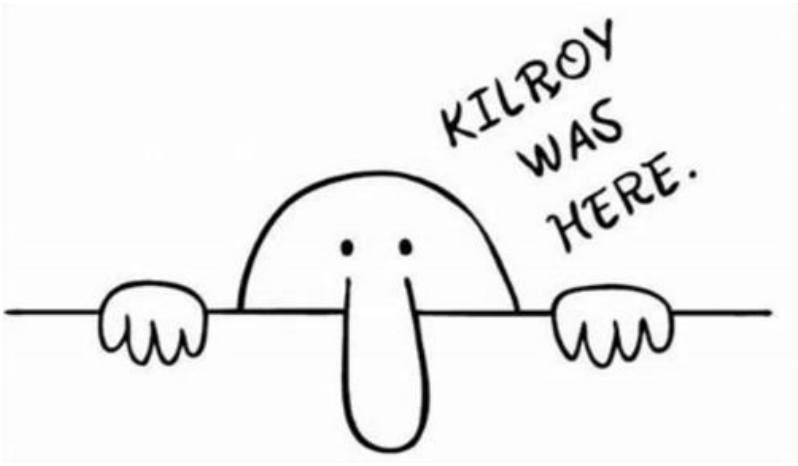
4 DE DICIEMBRE DE 2022



KILROY WAS HERE

PROYECTO JAVA MVC

ALBERTO GARCÍA NAVARRO
HORAS DE LIBRE CONFIGURACIÓN
2º CFGS DAW
Curso 2022-2023

ÍNDICE

A) Descripción del proyecto……pág. 2

A.1) Introducción……..pág. 2

A.2) Descripción del funcionamiento de la aplicación……..pág. 2

A.3) Proyección de futuro……pág. 2

B) Base de datos……..pág. 3

B.1) Descripción……..pág. 3

B.2) Esquema……pág. 4

C) Especificaciones……pág. 4

D) Objetivos de la rúbrica……pág. 5

A) Descripción del proyecto.

A.1) Introducción.

El proyecto surge a partir de una referencia cultural de un grafiti muy popular históricamente y que a día de hoy ha aparecido en multitud de películas y videojuegos. **Kilroy was here** da nombre e imagen a nuestra aplicación, explicando el objetivo principal de la misma.

La idea es crear una aplicación que guarde determinados puntos por geolocalización y, a medida que te acercas, te va dando pistas sobre la localización del tesoro. Al encontrarlo permite guardar un mensaje para el resto de usuarios como una firma personal, haciendo referencia a **Kilroy was here**.

A.2) Descripción del funcionamiento de la aplicación.

La aplicación comienza dándote la bienvenida. Te da la opción de iniciar sesión, consultando las credenciales en nuestra base de datos, o la opción de crear una cuenta, que validará que los campos son correctos para evitar “sql injection”.

Al hacer el login, podremos jugar e interactuar con la aplicación. Si pulsamos la opción de “buscar tesoro” la aplicación generará una coordenada aleatoria simulando la geolocalización por Google Maps. Si esta localización coincide con la de un tesoro, se le notifica al usuario de la cercanía del tesoro. Hay 3 tipos de tesoros que harán diferentes acciones:

- **Reliquia**: Es el tesoro básico, si el usuario lo encuentra se le da automáticamente.
- **Pista**: Concede una pista informativa de la localización de un tesoro.
- **Pieza**: Requiere reunir un número total de piezas para obtener un tesoro.

Una vez encontrado el tesoro (en la aplicación se simula, se quiere implementar radio de búsqueda con Google Maps) la aplicación guarda en la base de datos el tesoro junto al usuario que lo ha encontrado. A esto se le añade la firma del usuario.
La firma del usuario es un fichero txt que guarda en local un mensaje predefinido por el usuario. Así, cada vez que se encuentre un tesoro, el usuario no necesita reescribir su firma, si no que hará el proceso automáticamente.

Se tiene en consideración los tesoros que ya ha descubierto el usuario, para ello se guarda en una lista obtenida de la base de datos. Si el usuario busca un tesoro que ya ha descubierto, se le ofrecerá la opción de actualizar su firma con la actual.

Además de buscar tesoros, el usuario tendrá la opción de consultar los que ha descubierto (aparecerá el nombre del tesoro, su descripción y la firma), y también podrá modificar el fichero de la firma predefinida.

A.3) Proyección de futuro.

De esta forma, se busca una aplicación que el usuario pueda usar en su dispositivo móvil durante sus recorridos. Cuando se desee, se pulsará el botón de buscar y se le notificará al usuario cuando haya un tesoro cerca. Dependiendo del tipo de tesoro, el usuario podrá obtener el premio al encontrarlo o hacer una búsqueda más exhaustiva con las indicaciones del programa.

El logro de haber encontrado un tesoro será guardado con una firma, y cualquier otro usuario que encuentre ese tesoro podrá ver un mural con las firmas que han ido dejando el resto. Creando así una comunidad activa de buscadores de tesoros que vayan añadiendo tesoros y pistas por todo el mundo.

B) Base de datos.

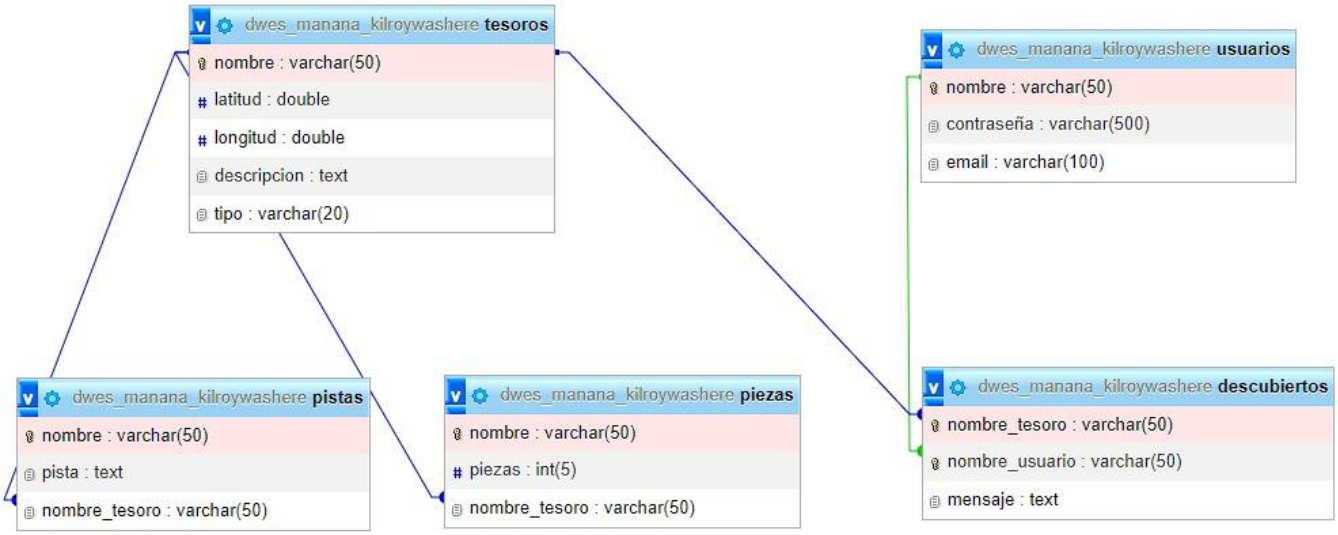
B.1) Descripción.

La base de datos para la gestión de la aplicación cuenta con 5 tablas que se detallarán a continuación:

- **Usuarios**: Datos de los usuarios que se van a registrar en nuestra aplicación.
    - Nombre: Nombre/Apodo que el usuario va a establecer. Será único por cada usuario.
    - Contraseña: Contraseña del usuario que deberá cumplir unos requisitos. Se guardará esta información encriptada en formato Hash MD5 para su seguridad.
    - Email: Correo electrónico del usuario, se usará para notificarle de cualquier novedad o incidente.
- **Tesoros**: Datos de los tesoros junto a su localización para buscarlos.
    - Nombre: Nombre que tendrá el tesoro. Será único e identificativo de cada uno.
    - Latitud: Coordenada de latitud donde se encuentra el tesoro.
    - Longitud: Coordenada de longitud donde se encuentra el tesoro.
    - Descripción: Mensaje descriptivo del tesoro.
    - Tipo: Permitirá clasificar el tipo de tesoro que es para establecer diferentes formas de obtención. Podrá ser Reliquia, Pista o Pieza.
- **Pistas**: Pista asociada a un tesoro, contendrá un mensaje que orientará al usuario de su localización.
    - Nombre: Nombre que tendrá la pista. Será única e identificativa de cada una.
    - Pista: Mensaje con la pista que ayudará al usuario a encontrar el tesoro.
    - Nombre Tesoro: Nombre del tesoro al que va asociado la pista.
- **Piezas**: Pieza asociada a un tesoro, al conseguir varias piezas obtendremos el tesoro.
    - Nombre: Nombre que tendrá la pieza. Será única e identificativa de cada una.
    - Piezas: Número de piezas que habrá que conseguir para obtener el tesoro.
    - Nombre Tesoro: Nombre del tesoro al que va asociado la pieza.
- **Descubiertos**: Registro de los tesoros que han descubiertos los usuarios. Cada combinación tesoro-usuario será única.
    - Nombre Tesoro: Nombre del tesoro que ha sido descubierto. Va asociado a la tabla tesoros.
    - Nombre Usuario: Nombre del usuario que ha descubierto el tesoro. Va asociado a la tabla usuarios.
    - Mensaje: Firma que el usuario hace al descubrir el tesoro.

