Pour créer l’app :

Express nomDelApp

On **ajoute** au package.json ( attention au formatage du texte avec word)

//On va ajouter nodemon pour le restart du serveur au save du fichier(nodemon est déjà installé globalement sur ma machine)

// on ajoute

},

”main”: “bin/www”,  
"dependencies": {

// Pour le cryptage pwd

"bcryptjs":"\*",

//Pour les messages flash

"connect-flash":"\*",

//Pour les sessions  
"express-session":"\*",

//Pour le templating

"express-handlebars": "\*",

//Pour la validation des form

"express-validator":"\*",

//Pour les messages flash ( a besoin de dépendances tels que connect-flash  
"express-messages":"\*",

// Pour le data modeling  
"mongoose":"\*",

/ Pour le login register

"passport":"\*",

//Pour le developpement en local  
"passport-local":"\*",

// Pour le login avec FB

"passport-facebook":"\*",

La suite c’est :

Installer les dépendances :

npm install

(J’ai sur ma machine d’installé en global « nodemon »)

On lance l’app avec

nodemon

On a une erreur car on veut utiliser handlebars et non jade qui est le moteur de vue par défaut(on a supprimé cette ligne dans package.json)

Il faut donc aller dans **app.js**

Faire appel à handlebars

**var** expressHandleBars= require('express-handlebars');

changer

app.set('view engine', 'jade');

en ça

app.set('view engine', 'handlebars');

et ajouter ça

app.engine('handlebars',expressHandleBars({defaultLayout:'layout'}));  
pour avoir ça :

// view engine setup  
app.set('views', path.join(\_\_dirname, 'views'));  
app.engine('handlebars',expressHandleBars({defaultLayout:'layout'}));  
app.set('view engine', 'handlebars');

Renommer les vue jade dans views pour avoir l’extension : .handlebars

Créer un dossier layouts dans le dossier views

On déplace le fichier layout.handlebars dans le dossier layouts

Dans app.js on ajoute les middleware

**var** expressValidator=require('express-validator');

**var** flash= require('connect-flash');  
**var** session= require('express-session');  
**var** passport= require('passport');  
**var** mongoose= require('mongoose');

On va maintenant faire le schema de la BDD

On crée à a racine un dossier config, qui contiendra les fichiers de passport pour les différentes stratégies

On y ajoute aussi un fichier db.js qui contiendra l’url de mongodb

Dans app.js on ajoute

**var** configDB= require('./config/db.js');  
mongoose.connect(configDB.url);

On se rend sur

<https://github.com/expressjs/express-messages>

pour récupérer le snippet

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Pour les messages flsh  
// Crée une variable globale "messages" qui peut etre utilisée dans les vues !  
//EXEMPLE  
//Adding Messages  
//  
//On the server:  
//  
//req.flash("info", "Email queued");  
//req.flash("info", "Email sent");  
//req.flash("error", "Email delivery failed");  
//  
app.use(require('connect-flash')());  
app.use(**function** (req, res, next) {  
 res.locals.messages = require('express-messages')(req, res);  
 next();  
});  
// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Pour les messages flsh  
//FIN

Que l’on colle dans app .js

On fait pareil pour express-validator sur

<https://github.com/ctavan/express-validator>

// Se charge des erreurs de validation  
// In this example, the formParam value is going to get morphed into form body format useful for printing.  
app.use(expressValidator({  
 errorFormatter: **function**(param, msg, value) {  
 **var** namespace = param.split('.')  
 , root = namespace.shift()  
 , formParam = root;  
  
 **while**(namespace.length) {  
 formParam += '[' + namespace.shift() + ']';  
 }  
 **return** {  
 param : formParam,  
 msg : msg,  
 value : value  
 };  
 }  
}));

On declare l’utilisation de session

Sur : https://github.com/expressjs/session

// Pour les sessions  
app.use(session({  
 secret: 'keyboard cat',  
 resave: **false**,  
 saveUninitialized: **true**}));

On test pour voir si tout va bien, sans oublier de lancer mongod

Tout va bien jusque ici………

On continue et on ajoute le binding des erreurs dans le template layout.handlebars

{{{messages}}}

Vient de la variable globale crée un peu plus tot dans app.js

Maintenant les erreurs pour la validation des form viendront du middleware expressvalidator, il contient une variable msg

Donc dans le template layout

{{#**if errors**}}  
 {{#**each errors**}}  
 {{**msg**}}  
 {{/**each**}}  
{{/**if**}}

**Mise en place front end :**

Je fais un bower init

bower install bootstrap --save