

# Torre de control

Grupo Alan Turing



# Índice:

Análisis

Diseño

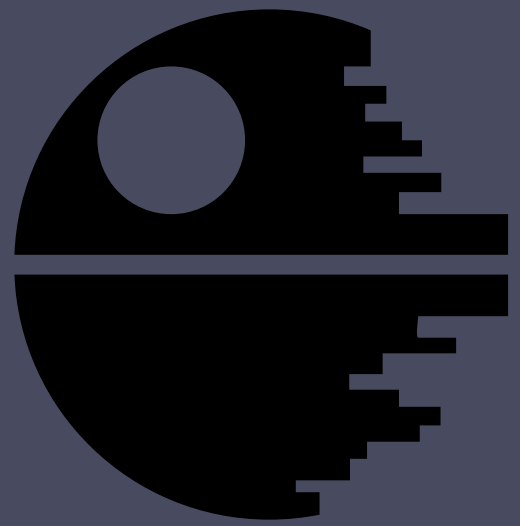
Ampliaciones

Conceptos

Salir

Empezar





# Introducción

Empresa: Alan Turing S.L.N.E

Cliente: La República Galáctica

Producto: 1000000 créditos galácticos

Mantenimiento: 1000 créditos mensuales



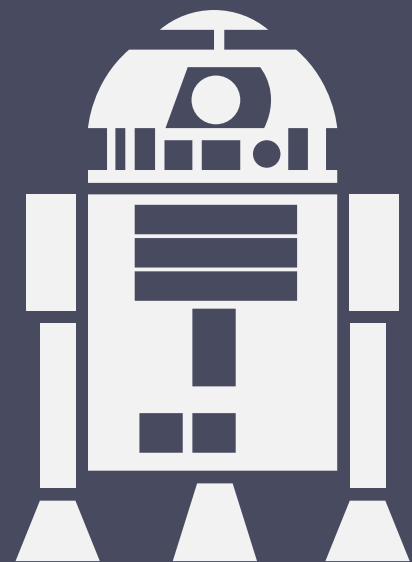
# Análisis

Ver, gestionar y controlar el flujo de naves naves en el hangar

Herencia: tipos de naves

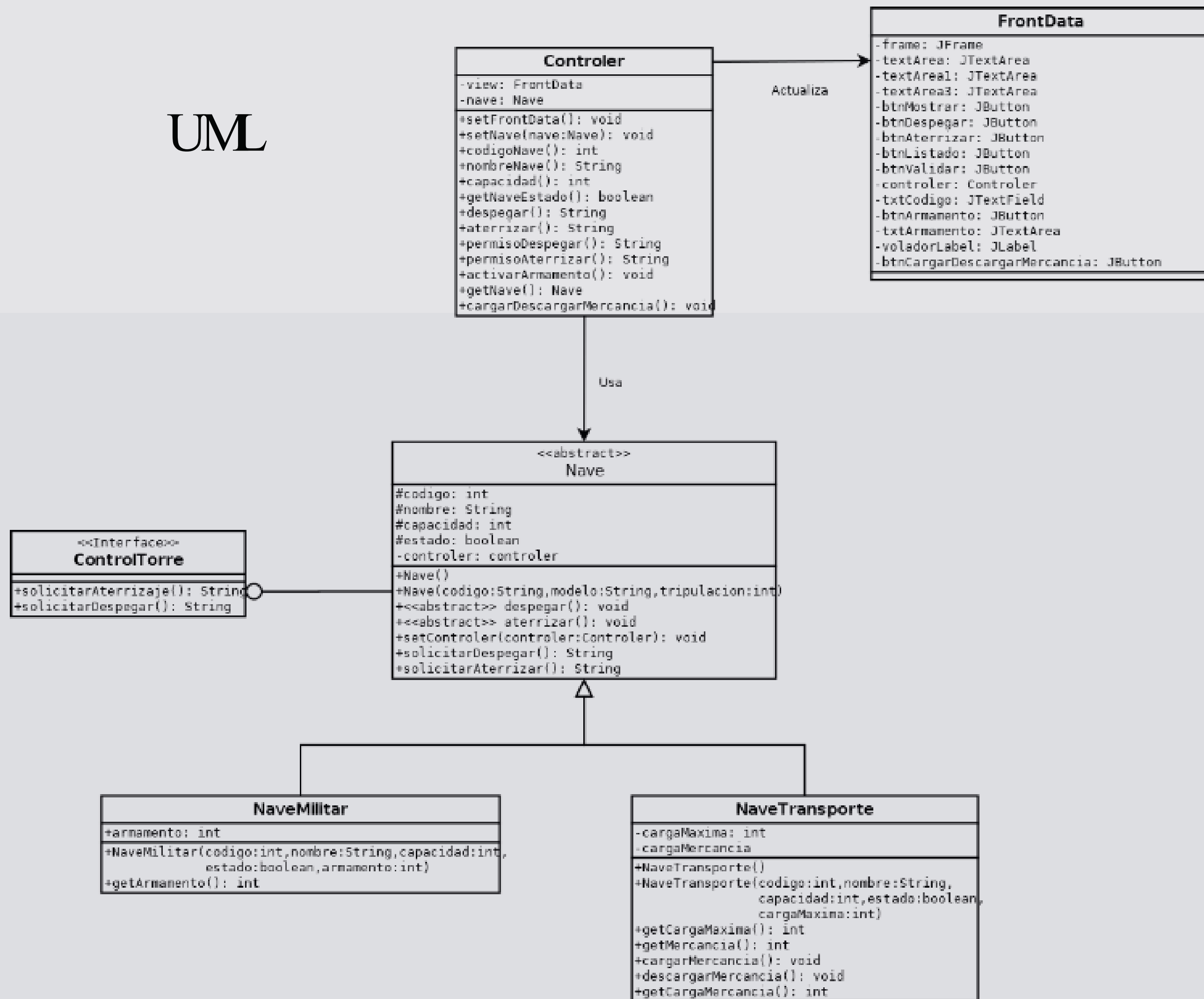
MVC: Mdelo Vista

Controlador



# Di se ño

## UML

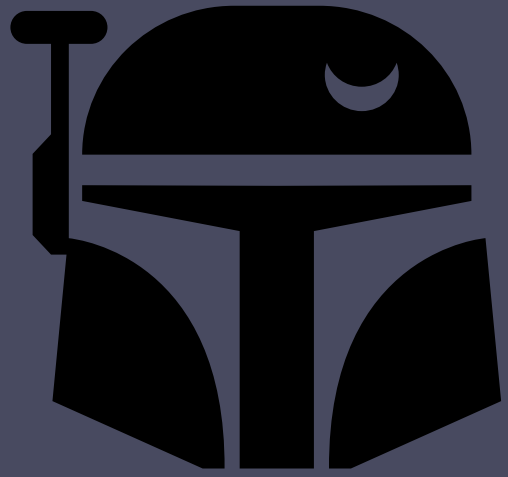




# Ampliaciones

El Grupo Alan Turing S. L. N. E propone ampliar la aplicación "Ground Control" para el control de contrabando y la defensa del hangar.

Esta expansión incluirá un sistema de escaneo como barrera en la entrada del hangar, que enviará información sobre cargas ilegales a la torre de control. Se implementará una opción de escaneo en la interfaz gráfica de usuario (GUI), que dirigirá la pantalla a una nueva ventana para revisar posibles puntos de carga ilegal.



# Conceptos teórico/práctico

## Polimorfismo

```
private Nave Nave transporte = new NaveTransporte(1112, "Transport Shuttle", 690, true, 1098)
```

# CASTING

## Casting explícito

```
public void actionPerformed(ActionEvent e) {  
    // Obtenemos el código ingresado por el usuario  
    try {  
        int codigoIngresado = Integer.parseInt(txtCodigo.getText());  
  
        // Verificamos si el código ingresado coincide con el código de la nave
```



# CARACTERES DE ESCAPE

## Salto de línea

```
if (naveTransporte.getCargaMercancia() > 0) {  
    info += "\nCantidad de mercancía: " + naveTransporte.getCargaMercancia();  
} else {  
    info += "\nLa nave está vacía.";  
}  
txtArmamento.setText(info);
```

# SOBRECARGA DE MÉTODOS

```
public Nave() {  
    this.codigo=0;  
    this.nombre="";  
    this.capacidad = 0;  
    this.estado=true;  
}  
public Nave(int codigo, String nombre,int capacidad, boolean estado) {  
    this.codigo=codigo;  
    this.nombre=nombre;  
    this.capacidad = capacidad;  
    this.estado=estado;  
}
```

