75.07 - Algoritmos y Programación III

Segundo Cuatrimestre 2021

Trabajo Práctico 2

Algo-thief

Grupo 14

Alumnos	*Padrón*
Guido Arribas garribas@fi.uba.ar	98287
Lucas Gabriel Burna lburna@fi.uba.ar	99608
Mateo Colombo mcolombo@fi.uba.ar	98615
Ariel Lowy alowy@fi.uba.ar	98298
Lola Segura lsegura@fi.uba.ar	99254

2º Cuatrimestre de 2021

Índice

1 Supuestos

- Los cargos detective y novato tienen asociada la misma dificultad
- Cuando un detective es acuchillado mas de una vez, se le resta 2 hora por vez
- Cuando un detective recibe una bala mas de una vez, se le resta 4 horas por vez
- Los jugadores son entidades distintas y comienzan con 0 arrestos
- Una ciudad sólo puede visitar otras tres ciudades, una de las cuales es la ciudad de la que se provino, con la excepción de la ciudad en la que se arranca cada Nivel.
- El ladrón se encuentra siempre en la misma ciudad.
- El detective es acuchillado o recibe una bala únicamente en la ciudad en la que se encuentra el ladrón.
- Las ciudades visitables por cada ciudad siempre son las mismas

2 Diagramas de clases

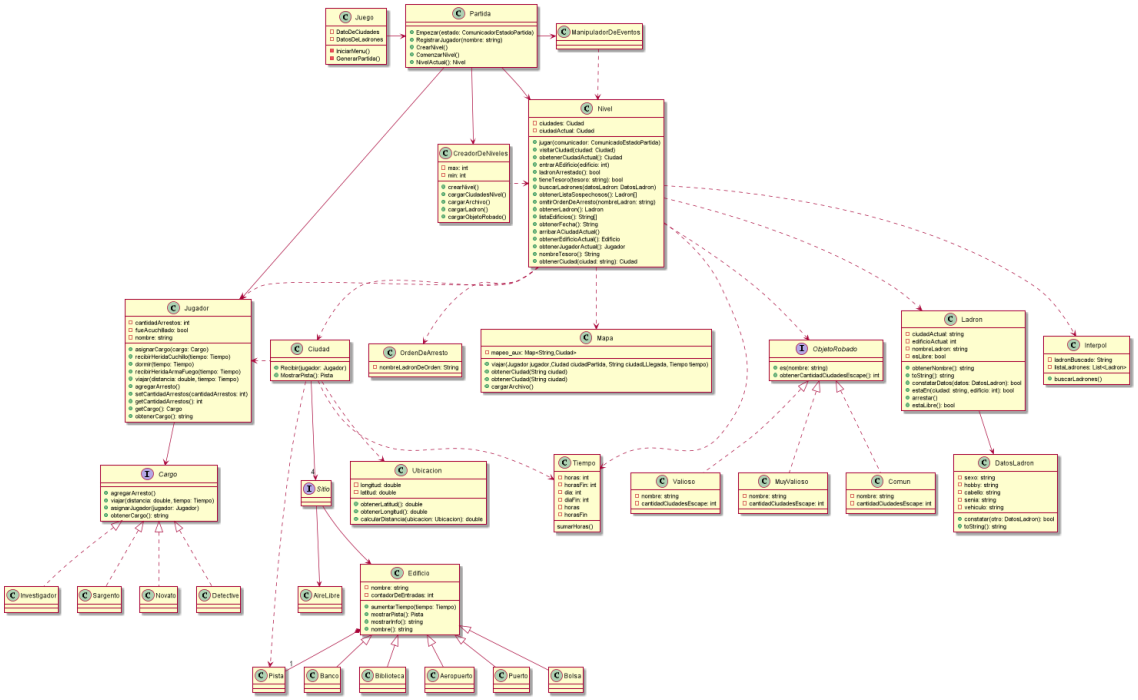

Figura 2.1: Esquema de clases para el modelo

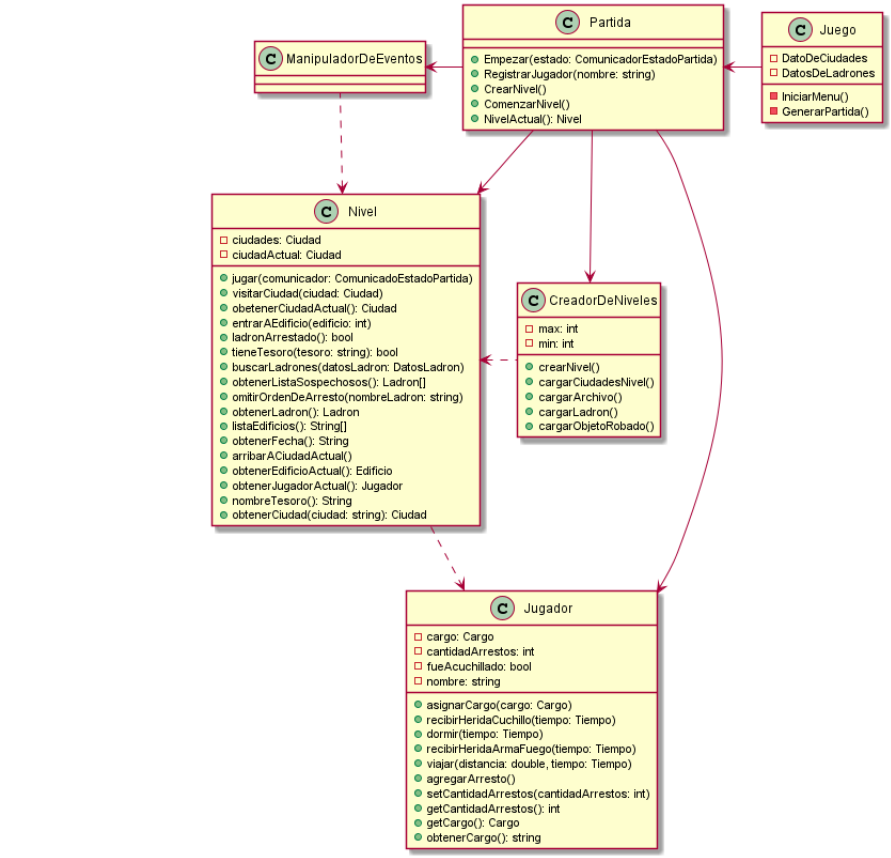

Figura 2.2: Diagrama de clase para el manejo del juego

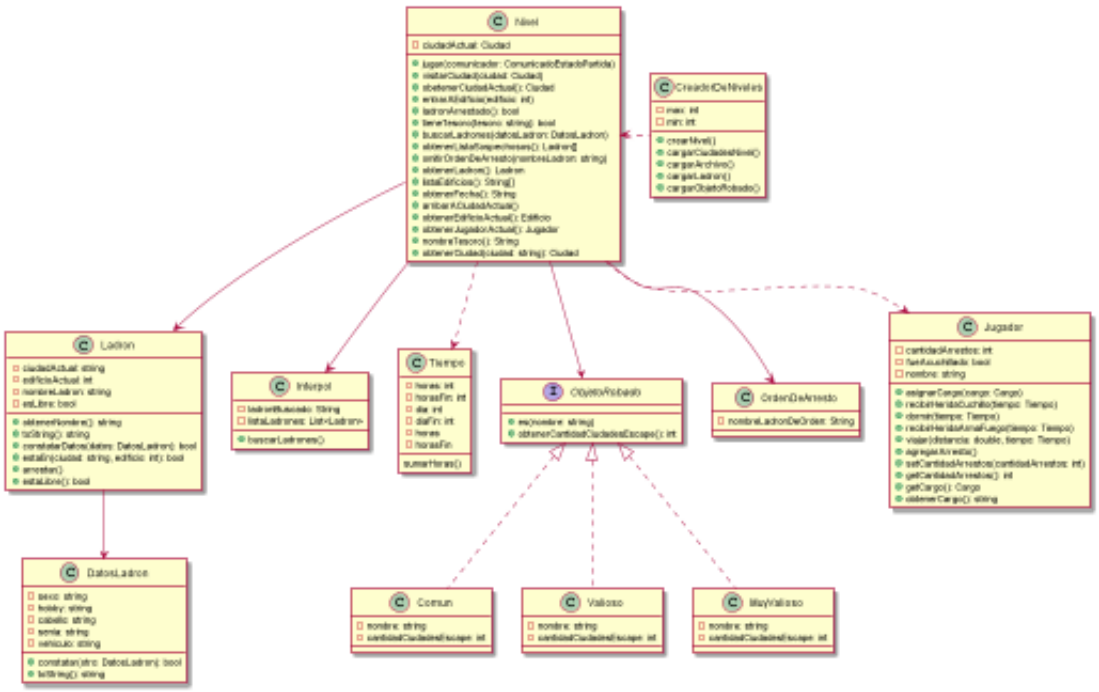

Figura 2.3: Primer diagrama de clase para el manejo del nivel

