|  |
| --- |
| Université de Paris |
| E-commerce |
| Aziz Djermani, Bastien Falchero |

|  |
| --- |
| M. Charroux  2021/2022 |

# Concept

Pour ce projet, nous avons réalisé un web service permettant une gestion simple d’un panier sur un site de e-commerce. Nous avons commencé par le développement de ce web service avec Spring boot en utilisant la version 11 de Java. Pour les données, nous avons choisi de les stocker dans une base de données MySQL externe au web service. Et enfin nous avons utilisé le Framework Angular pour développer une interface graphique, permettant d’essayer les différentes fonctionnalités de notre web service.

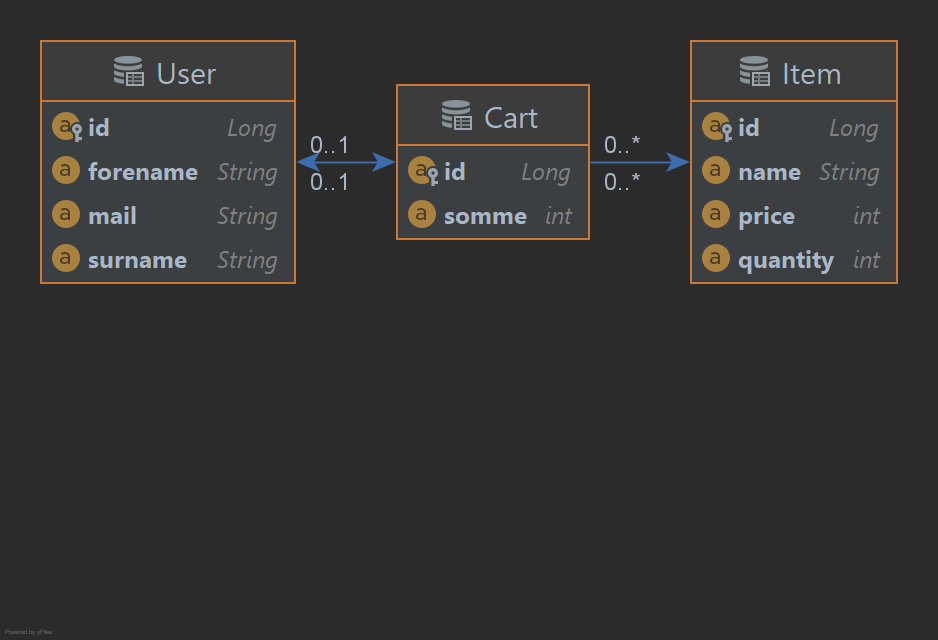
# Web service

## Architecture

Nous avons organisé notre code en 6 dossiers :

* Entities : il contient tous les objets principaux (user, cart, item) et la définition de leurs attributs.
* Dao : contiens les différentes classes pour connecter chaque entité à la base de données.
* Dto : contiens le formatage des données avant envoi.
* Exceptions : permet une gestion des exceptions et l’affichage de message d’erreurs.
* Service : les services permettent de réaliser des actions sur les données avant de les enregistrer en base de données.
* Controller : contiens les différentes classes permettant à travers les requêtes http implémenté de faire appel au service.

## Entités



Sur le diagramme ci-dessus, nous pouvons observer les attributs de chaque entité et les liens qui les associent. On a donc pour chaque user un unique panier (cart) qui lui est associé. Les paniers sont donc également reliés à un unique user et associé à aucun (vide) ou à plusieurs items. On note qu’à partir de l’item, on ne peut retrouver les paniers associés.

## Controller & fonctionnalités

#### User

Une image contenant texte

Description générée automatiquement

Requête Post permettant d’ajouter un user à la base de données.

* <http://localhost:8080/user/all>

Requête GET listant tous les users existant en base de données.

* [http://localhost:8080/user/{id}](http://localhost:8080/user/%7bid%7d)

Requête GET permettant de retourner un seul user identifié par son ID.

* [http://localhost:8080/user/{id}/update](http://localhost:8080/user/%7bid%7d/update)

Une image contenant texte

Description générée automatiquement

Requête PUT permettant de modifier une sélection d’attributs de l’user identifié par son ID.

* [http://localhost:8080/user/{id}](http://localhost:8080/user/%7bid%7d)

Requête DELETE permettant de supprimer un user identifié par son ID. On notera que lors de la suppression d’un utilisateur, son panier associé est également détruit. Si son panier contenait des items ceux-ci retourne automatiquement en stock disponible.

