Memoria

IIA CAFÉ

Cristian Delgado Cruz

Juan Jiménez Serrano

Ismael Da Palma Fernández

ÍNDICE

PUESTA EN PRÁCTICA

PLANTEAMIENTO INICIAL

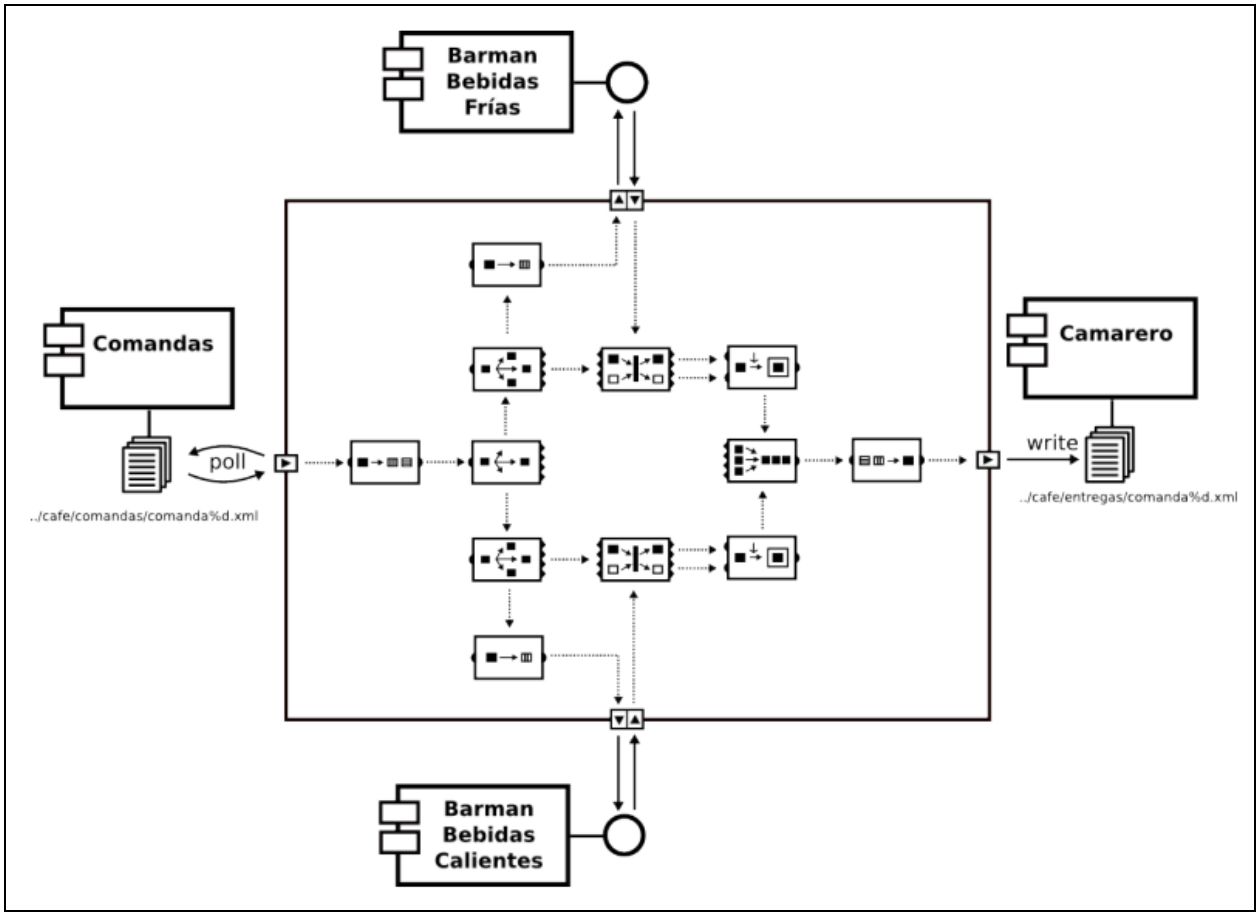


Fig. 1

PROCEDIMIENTO

Diagrama de Clases de la implementación del problema:

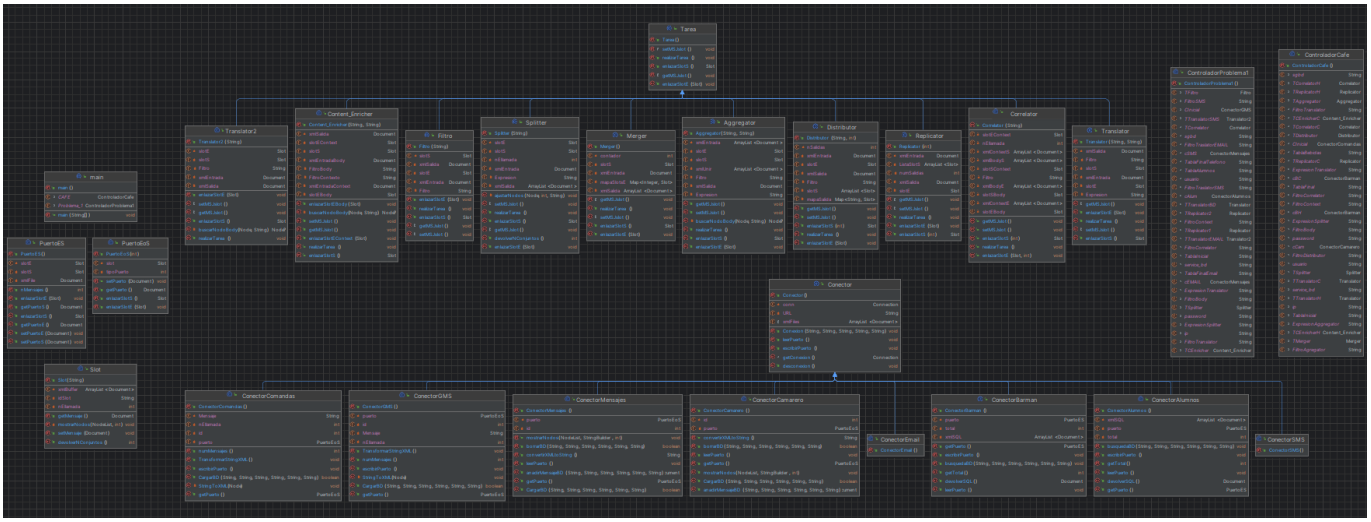


Fig. 2

Diagramas de Secuencia:

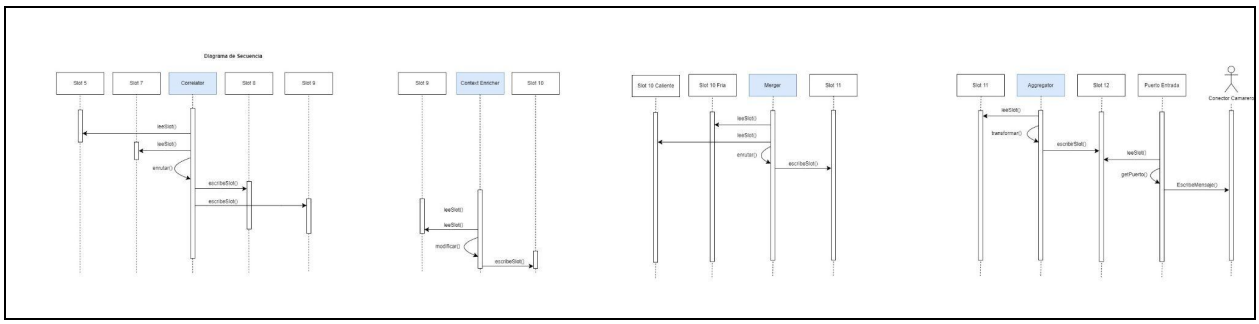


Fig. 3

Diagramas de recorrido:

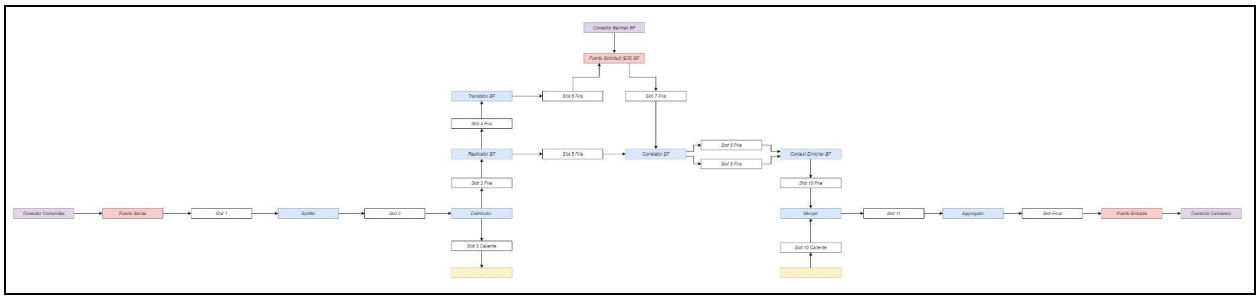


Fig. 4

CODIFICACIÓN DEL PROBLEMA:

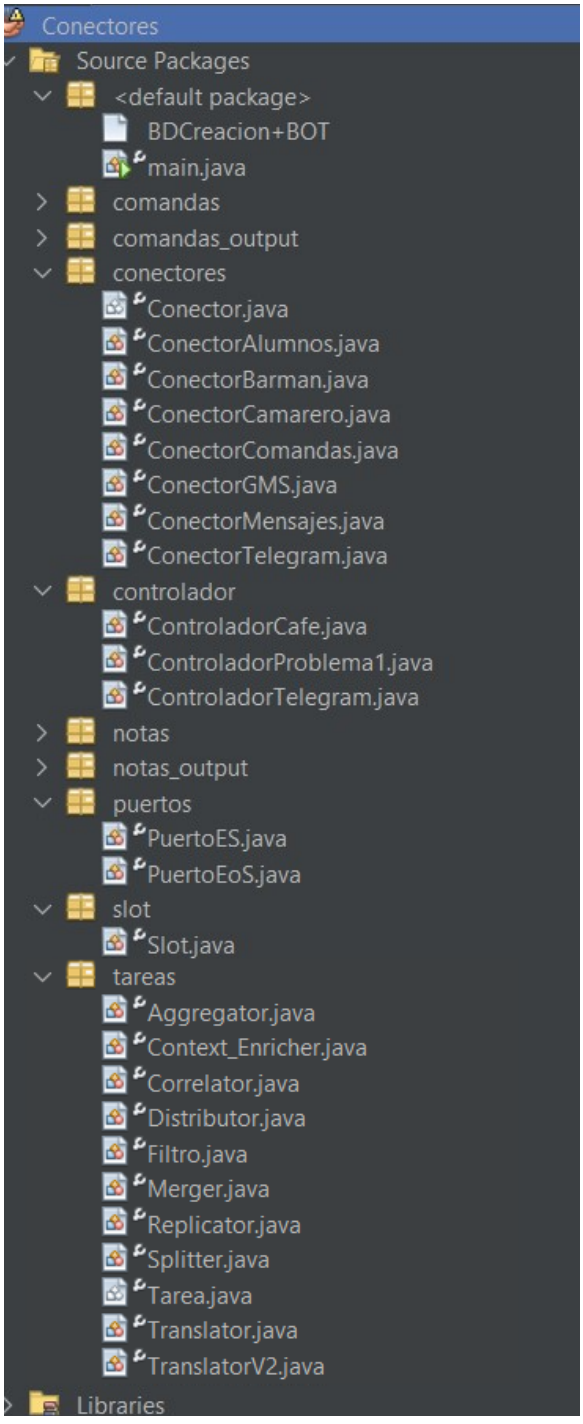


Fig. 5

Al finalizar el proceso de Codificación obtenemos el siguiente árbol de carpetas, donde podemos ver la estructura del proyecto.

En el **paquete default** tenemos el main y las sentencias para crear la BD y conectarse con el BOT de Telegram.

Dentro de **comandas** tenemos todos los datos de entrada respecto al problema *"CAFÉ"*, mientras que en el **output** se colocarán todas las comandas una vez procesadas, mientras que de forma similar **notas** y **output** serán del Problema 1.

Dentro de **conectores** tenemos el Conector que es una *clase abstracta* de la que dependen todos los demás: Barman, Camarero y Comandas del Problema "*CAFÉ*" y Alumnos, GMS y Mensajes son del *Problema 1* de clase y *Telegram* de la unión de "CAFÉ" con el bot.

Tres **controladores** que serán llamados por el Main.

Dos tipos de **puertos**, uno de E/S y otro sólo de Entrada o sólo de Salida.

slot, con una clase llamada Slot donde se colocan los datos que se van a usar.

tareas, donde la clase abstracta Tarea es de la que dependen todas las demás tareas.

A Continuación, procedemos a explicar la funcionalidad de las siguientes clases y paquetes.

CLASES COMUNES:

Main:

El main es un simple menú donde se puede elegir entre los tres problemas que hemos desarrollado, tras ello se llamará al controlador del problema que elijamos.

