

POKEDEX

Metodología de la Programación



Ingeniería de Sistemas de la Información
CEU | Universidad
San Pablo

CEU San Pablo

Escuela Politécnica Superior

Javier Linares Castrillón

Andrés Iturria Soler

Samuel Aragonés Lozano

Madrid, Diciembre de 2019.



Índice

Introducción	3
Diseño del Sistema	4
Diseño de la Base de Datos	7
Descripción de la UI	8
Observaciones y recursos utilizados	12



Introducción

El objetivo de la práctica es desarrollar una aplicación con interfaz gráfica que consulte una base de datos SQL. En este caso, la práctica está basada en una Pokédex.

“Se deben de implementar, al menos, dos patrones de diseño”. En este caso, se ha utilizado el patrón DAO y el patrón Observer.

- **Patrón DAO:** utilizado para desacoplar la aplicación de la base de datos. Se ha creado una clase *Conexión* encargada de interactuar entre la base de datos y la aplicación.
- **Patrón Observer:** para asegurar que, tras una acción del cliente con la aplicación, los componentes pertinentes se actualicen, se hace uso de este patrón. En este caso, se aplica por medio de una clase privada *Oyente* presente en varias clases. Cuando el usuario hace una acción determinada, *Oyente* se encarga de actualizar el estado de todos los objetos involucrados.

“Deben de haber, como mínimo, tres ventanas distintas con las que pueda interactuar el usuario”. Por ello, para no escatimar, este programa cuenta con 4 frames distintos, varios paneles diferentes asociados a un *tabbe panel*, dos ventanas modales, incontables botones distintos y mucho más.

“Esta aplicación debe consultar una base de datos de al menos tres tablas”. Se ha utilizado la base de datos **Pokedex.db** desarrollada en la asignatura de Base de Datos. El número de tablas distintas de ésta es de 20. No se redujo el número de tablas puesto que no se sabía en un principio cuáles se iban a utilizar. El grupo ha decidido dejarla así para futuras ampliaciones de la aplicación. Todos los datos mostrados por la aplicación, a excepción de las imágenes, se encuentran en la base de datos.

Diseño del Sistema

A priori, el diseño de pantallas original era algo más extenso de lo que al final ha terminado siendo. Esto no quiere decir que la empresa no haya sido capaz de cumplir las expectativas iniciales, simplemente ha tomado la decisión para sustituir trabajo de mono por innovación y reto.

La maqueta original (*Imagen 1.1, 1.2, 1.3, 1.4, 1.5*) del diseño de pantallas era extensiva. Contaba con 5 Ventanas (paneles), pero la funcionalidad de alguna de estas era similar a las otras. Por ello, se ha decidido eliminar la de gimnasios y equipo. En vez de eso, se han añadido otras ventanas como la ventana de Stats o la de Ataques de un Pokemon.



Pokemons	Mapa	Gimnasios	Habilidades	Equipo	
Lista de Pokemons	<input type="text"/> Nombre <input type="text"/> Clasificación <input type="text"/> Habilidad <input type="text"/> Peso/Altura <input type="text"/> Naturaleza <input type="text"/> Lugar	Descripción			



Imagen 1.1 - Ventana 1 – Pokemons

Pokemons	Mapa	Gimnasios	Movimientos	Habilidades	
Ciudades y lugares de interés.	Imagen de la ciudad o lugar seleccionado	Mapa general de la región			



