Práctica 2

Mentirosos en red

Aplicaciones Web
Año 2017/2018
Grado en Ing. Software
Fdi - UCM

Fecha de entrega: 28 de enero de 2018.

Porcentaje en la calificación de prácticas: 35 % (1.05 puntos en la calificación final de la asignatura).

Objetivos: La práctica tiene como objetivo la evaluación de las siguientes competencias:

- Implementación de APIs de tipo REST (Tema 8).
- Implementación de aplicaciones web utilizando AJAX y el modelo SPA (Temas 7 y 8).

Un aplicación SPA (*Single Page Application*) es una aplicación web contenida en un **único** documento HTML. Los componentes de este documento se muestran u ocultan a medida que el usuario interactúa con la aplicación web.

Evaluación: Esta práctica contiene tres apartados, de los cuales el primero (*Gestión de usuarios y partidas*) es obligatorio. Entregando solamente este apartado, la calificación máxima obtenible es de APTO con 4/10. Si se desea obtener una calificación mayor, se deberán realizar las extensiones opcionales de la práctica. La calificación máxima variará en función del número de extensiones implementadas.

- Apartado 1: Calificación máxima de 4/10 (0.42 puntos en la nota final de la asignatura).
- Apartados 1 y 2: Calificación máxima de 8/10 (0.84 puntos en la nota final).
- Apartados 1, 2 y 3: Calificación máxima de 10/10 (1.05 puntos en la nota final).

Se recuerda que la ausencia de entregas o la entrega de una práctica total o parcialmente copiada conllevará la calificación de NO APTO no recuperable. Los casos de copia serán notificados al Comité de Actuación ante Copias de la FdI.

En la corrección se valorará:

- El correcto funcionamiento de la aplicación web.
- El correcto diseño de la API RESTful y su implementación en *Express.js*. El servidor implementado ha de funcionar a través del protocolo **HTTPS**.

El diseño de la página web. No es necesario que sea igual a las capturas de pantalla que se muestran en las figuras de este enunciado. Para facilitar el trabajo, podéis utilizar *Bootstrap* o tecnologías similares del lado del cliente, aunque el uso de estas tecnologías no es obligatorio.

En esta práctica implementaremos un conocido juego de cartas: *el mentiroso*. Aunque este juego admite un número variable de jugadores, aquí nos restringiremos a partidas de cuatro jugadores (ni más, ni menos). Además, en el juego se utilizará la baraja francesa de 52 cartas, cuyas imágenes se proporcionan en el Campus Virtual.

Funcionamiento del juego

El juego del mentiroso comienza con el reparto de todas las cartas entre los cuatro jugadores. Cada jugador comienza, por tanto, con 13 cartas en su mano que mantiene ocultas del resto de jugadores. Para ganar el juego, cada jugador deberá librarse de todas sus cartas.

Se juega por turnos. La primera persona en jugar (que se determina aleatoriamente), debe colocar una, dos, o tres de sus cartas boca abajo, en el centro de la mesa. Tras esto, dirá en voz alta la combinación de cartas que ha colocado, la cual puede o no coincidir con la combinación que ha colocado realmente sobre la mesa. Tanto si dice la verdad como si miente, la combinación que el jugador declare haber puesto ha de consistir en cartas del mismo número o figura. Por ejemplo, un jugador puede afirmar haber colocado «*dos cuatros*», o «*dos reinas*», pero no puede decir que ha colocado «*un tres y una reina*». Esta restricción es independiente de lo que el jugador haya realmente colocado sobre la mesa, donde las cartas pueden tener distinto número.

