EXAMEN: Sistemes i Tecnologies Web Juny 2023

Niu: Nom:

Permutació A

☞ Bike Booking

Després de pujar el codi a moixero.uab.cat ompliu el següent requadre.

Entrega Electrònica

The SHA1 checksum of the received file is:

.....

Time stamp is:

.....

Guardeu-vos una copia de l'arxiu que entregueu.

Guia de correció:

La nota serà...	Quan...	Guia de puntuació
De 0 a 5 punts	Quan no funciona tot i hi ha errors de concepte *greus* en algun dels següents elements clau: callbacks, clausures, classes, herència, promises, i els diversos elements de vue (reactivitat, directives, events, props, components, ...).	Es parteix d'un 5 i es resta 1 punt per cada error de concepte.
De 5 a 7 punts	Quan no s'aconsegueix fer funcionar tot l'examen i no hi ha errors de concepte greus.	Es parteix d'un 7 i es resten 0.25 punts per cada error.
De 7 a 10 punts	L'examen passa tots els tests i feu entrega electrònica.	Teniu un 10. Us l'heu guanyat.

Instruccions

Seguiu les següents instruccions per a arrencar la màquina del laboratori i importar l'esquelet del projecte.

- Arrenqueu la màquina si no l'heu arrencada abans i seleccioneu la partició de Linux (user: examen i passwd: examen).
- Obriu una consola: Applications → Terminal.
- Descarregueu-vos l'esquelet del projecte executant la comanda:

  ```
  wget https://moixero.uab.cat/ExamenSTW.7z
  ```

- Descomprimiu el fitxer.

  ```
  7z x ExamenSTW.7z
  ```

- L'esquelet el teniu dins el directori **stw**. Feu l'examen (podeu fer servir el mateix terminal que ja teniu obert, i obrir el directori **stw** amb el Visual Studio Code).
- Per executar l'aplicació utilitzeu la comanda:

  ```
  node exam.js
  ```

- Nota: Si utilitzeu Chrome, veureu que s'obre en mode incògnit i no hi ha disponible l'addon de Vue. Obriu un nou Chrome (Ctrl+N). La nova finestra ja no és incògnit i el addon de Vue es pot utilitzar.
- Un cop hagueu acabat de desenvolupar el projecte, haureu d'entregar el vostre codi electrònicament. **Si us funciona tota l'aplicació, aviseu-nos abans d'entregar electrònicament.**
- Per entregar electrònicament, creeu un zip de la següent manera:

  ```
  7z a sol.zip exam.js vue/app.js
  ```

- Comproveu el contingut de l'arxiu que entregareu (obriu l'arxiu i mireu el contingut dels arxius que hi ha a dintre).
- Un cop sapigueu segur que voleu entregar aquest arxiu, pugeu-lo a: `https://moixero.uab.cat/`.
- Anoteu els dos valors (el checksum i el timestamp) a l'examen en paper i entregueu l'examen en paper.
- **Quan acabeu no sortiu de la sessió i no pareu la màquina.**

Context

Suposem que estem treballant en un portal de reserva de bicis tal el que es mostra a la Figura 1.

Bike 1 slots	Bike 2 slots	Bike 3 slots	Bike 4 slots	Bike 5 slots
Bike:1 Slot:1	Bike:2 Slot:1	Bike:3 Slot:1	Bike:4 Slot:1	Bike:5 Slot:1
Bike:1 Slot:2	Bike:2 Slot:2	Bike:3 Slot:2	Bike:4 Slot:2	Bike:5 Slot:2
Bike:1 Slot:3	Bike:2 Slot:3	Bike:3 Slot:3	Bike:4 Slot:3	Bike:5 Slot:3
Bike:1 Slot:4	Bike:2 Slot:4	Bike:3 Slot:4	Bike:4 Slot:4	Bike:5 Slot:4
Bike:1 Slot:5	Bike:2 Slot:5	Bike:3 Slot:5	Bike:4 Slot:5	Bike:5 Slot:5

Figura 1: Portal de reserva de bicis.