Imagen 1.2 - Ventana 2 - Localizaciones

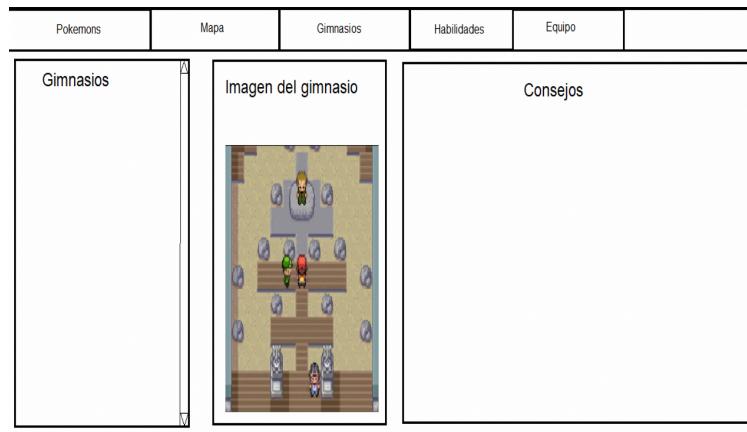


Imagen 1.3 - Ventana 3 – Gimnasios

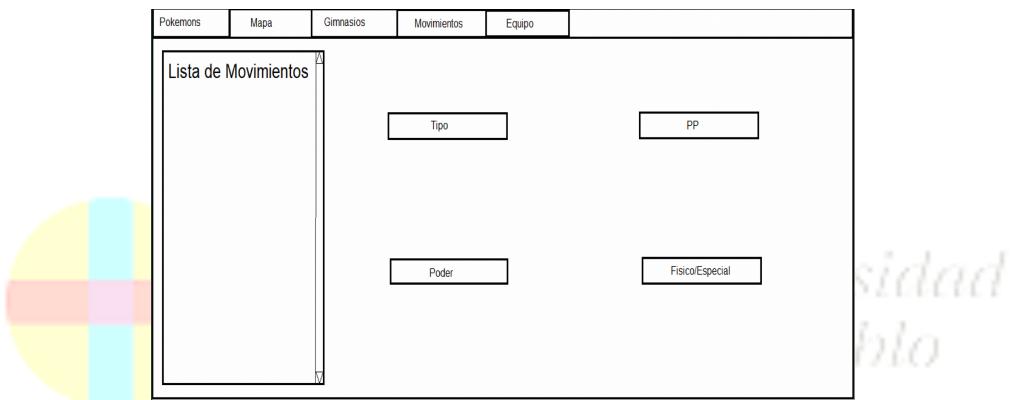


Imagen 1.4 - Ventana 4 – Movimientos

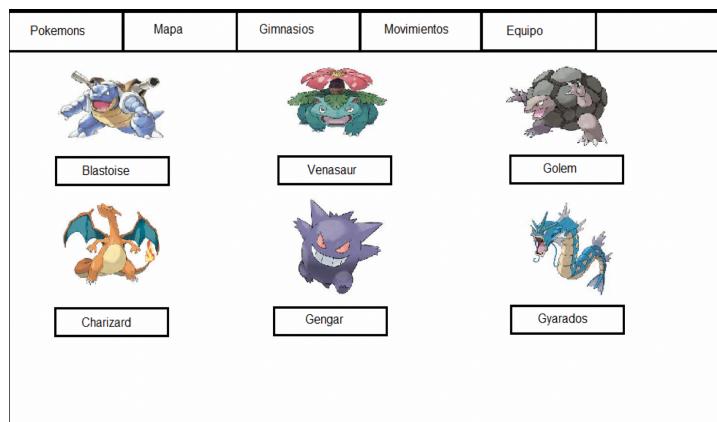


Imagen 1.5 – Ventana 5 – Equipo

Se ha tratado de aplicar el patrón Singleton a algunas ventanas, puesto que es de interés que el usuario solo sea capaz de generar dos de esas ventanas a la vez. Pero, puesto que cada frame tenía unas características distintas, se ha terminado utilizando una serie de métodos privados que se encargan de gestionar, simplemente, que no existan dos ventanas del mismo tipo a la vez.

En la *Imagen 2.1* se muestra el diagrama UML de la versión final de la aplicación.

