**El README es un archivo con extensión .md, es decir, está escrito en Markdown, que es un lenguaje de marcado utilizado para convertir texto en HTML válido.**

**En plataformas de repositorios remotos como  GitHub, la función de este archivo es presentar información del proyecto, como:**

* **Título del Proyecto**
* **Descripción de su proyecto**
* **Funcionalidades**
* **Uso**
* **Contribuir**
* **Historia**

**EJEMPLO**

**1. Clase Barco**

**Barco**

Este es un proyecto Java que define la clase Barco que extiende la clase Vehiculo.

**1.1. Instalación**

Si hubiese un método para instalar o ejecutar el proyecto, se indicaría aquí.

**1.2. Descripción**

La clase Barco representa un vehículo acuático y se utiliza para instanciar objetos que contienen información sobre el color, la marca, el precio, la matrícula y el tipo de barco. La clase también tiene un método para arrancar el barco y establecer la hora de inicio del viaje. Funcionalidades

**1.3. Funcionalidades**

La clase Barco tiene las siguientes funcionalidades:

Constructor: Crea una instancia de Barco con un color, marca, precio, matrícula y tipo de barco.

getTipoBarco(): Devuelve el tipo de barco.

arrancar(): Arranca el barco y establece la hora de inicio del viaje. Si el barco ya está en marcha, muestra un mensaje de aviso.

retroceder(int metros): Retrocede el barco la cantidad de metros indicada. Si el barco no está en marcha, muestra un mensaje de aviso.

parar(): Para el barco y muestra la hora de inicio y fin del viaje. Si el barco no está en marcha, muestra un mensaje de aviso.

avanzar(int distancia): Avanza el barco la cantidad de metros indicada. Si el barco no está en marcha, muestra un mensaje de aviso.

imprimirDatos(): Muestra los datos del barco.

Barco barco = new Barco("Azul", "Marca", "5000", "1234ABC", "Tipo de Barco");

String tipoBarco = barco.getTipoBarco();

barco.arrancar();

**1.4. Uso**

**Uso de la clase Barco**

Para utilizar la clase Barco, primero debe instanciar un objeto pasando los argumentos adecuados al constructor. Luego, puede llamar a los métodos getTipoBarco() y arrancar() según sea necesario.

**1.5. Contribuir**

1. Fork it!
2. Crea tu feature branch: git checkout -b my-new-feature
3. Haz Commit de tus cambios: git commit -am 'Add some feature'
4. Push al branch: git push origin my-new-feature
5. Submit un pull request

**1.6. Historia**

* Version 0.1 (2023-04-19) - lanzamiento inicial
* Version 0.2 (2023-04-20) - añadido método avanzar()
* Version 0.3 (2023-04-21) - añadido método retroceder()
* Version 0.4 (2023-04-22) - añadido método parar()
* Version 0.5 (2023-04-23) - añadido resto de métodos
* Version 0.6 (2023-04-24) - Pruebas unitarias y correcciones

**1.7. Créditos y atribuciones**

Lead Developer - Mentor - Pepito

**1.8. License**

The MIT License (MIT)

Copyright (c) 2015 Chris Kibble

**EJEMPLO 2**

**- API REST Cursos-Formación**

Este repositorio contiene dos microservicios desarrollados en Spring Boot para gestionar cursos. La aplicación utiliza una base de datos MySQL de la que se adjuntan los scripts en el repositorio. Expone sus funcionalidades a través de una API REST que se muestran a traves del navegador.

**Características Principales**

* Arquitectura basada en microservicios
* Uso base de datos MySQL.
* Relaciona el microservicio reserva con el microservicio de vuelo y con el microservicio de hotel.
* Exposición de endpoints para interactuar con todas las entidades.

**Tecnologías Utilizadas**

* **Spring Boot**: Infraestructura para microservicios eficiente.
* **Thymeleaf**: Motor de plantillas para vistas HTML.
* **Swagger**: Documentación y visualización de API RESTful.
* **JUnit**: Marco para pruebas unitarias en Java.

**Inicio Rápido**

1. **Clonar el repositorio**: git clone https://github.com/xxx.git
2. **Acceder al directorio del proyecto:** cd xxx

**Entidades**

1. **Curso:** Representa a un curso con las propiedades código de curso, nombre, duración y precio.
2. **Formación:** Representa una formación con las propiedades curso, asignaturas y precio**.**

**Servicio Curso**

Este microservicio proporciona funcionalidades para la gestión de cursos

**Funcionalidades**

1. Listar todos los cursos
2. Crear un nuevo curso

**Endpoints**

**Puerto 8080**

1. **Listar todos los cursos**
   * **URL: /cursos**
   * **Método: GET**
2. **Crear un nuevo curso**
   * **URL: /cursos**
   * **Método: POST**

**Servicio Formación**

Este interaccionará con el servicio de cursos para ofrecer su funcionalidad. Los datos que caracterizan una formación serán:

curso (texto)

asignaturas (numérico entero)

precio (numérico decimal)

**Funcionalidades**

* Devuelve la lista de cursos existentes. Como el microservicio de cursos no dispone del dato de asignaturas, se seguirá el siguiente algoritmo: Si un curso tiene una duración igual o superior a 50 horas, se considera que tiene 10 asignaturas, sino, serán 5 asignaturas
* Alta curso. A partir de los datos de formación recibidos en el cuerpo, se dará de alta el curso a través del microservicio de cursos, siempre que no exista un curso con el mismo nombre, en cuyo caso no se hará nada. La duración del curso se establecerá según la fórmula: asignaturas\*10

**Endpoints**

* **Puerto 9090**

1. **Listar todas las formaciones**
   * **URL: /formaciones**
   * **Método: GET**
2. **Dar de alta una formación**
   * **URL: /formaciones**
   * **Método: POST**

[**https://docs.github.com/es/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax**](https://docs.github.com/es/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax)