

ROUND BREAKER

Proyecto final Desarrollo de Aplicaciones Web

I.E.S. Clara del Rey | Grupo DW2E 2020

Autores

- Lucas Ariel González Morales
- Borja González Pereiro
- Pedro González-Tarrío Polo

Tutor

- Xabier Ganzábal García



Unión Europea

Fondo Social Europeo

“El FSE invierte en tu futuro”



Índice

- Descripción del proyecto
 - ¿Qué es Round Breaker?
 - Idea
- Instalación
 - Requisitos
 - Paso a paso
- Base de datos y entidades
 - Diagrama entidad relación
 - Diagrama lógico
 - Descripción de entidades
- Controladores y páginas
 - Controladores
 - Mapa de navegación
- Plantillas y formularios
 - Simplicidad del diseño: Bootstrap
 - Especificidad del diseño: Twig
- Estructura del proyecto
 - config
 - public
 - src
 - templates
 - tests
- Organización del equipo
- Tecnologías usadas
- Manual de usuario
 - Usuario anónimo
 - Registro de usuario
 - Usuario autenticado
 - Administrador
 - Recuperación de contraseña
- Conclusiones

Descripción del proyecto

¿Qué es Round Breaker?

Round Breaker es un sistema de gestión de competiciones de e-sports que abarca inscripciones, formaciones de equipos y gestión de rondas y resultados, además de integración con el sistema de retransmisión Twitch y con su chat.

Este sistema pretende abarcar un nicho poco atendido de la comunidad de deportes electrónicos como es el nivel profesional, centrándose más en la característica social de los torneos, agrupando jugadores de forma aleatoria y poniéndolos en contacto entre sí, resultando en competiciones informales y desenfadadas.

Es una aplicación web desde donde los usuarios pueden registrarse y gestionar su propio perfil e inscripciones a competiciones organizadas por otros usuarios. Además, cada usuario puede también organizar competiciones, obteniendo todos los permisos sobre ellas mediante un panel de organizador donde ver los detalles y editar las opciones y resultados. Desde este panel, el organizador se conectará con el chat de Twitch del canal indicado donde los jugadores inscritos deben confirmar que están presentes para marcar su inscripción en la competición como confirmada.

Una vez haya suficientes inscripciones confirmadas el organizador podrá generar los equipos, las rondas y el árbol de competición a través de un solo botón. La competición entonces se marcará como cerrada y todos los jugadores hayan sido elegidos aleatoriamente para participar podrán acceder a los datos de su equipo para ponerse en contacto con sus compañeros. Podrán ver, además, las credenciales necesarias para unirse a la sala del juego donde se disputarán los encuentros. En cada equipo, uno de los jugadores será elegido al azar como capitán y representará a su equipo, siendo además el único que puede cambiar el nombre del equipo.

Desde la aplicación web podrán consultarse públicamente los resultados y un histórico de competiciones de los usuarios.

Idea

La idea de Round Breaker surge de observar el crecimiento exponencial de los e-sports y cómo han aparecido muchas herramientas enfocadas a competiciones online pero siempre pensando desde un punto de vista profesional con equipos ya formados y de buen nivel.

Nosotros queremos ofrecer un enfoque distinto, orientado a competiciones más informales y fortaleciendo el componente social creando equipos en vivo y poniendo en contacto a sus integrantes.

Round Breaker provee una integración completa con la principal plataforma de retransmisión de e-sports del momento, Twitch. Los participantes confirman su inscripción desde el chat del propio canal de Twitch del organizador, para el que éste tenga visitas garantizadas.

Nuestra aplicación ofrece una forma rápida y ágil de organizar todo lo necesario para que pueda hacerse en vivo mientras se retransmite, para hacerlo así también atractivo para el espectador.

Instalación

Requisitos

- PHP >= 7.4.2
- Composer
- MySQL >= 5.7

Paso a paso

Clonación del proyecto

Puedes clonar todo nuestro proyecto desde nuestro repositorio público de github github.com/bgonp/round-breaker

```
$ git clone https://github.com/bgonp/round-breaker
```

Asegúrate que la carpeta `/var` tiene permisos de lectura y escritura, ya que es necesario para el uso de caché y l

Instalación de dependencias

Una vez con el repositorio clonado puedes instalar las dependencias necesarias usando composer ejecutando est comando desde la raíz del proyecto:

```
$ composer install
```

Base de datos

La aplicación necesita una base de datos MySQL para funcionar (versión mínima 5.7). Crea una base de datos y u usuario con permisos de lectura y escritura sobre ella. Necesitarás el nombre de la base de datos, el nombre de usuario y la contraseña para poder conectar la aplicación.

E-mail

Para el sistema de recuperación de contraseña necesitas una cuenta de correo electrónico y sus credenciales de autenticación para correo saliente.

Configuración del entorno

En la raíz del proyecto existe un archivo `.env` que contiene las variables de entorno. Copia este fichero y renómalo a `.env.local` y sustituye aquí los valores necesarios:

- Entorno de producción (dev para entorno de desarrollo):

```
APP_ENV=prod
```

- Correo saliente:

```
MAILER_DSN=smtps://cuenta:contraseña@servidor:puerto
```

- Base de datos:

```
DATABASE_URL=mysql://usuario:contraseña@servidor:puerto/nombre_db?serverVersion=5.7
```

Inicializar la base de datos

El siguiente paso es ejecutar las migraciones del proyecto para crear todas las tablas y relaciones usando el siguiente comando desde la raíz del proyecto:

```
$ php bin/console doctrine:migrations:migrate
```

Datos de prueba

Puedes cargar datos de prueba en la aplicación para ver su funcionamiento ejecutando el siguiente comando (so entornos dev y test):

```
$ php bin/console doctrine:fixtures:load
```

La contraseña inicial de todos los usuarios generados (incluido admin) es `randompassword`

Usuario administrador

Si no cargas los datos de prueba, puedes generar en base de datos un primer usuario con derechos de administrador a través de la ruta `/install`. Aquí aparece un formulario si en la base de datos no existe ningún usuario.

Tests del proyecto

Existe una carpeta `tests` con una batería de tests que puedes ejecutar para comprobar el correcto funcionamiento de la aplicación.