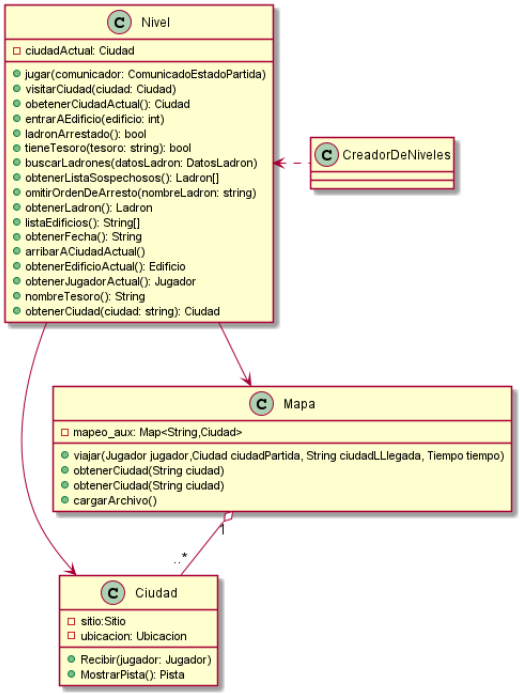

Figura 2.4: Segundo diagrama de clase para el manejo del nivel

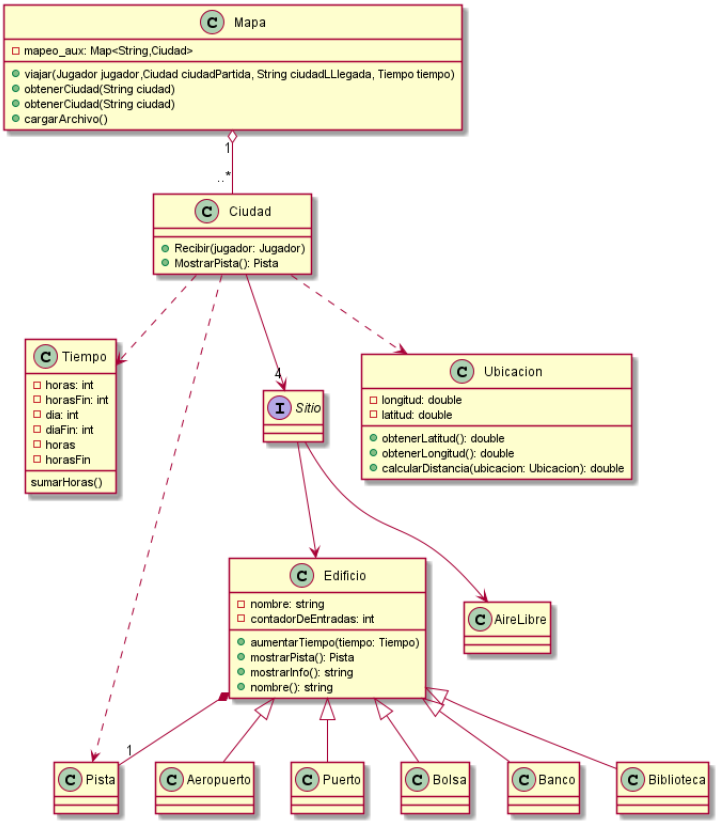

Figura 2.5: Diagrama de clase del mapa

3 Diagramas de secuencias

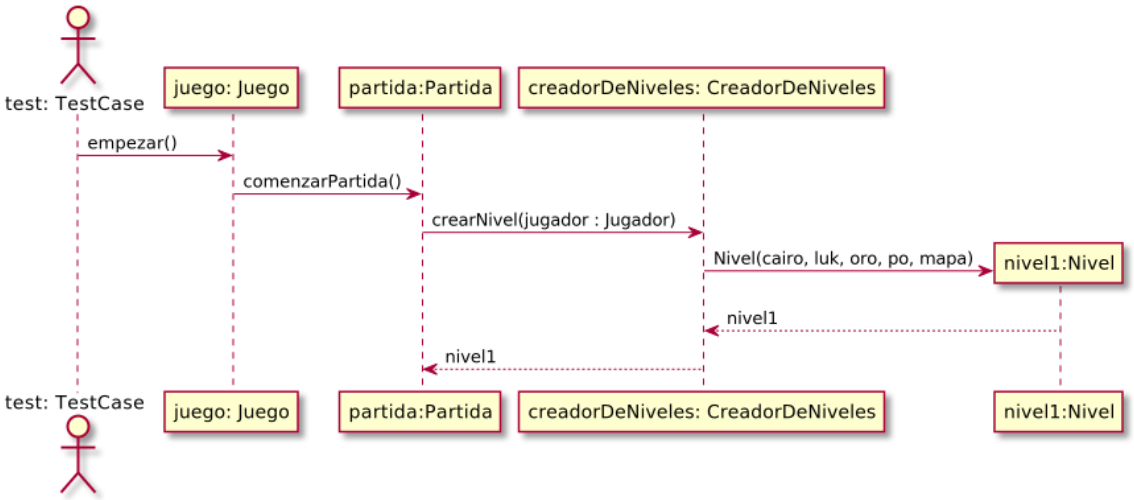

Figura 3.1: Esquema para construir una partida.

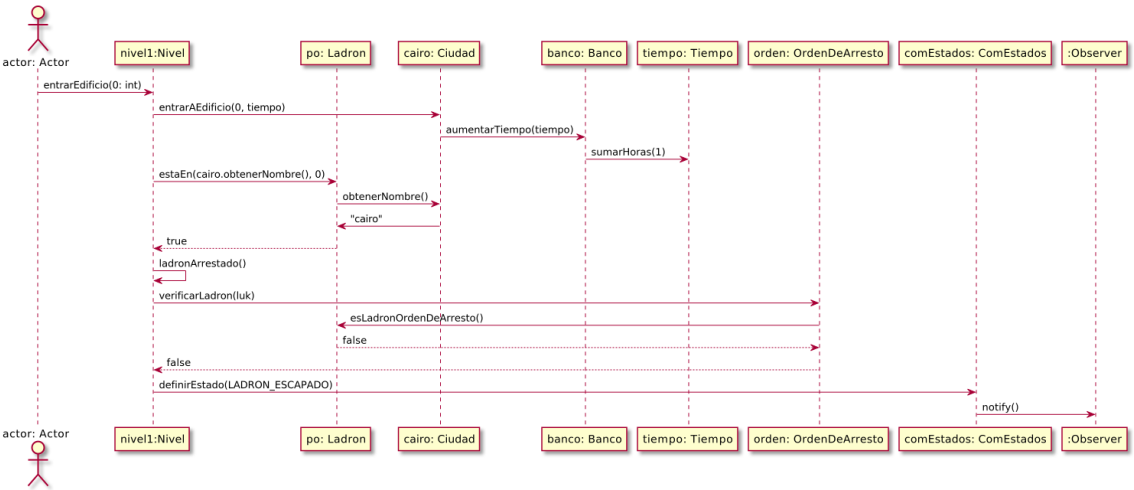

Figura 3.2: Esquema para jugador atrapando a un ladrón sin orden de arresto.

