Poker tournaments

Trabajo final de Grado Superior Desarrollo de Aplicaciones Web

Facundo Ezequiel Jadur Astray

ÍNDICE

Introducción

Mi nombre es Facundo Jadur y como proyecto de final de estudios he decidido hacer una aplicación web de mesas de torneos de poker.

He elegido este tipo de proyecto por varias razones:

- Me gusta el poker así que me implico más en ello y estoy más motivado.
- Considero que los videojuegos son complejos y requieren de más lógica que otro tipo de aplicaciones como por ejemplo las de gestión, por lo que me ayuda a mejorar más.
- Buscaba un proyecto el cual no abandonar al presentarlo y además poder usarlo como portafolio para mis futuras candidaturas a empleos.

El proyecto se trata de una aplicación en la que los usuarios se registran y pueden crear y participar en torneos. Tendrán mucha libertad para personalizar el torneo a su medida: eligiendo número de jugadores, cantidad de fichas inicial, tamaño de apuestas, duración de los turnos, coste de entrada, etc.

El usuario puede participar en varios torneos al mismo tiempo y además puede ser espectador de torneos en los que él no esté jugando.

La modalidad de poker escogida es la de Texas Holdem No Limit ya que es a la que estoy acostumbrado.

La aplicación está realizada con Laravel y es una aplicación basada en eventos utilizando websockets para la comunicación en tiempo real.

Tecnologías empleadas

Laravel:

Estuve planteándome utilizar Node.js con el Framework Loopback pero decidí desechar la idea debido a que en la empresa de prácticas en la que estaba estaba usando PHP con Laravel así que decidí utilizar Laravel.

Vue.js:

En la parte del front-end elegí Vue.js ya que Laravel tiene una implementación sencilla de componentes de Vue y además es la tecnología que estaba usando en la empresa de prácticas.

WebSockets:

Al estar realizando una aplicación en tiempo real y el cliente necesita actualizar la información dependiendo de las interacciones de otros usuarios no hay otra opción que utilizar WebSockets. Para ello utilicé un paquete de websockets para Laravel

Docker:

Docker se trata de un sistema de contenedores en los cuales se pueden virtualizar sistemas operativos y servicios. Para el entorno de desarrollo recibí ayuda de mi compañero Rubén Robles el cual me proporcionó una implementación de Laravel, Nginx, PHP, phpmyadmin y Mysql en Docker sobre la cual trabajar.

Guía de instalación

Para instalar la aplicación debemos contar con los siguientes requerimientos:

- Docker
- Make
- npm

Lo primero es conseguir los archivos del proyecto que se obtienen de mi repositorio en GitHub – Enlace

Lo siguiente es configurar los archivos .env tanto el de fuera de la aplicación como el de dentro

```
APP_SLUG=poker
APP_DEBUG=true
APP_ENV=local

DB_PORT=3306
DB_NAME=poker
DB_USER=user
DB_PASS=secret
DB_USER_PASS=secret

HTTP_PORT=80

PHPMYADMIN_PORT=8181
```