Tras la primera jugada, el segundo jugador en participar tiene dos opciones:

- **Creer al jugador anterior**. En este caso, el segundo jugador deberá añadir, al menos, una carta más, boca abajo, en el centro de la mesa, y afirmar que las cartas colocadas tienen el mismo número o figura que las del jugador anterior. Es decir, si el primer jugador afirma haber colocado dos cuatros, el segundo puede colocar otras tres cartas boca abajo y decir «*tres cuatros más*», aunque ninguna de las cartas colocadas realmente sea un cuatro. En este caso pasa el turno al tercer jugador, que deberá decidir si cree o no cree al segundo.
- **No creer al jugador anterior**. En este caso, el segundo jugador dará la vuelta a las cartas del primer jugador y las mostrará a todos los jugadores. Si el primer jugador había mentido, se llevará todas las cartas del centro de la mesa a su mano. En caso contrario, será el segundo jugador (el incrédulo) el que se lleve a su mano todas las cartas depositadas en el centro.

En resumen, cada ronda del juego consiste en un jugador colocando unas cartas en el centro de la mesa, y el resto de jugadores añadiendo más cartas (todas, supuestamente, del mismo número o figura) hasta que uno de ellos decide desconfiar del jugador anterior y levantar las cartas colocadas por este último. Si las sospechas del jugador que desconfía son ciertas, el jugador que mintió se lleva todas las cartas de la mesa; en caso contrario, es el jugador incrédulo el que se lleva las cartas. El jugador que comienza la siguiente ronda será el jugador siguiente al que se haya llevado las cartas de la ronda anterior.

En cualquier momento del juego (bien sea al principio de la partida, o bien cada vez que un jugador se lleva las cartas de la mesa), si un jugador acumula en su mano cuatro cartas del mismo valor, puede librarse de ellas descartándolas en un montón aparte. De este modo, cada vez habrá menos cartas en juego.

Gana el juego el primer jugador que se deshaga de todas sus cartas, con una excepción: Si un jugador, en su turno, coloca en el centro de la mesa todas las cartas que le quedan y miente al declarar su valor, el jugador siguiente aún tendrá la oportunidad de desconfiar de su jugada, levantando dichas cartas y obligando al jugador mentiroso a llevarse todas las cartas del centro.

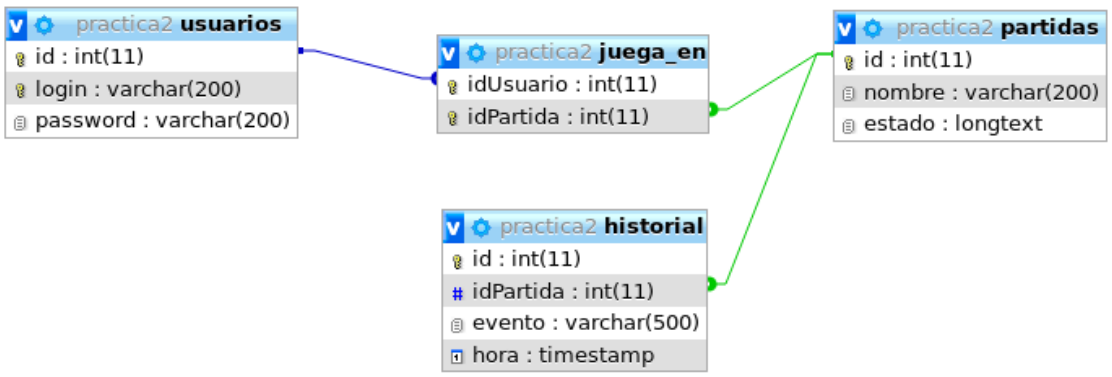

Figura 1: Diseño de la base de datos

Diseño de la base de datos

En esta práctica se sugiere seguir el diseño de la BD mostrado en la Figura 1, pero este diseño puede modificarse si es necesario. Tenemos dos entidades principales: `usuarios` y `partidas`, con una relación muchos-a-muchos entre ambas, representada mediante la tabla `juega_en`. Para cada usuario se almacena su nombre de usuario (*login*) y contraseña. Para cada partida se almacena su nombre y su estado actual. El estado es un texto en formato JSON que contiene la situación actual de la partida: cartas de cada jugador, cartas en el centro de la mesa, jugador que tiene el turno, etc. La tabla `historial`, que almacena los eventos que se van produciendo en cada partida, se utilizará en la última extensión opcional de la práctica.

