EXAMEN: Sistemes i Tecnologies Web Juny 2024

Niu: Nom:

☞ Tres en ratlla

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:

.....

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 8 punts	Quan no s'aconsegueix fer funcionar tot l'examen i no hi ha errors de concepte greus.	Es parteix d'un 8 i es resten 0.25 punts per cada error.
De 8 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:
  1. el client: **`npm run dev`**
  2. el servidor: Posicioneu-vos al directori **`src`** i executeu **`node exam.js`** o bé **`nodemon 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:
  Posicioneu-vos en el directori **`src`**

  ```
  7z a sol.zip exam.js App.vue components/SellForm.vue components/Shop.vue
  ```

- 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

Implementarem una versió simplificada del joc del tres en ratlla. Utilizarem dues pestanyes del navegador, a on cada pestanya representarà un jugador (figura 1). Per facilitar-ho tot, cada jugador veurà només els seus moviments. Hi ha dos casos d'ús en què el jugador veurà al seu tauler la fitxa de l'altre jugador:

- Si és el torn del jugador i fa clic a una cel·la ocupada per l'altre jugador (figura 9).
- Si no és el torn del jugador i fa clic a una cel·la ocupada per l'altre jugador (figura 8).

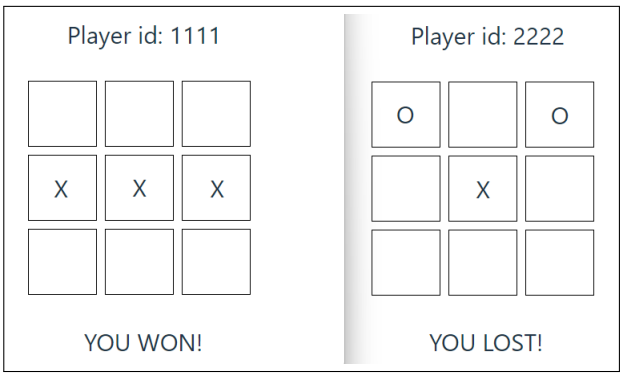

Figure 1: El jugador 1 (P1) juga amb "X" i el jugador 2 (P2) amb "0". P1 guanya. Fixeu-vos que P2 no pot veure els moviments de P1 i viceversa.

Podeu iniciar aquest portal executant:

1. El servidor: aneu al directori **`src`** i executeu **`node exam.js`** o **`nodemon exam.js`**.
   El servidor estarà disponible a través de la URL **`http`**:*`//localhost:3001`*.
2. El client: **`npm run dev`**
   El client estarà disponible a través de la URL **`http`**:*`//localhost:3000`*.

Veureu la vista a la figura 2.

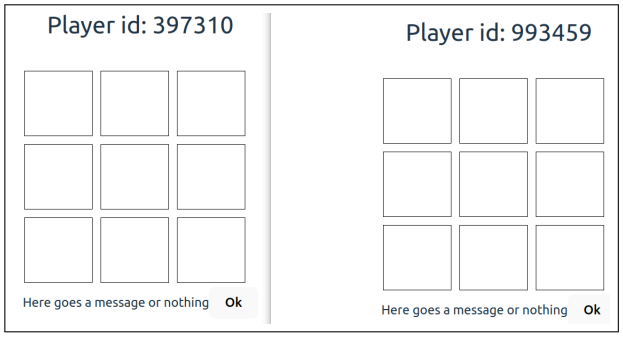

Figure 2: Vista de l'esquelet.

Exercici de frontend

Heu de completar el component **`RootComponent`** (**`App.vue`**) i el component **`Cell`**. A continuació es mostren les especificacions dels components.

Root Component (App.vue)

El **`RootComponent (App.vue)`**, crearà una graella 3x3 de components **`Cell`** (**fet**).

Especificacions

- Rep el **gameWinner** obtingut pel component **Cell** mitjançant un *binding* amb la directiva **v-model**. Si el **gameWinner** és *truthy* i correspon a la variable del **RootComponent**, **playerId**, mostra el missatge **YOU WON!**, en cas contrari mostra el missatge **YOU LOST!** (Figura 1)
- Reacciona a l'event **missatge** llençat pel component **Cell** i emmagatzema el paràmetre de l'event a la variable reactiva **missatge**. Mostra el missatge seguit d'un botó "OK" (figura 7).
- Quan es fa clic al botó **OK**, posa el **missatge** a *undefined*, de manera que ni el missatge ni el botó siguin visibles.

