

Ejercicios Tema 2. Conectores

Joan Gerard Camarena Estruch

Accés a Dades
Ejercicios



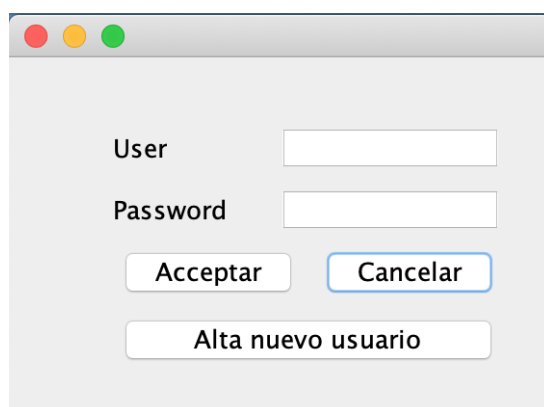
1. Control de acceso. SQLITE

Implementa el sistema de login de una aplicación (usuario y contraseña). Tener en cuenta que la información de los usuarios será una tabla de una BBDD SQLite. El usuario debe guardarse en texto plano y el password debe guardarse encriptado (MD5 o SHA).

Para ello primeramente se va a crear una base de datos `sqlite`, llamada `usuarios.db` con la siguiente estructura:

```
1 create table users (  
2     username char(20),  
3     password varchar(40) not null,  
4     primary key (username)  
5 );
```

Al abrir el programa aparecera una ventana (muy simple) con el siguiente aspecto, que se pasa ya implementada:



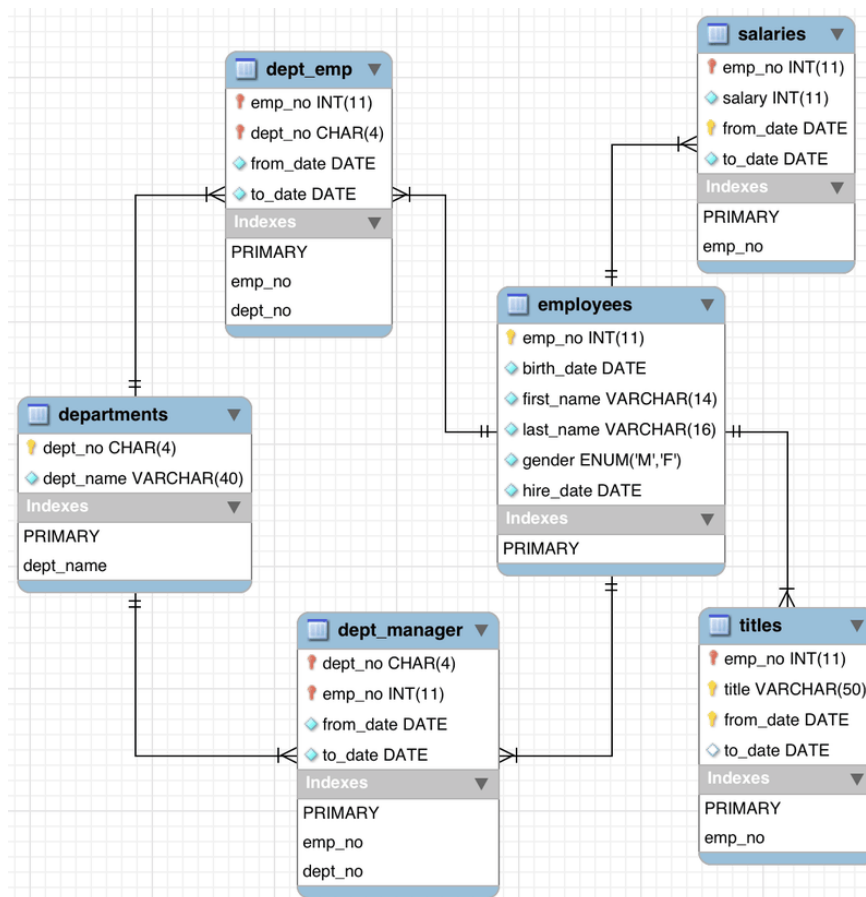
Se pide implementar:

- El boton de `Aceptar`, en el cual comprueba en la BBDD si el usuario y/o contraseña son válidos. El programa podrá informar de la siguiente casuística:
 - El usuario no existe
 - El password no corresponde con el usuario indicado (indica que el usuario si que es correcto pero el password está mal)
 - Los datos si que existen en la BBDD
- El botón `Alta nuevo usuario` insertará en la BBDD los datos. Controlar las excepciones

Nota: Recordar que hay que encriptar el password. Es decir, si el usuario indica como password 1234lo que se almacenará en la BBDD será algo como 81fe8bfe87576c3ecb22426f8e57847382917acf

2. Mantenimiento de bases de datos. Employees

La conocida BBDD de empleados es la que usaremos para esta práctica



1. Crear un mantenimiento de la tabla **Departamentos**. Opciones para crear, modificar, eliminar y buscar departamentos.
2. Crear un mantenimiento de la tabla **Empleados**. Opciones para crear, modificar, eliminar y buscar departamentos.
3. Otras opciones:
 - Dado el código de un empleado ver su información
 - Dado el código de un departamento ver su información
 - Dado el código de un departamento ver los empleados que tiene (y ha tenido)
 - Dado el código de un empleado ver a qué departamento pertenece actualmente.
 - Dado el código de un empleado y un departamento ver si el empleado ha pertenecido a ese departamento y si es afirmativo entre qué fechas (histórico de empleados)
 - Dado el código de un departamento ver los empleados (actuales) y el histórico (empleados que trabajaron allí pero que ahora no lo hacen).

- Realizar un cambio de departamento a un empleado dado. Debe actualizarse la fecha de finalización, y insertarse el nuevo departamento.
1. Por tanto el menú de la aplicación tendrá 3 opciones. Departamentos, Empleados y Consultas. Cada una de estas opciones tendrá las subopciones necesarias para todo lo que se nos pide en los puntos anteriores.

Nota. Se deja a criterio del alumno el presentarlo todo mediante consola o mediante interfaces gráficas.

3. Voluntaria - StarWars

Gracias a la página swapi.dev, se ofrece una API para recuperar datos de la saga StarWars. En ella podemos encontrar información detallada sobre, como se ve en el **JSON** resultado de la petición <https://swapi.dev/api/>:

```
1 {
2   "people": "https://swapi.dev/api/people/",
3   "planets": "https://swapi.dev/api/planets/",
4   "films": "https://swapi.dev/api/films/",
5   "species": "https://swapi.dev/api/species/",
6   "vehicles": "https://swapi.dev/api/vehicles/",
7   "starships": "https://swapi.dev/api/starships/"
8 }
```

Los distintos ítems que existen en dicha API se acceden con peticiones get y su identificador **GET** <https://swapi.dev/api/people/1/>, lo que devuelve algo como:

```
1 {
2   "name": "Luke Skywalker",
3   "height": "172",
4   "mass": "77",
5   "hair_color": "blond",
6   "skin_color": "fair",
7   "eye_color": "blue",
8   "birth_year": "19BBY",
9   "gender": "male",
10  "homeworld": "http://swapi.dev/api/planets/1/",
11  "films": [
12    "http://swapi.dev/api/films/1/",
13    "http://swapi.dev/api/films/2/",
14    "http://swapi.dev/api/films/3/",
15    "http://swapi.dev/api/films/6/"
16  ],
17  "species": [],
18  "vehicles": [
19    "http://swapi.dev/api/vehicles/14/",
20    "http://swapi.dev/api/vehicles/30/"
21  ],
22  "starships": [
23    "http://swapi.dev/api/starships/12/",
24    "http://swapi.dev/api/starships/22/"
25  ],
26  "created": "2014-12-09T13:50:51.644000Z",
27  "edited": "2014-12-20T21:17:56.891000Z",
28  "url": "http://swapi.dev/api/people/1/"
29 }
```

Se pide:

- Crear las clases necesarias para almacenar la información de cada elemento: `people`, `planets`, `films`, `species`, `vehicles`, y `starships`.
- Realizar las funciones de mapeo JSON a Clases. Buscar información librerías GSON.
- Cargar, mediante peticiones GET a la página indicada a las estructuras en memoria interna.
- En un futuro se mapeará las estructuras anteriores a una base relacional.