El esquema que se presenta en la Figura 1 no es puramente relacional, ya que la información contenida en el campo `partidas.estado` no es atómica. En este caso se trata de un esquema híbrido relacional/documental. El estado de una partida se compone de varios componentes relacionados entre sí, por lo que es conveniente tener toda la información relativa al estado en el mismo sitio, en lugar de tenerla dispersa entre varias tablas, que es lo que ocurriría en un modelo relacional «puro». Otra posibilidad consiste en utilizar un esquema puramente documental (por ejemplo, una base de datos *MongoDB*), pero la integración de este tipo de bases de datos con *Node* queda fuera de los objetivos de este curso.

1 Gestión de usuarios y partidas.

La parte obligatoria de la práctica consiste en implementar la creación de usuarios, conexión (*login*) y desconexión (*logout*) del sistema, creación de partidas, y la incorporación de usuarios a partidas existentes.

En el lado del servidor se deberán implementar los siguientes servicios[1]:

- **Creación de un usuario en la base de datos.** El servidor recibirá el nombre de usuario y contraseña, y devuelve un código `201 (Created)` en caso de éxito, o `400 (Bad request)` en caso de que la información sea incorrecta, o el usuario ya exista.

[1]Todos los servicios implementados en la práctica deben devolver un código `500 (Internal server error)` en caso de error de conexión o acceso a la base de datos.

- **Comprobación de identificación de un usuario**. El servidor recibirá un nombre de usuario y contraseña y comprobará si existe en la base de datos un usuario con el nombre y contraseña dados. Devolverá un código `200 (OK)`, junto con el resultado de la comprobación (por ejemplo, un valor booleano). **No debe establecerse ninguna *cookie* o atributo de sesión en esta práctica**.
- **Acceso a las partidas en las que participa un usuario**. El servidor deberá devolver, junto con el código `200`, la lista de identificadores y nombres de las partidas en las que participa el usuario identificado en la petición.
- **Creación de una partida**. El servidor recibirá el nombre de la partida a crear e insertará la entrada correspondiente en la base de datos, devolviendo al usuario un código `201`. Además, el servidor debe insertar el primer jugador de dicha partida, que es el usuario creador de la misma.
- **Incorporación a una partida**. El servidor recibirá un identificador de la partida e insertará al usuario identificado dentro de la lista de jugadores de la misma. Si la partida no existe, se devolverá un código `404`. Si la partida ya está completa, se devolverá el código `400`.
- **Estado de una partida**. El servidor recibe el identificador de una partida y devuelve los nombres de los jugadores actualmente inscritos en la misma, o el código `404` en caso de no existir una partida con el identificador dado.

Salvo en los dos primeros casos (crear usuario e identificar usuario), todas las peticiones que se hagan al servidor han de estar identificadas mediante la cabecera `Authorization`. En caso de que esta cabecera no se incluya, o se haga con un nombre de usuario y contraseña incorrectos, se deberá devolver el código `403 (Unauthorized)`.

Con respecto al lado del cliente, al acceder a la única página HTML de la aplicación deberá mostrarse un formulario solicitando un nombre de usuario y contraseña (Figura 2). Junto a este formulario existen dos botones: `[Aceptar]` y `[Nuevo usuario]`. Al pulsar el primero de ellos se solicita al servidor si el nombre y contraseña se encuentran en la BD. En caso afirmativo, se mostrará en la página el nombre del usuario actual junto con un botón `[Desconectar]`. A partir de este momento y hasta que se pulse el botón `[Desconectar]`, todas las peticiones incluirán la cabecera `Authorization` con el nombre y contraseña del usuario actualmente identificado.

