Activité 1 : Notions de base d'Angular

On veut créer une application Angular qui est composée d'un composant principal '***App'*** et de cinq autres composants à savoir : ***Header, Footer, Home***, ***Products*** et ***Contact***.

L'interface de l'application au lancement du serveur est comme le montre la **Figure1** suivante. Les contenus des différents composants sont aussi présentés dans cette figure. Pour le composant ***Contact*** on se contente seulement par sa création à ce stade sans l'intégrer dans l'arborescence des composants.

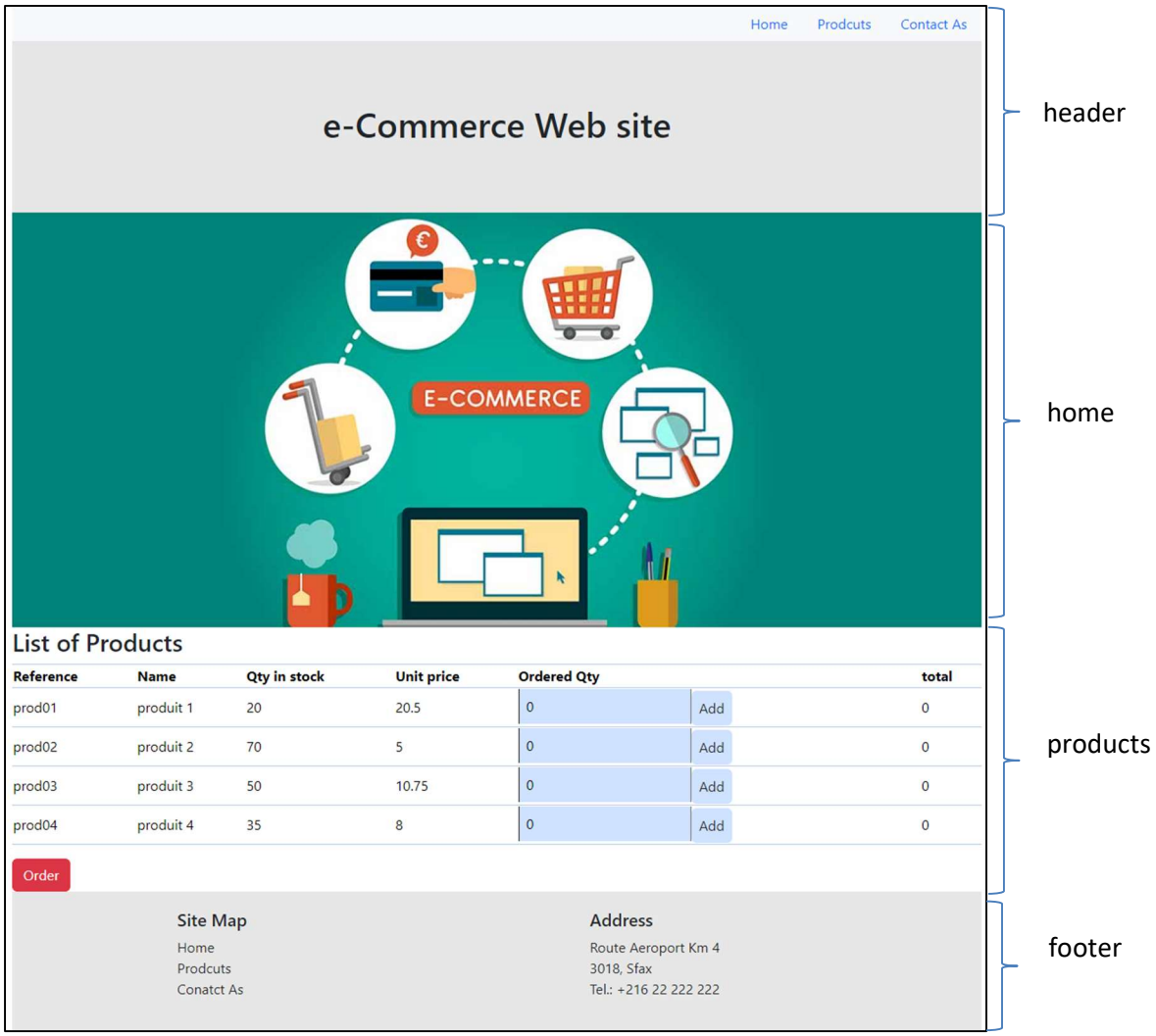

Figure 1

1. La première étape consite à reproduire les composants ci-dessus de l'application. Pour cela, on procède comme suit :
    a. En premmier lieu, on lance l'invite de commande de Windows et on se positionne dans le dossier qui va contenir l'application.
    b. Ensuite, il s'agit de créer le projet nommé '***rate-product*** de manière à ne pas générer des fichiers tests unitaires correpodants aux composants. Pour ce faire, on utilse la commande suivante : ***ng new rate-product --skip-tests=true***. Finalement, lorsque Angular, demande quel style adopter pour l'application maintenir le choix par défaut qui est CSS (on n'utilise pas des préprocesseurs de style à ce niveau).
    c. En dernier lieu, on doit créer les composants mentionnés ci-dessus à savoir : ***Home***, ***Products, Contact, Header*** et ***Footer***. Pour cela, il convient de se positionner dans l'invite de commande dans le dossier 'first-projet' créé lors de la création de l'application (cd rate-product). Ensuite, appliquer les commandes suivantes :
        - ng generate component home
        - ng generate component products
        - ng generate component contact
        - ng generate component header
        - ng generate component footer

