

TFC

Desarrollo de una Plataforma Web con PHP, MySQL, Plugin de Minecraft Java, Docker y Kubernetes

Realizado por: Adrián Gómez Morales

Tutor del proyecto: D. Nicolás José del Pozo Madroñal

Centro: I.E.S Triana

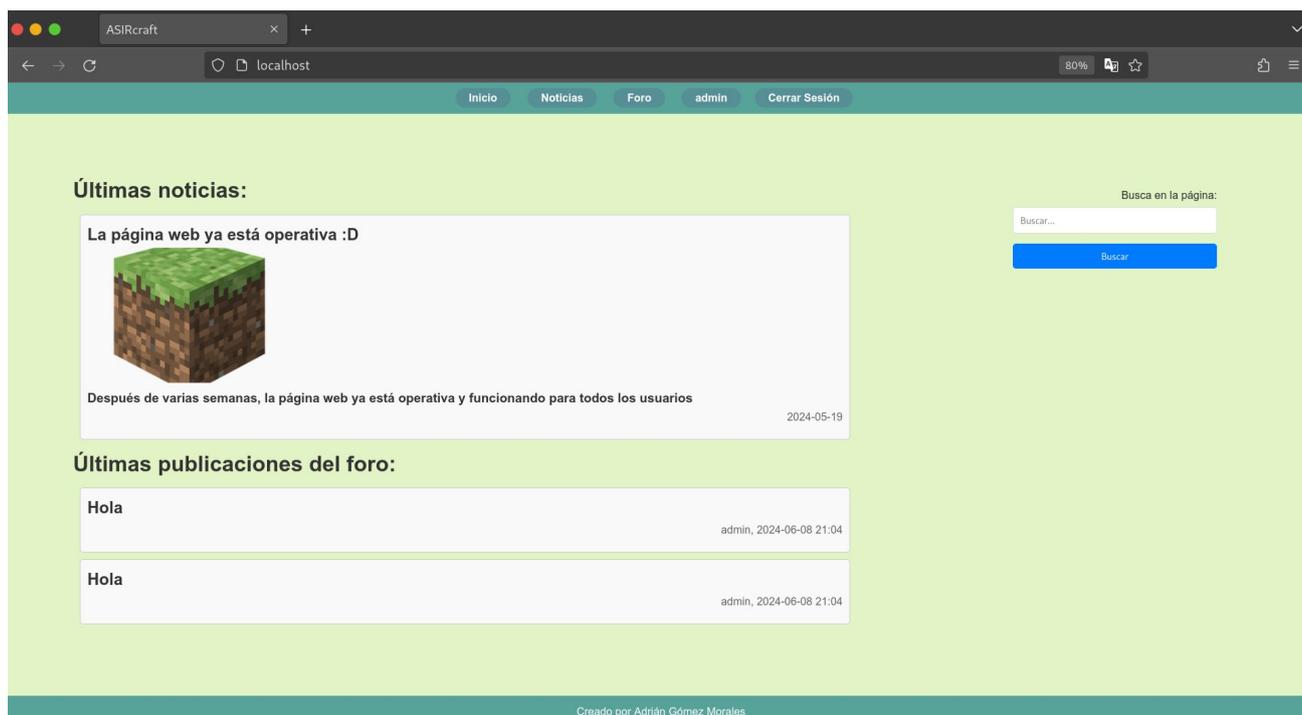
1.	Página web creada con PHP	3
	Inicio	3
	Noticias	4
	Foro	7
	Área de Usuarios	10
	Inciar Sesión	14
	Registrar Usuario	14
2.	Base de datos MySQL	15
	Tabla web	16
	Tabla Minecraft	16
	Usuario de la Base de datos	17
3.	Plugin para Minecraft Server usando Java	18
	Importación de librerías	18
	Clase principal	18
	Método onEnable	19
	Método onDisable	20
	Manejo del evento PlayerJoinEvent	21
	Método actualizarHorasOnline	22
	Manejo del evento PlayerQuitEvent	23
	Manejo del evento PlayerMoveEvent	24
	Manejo del evento PlayerDropItemEvent	24
	Manejo del evento PlayerDeathEvent	25
	Manejo del evento BlockBreakEvent	25
	Método onCommand	26
	Método estaJugadorRegistrado	29
	Método comprobarContrasenia	30
	Manejo del evento PlayerDeathEvent	31
	Manejo del evento BlockPlaceEvent	32
4.	Docker	33
	Instalación de docker	33
	Imágenes personalizadas	38
	Minecraft	38
	Web	41
5.	Kubernetes	44
	Secret para acceder al repositorio privado de Docker Hub	46
	Minecraft en Kubernetes	46
	Base de datos MySQL en Kubernetes	50
	Archivo yaml	50
	Web en Kubernetes	53
	Nota Importante:	58
6.	Cliente Minecraft	59
7.	Vídeo demostrativo	61

1. Página web creada con PHP

Es una página creada desde cero, sin usar ningún CMS como WordPress (El código está comentado con las explicaciones)

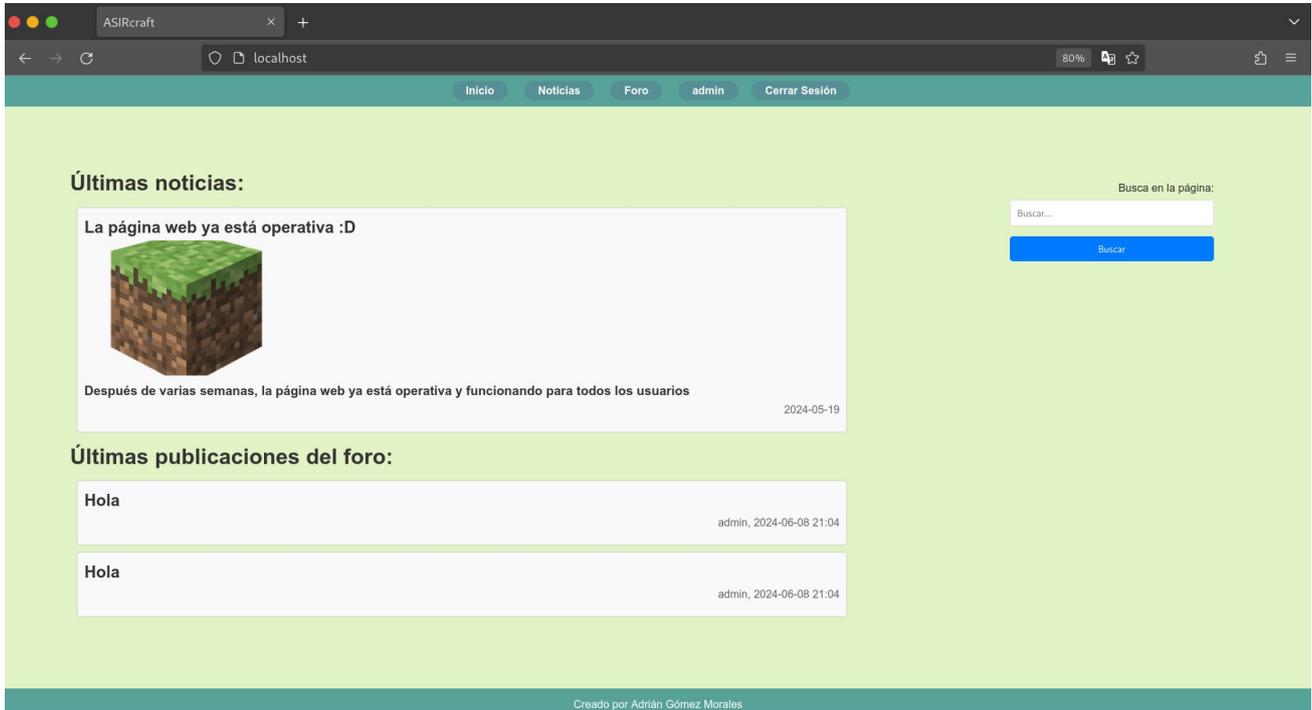
Inicio

En el inicio del sitio, aparecerán las dos últimas noticias y los dos últimos posts del foro, además de la posibilidad de buscar noticias, además de forma interna, detecta si existe un usuario con rol admin y en caso de que no, crea un usuario llamado admin con contraseña changeme

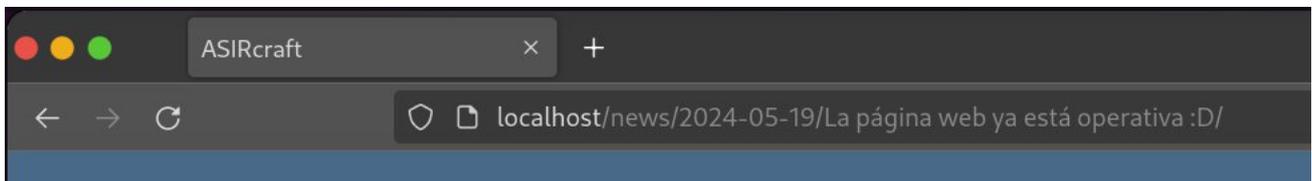


Noticias

En la página de noticias, podemos ver todas las dos últimas noticias que se han publicado en la página Web



Además, tenemos la opción de buscar las noticias si indicamos un texto dentro del formulario de la derecha



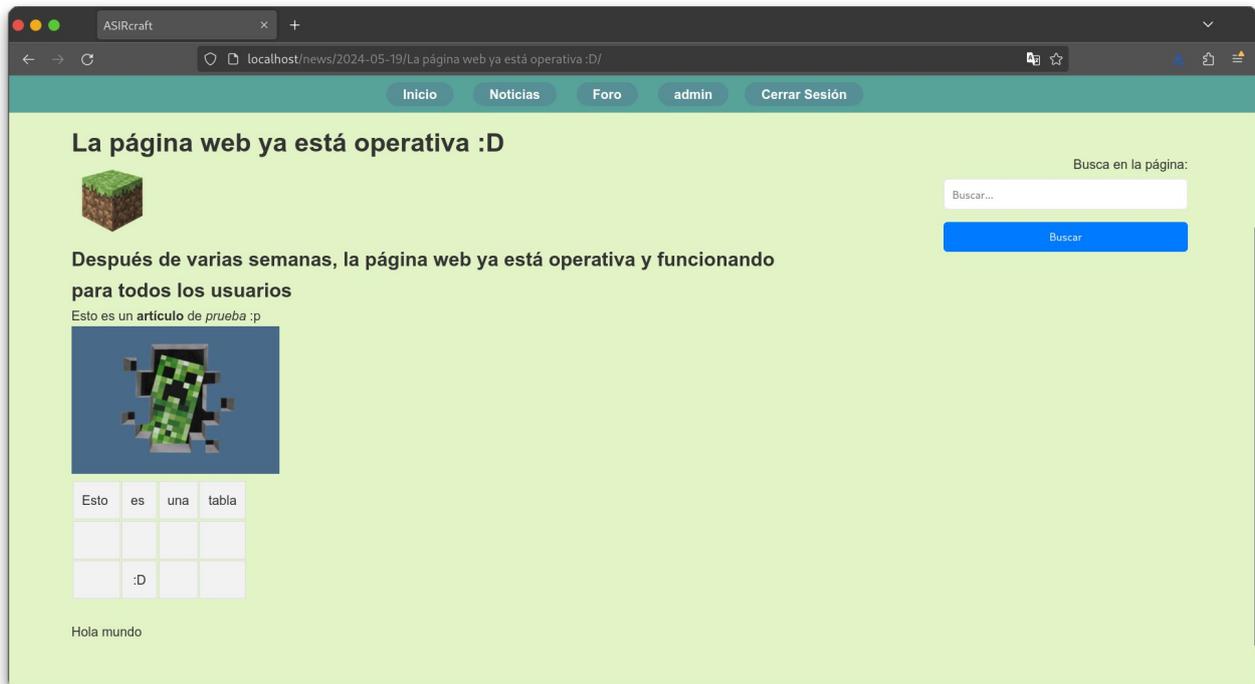
Las noticias se guardan en un archivo json ubicado en /news, al momento de publicar la noticia, creará una carpeta con la fecha en formato YYYY-MM-DD y dentro de esta carpeta, creará otra carpeta con el titular de la noticia y un archivo index.php con la noticia en sí

Noticias.json:

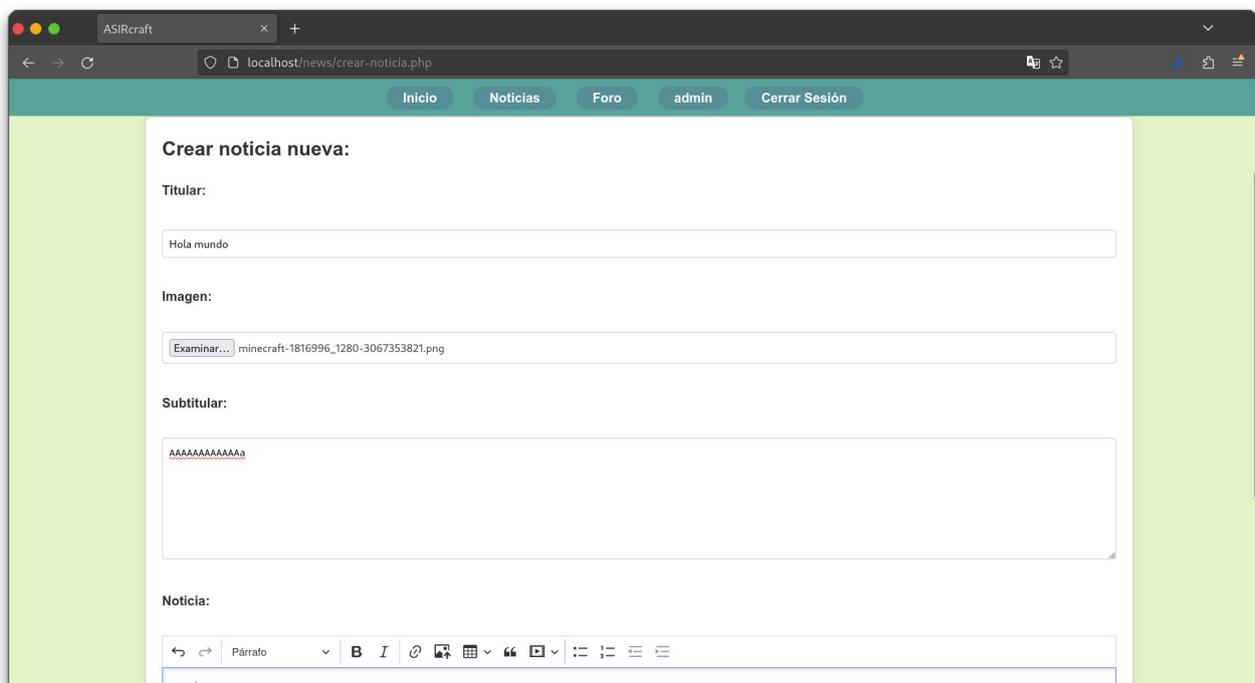
```
news > f) noticias.json > ...
1  [
2  {
3    "fecha": "2024-05-19",
4    "titular": "La página web ya está operativa :D",
5    "imagen": ".../uploads/664a662a7d985-1716151850-minecraft-1816996_1280-3067353821.png",
6    "texto": "Después de varias semanas, la página web ya está operativa y funcionando para todos los usuarios",
7    "noticia": "<p>Esto es un <strong>artículo</strong> de <i>prueba</i> :p</p><figure class='image'><img style='aspect-ratio:255/182;' src"
8  }
9  ]
```

Aunque esto no guarda en sí la noticia, es usado para crear el enlace a la noticia para filtrar usando el buscador de noticias

Si damos click en cualquier parte del rectángulo de la noticia, podemos entrar a verla:



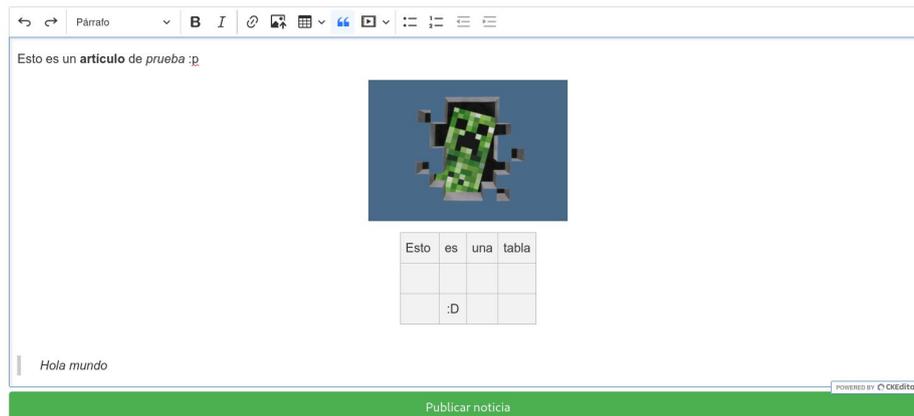
Solo los usuarios administradores pueden crear noticias dentro de la página web mediante un formulario, que pregunta el titular, el subtítular, una imagen destacada y el texto o el cuerpo de la noticia mediante CKEditor



