Universidad Tecnológica de Pereira

Misión TIC 2022

El futuro digital es de todos

MinTIC

Diferenciando Capas

Persistencia Basada en Archivos

Hechos que conectan

Introducción

En esta sección de la Unidad 5 enlazaremos varios conceptos en un caso de estudio pequeño:

- Separación de interacción y lógica (Presentada en la Unidad 2)
- Aplicación CRUD (Presentada desde la Unidad 3, modificando colecciones)
- Manejo de archivos para persistencia de datos (Presentado en esta Unidad)

Caso de Estudio

- Recordemos la Aplicación CRUD de la Unidad 3 para manejar un listado de tareas, manipulando un diccionario de diccionarios: https://github.com/luismescobarf/clasesCiclo1/blob/master/AppCRUD_ParadigmaProceduralEstructurado.py

- Ahora vamos a darle persistencia al diccionario que contiene las tareas, es decir, cuando el programa se cierre, estarán todos los cambios realizados por la aplicación disponibles para una nueva ejecución.

Capa de Datos

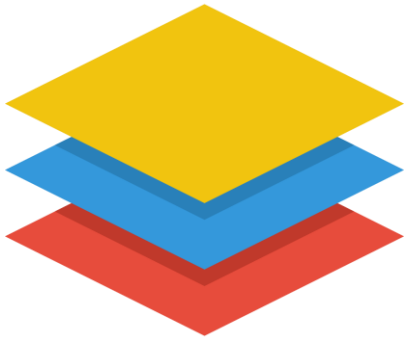
- Hemos hablado en unidades previas acerca de la diferenciación de la **interacción y la lógica** cuando escribimos nuestras soluciones, y las **funciones** son nuestras aliadas para esto.

- Iremos un poco más allá, separaremos la carga de los datos en una **capa separada**, que en este caso será un archivo para guardar de manera permanente el registro que lleva nuestra aplicación para la gestión de tareas: **capa de datos**.

Programando por Capas

Introduciremos el concepto de capas, aunque lo hemos utilizado de forma implícita, vamos a realizar la siguiente diferenciación:

1. **Capa de presentación:** las interfaces que se presentan al usuario.
2. **Capa de negocio:** donde se encuentra la lógica de los programas, recibe de la capa de presentación la información y se comunica con la capa de datos para reflejar de manera permanente estas acciones.
3. **Capa de datos:** donde se guardan los datos (archivos o bases de datos), dando lugar a modelos que representan el dominio del problema.



