



Trabajo Práctico 2 — AlgoThief

[7507/9502] Algoritmos y Programación III
Curso 2
Segundo cuatrimestre de 2021

Grupo 11
Corrector: Pablo Suarez

Alumno	Padrón	Mail
Matias Montero Kondratiuk	104633	mmontero@fi.uba.ar
Ignacio Agustín Pereyra	106022	iapereyra@fi.uba.ar
Tobias Luciano Rivero Trujillo	106302	trivero@fi.uba.ar
Agustín Nicolás González	106086	agusegonza@gmail.com
Gonzalo Joaquin Shimabukuro	107185	gshimabukuro@fi.uba.ar

Índice

1. Introducción	2
2. Supuestos	2
3. Modelo de dominio	2
4. Diagramas de clase	3
5. Detalles de implementación	4
5.1. Clases en particular	4
6. Diagramas de secuencia	5
7. Semana 0	6
7.1. Clases:	6
8. Semana 1	8
9. Semana 1 BIS	11
10. Semana 2	14
10.1. Inicio de semana	14
10.2. Fin de semana	14
10.3. Diagramas de clase	15
10.4. Caso de uso	16
11. Semana 3	19
11.1. Diagrama de clases final	19
11.2. Interfaz gráfica Básica	22
12. Semana 4: Entrega final	28
12.1. Patrones	28
12.2. Patrón Factory	28
12.3. Patrón: Singleton	29
12.4. Excepciones	29
12.5. Diagramas de Paquete	30
12.6. Diagramas de Estado	36
12.7. Interfaz	38

1. Introducción

El presente informe reune la documentación de la solución del segundo trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar una versión propia del juego Carmen Sandiego, utilizando los conceptos del paradigma de la orientación a objetos vistos hasta ahora en el curso.

2. Supuestos

- **Horas de sueño:** La hora en la que nuestro personaje se pone a dormir va a estar prefijada a las X horas. De ejecutarse una acción que nos cueste más tiempo, y nos deje en un horario mayor, se penaliza de las horas "diurnas".

Ejemplo práctico: Nuestro personaje duerme siempre a las 10. Son las 9 y recibe una herida que lo inhabilita por 2 horas ->Duerme a las 11 pero se despierta 1 hora más tarde.

- **Visitas a edificios:** Cuando el agente visita mas de un edificio en un mismo país, el tiempo transcurrido aumenta de 1 hasta 3 con incremento de 1 hora, en lugar de aumentar 1 hora solo al volver a un edificio ya visitado a modo de castigo. Cuando se cambia de país la duración de la visita se resetea, esto basado en el gameplay del juego original.
- **Rango:** Al iniciar el juego, el policía comienza con rango de novato, y al ir completando casos, promueve de rango, pero al salir e iniciar el juego nuevamente, vuelve a ser novato.

3. Modelo de dominio

El modelo del juego se hizo a partir de un primer diagrama de clases tentativo, y su posterior refactorización para lograr una solución de forma incremental. Sumado a las refactorizaciones post clase, resolvimos resolver muchos casos de duda con referencia al juego original.

Acá vamos a explicar por qué hicimos lo que hicimos y de qué forma lo hicimos.

4. Diagramas de clase

Diagrama de clases definitivo (Se actualizará semana a semana)

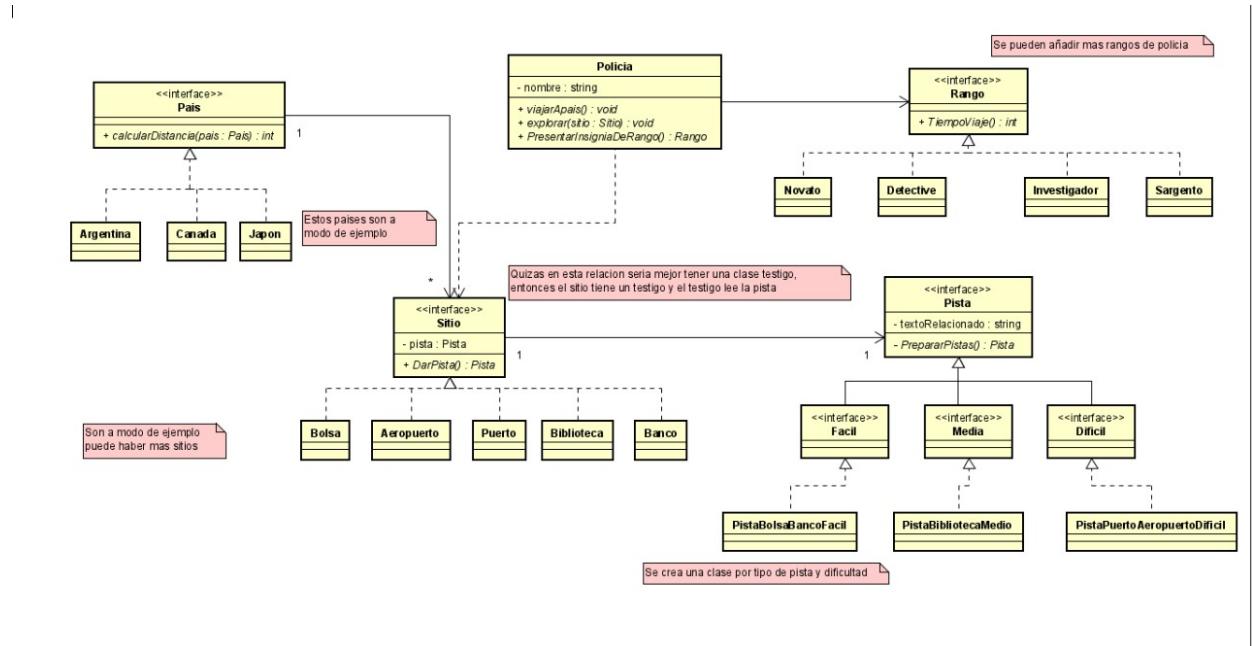


Figura 1: Diagrama de clases tentativo SEMANA 0.

5. Detalles de implementación

5.1. Clases en particular

- **Policía:**

Es la clase que representa a nuestro protagonista, el jugador. Posee varios atributos, entre los cuales se encuentran un nombre, el país actual en donde se encuentra, un rango, etc.

Al ser el policía el personaje principal, contiene también una gran cantidad de métodos representan todas las acciones que se espera que haga un policía, siempre en el contexto de nuestro juego. Algunas de estas pueden ser viajar a otro país, visitar algún edificio en específico, emitir una orden de arresto, etc.

- **Partida:**

La clase partida es la que prepara todo el juego, es la que inicializa todo lo que sea necesario para comenzar la partida, y también es la que determina cuando el juego acaba.

- **País:**

La clase país es el sitio donde interactúa el policía. Cada país tiene sus sitios y por lo tanto sus pistas, además de países vecinos a determinadas distancias.

- **Edificio:**

La clase donde el policía investiga para obtener pistas, al ser una abstracción existen edificios de todo tipo: banco, puerto, biblioteca y cada una entregara pistas de cierto tipo.

- **Rango:**

Cada policía tiene un rango, una clase para poder afectar el comportamiento que tendrá y las herramientas y dificultades que se presentaran. El mismo evoluciona a la vez que se resuelven casos

- **Arma:**

Clase la cual lastima al policia, para hacerle perder tiempo.

- **Caso:**

Cada partida permitirá jugar al usuaria todos los casos que quiera, los mismo representan .el nivel.en el que se encuentra, la dificultad de caso, en relación a la pistas dependerá del rango del policía.

- **ComputadoraInterpol:**

Esta clase representa la herramienta que tiene un policía para poder filtrar criminales y poder llegar al culpable, emitir una orden de arresto y concretar (o no) el caso de manera satisfactoria.

- **Criminal:**

Esta clase representa al .enemigo"que tendremos que atrapar con las pistas del caso para poder concretar el nivel con una victoria. El mismo tiene un seria de características, las cuales sirven para poder encontrarlo.

- **Pista:**

La clase necesaria para poder desarrollar un caso y saber donde viajar y que características tiene el criminal para poder filtrarlo a través de la computadora, las mismas se obtiene al explorar edificios de un país.

■ EstadoJuego:

Esta clase representa el estado que puede tener un caso, "nivelz variara su comportamiento en relacion a como el jugador se mueva y accione.

■ Reloj:

La clase encargada de mover el tiempo, la condición de victoria de un caso, cada acción que se realice consume tiempo de una manera, y el reloj que tiene el policía permite modificar el tiempo para que se desarrolle el caso, la hora del juego sera afectada por intervalos de tiempo.

■ IntervaloTiempo:

Abstracción para poder administrar el consumo de tiempo de cada acción que puede ser diferente, el reloj entiende estos intervalos y actúa en base a ellos pudiendo hacer que exista el transcurso de tiempo en el juego.

Existen mas clases en el sistema pero que no son de relevancia total, para el entendimiento del programa y que solo oscurecerian, mas que iluminar, tales como clases de la interfaz o recursos del propio software como por ejemplo: clases de JavaFX, listable, Factory (que se tratará mas adelante en el informe), Paises, Criminales, botonHanddler etc..

6. Diagramas de secuencia

Los diagramas de secuencia se van a basar en los casos de uso solicitados para cada semana, y los mismos estarán en el apartado que le corresponde a dicha semana en particular.

7. Semana 0

Durante la primera semana de trabajo, tuvimos mucha comunicación para tratar de encontrar un buen modelo, con bases sólidas para encarar el problema de manera correcta. Si bien obviamente se presentaran algunos o muchos cambios en el transcurso del trabajo decidimos comenzar por lo siguiente:

7.1. Clases:

- policía
- ladrón
- objeto
- tiempo
- país
- sitio
- pista
- rango

Como idea general tratamos de ir a lo mas concreto del juego, es decir a lo primero que resuena en nuestra cabeza cuando leímos las reglas del juego publicadas. Se destaca que en los diagramas presentados puede haber clases de mas o faltantes que todavía no logramos decidir en donde encajar.

Además destacamos que el modelo esta muy sujeto a cambios por las tempranas fechas de desarrollo, en la próxima semana seguiremos agregando/ modificando el sistema.

A continuación se adjuntan los modelos requeridos para las primeras semanas de desarrollo.

Casos de uso y diagramas de la semana 0.

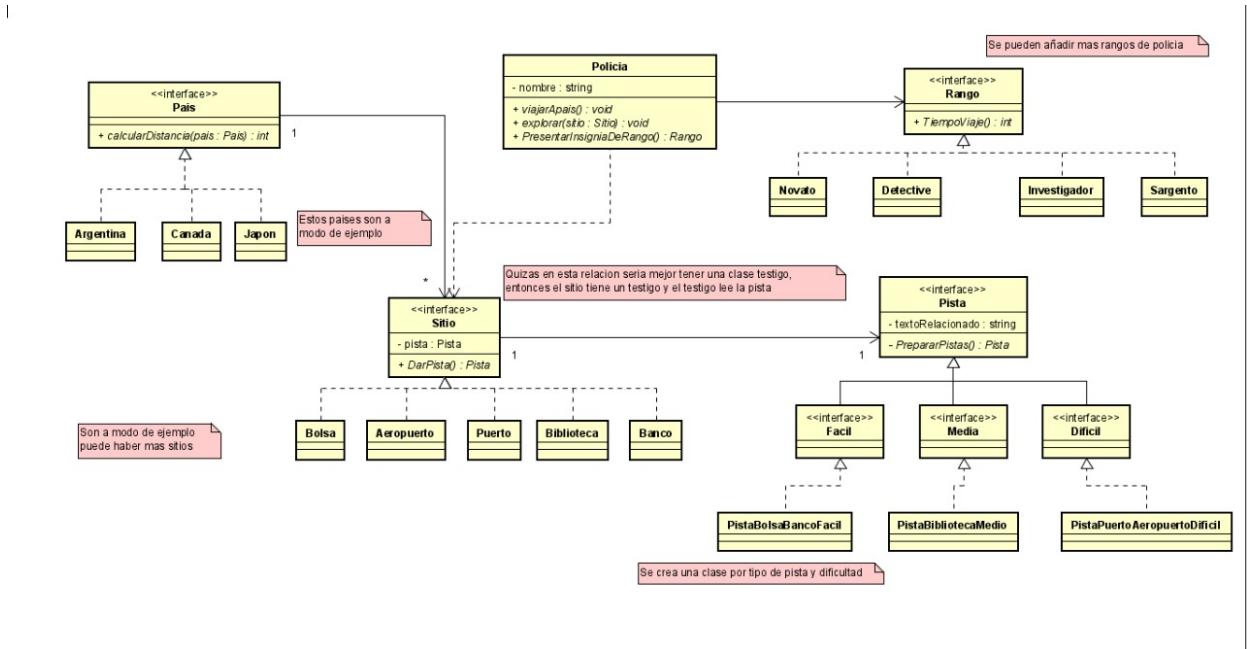


Figura 2: Diagrama de clases tentativo SEMANA 0.

8. Semana 1

Casos de uso y diagramas de secuencia de la semana 1.

Figura 3: Diagrama de clases tentativo SEMANA 1.

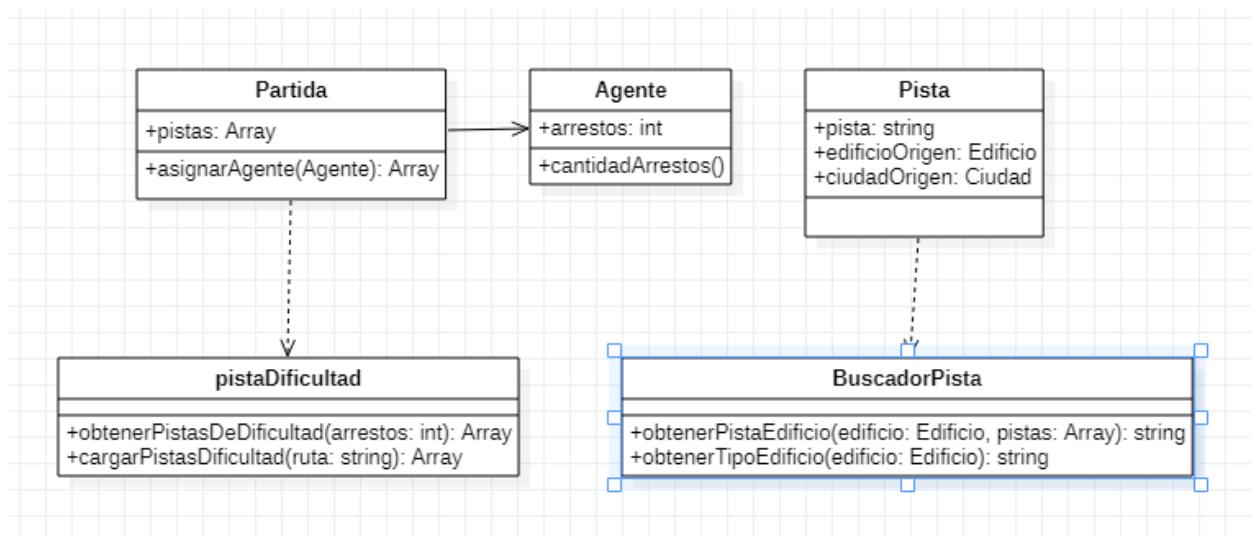


Figura 4: Diagrama de clases tentativo SEMANA 1.

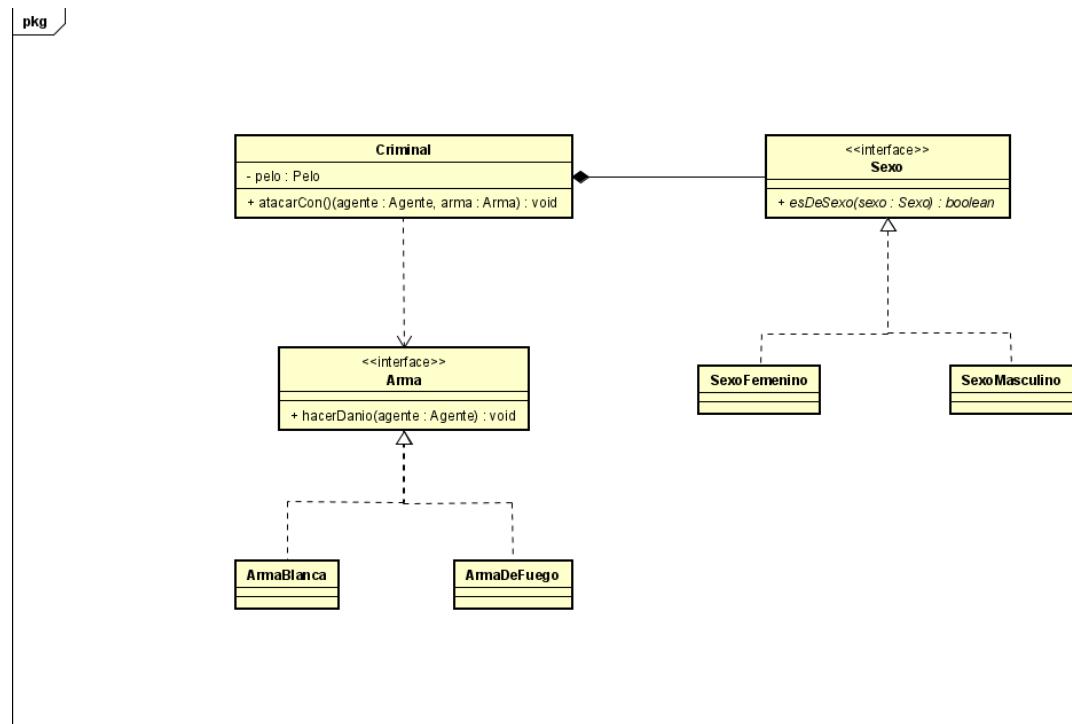


Figura 5: Diagrama de clases tentativo SEMANA 1.

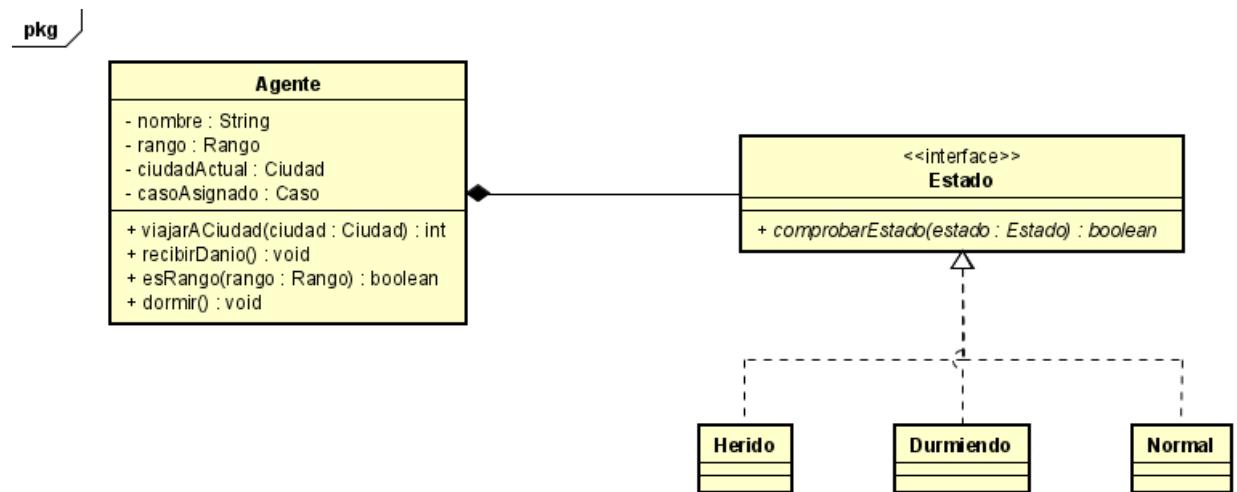


Figura 6: Diagrama de clases tentativo SEMANA 1.

