FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2 — Kahoot

[7507/9502] Algoritmos y Programación
III Curso 1
Primer cuatrimestre de 2020

Alumno	Padrón	Mail
Agustin Brasburg	104733	abrasburg@fi.uba.ar
Damian Ganopolsky	101168	dganopolsky@fi.uba.ar
Andres Jalife	104342	ajalife@fi.uba.ar
Maximiliano Levi	104288	mlevif@fi.uba.ar
Mathias Welz	101552	mwelz@fi.uba.ar

Índice

1. Introducción

El presente informe reúne la documentación de la solución del segundo trabajo práctico de la materia Algoritmos y Programación III que consiste desarrollar un juego de quiz que se asimila al Kahoot utilizando las técnicas de Java, TDD y Git para lograr un trabajo grupal.

2. Supuestos

A continuación vamos a enumerar algunos supuestos que tuvimos que tener en cuenta, ya que no estaban especificados por el enunciado.

Un supuesto importante que tuvimos a la hora de desarrollar el modelo fue el de mostrar y guardar la decisión de los jugadores acerca del uso o no de los distintos modificadores previo a que alguno de los jugadores vea la pregunta. Hicimos esto debido a que nos pareció lógico que los modificadores se usen con el fin de tener algún 'boost' para alcanzar al otro jugador, pero sin aprovecharse de conocer la pregunta, por lo que creemos que es un tanto más divertido e impredecible el desarrollo del juego de este modo.

Toda pregunta escrita en el "preguntas.json" que no sea válida no será añadida al kahoot.

Además supusimos que los nombres de los usuarios ingresados deben ser tanto diferentes de vacío como distintos entre sí.

3. Modelo de dominio

El Kahoot fue diseñado para que sea fácilmente extensible ya sea en nuevos modificadores, nuevos tipos de preguntas o nuevos modos de pregunta. Para poder lograr esto se trato de que la mayoría de las funcionalidades sean delegadas a objetos que puedan ser reemplazados fácilmente.

La unidad base de nuestro programa es el objeto Kahoot, este maneja y conecta las distintas partes del modelo, comenzando por las rondas. El Kahoot contiene una lista de objetos del tipo RondaBase donde cada ronda representa una ronda del juego osea una pregunta con sus jugadores y sus modificadores. Luego de que ambos jugadores contesten una pregunta Kahoot creará una ronda nueva con la siguiente pregunta y los modificadores a utilizar.

Las rondas son el objeto principal de nuestro modelo. Estas mantienen un estado de las respuestas dadas, los modificadores utilizados y el tiempo restante de la pregunta. También se encargan de calcular los puntajes dependiendo de los modificadores que se utilizaron. Existen 2 implementaciones de RondaBase, RondaNormal y RondaExclusividad que representan rondas donde se pueden usar multiplicadores o exclusividades respectivamente.

Por el lado de las preguntas estas mismas fueron modeladas con atributos base como texto y opciones posibles pero las funcionalidades más específicas como la forma de calcular el puntaje o
verificar las respuestas son delegadas a otros objetos del tipo ModoDePregunta y

TipoDePregunta. Esto nos permite mantener el mismo objeto pregunta pero con distintas funcionalidades y sin utilizar herencia, manteniendo así una jerarquía más simple.

Dentro de las rondas cada jugador es representado por un objeto del tipo Jugador que contiene su puntaje, su nombre y sus modificadores y exclusividades restantes. Los modificadores son objetos del tipo Multiplicador y Exclusividad los cuales mantienen un estado de sus usos y definen como modificar los puntajes.

El objeto Kahoot internamente carga las preguntas desde un archivo JSON que es indicado internamente. Esto nos permite editar las preguntas sin necesidad de recompilar el código. Para maximizar la versatilidad del modelo esto fue abstraido a través de una interfaz lo que nos permite tener distintos lectores con la misma firma. Más adelante mostraremos un ejemplo de otros lectores de preguntas implementadas.

4. Diagramas de clase

Kahoot es la clase encargada de empezar el juego y guardar el panel que muestra el juego, las preguntas que se mostrarán con sus respectivas respuestas y puntos por responder bien, y sus jugadores con sus respectivos nombres, puntaje y modificadores de puntajes.