* <http://localhost:8080/user/search?search=’’john>’’

Requête GET permettant de lister les users don le mail contient la chaîne de caractère spécifié en paramètre.

### Item

* <http://localhost:8080/item/add>

Une image contenant texte

Description générée automatiquement

Requête Post permettant d’ajouter un item à la base de données.

* <http://localhost:8080/item/all>

Requête GET listant tous les items existant en base de données.

* [http://localhost:8080/item/{id}](http://localhost:8080/item/%7bid%7d)

Requête GET permettant de retourner un seul item identifié par son ID.

* [http://localhost:8080/item/{id}/update](http://localhost:8080/item/%7bid%7d/update)

Une image contenant texte

Description générée automatiquement

Requête PUT permettant de modifier une sélection d’attributs de l’item identifié par son ID.

* [http://localhost:8080/item/{id}](http://localhost:8080/item/%7bid%7d)

Requête DELETE permettant de supprimer un item identifié par son ID. On notera que lors de la suppression d’un item, s’il était présent dans un cart, il sera supprimé de celui-ci.

* <http://localhost:8080/item/search?search=’’Iphone>’’

Requête GET permettant de lister les items donc le nom contient la chaîne de caractère spécifié en paramètre.

### Cart

* [http://localhost:8080/user/{id}/cart](http://localhost:8080/user/%7bid%7d/cart) Requête GET permettant de lister les items d’un cart identifié par l’ID de l’user.
* [http://localhost:8080/user/{id}/cart/price](http://localhost:8080/user/%7bid%7d/cart/price)

Requête GET permettant de récupérer la valeur du panier.

* [http://localhost:8080/user/{id}/cart/add/{itemID}](http://localhost:8080/user/%7bid%7d/cart/add/%7bitemID%7d) Requête GET permettant d’ajouter un item à un panier. Le stock de l’item est décrémenté.
* [http://localhost:8080/user/{id}/cart/remove/{itemID}](http://localhost:8080/user/%7bid%7d/cart/remove/%7bitemID%7d) Requête GET permettant de retirer un item d’un panier. Le stock de l’item est incrémenté.
* [http://localhost:8080/user/{id}/cart/purchase](http://localhost:8080/user/%7bid%7d/cart/purchase)

Requête GET permettant d’acheter les items d’un panier. Le panier est vidé sans que le stock des items ne soit réincrémenté.

## Dockerfile

Une image contenant texte

Description générée automatiquement

Nous avons divisé notre dockerfile en deux parties. Dans un premier temps, nous utilisons une image de maven pour build le web service sans oublier d’inclure les dépendances présentes dans le fichier pom.xml. Dans un second temps une fois le build terminé, on utilise l’image openjdk:11 pour lancer le web service avec le port 8080 exposé.

# 3. Angular

## A. Architecture

Le principe d’Angular est de diviser l’implémentation en composant. Nous en avons donc créé plusieurs et les détails ci-dessous à travers les fonctionnalités.

## B. Fonctionnalités

### 1. Utilisateurs

En accédant à l’url : ‘’localhost/users’’. Nous arrivons sur la page d’accueil contenant la liste de tous les utilisateurs. Sur cette page, nous retrouvons deux barres de navigation la première est présente sur toutes les pages et permet de naviguer entre la liste des utilisateurs et la liste des items. La seconde évolue en fonction des fonctionnalités. Ici, nous avons All pour lister tous les utilisateurs, New pour ajouter un nouveau user et une barre de recherche pour filtrer les utilisateurs en fonctions de leur mail.

De plus, chaque utilisateur possède 4 boutons pour réaliser différentes actions :

Permet de supprimer l’utilisateur.

Permet d’éditer le profil de l’utilisateur.

Permet d’accéder au panier de l’utilisateur.

Permet d’accéder à l’interface pour ajouter des articles au panier de l’utilisateur.