Cada columna representa una bici a reservar, i cada fila un slot de reserva. Com podem veure a la Figura 1, el portal tindrà una llista de cinc bicis a poder reservar a on per cada bici, tindrem cinc slots de reserva. La idea és que cinc possibles usuaris opten a llogar una mateixa bicicleta. El servidor adjudicarà la bicicleta al darrer usuari que hagi clicat un dels slots de la bici. Cada una de les bicletes és independent.

Podeu arrencar aquest portal executant **`node exam.js`**.

Exercici frontend

Només caldrà que programem el component anomenat **`BookingSlotComponent`**, que representarà cadascun dels slots per reservar una bici.
No s'ha de modificar cap de les altres parts del frontend. Només el component **`BookingSlotComponent`**. En particular **no** s'ha de modificar el component **`RootComponent`**.

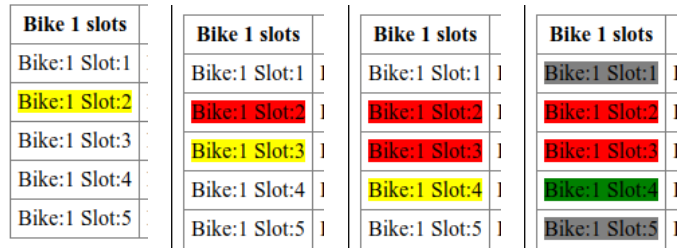
Bike 1 slots
Bike:1 Slot:1
Bike:1 Slot:2
Bike:1 Slot:3
Bike:1 Slot:4
Bike:1 Slot:5

(a) Cliquem la Bike1, Slot2.

Bike 1 slots
Bike:1 Slot:1
Bike:1 Slot:2
Bike:1 Slot:3
Bike:1 Slot:4
Bike:1 Slot:5

(b) Passats 5 segons se li adjudica la Bike1 al Slot2.

Figura 2: Cas d'us a on s'adjudica la Bike1 al Slot2 passats 5 segons d'haver clicat el slot.



(a) En t_0 cliquem sobre la Bike1 (b1), Slot2 (s2).

(b) En t_1 cliquem sobre (b1, s3).

(c) En t_2 cliquem sobre (b1, s4).

(d) En t_5 se li assigna la bike1 al slot4

Figura 3: Cas d'us a on s'adjudica la Bike1 al slot4 passats 5 segons d'haver clicat el Slot2.

Especificacions

– El component **`BookingSlotComponent`** ha de mostrar l'identificador de la bici i del slot com a la figura 1. Podeu utilitzar el següent template HTML:

```
<span style="background-color: '...'"></span>
```

a on **`background-color`** pot prendre els valors: **`'', yellow, red, green, or gray`**

– El component que heu d'implementar s'ha de poder fer servir de la següent manera:

```
<BookingSlotComponent :bikeId="bikeIndex" :slotId="slotIndex"
  v-model="bikeBookedArray[index]"></BookingSlotComponent>"
```

a on **bikeId** és l'identificador de la bici, **slotId** és l'identificador del slot de la bici i **`bikeBookedArray[bikeId]`** és un booleà que ens indica si la *bikeId* associada al *slotId* ja està reservada.

Aquest booleà se li passa al component via **`v-model`** (two way data binding), de manera que si des d'un component es reserva una bicicleta, el component "avisa" al RootComponent per a que actualitzi el booleà.

– El component ha de ser clicable només un únic cop. En clicar-lo canviarà l'estat del slot tal i com indiquem a continuació i ja no tindrà cap efecte tornar-lo a clicar.

– El component ha de mostrar l'estat de la reserva de la bici amb els colors de fons següents:

cap color: El slot està lliure, és a dir no ha estat clicat prèviament. En aquest cas el slot no té cap color de fons i s'ha de poder clicar.

groc: Un usuari clica sobre un slot lliure: (1) el color de fons es posa a groc (Figura 2), (2) el component fa la petició de reserva de la bici al servidor i espera la seva resposta, que pot ser la cadena **'booked'** o bé **'rejected'**.