# REDEFINICIÓN DE MÉTODOS

```
@Override
public String despegar() {
    return "La nave militar "+super.nombre+" está despegando.";
}

@Override
public String aterrizar() {
    return "La nave militar "+super.nombre+" está aterrizando.";
}

@Override
public String solicitarAterrizar() {
    String mensaje="Nave militar "+super.nombre+" solicitando permiso para aterrizar...";
    return mensaje;
}

@Override
public String solicitarDespegar() {
    String mensaje="Nave militar "+super.nombre+" solicitando permiso para despegar...";
    return mensaje;
}
}
```

# HERENCIA

```
public class NaveMilitar extends Nave{  
    private int armamento;  
}
```

# HERENCIA Y CONSTRUCTORES ( VACÍ O Y PARAMETRIZADO)

```
public class NaveMilitar extends Nave{  
    private int armamento;  
  
    public NaveMilitar() {  
        super();  
        this.armamento=0;  
    }  
    public NaveMilitar(int codigo, String nombre,int capacidad, boolean estado, int armamento) {  
        super(codigo, nombre, capacidad, estado);  
        this.armamento=armamento;  
    }  
}
```

# CLASES ABSTRACTAS

```
public abstract class Nave implements ControlTorre{
    protected int codigo;
    protected String nombre;
    protected int capacidad;
    protected boolean estado;
    private Controller controller;

    public Nave() {
        this.codigo=0;
        this.nombre="";
        this.capacidad = 0;
        this.estado=true;
    }
    public Nave(int codigo, String nombre,int capacidad, boolean estado) {
        this.codigo=codigo;
        this.nombre=nombre;
        this.capacidad = capacidad;
        this.estado=estado;
    }

    public abstract String despegar();
    public abstract String aterrizar();
}
```

# INTERFACES

```
3 public interface ControlTorre {  
4     String solicitarAterrizar();  
5     String solicitarDespegar();  
6  
7  
8 }  
3 public abstract class Nave implements ControlTorre{  
4     protected int codigo;  
5     protected String nombre;  
6     protected int capacidad;  
7     protected boolean estado;  
8     private Controller controller;  
9
```

# EXCEPCIONES

```
public boolean validarCodigoNave(String codigo) {  
    try {  
        // Intenta convertir el código a un número  
        int codigoNave = Integer.parseInt(codigo);  
        // Verifica si el código convertido es positivo  
        return codigoNave >= 0;  
    } catch (NumberFormatException e) {  
        // Si ocurre una NumberFormatException, significa que el código no es un número válido  
        return false;  
    }  
}
```



# INTERFACES DE USUARIO

Control de Torre de Naves Espaciales

Estado: Código: 1112 | Nombre: Transport Shuttle

Ingrese el código de la nave:

Código: 1112  
Nombre: Transport Shuttle  
La capacidad de la nave es de: 690.  
La nave está: volando.  
Nave de transporte Transport Shuttle solicitando permiso para aterrizar.

La nave de transporte Transport Shuttle está aterrizando.

Transport Shuttle || Estado: Sin armamento  
Carga Máxima: 1098  
Cantidad de mercancía: 1098

# Componentes

```
private JFrame frame;  
private JTextArea textArea;  
private JTextArea textArea1;  
private JTextArea textArea3;  
private JButton btnMostrar;  
private JButton btnDespegar;  
private JButton btnAterrizar;  
private JButton btnListado;  
private JButton btnValidar; // Botón de validación  
private Controler controler;  
private JTextField txtCodigo;  
private JButton btnArmamento;  
private JTextArea txtArmamento;  
private JLabel voladorLabel;  
  
private JButton btnCargarDescargarMercancia;
```

# Contenedores

```
private JFrame frame;
```

```
    JPanel panel = new JPanel();
```

```
    panel.setLayout(null);
```

```
    panel.setBounds(88, 500, 1500, 600);
```

```
    frame.getContentPane().add(panel);
```

# Contenedores

```
private JFrame frame;
```

```
    JPanel panel = new JPanel();
```

```
    panel.setLayout(null);
```

```
    panel.setBounds(88, 500, 1500, 600);
```

```
    frame.getContentPane().add(panel);
```

# Contenedores de eventos

```
btnArmamento.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        activarArmamento();  
    }  
});
```



¡ Muchas Gracias !

“Este es el camino.”

