FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2

[7507/9502] Algoritmos y Programación III
2 de diciembre, 2021

Integrantes

Alumno	Padrón	Email
Rizzo Ehrenbock, Gonzalo Daniel	106475	grizzoe@fi.uba.ar
Dieguez, Jose Manuel	106146	mdieguez@fi.uba.ar
Busto, Fabrizio Alejandro	106856	fbusto@fi.uba.ar
Vetrano, Ignacio Ezequiel	106129	ivetrano@fi.uba.ar
Testa, Santiago Agustin	106689	stesta@fi.uba.ar

Índice

1. Introducción

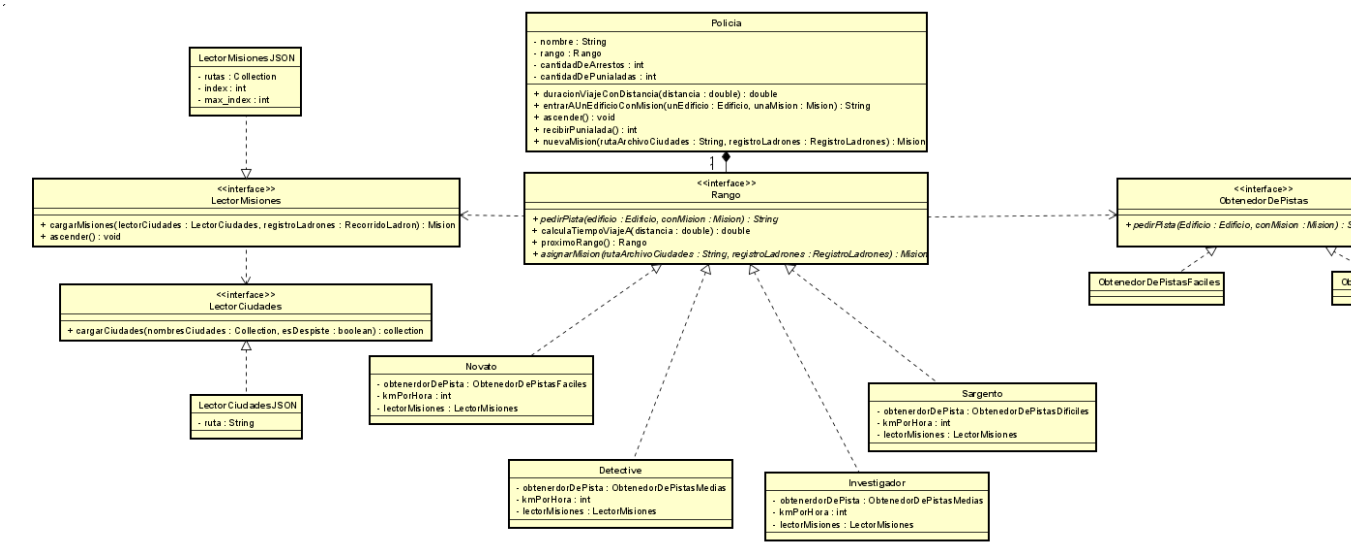
El presente informe reune la documentación de la solución del segundo trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar un juego en donde se simulan persecuciones a ladrones utilizando los conceptos del paradigma de la orientación a objetos vistos hasta ahora en el curso. Basado en el juego Carmen De San Diego.

2. Supuestos

A continuación se describen los supuestos a partir de los cules se desarroll el proyecto:

- Todas las ciudades deben tener un aeropeurto/puerto, banco/bolsa y biblioteca.
- El recorrido del ladron no puede atravesar dos veces por la misma ciudad.
- El rango de un policia no podra ser degradado a pesar de fallar en la resolución de uno o más casos.
- El juego finaliza (y el jugador gana) al terminar todas las misiones de todos los rangos.
- A lo largo de una partida, puede que un mismo ladron robe más de una vez.
- A lo largo de una partida, puede que un mismo objeto sea robado más de una vez.
- Una vez que se abandono una ciudad, no es posible regresar a la misma en una misma mision.
- Todas las misiones empiezan un Lunes, sin importar cuando termine la anterior.
- Existe la probabilidad de que al entrar a un edifico, ademas de la pista del lugar al que escapó, se obtenga una caracteristica del sospechoso.
- Al entrar a un edificio existe una probabilidad de ser apuñalado o disparado por un complice del sospechoso.
- Entrar a un edificio de una ciudad por donde no haya pasado el ladron, no nos dara información alguna acerca de donde se fue el sospechoso.
- Todas las noches a las 00:00hs el policia deberá dormir 8hs. Si una accion realizada por el policía sobrepasa las 00:00, el mismo dormira al momento de finalizar esta accion.

3. Diagramas de clase

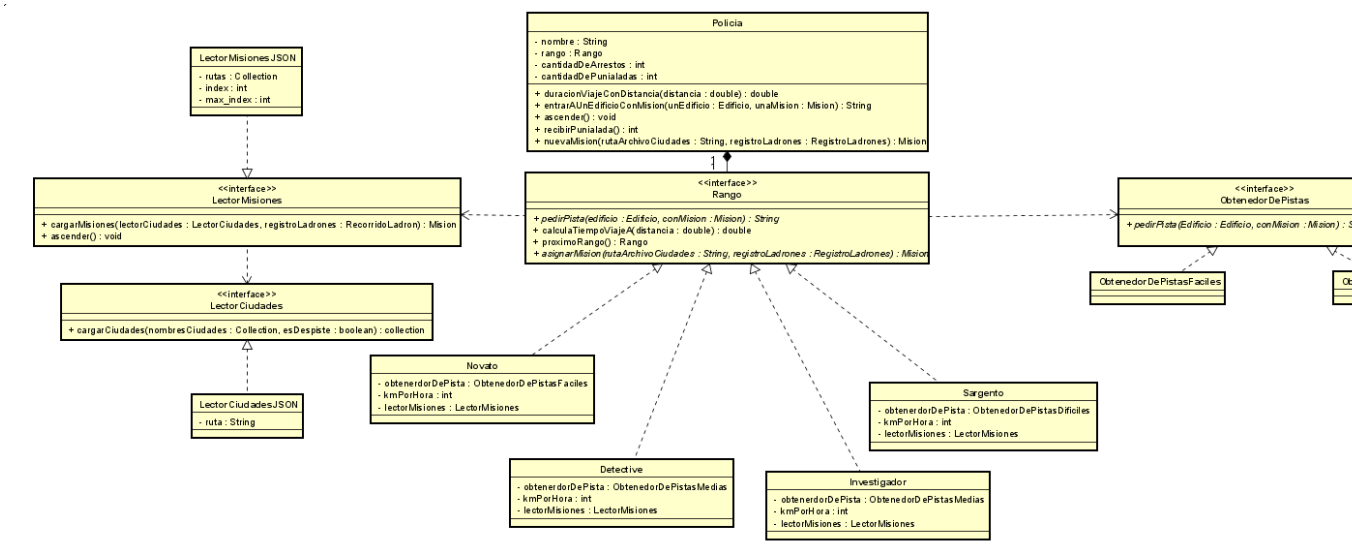


4. Detalles de implementación

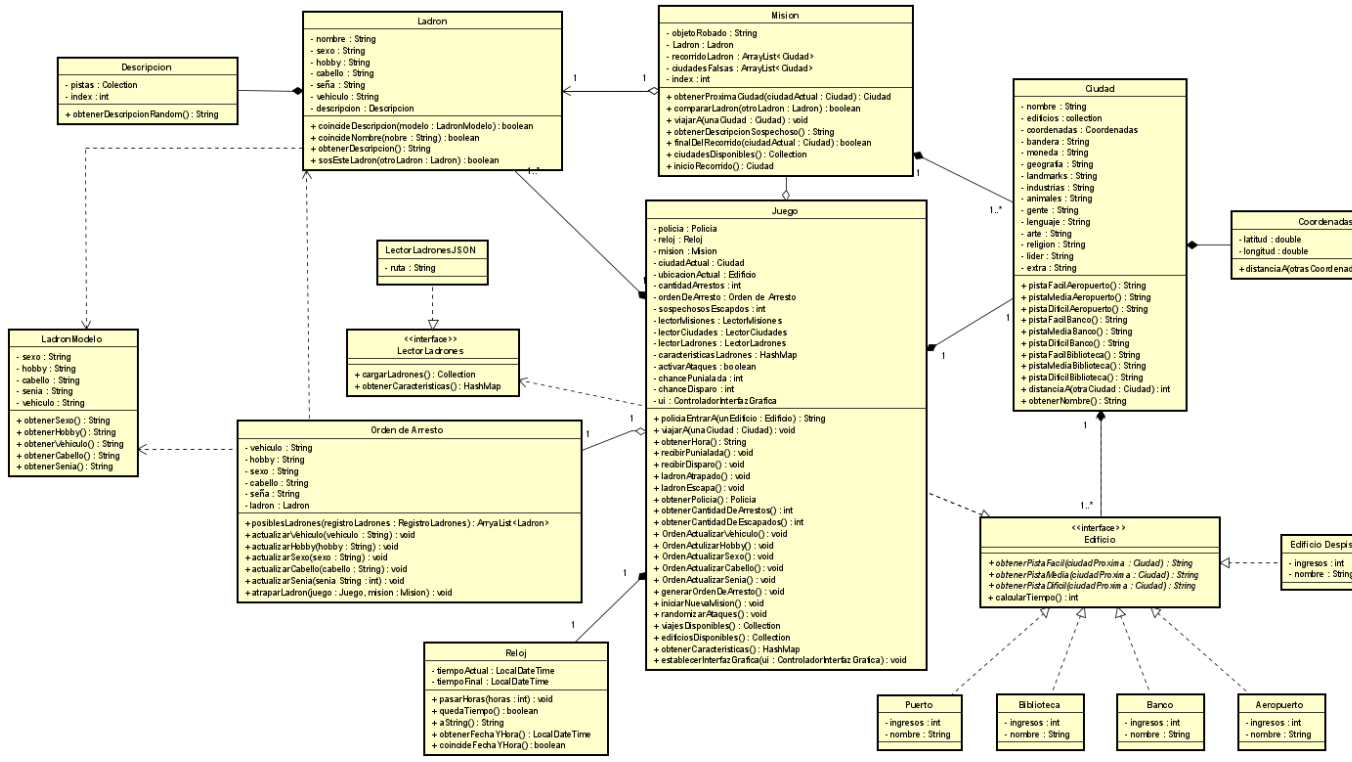
4.1. Obtención de pistas de la proxima ciudad (ObtenerdorDePistas)

Para la obtencion de las pistas de la proxima ciudad se dependia de 3 cosas:

- El rango del policia.
- El edificio al que entro el policia.
- La ciudad siguiente (solo en caso de que es sospechoso haya estado en la ciudad actual).