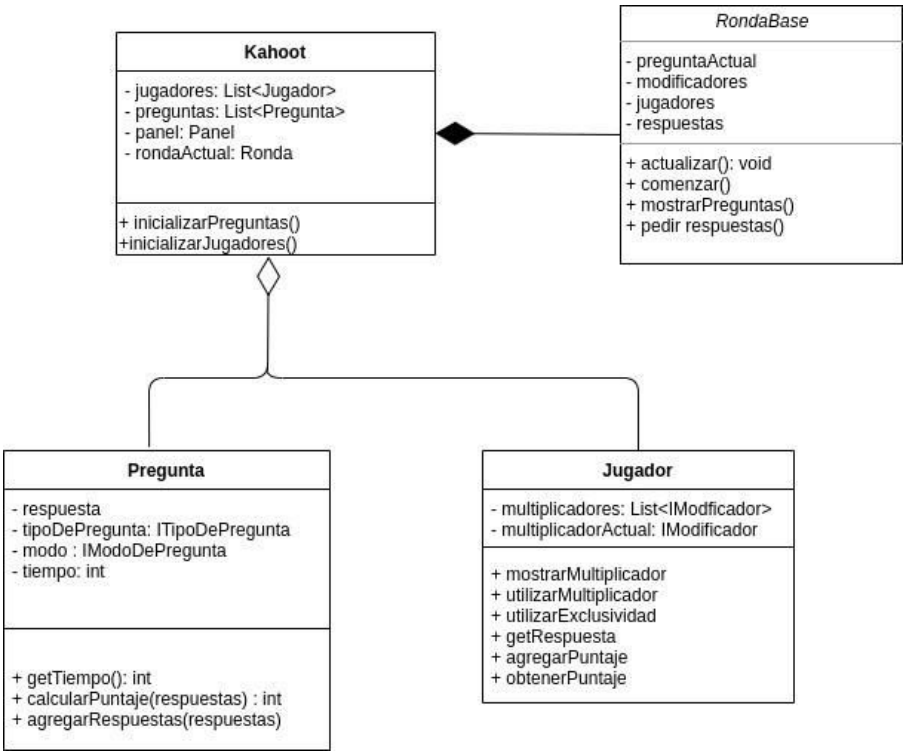

Figura 1: Diagrama del Kahoot.

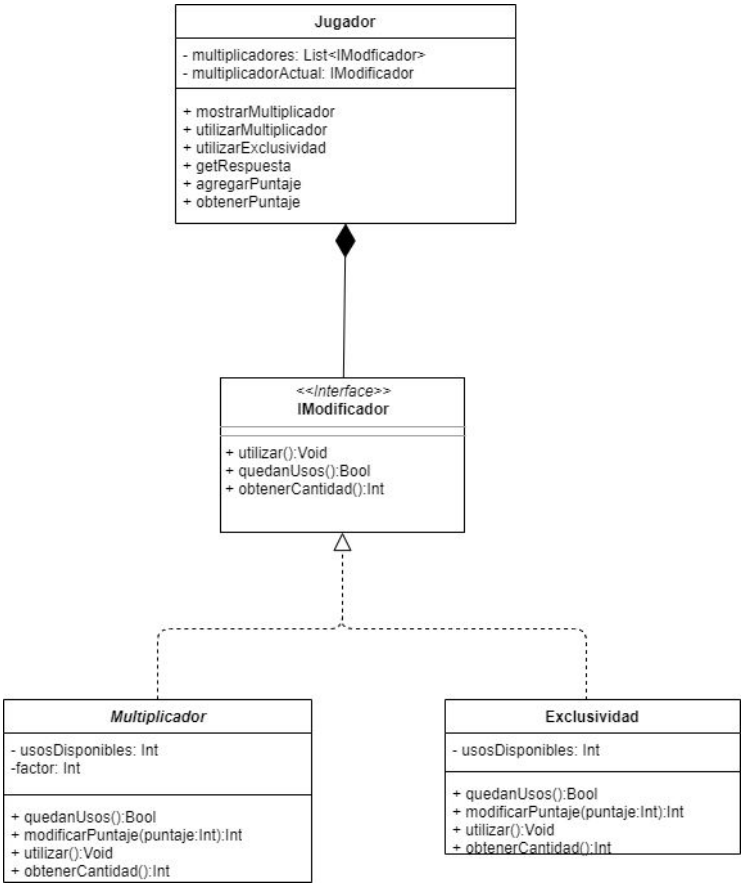

Figura 2: Diagrama del Jugador.