Introducción MVC

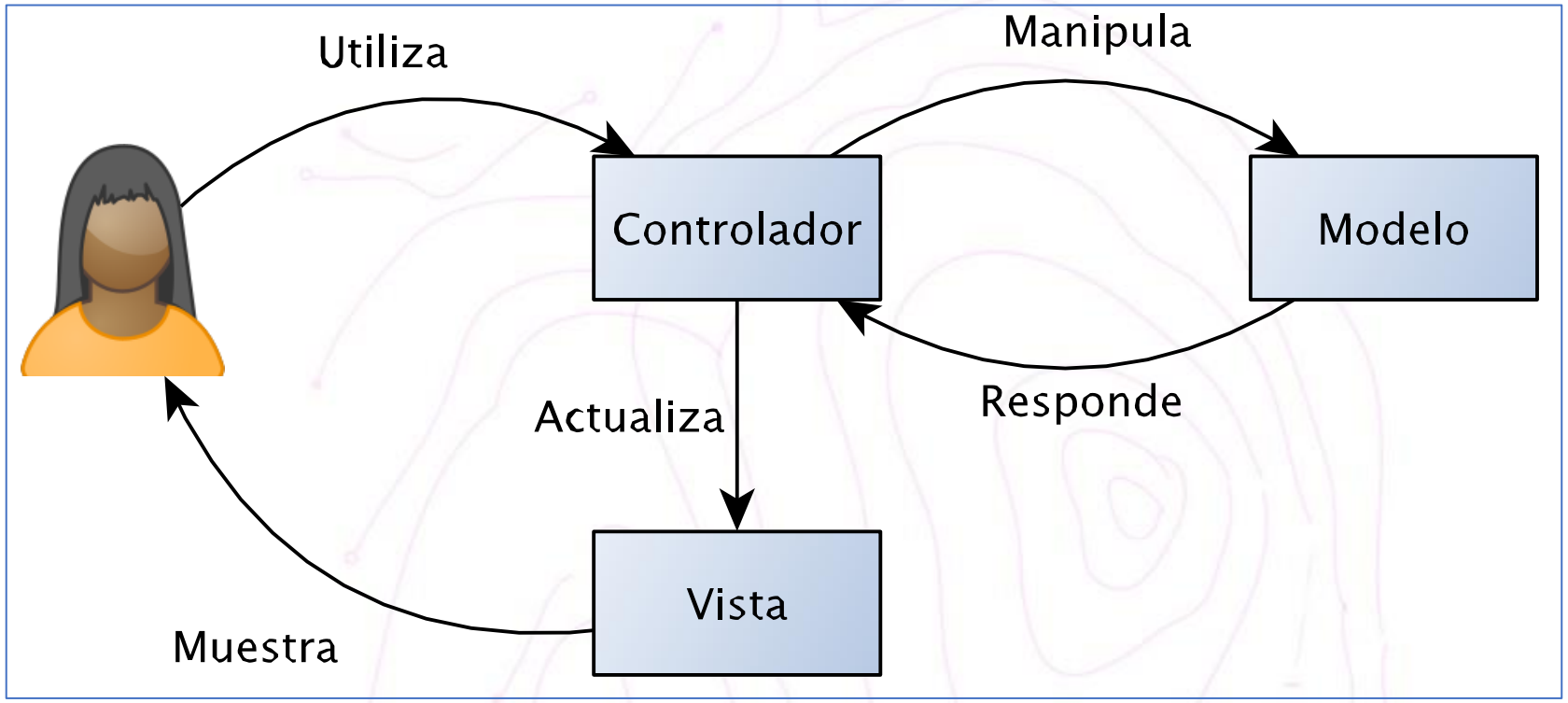
- La diferenciación de capas sentará las bases para el Modelo Vista Controlador (MVC), el cual se trabajará en el Ciclo 2.

- El caso de estudio que se presenta a continuación, nos permitirá dar los primeros pasos para escribir nuestros programas de una forma conveniente para realizar mantenimiento y agregar nuevas funcionalidades.

- En la siguiente ubicación se encuentra la Aplicación CRUD que analizaremos: https://github.com/luismescobarf/clasesCiclo1/tree/master/AppCRUD_CapaDatos

Introducción MVC

- Observemos una separación de las capas más específica aún:



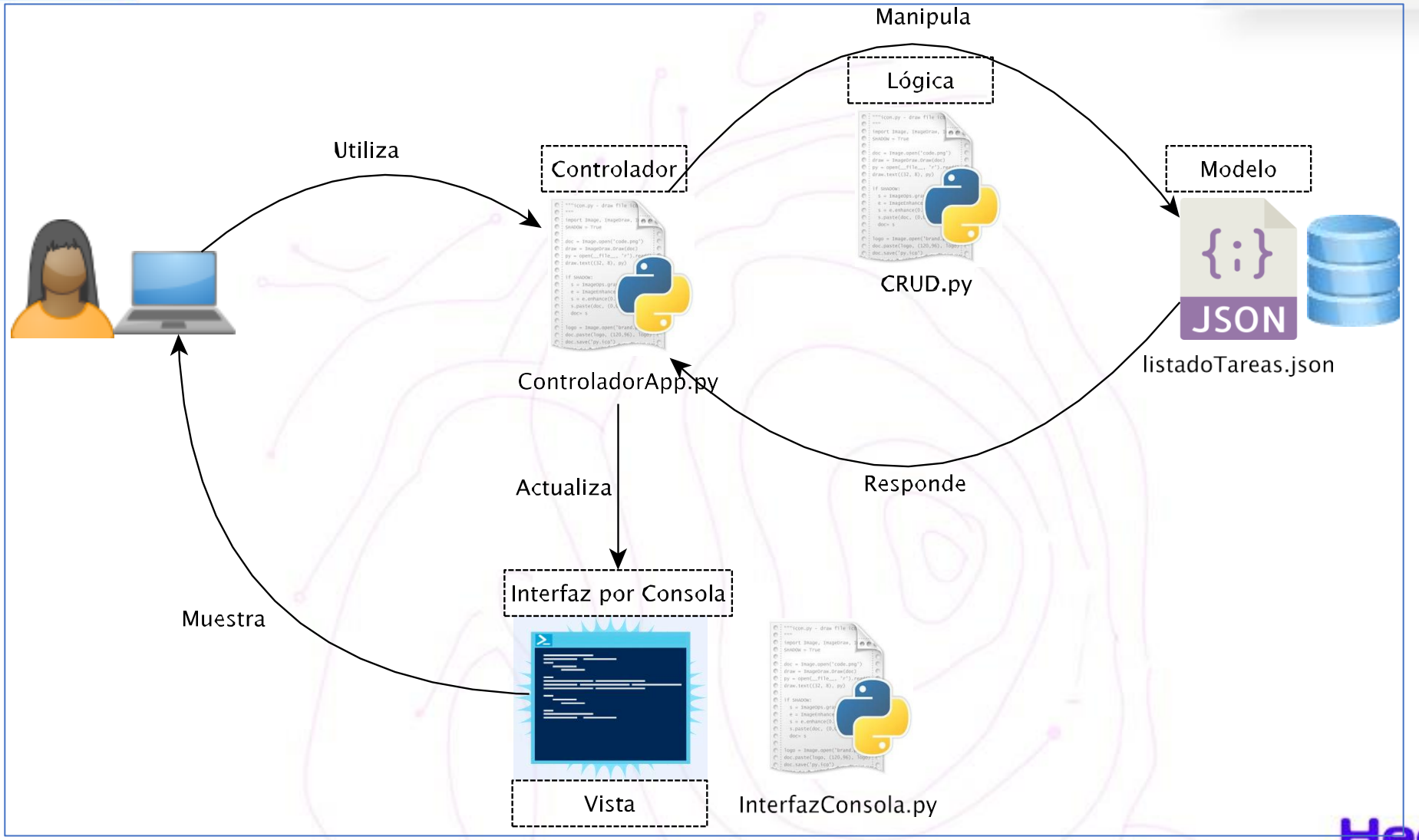
Introducción MVC

- Cuando el usuario utilice la aplicación, se estará entendiendo con el **Controlador**, mediante la **Vista** ofrecida. En este caso de estudio es a través del terminal o línea de comandos.