Component Cell

El component **Cell** serà el que gestionarà els clics a la cel·la, la comunicació amb el backend i mostrarà la ficha (token) ('X' o 'O').

Especificacions

- Rebrà les següents **props**:
  - **fila** (*number*): representa la fila de la cel·la. Va de 0 (més a munt) a 2 (més aball).
  - **columna** (*number*): representa la columna de la cel·la. Va de 0 (la columna més a l'esquerra) a 2 (la columna més a la dreta).
  - **playerId** (*number*): representa l'identificador del nostre jugador.
  - **gameWinner** (*number*): representa l'identificador del jugador que ha guanyat la partida. El valor per defecte és **null**. Està lligat bidireccionalment amb el **RootComponent**.
    Recordeu que les dues etiquetes següents són equivalents:

```
<Component v-model:specificProp="data" />
<Component :specificProp="data" @update:specificProp="x => data = x" />
```

- Al **template**, el tag `<div class="cellExterior">` representa tota la cel·la. És clicable. Quan es cliqui, hem de fer una petició al endpoint **/cell_click**:

```
fetch(`http://localhost:3001/cell_click?playerId=11111&row=0&column=0`)
 .then(res => {this.status=res.status; return res.json()})
 .then(json => { ... })
 .catch(() => { ... })
```

  ☞ Fer clic a una cel·la no tindrà cap efecte si ja estem realitzant una petició, o ja hi ha un guanyador o hi ha un token a la cel·la.

- El endpoint respon amb un **JSON** i un **codi d'estat**. El JSON pot ser:
  - **{gameWinner: null, token: X}** on el token pot ser 'X' o 'O'. Encara no hi ha guanyador. En aquest cas només mostrem el token rebut a la cel·la. Per fer-ho, al **template**, el tag **<p>** mostra el token. El valor del token per defecte és una cadena buida **""**. Canviarà a **"..."** mentre s'espera una resposta a la petició al endpoint **/cell_click**. En rebre la resposta, canviarà al valor del token rebut 'O' o 'X' (Figura 3).

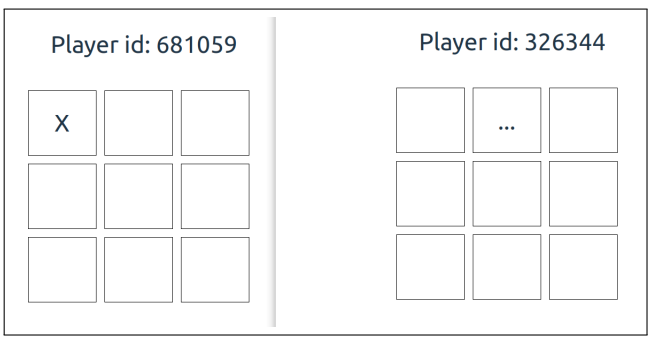

Figure 3: P1 ha fet clic a la cel·la[0][0] i ha obtingut el token "X". P2 ha fet clic a c[0][1] i està esperant la resposta del servidor.

– **`{gameWinner: 11111, token: X}`** En aquest cas s'ha acabat la partida. Tenim un guanyador, el jugador 11111, que ha realitzat l'últim moviment. Hem de notificar al **`RootComponent`** el guanyador perquè mostri el missatge "YOU WON" (Figura 4).

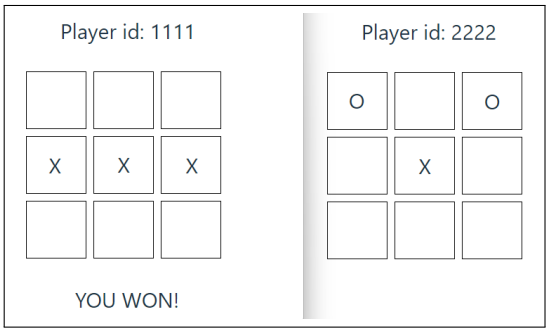

Figure 4: P1 guanya, P2 encara no ho sap perquè és el seu torn de joc.

– **`{gameWinner: 11111}`** El jugador 1111 va guanyar el joc en l'últim moviment. El jugador 2222 encara no ho sap perquè és el seu torn de joc. El jugador 2222 fa clic a una cel·la i obté aquest JSON. Hem de notificar al **`RootComponent`** el guanyador perquè pugui mostrar el missatge "YOU LOST" (Figura 5).