Tras la correcta identificación, debe mostrarse la página de partidas en las que juega el usuario, como se muestra en la Figura 3. Aunque en esta figura se muestran dichas partidas (*Partida amiguetes* y *Familiar*) en forma de pestañas, cualquier otra disposición es admitida. En cualquier caso, se debe permitir al usuario, mediante sendos formularios, crear una nueva partida y de unirse a una partida ya existente introduciendo su identificador.

Al hacer clic en el nombre de una de las partidas a las que pertenece el usuario, deberá mostrarse una vista con la lista de jugadores inscritos en la misma, y un mensaje con el identificador de la misma (Figura 4). El botón `[Actualizar partida]` sirve para volver a pedir al servidor la lista de jugadores actualmente inscritos con el fin de actualizar la lista de la Figura 4.

2 ¡Comienza el juego!

La primera y principal extensión opcional de la práctica consiste en la implementación de la mecánica del juego en sí. Es decir, la gestión de los turnos de los jugadores, de las acciones que pueden realizarse, y del estado de cada partida.

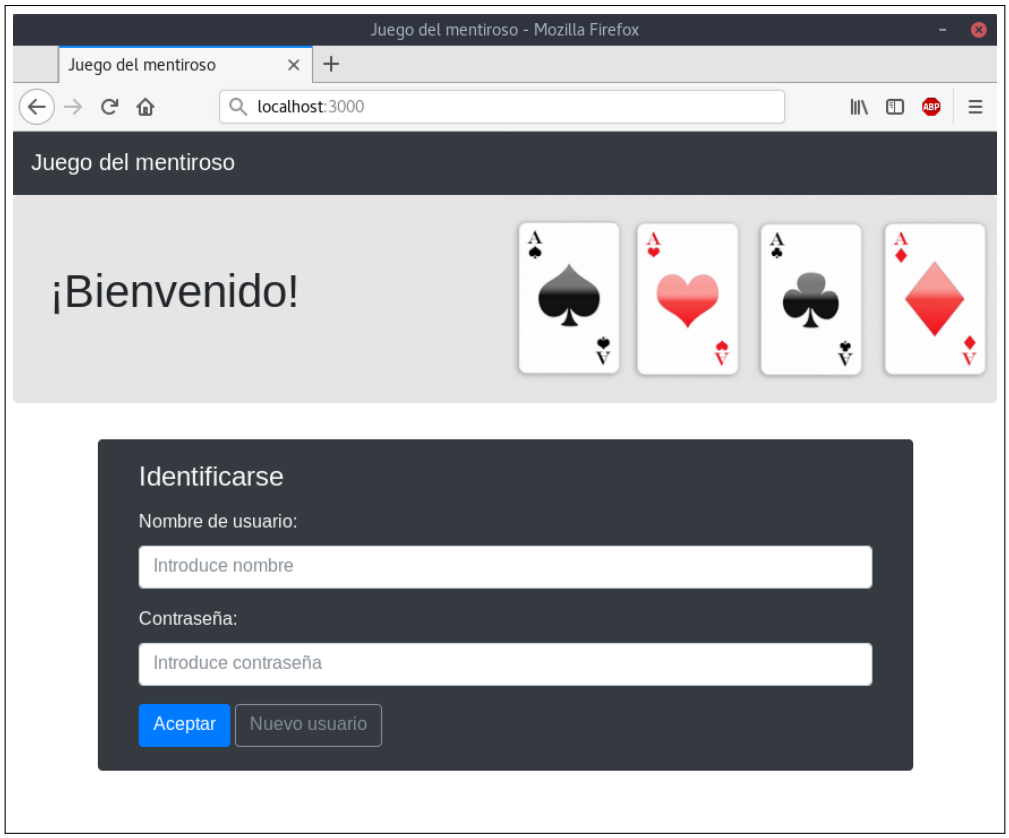

Figura 2: Pantalla de bienvenida e identificación.