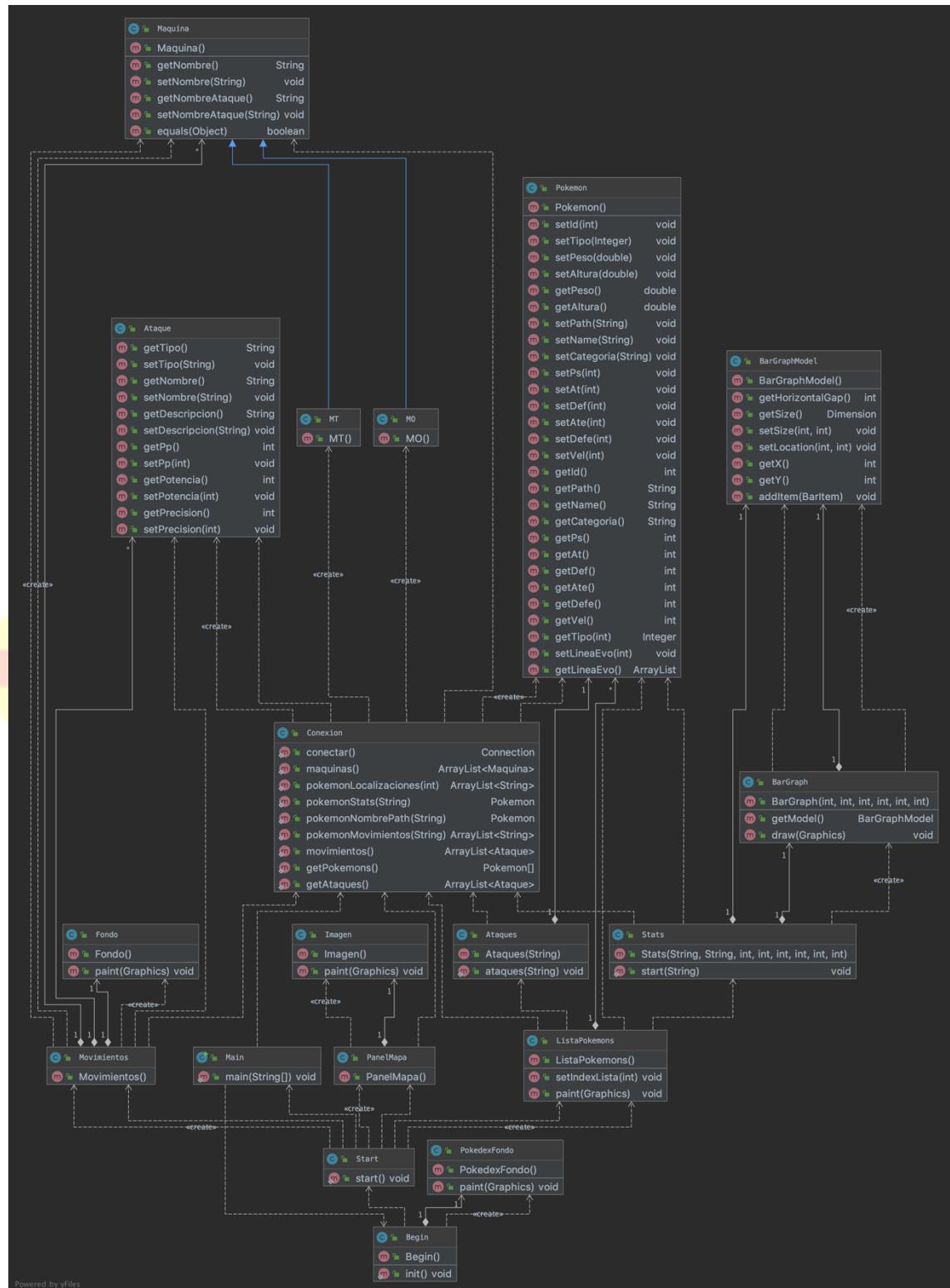


Imagen 2.1 – Diagrama UML – clases y métodos públicos

Diseño de la Base de Datos

La *Imagen 3.1* muestra el diseño de la base de datos. Las tablas utilizadas se numerarán a continuación:

1. **Pokemon:** contiene todos los pokemons.
2. **LíneaEvo:** contiene la línea evolutiva de cada Pokemon.
3. **Localización:** contiene todas las localizaciones.
4. **PokemonLocalización:** contiene la localización de cada pokemon.
5. **Movimiento:** contiene todos los movimientos que puede aprender un pokemon.
6. **PokemonMovimientoForma:** almacena los movimientos que puede aprender un pokemon y cómo lo aprende.
7. **FormaAprendizaje:** contiene la forma en la que un pokemon aprende un movimiento.
8. **MT:** almacena el nombre de las Máquinas técnicas que pueden enseñar un movimiento X a un pokemon.
9. **MO:** almacena el nombre de las Máquinas ocultas que pueden enseñar un movimiento M X un pokemon.
10. **Tipo:** contiene todos los tipos que pueden ser los movimientos y los pokemons.
11. **PokemonTipo:** contiene los tipos que puede ser un mismo Pokemon.