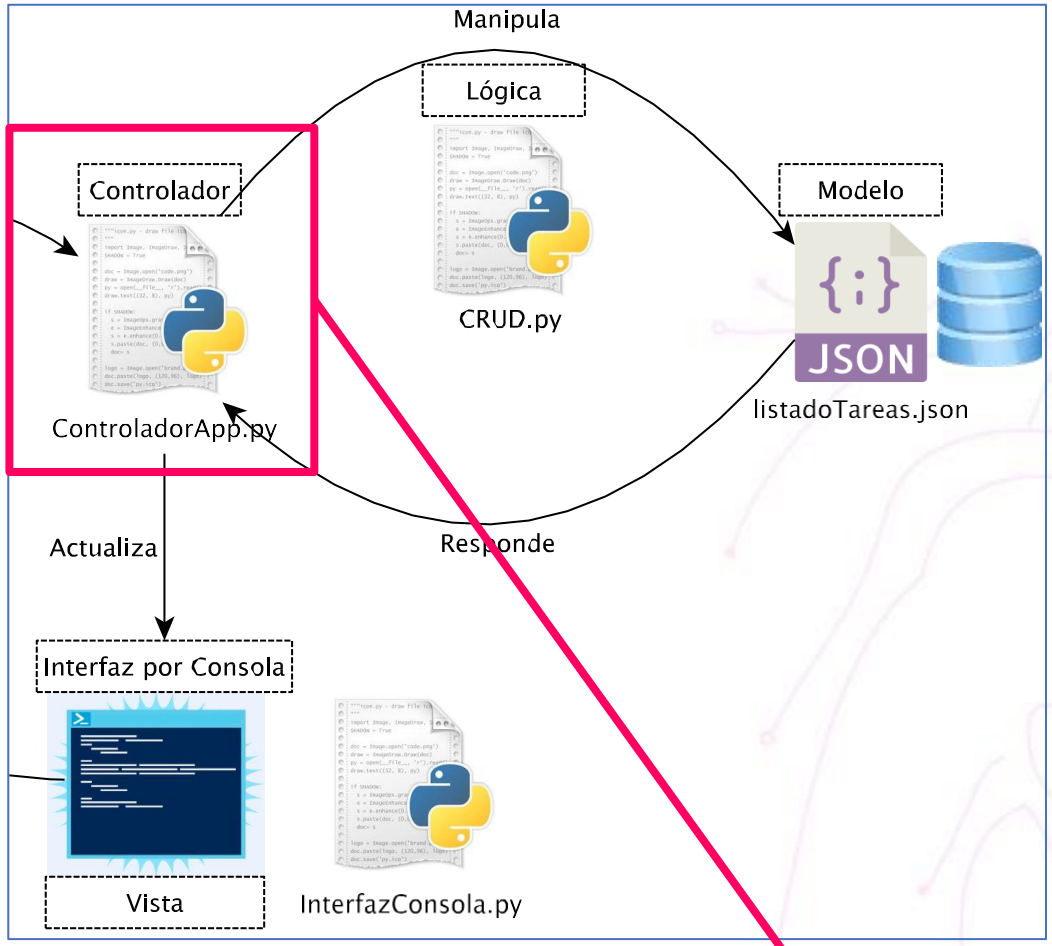
- Los datos serán almacenados en un archivo, el cual aloja el **Modelo**, que corresponde a la forma como hemos diseñado la colección de tareas: diccionario de diccionarios.