Archivo .env principal

```
APP_KEY=base64:52Fj1w/doIIMtzEbivsWqWw9TjpLi6taMgbcw/Pqi4A=

BROADCAST_DRIVER=pusher
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

PUSHER_APP_ID=123456
PUSHER_APP_KEY=ASDFGH
PUSHER_APP_SECRET=ZXCVBN
PUSHER_APP_CLUSTER=mt1

MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```

Archivo .env en el cual omitiremos la primera línea ya que la generaremos con un comando más adelante

Acto seguido ejecutamos “make build” sobre la carpeta principal del proyecto

Luego nos movemos a la carpeta *application* y ejecutamos “npm install” y “npm run dev”

Volvemos a la carpeta principal y ejecutamos “make jumpin” para entrar dentro del contenedor php.

Una vez dentro ejecutamos “composer install”, “php artisan key:generate”, “php artisan migrate” y “php artisan db:seed”.

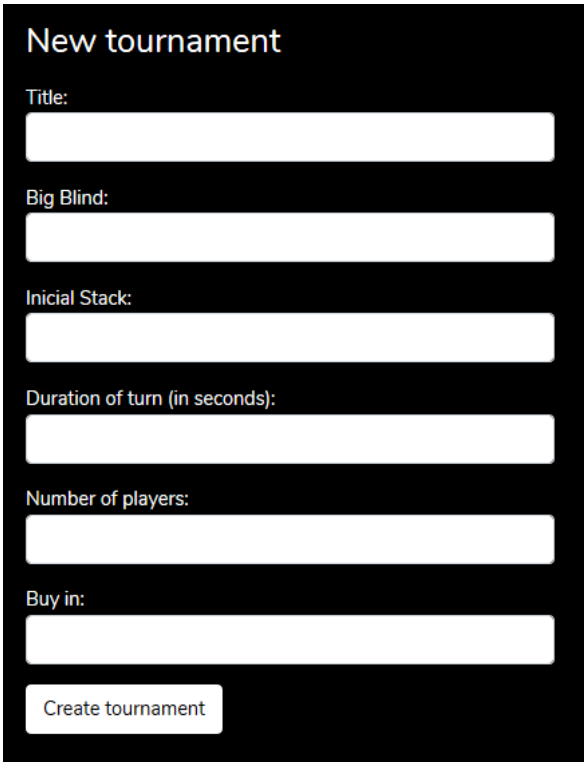
Para arrancar el servidor websockets utilizamos dentro del contenedor php el comando “php artisan websockets:serve”.

Guía de usuario

La página principal de la aplicación es la lista de torneos en la que se verán los torneos activos junto a algunos de los datos del torneo: tamaño de ciegas, stack inicial, coste de la entrada, número de jugadores.

Lista de torneos

En la página de la lista en la esquina inferior derecha se encuentra el botón de crear torneo, con la cual se accede al formulario donde el usuario podrá crear su torneo a medida.

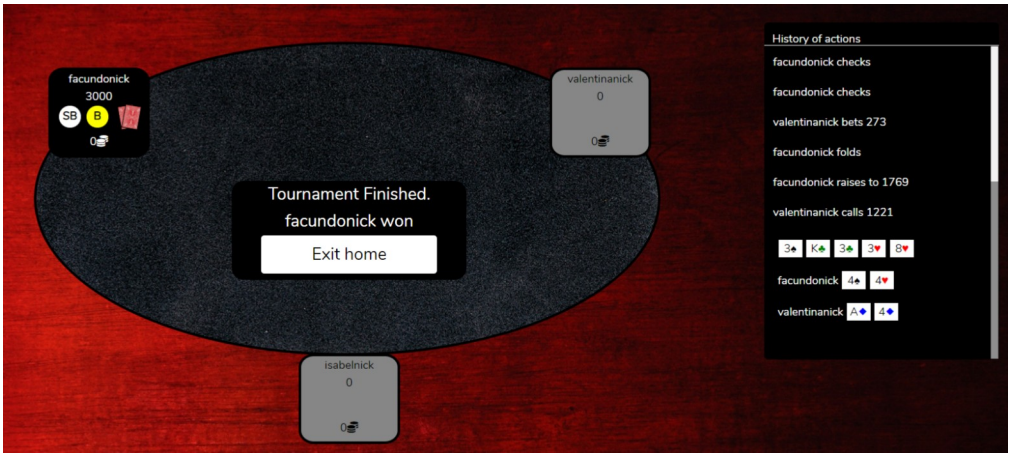


Formulario de creación de torneos

Una vez entremos a una mesa de torneo tendremos que sentarnos en ella usando el botón “Join” en el cual se muestra el número de jugadores actuales.

Mesa en la cual el usuario no se ha sentado todavía

Mesa en la cual el usuario se ha sentado pero aun no está completa



Mesa de torneo finalizado en la que se muestra al ganador

Los jugadores que se queden sin fichas serán eliminados del torneo y se mostrarán con fondo gris

Bote de la ronda

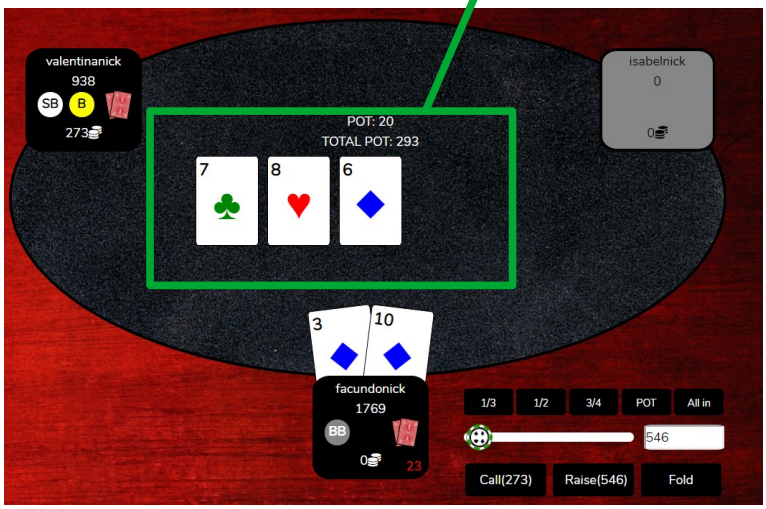
El icono de cartas indica que el usuario está jugando esta ronda. El icono circular indica si el jugador es una de las ciegas o el botón.

Información del jugador:
- Cartas de esta ronda
- Nick
- Stack actual
- Indicadores
- Apuesta actual
- Tiempo de turno

Botones de acción

Historial de acciones

Cartas comunitarias de la ronda actual