La petició de reserva es fa cridant a l'endpoint **`/book`** del servidor, passant-li com a *query params* els atributs **`bikeId`** i **`slotId`**. Per exemple, si hem clicat el slot (b1, s2), la petició al servidor seria: **`/book?bikeId=1&slotId=2`**. Per a fer aquesta petició utilitzeu la funció:

```
fetch(url).then(res => res.text()).then()
```

verd: En fer la petició de reserva anterior, el servidor ens ha respost amb la cadena **'booked'** (ens ha reservat la bici). En aquest cas el color de fons canvia de groc a verd i cal informar al **`RootComponent`** per a que canviï el **`booleà bikeBookedArray[bikeX]`** a **`true`**.

gris: Si el slot no ha sigut clicat però la bicicleta associada a aquest slot ha sigut reservada des d'un altre slot (**`bikeBookedArray[bikeX] = true`**), aquest slot passa a tenir un color de fons gris i clicar-lo no té cap efecte.

vermell: Si estem en estat 'groc' i el servidor ens respon amb la cadena **'rejected'** indica que el servidor no ens adjudica la bicicleta i per tant el color de fons passarà de groc a vermell. Altres slots de la bici poden estar encara disponibles per ser clicables.

– Les reserves de les diferents bicicletes son independents entre elles.

A continuació us mostrem un altre exemple de com hauria de funcionar el portal un cop implementades les funcionalitats esmentades anteriorment:

Bike 3 slots	Bike 4 slots
Bike:3 Slot:1	Bike:4 Slot:1
Bike:3 Slot:2	Bike:4 Slot:2
Bike:3 Slot:3	Bike:4 Slot:3
Bike:3 Slot:4	Bike:4 Slot:4
Bike:3 Slot:5	Bike:4 Slot:5

(a) Es clica sobre (b3,s1) i sobre (b4,s2).

Bike 3 slots	Bike 4 slots
Bike:3 Slot:1	Bike:4 Slot:1
Bike:3 Slot:2	Bike:4 Slot:2
Bike:3 Slot:3	Bike:4 Slot:3
Bike:3 Slot:4	Bike:4 Slot:4
Bike:3 Slot:5	Bike:4 Slot:5

(b) Es clica sobre (b4, s3).

Bike 3 slots	Bike 4 slots
Bike:3 Slot:1	Bike:4 Slot:1
Bike:3 Slot:2	Bike:4 Slot:2
Bike:3 Slot:3	Bike:4 Slot:3
Bike:3 Slot:4	Bike:4 Slot:4
Bike:3 Slot:5	Bike:4 Slot:5

(c) Després de 5 segons d'haver seleccionat (b3,s1) se li assigna la bici3 al slot s1. Després de 5 segons d'haver seleccionat (b4,s2) se li assigna la bici4 al darrer slot que ha fet la selecció, s3

Figura 4: Cas d'us a on es reserven dues bicicletes de forma independent.

Notes

- Recordeu que els dos tags següents són equivalents:

```
<Component v-model="data" />
<Component v-bind:modelValue="data" v-on:update:modelValue="x => data = x" />
```

- No es pot fer servir async/await.

Exercici backend

Context

Cal implementar una part del backend del portal.
El servidor només tindrà un endpoint: /**book**. Rebrà dos *query params*: **bikeId**, amb l'identificador de la bici a reservar i **slotId**, amb l'identificador del slot de reserva. Cal que implementeu el mòdul **`bookBikeModule`** que inclourà la funció **`bookBike`**. Quan el servidor rebi una petició a l'endpoint /**book** es farà una crida a la funció **bookBike**. Aquesta funció retornarà una promesa que resoldrà a la cadena **'booked'** si assigna aquesta bicicleta a aquest slot, o bé farà un reject amb la cadena 'rejected' si no assigna aquesta bici a aquest slot. La cadena a la que resol/rejecta la promesa és el que li envieu al client com a resposta a la petició '/**book**'.
Mentres la funció **`bookBike`** no estigui implementada, l'esquelet proporciona una resposta hardcodejada per al client, on retorna la cadena **'booked'** o **'rejected'** immediatament de manera aleatòria.