    Il est possible aussi de créer tous les composants en une seule ligne de commande (ng generate component home products contact header footer)
2. La deuxième étape consiste à définir le contenu des composants ***Header***, ***Footer***, ***Home*** et ***Products*** selon leurs présentations dans la ***figure 1*** ci-dessus sachant que la mise en forme des composants se base sur Bootstarp.
    a. Intégrer BootStrap dans l'application.
        - Dans l'invite de commande, toujours positionné dans le répertoire du projet, lancer la commande : ***npm install bootstrap --save***
        - Inclure le Bootstrap dans la liste des styles du projet. Pour cela, ouvrir dans l'éditeur VS, le fichier '***angular.json***' et ajouter le chemin du fichier de style de bootstrap dans le tableau des fichiers de styles du projet. Le chemin de bootstratp est : "***node_modules/bootstrap/dist/css/bootstrap.css***".

    ⇨ Il est possible à partir de là d'appliquer des styles bootstrap sur tous les composants du projet.

        - Inclure le bundle Javascript de BootStrap dans l'application. Pour cela, ouvrir dans l'éditeur VS, le fichier '***angular.json***' et ajouter le chemin du bundle de bootstrap dans le tableau des scripts du projet. Le chemin de bootstratp est : "***node_modules/bootstrap/dist/js/bootstrap.bundle.min.js***".

    b. Dans le dossier '***public'*** de l'application, mettre l'image (à votre choix) qui sera utilisée par le composant ***Home***.

c. Définir le contenu HTML du composant ***Home*** sachant qu'il contient seulement l'image insérée ci-dessus dans le répertoire '***public'***. Cette image s'étale sur toute la largeur du composant.
d. Définir le contenu HTML du composant ***Header*** et appliquer les styles nécessaires afin d'obtenir une représentation similaire à celle de la figure1 ci-dessus. Le composant est composé d'un navigateur et d'une zone pour le titre.
e. Définir le contenu HTML du composant ***Footer*** et appliquer les styles nécessaires afin d'obtenir une représentation similaire à celle de la figure1 ci-dessus.
f. Définir le contenu du composant ***Products*** et appliquer les styles nécessaires afin d'obtenir une représentation similaire à celle de la figure1 ci-dessus. Il convient de mentionner que le contenu du tableau présenté dans ce composant est tiré à partir du code TypeScript du composant en appliquant l'inetrpolation et en utilisant ***@for***.
   - Dans le fichiers '***products.ts***', déclarer un tableau appelé ***products*** cotenant quatre objets produits caractérisés chacun par les propriétés suivantes : ***refrence***, ***name***, ***stockQte*** et ***unitPrice***. Les valeurs correspondantes aux différentes propriétés de chaque produit sont à extraire à partir de la ***figure 1*** ci-dessus.
   - Dans le fichier '***prodcuts.html***', insérer un titre, un tableau dont les lignes sont ajoutées dynamiquement à travers ***@for*** sachant que chaque ligne comporte 4 colonnes dont le contenu tiré à partir du TypeScript en plus des 2 colonnes à définir le traitement correspondant ultérieuerement (***Ordered Qty*** et ***total***). Le composant contient aussi un bouton ***order***.

3. La troisième partie consiste à définir les traitements correspondants aux boutons ***add*** et ***order*** du composant ***Products*** tous en utilisant les notions de event Binding tow-way binding. Ainsi, la cinquième colonne du tableau est un '***input***' permettant la saisie de la quantité à commander pour le produit correspondant. Une fois la quantité à commander est saisie, il s'agit de cliquer sur le bouton '***Add***' correspondant afin de calculer le total correspondant qui est égale à ***'quantité commandée * prix unitaire'***. Si la quantité saisie est supérieure à celle en stock le bouton ***Add*** sera masqué (utilisation du block de contrôle de flux ***@if***). On obtient donc une vue similaire à celle de la ***Figure2*** ci-dessous.