Puertos:

Los puertos serán creados por los conectores, estos a su vez crearán los slots de salida o recibirán los slots de entrada según sea necesario

PuertoES: (Realizado por Ismael)

Este puerto será usado por aquellos conectores los cuales necesiten recibir y enviar mensajes de las tareas, crean su slot de salida y reciben su slot de entrada.

PuertoEoS: (Realizado por Cristian)

Este puerto podrá ser de entrada, o salida, a la hora de instanciarlo en el controlador se definirá su tipo (1 = Entrada, 2 = Salida).

Slots:

Slot: (Realizada por Juan)

Es una clase intermedia que solo se usa de manera parecida a un buffer, son creados por las tareas o puertos cuando necesitan slots de salida, los de entrada son vinculados con otros slots de salidas ya creados, a parte muestra por pantalla el documento que llega al slot.

Tareas:

Tarea: (Realizada por Cristian)

Es la clase de la cual todas las tareas heredan, tiene los métodos para realizar la tarea, y enlazar los slots, que son públicos, por otra parte tiene los de recoger y enviar mensajes al slot, que son protegidos.

Aggregator: (Realizado por Cristian, Ismael y Juan)

El Aggregator combina documentos XML según una expresión XPATH y un filtro los cuales reciben por parámetros, busca conjuntos de documentos con el mismo filtro. Cuando encuentra un conjunto, agrega los documentos en uno solo, por el nodo que se le pasa en la expresión XPATH, y colocándolo en el slot de salida.

Context Enricher: (Realizado por Cristian y Juan)

El Context Enricher combina información de un contexto y un cuerpo en documentos XML. Utiliza dos XPATH en forma de filtro para primero buscar el nodo que queremos añadir del contexto al cuerpo y el nodo del cual queremos que esa información sea hijo, creando un nuevo documento combinado que se pasará al slot de salida.

Correlator: (Realizado por Ismael y Juan)

El Correlator realiza la correlación de datos entre dos conjuntos de documentos XML, uno para el cuerpo (Body) y otro para el contexto. Utiliza una expresión XPATH (Filtro) para encontrar coincidencias entre los elementos de ambos conjuntos, una vez encontrado lo coloca en sus slots correspondientes.

Distributor: (Realizado por Juan)

El Distributor recoge por parámetros la expresión XPATH por la cual sabemos que valor de nodo hay que comparar y el número de salidas a cual deberá distribuir, para crear tantos slots como sean necesarios. Con un mapa vamos metiendo todos los mensajes que compartan el mismo valor en el nodo, en un mismo slot.

Merger: (Realizado por Ismael)

El Merger recoge en un mapa todos los slots de entrada, y se encarga de guardar todos los documentos recogidos de estos en el slot de salida.

Replicator: (Realizado por Cristian)

El Replicator recoge por parámetros el número de salidas que tiene, tras ello copia el mensaje de entrada en todos los slots de salida que se hayan creado debido a ese parámetro de entrada.

Splitter: (Realizado por Cristian e Ismael)

El Splitter recoge por parámetros la expresión XPATH por la cual vamos a recortar el mensaje, tras eso realiza el corte, indicando en cuántos cortes se ha partido el mensaje, y colocándolo con la etiqueta size dentro del mensaje, también debe mantener el id, para que al final podamos montarlo.

Filtro: (Realizado por Cristian)

El Filtro recoge por el slot de entrada el documento XML y a través de una expresión XPATH pasada por parámetro comprueba si el nodo del documento cumple con dicha expresión, si es así, se guardará en el slot de salida.

Translator: (Realizado por Cristian, Ismael y Juan)

El Translator está pensado para acceder a una BD, por esto, usa la expresión que se le pasa por parámetro como consulta SQL, además como debe buscar una fila en especifico, por lo tanto se le debe pasar también el nodo del cual hay que recoger el id, nombre o por lo que quieras buscar en la fila.

TranslatorV2: (Realizado por Cristian e Ismael)

Al igual que el Translator, la diferencia es que este sirve para recortar el mensaje y además eliminar un nodo que no sea necesario, por parámetros se le envía la expresión XPATH por la que recortar el mensaje, y el nombre del nodo el cual queremos eliminar.

CAFE:

Conectores:

Comandas: (Realizado por Ismael y Cristian)

Este conector lee los mensajes de una carpeta, que está dentro del mismo proyecto llamada comandas, los transforma en un documento y los pasa al puerto.

