08GIIN Metodología de Programación

José Manuel Galván Díaz

Contenido

1. Menú principal

El proyecto cuenta con una base de datos. Los datos se encuentran registrados en documentos JSON dentro del directorio system_data.

Al iniciar el programa se cargan los datos de la base de datos y se muestra el menú principal.

1.1 Tipos de cohetes

1.1.1 Introducir nuevo cohete

En esta pantalla se le pide al usuario que indique:

- Un identificador único válido, este identificador se cotejará con los ya establecidos en la base de datos para asegurar que sean únicos. Como mínimo el identificador debe de constar de 3 caracteres. A la hora de procesar el identificador se realiza una limpieza del texto, eliminando espacios, sustituyendo las vocales acentuadas y las mayúsculas. Por ejemplo, “Falcón 3” después del proceso de limpieza quedaría de la siguiente forma “falcon3”.
- Carga útil (OTB) en Kg. Se comprueba que la carga sea mayor que cero y se introduzca un valor transformable a entero. Para facilitar las operaciones relacionadas con el peso, en la base de datos todos datos que hagan referencia a esto se guardan en gramos. El peso máximo establecido es de 99999Kg

1.1.2 Ver cohetes

En esta pantalla se muestran todos los tipos de cohetes existentes.

Como el listado de tipos de cohetes se muestra en esta opción, en la opción “Información avanzada” no se da la posibilidad de visualizarlo, para no tener funcionalidades repetidas dentro del proyecto.

1.1.3 Salir

El usuario vuelve al menú principal

1.2 Peticiones de la estación

1.2.1 Introducir nueva petición

En esta pantalla se le pide al usuario que indique:

- Un identificador único válido, este identificador se cotejará con los ya establecidos en la base de datos para asegurar que sean únicos. Como mínimo el identificador debe de constar de 3 caracteres. A la hora de procesar el identificador se realiza una limpieza del texto, eliminando espacios, sustituyendo las vocales acentuadas y las mayúsculas. Por ejemplo, “r8 P” después del proceso de limpieza quedaría de la siguiente forma “r8p”.
- Peso Kg. Se comprueba que la carga sea mayor que cero y el valor no contenga más de tres decimales. En caso de que un usuario introduzca por error una coma en lugar de un punto, al ser un número decimal, se sustituye esa coma por un punto para poder procesar el valor. Para facilitar las operaciones relacionadas con el peso, en la base de datos todos datos que hagan referencia a esto se guardan en gramos. El peso máximo establecido es de 99999Kg
- Descripción de la petición, este campo acepta una cadena de entre 3 y 100 caracteres
- Días máximos. Se comprueba que sea un número entero superior a cero. Como máximo se podrán establecer 365 días.

1.2.2 Ver peticiones

En esta pantalla se muestran solo las peticiones que están pendientes en espera.

Para visualizar todas las peticiones de la base de datos, se accede a través del menú “Información avanzada” apartado “Peticiones”

1.2.3 Salir

El usuario vuelve al menú principal

1.3 Lanzamientos disponibles

1.3.1 Introducir nuevo lanzamiento

En esta pantalla se le pide al usuario que indique:

- Los lanzamientos disponen de un identificador único, en este caso el usuario no indicará este identificador, lo generará el sistema a partir de la fecha actual pasando la fecha al siguiente formato “YYYYMMDDhhmmss”, por si existiese la posibilidad de que coincidan la creación de dos lanzamientos en el mismo instante, el sistema comprueba que el identificador no este registrado.
- Un tipo de cohete, se mostrará un menú con la lista de tipos de cohetes disponibles, el usuario introducirá la opción del menú correspondiente al tipo de cohete que quiere seleccionar.
- Días para la estación. Se comprueba que sea un número entero superior a cero. Como máximo se podrán establecer 365 días.

1.3.2 Ver lanzamientos

En esta pantalla se muestran solo los lanzamientos que están pendientes en espera.

Para visualizar todas los lanzamientos de la base de datos, se accede a través del menú “Información avanzada” apartado “Lanzamientos”

1.3.3 Salir

El usuario vuelve al menú principal

1.4 Asignar

Al seleccionar esta opción el sistema automáticamente asignará las peticiones sin asignar, en los lanzamientos disponibles.

A la hora de realizar la asignación, se han ordenado tanto las peticiones como los lanzamientos, en base a los días establecidos para cada uno de forma ascendente.

