INTEGRANTE: Manuel Carrizo

APLICACIÓN QT- LOGIN

En la materia de Programación Orientada a Objetos con el profesor Cesar Osimani, consistió en armar un programa en Qt creator un login con el lenguaje C++.

Primero que todo para comenzar a hacer este proyecto, tuve que descargar el instalador de Qt Creator y el OpenSSL. También era necesario tener un web service.
A partir de esto ya en la clase 9, ejercicio 1, tuvimos que crear una base de datos que tenga una tabla con los campos que contenga datos personales, pero lo más importante es que tenga un campo de “Usuario” y “Clave”. Con esto cree una API en la que al poner los atributos de usuario y clave me devuelva los datos del usuario (si es que existe en la base de datos) o denegado.
https://tusrutashoy.com.ar/api_manu/apimanu.php?user=admin&pass=1234 //API
Luego en Qt hacer la interfaz del login de usuarios que tiene de fondo una imagen. Lo programe de tal manera que el login no aparezca hasta que la imagen se descargue.

//Aca esta en connect que va a hacer que se ejecute el slot una vez que la imagen esté descargada

```
//Imagen de fondo
connect(managerparaimagen, SIGNAL(finished(QNetworkReply*)), this, SLOT(slot_descargadeimagenFinalizada(QNetworkReply*)));
managerparaimagen->get(QNetworkRequest(QUrl("https://us.123rf.com/450wm/onairjiw/onairjiw1702/onairjiw170200150/72952333-alt
```

//Una vez que descarga, se muestra el login

```
void Login::paintEvent(QPaintEvent*)
{
    if (imagenCargada) {
        QPainter painter(this);
        painter.drawImage(0, 0, imagenBackground.scaled(this->width(), this->height()));
    }
}

void Login::slot_descargadeimagenFinalizada(QNetworkReply* reply)
{
    imagenBackground = QImage::fromData(reply->readAll());
    imagenCargada = true;
    this->repaint();
    this->show(); //Esto hace que no se habra el login hasta que la imagen este descargada
}
```

Tiene una funcionalidad de seguridad, de que si el usuario ingresa una contraseña incorrecta 3 veces, se le bloqueará el ingreso en esa cuenta por 5 minutos.

```
//        ventana->show();

    }else if( ba.contains( ":" )){
        ba = ba.replace( ":", " " );
        ba.remove(0, 2);

        qDebug() << ba;
        QMessageBox::critical(this,"Error de validacion", "Ha alcanzado el limite de 3 intentos, puede intentar de vuelta a las " + ba);
```

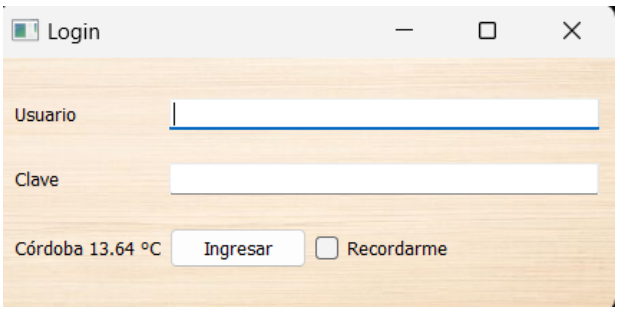
La variable “ba” es traída del php, que es el que le da la información si el usuario alcanzo el límite de 3 intentos y le bloquea la cuenta por 5 minutos.

```php
if ($fila["INTENTOS"] >= 3){
    // Si la diferencia de minutos entre la fecha actual y el ultimo intento es mayor a 5
    $fecha_actual = new DateTime(); // Fecha actual
    $ultimo_inicio = new DateTime($ultimo_inicio); // Fecha obtenida de la base de datos como objeto DateTime
    $diferencia = $fecha_actual->diff($ultimo_inicio); // Diferencia entre las dos fechas
    $minutos = $diferencia->i; // Obtener los minutos de la diferencia
    //echo $minutos; // Imprimir la diferencia de minutos
    if($minutos < 5){
        echo $nueva_fecha . "!";
        exit;
    } else{
        $cod_usuario = $fila["USUARIO"];
        if($fila["CLAVE"] == $password){
            echo $fila["NOMBRE"] . "::";
            echo $fila["APELLIDO"];
            $sql3 = "UPDATE USUARIOS_MANU SET INTENTOS = 0, ULTIMO_INICIO = '$ultimo_intento' WHERE USUARIO = '$cod_usuario'";
            mysqli_query($DB_conn, $sql3);
        }
        else {
            $intentos = $fila["INTENTOS"] + 1 ;
            $sql2 = "UPDATE USUARIOS_MANU SET INTENTOS = '$intentos', ULTIMO_INICIO = '$ultimo_intento' WHERE USUARIO ='$cod_usuario' ";
            mysqli_query($DB_conn, $sql2);
        }
    }
```

- A continuación utilicé una API en la que me trae la temperatura actual de Córdoba.