Configuración de servidor

El servidor que aloje la aplicación debe configurarse para que apunte a la carpeta `public/` que está en la raíz de proyecto. Se necesita el módulo `rewrite` para atender correctamente las peticiones a las rutas. Dejamos una configuración de ejemplo para servidores Apache, es necesaria la instrucción `AllowOverride All` para usar el `.htaccess` ubicado en dicha carpeta `public/`:

```
<VirtualHost *:80>
    DocumentRoot /var/www/round-breaker/public
    <Directory /var/www/round-breaker/public>
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

Versión online

Hemos instalado la última versión de la aplicación en un servidor y está disponible en el dominio roundbreaker.dw2e.online y configurado en entorno de producción para usarlo libremente.

Base de datos y entidades

Diagrama entidad relación

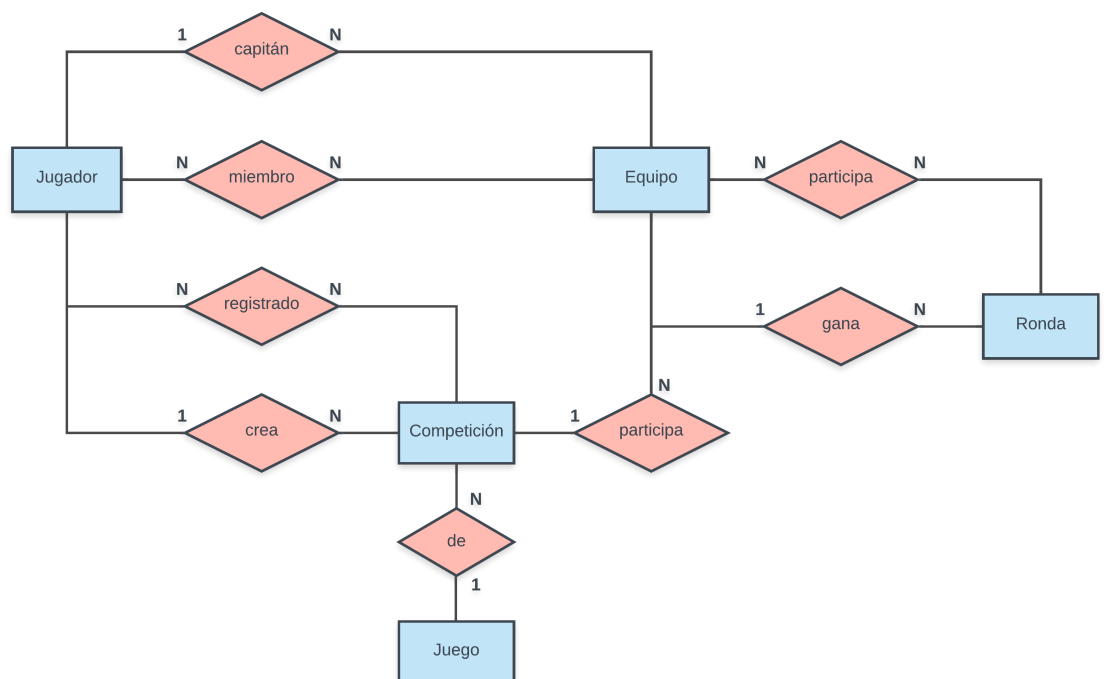
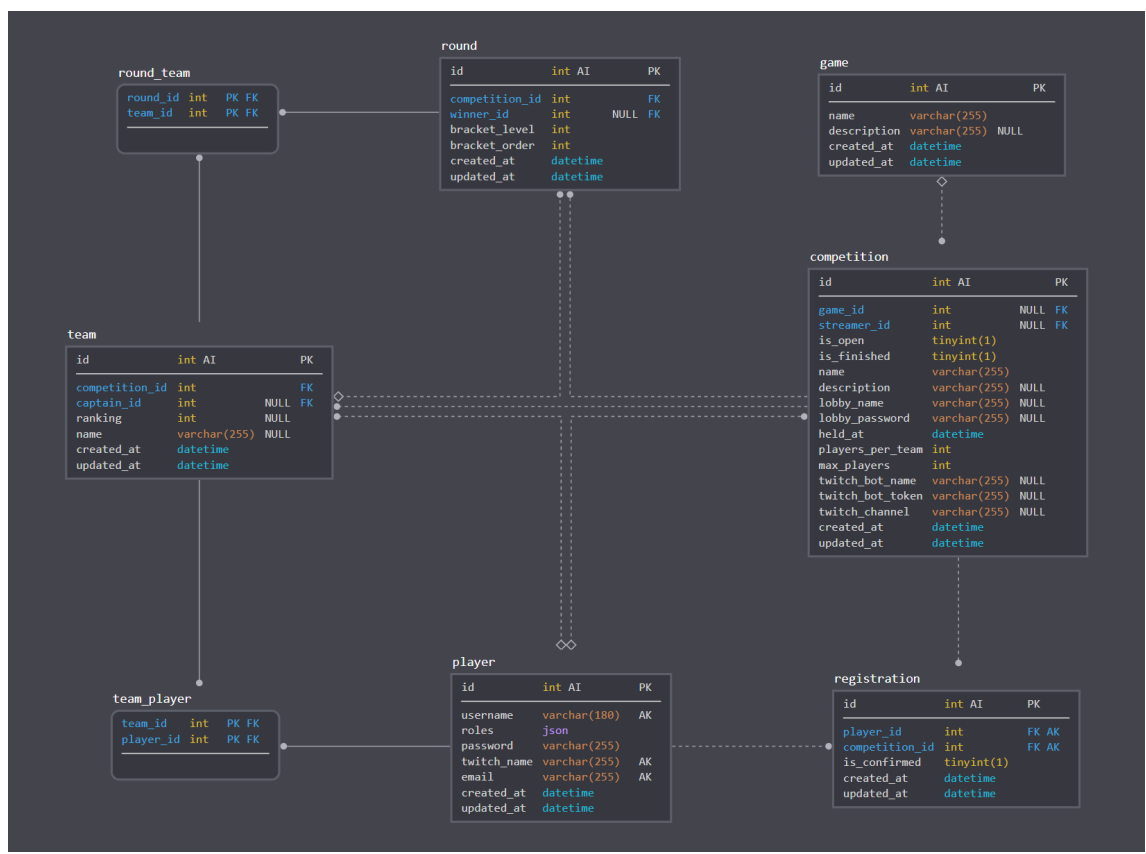


Diagrama lógico



Descripción de entidades

Definimos las entidades que forman el proyecto y sus relaciones. Hemos definido una clase abstracta `Base` de la que heredan el resto de entidades. Esta clase contiene los atributos que son comunes a todas las demás clases (`id` , `createdAt` , `updatedAt`) y un método `equals` para comparar instancias. Además tiene un método que, siguiendo el `lifecycle` de doctrine `PreUpdate`, actualiza en cada escritura el campo `updatedAt` con la hora y fechas actuales. Con esto hemos conseguido evitar la repetición de código y despejar mucho las entidades. Para el correcto mapeo de ORM hemos tenido que etiquetarla como `MappedSuperclass` para que lo interpretase correctamente.

Player

Es la entidad que representa a los usuarios, implementa la interfaz `UserInterface` para ser la entidad sobre la que actúa el sistema de autenticación, implementando los métodos necesarios. El usuario se identifica usando el campo `username` y tanto éste como `email` y `twitch_name` son obligatorios y deben ser únicos (este último debe serlo y que nuestro sistema de competiciones basa su sistema de confirmaciones en la conexión con el chat de Twitch).

Es importante indicar que cada uno de estos jugadores, puede crear competiciones y actuará como organizador misma, teniendo permisos especiales sobre ella (esto no implica que no pueda inscribirse, un usuario puede ser organizador de una competición y participar en la misma).

Game

Es una entidad sencilla, que contiene únicamente nombre y descripción de un juego. Se usa como atributo de competición, ya que las competiciones se disputan en juegos concretos. Solo el administrador puede dar de alta nuevos juegos a los que dar soporte y editarlos o borrarlos (borrarlos solo si no existe ya alguna competición de ese juego).

Competition

Esta es la entidad más compleja y alrededor de la cual se desarrolla toda la lógica de la aplicación. Además de campos sencillos como el nombre (`name`) o la fecha de celebración de la competición (`heldAt`) podemos ampliar información de las siguientes propiedades y relaciones:

- Las propiedades booleanas `isOpen` y `isFinished` definen el estado de la competición. Una competición admite inscripciones hasta que se marca `isOpen` a `false` que es cuando se generan los equipos y puede comenzar competición. Una vez que se ha llegado a la final y se ha declarado un vencedor, la competición pasa a tener `isFinished` a `true` y ya no admite más cambios. Resumiendo:
 - `isOpen true & isFinished false`: Competición programada y aceptando inscripciones.
 - `isOpen false & isFinished false`: Competición en curso, equipos generados y no se admiten más inscripciones.
 - `isOpen false & isFinished true`: Competición finalizada, tiene un ganador y no admite más cambios.
- `lobbyName` y `lobbyPassword` contienen las claves necesarias para conectarse a la sala del juego donde se celebrará la competición. Esta información será solo visible por los jugadores seleccionados para participar.
- `playersPerTeam` y `maxPlayers` son dos enteros que definen el tipo de competición a celebrar, pudiendo ser equipos de entre 1 (individual) y 5 jugadores y un árbol de 2, 4, 8 o 16 equipos.
- `streamer` es la relación `ManyToOne` con la entidad `Player` que representa al jugador que ha creado la competición y que actúa de organizador.
- `game` es la relación `ManyToOne` con la entidad que representa el juego sobre el que se disputa la competición.
- `registrations` es la relación `OneToMany` con la entidad que representa la inscripción de cada jugador a esta competición.
- `teams` es la relación `OneToMany` con los equipos generados para esta competición.

- `rounds` es la relación `OneToMany` con las rondas que componen el árbol de la competición.
- `twitchBotName` y `twitchBotToken` son las credenciales necesarias para conectarse al chat de Twitch del canal indicado en `twitchChannel` y escuchar las confirmaciones de las inscripciones.

Registration

Representa la inscripción de un jugador a una competición. Esta inscripción solo se tendrá en cuenta para la generación de equipos de la competición si se marca la propiedad `isConfirmed` a `true`. Para hacer esto el usuario debe confirmar que está presente en el directo de Twitch donde se retransmita escribiendo en el chat el comando `!confirmo`, una vez el organizador abra las confirmaciones de inscripciones desde el panel de competición.

Team

Representa una agrupación puntual de jugadores en un equipo para participar en una competición. Tal y como es pensado nuestro sistema, siendo inscripciones individuales y equipos aleatorios, un equipo existe únicamente para una competición puntual, es por esto que la relación entre `Team` y `Competition` es `ManyToOne` en lugar de `ManyToMany`. Hay dos relaciones entre `Team` y `Player`:

- `captain` es una relación `ManyToOne` que representa a un único capitán de cada equipo. Este capitán será el portavoz del equipo y será el único con derechos para editarlo.
- `players` es una relación `ManyToMany` que representa a cada uno de los integrantes del equipo. Aunque hemos limitado en el código a un máximo de 5 jugadores por equipo, esto se puede modificar fácilmente si el futuro queremos equipos más amplios.

Además tiene el atributo `name` con el nombre del equipo (editable por su capitán), `ranking` que almacena el puntaje final en la competición y se actualiza según vayan finalizando las rondas, y una relación `ManyToMany` con las rondas en las que juega este equipo.

Round