Dificultades del desarrollo

Diseño de la base de datos

Tengo dos tablas principales:

- *users*
- *tournaments*

Los usuarios no interactúan directamente con los torneos, sino que lo hacen a través de la tabla *players*.

Cuando un usuario se apunta a un torneo se genera una instancia del modelo Player que contiene la información de ese usuario en ese torneo. Un usuario al participar en varios torneos tendrá varias instancias Player.

- El **torneo** se compone de varias **rondas** representadas con la tabla *rounds*.
- Cada **ronda** tiene varias **rondas de apuestas** (Preflof, Flop, Turn y River) representadas con la tabla *bet_rounds*.
- A su vez cada **ronda de apuestas** tiene **acciones** realizadas por diferentes Players.
- Las **cartas** están en la tabla *cards* y se relacionan con los Players, con las **rondas** y con los **torneos** (para saber qué cartas están disponibles en cada torneo debido a que cada torneo tiene su propia baraja).

Evaluación de manos

Este problema me llevó más tiempo pensarlo que convertirlo a código. Es un sistema diseñado totalmente por mí sin ninguna fuente externa.

Se trata de una función a la cual se le pasan las 5 cartas de la mesa más las 2 cartas de cada jugador que quede en pie.

Cada jugador dispone de su combinación de 7 cartas (las 5 de la mesa más las 2 suyas) la cual pasa por una serie de filtros y se devuelve un array con la valoración de la mano.

Cada filtro es una posible jugada de poker y empieza por los filtros de las jugadas más valiosas hasta las menos valiosas para así poder detenerse ya que al tener una jugada valiosa no es necesario comprobar las demás posibles jugadas.

El primer valor del array es la jugada, que va desde “carta alta” hasta “escalera de color” (valores 1 a 9).

A partir de aquí los siguientes valores definen el valor de la jugada ya que no es lo mismo una pareja de Ases que una pareja de doses. El último valor del array es el id del jugador portador de esa mano.

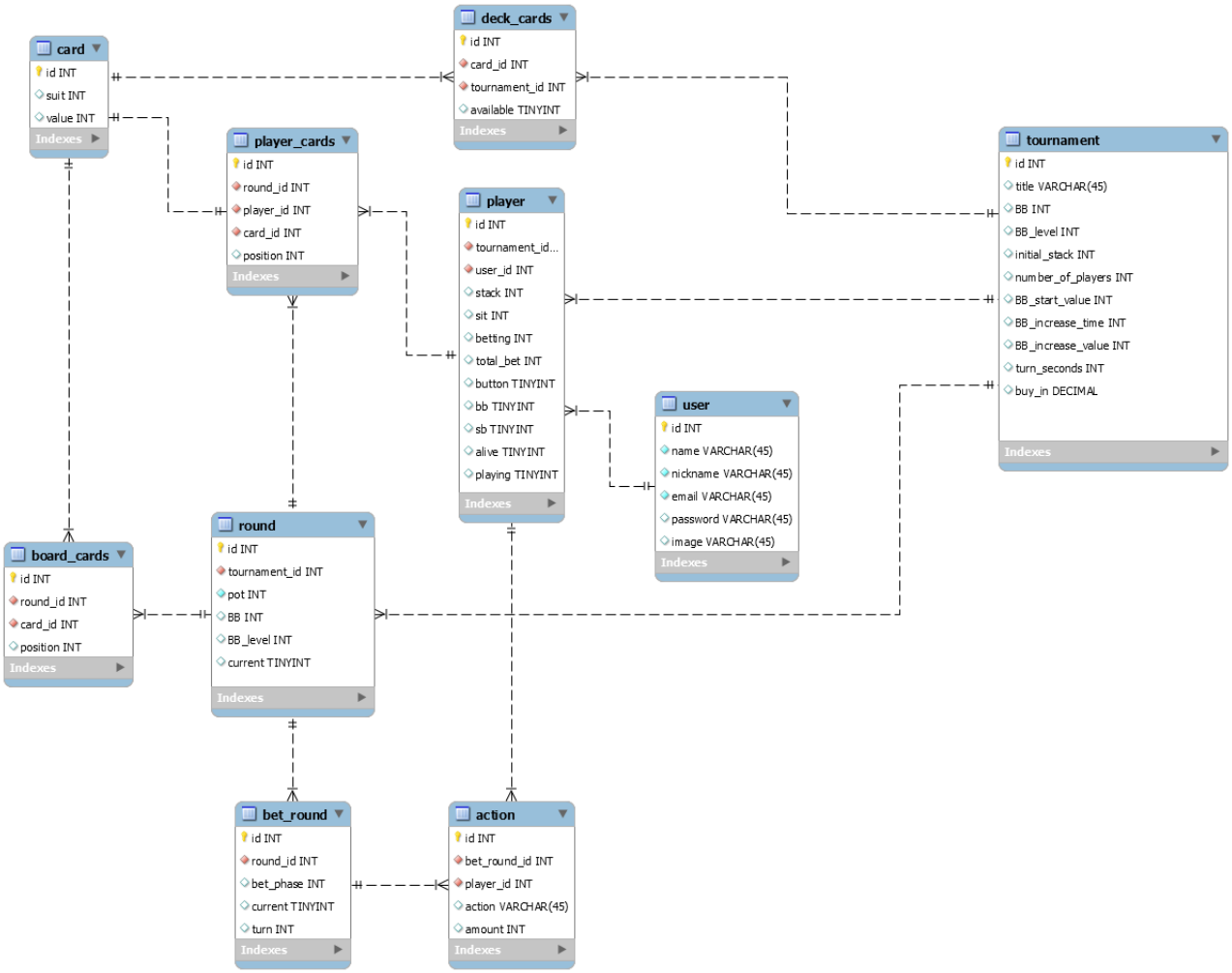


Diagrama EER de la base de datos

Mejoras

He realizado una lista con las mejoras que iré implementando con el tiempo dado que este proyecto no lo voy a abandonar sino que pienso desplegarlo en internet y actualizarlo continuamente.

-Diseño responsive:

El diseño actual es de tamaño fijo y no es cómodo para dispositivos móviles.

