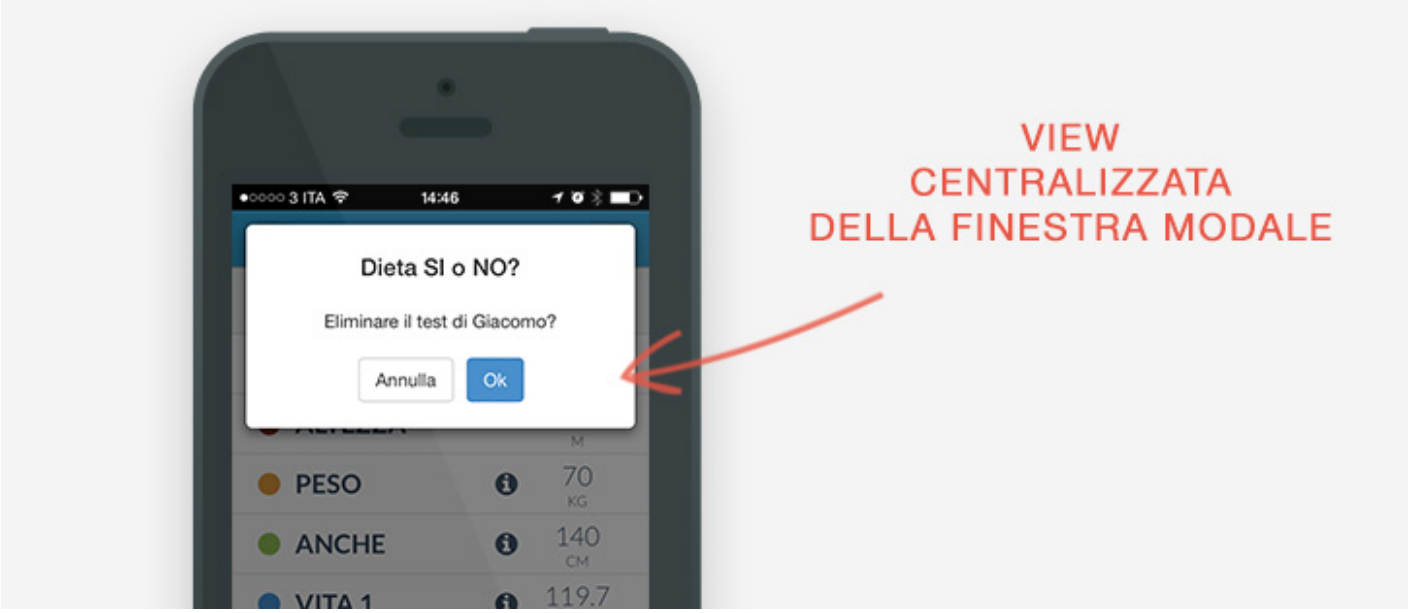


Schermata dell’applicazione Dieta Si o NO? che mostra i risultati di un test sulla massa corporea

In questo modo ho pootuto centralizzare la gestione dei test, ovvero crearne di nuovi e modificare i parametri di quelli già esistenti da qualunque parte dell’applicazione, senza preoccuparmii di ripetere ogni volta il codice necessario per calcolare i risultati, oppure richiamare la funzione che se ne occupa.

Tutto è definito nel model principale, e se voglio modificare il modo in cui vengono calcolati i risultati del test, mi basterà editare solamente il file javascript in cui quel model è stato definito.

Lo stesso funzionamento vale ovviamente per le view: possiamo ad esempio definire in modo centralizzato il comportamento e le caratteristiche di una view che mostra un messaggio informativo all’utente tramite una finestra modale, ed utilizzarla da qualsiasi parte o in qualsiasi altra view del nostro progetto.