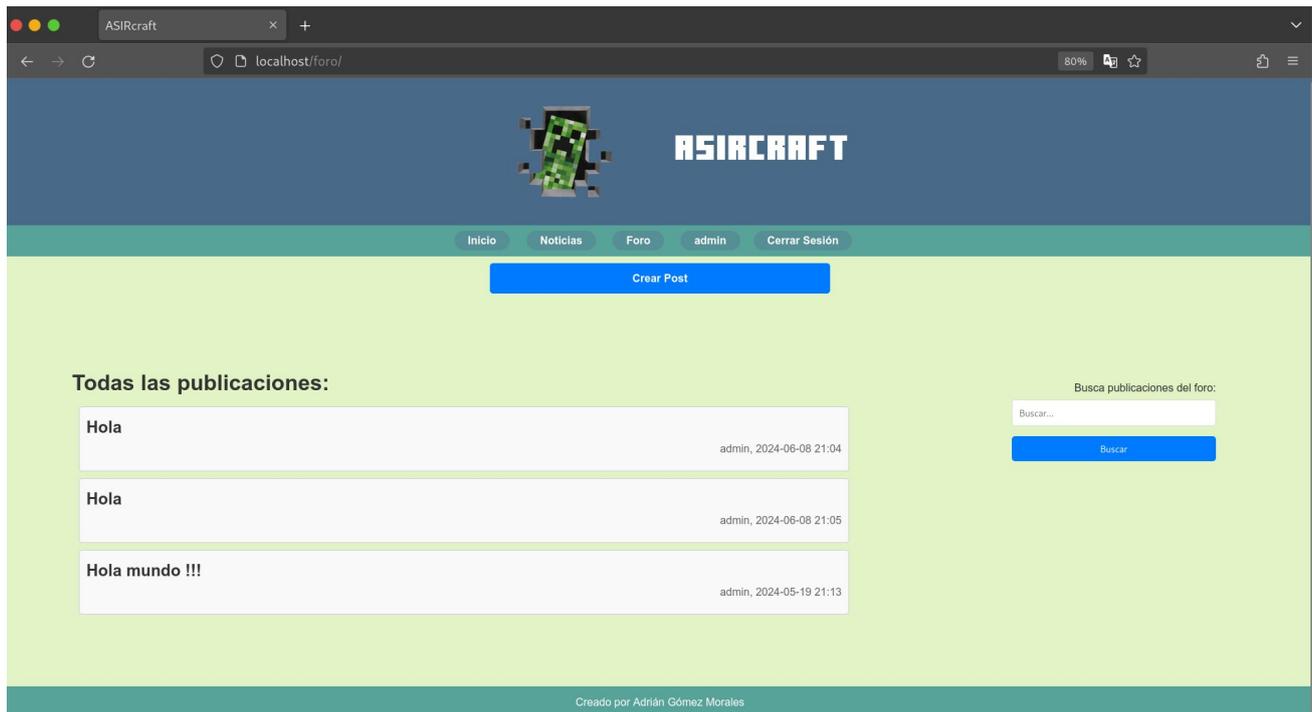
CKEditor y CKFinder

CKEditor es un editor de texto enriquecido que se utiliza comúnmente en aplicaciones web para permitir a los usuarios formatear texto de manera similar a un procesador de texto. Proporciona una interfaz de usuario intuitiva que incluye opciones para cambiar el estilo del texto, insertar imágenes, crear listas, etc... [<https://ckeditor.com/ckeditor-5/>]

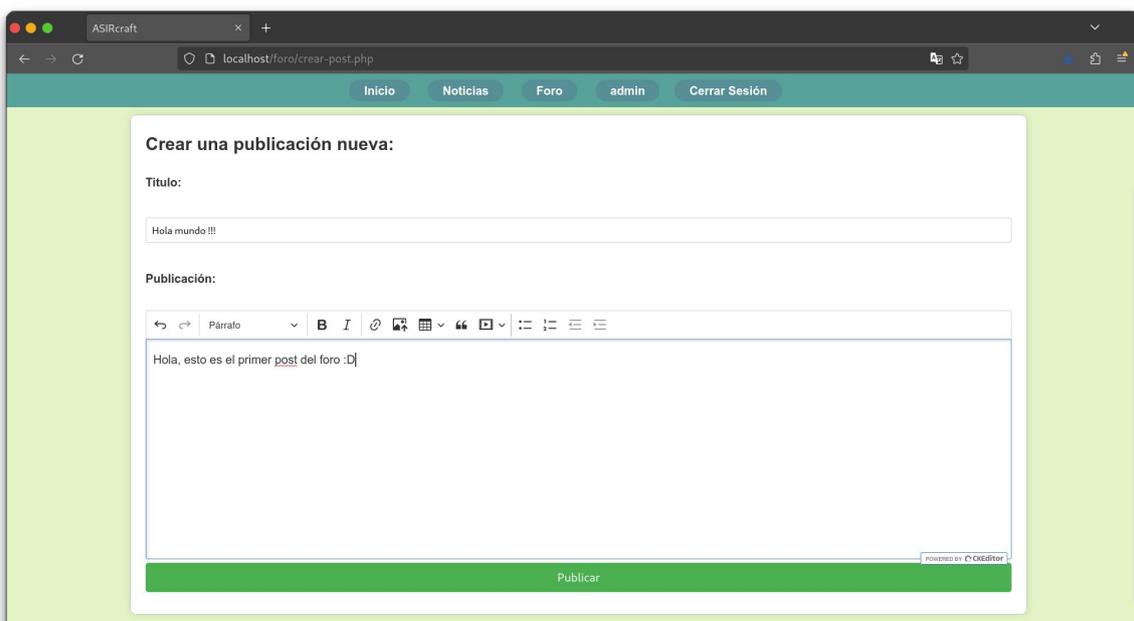
CKFinder es una herramienta complementaria de CKEditor que se utiliza para la gestión de archivos en aplicaciones web. Permite a los usuarios subir, organizar y gestionar archivos de forma sencilla a través de una interfaz intuitiva. [<https://ckeditor.com/ckfinder/>]



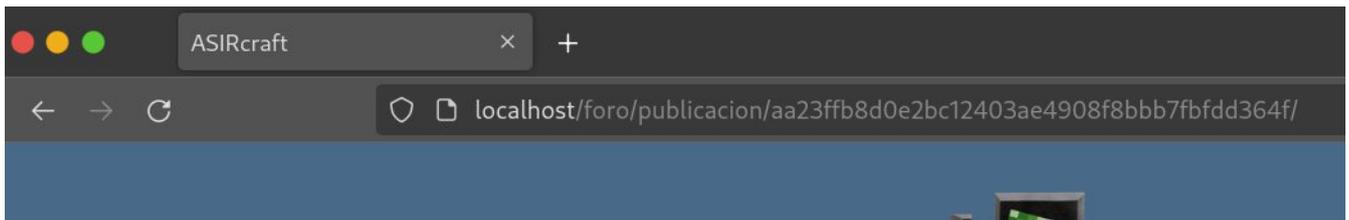
Foro



El foro es donde los usuarios identificados pueden crear Posts para que estos sean comentados por otros usuarios, la idea sería en un supuesto de que si por ejemplo un usuario encuentra algo en el juego o le ocurre cualquier tipo de problema técnico, pueda crear un post para pedir ayuda a otros usuarios. Para abrir el formulario de crear el post, tenemos que dar click en el “botón” de arriba de “Crear Post”



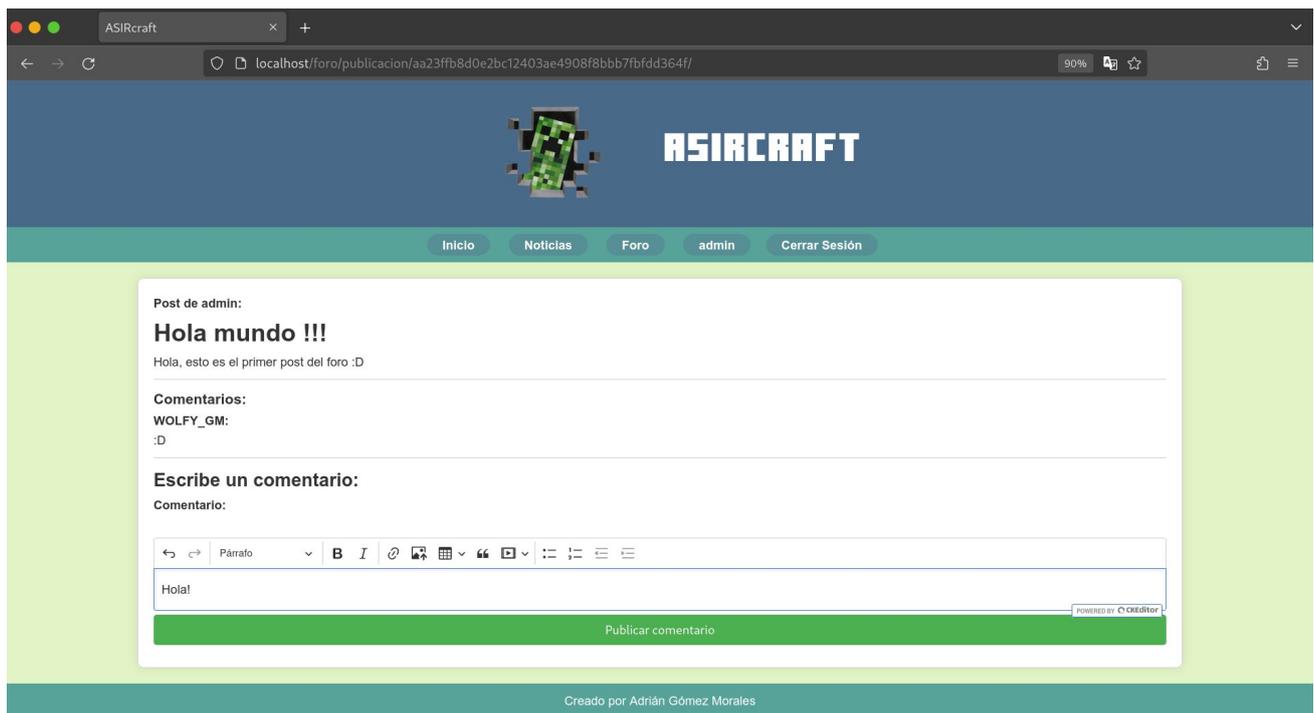
Al igual que con las noticias, CkEditor es usado como procesador de texto



Los posts se crean dentro del directorio publicacion con un hash que es la union de la fecha, titular y usuario cifrado en sha1, para evitar conflictos de varias carpetas con el mismo nombre en la misma ruta

```
{ } foro.json x
foro > { } foro.json > { } 1 > noticia
1  [
2    {
3      "fecha": "2024-06-08 21:04",
4      "titular": "Hola",
5      "noticia": "<p>Hola</p>",
6      "usuario": "admin",
7      "ruta": "0a2ae4ee7db6b6e41a92ec76df0992abf02cea34f"
8    },
9    {
10     "fecha": "2024-06-08 21:05",
11     "titular": "Hola",
12     "noticia": "<p>Hola</p>",
13     "usuario": "admin",
14     "ruta": "1f7df71dd1c1357b75054ffc6afa321d730bb7eb"
15   },
16   {
17     "fecha": "2024-05-19 21:13",
18     "titular": "Hola mundo !!!",
19     "noticia": "<p>Hola, esto es el primer post del foro :D</p>",
20     "usuario": "admin",
21     "ruta": "aa23ffb8d0e2bc12403ae4908f8bbb7fbfdd364f"
22   }
23 ]
```

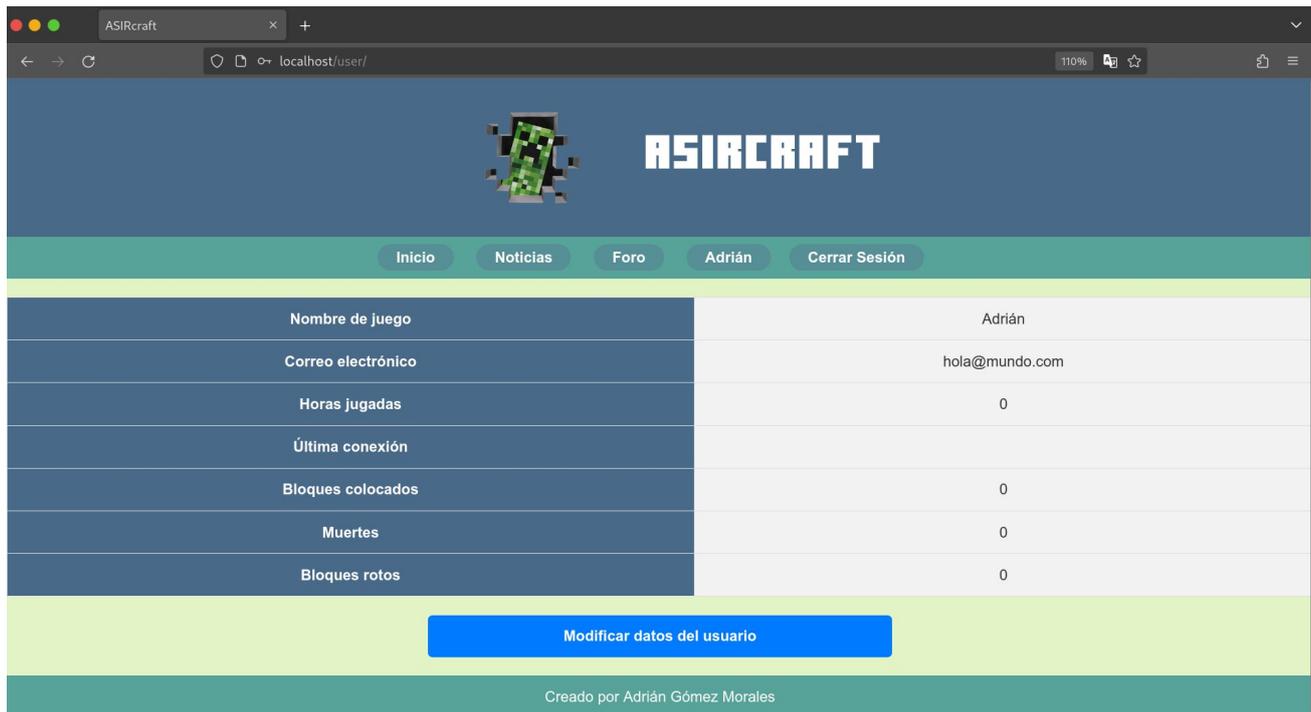
Este sería el json con la información del foro, aunque esto no guarda en sí el post del foro, es usado para crear el enlace al post y para filtrar usando el buscador de post, la página del post se crea usando una plantilla predefinida



Si estás autenticado, podrás enviar comentarios al post, usando CkEditor, los comentarios se guardarán en un archivo json dentro del directorio de la publicación

```
{} comentarios.json x
foro > publicacion > aa23ffb8d0e2bc12403ae4908f8bbb7fbfd364f > {} comentarios.json > ...
1  [
2    {
3      "usuario": "WOLFY_GM",
4      "comentario": "<p>:D</p>"
5    }
6  ]
```

Área de Usuarios

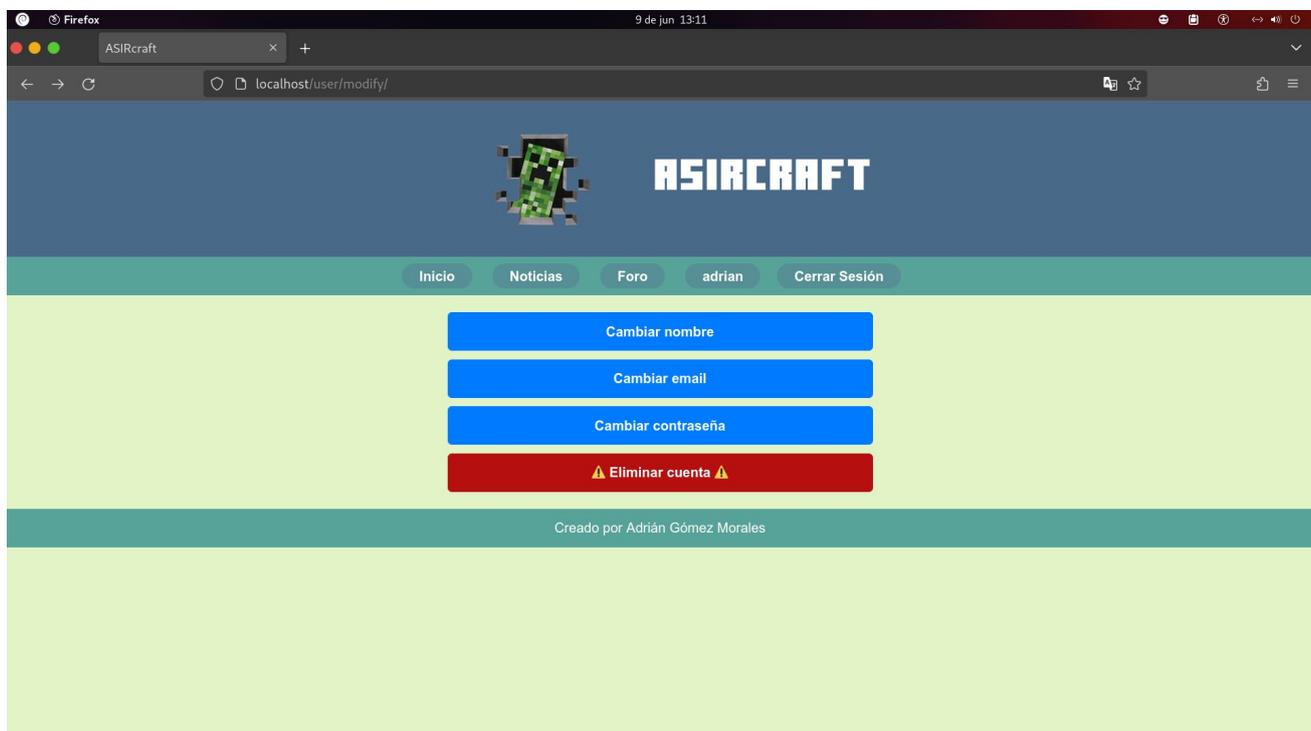


Nombre de juego	Adrián
Correo electrónico	hola@mundo.com
Horas jugadas	0
Última conexión	
Bloques colocados	0
Muertes	0
Bloques rotos	0

Modificar datos del usuario

Creado por Adrián Gómez Morales

Este es el área de usuarios, en esta página, los usuarios podrán ver las estadísticas de su juego y además podrán acceder a modificar los datos de su propio usuario



