
	Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLD-001	Página: 1

INFORME DE LABORATORIO

INFORMACION BASICA					
ASIGNATURA:	Tecnología de Objetos				
TITULO DE LA PRACTICA:	Funciones lambda e introducción a Widget App Qt				
NUMERO DE LA PRACTICA:	05	AÑO LECTIVO:	2025 - B	N° SEMESTRE:	VI
FECHA DE PRESENTACION:	19 / 10 / 2025	HORA DE PRESENTACION:	--:-- PM		
INTEGRANTE (s): ■ Huayhua Hillpa, Yourdyy Yossimar				NOTA:	
DOCENTE (s): ■ Mg. Escobedo Quispe, Richart Smith					



1. Tarea

1.1. Problema propuesto:

- [1] Contar los números impares y menores de 20 de un vector de números enteros generados de forma aleatoria (la solución puede usar el método `std::count_if`)
- [2] Diferencias, ventajas y desventajas de las clases `QVector` y `QList`.
- [3] Crear una interface gráfica (que implemente señales y slots) que muestre una lista de países, al dar clic sobre alguno que se muestre un `Label` o `Text` con el idioma y capital correspondiente.

2. Equipos, materiales y temas utilizados

- Subsistema de Windows para Linux (WSL) con Ubuntu.
- Sistema operativo: Microsoft Windows [Versión 10.0.26100.6584]
- TeX Live 2025
- Helix 25.01.1 (e7ac2fcd)
- Visual Studio Code 1.104.0 x64
- Git version 2.41.0.windows.1
- Cuenta activa en GitHub para la gestión de repositorios remotos.
- Qt Creator
- Leguaje de programación C++
- Librería Qt

	<p>Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 2</p>

3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- <https://github.com/yhuayhuahi/Teo.git>
- URL para el laboratorio (05) en el Repositorio GitHub.
- <https://github.com/yhuayhuahi/Teo/tree/main/laboratorios/lab05>

4. Desarrollo de las actividades

4.1. Actividad 01: Implementación en C++



4.1.1. Función Main en C++

A continuación se muestra la implementación de la función main en C++ que genera un vector de números enteros aleatorios y cuenta cuántos de ellos son impares y menores de 20 utilizando el método.

Listing 1: Función Main en cpp - Primera implementación

```

1  #include <iostream>
2  #include <vector>
3  #include <algorithm> // std::count_if
4  #include <cstdlib>
5  #include <ctime>
6
7  int main() {
8      //para generar números aleatorios
9      srand(static_cast<unsigned int>(time(nullptr)));
10
11     // Generamos un vector de numeros enteros aleatorios
12     std::vector<int> numeros;
13     int n = 20;
14
15     for (int i = 0; i < n; ++i) {
16         numeros.push_back(rand() % 50); //números aleatorios entre 0 y 49
17     }
18
19     //mostramos los números generados
20     std::cout << "Numeros generados a continuación: ";
21     for (int num : numeros) {
22         std::cout << num << " ";
23     }
24     std::cout << std::endl;
25
26     // Contar cuántos son impares y menores de 20 ! usando count_if
27     int conteo = std::count_if(numeros.begin(), numeros.end(), [](int x) {
28         return (x % 2 != 0) && (x < 20);
29     });
30
31     std::cout << "Cantidad de numeros impares y menores de 20 encontrados: " << conteo
32         << std::endl;
33
34     return 0;
35 }
```

	<p>Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 3</p>

4.1.2. Prueba de ejecución

Se realizó la prueba de ejecución del código en C++ y se obtuvo el siguiente resultado:

```
WSL at ~ / ... /laboratorios/lab05/ejer01 0/main = 71 16ms
06:06:20 ls
contador.cpp
WSL at ~ / ... /laboratorios/lab05/ejer01 0/main = 71 15ms
06:06:23 g++ contador.cpp -o contador
WSL at ~ / ... /laboratorios/lab05/ejer01 0/main = 72 2.209s
06:06:47 ./contador
Numeros generados a continuación: 46 3 43 29 47 47 23 41 32 39 21 36 44 39 0 5 11 8 43 10
Cantidad de numeros impares y menores de 20 encon: 3
WSL at ~ / ... /laboratorios/lab05/ejer01 0/main = 72 24ms
06:06:52
```