En el lado del servidor, deben modificarse los siguientes dos servicios del apartado anterior:

- **Incorporación a una partida**. Si la partida queda completa tras la incorporación del jugador, se dará comienzo a la misma. Para ello se deben repartir aleatoriamente las 52 cartas de la baraja entre los cuatro jugadores, determinar el orden de los mismos, y seleccionar el jugador que comenzará la partida.
- **Estado de una partida**. El servidor, en lugar de devolver únicamente los nombres de los jugadores participantes, debe incluir además la información sobre la partida que sea relevante para el usuario que está realizando la petición. Entre esta información se encuentra: el número de cartas que cada jugador tiene en la mano, las cartas del jugador que realiza la petición, el número de cartas que hay sobre la mesa, el valor que (supuestamente) tienen dichas cartas, cuántas cartas introdujo el último jugador en el centro, etc.

Además, debe incluirse un nuevo servicio:

- **Realizar acción de juego**. El servidor recibirá el identificador de una partida y la acción a realizar por el jugador que envía la petición. Debe comprobar que la partida existe en la BD, que dicha partida no ha terminado, que el jugador que quiere realizar la acción es el jugador que tiene el

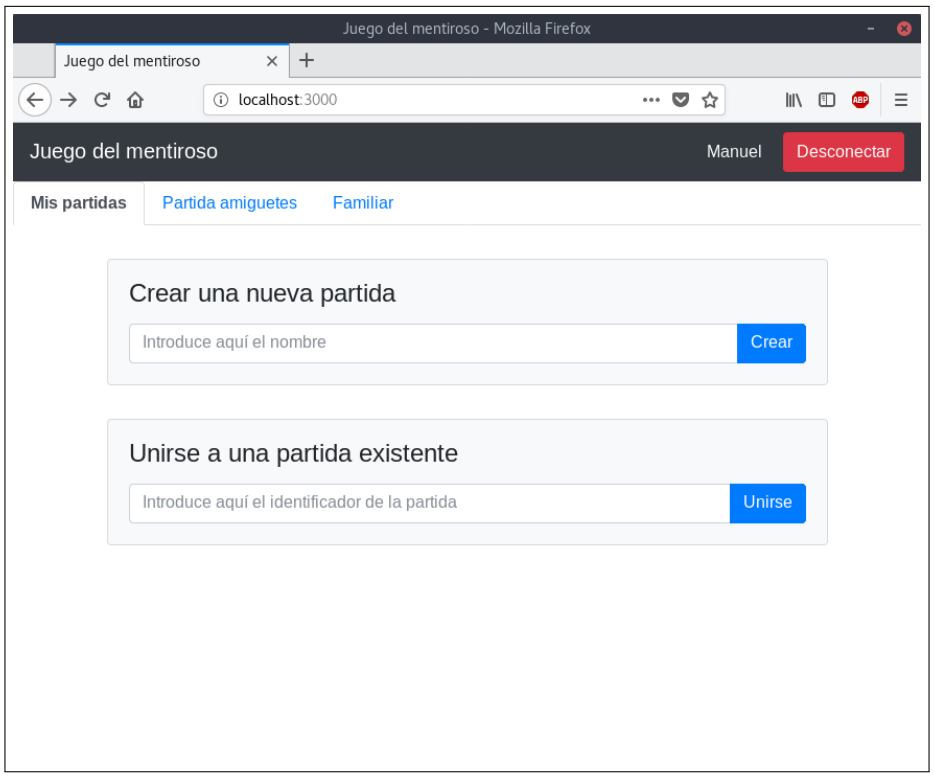

Figura 3: Pantalla de partidas de un usuario.

