Práctica 5: Despliegue de un sistema de información Web con persistencia de datos utilizando contenedores Docker

Peña Costa, Mateo (874162)
Cámara Domene, Raquel (875001)
Ibañez Sánchez, Marcos (869021)

viernes, 29 de noviembre de 2024

Práctica 5: Despliegue de un sistema de información Web con persistencia de datos utilizando contenedores Docker.....

Funcionalidades del sistema

El sistema permite loguearse y registrarse al usuario además de el cambio de contraseña o datos de perfil.

Previamente habiéndose registrado, el sistema permite al usuario guardar formaciones de equipos de campeones. Esto se hace para que si en una partida le ha ido muy bien con esos campeones, pueda guardar la combinación de estos para poder verla en futuras partidas y recrearla.

Otra funcionalidad es que habiendo seleccionado primero el nivel de la ronda en la que se encuentra, y más adelante los campeones que posee y emblemas que tiene, le aparecerán los personajes recomendados que mejor le irían para ganar la partida, y que tendría que buscar si los tiene ya o comprarlos en otro caso.

Para buscar los campeones, tiene la opción de utilizar la barra de búsqueda del menú de personajes, presionando el botón de 'buscar' o dándole a la tecla 'enter'. Además podrá ordenar el menú por el atributo de 'coste' del personaje, o por orden alfabético.
Asimismo, el sistema permite realizar una búsqueda de personajes en función de sus características principales (Nombre y coste).

En el menú de 'recomendaciones' que tenga, se le enseñarán los personajes a escoger, ordenados por probabilidad de aparición según el nivel de la ronda en la que se encuentre.

Hay una pestaña de "Como usar" en la que el usuario puede leer una guía para usar la aplicación.

A continuación se resume el funcionamiento del algoritmo de "Recomendación" de la aplicación que es nuestro código principal y más complejo:

- En el juego hay un listado de "fichas" o "campeones" que juegan por rondas en un tablero. El juego a menudo se compara con un ajedrez automático.

- Estas fichas varían en coste de 1-5, además cuanto mayor sea el coste menor será su probabilidad de aparición dependiendo también del nivel del jugador en cada ronda, pudiendo a llegar a tener una probabilidad de aparición de 0%.

- Las fichas tienen unas "sinergias" que si se acumulan varias de las mismas dan poderosas bonificaciones.

Con todo esto en cuenta nuestro algoritmo se encarga de recopilar información como:
- El nivel actual del jugador (y por lo tanto la probabilidad de aparición)
- El tablero actual del jugador (y por lo tanto las sinergias y combinaciones)

El algoritmo recomendará al jugador los campeones ideales, es decir, los que más sinergias aporten al equipo y sean más óptimos y los ordenará por probabilidad de aparición (si la probabilidad es 0 no aparecerán).

Perfiles de usuario que interactúan con el sistema

Hay dos perfiles de usuario que serán, Usuario Estándar y Administrador.

El usuario estándar puede acceder a las funcionalidades de crear composiciones de campeones, ver el manual de uso de la aplicación web, y seleccionar los personajes y emblemas que tenga en la ronda actual para que el sistema le recomiende que personaje nuevo escoger.

El administrador tiene acceso a eliminar usuarios y composiciones de estos. Esto estaría directamente relacionado con el mantenimiento y seguridad de la web, al poder eliminar usuarios con nombres ofensivos o que atentaran contra la aplicación web (por ejemplo, subiendo excesivas composiciones de campeones para sobrecargar la web).

Entidades de información que deben gestionarse de forma persistente

La aplicación web tiene una base de datos estática para almacenar campeones, emblemas y sinergias desde la que los usuarios podrán buscar y sacar información permitiéndoles interactuar con la página.

Además de eso, también contamos con una base de datos para almacenar los registros de usuarios y otra funcionalidad que supone tener una base de datos donde los usuarios puedan almacenar, borrar y buscar formaciones de equipos creados y guardados por ellos.

La información que usaremos para rellenar la base de datos estática serán nuestros propios datos obtenidos manualmente.

Recursos utilizados

Para completar las funcionalidades de nuestra aplicación web hemos añadido alguna funcionalidad que nos ha gustado de dos páginas que tratan también sobre el mismo videojuego:

- Mobalytics: https://mobalytics.gg/tft?int_source=homepage&int_medium=mainbutton
- TFTactics: https://tftactics.gg/

En este caso para hacer la funcionalidad específica de poder ver las composiciones de campeones de los demás usuarios nos hemos basado en un apartado que tienen ellos para poder ver formaciones de campeones.

La segunda página (TFTactics) la hemos utilizado para ver cuantos personajes serían, que son 60 personajes, y para más adelante hacer nuestra base de datos.

Tecnología usadas

Para la parte del frontend hemos usado herramientas estándar como HTML, CSS y JavaScript que aportan el cuerpo, el estilo y la funcionalidad a la página web.