Barman: (Realizado por Cristian)

Este conector lee los mensajes del puerto (slot de entrada) realiza la búsqueda en la BD y envía al puerto el resultado (slot de salida).

Camarero: (Realizado por Ismael)

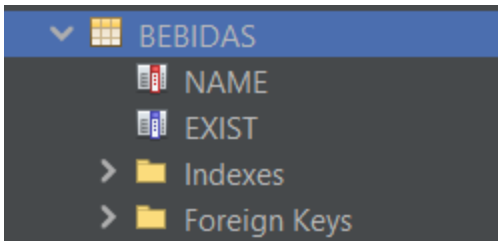
Este conector lee los mensajes del puerto y los crea en una carpeta que está dentro del proyecto llamada comandas_output.

Controlador:

Controlador Café: (Realizado por Ismael y Cristian)

En el controlador podemos observar tres partes principales, la primera donde se indican todos los datos necesarios para iniciar la integración, la segunda, donde el constructor enlaza los slots, y la tercera, donde se realiza el problema.

BD:




```
CREATE TABLE BEBIDAS(NAME VARCHAR(80),
EXIST INTEGER(10) NOT NULL, CONSTRAINT
cp_nameBEBIDAS PRIMARY KEY (NAME));
```

Fig.6

PROBLEMA 1:

Conectores:

GMS: (Realizado por Juan)

Este conector lee los mensajes de una carpeta, que está dentro del mismo proyecto llamada notas, los transforma en un documento y los pasa al puerto.

Alumnos: (Realizado por Juan)

Este conector lee los mensajes del puerto (slot de entrada) realiza la búsqueda en la BD y envía al puerto el resultado (slot de salida).

Mensaje: (Realizado por Juan)

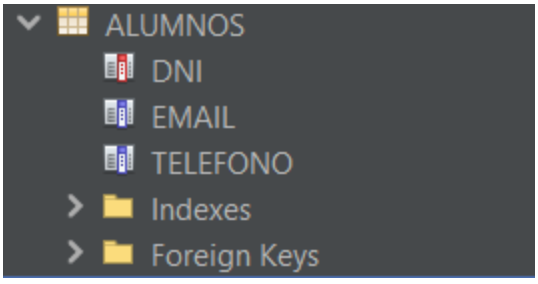
Este conector lee los mensajes del puerto y los crea en una carpeta que está dentro del proyecto llamada notas_output.

Controlador:

Controlador Problema 1: (Realizado por Cristian, Ismael, Juan)

En el controlador podemos observar tres partes principales, la primera donde se indican todos los datos necesarios para iniciar la integración, la segunda, donde el constructor enlaza los slots, y la tercera, donde se realiza el problema.

BD:




```
CREATE TABLE ALUMNOS(DNI VARCHAR(20),
EMAIL VARCHAR(100) NOT NULL, TELEFONO
VARCHAR(20), CONSTRAINT cp_dniALUMNOS
PRIMARY KEY (DNI));
```

Fig.7

TELEGRAM:

Conectores:

Telegram: (Realizado por: Cristian)

Este conector lee los mensajes del puerto y los envía por el bot a la persona que hable, si no, se espera.

Controlador:

Controlador Telegram: (Realizado por: Cristian, Ismael y Juan)

En el controlador podemos observar tres partes principales, la primera donde se indican todos los datos necesarios para iniciar la integración, la segunda, donde el constructor enlaza los slots, y la tercera, donde se realiza el problema.

BOT: (Realizado por Ismael)

Para este tercer problema hemos usado la misma base de datos que en el problema CAFÉ, pues en esencia es el mismo problema, lo único que lo distingue es que para enviar la información lo hacemos a un bot, el cual su enlace es el siguiente:

https://t.me/IIA_GPT_bot

(Es necesario tener instalado Telegram en el ordenador, en caso de que no funcione pruebe con el teléfono móvil)