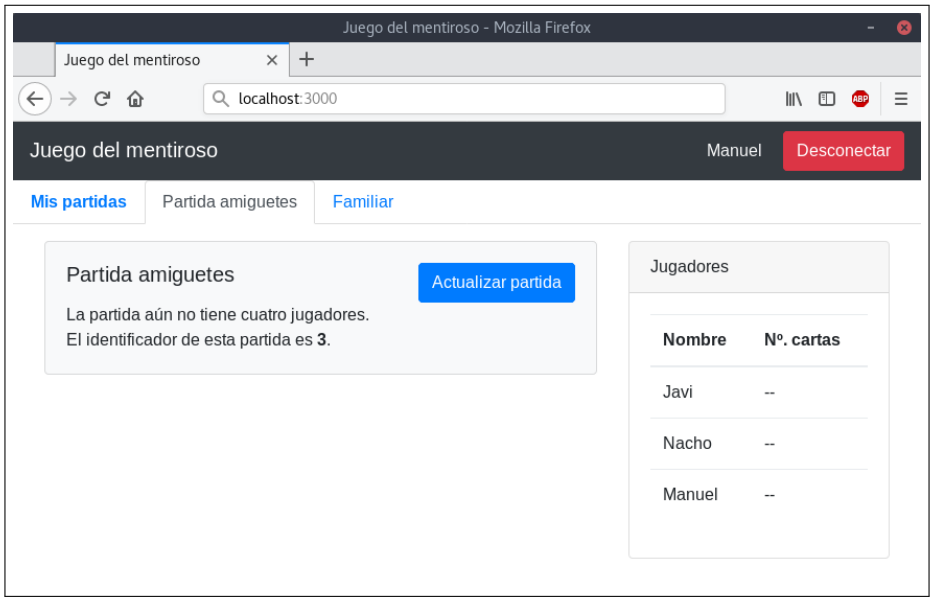

Figura 4: Partida a la espera de un jugador.

turno en ese momento y que la acción que quiere realizar es válida. En este caso, el servidor modificará el estado de la partida conforme a la acción realizada, y responderá al cliente con el resultado de la acción[2].

En el lado del cliente, la vista de partida tendrá el aspecto de las Figuras 5 y 6. Entre la lista de jugadores debe mostrarse resaltado el jugador que tiene el turno. También aparecerán las cartas que hay boca abajo en el centro de la mesa, y su supuesto valor. Por otra parte, se mostrarán las cartas del jugador actual. En el caso en el que el jugador tenga el turno, se mostrarán dos botones: `[Jugar cartas seleccionadas]` y `[¡Mentiroso!]`. El primer botón permite al usuario añadir al centro de la mesa las cartas seleccionadas previamente por él. Además, si en la mesa no había cartas, se le solicitará además un valor (As, 2, 3, 4, ..., J, Q, K) que será el que el usuario afirme al resto de jugadores. El segundo botón permite levantar las cartas introducidas por el jugador anterior, para así comprobar si éste decía la verdad o no.

3 Historial de partida.

El sistema, tal y como está desarrollado hasta ahora tiene algunos problemas de usabilidad. En particular, los usuarios no conocen las acciones realizadas por los demás jugadores. Para solucionar esto se propone, en la última extensión opcional, almacenar en la tabla `historial` la lista de eventos que se producen en cada partida. Entre estos eventos se incluyen:

- La incorporación de jugadores a la partida, y el inicio de la misma.
- Las acciones y resultados de las acciones realizadas por cada jugador.
- Los descartes que se producen cuando un jugador tiene cuatro cartas del mismo valor en su mano.
- La finalización de la partida.
- etc.

Para realizar esta parte, el servidor debe proporcionar un servicio que permita al cliente obtener el historial de acciones de una determinada partida. Por otro lado, la vista de partida mostrada anteriormente en las Figuras 5 y 6 debe extenderse para incluir la lista de eventos ocurridos en dicha partida.

[2] La estructura concreta del JSON con el que se representan las acciones y los resultados de las mismas queda a vuestra elección.