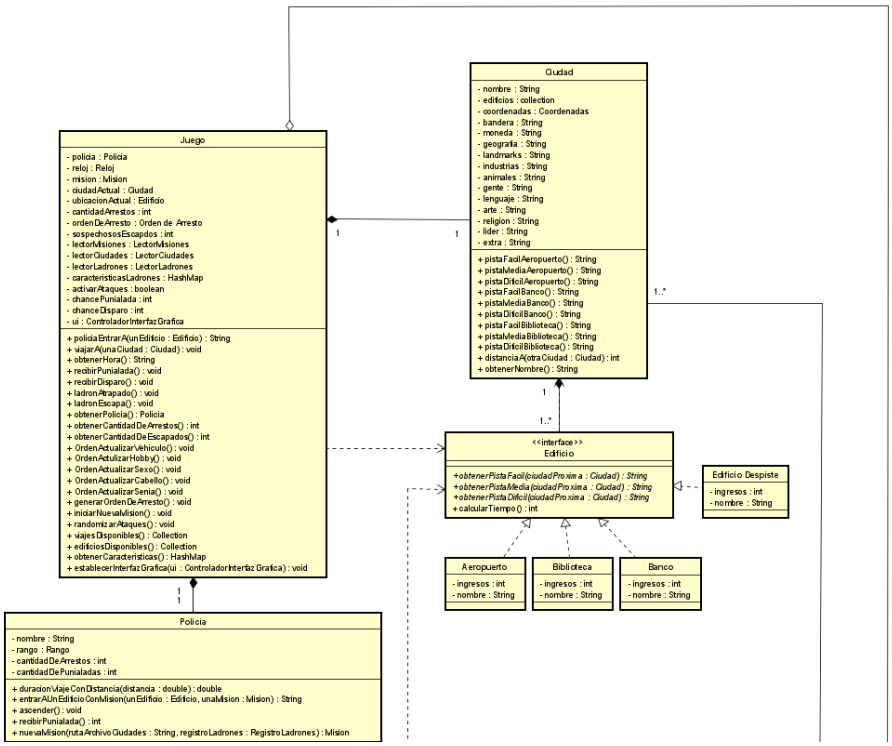


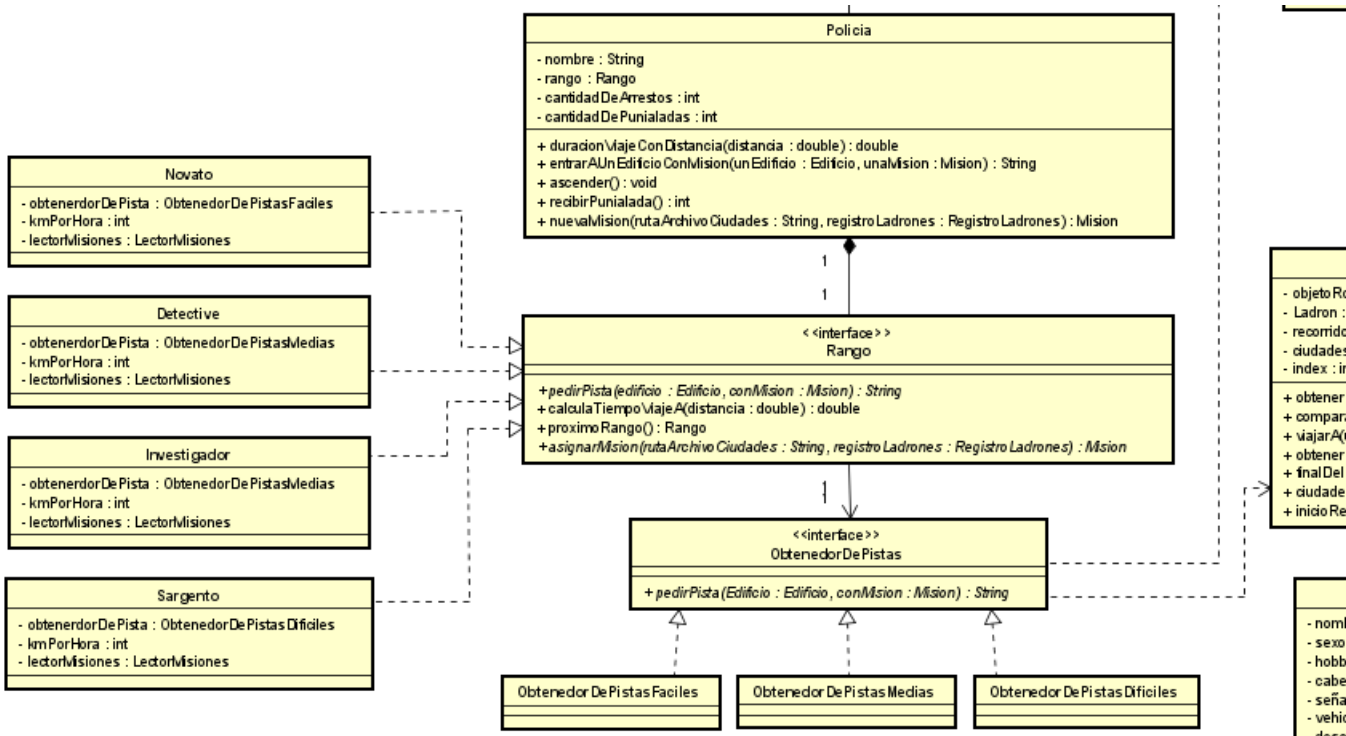
Por lo que para la obtencion de pistas se implementó un sistema de **Double Dispatch** en donde el policia tiene un rango, cada tipo de rango (Novato, Detective, Investigador, Sargento) tiene un ObtenedorDePistas diferente (ObtenedorDePistasFaciles, ObtenedorDePistasMedias, ObtenedorDePistasDificiles), que le envian un mensaje diferente al edificio actual (obtenerPistaFacil, obtenerPistaMedia, obtenerPistaDificil) el cual le envia un mensaje diferente a la proxima ciudad (PistaFacilAeropuerto, PistaMediaBiblioteca, etc).