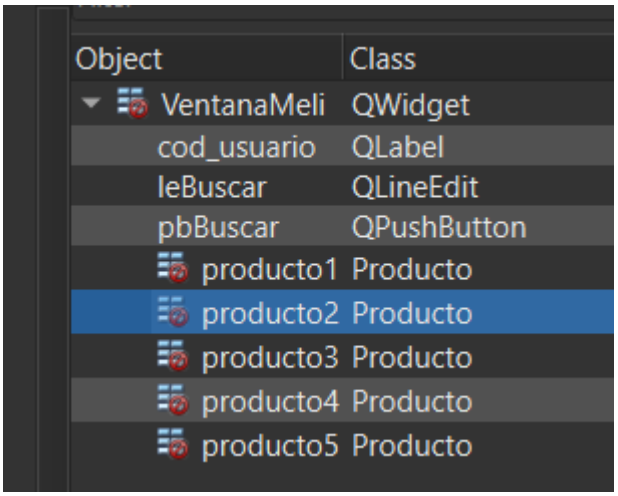
La API es sacada de la página www.openweathermap.org , que tiene su opción Free. Tiene su documentación, en donde te explica como incluirlo y como exprimirlo al máximo. Obviamente que su opción Free es limitada.
Esta API exige tener un “token” que se puede obtener registrándose en la página.
Hay diferentes maneras de llamar a la API, una de las que vi que me parece mas práctico es la siguiente:
https://api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}
Como dice en letra naranja, exige dos tipos de variables para traerte la información. Primero y lo mas importante es la API key, que sería en token. Sin esto nunca funcionaria la API. La siguiente variable es la del nombre de la ciudad. Ingresando esas dos variables te va a traer una información en formato JSON en la que te va a extraer información del clima de esa ciudad y vos podés elegir que quieres mostrar.
{"coord":{"lon":36.3076,"lat":34.7301},"weather":[{"id":800,"main":"Cl ear","description":"clear sky","icon":"01n"}],"base":"stations","main":{"temp":10.69,"feels_like ":10.19,"temp_min":10.69,"temp_max":10.69,"pressure":1019,"humidity":9 1,"sea_level":1019,"grnd_level":984},"visibility":10000,"wind":{"speed ":3.07,"deg":251,"gust":8.9},"clouds":{"all":6},"dt":1681429421,"sys": {"country":"SY","sunrise":1681441492,"sunset":1681488302},"timezone":1 0800,"id":163533,"name":"Bsas","cod":200}
Una cosa importante aclarar es que podemos aclararle a la API en que unidad queremos que se nos muestre la temperatura, ya sea en “metric”, “standard" o “imperial”. La que nosotros usamos es “metric” (grados Celsius). Esto se le aclara metiendo una variable más en el link de la API.
https://api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}&units={unit}

Yo traigo esa información desde un php a través de este link:
http://tusrutashoy.com.ar/api_manu/apiclima.php y lo muestro en el login.



- Una vez que se loguea correctamente, se va a abrir una ventana en donde va a tener un campo para escribir lo que desea buscar. Tiene vinculada una API de mercadolibre, en la que le va a traer los 5 resultados más baratos de lo que deseó.

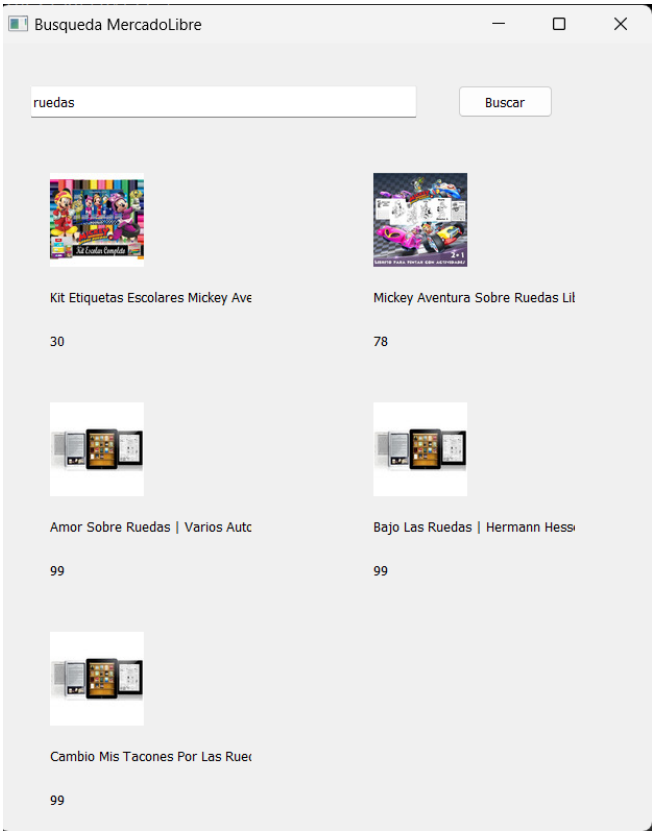
Cada búsqueda se va a mostrar en un QWidget y se mostrarán en la ventana de búsqueda.



La API me trae la información en formato JSON, y cada producto es un objeto. Por lo que tengo que acceder a cada uno y extraerle la información de: foto, título y precio.
Esto lo haría 5 veces asi muestro los 5 productos.