Las bases de datos se almacenarán con SQLite y accederemos a ellas con Python y sus bibliotecas. Hemos elegido SQLite debido a su ligereza y portabilidad, la base de datos es ideal para una aplicación o prototipo pequeño/mediano como el nuestro. Además, SQLite cuenta con compatibilidad multiplataforma entre sistemas operativos.

Para guardar nuestro trabajo y poder ir viendo las versiones realizadas, utilizaremos un repositorio de GitHub compartido entre los 3 miembros del grupo.

Enfoque del sistema

Los ficheros de la aplicación se encuentran en la carpeta “app/src”. Allí tenemos el fichero app.py y una carpeta dedicada a la base de datos “app/src/bd”, una carpeta dedicada a los html “app/src/templates”, una carpeta para las clases VO y DAO “app/src/classes” y una carpeta para los elementos estáticos “app/src/static”. En está última carpeta, tenemos los css (“app/src/static/css”), las imágenes (“app/src/static/images”) y algunos javascript (“app/src/static/js”).

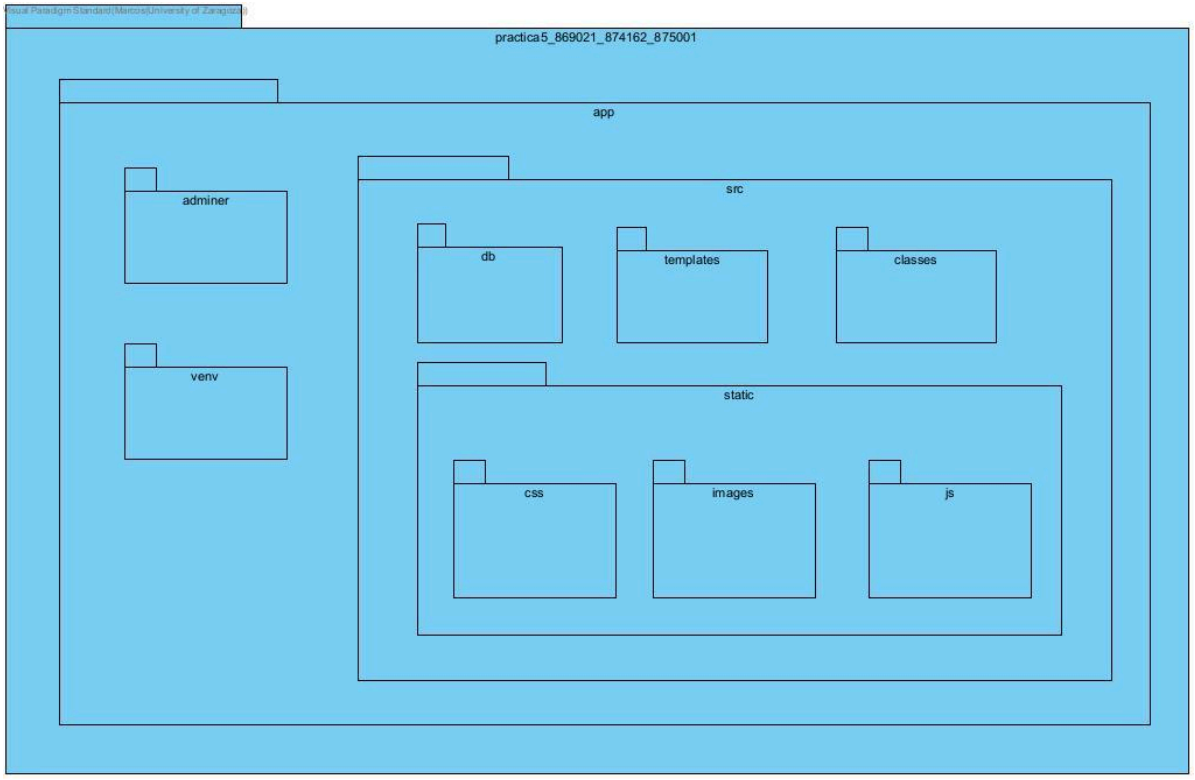

Figura 1: Diagrama de paquetes del proyecto

Los html poseen javascript insertados y también ejecutan los ficheros .js de “app/src/static”. A través de estos scripts interactuamos con app.py (“app/src/app.py”) . Por otro lado, app.py nos permite la interacción con los DAO y VO, que sirven de puente para manipular la base de datos (“app/src/bd/database.db”). A continuación observamos un diagrama de clases con los métodos de las clases DAO y los atributos de sus correspondientes VO.