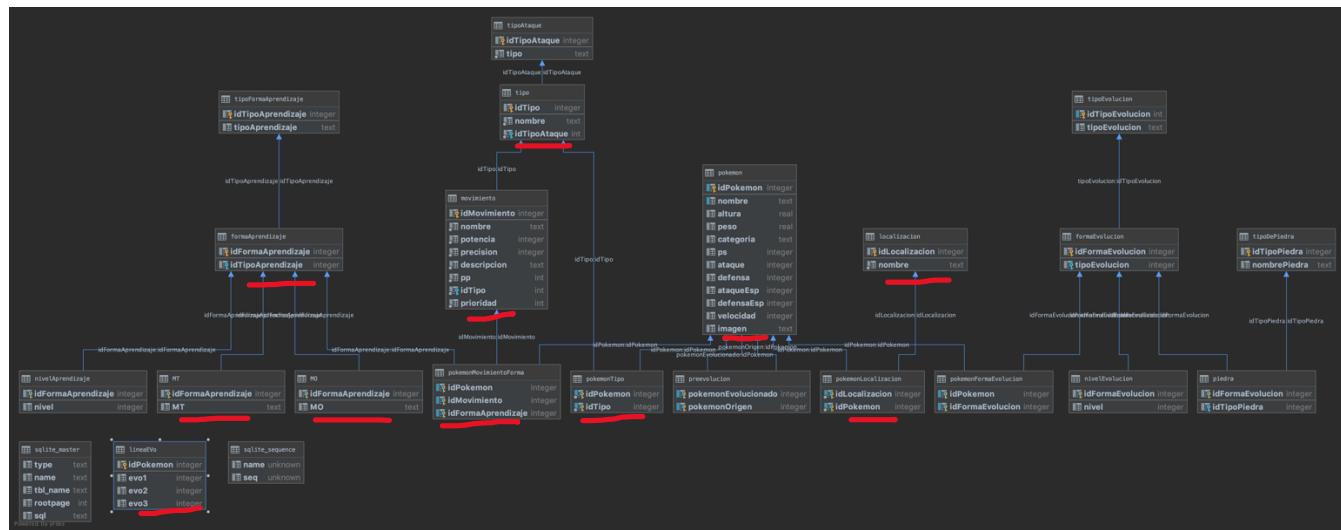
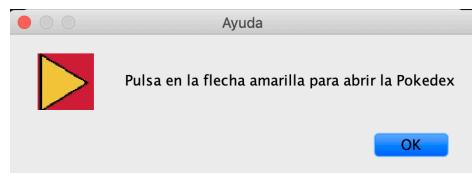
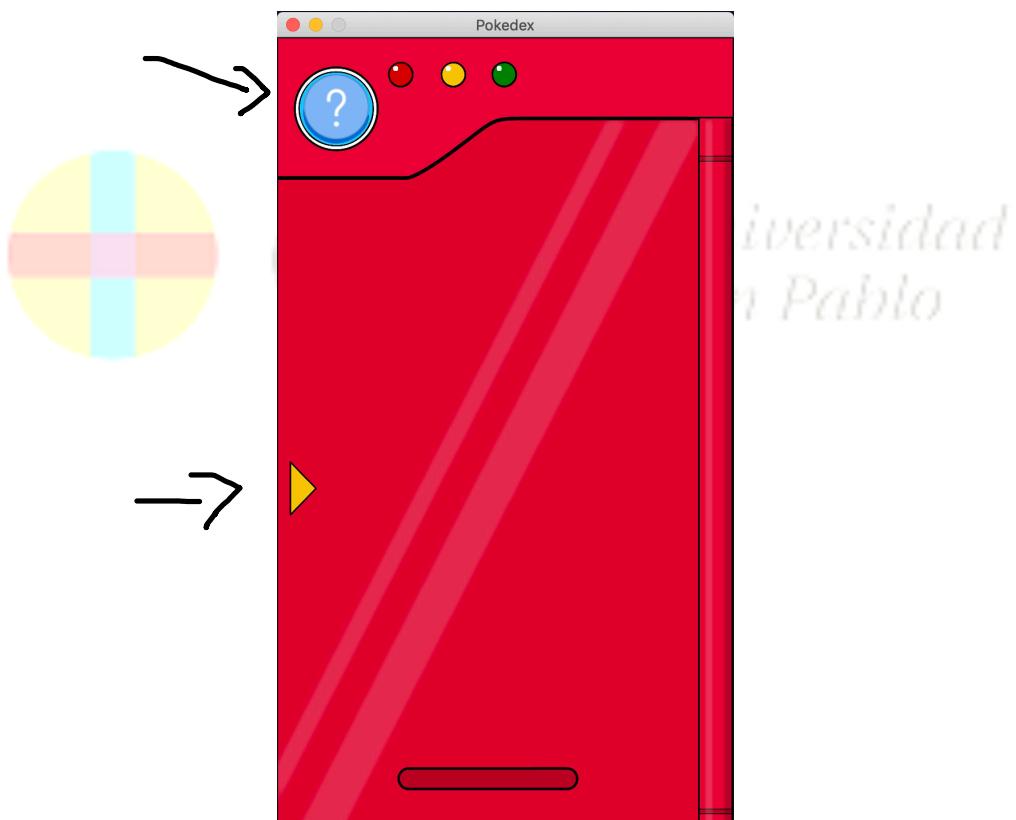