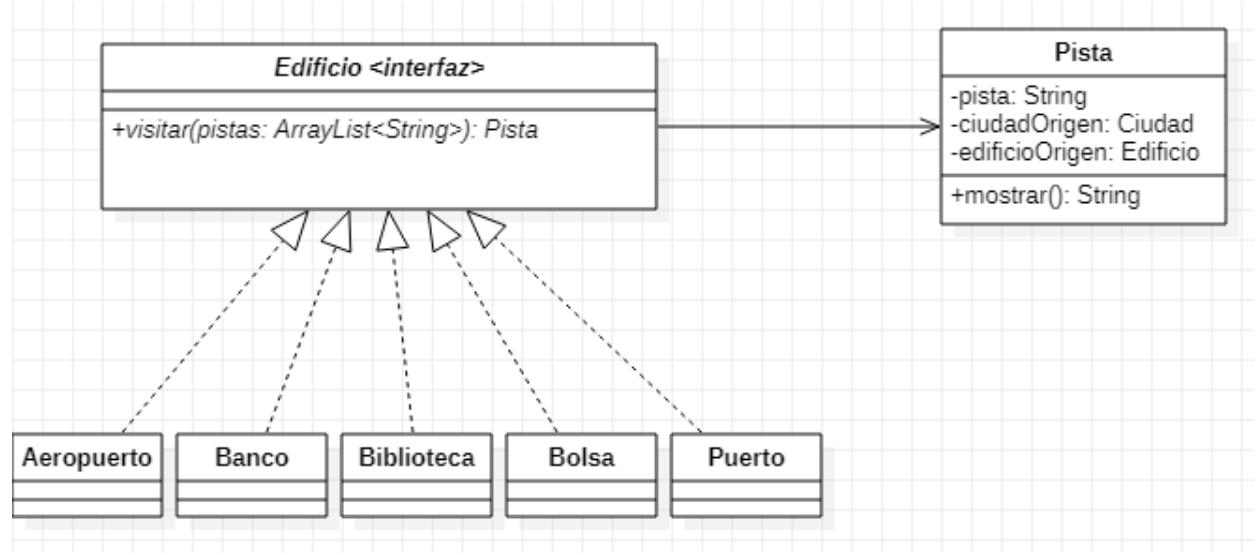


Figura 7: Diagrama de clases tentativo SEMANA 1.

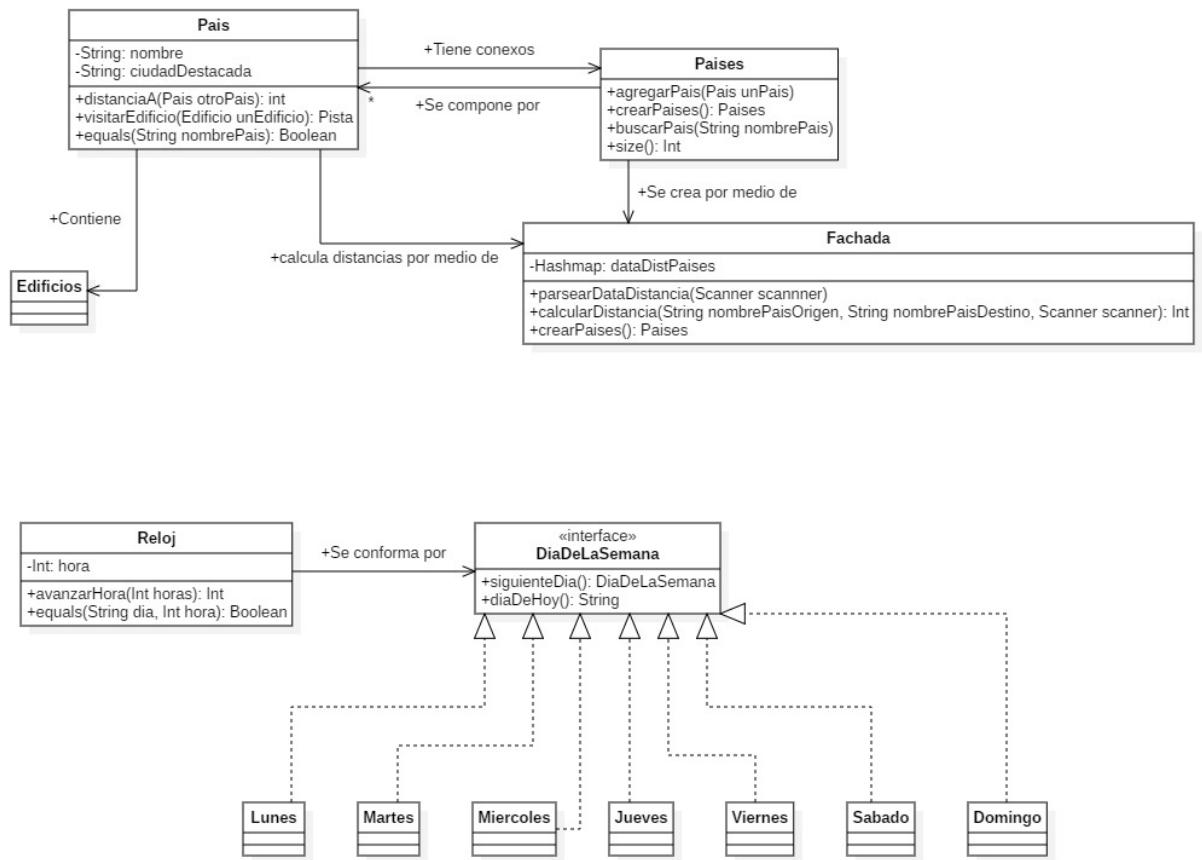


Figura 8: Diagrama de clases tentativo SEMANA 1.

9. Semana 1 BIS

Casos de uso y diagramas de secuencia de la semana 1.

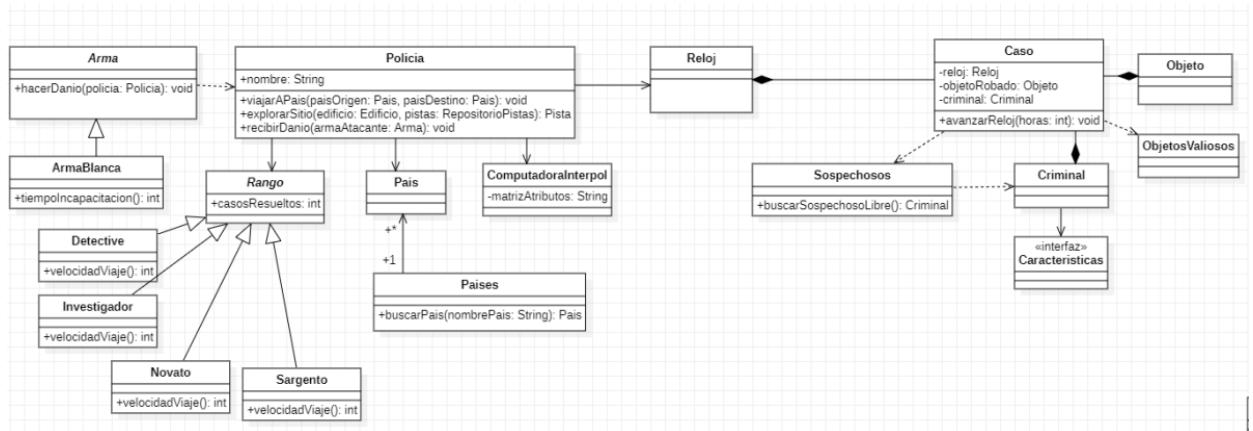


Figura 9: Diagrama de clases tentativo SEMANA 1.

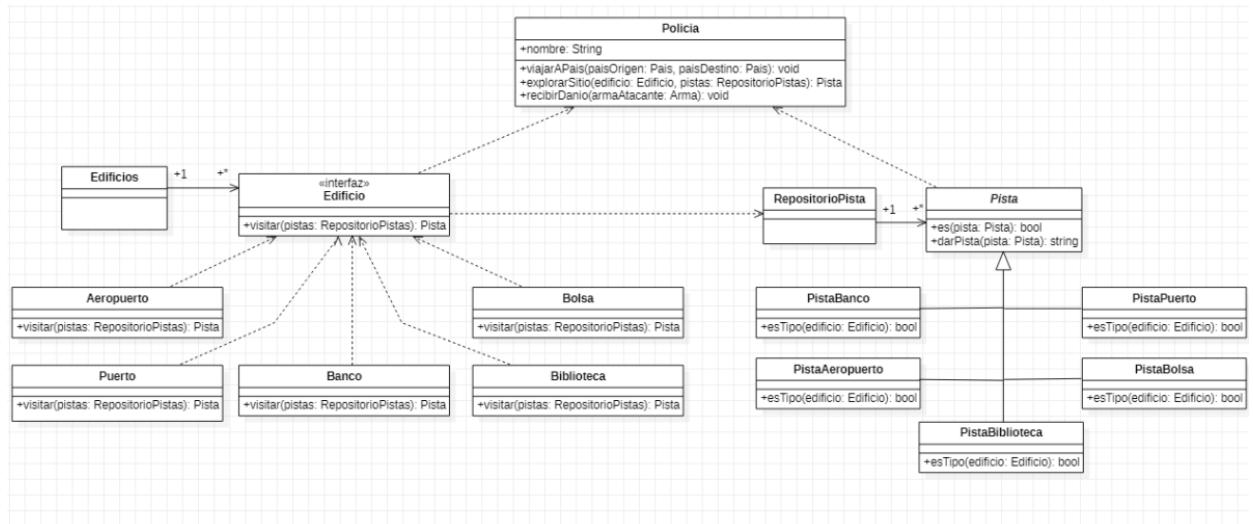


Figura 10: Diagrama de clases tentativo SEMANA 1.

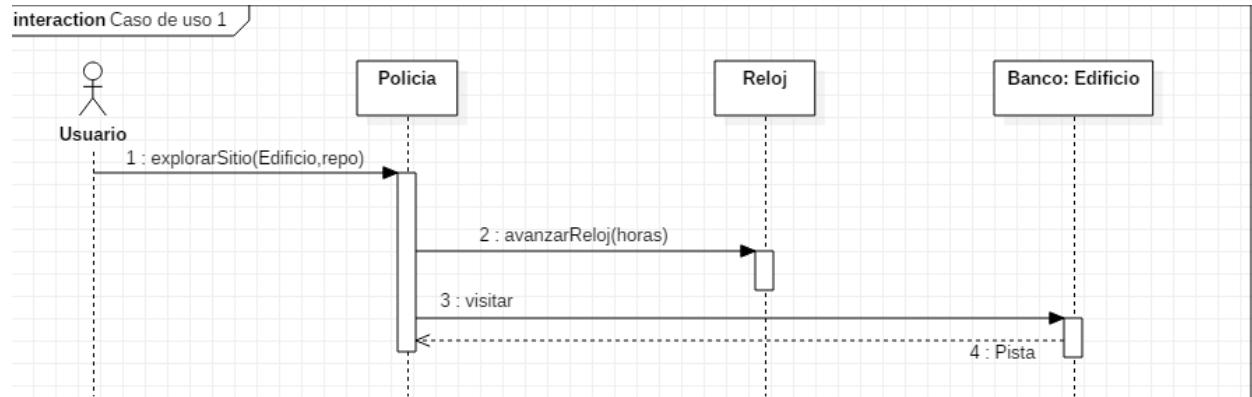


Figura 11: Seq. Caso 1

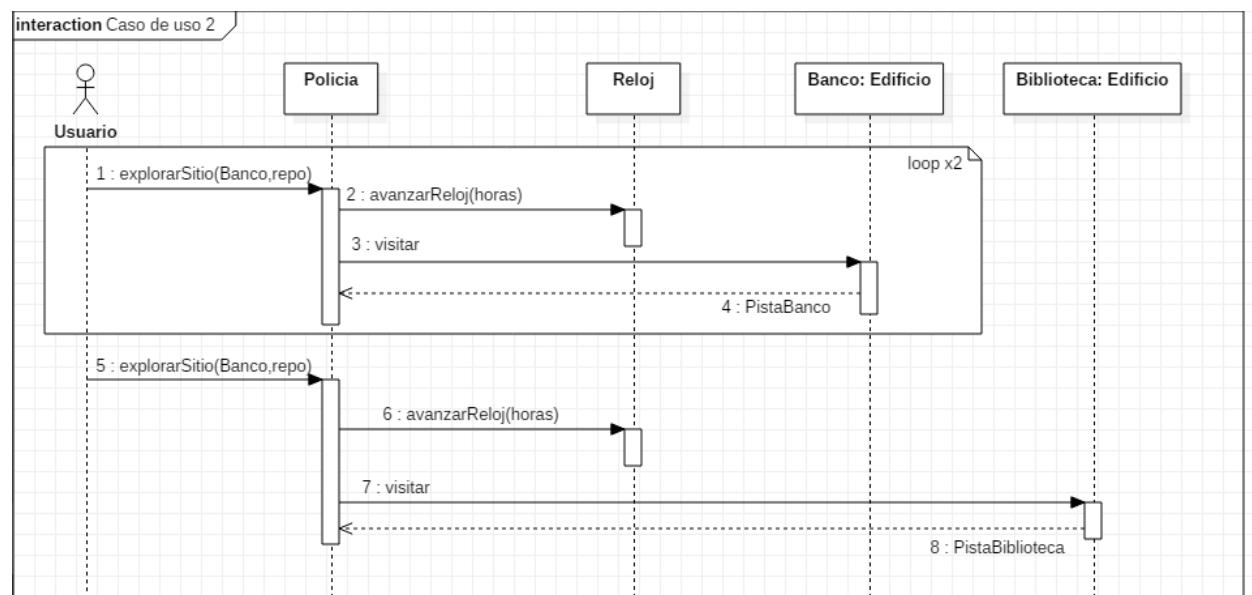


Figura 12: Seq. Caso 2

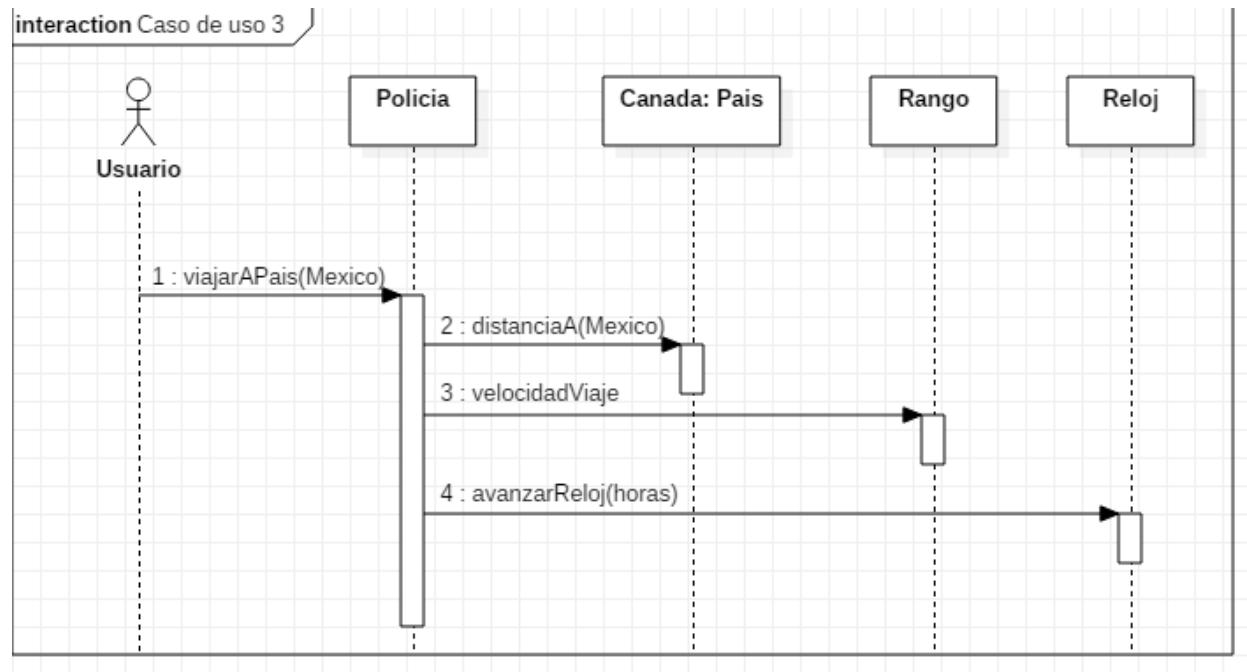


Figura 13: Seq. Caso 3

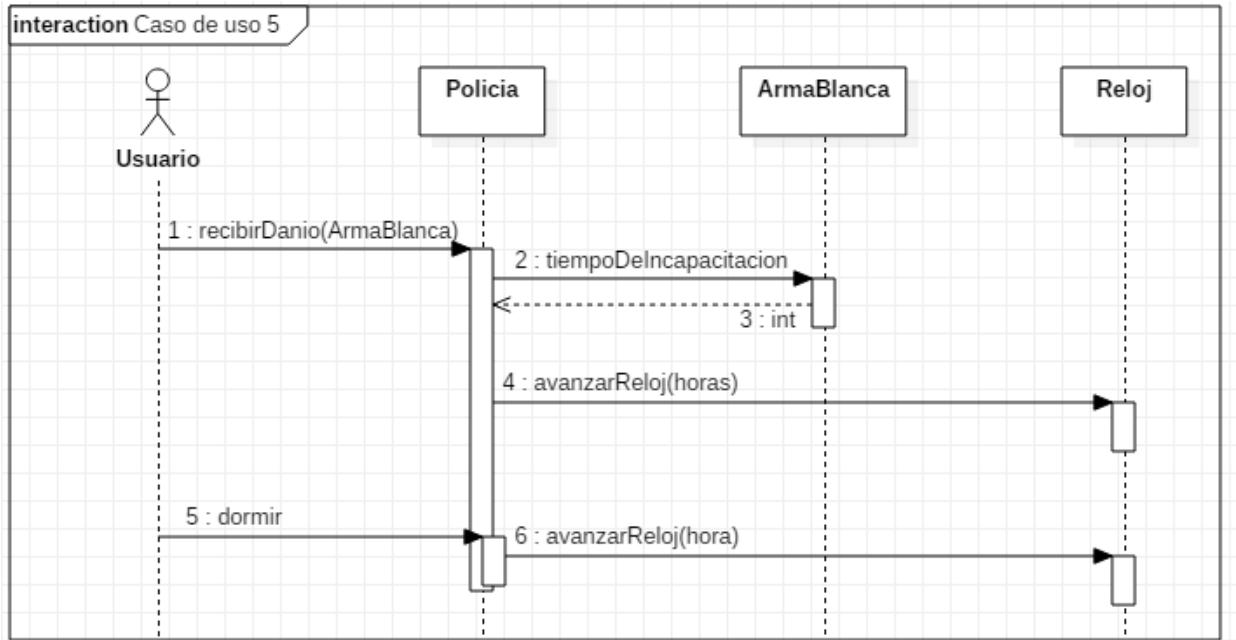


Figura 14: Seq. Caso 5

10. Semana 2

10.1. Inicio de semana

En esta semana decidimos en concentrarnos plenamente en el modelo, sin aún haber programado exhaustivamente nada, para no construir sobre una base poca sólida.

Luego de esta semana haber descargado y probado un tiempo del juego en el que se baso la idea del trabajo práctico, logramos encontrar información muy útil para el desarrollo del mismo, permitiéndonos corregir o agregar mas objetos al modelo presentado en la semana anterior.

Esto nos permitió encontrar o pensar en clases muy útiles como lo son "Partida." "Tempo." ambos conceptos fundamentales para el desarrollo del mismo.

En esta semana pudimos agregar un objeto fundamental, la clase Reloj, la cual nos permitira interactuar con uno concepto importante como es el tiempo.

Esta nos parece uno de los objetos mas importantes ya que es uno de los limitantes del juego, todo acción influye en el tiempo y el mismo determina una victoria o no frente al caso (además de otros factores claro).

Expandimos la visión sobre el juego introduciendo objetos como al clase Partida, DiaDeLaSemana. Permitiendo una visión mas global sobre el sistema y lo propio a ser programado.

10.2. Fin de semana

Para este punto ya logramos tener una imagen mas o menos clara de lo pedido, y luego de la clase de consulta con nuestro corrector pudimos ponernos manos a la obra.

Comenzamos programando las clases mas básicas, que venían siendo parte del modelo desde sus inicios clases como policía, ladrón, pista, país entre otras fueron nuestros primeros objetivos divididos entre cada uno del grupo.

Un posible patrón que habíamos pensado en usar, fue el patrón fachada para la carga y seteo de los elementos necesarios para la partida, por ejemplo los países con sus distancias o la instancia de partida en si.

Dicha idea luego de la charla con nuestro corrector termino siendo descartada porque no estábamos cumpliendo con los requisitos propios del patrón para cuando y porque usarlo. Por lo tanto fue retirado del modelo.