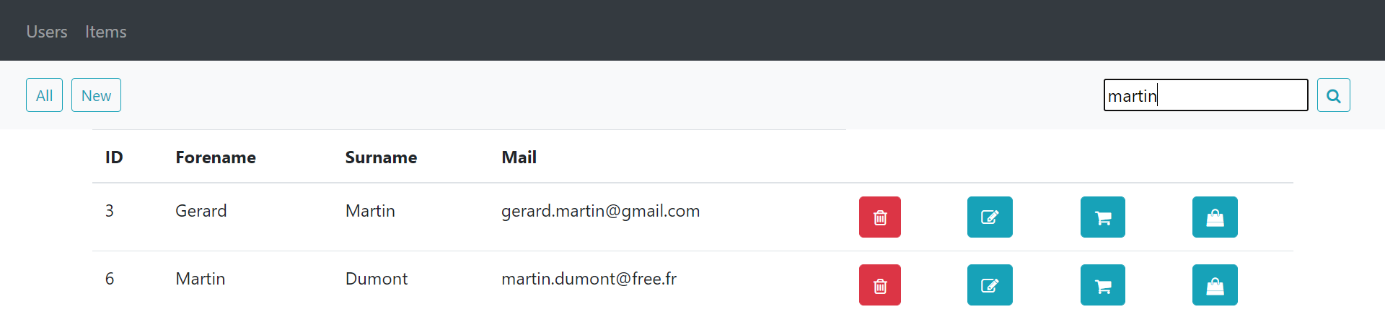
Cliquons sur le bouton New.

### Ajout utilisateur

Nous arrivons sur cette interface et pouvons donc remplir les champs et enregistrer ce nouvel utilisateur. On notera que le mail spécifié ne doit pas déjà être enregistré dans la base.

En revenant sur la page des utilisateurs, nous souhaitons le retrouver et nous utilisons donc la recherche par mail.

### Recherche par mail



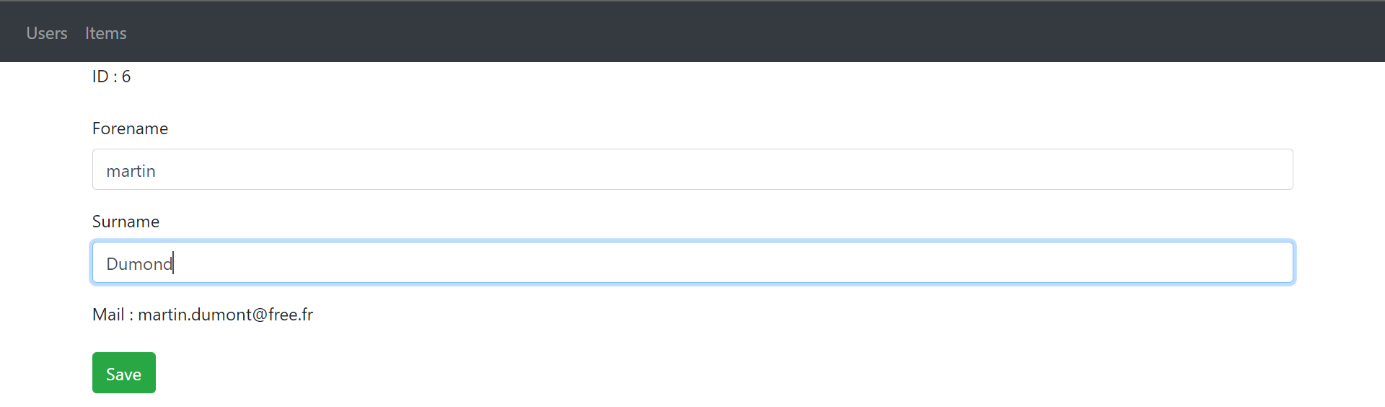
Nous avons filtré les utilisateurs par leur mail. Deux mails contiennent le mot « martin » donc ces deux utilisateurs s’affichent.