Imagen 3.1 – Diagrama de tablas – Pokedex.db

Descripción de la UI

El programa cuenta con dos Frames. Uno principal que engloba la totalidad de la aplicación y uno previo que se muestra al principio a modo de inicio.

Frame de inicio: trata de imitar el diseño clásico de la Pokédex. A pesar de ser una simple pantalla de inicio, guarda dos funcionalidades ocultas. La primera, y más evidente, un botón para iniciar la aplicación. Este botón es invisible y se encuentra en el triángulo amarillo del dibujo de la Pokédex (ver *Imagen 4.1*). La segunda funcionalidad es un botón con el icono de "?" que genera una ventana modal donde se muestra la información de cómo abrir la Pokédex.



Frame principal: este Frame posee un JTABBE panel con tres iconos distintos. Dependiendo del icono que seleccione el usuario, se mostrará un panel u otro.

Panel Pokemons: es el panel seleccionado por defecto (*Imagen 5.1*) y el panel principal de la aplicación. Consta de una lista de Pokemons a la izquierda. Dependiendo de la selección de la lista, los datos del panel se actualizan. Ya sea la imagen del pokemon, las imágenes de su línea evolutiva, sus datos, sus tipos. Además, este panel posee dos botones en su derecha. El de arriba inicializa un frame con las estadísticas del pokemon seleccionado en la lista (*Imagen 5.2*). Las estadísticas se dibujan en forma de histograma. El botón de abajo genera otro frame donde se muestra una lista de los ataques que el pokemon seleccionado en la lista es capaz de aprender (*Imagen 5.3*).

