

1. En este ejercicio utilizamos un software capaz de transformar texto simple en diagramas, para llevar a cabo el ejercicio investigué cómo funcionaba la aplicación y qué debería escribir para que lo que yo quisiese mostrar en pantalla apareciese.

Los caracteres `o--` y `<|--` escritas entre clases creaban relaciones y “class” se utilizaba para guardar la información de las clases. Escribí el código o texto según se iba creando el diagrama a modo de apoyo.

Es una muy buena herramienta para observar el flujo de trabajo al crear clases o otros tipos de diagrama.

@startuml

Almacén o-- Bebida

Bebida <|-- Refresco

Bebida <|-- Cerveza

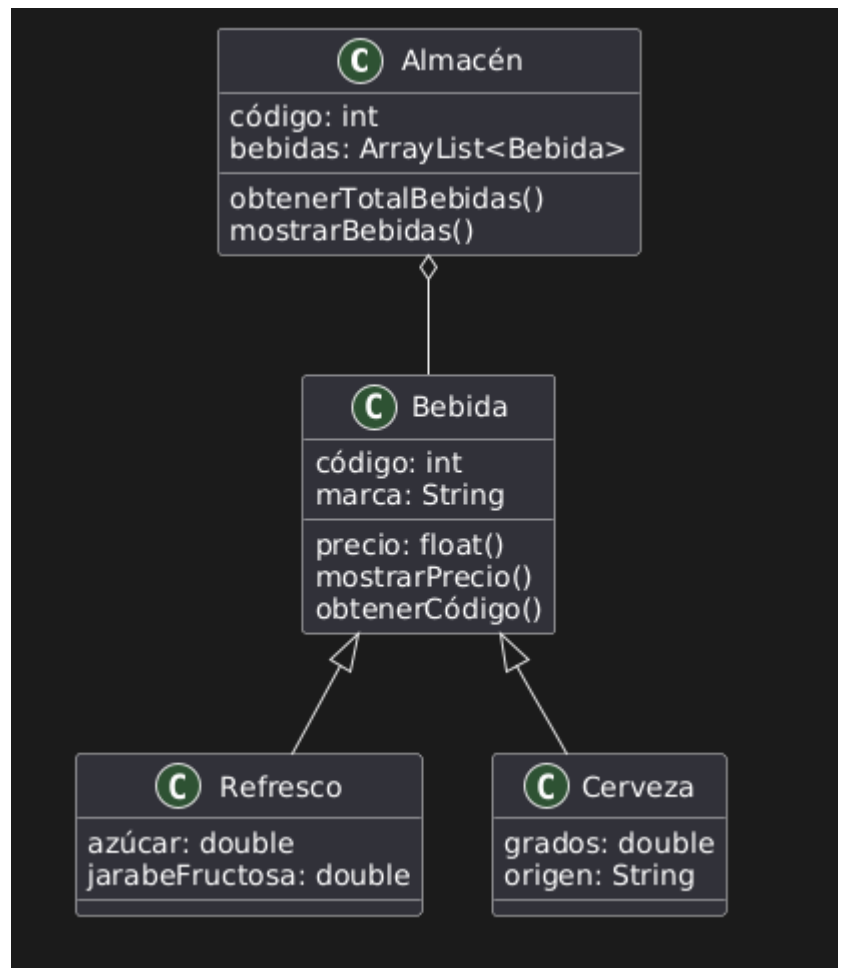
```
class Almacén {  
    código: int  
    bebidas: ArrayList<Bebida>  
    obtenerTotalBebidas()  
    mostrarBebidas()  
}
```

```
class Bebida {  
    código: int  
    marca: String  
    precio: float()  
    mostrarPrecio()  
    obtenerCódigo()  
}
```

```
class Refresco {  
    azúcar: double  
    jarabeFructosa: double  
}
```

```
class Cerveza {  
    grados: double  
    origen: String  
}
```

@enduml



YouTube | Tauler | AULES | Curso: Entornos de | | 2.1 Documentación | | Práctica 1. Probando | | Class Diagram synta | | PlantUML Web Serv | | Entornos\_UD2\_Prac | | que es plantuml - B | | + |

https://www.plantuml.com/plantuml/dum/5oWklmgASiDuN8ElmKDZ1LoDVLO2mgRW5KD0W780

LG OLED LG OLED LG OLED LG OLED  
LG 65 pulgadas TV... 4.499 € 1.029 €  
LG 75 pulgadas Smart... 999,01 € 2.599 €

VILLAGE BOOK  
JUST \$1 A MONTH GIVES LIBRARY ACCESS  
TO A STUDENT FOR AN ENTIRE YEAR  
DONATE

We believe

```
@startuml  
Almacén o-- Bebida  
Bebida <|-- Refresco  
Bebida <|-- Cerveza  
  
class Almacén {  
    código: int  
    bebidas: ArrayList<Bebida>  
    obtenerTotalBebidas()  
    mostrarBebidas()  
}  
  
class Bebida {  
    código: int  
    marca: String  
    precio: float()  
    mostrarPrecio()  
    obtenerCódigo()  
}  
  
class Refresco {  
    azúcar: double  
    jarabeFructosa: double  
}  
  
class Cerveza {  
    grados: double  
    origen: String  
}  
  
@enduml
```

Submit | Pure Javascript | PNG SVG ASCII Art

Página 1

https://www.plantuml.com/plantuml/dpng/RP111Min4M1VDFQc8kgmcouh5w9Ue98KD94a9LNLkgCuoBKsJtCduTktHtLlly\_vfIr51ipdRfup--5c0nk9X531Ks0xKj5ztGKK1z\_ub7axx0T7wXAVnTrns0Ft-5E70F1LYLReff31IHV8sD0j1M1 Decode URL

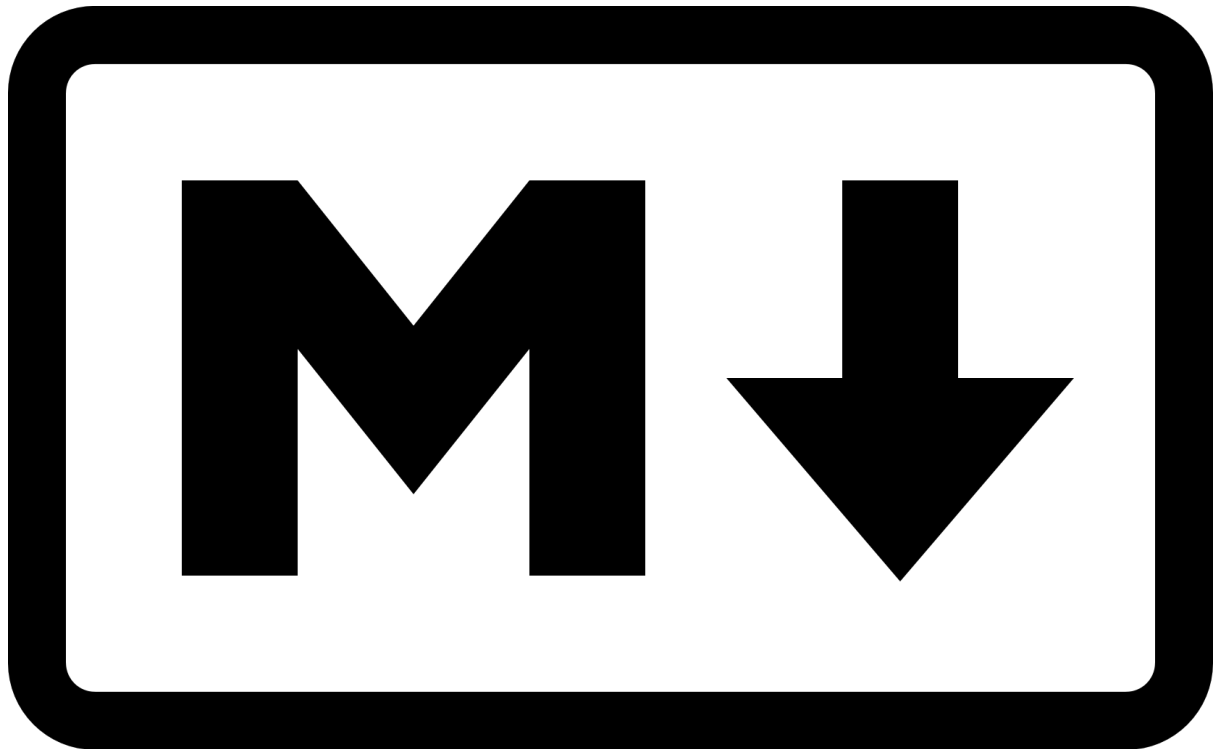
```
classDiagram  
    class Almacén {  
        código: int  
        bebidas: ArrayList<Bebida>  
        obtenerTotalBebidas()  
    }  
    class Bebida {  
        código: int  
        marca: String  
        precio: float()  
        mostrarPrecio()  
        obtenerCódigo()  
    }  
    class Refresco {  
        azúcar: double  
        jarabeFructosa: double  
    }  
    class Cerveza {  
        grados: double  
        origen: String  
    }  
    Almacén o--> Bebida  
    Bebida <|-- Refresco  
    Bebida <|-- Cerveza
```

2. En la siguiente práctica usamos Markdown, un lenguaje de marcado para formatear en texto plano, muy comúnmente usado para mostrar la documentación técnica de un programa.