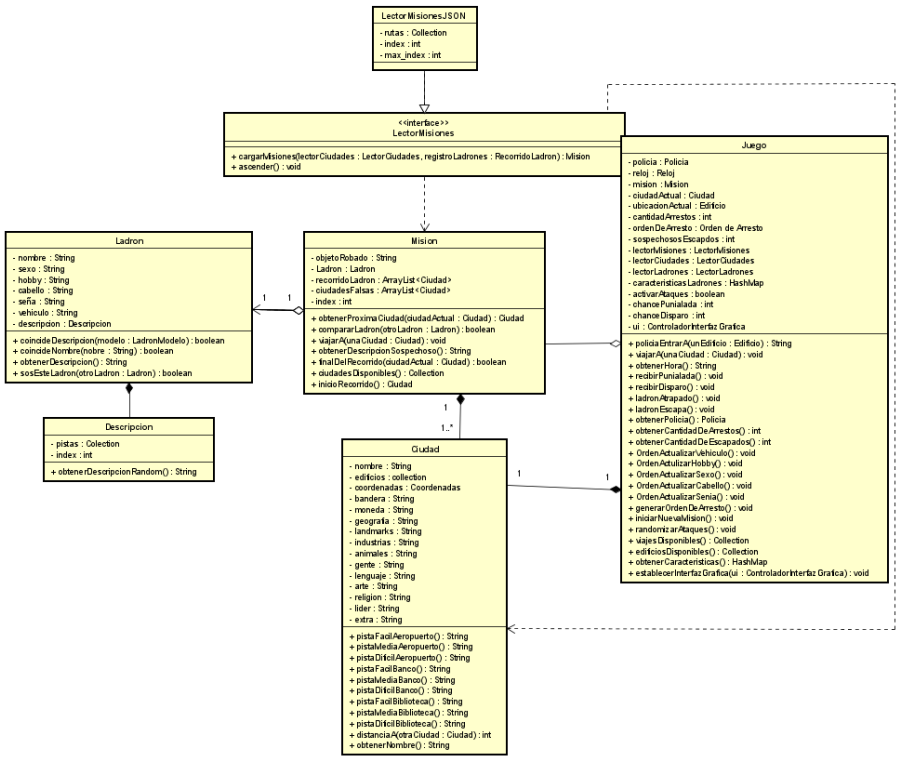
4.2. Obtención de pistas de la proxima ciudad desde una ciudad falsa (ObtenerdorDePistas)

Cuando estamos en una ciudad en la que no estuvo el ladron y entramos a un edificio no nos deberian dar pistas sobre la proxima ciudad, por lo que se utilizo **Polimorfismo** con edificios falsos de la siguiente manera: Cuando se instancia una ciudad falsa se instancia con edificios falsos (o de despiste), los cuales reciben los mismos mensajes que los edificios reales, pero cuya implementacion es diferente, en vez de dar un String con una pista cuando se la piden, devuelve un String en donde se comenta que el sospechoso no estuvo en ese lugar.