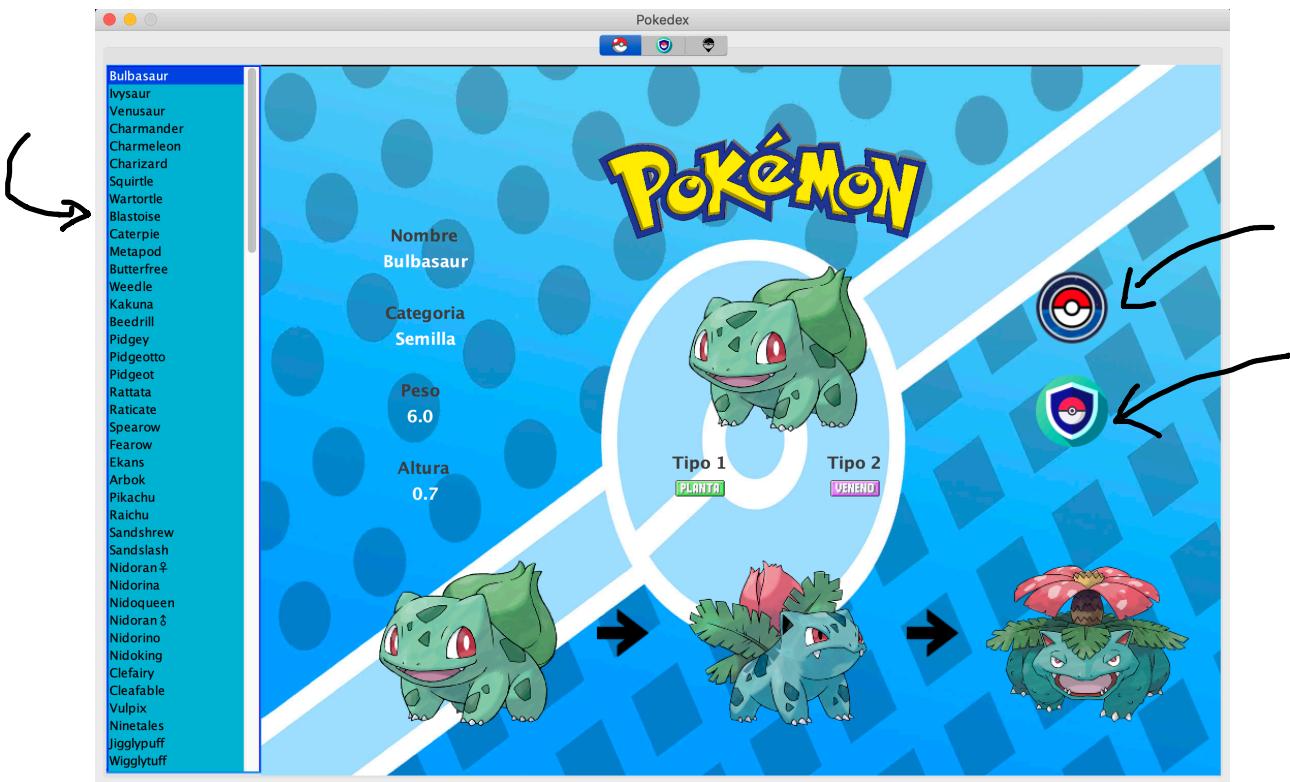


Imagen 5.1 – Panel Pokémon

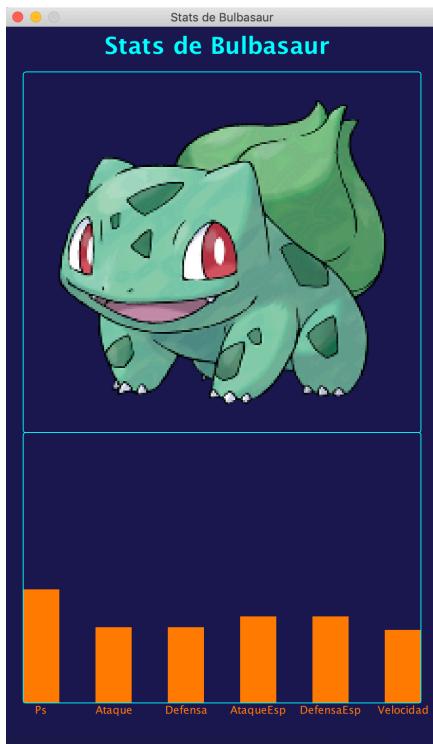


Imagen 5.2 – Stats del pokemon

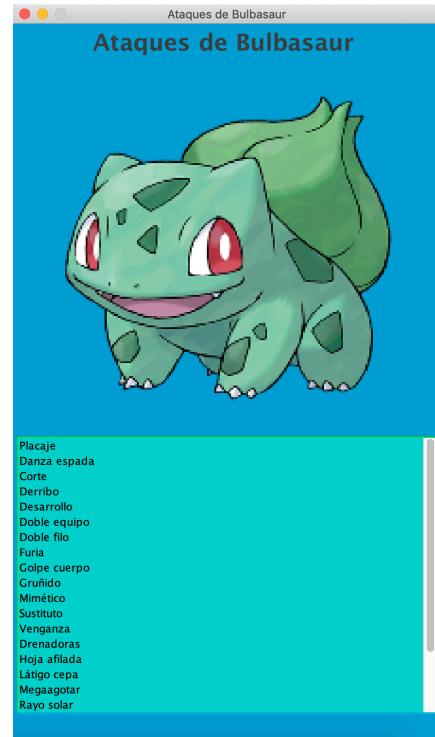


Imagen 5.3 – Ataques del Pokemon

Panel Movimientos: Este panel consta de una lista a la izquierda con todos los movimientos que existen. Los datos mostrados varían dependiendo de la selección (Imagen 6.1).

*Nótese que existen ataques que se aprenden por medio de una Máquina Técnica(MT) o Máquina Oculta(MO), en esos casos, también se mostrará al lado del tipo el nombre de la máquina.