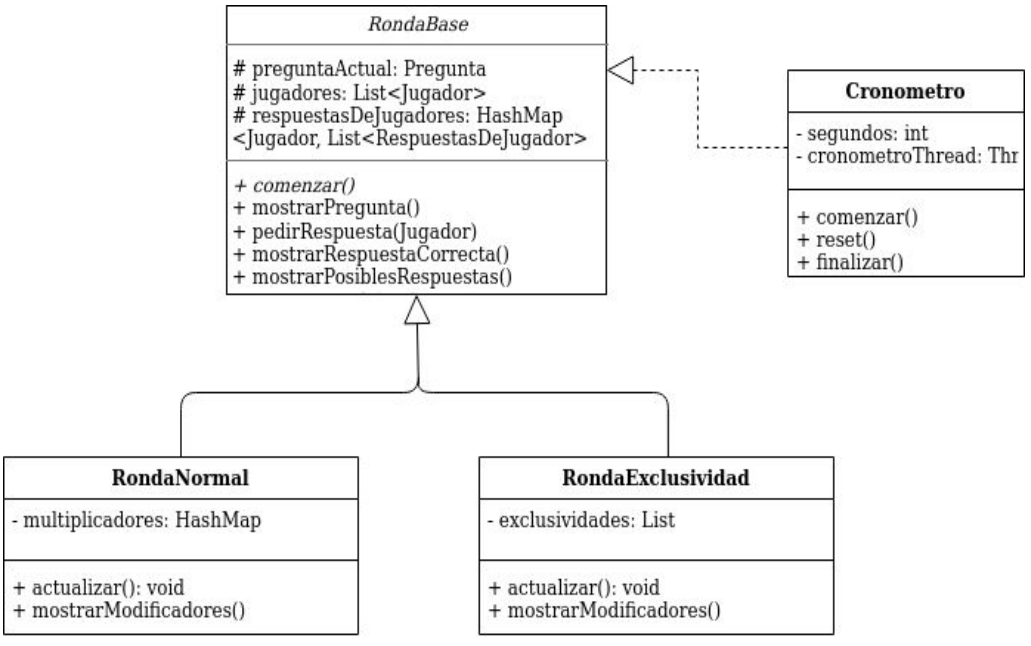

Figura 3: Diagrama de Ronda.

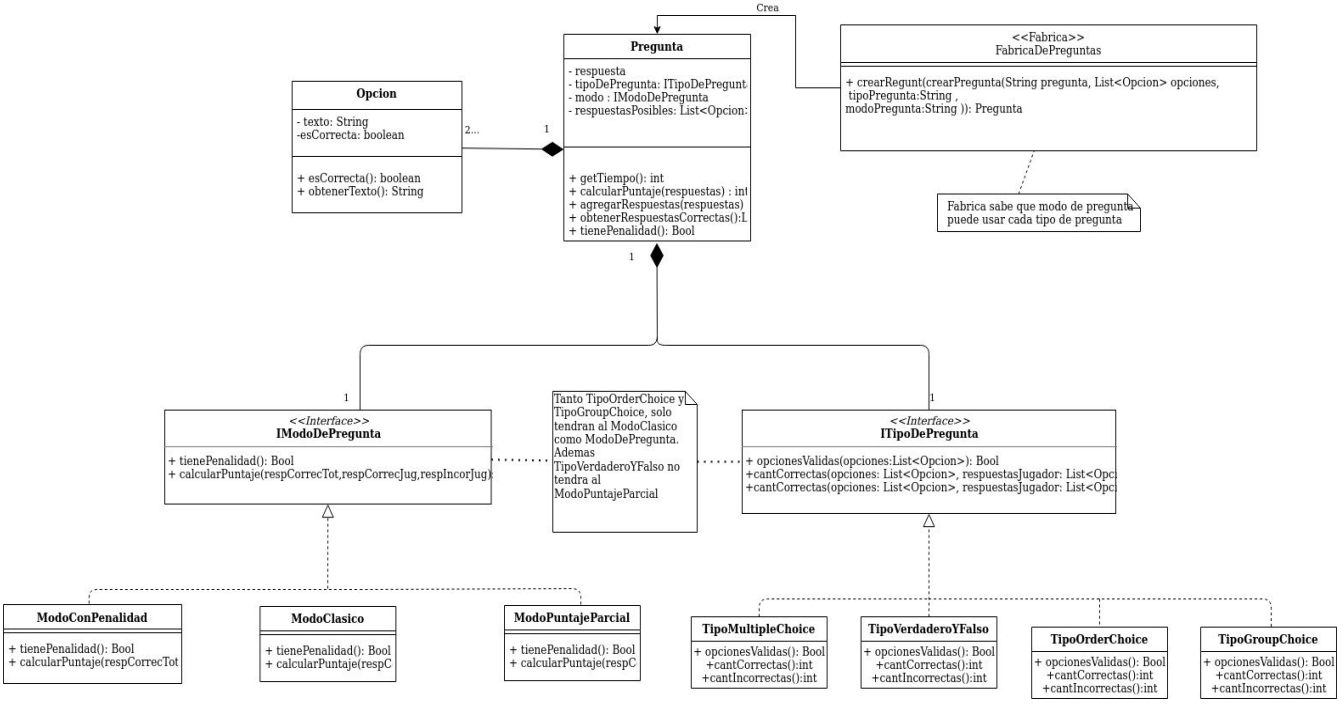

Figura 4: Diagrama de Pregunta.

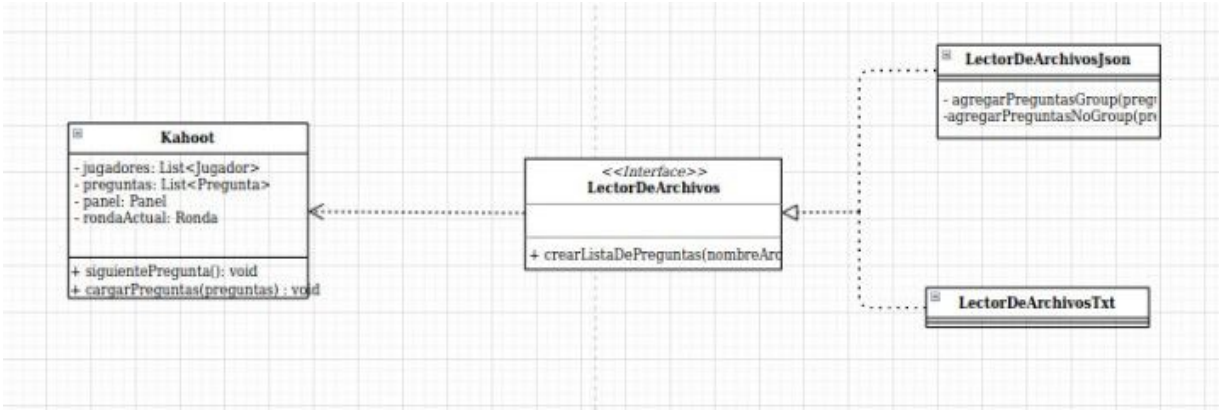

Figura 5: Diagrama de interfaz de LectorDeArchivos.

