Obelix Group
obelixswe@gmail.com

Verbale esterno 2017-05-03

Versione	*v1_0_0*
Data creazione	2017/05/03
Redattori	Emanuele Crespan
	Federica Schifano
Verificatori	Riccardo Saggese
Approvazione	Nicolò Rigato
Stato	Approvato
Uso	interno
Distribuzione	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
	Red Babel
	Gruppo Obelix

Sommario

Verbale dell'incontro tra il gruppo *Obelix* e il proponente *RedBabel* in data 2017-05-03

1 Informazioni sulla riunione

Data: 2017-05-03

Luogo: Residenza membro gruppo Obelix e sede Red Babel - videoconferenza

Ora: 18:00

Durata: 40'

Partecipanti interni: Obelix

Emanuele Crespan
Federica Schifano
Nicolò Rigato
Riccardo Saggese
Silvio Meneguzzo
Tomas Mali

Partecipanti esterni: Red Babel

Milo Ertola

2 Conversazione

Di seguito vengono riportate in grassetto le domande e affermazioni effettuate dal gruppo Obelix nel corso della videoconferenza e in corsivo le risposte e osservazioni date dal Proponente Red Babel.

Innanzitutto volevamo aggiornarvi su quello che stavamo facendo e pensando. Siamo nella fase di progettazione architetturale, in realtà siamo molto confusi. Vi dico quello che abbiamo fatto fino ad adesso...
L'ultima volta che ci siamo sentiti vi eravate appena formati, abbiamo discusso di come trovare l'idea per il progetto, giusto?? Come è andata?

abbiamo fatto la prima presentazione con una serie di idee che erano forse molto abbozzate, ce l'ha fatta passare con una serie di note e correzioni da apportare.
Potete mostrarci la presentazione? L'idea con cui siete venuti fuori?

Per quanto riguarda l'SDK abbiamo una parte più verso la grafica e una parte di altro, di funzionalità.
Dunque, il progetto è diviso in due parti: uno il cosiddetto contenuto tecnologico quindi l'SDK, lo strumento che utilizzano gli sviluppatori per venir fuori con le loro app e l'altro è l'utilizzo di questo SDK. Dunque l'SDK come è suddivisa?

Una parte è il front-end che è pensata più che altro per semplificare la composizione, abbiamo pensato ad un sistema di elementi e

contenitori inspirati al modo in cui funziona QT nel C++ o alle swing del Java. I contenitori contengono o contenitori o elementi, i contenitori si occupano di impilare o di accostare in orizzontale le cose (per realizzare il layout), abbiamo fatto una cosa così componibile con misure e dati tutti di default per fare tutto nel modo più automatico e veloce possibile.
Benissimo, ma se lo sviluppatore vuole può fare delle customizzazioni un po' più spinte, giusto?

Certo! Copriamo gli elementi dell'HTML principali in modo molto superficiale. Ad esempio se abbiamo un <p> (paragrafo) noi curiamo solo il contenuto, per il resto abbiamo pensato di dare la possibilità di riferirsi a un id o alle classi dell'HTML e se lo sviluppatore vuole si arrangia.
Una cosa, non so a che livello avete progettato, ma se dovete avere altre ispirazioni ci sono dei tool che funzionano più nella direzione di React o quantomeno più legate all'HTML e CSS. Sapete cos'è bootstrap?

si, sappiamo cos'è.
Bene, in bootstrap c'è un sistema di greeding cioè una griglia senza cazzi e ramazzi, già impostata, vi consiglio di guardare e prendere spunto da cose che ci sono già in giro, non reinventare ovviamente questo nei limiti del framework Meteor e Rocket.chat.

Perfetto, anche se in realtà questa è una delle cose che ci blocca di più perchè facciamo fatica a capire che strumento fa cosa..
Cosa non vi è chiaro più in particolare?

Ad esempio come attaccarsi a Rochet.chat, cioè noi facciamo il nostro SDK e dobbiamo dare e ricevere delle informazioni da rocket.chat..
Esatto, Meteor è un sistema monolitico che utilizza Javascript sia su front-end e sia su back-end quindi il nostro SDK avrà una parte che girerà su fron-end quindi scritta in javascript e una parte che girerà nel back-end che può essere anche solo anch'esso scritta in javascript..Voi avete visto come funziona Rochet-chat e cosa fornisce?
A livello di..?
Ad esempio se vi permette di usare bootstrap?

Si, ci siamo informati un po'. Però non ci è ciaro come attaccarci a Rochet.chat..Cioè noi facciamo il nostro SDK e dobbiamo avere delle informazioni e dare delle informazioni a Rocket.chat.
Esatto, Rochet.chat è un'applicazione che gira su meteor, ed esso necessità dei package, quindi quando avete messo il vostro package nella vostra istanza di Rocket.chat, in sostanza uno script meteor che utilizza le API di Rochet.chat .

Ok, andremo subito a guardarlo.
Per saperne di più andate a vedere i pacchetti che ttrovate nella Repo di Rochet.chat, proprio il codice. I pacchetti che ho utilizzato per il mio esempio sono ad esempio gli action link per vedere come si poteva avere dei link azzionabili

all'interno della bolla.

Ok. Questo è già di aiuto su molte cose. Un altra cosa, è che nel capitolato si parlava di SCSS..
Si, SCSS come avete visto, SCSS compila un css e se scrivi css il css è scss valido. Vi permette di risparmiare lavoro con il css, inoltre SCSS è integrato in meteor.

Perfetto, questo è chiaro, inoltre riguardo React, angular c'è qualche consiglio o avvertimento utile che puoi fornirci?
Di default Metor utilizza Blaze.js, ma comunque vi consiglio React o comunque è a vostra discrezione la scelta. Scusate, ma tra poco devo andare, per la Demo? cosa avete pensato?

Ok, quindi dato che dobbiamo accellerare la demo era pensata come una versione più evoluta di una to do list..ad esempio posta in una situazione come magazzino/dispensa c'è un elenco di cose che devono esserci, quindi la bolla permette di spuntare da liste predefinite le cose che ci sono e quelle che non ci sono, quello che non c'è viene automaticamente messo in lista ev in quella lista si possono aggiungere ulteriori cose che non ci sono nelle liste di controllo.
Ok, quindi ci sono almeno due tipi di utente?

Si, ci sarà un Mittente e un ricevente.
Ottimo. Come consiglio per l'SDK se riuscite a farvi il prototipo avete già un idea di come buttare giù gli schemi. Io adesso devo andare, se c'è altro scrivete su slack.

Va bene, Grazie mille.

3 Decisioni prese

1. Confermato struttura Demo, approvata l'idea da parte dei proponenti
2. Uso di bootsrap
3. valutazione tra Blaze.js, React e Angular con scelta finale di React
4. Affermate due tipologie di utente per la Demo (mittente e ricevente)