Mapa de navegación

Figura 2: Mapa de navegación

Descripción de las pantallas

P1. Pantalla de inicio.
P2. Pantalla con la opción en el usuario de configurar su perfil, o desloguearse.
P3. Pantalla para cambiar los datos de la cuenta.
    P3.1. Pantalla con las opciones de cambio de avatar.
P4. Pantalla de inicio de sesión.
P5. Pantalla de registro del usuario.
    P5.1. Pantalla de registro del usuario con posibles errores que podría hacer.
P6. Pantalla con todas las herramientas necesarias para que el sistema responda al usuario con las recomendaciones correspondientes.
    P6.1. Pantalla con un formulario a rellenar para guardar o publicar la composición de campeones.
        P6.1.1. Pantalla de confirmación de guardar una composición.
P7. Pantalla para ver 'mis composiciones' guardadas.
    P7.1. Pantalla que permite ver la descripción de una composición.
    P7.2. Pantalla de confirmación de borrar composición, con la opción de cancelar la acción.
    P7.3. Pantalla para editar la descripción o nivel de dificultad de la composición.
    P7.4. Pantalla de confirmación de publicación y despublicación de la composición.
P8. Pantalla con las composiciones que los usuarios pueden ver de los que las han publicado.
P9. Pantalla con un video sobre cómo utilizar la página web.
P10. Pantalla con el perfil de administrador para poder eliminar usuarios.
P11. Pantalla con el perfil de administrador para poder eliminar composiciones de campeones.

Leyenda / Comentarios:

La pantalla P1 es una pantalla a la que podemos acceder con la barra de scroll, y que tiene el menú principal de nuestra página web.

Los símbolos y botones que aparecen en todas las pantallas pero no tienen flecha (o solo en alguna para poder ver un ejemplo de ejecución) ha sido una elección nuestra para no sobrecargar el mapa, y por ello se explican a continuación:

- En todas las pantallas tenemos el logo, el cual nos lleva siempre a la pantalla inicial (P1).
- Todas las pantallas menos la P4 y P5, que son de registro e inicio de sesión, tienen arriba a la derecha un botón con el simbolo de interrogacion, que si se clica en el llevara a la P9.
- Todas las pantallas tienen además si el usuario se ha registrado o no, marcando arriba a la derecha si se ha realizado la acción con un mensaje de 'Hola, …' y en caso contrario el botón para loguearse.

Todas las acciones disponibles del menú de inicio se pueden acceder sin registrarse a excepción de la opción 'My Team Comps'. También se pedirá haber iniciado sesión en la pantalla referente a guardar composiciones (P6.1), o en el caso de la opción del menú 'Community comps' se pedirá loguearse cuando quiera guardar una composición, votar o reportar.

Además, no se muestra en el mapa de navegación, pero en todos los casos que el contenido de un contenedor ya sea de campeones, emblemas, sinergias o recomendaciones que aparecen en la opción de 'Start Team', y las composiciones guardadas en 'My community comps', tienen un scroll invisible para en caso de no caber todos los elementos en la pantalla, poder observar el resto de estos. En el caso de las composiciones guardadas, el scroll es visible.