Figura 1: Interfaz inicial de la aplicación Qt

4.2. Actividad 02: Diferencias, ventajas y desventajas de las clases QVector y QList.

4.2.1. Diferencias identificadas

Comportamiento por versión de Qt

1. Si se trabaja con Qt5, las diferencias todavía importan: QVector suele ofrecer mejor rendimiento para elementos grandes y acceso contiguo; QList puede comportarse distinto para tipos grandes (indirección). MIT Stuff
2. Si se trabaja con Qt6, las diferencias son prácticamente nominales: la implementación es la misma/compatible y muchos de los viejos problemas de QList se resolvieron por unificación. Qt

Almacenamiento en memoria

1. Qt5: QVector<T>= elementos contiguos. QList<T>= en muchos casos elementos alocados individualmente y punteros en un array.
2. Qt6: ambos almacenan elementos de forma contigua, un comportamiento tipo std::vector.

Operaciones como prepend o append

1. Qt5: QList históricamente tenía mejor coste amortizado para prepend en ciertas implementaciones; QVector era más eficiente para crecimiento cuando se requiere memoria contigua.
2. Qt6: se añadió optimización de prepend a QList (y por tanto a QVector vía alias) para no romper expectativas. Qt



Compatibilidad y estilo de API

1. Muchas APIs de Qt históricamente devolvían QList; por eso se tuvo decisión de unificar para facilitar migraciones entre versiones y evitar confusión en la API pública.

4.2.2. Ventajas y desventajas (prácticas)

Ventajas de QVector:

1. Almacenamiento contiguo → mejor localidad de caché → mejor rendimiento para acceso secuencial/aleatorio.

	<p>Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLD-001	Página: 4

2. Menos sobrecarga al almacenar tipos grandes (evita aloca cada elemento por separado).

Desventajas de QVector:

1. `prepend()` no era tan eficiente como en `QList` esto claro antes de las actualizaciones, dependiendo del uso concreto.
2. Estado (Qt6): alias de `QList` – las diferencias previas desaparecen en gran medida.

Ventajas de QList:

1. API cómoda y era la recomendada por la documentación en muchos casos; en ciertas situaciones `prepend()` tenía mejor coste amortizado. Para tipos pequeños no siempre era peor; comportamiento dependía de `sizeof(T)` declaraciones como `Q_DECLARE_TYPEINFO`.

Desventajas de QList:

1. Para tipos no triviales o grandes, podía inducir muchas asignaciones individuales en el heap y pérdida de localidad de referencia → peor rendimiento y mayor uso de memoria. Por eso muchos desarrolladores preferían `QVector` como "default".
2. Estado (Qt6): implementación unificada, por lo que la mayoría de las desventajas históricas ya no aplican.

4.3. Actividad 03: Implementación de una interfaz gráfica con Qt

4.3.1. Implementación en C++ con Qt



A continuación se muestra la implementación en C++ con Qt de una interfaz gráfica que muestra una lista de países y al dar clic sobre alguno se muestra un Label o Text con el idioma y capital correspondiente.

Listing 2: mainwindow.h

```

1  #ifndef MAINWINDOW_H
2  #define MAINWINDOW_H
3
4  #include <QMainWindow>
5  #include <QStringListModel>
6  #include <QMap>
7
8  QT_BEGIN_NAMESPACE
9  namespace Ui { class MainWindow; }
10 QT_END_NAMESPACE
11
12 class MainWindow : public QMainWindow
13 {
14     Q_OBJECT
15
16 public:
17     MainWindow(QWidget *parent = nullptr);
18     ~MainWindow();
19
20 private slots:
21     void on_countrySelected(const QModelIndex &index); // slot
22
23 private:

```

	<p>Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 5</p>

```

24     Ui::MainWindow *ui;
25     QStringListModel *model; // lista
26     QMap<QString, QPair<QString, QString>> datosPaíses; // país -> (idioma, capital)
27 };
28
29 #endif // MAINWINDOW_H

