Cahier de recette

Test utilisateur Angular

MITTON BENJAMIN

PINARD CEDRIC

Sommaire

1. Introduction

Ce document a pour objectif de décrire le contexte de production du projet ainsi que les modalités de rendu et d'évaluation. Dans un premier temps nous définirons comment ce travail doit être produit et dans un second temps comment il pourra être évaluer.

2. Description de la fourniture

L'application est une librairie Angular. Elle doit être utilisable dans d'autres projets Angular en suivant le guide d'installation disponible sur les pages de publication du projet. Le projet doit être disponible sur GitHub et sur npm.

Dans le cadre de la publication de la librairie sur GitHub et npm il faut rédiger des documents complémentaires. En effet le projet doit comporter un « ReadMe » en anglais, contenant une description brève de la librairie, un guide d'installation complet, une notice d'utilisation regroupant toutes les options de la librairie, un exemple d'utilisation sous forme de tutoriel et un lien vers une page d'exemple d'implémentation (hébergé avec GitHub Pages). Ainsi qu'un fichier « package.json » contenant, les spécifications de la librairie (nom, version, description et auteur) ainsi que l'adresse GitHub du code source et des mots-clés pour qu'elle soit correctement référencée.

En parallèle le projet doit aussi être compléter par la liste de document suivante :

- le cahier des charges
- la conception détaillée/Plan de développement
- le manuel d'utilisation
- le manuel d'installation
- le plan de tests
- la documentation interne
- le code source du programme
- le rapport du projet
- le résumé en français et en anglais
- la vidéo de démonstration
- les compte-rendus de réunion

3. Conformité aux spécifications générales

Pour que le projet corresponde aux spécifications générales il faut :

- Qu'il soit disponible sur npm
- Qu'il soit utilisable en suivant le guide d'installation et le guide d'utilisation
- Qu'il permette l'affichage de linechart
- Qu'il permette de l'intégrer au sein de projet plus complexe

4. Environnement de test

L'ensemble des tests sera réalisé par vérification manuelle, étant donné l'importance de l'aspect visuel dans ce projet. Un projet Angular contenant la librairie téléchargé depuis GitHub en suivant le « readme », sera utilisé. Chaque fonctionnalité donne lieu à un affichage à faire valider par un membre du projet qui a connaissance de la nature du test.

5. Conformité aux spécifications fonctionnelles

5.1. Affichage linechart simple

5.1.1. Identification

Afficher une linechart pour des données.

5.1.2. Descriptions

Une linechart est une courbe représentant des données, on souhaite donc qu'en appelant la librairie et en lui passant des données en paramètres que le résultat soit effectivement, sous la forme d'une linechart.

5.1.3. Procédure de test

Les données utilisées sont celles d'un jeu de données proposé par le client du projet. Ces données sont de la forme suivante :

<date h:m:s:ms ; nomducapteur ; valeur>.

On utilise donc le composant de la librairie avec pour nom de capteur, le capteur PC5 et on observe le résultat pour savoir si il est conforme selon le client.

5.2. Affichage linechart multiple

5.2.1. Identification

Afficher une linechart multiple pour des données

5.2.2. Descriptions

Une linechart multiple comme son nom l'indique, est une linechart contenant plusieurs lignes superposées sur un même graphique. On cherche à vérifier qu'il est bien possible d'afficher plusieurs lignes dans le même composant et qu'il n'y a pas de problème visuel.

5.2.3. Procédure de test

Les données utilisées sont celles d'un jeu de données proposé par le client du projet. Ces données sont de la forme suivante :

<date h:m:s:ms ; nomducapteur ; valeur>.

On utilise donc le composant de la librairie avec pour nom de capteur, les capteurs PC5, PC6, PC7 et on observe le résultat pour savoir s'il est conforme selon le client.

On viendra ici observer si les axes restent correctes, et si il n'y a pas de problème de superposition et que l'intégrité des courbes est respectée.

5.3. Affichage de différents type de données

5.3.1. Identification

Prendre en compte des données de type booléen, flottante et entière.