5. Detalles de implementación

5.1. Inicialización de preguntas

Actualmente le proveemos dos formas distintas de cargar las preguntas al programa.

Formato Propio
El archivo con el que inicializamos las preguntas contenía las líneas con el siguiente formato Modo/Tipo/Pregunta/Opciones/OpcionCorrecta .

Siendo el contenido de cada campo el siguiente:

Modo: String
- ConPuntajePenalidad

- Clásico
- Parcial

Tipo: String
- MultipleChoice
- VerdaderoFalso
- OrderedChoice
- GroupChoice

Pregunta:

String

Opciones: String,String,String

OpciónCorrecta: String

Ejemplo:

Clasico/MultipleChoice/¿Como se llama tu mascota?/Nestor,Chewbacca,Godzilla/Godzilla

Formato JSON

El formato JSON mantiene la misma idea de formato propio, representando a los modos, las opciones y las preguntas como strings pero la diferencia principal es que están separadas por categorías y que se utilizan los arrays que provee el formato JSON

Ejemplo de MultipleChoice / Verdadero Falso:

```
{
            "MultipleChoice":
            [
                {
                "modo": "ConPuntajeParcial",
                "texto" : "¿Quién ganó gran hermano 2015?",
                "opciones" : ["Matías Schrank", "Francisco Delgado", "Vos"],
                "opcionesCorrectas": ["Francisco Delgado"],
                }
            ]
}
```

Ejemplo de archivo OrderedChoice / GroupChoice

```
{
        "GroupChoice" :
            [
                    {
                            "modo": "Clásico",
                            "texto" : "Separe entre caballos (Grupo 1) y flores (Grupo 2).",
                            "opciones" : {"1": ["ANDALUZ", "APALUSA"], "2": ["NARCISOS"]}
                    }
            ]
}
```

```
{
        "OrderedChoice":
        [
                {
                        "modo": "Clásico",
                        "texto": "¿Cuál es el orden de nacimientos?",
                        "opciones" :
                        {
                                "1":["Jesús"],
                                "2":["El Dieguito (maradona)"],
                                "3":["Britney Spears"],
                                "4":["Bad Luck Brian"]
                        }
                }
        ]
}
```

5.2. Preguntas

Decidimos hacer una sola pregunta que es la que se encarga de realizar todas las cosas que tienen en común todos los tipos de pregunta, como obtener las preguntas o obtener las respuestas correctas pero delegando en el modo o el tipo para hacer cosas como calcular el puntaje o chequear si la cantidad de respuestas ingresadas en una primera instancia es la correcta.

5.3. Modo

Modo es el encargado de calcular el puntaje de cada pregunta y de los cuales hay 3 distintos.

ModoClasico es la encargada de calcular los puntajes de las preguntas que solo se gana puntos si se contesta todo bien y no tienen penalidad.

ModoConPenalidad es la encargada de calcular los puntajes de las preguntas que se ganan viendo cuantas contestaste bien y viendo cuantas contestaste mal y así pudiendo sumar como máximo la cantidad de respuestas correctas y como mínimo la cantidad de respuestas mínimas

ModoPuntajeParcial es la encargada de calcular los puntajes de las preguntas que tienen múltiples opciones pero no se puede conseguir una cantidad de puntos negativa

5.4. Tipo

Tipo es la encargada de ver si la cantidad de opciones que tiene una pregunta es la válida y de ver cuántas respuestas correctas e incorrectas contestó cada usuario.

6. Excepciones

ExtensionInvalidaExcepcion Esta excepción fue creada con el fin de notificar al usuario que el archivo de preguntas que se dio tiene una extensión erronea.

NoQuedanUsosExcepcion Esta excepcion fue creada con el fin de notificar al usuario que el Multiplicador que se utilizo no tenia usos disponibles y por ende la ejecucion no puede continuar.