Modelo entidad-relación

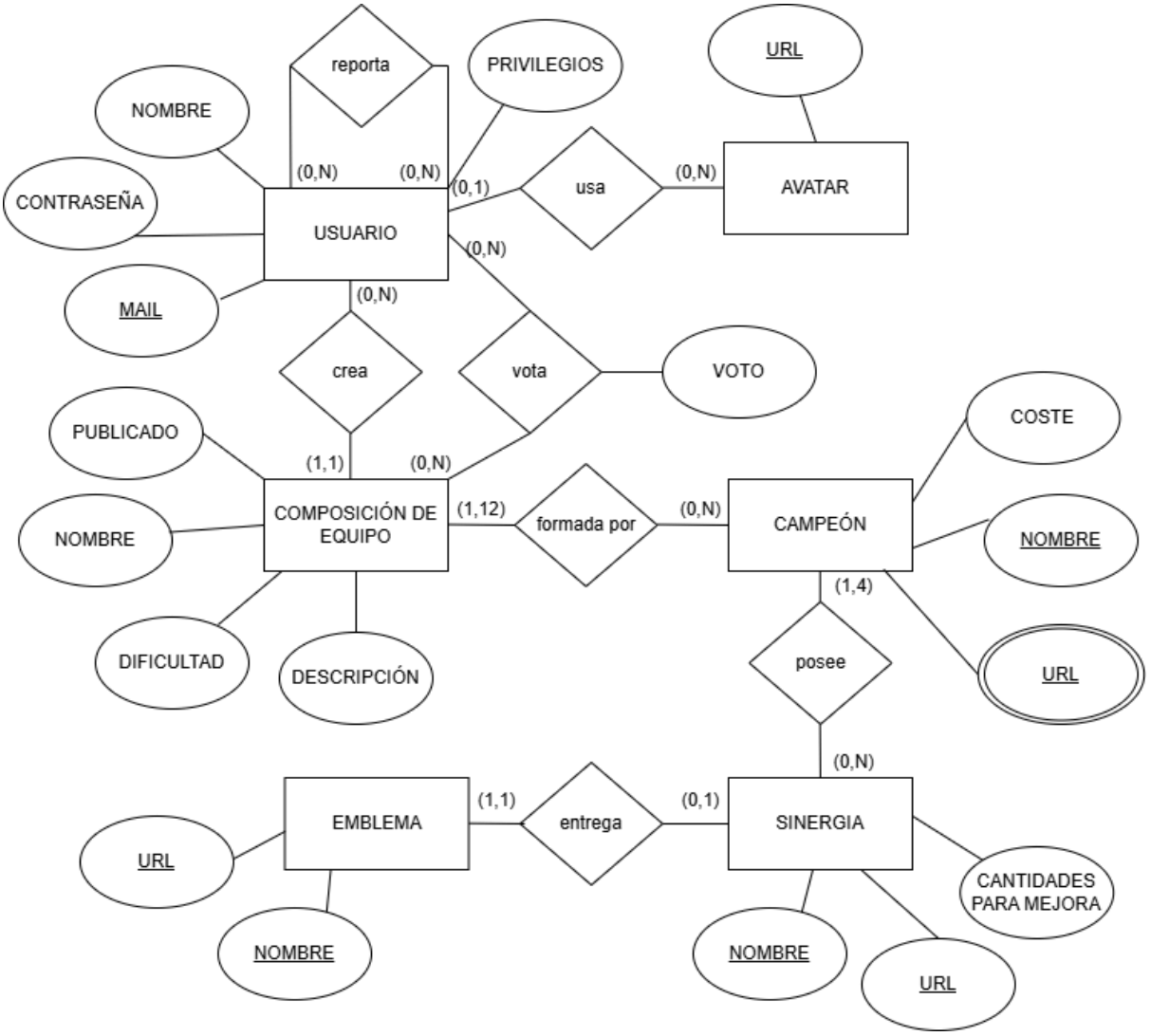

Figura 3: Esquema Entidad-Relación

Restricciones

- No pueden existir dos composiciones de equipo que tengan el mismo nombre y que hayan sido creadas por el mismo usuario.
- Un usuario no se puede reportar a sí mismo.