En caso de que al realizar la ordenación coincidan el número de días, se procederá de la siguiente forma:

- Peticiones: Se ordenara de forma descendente tomando como referencia el peso.
- Lanzamientos: Se ordenara de forma ascendente tomando como referencia su carga máxima.

De esta forma intentamos que los lanzamientos se rellenen con la mayor carga posible.

A medida que se vayan asignando las peticiones a los lanzamientos, se irán mostrando por pantalla estas asignaciones.

En caso de quedar alguna petición sin asignar, se mostrará un mensaje indicándolo.

1.5 Días

Al seleccionar esta opción, el sistema pedirá al usuario que indique el número de días que quiere avanzar. Este valor debe de ser un número entero superior a cero y como máximo 365.

Una vez establecido, se irá incrementando el contador del día actual en el sistema en base a la cantidad de días indicados por el usuario y se realizara este proceso cada día:

1º Se registra el nuevo día en la base de datos, creando su correspondiente JSON.

2º Se resta uno a los días máximos de las peticiones.

En caso de llegar a cero y no estar asignada a ningún lanzamiento, la petición queda cancelada.

3º Se resta uno a los días de los lanzamientos.

Si el lanzamiento aún no había despegado se revisa el número de peticiones que tiene asignado. Si no tiene asignada ninguna petición se cancela el lanzamiento (reseteamos el número de días a cero), ya que no se va a realizar el envío de un lanzamiento sin carga. Si tiene asignado alguna carga su estado pasa a “En tránsito”.

Si el lanzamiento ya había despegado y el contador llega a cero, entonces establecemos el estado como “Entregado”. Posteriormente recorremos los pedidos asignados al lanzamiento para cambiar su tiempo máximo a cero, indicando que ya ha sido entregado.

4º Guardamos todos los registros del día. En un principio se consideró obviar la generación de un fichero de registro para los días en los que no se hubiese registrado nada, pero la ausencia de eventos también es un dato a considerar, así que también se guardan los días sin registros.

1.6 Información avanzada

1.6.1 Peticiones

Esta opción muestra todas las peticiones registradas en el sistema y su estado actual.

Los posibles estados de una petición son los siguientes:

- **En tránsito**: La petición ya ha sido asignada a un lanzamiento.
- **Sin asignar**: La petición no está asignada a ningún lanzamiento.
- **Entregado**: La petición ha llegado a la estación.
- **Cancelado**: La petición no ha sido asignada a ningún lanzamiento y el tiempo para el envío se ha agotado.

1.6.2 Lanzamientos

Esta opción muestra todos los lanzamientos registrados en el sistema y su estado actual.

Los posibles estados de un lanzamiento son los siguientes:

- **En tránsito**: El lanzamiento ya ha sido enviado.
- **En espera**: El lanzamiento está en espera de ser asignado y enviado.
- **Entregado**: El lanzamiento ha llegado a la estación.
- **Cancelado**: El lanzamiento no tiene asignada ninguna petición, se cancela para no realizar un envío vacío.

1.6.3 Historial

Muestra todos los eventos registrados en la base de datos. Los eventos están agrupados por días, el sistema pide al usuario que indique de qué día quiere consultar los registros.

1.6.4 Salir

El usuario vuelve al menú principal

1.7 Configuración

1.7.1 Cambiar idioma

Esta opción permite al usuario cambiar el idioma del sistema, los idiomas disponibles actualmente son español e inglés.

El cambio de idioma del sistema, queda guardado en la configuración como idioma predeterminado en futuras sesiones.

1.7.2 Salir

El usuario vuelve al menú principal

1.8 Salir

Finaliza el programa.

Como todas las interacciones del usuario se registran y guardan automáticamente dentro de la base de datos, no es necesario realizar ningún proceso cuando el usuario decide finalizar el programa.

2. Clases

Estos archivos y directorios se encuentran dentro del directorio **classes** del proyecto

2.1 Rocket

NOMBRE DE FUNCIÓN	PARÁMETROS	RETURN
__init__: Inicializa la clase	• **name (String):** Identificador del cohete • **shipload (Integer):** Carga máxima del cohete	
__str__: Retorna los datos del cohete para imprimirlos por pantalla		**String**
getId: Retorna el identificador del cohete		**String**
setId: Actualiza el identificador del cohete	• **name (String):** Nuevo identificador para el cohete	
getShipload: Retorna la carga máxima del cohete		**Integer**
setShipload: Actualiza la carga máxima del cohete	• **shipload (Integer):** Nueva carga máxima para el cohete	
getRocketData: Devuelve un objeto con los datos del cohete		**dict**