7. Diagramas de secuencia

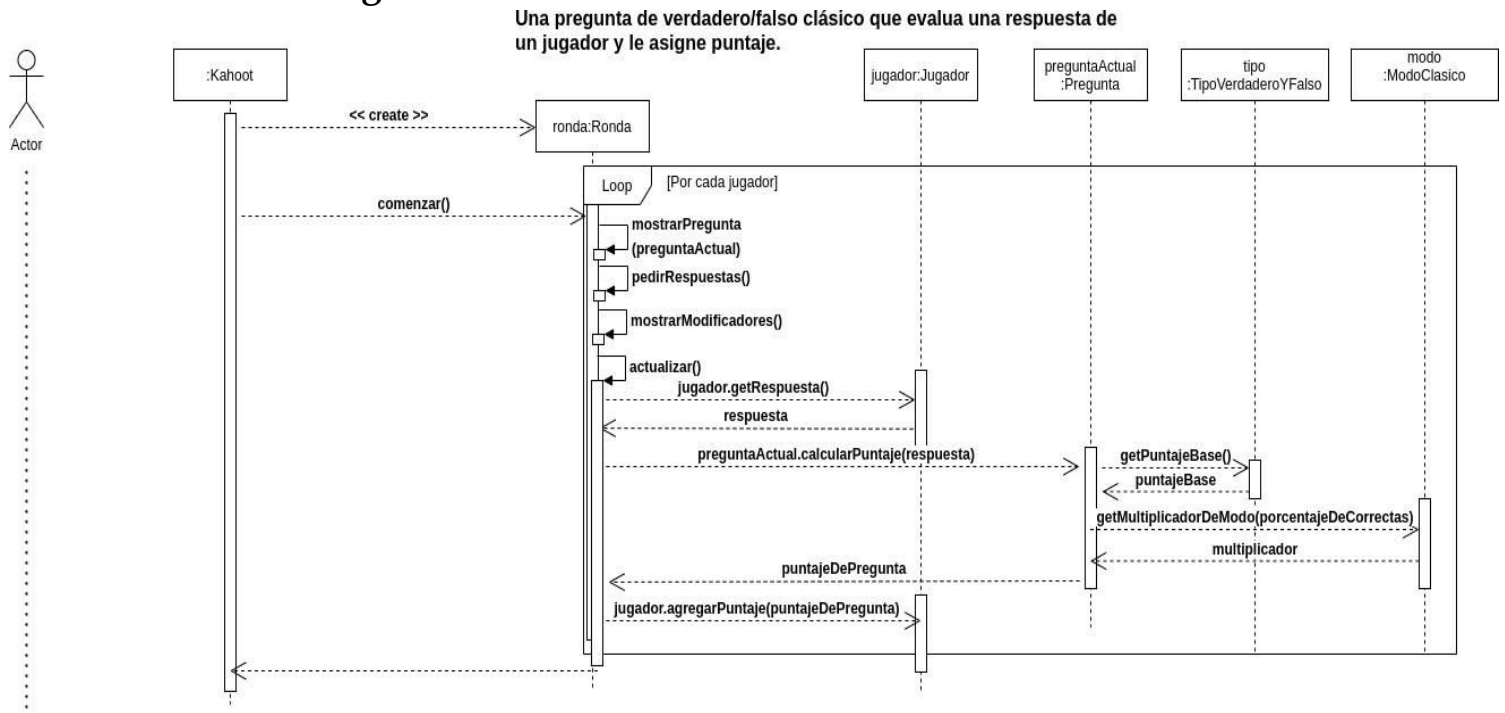

Figura 6: Se muestra el proceso de una ronda en el que un jugador responde una pregunta, se calcula el puntaje y se le agregan los puntos obtenidos.