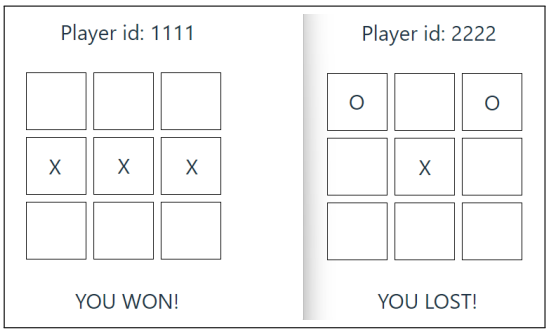

Figure 5: P1 guanya, P2 fa clic a una cel·la i el servidor l'informa de que el joc ha acabat i P1 és el guanyador.

– El codi d'estat (**`this.status`**) pot ser:

200: - És el nostre torn i la cel·la està buida. En aquest cas mostrem el token rebut del servidor (Figura 6).

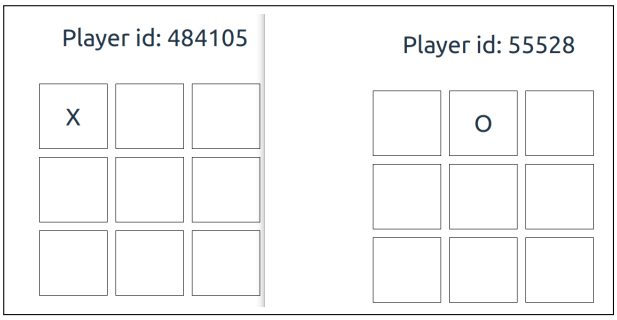

Figure 6: P1 selecciona c[0][0] i P2 c[0][1].

- Heu guanyat la partida (figura 4) o heu perdut la partida (figura 5).

401: No és el nostre torn (Figures 7 i 8). Llencem l'event **missatge** passant-li com a paràmetre la cadena "It is not your turn" perquè el **`RootComponent`** pugui mostrar-lo.

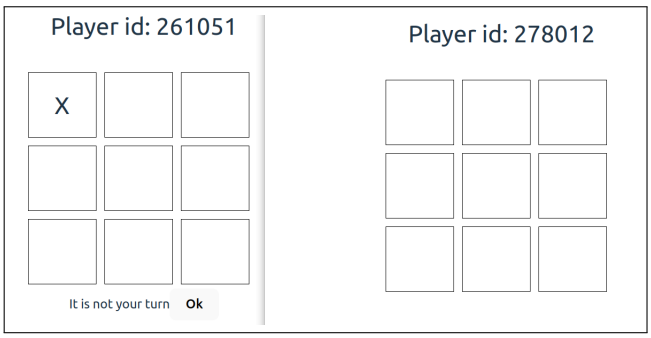

Figure 7: P1 selecciona c[0][0] i abans que P2 faci la seva jugada, P1 selecciona c[0][1]. Observeu el missatge a la part inferior de la vista de P1.

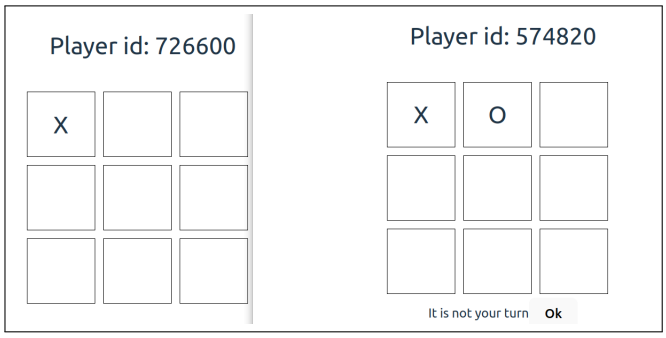

Figure 8: P1 selecciona c[0][0], P2 selecciona c[0][1] i abans que P1 faci la seva jugada, P2 selecciona c[0][0]. Mostrem el token de P1 al tauler de P2 i el missatge "It is not your turn".

400: L'altre jugador ja ha posat un token a la cel·la que acabem de clicar (figura 9). En aquest cas, mostrarem un borde al voltant de la cel·la: `style="border: 2px solid red"` i el token rebut a la cel·la.