Representa cada una de las rondas de las que se compone un árbol de competición. Cada ronda enfrenta a dos equipos, pero la relación entre `Round` y `Team` es `ManyToMany` (en lugar de una relación `ManyToOne` para cada equipo porque queremos dar soporte a encuentros entre un mayor número de equipos, aunque de momento la lógica de nuestra aplicación lo limita a 2. Las rondas pertenecen a una única `Competition` y se determina su lugar en el árbol mediante dos atributos:

- `bracketLevel`: un entero que representa el nivel en el árbol (cuartos, semifinales, final...)
- `bracketOrder`: un entero que representa el orden de esta ronda dentro de su nivel.

Con estas dos variables se puede ubicar correctamente cada ronda en el árbol y determina cuál es la siguiente ronda de cada equipo ganador. Por último existe otra relación con `Team`, esta vez `ManyToOne` que es un nullable y pasa a apuntar al equipo ganador una vez que ha terminado la ronda.

ResetPasswordRequest

Es una entidad generada por el bundle `SymfonyCastsResetPasswordBundle` que se encarga de almacenar los tokens y su fecha de caducidad cada vez que un usuario pide recuperar su contraseña.

Controladores y páginas

Controladores

Los controladores se encargan de gestionar las peticiones y devolver respuestas. Hemos recopilado en esta tabla cada una de las rutas existentes en el proyecto con los controladores que las gestionan.

Hemos indicado permisos de player, competitor y streamer aunque todas ellas se refieren a la misma entidad `Player` ya que cada jugador ejerce de `streamer` sobre las competiciones que organiza él mismo y ejerce de `competitor` sobre las inscripciones en las que se registra él mismo o sobre los equipos a los que pertenece. Indicamos `player` las páginas en las que solo se necesita haber iniciado sesión.

Hemos omitido en esta tabla las páginas de recuperación de contraseña y de instalación por no formar parte de navegación principal.

Vista	Permisos	Controlador	Nombre ruta	Descripción	método
SI	public	Main	main	Página principal	GET/POST
SI	public	Competition	competition_list	Lista de competiciones	GET
SI	player	Competition	competition_new	Crear nueva competición	GET/POST
NO	streamer	Competition	competition_delete	Borrar una competición	POST
SI	streamer	Competition	competition_edit	Editar una competición	GET/POST
SI	public	Competition	competition_show	Mostrar detalles de una competición	GET
NO	player	Registration	registration_new	Crear nueva inscripción	POST
NO	competitor	Registration	registration_delete	Borrar una inscripción	POST
NO	streamer	Registration	toggle_confirmation	Confirmar/Desconfirmar una inscripción	POST
SI	public	Game	game_list	Lista de juegos	GET
SI	public	Game	game_show	Ver detalles de un juego	GET
SI	admin	Game	game_new	Crear un nuevo juego	GET/POST
NO	admin	Game	game_delete	Borrar un juego existente	POST
SI	admin	Game	game_edit	Editar un juego	GET/POST
SI	competitor	Team	team_show	Detalles del equipo y credenciales	GET/POST
SI	public	Player	player_show	Información del jugador e histórico	GET
SI	admin	Player	player_edit	Edición de datos de jugador	GET/POST
SI	player	Player	profile	Edición de datos de jugador e histórico propio	GET/POST
NO	streamer	Api	api_winner	Indicar el ganador de una ronda	POST

Vista	Permisos	Controlador	Nombre ruta	Descripción	método
NO	streamer	Api	api_confirm	Confirmar una inscripción	POST
NO	streamer	Api	open_confirmations	Abrir confirmaciones (inicia la escucha de Twitch)	POST
NO	public	Security	app_login	Inciar sesión	GET/POST
NO	player	Security	app_logout	Cerrar sesión	POST
NO	public	Security	user_registration	Registro de usuario	POST

En [este enlace](#) se puede ver una versión más extensa de esta tabla, con las rutas completas, los parámetros que se pasan y la plantilla que renderiza la vista.

Mapa de navegación

La navegación entre las citadas páginas puede realizarse siguiendo el siguiente mapa de navegación, donde se p ver las rutas con las que tiene contacto directo cada página. Hemos indicado como restringido (en naranja) las páginas `competition_edit`, que necesita además de estar logueado ser el creador de la competición, y `team_show` que necesita que seas miembro del equipo que intentas ver, también desde `team_show` el `captain` del equipo tiene un pequeño formulario donde puede renombrarlo.



Plantillas y formularios

Simplicidad del diseño: Bootstrap

Para dotar a toda nuestra página web de un diseño responsive, simple, minimalista e intuitivo de navegar hemos hecho uso del framework de CSS bootstrap y del diseño GRID de 12 columnas lo que nos ha permitido que páginas más complejas y con varios formularios como `competition_edit` no parezcan excesivamente recargadas.

Especificidad del diseño: Twig

A la hora de mostrar los datos procesados por los controladores de nuestra aplicación a los distintos usuarios que van a hacer uso de la misma aprovechamos la potencia del motor de plantillas Twig, desarrollado para PHP para facilitar el desarrollo de vistas en el patrón MVC.

Gracias a él podemos, en función del tipo de usuario, o del estado de la entidad que se está consultando o editar mostrar o no ciertas secciones de HTML. También nos facilita la creación de listas de competiciones o usuarios con enlaces mediante el uso de bucles en las plantillas.

Otra ventaja que hemos aprovechado de Twig es su sistema de herencia, cuyo principal objetivo es la reutilización de código. Gracias a esto hemos podido crear una plantilla base de la cual el resto de plantillas del proyecto han heredado (`{% extends 'base.html.twig' %}`) mientras que alguna otra plantilla de ciertas vistas específicas han contenido en su interior otras plantillas (`{% include 'brackets/bracket_2.html.twig' %}`).

Estructura del proyecto

El proyecto se ha desarrollado en el framework PHP Symfony. Descripción de las principales carpetas y ficheros:

config

Contiene los archivos de configuración del proyecto.

public

Carpeta pública hacia la que van las peticiones y las resuelve a través del fichero `index.php`, también contiene la configuración del directorio de apache (`.htaccess`). Todos los recursos públicos se ubican aquí:

- **css/**: Contiene los archivos de estilos de la web. Es un archivo principal `main.css` y archivos extra para cada tipo de árbol de competición.
- **fonts/**: Contiene las fuentes utilizadas para mostrar los textos de la web (Octin y BrokenGlass)
- **img/**: Imágenes recurso (logo, favicon, imágenes de la home y logos de redes sociales).
- **js/**:
 - **main.js**: Principal funcionamiento y listeners de la interfaz de la web.