Cambiar nombre

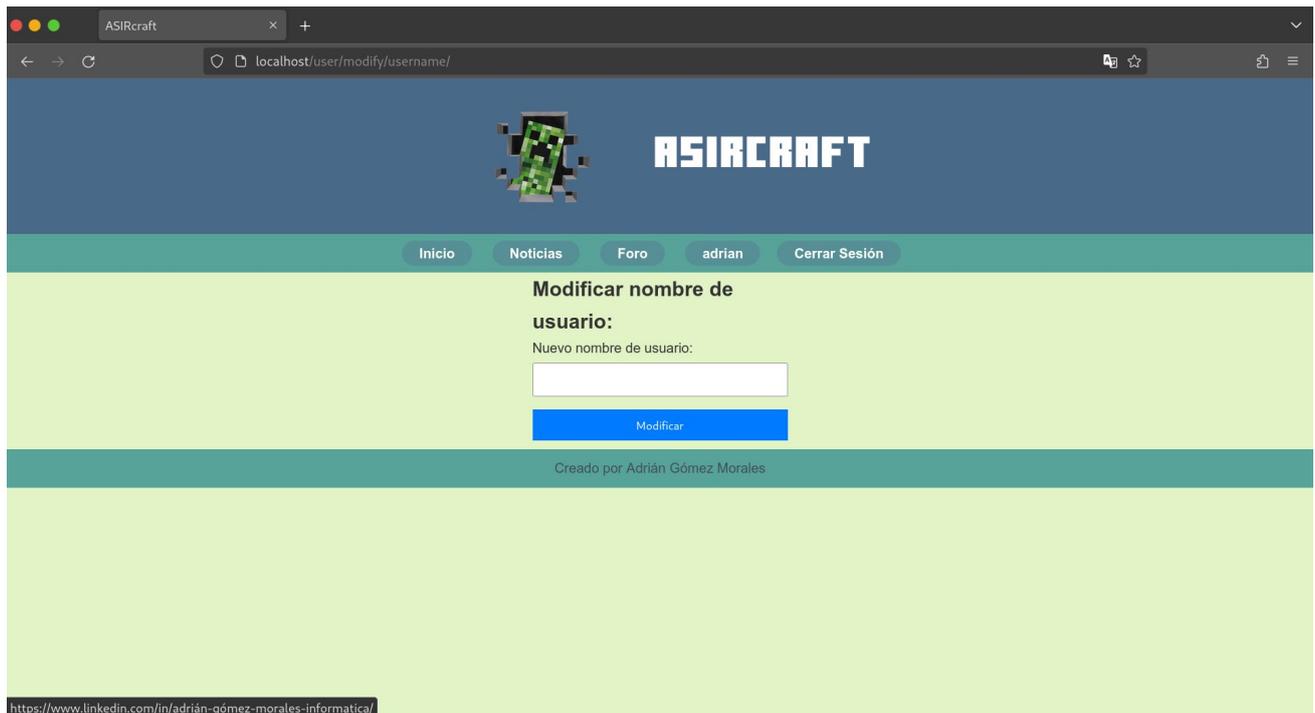
Cambiar email

Cambiar contraseña

⚠ Eliminar cuenta ⚠

Creado por Adrián Gómez Morales

Si damos click en “Modificar datos del usuario” entraremos a un menú para cambiar los datos como el nombre, email o la contraseña, también podremos eliminar nuestra cuenta



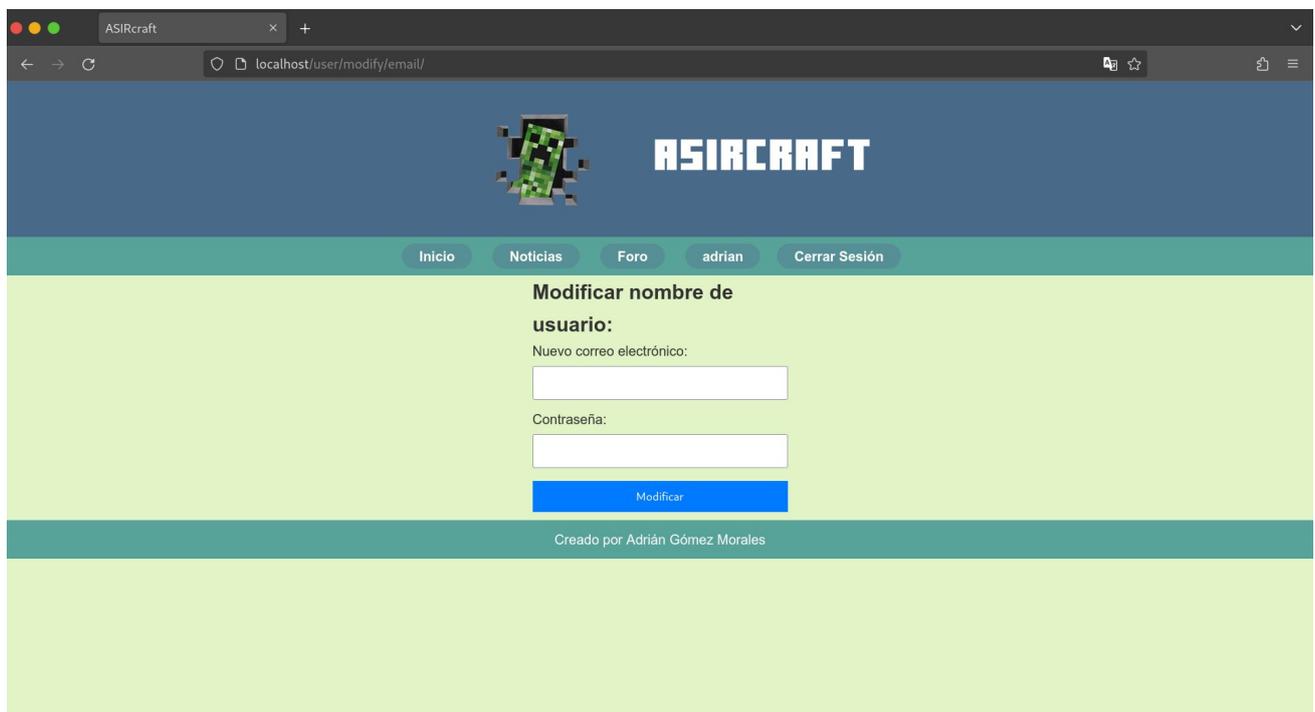
The screenshot shows a web browser window with the URL `localhost/user/modify/username/`. The page features a dark blue header with the ASIRCRAFT logo and a green navigation bar with buttons for 'Inicio', 'Noticias', 'Foro', 'adrian', and 'Cerrar Sesión'. The main content area is light green and contains the following text and form:

Modificar nombre de usuario:
Nuevo nombre de usuario:

Created by Adrián Gómez Morales

<https://www.linkedin.com/in/adrián-gómez-morales-informatica/>

Si entramos en “Cambiar nombre”, entramos en una página con un formulario para actualizar el nuevo nombre_juego de la BBDD, antes de realizar el UPDATE en la BBDD, comprobamos que ese usuario no exista



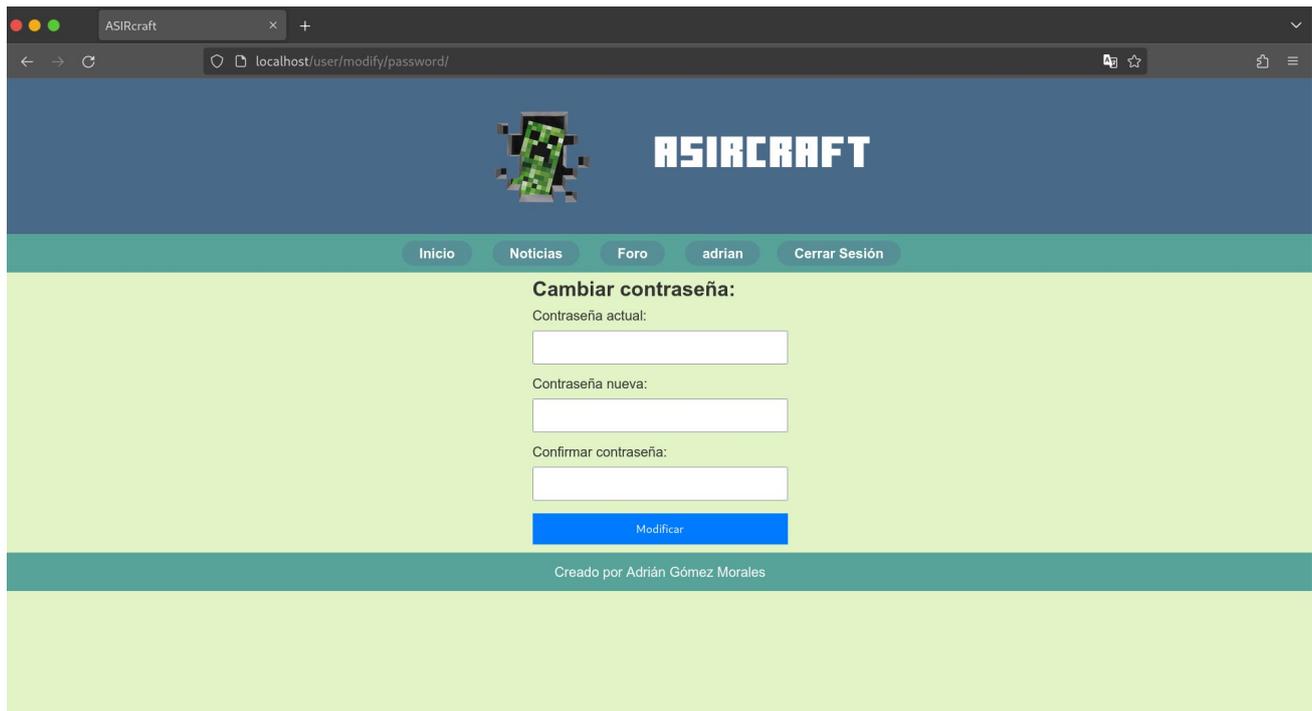
The screenshot shows a web browser window with the URL `localhost/user/modify/email/`. The page features the same ASIRCRAFT logo and navigation bar as the previous screenshot. The main content area is light green and contains the following text and form:

Modificar nombre de usuario:
Nuevo correo electrónico:

Contraseña:

Created by Adrián Gómez Morales

Si entramos a “Cambiar email”, accedemos a un formulario para cambiar nuestro correo, nos pedirá el nuevo correo y la contraseña que está usando el usuario, comprueba que la contraseña coincida con la almacenada en la BBDD y si es así, actualiza el correo



The screenshot shows a web browser window with the URL `localhost/user/modify/password/`. The page features a dark blue header with a Creeper character icon and the text "ASIRCRAFT". Below the header is a navigation bar with buttons for "Inicio", "Noticias", "Foro", "adrian", and "Cerrar Sesión". The main content area is light green and contains the following form:

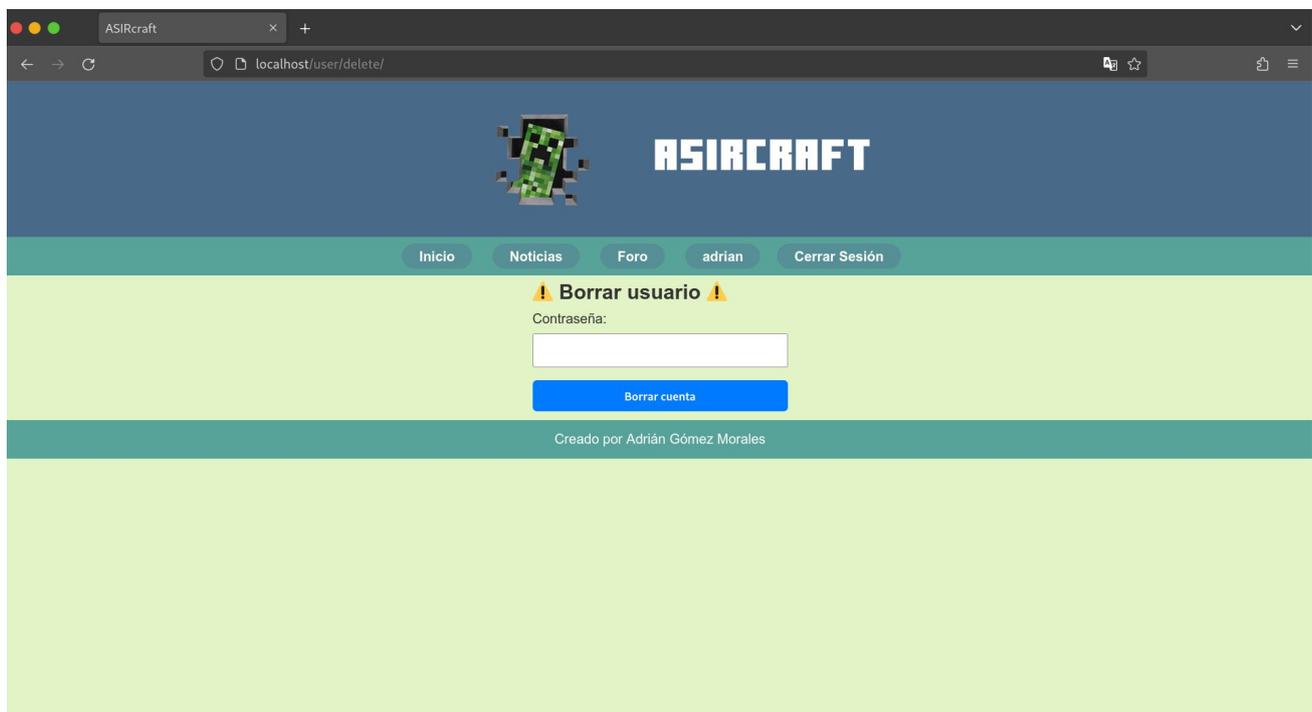
Cambiar contraseña:
Contraseña actual:

Contraseña nueva:

Confirmar contraseña:

At the bottom of the form area, it says "Creado por Adrián Gómez Morales".

Si entramos al botón de “Cambiar contraseña”, accedemos a otro formulario donde nos pedirá la contraseña actual y dos veces la nueva contraseña, por detrás, comprobará que las dos contraseñas actuales coincidan y si es así, actualizará la contraseña en la BBDD



The screenshot shows a web browser window with the URL `localhost/user/delete/`. The page features the same header and navigation bar as the previous screenshot. The main content area is light green and contains the following form:

⚠ Borrar usuario ⚠
Contraseña:

At the bottom of the form area, it says "Creado por Adrián Gómez Morales".

Si entramos a “Eliminar cuenta” entramos en un formulario que nos pedirá la contraseña actual y en caso de que coincida con la de la BBDD, el usuario se eliminará de ambas tablas

The screenshot shows a web browser window with the URL localhost/user/. The page has a navigation bar with links: Inicio, Noticias, Foro, admin, and Cerrar Sesión. Below the navigation bar, there are four statistics cards: Última conexión, Bloques colocados, Muertes, and Bloques rotos, all showing a value of 0. A blue button labeled 'Modificar datos del usuario' is visible. Below this, the section 'Administración web:' contains a blue button 'Crear una nueva noticia'. The 'Usuarios:' section features a table with the following data:

Usuario	Correo electrónico	Horas jugadas	Última conexión	Rol	Status
admin	example@asircraft.com	0		admin	Offline
Adrian	hola@mundo.com	0		usuario	Offline

At the bottom of the page, it says 'Creado por Adrián Gómez Morales'.

Volviendo al panel del usuario, si entramos al mismo panel pero como un usuario con rol de admin, podemos ver una lista de usuarios registrados, su correo, el número de horas jugadas, así como su última conexión rol y si están conectados o no, además tendremos acceso a crear noticias

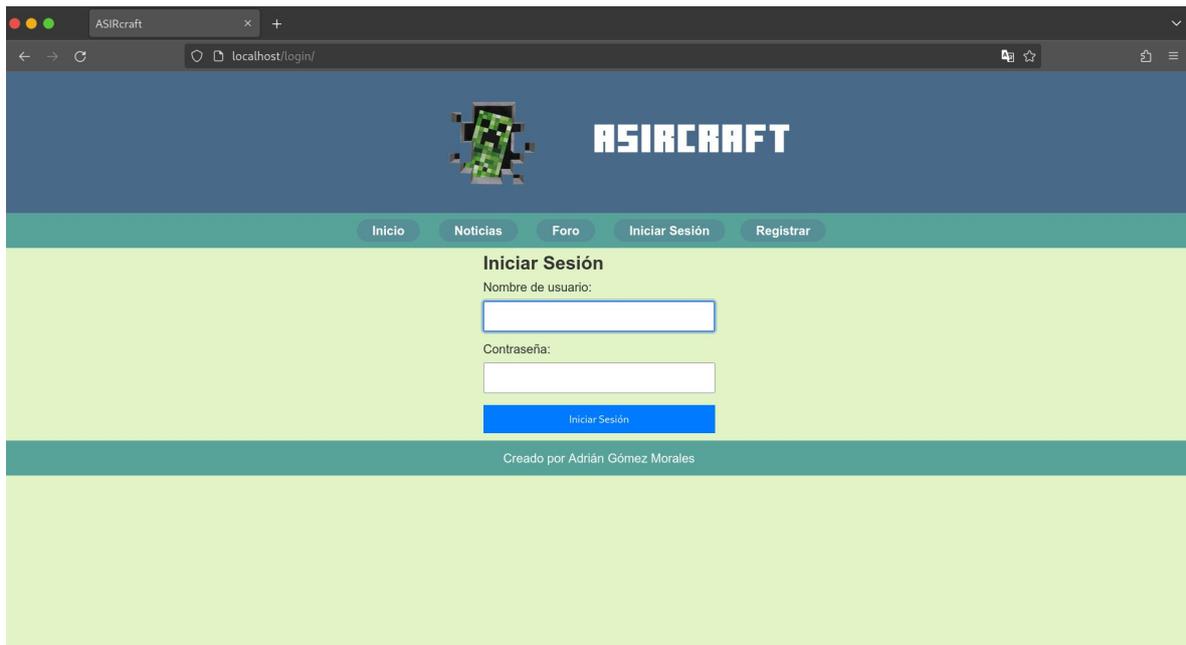
The screenshot shows a web browser window with the URL localhost/news/crear-noticia.php. The page has a navigation bar with links: Inicio, Noticias, Foro, admin, and Cerrar Sesión. The main content area is a form titled 'Crear noticia nueva:'. The form has the following fields:

- Titular:** A text input field.
- Imagen:** A file upload field with a button labeled 'Elegir...' and a message 'no se ha seleccionado ningún archivo'.
- Subtitular:** A text input field.
- Noticia:** A rich text editor with a toolbar containing options like Bold, Italic, Underline, and others.

At the bottom of the form, there is a green button labeled 'Publicar noticia'. At the bottom of the page, it says 'Creado por Adrián Gómez Morales'.