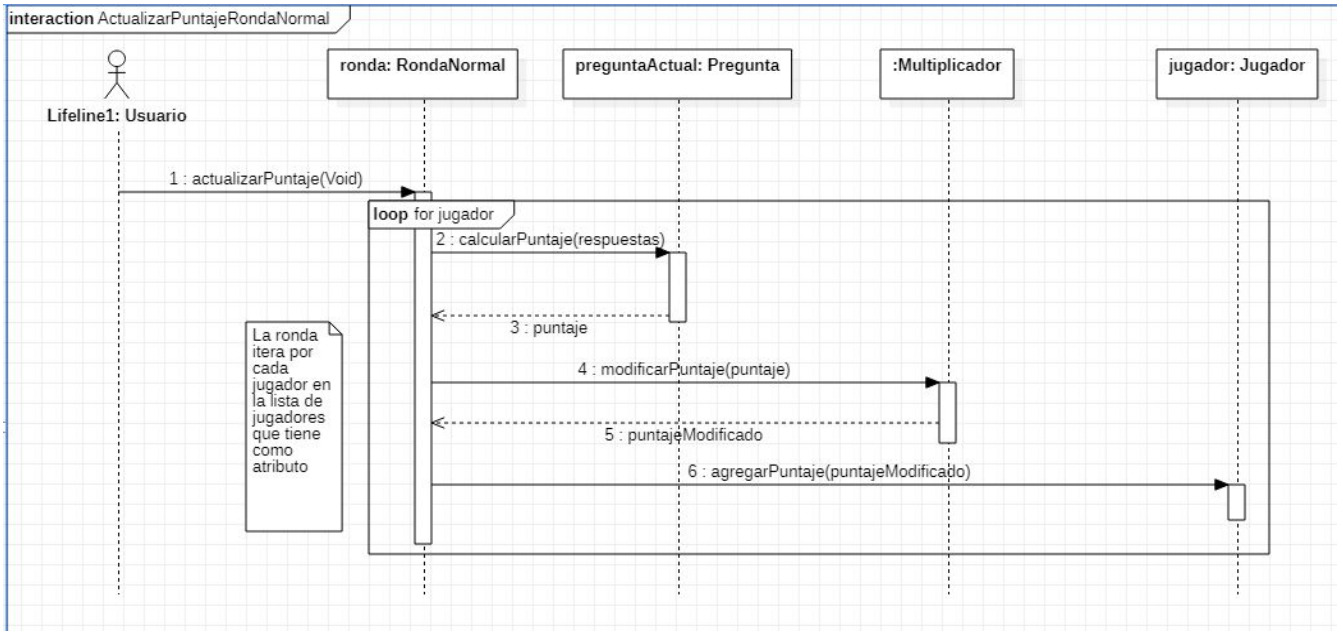

Figura 7: Se muestra el comportamiento de RondaNormal cuando le llega el mensaje actualizarPuntaje.

Debido a que el funcionamiento de la clase Pregunta al recibir el mensaje CalcularPuntaje, no es trivial se decidió extender su comportamiento en el siguiente diagrama de secuencia.

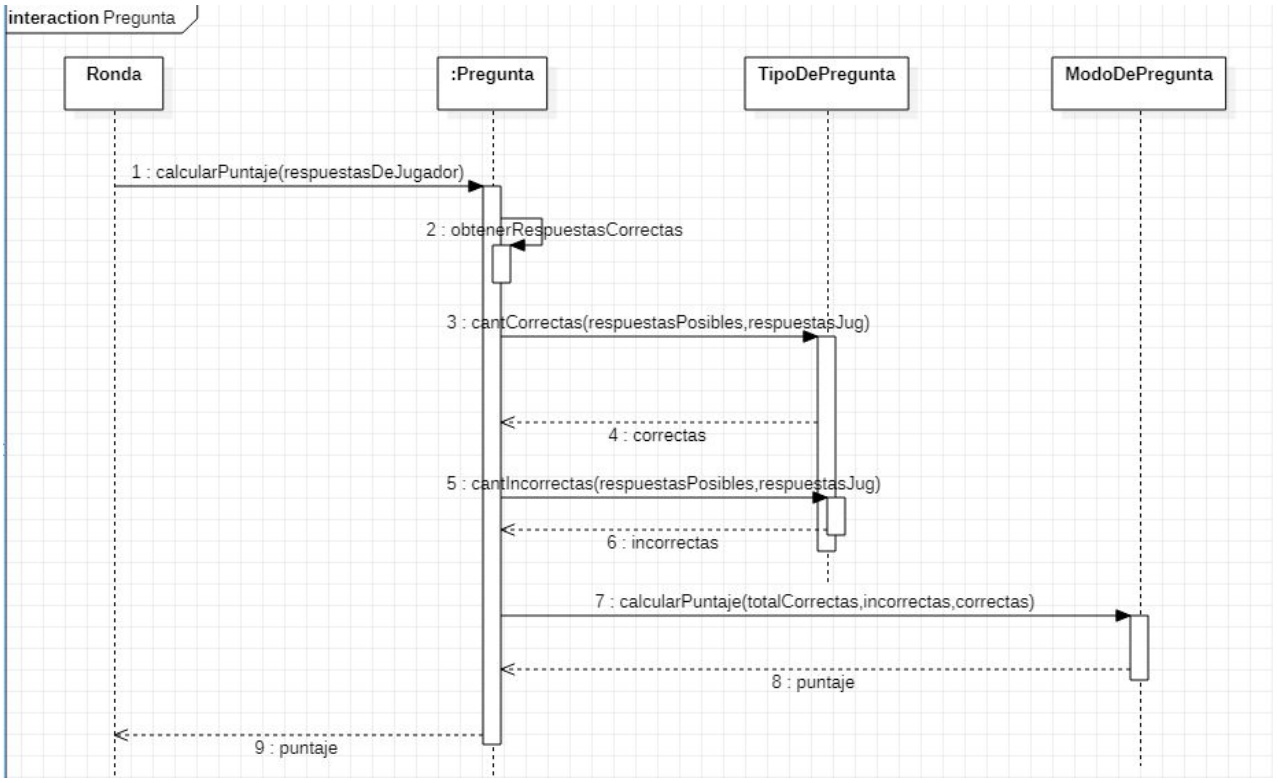

Figura 8: Se muestra el proceso interno por el cual en el diagrama anterior Pregunta calculaba el puntaje

8. Diagrama de paquetes

En esta sección vamos a mostrar los paquetes del proyecto, así como sus dependencias, quién conoce a quién y con quién se comunican los jugadores.