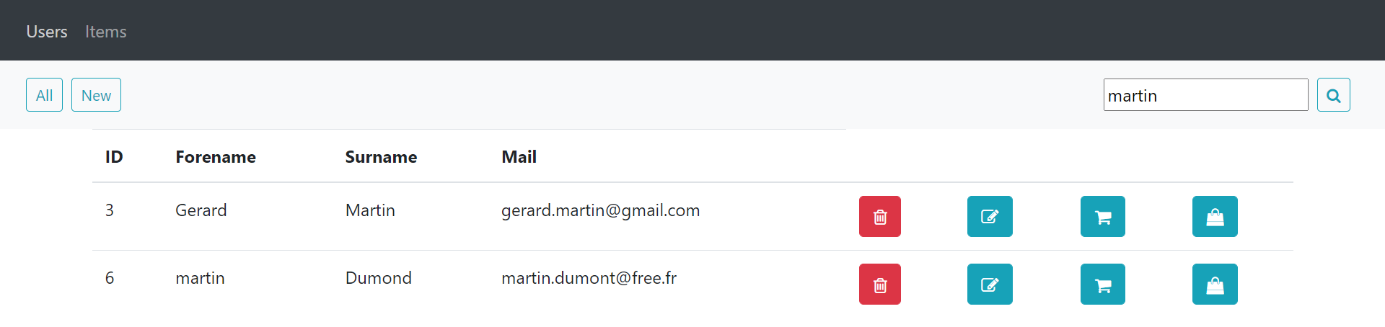
Nous allons modifier le profil utilisateur en cliquant sur le bouton d’édition.

Une image contenant texte

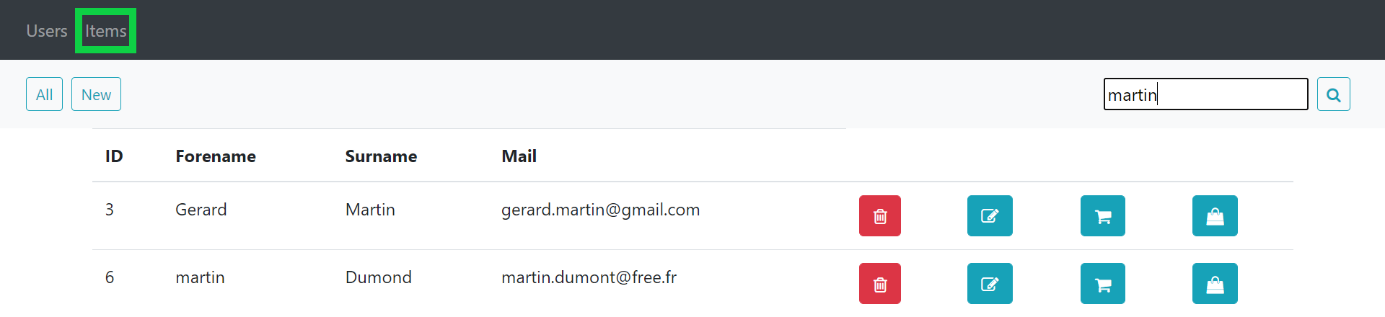
Description générée automatiquement

### Edition utilisateur



Nous arrivons donc sur cette page et pouvons modifier le prénom et le nom de l’utilisateur. Comme le mail est unique, nous avons décidé de ne pas le rendre éditable.

En retournant sur la liste des utilisateurs et en les filtrant nous constatons que les champs de l’utilisateur ont bien été modifiés.

Nous allons maintenant lister les items.

### Une image contenant table Description générée automatiquementItems

L’interface est très similaire avec celle des utilisateurs et les fonctionnalités sont :

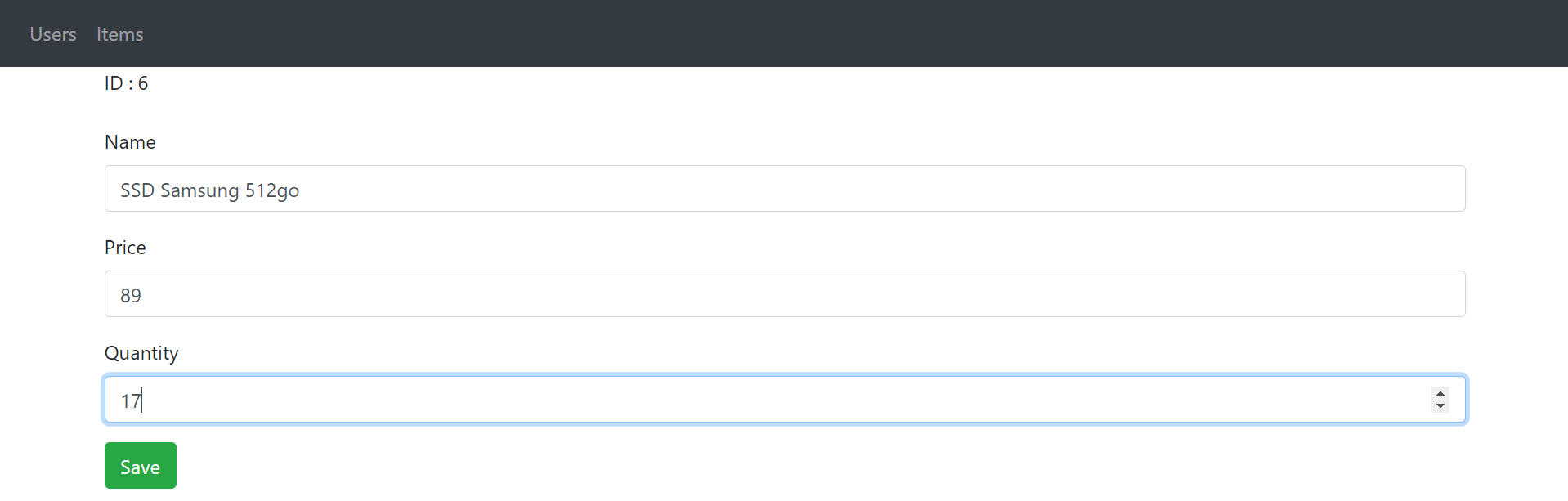
* + - Lister tous les items
    - L’ajout d’un item
    - La recherche par nom d’un item
    - La suppression
    - La modification