En primer lugar instalé VS Code y trasteando un poco puse un proyecto con el lenguaje de marcado Markdown. Para previsualizar el contenido instalé una extensión.

Seguí las instrucciones para hacer títulos, listas ordenadas, insertar links e insertar bloques de código hasta que tuve el ejercicio hecho.

Me parece un lenguaje de marcado muy bueno y en el futuro lo usaría para dar información sobre un programa hecho por mí o simplemente usarlo para tener una lista ordenada de cosas.



# Mi Proyecto Increíble

Este es un proyecto increíble que hace cosas sorprendentes.

## Descripción

Este proyecto realiza tareas asombrosas y útiles. Aquí hay una breve descripción de lo que puede hacer:

- Realiza cálculos avanzados en segundos.
- Genera gráficos impresionantes en tiempo real.
- ¡Hace tu vida más fácil!

## Instalación

Para utilizar este proyecto, sigue estos pasos:

1. Clona este repositorio: `git clone https://github.com/tuusuario/mi-proyecto-increible.git`
2. Ve al directorio del proyecto: `cd mi-proyecto-increible`
3. Instala las dependencias: `npm install`

## Uso

Aquí tienes un ejemplo de cómo usar este proyecto:

```
const proyectoIncreible = require('proyecto-increible');

// Realiza un cálculo asombroso
const resultado = proyecto.increible.calcular(42);

console.log(resultado);
```

### # Mi Proyecto Increíble

Este es un proyecto increíble que hace cosas sorprendentes.

#### ## Descripción

Este proyecto realiza tareas asombrosas y útiles. Aquí hay una breve descripción de lo que puede hacer:

- Realiza cálculos avanzados en segundos.
- Genera gráficos impresionantes en tiempo real.
- ¡Hace tu vida más fácil!

#### ## Instalación

Para utilizar este proyecto, sigue estos pasos:

1. Clona este repositorio: `git clone https://github.com/tuusuario/mi-proyecto-increible.git`
2. Ve al directorio del proyecto: `cd mi-proyecto-increible`
3. Instala las dependencias: `npm install`

#### ## Uso

Aquí tienes un ejemplo de cómo usar este proyecto:

~~~

```
const proyectoIncreible = require('proyecto-increible');
```

```
// Realiza un cálculo asombroso
```

```
const resultado = proyecto.increible.calcular(42);
```

```
console.log(resultado);
```

~~~

3. A continuación hacemos uso de RAML, un lenguaje de modelado diseñado para describir APIs RESTful.

Esta aplicación ha sido un poco más complicada de entender, nunca había utilizado una aplicación parecida, he tenido también que buscar ejemplos para facilitar el aprendizaje.

Cambié la versión a “v4”, “/Items’ por Personajes de Star Wars; asigné un personaje a cada “/{ItemID}” y cambié su “description:”, puse la principal descripción de los personajes Yoda y Anakin. Finalmente introduje un “delete:” a cada personaje cuya respuesta fuese 204 (“204:”)

Me parece útil pero complicada, requiere algo de tiempo para aprender su metodología y sintaxis, pero puede ser interesante su uso para definir este tipo de APIs.

Star Wars

API Title

Star Wars

Version

v4

Base URI

http://api.ejemplo.com/{version}

Endpoints

/Personajes de Star Wars

/Personajes de Star Wars/{Yoda} GET DELETE

/Personajes de Star Wars/{Anakin} GET DELETE

GET

↓

↓

↓

↓

Star Wars / Personajes de Star Wars / {Yoda} / delete

Endpoint URL

http://api.ejemplo.com/v4/Personajes de Star Wars/{Yoda}

Code samples

URI parameters

version

String Required

Yoda

String Required

Responses

< 204 >

Body

Media type: application/json

Any instance of data is allowed.

The API file specifies body for this request but it does not specify the data model.