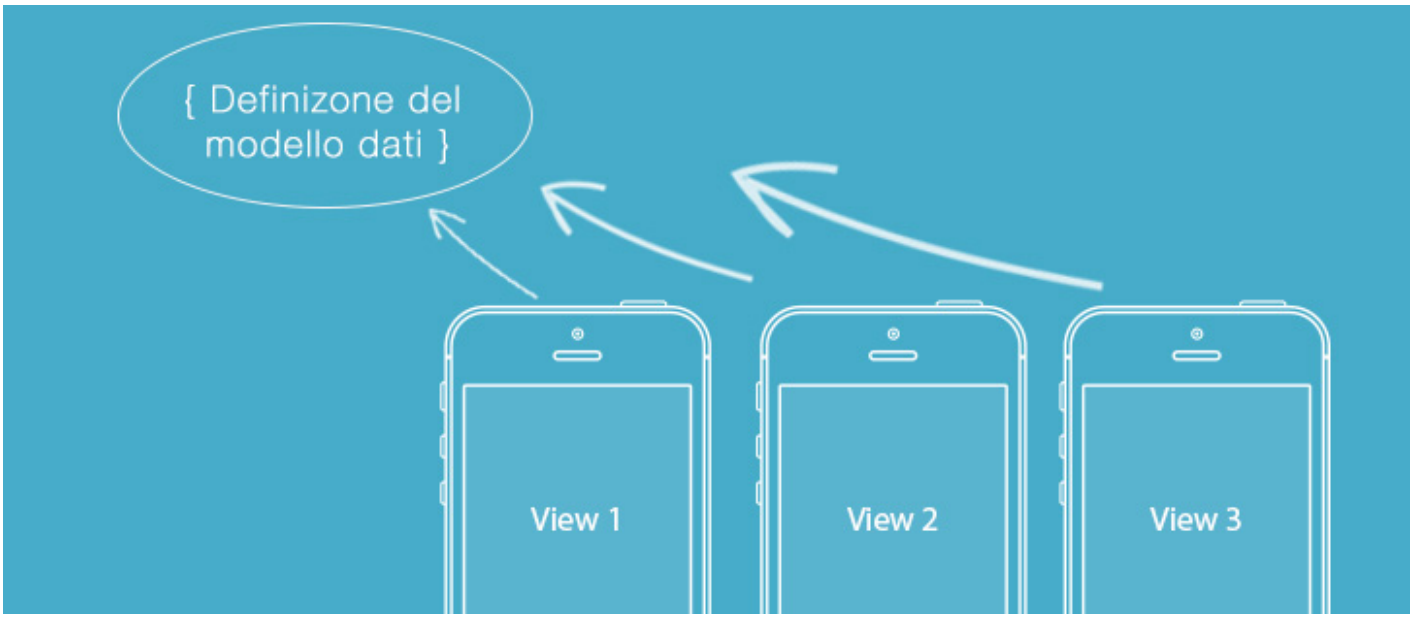
Su Dieta Si o NO? la view della finestra modale per i messaggi informativi può essererichiamata da qualsiasi parte dell’applicazione, senza ripetere

Se volessimo poi modificare l’effetto di entrata del messaggio basterebbe andare a modificare la definizione della view principale.

Tutto ciò è sinonimo di velocità di sviluppo accelerata.

Possibilità di lavorare simultaneamente nello stesso progetto.

In base alle dimensioni dell'applicazione e la separazione dei ruoli degli sviluppatori, la modularità (separazione dei vari componenti in moduli distinti a seconda dei compiti a loro assegnati) offerta dai vari framework MVC permette a più individui "addetti ai lavori" di editare allo stesso tempo diverse parti dell'applicazione.



Anche questo aspetto è sinonimo di un incremento della veolcità del flusso di lavoro: sviluppatori che si occupano della logica dei dati possono lavorare contemporaneamente a quelli che si occupano dell'interfaccia, oppure a più sviluppatori che si occupano dell'interfaccia possono lavorare allo stesso tempo sullo sviluppo di schermate diverse.

HTML Mobile App: perchè utilizzare l'MVC: conclusioni

Sebbene superare la fase di apprendimento iniziale rappresenti un ostacolo non indifferente, numerosi sono i benefici che possiamo trarre dall'utilizzo di un framework MVC, e in questo post ti ho mostrato quelli che secondo me spiccano rispetto agli altri.

Uno dei più grandi valori del pattern MVC è che esso rappresenta **il risultato dell'esperienza maturata da una grandissima comunità di sviluppatori** e software-engineer nel corso degli anni; si tratta pertanto di un modello di sviluppo adattabile a pressoché qualsiasi tipo di necessità, senza contare il fatto che tutti i framework MVC vengono costantemente aggiornati e mantenuti.

Crea la tua applicazione mobile

Utilizzando il pattern MVC insieme a molte altre tecniche ho creato per i miei clienti oltre 10

applicazioni mobile, tra cui due progetti personali: Tint Weather App e Dieta SI o NO?.

Se anche tu sei un web designer e vuoi realizzare la tua applicazione mobile da distribuire nei vari stores, forse sarai interessato al mio e-book: HTML Mobile Accelerato.

Grazie ad esso verrai a conoscenza di tutte quelle tecniche che renderanno la tua mobile app fluida e reattiva come quelle native.