Para tratar la clase tiempo nos vimos en un problema, todas las acciones consumen tiempo, es decir que de una u otra forma casi todos los objetos propios del juego deberían afectar el tiempo, lo cual nos presentaba una situación en la cual todos los objetos necesitarían tener acceso al reloj como atributo. Con esta duda nos presentamos a la revisión en la cual nuestro corrector luego de escucharnos nos recomendó el uso del patrón Singleton, con su debido advertencia.

Luego de pensar un poco y encontrarnos con que agregar singleton estaba presentando dificultades sumado a que al investigar vimos que no solo era un patrón algo problemático (pero aveces necesario) sino que también es un patrón que al aplicarlo es muy difícil desacoplarlo del programa. Por lo tanto después de pensar un poco logramos encontrar una solución al problema... No aplicar dicho patrón.

Todo esto debido a que pudimos encontrar y delegar la responsabilidad de "portar." el reloj a la clase policía la cual si bien no hace todas las acciones explícitamente interactúa con casi todos los objetos que afectan el tiempo, permitiéndonos eludir dicho patrón.

Para el final de la semana 2 ya tenemos presente un modelo relativamente estable y definitivo, que presentaremos la próxima semana. Pudimos programar y probar ya bastantes aspectos claves del sistema, presentado y adjuntando pruebas unitarias al mismo como medio de documentación, todavía tenemos ciertas dudas con el sistema de pistas y como estas relacionarlas con los edificio y el rango del policía.

10.3. Diagramas de clase

Durante la semana 2 Priorizamos la programación del modelo, siguiendo las líneas pautadas en las semanas anteriores por lo tanto no se presentaron cambios significativos en los diagramas de clase y los casos de uso fueron diagrama dos con las clases existentes de la semana anterior.

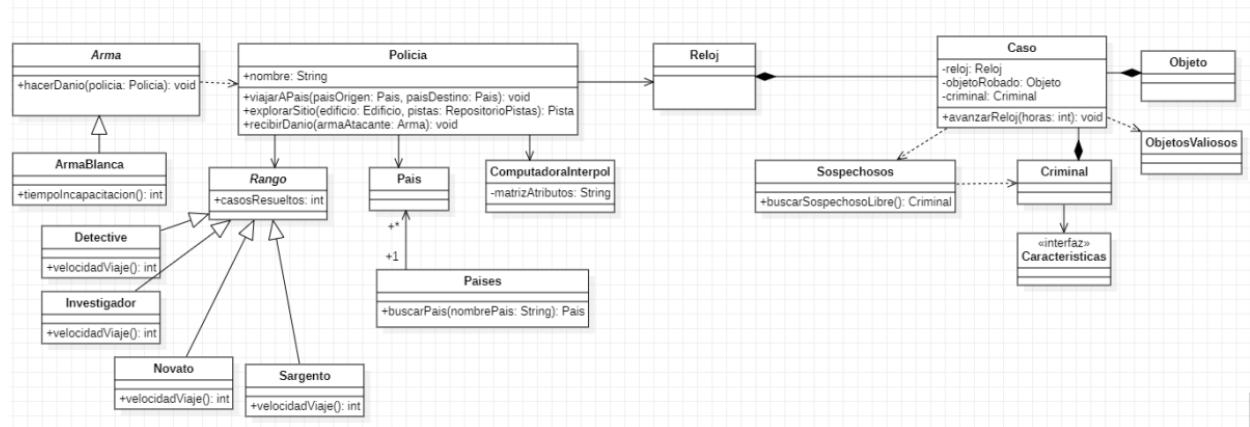


Figura 15: Diagrama de clases tentativo SEMANA 1.

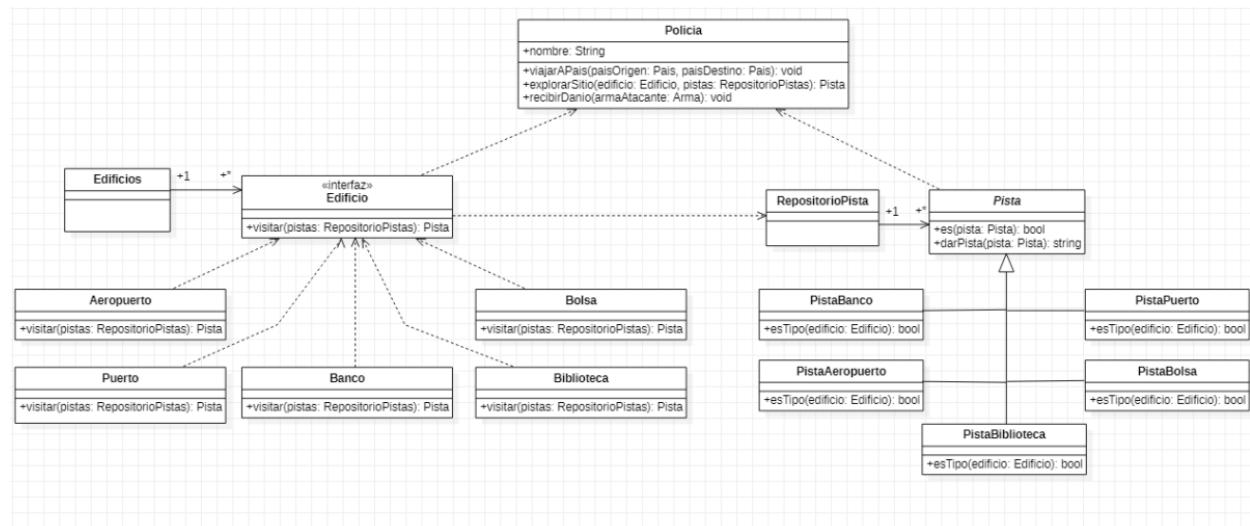


Figura 16: Diagrama de clases tentativo SEMANA 1.

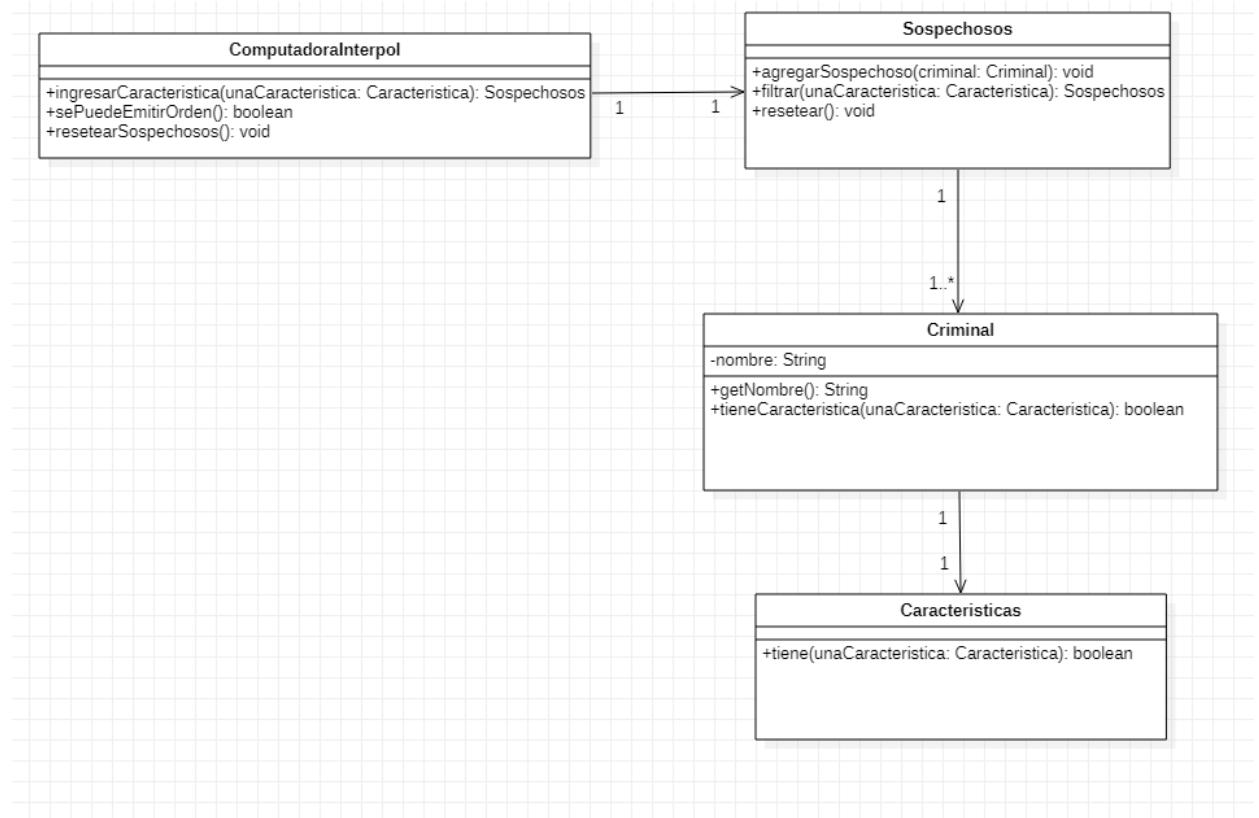


Figura 17: Diagrama de clases tentativo SEMANA 1.

10.4. Caso de uso

Se adjunta a continuación los casos de uso de la semana pertinente.

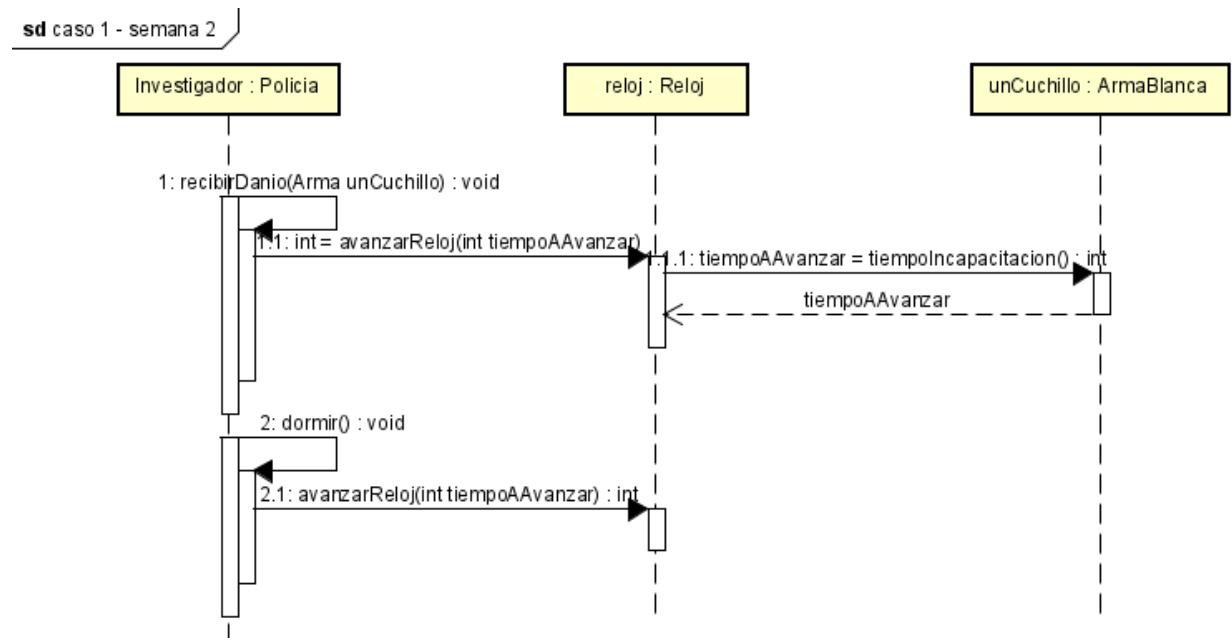


Figura 18: Seq. Caso 1

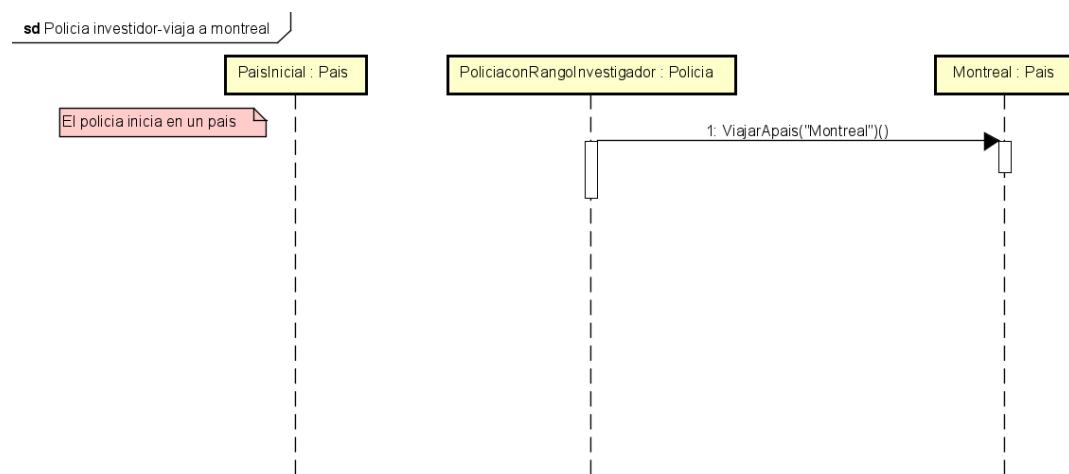


Figura 19: Seq. Caso 1

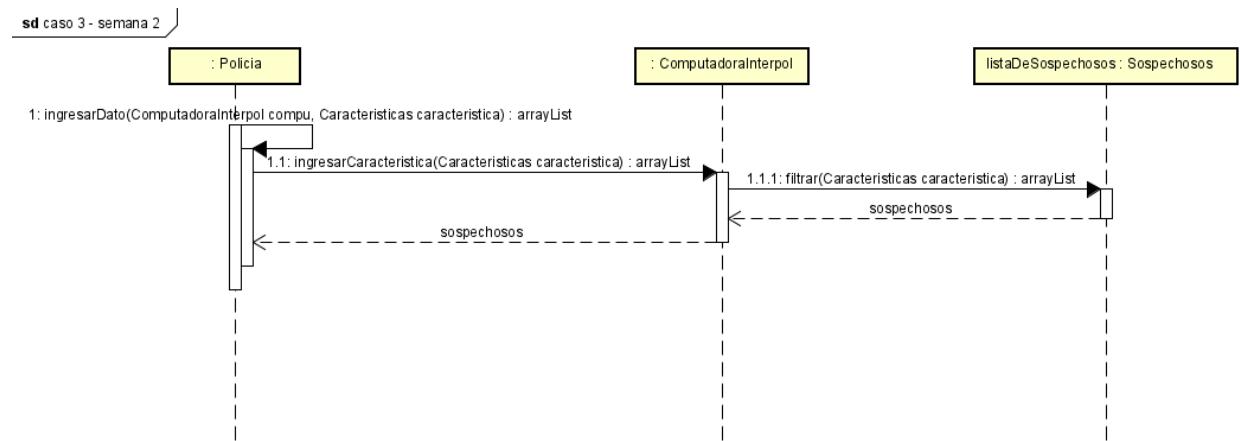


Figura 20: Seq. Caso 1

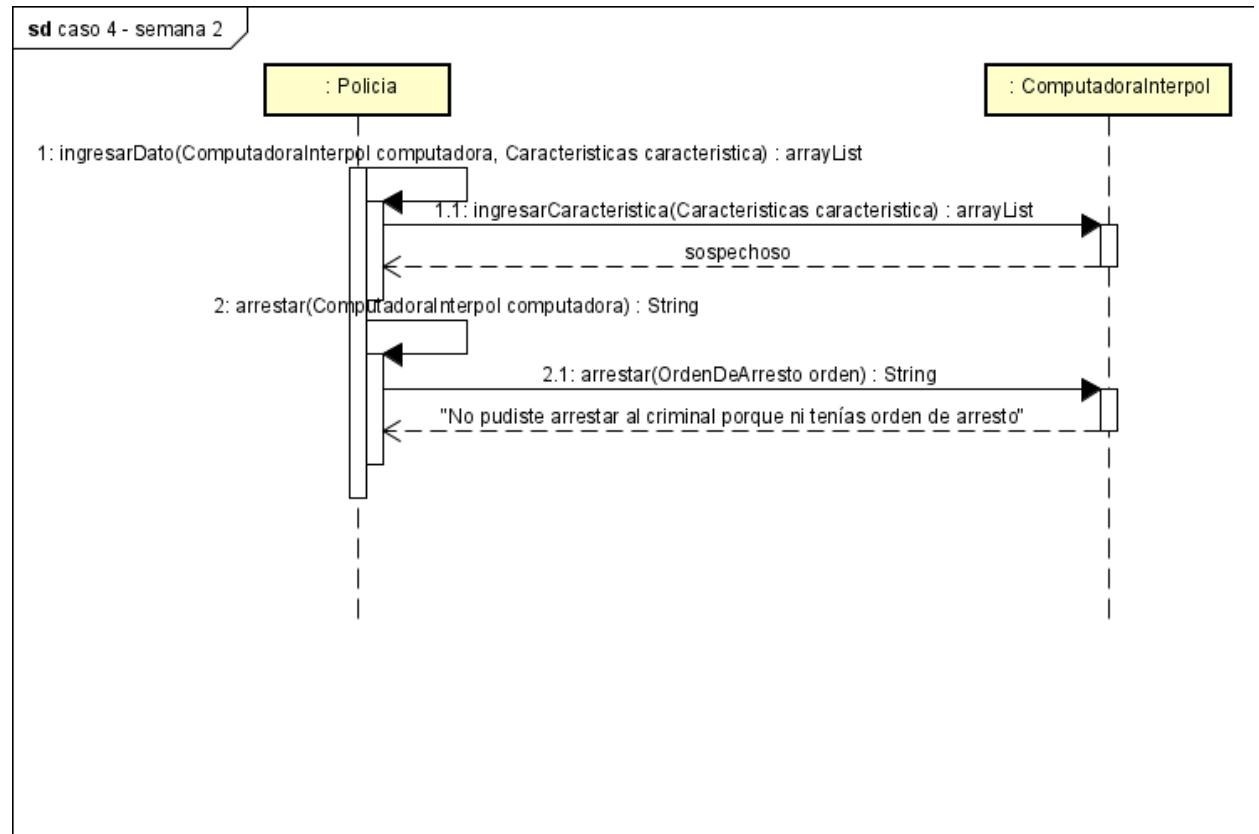


Figura 21: Seq. Caso 1

11. Semana 3

La semana 3 no presenta casos de uso, debido a que se presentan los diagramas de clases finales junto a un modelo definitivo del mismo acompañada de una interfaz gráfica básica y provisoria.

11.1. Diagrama de clases final

Luego de programar ya una gran parte del juego, pudimos encontrar problemas o baches de desarrollo en cuanto a las clases pensadas, su enfoque o responsabilidad, permitiéndonos poder arreglar dichas clases y crear nuevas con el objetivo de seguir los pilares de POO y sus buenas prácticas.

De acá en mas el proyecto avanzara en base a los diagramas que se presentaran a continuación y se hará un breve resumen sobre las decisiones tomadas y el porque de las mismas.