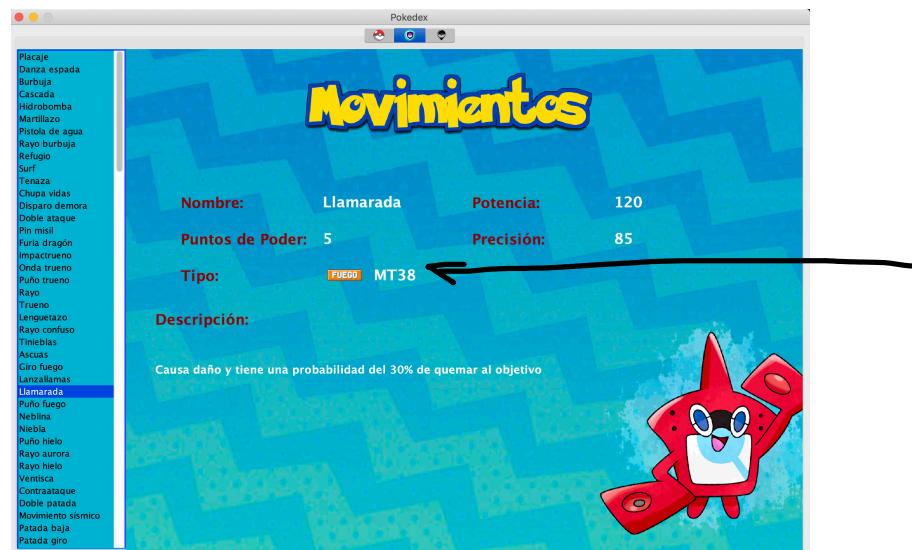


Imagen 6.1 – Panel Movimientos

Panel Localizaciones: tercer y último panel. Consta de una Imagen con botones repartidos a lo largo y ancho de esta simulando las distintas localizaciones (*Imagen 7.1*). En función de qué botón se seleccione, se imprimirán en la lista de al lado los nombres de los pokemons localizables en ese sitio. También hay la posibilidad de seleccionar el botón de "?". En ese caso, se generará una ventana modal donde se podrá seleccionar la localización deseada (*Imagen 7.2*).