5.3.2. Descriptions

Les types de données que doit pouvoir lire le composant sont les booléens, les entiers et les flottants. Il faut donc vérifier pour chacun de ces trois types de données .

5.3.3. Procédure de test

Les données utilisées sont celles d'un jeu de données proposé par le client du projet. Ces données sont de la forme suivante :

<date h:m:s:ms ; nomducapteur ; valeur>.

On utilise donc le composant de la librairie avec pour nom de capteur, les capteurs Température_Cuisine (flottant), Présence_Cuisine (entier) et PC5 (booléen) et on observe le résultat pour savoir s'il est conforme selon le client.

On viendras ici observer que pour chacun des types de données, la conservation de son intégrité visuel et la correspondance aux données d'entrées.

5.4. Paramétrage des axes

5.4.1. Identification

Les axes sont paramétrables

5.4.2. Descriptions

Les axes des linecharts doivent pouvoir se calibrer automatiquement en fonction de la plage de données et/ou d'une plage fictive donnée en paramètres.

5.4.3. Procédure de test

Les données utilisées sont celles d'un jeu de données proposé par le client du projet. Ces données sont de la forme suivante :

<date h:m:s:ms ; nomducapteur ; valeur>.

On utilise donc le composant de la librairie avec pour nom de capteur, le capteur Température_Cuisine (flottant) ainsi que le domaine [0,30] et on observe le résultat pour savoir s'il est conforme selon le client.

On viendra ici observer que pour chacun des types de création des axes que la courbe reste fidèle et que les axes sont conformes aux paramètres transmis.

5.5. Style de courbe

5.5.1. Identification

Le style de la courbe peut être : une ligne, une aire ou les deux.

5.5.2. Descriptions

Les lignes des composants peuvent être soit une ligne, soit une aire, soit les deux en même temps selon les paramètres donnés par l'utilisateur .

5.5.3. Procédure de test

Les données utilisées sont celles d'un jeu de données proposé par le client du projet. Ces données sont de la forme suivante :

<date h:m:s:ms ; nomducapteur ; valeur>.

On utilise donc le composant de la librairie avec pour nom de capteur, le capteur Température_Cuisine (flottant) ainsi que le style line puis area puis both et on observe le résultat pour savoir s'il est conforme selon le client.

On viendra ici observer que pour chacun des styles que la courbe reste fidèle et que le format d'affichage des lignes est conforme aux paramètres.

5.6. Interpolation

5.6.1. Identification

L'interpolation de la courbe peut être linéaire ou en escalier.

5.6.2. Descriptions

Pour relier deux points d'une courbe, on peut tracer une ligne on obtient donc une interpolation linéaire mais on peut aussi tracer une ligne horizontale puis une ligne verticale on obtient donc une interpolation en escalier.

5.6.3. Procédure de test

Les données utilisées sont celles d'un jeu de données proposé par le client du projet. Ces données sont de la forme suivante :

<date h:m:s:ms ; nomducapteur ; valeur>.

On utilise donc le composant de la librairie avec pour nom de capteur, le capteur Température_Cuisine (flottant) ainsi que l'interpolation linear puis step et on observe le résultat pour savoir s'il est conforme selon le client.

On viendra ici observer que pour chacune des interpolation que la courbe reste fidèle et que le format d'affichage des lignes est conforme aux paramètres.

5.7. Affichage d'une bulle d'information

5.7.1. Identification

Au passage de la souris sur la linechart une bulle d'information doit s'afficher correctement.

5.7.2. Descriptions

Une bulle contenant la date et la valeur sur laquelle le pointeur de la souris est placé doit s'afficher dynamiquement en fonction des déplacements de la souris

5.7.3. Procédure de test

Les données utilisées sont celles d'un jeu de données proposé par le client du projet. Ces données sont de la forme suivante :

<date h:m:s:ms ; nomducapteur ; valeur>.

On utilise donc le composant de la librairie avec pour nom de capteur, le capteur Température_Cuisine (flottant). On déplace ainsi notre souris au sein du composant et on vérifie que les valeurs affichées sont conformes.

On viendra ici observer que sur les bords cette bulle s'affiche toujours, qu'elle reste visible et qu'elle est capable de s'inverser si elle doit sortir du composant.