 - **twitch.js**: Se carga en la página de edición de competición y se encarga de comunicarse con Twitch usando la librería `tmi.js` y envía peticiones al servidor para confirmar inscripciones.
 - **bracket.js**: Se carga en las competiciones con rondas generadas y se encarga de enviar al servidor información del resultado de cada ronda y actualizar el árbol.
 - **tmi.min.js**: Librería para comunicarse con Twitch usada por `twitch.js`.

src

Carpeta principal con el código fuente de la aplicación, dividida en subcarpetas según la funcionalidad y el tipo de lógica de las clases que contiene:

Migrations

Contiene las migraciones de Doctrine de la base de datos. Las migraciones guardan los cambios a la base de datos nos permiten actualizarla de forma segura a la vez de mantener un historial. Después de varios cambios hemos sustituido muchas de las migraciones que hemos ido generando por una migración con nuestras tablas principales y sus relaciones en su estado final, de forma que hay que ejecutar muchas menos instrucciones sobre la base de datos y agiliza el proceso de puesta en marcha.

Se pueden ejecutar en orden en una base de datos vacía para tener todas las entidades requeridas por la aplicación.

Entity

Clases con las que crear objetos que representan cada uno de los registros de nuestras tablas principales de base de datos con sus relaciones entre ellas. Es importante que exista una fuerte sincronización entre los esquemas de base de datos y estas entidades, para que Doctrine pueda mapearlo todo correctamente. Las entidades principales con las que trabaja nuestra aplicación son: `Game`, `Competition`, `Player`, `Registration`, `Team` y `Round`. El detalle de cada entidad está explicado más extensamente en la sección [Descripción de entidades](#).

Al trabajar con entidades generadas con los datos de base de datos en lugar de directamente con la base de datos consigue disminuir la necesidad de consultas y facilitar el manejo de información.

Repository

Los repositorios son clases destinadas a proporcionar instancias de entidades a partir de datos de la base de datos. Existe un repositorio para cada entidad que se encarga de gestionar el guardado, borrado y obtención de dichas entidades siguiendo diferentes directrices que difieren en cada caso concreto. Cada método de estas clases se encarga de uno de estos casos concretos, y usando estos métodos se consigue agilizar mucho el trabajo en los controladores y evitar la duplicación de código.

Usando repositorios conseguimos desacoplar mucho el proyecto, evitando que los controladores tengan cualquier contacto directo con el tipo de consultas que se hacen a base de datos, haciéndolo mucho más escalable y mantenible.

Los repositorios, además, son gestionados por defecto por el sistema de inyección de dependencias de Symfony, haciendo posible su uso en los controladores simplemente llamándolos en los parámetros de los métodos.

Service

Son clases que contienen métodos con la lógica de funcionalidades concretas de la aplicación y en las que intervienen las entidades:

- `CompetitionService` : Acciones sobre la entidad `Competition`. Tiene la responsabilidad de generar los equipos y las rondas al comenzar una competición (haciendo uso de `TeamService` y `RoundService`) y actualizar su estado en base de datos.
- `TeamService` : Acciones sobre la entidad `Team`. Reside aquí la lógica para generar equipos al comenzar una competición y la de reemplazar jugadores de un equipo.
- `RoundService` : Acciones sobre la entidad `Round`. Contiene la lógica para generar las rondas dada una competición con sus equipos y el hecho de marcar o desmarcar ganadores de rondas, actualizando resultados de las siguientes rondas.

- `PlayerService` : Contiene únicamente un método para editar los datos de un jugador, ya que es una acción (o acción) que se repite en varios controladores.

El inyector de dependencias de Symfony actúa al instanciar los servicios, por lo que hemos añadido constructores para las dependencias necesarias para efectuar la lógica en cada caso.

Controller

Contiene todos los controladores encargados de atender las peticiones web. Cada controlador tiene una ruta asignada a través del sistema de Router de symfony (nosotros hemos usado anotaciones, lo consideramos más cómodo que la configuración a través de yaml o xml) y devuelve una respuesta, ya sea con un `JsonResponse` o renderizan un template concreto para devolver un `Response`.

A cada controlador se le indica el nombre y dirección de la ruta a la que atender, además de los métodos aceptados. El sistema de inyección de dependencias de Symfony actúa sobre cada método de los controladores (al contrario de lo que pasa en servicios), por lo que es fácil usar los repositorios y servicios necesarios en cada caso.

Para la gestión de mensajes de error o éxito hemos usado el método `addFlash` de `AbstractController` (del que heredan todos los controladores), que almacena en session el mensaje etiquetado y lo elimina una vez que es consultado por la plantilla para mostrarlo, asegurando que se muestre solo una vez.

Se puede ver una descripción completa de los controladores con sus características en la sección [Páginas](#).

ArgumentResolver

Como herramienta para hacer más ágil el manejo de entidades en los controladores hemos creado unas clases en este directorio que, al implementar `ParamConverterInterface`, pueden interceptar las llamadas a los controladores pudiendo diseñar así nuestras propias reglas de inyección de dependencias.

Hemos utilizado estas clases para que, cada vez que un controlador indique una entidad en sus parámetros, si la petición tiene un atributo que contenga el nombre de esa entidad seguido de `_id` (por ejemplo `competition_id`, `player_id` ...) entre en acción nuestro resolver y se encargue de buscar la entidad correspondiente a través de su repositorio e inyectarla al controlador que lo haya pedido.

De esta forma hemos podido ahorrarnos mucho código repetido y agilizar el desarrollo de los controladores, evitando en muchos casos las consultas a los repositorios en ellos.

Security

Contiene la clase encargada de la lógica de validar credenciales de usuario y autenticación y efectuar las acciones necesarias en caso de inicio de sesión correcto o incorrecto. Esta clase intercepta las peticiones a la ruta `app_login` de `SecurityController` para realizar las operaciones antes de entrar en juego este controlador. Es aquí, en `SecurityController` donde comprobamos si se ha realizado la autenticación correctamente y manejamos la redirección con o sin mensaje de error a donde proceda.

