# Índice General

Capítulo 1 Introducción 3

Capítulo 2 Organización De Ficheros 4

Capítulo 3 Descripción del Código 7

3.1 Htdocs/lookup 7

3.1.1 DatosEstrellas 8

3.1.2 GetGameData 8

3.1.3 LogIn 10

3.1.4 Register 10

3.1.5 UpdateGameData 11

3.2 Servidor/Web 11

3.3 Negocio/GestionUsuarios 18

3.3.1 Login 19

3.3.2 Main 19

3.3.3 Navigator 20

3.3.4 Register 21

3.4 Negocio/Juego/Astros 21

3.4.1 CieloManager 22

3.4.2 CoordsManager 26

3.4.3 StellariumManager 26

3.5 Negocio/Juego/General 28

3.5.1 DatosEntreEscenas 29

3.5.2 Estructuras 29

3.5.3 MainManager 31

3.5.4 Miscelanea 32

3.5.5 SceneLoadManager 35

3.6 Negocio/Juego/MenuOpciones/Ajustes 36

3.6.1 ControlVolumen 37

3.6.2 FondoSettings 38

3.6.3 LocaleSelector 40

3.7 Negocio/Juego/MenuOpciones/Biblioteca 41

3.7.1 AprendizajeManager 42

3.7.2 BibliotecaManager 42

3.8 Negocio/Juego/MenuOpciones/Logros 45

3.8.1 LogrosManager 46

3.8.2 ProgresoLogrosManager 47

3.9 Negocio/Juego/MenuOpciones/Ubicación 51

3.9.1 UbicacionManager 52

# Introducción

En este manual constará el código realizado para el desarrollo del proyecto. Se detallará tanto el código de la aplicación en concreto como el que ha sido necesario desarrollar por la parte del servidor.

Se dividirá el manual en cada uno de los ficheros desarrollados, correspondiendo la gran mayoría de estos con una clase puesto que la aplicación Look Up se ha desarrollado con C#, lenguaje orientado a objetos. Sin embargo, encontraremos excepciones en ficheros creados con fines específicos y en archivos php que también han sido necesarios crear para la gestión de la base de datos y API de Stellarium.

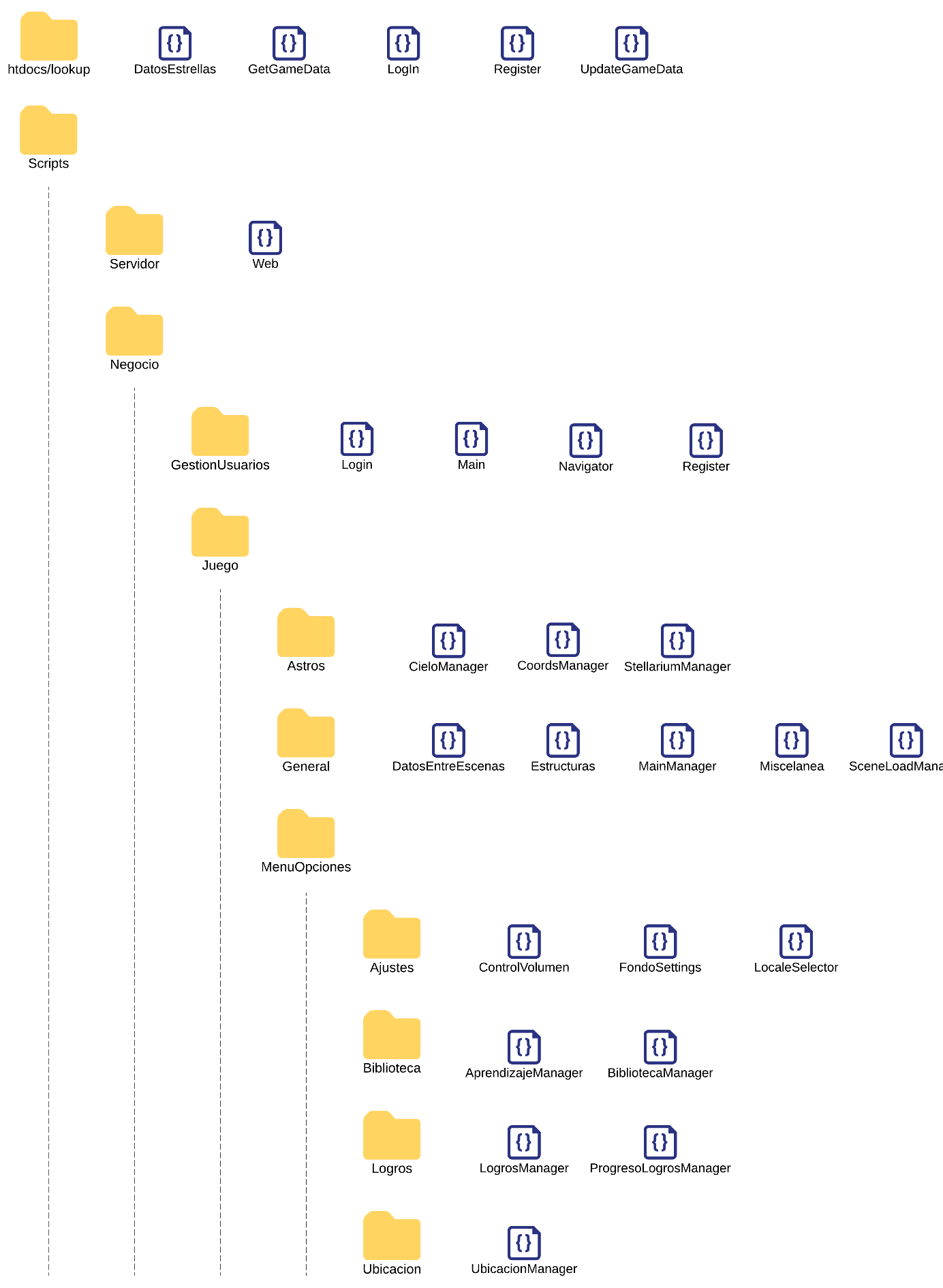
También será objetivo de este manual explicar la organización en directorios de los ficheros de código del proyecto.

# Organización De Ficheros

Tener una buena organización de los ficheros de código es esencial en un proyecto de informática. ya que influye directamente en la eficiencia, el mantenimiento y la escalabilidad del proyecto. Cuando los archivos están bien organizados, se puede trabajar de manera más productiva, encontrando rápidamente lo que sea necesario y reduciendo el tiempo dedicado a buscar o corregir errores.

En el desarrollo de este proyecto ha habido dos directorios principales: “Scripts”, directorio de código predeterminado de Unity cuyos ficheros tendrá el formato **cs**; y “htdocs”, carpeta de Xampp en la cual almacenaremos las funciones en forma de archivos **php** a los que los usuarios accederán de manera indirecta al interactuar con el sistema.

En cuando a la carpeta Scripts, presenta la siguiente estructura que podemos observar en la Figura 1:



**Figura 1:** Organización Scripts

Tenemos Servidor y Negocio como subdirectorios principales. En Servidor se encuentra el fichero Web.cs que se encarga de todas las funciones que contactan con el servidor, Mientras que en Negocio encontraremos todo el código que controla la lógica de nuestra aplicación de forma local.

Continuando por Negocio encontramos GestionUsuarios, cuyos ficheros se enfocarán en funciones de las pantallas de inicio y registro, y Juego, por donde nos adentraremos a todos los archivos que toman partida una vez hemos iniciado sesión. Siendo estos:

* Astros: en esta carpeta están todas las clases que toman partida en la realidad aumentada, es decir, a la hora de interactuar con astros.
* General: aquí encontramos ficheros de carácter general. Estos presentan métodos que se les da uso en múltiples lugares del sistema.
* MenuOpciones: nos ofrece los ficheros de las diferentes opciones y pantallas accesibles desde el menú principal de la aplicación. Esto es:
  + Ajustes: ficheros para configurar so nido, idioma y opacidad del fondo de los astros.
  + Biblioteca: AprendizajeManager es el encargado de redirigirnos a YouTube al interactuar con un vídeo. Mientras que Biblioteca Manager construye el apartado de Progreso y Detalles.
  + Logros: presenta una clase para construir la pantalla de logros y otra para controlar el desbloqueo de los mismos.
  + Ubicación: es clase se encarga de todas las tareas relacionadas con la ubicación, es decir, de la búsqueda de ciudad y del cambio de ubicación.

# Descripción del Código

## Htdocs/lookup

En este apartado tendremos presentes archivos php los cuales se accederán desde la clase Web como se verá en el punto **3.2 Servidor/Web**.

### DatosEstrellas

**<?**php

/\*\*

\* Script para obtener datos de estrellas del servidor.

\* Este script recibe las coordenadas del usuario (latitud y longitud) y devuelve

\* los datos de los cuerpos celestes visibles desde esa ubicación.

\*/

include 'db\_connection.php'**;**

**if** **(**$\_SERVER**[**'REQUEST\_METHOD'**]** **==** 'POST'**)** **{**

$correo **=** $\_POST**[**'loginUser'**];**

$longitud **=** $\_POST**[**'longitud'**];**

$latitud **=** $\_POST**[**'latitud'**];**

// Consulta SQL para obtener datos de estrellas basados en la ubicación del usuario

$query **=** "SELECT \* FROM estrellas WHERE longitud = ? AND latitud = ?"**;**

$stmt **=** $conn**->**prepare**(**$query**);**

$stmt**->**bind\_param**(**'ss'**,** $longitud**,** $latitud**);**

$stmt**->**execute**();**

$result **=** $stmt**->**get\_result**();**

$estrellas **=** array**();**

// Itera sobre los resultados y almacena los datos en un array

**while** **(**$row **=** $result**->**fetch\_assoc**())** **{**

$estrellas**[]** **=** $row**;**

**}**

echo json\_encode**(**$estrellas**);**

**}** **else** **{**

echo json\_encode**(**array**(**"error" **=>** "Método no soportado."**));**

**}**

$conn**->**close**();**

**?>**

### GetGameData

**<?**php

/\*\*

\* Script para obtener los datos del juego de un usuario específico.

\* Este script busca los datos de progreso del jugador utilizando su correo electrónico.

\*/

include 'db\_connection.php'**;**

**if** **(**$\_SERVER**[**'REQUEST\_METHOD'**]** **==** 'POST'**)** **{**

$correo **=** $\_POST**[**'loginUser'**];**

$query **=** "SELECT \* FROM progreso\_jugador WHERE correo = ?"**;**

$stmt **=** $conn**->**prepare**(**$query**);**

$stmt**->**bind\_param**(**'s'**,** $correo**);**

$stmt**->**execute**();**

$result **=** $stmt**->**get\_result**();**

$datosJuego **=** array**();**

// Almacena los datos del jugador en un array

**if** **(**$row **=** $result**->**fetch\_assoc**())** **{**

$datosJuego **=** $row**;**

**}**

// Retorna los datos del juego en formato JSON

echo json\_encode**(**$datosJuego**);**

**}** **else** **{**

echo json\_encode**(**array**(**"error" **=>** "Método no soportado."**));**

**}**

$conn**->**close**();**

**?>**

### LogIn

**<?**php

/\*\*

\* Script de inicio de sesión.

\* Verifica las credenciales del usuario (correo y contraseña) y responde

\* con un mensaje indicando si el inicio de sesión es exitoso o no.