5.8. Zoom

5.8.1. Identification

Lors de l'action de la molette, un zoom s'active sur l'axe des abscisses (temps). On agrandit ou on réduit la plage de temps des données que l'on affiche.

5.8.2. Descriptions

Il est possible de zoomer à l'intérieur d'un composant. Visuellement cela doit se traduit par un gain de précision au niveau de l'axe des abscisses et une adaptation de la ligne à la nouvelle plage de données.

5.8.3. Procédure de test

Les données utilisées sont celles d'un jeu de données proposé par le client du projet. Ces données sont de la forme suivante :

<date h:m:s:ms ; nomducapteur ; valeur>.

On utilise donc le composant de la librairie avec pour nom de capteur, le capteur Température_Cuisine (flottant). On zoom et dézoom au maximum au sein du composant.

On viendra ici observer que suite au zoom max et au dézoom max, que l'on retombe sur la ligne de départ, ainsi que le respect de l'intégrité visuelle du composant par cette fonctionnalité, c'est à dire qu'elle ne le détruit pas ou ne le déforme pas.

5.9. Scrollbar horizontale

5.9.1. Identification

Une scrollbar horizontale permet de se déplacer sur l'axe des abscisses (temps) dans les données. On déplace horizontalement la plage de temps des données que l'on affiche, sans changer la taille de cette plage.

5.9.2. Descriptions

Une scrollbar est disponible en dessous du composant, tant qu'il n'y a pas eu de zoom et elle prend l'entièreté de l'espace qui lui est attribué et ne peut être déplacée. Elle permet ensuite de déplacer la plage de temps visible avec le zoom.

5.9.3. Procédure de test

Les données utilisées sont celles d'un jeu de données proposé par le client du projet. Ces données sont de la forme suivante :

<date h:m:s:ms ; nomducapteur ; valeur>.

On utilise donc le composant de la librairie avec pour nom de capteur, le capteur Température_Cuisine (flottant) . On zoom pour ensuite se déplacer a droite puis a gauche dans la plage de données et on dézoom au maximum au sein du composant.

On viendra ici observer que suite au zoom max et au dézoom max, que l'on retombe sur la ligne de départ. Ainsi que le respect de l'intégrité visuelle du composant par cette fonctionnalité, c'est à dire qu'elle ne le détruit pas ou ne le déforme pas.

5.10. Synchronisation

5.10.1. Identification

On doit pouvoir synchroniser la plage de temps et le temps actuel, en entrée et en sortie entre les linecharts.

5.10.2. Descriptions

La synchronisation des fonctionnalités de navigation comme le zoom et la scrollbar permet (si spécifié dans les paramètres) à plusieurs linecharts de réagir de la même manière aux actions réalisées dans une seule d'entre elles.

5.10.3. Procédure de test

Les données utilisées sont celles d'un jeu de données proposé par le client du projet. Ces données sont de la forme suivante :

<date h:m:s:ms ; nomducapteur ; valeur>. .

On utilise donc les composant de la librairie avec pour nom de capteur, le capteur Température_Cuisine (flottant). On les synchronise depuis les paramètres et on zoom pour se déplacer avec la scrollbar à droite puis à gauche dans la plage de données et on dézoom au maximum au sein du composant.

On viendra ici observer que suite au zoom max et au dézoom max, que l'on retombe sur la ligne de départ. Ainsi que le respect de l'intégrité visuel du composant par cette fonctionnalité, c'est à dire qu'elle ne le détruit pas ou ne le déforme pas.

6. Conformité de la documentation

Le « ReadMe » doit être construit selon le modèle suivant :

- Une présentation : description brève et exemples d’implémentation
- Un guide d'installation : installation de la librairie et des dépendances
- Une notice d'utilisation : détaillant tout le contenu de la librairie et toutes les options
- Un exemple d'utilisation : sous la forme de tutoriel

Le « package.json » doit être construit selon le modèle suivant :

- nom
- version
- description
- auteur
- dépôt
- page d'accueil
- mots-clés
- dépendances

7. Annexes

Tableau des fonctionnalités attendues par le client :