Modelo relacional

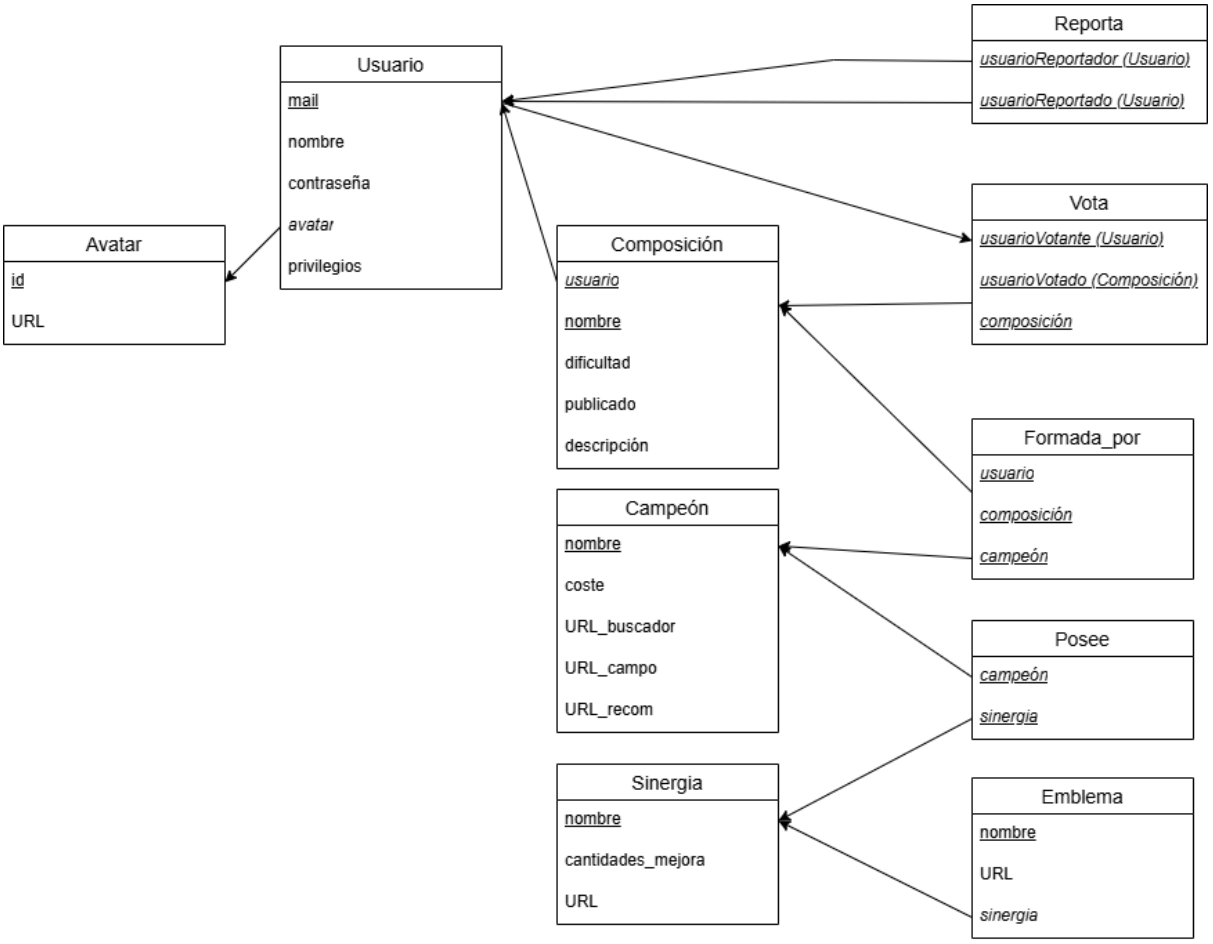

Figura 4: Esquema Relacional

Para comprobar que nuestro modelo está normalizado hemos comprobado que cumplía la 2FN ya que no hay dependencias con parte de la clave, teniendo en cuenta que la única clave compuesta que no es todos los atributos es la clave compuesta de 'Composición', y está normalizada.
En cuanto a la 3FN, los atributos de composición no tienen ninguna relación entre ellos, los de la tabla 'Usuario' tampoco, y en 'Campeón' o 'Emblema', el atributo URL solo fue clave candidata, por lo que no hay dependencias.

Creación de tablas

Para la creación de tablas y su propia población hemos utilizado el lenguaje de sqlite.

Para ello hemos creado un archivo app.py que ejecuta la funcion 'init_db', lee del archivo 'schema.sql' que contiene el esquema de las tablas a crear con sus determinados 'CREATE TABLE' y va creando la base de datos 'database.db'.

Estamos trabajando con la herramienta de 'Docker Desktop' por lo que cuando ejecutamos el programa 'app.py' ya se crearon las tablas en el contenedor, y por lo que no hará falta que se vuelva a ejecutar el archivo cada vez que se quiera ejecutar el contenedor.

Diseño de las clases

Para la implementación de la capa de acceso a datos de la aplicación Web hemos decidido utilizar las clases DAO y VO. Hemos optado por trabajar con python para crear la base de datos, y sus clases. Para organizar mejor el código, por cada tabla creada hemos creado un archivo que incluye su clase VO, y la clase DAO que realiza la comunicación entre esta y la base de datos mediante consultas.
El nombre de los archivos de cada clase tiene el formato de 'nombre_de_la_tabla.py'