\*/

include 'db\_connection.php'**;**

**if** **(**$\_SERVER**[**'REQUEST\_METHOD'**]** **==** 'POST'**)** **{**

$correo **=** $\_POST**[**'loginUser'**];**

$password **=** $\_POST**[**'loginPass'**];**

$query **=** "SELECT \* FROM usuarios WHERE correo = ?"**;**

$stmt **=** $conn**->**prepare**(**$query**);**

$stmt**->**bind\_param**(**'s'**,** $correo**);**

$stmt**->**execute**();**

$result **=** $stmt**->**get\_result**();**

**if** **(**$row **=** $result**->**fetch\_assoc**())** **{**

**if** **(**password\_verify**(**$password**,** $row**[**'password'**]))** **{**

echo json\_encode**(**array**(**"message" **=>** "Login correcto."**));**

**}** **else** **{**

echo json\_encode**(**array**(**"message" **=>** "Contraseña incorrecta."**));**

**}**

**}** **else** **{**

echo json\_encode**(**array**(**"message" **=>** "Usuario no encontrado."**));**

**}**

**}** **else** **{**

echo json\_encode**(**array**(**"error" **=>** "Método no soportado."**));**

**}**

$conn**->**close**();**

**?>**

### Register

**<?**php

/\*\*

\* Script de registro de usuarios.

\* Permite a los nuevos usuarios registrarse proporcionando su correo y contraseña.

\*/

include 'db\_connection.php'**;**

**if** **(**$\_SERVER**[**'REQUEST\_METHOD'**]** **==** 'POST'**)** **{**

$correo **=** $\_POST**[**'loginUser'**];**

$password **=** $\_POST**[**'loginPass'**];**

// Encripta la contraseña antes de guardarla

$passwordHash **=** password\_hash**(**$password**,** PASSWORD\_BCRYPT**);**

$query **=** "INSERT INTO usuarios (correo, password) VALUES (?, ?)"**;**

$stmt **=** $conn**->**prepare**(**$query**);**

$stmt**->**bind\_param**(**'ss'**,** $correo**,** $passwordHash**);**

**if** **(**$stmt**->**execute**())** **{**

echo json\_encode**(**array**(**"message" **=>** "Registro exitoso."**));**

**}** **else** **{**

echo json\_encode**(**array**(**"error" **=>** "Error en el registro."**));**

**}**

**}** **else** **{**

echo json\_encode**(**array**(**"error" **=>** "Método no soportado."**));**

**}**

$conn**->**close**();**

**?>**

### UpdateGameData

**<?**php

/\*\*

\* Script para actualizar los datos del juego.

\* Permite que los datos de progreso del jugador sean actualizados en la base de datos.

\*/

include 'db\_connection.php'**;**

// Verifica si la solicitud es POST

**if** **(**$\_SERVER**[**'REQUEST\_METHOD'**]** **==** 'POST'**)** **{**

$correo **=** $\_POST**[**'correo'**];**

$gameData **=** $\_POST**[**'gameData'**];**

// Consulta SQL para actualizar los datos del jugador

$query **=** "UPDATE progreso\_jugador SET datos\_juego = ? WHERE correo = ?"**;**

$stmt **=** $conn**->**prepare**(**$query**);**

$stmt**->**bind\_param**(**'ss'**,** $gameData**,** $correo**);**

**if** **(**$stmt**->**execute**())** **{**

echo json\_encode**(**array**(**"message" **=>** "Datos del juego actualizados correctamente."**));**

**}** **else** **{**

echo json\_encode**(**array**(**"error" **=>** "Error al actualizar los datos del juego."**));**

**}**

**}** **else** **{**

echo json\_encode**(**array**(**"error" **=>** "Método no soportado."**));**

**}**

$conn**->**close**();**

**?>**

## Servidor/Web

Como se explicó anteriormente, la clase Web es la encargada de todas las funciones que requieren una conexión con el servidor, ya sea para realizar una consulta a la base de datos o a la API de Stellarium.

/\*\*

\* Clase Web contiene los diferentes métodos para gestionar la autenticación,

\* registro y actualización de datos del juego a través de la web.

\*

\* **@author** Carlos Checa Moreno

\*

\*/

**using** UnityEngine**;**

**using** UnityEngine**.**Networking**;**

**using** System**.**Collections**;**

**using** UnityEngine**.**SceneManagement**;**

**using** UnityEngine**.**UI**;**

**using** System**.**IO**;**

**using** TMPro**;**

**using** UnityEngine**.**Localization**;**

**using** UnityEngine**.**Localization**.**Settings**;**

**public** class Web **:** MonoBehaviour

**{**

**public** GameObject feedbackLogin**;**

**public** GameObject errorTextPanel**;**

**public** GameObject iniciarSesion**;**

**public** GameObject registrarse**;**

**public** GameObject errorText**;**

/\*\*

\* Método Start inicializa los objetos de interfaz de usuario, ocultando

\* paneles de error y login al comienzo.

\*/

void Start**()**

**{**

**if** **(**errorTextPanel **!=** **null)**

**{**

errorTextPanel**.**SetActive**(false);**

**}**

**if** **(**feedbackLogin **!=** **null)**

**{**

Debug**.**Log**(**"no es null"**);**

feedbackLogin**.**SetActive**(false);**

**}**

**}**

/\*\*

\* Método GetGameData obtiene los datos del jugador desde el servidor

\* y los guarda localmente en un archivo JSON.

\*

\* **@param** username Nombre de usuario para obtener los datos de juego

\* **@return** IEnumerator para manejar la solicitud web asíncrona

\*/

**public** IEnumerator GetGameData**(**string username**)**

**{**

WWWForm form **=** **new** WWWForm**();**

form**.**AddField**(**"loginUser"**,** username**);**

string path **=** DatosEntreEscenas**.**urlToLocalHost **+** "/lookupbbdd/GetGameData.php"**;**

**using** **(**UnityWebRequest www **=** UnityWebRequest**.**Post**(**path**,** form**))**

**{**

**yield** **return** www**.**Send**();**

**if** **(**www**.**isNetworkError **||** www**.**isHttpError**)**

**{**

errorTextPanel**.**SetActive**(true);**

Debug**.**Log**(**www**.**error**);**

**}**

**else**

**{**

Debug**.**Log**(**"Terminando de cargar datos de juego..."**);**

string jsonArray **=** www**.**downloadHandler**.**text**;**

string filePath **=** Application**.**persistentDataPath **+** "/playerProgression.json"**;**

File**.**WriteAllText**(**filePath**,** jsonArray**);**

Debug**.**Log**(**"Datos del jugador guardados en: " **+** filePath**);**

**}**

**}**

**}**

/\*\*

\* Método Login maneja el inicio de sesión del usuario, verificando las credenciales

\* en el servidor y cargando los datos del jugador si son correctas.

\*

\* **@param** username Nombre de usuario para iniciar sesión

\* **@param** userpassword Contraseña del usuario

\* **@return** IEnumerator para manejar la solicitud web asíncrona

\*/

**public** IEnumerator Login**(**string username**,** string userpassword**)**

**{**

Button button **=** iniciarSesion**.**GetComponent**<**Button**>();**

ColorBlock colors **=** button**.**colors**;**

Color originalNormalColor **=** colors**.**normalColor**;**

colors**.**normalColor **=** Color**.**gray**;**

button**.**colors **=** colors**;**

button**.**interactable **=** **false;**

TextMeshProUGUI errorTextonly **=** feedbackLogin**.**GetComponent**<**TextMeshProUGUI**>();**

WWWForm form **=** **new** WWWForm**();**

form**.**AddField**(**"loginUser"**,** username**);**

form**.**AddField**(**"loginPass"**,** userpassword**);**

string path **=** DatosEntreEscenas**.**urlToLocalHost **+** "/lookupbbdd/LogIn.php"**;**

**using** **(**UnityWebRequest www **=** UnityWebRequest**.**Post**(**path**,** form**))**

**{**

**yield** **return** www**.**SendWebRequest**();**

**if** **(**www**.**result **!=** UnityWebRequest**.**Result**.**Success**)**

**{**

errorTextonly**.**text **=** LocalizationSettings**.**StringDatabase**.**GetLocalizedString**(**"errores"**,** "e1001"**);**

feedbackLogin**.**SetActive**(true);**

Debug**.**Log**(**www**.**error**);**

**}**

**else**

**{**

**if** **(!**www**.**downloadHandler**.**text**.**Contains**(**"Login correcto."**))**

**{**

errorTextonly**.**text **=** LocalizationSettings**.**StringDatabase**.**GetLocalizedString**(**"preGame"**,** "credencialE"**);**

feedbackLogin**.**SetActive**(true);**

**}**

**else**

**{**

**yield** **return** StartCoroutine**(**GetGameData**(**username**));**

DatosEntreEscenas**.**correo **=** username**;**

DatosEntreEscenas**.**logged **=** **true;**

SceneManager**.**LoadScene**(**"Juego"**);**

**}**

**}**

**}**

colors**.**normalColor **=** originalNormalColor**;**

button**.**colors **=** colors**;**

button**.**interactable **=** **true;**

**}**

/\*\*

\* Método Register registra un nuevo usuario en el servidor, verificando que

\* los datos de registro sean válidos antes de proceder.

\*

\* **@param** username Nombre de usuario para el registro

\* **@param** password Contraseña del usuario

\* **@param** passwordrepeat Confirmación de la contraseña

\* **@return** IEnumerator para manejar la solicitud web asíncrona

\*/

**public** IEnumerator Register**(**string username**,** string password**,** string passwordrepeat**)**

**{**

Button button **=** registrarse**.**GetComponent**<**Button**>();**

ColorBlock colors **=** button**.**colors**;**

Color originalNormalColor **=** colors**.**normalColor**;**

colors**.**normalColor **=** Color**.**gray**;**

button**.**colors **=** colors**;**

button**.**interactable **=** **false;**

**try**

**{**

string path **=** DatosEntreEscenas**.**urlToLocalHost **+** "/lookupbbdd/Register.php"**;**

Text errorTextonly **=** feedbackLogin**.**GetComponent**<**Text**>();**

**if** **(!**IsValidEmail**(**username**))**

**{**

errorTextonly**.**text **=** LocalizationSettings**.**StringDatabase**.**GetLocalizedString**(**"preGame"**,** "noValidMail"**);**

feedbackLogin**.**SetActive**(true);**

**yield** **break;**

**}**

**if** **(**password **!=** passwordrepeat**)**

**{**

errorTextonly**.**text **=** LocalizationSettings**.**StringDatabase**.**GetLocalizedString**(**"preGame"**,** "noMatchPass"**);**

feedbackLogin**.**SetActive**(true);**

**yield** **break;**

**}**

**if** **(**password**.**Length **<** 6**)**

**{**

errorTextonly**.**text **=** "La contraseña debe tener más de 6 caracteres."**;**

feedbackLogin**.**SetActive**(true);**

**yield** **break;**

**}**

WWWForm form **=** **new** WWWForm**();**

form**.**AddField**(**"loginUser"**,** username**);**

form**.**AddField**(**"loginPass"**,** password**);**

**using** **(**UnityWebRequest www **=** UnityWebRequest**.**Post**(**path**,** form**))**

**{**

**yield** **return** www**.**SendWebRequest**();**

**if** **(**www**.**result **!=** UnityWebRequest**.**Result**.**Success**)**

**{**

errorTextonly**.**text **=** LocalizationSettings**.**StringDatabase**.**GetLocalizedString**(**"errores"**,** "e1001"**);**

feedbackLogin**.**SetActive**(true);**

Debug**.**Log**(**www**.**error**);**

**}**

**else**

**{**

**yield** **return** StartCoroutine**(**Login**(**username**,** password**));**

**}**

**}**

**}**

**finally**

**{**

colors**.**normalColor **=** originalNormalColor**;**

button**.**colors **=** colors**;**

button**.**interactable **=** **true;**

**}**

**}**

/\*\*

\* Método IsValidEmail verifica si una cadena tiene el formato correcto de un correo electrónico.