Si damos click en “Crear nueva noticia”, nos redirigirá a una página con un formulario para poder crear noticias, solo los usuarios administradores pueden entrar

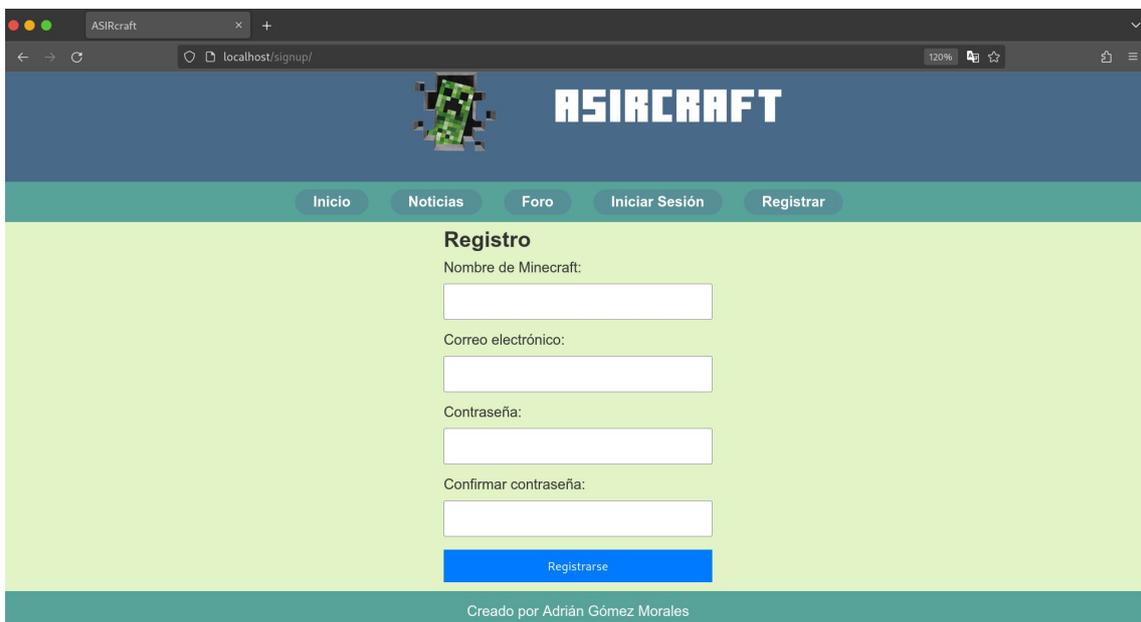
Iniciar Sesión



The screenshot shows a web browser window with the URL localhost/login/. The page features a dark blue header with the ASIRCRAFT logo and a green navigation bar with links for Inicio, Noticias, Foro, Iniciar Sesión, and Registrar. The main content area is light green and contains a login form titled 'Iniciar Sesión'. The form has two input fields: 'Nombre de usuario:' and 'Contraseña:'. Below the fields is a blue button labeled 'Iniciar Sesión'. At the bottom of the page, it says 'Creado por Adrián Gómez Morales'.

Para ello debemos de entrar en la página /login, donde nos pedirá el usuario y la contraseña en un formulario y en caso de que el usuario y la contraseña coincida con la de la BBDD, podremos iniciar sesión

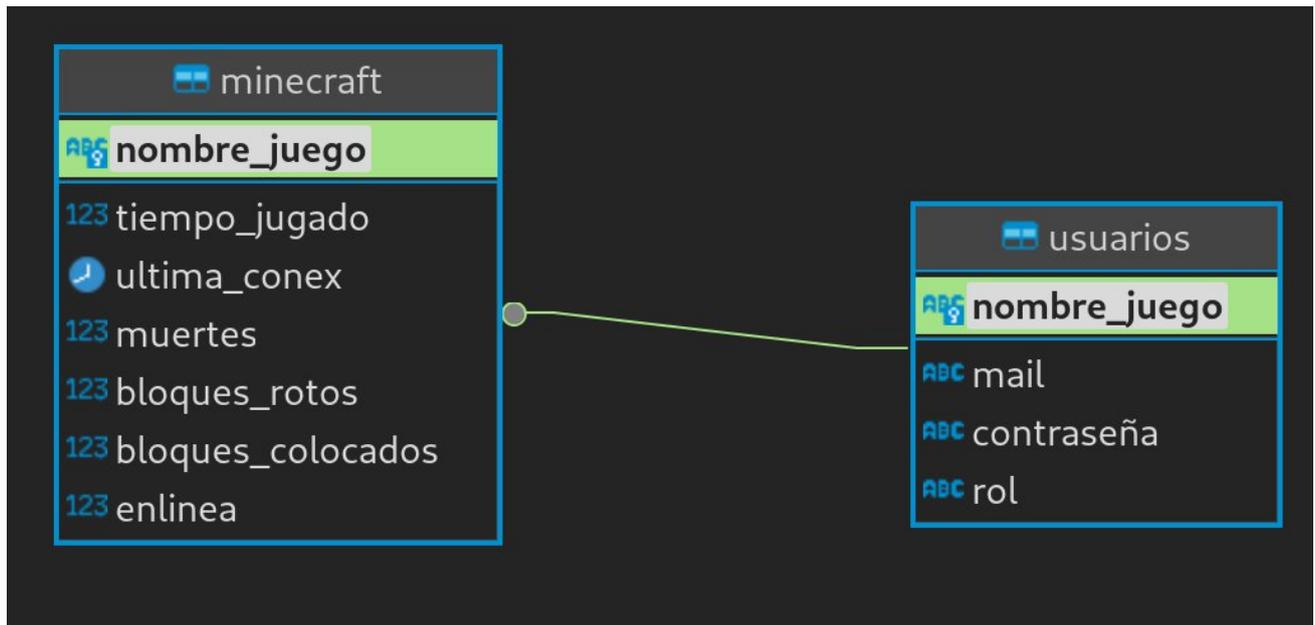
Registrar Usuario



The screenshot shows a web browser window with the URL localhost/signup/. The page features a dark blue header with the ASIRCRAFT logo and a green navigation bar with links for Inicio, Noticias, Foro, Iniciar Sesión, and Registrar. The main content area is light green and contains a registration form titled 'Registro'. The form has four input fields: 'Nombre de Minecraft:', 'Correo electrónico:', 'Contraseña:', and 'Confirmar contraseña:'. Below the fields is a blue button labeled 'Registrarse'. At the bottom of the page, it says 'Creado por Adrián Gómez Morales'.

La página de registro se encuentra en /signup, nos pedirá un nombre de usuario (debe ser distinto a los que están guardados en la BBDD), un email, una contraseña y la confirmación de la contraseña, por defecto los usuarios que se registren mediante este formulario, se crearán con el rol de usuario

2. Base de datos MySQL



```

CREATE DATABASE IF NOT EXISTS proyecto;

USE proyecto;

CREATE TABLE IF NOT EXISTS usuarios (
  nombre_juego VARCHAR(255),
  mail VARCHAR(255),
  contraseña VARCHAR(255),
  rol VARCHAR(50),
  PRIMARY KEY (nombre_juego)
);

CREATE TABLE IF NOT EXISTS minecraft (
  nombre_juego VARCHAR(255) NOT NULL,
  tiempo_jugado INT DEFAULT 0,
  ultima_conex DATE DEFAULT NULL,
  muertes INT DEFAULT 0,
  bloques_rotos INT DEFAULT 0,
  bloques_colocados INT DEFAULT 0,
  enlinea SMALLINT NOT NULL DEFAULT 0,
  PRIMARY KEY (nombre_juego),
  FOREIGN KEY (nombre_juego) REFERENCES usuarios(nombre_juego)
  ON DELETE CASCADE
  ON UPDATE CASCADE
);

CREATE USER IF NOT EXISTS 'user_proyect'@'%' IDENTIFIED WITH mysql_native_password BY 'H4-D14s*fVj.4_P35k4R';

GRANT ALL PRIVILEGES ON proyecto.* TO 'user_proyect'@'%';

FLUSH PRIVILEGES;

```

Script de la base de datos, creará una base de datos si no existe llamada proyecto, que tendrá dos tablas, tabla web:

Tabla web

Esta tabla almacena información de los usuarios de la web con los siguientes campos:

nombre_juego (VARCHAR(255))

- Es el identificador único para cada usuario. Representa el nombre que el usuario utiliza en el juego
- Restricción: Clave primaria (PRIMARY KEY), lo que significa que cada valor debe ser único y not null

mail (VARCHAR(255))

- Almacena la dirección de correo electrónico del usuario

contrasea (VARCHAR(255))

- Contraseña que será usada para iniciar sesión por parte del usuario

Nota: No se puede usar la 'ñ', da problemas luego para kubernetes

rol (VARCHAR(50))

- Indica el rol que tendrá el usuario (usuario o admin)

Tabla Minecraft

Esta tabla almacena la información relacionada con el juego en sí

nombre_juego (VARCHAR(255))

- Es el identificador único para cada usuario. Representa el nombre que el usuario utiliza en el juego
- Restricción: Clave primaria (PRIMARY KEY), lo que significa que cada valor debe ser único y not null y además es clave foránea de la tabla usuarios.

tiempo_jugado (INT)

- Usada para guardar el tiempo que pasa el jugador online, es de tipo numérico que por defecto asignará el valor de 0

Ultima_conex (DATE)

- Usada para guardar la última fecha a la que se conectó el usuario al juego, es de tipo fecha y por defecto es NULL

Bloques_rotos (INT)

- Usada para guardar el número de bloques que rompe el jugador, es de tipo numérico que por defecto asignará el valor de 0

Bloques_colocados (INT)

- Usada para guardar el número de bloques que coloca el jugador, es de tipo numérico que por defecto asignará el valor de 0

enlinea (SMALLINT)

- Usada para guardar el estado del jugador, 1 representa online y 0 representa que está offline, es de tipo numérico que por defecto asignará el valor de 0

ON DELETE/UPDATE CASCADE

La cláusula ON DELETE y ON UPDATE con la opción CASCADE en bases de datos SQL se utiliza para mantener la integridad referencial entre las tablas. La integridad referencial asegura que los valores en una tabla (la tab hija) siempre sean válidos en relación con los valores en otra tabla (la tab padre). Cuando se realiza una operación de eliminación o actualización en la tabla padre, estas opciones permiten que automáticamente se realicen cambios correspondientes en la tabla hija.

Usuario de la Base de datos

Creamos un usuario que permita la conexión desde cualquier IP ("%"), esto se debe a que no sabemos que IP tendrá los distintos servicios que usarán la BBDD. Además asignamos todos los privilegios al usuario para todas las tablas de esa BBDD

3. Plugin para Minecraft Server usando Java

Importación de librerías

```
Verysimplesqllogin.java x
1 package es.adriangomezmorales.verysimplesqllogin;
2
3 import org.bukkit.command.Command;
4 import org.bukkit.command.CommandSender;
5 import org.bukkit.command.CommandExecutor;
6 import org.bukkit.entity.Player;
7 import org.bukkit.event.EventHandler;
8 import org.bukkit.event.Listener;
9 import org.bukkit.event.entity.PlayerDeathEvent;
10 import org.bukkit.event.player.PlayerMoveEvent;
11 import org.bukkit.event.player.PlayerJoinEvent;
12 import org.bukkit.event.player.PlayerQuitEvent;
13 import org.bukkit.plugin.java.JavaPlugin;
14 import org.bukkit.event.block.BlockBreakEvent;
15
16 import java.sql.*;
17 import java.util.HashMap;
18 import java.util.Map;
19 import java.util.UUID;
20
```

Se importa las librerías necesarias para trabajar con Bukkit API (La API de minecraft server) y JDBC para las conexiones a la base de datos

Clase principal

```
21 public final class Verysimplesqllogin extends JavaPlugin implements CommandExecutor, Listener {
22     private Map<UUID, Boolean> sesionesJugador = new HashMap<>();
23     private Connection conexion;
24
```

Define la clase principal Verysimplesqllogin que extiende JavaPlugin, implementando CommandExecutor y Listener para manejar comandos y eventos de Minecraft. Declara un mapa sesionesJugador para rastrear las sesiones de los jugadores y una conexión conexion a la base de datos. El resto del código está dentro de esta clase principal

Método onEnable

```

27  @Override no usages
28  public void onEnable() {
29      getLogger().info(msg: "Inicio del plugin de login SQL simple por Adrian Gomez Morales");
30      getServer().getPluginManager().registerEvents(listener: this, plugin: this);
31      getCommand(name: "login").setExecutor(this);
32
33      // Conexión a la base de datos
34      try {
35          conexion = DriverManager.getConnection(url: "jdbc:mysql://mysql:3306/proyecto", user: "user_proyecto", password: "H4-D14s*fVj.4_P35k4R");
36      } catch (SQLException e) {
37          getLogger().severe(msg: "Error al conectar a la base de datos: " + e.getMessage());
38      }
39  }

```

El método `onEnable()` de los plugins de Minecraft Server hace referencia al momento de cargarse el plugin dentro del servidor, que por norma general es cuando se está iniciando el propio servidor de juego.

`getLogger().info("Hola");`

Sirve para registrar a "Hola" dentro del log de Minecraft

`getServer().getPluginManager().registerEvents(this, this);`

Indica que el plugin es un listener de eventos. El método `registerEvents` del `PluginManager` se utiliza para que este plugin (representado por `this`) pueda manejar eventos que ocurren en el servidor. El primer `this` se refiere al listener que maneja los eventos, y el segundo `this` se refiere al propio plugin.

`getCommand("login").setExecutor(this);`

Establece que el comando "login" del plugin será gestionado por este mismo objeto. `setExecutor(this)` significa que la clase actual implementa la interfaz `commandExecutor`, lo que permite manejar lo que ocurre cuando un jugador usa el comando `/login` en el juego.

`try{`

`conexion = DriverManager.getConnection("jdbc:mysql://mysql:3306/proyecto", "user_proyecto", "H4-D14s*fVj.4_P35k4R");`

`}`

Intenta establecer una conexión a MySQL. Usa la clase `DriverManager` para obtener una conexión (`getConnection`) a la base de datos ubicada en `mysql` (Nombre de DNS del servicio `mysql` de Kubernetes que crearemos posteriormente) en el puerto 3306, usando el usuario y su contraseña. La conexión establecida se asigna a la variable `conexion`.

```

catch (SQLException e) {

    getLogger().severe("Error al conectar a la base de datos: " +
e.getMessage());

}

```

Si ocurre una excepción, se guarda y se registra en el log del servidor con `getLogger().severe`.

Método onDisable

```

39      @Override no usages
40      public void onDisable() {
41          // Cierre de la conexión a la base de datos
42          if (conexion != null) {
43              try (PreparedStatement statement = conexion.prepareStatement(
44                  "UPDATE minecraft SET online = 0 WHERE 1")) {
45                  statement.executeUpdate();
46              } catch (SQLException e) {
47                  getLogger().severe(msg: "Error al establecer a 'Desconectado' a los jugadores: " + e.getMessage());
48              }
49              try {
50                  conexion.close();
51              } catch (SQLException e) {
52                  getLogger().severe(msg: "Error al cerrar la conexión a la base de datos: " + e.getMessage());
53              }
54          }
55      }

```

`onDisable()` hace referencia a cuando el plugin se desactiva del servidor que por norma general es cuando el servidor se apaga o reinicia

if (conexion != null) {

Comprueba que la conexión a la BBDD se estableció anteriormente y en el caso contrario no hará nada. Si la condición `conexion != null`, es true, entonces ejecutará lo siguiente (En todas los try hay catch que recogen el error y lo almacenan en el log si algo va mal durante la ejecución):

```

try (PreparedStatement statement = conexion.prepareStatement(
    "UPDATE minecraft SET online = 0 WHERE 1")) {

```

```

statement.executeUpdate();

```

Como es cuando por norma general se apaga, debemos de hacer que todos los usuarios que estaban marcados como "Online" deben de pasar a "Offline" ya que si el servidor se apaga o se reinicia, los jugadores se desconectarán del servidor

```

try {
    conexion.close();
}

```

Esta línea cierra la conexión a la BBDD

Manejo del evento PlayerJoinEvent

```

57      @EventHandler
58      public void jugadorSeUne(PlayerJoinEvent event) {
59          Player jugador = event.getPlayer();
60          sesionesJugador.put(jugador.getId(), false);
61          jugador.sendMessage("Usa /login [contraseña] para iniciar sesión");
62
63          // Expulsar al jugador si no está registrado en la base de datos
64          if (!estaJugadorRegistrado(jugador)) {
65              jugador.kickPlayer("¡Por favor, regístrate en nuestra web!");
66          }
67      }

```

Este evento hace referencia a cuando un jugador entra al servidor

Player jugador = event.getPlayer();

Este código obtiene el objeto Player que representa al jugador que se acaba de unir al servidor.

sesionesJugador.put(jugador.getId(), false);

Agrega la UUID (Identificador Único Universal) del jugador al mapa sesionesJugador con un valor de false, indicando que el jugador no ha iniciado sesión todavía.

jugador.sendMessage("Usa /login [contraseña] para iniciar sesión");

El jugador recibe por el chat del juego "Usa /login [contraseña] para iniciar sesión" y solo lo recibe ese jugador en concreto

if (!estaJugadorRegistrado(jugador)) {

jugador.kickPlayer("¡Por favor, regístrate en nuestra web!");

}

Aquí se verifica si el jugador está registrado en la base de datos utilizando el método estaJugadorRegistrado que se definirá más adelante. Si el jugador no está registrado, es expulsado del servidor.

Método actualizarHorasOnline

```

68
69 @ public void actualizarHorasOnline(Player jugador) { usage
70     try (PreparedStatement statement = conexion.prepareStatement(
71         s: "UPDATE minecraft SET tiempo_jugado = tiempo_jugado + TIMESTAMPTDIFF(HOUR, ultima_conex, NOW()) WHERE nombre_juego = ?") {
72         statement.setString(1, jugador.getName());
73         statement.executeUpdate();
74     } catch (SQLException e) {
75         getLogger().severe(msg: "Error al actualizar las horas en línea del jugador: " + e.getMessage());
76     }
77 }
78

```

Este método se usa para actualizar el tiempo de juego de un jugador en la base de datos.