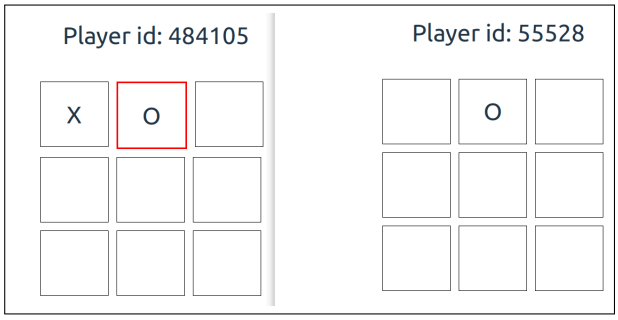

Figure 9: És el torn de P1 i selecciona c[0][1] on ja hi ha un token de P2. Mostrem el token de P2 al tauler de P1 i un borde vermell a la cel·la.

Test del frontend sense implementar el backend

Us proporcionem un servidor *dummy* que després d'un segon de rebre la petició al endpoint `/cell_click`, sempre retorna un codi d'estat de **200** i el següent JSON: `{gameWinner: nul, token: "X"}`
Podeu canviar els valors dels camps del JSON i del codi d'estat per provar la vostra interfície pels casos:

- **Jugada normal**: No heu de tocar el JSON. Clicar de nou la mateixa cel·la no generarà cap petició nova.
- **Cel·la ocupada**: Modifiqueu el status que retorna el server a `400`.
- **No és el vostre torn**: Modifiqueu el status que retorna el server a `401`
- **Heu guanyat**: Modifiqueu el status que retorna el server a `200`. Modifiqueu el JSON i poseu el vostre `playerId` en el camp `gameWinner`.Com ha acabat la partida ja no podreu clicar cap cel·la.
- **Heu perdut**: Modifiqueu el JSON i i poseu com a `gameWinner` el id: 111. Cliqueu una cel·la en la segona pestanya del navegador.

Exercici de backend

L'objectiu general del backend és fer un seguiment dels torns, dels clics a les cel·les i determinar el guanyador del joc. Està gairebé **fet**. Heu de completar la declaració del mòdul **game**, que oferirà mètodes per utilitzar el endpoint `/cell_click`. Al fitxer **exam.js**, només s'han d'implementar les parts marcades com a `// TODO`.

Especificacions del mòdul game

– Aquest mòdul s'implementarà utilitzant el *module pattern*. Aquí teniu un exemple:

```
const testModule = (()=>{
 let _a = 1;
 const inc = () => { _a++; },
 const getValue = () => { return _a; }
 return {
 inc,
 getValue,
 }
})();
```

– (**fet**) Defineix les variables privades següents:

- **`currentTurnToken`** (*string*): aquesta variable pot ser `"X"` o `"O"`. Indica el token que es col·locarà a la següent cel·la a la que s'hi farà clic. El valor per defecte és `"X"`, el que significa que al primer jugador que faci clic en una cel·la, se li assignaran creus.
- **`tokenOwner`** (*diccionari*): aquesta variable emmagatzema l'associació entre un token i el jugador que l'utilitza. Per exemple, si conté `{"X": 1111, "O": null}`, vol dir que les creus són del jugador amb id 1111 i que els cercles encara no estan assignats a cap jugador. El valor per defecte és `{"X": null, "O": null}`, donat que al començament del joc, cap jugador té una fitxa assignada.
- **`board`** (*array*): és una matriu de 3x3 que representa el tauler del joc. Inicialment, cada posició de la matriu (`board[row][col]`) contindrà `null`. A mesura que es vagin clicant cel·les hi haurà el token (`"X"` o `"O"`).

– Defineix els mètodes següents:

- **`isMyTurn(playerId)`** : **`boolean`** **(ja implementat)**: donat un **`playerId`** retorna *true* o *false* depenent de si és el torn del jugador o no.
- **`getCellToken(fila, columna)`** : **`'X'/'O'`** **(ja implementat)**: retorna el contingut d'una cel·la (`'X'/'O'`) o `null`.
- **`getGameWinner()`** : **`playerId`** **(ja implementat)**: aquest mètode retornarà l'Id del **`gameWinner`** o `null` si hi encara no hi ha guanyador.
- **`endTurn()`** : **`void`** **(ja implementat)**: **Aquest mètode s'ha de cridar quan el jugador faci clic a una cel·la i la cel·la estigui buida.** Canviarà el valor de **`currentTurnToken`** (`'X'/'O'`) i comprovarà si hi ha un tres en ratlla al tauler.
- **`refreshTokenOwner(playerId)`** : **`promise`** **(TODO)**: aquest mètode rep el **`playerId`** del jugador que ha clicat una cel·la i **retorna una promesa**.