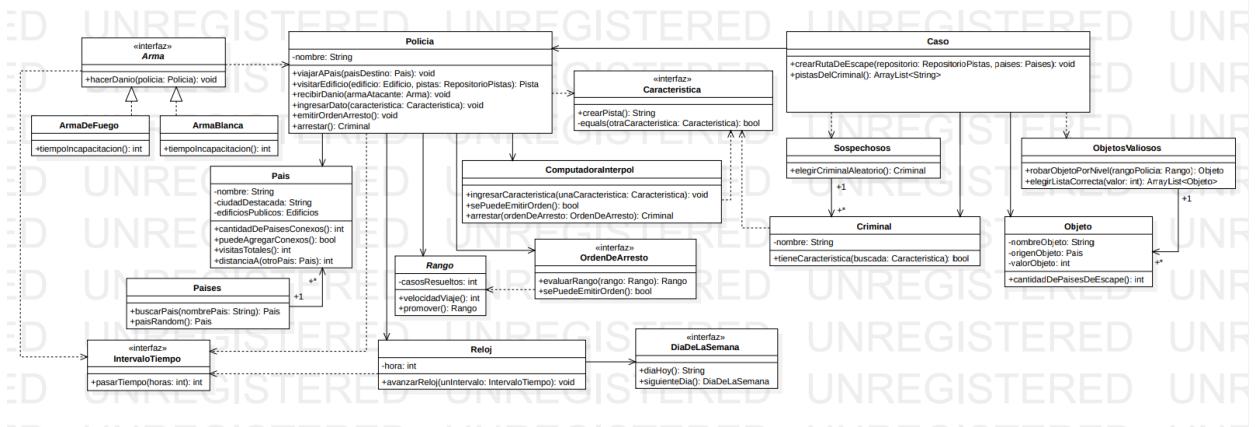


Figura 22: Diagrama de clase Modelo principal

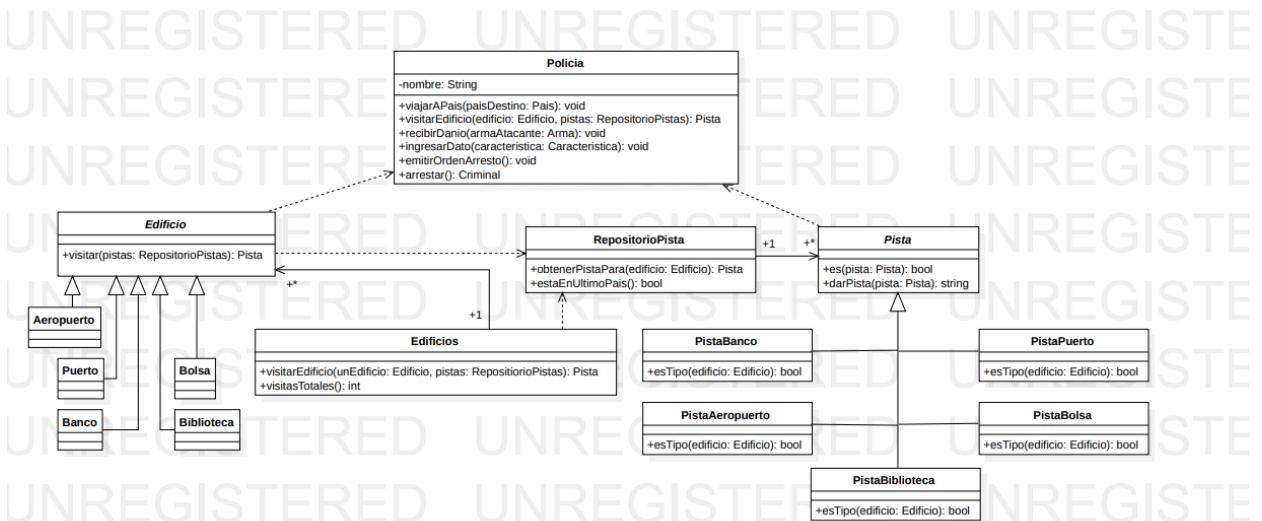


Figura 23: Diagrama de clase Policía-pista

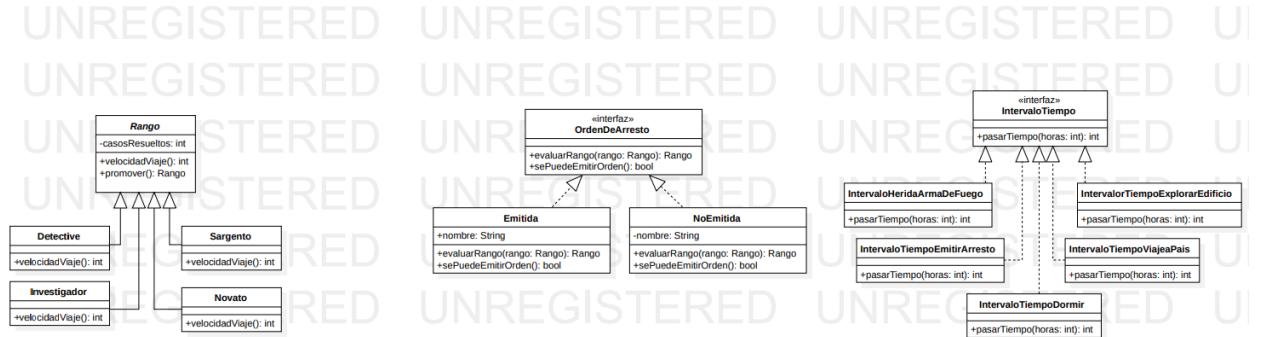


Figura 24: Diagrama de clase Rango-Arresto

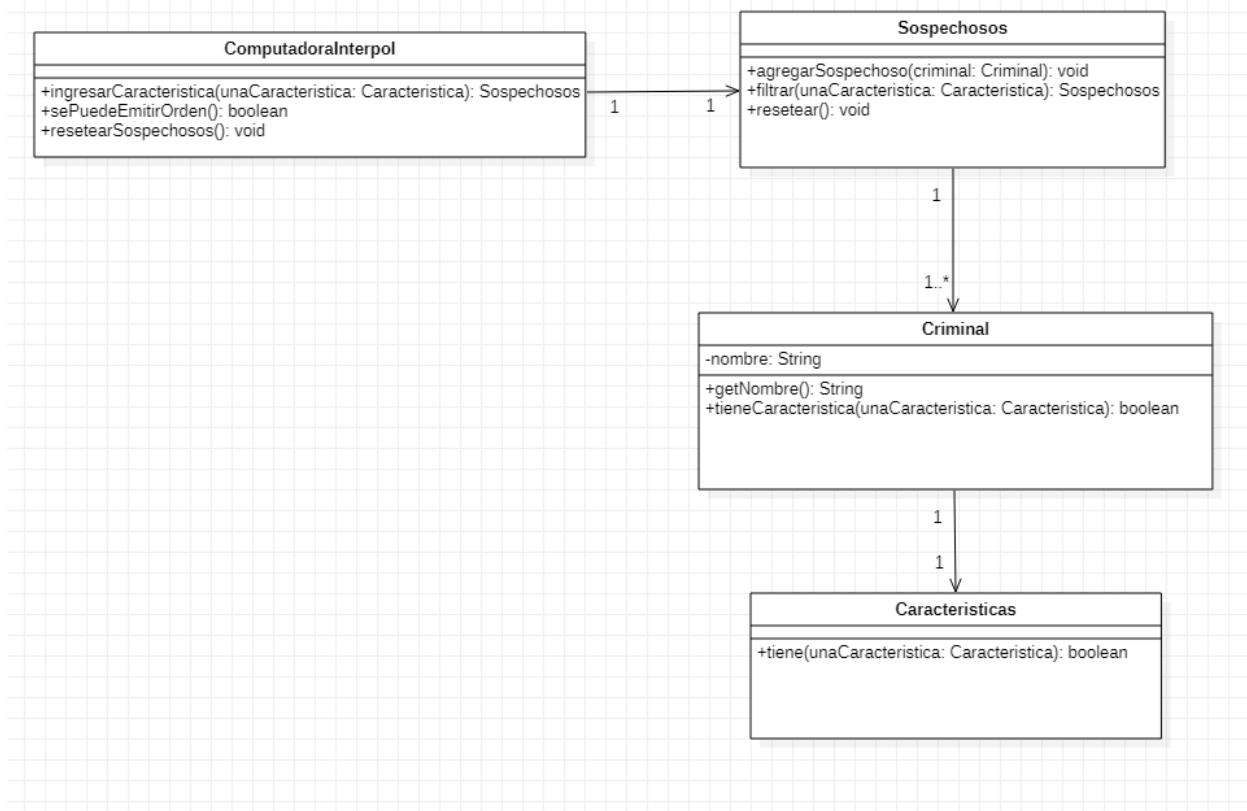


Figura 25: Diagrama de clase Computadora

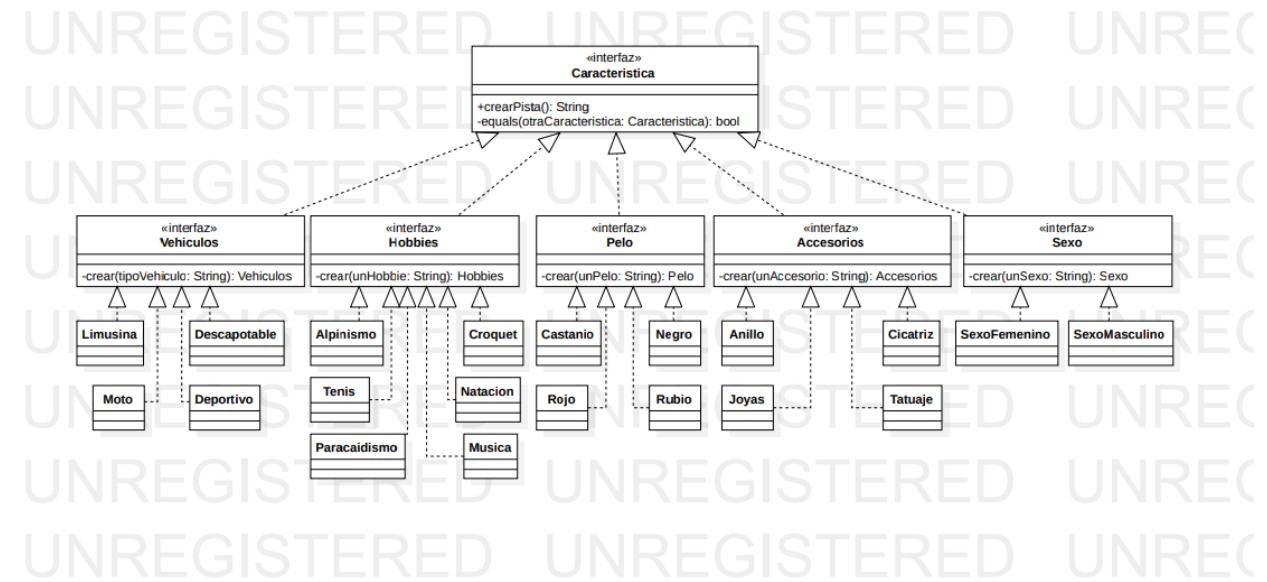


Figura 26: Diagrama de clase Características

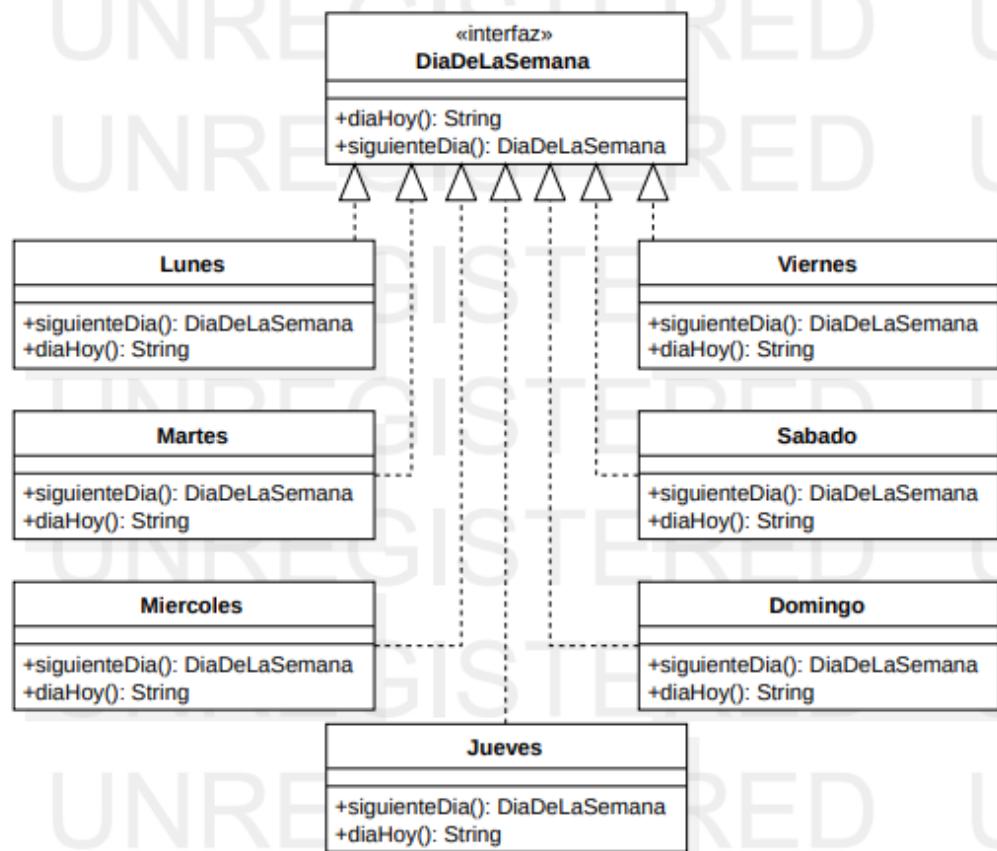


Figura 27: Diagrama de clase DiaSemana

11.2. Interfaz gráfica Básica

Se adjuntan las imágenes del prototipo de interfaz gráfica.

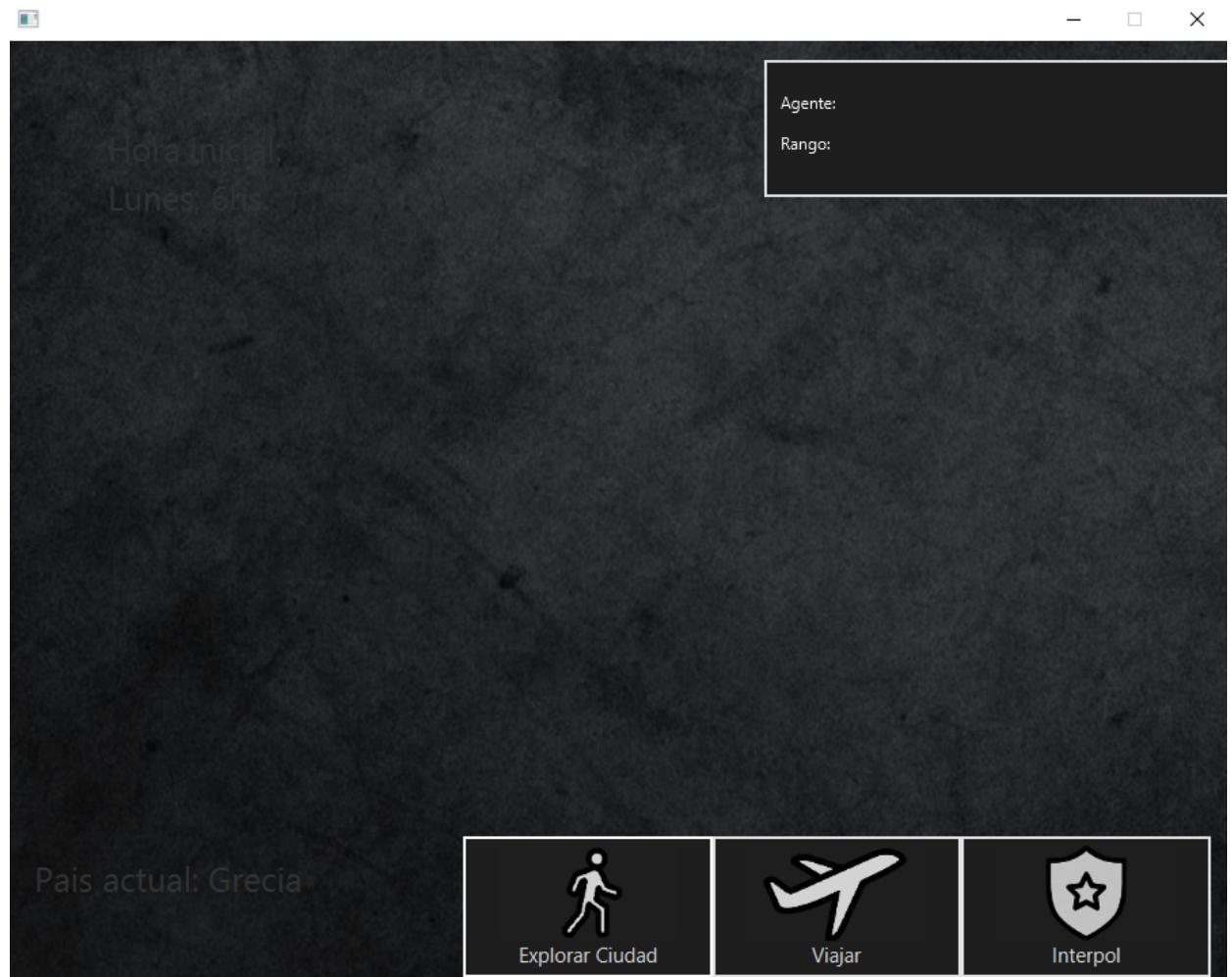


Figura 28: Diagrama de clase Características

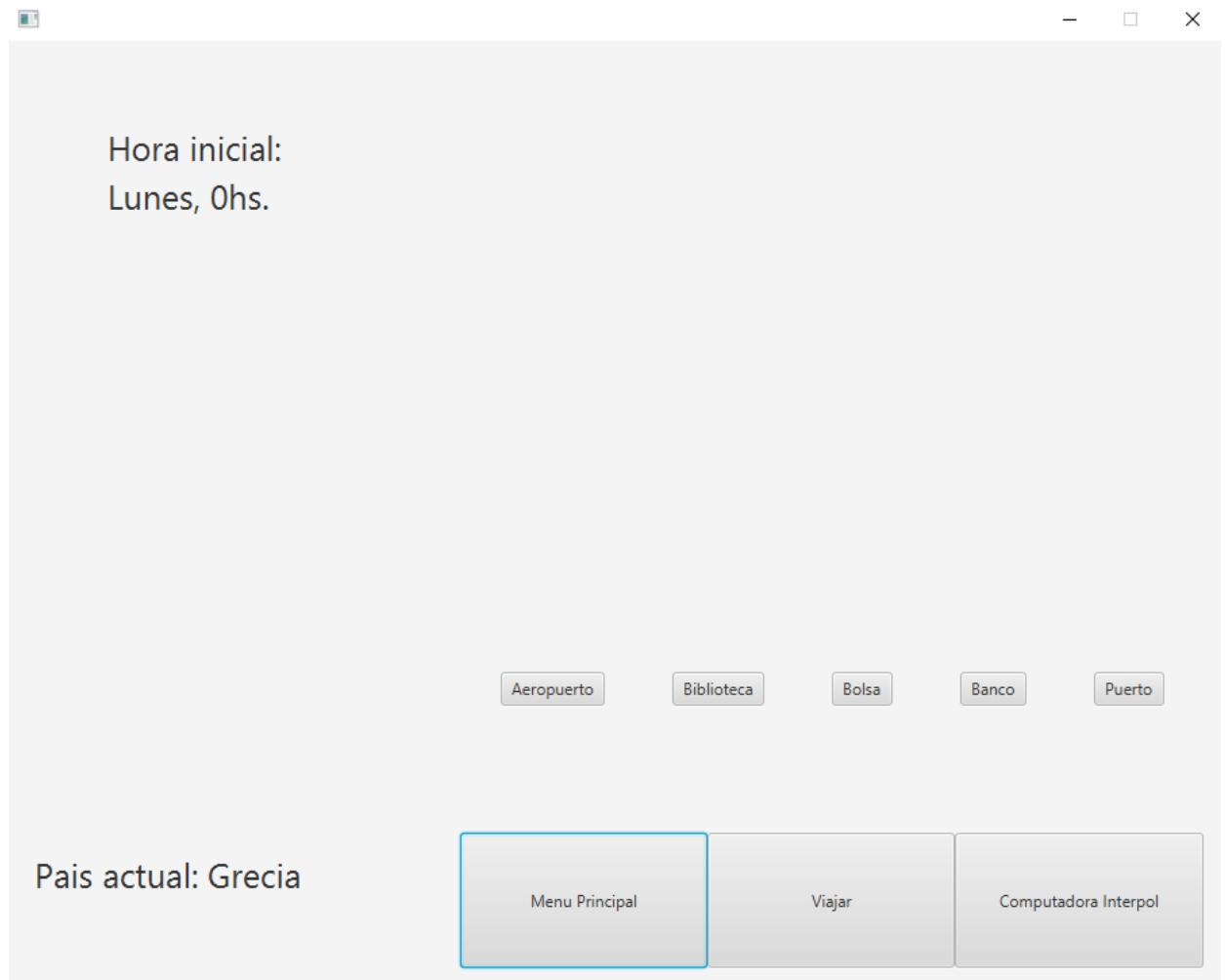


Figura 29: Diagrama de clase Características

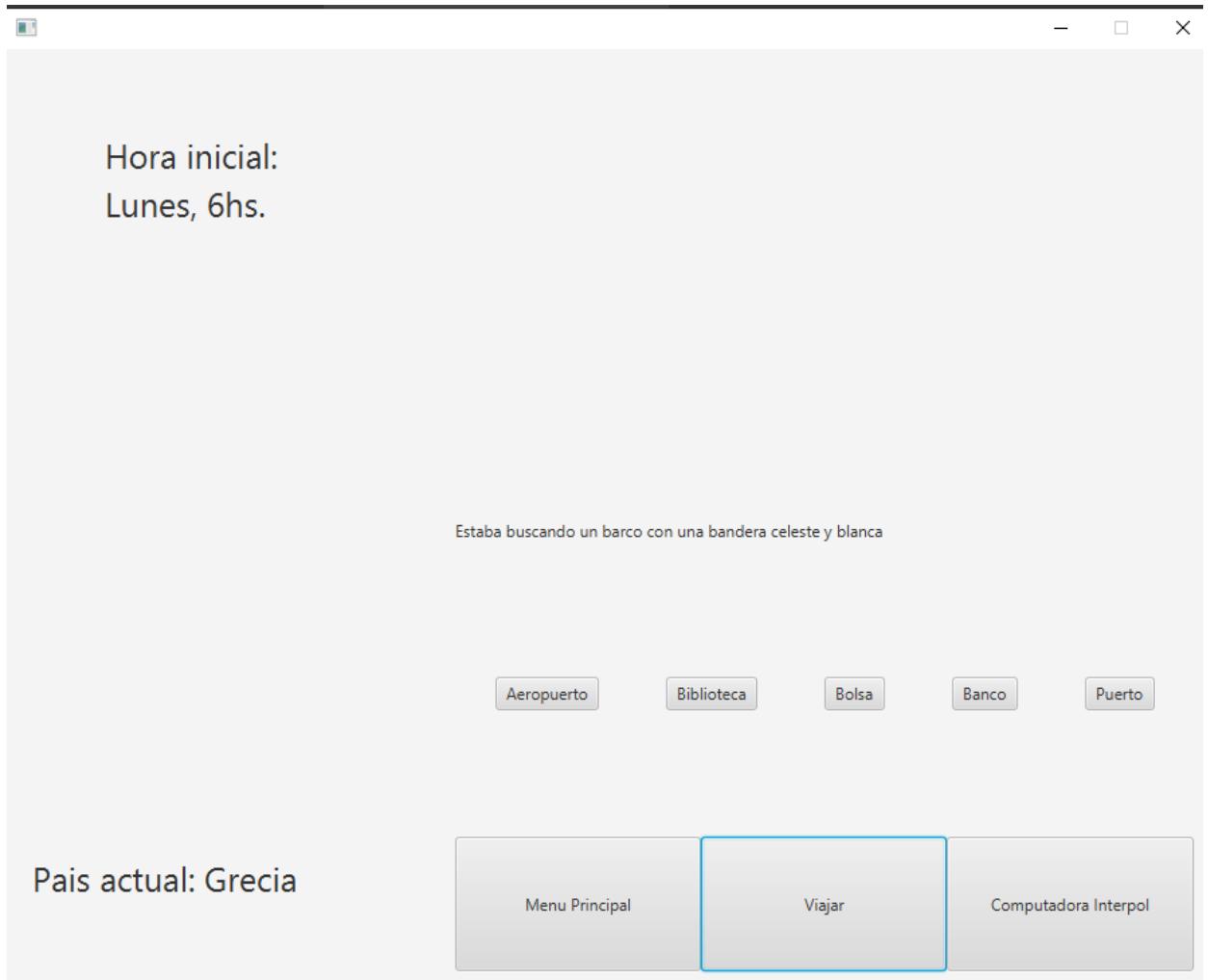


Figura 30: Diagrama de clase Características

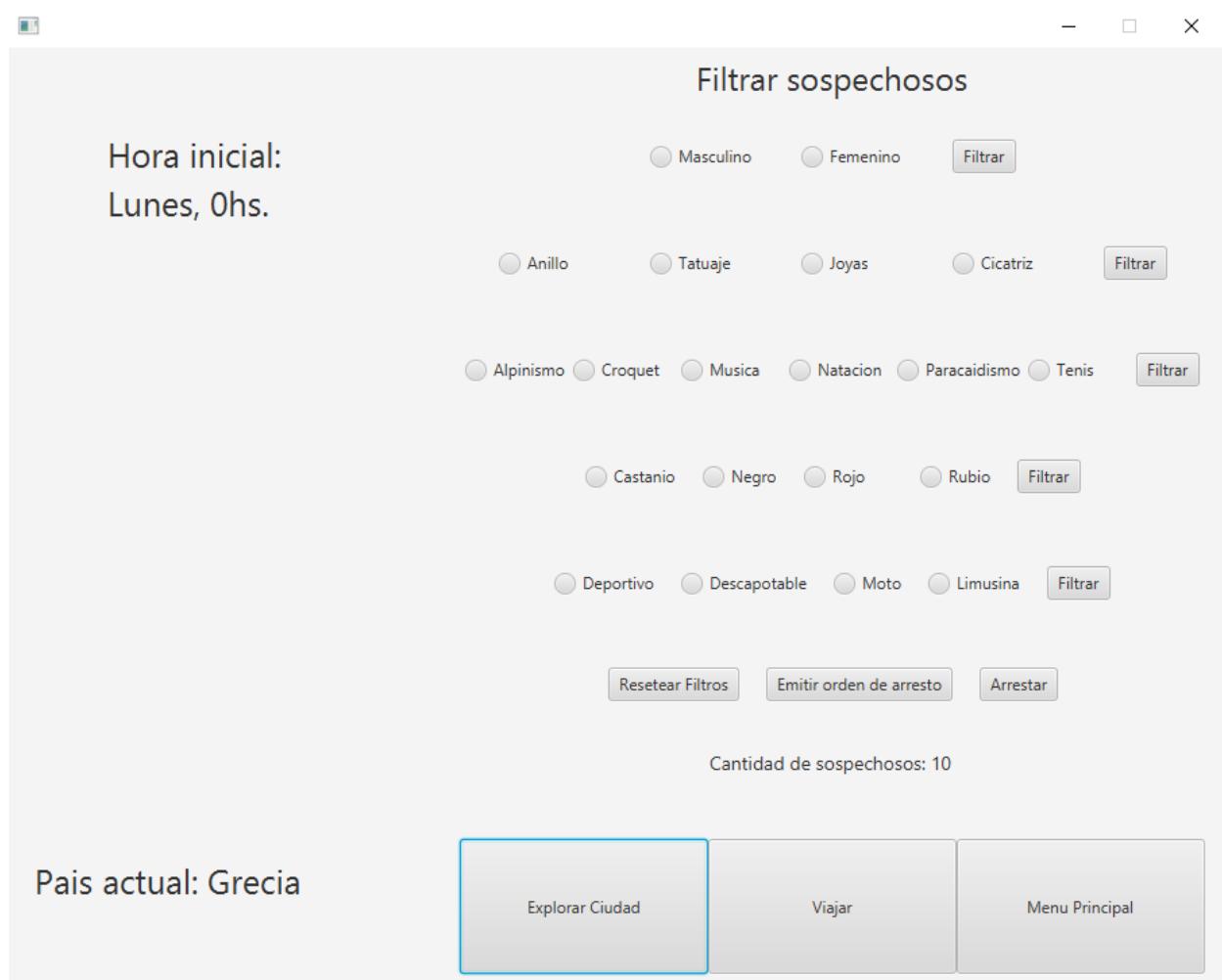


Figura 31: Diagrama de clase Características

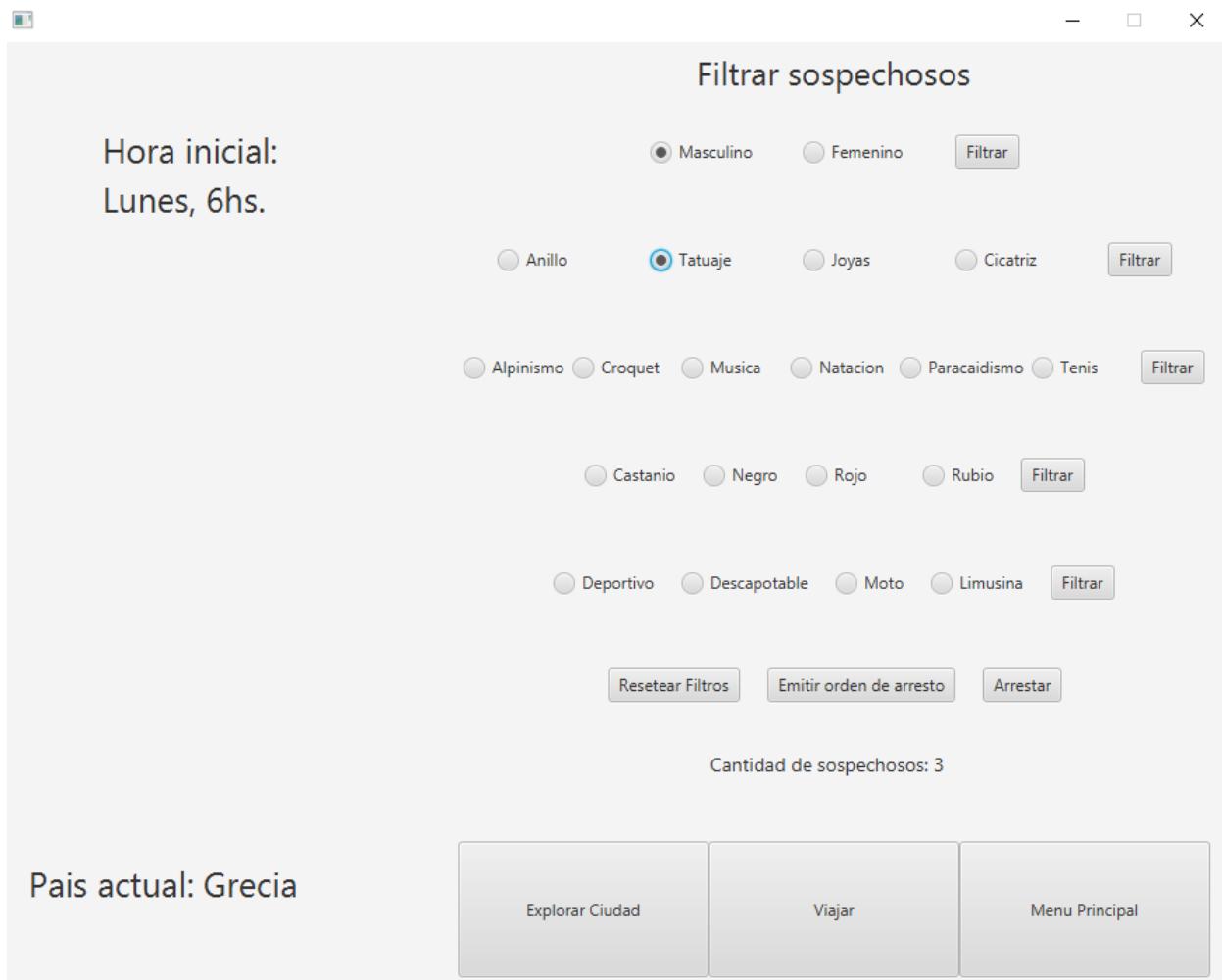


Figura 32: Diagrama de clase Características

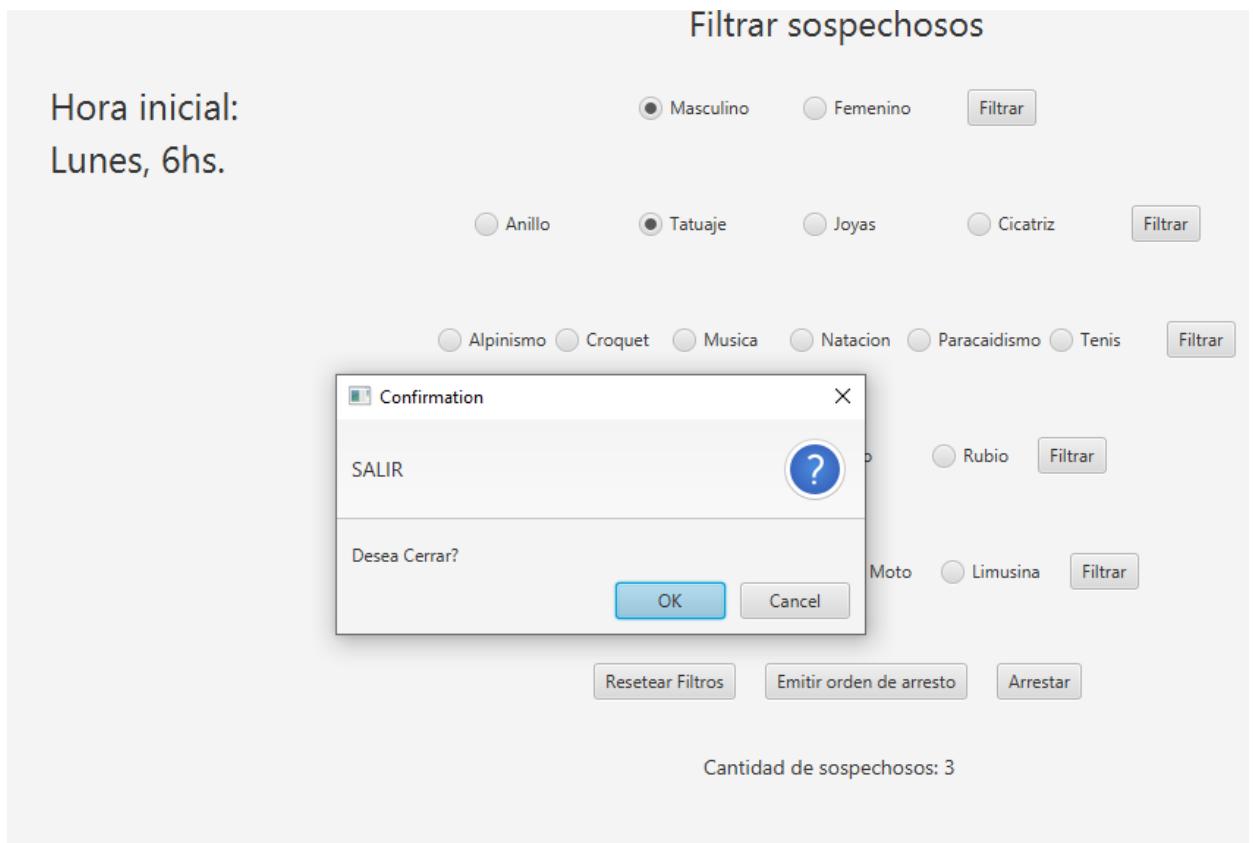


Figura 33: Diagrama de clase Características

12. Semana 4: Entrega final

En la ultima semana de entrega sobre el programa, presentaremos todos los diagramas finales requeridos: Diagramas de estado y Diagramas de paquetes.

También justificaremos los patrones aplicados y se presentaran imágenes del producto final, incluyendo interfaz gráfica.

12.1. Patrones

Para el desarrollo del software utilizamos los siguiente patrones:

- Patrón Factory
- Patrón Singleton

12.2. Patrón Factory

Hicimos uso del patrón creacional factory, en principal debido a que queríamos que al iniciar el juego, la clase partida instanciara todo lo necesario para el desarrollo de una partida. Esto implicaba que en partida se alojara toda la lógica sobre la creación e instancia de todo los elementos propios del juego para la partida, por ej: criminal, países etc..

Utilizando de esta forma teñíamos fragmentos grandes de código suelto, que no tenían nada que

ver con la propia partida, como por ejemplo el proceso de crear los países para el cual se lee un archivo y se ejecuta un pequeño algoritmo. Este tipo de situaciones se repetía para cada clase que necesitábamos no solo entorpeciendo la vista de la clase Partida, sino también penetraba la capa de abstracción.

Por lo tanto con el patrón factory no solo logramos un código mas limpio sino que también nos permitió mantener una abstracción sobre como y de que manera se instancia todos los objetos relevantes a partida, permitiendo a futuro una ampliación que no afecte a ninguna otra parte del software ya que si se desea agregar mas objetos a la partida o se produce algún cambio en el juego para el cual ahora la partida tenga por ejemplo un investigador privado que compita con el jugador en atrapar al ladrón y este a su vez base su complejidad en el rango del policía, simplemente se agregaría un FactoryCompetidor y no habría repercusión en ninguna otra parte del software.

12.3. Patrón: Singleton

Si bien habíamos logrado evitar el uso de singleton para el caso del reloj en relación al tiempo, terminamos encontrándonos en una situación similar en donde consideramos que era una opción fiable y lo que nos permitía solucionar mas problemas que molestias. El caso se presentó con la clase partida, que al igual que la clase tablero se relaciona con todos los objetos en un juego de mesa, por ejemplo las piezas, el dado etc.. Este fue el caso con Partida, ya que todos los elementos necesitaban interactuar con la partida para poder establecer un sentido al juego y permitir que el mismo funcione, evitando algoritmos o programación forzada que solo entorpecía la libertad de nuestro software. Por lo tanto aplicando el patrón Singleton podemos lograr que todas las clases que tengan que interactuar con la partida, puedan hacerlo por medio de este patrón, siendo una implementación correcta del mismo.

12.4. Excepciones

En el apartado se explican las excepciones creadas y su uso.

- NoExiste: Excepción creada y utilizada para cuando se trata de buscar un objeto que no existe. Es decir, si se trata de buscar un país que no existe ej: Bonta—>como no existe —>arroja la excepción "No Existe Error".