```



4.3.2. Implementación del archivo mainwindow.cpp

Listing 3: mainwindow.cpp

```

1  #include "mainwindow.h"
2  #include "../ui_mainwindow.h"
3  #include <QStringList>
4
5  MainWindow::MainWindow(QWidget *parent)
6      : QMainWindow(parent)
7      , ui(new Ui::MainWindow)
8  {
9      ui->setupUi(this);
10
11      // lista de países
12      QStringList países = {
13          "Peru",
14          "Mexico",
15          "Colombia",
16          "Argentina",
17          "Chile",
18          "Estados Unidos",
19          "Canada",
20          "Francia",
21          "Italia",
22          "Japon"
23      };
24
25      // idioma y capital
26      datosPaíses = {
27          {"Peru", {"Español", "Lima"}},
28          {"Mexico", {"Español", "Ciudad de Mexico"}},
29          {"Colombia", {"Español", "Bogota"}},
30          {"Argentina", {"Español", "Buenos Aires"}},
31          {"Chile", {"Español", "Santiago"}},
32          {"Estados Unidos", {"Ingles", "Washington D.C."}},
33          {"Canada", {"Ingles y Frances", "Ottawa"}},
34          {"Francia", {"Frances", "Paris"}},
35          {"Italia", {"Italiano", "Roma"}},
36          {"Japon", {"Japones", "Tokio"}}
37      };
38
39      // se crea el modelo y se asocia al QListView
40      model = new QStringListModel(this);
41      model->setStringList(países);
42      ui->listView->setModel(model);
43
44      // conectamos la señal y slot
45      connect(ui->listView->selectionModel(), &QItemSelectionModel::currentChanged,

```

	<p>Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 6</p>

```

46         this, &MainWindow::on_countrySelected);
47     }
48
49     MainWindow::~MainWindow()
50     {
51         delete ui;
52     }
53
54     void MainWindow::on_countrySelected(const QModelIndex &index)
55     {
56         QString pais = model->data(index, Qt::DisplayRole).toString();
57
58         if (datosPaíses.contains(pais)) {
59             QString idioma = datosPaíses[pais].first;
60             QString capital = datosPaíses[pais].second;
61             ui->labelInfo->setText("Idioma: " + idioma + "\nCapital: " + capital);
62         } else {
63             ui->labelInfo->setText("Informacion no disponible");
64         }
65     }

```

4.3.3. Pruebas de ejecución

Se realizaron pruebas de ejecución para verificar que la interfaz gráfica funciona correctamente. Se seleccionaron diferentes países de la lista y se comprobó que la información mostrada en el Label o Text era la correcta.

