# Présentation Orale projet MonDressing

## Présentation de l’entreprise et/ou du service et contexte du projet (cahier des charges, environnement humain et technique)

* Léa Delannay
* Dernière promotion Développeur Web et Web Mobile
* Présentation d’un projet personnel car le contenu de mon stage ne permettait pas de couvrir toutes les compétences requises pour le titre
* Projet nommé MonDressing
* Choix de réaliser ce projet car c’est quelque chose que je souhaite faire évoluer dans le temps, durant mes études (intégration d’intelligence artificielle notamment – machine learning sur les habitudes et les gouts des utilisateurs)
* But :
  + 1e version : avoir visibilité sur vêtements enregistrés, et les gérer
  + 2e version : générer aléatoirement des tenues en fonction de certains critères définis par utilisateur tels que couleur ou occasion
* Cible : hommes et femmes qui souhaitent déléguer le choix de leur tenue
* Obj quantitatifs : 5 bêta-testeurs dans un premier temps. Pas destinée à être vendue.
* Contraintes techniques : hébergement en local dans un premier temps, site responsive.
* Fait en Angular 7, Bootstrap 4, NodeJs, Express, MySQL, VSCode, Xampp
* 8 pages et 2 modales :
  + page d’accueil,
  + page de création du compte,
  + page de connexion,
  + page de création d’un vêtement,
  + page de création d’une tenue,
  + page de visualisation de tous les vêtements,
  + modale de visualisation d’un vêtement en particulier,

page de modification d’un vêtement,

* + modale de suppression d’un vêtement,
  + page de météo(module ext).
* Si ça vous intéresse, charte graphique, moodboard et maquettes graphiques en annexe
* Livrables : conception, développements, tests, déploiement et maintenance.
* User stories et spécifications techniques correspondantes en annexe.