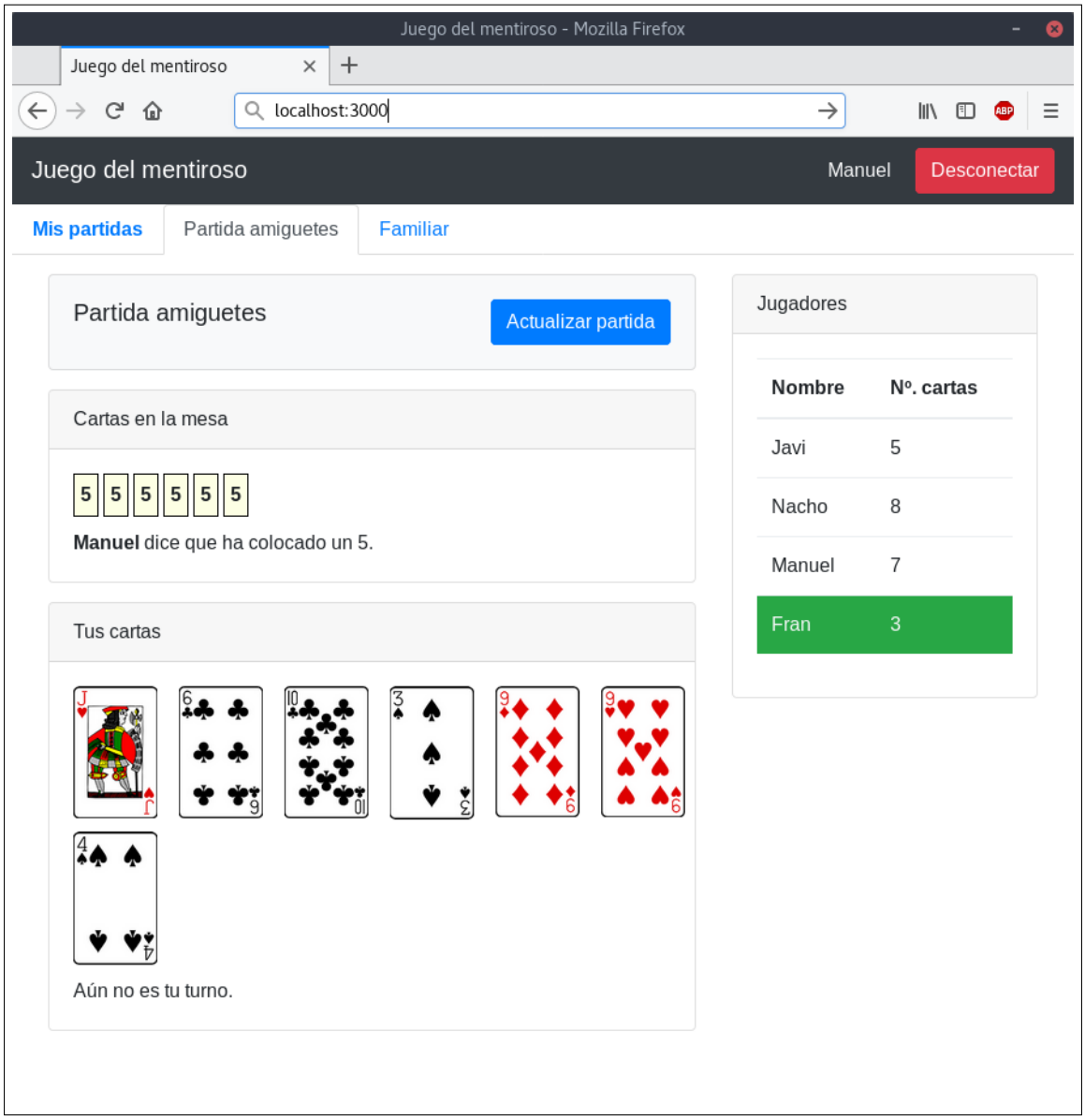

Figura 5: Desarrollo de una partida cuando el jugador actual no tiene el turno.