Ajoutons un item en cliquant sur New.

### Ajout item

Puis éditons-le.

### Edition item



### Une image contenant table Description générée automatiquementRecherche item

A l’aide de la recherche par nom d’article, nous retrouvons deux articles dont celui que nous venons de créer puis d’éditer.

Maintenant, revenons à nos utilisateurs puis allons ajouter quelques articles au panier.

### Shopping

Une image contenant table

Description générée automatiquementAprès avoir cliqué sur le bouton marqué, nous arrivons à une interface reprenant les bases de la liste des items avec d’autres fonctionnalités.

Nous avons trois fonctionnalités : afficher tous les articles, les filtrer par leur nom et ajouter au panier. Nous ajoutons 4 articles au panier de John.

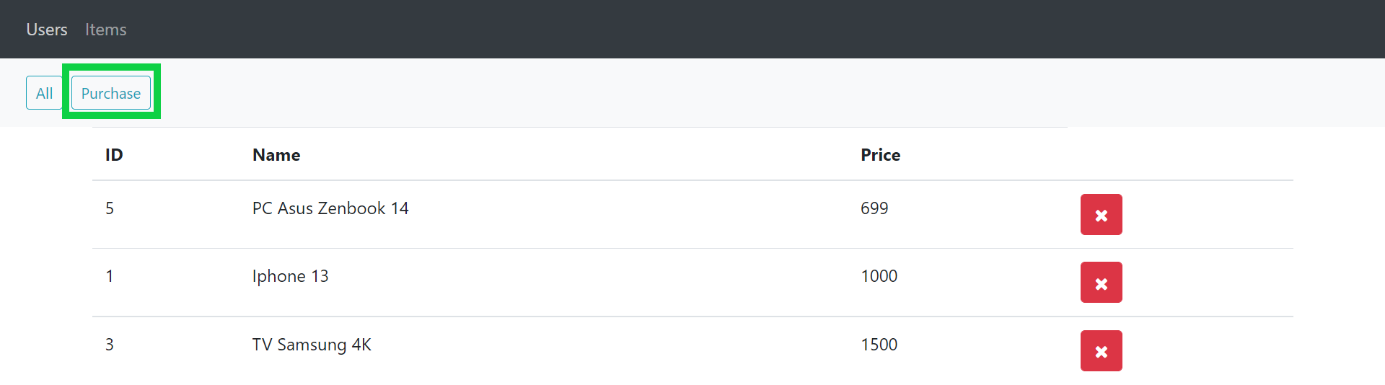
### Une image contenant table Description générée automatiquementPanier