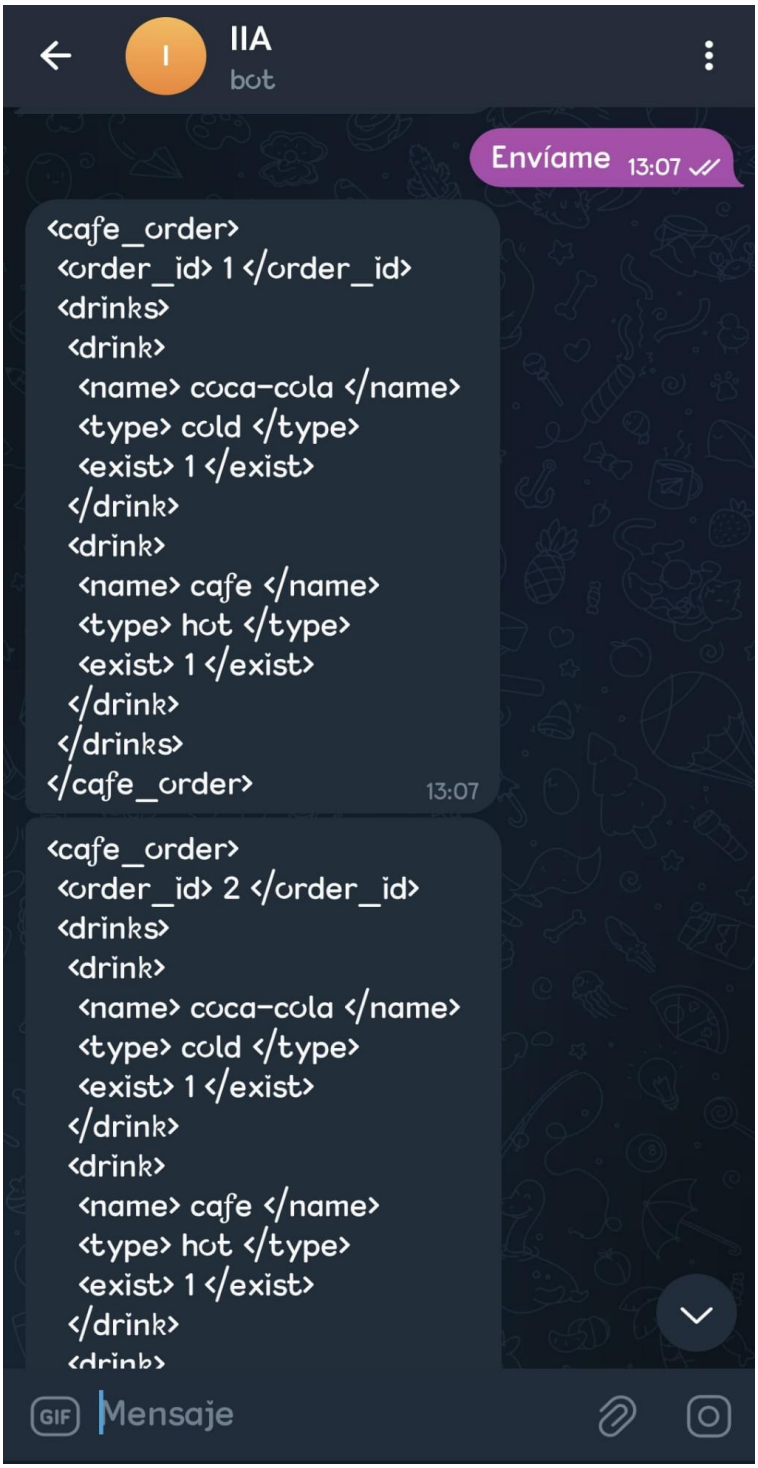

IIA
bot
Envíame 13:07
<cafe_order>
<order_id> 1 </order_id>
<drinks>
<drink>
<name> coca-cola </name>
<type> cold </type>
<exist> 1 </exist>
</drink>
<drink>
<name> cafe </name>
<type> hot </type>
<exist> 1 </exist>
</drink>
</drinks>
</cafe_order>
13:07
<cafe_order>
<order_id> 2 </order_id>
<drinks>
<drink>
<name> coca-cola </name>
<type> cold </type>
<exist> 1 </exist>
</drink>
<drink>
<name> cafe </name>
<type> hot </type>
<exist> 1 </exist>
</drink>
GIF
Mensaje

Fig.8

DISTRIBUCIÓN DE LA PRÁCTICA

Cristian Delgado Cruz

Tiempo dedicado aproximado: 30 horas

Juan Jiménez Serrano

Tiempo dedicado aproximado: 25 horas

Ismael Da Palma Fernández

Tiempo dedicado aproximado: 25 horas

Nota: Al usar la aplicación **IntelliJ IDEA**, hemos podido trabajar de forma concurrente pudiendo realizar el trabajo a la vez, aún así hemos dividido de forma equitativa las tareas, por otra parte, Cristian ha realizado el seguimiento de los bugs que iban apareciendo en el código y también de exponer de forma simplificada y sencilla el código final.

FIGURAS