El Parametro jugador es una instancia de la clase Player, que representa al jugador cuyo tiempo de juego se desea actualizar.

try (PreparedStatement statement = conexion.prepareStatement(

"UPDATE minecraft SET tiempo_jugado = tiempo_jugado + TIMESTAMPTDIFF(HOUR, ultima_conex, NOW()) WHERE nombre_juego = ?") {

Aquí se prepara una consulta SQL para actualizar la columna tiempo_jugado en la tabla minecraft. La consulta utiliza la función TIMESTAMPTDIFF para calcular la diferencia en horas (HOUR) entre ultima_conex (la última vez que el jugador se conectó) y NOW() (el momento actual).

statement.setString(1, jugador.getName());

Aquí se establece el valor del parámetro nombre_juego en la consulta SQL. El nombre del jugador se obtiene utilizando jugador.getName()

statement.executeUpdate();

Esta línea ejecuta la consulta

} catch (SQLException e) {

getLogger().severe("Error al actualizar las horas en línea del jugador: " + e.getMessage());

}

Si ocurre una excepción SQL se registra y se guarda en el log

Manejo del evento PlayerQuitEvent

```

79      @EventHandler
80      public void jugadorAbandona(PlayerQuitEvent event) {
81          Player jugador = event.getPlayer();
82          UUID uuid = jugador.getUniqueId();
83          actualizarHorasOnline(jugador);
84          try (PreparedStatement statement = conexion.prepareStatement(
85              "UPDATE minecraft SET online = 0 WHERE nombre_juego = ?") {
86              statement.setString(1, jugador.getName());
87              statement.executeUpdate();
88          } catch (SQLException e) {
89              getLogger().severe("Error al establecer que el jugador NO está en línea: " + e.getMessage());
90          }
91          sesionesJugador.remove(event.getPlayer().getUniqueId());
92      }
93  }

```

Este método es un manejador de eventos (event handler) que se ejecuta cada vez que un jugador se desconecta del servidor

Player jugador = event.getPlayer();

UUID uuid = jugador.getUniqueId();

Obtiene el objeto Player que representa al jugador que se está desconectando y su UUID.

actualizarHorasOnline(jugador);

Llama al método actualizarHorasOnline con el parámetro jugador

```

try (PreparedStatement statement = conexion.prepareStatement(
    "UPDATE minecraft SET online = 0 WHERE nombre_juego = ?") {
    statement.setString(1, jugador.getName());
    statement.executeUpdate();
} catch (SQLException e) {
    getLogger().severe("Error al establecer que el jugador NO está en línea: " +
e.getMessage());
}

```

En esta sección, se intentará ejecutar un update en la tabla minecraft para poner online con valor 0, que significa offline al jugador que deja el servidor y en caso de que falle, registrará la excepción y la guardará en el log

sesionesJugador.remove(event.getPlayer().getUniqueId());

Esta línea elimina la sesión del jugador que abandona

Manejo del evento PlayerMoveEvent

```
94      @EventHandler
95      public void jugadorSeMueve(PlayerMoveEvent event) {
96          Player jugador = event.getPlayer();
97          UUID uuid = jugador.getUniqueId();
98          if (sesionesJugador.containsKey(uuid) && !sesionesJugador.get(uuid)) {
99              event.setCancelled(true); // Cancela el movimiento mientras no haya iniciado sesión
100          }
101      }
```

Este bloque está destinado a cancelar el movimiento del jugador en caso de que no haya iniciado sesión

```
Player jugador = event.getPlayer();
```

```
UUID uuid = jugador.getUniqueId();
```

Obtiene el objeto Player que representa al jugador que se está intentando mover y su UUID.

```
if (sesionesJugador.containsKey(uuid) && !sesionesJugador.get(uuid)) {  
    event.setCancelled(true);  
}
```

Comprueba si el uuid del jugador está presente en el mapa sesionesJugador y en caso de que no sea así, cancelará el evento, es decir, no se podrá mover y por lo tanto no podrá jugar

Manejo del evento PlayerDropItemEvent

```
104      @EventHandler
105      public void soltarItem(PlayerDropItemEvent event) {
106          Player jugador = event.getPlayer();
107          UUID uuid = jugador.getUniqueId();
108          if (sesionesJugador.containsKey(uuid) && !sesionesJugador.get(uuid)) {
109              event.setCancelled(true); // Cancela el dropeo de items mientras no haya iniciado sesión
110          }
111      }
```

Prácticamente hace lo mismo que el anterior, solo que en lugar de cancelar el evento de moverse, cancela el evento de tirar un objeto del inventario del jugador, para evitar que otra persona pueda intentar conectarse como el jugador original y tirar su inventario

Manejo del evento PlayerDeathEvent

```

113 @EventHandler
114 public void cancelarMuerte(PlayerDeathEvent event) {
115     Player jugador = event.getPlayer();
116     UUID uuid = jugador.getUniqueId();
117     if (sesionesJugador.containsKey(uuid) && !sesionesJugador.get(uuid)) {
118         event.setCancelled(true); // Cancela la muerte mientras no haya iniciado sesión
119     }
120 }

```

Más o menos lo mismo que los otros dos, pero esta vez cancela el evento de muerte del jugador, evitando que pueda morir mientras aún no ha iniciado sesión

Manejo del evento BlockBreakEvent

```

121 @EventHandler
122 public void bloqueDestruido(BlockBreakEvent event) {
123     Player jugador = event.getPlayer();
124     try (PreparedStatement statement = conexion.prepareStatement(
125         "UPDATE minecraft SET bloques_rotos = bloques_rotos + 1 WHERE nombre_juego = ?")) {
126         statement.setString(1, jugador.getName());
127         statement.executeUpdate();
128     } catch (SQLException e) {
129         getLogger().severe("Error al incrementar el número de bloques rotos: " + e.getMessage());
130     }
131 }

```

Esta sección se dedica a actualizar la BBDD en la tabla minecraft para contar cuantos bloques destruye el jugador

Player jugador = event.getPlayer();

Obtiene el jugador que activa el evento

```

try (PreparedStatement statement = conexion.prepareStatement(
    "UPDATE minecraft SET bloques_rotos = bloques_rotos + 1 WHERE
nombre_juego = ?")) {
    statement.setString(1, jugador.getName());
    statement.executeUpdate();
} catch (SQLException e) {
    getLogger().severe("Error al incrementar el número de bloques rotos: " +
e.getMessage());
}

```

Esta sección intentará actualizar en la BBDD la tabla minecraft sumando 1 a los bloques_rotos donde el nombre_juego es el jugador que ha activado el evento, en caso de excepción, se guardará en el log del servidor

Método onCommand

```

GNU nano 7.2                                                                    Verysimples
public boolean onCommand(CommandSender sender, Command command, String label, String[] args) {
    Player jugador = (Player) sender;
    UUID uuidJugador = jugador.getUniqueId();
    if (!sesionesJugador.get(uuidJugador)) {
        // El jugador aún no ha iniciado sesión
        if (args.length != 1) {
            jugador.sendMessage("Uso: /login <contraseña>");
            return false;
        }
        String contrasena = args[0];
        if (!comprobarContrasenia(jugador, contrasena)) {
            jugador.kickPlayer("Contraseña incorrecta.");
            return false;
        }
        // El jugador ha iniciado sesión correctamente
        sesionesJugador.put(uuidJugador, true);
        jugador.sendMessage("¡Has iniciado sesión correctamente!");
        // Actualizar base de datos para marcar jugador como en línea
        try (PreparedStatement statement = conexion.prepareStatement(
            "UPDATE minecraft SET enlinea = 1 WHERE nombre_juego = ?") {
            statement.setString(1, jugador.getName());
            statement.executeUpdate();
        } catch (SQLException e) {
            getLogger().severe("Error al establecer que el jugador está en línea: " + e.getMessage());
        }
        try (PreparedStatement statement = conexion.prepareStatement(
            "UPDATE minecraft SET ultima_conex = ? WHERE nombre_juego = ?") {
            statement.setTimestamp(1, new Timestamp(System.currentTimeMillis()));
            statement.setString(2, jugador.getName());
            statement.executeUpdate();
        } catch (SQLException e) {
            getLogger().severe("Error al establecer la última conexión: " + e.getMessage());
        }
    } else {
        jugador.sendMessage("Ya has iniciado sesión.");
    }
    return true;
}

```

Maneja el comando /login ejecutado por el jugador, comprueba la contraseña y también maneja las posibles excepciones y las registra en el log

public boolean onCommand(CommandSender sender, Command command, String label, String[] args) {

public boolean onCommand: Define un método que devuelve un booleano.

CommandSender sender: El objeto que envió el comando. Puede ser un jugador u otro tipo de emisor de comandos.

Command command: El comando que se ejecuta.

String label: La etiqueta del comando.

String[] args: Los argumentos del comando.

Player jugador = (Player) sender;

Convierte el sender a un objeto Player, asumiendo que el comando solo puede ser ejecutado por un jugador.

UUID uuidJugador = jugador.getUniqueId();

Obtiene el UUID único del jugador.

```
if (!sesionesJugador.get(uuidJugador)) {
```

Primero se verifica que la sesión no haya sido creada

```
if (args.length != 1) {
```

```
    jugador.sendMessage("Uso: /login <contraseña>");
```

```
    return false;
```

```
}
```

Verifica que el argumento si es distinto a uno, significa que o no ha enviado la contraseña o que ha enviado como mínimo 2, así que se entiende que el jugador ha cometido un error y se explica como se usa el comando

```
String contraseña = args[0];
```

Guardamos el primer argumento del comando en un string contraseña

```
if (!comprobarContrasenia(jugador, contraseña)) {
```

```
    jugador.kickPlayer("Contraseña incorrecta.");
```

```
    return false;
```

```
}
```

Llama a un método para verificar la contraseña. Si es incorrecta, expulsa al jugador con un mensaje de que su contraseña es incorrecta y devuelve false

```
sesionesJugador.put(uuidJugador, true);
```

Marca al jugador como que ha iniciado sesión

```
jugador.sendMessage("¡Has iniciado sesión correctamente!");
```

Envía al jugador que ha iniciado sesión correctamente

```

try (PreparedStatement statement = conexion.prepareStatement(
    "UPDATE minecraft SET enlinea = 1 WHERE nombre_juego = ?") {
    statement.setString(1, jugador.getName());
    statement.executeUpdate();
} catch (SQLException e) {
    getLogger().severe("Error al establecer que el jugador está en línea: " +
e.getMessage());
}

```

Después de marcar al jugador como que ha iniciado sesión, hará un update en la base de datos para marcar que está en línea, si hay un excepción, se enviará al log

```

try (PreparedStatement statement = conexion.prepareStatement(
    "UPDATE minecraft SET ultima_conex = ? WHERE nombre_juego = ?") {
    statement.setTimestamp(1, new Timestamp(System.currentTimeMillis()));
    statement.setString(2, jugador.getName());
    statement.executeUpdate();
} catch (SQLException e) {
    getLogger().severe("Error al establecer la última conexión: " +
e.getMessage());
}

```

Este bloque de código registra la última vez que un jugador se conectó al servidor haciendo un UPDATE en la base de datos, también, en caso de excepción, se envía al log del servidor

```

} else {
    jugador.sendMessage("Ya has iniciado sesión.");
}

```

Si el jugador vuelve a usar /login <contraseña> después de haber iniciado sesión, recibirá el mensaje de que ya había iniciado sesión

```

return true;
}

```

Devuelve true para indicar que el comando se ejecutó correctamente

Método estaJugadorRegistrado

```

173 @ private boolean estaJugadorRegistrado(Player jugador) { 1usage
174     try (PreparedStatement statement = conexion.prepareStatement("SELECT * FROM usuarios WHERE nombre_juego = ?")) {
175         statement.setString(1, jugador.getName());
176         try (ResultSet resultSet = statement.executeQuery()) {
177             return resultSet.next();
178         }
179     } catch (SQLException e) {
180         getLogger().severe(msg, "Error al comprobar si el jugador está registrado: " + e.getMessage());
181         return false;
182     }
183 }
184

```

Este método privado comprueba de que el jugador exista en la base de datos y por ende esté registrado

private boolean estaJugadorRegistrado(Player jugador) {

Método privado que se le indica un objeto (player) como parámetro (jugador) y devuelve true or false

try (PreparedStatement statement = conexion.prepareStatement("SELECT * FROM usuarios WHERE nombre_juego = ?")) {

En esta línea se prepara la consulta SQL que seleccionará todos los datos de la tabla usuarios donde el nombre_juego será

statement.setString(1, jugador.getName());

El nombre del jugador que hemos pasado antes como parámetro

try (ResultSet resultSet = statement.executeQuery()) {
return resultSet.next();
}

Este fragmento ejecuta la consulta SQL y verifica si el resultado contiene al menos una fila para mirar si un jugador está registrado en la base de datos

} catch (SQLException e) {

getLogger().severe("Error al comprobar si el jugador está registrado: " + e.getMessage());

return false;

}

}

En caso de excepción, se guarda en el log y devuelve false

Método comprobarContrasenia

```

185  @
186  private boolean comprobarContrasenia(Player jugador, String contrasena) { 1usage
187      try (PreparedStatement statement = conexion.prepareStatement("SELECT contrasena FROM usuarios WHERE nombre_juego = ?")) {
188          statement.setString(1, jugador.getName());
189          try (ResultSet resultSet = statement.executeQuery()) {
190              if (resultSet.next()) {
191                  String contrasenaAlmacenada = resultSet.getString("contrasena");
192                  return contrasena.equals(contrasenaAlmacenada);
193              }
194          } catch (SQLException e) {
195              getLogger().severe(msg: "Error al comprobar la contraseña del jugador: " + e.getMessage());
196          }
197          return false;
198      }

```

Este método que es llamado en el método comando, para verificar si la contraseña proporcionada por el jugador coincide con la almacenada en la base de datos.

private boolean comprobarContrasenia(Player jugador, String contrasena) {

Método privado que devuelve true o false. Toma como parámetros un objeto Player, que es el jugador que ejecutará el comando y un string contrasena, que es el parámetro 0 del comando

try (PreparedStatement statement = conexion.prepareStatement("SELECT contrasena FROM usuarios WHERE nombre_juego = ?")) {
statement.setString(1, jugador.getName());

Prepara para la consulta SQL que seleccionará la contraseña de la tabla usuarios donde la columna nombre_juego coincide con el nombre del jugador.

try (ResultSet resultSet = statement.executeQuery()) {

Ejecuta la consulta y guarda el resultado en un ResultSet

if (resultSet.next()) {
String contrasenaAlmacenada = resultSet.getString("contrasena");
return contrasena.equals(contrasenaAlmacenada);
}
}

Con resultSet.next() verifica si hay al menos una fila en el resultado de la consulta y si hay al menos una fila, guarda la contraseña en un string llamado contrasenaAlmacenada

return contrasena.equals(contrasenaAlmacenada): Compara la contraseña proporcionada (contrasena) con la contraseña de la BBDD (contrasenaAlmacenada). Devuelve true si son iguales, de lo contrario false.

```

    } catch (SQLException e) {
        getLogger().severe("Error al comprobar la contraseña del jugador: " +
e.getMessage());
    }

```

En caso de excepción SQL, se guardará en el log del servidor

```

    return false;
}

```

Si la contraseña no coincide o hay excepción SQL, se devuelve false

Manejo del evento PlayerDeathEvent

```

199
200 @EventHandler
201 public void muerteJugador(PlayerDeathEvent event) {
202     Player jugador = event.getEntity();
203     try (PreparedStatement statement = conexion.prepareStatement(
204         s: "UPDATE minecraft SET muertes = muertes + 1 WHERE nombre_juego = ?") {
205         statement.setString(1, jugador.getName());
206         statement.executeUpdate();
207     } catch (SQLException e) {
208         getLogger().severe(msg: "Error al incrementar el número de muertes del jugador: " + e.getMessage());
209     }
210 }

```

Método público que se ejecuta cuando ocurre un evento de tipo PlayerDeathEvent y no devuelve nada ya que es un public void

```

Player jugador = event.getEntity();

```

Obtiene el jugador que muere

```

try (PreparedStatement statement = conexion.prepareStatement(
    "UPDATE minecraft SET muertes = muertes + 1 WHERE nombre_juego
= ?") {
    statement.setString(1, jugador.getName());
    statement.executeUpdate();
}

```

Intentará ejecutar la sentencia SQL que sumará +1 al número que ya tenga la BBDD en la columna muertes de la tabla minecraft donde el nombre_juego es el nombre del jugador que ha muerto

```

catch (SQLException e) {
    getLogger().severe("Error al incrementar el número de muertes del
jugador: " + e.getMessage());
}
}

```

En caso de excepción SQL, se registrará en el log del servidor

Manejo del evento BlockPlaceEvent

```

213
214 @EventHandler
215 public void bloquePuesto(BlockPlaceEvent event) {
216     Player jugador = event.getPlayer();
217     try (PreparedStatement statement = conexion.prepareStatement(
218         "UPDATE minecraft SET bloques_colocados = bloques_colocados + 1 WHERE nombre_juego = ?") {
219         statement.setString(1, jugador.getName());
220         statement.executeUpdate();
221     } catch (SQLException e) {
222         getLogger().severe("Error al incrementar el número de bloques colocados: " + e.getMessage());
223     }
224 }

```

Este bloque de código actualiza el número de bloques que ha colocado el jugador cada vez que ocurra el evento

Player jugador = event.getPlayer();

Obtiene el jugador que colocó el bloque

```

try (PreparedStatement statement = conexion.prepareStatement(
    "UPDATE minecraft SET bloques_colocados = bloques_colocados + 1
WHERE nombre_juego = ?")) {
    statement.setString(1, jugador.getName());
    statement.executeUpdate();
}

```

Esta sección intentará hacer el UPDATE en la BBDD al jugador en concreto que ejecutó el evento

```

catch (SQLException e) {
    getLogger().severe("Error al incrementar el número de bloques colocados:
    " + e.getMessage());
}

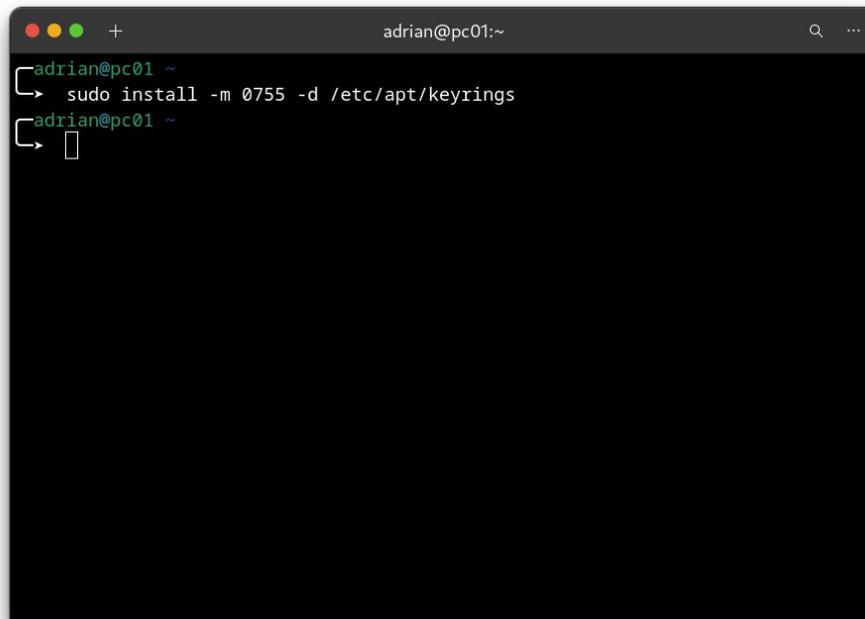
```