Mòdul `bookBikeModule`

Aquest mòdul el desenvupareu seguint el patró *module pattern*. Aquí teniu un exemple.

```
const testModule = (function() {
  let _a = 1;
  const _increase = function () { _a++; },
  const _value = function() { return _a; }
  return {
    increase :  _increase,
    value:  _value
  };
})();
```

Funció `bookBike`

- Aquesta funció s'executa en rebre una petició a l'endpoint **`/book`**.
- La funció **`bookBike`** del mòdul **`bookBikeModule`** rep com a paràmetres **`bikeId`**, **`slotId`** i retorna una *promise*: **`bookBike(bikeId, slotId) : promise`**

- Quan s'executi aquesta funció sobre un **`bikeId`** per **primer cop**, cal que **iniciï un comptador de cinc segons específic per aquesta bici**.
- Si passats cinc segons no s'ha rebut cap altra petició de reserva sobre aquesta bici es resol la promesa a **'booked'**.
- Si durant aquests cinc segons s'executa la funció **`bookBike`** per reservar aquesta bici a través d'un altre slot, **no** s'ha d'iniciar cap comptador, i immeditament, s'ha de *rejectar* la promesa de la petició de reserva anterior.

  Exemple:

  1. A t_0 fem una primera petició de reserva (b3,s1). Obtenim la promesa p_3_1 que inicia un timeout de 5 segons.
  2. A t_1, rebem una segona petició de reserva (b3,s5). Obtenim la promesa p_3_5. La promesa p_3_1 s'ha de rejectar a 'rejected'.
  3. A t_2, rebem una tercera petició de reserva (b3,s4). Obtenim la promesa p_3_4. La promesa p_3_5 s'ha de rejectar a 'rejected'.
  4. A t_5 resolem la promesa p_3_4 a 'booked'.
- Atenció, en cas que el **`bikeId`** sigui idèntic al **`slotId`**, cal que la promise faci *reject* immediatament, p.e. (b1,s1), (b2, s2), etc. (Figura 5)

Bike 1 slots	Bike 2 slots	Bike 3 slots	
Bike:1 Slot:1	Bike:2 Slot:1	Bike:3 Slot:1	I
Bike:1 Slot:2	Bike:2 Slot:2	Bike:3 Slot:2	I
Bike:1 Slot:3	Bike:2 Slot:3	Bike:3 Slot:3	I
Bike:1 Slot:4	Bike:2 Slot:4	Bike:3 Slot:4	I
Bike:1 Slot:5	Bike:2 Slot:5	Bike:3 Slot:5	I

Figura 5: S'han seleccionat els slots amb el mateix id que la bicicleta.

- Fixeu-vos que no sabeu quan podreu executar la funció **`resolve`** i **`reject`** de a promesa retornada per la funció **`bookBike`**. Pot ser us cal guardar-vos aquestes funcions. Si voleu les podeu encapsular en un objecte.

  Penseu si us cal guardar els *rejects* i *resolves* de les promeses dels diferents slots sobre una mateixa bici. Pot ser només us cal guardar els del darrer slot?
- Opcional: Per gestionar el timeout iniciat en rebre la primera petició sobre un **`bikeId`** us pot ser útil la funció **`buildTimeOut`** (opcional):

```
const buildTimeOut = (bikeId)=>
  return {
    state: 0, //0: off  1: on
    startTimeOut: function(){
    //TODO  }}}
```

  Aquesta funció rep com a paràmetre un **`bikeId`** i retorna un objecte que encapsula un timeout per a aquesta bici. Aquest objecte conté la propietat **`state`** que indica si el timeout està en marxa (1) o parat (0). La propietat **`startTimeOut`** és una funció que canvia l'estat de l'objecte, inicia el timeout de cinc segons.
- Opcional: També podeu utilitzar un array per tenir un d'aquests objectes per cada bici:

```
let bikesTimeouts=[buildTimeOut(1), buildTimeOut(2), ..., buildTimeOut(5)]
```

- Nota: Per a fer un reset de les reserves us caldrà parar el server.

Notes

- S'ús dona un esquelet d'aquesta interfície web configurat amb:

```
app.use(express.static(path.join(__dirname, 'vue')));
```

- No es pot fer servir async/await.