- Al tener una capa de datos, debemos diferenciar qué funciones los manipulan, y por tanto, separar la lógica de la interacción.

MVC Caso de Estudio



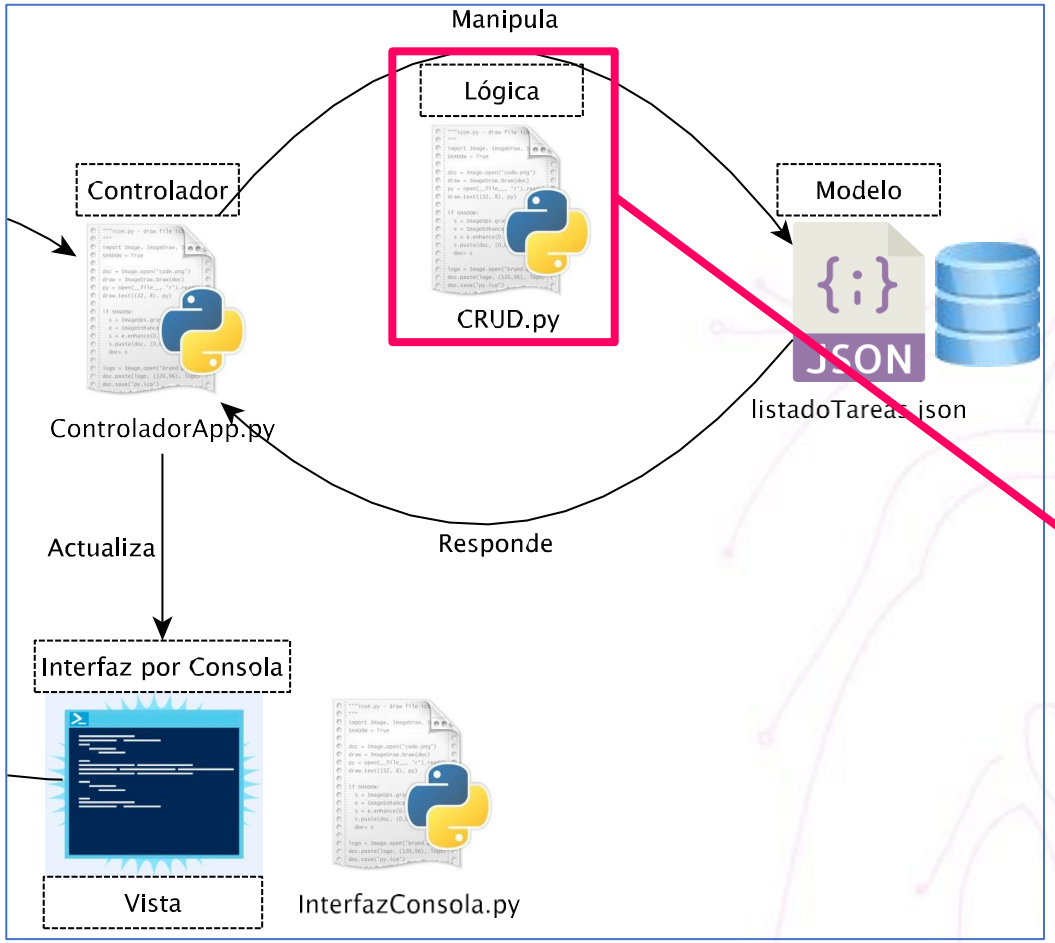
Controlador (App)