\*

\* **@param** email Cadena que representa el correo electrónico a validar

\* **@return** bool Devuelve true si el correo es válido, false en caso contrario

\*/

**private** bool IsValidEmail**(**string email**)**

**{**

**try**

**{**

var addr **=** **new** System**.**Net**.**Mail**.**MailAddress**(**email**);**

**return** addr**.**Address **==** email**;**

**}**

**catch**

**{**

**return** **false;**

**}**

**}**

/\*\*

\* Método DatosEstrellas obtiene los datos de progreso de las estrellas para un usuario

\* desde el servidor.

\*

\* **@param** username Nombre de usuario para obtener los datos de estrellas

\* **@return** IEnumerator para manejar la solicitud web asíncrona

\*/

**public** IEnumerator DatosEstrellas**(**string username**)**

**{**

string path **=** DatosEntreEscenas**.**urlToLocalHost **+** "/lookupbbdd/DatosEstrellas.php"**;**

Text errorTextonly **=** feedbackLogin**.**GetComponent**<**Text**>();**

WWWForm form **=** **new** WWWForm**();**

form**.**AddField**(**"loginUser"**,** username**);**

**using** **(**UnityWebRequest www **=** UnityWebRequest**.**Post**(**path**,** form**))**

**{**

**yield** **return** www**.**SendWebRequest**();**

**if** **(**www**.**result **!=** UnityWebRequest**.**Result**.**Success**)**

**{**

Debug**.**Log**(**www**.**error**);**

**}**

**else**

**{**

Debug**.**Log**(**www**.**downloadHandler**.**text**);**

**}**

**}**

**}**

/\*\*

\* Método UpdateGameData actualiza los datos de juego de un usuario en el servidor,

\* después de procesar los cambios localmente.

\*

\* **@param** astro Objeto con el progreso de un cuerpo celeste

\* **@param** callback Acción que devuelve un booleano indicando éxito o fallo

\* **@return** IEnumerator para manejar la solicitud web asíncrona

\*/

**public** IEnumerator UpdateGameData**(**infoCuerpoProgreso astro**,** System**.**Action**<**bool**>** callback**)**

**{**

**if** **(**Miscelanea**.**UpdatePlayerProgress**(**astro**))**

**{**

WWWForm form **=** **new** WWWForm**();**

form**.**AddField**(**"correo"**,** DatosEntreEscenas**.**correo**);**

form**.**AddField**(**"gameData"**,** Miscelanea**.**CombinePlayerData**());**

string path **=** DatosEntreEscenas**.**urlToLocalHost **+** "/lookupbbdd/UpdateGameData.php"**;**

**using** **(**UnityWebRequest www **=** UnityWebRequest**.**Post**(**path**,** form**))**

**{**

**yield** **return** www**.**SendWebRequest**();**

**if** **(**www**.**result **!=** UnityWebRequest**.**Result**.**Success**)**

**{**

callback**(false);**

**}**

**else**

**{**

callback**(true);**

**}**

**}**

**}**

**else**

**{**

callback**(false);**

**}**

**}**

/\*\*

\* Método UpdateLogroProgreso actualiza el progreso de un logro de usuario en el servidor,

\* después de validar el progreso localmente.

\*

\* **@param** nombreLogro Nombre del logro a actualizar

\* **@param** callback Acción que devuelve un booleano indicando éxito o fallo

\* **@return** IEnumerator para manejar la solicitud web asíncrona

\*/

**public** IEnumerator UpdateLogroProgreso**(**string nombreLogro**,** System**.**Action**<**bool**>** callback**)**

**{**

bool logroActualizado **=** Miscelanea**.**UpdateLogroProgreso**(**nombreLogro**);**

**if** **(**logroActualizado**)**

**{**

WWWForm form **=** **new** WWWForm**();**

form**.**AddField**(**"correo"**,** DatosEntreEscenas**.**correo**);**

form**.**AddField**(**"gameData"**,** Miscelanea**.**CombinePlayerData**());**

string path **=** DatosEntreEscenas**.**urlToLocalHost **+** "/lookupbbdd/UpdateGameData.php"**;**

**using** **(**UnityWebRequest www **=** UnityWebRequest**.**Post**(**path**,** form**))**

**{**

**yield** **return** www**.**SendWebRequest**();**

**if** **(**www**.**result **!=** UnityWebRequest**.**Result**.**Success**)**

**{**

callback**(false);**

**}**

**else**

**{**

callback**(true);**

**}**

**}**

**}**

**else**

**{**

callback**(false);**

**}**

**}**

**}**

## Negocio/GestionUsuarios

Como se explicó anteriormente, este apartado agrupa las clases que manejan la gestión de usuarios, como el inicio de sesión, registro y navegación en la aplicación. Aquí se encuentran clases esenciales para el flujo de autenticación y cambio de pantallas dentro del juego.

### Login

/\*\*

\* Clase Login gestiona el proceso de inicio de sesión,

\* capturando los datos de usuario y lanzando la solicitud de autenticación.

\*

\* **@author** Carlos Checa Moreno

\*

\*/

**using** System**.**Collections**;**

**using** System**.**Collections**.**Generic**;**

**using** UnityEngine**;**

**using** UnityEngine**.**UI**;**

**public** class Login **:** MonoBehaviour

**{**

**public** InputField UsernameInput**;**

**public** InputField PasswordInput**;**

**public** Button LoginButton**;**

/\*\*

\* Método Start inicializa el botón de login, añadiendo el evento

\* que lanzará el proceso de autenticación del usuario.

\*/

void Start **()**

**{**

LoginButton**.**onClick**.**AddListener**(()** **=>** **{**

StartCoroutine**(**Main**.**Instance**.**Web**.**Login**(**UsernameInput**.**text**,** PasswordInput**.**text**));**

**});**

**}**

**}**

### Main

/\*\*

\* Clase Main es el controlador principal de la aplicación,

\* gestiona la configuración inicial y provee acceso a instancias compartidas.

\*

\* **@author** Carlos Checa Moreno

\*

\*/

**using** System**.**Collections**;**

**using** System**.**Collections**.**Generic**;**

**using** UnityEngine**;**

**public** class Main **:** MonoBehaviour

**{**

**public** static Main Instance**;**

**[**SerializeField**]** **private** AudioSource myAudioSource**;**

**public** Web Web**;**

/\*\*

\* Método Start inicializa la instancia de la clase Main y configura

\* el volumen de los efectos de audio desde las preferencias del jugador.

\*/

void Start **()**

**{**

float volume **=** PlayerPrefs**.**GetFloat**(**"volumenEfectos"**,** 0.5f**);**

myAudioSource**.**volume **=** volume**;**

Instance **=** **this;**

Web **=** GetComponent**<**Web**>();**

**}**

**}**

### Navigator

/\*\*

\* Clase Navigator permite la navegación entre diferentes escenas de la aplicación,

\* proporcionando métodos para cambiar entre las pantallas de registro e inicio de sesión.

\*

\* **@author** Carlos Checa Moreno

\*

\*/

**using** System**.**Collections**;**

**using** System**.**Collections**.**Generic**;**

**using** UnityEngine**;**

**using** UnityEngine**.**SceneManagement**;**

**using** UnityEngine**.**UI**;**

**public** class Navigator **:** MonoBehaviour

**{**

/\*\*

\* Método toRegister cambia la escena actual a la pantalla de registro.

\*/

**public** void toRegister**()**

**{**

SceneManager**.**LoadScene**(**"Register"**);**

**}**

/\*\*

\* Método toLogIn cambia la escena actual a la pantalla de inicio de sesión.

\*/

**public** void toLogIn**()**

**{**

SceneManager**.**LoadScene**(**"LogIn"**);**

**}**

**}**

### Register

/\*\*

\* Clase Register gestiona el proceso de registro de nuevos usuarios,

\* capturando la información del formulario y lanzando la solicitud de registro.

\*

\* **@author** Carlos Checa Moreno

\*

\*/

**using** System**.**Collections**;**

**using** System**.**Collections**.**Generic**;**

**using** UnityEngine**;**

**using** UnityEngine**.**UI**;**

**public** class Register **:** MonoBehaviour

**{**

**public** InputField UsernameInput**;**

**public** InputField PasswordInput**;**

**public** InputField PasswordRepeatInput**;**

**public** Button RegisterButton**;**

/\*\*

\* Método Start inicializa el botón de registro, añadiendo el evento

\* que lanzará el proceso de registro del usuario.

\*/

void Start **()**

**{**

RegisterButton**.**onClick**.**AddListener**(()** **=>** **{**

StartCoroutine**(**Main**.**Instance**.**Web**.**Register**(**UsernameInput**.**text**,** PasswordInput**.**text**,** PasswordRepeatInput**.**text**));**

**});**

**}**

**}**

## Negocio/Juego/Astros

Este punto aborda las clases responsables de manejar los datos y visualización de los cuerpos celestes dentro del juego. Incluye funcionalidades que permiten obtener la información de las estrellas y planetas, así como la gestión de sus coordenadas y representación en la interfaz de usuario.

### CieloManager

/\*\*

\* Clase CieloManager se encarga de crear representaciones visuales de los objetos celestes

\* en la escena de juego, incluyendo constelaciones y planetas.

\*

\* **@author** Carlos Checa Moreno

\*

\*/

**using** System**.**Collections**;**

**using** System**.**Collections**.**Generic**;**

**using** UnityEngine**;**

**using** UnityEngine**.**UI**;**

**using** TMPro**;**

**public** class CieloManager **:** MonoBehaviour

**{**

**public** GameObject constelacionPrefab**;** // Prefab de las constelaciones

**public** Transform parentElementConstelacion**;** // Elemento padre bajo el cual se crearán las constelaciones

**public** Web webScript**;**

**public** AudioSource AudioSuccess**;**

**public** AudioSource AudioYaDescubierto**;**

**public** GameObject PlanetaPrefab**;** // Prefab para los planetas

**public** GameObject dialogoDescubierto**;**

**private** CanvasGroup canvasGroup**;**

/\*\*

\* Método Start inicializa los elementos de la interfaz de usuario y configura el diálogo de descubrimiento.

\*/

void Start**()**

**{**

canvasGroup **=** dialogoDescubierto**.**GetComponent**<**CanvasGroup**>();**

**}**

/\*\*

\* Método ShowDialogo muestra un diálogo temporal de descubrimiento en pantalla.

\* **@return** IEnumerator para manejar la animación de aparición y desaparición del diálogo

\*/

IEnumerator ShowDialogo**()**

**{**

dialogoDescubierto**.**SetActive**(true);**

canvasGroup**.**alpha **=** 1**;**

**yield** **return** **new** WaitForSeconds**(**2**);**

float fadeDuration **=** 1.5f**;**

float fadeSpeed **=** 1f **/** fadeDuration**;**

**while** **(**canvasGroup**.**alpha **>** 0**)**

**{**

canvasGroup**.**alpha **-=** Time**.**deltaTime **\*** fadeSpeed**;**

**yield** **return** **null;**

**}**

dialogoDescubierto**.**SetActive**(false);**

**}**

/\*\*

\* Método CreateObjetosCielo crea visualmente los objetos celestes en la escena,

\* asignando sprites o modelos 3D dependiendo del tipo de objeto (constelación o planeta).