Star Wars / Personajes de Star Wars / {Yoda} / get

Endpoint URL

http://api.ejemplo.com/v4/Personajes de Star Wars/{Yoda}

Code samples

URI parameters

version

String Required

Yoda

String Required

Lo que ocurre si pulso



DELETE

#### Endpoint URL

http://api.ejemplo.com/v4/Personajes de Star Wars/{Yoda}

#### Code samples

#### URI parameters

version

String Required

Yoda

String Required

#### Responses

< 204 >

#### Body

Media type: application/json

Any instance of data is allowed.

The API file specifies body for this request but it does not specify the data model.

```

#%RAML 1.0
title: Star Wars
baseUri: http://api.ejemplo.com/{version}
version: v4

uses:
  assets: assets.lib.raml

annotationTypes:
  monitoringInterval:
    type: integer

/Personajes de Star Wars:
  /{Yoda}:
    get:
      description: Yoda (896 ABY-Dagobah, 4 DBY), uno de los más renombrados
y poderosos maestros Jedi durante toda la historia de la Galaxia.
      responses:
        200:
          body:
            application/json:
              type: String
    delete:
      responses:
        204:
          body:
            application/json:
              type: Item
  /{Anakin}:
    get:
      description: Anakin Skywalker (Tatooine, 42 ABY - Estrella de la
Muerte, 4 DBY), más tarde Darth Vader.
      responses:
        200:
          body:
            application/json:
              type: String
    delete:
      responses:
        204:
          body:
            application/json:
              type: Item

```

4. En esta ocasión (para mí) lo más difícil ha sido la instalación y saber usar el programa. He instalado PHPDocumentor a través de Docker; Docker crea una máquina virtual en la que puedes iniciar programas y ejecutar comandos.

Una vez instalado PHPDocumentor, busqué un ejemplo escrito en php con clases para documentarlo (el código también tenía métodos).

To run phpDocumentor using docker, the following should suffice:

```
$ docker run --rm -v "$(pwd):/data" "phpdoc/phpdoc:3"
```

ESCRIBI EN DOCKER. `docker run --rm -v $(pwd):/data phpdoc/phpdoc:3 -f C:\Users\david\Downloads\Frutas.php -t C:\Users\david\Downloads`

Este comando inicia PHPDocumentor y recoge mi documento php (Frutas.php) y envía el documento resultante a una carpeta contenedora (david\downloads)

**-d**

specifies the directory, or directories, of your project that you want to document.

**-f**

specifies a specific file, or files, in your project that you want to document.

**-t**

specifies the location where your documentation will be written (also called 'target folder').

The above options are all you need to generate your documentation as demonstrated in this example:

```
$ phpdoc -d path/to/my/project -f path/to/an/additional/file -t path/to/my/output/folder
```

Una vez hecho, te da la documentación en un .html, el cual abres tienes información sobre las clases, sus propiedades (Properties) y los métodos (Methods), así como reportes relacionados con tu archivo, como posibles errores en el proyecto.

Una muy útil herramienta para documentar PHP una vez sabes cómo funciona y ya la tienes instalada. El año que viene, cuando tenga la asignatura de servidor seguramente la use de nuevo.

File | C:/Users/david/Downloads/.phpdoc/build/classes/Fruit.html

Documentation

Packages

Application

Reports

Deprecated

Errors

Markers

Indices

Files

Application

**Fruit**

in package Application

Table of Contents

Properties

P

 \$color : mixed

P

 \$name : mixed

Methods

M

 get\_color() : mixed

M

 get\_name() : mixed

M

 set\_color() : mixed

M

 set\_name() : mixed

Properties

\$color

public mixed \$color

\$name

public mixed \$name

Methods

get\_color()

public get\_color() : mixed

get\_name()

public get\_name() : mixed

Página 6

Frutas.php : 11

```

<?php
class Fruit {
    // Properties
    public $name;
    public $color;

    // Methods
    function set_name($name) {
        $this->name = $name;
    }
    function get_name() {
        return $this->name;
    }
    function set_color($color) {
        $this->color = $color;
    }
    function get_color() {
        return $this->color;
    }
}

// Creating objects
$apple = new Fruit();
$banana = new Fruit();

// Setting properties
$apple->set_name('Apple');
$apple->set_color('Red');
$banana->set_name('Banana');
$banana->set_color('Yellow');

// Getting properties
echo "Name: " . $apple->get_name() . ", Color: " .
$apple->get_color();
echo "<br>";
echo "Name: " . $banana->get_name() . ", Color: " .
$banana->get_color();
?>

```