Examen STW: DEMO 2

1 Exercici Vue

Heu de programar una web amb Vue.js per mostrar una llista de la compra tal com es descriu a continuació. S'han de poder eliminar elements de la llista.

1.1 Les dades

La llista de la compra la teniu ja disponible a l'objecte JSON anomenat **`shoppingList`**.

```
const shoppingList = [
  { item: "Ous", quantity: 2, units: ["dotzena", "dotzenes"] },
  { item: "Llet", quantity: 1, units: ["litre", "litres"] },
  { item: "Oli", quantity: 2, units: ["litre", "litres"] },
        /* an arbitrary number of elements may appear here */
  { item: "Patates", quantity: 15, units: ["kilogram", "kilograms"] },
];
```

Nota: al camp **`units`** hi ha una llista amb les unitats en singular i plural.

1.2 El resultat

La llista s'ha presentar tal i com es mostra a les Figs. 1 i 2.

Figure 1: Com s'ha de veure la llista de la compra.

Figure 2: Resultat d'eliminar la llet de la llista.

Detalls

- S'han de mostrar les unitats en plural o singular segons correspongui.
- No es pot assumir que hi ha un número constant d'elements a la llista.
- Idea: **`v-for="(item, index) in items"`**.

2 Exercici Javascript / Nodejs

En aquest exercici cal implementar el metode **waitForAll**.

```
result = waitForAll(list)
```

Aquest mètode rep com a paràmetres una llista de funcions (**list**). Aquest mètode retorna una promise (**result**).

Ha de fer el següent:

a) Cada 10 ms ha d'executar totes les funcions de **list**. Quan totes les funcions retornin un valor *truthy*[1] al mateix moment, es resoldrà (amb **resolve**) la promise **result** amb un valor **true**.

b) Si alguna de les funcions de **list** fa un throw, es rebutjarà (amb **reject**) la promise **result** amb l'error capturat pel **try { ... } catch (error) { }**.

Nota: un cop resolta o rebutjada la promise **result**, cal que no s'executin més les funcions de **list**.

Exemples d'ús:

```
// 1
result = waitForAll([() => true]);
result.then(m => console.log("1) ", m));

// Resultat: 1) true

// 2
result = waitForAll([() => "hola", () => 23]);
result.then(m => console.log("2) ", m));

// Resultat: 2) true

// 3
var bool = false;

result = waitForAll([() => bool, () => true]);
setTimeout(() => bool = true, 100);
console.log("3) ", bool);
result.then(m => console.log("3) ", m));

// Resultat: 3) false
//           3) true

// 4
var bool1 = false, bool2 = false;

result = waitForAll([() => bool1, () => bool2]);
var id1 = setInterval(() => bool1 = !bool1, 40);
var id2 = setInterval(() => bool2 = !bool2, 60);
setTimeout(() => { clearInterval(id1); clearInterval(id2) }, 500);
result.then(m => console.log("4) ", m));

// Resultat: 4) true

// 5
var anotherBool = false;
```

[1] Equivalent a **true** a efectes d'un **if(valor)**.

```
result = waitForAll([() => true, () => { if(anotherBool) throw "Error 123" }]);
setTimeout(() => anotherBool = true, 100);
result.catch(m => console.log("5) ", m));

// Resultat: 5) Error 123

// Your tests here
```

NOTA: SI EL VOSTRE SCRIPT DE TEST NO ACABA MAI, NO HO ESTEU FENT BÉ.
