**JIFERAY CE V1.0.0**

Jiferay CE - Community Edition - è un Portale orientato alla costruzione rapida di applicazioni web Enterprise, basato sulla tecnologia tecnologia "Angular Single-Page" ; esso nasce dalle esperienze di TPL informatica in Node.JS, Express.JS, Angular.JS e MySql

Abbiamo scelto la piattaforma Angular/node, in alternativa a java, per la sua estrema snellezza, e velocità di sviluppo.

Abbiamo notato da questa esperienza la necessità di una standardizzazione dello sviluppo SW, per aumentare la mantenibilità del codice, la velocità di sviluppo, unitamente con la affidabilità globale dell'applicazione.

Abbiamo utilizzato la tecnologia "Angular One-Page" per costruire la migliore e lo più efficace Digital eXperience possibile per l’utente, costruendo così una DXP (Digital eXperience Platform) particolarmente interessante sia per l’utente che per gli sviluppatori che approntano le applicazioni.

La tecnologia permette anche un controllo puntuale del codice e degli aggiornamenti dei vari moduli client/server.

L'idea è di costruire una community di Developers che adottano gli stessi standard di sviluppo per applicativi aziendali e IoT, per poter condividere tali applicazioni su un MarketPlace di prossima costituzione.

In tal modo la community può avere a disposizione moduli applicativi di tutti i partners per costruire in tempo limitato portali anche complessi, di tipo enterprise.

# LA VERSIONE JIFERAY CE V1.0.0

Come detto, JIFERAY utilizza Angular Single Page, una tecnologia che uilizza una sola pagina html (index.html) , suddivisa in

* header,
* ui-view centrale (<ui-view></ui-view>) in cui vengono caricate via via le pagine dell’applicazione, e
* footer (collocati nei moduli headerModule, homePagesModule, footerModule del Client);

L’ **header** contiene la navigation bar, il **footer** è fisso, e nella **ui-view centrale** vengono caricate le pagine dell’applicazione, cliccando il link corrispondente alla variabile *nomemoduloClient* in $stateProvider.state(“<*nomemoduloClient*>”);

# SERVER

il server nel Jiferay CE V1.0.0, in Nodejs , è composto dai file:

* ./app.js
* ./routes/users\_<nomemodulo>.js
* ./node\_modules
* ./bin/www
* ./package.json

ove il <nomemodulo> corrisponde con il nome del modulo, per esempio: users\_mysql.js

il modulo può corrispondere anche ad una applicazione web, in tal caso vi è pure un modulo sul Client con lo stesso nome , vedi oltre.

I moduli **users\_<nomemodulo>.js** vengono referenziati in app.js con le due istruzioni:

require(“”) e

app.use()

Quindi il server risulta essere costituito da servizi tutti connessi ad app.js , ciascuno dei quali specializzato in base alle funzionalità comuni di base, ovvero alle singole applicazioni.

# CONNESSIONE A DATABASE

Il collegamento a DB MySql viene effettuato con parametri di connessione descritti in variabili globali nel modulo **users\_mysql.js**

Qualsiasi chiamata a DB, dal server e dai servizi rappresentati dagli endpoint chiamati da Post Angular del Client, vengono fatti nei moduli applicativi server **users\_<nomemodulo>.js** utilizzando i parametri globali dichiarati nel modulo users\_mysql.js , aprendo una connessione , effettuando le query richieste, e chiudendo sempre la connessione dopo tali query.

Il backup del DB si trova all’interno dell’applicazione, nominato “jiferay.sql”;

per ripris

# CLIENT

Il Client è costituito dalle pagine che vengono alimentate nella ui-view della Single Page di Angular Single Page , la pagina **index.html**.

Il Client è composto dai seguenti file / cartelle:

**./views/index.html**

**./public/javascript/content**

che contiene i file .js delle librerie Client (Bootstrap, Angular, Api varie);

**./public/javascript/modules/**

che contiene le cartelle dei moduli di base (contenenti i relativi file .html e .js)

**./public/javascript/modules/headerModule**

**./public/javascript/modules/footerModule**

Inoltre contiene le cartelle dei moduli applicativi (ognuno con i relativi file .html e .js)

tutti nella cartella:

**./public/javascript/modules/homePageModules/**

il file:

**./public/javascript/ClientApp.js**

Descriviamo le funzioni e i contenuti di tali file:

**./views/index.html**

questa è la Single e unica Pagina del Portale, nella quale vengono dichiarati:

* tutti i **file .js e .css;**
* il  **file clientApp.js**, il punto di partenza di tutte le applicazioni Client:

./public/javascripts/**ClientApp.js**

nel quale **sono dichiarati tutti gli “*angular.module('<moduloClient>', []*)** ,

presenti **nel file homeModule.js** di ciascun modulo applicativo Client:

./public/javascript/modules/homePageModules/<modulopplicativoClient>/

**homeModule.js**

inoltre in **index.html** sono definiti

* **l'header** , la navigation bar nella barra superiore, usato dal headerModule in

**./public/javascript/module/headerModule**

* la **ui-view** , la fascia orizzontale centrale, in cui pubblicano le loro pagine i moduli applicativi presenti in:

**./public/javascript/module/homePagesModules/<nomemoduloapplicativo>**

* e il footer nella barra superiore dello schermo, **utilizzato** dal footerModule in

**./public/javascript/module/footerModule**

# I MODULI APPLICATIVI LATO CLIENT

Nella cartella di un modulo applicativo client sono presenti i file:

* homeModule.html
* homeModule.js
* homeModule.css

il file **homeModule.html** contiene i tag della pagina da visualizzare: esso non richiede i tag <html><head><body> , in quanto è un’estensione del index.html, ove sono gia presenti i tag.

il file **homeModule.js** contiene invece il codice javascript:

angular.module('ContattaciModule', [])

.config(function($stateProvider) {

$stateProvider.state('Contattaci', {

url: '/Contattaci',

controller: 'ContattaciController',

templateUrl: '/javascripts/module/homePagesModule/Contattaci/homeModule.html',

jp\_app\_name: 'Contattaci',

Jp\_app\_version: 'v1.0.0',

Jp\_app\_menu: 'Contattaci'

});

})

angular.module('ContattaciModule')

.controller('ContattaciController', function($scope, $http, fileReader, $sce, $state, $rootScope) {

...

(codice applicativo in Angular/javascript)

...

}

in cui si osservano le due direttive **angular.module**:

**angular.module(**'ContattaciModule', [])

**.config**(function($stateProvider) {

**$stateProvider.state**('Contattaci', {

in cui

* **il nome del modulo 'ContattaciModule'** deve essere **unico** nell’applicazione, e viene riportato nel file C**lientApp.js ,** e lo
* ***$stateProvider.state*** **'Contattaci'** pure **unico** nell’applicazione, rappresenta il link da chiamare per visualizzare la pagina del modulo, eseguendone il codice javascript.

e

**angular.module**('ContattaciModule')

.**controller**('ContattaciController',

in cui

* **il nome del controller 'ContattaciController'** deve essere **unico** nell’applicazione.

All’interno del controller viene scritto il codice javascript da eseguire nell’applicazione, all’attivazione della pagina.

# FUNZIONALITA’ DI Jiferay CE V1.0.0

Jiferay CE V1.0.0 comprende le seguenti funzioni:

## Gestione profilo e autenticazione:

-signup: registrazione al sito di un nuovo account con convalida tramite mail;

-login

-personalizzazione profilo, quale immagine, dati e informazioni contatto

## Gestione utenti/gruppi utenti

è presente un pannello di amministrazione, raggiungibile dopo aver effettuato l'accesso con un account amministratore,

il quale presenta due sezioni:

la prima, di modifica degli account, permette di:

-attribuire ruoli ai vari account(moderatore,webMaster,Admin,tecnico,etc);

-la modifica dell'account(username,password);

-l'eliminazione del suddetto;

la seconda consente la gestione dei gruppi di account, in particolare, selezionandone uno,

vi è la possibilità di aggiungere, modificare o rimuovere pagine alle quali

tale gruppo potrà accedere, tramite un semplice menù Drag & Drop che presenta le seguenti sezioni:

-una griglia, che consente il posizionamento, e quindi la visibilità delle pagine per il gruppo;

-in alto presenta il blocco "nuova voce", che trascinata nella griglia, consente l'inserimento di un nuovo modulo;

-a destra è presente un elenco dei moduli client-side presenti all'interno dell'applicazione;

trascinandone uno nel blocco precendentemente inserito, il modulo sarà attivato. ora basta solo salvare sul fondo per renderlo attivo.

## Gestione contenuti Linguistici

Si tratta di un modulo che consente di modificare i contenuti letterali all'interno dell'app senza andare a modificare codice,

creare nuove pagine in altre lingue ecc...

Perche sia attivo, è necessario che i contenuti letterali nelle pagine HTML siano chiamati da una funzione,

ES: {{ lr("in amicizia, fiducia e lealtà!") }}

si tratta di una funzione Angular(la si riconosce dal {{ }} ) chiamata $scope.lr e provvede a sostituire tutti i contenuti letterali

con la loro variante in un altra lingua, facendo una ricerca su DB.

il modulo "reslang" è un interfaccia che permette:

-personalizzazione di contenuti linguistici(frasi,parole,ecc)

-modifica e correzione in tempo reale dei contenuti

il tutto in tempo reale

## CMS Content Management System

La personalizzazione delle Pagine HTML dei singoli moduli applicativi client può essere fatta direttamente dalla medesima tramite apposito pannello.

Accedendo al portale come Webmaster o Amministratore, nelle pagine vetrina del menu Home dedicato ai visitatori non registrati, sarà visibile sotto al contenuto della pagina, un pannello che permette di:

**-modificare** la pagina HTML in maniera diretta, senza accedere al codice sorgente;

-**effettuare upload di file di foto,di file HTML** dal computer locale sul browser per l'utilizzo immediato nella propria pagina, con possibilità di salvarlo nel Repository, nel server, e possono essere utilizzati anche in altre pagine;

-**accedere al Repository su server**, contenente file e foto ivi caricati,per riprendere file e foto e modificarli;

-**pubblicare la pagina di cui si è fatto l’upload**. In tal caso per vederla occorre fare il Refresh dell’applicazione.