DataFixtures

Contiene las clases encargadas de cargar los datos de prueba en la base de datos. Existe una clase `Fixture` para cada entidad, excepto en el caso de `BracketFixtures` que hace uso del servicio correspondiente para generar equipos y rondas de una competición por lo que genera ambos en un solo fichero.

Para generar direcciones de correo y nombres de usuario aleatorios y únicos hemos hecho uso de la librería [fzaninotto/Faker](#).

Al haber entidades que tienen relaciones con otras entidades es importante el orden en el que ejecutar estas fixtures para evitar errores de inconsistencia. Es por ello que algunas de estas clases implementan la interfaz `DependentFixtureInterface` que nos permite definir (implementando el método `getDependencies` de esta interfaz) las fixtures que deben haberse ejecutado antes de esta.

Al igual que en los servicios, las dependencias necesarias para la ejecución de estos métodos se inyectan en el constructor y se guardan como propiedades de objeto para hacer uso de ellas.

Esta funcionalidad no está disponible si el entorno del proyecto está configurado como producción.

Exception

Hemos creado varias excepciones propias que recogen errores que pueden ocurrir durante la ejecución de nuestra aplicación. Estas excepciones pueden ser capturadas por los controladores para mostrar su mensaje a través de `addFlash` si procede.

TwigExtension

Hemos definido extensiones accesibles desde los templates de twig para facilitar su uso en el diseño de plantillas.

Estas funcionalidades son:

- `DateTimeExtension` : Proporciona funcionalidades para formatear objetos `DateTime` a la forma correcta para pasarlos a inputs de tipo `datetime`.
- `GetTeamsNumberExtension` : Agiliza una operación común que hay que realizar al mostrar información sobre competiciones a través de la función `getTeamsNumber`.
- `CompetitionsCountExtension` : Crea una variable global con el número de competiciones existente en base de datos accesible desde cualquier template en cualquier momento. Usado para poner en el `header` un mensaje: ¡Más de X torneos organizados!.

Form

Contiene las clases encargadas de generar los formularios de cambio de contraseña y petición de cambio de contraseña gestionados por el bundle `SymfonyCastsResetPasswordBundle`.

templates

Modelos twig que reciben datos de los controladores y proporcionan una vista a la aplicación. Cada ruta tiene su propia vista, la cual renderiza los datos adecuadamente. El uso de archivos twig nos permite enseñar y esconder código html de forma inteligente, dependiendo del tipo de usuario, sus permisos o el estado de la entidad que se esté consultado o editando (Competición abierta o cerrada, finalizada o no...)

tests

Para la revisión del correcto funcionamiento de nuestra aplicación hemos diseñado una batería de tests que comprueban las funcionalidades de cada página para cada caso que pueda ocurrir. Una de las razones por las que hemos migrado a Symfony 5.1 es que se añade un método `loginUser` a la clase `KernelBrowser` que facilita el inicio de sesión a realizar en el cliente proporcionado por `WebTestCase` para cada test y así poder probar las funcionalidades desde los diferentes roles posibles.

Para facilitar la confección de los tests hemos creado una clase `TestBase` de la que heredarán todos nuestros tests. Esta clase hereda de `WebTestCase` e implementa funcionalidades que usaremos durante toda nuestra batería de pruebas. Además de alias para acortar llamadas a los métodos de `request`, `loginUser` y `followRedirect` (entre otros) esta clase nos ofrece:

- Un método `getRepository` que, al pasarle el nombre de una clase como string, nos proporciona su repositorio. Muy útil ya que hay muchas llamadas a repositorios durante los tests y no son de acceso trivial.
- Un método `reloadFixtures` que, usando el bundle `liip/test-fixtures-bundle` que proporciona el trait `FixturesTrait`, permite recargar todas las fixtures. Este método es llamado al finalizar los tests que escriben base de datos para asegurar el correcto funcionamiento de los tests posteriores.
- Un método privado `initDatabase` ejecutado una sola vez que se encarga de inicializar la base de datos, con esquema actual y con los datos de las fixtures.

Los tests que realizamos comprueban, de cada página y desde cada rol posible, que se permite efectuar las operaciones solo cuando procede y que la ejecución de estas operaciones termina con el resultado esperado. Ante cada push al repositorio remoto, cada uno de nosotros ejecuta estas pruebas para comprobar que todo sigue funcionando correctamente y diseñar nuevos tests si procede.

Organización del equipo

El código completo del proyecto se puede consultar en el repositorio de GitHub [bgonp/round-breaker](https://github.com/bgonp/round-breaker), donde se puede consultar tanto el código final como los diferentes commits y ramas que hemos ido haciendo los tres integrantes del equipo.

Hemos dividido el proyecto en áreas generales para organizar el equipo, cada uno de nosotros será el principal encargado de algunas de estas áreas, aunque los tres colaboramos en todas ellas:

- Base de datos (Borja)
- Entidades (Borja)
- Repositorios (Borja)
- Controladores (Lucas)
- Back-end (Lucas)
- Seguridad (Lucas)
- Diseño (Pedro)
- Formularios (Pedro)
- Templates (Pedro)
- Integración con Twitch (Borja)
- Tests (Borja)

Flujo de trabajo

Los tres somos autores del repositorio remoto alojado en **GitHub**, teniendo los tres permisos para editarlo. De esta forma, hemos organizado el flujo de trabajo para que cada uno de nosotros se cree una rama nueva para una característica de la aplicación, la desarrolle en ella y cuando estuviera suficientemente estable, hiciera un `pull` para traer el estado remoto actual y resolviera los posibles conflictos con la rama master. Una vez resueltos esos conflictos, hacemos un `merge` para unir nuestra rama a la rama master y `push` al repositorio remoto.

Haciéndolo así y estando en contacto en todo momento no hemos tenido demasiados problemas para trabajar simultáneamente en un mismo repositorio.

Tecnologías usadas

PHPStorm y VSCode

El editor escogido por Lucas y Borja para desarrollar código ha sido **JetBrains PHPStorm** por considerarlo el editor más completo y con mayor integración con Symfony. Pedro ha preferido usar **Microsoft Visual Studio Code** al ser más flexible y adaptado al diseño y desarrollo de interfaces web.

