Angular.JS

date	révision
Août 2018	Création (v.38)
01/10/2018	Ajout utilisation Firefox recommandé (Chrome analyse une attaque Cross Origin)
18/07/2019	Ajout de fonctionnement général
20/09/2019	Correction <body> exemple 3.2.1
20/09/2020	V50 Correction p.7 (type=text)
25/06/2021	Amélioration compréhension CDN
08/09/2022	Intégration NodeJS/npm et mise à jour AngularJS déprécié
19/09/2023	Ajout des pages d'aides (démarche)

TABLE OF CONTENTS

1 Projet Angular.JS

Pour vérifier la bonne compréhension des concepts d'AngularJS, nous allons réaliser une petite application, de type 'Todo list'.

2 ToDo list

L'idée est de permettre à un utilisateur de se créer une liste de tâches dans une page web.

Vous devez donc imaginer un système permettant la Création, Lecture, Mise à jour et suppression de tâches dans cette page.

1. Imaginez un design d'application (Kanban ? Simple liste ? Verticalement ou horizontalement?)
2. Définissez le contenu de votre liste (tâche uniquement ? Un objet contenant la tâche, la priorité, la zone du Kanban ?)
3. Envisagez la structure du code (des vues ? Un contrôleur ou plusieurs ? Utilisation d'une classe de données ?)
4. Évaluez la durée de votre projet (en heure : 1, 2, 3, 5 ou 8 heures?)
5. Échangez avec les autres étudiants (stand-up)
6. Programmez
7. Testez (déboguez)

Pour ajouter de la complexité (mais aussi de la maîtrise), on peut imaginer que l'application compte plusieurs pages (gestion du routage). Par exemple, une page pour les tâches en cours et une pour les tâches terminées.

Vous pouvez vous aider d'Internet mais ne copiez pas une solution toute faite, ce serait dommage.

3 Démarche projet (aide)

Voici quelques conseils, pour démarrer plus facilement votre projet et éviter la "page blanche".

3.1 Définir l'apparence de l'application

En effet, en dessinant votre application, vous allez placer des champs, des boutons et des éléments qui vous permettront de déterminer les données à gérer.

N'ayez pas peur d'annoter vos dessins, en expliquant le comportement dynamique :

- Zones sensibles au survol (mouseover)
- Légende (couleur des champs, par exemple, pour la priorité ; choix d'icônes, etc.)
- Éléments cliquables et limitation de valeurs de champs

Faites un dessin par vue (les différents écrans et fenêtres que verra l'utilisateur)

Une méthode complémentaire est de tracer un diagramme d'utilisation UML.

3.2 Préparer les données

Aidez-vous des mockups (dessins faits précédemment) pour trouver les données utiles de votre application.

Si vous avez un champ catégorie dans votre ToDoList, cela implique de créer une classe Tâche qui contiendra un champ catégorie.

Profitez de ce travail pour tracer un diagramme de classe UML.

3.3 Préparer le projet

Choisissez la technologie à utiliser : ici, c'est de l'Angular.JS mais dans un projet réel, les décideurs font parfois un SWOT pour décider de l'opportunité de choisir une technologie innovante.

Imaginez ensuite la durée de votre projet, en prenant en compte les éléments suivants :

- Configuration de l'environnement (NodeJS ? NPM ? Création d'un serveur ? Etc.)
- Durée de création de la première vue (écran d'accueil)
- Durée de création des données d'une classe et de leur gestion (CRUD)
- Durée supplémentaire pour chaque nouvelle vue (et les CRUD associés)
- Durée d'auto-formation (vous aurez besoin de lire de la documentation, regarder des vidéos, tester des bouts de codes).
- Durée des tests (vérifier que tout fonctionne bien, testez des mauvaises valeurs...) : dans cette étape, on compte la création de valeurs pré-remplies pour tester toutes les conditions. Le terme est "jeu de test".

3.4 COMMUNIQUER SUR LE PROJET

Dans cette étape, l'échange avec d'autres personnes permet de vérifier que chacun à bien compris les attentes du client et qu'il n'y a pas plusieurs interprétations possibles.

Pour formaliser cela, on peut créer les diagrammes d'utilisation et diagrammes de séquences en UML.

Il faut être ouvert d'esprit et ne pas critiquer inutilement : souvent, cette phase permet de consolider le projet et éviter les développements inutiles.

Note : il est possible d'exprimer des idées complexes, mais au commencement du projet, il est préférable de garder un fonctionnement simplifié. Lors d'un prochain stand-up, il sera toujours temps de rajouter de la complexité si le projet avance bien.

3.5 PROGRAMMER

Programmer seul, se partager le travail, programmer en pair (deux personnes font le même code ensemble), etc.

En cours, le pair-programming n'est envisageable que pour deux personnes ayant le même niveau.

Commencez par la partie HTML puis langage de programmation.

La partie CSS ne devrait pas prendre de temps au début du projet : lorsque tout fonctionnera bien, il sera possible d'améliorer l'ergonomie du programme.

Lorsque votre projet fonctionne, ajoutez des frameworks comme chartJS ou introJS pour fournir des graphiques ou un tutoriel d'utilisation du programme.

Pensez à commenter votre code, nettoyez les commandes de débogage, etc.