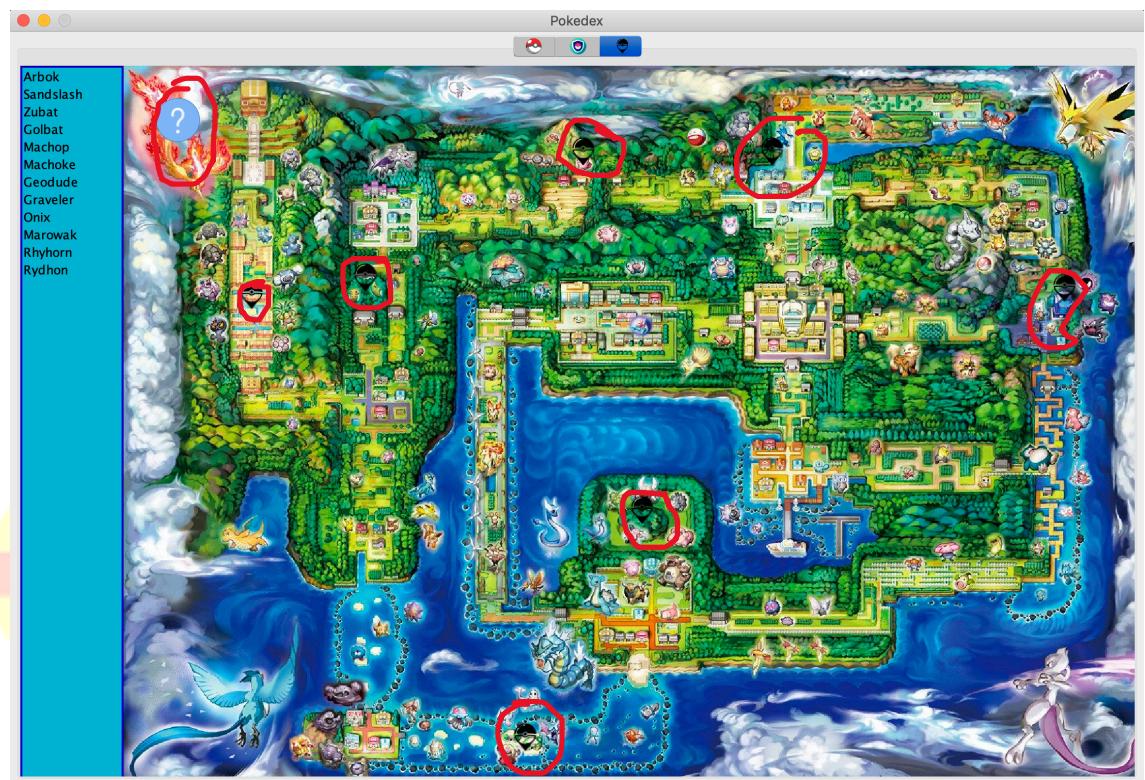


Imagen 7.1 – Panel Localizaciones

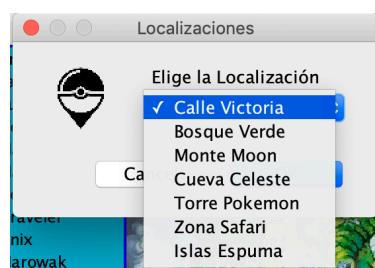


Imagen 7.2 – Ventana Modal - Localizaciones

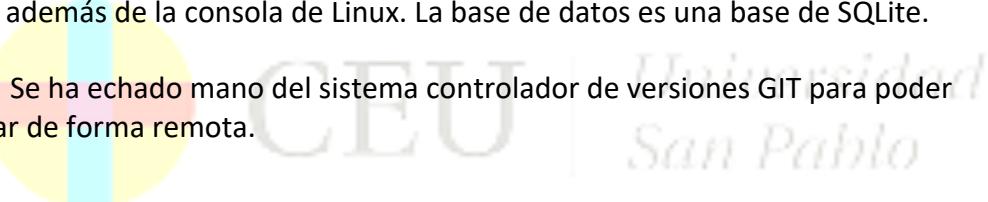
Observaciones y recursos utilizados

Existe un pequeño fallo, propio de java, al menos para la plataforma MAC (no se ha probado en Windows).

Cuando tienes la pantalla maximizada de la consola y ejecutas el programa, se abre una pestaña en el escritorio con la aplicación. Si mientras esto sucede, no va directamente al escritorio, sino que, a otra ventana distinta, el programa se queda en blanco sin llegar a cargar la imagen ni los componentes. Seguramente se trate de un fallo del método *paint()*. Si se detecta este problema al ejecutar, se recomienda que ejecute el programa con su consola/entorno de desarrollo abierto en el escritorio y no realice ninguna acción hasta que el *frame* de inicio esté totalmente cargado.

Para la realización de esta práctica, se ha utilizado el entorno de desarrollo intelliJ además de la consola de Linux. La base de datos es una base de SQLite.

Se ha echado mano del sistema controlador de versiones GIT para poder trabajar de forma remota.



Todo el código se encuentra en <https://github.com/Romuloo/pokedexMTP.git>