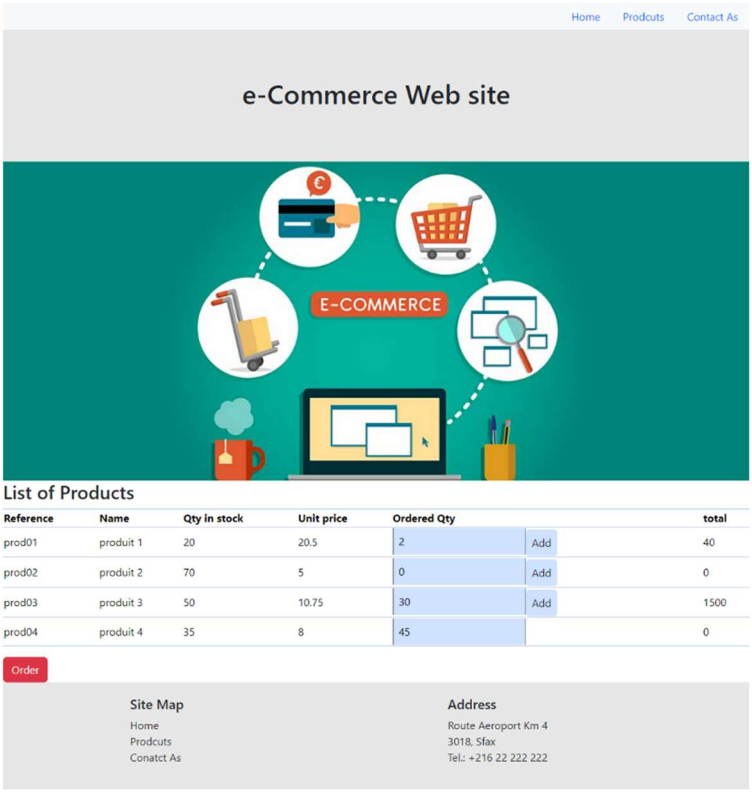

Figure2

Finalement, lorsqu'on clique sur le bouton '***Order***' qui au-dessous du tableau, il s'agit de calculer le total global de la commande et d'afficher un message comportant le résultat obtenu. On obtient une vue similaire à la ***Figure3***.

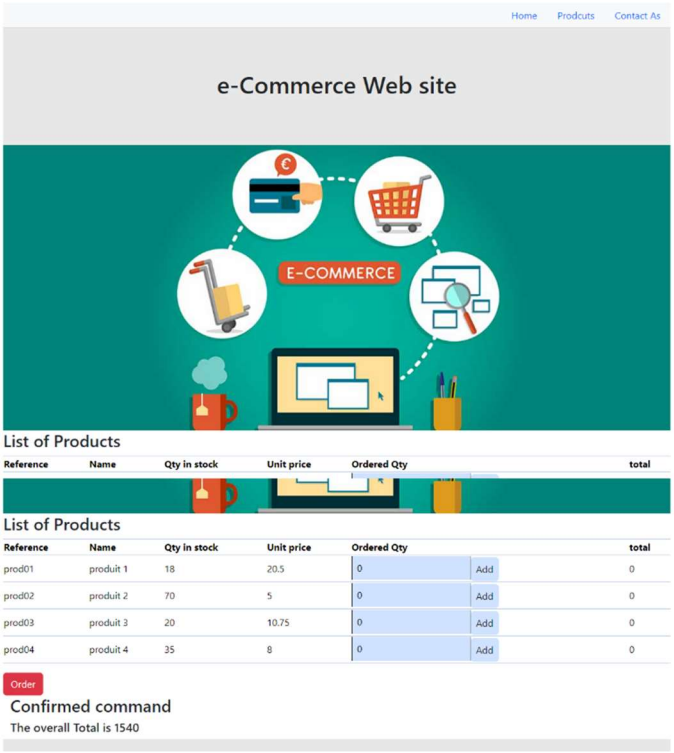

Figure3

Ainsi, le composant ***Products*** préconise dans son code en plus du tableau products:

- Un tableau d'entiers '**qte'** de taille 4 dont toutes les cases sont initialisées à 0. Les cases du tableau sont reliées aux '***input'*** correspondants de la colonne '***Ordered Qty***' dans le tableau de la partie html du composant.
- Un tableau d'entiers '**total'** de taille 4 dont toutes les cases sont initialisées à 0. Chacune des cases de ce tableau peut être remplie à la suite du clic sur le bouton correspondant '***Add'*** de la partie html. Le contenu de ce tableau est affiché dans la colonne '***Total'***.
- Une fonction ***addCommand (index)*** qui est déclenchée en cliquant sur le bouton '***Add'***. Elle permet de calculer pour le produit de la ligne où se trouve le bouton '***Add'***, le total correspondant selon la formule mentionnée ci-dessus.
- Une fonction ***order()*** qui est déclenchée en cliquant sur le bouton '***Order'*** qui permet de calculer le total global de la commande et de le mettre dans une variable appelée ***finalTotal*** tout en se basant sur les différents tableaux du composant. Une fois le total est calculé, cette fonction fait appel à une fonction appelée ***changeQte(index, qte)*** permettant de retrancher la quantité commandée du stock pour chaque produit commandé. Finalement, la fonction '***order()'*** doit remettre les cases des tableaux ***qte*** et ***total*** à 0.