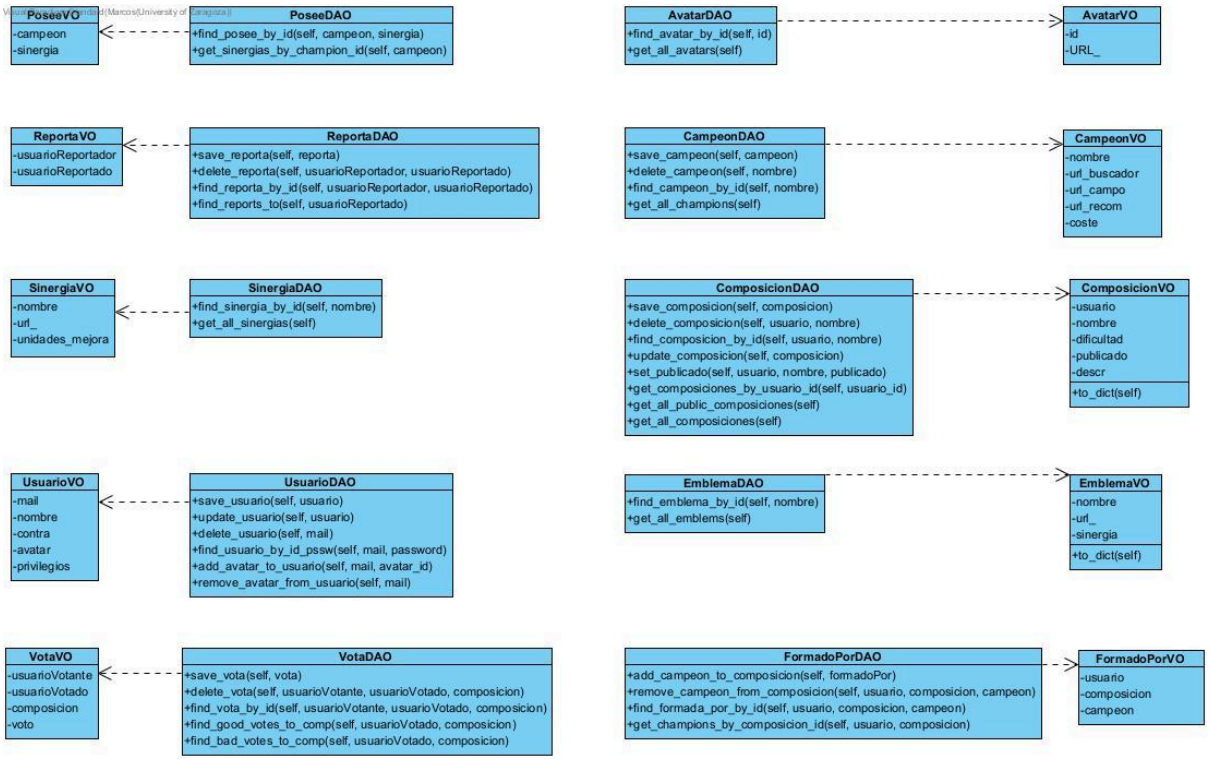

Figura 5: Diagrama de clases del VO y el DAO

Proceso de instalación

La instalación de nuestra aplicación es sencilla y tan solo requiere unos pocos pasos.
Pero sí será necesaria la instalación del siguiente software:

-Docker Desktop

1º.- Instalar Docker Desktop:

Puede ser descargada desde el sitio oficial: https://www.docker.com/products/docker-desktop/

2º.- Iniciar Docker Engine:

Es importante tener ejecutándose la aplicación Docker Desktop y su Docker Engine para poder lanzar la aplicación en todo momento.

3º.- Descargar el .zip de la aplicación:

Descargar y descomprimir el zip incluido en la entrega.

4º.- Acceder al directorio del proyecto:

Navegar hasta el directorio en el que se encuentra el proyecto,.
Deberían verse los siguientes archivos y directorios:

app
docker-compose.yml

5º.- Ejecutar la aplicación:

Desde una terminal en el directorio mencionado anteriormente, ejecutar el comando *"docker-compose up"* para construir la aplicación. Importante tener el Docker Engine funcionando de fondo.

6º.- Visualizar la web:

Abrir un navegador web y escribir en la URL: *"localhost:7000"*
Adicionalmente se puede abrir en otra pestaña: *"localhost:8080"* para visualizar la base de datos en ejecución mediante la herramienta Adminer.

Datos requeridos para el uso del sistema

Cuenta de administrador:

- Mail: admin@gmail.com Contraseña: @serendipia7#

Cuentas de usuario (También es posible registrarse con una nueva cuenta):

- Mail: user1@example.com Contraseña: password1
- Mail: user2@example.com Contraseña: password2
- Mail: user3@example.com Contraseña: password3

Metodología de trabajo, planificación y distribución de tareas

A lo largo de la práctica hemos ido siguiendo la hoja de ruta facilitada por los profesores. En algunas ocasiones hemos dividido las tareas entre los distintos integrantes del grupo y otras veces hemos dejado que cada uno avance en lo que más desee, siempre y cuando avise a los demás de lo que va a intentar y del trabajo realizado.
El trabajo común se ha organizado a través de Github realizando commits con un título explicativo de los cambios realizados. Además, se ha utilizado Discord para la comunicación del equipo. Por otra parte, hemos trabajado a través de un Docker desde el comienzo de la implementación, lo cual ha facilitado sustancialmente el trabajo. En ocasiones también hemos aprovechado la extensión de Visual Studio Live Share para trabajar simultáneamente sobre el mismo código.
Cabe destacar que, aunque hemos tratado de conocer todos los distintos elementos que conforman el proyecto, cada uno de nosotros hemos invertido una mayor cantidad de tiempo en determinados aspectos de la aplicación o en pantallas específicas de ellas. A continuación indicaremos algunos ejemplos de cada integrante:

- Mateo se ha encargado de la interacción con la base de datos creando la capa de persistencia, de la utilización de la sesión en las distintas pantallas, de la búsqueda y solución de problemas y de pantallas como Start Team, Profile Settings y el Index.
- Raquel se ha encargado de la adaptabilidad y accesibilidad, de los css, de la realización de parte de la documentación como la producción del vídeo y de pestañas como login, register y My Team Comps.
- Marcos se ha encargado de la creación de la base de datos, la seguridad y el mantenimiento, además de pantallas como Community Comps o Admin User List y Admin Comps List.

A continuación se muestra una tabla con la distribución del tiempo de cada integrante.