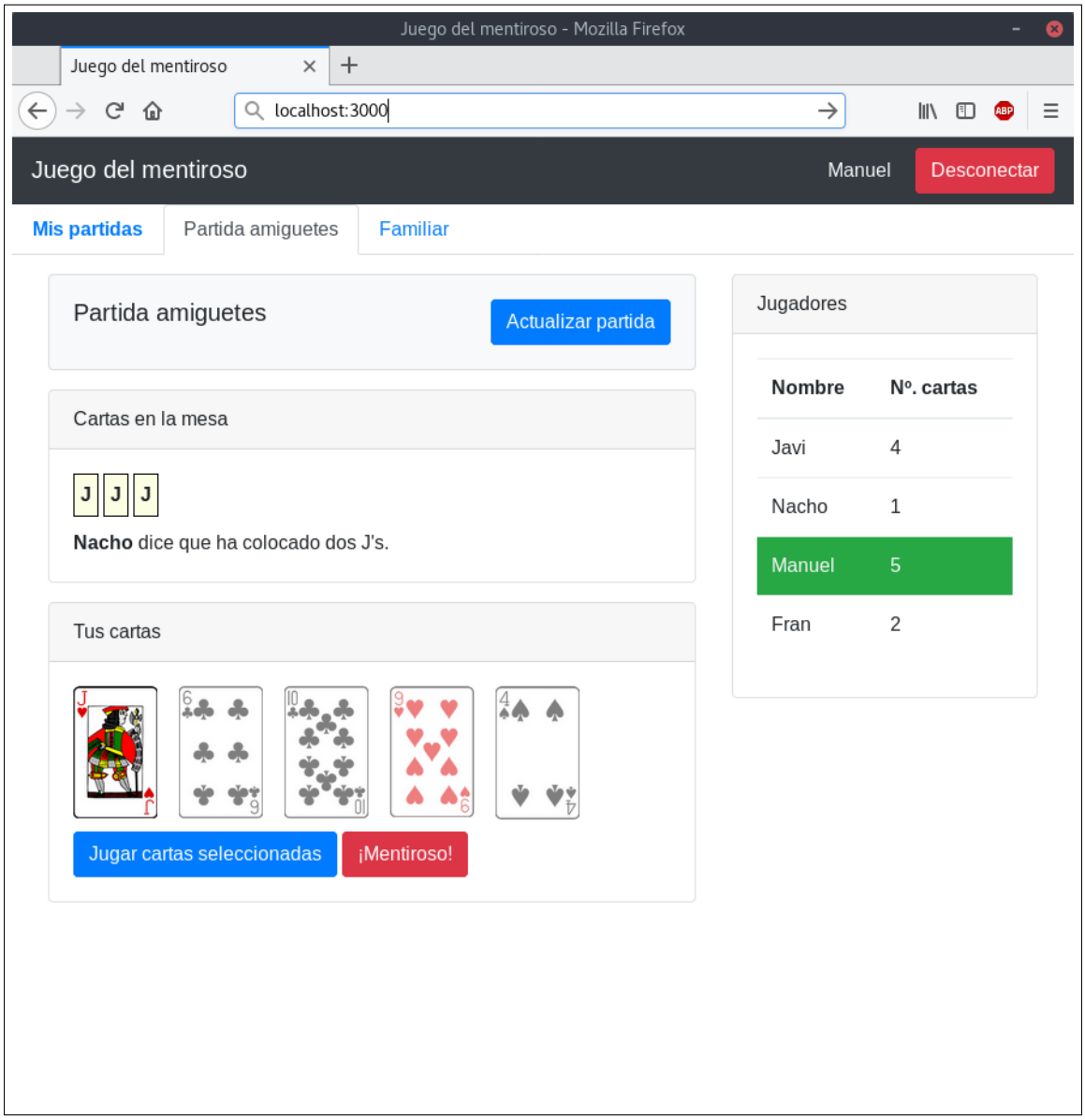

Figura 6: Desarrollo de una partida cuando el jugador actual tiene el turno.