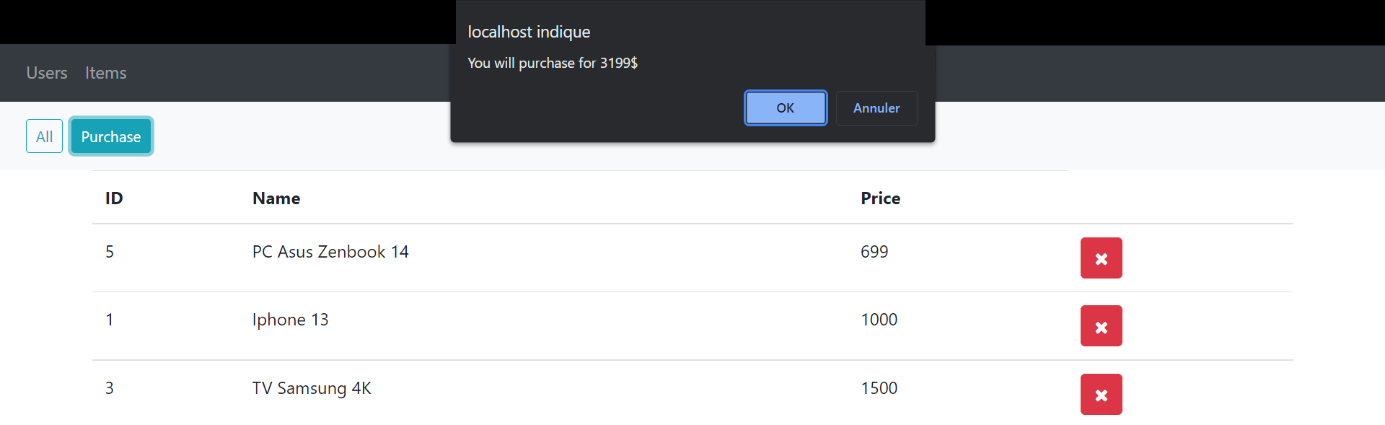
Accédons au panier de John en cliquant sur le bouton marqué.

Une image contenant table

Description générée automatiquement

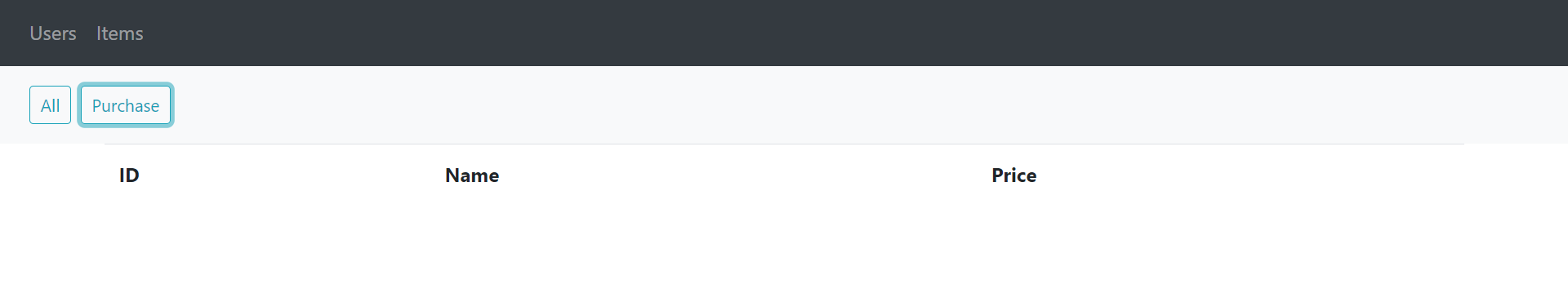
Nous voyons tous les articles de John et avons également quelques fonctionnalités. Nous pouvons acheter tous les articles présents dans le panier et retirer des articles du panier. Commençons par retirer un article en cliquant sur le bouton marqué.

Nous constatons que l’article a bien été retiré du panier. Nous pouvons acheter ces articles en cliquant sur le bouton marqué.

Une alerte nous indiquant le montant total nous demande confirmation avant l’achat.

Une image contenant texte

Description générée automatiquementUne fois validé une deuxième alerte nous confirme la bonne réalisation de l’achat.



Le panier est maintenant vide.

