Python No Muerde

Capítulo: Las Capas de una Aplicación

> "Que tu mano izquierda no sepa lo que hace tu mano derecha"
>
> Anónimo

En el capítulo anterior cuando estaba mostrando el uso del ORM puse

> Si tenemos cuidado y aislamos el ORM del resto de la aplicación, es posible reemplazarlo con otro más adelante (o eliminarlo y "bajar" a SQL o a NoSQL).

¿Qué significa, en ese contexto, "tener cuidado"? Bueno, estoy hablando básicamente de lo que en inglés se llama multi-tier architecture.

Sin entrar en detalles formales, la idea general es decidir un esquema de separación en capas dentro de tu aplicación.

Siguiendo con el ejemplo del ORM: si todo el acceso al ORM está concentrado en una sola clase, entonces para migrar el sistema a NoSQL alcanza con reimplementar esa clase y mantener la misma semántica.

Algunos de los "puntos" clásicos en los que partir la aplicación son: Interfaz/Lógica/Datos y Frontend/Backend.

Por supuesto que esto es un formalismo: Por ejemplo, para una aplicación puede ser que todo twitter.com sea el backend, pero para los que lo crean, twitter.com a su vez está dividido en capas.

Yo no creo en definiciones estrictas, y no me voy a poner a decidir si un método específico pertenece a una capa u otra, normalmente uno puede ser flexible siempre que siga al pie de la letra tres reglas:

Una vez definida que tu arquitectura es en capas "A"/"B"/"C"/"D" (exagerando, normalmente dos o tres capas son suficiente):

- Las capas son una lista ordenada, se usa hacia abajo.

  Si estás en la capa "B" usás "C", no "A".

- Nunca dejes que un componente se saltee una capa.

  Si estás en la capa "A" entonces podés usar las cosas de la capa "B". "B" usa "C". "C" usa "D". Y así. Nunca "A" usa "C". Eso es joda.

- Tenés que saber en qué capa estás en todo momento.
  Apenas dudes "¿estoy en B o en C?" la respuesta correcta es "estás en el horno."

¿Cómo sabemos en qué capa estamos? Con las siguientes reglas:

1. Si usamos el ORM estamos en la capa datos.
2. Si el método en el que estamos es accesible por el usuario, estamos en la capa de interfaz.
3. Si `not 1 and not 2` estamos en la capa de lógica.

No es exactamente un ejemplo de formalismo, pero este libro tampoco lo es.

Proyecto

Vamos a hacer un programa dividido en capas capas, interfaz/lógica/datos. Vamos a implementar dos veces cada capa, para demostrar que una separación clara independiza las implementaciones y mejora la claridad conceptual del código.

El Problema

Pensemos en juegos de tablero multijugador. ¿Cómo definirías el juego de damas de forma muy genérica?

- Hay un tablero de NxN
- Hay X cantidad de fichas de cada color.
- Las fichas comienzan el juego en cierta posición.
- Hay dos jugadores
- Cada jugador tiene un turno en el que puede mover ciertas fichas según reglas específicas.
- Luego de una movida, tal vez alguna ficha se saque del tablero.
- Luego de una movida, tal vez alguna ficha sea reemplazada por otra.

Ok, ahora pensemos en ajedrez. O en go. O en ta-te-ti…

¡Resulta que todos esos juegos se pueden describir de exactamente la misma manera!

Entonces dividamos esta teórica aplicación en capas:

- Interfaz: muestra el tablero y las fichas. Acepta las movidas.
- Lógica: procesa las movidas que manda la interfaz, las valida y acepta o rechaza.
- Datos: Luego de que una movida es validada, guarda un historial de las mismas, y el estado del tablero y las fichas.

Vamos a implementar esta aplicación de una manera… peculiar. Cada capa va a ser implementada dos veces, de maneras lo más distintas posible.

La manera más práctica de implementar estas cosas es de atrás para adelante:

FIXME hacer diagrama

Datos -> Lógica -> Interfaz

Capa de Datos: Diseño

Necesitamos describir completamente y de forma genérica todos estos juegos.

Qué tenemos en común:

Fichas

Son objetos que tienen un tipo (“alfil negro”, “cruz”, “piedrita blanca”), una posición en el tablero (o no), y un dueño.

Jugadores

Los dueños de las fichas. Juegan por turno. Aplican acciones a las fichas. Tienen un nombre.

Tablero

Donde se ponen las fichas. Tiene una cantidad de posiciones donde una ficha puede ponerse. Esas posiciones se pueden representar en una superficie.

Creo que con esos elementos puedo representar cualquier juego de tablero común. [1]

1 La ventaja que tengo al ser el autor del libro es que si no es así vengo, edito la lista, y parece que tengo todo clarísimo desde el principio. No es el caso.

El Tablero

Las Fichas

El Jugador