-App para móvil:

Es más cómodo jugar en una aplicación que en un navegador, por lo que planeo modificar mi aplicación para crear una API robusta para atacarla desde una app.

-Turnos controlados por Servidor:

Actualmente los turnos están controlados por el cliente y eso no es viable para una aplicación seria. Para ello he de implementar el sistema de Schedules de Laravel.

-Incremento de tamaño de ciegas:

En un torneo de Poker las Ciegas (apuestas obligatorias) van incrementando su valor con el paso del tiempo para hacer que la partida avance y no se estanque, ya que cuantos menos jugadores queden, más fichas tienen. Para ello también haré uso de Schedules.

-Control de AFK:

Actualmente no tengo control sobre qué usuarios están jugando o están AFK, eso lo implementaré cambiando los canales de websockets de públicos a privados.

-Imágen usuarios:

En un futuro me gustaría que cada usuario pudiera tener una imágen de perfil para mostrarse mientras están jugando.

-Cambiar a canales privados:

Los canales por los que los eventos están siendo enviados son públicos. Pienso implementar canales privados para poder crear torneos privados en los que no puedan haber espectadores y se acceda mediante contraseña.

-Animaciones y sonidos:

Se hace complicado seguir el ritmo de la ronda sin animaciones y sonidos que indiquen quién acaba de apostar o cuándo es nuestro turno.

-Personalización de la interfaz:

Los usuarios podrán elegir lenguaje de la aplicación, color del tapiz de la mesa, fondo de pantalla, diseños de cartas, etc.

-Opciones extra:

Añadir más personalización a los torneos, por ejemplo: elegir si queremos jugar con la regla de “Dead button” o “Moving button”, elegir si queremos jugar un torneo en el que además de las ciegas también existan “antes” (apuestas obligatorias para todos los usuarios de la mesa), posibilidad de recompra, etc

-Chat:

Chat para que los jugadores puedan comunicarse entre ellos.

Bibliografía

Para realizar mi proyecto he utilizado las siguientes fuentes:

Serie de video-tutoriales de implementación de WebSockets en Laravel – Enlace

Documentación oficial del paquete laravel-websockets de Beyondcode - Enlace

StackOverflow - Enlace

w3schools – Enlace

Documentación oficial de PHP - Enlace

Documentación oficial de Laravel - Enlace