12.5. Diagramas de Paquete

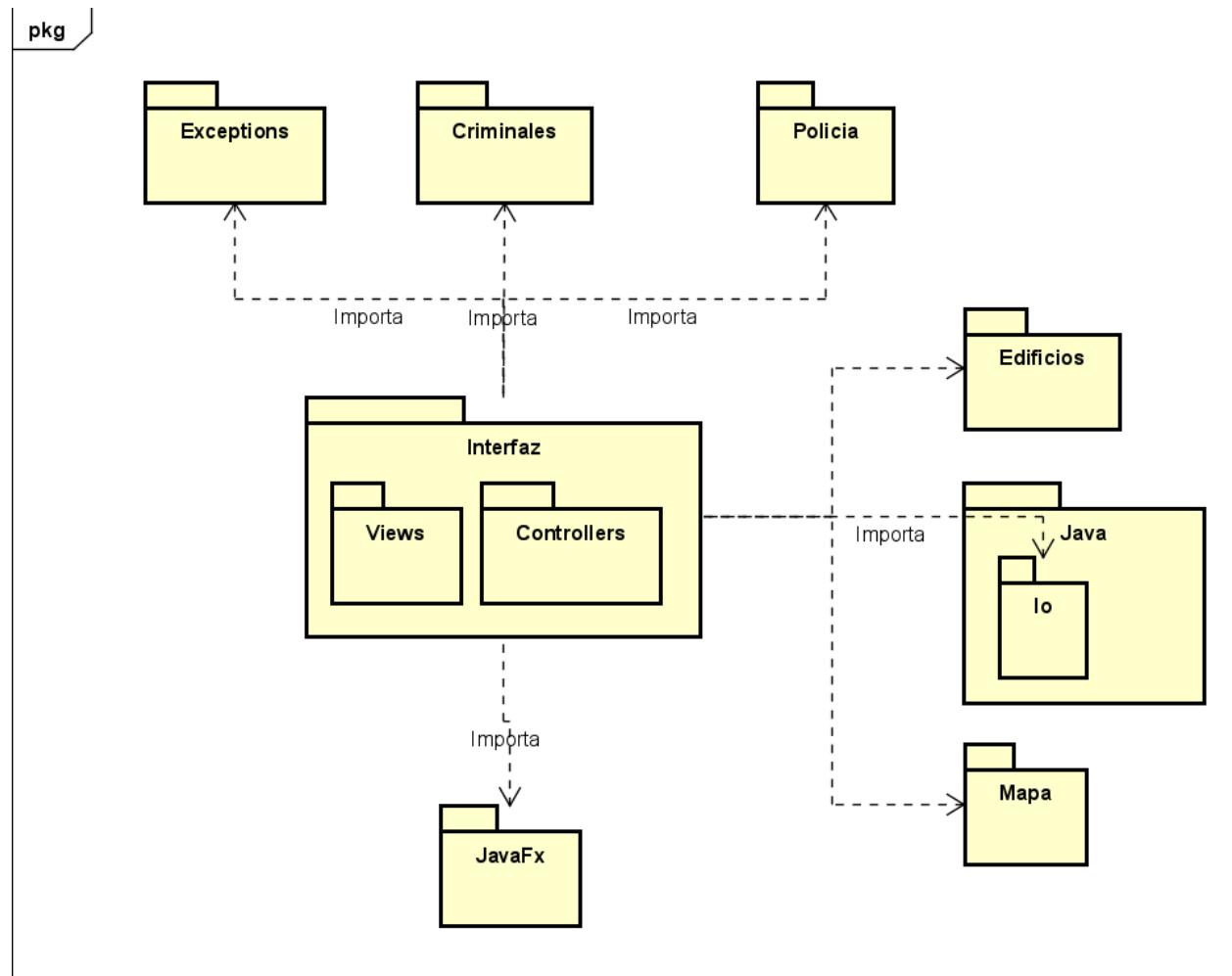


Figura 34: Diagrama de Paquete de la interfaz

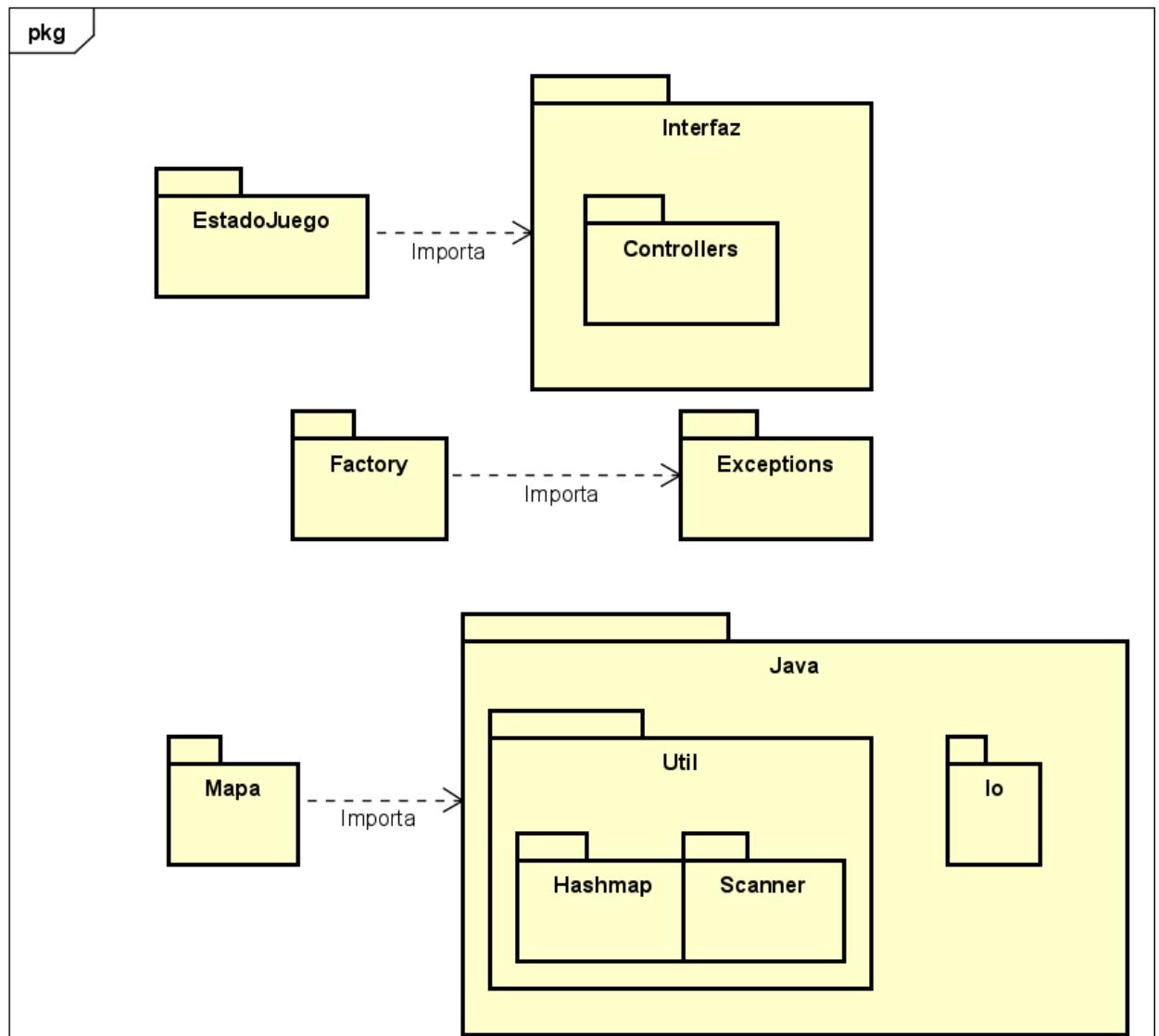


Figura 35: Diagrama de paquete del estado juego

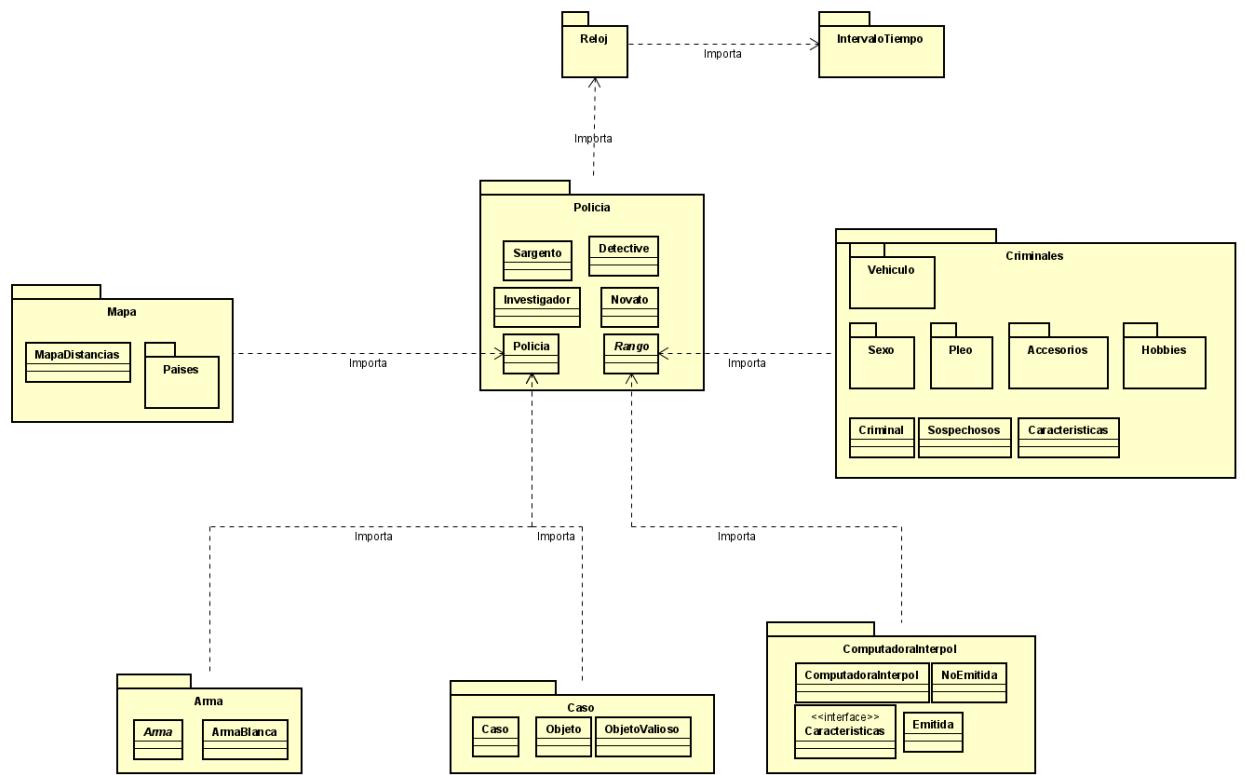


Figura 36: Diagrama de paquete de policía

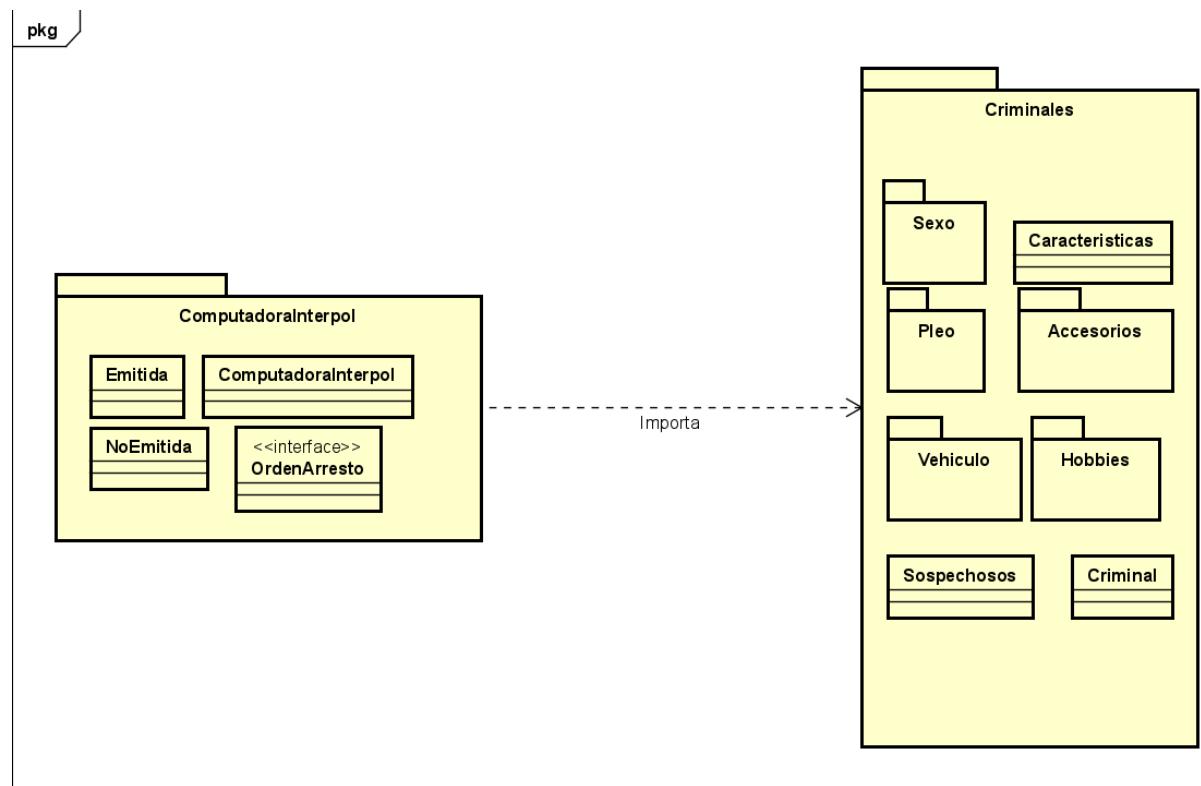


Figura 37: Diagrama de paquete de computadora Interpol

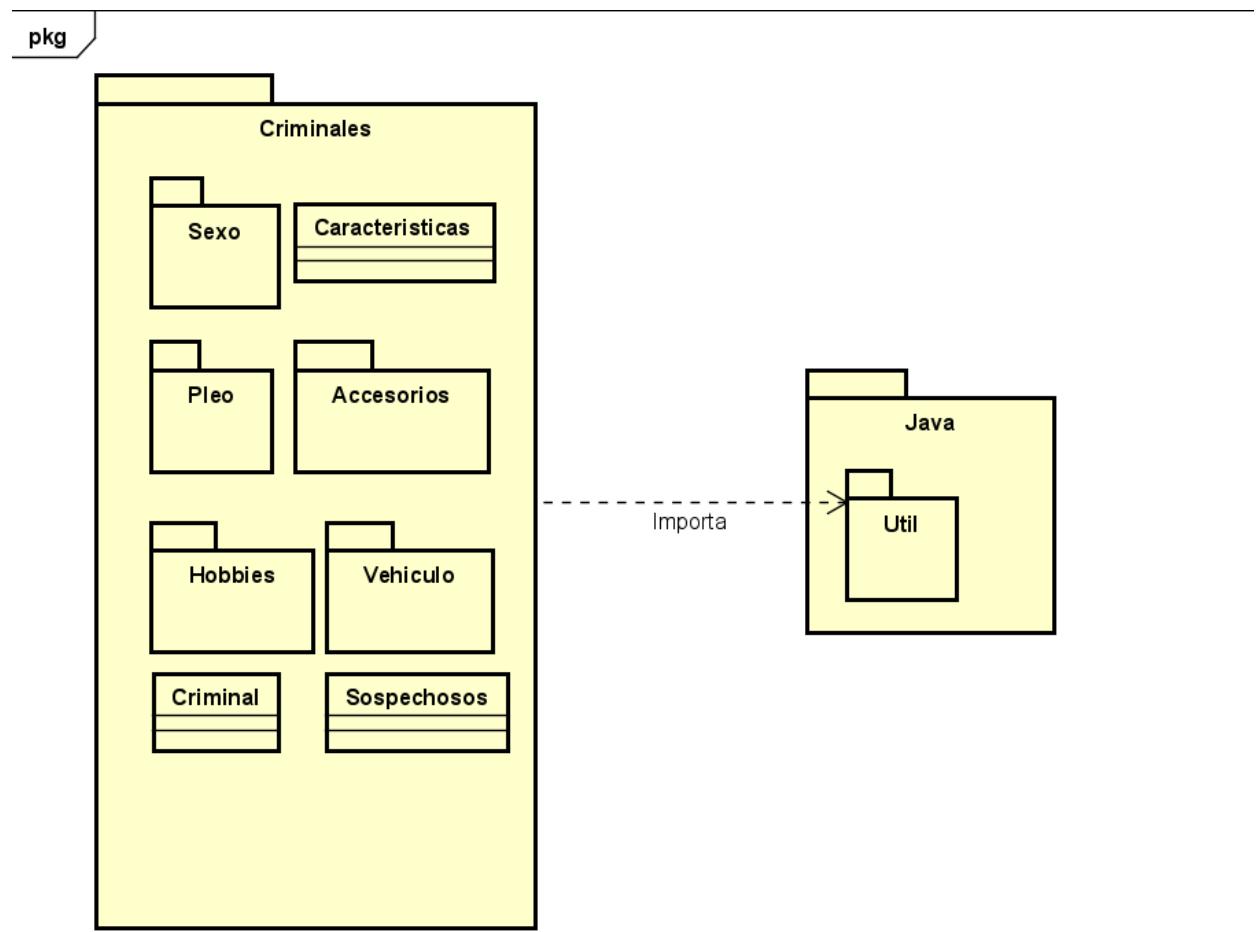


Figura 38: Diagrama de paquete de criminales

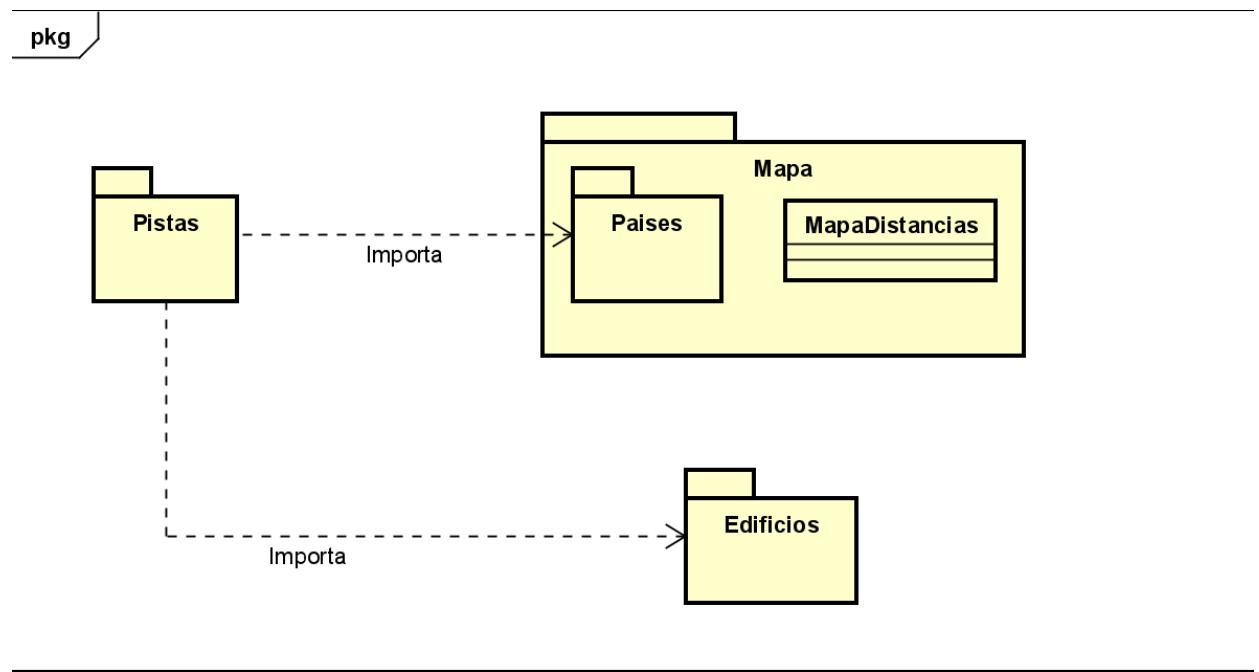


Figura 39: Diagrama de paquete de pistas

