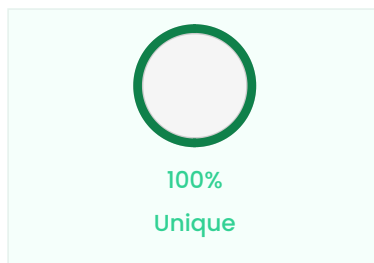
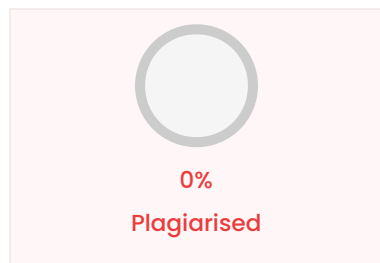


## Plagiarism Scan Report



## Words Statistics

Words	774
Characters	7369

Exclude URL None

## Content Checked For Plagiarism

```
#include "ListaPerson.h"
ListaPerson::ListaPerson() { primero = NULL; }
void ListaPerson::insertar(Persona dato) {
    NodoPerson* nuevo = new NodoPerson(dato);
    if (primero == NULL) {
        primero = nuevo;
        primero->setSiguiente(primero);
        primero->setAnterior(primero);
    } else if (primero->getSiguiente() == primero) {
        ultimo = nuevo;
        ultimo->setSiguiente(primero);
        ultimo->setAnterior(primero);
        primero->setSiguiente(ultimo);
        primero->setAnterior(ultimo);
    } else {
        ultimo->setSiguiente(nuevo);
        nuevo->setAnterior(ultimo);
        nuevo->setSiguiente(primero);
        primero->setAnterior(nuevo);
        ultimo = nuevo;
    }
}
bool ListaPerson::eliminar(string dato) {
    if (primero == NULL) {
        return false;
    } else if (primero->getSiguiente() == primero && primero->getDato().getPlaca() == dato) {
        primero = NULL;
        return true;
    } else {
        NodoPerson* auxiliar = primero;
        while (auxiliar->getSiguiente() != primero && auxiliar->getDato().getPlaca() != dato) {
            auxiliar = auxiliar->getSiguiente();
        }
        if (auxiliar == primero && auxiliar->getDato().getPlaca() == dato) {
            primero->getSiguiente()->setAnterior(primero->getAnterior());
            primero->getAnterior()->setSiguiente(primero->getSiguiente());
            primero = primero->getSiguiente();
            delete auxiliar;
            return true;
        } else if (auxiliar->getDato().getPlaca() == dato) {
            auxiliar->getAnterior()->setSiguiente(auxiliar->getSiguiente());
            auxiliar->getSiguiente()->setAnterior(auxiliar->getAnterior());
            delete auxiliar;
            return true;
        } else {
            return false;
        }
    }
}
bool ListaPerson::buscar(string dato) {
    if (primero == NULL) {
        cout << "=====NO SE ENCONTRO INFORMACION===== " << endl;
        system("pause");
        return false;
    } else if (primero->getSiguiente() == primero && primero->getDato().getPlaca() == dato) {
        cout << "=====Informacion - Usuario=====1" << endl;
        cout << ">>Nombre: " << primero->getDato().getNombre() << endl;
        cout << ">>Apellido: " << primero->getDato().getApellido() << endl;
        cout << ">>Direccion: " << primero->getDato().getDireccion() << endl;
        cout << ">>Telefono: " << primero->getDato().getTelefono() << endl;
        cout << ">>Correo: " << primero->getDato().getCorreo() << endl;
        cout << ">>Placa: " << primero->getDato().getPlaca() << endl;
        cout << "\n=====Informacion - Vehiculo===== " << endl;
        return true;
    } else {
        NodoPerson* auxiliar = primero;
        while (auxiliar->getSiguiente() != primero && auxiliar->getDato().getPlaca() != dato) {
            auxiliar = auxiliar->getSiguiente();
        }
        if (auxiliar->getSiguiente() == primero && auxiliar->getDato().getPlaca() == dato) {
            cout << "=====Informacion - Usuario=====2" << endl;
            cout << ">>Nombre: " << auxiliar->getDato().getNombre() << endl;
            cout << ">>Apellido: " << auxiliar->getDato().getApellido() << endl;
            cout << ">>Direccion: " << auxiliar->getDato().getDireccion() << endl;
            cout << ">>Telefono: " << auxiliar->getDato().getTelefono() << endl;
            cout << ">>Correo: " << auxiliar->getDato().getCorreo() << endl;
            cout << ">>Placa: " << auxiliar->getDato().getPlaca() << endl;
            cout << "\n=====Informacion - Vehiculo===== " << endl;
            return true;
        } else {
            return false;
        }
    }
}
```

```
" "" "" "" "" "" """; html << ""; html << ""; html << ""; html << ""; html << ""; html << ""; html << "
```

```
; std::ostringstream plain; std::string html_filename = "Reporte_Clientes.html"; //std::string pdf_filename =
"Reporte_Clientes.pdf"; std::ofstream out_html(html_filename, std::ios::trunc); // std::ofstream out_txt("data.txt",
std::ios::trunc); out_html << html.str(); // out_txt.close(); out_html.close(); lista.imprimir(); } void
ListaPerson::iterar(std::function lambda) { NodoPerson* tmp = this->primero; while (tmp != nullptr) { lambda(tmp-
>getDato()); tmp = tmp->getSiguiente(); } }
```