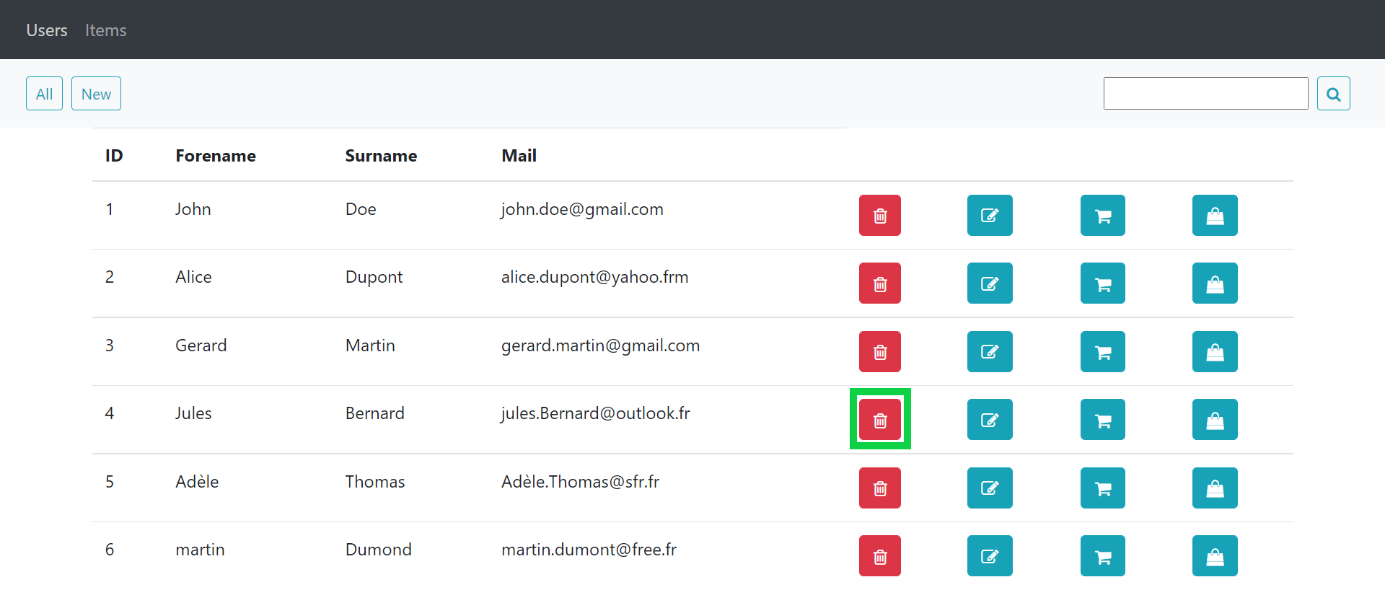
Nous allons nous intéresser aux quantités des articles qui ont été décrémentées lors de l’ajout d’un article à un panier et décrémentées lors de son retrait, mais pas lors de l’achat. Souvenons-nous, il y avait 9 Iphone 13, 1 TV Samsung 4K et 12 PC Asus Zenbook. À la suite de notre achat, il doit donc rester 8 Iphone 13, 0 TV Samsung 4K et 11 PC Asus Zenbook. On notera que la quantité de TV est à zéro donc il n’est plus possible d’ajouter l’article à un panier.