} → Cierre de la clase principal

Si ocurre una excepción SQL, se registrará en el log

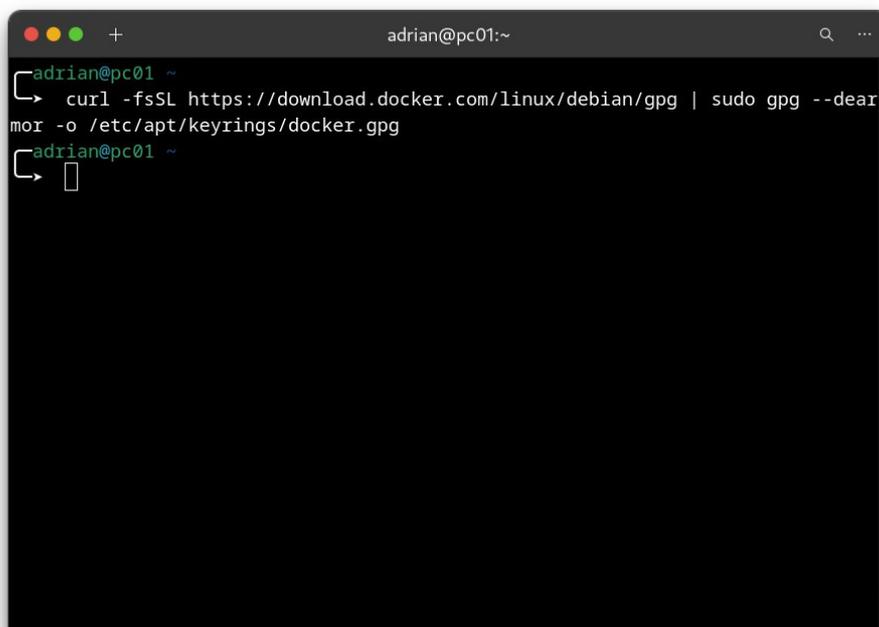
4. Docker

Instalación de docker



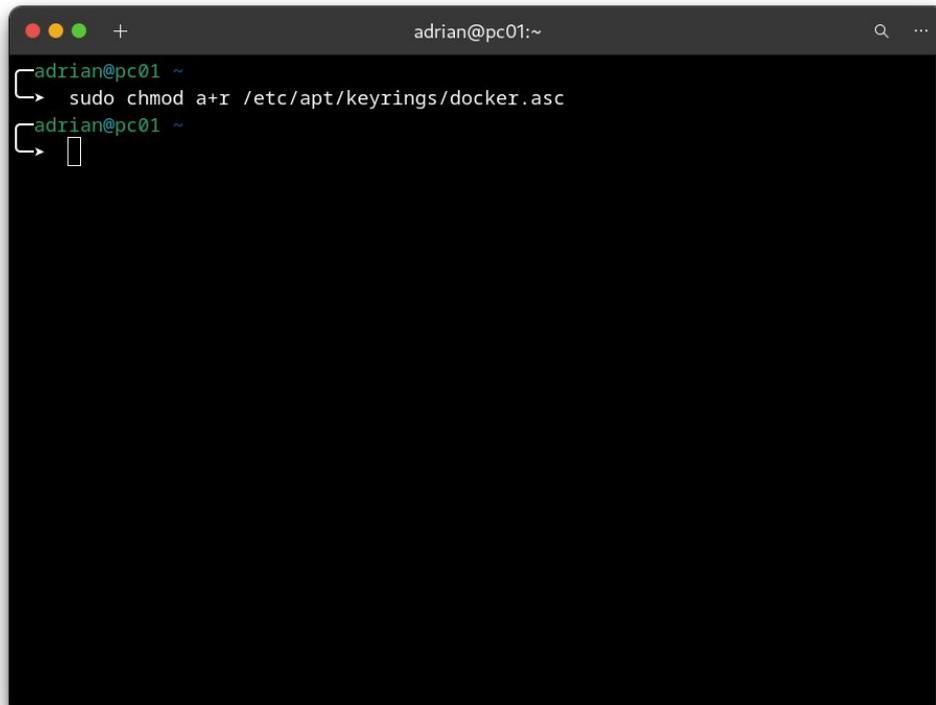
```
adrian@pc01:~  
└─$ sudo install -m 0755 -d /etc/apt/keyrings  
adrian@pc01:~  
└─$
```

Creamos el directorio keyrings con permisos 755 en /etc/apt



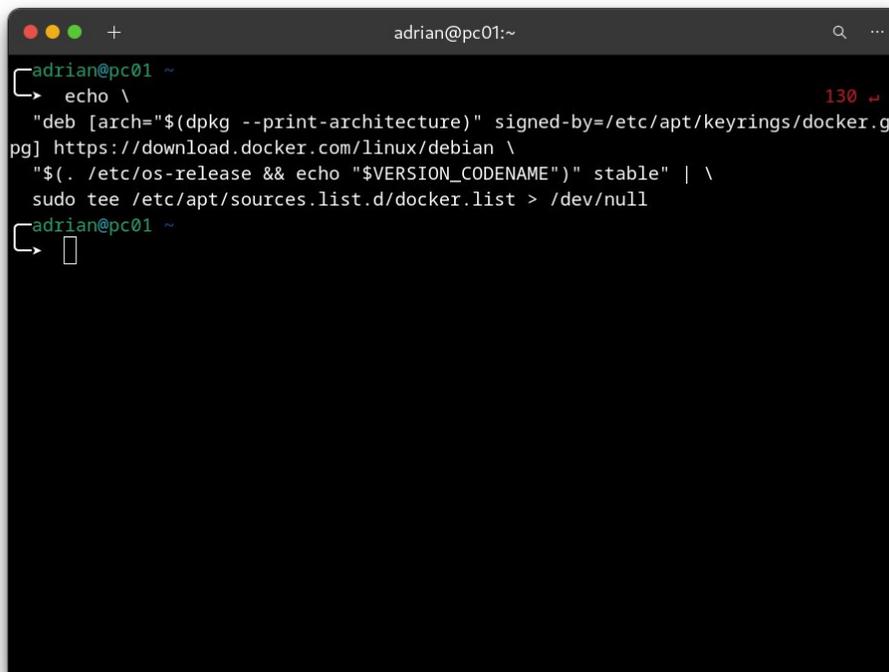
```
adrian@pc01:~  
└─$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg  
adrian@pc01:~  
└─$
```

Ahora necesitamos descargar y guardar la clave GPG desde el sitio web oficial de Docker



```
adrian@pc01:~  
└─$ sudo chmod a+r /etc/apt/keyrings/docker.asc  
adrian@pc01:~  
└─$
```

Asignamos a todo el mundo permisos de lectura a la clave GPG



```
adrian@pc01:~  
└─$ echo \  
"deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.g  
pg] https://download.docker.com/linux/debian \  
"$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \  
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null  
adrian@pc01:~  
└─$
```

Agregamos el repositorio al source list

```

adrian@pc01 ~
└─> sudo apt update
Obj:1 http://repo.mysql.com/apt/debian bookworm InRelease
Obj:3 https://dl.winehq.org/wine-builds/debian bookworm InRelease
Obj:4 https://download.docker.com/linux/debian bookworm InRelease
Obj:5 https://linux.teamviewer.com/deb stable InRelease
Obj:6 http://ftp.es.debian.org/debian bookworm InRelease
Obj:7 http://ftp.es.debian.org/debian bookworm-updates InRelease
Obj:8 http://ppa.launchpad.net/polychromatic/stable/ubuntu focal InRelease
Des:2 https://packages.microsoft.com/repos/code stable InRelease [3.590 B]
Obj:9 https://packages.mozilla.org/apt mozilla InRelease
Descargados 3.590 B en 1s (5.298 B/s)
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se puede actualizar 1 paquete. Ejecute «apt list --upgradable» para verlo.
adrian@pc01 ~
└─> █

```

Actualizamos la lista de paquetes

```

adrian@pc01 ~
└─> sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
docker-ce ya está en su versión más reciente (5:26.1.3-1~debian.12~bookworm).
docker-ce-cli ya está en su versión más reciente (5:26.1.3-1~debian.12~bookworm).
docker-buildx-plugin ya está en su versión más reciente (0.14.0-1~debian.12~bookworm).
docker-compose-plugin ya está en su versión más reciente (2.27.0-1~debian.12~bookworm).
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
 libestr0 libfastjson4 liblightdm-gobject-1-0 liblognorm5
 lightdm-gtk-greeter
Utilice «sudo apt autoremove» para eliminarlos.
Se actualizarán los siguientes paquetes:
 containerd.io
1 actualizados, 0 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 30,0 MB de archivos.
Se utilizarán 495 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s

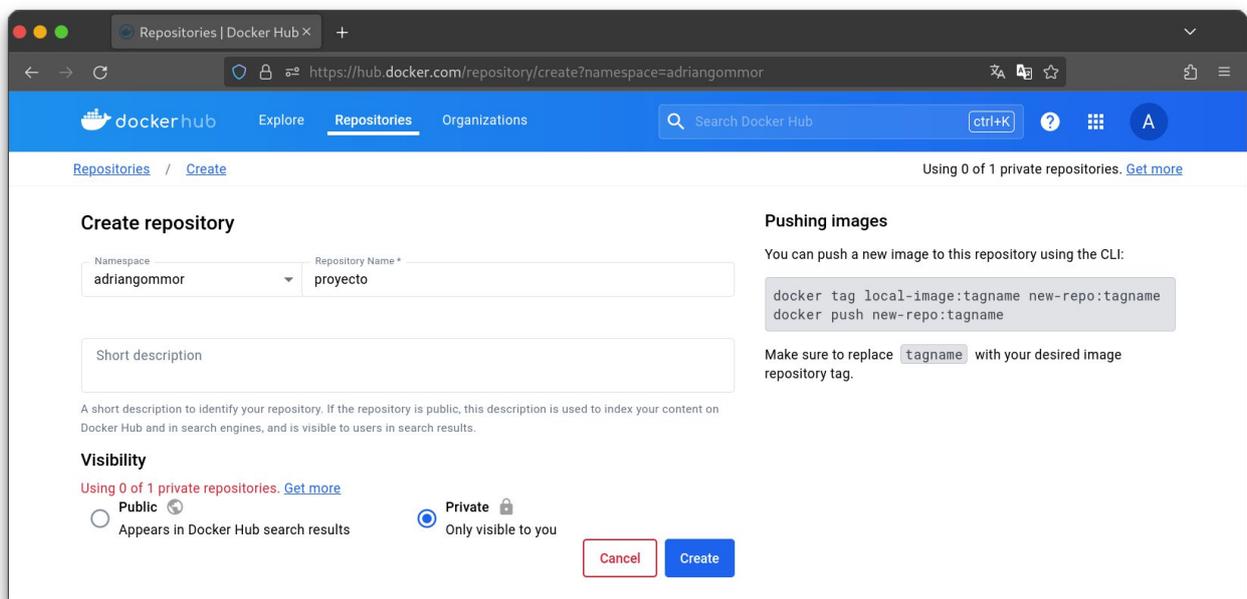
```

Instalamos los paquetes necesarios


```
adrian@pc01 ~
└─$ docker login --username=adriangommor
Password:
WARNING! Your password will be stored unencrypted in /home/adrian/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
adrian@pc01 ~
└─$
```

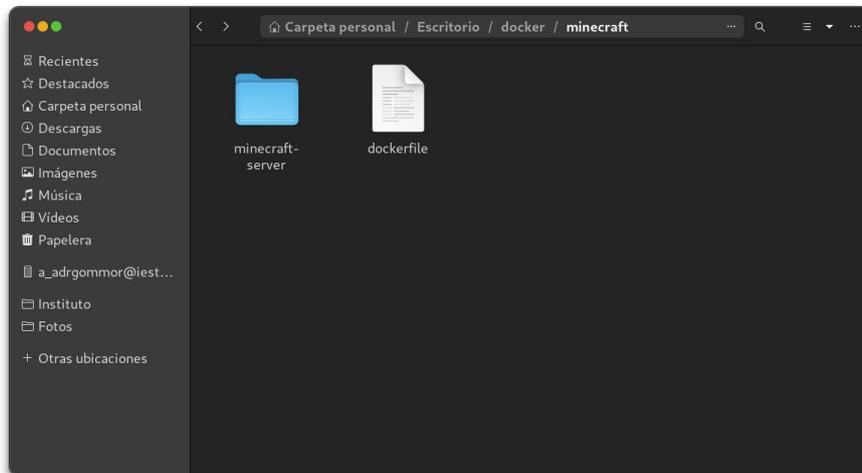
Para en un futuro poder subir nuestras imágenes a un repositorio, debemos iniciar sesión con las credenciales de docker hub (<https://hub.docker.com/>)



En docker hub, creamos un repositorio privado, donde subiremos las imágenes

Imágenes personalizadas

Minecraft



En el directorio de trabajo, donde tenemos los archivos del servidor de Minecraft, creamos un archivo llamado “dockerfile”

```
FROM ubuntu:latest

RUN apt update && apt upgrade -y

RUN apt install -y openjdk-17-jdk

RUN mkdir /minecraft

COPY ./minecraft-server /minecraft

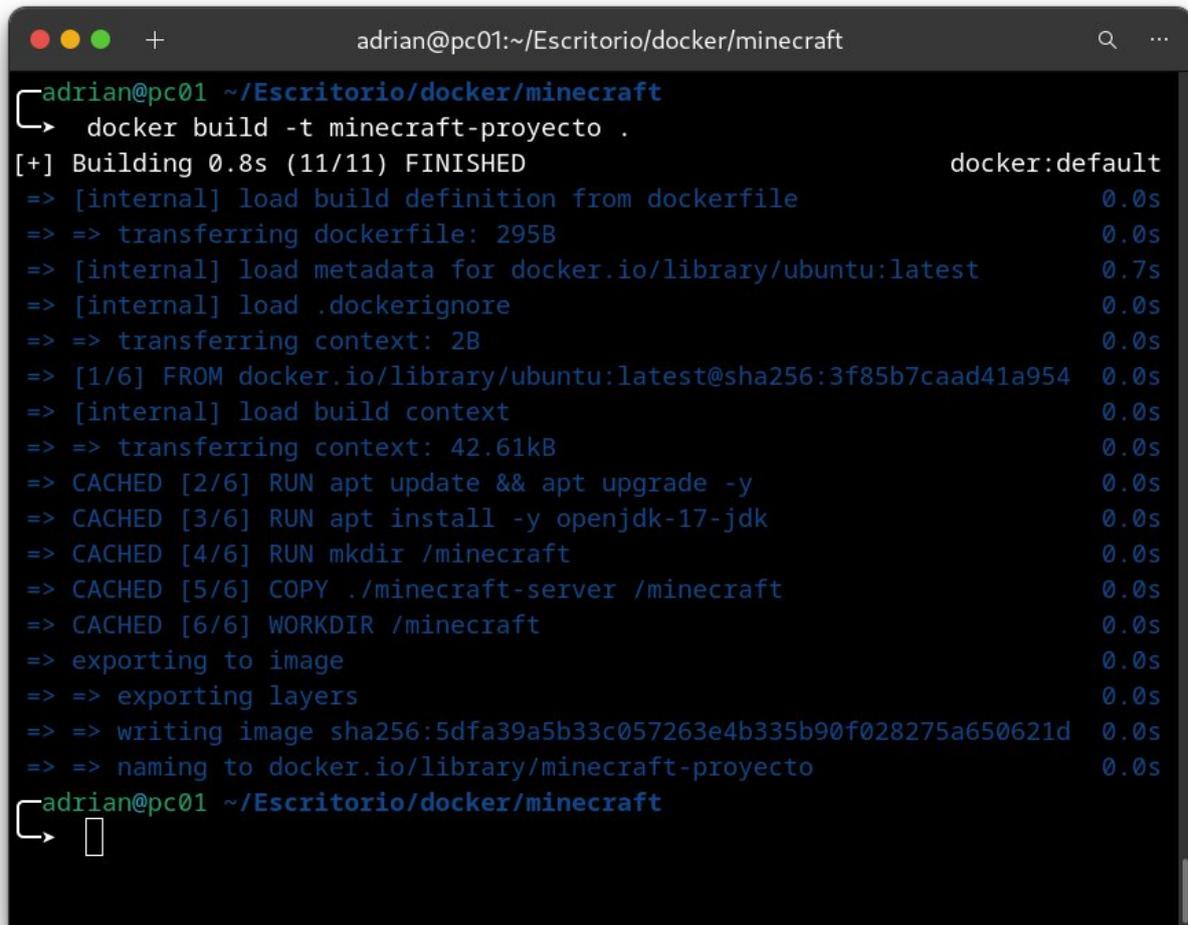
WORKDIR /minecraft

EXPOSE 25565

CMD ["java", "-Xms1G", "-Xmx2G", "-jar", "paper-1.20.4-477.jar", "noqui"]
```

Establecemos que vamos a partir de la imagen ubuntu:latest (la última LTS) vamos a copiar nuestra carpeta que contiene el servidor a /minecraft del contenedor (previamente hay que aceptar el eula), establecemos el directorio de trabajo en /minecraft y establecemos que se ejecute automáticamente paper-1.20.4-477.jar usando java, mínimo usará 1GB de RAM y como máximo

usará 2GB de RAM, además se lanzará con el parametro nogui para que no se ejecute la interfaz gráfica y solo se ejecute la versión de terminal

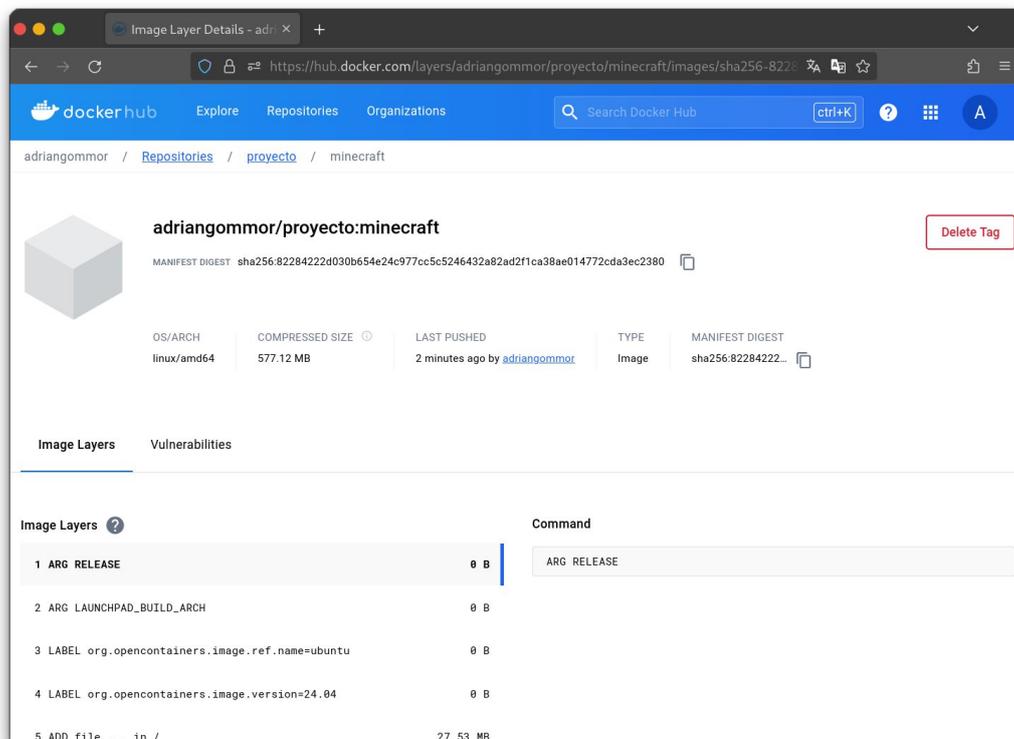
A terminal window titled 'adrian@pc01:~/Escritorio/docker/minecraft' showing the output of the 'docker build -t minecraft-proyecto .' command. The output indicates a successful build of a Docker image. The process starts with 'Building 0.8s (11/11) FINISHED' and lists various steps such as loading build definitions, transferring files, and installing dependencies like 'apt update', 'apt upgrade', and 'openjdk-17-jdk'. The final step is 'exporting to image' and 'naming to docker.io/library/minecraft-proyecto'. The terminal ends with a prompt 'adrian@pc01 ~/Escritorio/docker/minecraft' and a cursor.

```
adrian@pc01 ~/Escritorio/docker/minecraft
└─> docker build -t minecraft-proyecto .
[+] Building 0.8s (11/11) FINISHED                                docker:default
=> [internal] load build definition from dockerfile                0.0s
=> => transferring dockerfile: 295B                               0.0s
=> [internal] load metadata for docker.io/library/ubuntu:latest   0.7s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                     0.0s
=> [1/6] FROM docker.io/library/ubuntu:latest@sha256:3f85b7caad41a954 0.0s
=> [internal] load build context                                  0.0s
=> => transferring context: 42.61kB                               0.0s
=> CACHED [2/6] RUN apt update && apt upgrade -y                  0.0s
=> CACHED [3/6] RUN apt install -y openjdk-17-jdk                 0.0s
=> CACHED [4/6] RUN mkdir /minecraft                             0.0s
=> CACHED [5/6] COPY ./minecraft-server /minecraft               0.0s
=> CACHED [6/6] WORKDIR /minecraft                               0.0s
=> exporting to image                                             0.0s
=> => exporting layers                                           0.0s
=> => writing image sha256:5dfa39a5b33c057263e4b335b90f028275a650621d 0.0s
=> => naming to docker.io/library/minecraft-proyecto             0.0s
adrian@pc01 ~/Escritorio/docker/minecraft
└─> █
```