12.6. Diagramas de Estado

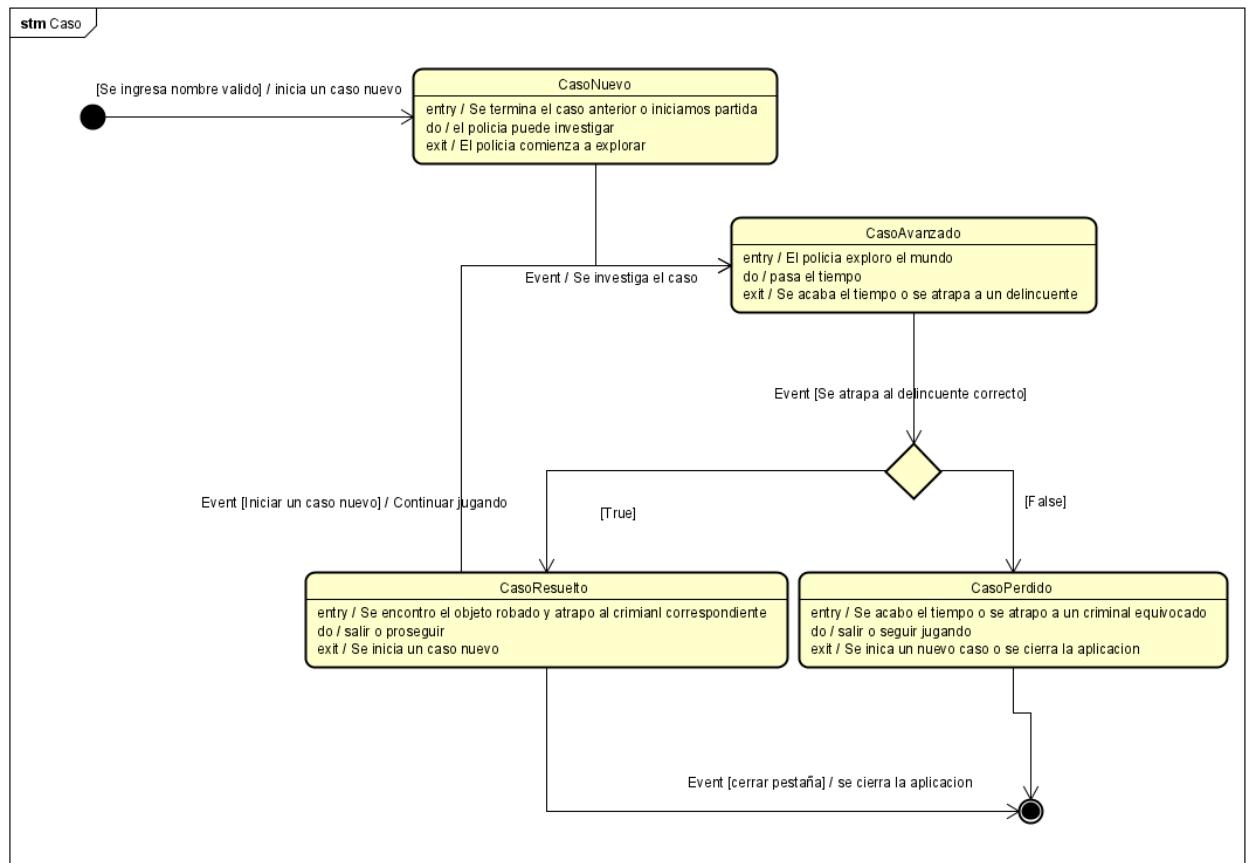


Figura 40: Diagrama de estado de caso

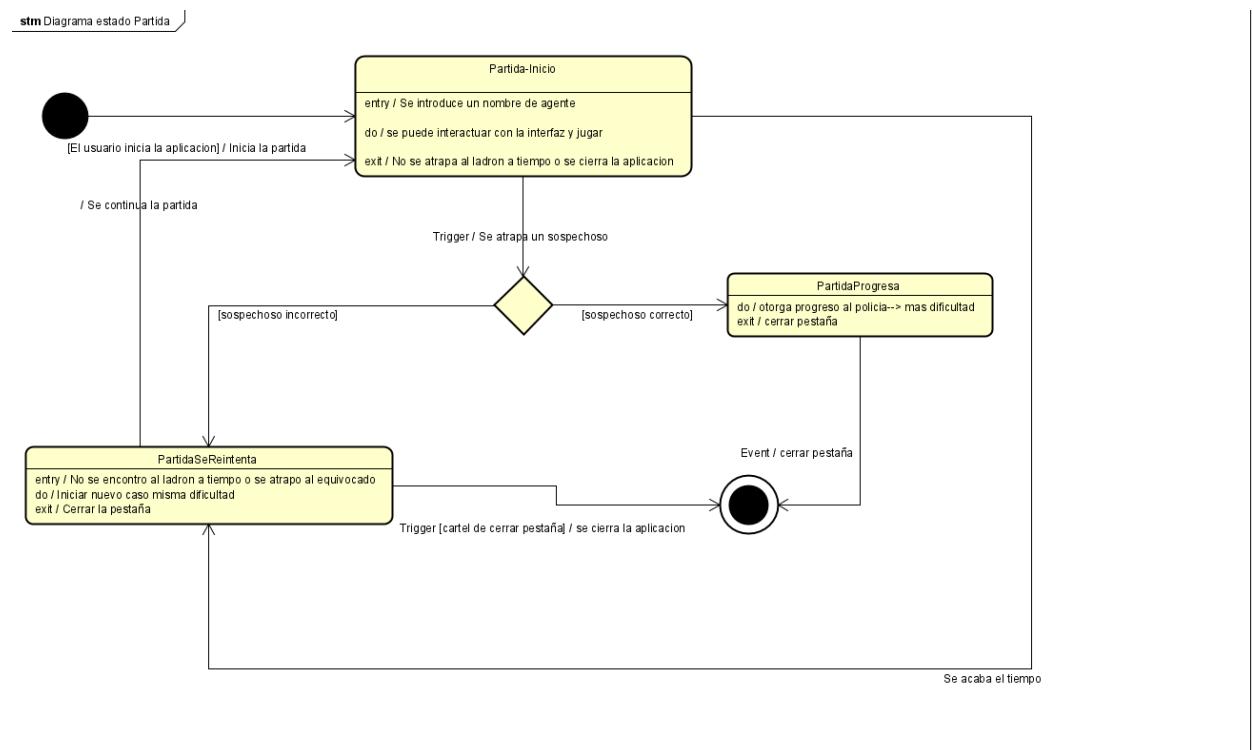


Figura 41: Diagrama de estado de partida

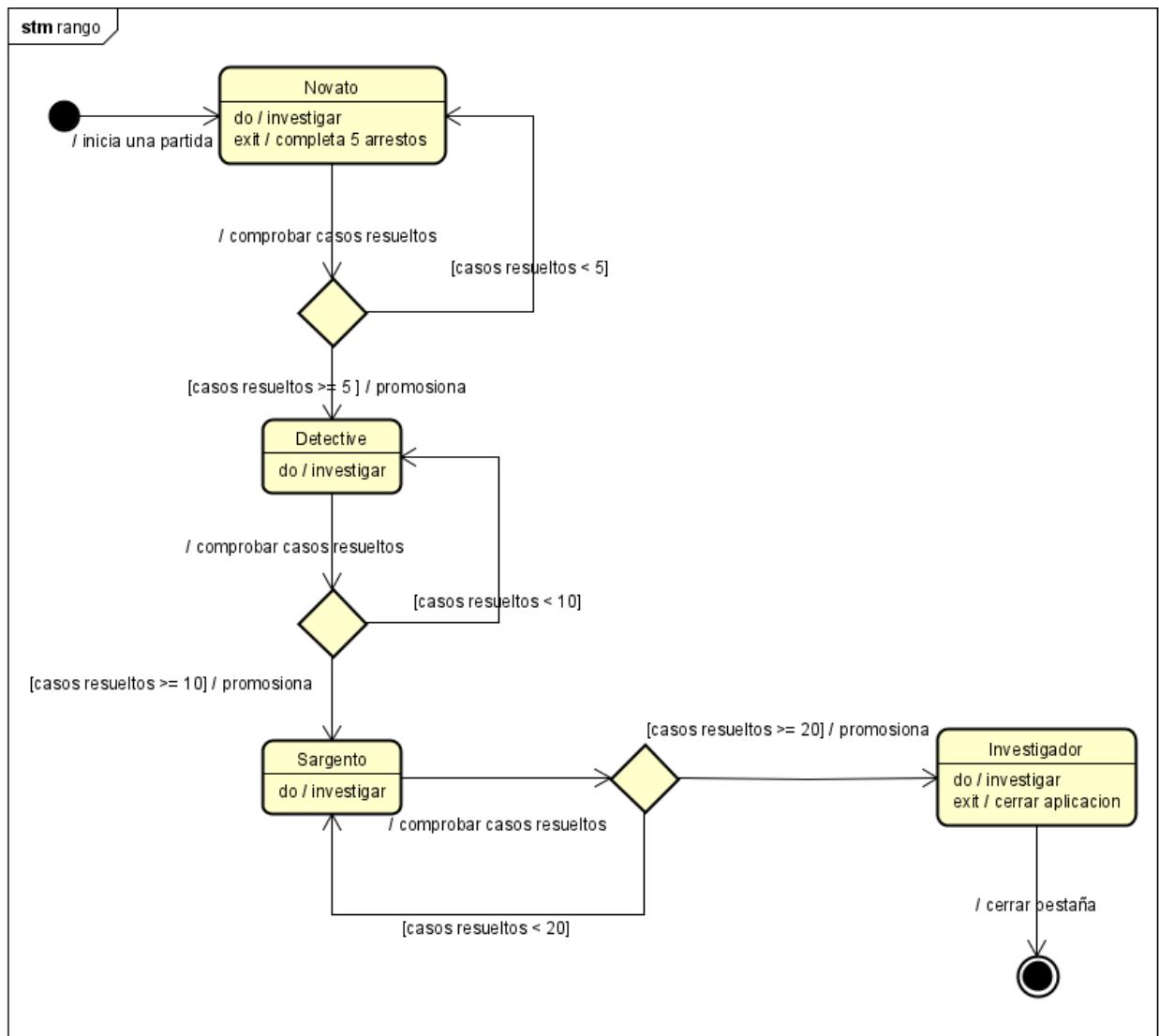


Figura 42: Diagrama de estado de policía

12.7. Interfaz

Se adjuntan las imágenes de la interfaz definitiva



Figura 43: Pantalla de inicio

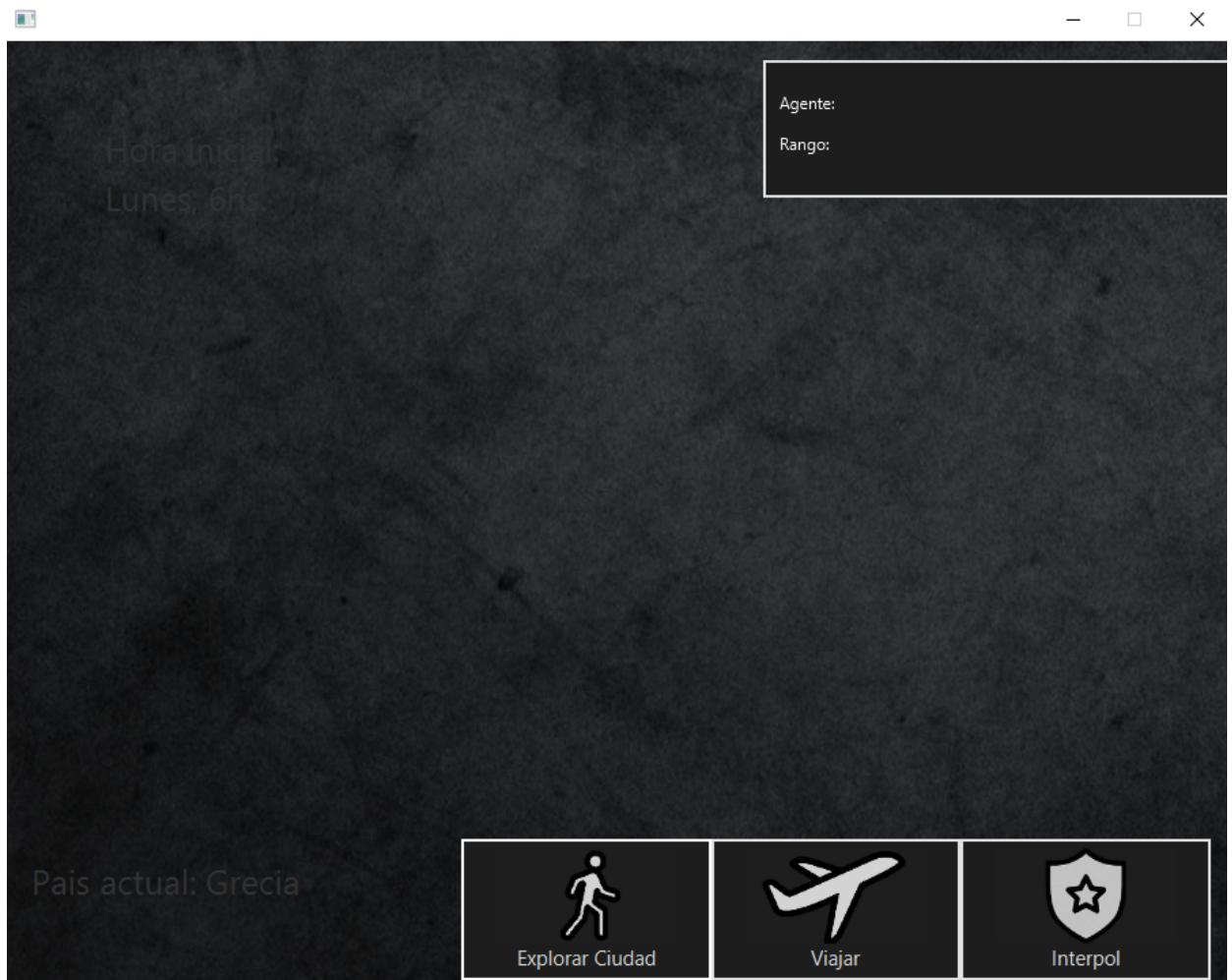


Figura 44: Menú principal

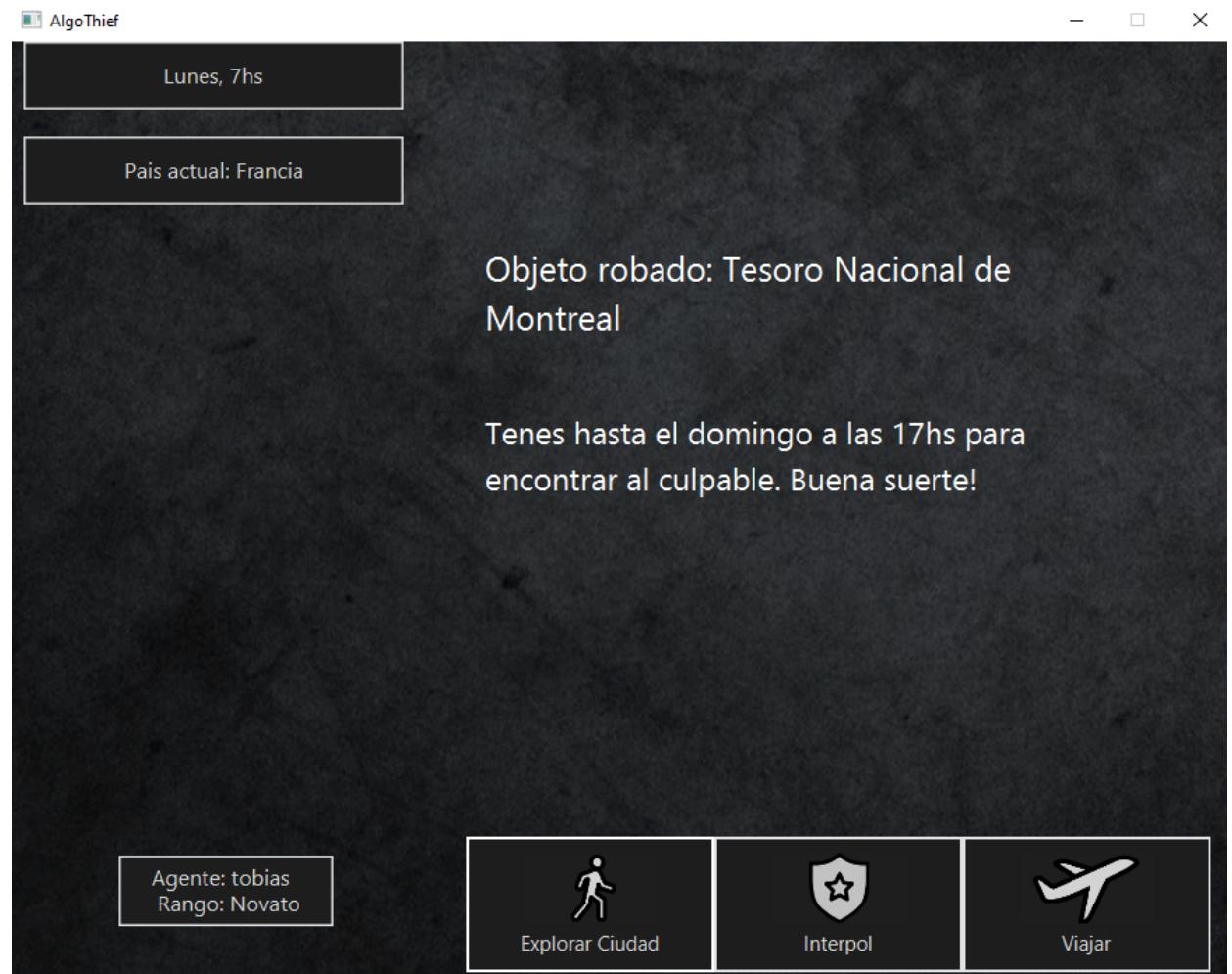


Figura 45: pantalla de caso