\*

\* **@param** celestialObjects Lista de objetos celestes a visualizar

\* **@param** bibliotecaManager Referencia al manager de la biblioteca de datos del jugador

\* **@return** IEnumerator para manejar la creación asíncrona de los objetos

\*/

**public** IEnumerator CreateObjetosCielo**(**List**<**CelestialObjectInfo**>** celestialObjects**,** BibliotecaManager bibliotecaManager**)**

**{**

**foreach** **(**CelestialObjectInfo objetoCielo **in** celestialObjects**)**

**{**

**if** **(**objetoCielo**.**objectType **!=** "planeta"**)**

**{**

GameObject spriteObject **=** Instantiate**(**constelacionPrefab**,** parentElementConstelacion**);**

spriteObject**.**name **=** objetoCielo**.**name**;**

SpriteRenderer spriteRenderer **=** spriteObject**.**GetComponent**<**SpriteRenderer**>();**

**if** **(**spriteRenderer **==** **null)**

**{**

Debug**.**LogWarning**(**"El prefab no contiene un componente SpriteRenderer."**);**

**continue;**

**}**

var sprite **=** Resources**.**Load**<**Sprite**>(**"Constelaciones\_cielo/" **+** objetoCielo**.**name**);**

**if** **(**sprite **!=** **null)**

**{**

spriteRenderer**.**sprite **=** sprite**;**

**}**

**else**

**{**

Debug**.**LogWarning**(**$"No se pudo cargar el sprite para {objetoCielo.name} "**);**

**}**

spriteRenderer**.**transform**.**localPosition **=** objetoCielo**.**unityPosition**;**

**if** **(**objetoCielo**.**unityPosition**.**y **<** 2**)**

**{**

spriteRenderer**.**transform**.**localScale **=** **new** Vector3**(**0**,** 0**,** 0**);**

**}**

spriteRenderer**.**transform**.**LookAt**(**Camera**.**main**.**transform**);**

TextMeshPro titleTransform **=** spriteObject**.**transform**.**Find**(**"Title"**).**GetComponent**<**TextMeshPro**>();**

**if** **(**titleTransform **!=** **null)**

**{**

titleTransform**.**text **=** objetoCielo**.**name**;**

**}**

**else**

**{**

Debug**.**LogWarning**(**"No se encontró el objeto hijo 'Title'."**);**

**}**

Transform buttonTransform **=** spriteObject**.**transform**.**Find**(**"Canvas/Button"**);**

**if** **(**buttonTransform **!=** **null)**

**{**

Button button **=** buttonTransform**.**GetComponent**<**Button**>();**

**if** **(**button **!=** **null)**

**{**

button**.**onClick**.**AddListener**(()** **=>** **{**

infoCuerpoProgreso astro **=** **new** infoCuerpoProgreso

**{**

name **=** objetoCielo**.**name**,**

altitude **=** objetoCielo**.**altitude**,**

azimuth **=** objetoCielo**.**azimuth**,**

distance **=** objetoCielo**.**distance**,**

location **=** DatosEntreEscenas**.**ubicacion**,**

date **=** DatosEntreEscenas**.**horaDescargaDatos**.**ToString**()**

**};**

ProgresoLogrosManager progresoLogrosManager **=** button**.**GetComponent**<**ProgresoLogrosManager**>();**

**if** **(**progresoLogrosManager **!=** **null)**

**{**

progresoLogrosManager**.**IncrementarProgreso**(**"constDesc1"**);**

progresoLogrosManager**.**IncrementarProgreso**(**"constDesc2"**);**

**}**

StartCoroutine**(**webScript**.**UpdateGameData**(**astro**,** **(**success**)** **=>** **{**

**if** **(**success**)**

**{**

StartCoroutine**(**bibliotecaManager**.**CrearPanelIndividual**(**astro**));**

AudioSuccess**.**Play**();**

**}**

**else**

**{**

**if(!**dialogoDescubierto**.**activeSelf**)**

**{**

StartCoroutine**(**ShowDialogo**());**

**}**

AudioYaDescubierto**.**Play**();**

**}**

**}));**

**});**

**}**

**}**

**}**

**else**

**{**

GameObject object3D **=** Instantiate**(**PlanetaPrefab**,** parentElementConstelacion**);**

object3D**.**name **=** objetoCielo**.**name**;**

MeshRenderer meshRenderer **=** object3D**.**GetComponent**<**MeshRenderer**>();**

Material material **=** Resources**.**Load**<**Material**>(**$"Materiales/{objetoCielo.name}"**);**

**if** **(**material **!=** **null)**

**{**

meshRenderer**.**material **=** material**;**

**}**

object3D**.**transform**.**localPosition **=** objetoCielo**.**unityPosition**;**

**if** **(**objetoCielo**.**unityPosition**.**y **<** 2**)**

**{**

object3D**.**transform**.**localScale **=** **new** Vector3**(**0**,** 0**,** 0**);**

**}**

object3D**.**transform**.**LookAt**(**Camera**.**main**.**transform**);**

Transform titleTransform **=** object3D**.**transform**.**Find**(**"ConstelacionPrefab/Title"**);**

**if** **(**titleTransform **!=** **null)**

**{**

TextMeshPro titleText **=** titleTransform**.**GetComponent**<**TextMeshPro**>();**

titleText**.**text **=** objetoCielo**.**name**;**

**}**

Transform buttonTransform **=** object3D**.**transform**.**Find**(**"ConstelacionPrefab/Canvas/Button"**);**

**if** **(**buttonTransform **!=** **null)**

**{**

Button button **=** buttonTransform**.**GetComponent**<**Button**>();**

button**.**onClick**.**AddListener**(()** **=>** **{**

infoCuerpoProgreso astro **=** **new** infoCuerpoProgreso

**{**

name **=** objetoCielo**.**name**,**

altitude **=** objetoCielo**.**altitude**,**

azimuth **=** objetoCielo**.**azimuth**,**

distance **=** objetoCielo**.**distance**,**

location **=** DatosEntreEscenas**.**ubicacion**,**

date **=** DatosEntreEscenas**.**horaDescargaDatos**.**ToString**()**

**};**

ProgresoLogrosManager progresoLogrosManager **=** button**.**GetComponent**<**ProgresoLogrosManager**>();**

progresoLogrosManager**.**IncrementarProgreso**(**"planet1"**);**

StartCoroutine**(**webScript**.**UpdateGameData**(**astro**,** **(**success**)** **=>** **{**

**if** **(**success**)**

**{**

StartCoroutine**(**bibliotecaManager**.**CrearPanelIndividual**(**astro**));**

AudioSuccess**.**Play**();**

**}**

**else**

**{**

**if(!**dialogoDescubierto**.**activeSelf**)**

**{**

StartCoroutine**(**ShowDialogo**());**

**}**

AudioYaDescubierto**.**Play**();**

**}**

**}));**

**});**

**}**

**}**

**}**

**yield** **return** **null;**

**}**

**}**

### CoordsManager

/\*\*

\* Clase CoordsManager convierte coordenadas ecuatoriales (azimut y altitud)

\* a coordenadas cartesianas para su uso en el motor gráfico.

\*

\* **@author** Carlos Checa Moreno

\*

\*/

**using** System**.**Collections**;**

**using** System**.**Collections**.**Generic**;**

**using** UnityEngine**;**

**public** class CoordsManager **:** MonoBehaviour

**{**

/\*\*

\* Método EquatorialToCartesian convierte coordenadas ecuatoriales a cartesianas,

\* utilizando la distancia y las coordenadas angulares azimut y altitud.

\*

\* **@param** azimuth Azimut del objeto celeste

\* **@param** altitude Altitud del objeto celeste

\* **@param** distance Distancia al objeto

\* **@return** Vector3 Devuelve las coordenadas cartesianas

\*/

**public** static Vector3 EquatorialToCartesian**(**float azimuth**,** float altitude**,** float distance**)**

**{**

distance **=** 15f**;**

float azimuthRad **=** azimuth **\*** Mathf**.**Deg2Rad**;**

float altitudeRad **=** altitude **\*** Mathf**.**Deg2Rad**;**

float x **=** Mathf**.**Cos**(**altitudeRad**)** **\*** Mathf**.**Cos**(**azimuthRad**)** **\*** distance**;**

float y **=** Mathf**.**Cos**(**altitudeRad**)** **\*** Mathf**.**Sin**(**azimuthRad**)** **\*** distance**;**

float z **=** Mathf**.**Sin**(**altitudeRad**)** **\*** distance**;**

**return** **new** Vector3**(**x**,** y**,** z**);**

**}**

**}**

### StellariumManager

/\*\*

\* Clase StellariumManager se encarga de obtener los datos celestiales

\* desde un servidor remoto y convertirlos en objetos manejables por el motor gráfico.

\*

\* **@author** Carlos Checa Moreno

\*

\*/

**using** System**.**Collections**;**

**using** System**.**Collections**.**Generic**;**

**using** UnityEngine**;**

**using** UnityEngine**.**Networking**;**

**using** System**;**

**using** System**.**Linq**;**

**public** class StellariumManager **:** MonoBehaviour

**{**

/\*\*

\* Método FetchCelestialData obtiene los datos de los objetos celestes del servidor y

\* los convierte a un formato usable en Unity.

\*

\* **@param** celestialObjects Lista donde se almacenarán los datos de los objetos celestes

\* **@return** IEnumerator para manejar la solicitud web asíncrona

\*/

**public** IEnumerator FetchCelestialData**(**List**<**CelestialObjectInfo**>** celestialObjects**)**

**{**

Debug**.**Log**(**"Obteniendo datos de astros..."**);**

string path **=** DatosEntreEscenas**.**urlToLocalHost **+** "/lookupbbdd/DatosEstrellas.php"**;**

WWWForm form **=** **new** WWWForm**();**

form**.**AddField**(**"loginUser"**,** DatosEntreEscenas**.**correo**);**

form**.**AddField**(**"longitud"**,** DatosEntreEscenas**.**longitud**.**ToString**());**

form**.**AddField**(**"latitud"**,** DatosEntreEscenas**.**latitud**.**ToString**());**

**using** **(**UnityWebRequest www **=** UnityWebRequest**.**Post**(**path**,** form**))**

**{**

**yield** **return** www**.**SendWebRequest**();**

**if** **(**www**.**result **!=** UnityWebRequest**.**Result**.**Success**)**

**{**

Debug**.**Log**(**www**.**error**);**

**}**

**else**

**{**

string jsonResponse **=** "{\"constellations\":" **+** www**.**downloadHandler**.**text **+** "}"**;**

**try**

**{**

ConstellationDataList celestialDataList **=** JsonUtility**.**FromJson**<**ConstellationDataList**>(**jsonResponse**);**

**foreach** **(**var data **in** celestialDataList**.**constellations**)**

**{**

CelestialObjectInfo info **=** **new** CelestialObjectInfo

**{**

name **=** data**.**name**,**

objectType **=** data**.**objectType**,**

altitude **=** data**.**altitude**,**

azimuth **=** data**.**azimuth**,**

distance **=** data**.**distance**,**

unityPosition **=** CoordsManager**.**EquatorialToCartesian**(**data**.**azimuth**,** data**.**altitude**,** data**.**distance**),**

discovered **=** DatosEntreEscenas**.**playerProgress**.**discovered\_constellations**.**Any**(**c **=>** c**.**name **==** data**.**name**),**

**};**

celestialObjects**.**Add**(**info**);**

**}**

**}**

**catch** **(**Exception e**)**

**{**

Debug**.**LogError**(**"Error reading JSON file: " **+** e**.**Message**);**

**}**

Debug**.**Log**(**"Fetched all celestial object data."**);**

**}**

**}**

**}**

**}**

## Negocio/Juego/General

Este apartado engloba clases con funciones generales del juego, como el manejo de los datos compartidos entre escenas, estructuras de datos clave para el progreso del jugador, y la carga de escenas. Estas clases son fundamentales para la continuidad de la experiencia de juego.

Cabe mencionar que aquí se encuentra el fichero de “Estructuras” que forma para de los archivos excepcionales de los que se habló al principio del documento. Este archivo es un conjunto de clases que se usan a modo de estructuras para facilitar el traspaso de información entre diferentes clases del sistema.

### DatosEntreEscenas

/\*\*

\* Clase DatosEntreEscenas contiene variables estáticas que se utilizan para

\* compartir datos entre diferentes escenas, como información del jugador,

\* ubicación y progreso.

\*

\* **@author** Carlos Checa Moreno

\*

\*/

**using** System**;**

**using** UnityEngine**;**

**public** static class DatosEntreEscenas

**{**

**public** static string correo **{** **get;** **set;** **}** **=** "admin"**;**

**public** static string pathDatosUsuario **{** **get;** **set;** **}** **=** Application**.**persistentDataPath **+** "/playerProgression.json"**;**

**public** static string urlToLocalHost **{** **get;** **set;** **}** **=** "https://rested-fox-accurately.ngrok-free.app"**;**

**public** static string ubicacion **{** **get;** **set;** **}** **=** "Córdoba"**;**

**public** static float longitud **{** **get;** **set;** **}** **=** **-**4.7727500f**;**

**public** static float latitud **{** **get;** **set;** **}** **=** 37.8915500f**;**

**public** static DateTime horaDescargaDatos **{** **get;** **set;** **}** **=** DateTime**.**Now**;**

**public** static bool logged **{** **get;** **set;** **}** **=** **false;**

**public** static int biblioLogro **{** **get;** **set;** **}** **=** 0**;**

**public** static int constDescLogro **{** **get;** **set;** **}** **=** 0**;**

**public** static int generalLogro **{** **get;** **set;** **}** **=** 0**;**

**public** static int planetLogro **{** **get;** **set;** **}** **=** 0**;**

**public** static int ubiLogro **{** **get;** **set;** **}** **=** 0**;**

**public** static PlayerProgress playerProgress**;**

**public** static PlayerLogros playerLogros**;**

**public** static GameObject logros**;**

**public** static GameObject notificacionLogros**;**

**public** static bool conseguido **=** **false;**

**}**

### Estructuras

/\*\*

\* Archivo Estructuras contiene las clases de datos y estructuras utilizadas

\* en la aplicación, como información sobre los logros, progreso del jugador,

\* datos celestiales y ciudades.

\*

\* **@author** Carlos Checa Moreno

\*

\*/

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** UnityEngine**;**

**[**Serializable**]**

**public** class infoLogros

**{**

**public** string name**;**

**public** bool conseguido**;**

**public** int progreso**;**

**}**

**[**Serializable**]**

**public** class PlayerLogros

**{**

**public** List**<**infoLogros**>** listaLogros**;**

**}**

**[**Serializable**]**

**public** class infoCuerpoProgreso

**{**

**public** string name**;**

**public** float altitude**;**

**public** float azimuth**;**

**public** float distance**;**

**public** string location**;**

**public** string date**;**

**}**

**[**Serializable**]**

**public** class PlayerProgress

**{**

**public** List**<**infoCuerpoProgreso**>** discovered\_constellations**;**

**}**

**[**Serializable**]**

**public** class ConstellationDataList

**{**

**public** List**<**CelestialData**>** constellations**;**

**}**

**public** class CelestialObjectInfo

**{**

**public** string name**;**

**public** string objectType**;**

**public** float altitude**;**

**public** float azimuth**;**

**public** float distance**;**

**public** Vector3 unityPosition**;**

**public** bool discovered**;**

**}**

**[**Serializable**]**

**public** class CelestialData

**{**

**public** bool found**;**

**public** string name**;**

**public** string objectType**;**

**public** float azimuth**;**

**public** float altitude**;**

**public** float distance**;**

**public** string descripcion**;**

**}**

**[**Serializable**]**

**public** class CiudadData

**{**

**public** string name**;**

**public** string country**;**

**public** float lat**;**

**public** float lng**;**

**}**

### MainManager

/\*\*

\* Clase MainManager coordina la carga inicial de datos y objetos celestes,

\* gestionando diferentes managers relacionados con la biblioteca, logros,

\* ubicaciones y el cielo.

\*

\* **@author** Carlos Checa Moreno

\*

\*/

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** UnityEngine**;**

**using** System**.**Collections**;**

**public** class MainManager **:** MonoBehaviour

**{**

**[**SerializeField**]** **private** BibliotecaManager bibliotecaManager**;**

**[**SerializeField**]** **private** UbicacionManager ubicacionManager**;**

**[**SerializeField**]** **private** LogrosManager logrosManager**;**

**[**SerializeField**]** **private** CieloManager cieloManager**;**

**[**SerializeField**]** **private** StellariumManager stellariumManager**;**

**private** List**<**CelestialObjectInfo**>** celestialObjects **=** **new** List**<**CelestialObjectInfo**>();**

**[**SerializeField**]** **private** GameObject tutorial**;**

**private** bool tutorialActive **=** **false;**

/\*\*

\* Método Start inicializa la aplicación cargando datos de progreso, logros

\* y mostrando el tutorial si es necesario.

\*/

void Start**()**

**{**

DatosEntreEscenas**.**notificacionLogros **=** GameObject**.**Find**(**"GONotificacionLogro"**);**

DatosEntreEscenas**.**notificacionLogros**.**SetActive**(false);**

DatosEntreEscenas**.**playerProgress **=** Miscelanea**.**LoadPlayerProgress**();**

DatosEntreEscenas**.**playerLogros **=** Miscelanea**.**LoadPlayerLogros**();**

**if** **(**PlayerPrefs**.**GetInt**(**"tutorial"**,** 1**)** **==** 1**)**

**{**

tutorial**.**SetActive**(true);**

PlayerPrefs**.**SetInt**(**"tutorial"**,** 0**);**

PlayerPrefs**.**Save**();**

tutorialActive **=** **true;**

**}**

**else**

**{**

tutorial**.**SetActive**(false);**

**}**

**}**

/\*\*

\* Método Inicio coordina la carga de datos celestes y la creación de paneles

\* visuales relacionados con la biblioteca, logros y ubicaciones.

\*

\* **@return** IEnumerator para manejar el flujo asíncrono de la carga

\*/

**public** IEnumerator Inicio**()**

**{**

DatosEntreEscenas**.**horaDescargaDatos **=** DateTime**.**Now**;**

**yield** **return** StartCoroutine**(**stellariumManager**.**FetchCelestialData**(**celestialObjects**));**

**yield** **return** StartCoroutine**(**bibliotecaManager**.**CreatePanelsBiblioteca**());**

**yield** **return** StartCoroutine**(**logrosManager**.**CreatePanelsLogros**());**

**yield** **return** StartCoroutine**(**ubicacionManager**.**CreatePanelsCiudades**());**

**yield** **return** StartCoroutine**(**cieloManager**.**CreateObjetosCielo**(**celestialObjects**,** bibliotecaManager**));**

**if** **(!**tutorialActive**)**

**{**

DatosEntreEscenas**.**logros**.**SetActive**(false);**

**}**

**}**

**}**

### Miscelanea

/\*\*

\* Clase Miscelanea gestiona operaciones auxiliares como la carga y actualización

\* del progreso del jugador y la combinación de datos de progreso y logros en JSON.

\*

\* **@author** Carlos Checa Moreno

\*

\*/

**using** System**.**Collections**;**

**using** System**.**Collections**.**Generic**;**

**using** UnityEngine**;**

**using** System**.**IO**;**

**using** Newtonsoft**.**Json**;**

**public** class Miscelanea **:** MonoBehaviour

**{**

/\*\*

\* Método LoadPlayerProgress carga el progreso del jugador desde un archivo JSON local.

\*

\* **@return** PlayerProgress Objeto con el progreso del jugador

\*/

**public** static PlayerProgress LoadPlayerProgress**()**

**{**

string progressFilePath **=** DatosEntreEscenas**.**pathDatosUsuario**;**

**if** **(**File**.**Exists**(**progressFilePath**))**

**{**

string json **=** File**.**ReadAllText**(**progressFilePath**);**

**return** JsonUtility**.**FromJson**<**PlayerProgress**>(**json**);**

**}**

**else**

**{**

**return** **new** PlayerProgress**();**

**}**

**}**

/\*\*

\* Método LoadPlayerLogros carga los logros del jugador desde un archivo JSON local.

\*

\* **@return** PlayerLogros Objeto con los logros del jugador

\*/

**public** static PlayerLogros LoadPlayerLogros**()**

**{**

string progressFilePath **=** DatosEntreEscenas**.**pathDatosUsuario**;**

**if** **(**File**.**Exists**(**progressFilePath**))**

**{**

string json **=** File**.**ReadAllText**(**progressFilePath**);**

**return** JsonUtility**.**FromJson**<**PlayerLogros**>(**json**);**

**}**

**else**

**{**

**return** **new** PlayerLogros**();**

**}**

**}**

/\*\*

\* Método UpdatePlayerProgress actualiza el progreso del jugador añadiendo

\* una nueva constelación descubierta.

\*

\* **@param** newConstellation Objeto infoCuerpoProgreso con la nueva constelación

\* **@return** bool Devuelve true si el progreso fue actualizado, false si ya existía

\*/

**public** static bool UpdatePlayerProgress**(**infoCuerpoProgreso newConstellation**)**

**{**

string progressFilePath **=** DatosEntreEscenas**.**pathDatosUsuario**;**

bool constellationExists **=** DatosEntreEscenas**.**playerProgress**.**discovered\_constellations**.**Exists**(**c **=>** c**.**name **==** newConstellation**.**name**);**

**if** **(!**constellationExists**)**

**{**

DatosEntreEscenas**.**playerProgress**.**discovered\_constellations**.**Add**(**newConstellation**);**

string updatedJson **=** CombinePlayerData**();**

File**.**WriteAllText**(**progressFilePath**,** updatedJson**);**

**return** **true;**

**}**

**else**

**{**

**return** **false;**

**}**

**}**

/\*\*

\* Método CombinePlayerData combina los datos de progreso y logros en un único JSON.

\*

\* **@return** string Devuelve la representación JSON combinada

\*/

**public** static string CombinePlayerData**()**

**{**

var combinedData **=** **new**

**{**

discovered\_constellations **=** DatosEntreEscenas**.**playerProgress**?.**discovered\_constellations**,**

listaLogros **=** DatosEntreEscenas**.**playerLogros**?.**listaLogros

**};**

**return** JsonConvert**.**SerializeObject**(**combinedData**,** Formatting**.**Indented**);**

**}**

/\*\*

\* Método UpdateLogroProgreso actualiza el progreso de un logro, marcándolo como conseguido

\* si se cumplen los requisitos.

\*

\* **@param** nombreLogro Nombre del logro a actualizar

\* **@return** bool Devuelve true si el logro fue conseguido, false en caso contrario

\*/

**public** static bool UpdateLogroProgreso**(**string nombreLogro**)**

**{**

infoLogros logro **=** DatosEntreEscenas**.**playerLogros**.**listaLogros**.**Find**(**l **=>** l**.**name **==** nombreLogro**);**

**if** **(**logro **!=** **null** **&&** **!**logro**.**conseguido**)**

**{**

logro**.**progreso **+=** 1**;**

**if** **(**ProgresoLogrosManager**.**valoresParaConseguirLogro**.**ContainsKey**(**nombreLogro**))**

**{**

int valorParaConseguir **=** ProgresoLogrosManager**.**valoresParaConseguirLogro**[**nombreLogro**];**

**if** **(**logro**.**progreso **>=** valorParaConseguir**)**

**{**

logro**.**conseguido **=** **true;**

DatosEntreEscenas**.**conseguido **=** **true;**

string updatedJson **=** CombinePlayerData**();**

File**.**WriteAllText**(**DatosEntreEscenas**.**pathDatosUsuario**,** updatedJson**);**

**return** **true;**

**}**

**}**

**}**

**return** **false;**

**}**

**}**

### SceneLoadManager

/\*\*

\* Clase SceneLoadManager gestiona la carga de escenas en la aplicación,

\* mostrando una barra de progreso durante el proceso.

\*

\* **@author** Carlos Checa Moreno

\*

\*/

**using** System**.**Collections**;**

**using** UnityEngine**;**

**using** UnityEngine**.**UI**;**

**public** class SceneLoadManager **:** MonoBehaviour

**{**

**[**SerializeField**]** **private** Slider loadbar**;**

**[**SerializeField**]** **private** GameObject loadPanel**;**

**[**SerializeField**]** **private** MainManager mainManager**;**

**[**SerializeField**]** **private** Canvas canvasToDisable**;**

/\*\*

\* Método Start inicia la corrutina para cargar el StellariumAPI y mostrar

\* el progreso en la barra de carga.

\*/

void Start**()**

**{**

StartCoroutine**(**LoadStellariumAPI**());**

**}**

/\*\*

\* Método LoadStellariumAPI controla la animación de la barra de carga,

\* avanzando en dos fases mientras se carga el StellariumAPI.

\*

\* **@return** IEnumerator para manejar la animación y la carga asíncrona

\*/

IEnumerator LoadStellariumAPI**()**

**{**

float duration **=** 1.0f**;**

float targetProgress **=** 0.5f**;**

float elapsed **=** 0.0f**;**

**while** **(**elapsed **<** duration**)**

**{**

elapsed **+=** Time**.**deltaTime**;**

loadbar**.value** **=** Mathf**.**Lerp**(**0**,** targetProgress**,** elapsed **/** duration**);**

**yield** **return** **null;**

**}**

**yield** **return** StartCoroutine**(**mainManager**.**Inicio**());**

targetProgress **=** 2.0f**;**

elapsed **=** 0.0f**;**

**while** **(**elapsed **<** duration**)**

**{**

elapsed **+=** Time**.**deltaTime**;**

loadbar**.value** **=** Mathf**.**Lerp**(**0.5f**,** targetProgress**,** elapsed **/** duration**);**

**yield** **return** **null;**

**}**

canvasToDisable**.**gameObject**.**SetActive**(false);**

**}**

**}**

## Negocio/Juego/MenuOpciones/Ajustes

Este apartado agrupa las clases encargadas de gestionar las opciones de ajustes del juego, incluyendo la configuración del volumen, la opacidad del fondo, y la selección de idioma. Tendremos una clase específica para cada una de estas configuraciones.

### ControlVolumen

/\*\*

\* Clase VolumeSettings gestiona el volumen de los efectos de sonido y la música,

\* permitiendo al jugador ajustar los niveles y guardar sus preferencias.

\*

\* **@author** Carlos Checa Moreno

\*

\*/

**using** UnityEngine**;**

**using** UnityEngine**.**UI**;**

**public** class VolumeSettings **:** MonoBehaviour

**{**

**public** bool efectos**;**

**[**SerializeField**]** **private** AudioSource myAudioSource**;**

**[**SerializeField**]** **private** Slider musicSlider**;**

**[**SerializeField**]** **private** Image handleImage**;**

/\*\*

\* Método Start inicializa el volumen según las preferencias guardadas

\* y configura el listener del slider.

\*/

**private** void Start**()**

**{**

float volume **=** 1.0f**;**

**if** **(**efectos**)**

**{**

volume **=** PlayerPrefs**.**GetFloat**(**"volumenEfectos"**,** 1.0f**);**

**}**

**else**

**{**

volume **=** PlayerPrefs**.**GetFloat**(**"volumenMusica"**,** 1.0f**);**

**}**

musicSlider**.value** **=** volume**;**

myAudioSource**.**volume **=** volume**;**

musicSlider**.**onValueChanged**.**AddListener**(delegate** **{** SetMusicVolume**();** **});**

UpdateHandleImage**(**volume**);**

**}**

/\*\*

\* Método SetMusicVolume ajusta el volumen de la música o los efectos de sonido

\* dependiendo de la selección actual.

\*/

**public** void SetMusicVolume**()**

**{**

float volume **=** musicSlider**.value;**

myAudioSource**.**volume **=** volume**;**

UpdateHandleImage**(**volume**);**

**}**

/\*\*

\* Método GuardarPreferencias guarda el volumen de la música o los efectos de sonido

\* en las preferencias del jugador.

\*/

**public** void GuardarPreferencias**()**

**{**

**if** **(**efectos**)**

**{**

PlayerPrefs**.**SetFloat**(**"volumenEfectos"**,** musicSlider**.value);**

PlayerPrefs**.**Save**();**

**}**

**else**

**{**

PlayerPrefs**.**SetFloat**(**"volumenMusica"**,** musicSlider**.value);**

PlayerPrefs**.**Save**();**

**}**

**}**

/\*\*

\* Método UpdateHandleImage actualiza la imagen del control deslizante dependiendo

\* del nivel de volumen.

\*

\* **@param** volume Nivel de volumen

\*/

**private** void UpdateHandleImage**(**float volume**)**

**{**

**if** **(**volume **==** 0**)**

**{**

handleImage**.**sprite **=** efectos

**?** Resources**.**Load**<**Sprite**>(**"Botones\_ajustes/sonidoMuted"**)**

**:** Resources**.**Load**<**Sprite**>(**"Botones\_ajustes/musicMuted"**);**

**}**

**else**

**{**

handleImage**.**sprite **=** efectos

**?** Resources**.**Load**<**Sprite**>(**"Botones\_ajustes/sonidoOn"**)**

**:** Resources**.**Load**<**Sprite**>(**"Botones\_ajustes/musicOn"**);**

**}**

**}**

**}**

### FondoSettings

/\*\*

\* Clase FondoSettings gestiona la configuración del fondo de la aplicación,

\* permitiendo modificar la translucidez y guardar las preferencias del jugador.

\*

\* **@author** Carlos Checa Moreno

\*

\*/

**using** UnityEngine**;**

**using** UnityEngine**.**UI**;**

**public** class FondoSettings **:** MonoBehaviour

**{**

**[**SerializeField**]** **public** Material fondoMaterial**;** // Asigna el material del plano en el inspector.

**[**SerializeField**]** **public** Slider translucencySlider**;** // Slider para modificar la translucidez.

**[**SerializeField**]** **public** Image handleImage**;**

/\*\*

\* Método Start inicializa la translucidez del fondo según las preferencias guardadas

\* y configura el listener del slider.

\*/

**private** void Start**()**

**{**

float translucency **=** PlayerPrefs**.**GetFloat**(**"translucencyLevel"**,** 0.25f**);**

translucencySlider**.value** **=** translucency**;**

SetTranslucencyLevel**(**translucency**);**

translucencySlider**.**onValueChanged**.**AddListener**(delegate** **{** SetTranslucencyLevel**(**translucencySlider**.value);** **});**

UpdateHandleImage**(**translucency**);**

**}**

/\*\*

\* Método SetTranslucencyLevel ajusta el nivel de translucidez del fondo

\* y actualiza la imagen del control deslizante.

\*

\* **@param** value Valor de la translucidez

\*/

**public** void SetTranslucencyLevel**(**float **value)**

**{**

Color color **=** fondoMaterial**.**color**;**

color**.**a **=** **value;** // Modificar la transparencia del material.

fondoMaterial**.**color **=** color**;**

UpdateHandleImage**(value);**

**}**

/\*\*

\* Método GuardarPreferencias guarda el nivel de translucidez en las preferencias del jugador.

\*/

**public** void GuardarPreferencias**()**

**{**

PlayerPrefs**.**SetFloat**(**"translucencyLevel"**,** translucencySlider**.value);**

PlayerPrefs**.**Save**();**

**}**

/\*\*

\* Método UpdateHandleImage actualiza la imagen del control deslizante

\* dependiendo del nivel de translucidez.

\*

\* **@param** translucency Valor de la translucidez

\*/

**private** void UpdateHandleImage**(**float translucency**)**

**{**

**if** **(**translucency **==** 0**)**

**{**

handleImage**.**sprite **=** Resources**.**Load**<**Sprite**>(**"Botones\_ajustes/translucencyOff"**);**

**}**

**else**

**{**

handleImage**.**sprite **=** Resources**.**Load**<**Sprite**>(**"Botones\_ajustes/translucencyOn"**);**

**}**

**}**

**}**

### LocaleSelector

/\*\*

\* Clase LocaleSelector permite cambiar el idioma de la aplicación utilizando

\* el sistema de localización de Unity y guardando la selección en las preferencias del jugador.

\*

\* **@author** Carlos Checa Moreno

\*

\*/

**using** System**.**Collections**;**

**using** UnityEngine**;**

**using** UnityEngine**.**Localization**.**Settings**;**

**public** class LocaleSelector **:** MonoBehaviour

**{**

**private** bool active **=** **false;**

/\*\*

\* Método Start inicializa la selección de idioma según las preferencias guardadas

\* y aplica el idioma correspondiente.

\*/

void Start**()**

**{**

int savedLocaleID **=** PlayerPrefs**.**GetInt**(**"idioma"**,** 0**);**

ChangeLocale**(**savedLocaleID**);**

**}**

/\*\*

\* Método ChangeLocale cambia el idioma de la aplicación y guarda la selección en las preferencias.

\*

\* **@param** localeID ID del idioma seleccionado

\*/

**public** void ChangeLocale**(**int localeID**)**

**{**

**if** **(**active**)**

**{**

**return;**

**}**

StartCoroutine**(**SetLocale**(**localeID**));**

PlayerPrefs**.**SetInt**(**"idioma"**,** localeID**);**

PlayerPrefs**.**Save**();**

**}**

/\*\*

\* Método SetLocale cambia el idioma de la aplicación de manera asíncrona

\* para evitar bloqueos durante la inicialización.

\*

\* **@param** \_localeID ID del idioma a cambiar

\* **@return** IEnumerator para manejar el flujo asíncrono

\*/

**private** IEnumerator SetLocale**(**int \_localeID**)**

**{**

active **=** **true;**

**yield** **return** LocalizationSettings**.**InitializationOperation**;**

LocalizationSettings**.**SelectedLocale **=** LocalizationSettings**.**AvailableLocales**.**Locales**[**\_localeID**];**

active **=** **false;**

**}**

**}**

## Negocio/Juego/MenuOpciones/Biblioteca

En este apartado se describen las clases que gestionan la sección de la biblioteca, tenemos tanto el encargado de la pantalla de Aprendizaje como el de gestor de la pantalla de Progreso.

### AprendizajeManager

/\*\*

\* Clase AprendizajeManager gestiona la apertura de enlaces de aprendizaje

\* en el navegador predeterminado del sistema.

\*

\* **@author** Carlos Checa Moreno

\*

\*/

**using** System**.**Collections**;**

**using** System**.**Collections**.**Generic**;**

**using** UnityEngine**;**

**public** class AprendizajeManager **:** MonoBehaviour

**{**

/\*\*

\* Método OpenLink abre una URL en el navegador predeterminado si la URL no es nula o vacía.

\*

\* **@param** url La URL que se abrirá

\*/

**public** void OpenLink**(**string url**)**

**{**

**if** **(!**string**.**IsNullOrEmpty**(**url**))**

**{**

Application**.**OpenURL**(**url**);**

**}**

**else**

**{**

Debug**.**LogWarning**(**"La URL proporcionada está vacía o es nula."**);**

**}**

**}**

**}**

### BibliotecaManager

/\*\*

\* Clase BibliotecaManager gestiona la creación de paneles de la biblioteca

\* y la visualización de los detalles de los objetos celestes descubiertos.

\*

\* **@author** Carlos Checa Moreno

\*

\*/

**using** System**.**Collections**;**

**using** System**.**Collections**.**Generic**;**

**using** UnityEngine**;**

**using** UnityEngine**.**UI**;**

**using** TMPro**;**

**using** UnityEngine**.**Localization**.**Settings**;**

**public** class BibliotecaManager **:** MonoBehaviour

**{**

**private** TextMeshProUGUI titleText**;**

**private** TextMeshProUGUI descriptionText**;**

**private** TextMeshProUGUI fechaYDatosText**;**

**public** GameObject content**;**

**public** RawImage rawImageComponent**;**

**public** GameObject bibliotecaObject**;**

**public** GameObject bibliotecaDetallesObject**;**

**public** AudioSource efectoSonido**;**

/\*\*

\* Método Start inicializa los componentes de texto de la biblioteca

\* para mostrar los detalles de los objetos celestes.

\*/

void Start**()**

**{**

titleText **=** content**.**transform**.**Find**(**"Title"**).**GetComponent**<**TextMeshProUGUI**>();**

descriptionText **=** content**.**transform**.**Find**(**"Descripcion"**).**GetComponent**<**TextMeshProUGUI**>();**

fechaYDatosText **=** content**.**transform**.**Find**(**"FechaDatos"**).**GetComponent**<**TextMeshProUGUI**>();**

**}**

**public** GameObject panelPrefabBiblioteca**;** // Prefab del panel de biblioteca

**public** Transform parentElementBiblioteca**;** // Elemento padre bajo el cual se crearán los paneles

/\*\*

\* Método CreatePanelsBiblioteca crea los paneles de la biblioteca para cada

\* objeto celeste descubierto.

\*

\* **@return** IEnumerator para manejar la creación asíncrona

\*/

**public** IEnumerator CreatePanelsBiblioteca**()**

**{**

**foreach** **(**infoCuerpoProgreso objetoCielo **in** DatosEntreEscenas**.**playerProgress**.**discovered\_constellations**)**

**{**

**yield** **return** StartCoroutine**(**CrearPanelIndividual**(**objetoCielo**));**

**}**

**yield** **return** **null;**

**}**

/\*\*

\* Método CrearPanelIndividual crea un panel para un objeto celeste específico

\* y configura el botón de detalles.

\*

\* **@param** objetoCielo Datos del objeto celeste

\* **@return** IEnumerator para manejar la creación asíncrona del panel

\*/

**public** IEnumerator CrearPanelIndividual**(**infoCuerpoProgreso objetoCielo**)**

**{**

GameObject panel **=** Instantiate**(**panelPrefabBiblioteca**,** parentElementBiblioteca**);**

panel**.**GetComponentInChildren**<**Text**>().**text **=** objetoCielo**.**name**;**

var texture **=** Resources**.**Load**<**Texture2D**>(**"Cielo\_ilustraciones/" **+** objetoCielo**.**name**);**

RawImage rawImage **=** panel**.**GetComponentInChildren**<**RawImage**>();**

**if** **(**rawImage **!=** **null)**

**{**

**if** **(**texture **!=** **null)**

**{**

rawImage**.**texture **=** texture**;**

**}**

**else**

**{**

Debug**.**LogWarning**(**$"No se pudo cargar la imagen para {objetoCielo.name}"**);**

**}**

**}**

**else**

**{**

Debug**.**LogWarning**(**"No se encontró el componente RawImage en el panel prefab."**);**

**}**

Button button **=** panel**.**GetComponent**<**Button**>();**

**if** **(**button **==** **null)**

**{**

button **=** panel**.**AddComponent**<**Button**>();**

**}**

button**.**onClick**.**AddListener**(()** **=>**

**{**

efectoSonido**.**Play**();**

StartCoroutine**(**SetContent**(**objetoCielo**.**name**,** objetoCielo**.**location**,** objetoCielo**.**date**,** objetoCielo**.**azimuth**,** objetoCielo**.**altitude**));**

Texture2D newTexture **=** Resources**.**Load**<**Texture2D**>(**"Ilustraciones\_png/" **+** objetoCielo**.**name**);**

**if** **(**newTexture **!=** **null)**

**{**

rawImageComponent**.**texture **=** newTexture**;**

**}**

**else**

**{**

rawImageComponent**.**texture **=** Resources**.**Load**<**Texture2D**>(**"Ilustraciones\_png/noconstelacion"**);**

Debug**.**LogWarning**(**"Texture not found in Resources/Ilustraciones\_png/" **+** objetoCielo**.**name**);**

**}**

bibliotecaObject**.**SetActive**(false);**

bibliotecaDetallesObject**.**SetActive**(true);**

**});**

**yield** **return** **null;**

**}**

/\*\*

\* Método SetContent actualiza el contenido de la vista de detalles con la información

\* del objeto celeste seleccionado.

\*

\* **@param** title Título del objeto celeste

\* **@param** location Ubicación del objeto celeste

\* **@param** fecha Fecha de descubrimiento

\* **@param** azimuth Azimut del objeto celeste

\* **@param** altitude Altitud del objeto celeste

\* **@return** IEnumerator para manejar la actualización asíncrona

\*/

**public** IEnumerator SetContent**(**string title**,** string location**,** string fecha**,** float azimuth**,** float altitude**)**

**{**

**if** **(**titleText **!=** **null** **&&** descriptionText **!=** **null)**

**{**

titleText**.**text **=** title**;**

descriptionText**.**text **=** LocalizationSettings**.**StringDatabase**.**GetLocalizedString**(**"biblioteca"**,** title**);**

fechaYDatosText**.**text **=** location **+** " " **+** fecha **+** " Azimuth " **+** azimuth **+** " Altitude " **+** altitude**;**

**}**

**else**

**{**

Debug**.**LogError**(**"¡No se encontró el componente de texto de Título o Descripción!"**);**

**}**

**yield** **return** **null;**

**}**

**}**

## Negocio/Juego/MenuOpciones/Logros

Este punto aborda las clases que controlan el progreso y la visualización de los logros del jugador. Se encargan de actualizar los logros conseguidos, gestionar las notificaciones y mostrar el progreso en la pantalla de logros de Look Up.

### LogrosManager

/\*\*

\* Clase LogrosManager gestiona la creación de los paneles de logros conseguidos por el jugador,

\* mostrando la información correspondiente y cargando las imágenes asociadas.

\*

\* **@author** Carlos Checa Moreno

\*

\*/

**using** System**.**Collections**;**

**using** System**.**Collections**.**Generic**;**

**using** UnityEngine**;**

**using** UnityEngine**.**UI**;**

**using** TMPro**;**

**using** UnityEngine**.**Localization**.**Components**;**

**using** UnityEngine**.**Localization**.**Settings**;**

**public** class LogrosManager **:** MonoBehaviour

**{**

**public** GameObject panelPrefabBiblioteca**;** // Prefab del Panel

**public** Transform parentElementBiblioteca**;** // Elemento padre bajo el cual se crearán los paneles

/\*\*

\* Método CreatePanelsLogros crea paneles para cada logro conseguido por el jugador,

\* localizando y mostrando su nombre y descripción.

\*

\* **@return** IEnumerator para manejar la creación asíncrona de los paneles

\*/

**public** IEnumerator CreatePanelsLogros**()**

**{**

**foreach** **(**infoLogros logroObj **in** DatosEntreEscenas**.**playerLogros**.**listaLogros**)**

**{**

**if** **(!**logroObj**.**conseguido**)**

**continue;**

GameObject panel **=** Instantiate**(**panelPrefabBiblioteca**,** parentElementBiblioteca**);**

// Obtener los componentes LocalizeStringEvent en el panel

LocalizeStringEvent**[]** localizeStringEvents **=** panel**.**GetComponentsInChildren**<**LocalizeStringEvent**>();**

**if** **(**localizeStringEvents**.**Length **>=** 2**)**

**{**

// Configurar la referencia de la tabla y la clave de localización para el nombre del logro

localizeStringEvents**[**0**].**StringReference**.**TableReference **=** "logros"**;**

localizeStringEvents**[**0**].**StringReference**.**TableEntryReference **=** "nom" **+** logroObj**.**name**;**

// Configurar la referencia de la tabla y la clave de localización para más información del logro

localizeStringEvents**[**1**].**StringReference**.**TableReference **=** "logros"**;**

localizeStringEvents**[**1**].**StringReference**.**TableEntryReference **=** logroObj**.**name**;**

**}**

**else**

**{**

Debug**.**LogError**(**"No se encontraron suficientes componentes LocalizeStringEvent en los hijos."**);**

**}**

// Configurar la imagen del logro

var texture **=** Resources**.**Load**<**Texture2D**>(**"Logros/" **+** logroObj**.**name**);**

RawImage rawImage **=** panel**.**GetComponentInChildren**<**RawImage**>();**

**if** **(**rawImage **!=** **null)**

**{**

**if** **(**texture **!=** **null)**

**{**

rawImage**.**texture **=** texture**;**

**}**

**else**

**{**

Debug**.**LogWarning**(**$"No se pudo cargar la imagen para {logroObj.name}"**);**

**}**

**}**

**else**

**{**

Debug**.**LogWarning**(**"No se encontró el componente RawImage en el panel prefab."**);**

**}**

Button buttonComponent **=** panel**.**GetComponent**<**Button**>();**

**if** **(**buttonComponent **!=** **null)**

**{**

Destroy**(**buttonComponent**);**

**}**

**}**

**yield** **return** **null;**

**}**

**}**

### ProgresoLogrosManager

/\*\*

\* Clase ProgresoLogrosManager controla el progreso de los logros del jugador,

\* actualizando los logros alcanzados y mostrando notificaciones y paneles de logros.

\*

\* **@author** Carlos Checa Moreno

\*

\*/

**using** System**.**Collections**;**

**using** System**.**Collections**.**Generic**;**

**using** UnityEngine**;**

**using** TMPro**;**

**using** UnityEngine**.**UI**;**

**using** UnityEngine**.**Localization**.**Settings**;**

**public** class ProgresoLogrosManager **:** MonoBehaviour

**{**

**public** Web webScript**;**

**private** GameObject panelPrefabBiblioteca**;**

**private** Transform parentElementBiblioteca**;**

/\*\*

\* Método Start inicializa las referencias necesarias para gestionar el progreso de los logros,

\* incluyendo el script Web y los objetos de la interfaz de usuario.

\*/

void Start**()**

**{**

GameObject mainObject **=** GameObject**.**Find**(**"Main"**);**

**if** **(**mainObject **!=** **null)**

**{**

webScript **=** mainObject**.**GetComponent**<**Web**>();**

**if** **(**webScript **==** **null)**

**{**

Debug**.**LogError**(**"No se encontró el componente Web en el objeto Main."**);**

**}**

**}**

**else**

**{**

Debug**.**LogError**(**"No se encontró el objeto Main en la escena."**);**

**}**

DatosEntreEscenas**.**logros **=** GameObject**.**Find**(**"Logros"**);**

panelPrefabBiblioteca **=** Resources**.**Load**<**GameObject**>(**"Prefabs/PanelPrefab"**);**

**if** **(**DatosEntreEscenas**.**logros **!=** **null)**

**{**

parentElementBiblioteca **=** DatosEntreEscenas**.**logros**.**transform**.**Find**(**"Scroll View/Viewport/Content"**);**

**if** **(**parentElementBiblioteca **==** **null)**

**{**

Debug**.**LogError**(**"Content Transform not found!"**);**

**}**

**}**

**else**

**{**

Debug**.**LogError**(**"Logros object not found!"**);**

**}**

**}**

static **public** Dictionary**<**string**,** int**>** valoresParaConseguirLogro **=** **new** Dictionary**<**string**,** int**>()**

**{**

**{** "biblio1"**,** 5 **},**

**{** "constdesc1"**,** 10 **},**

**{** "planet1"**,** 1 **}**

**};**

/\*\*

\* Método IncrementarProgreso incrementa el progreso de un logro específico y actualiza la interfaz

\* si el logro ha sido conseguido.