Creamos la imagen con **docker build -t minecraft-proyecto .**

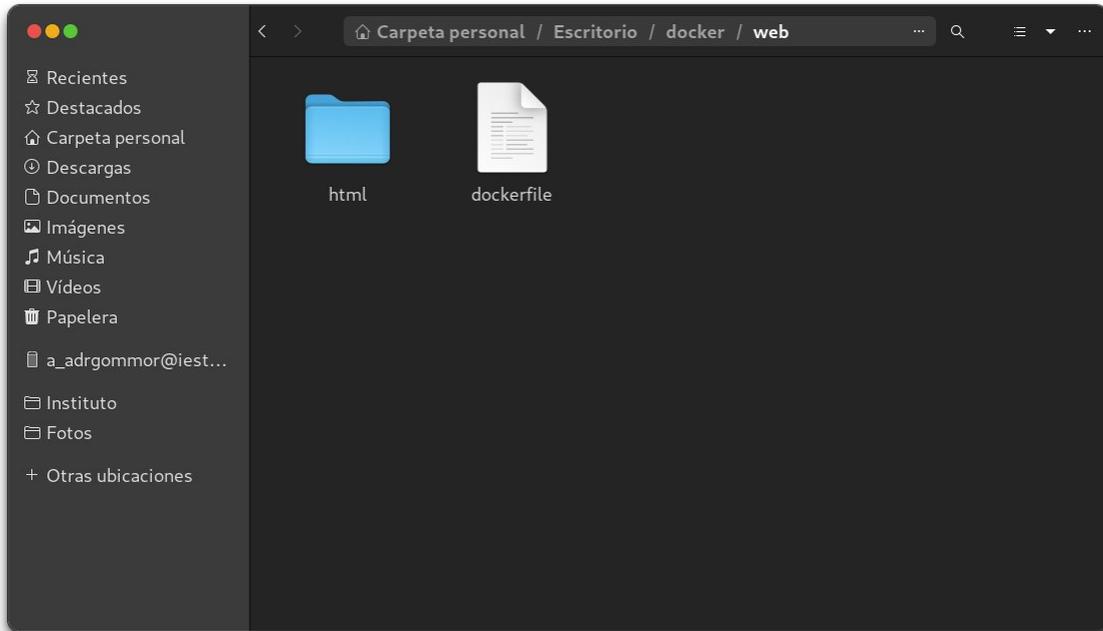
```
adrian@pc01 ~
└─> docker tag minecraft-proyecto adriangommor/proyecto:minecraft
adrian@pc01 ~
└─> docker push adriangommor/proyecto:minecraft
The push refers to repository [docker.io/adriangommor/proyecto]
5f70bf18a086: Pushed
dba532da3a0e: Pushed
9cf104663da0: Pushed
5e08b690e734: Pushed
16c248650537: Pushed
80098e3d304c: Mounted from library/ubuntu
minecraft: digest: sha256:82284222d030b654e24c977cc5c5246432a82ad2f1ca38ae014772cda3ec2380 size: 1579
adrian@pc01 ~
```

Etiquetamos la imagen con **docker tag minecraft-proyecto adriangommor/proyecto:minecraft** y subimos la imagen a nuestro repositorio privado con **docker push adriangommor/proyecto:minecraft**



Y podemos ver la imagen subida correctamente

Web



Ahora cambiamos de directorio de trabajo, vamos a crear la imagen para nuestra página web, para ello, en el mismo directorio, dejamos nuestra web y hacemos un dockerfile

```
dockerfile
~/Escritorio/Proyecto - Adrian Gomez Morales/Docker/Web

FROM php:8.2-apache

RUN apt-get update && apt-get upgrade -y

RUN docker-php-ext-install mysqli && apt-get install -y libpng-dev && docker-php-ext-install gd

COPY html/ /var/www/html

RUN chown -R www-data:www-data /var/www/html

EXPOSE 80

EXPOSE 443
```

Añadimos las dependencias de mysqli para la conexión a MySQL y gd que es necesario para CkEditor

```

adrian@pc01 ~/Escritorio/docker/web
└─$ docker build -t adriangommor/proyecto:web .
[+] Building 15.6s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring Dockerfile: 185B
=> [internal] load metadata for docker.io/library/php:8.2-apache
=> [internal] load .dockerignore
=> => transferring context: 2B
=> CACHED [1/4] FROM docker.io/library/php:8.2-apache@sha256:7559e3d22c0f16b7ebcf5ef95ce340288d9e94ef20d88957755de7b186962c6a
=> [internal] load build context
=> => transferring context: 781.50kB
=> [2/4] RUN apt-get update && apt-get upgrade -y
=> [3/4] RUN docker-php-ext-install mysql
=> [4/4] COPY ./html /var/www/html
=> exporting to image
=> exporting layers
=> writing image sha256:5467389b6503ccb8cc39c63faf1fe2c500bb8af20772d8a5179401cf2da9040
=> naming to docker.io/adriangommor/proyecto:web
adrian@pc01 ~/Escritorio/docker/web

```

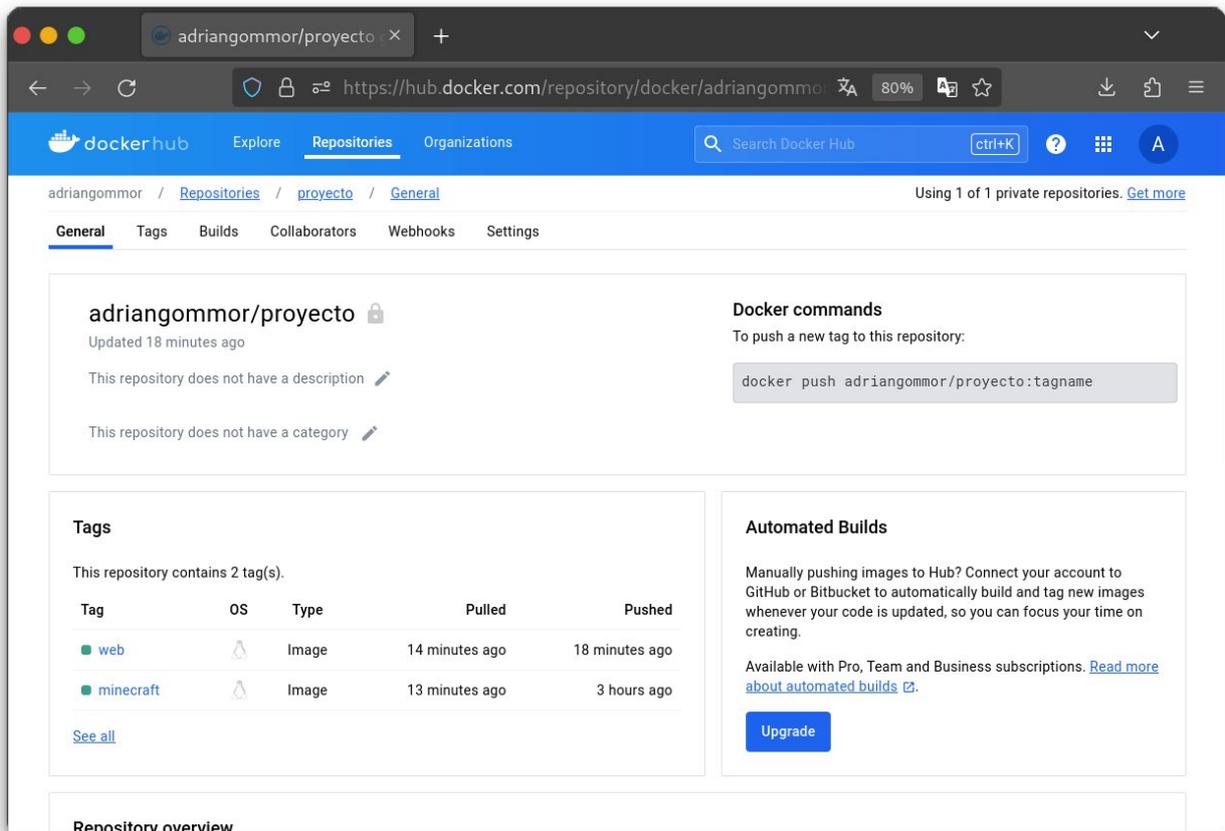
Creamos la imagen con **docker build -t adriangommor/proyecto:web .** además ya se crea etiquetada

```

adrian@pc01 ~/Escritorio/docker/web
└─$ docker push adriangommor/proyecto:web
The push refers to repository [docker.io/adriangommor/proyecto]
08093cb95353: Pushed
8d823ed36be2: Pushed
641c00c9f2a5: Pushed
fe8c7f6c4a88: Mounted from library/php
55a3199bfde5: Mounted from library/php
8e73026e7d55: Mounted from library/php
e593f731d682: Mounted from library/php
45ba90d36265: Mounted from library/php
0702b1aba921: Mounted from library/php
7ea0df9b18f9: Mounted from library/php
11b8a7bb5a34: Mounted from library/php
62345f7786c0: Mounted from library/php
33c44db12c6c: Mounted from library/php
b354ecd4789a: Mounted from library/php
fb5221356fa2: Mounted from library/php
5d4427064ecc: Mounted from library/php
web: digest: sha256:16b4c9193f5024ef6d293756dc38b390d1a5090f23d48dd915ff287a1a5e6e1f size: 3669
adrian@pc01 ~/Escritorio/docker/web

```

Subimos la imagen a nuestro repositorio privado de docker hub



Así quedaría nuestro repositorio con las 2 imágenes


```

adrian@pc01 ~
└─> sudo apt install kubernetes-client
[sudo] contraseña para adrian:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no
son necesarios.
  libestr0 libfastjson4 liblightdm-gobject-1-0 liblognorm5 lightdm-gtk-greeter
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes NUEVOS:
  kubernetes-client
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 8.195 kB de archivos.
Se utilizarán 38,4 MB de espacio de disco adicional después de esta operación.
Des:1 http://ftp.es.debian.org/debian bookworm/main amd64 kubernetes-client amd6
4 1.20.5+really1.20.2-1.1 [8.195 kB]
Descargados 8.195 kB en 1s (7.044 kB/s)
Seleccionando el paquete kubernetes-client previamente no seleccionado.
(Leyendo la base de datos ... 334881 ficheros o directorios instalados actualmen
te.)
Preparando para desempaquetar .../kubernetes-client_1.20.5+really1.20.2-1.1_amd6
4.deb ...

```

Ahora instalamos kubectl desde los repositorios de Debian

```

adrian@pc01 ~/Escritorio/kubernetes
└─> minikube start --memory 8192 --cpus 4 --driver=docker
🌻 minikube v1.33.1 en Debian 12.5
👉 Using the docker driver based on existing profile
👉 Starting "minikube" primary control-plane node in "minikube" cluster
📡 Pulling base image v0.0.44 ...
🔄 Restarting existing docker container for "minikube" ...

🚨 Docker is nearly out of disk space, which may cause deployments to fail! (95% of capacity). You can pass '--force' to skip this check.
💡 Suggestion:

Try one or more of the following to free up space on the device:

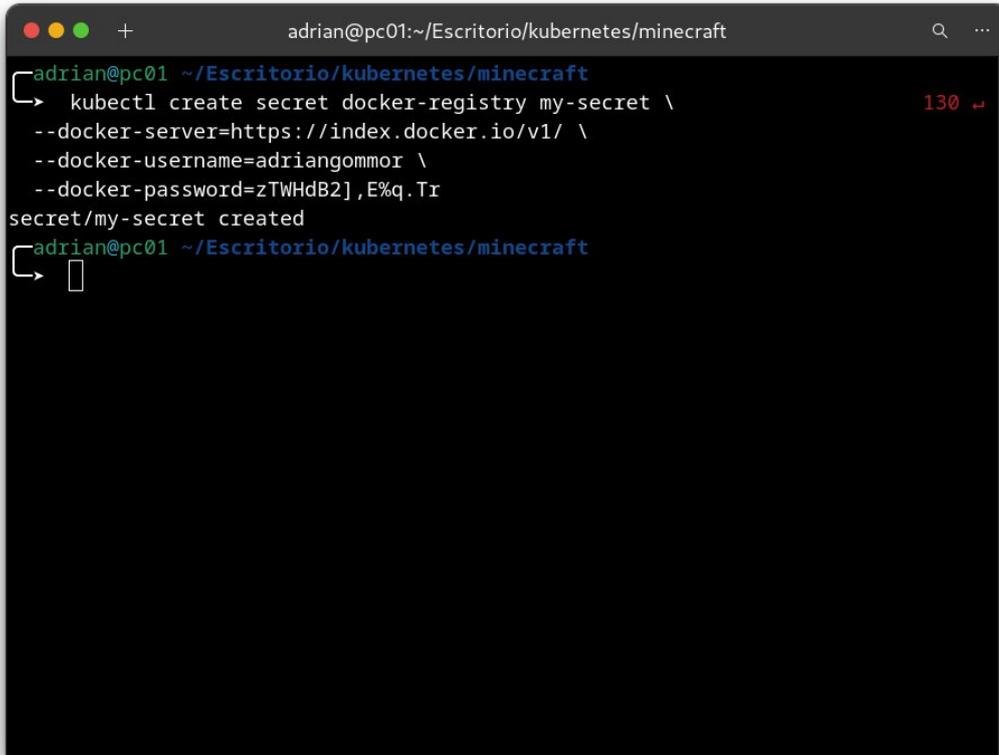
1. Run "docker system prune" to remove unused Docker data (optionally with "-a")
2. Increase the storage allocated to Docker for Desktop by clicking on:
   Docker icon > Preferences > Resources > Disk Image Size
3. Run "minikube ssh -- docker system prune" if using the Docker container runtime
🔗 Related issue: https://github.com/kubernetes/minikube/issues/9024

📡 Preparando Kubernetes v1.30.0 en Docker 26.1.1...
🔍 Verifying Kubernetes components...
  • Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟 Complementos habilitados: storage-provisioner, default-storageclass
🎉 Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
adrian@pc01 ~/Escritorio/kubernetes
└─>

```

Iniciamos Minikube usando como driver docker y asignando 8GB de memoria RAM y 4 núcleos de mi procesador para que tenga los suficientes recursos

Secret para acceder al repositorio privado de Docker Hub



```
adrian@pc01:~/Escritorio/kubernetes/minecraft
└─$ kubectl create secret docker-registry my-secret \
  --docker-server=https://index.docker.io/v1/ \
  --docker-username=adriangommor \
  --docker-password=zTWHdB2],E%q.Tr
secret/my-secret created
adrian@pc01:~/Escritorio/kubernetes/minecraft
└─$
```

Creamos el secret

minecraft en Kubernetes

Este yaml crea un despliegue de un servidor Minecraft utilizando una imagen propia de Docker Hub. Define un PersistentVolumeClaim de 3GiB para almacenar los datos del mundo de Minecraft, para que sean persistentes. Monta el volumen persistente en /minecraft/world. Además, expone el servicio a través de un NodePort en el puerto 30000, permitiendo el acceso externo al servidor Minecraft. Se utiliza un secret para el pull de la imagen

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: pvc-minecraft

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 3Gi

apiVersion: apps/v1

kind: Deployment

metadata:

name: deployment-minecraft

labels:

app: minecraft

service: game

spec:

replicas: 1

selector:

matchLabels:

app: minecraft

service: game

template:

metadata:

labels:

app: minecraft

service: game

spec:

volumes:

- name: volumen-minecraft

persistentVolumeClaim:

claimName: pvc-minecraft

containers:

- name: contenedor-minecraft

image: adriangommor/proyecto:minecraft

volumeMounts:

- mountPath: "/minecraft/world"

name: volumen-minecraft

imagePullSecrets:

- name: my-secret

apiVersion: v1

kind: Service

metadata:

name: service-minecraft

labels:

app: minecraft

service: game

spec:

type: NodePort

selector:

app: minecraft

service: game

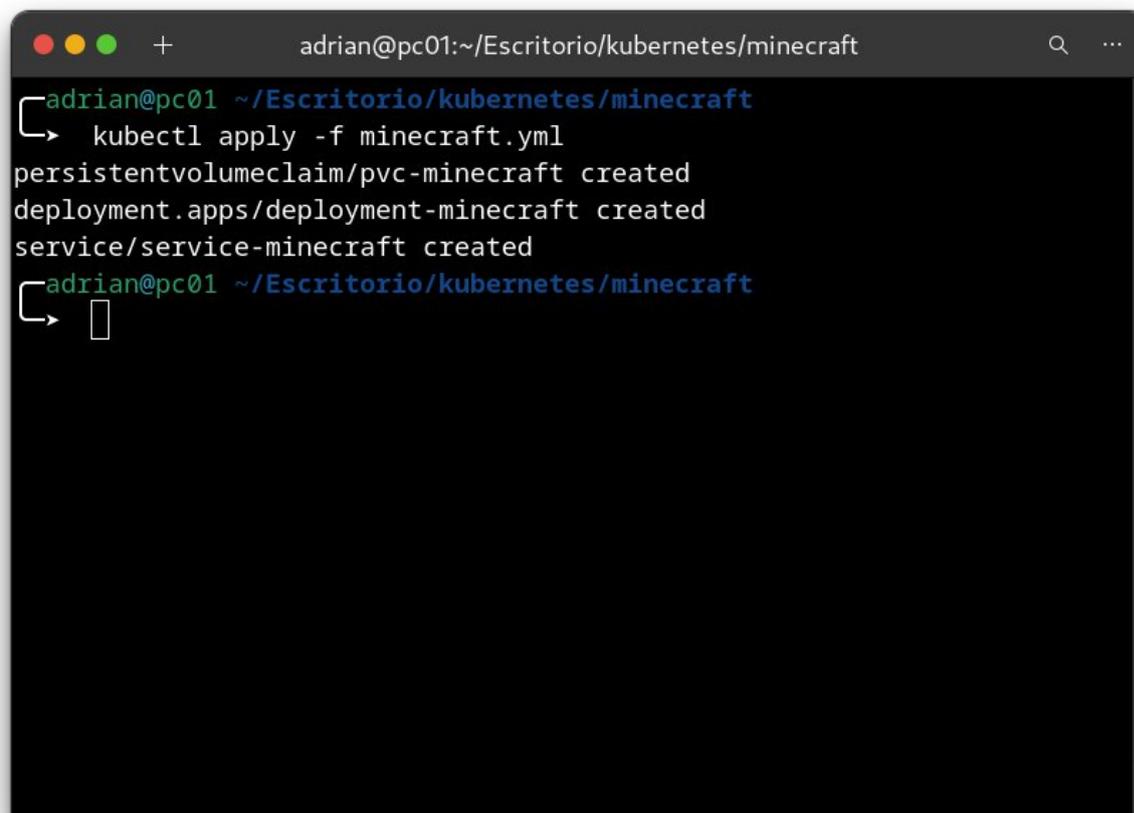
ports:

- name: minecraft

port: 25565

targetPort: 25565

nodePort: 30000



```
adrian@pc01:~/Escritorio/kubernetes/minecraft
└─┬─┘ kubectl apply -f minecraft.yml
    persistentvolumeclaim/pvc-minecraft created
    deployment.apps/deployment-minecraft created
    service/service-minecraft created
└─┬─┘
    └─┬─┘
```

Aplicamos el yaml con “kubectl apply -f minecraft.yml”

Base de datos MySQL en Kubernetes

```
adrian@pc01 ~/Escritorio/kubernetes/mysql
└─> kubectl create secret generic mysql-pass --from-literal=password='H4-D14s*fV;.4_P35k4R'
```

Creamos el secret para nuestra contraseña de la BBDD

```
adrian@pc01 ~/Escritorio/kubernetes/mysql
└─> kubectl create secret generic mysql-root-pass --from-literal=password=root
secret/mysql-root-pass created
adrian@pc01 ~/Escritorio/kubernetes/mysql
```

Creamos el secret para la contraseña de root

Archivo yaml

Este yaml configura el despliegue de una BBDD MySQL usando la imagen de Docker Hub. Define un volumen de 1GiB para almacenar los datos de MySQL, para que los datos sean persistentes. Además, expone el servicio MySQL a través de un Service ClusterIP para que pueda ser accedido por otros pods en el clúster usando la resolución DNS

apiVersion: apps/v1

kind: Deployment

metadata:

name: mysql

labels:

app: mysql

spec:

replicas: 1

selector:

matchLabels:

app: mysql

template:

metadata:

labels:

app: mysql

spec:

containers:

- name: mysql

image: mysql:latest

env:

- name: MYSQL_ROOT_PASSWORD

valueFrom:

secretKeyRef:

name: mysql-root-pass

key: password

- name: MYSQL_PASSWORD

valueFrom:

secretKeyRef:

name: mysql-pass

key: password

- name: MYSQL_DATABASE

value: proyecto

- name: MYSQL_USER

value: user_proyect

ports:

- containerPort: 3306

volumeMounts:

- name: mysql-persistent-storage

mountPath: /var/lib/mysql

volumes:

- name: mysql-persistent-storage

persistentVolumeClaim:

claimName: mysql-pv-claim

apiVersion: v1

kind: Service

metadata:

name: mysql

spec:

selector:

app: mysql

ports:

- protocol: TCP

port: 3306

targetPort: 3306

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: mysql-pv-claim

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 1Gi

```

adrian@pc01:~/Escritorio/kubernetes/mysql x watch -n 1 ka Puede pegar la imagen desde el portapapeles. mysql -- /bin/bash x adrian@pc01:~/Escritorio/kubernetes/minecraft x
mysql> USE proyecto;
Database changed
mysql>
mysql> CREATE TABLE IF NOT EXISTS usuarios (
-> nombre_juego VARCHAR(255),
-> mail VARCHAR(255),
-> contrasea VARCHAR(255),
-> rol VARCHAR(50),
-> PRIMARY KEY (nombre_juego)
-> );
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> CREATE TABLE IF NOT EXISTS minecraft (
-> nombre_juego VARCHAR(255) NOT NULL,
-> tiempo_jugado INT DEFAULT 0,
-> ultima_conex DATE DEFAULT NULL,
-> muertes INT DEFAULT 0,
-> bloques_rotos INT DEFAULT 0,
-> bloques_colocados INT DEFAULT 0,
-> enlinea SMALLINT NOT NULL DEFAULT 0,
-> PRIMARY KEY (nombre_juego),
-> FOREIGN KEY (nombre_juego) REFERENCES usuarios(nombre_juego)
-> ON DELETE CASCADE
-> ON UPDATE CASCADE
-> );
Query OK, 0 rows affected (0.02 sec)

```

Entramos dentro de una bash del deployment y cargamos las tablas de la base de datos

Web en Kubernetes

Con este yaml creamos un despliegue de mi página usando mi imagen propia de docker hub, creamos dos volúmenes (de 1GiB cada uno), uno para guardar la web (/var/www/html) en sí y otro para guardar /tmp, ya que en /tmp se guardan las sesiones de PHP y si vamos a tener varias replicas, necesitamos que usen las mismas sesiones, además creamos un servicio de tipo NodePort para exponer nuestra web. El comando ejecutado en el contenedor de inicialización copia la web al volumen persistente si está vacío, asegurando que la web no se borre por culpa del volumen.

apiVersion: v1

kind: PersistentVolume

metadata:

name: pv-web

spec:

capacity:

storage: 1Gi

accessModes:

- ReadWriteOnce

hostPath:

path: "/mnt/data"

persistentVolumeReclaimPolicy: Retain

storageClassName: manual

apiVersion: v1

kind: PersistentVolume

metadata:

name: pv-tmp

spec:

capacity:

storage: 1Gi

accessModes:

- ReadWriteOnce

hostPath:

path: "/mnt/tmp"

persistentVolumeReclaimPolicy: Retain

storageClassName: manual

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: pvc-web

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 1Gi

storageClassName: manual

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: pvc-tmp

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 1Gi

storageClassName: manual

apiVersion: apps/v1

kind: Deployment

metadata:

name: deployment-web

labels:

app: web

service: http

spec:

replicas: 2

selector:

matchLabels:

app: web

service: http

template:

metadata:

labels:

app: web

service: http

spec:

volumes:

- name: volumen-web

persistentVolumeClaim:

claimName: pvc-web

- name: volumen-tmp

persistentVolumeClaim:

claimName: pvc-tmp

initContainers:

- name: init-web

image: adriangommor/proyecto:web

command: ['sh', '-c', 'if [-z "\$(ls -A /mnt/html)"]; then cp -r /var/www/html/* /mnt/html; fi']

volumeMounts:

- name: volumen-web

mountPath: /mnt/html

containers:

- name: contenedor-web

image: adriangommor/proyecto:web

ports:

- containerPort: 80

- containerPort: 443

volumeMounts:

- mountPath: "/var/www/html"

name: volumen-web

- mountPath: "/tmp"

name: volumen-tmp

imagePullSecrets:

- name: my-secret

apiVersion: v1

kind: Service

metadata:

- name: service-web

labels:

- app: web

- service: http

spec:

type: NodePort

selector:

- app: web

- service: http

ports:

- name: http

- port: 80

- targetPort: 80

- nodePort: 30001

- name: https

- port: 443

- targetPort: 443

- nodePort: 30002

```
adrian@pc01 ~/Escritorio/Proyecto - Adrian Gomez Morales/Kubernetes/Web
└─┬─▶ kubectl apply -f .
    persistentvolume/pv-web unchanged
    persistentvolume/pv-tmp unchanged
    persistentvolumeclaim/pvc-web unchanged
    persistentvolumeclaim/pvc-tmp unchanged
    deployment.apps/deployment-web unchanged
    service/service-web unchanged
└─┬─▶ adrian@pc01 ~/Escritorio/Proyecto - Adrian Gomez Morales/Kubernetes/Web
    └─▶
```

Aplicamos el archivo yaml

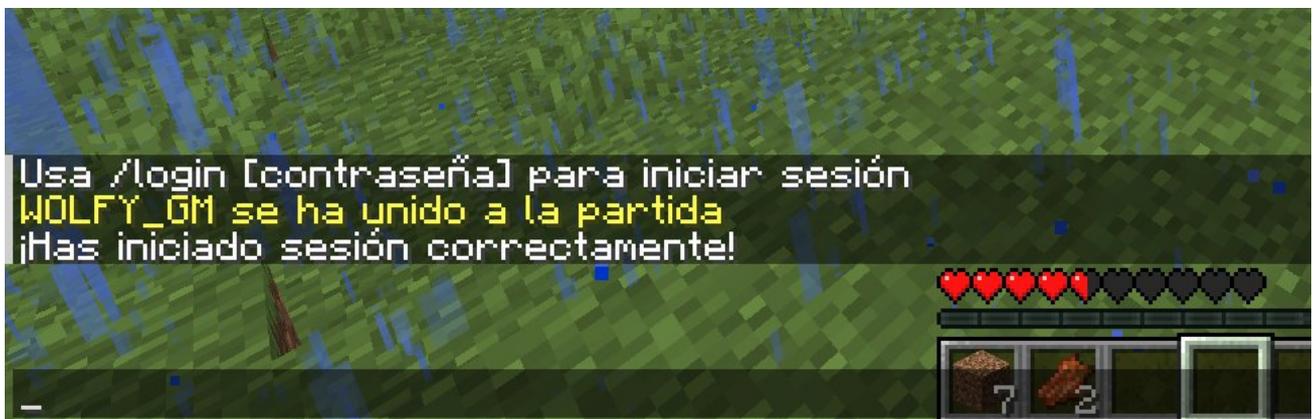
Nota Importante:

Es importante lanzar antes el deployment de MySQL antes que el de Minecraft ya que si no funciona la conexión a la BBDD, el plugin no funcionará de forma correcta, también por primera vez tenemos que cargar la BBDD usando `kubectl exec -it [deployment_mysql] -- /bin/bash` y de la misma forma, en el deployment de la web, hay que establecer /tmp con los permisos 777 y hay que cambiar el propietario de todos los archivos de /var/www/html a www-data de lo contrario no se podrá subir nada a la web

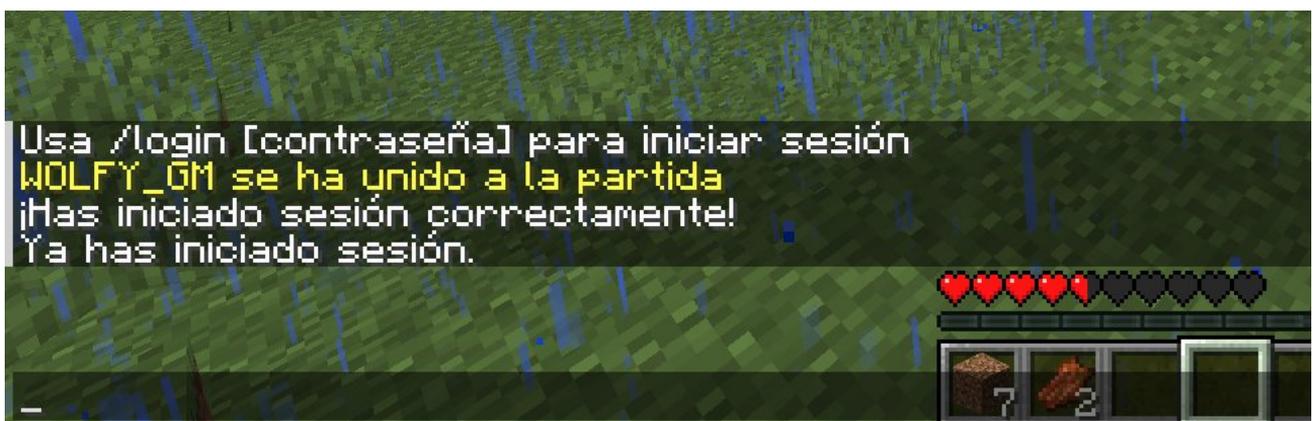
6. Cliente Minecraft



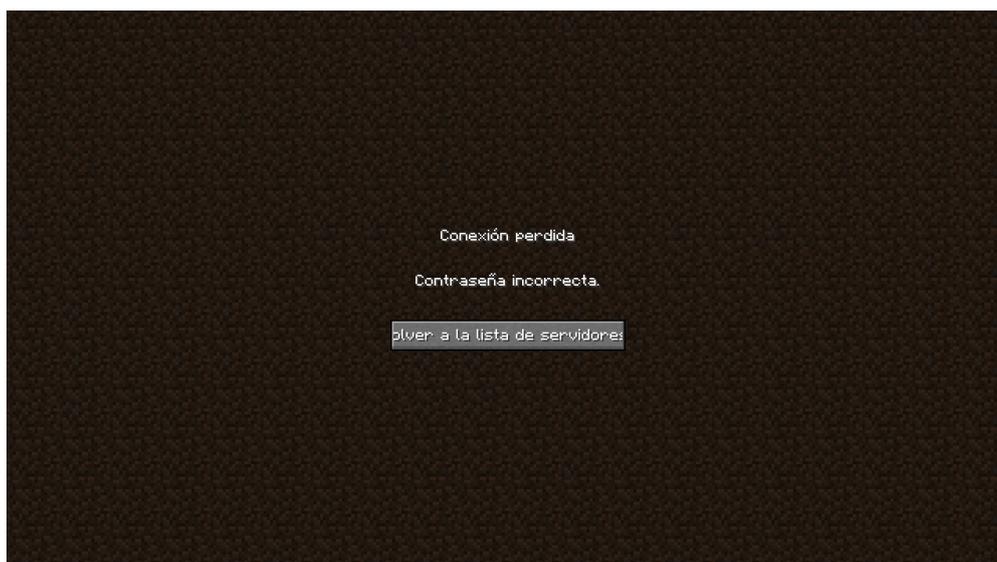
Cuando un jugador se une, se le solicitará la contraseña



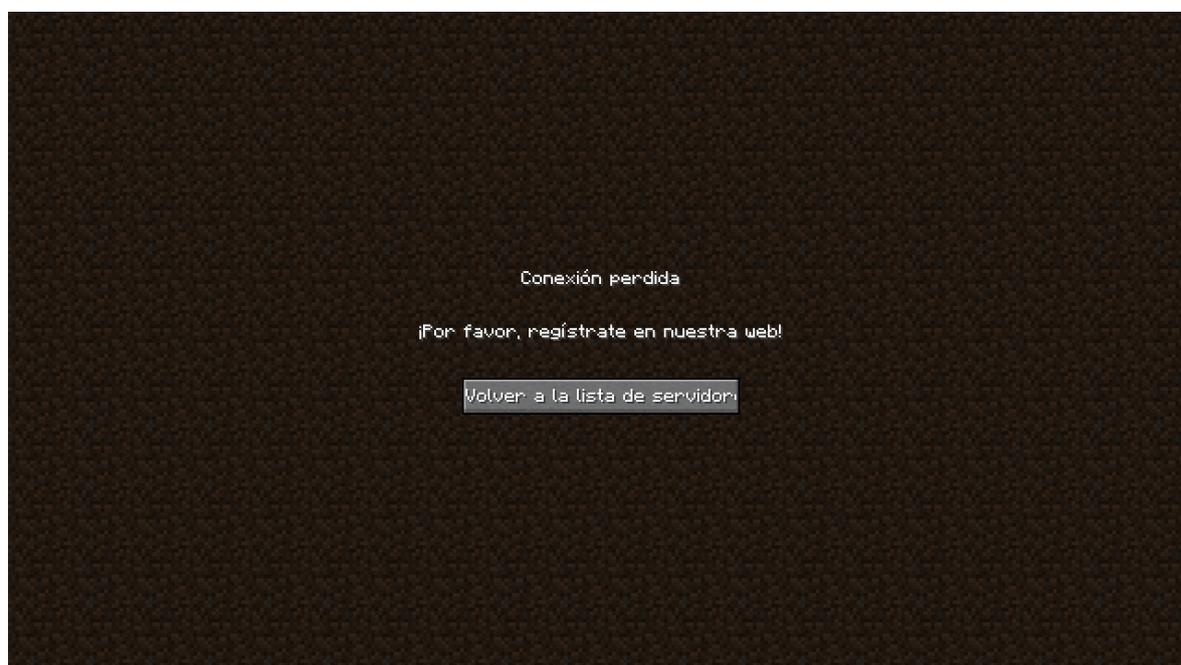
Si es correcta saldrá el mensaje de aviso de que es correcta



Si ya había iniciado sesión, se lo notificará al usuario



En caso de que sea incorrecta, se expulsará al jugador del servidor



Si no estamos registrados, ni si quiera podríamos entrar al servidor



Si hacemos un /help login muestra los detalles del comando, debido a que en nuestro plugin.yml del proyecto es

```
plugin.yml
name: verysimplesqllogin
version: '${project.version}'
main: es.adriangomezmorales.verysimplesqllogin.Verysimplesqllogin
api-version: '1.20'
commands:
  login:
    description: Permite a los jugadores iniciar sesión. Creado por Adrian Gomez Morales para el TFC
    usage: /login [contraseña]
```

7. Vídeo demostrativo

Para ver el vídeo:

[[Click aquí](https://www.youtube.com/watch?v=e2ZDbtxC61U)] o copie este enlace <https://www.youtube.com/watch?v=e2ZDbtxC61U>