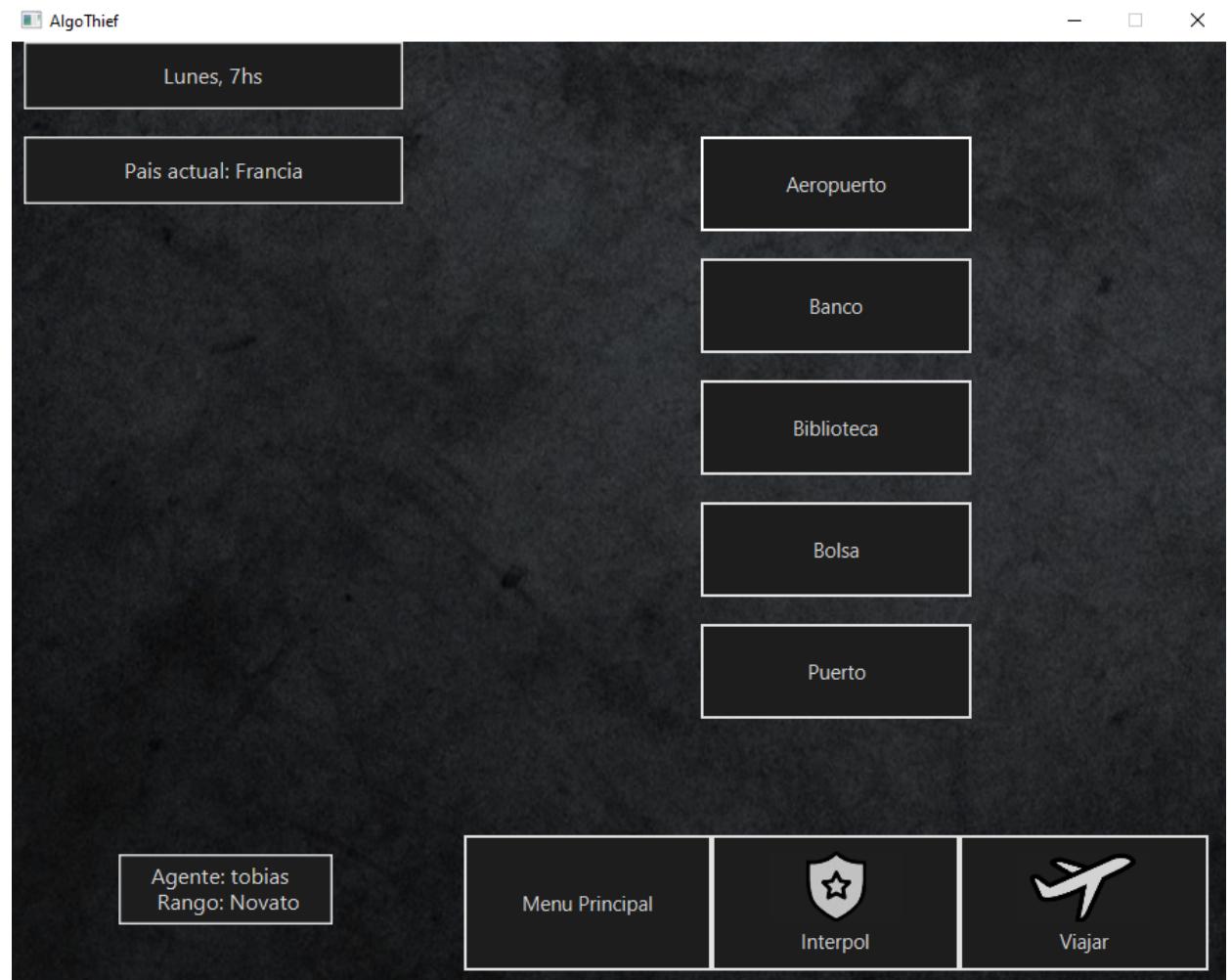


Figura 46: explorar la ciudad

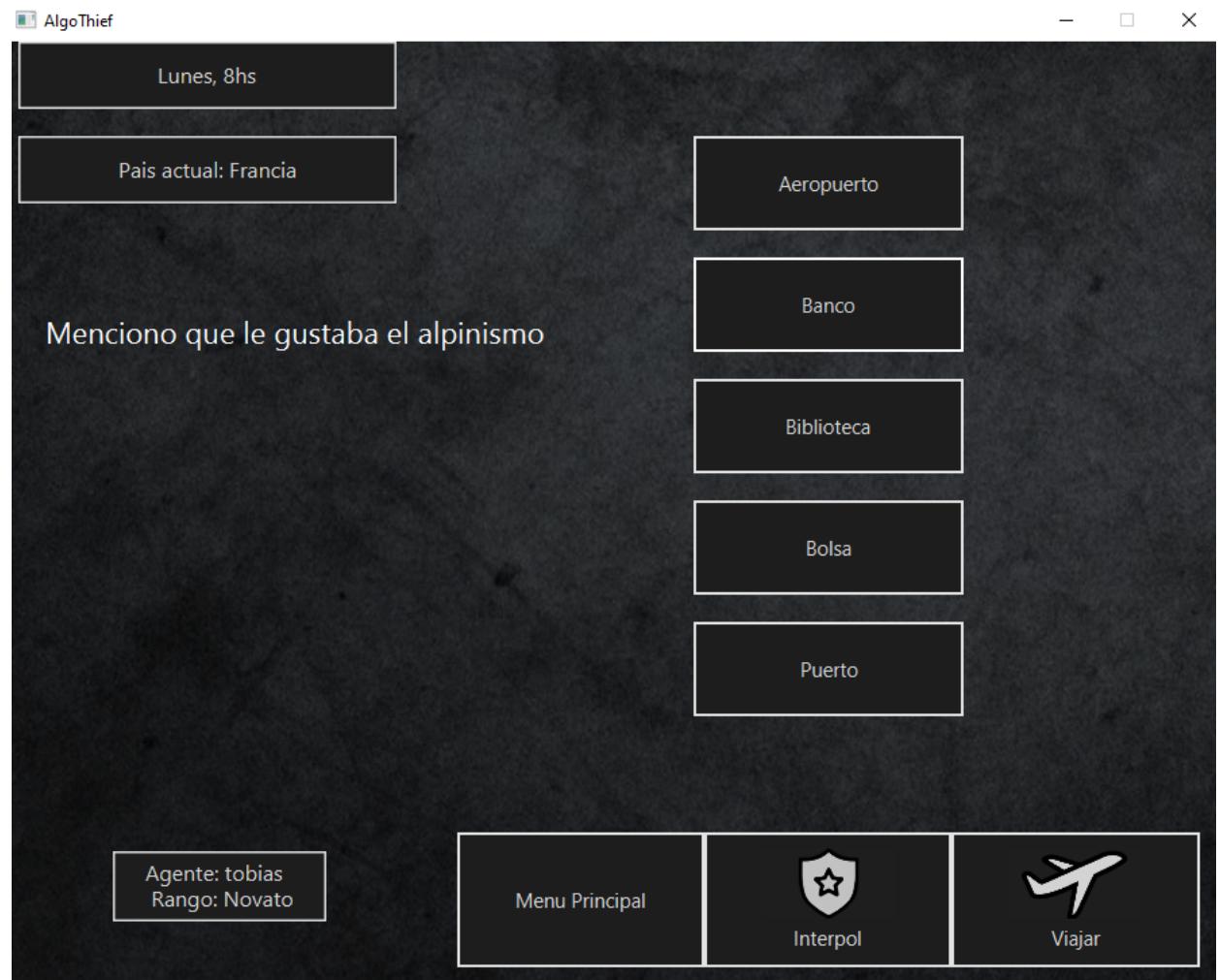


Figura 47: Pista luego de explorar sitio

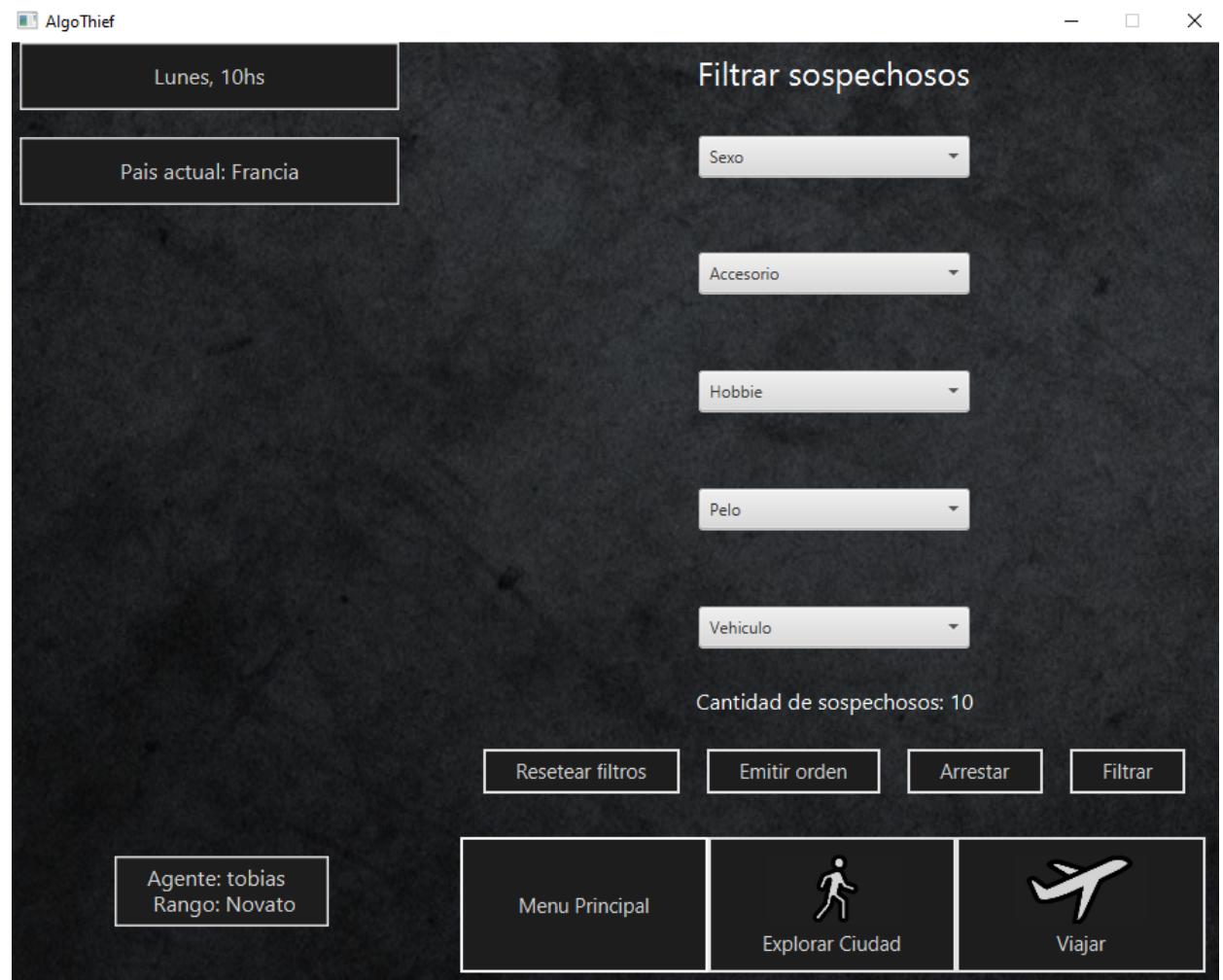


Figura 48: Computadora de interpol

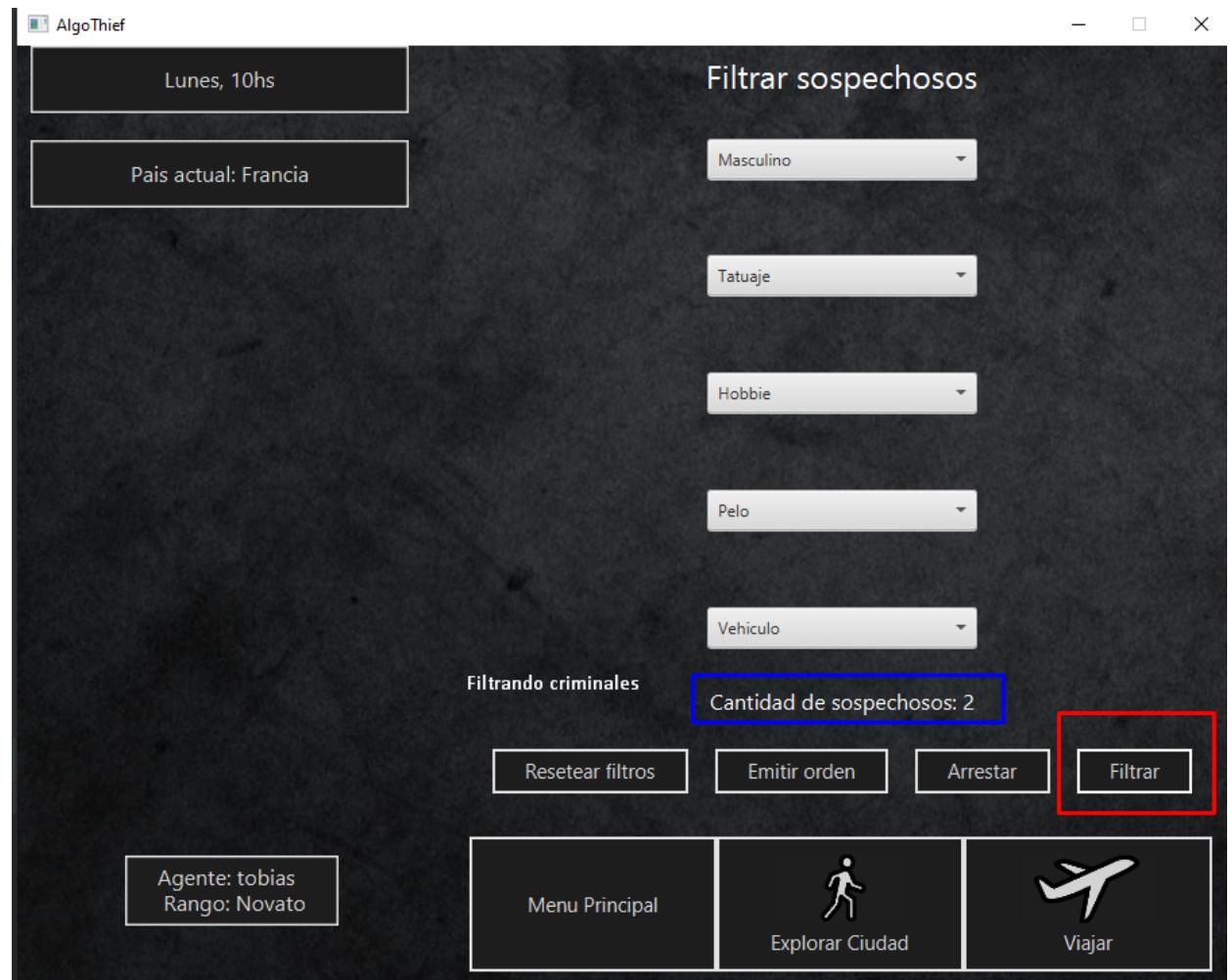


Figura 49: Filtrando criminal

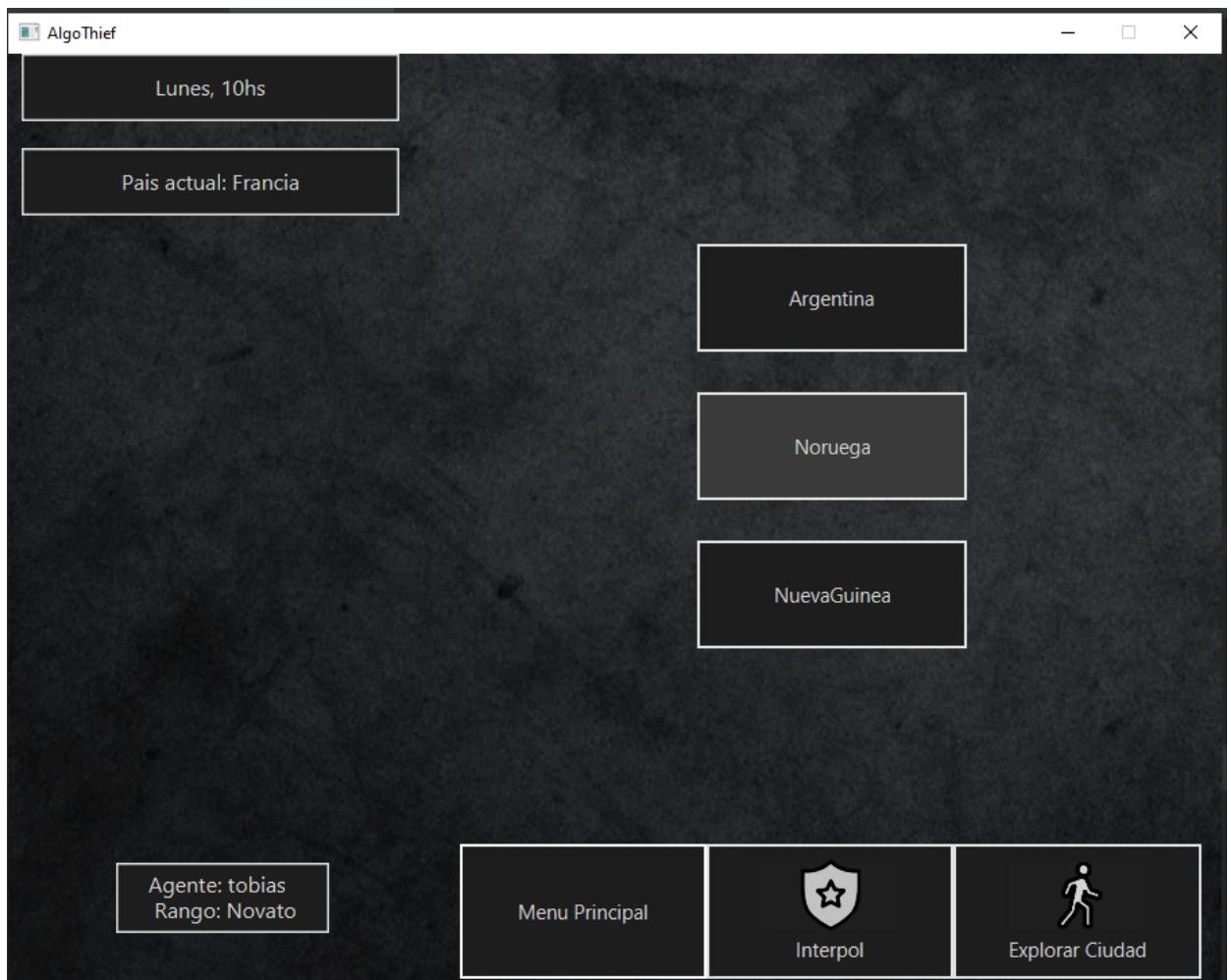


Figura 50: pantalla para viajar a otro país

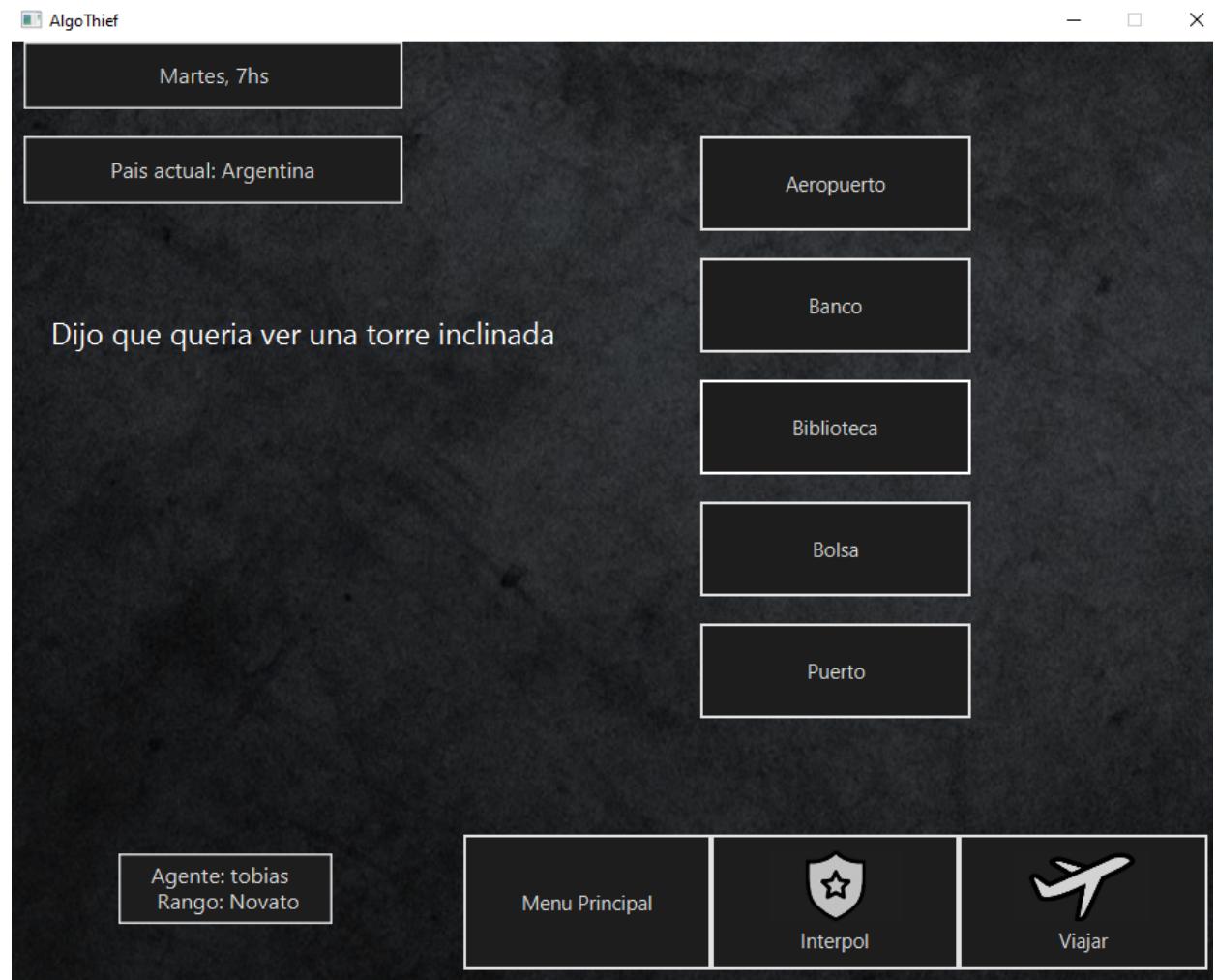


Figura 51: Nueva ciudad

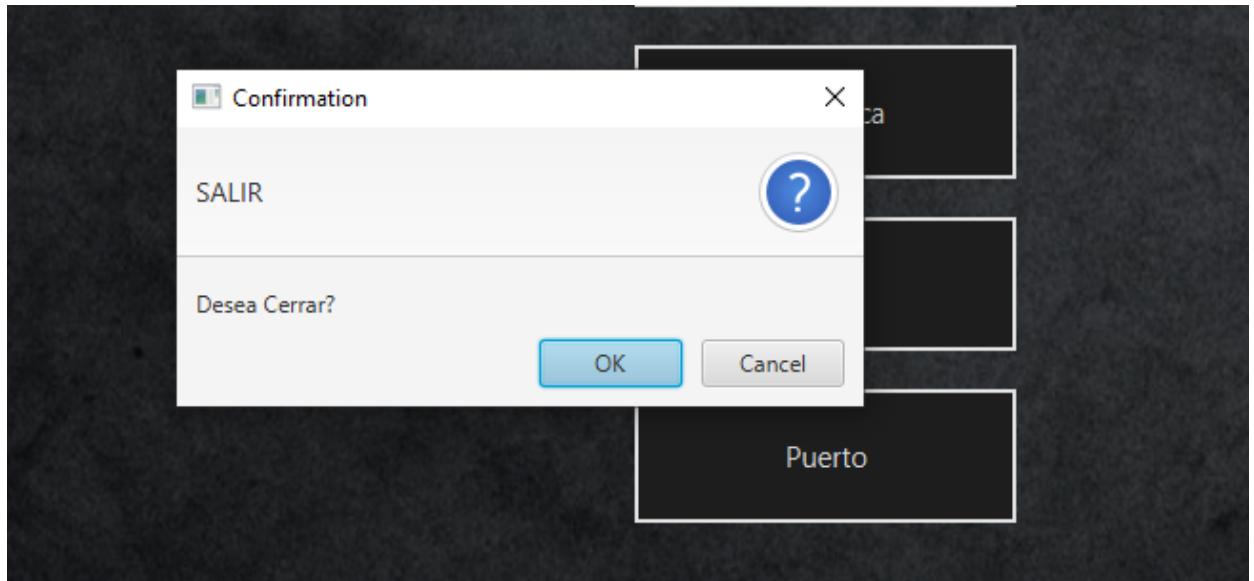


Figura 52: Pantalla de salida de la aplicación