  El mètode comprova si és el torn d'un jugador (utilitza el mètode **`isMyTurn(playerId)`**), si és així, actualitza el propietari del token (**`tokenOwner[currentTurnToken] = playerId`**) i **resol a res** la promesa. En cas contrari, **rebutja la promesa també a res**.
- **`refreshCell(row,col)`** : **`promise`** **(TODO)**: aquest mètode rep la **`fila`** i **`columna`** d'una cel·la del tauler i retorna una promesa. **El mètode comprova si la cel·la està buida**. Si la cel·la està buida (**`!board[fila][columna]`**), col·loca la fitxa del jugador actual a la cel·la:

  `board[fila][columna] = currentTurnToken`

  i **resol la promesa amb el token que ha guardat a la cel·la** (**`currentTurnToken`**). Si la cel·la no està buida, **rebutja la promesa amb token que ja hi ha a la cel·la** (**`board[fila][columna]`**).

– El mòdul **`game`** ha de revelar (exportar) els mètodes: **`refreshTokenOwner`**, **`refreshCell`**, **`getCellToken`**, **`getGameWinner`** i **`endTurn`**.

Especificacions del servidor web

El servidor implementarà un endpoint: `/cell_click`, que rebrà el playerId i la fila i la columna de la cel·la que s'ha clicat:
`http://localhost:3001/cell_click?playerId=11111&row=0&column=0`
Hem de convertir els tres paràmetres de la *query request* a números abans de passar-los com a paràmetres als mètodes del mòdul **game**. Els podem convertir en números utilitzant la funció Javascript `parseInt` (és a dir, `parseInt('123')`).

Aquests són els passos que ha de realitzar el endpoint `/cell_click`:

1. Crida a la funció `getGameWinner` per comprovar ja hi ha un guanyador.
2. Si ja hi ha un guanyador, envia al client el fitxer JSON `{gameWinner: thegame_winner}` i l'estat `200`.
   `res.status(200).json({gameWinner: the_gameWinner})`
3. En cas contrari, crida a la funció `refreshTokenOwner` per comprovar que és el vostre torn de joc.
4. Si no és el vostre torn, envia el codi d'estat `401` al client. Utilitzeu la funció `getCellToken` per obtenir el token que hi ha a la cel·la:
   `res.status(401).json({gameWinner: the_gameWinner/null, token: 'X'/'O'/null})`
5. Si és el vostre torn, comprova si la cel·la seleccionada està buida (`refreshCell`).
6. Si la cel·la està buida, el codi d'estat serà `200`, si no `400`.
7. Tant si la cel·la està buida com si no, obteniu el token amb el qual s'ha resolt/rebutjat la promesa `refreshCell`.
8. En cas que la cel·la estigui buida, heu d'acabar el vostre torn cridant a la funció `endTurn()`.
9. Respon al client amb el json:
   `{gameWinner: the_gameWinner/null, token: theCurrentToken}` i l'estat actual (`200`, `400` o `401`):
   `res.status(estat).json({gameWinner: the_gameWinner/null, token: theCurrentToken})`

Test de l'aplicació

☞ Per iniciar un joc nou, heu de reiniciar el servidor i tornar a carregar les pestanyes del navegador.

1. **Jugada normal**: P1 a c[0][0] i P2 a c[0][1]. **Output:** Figures 3 i 6.
2. **Feu clic dues vegades sobre la mateixa cel·la**: Test_1 i P1 a c[0][0]. **Output:** No es genera cap petició.
3. **La cel·la no està buida**: Test_1 i P1 a c[0][1]. **Output:** Figura 9.
4. **No és el vostre torn i feu clic sobre una cel·la buida**: Test_3 i P2 mou a c[0][2]. **Output:** Figura 7. Després de fer click al botó `OK`, el missatge "It is not your turn" i el botó desapareixen.
5. **No és el vostre torn i feu clic sobre una cel·la ocupada**: Test_3 i P2 mou a c[0][0]. **Output:** Figura 8. Després de fer clic al botó `OK`, el missatge "It is not your turn" i el botó desapareixen.
6. `P1 guanya`: Continueu el joc i feu que P1 sigui guanyador. **Output:** Figura 4. P2 fa clic a una cel·la buida. **Output:** Figura 5.

Aquesta és un full d'esborrany. No el desgrapis!