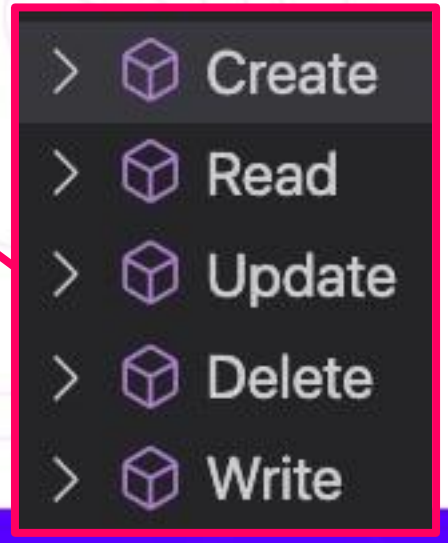
- El **controlador** manipulará el modelo o contenedor a través de las funciones que contienen la lógica de la aplicación, ubicada en CRUD.py
- De forma análoga, interactuará con el usuario, haciendo uso de la interfaz implementada en InterfazConsola.py
- El controlador integrará las capas de presentación y datos.

```
import CRUD #Capa lógica o backend básico
import InterfazConsola as ic #Interfaz para interacción
```

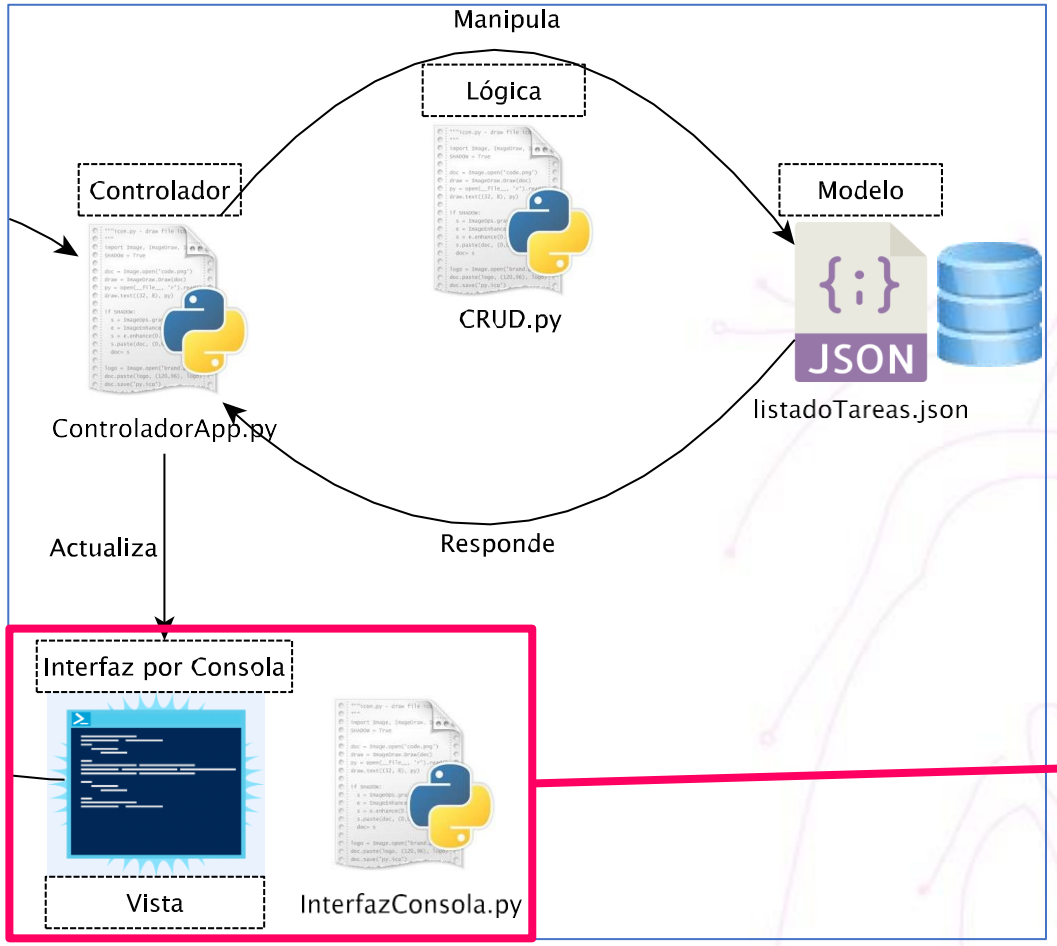
Lógica



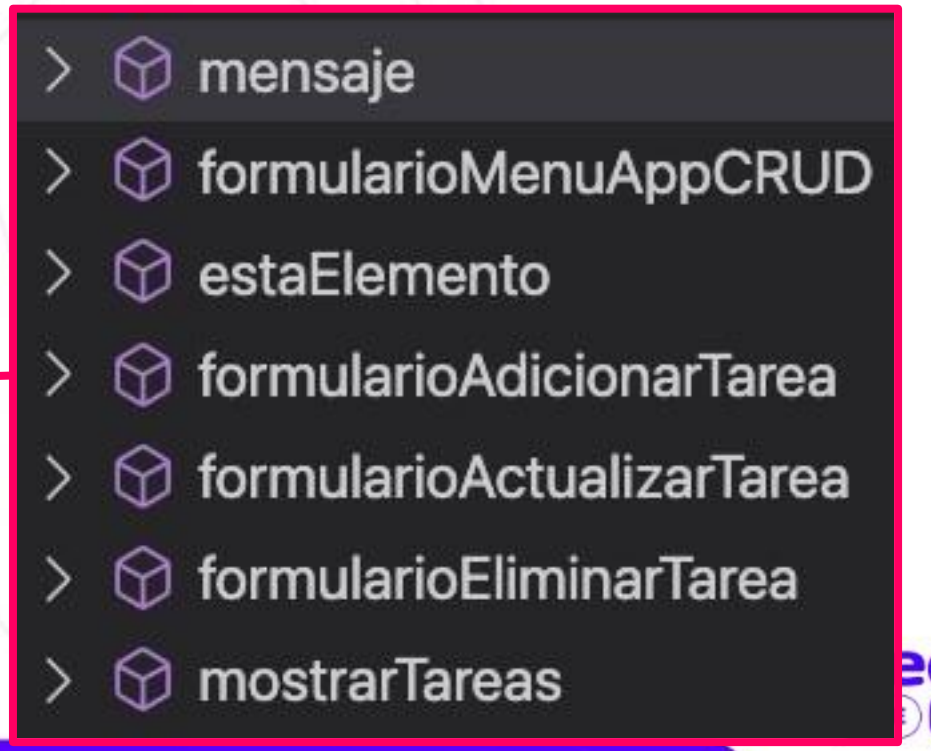
- La **capa de negocio o lógica,** implementa las operaciones que manipulan el modelo de datos: diccionario de diccionarios en memoria y almacenado como archivo JSON de forma persistente.
- Los diccionarios tienen correspondencia directa con JSON.



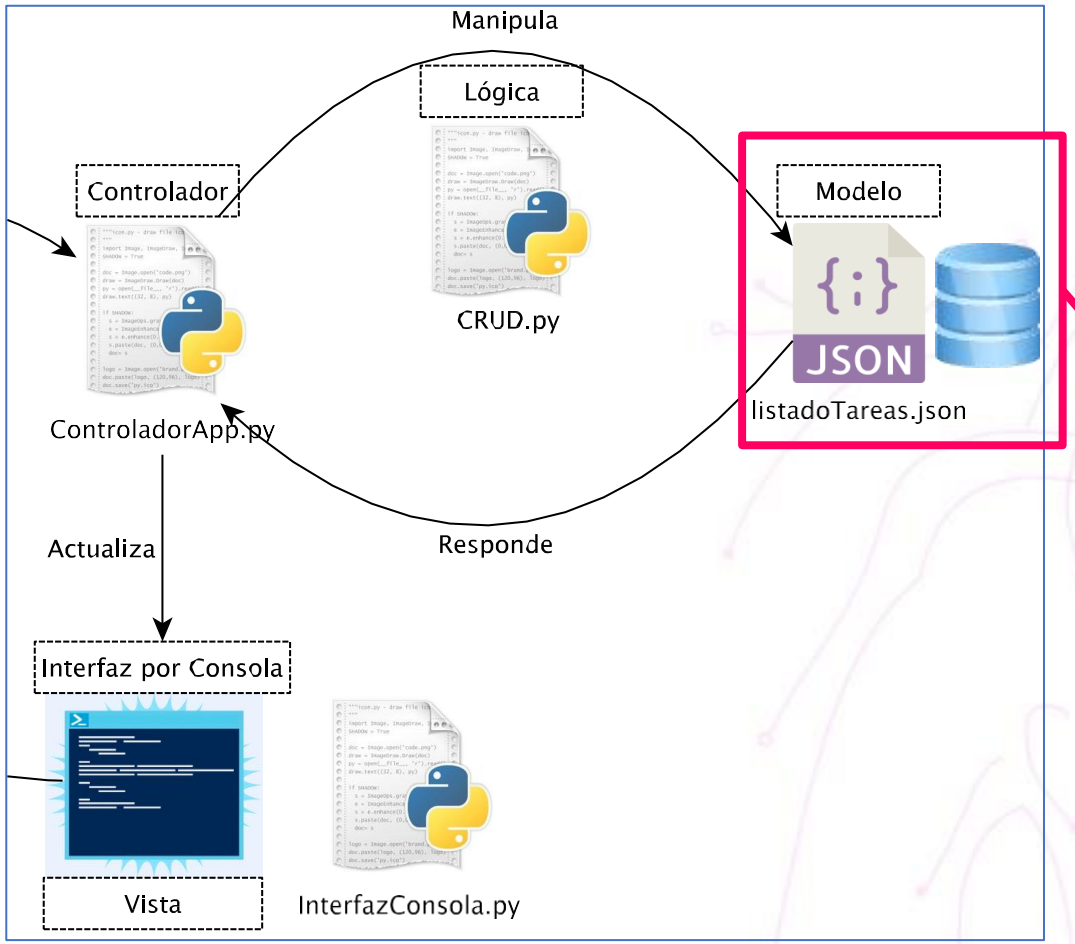
Vista/Interfaz



- Ofrecerá la información y posibilidad de interacción al usuario, según lo que establezca el controlador al recibir respuesta de la capa lógica y el modelo.



Persistencia en Archivo



JSON Datos sin procesar Cabeceras

Guardar Copiar Contraer todo Expandir todo Filtrar JSON

```
▼ 01:
    descripcion:    "Ir a mercar"
    estado:         "pendiente"
    tiempo:         60
▼ 02:
    descripcion:    "Estudiar"
    estado:         "pendiente"
    tiempo:         180
▼ 03:
    descripcion:    "Hacer ejercicio"
    estado:         "pendiente"
    tiempo:         50
```

Trabajo Autónomo

- Revisar detenidamente la implementación del caso de estudio, interactuar con este, observar la evolución del archivo JSON (base de datos).
- Agregar nuevas funcionalidades a la Aplicación CRUD para Listado de Tareas.
- Plantear aplicaciones CRUD como esta para otros requerimientos, con niveles de anidación mayores en el diccionario:
  - Pedidos de mesas en un restaurante, especificando cada puesto de la mesa.
  - Gestión estudiantes en grupos de materias.
  - CRUD para gestión de árboles genealógicos.