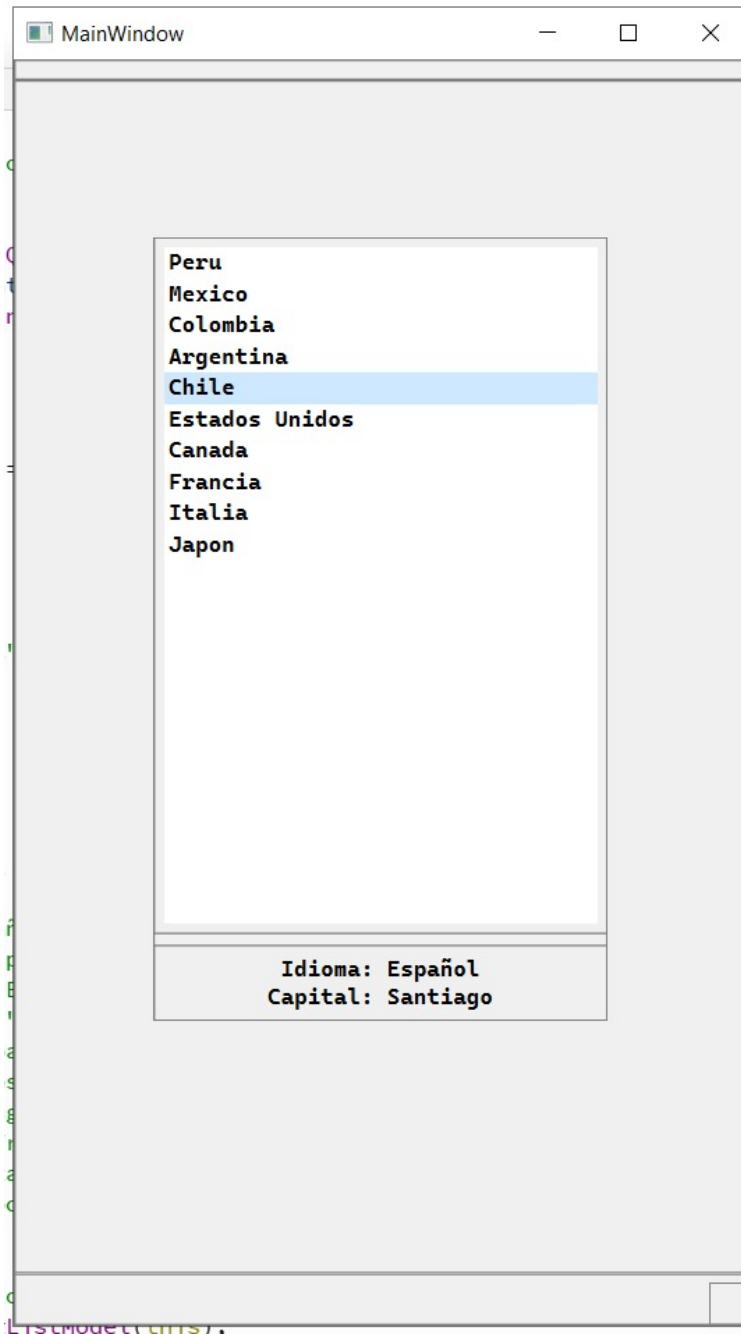


Figura 2: Interfaz mostrando información de un país seleccionado

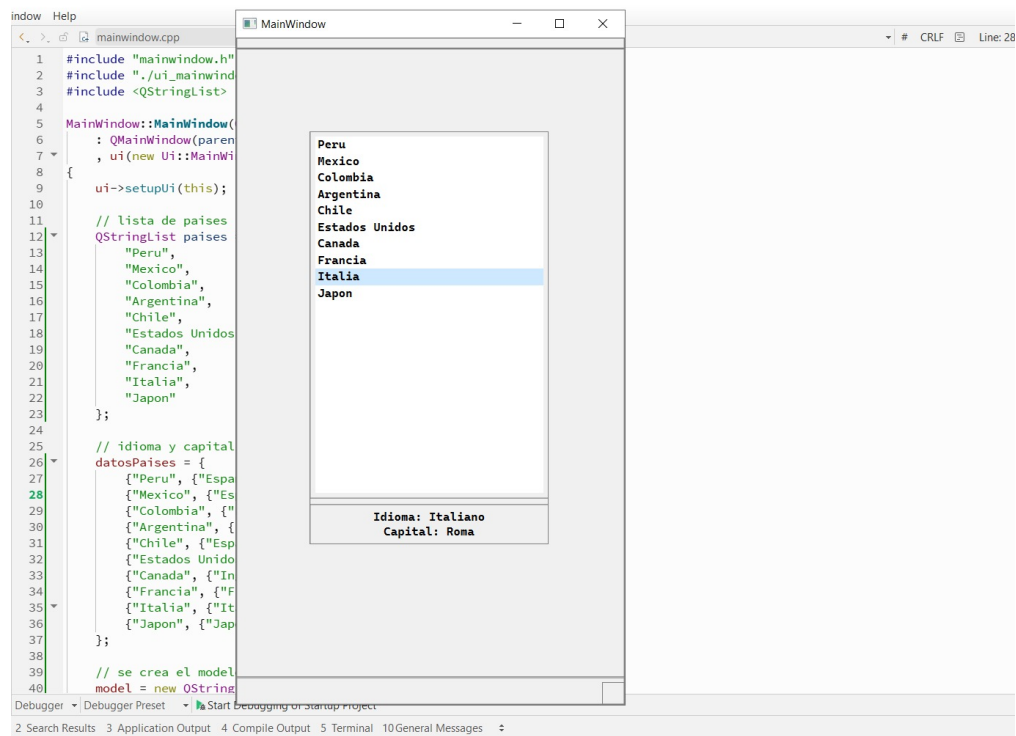


Figura 3: Interfaz mostrando información de otro país seleccionado

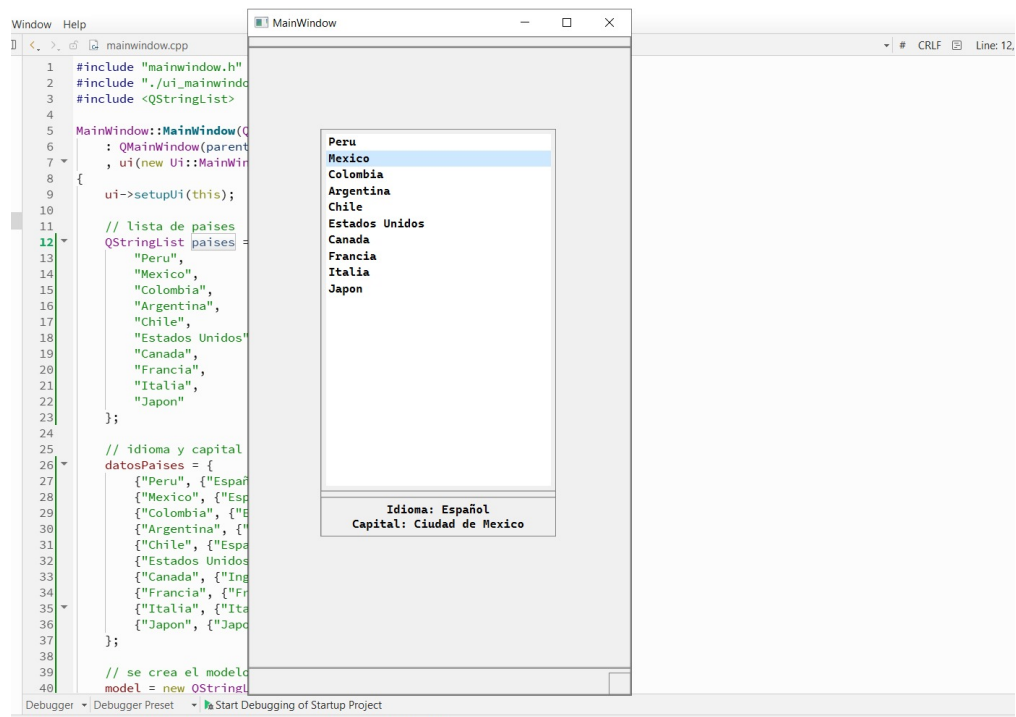




Figura 4: Interfaz mostrando información de otro país seleccionado

	<p>Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 9</p>

4.4. Commits realizados

4.4.1. Primer Commit

- Este commit se hizo después de terminar la primera implementación de código para C++ con Qt para el ejercicio propuesto en el laboratorio.

```

~/. . . /Teo/laboratorios/lab05  O  |main  ≡  ?1  ~2  4.137s  ♥
02:26:56  git add ejer03/
~/. . . /Teo/laboratorios/lab05  O  |main  ≡  ?1  ~2  |  ? +5  246ms
06:46:24  git commit -m "Ejercicio 03 completo"
[main d436d04] Ejercicio 03 completo
5 files changed, 260 insertions(+)
create mode 100644 laboratorios/lab05/ejer03/CMakeLists.txt
create mode 100644 laboratorios/lab05/ejer03/main.cpp
create mode 100644 laboratorios/lab05/ejer03/mainwindow.cpp
create mode 100644 laboratorios/lab05/ejer03/mainwindow.h
create mode 100644 laboratorios/lab05/ejer03/mainwindow.ui
~/. . . /Teo/laboratorios/lab05  O  |main  ↑1  ?1  ~2  286ms  ♥
06:47:37  git push

```

Figura 5: Primer Commit - Ejercicio 03 completo

4.5. Estructura del laboratorio

A continuación se muestra la estructura de archivos y carpetas del laboratorio realizado: Claramente los archivos de compilación y otros que se pudieron generar no se subieron al repositorio.

```

~/.../Teo/laboratorios/lab05  ↵ main
06:47:45 tree /f
El número de serie del volumen es A025-E3A0