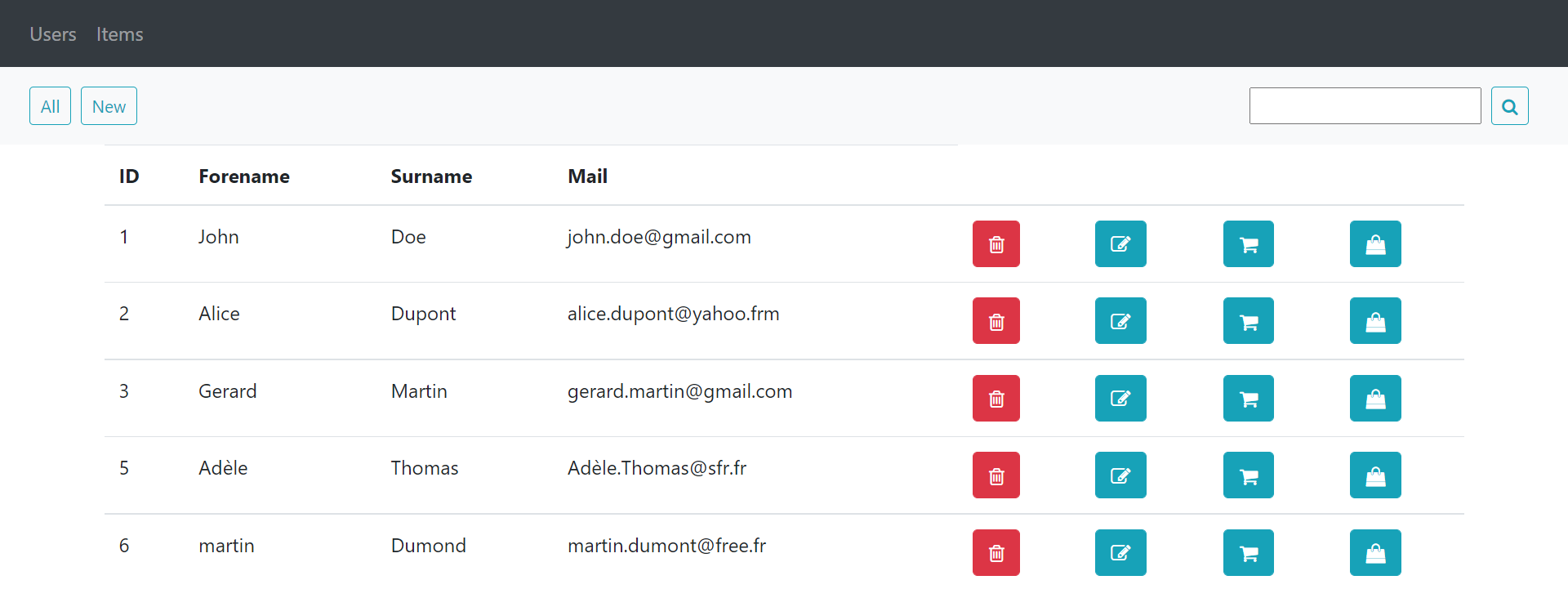
Une image contenant table

Description générée automatiquement

Les quantités ont bien été mises à jour.

1. La suppression

Pour supprimer un utilisateur, il suffit de cliquer sur le bouton marqué, s’il possédait des articles dans son panier ceux-ci sont remis en stock automatiquement.



L’utilisateur à bien été supprimé.

Pour la suppression d’un article, le procédé est strictement le même. Si l’article était présent dans un ou plusieurs paniers d’utilisateurs, il sera automatiquement retiré.

## Une image contenant texte Description générée automatiquementC. Dockerfile

Comme pour le web service le Dockerfile en une partie build et une partie run. Nous récupérons l’image node:14-alpine, créons un répertoire de travail /usr/src/app ou nous y copions nos fichiers de configurations puis nous lançons la commande ‘npm install’ pour installer les dépendances qui sont spécifiées dans ces fichiers de configuration. Par la suite, nous copions tous les fichiers pour ensuite lancer le build avec la commande ‘npm run build’

Pour rendre le projet accessible, nous avons besoin de configuré un serveur web, nous avons choisi la dernière version de Nginx. Nous copions d’abord le fichier de configuration préalablement rempli puis le projet build à l’étape précédente.

Une image contenant texte, écran, capture d’écran

Description générée automatiquement

# Docker-compose

Nous avons décidé de diviser chacun des trois services dans des conteneurs distinct. Pour cela, nous avons créé un fichier docker-compose.yml.

Une image contenant texte

Description générée automatiquement Il se divise en trois services :

* La base de données MySQL, dans laquelle on spécifie notre environnement notre mot de passe root et le nom de notre base de données. Cela nous servira pour connecter la base au web service. Enfin, les services en parallèles pourront accéder à la base par le port 3306.
* Le web service, nous construisons l’image avec le Dockerfile que nous avons présenté plus haut. Ce service sera accessible par le port 8080 et redémarre après un échec.
* Angular, on construit également l’image avec son propre Dockerfile présenté plus haut. Ce service est accessible par le port 80.

Pour que la base de données soit connectée au web service nous avons édité le fichier « application.properties » comme suit :

Une image contenant texte

Description générée automatiquementLa première ligne permet à chaque redémarrage de recréé une base de données en supprimant la précédente. La ligne qui suit permet de préciser le chemin accès à la base. Les deux lignes qui suivent permettent de se connecter à la base avec l’identifiant et le mot de passe définit lors de la création du conteneur.

# Démo

Pour tester le projet, vous pouvez le cloner avec la commande :

* git clone <https://github.com/babafafa7/project.git>

Placez-vous dans le répertoire contenant le fichier docker-compose.yml puis lancer la commande :

* docker-compose up

Une image contenant texte, moniteur, télévision, écran

Description générée automatiquementVous devriez voir les trois images créées et en utilisation :

Une image contenant texte

Description générée automatiquementAinsi que les conteneurs :

Vous pouvez maintenant accéder à la page web à l’url : « http://localhost/ »