4 Detalles de implementación

La lógica de la aplicación parte de un ciclo que comienza cuando se abre el juego y se mantiene corriendo siempre y cuando no se lo cierre. En pimer lugar, la clase **Partida** se encarga de generar los niveles mediante el **Creador de Niveles**; **Partida** se encarga de cargar y guardar los niveles, junto con el estado del jugador en cuanto a sus arrestos luego de finalizar los niveles; y el **Nivel** maneja el entorno actual del jugador. Esto lo logra mediante la carga y escritura de un archivo JSON que contiene la información relevante para la creación de los niveles, basada en los jugadores y la cantidad de arrestos realizados por los mismos.

Se pensó en una estructura piramidal de manera que **Juego** resulta la cúspide de la pirámide, teniendo como capa inmediatamente inferior a **Partida**, **Nivel** y **CreadorDeNiveles**. El **ManipuladorDeEventos** permite manejar los estados del modelo, haciendo las veces de controlador del mismo.

Nivel es la clase más importante para el juego. El mismo se encarga de la mayor parte de la lógica. Es quien tiene acceso y métodos para el manejo del juego como tal, obteniendo los datos necesarios para el uso de una instancia de esta clase de archivos JSON. En sus atributos se pueden encontrar clases **Jugador, Ciudad, OrdenDeArresto, Ladron, Interpol, Mapa y ObjetoRobado** con las cuales **Nivel** va interactuando y modelando para llevar a cabo cada nivel, valga la redundancia. Cada **Nivel** se intancia a partir del **CreadorDeNiveles**. Entre los métodos destacables que lleva a cabo esta clase podemos encontrar, *entrarAEdificio* (encargado de la lógica del botón INVESTIGAR), *visitarCiudad* (logica botón VIAJAR), *emitirOrdenDeArresto*, *buscarLadrones* (lógica de botón VISITAR INTERPOL).

A su vez, el **Nivel** implementa un **Mapa**, al cual delega la responsabilidad de traducir las cadenas a ciudades y realizar los viajes al jugador. El mapa y el nivel utilizan ciudades, que tienen la ubicación: latitud y longitud, para permitir calcular los tiempos de viaje desde una ciudad a la otra; junto con los edificios que pueden visitarse en esta ciudad y sus respectivas pistas.

En cuanto a **Jugador**, contiene las interacciones básicas relativas a las acciones que puede realizar o recibir el jugador, como por ejemplo: recibirHeridaCuchillo() o viajar(). Dado que el juego tiene memoria, fue posible que esta clase contenga la cantidad de arrestos que realizó a lo largo del tiempo.

El **Ladron** está conformado por una clase de datosLadron, que sirve para separar las responsabilidades del ladrón de aquellas generadas por los atributos del mismo, como la comparación de datos tanto en conjunto como individualmente.

La **Interpol** recibe un archivo JSON para la creación de los distintos ladrones y a partir de los mismos (**Nivel** le delega la responsabilidad) se encarga de comparar los datos ingresados por pantalla para relacionarlos con los posibles ladrones.

Para la implementación del cargo del jugador se pensó en la lógica de estados *patron state*, en donde cada tipo de cargo se hace responsable de delegar las tareas a sus clases "hermanas", permitiendo así que **Jugador** no tenga en sus métodos `if`'s o `switch`es.

En cuanto a las parte gráfica del juego, se tiene un **Constructor de Escenas** el que se encarga de dibujar la diferentes pantallas. Esta clase se complementa con una clase **Oyente Partida** que implementa la interfaz **Observer** la cual se encarga de escuchar e interpretar los distintos estados en los que se puede encontrar el modelo.

Para comunicar el estado del modelo se implemento una clase **Comunicador Estado Partida** que hereda de **Observable**, por lo que cada vez que el modelo cambia, se cambia a su vez el estado y se notifica al **Observer**, permitiendo así conectar el modelo con la interfaz gráfica.

5 Excepciones

No se implementaron excepciones en el programa.