## Conception et codage des composants front-end et des composants back-end

* Base de données
  + Réflexion et analyse des processus de l’application souhaotée sur papier
  + MCD MLD en annexe, faits sur Power AMC
  + SQL généré, puis modifié ex : clés étrangères pas les mêmes que clés primaires, ajout des instructions SQL Create database if not exists, use dressing, drop table if exists caracteristiques etc pour éviter éventuelles incohérences
  + Intégré le code SQL dans l’onglet SQL de phpMyAdmin
  + Ajout de jeux de données pour tester l’appli : via phpMyAdmin et via le code SQL
* Front et back
  + Création d’un compte :
* Récupération des données saisies par utilisateur: data binding dans le composant Angular, variable dans .ts, directive ngModel dans html
* Au clic sur le bouton s’enregistrer, lance une fonction OnRegister qui vérifie @, crée objet JSON contenant données formulaire, appel au service, souscrit au retour du service, puis lors du retour du service contenant le code http retourné par le serveur, data binding et directive structurelle ngIf avec html pour affichage d’un message (erreur ou ok) à destination de l’utilisateur
* Service fait appel à ressource mise à dispo par serveur. Requete http en post car création, id inexistant au préalable, et je souhaite appliquer les principes REST (representational state transfer, basé sur http (hypertext transfer protocol), je considère les éléments sur mon serveur comme des ressources).
* Afin que données qui transitent entre le front et serveur chiffrées, installer sur serveur certificat https permettant d’utiliser protocole https, puis lier au site web.
* Sur le serveur : création de la route / point d’entrée
* Données du front envoyées à la ressource users sur serveur. Ressource joue la fonction router.post /. Fonction vérifie que données attendues sont bien dans l’objet Json envoyé par front (Si tout est ok, renvoie code http 200 OK, si NOK, envoie code 400 bad request.), puis fait appel à fonction createUser qui gère l’appel à la BDD. Si ok, renvoie code http 200 OK, si NOK, envoie code 500 Internal Server Error.
* La fonction createUser dans dbaccess/index.js fait la requete insert into sous la forme d’une requête préparée pour éviter injections sql. Si ok, renvoie code http 200 ok, si NOK, envoie code 500 Internal Server Error. Le mdp est stoché hashé en sha256.
  + Connexion :
* Récupération des données comme précédemment
* Au clic sur bouton se connecter, fonction onLogin se lange, crée objet JSON dans lequel je mets les données récupérées du formulaire pour envoi au serveur
* Appel au service en lui passant l’objet créé, abonnement au retour du service
* Lors du retour du service, récupère le token JWT généré par le back que je stocke dans le localstorage jusqu’à ce que l’utilisateur se déco
* Récupère code http retourné par serveur et renvoie utilisateur vers page accueil si code 200ok, affiche un message d’erreur si code http 500 Internal Server error ou 403 Forbidden
* Service appel a la ressource en post car je dois envoyer les infos de l’utilisateur au serveur pour pouvoir les comparer aux infos présentes en bdd
* Données envoyées à ressource users, route définie, sur router.post /login/. Cette fonction renvoie 500 internal server error si erreur, si ok vérifie que que données attendues sont bien dans l’objet (sinon 400 Bad Request), puis fait appel a fonction dans index.js readUser qui joue la requete sql
  + La 1e requete vérifie que login saisi existe bien en bdd
  + 2e requete récupère id et mdp en bdd
  + 3e requete permet de comparer le mdp transmis avec le mdp en base.
* Toutes les requêtes SQL sont préparées afin d’éviter les injections SQL
* Si NOK, renvoi code http 403 forbidden
* Si ok, serveur crypte l’id dans un token JWT (grâce à librairie JWT), renvoie code http 200ok + token.
  + Création d’un vêtement :
* Fait de la même manière que le user mais le service appelle la ressource clothes
* Possibilité de créer des couleurs (ressource colors), catégories (ressource catégories), marques (ressource brands), caractéristiques (ressource features), occasions (ressource occasions)
* Concernant les couleurs, occasions et caractéristiques relatives à un vêtement, le traitement est un peu spécial car un vêtement peut en avoir plusieurs. Dans le ts, je récupère les couleurs, caractéristiques, occasions grâce à un get sur lequel je filtre uniquement les éléments cochés et pour lesquels je ne récupère que les ID. Côté back, pour insérer ces tableaux d’éléments en base, j’itère dessus grâce à une boucle forEach.
  + Modification d’un vêtement :
* La gestion du formulaire et récupération est la même que précédemment
* Ce qui change :
* A l’affichage du formulaire, il est rempli avec les données provenant de bdd grâce au binding bidirectionnel directive ngModel. A l’initialisation de la page j’appelle le service qui fait lui-même appel à la ressource clothes en get (car c’est juste de la lecture, n’envoie pas d’objet). La ressource appelle la fonction readSpecificClothes avec l’id en paramètre. Cette fonction fait un select dans la bdd pour récupérer toutes les informations concernant le vêtement concerné. Toutes ces informations sont ensuite transmises au front dans la variable clotheDetail, liée au html grâce à ngModel
* Etant donné le binding bidirectionnel entre html et ts sur variable clotheDetail, toute modification est prise en compte dans la variable, et c’est cette même variable qui est renvoyée au back lors de la soumission du formulaire.
* Côté back, cela joue la fonction router.put car il s’agit d’une modification selon les principes REST. Cette fonction jour la fonction updateClothe qui va mettre à jour la table vetement, et supporimer le contenu des tables associatives vet\_caract\_assoc, vet\_coul\_assoc et vet\_occas\_assoc pour les reremplir avec les données transmises par le front
  + Visualisation et filtres :
  + Suppression d’un vêtement :

## Présentation des éléments les plus significatifs de l’interface de l’application

## Présentation du jeu d’essai de la fonctionnalité la plus représentative (données en entrée, données attendues, données obtenues) et analyse des écarts éventuels

## Présentation d’un exemple de recherche effectuée à partir de site anglophone

## Synthèse et conclusion (satisfactions et difficultés rencontrées)