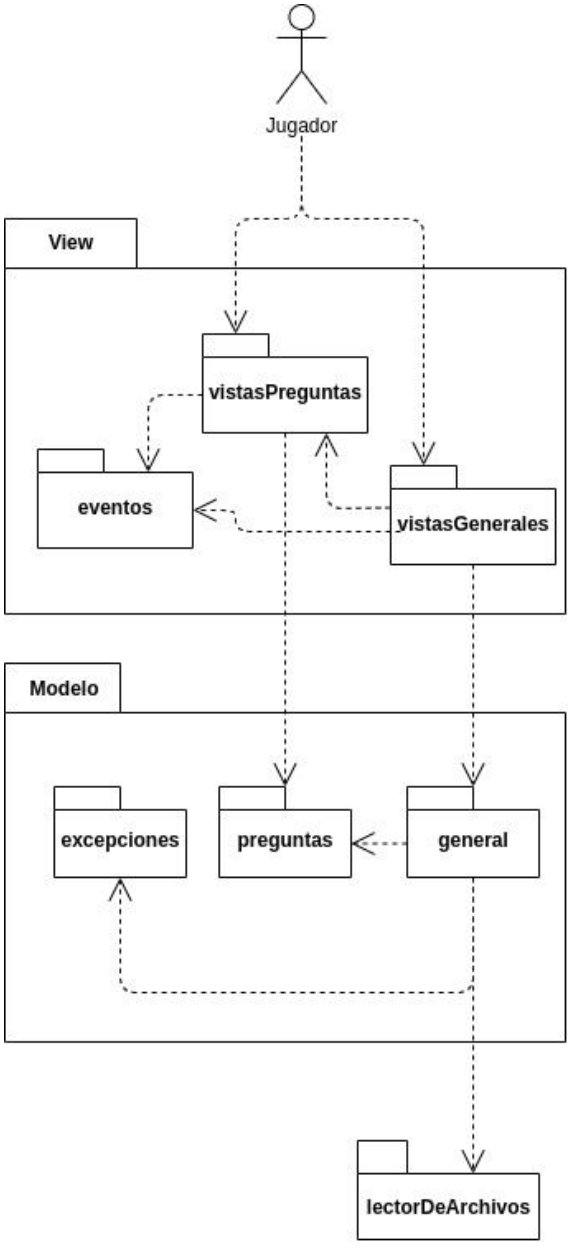

Figura 9: Se muestra el proceso de una ronda en el que un jugador responde una pregunta, se calcula el puntaje y se le agregan los puntos obtenidos.

9. Diagramas de estado

En el primer diagrama se pueden ver los distintos estados que existen en la clase Exclusividad.

Estos van a depender de la cantidad de usos disponibles que haya para el dado modificador de tipo exclusividad, siendo la cantidad inicial 2. En los primeros 2 estados, el comportamiento de la clase es el esperado y normal(aplica la exclusividad), pero una vez que pase por estos levantara la excepción "no tiene usos". También se puede notar que una vez que se utiliza la exclusividad, no se puede volver a un estado anterior por lo que solo hay un camino de transición de estados.

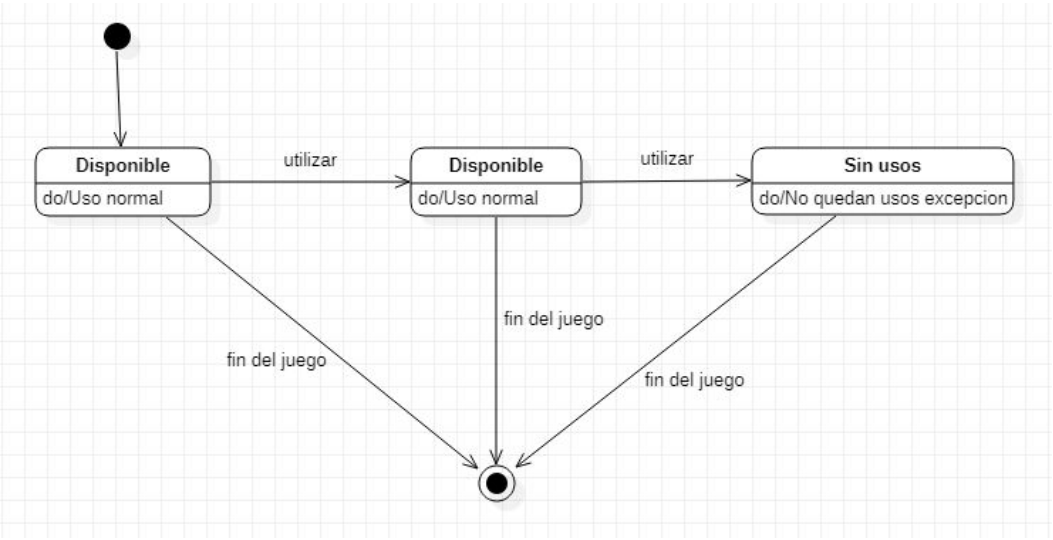

Figura 10: Diagrama de exclusividad.

El segundo diagrama trata de una clase similar a la mencionada anteriormente, con la excepción de que solo se tiene un uso al inicializarse. Además, al ingresar al estado "Disponible"que vendría a ser cuando se inicialice el objeto para un dado jugador, se creará un factor dependiendo del parámetro.