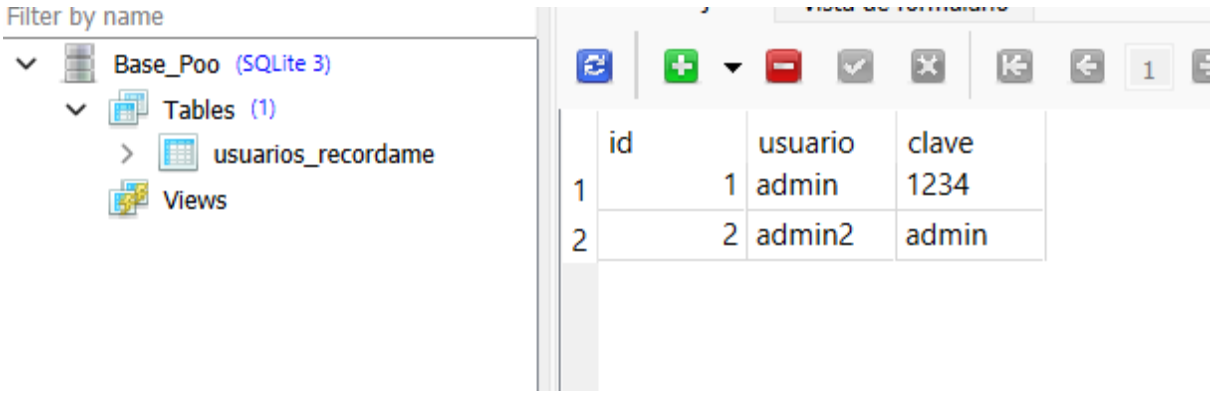
```
if(cantResultados == 0){
    if (result.isObject()) {
        QJsonObject resultObject = result.toObject();
        if (resultObject.contains("thumbnail")) {
            QString foto = resultObject.value("thumbnail").toString();
            qDebug() << "Foto: " << foto;
            ui->producto1->setImagen(foto);
        }
        if (resultObject.contains("title")) {
            QString descripcion = resultObject.value("title").toString();
            qDebug() << "Descripcion: " << descripcion;
            ui->producto1->setDescripcion(descripcion);
        }
        if (resultObject.contains("price")) {
            int precio = resultObject.value("price").toInt();
            qDebug() << "Precio: " << precio;
            ui->producto1->setPrecio(precio);

        }
    }
}
```

Y en la ventana de búsqueda se mostrarán de esta manera, ordenados de menor precio a mayor.



- Por último, le agregué la funcionalidad de “recordarme”. Cuando el checkbox este tildado, el usuario y clave se almacenan en una base de datos local en SQlite. Esto servirá para que la próxima vez que ingrese el usuario, el campo clave se le autocompleta.

//Aquí quedan almacenados los usuarios y las claves



//Esto es un if en donde si el usuario tilda el checkbox y ese usuario no existe en la base, entonces la guarda

```
if(ui->leRecordarme->isChecked() && !usuarioValido){
    QSqlQuery * query = new QSqlQuery( adminDB->getDB() );
    query->exec( "INSERT INTO usuarios_recordame (usuario, clave) values ('" + ui->leUsuario->text() + "' , '" + ui->leClave->text() + "' )
}
```

//Una vez que leUsuario se ingreso, se ejecuta slot_recordar()

```
// Connect para validar el recordarme
connect(ui->leUsuario, SIGNAL(editingFinished()), this, SLOT(slot_recordar()));
```

//Y este slot básicamente lo que hace es consultar a la base si contiene esa información. Si existe, entonces autocompleta leClave

```
void Login::slot_recordar(){

    if ( adminDB->getDB().isOpen() )  {
        QSqlQuery * query = new QSqlQuery( adminDB->getDB() );

        query->exec( "SELECT usuario, clave FROM usuarios_recordame WHERE usuario='" +
                    ui->leUsuario->text() + "'" );
        //query->exec( "SELECT usuario, clave FROM usuarios");
        qDebug() << query->lastQuery();

        // Si los datos son consistentes, devolverá un único registro.
        while ( query->next() )  {

            qDebug() << "1";

            QSqlRecord record = query->record();

            // Obtenemos el número de la columna de los datos que necesitamos.
            //int columnaNombre = record.indexOf( "usuario" );
            int columnaClave = record.indexOf( "clave" );


            // Obtenemos los valores de las columnas.
            //qDebug() << "Nombre=" << query->value( columnaNombre ).toString();
            //qDebug() << "Apellido=" << query->value( columnaApellido ).toString();

            ui->leClave->setText(query->value( columnaClave ).toString());
            usuarioValido = true;
        }
```

En conclusión, nos quedó una aplicación compleja. Un login en la que tiene su respectiva base de datos en la web server, que a través de un php trae la información. Tiene la seguridad de que si llega a los 3 intentos de ingreso, se bloquee la cuenta por 5 minutos. En la interfaz de usuario además de tener un background, se muestra la temperatura actual de Córdoba. Tiene la función de recordar el usuario, vinculada a una base de datos local y una de las cosas más complejas que tiene es traerte 5 resultados de una búsqueda, a través de la api de mercadolibre.