Horas trabajadas

Práctica 1

	RAQUEL	MATEO	MARCOS	TOTAL
INFORME INICIAL	1h 45min	1h 45min	1h 20min	4h 50min
TOTAL	1h 45min	1h 45min	1h 20min	4h 50min

Práctica 2

	RAQUEL	MATEO	MARCOS	TOTAL
Resumen	30min	15min	15min	1h
Recursos utilizados	2h	2h	2h	6h
Tecnologías usadas	2h 30min	2h 30min	2h 30min	7h 30min
Mapa de navegación y navegabilidad	3h	3h	2h	8h
Memoria	4h	3h	3h	10h
TOTAL	12h	10h 45min	9h 45min	32h 30min

Práctica 3

	RAQUEL	MATEO	MARCOS	TOTAL
Resumen	-	15min	-	15min
Modelo entidad-relación	15min	15min	2h	2h 30min
Modelo relacional	1h	15min	15min	1h 30min
Creación de tablas	15min	2h	15min	2h 30min
Diseño de clases	2h	2h	2h	6h
Memoria	2h	1h	1h	4h
TOTAL	5h30min	5h45min	5h30min	16h 45min

Práctica 4 y 5

	RAQUEL	MATEO	MARCOS	TOTAL
Capa de persistencia de datos	-	15min	-	
Capa modelo	10h	20h	15h	45h
Capa vista	30h	30h	20h	80h
Adaptabilidad y usabilidad	20h	10h	15h	45h
Seguridad y mantenimiento	10h	15h	20h	45h
Testing	15h	20h	15h	50h
Memoria	3h	3h	5h	11h
Docker	10h	10h	5h	25h
TOTAL	98h	108h	95h	301h

Apartados extras

Seguridad

Cuenta de administrador

El principal elemento usado para reforzar la seguridad y facilitar el mantenimiento es la existencia de cuentas con privilegios. La base de datos ha sido ligeramente modificada para agregar a la entidad Usuario un atributo Privilegios que indique si posee o no privilegios.

Las cuentas con privilegios permitirán el acceso a dos pantallas, una para ver todos los usuarios del sistema y el número de veces que ha sido reportado cada uno de ellos, y otra para ver las composiciones del sistema. Todas estas cuentas de usuario y composiciones tendrán a su lado un botón que permitirá el borrado de las mismas. Esta funcionalidad permitirá al administrador eliminar usuarios que hayan causado problemas o revisar y eliminar aquellas cuentas o composiciones con nombres ofensivos o peligrosos. Además, ambas pantallas poseen una barra de búsqueda para facilitar encontrar determinados nombres de usuario o composición.

Al intentar acceder a estás pantallas la aplicación pregunta a la base de datos si el usuario tiene privilegios, si la respuesta es negativa lo redirige al index.

Cabe destacar que los usuarios pueden reportar otros usuarios a través de la pantalla community comps. Esto facilita el mantenimiento.

Contraseñas hash

Como método para proteger las contraseñas hemos decidido aplicarles una función hash y guardar el resultado obtenido en la base de datos. Cuando un usuario quiera loguearse aplicaremos hash a su entrada y la compararemos con la cadena hash guardada en la base de datos.

Inyecciones SQL

Las inyecciones SQL son uno de los métodos de ataque a aplicaciones web más comunes, el tercero más común según el top 10 de OWASP.
Para evitar este tipo de ataques hemos utilizado SQL parametrizado con cursor.execute(). El SQL parametrizado soluciona muchos de los problemas de seguridad presentes en el sql concatenado. Se encarga de sanitizar las entradas automáticamente para evitar que contengan caracteres o secuencias que puedan alterar la lógica de la consulta SQL. Algunos ejemplos serían que impide realizar más de una consulta con una misma llamada a cursor.execute() o el tratamiento de carácteres como ‘ o “.

Además, usar SQL parametrizado facilita la legibilidad y mantenibilidad.

Fugas de datos y Plan de contingencia

Se ha tenido cuidado de evitar la presencia de errores que aporten información interna del programa comprobando constantemente la información que aparece en la consola y en los mensajes de error.

Además, se ha tratado de evitar todo tipo de errores Internal Server Error o Unhandled Exception que puedan dar información del funcionamiento del sistema. Por ejemplo, el acceso sin logear a pestañas que requieren login generaba un KeyError, actualmente produce una redirección a la pestaña del login antes de que se produzca dicho KeyError. Esto ocurre aunque el usuario intente acceder escribiendo la url de la pantalla que requiere login en el navegador.

También se ha evitado que los errores de las consultas a la base de datos muestren información interna del funcionamiento de esta.

Por otro lado, se evita la propagación de errores mediante la recogida de excepciones, que permitirá tratar como corresponda el error y regresar a una situación controlada.

Plan de mantenimiento

Revisiones en la base de datos

Inicialmente se realizarán revisiones semanales manuales de las tablas con pocas tuplas de la base de datos para comprobar que los valores se mantienen consistentes y no hay valores fuera de lo común. Estas revisiones se harán a través de la cuenta de administrador con las pestañas admin_user_list.html y admin_comps_list.html y con adminer para la revisión de otras tablas. Esto se hará siguiendo un procedimiento igual en todas las ocasiones. Las pantallas de administrador permiten ver y eliminar los nombres peligrosos, además de buscar nombres específicos.
A lo largo del tiempo de vida de la aplicación y especialmente si aumenta el número de usuarios, se crearán scripts y ficheros python que realicen o faciliten estás comprobaciones. Estos scripts y ficheros .py se lanzarán periódicamente cron o crontab.También se mejorarán las capacidades del administrador.