\*

\* **@param** nombreLogro Nombre del logro a actualizar

\*/

**public** void IncrementarProgreso**(**string nombreLogro**)**

**{**

bool logroConseguido **=** **false;**

StartCoroutine**(**webScript**.**UpdateLogroProgreso**(**nombreLogro**,** **(**success**)** **=>**

**{**

logroConseguido **=** success**;**

**}));**

**if** **(**DatosEntreEscenas**.**conseguido**)**

**{**

UpdateNotificacionLogro**(**nombreLogro**);**

NotificarLogro**(**nombreLogro**);**

CreatePanelsLogroNuevo**(**nombreLogro**);**

DatosEntreEscenas**.**conseguido **=** **false;**

**}**

**}**

/\*\*

\* Método NotificarLogro muestra una notificación visual en la interfaz

\* cuando el jugador consigue un logro.

\*

\* **@param** nombreLogro Nombre del logro conseguido

\*/

**public** void NotificarLogro**(**string nombreLogro**)**

**{**

**if** **(**DatosEntreEscenas**.**notificacionLogros **!=** **null)**

**{**

DatosEntreEscenas**.**notificacionLogros**.**SetActive**(true);**

**}**

**else**

**{**

Debug**.**LogWarning**(**"notificacionLogros es null."**);**

**}**

Debug**.**Log**(**"Notificación de logro instanciada: " **+** nombreLogro**);**

**}**

/\*\*

\* Método UpdateNotificacionLogro actualiza el contenido de la notificación

\* de logros con el nombre y la imagen del logro conseguido.

\*

\* **@param** logroName Nombre del logro conseguido

\*/

**public** void UpdateNotificacionLogro**(**string logroName**)**

**{**

GameObject notificationObj **=** DatosEntreEscenas**.**notificacionLogros**;**

**if** **(**notificationObj **==** **null)**

**{**

Debug**.**LogError**(**"No se encontró el objeto GONotificacionLogro en DatosEntreEscenas."**);**

**return;**

**}**

Transform panelTransform **=** notificationObj**.**transform**.**Find**(**"NotificacionLogro/Panel"**);**

**if** **(**panelTransform **==** **null)**

**{**

Debug**.**LogError**(**"No se encontró el Panel en GONotificacionLogro."**);**

**return;**

**}**

TextMeshProUGUI textNombreLogro **=** panelTransform**.**Find**(**"TextNombreLogro"**).**GetComponent**<**TextMeshProUGUI**>();**

**if** **(**textNombreLogro **!=** **null)**

**{**

textNombreLogro**.**text **=** LocalizationSettings**.**StringDatabase**.**GetLocalizedString**(**"logros"**,** "nom" **+** logroName**);**

**}**

**else**

**{**

Debug**.**LogError**(**"No se encontró el componente TextMeshProUGUI en TextNombreLogro."**);**

**}**

RawImage rawImage **=** panelTransform**.**Find**(**"Image"**).**GetComponent**<**RawImage**>();**

**if** **(**rawImage **!=** **null)**

**{**

var texture **=** Resources**.**Load**<**Texture2D**>(**"Logros/" **+** logroName**);**

**if** **(**texture **!=** **null)**

**{**

rawImage**.**texture **=** texture**;**

**}**

**else**

**{**

Debug**.**LogWarning**(**$"No se pudo cargar la imagen para {logroName}."**);**

**}**

**}**

**else**

**{**

Debug**.**LogError**(**"No se encontró el componente RawImage en el panel."**);**

**}**

**}**

/\*\*

\* Método CreatePanelsLogroNuevo crea un nuevo panel en la interfaz de logros

\* cuando el jugador consigue un nuevo logro.

\*

\* **@param** logroName Nombre del nuevo logro conseguido

\*/

**public** void CreatePanelsLogroNuevo**(**string logroName**)**

**{**

GameObject panel **=** Instantiate**(**panelPrefabBiblioteca**,** parentElementBiblioteca**);**

Text**[]** textsLogros **=** panel**.**GetComponentsInChildren**<**Text**>();**

**if** **(**textsLogros**.**Length **>** 1**)**

**{**

textsLogros**[**0**].**text **=** LocalizationSettings**.**StringDatabase**.**GetLocalizedString**(**"logros"**,** "nom" **+** logroName**);**

textsLogros**[**1**].**text **=** LocalizationSettings**.**StringDatabase**.**GetLocalizedString**(**"logros"**,** logroName**);**

**}**

**else**

**{**

Debug**.**LogError**(**"No se encontraron suficientes componentes Text en los hijos."**);**

**}**

var texture **=** Resources**.**Load**<**Texture2D**>(**"Logros/" **+** logroName**);**

RawImage rawImage **=** panel**.**GetComponentInChildren**<**RawImage**>();**

**if** **(**rawImage **!=** **null)**

**{**

**if** **(**texture **!=** **null)**

**{**

rawImage**.**texture **=** texture**;**

**}**

**else**

**{**

Debug**.**LogWarning**(**$"No se pudo cargar la imagen para {logroName}"**);**

**}**

**}**

**else**

**{**

Debug**.**LogWarning**(**"No se encontró el componente RawImage en el panel prefab."**);**

**}**

**}**

**}**

## Negocio/Juego/MenuOpciones/Ubicación

Finalmente, este apartado cubre la única clase que permite al jugador buscar y seleccionar una ubicación. Las clase maneja la búsqueda de ciudades y la actualización de la ubicación del jugador dentro del sistema.

### UbicacionManager

/\*\*

\* Clase UbicacionManager gestiona la selección de la ubicación del jugador,

\* permitiendo buscar ciudades, mostrar resultados y actualizar la ubicación en la aplicación.

\*

\* **@author** Carlos Checa Moreno

\*

\*/

**using** System**.**Collections**;**

**using** System**.**Collections**.**Generic**;**

**using** UnityEngine**;**

**using** TMPro**;**

**using** UnityEngine**.**UI**;**

**using** System**.**Linq**;**

**using** UnityEngine**.**Localization**;**

**public** class UbicacionManager **:** MonoBehaviour

**{**

**public** Button botonSi**;**

**public** Button botonNo**;**

**public** GameObject textoPrefabCiudades**;** // Prefab del Panel

**public** Transform parentElementCiudades**;** // Elemento padre bajo el cual se crearán los paneles

**public** TextMeshProUGUI ubicacionText**;**

**public** GameObject textoPlaceHolder**;**

**private** List**<**CiudadData**>** listaCiudades **=** **new** List**<**CiudadData**>();**

/\*\*

\* Método Start inicializa el texto de la ubicación actual del jugador

\* con los datos almacenados.

\*/

void Start**()**

**{**

ubicacionText**.**text **=** DatosEntreEscenas**.**ubicacion**;**

**}**

/\*\*

\* Método CreatePanelsCiudades carga una lista de ciudades desde un archivo CSV

\* y crea paneles para cada ciudad utilizando un prefab.

\*

\* **@return** IEnumerator para manejar la creación asíncrona de los paneles

\*/

**public** IEnumerator CreatePanelsCiudades**()**

**{**

TextAsset csvData **=** Resources**.**Load**<**TextAsset**>(**"Datos/worldcitiesLookUp"**);**

**if** **(**csvData **==** **null)**

**{**

Debug**.**LogError**(**"No se pudo encontrar el archivo CSV en la carpeta Resources."**);**

**yield** **return** **null;**

**}**

string**[]** rows **=** csvData**.**text**.**Split**(new** char**[]** **{** '\n' **});**

**for** **(**int i **=** 1**;** i **<** rows**.**Length **-** 1**;** i**++)**

**{**

string row **=** rows**[**i**];**

string**[]** columns **=** row**.**Split**(new** char**[]** **{** ';' **});**

**if** **(**columns**.**Length **==** 4**)**

**{**

string city **=** columns**[**0**].**Trim**();**

float latItFloat **=** float**.**Parse**(**columns**[**1**].**Trim**());**

float lngItFloat **=** float**.**Parse**(**columns**[**2**].**Trim**());**

string pais **=** columns**[**3**].**Trim**();**

listaCiudades**.**Add**(new** CiudadData **{** name **=** city**,** country **=** pais**,** lat **=** latItFloat**,** lng **=** lngItFloat **});**

**}**

**else**

**{**

Debug**.**LogWarning**(**$"La línea {i} no tiene el número correcto de columnas: {row}"**);**

**}**

**}**

**yield** **return** **null;**

**}**

**public** TMP\_InputField ciudadBuscar**;**

**private** List**<**CiudadData**>** ciudadesFiltradas **=** **new** List**<**CiudadData**>();**

**public** GameObject panelMsgConfirmacion**;**

**public** TextMeshProUGUI textoConfirmacion**;**

/\*\*

\* Método BuscarCiudad filtra la lista de ciudades según el texto ingresado por el jugador

\* y crea paneles para las ciudades que coinciden con la búsqueda.

\*/

**public** void BuscarCiudad**()**

**{**

string busqueda **=** ciudadBuscar**.**text**;**

ciudadesFiltradas **=** listaCiudades

**.**Where**(**c **=>** c**.**name**.**StartsWith**(**busqueda**,** System**.**StringComparison**.**OrdinalIgnoreCase**))**

**.**ToList**();**

**foreach** **(**Transform child **in** parentElementCiudades**)**

**{**

Destroy**(**child**.**gameObject**);**

**}**

**foreach** **(**var ciudad **in** ciudadesFiltradas**)**

**{**

GameObject instantiatedObject **=** Instantiate**(**textoPrefabCiudades**,** parentElementCiudades**);**

TextMeshProUGUI textMeshPro **=** instantiatedObject**.**GetComponentInChildren**<**TextMeshProUGUI**>();**

**if** **(**textMeshPro **!=** **null)**

**{**

textMeshPro**.**text **=** $"{ciudad.name}, {ciudad.country}, Lat: {ciudad.lat}, Lng: {ciudad.lng}"**;**

**}**

**else**

**{**

Debug**.**LogError**(**"No se encontró un componente TextMeshProUGUI en el objeto instanciado."**);**

**}**

Button button **=** instantiatedObject**.**GetComponentInChildren**<**Button**>();**

**if** **(**button **!=** **null)**

**{**

button**.**onClick**.**AddListener**(()** **=>**

**{**

nameObjetivo **=** ciudad**.**name**;**

latitudObjetivo **=** ciudad**.**lat**;**

longitudObjetivo **=** ciudad**.**lng**;**

panelMsgConfirmacion**.**SetActive**(true);**

textoConfirmacion**.**text **=** $"{ciudad.name}, {ciudad.country}?"**;**

**});**

**}**

**else**

**{**

Debug**.**LogError**(**"No se encontró un componente Button en el objeto instanciado."**);**

**}**

**}**

**}**

**private** string nameObjetivo **=** "linares"**;**

**private** float longitudObjetivo **=** **-**3.6445510f**;**

**private** float latitudObjetivo **=** 38.0973414f**;**

/\*\*

\* Método CambiarDatosUbicacion actualiza los datos de ubicación del jugador

\* según la ciudad seleccionada.

\*/

**public** void CambiarDatosUbicacion**()**

**{**

DatosEntreEscenas**.**ubicacion **=** nameObjetivo**;**

ubicacionText**.**text **=** nameObjetivo**;**

DatosEntreEscenas**.**longitud **=** longitudObjetivo**;**

DatosEntreEscenas**.**latitud **=** latitudObjetivo**;**

**}**

**}**