2.2 Request

NOMBRE DE FUNCIÓN	PARÁMETROS	RETURN
__init__: Inicializa la clase	• **name (String):** Identificador de la petición • **weight (Integer):** Peso de la petición • **description (String):** Descripción de la petición • **time_max (Integer):** Tiempo máximo de la petición • **dispatched (Boolean):** Estado de la petición	
__str__: Retorna los datos de la petición para imprimirlos por pantalla		**String**
getId: Retorna el identificador de la petición		**String**
setId: Actualiza el identificador de la petición	• **name (String):** Nuevo identificador para la petición	
getWeight: Retorna peso de la petición		**Integer**
setWeight: Actualiza el peso de la petición	• **weight (Integer):** Nueva peso de la petición	
getDescription: Retorna la descripción de la petición		**String**
setDescription: Actualiza la descripción de la petición	• **description (String):** Nueva descripción de la petición	
getTimeMax: Retorna tiempo máximo de la petición		**Integer**
setTimeMax: Actualiza el tiempo máximo de la petición	• **time_max (Integer):** Nuevo tiempo máximo de la petición	
getDispatched: Retorna el estado de la petición		**Boolean**
setDispatched: Actualiza el estado de la petición	• **dispatched (Boolean):** Nuevo estado de la petición	
getRequestData: Devuelve un objeto con los datos de la petición		**dict**

2.3 Launch

NOMBRE DE FUNCIÓN	PARÁMETROS	RETURN
__init__: Inicializa la clase	• **name (String):** Identificador del lanzamiento • **id_rocket (String):** Identificador del cohete del lanzamiento • **max_weight (Integer):** Carga máxima del lanzamiento • **shipload (Integer):** Carga actual del lanzamiento • **requests ([String]):** Peticiones del lanzamiento • **time (Integer):** Tiempo máximo del lanzamiento • **dispatched (Boolean):** Estado del lanzamiento	
__str__: Retorna los datos del lanzamiento para imprimirlos por pantalla		**String**
getId: Retorna el identificador del lanzamiento		**String**
setId: Actualiza el identificador del lanzamiento	• **name (String):** Nuevo identificador para el lanzamiento	
getIdRocket: Retorna el identificador del cohete del lanzamiento		**String**
setIdRocket: Actualiza el identificador del cohete del lanzamiento	• **id_rocket (String):** Nuevo identificador para el cohete del lanzamiento	
getMaxWeight: Retorna el peso máximo del lanzamiento		**Integer**
setMaxWeight: Actualiza el peso máximo del lanzamiento	• **max_weight (Integer):** Nueva peso máximo del lanzamiento	
getShipload: Retorna la carga de la petición		**Integer**
setShipload: Actualiza la carga de la petición	• **shipload (Integer):** Nueva carga de la petición	
getRequests:		**[String]**

Retorna las peticiones del lanzamiento	
setRequests: Actualiza las peticiones del lanzamiento	• **requests ([String]):** Nuevos peticiones del lanzamiento
getTime: Retorna tiempo del lanzamiento	**Integer**
setTime: Actualiza el tiempo del lanzamiento	• **time (Integer):** Nuevo tiempo del lanzamiento
getDispatched: Retorna el estado del lanzamiento	**Boolean**
setDispatched: Actualiza el estado del lanzamiento	• **dispatched (Boolean):** Nuevo estado del lanzamiento
getLaunchData: Devuelve un objeto con los datos del lanzamiento	**dict**

2.4 Record

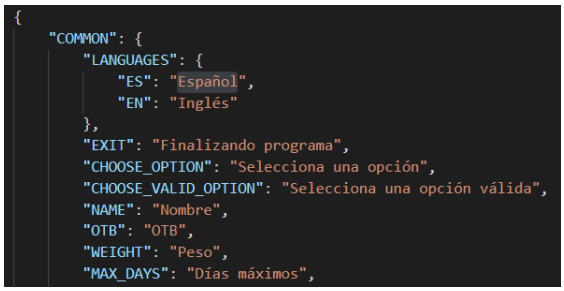
NOMBRE DE FUNCIÓN	PARÁMETROS	RETURN
__init__: Inicializa la clase	• **name (String):** Identificador del registro • **shipload (Integer):** Valor del registro	
__str__: Retorna los datos del registro para imprimirlos por pantalla		**String**
getId: Retorna el identificador del registro		**String**
setId: Actualiza el identificador del registro	• **name (String):** Nuevo identificador para el registro	
getValue: Retorna el valor del registro		**Integer**
setValue: Actualiza el valor del registro	• **value (Integer):** Nuevo valor para el registro	
getRecordData: Devuelve un objeto con los datos del registro		**dict**

3. Base de datos

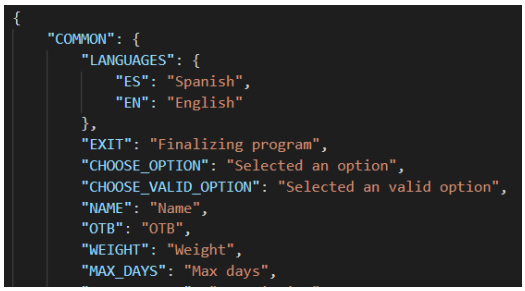
Estos archivos y directorios se encuentran dentro del directorio **system_data** del proyecto

3.1 Languages

3.1.1 {{language}}.json



```
{
    "COMMON": {
        "LANGUAGES": {
            "ES": "Español",
            "EN": "Inglés"
        },
        "EXIT": "Finalizando programa",
        "CHOOSE_OPTION": "Selecciona una opción",
        "CHOOSE_VALID_OPTION": "Selecciona una opción válida",
        "NAME": "Nombre",
        "OTB": "OTB",
        "WEIGHT": "Peso",
        "MAX_DAYS": "Días máximos",
```



```
{
    "COMMON": {
        "LANGUAGES": {
            "ES": "Spanish",
            "EN": "English"
        },
        "EXIT": "Finalizing program",
        "CHOOSE_OPTION": "Selected an option",
        "CHOOSE_VALID_OPTION": "Selected an valid option",
        "NAME": "Name",
        "OTB": "OTB",
        "WEIGHT": "Weight",
        "MAX_DAYS": "Max days",
```

3.2 Records

3.2.1 {{day}}.json

```
{
    "records": [
        {
            "id": String;
            "value": String;
        {
      ]
}
```

3.3 Launchs

```
{
    "launchs": [
        {
            "id": String;
            "id_rocket": String;
            "max_weight": Integer;
            "shipload": Integer;
            "requests": [String];
            "time": Integer;
            "dispatched": Boolean;
        }
      ]
}
```

3.4 Requests

```
{
```

```
    "requests": [
        {
            "id": String;
            "weight": Integer;
            "description": String;
            "time_max": Integer;
            "dispatched": Boolean;
        }
      ]
}
```

3.5 Rockets

```
{
    "rockets": [
        {
            "id": String;
            "shipload": Integer;
        }
      ]
}
```

3.6 Settings

```
{
    "language": String;
    "day": Integer;
}
```

4. Validaciones

Estos archivos se encuentran dentro del directorio **validations** del proyecto

4.1 Float_validations

NOMBRE DE FUNCIÓN	PARÁMETROS	RETURN
isValidFloat: Comprueba si el valor es un número Real, en caso de no serlo, revisa si se puede transformar a Real	• **value (Integer):** Valor a revisar	**Boolean**
valueFloatIncluded: Comprueba si el valor recibido, está comprendido entre el valor mínimo y máximo establecido	• **value (Float):** Valor que se tiene que validar. • **minValue (Float):** Valor mínimo válido, el valor introducido por el usuario debe de ser mayor o igual a este. • **maxValue (Float):** Valor máximo válido, el valor introducido por el usuario debe de ser menor a este. • **decimals (Integer):** Número máximo de decimales.	**Boolean**
minDecimals: Comprueba si el valor recibido, no supera los decimales máximos	• **value (Float):** Valor que se tiene que validar • **decimals (Integer):** Número máximo de decimales	**Boolean**

4.2 Integer_validations

NOMBRE DE FUNCIÓN	PARÁMETROS	RETURN
isValidInteger: Comprueba si el valor es un número entero, en caso de no serlo, revisa si se puede transformar a entero	• **value (Integer):** Valor a revisar	**Boolean**
valueFloatIncluded: Comprueba si el valor recibido, está comprendido entre el valor mínimo y máximo establecido	• **value (Integer):** Valor que se tiene que validar. • **minValue (Integer):** Valor mínimo válido, el valor introducido por el usuario debe de ser mayor o igual a este. • **maxValue (Integer):** Valor máximo válido, el valor introducido por el usuario debe de ser menor a este.	**Boolean**