GitHub

Se ha creado un repositorio en GitHub donde subir el proyecto con los tres integrantes del equipo con derechos escritura para poder editar todos en paralelo como se ha explicado en [flujo de trabajo](#).

Symfony

El proyecto se ha desarrollado en PHP usando el framework Symfony, comenzando en la versión 5.0.7 y actualizando posteriormente a 5.1.0.

DebugBundle

Proporciona herramientas de ayuda durante el desarrollo, accesibles desde la barra mostrada en cada página de aplicación (gracias a `symfony/profiler-pack`). Nos ha servido para hacer un seguimiento más profundo de los errores y comprobar el buen funcionamiento y las llamadas a base de datos que se hacían.

TwigBundle

Gestor de templates para generar las vistas.

DoctrineFixturesBundle

Para generar datos de prueba útiles para probar la aplicación y ejecutar los tests basados en estas fixtures.

LiipTestFixturesBundle

Proporciona un trait con herramientas para cargar las fixtures en tiempo de ejecución de la aplicación, usado para cargar los datos de prueba durante la ejecución de los tests.

SymfonyCastsResetPasswordBundle

Para crear el sistema de recuperación de contraseñas a través de tokens con caducidad establecida en 2 horas.

friendsofphp/php-cs-fixer

Para unificar el estilo del código de los tres integrantes del equipo y prevenir errores aplicamos PHP CS Fixer antes de cada push al repositorio. La configuración de las reglas a aplicar está en el fichero `.php_cs.dist` en la raíz del proyecto.

beberlei/doctrineextensions

Para añadir a las consultas efectuadas en los repositorios funcionalidades extra de MySQL que no están disponibles por defecto desde el query builder, como `RAND()` para devolver registros ordenados aleatoriamente.

fzaninotto/faker

Para generar datos aleatorios como nombres de usuario. Lo hemos usado en las fixtures para crear los datos de prueba y al generar equipos para ponerles un nombre inicial.

sensio/framework-extra-bundle

Herramientas para controladores, necesario para resolver entidades en los parámetros de los métodos de los controladores cuando la petición contiene un `id`.

symfony/mailer

Para enviar correos, usado en el sistema de recuperación de contraseñas.

symfony/phpunit-bridge

Necesario para ejecutar los tests correctamente en Symfony usando PHPUnit.

tmijs/tmi.js

Para la integración con el chat de twitch hemos usado la librería [tmi.js](#) para conectar con el websocket del chat de twitch. Usado para confirmar inscripciones de jugadores.

bootstrap

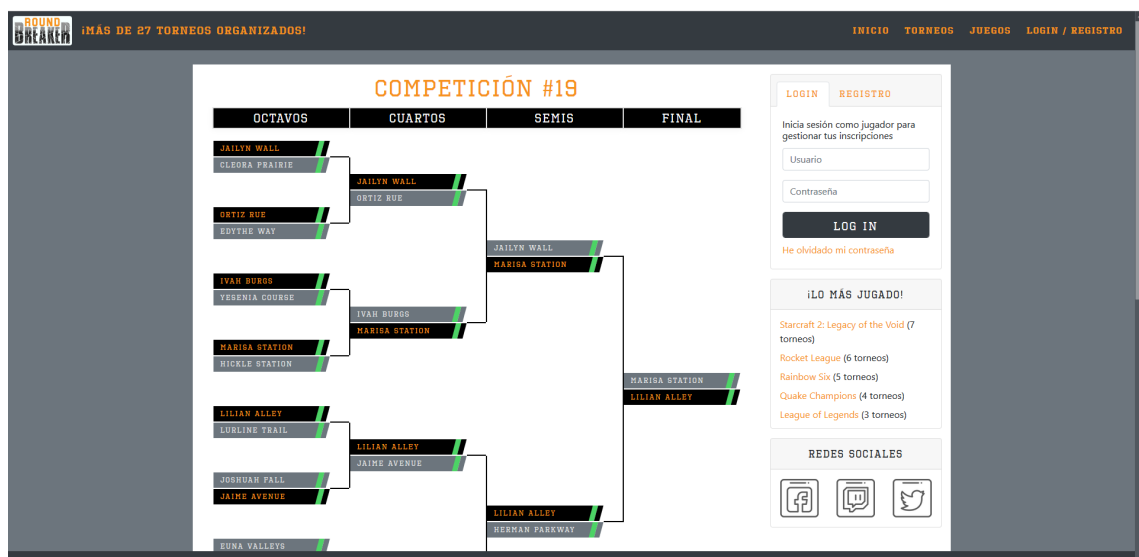
Herramienta para diseñar toda la parte front-end de la aplicación web responsive.

fontawesome

Set de iconos ligero para mostrar y ayudar en la navegación.

Manual de usuario

Este manual cubrirá las dudas que puedan sugerir al navegar nuestra página por primera vez. Empezaremos por la página de inicio.



Esta es la primera página que verá un nuevo usuario. Desde aquí podrá hacer login/registro y acceder a las 3 ram principales de la página: **Competiciones**, **Juegos** y su **Perfil**. Además tiene acceso a links a nuestras redes sociales: los juegos más jugados en la plataforma. Lo competición mostrada en la portada es un torneo elegido aleatoriamente entre todos los torneos de entre 4 y 16 equipos finalizados hasta el momento.



IMÁS DE 27 TORNEOS ORGANIZADOS!

[INICIO](#) [TORNEOS](#) [JUEGOS](#) [LOGIN / REGISTRO](#)

TORNEOS

FILTRAR POR JUEGOS

COMPETICIÓN #28 (STARCRAFT 2) - 26/06/2020 12:56

COMPETICIÓN #27 (STARCRAFT 2) - 14/06/2020 14:31

COMPETICIÓN #26 (STARCRAFT 2) - 05/06/2020 23:53

COMPETICIÓN #25 (STARCRAFT 2) - 22/05/2020 21:26

COMPETICIÓN #24 (STARCRAFT 2) - 16/05/2020 22:20

COMPETICIÓN #23 (STARCRAFT 2) - 08/05/2020 09:18

COMPETICIÓN #22 (STARCRAFT 2) - 26/04/2020 19:21

COMPETICIÓN #21 (ROCKET LEAGUE) - 18/04/2020 04:33

COMPETICIÓN #20 (ROCKET LEAGUE) - 04/04/2020 07:51

COMPETICIÓN #19 (ROCKET LEAGUE) - 30/03/2020 04:42

COMPETICIÓN #18 (ROCKET LEAGUE) - 15/03/2020 15:17

COPYRIGHT 2020: THE ROUNDBREAKER TEAM ©



IMÁS DE 27 TORNEOS ORGANIZADOS!

[INICIO](#) [TORNEOS](#) [JUEGOS](#) [LOGIN / REGISTRO](#)

TORNEOS

STARCRAFT 2

COMPETICIÓN #28 - 26/06/2020 12:56

COMPETICIÓN #27 - 14/06/2020 14:31

COMPETICIÓN #26 - 05/06/2020 23:53

COMPETICIÓN #25 - 22/05/2020 21:26

COMPETICIÓN #24 - 16/05/2020 22:20

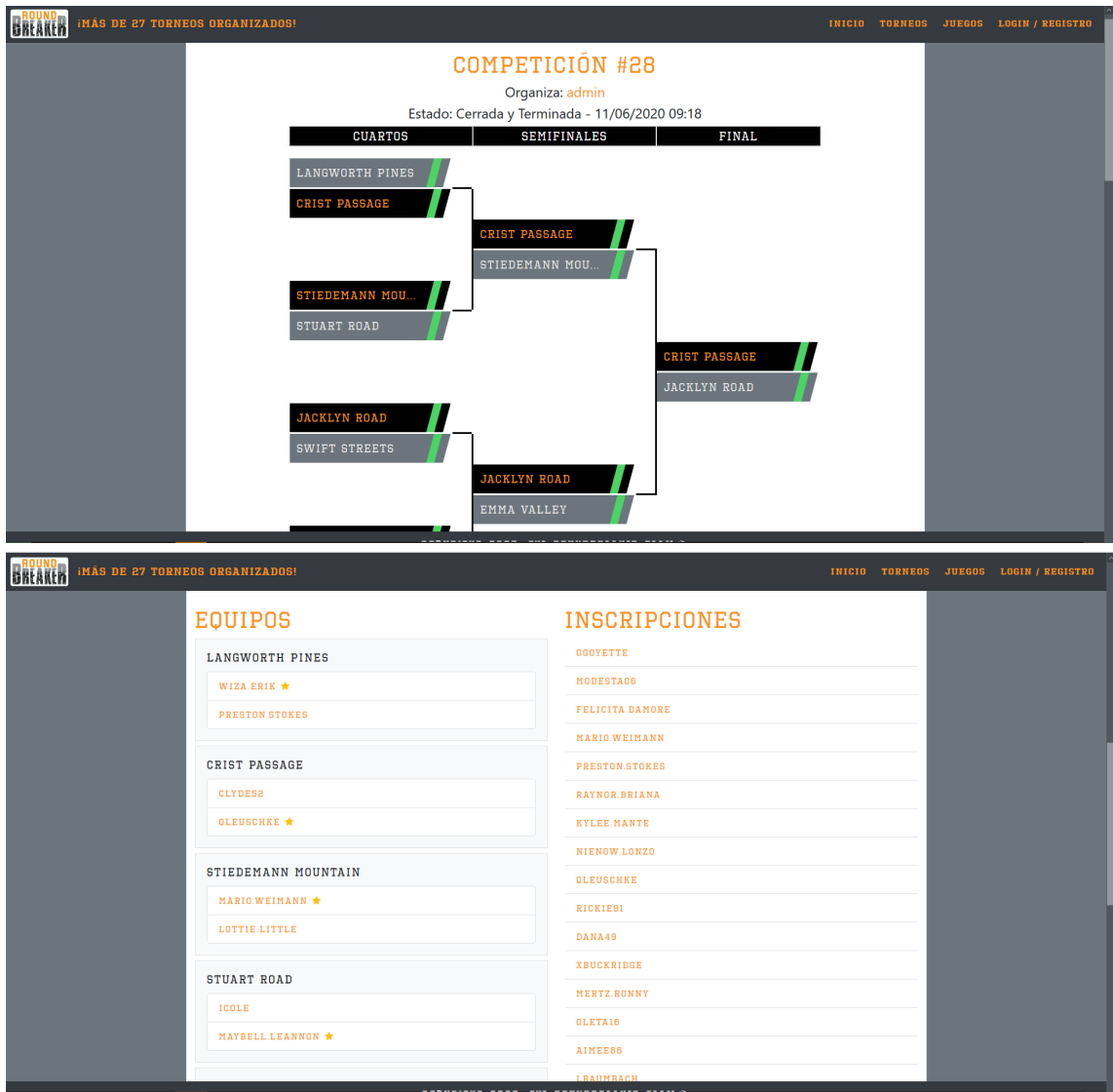
COMPETICIÓN #23 - 08/05/2020 09:18

COMPETICIÓN #22 - 26/04/2020 19:21

COPYRIGHT 2020: THE ROUNDBREAKER TEAM ©

Usuario anónimo

Como usuario anónimo se pueden ver todas las competencias que se están jugando y se han jugado en la página **Torneos**. Esta disponible el juego de la competición y la fecha en la que se disputará o ha disputado. Las competencias que ya tengan sus equipos generados y por lo tanto no admitan más inscripciones, aparecen con ligero sombreado gris como se puede ver en la imagen anterior. Desde esta página el usuario puede hacer click y una competición con más detalle. Utilizando el filtro, el usuario tambien puede filtrar por juego para tener una lis más específica.



Desde la página de un torneo se puede ver el estado de la competición, los brackets actuales, los equipos e inscripciones. Sin hacer login el usuario no podrá unirse a una competición, así que lo haremos antes de continuar.

Registro de usuario

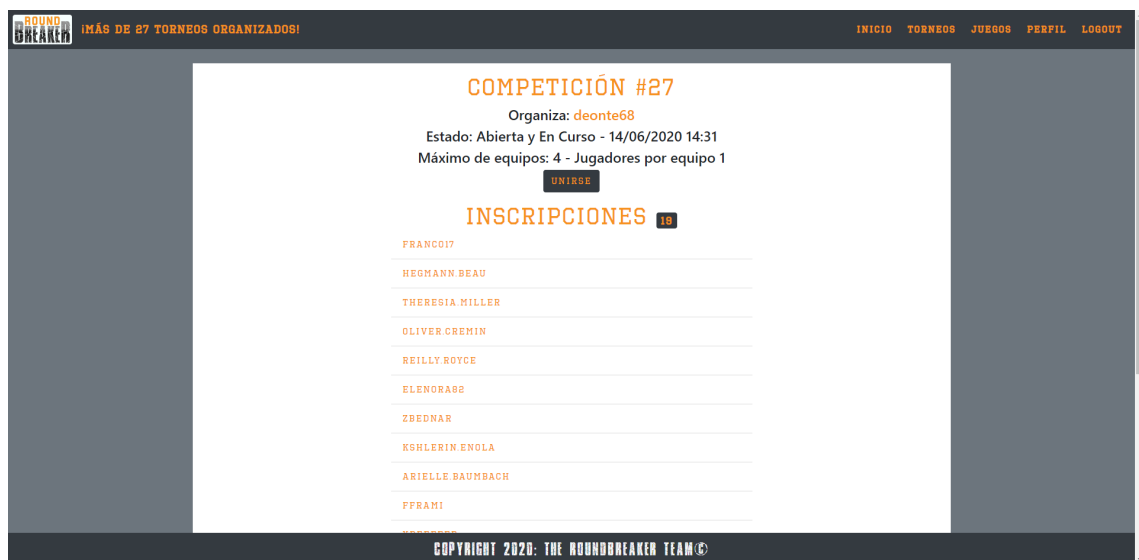


The screenshot shows a registration form on a dark-themed website. At the top, there are navigation links: INICIO, TORNEOS, and JUEGOS. The form has two tabs: LOGIN and REGISTRO, with REGISTRO being the active tab. Below the tabs, there is a prompt: 'Regístrate y participa en innumerables torneos!'. The form contains five input fields: Usuario, Contraseña, Repetir Contraseña, Email, and Twitchname. A dark button labeled REGISTRAR is positioned below the fields. At the bottom of the form, there is a small orange logo and a button labeled ¡LO MÁS JUGADO!.

Rellenando este formulario el usuario se da de alta en nuestra plataforma. Desde la página de inicio podrá hacer inmediatamente después.

Usuario autenticado

Al haber iniciado sesión, podemos ver que tenemos opciones en la página de torneo si nos vamos a una competición abierta:



The screenshot displays the 'COMPETICIÓN #27' page. The header includes the Round Breaker logo, a notification '¡MÁS DE 27 TORNEOS ORGANIZADOS!', and navigation links: INICIO, TORNEOS, JUEGOS, PERFIL, and LOGOUT. The main content area shows the tournament details: 'Organiza: deonte68', 'Estado: Abierta y En Curso - 14/06/2020 14:31', and 'Máximo de equipos: 4 - Jugadores por equipo 1'. A dark button labeled UNIRSE is present. Below this, the 'INSCRIPCIONES' section shows a list of 19 registered players: FRANCO17, HEGMANN BEAU, THERESIA MILLER, OLIVER CREMIN, REILLY ROYCE, ELENORAB2, ZBEDNAR, KSHLERIN ENOLA, ARIELLE BAUMBACH, and FFRAMI. The footer contains the copyright notice: 'COPYRIGHT 2020: THE ROUND BREAKER TEAM ©'.

Ahora podemos ver el número de jugadores ya inscritos y una lista de los mismos, además de los detalles del torneo, el número de equipos máximo y el número de jugadores por equipo. Se presenta también un botón para **Unirse**, hacemos click en este botón nos inscribiremos en la competición, de esta forma tendremos la oportunidad de competir cuando el creador asigne los equipos aleatoriamente. Ahora que ya tenemos cuenta, tenemos nuestro Perfil:

ROUND BREAK MÁS DE 27 TORNEOS ORGANIZADOS! INICIO TORNEOS JUEGOS PERFIL LOGOUT

WALTON37

USERNAME
walton37

CONTRASEÑA
Contraseña

REPETIR CONTRASEÑA
Repetir contraseña

EMAIL
boyle.jaquefin@cruickshank.com

TWITCHNAME
walton37

EDITAR MIS DETALLES

TUS TORNEOS

No has organizado ningún torneo. [Crea un torneo!](#)

REGISTROS

COMPETICIÓN #25 (STARCRAFT 2- LEGACY OF THE VOID)	14/05/2020 03:27
COMPETICIÓN #19 (ROCKET LEAGUE)	06/03/2020 06:49
COMPETICIÓN #16 (ROCKET LEAGUE)	16/02/2020 04:08
COMPETICIÓN #14 (RAINBOW SIX)	29/01/2020 16:37
COMPETICIÓN #10 (QUAKE CHAMPIONS)	19/12/2019 03:11
COMPETICIÓN #09 (QUAKE CHAMPIONS)	03/12/2019 16:18
COMPETICIÓN #04 (LEAGUE OF LEGENDS)	13/10/2019 17:36

TUS EQUIPOS

ALFRED VILLAGE (QUAKE CHAMPIONS) - RANKING: 7

En nuestro perfil podemos editar nuestros datos, ver nuestros torneos, inscripciones y equipos. Accederemos a uno de nuestros equipos para ver más información:

ROUND BREAK MÁS DE 27 TORNEOS ORGANIZADOS! INICIO TORNEOS JUEGOS PERFIL LOGOUT

ALFRED VILLAGE

WALTON37

BODE JAKOB

DARLENE76

NHAROUARDT

COMPETICIÓN:

COMPETICIÓN #10

ORGANIZA: ZMCKENZIE

CELEBRADO EN: 19/12/2019 03:11

TEAM RANKING: 7

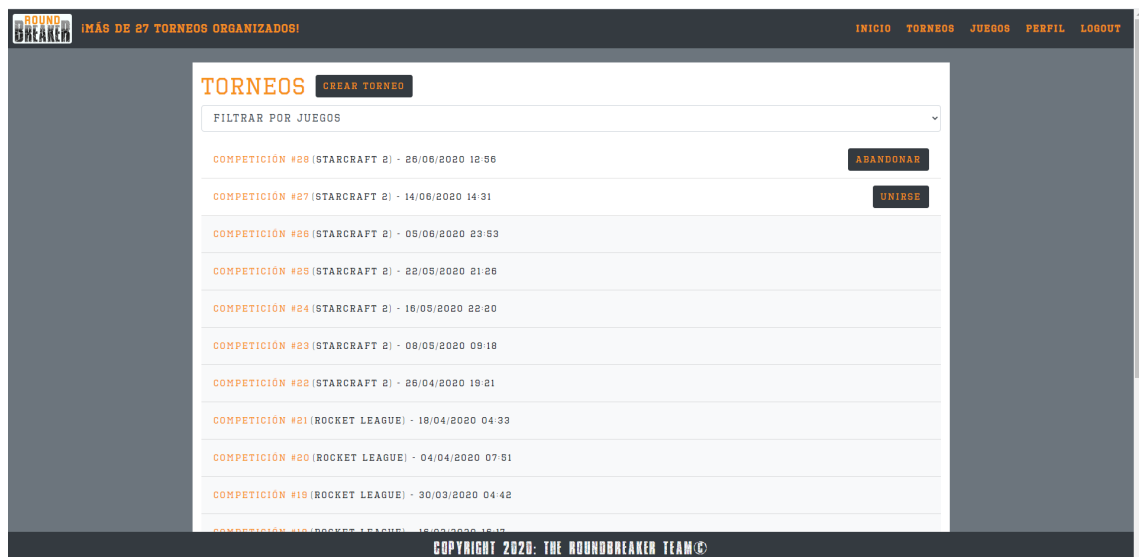
CERRADA Y TERMINADA

ERES CAPITÁN. CAMBIA EL NOMBRE A TU EQUIPO

Alfred Village

CAMBIA!

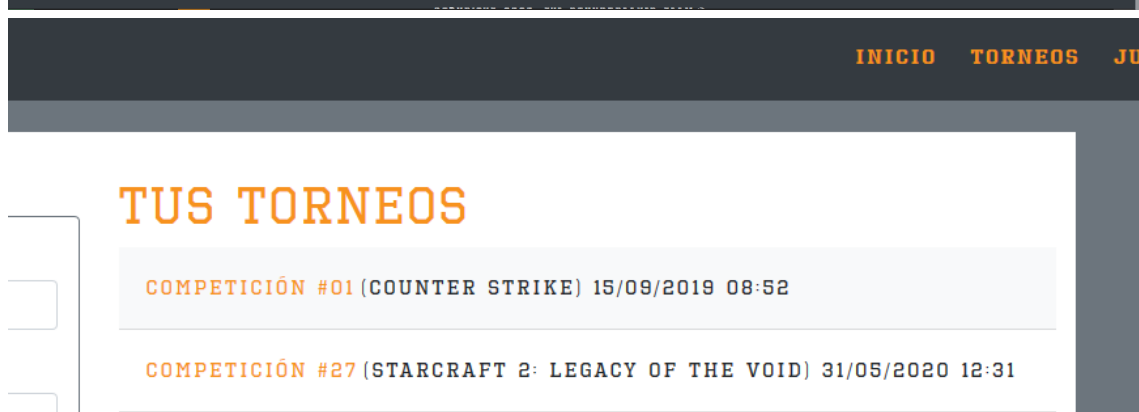
Aquí tenemos todos los jugadores de nuestro equipo, información sobre el torneo en el que está compitiendo y somos el capitán del equipo podemos editar el nombre. Además, podemos ver el ranking actual de nuestro equipo en el torneo en el que está participando. Continuaremos creando nuestro propio torneo:



Como podemos ver, ahora hay más opciones en la lista de torneos. Podemos unirnos a competiciones desde aquí o podemos crear una competición dándole al botón correspondiente.

CREAR COMPETICIÓN

NOMBRE*	DESCRIPCIÓN
<input type="text"/>	<input type="text"/>
NOMBRE DE SALA	CONTRASEÑA DE SALA
<input type="text"/>	<input type="text"/>
NÚMERO DE EQUIPOS*	
ESCOGE...	
JUGADORES POR EQUIPO*	
ESCOGE...	
JUEGO*	
ESCOGE...	
FECHA*	HORA*
dd/mm/aaaa	--:--
<input type="button" value="CREAR!"/>	



Rellenando este formulario podremos tener nuestra propia competición. Desde nuestro perfil podemos acceder a la competición que acabamos de crear.

ROUND
GRABER

MÁS DE 27 TORNEOS ORGANIZADOS!

INICIO TORNEOS JUEGOS PERFIL LOGOUT

COMPETICIÓN #27

Organiza: ellis.feest

Estado: Abierta y En Curso - 31/05/2020 12:31

ASIGNAR EQUIPOS!

NOMBRE*

Competición #27

DESCRIPCIÓN

NOMBRE DE SALA

CONTRASEÑA DE SALA

NÚMERO DE EQUIPOS*

4

JUGADORES POR EQUIPO*

1

JUEGO*

NOMBRE DEL TWITCH BOT

TOKEN DEL TWITCH BOT (OBTÉN UN TOKEN)

CANAL DE TWITCH

ABRIR CONFIRMACIONES!

EDITAR!

INSCRIPCIONES

ADRIANNA FRIESEN

☒

MODESTA06

☒

OLEUSCHKE

☒

ARLIE OCONNER

☐

BODE JAKOB

☐

NADER TIA

☐

BEER HOLDEN

☐

DAMORE WATSON

☐

ELIMINAR TORNEO

UNIRSE

Esta es la página de edición de torneo, aquí podremos editar sus datos, randomizar los equipos, banear/confirmar/desconfirmar jugadores y añadir un **Twitch Bot**. Este bot podrá leer el chat de nuestro canal de Twitch para que nuestros espectadores puedan confirmar su presencia en un torneo mediante un simple mensaje el chat. Debemos rellenar el formulario así:

NOMBRE DEL TWITCH BOT

TwitchBot

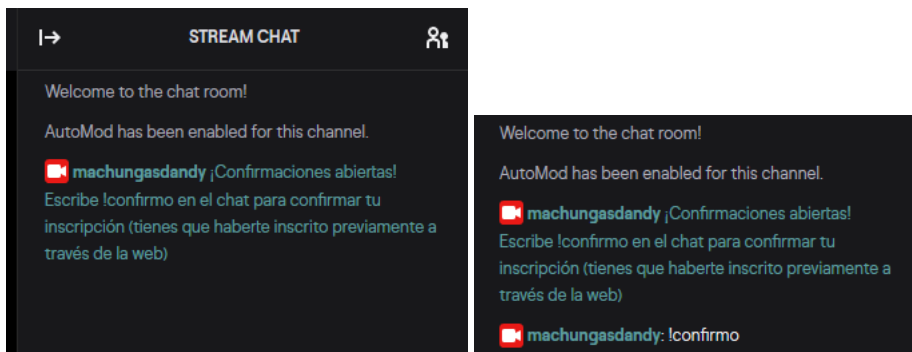
TOKEN DEL TWITCH BOT (OBTÉN UN TOKEN)

CANAL DE TWITCH

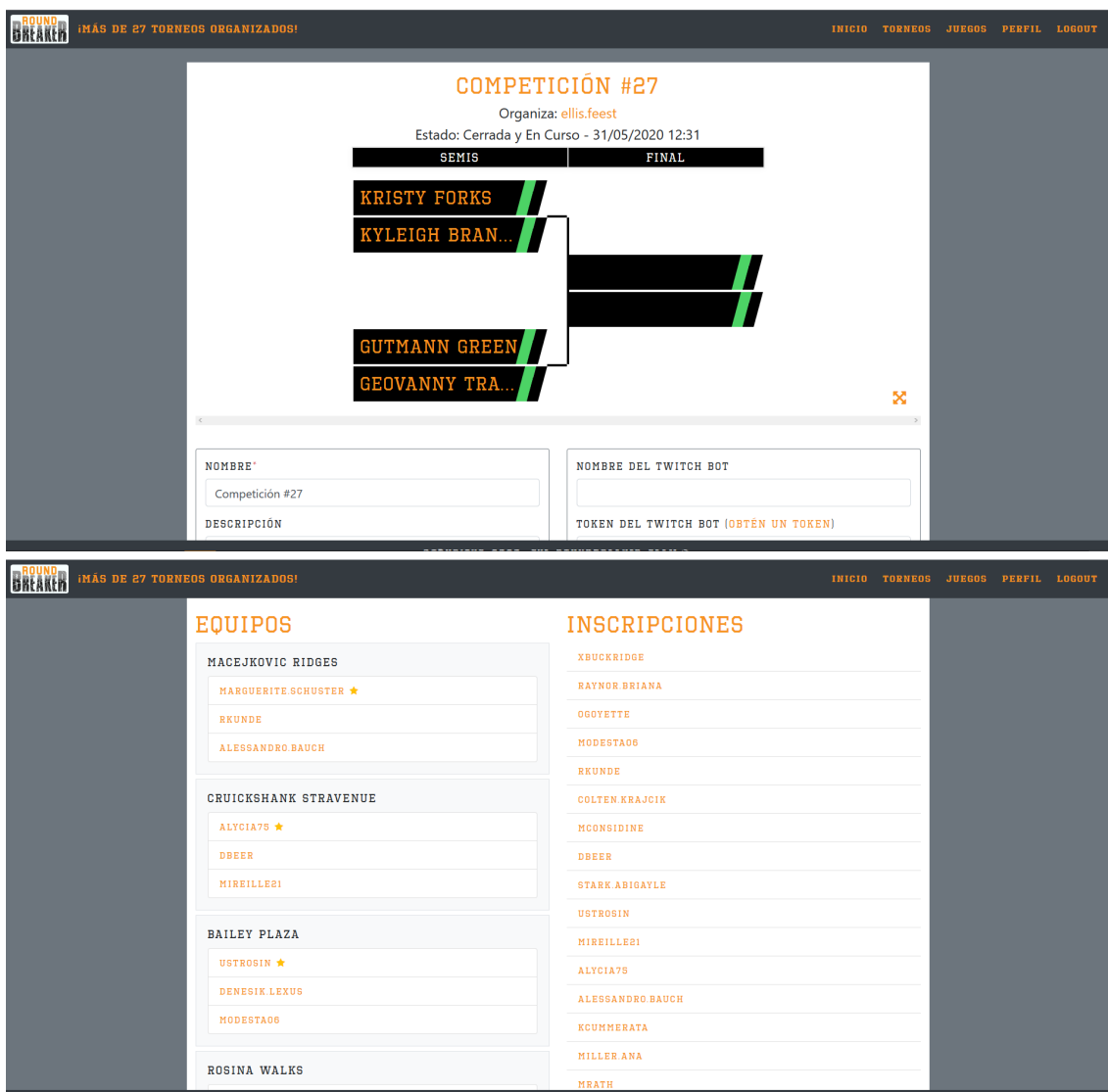
machungasdandy

ABRIR CONFIRMACIONES!

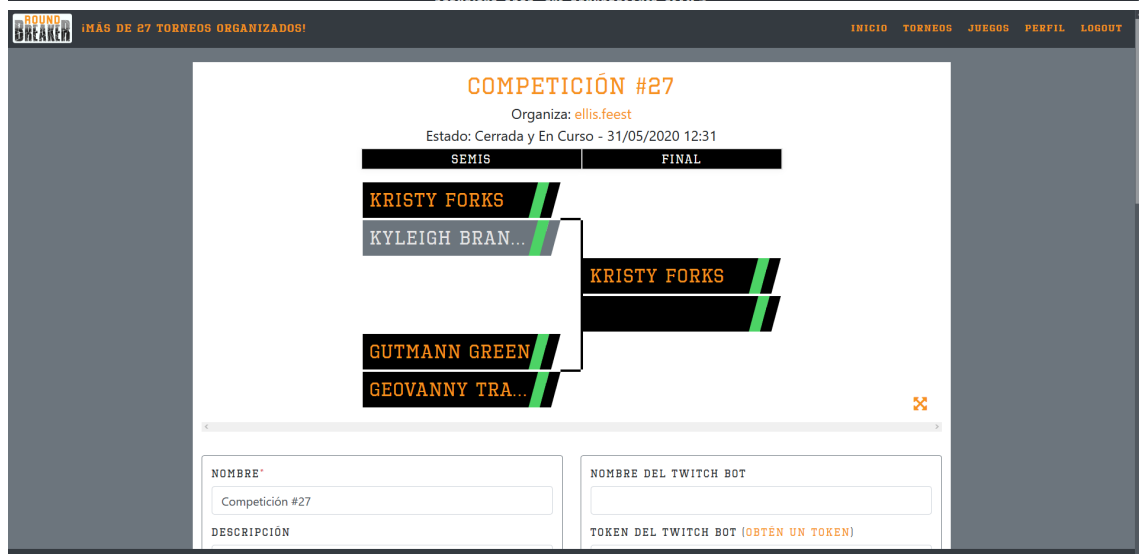
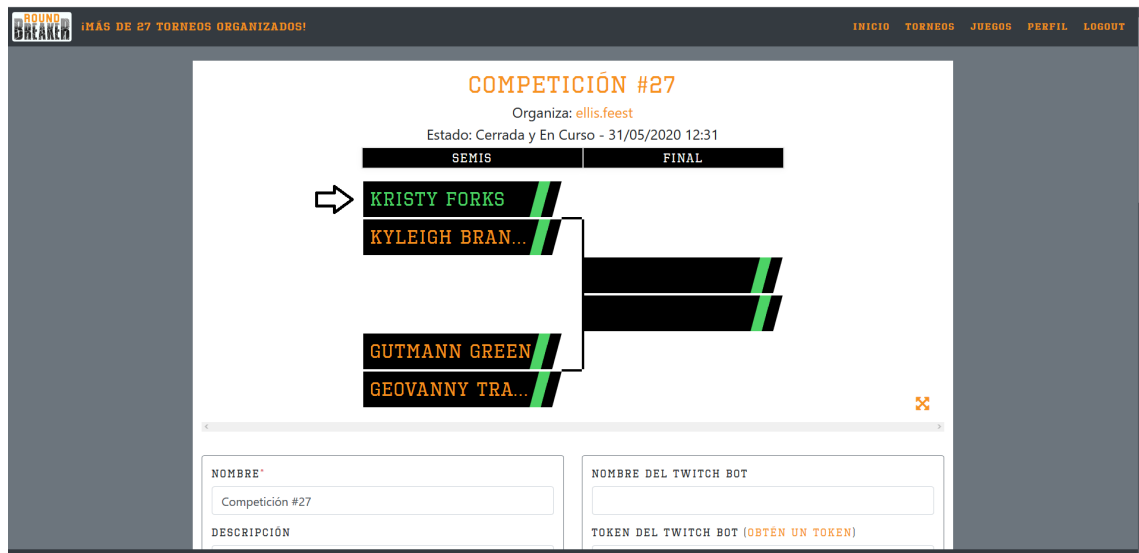
Tras abrir confirmaciones en nuestro canal de twitch aparecerá nuestro bot con instrucciones:



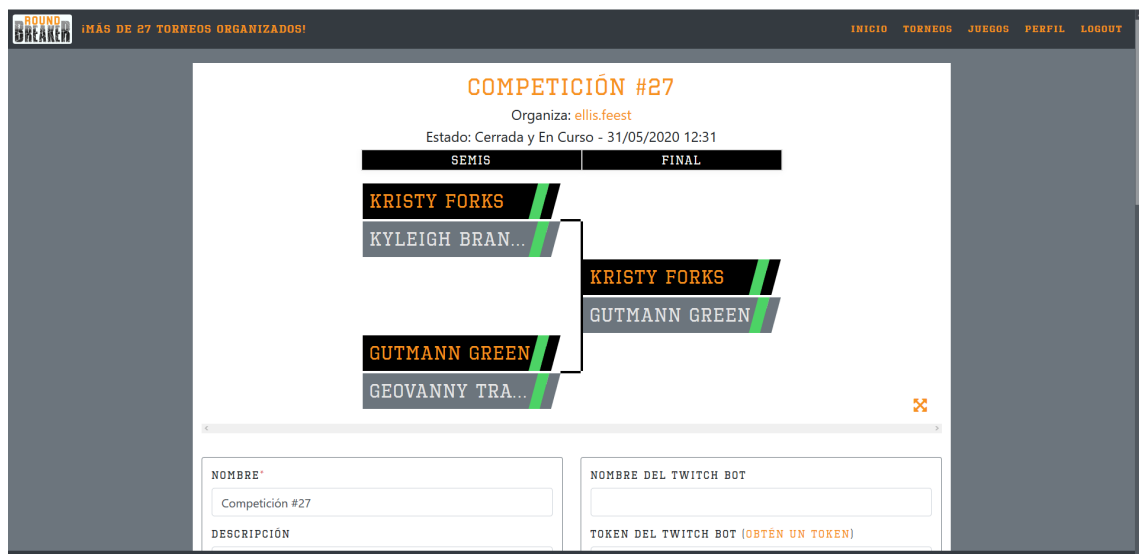
Una vez estemos satisfechos con las opciones de nuestro torneo y las confirmaciones obtenidas, podemos asignar equipos:



De esta forma, ya han sido creados los 4 equipos de nuestro torneo, con jugadores aleatorios de entre los inscritos confirmados. Ahora se pueden ver los equipos con sus capitanes en el lado izquierdo de la pantalla. Este árbol de competición está pensado para que el streamer pueda retransmitirlo, poniéndolo a pantalla completa haciendo clic en el icono de ampliar de abajo a la derecha. También podemos hacer click en el nombre del equipo ganador para que avance a la siguiente ronda:



Una vez hayamos elegido todos nuestros ganadores, la competición quedará finalizada.



Esto cubre lo básico para usar nuestra plataforma como jugador o streamer, por lo tanto pasaremos a mostrar la funcionalidades de **administrador**.

Administrador

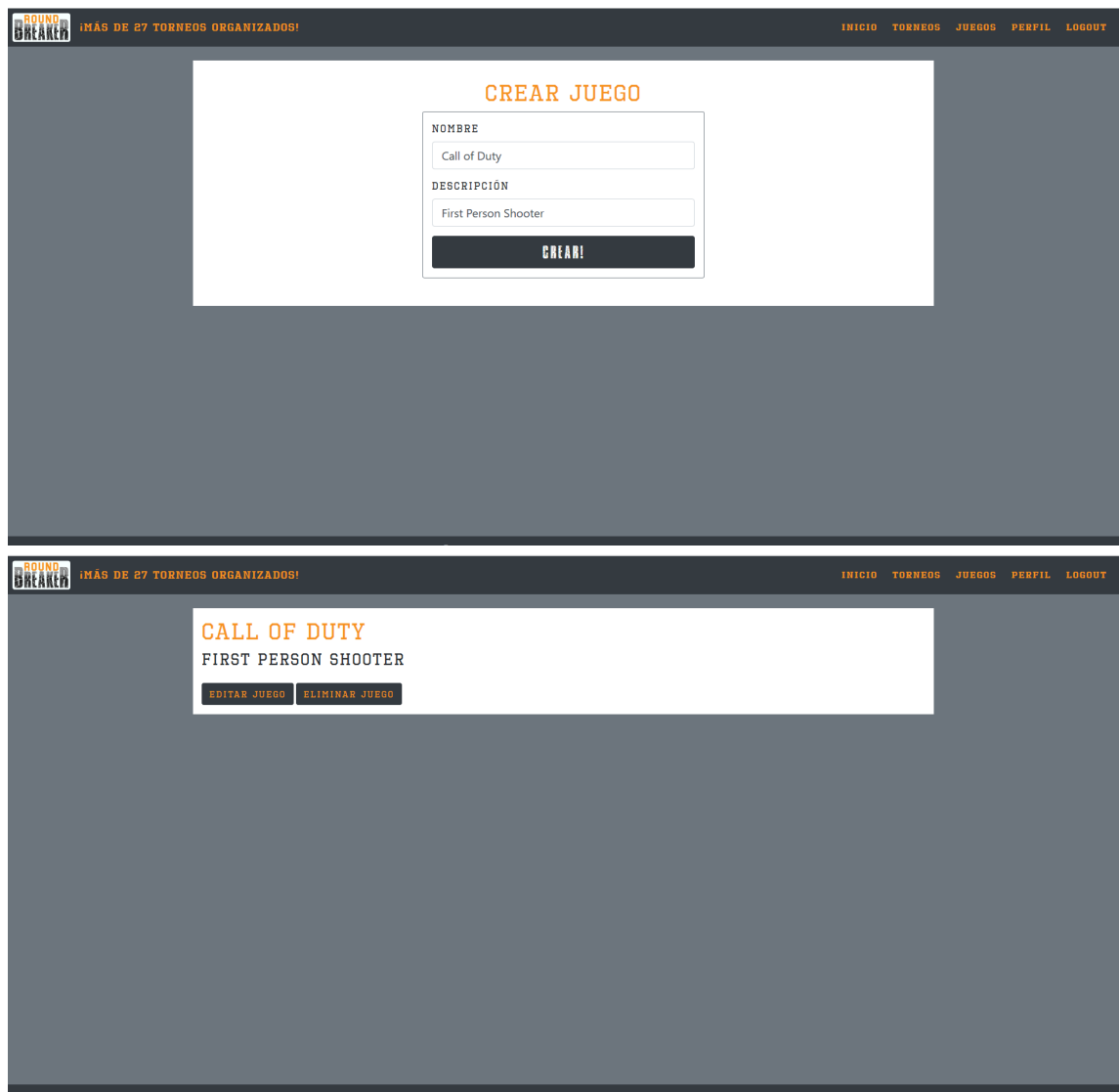
The image displays two screenshots of the Round Breaker administrator interface. The top screenshot shows the 'JUEGOS' (Games) management page. It features a header with the Round Breaker logo and navigation links. The main content area has a 'JUEGOS' title and a 'CREAR JUEGO' button. Below this is a table listing games and their associated tournaments, with 'EDITAR' and 'BORRAR' buttons for each row. The bottom screenshot shows the 'ROCKET LEAGUE' tournament details page. It features a header with the Round Breaker logo and navigation links. The main content area has a 'ROCKET LEAGUE' title and a 'SUPERSONIC ACROBATIC ROCKET-POWERED BATTLE-CARS' subtitle. Below this is a table listing competitions and their dates, with a 'UNIRSE' button for each row. At the bottom, there are 'EDITAR JUEGO' and 'ELIMINAR JUEGO' buttons.

JUEGOS	COMPETICIÓN	FECHA
COUNTER STRIKE PUM PUM	COMPETICIÓN #21	07/04/2020 03:42
DOTA 2	COMPETICIÓN #20	23/03/2020 07:51