Actualización de la base de datos

La aplicación está relacionada con el videojuego Teamfight Tactics, cuyos campeones cambian cada 3 o 4 meses. Estos campeones son necesarios para la aplicación por lo que será necesario eliminar actualizarlos. En caso de que la aplicación tenga usuarios y siga funcionando a largo plazo se planea automatizar este proceso mediante el uso de la API de la empresa, Riot Games.

Revisiones del código fuente

Periódicamente, una vez al mes, se hará una búsqueda de los ataques de ciberseguridad más comunes a aplicaciones web y se planteará la posibilidad de añadir mecanismos al código para su

protección.
De la misma forma, se recopilará información mediante cuestionarios sobre los errores más comunes realizados por los usuarios y se buscará evitarlos o facilitar su solución.

Revisiones generales

Diariamente, se realizará una comprobación del correcto funcionamiento de la web. Se comprobará que está activa y que no existe ningún problema evidente en ninguna de sus funcionalidades.

Modelo de negocio

Considerando que tenemos un prototipo de aplicación web fuerte y con potencial, podríamos decidir en un futuro lanzarlo como servicio al público.

Para ello seguiríamos una serie de pasos para asegurarnos que nuestra app tuviese el mayor éxito posible y llegase al mayor número de personas posible.

Buscaríamos y compraríamos un dominio propio para la web como por ejemplo: "*www.tftmanager.com"*.

Elegiríamos un proveedor de hosting de web fiable como podrían ser Bluehost o Hostinger.
También se podrían barajar opciones gratuitas como GitHub.

Puliríamos el apartado de seguridad para que no hubiese brechas y dedicaríamos recursos al marketing de la web para anunciarla.

Buscaríamos un grupo de usuarios dispuestos a testear la app antes del lanzamiento y una vez online tendríamos un sistema de feedback para seguir mejorando la app.

Tendríamos que decidir si queremos recaudar dinero de la página mediante anuncios o no, o habilitar algún tipo de donaciones.

Accesibilidad y usabilidad

Usabilidad

El diseño de la interfaz ha tenido siempre presente la usabilidad, comprobando reiteradamente que se cumplen tanto Los 8 principios de usabilidad del Software de
Ben Shneiderman como las 10 reglas heurísticas de Nielsen u otros principios comunes de usabilidad. Aquí se indican algunos ejemplos.

- **Contraste:** Se ha intentado mantener siempre un contraste suficiente que permita diferenciar los distintos elementos entre ellos.
- **Coherencia:** Se ha buscado desde el prototipo la coherencia externa mediante el seguimiento de estándares y la coherencia interna.
    - Ejemplos coherencia interna: Los botones casi siempre son amarillos y redondeados por los bordes, los formularios de “My Team Comps” y “Start Team” tienen diseños comunes
    - Ejemplos coherencia externa: Uso del interrogante para la pantalla de ayuda, botón de login posicionado en la parte superior derecha de la pantalla.
- **Errores:** El diseño trata de informar de los errores de forma clara, aportando posibles soluciones y permitiendo volver atrás en caso de necesidad sin que ello suponga grandes problemas.
    - Ejemplos identificación de errores: Aviso en la pestaña de login al intentar loguearte con campos vacíos, avisos en la pestaña de register al introducir valores no válidos.
    - Ejemplos de ayuda en la solución de errores: Redirección del usuario a la pestaña de login al tratar de utilizar funciones o entrar a páginas que requieren loguearse.
    - Ejemplos de reversión de acciones: Las composiciones publicadas pueden volver a hacerse privadas, podemos borrar o editar los campos de composiciones ya creadas, podemos alterar los datos del perfil de usuario en cualquier momento.
- **Feedback:** Las acciones realizadas en el sistema obtienen una retroalimentación inmediata que permite al usuario saber que han surtido efecto.
    - Ejemplos: Las tres opciones iniciales (My Team Comps, Start Team y Community Comps se mueven al posicionar el ratón encima, los votos en la pantalla de Community Comps alteran su número y las flechas alteran su color cuando el usuario vota, las composiciones editadas muestran sus nuevos valores nada más guardar su edición.
- **Ayuda y documentación:** Existe una sección de ayuda y soporte con datos de contacto y un texto informativo. Además existe el manual de instalación.

Accesibilidad

Se ha utilizado la herramienta web examinator para medir la accesibilidad, esta aplicación devuelve una nota de 1 a 10 y el número de elementos “Muy buenos”, “Buenos”, “Malos” y “Muy malos” presentes nuestro código html en función de los criterios de conformidad de las WCAG 2.0 comprobables automáticamente. Hemos tratado de aumentar la nota de las distintas pantallas y de eliminar los errores más graves. La nota de todas las pantallas supera el 5,5, siendo en la mayoría de ellas superior a 6.