B.2) Esquema.

A continuación, mostramos un esquema aclarativo de la estructura de la base de datos va a utilizar la aplicación:



C) Especificaciones.

El proyecto se ha creado con las siguientes tecnologías:

- **Sistema operativo**: Windows 10.
- **IDE**: Apache NetBeans IDE 12.5.
- **Lenguaje**: Java 17.0.1.
- **Project Management Framework**: Maven 3.6.3.
- **Compilador**: Maven JDK 17**.**
- **Conector Base de Datos (JDBC Driver)**: MySQL Connector Java 8.0.31.
- **Base de datos**: MySQL 8.1.6 gestionado en phpMyAdmin 5.2.0.
- **Documentación**: JavaDoc generado por el propio IDE.
- **Control de calidad**: El proyecto ha sido analizado por SonarLint, acorde con los estándares establecidos.

D) Objetivos de la rúbrica.

Se detallan a continuación los objetivos de la rúbrica conforme al proyecto:

1. **Base de datos.**
    a. Creación de tablas necesarias para la gestión de nuestra base de datos.
    b. Sentencias Select (Consulta de la existencia del usuario), Insert(Insertar nuevos usuarios), Update(Actualizar la firma del usuario)… Que vincula nuestra aplicación con la base de datos.
    c. SQLExceptions claramente recogidas y tratadas, cerrando siempre el flujo de la aplicación e informando al usuario del error.
2. **Ficheros.**
    a. Si se borra por error el fichero, siempre se crea uno, evitando así errores en su ejecución.
    b. IOExceptions y FileNotFoundExceptions claramente recogidas y tratadas, cerrando siempre el flujo del fichero e informando al usuario del error.
    c. Se accede al fichero de texto para consultar la firma del usuario y para modificarla.
3. **Herencia y polimorfismo.**
    a. Se crea una clase abstracta Tesoro, del que va a heredar clases hijas según el tipo de tesoro, cada una implementando de una forma distinta la acción al buscar un tesoro.
    b. Polimorfismo llamando a la superclase Tesoro e instanciado por sus clases hijas.
4. **Constructores**
    a. Constructor por defecto y parametrizado para poder instanciar todas las clases, salvo las clases que sean de utilidad con todo estático (no permitiremos que sean instanciadas).
    b. Uso del “this” en la clase Usuario para instanciar un usuario según los diferentes parámetros que le pasemos (si queremos hacer un login, solo le pasaremos el nombre y la contraseña, pero si creamos una cuenta, le pasaremos además el email).
5. **Collections**: Creación de listas (List - ArrayList) para guardar los tesoros descubiertos por el usuario.
6. **Patrón MVC**: Se separan los datos (Las tablas que hemos creado en nuestra base de datos), la lógica (los gestores que realizan las acciones para obtener los datos de la base de datos y mostrarlos) y la visualización (en nuestro caso, por consola, para futura implementación en interfaz de móvil).
7. **Gestión de excepciones:**
    a. Se usan try-catch y throws para tratar las excepciones de nuestro programa.
    b. Guardamos la información de las excepciones y su traza en un fichero .tmp, que será borrado al cerrar la aplicación.
    c. Mostramos al usuario por consola la excepción producida.
8. **Visualización de la información**: Usamos la consola como interfaz mostrando un menú para que el usuario pueda interactuar con la aplicación. La finalidad es hacer una interfaz final al trasladar este proyecto a nuestro dispositivo móvil.