LEAGUE OF LEGENDS BEST COMMUNITY EVER	COMPETICIÓN #19	14/03/2020 08:57
QUAKE CHAMPIONS	COMPETICIÓN #18	06/03/2020 06:49
RAINBOW SIX	COMPETICIÓN #17	25/02/2020 07:59
ROCKET LEAGUE SUPERSONIC ACROBATIC ROCKET-POWERED BATTLE-CARS	COMPETICIÓN #16	16/02/2020 04:08
STARCRRAFT 2: LEGACY OF THE VOID		

ROCKET LEAGUE	COMPETICIÓN	FECHA
SUPERSONIC ACROBATIC ROCKET-POWERED BATTLE-CARS	COMPETICIÓN #21	07/04/2020 03:42
	COMPETICIÓN #20	23/03/2020 07:51
	COMPETICIÓN #19	14/03/2020 08:57
	COMPETICIÓN #18	06/03/2020 06:49
	COMPETICIÓN #17	25/02/2020 07:59
	COMPETICIÓN #16	16/02/2020 04:08

Desde la lista de juegos, podemos ver todos los juegos disponibles en la plataforma. Si hacemos click en uno, podemos ver más información y una lista de todas las competencias que utilizan ese juego.

Un admin puede crear torneos y editar todos los torneos de la plataforma, no solo los que haya creado él/ella. Además puede crear, editar y eliminar **Juegos**. Esto es una característica específica del administrador. Un usuario normal puede ver la lista de juegos y acceder a ellos, pero no modificarlos de ninguna forma. Si hacemos click en botón "Crear Juego", se nos presentará este formulario.



Tras finalizar el formulario, todos los usuarios podrán utilizar nuestro nuevo juego para sus torneos.

ROUND
CHALKY

IMÁS DE 27 TORNEOS ORGANIZADOS!

INICIO TORNEOS JUEGOS PERFIL LOGOUT

BODE.JAKOB

EDITAR JUGADOR

TORNEOS

No ha organizado ningún torneo.

REGISTROS

COMPETICIÓN #26 (STARCRRAFT 2: LEGACY OF THE VOID)	22/05/2020 12:21
COMPETICIÓN #19 (ROCKET LEAGUE)	14/03/2020 09:57
COMPETICIÓN #12 (RAINBOW SIX)	04/01/2020 05:48
COMPETICIÓN #10 (QUAKE CHAMPIONS)	19/12/2019 03:11
COMPETICIÓN #09 (QUAKE CHAMPIONS)	03/12/2019 15:18
COMPETICIÓN #06 (LEAGUE OF LEGENDS)	05/11/2019 00:36
COMPETICIÓN #04 (LEAGUE OF LEGENDS)	13/10/2019 17:36
COMPETICIÓN #01 (COUNTER STRIKE)	15/09/2019 08:52

EQUIPOS

ROUND
CHALKY

IMÁS DE 28 TORNEOS ORGANIZADOS!

INICIO TORNEOS JUEGOS PERFIL LOGOUT

PERFIL

BODE.JAKOB

USERNAME

bodejakob

CONTRASEÑA

Contraseña

REPETIR CONTRASEÑA

Repetir contraseña

EMAIL

vaughn.mertz@gmail.com

TWITCHNAME

bodejakob

EDITAR DETALLES

Como admin, también podemos editar información de un jugador desde su perfil.

Recuperación de contraseña

Para concluir este manual de usuario, demostraremos como recuperar nuestra contraseña si la hemos perdido:

The image shows two screenshots of the Round Breaker website's password recovery process. The top screenshot shows the 'LOG IN' button and a link 'He olvidado mi contraseña' (I forgot my password). The bottom screenshot shows the 'RECUPERA TU CUENTA' (RECOVER YOUR ACCOUNT) form, which prompts the user to enter their email address to receive a recovery link. The form includes a text input field and a button labeled 'ENVIAR CORREO DE RECUPERACIÓN' (SEND RECOVERY EMAIL).

Si pinchamos en este enlace, podremos introducir nuestro email para recuperar nuestra cuenta. Tras unos momentos llegará un correo a nuestro email:

The image shows an email received from Round-Breaker <roundbreaker@bgon.es> for 'para mí' (for me). The email body contains the 'Round Breaker' logo, the heading 'Obtén una nueva contraseña' (Get a new password), and instructions to click on a provided link to create a new password. It also states that the link will expire within 2 hours. At the bottom, there are two buttons: 'Responder' (Reply) and 'Reenviar' (Forward).

Haciendo click en el enlace proporcionado, podremos recuperar nuestra contraseña introduciendo una nueva:

The image shows the 'NUEVA CONTRASEÑA' (NEW PASSWORD) form on the Round Breaker website. The form prompts the user to enter a new password and confirm it by repeating it. It includes two text input fields labeled 'Nueva contraseña' (New password) and 'Repite contraseña' (Repeat password), and a button labeled 'CAMBIAR CONTRASEÑA' (CHANGE PASSWORD).

Futuras mejoras

Durante el desarrollo de Round Breaker han ido surgiendo nuevas ideas, otras ideas las hemos descartado y en general el proyecto han ido cambiando algunas características según avanzábamos.

Como ya hemos comentado en [descripción de entidades](#), la relación entre Round y Team es ManyToMany, por lo que la limitación de dos equipos por ronda es prescindible y en un futuro nos gustaría dar soporte a competiciones que hay rondas con 3 o más equipos, ya que en varios juegos esto es habitual y nos daría capacidad para hacer competiciones con más gente en menos tiempo, ya que de cada ronda solo sale un ganador igualmente.

Algo parecido pasa con la relación entre Team y Player, que aunque hemos limitado a 5 usuarios por equipo, podríamos hacerlos mucho más multitudinarios, aunque habría que estudiarlo bien porque al estar pensado para en directo podría llegar a ser incontrolable manejar a tanta gente.

También habíamos pensado en desarrollar alguna característica más de cara a hacer más ágil la competición durante su desarrollo, como tener integración con Discord para poder crear al vuelo salas de audio privadas para que contacten a un solo click los integrantes de un equipo, pero si esto conlleva tener que pedirle al usuario un dato (el usuario de Discord) creemos que el registro puede ser más engorroso y perder usuarios potenciales.

Y por último, nos gustaría intentar lanzar esta plataforma al público, ponernos en contacto con algún streamer de sports que le interese usarla y después de ver su funcionamiento en vivo y realizar los cambios que consideremos dejarlo abierto para su libre uso. Pero todo esto hay que estudiarlo bien antes de llevarlo a cabo.

Conclusiones

Este proyecto nos ha empujado más que cualquier otro, pero también ha sido en el que más hemos aprendido. Hemos tenido que trabajar y solucionar problemas imprevistos en cada paso del camino pero siempre hemos sacado una lección.

Durante el desarrollo hemos replanteado conceptos clave, reescrito funcionalidades y solucionado errores que eran incomprensibles en un principio. Desde control de versiones hasta compartimentación de código, todo ha sido experiencia inestimable como programadores. Nos ha dado una pista de lo que es trabajar en una aplicación extendida de trabajo en grupo y bajo presión. Aprender a poner una aplicación en producción ha sido una prueba considerativa pero ha valido la pena.

Nos ha dado mas ganas que nunca de dedicarnos a esto y hacer realidad nuestras ideas y proyectos. Sabemos que esto es solo el principio y hay mucho más trabajo por hacer, pero ver el fruto del trabajo siempre será especial.