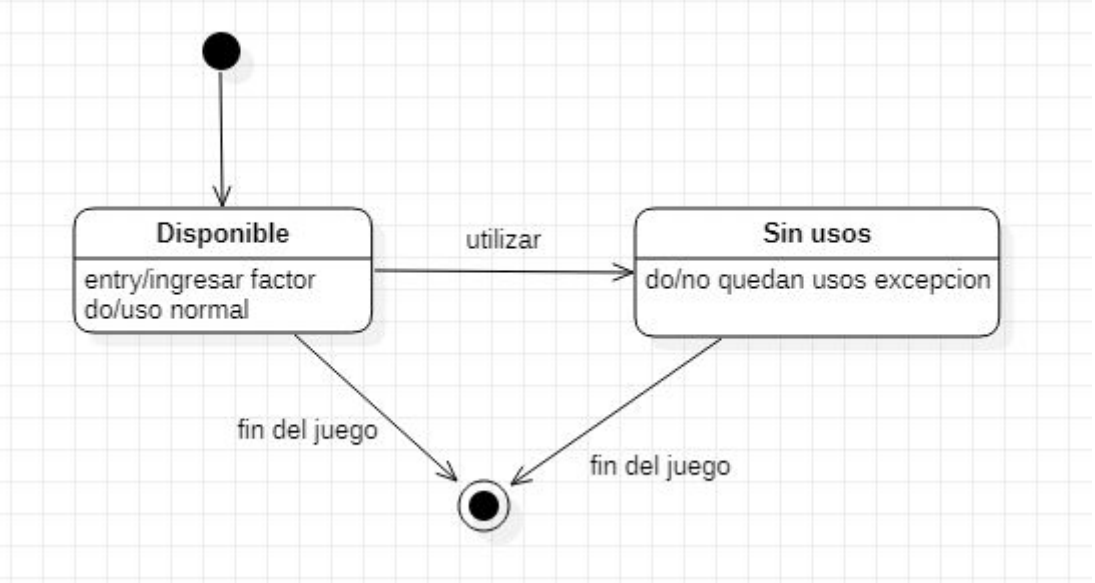

Figura 11: Diagrama de multiplicador.

En el último diagrama se pueden ver las distintas transiciones de estado internas que tiene el cronómetro. Se puede observar que si se usa el método de esperar 1 segundo, se vuelve a un estado anterior. Los distintos estados van a estar creados por los pulsos de reloj.

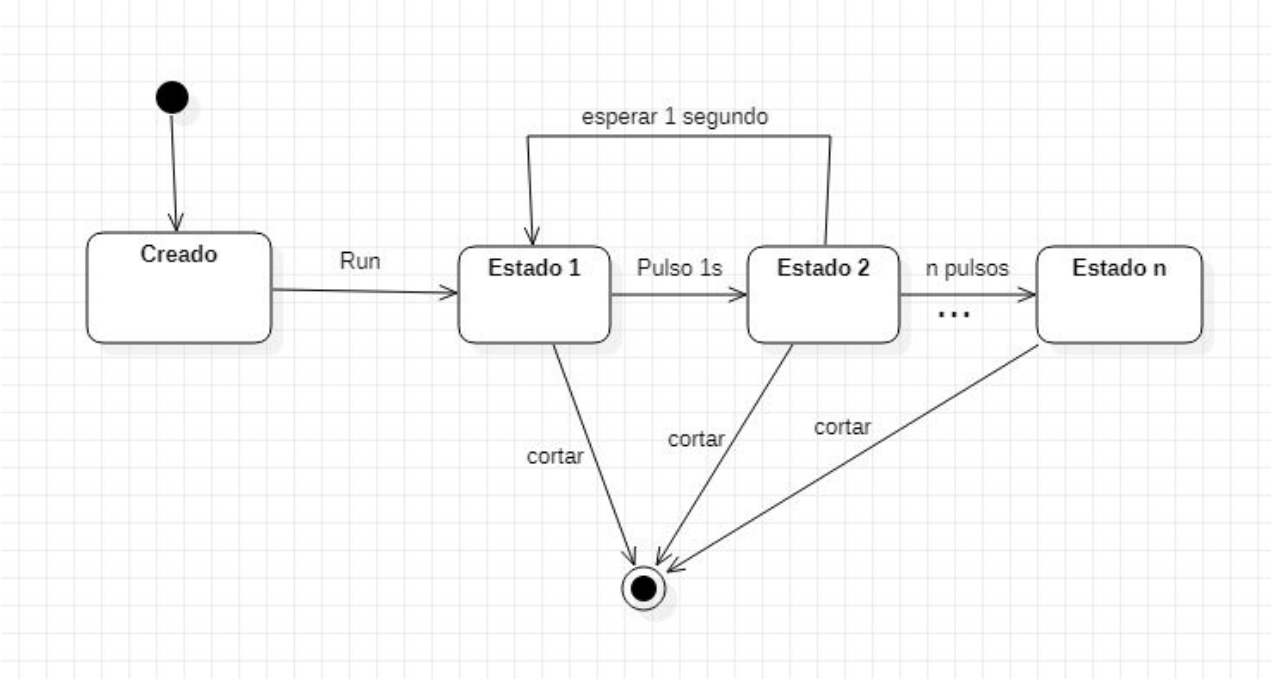

Figura 12: Diagrama de cronómetro.