C:.\
├── .gitignore
├── Readme.md
├── ejer01
│   └── contador.cpp
├── ejer03
│   ├── CMakeLists.txt
│   ├── main.cpp
│   ├── mainwindow.cpp
│   ├── mainwindow.h
│   └── mainwindow.ui
└── informe-latex
    ├── desarrollo.tex
    ├── main.tex
    └── rubricas.tex
  
```

Figura 6: Estructura de archivos y carpetas del laboratorio

5. Cuestionario



No hay un cuestionario para este laboratorio.

6. Rúbricas

6.1. Entregable Informe

Cuadro 1: Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y facil de leer.



	<p>Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 11</p>

6.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna Checklist si cumple con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos los ítems.
- El alumno debe autocalificarse en la columna Estudiante de acuerdo a la siguiente tabla:

Cuadro 2: Niveles de desempeño

Nivel				
Puntos	Insatisfactorio 25%	En Proceso 50%	Satisfactorio 75%	Sobresaliente 100%
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

	<p>Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLD-001	Página: 12

Cuadro 3: Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Chec- klist	Estudian- te	Profe- sor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	15	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	15	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente estan dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	3	
Total		20		16	

7. Referencias

- [1] <https://doc.qt.io/qt-6/signalsandslots.html>
- [2] <https://doc.qt.io/qt-6/qtwidgets-widgets-mainwindow-example.html>
- [3] <https://doc.qt.io/qt-6/qtwidgets-widgets-qmainwindow.html>