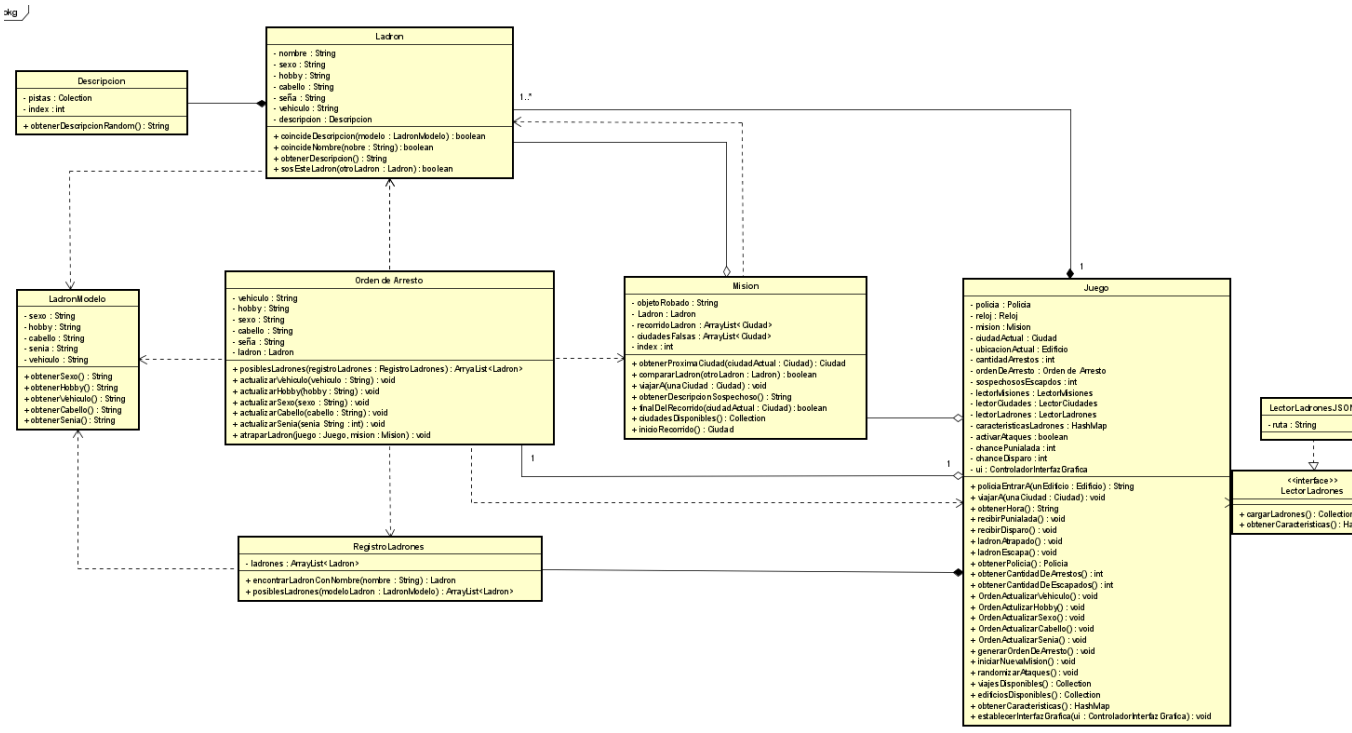



Policia
- nombre : String
- rango : Rango
- cantidadDeArrestos : int
- cantidadDePunialadas : int
+ duracionViajeConDistancia(distancia : double) : double
+ entrarAUnEdificioConMision(unEdificio : Edificio, unaMision : Mision) : String
+ ascender() : void
+ recibirPunialada() : int
+ nuevaMision(rutaArchivoCiudades : String, registroLadrones : RegistroLadrones) : Mision
1
1
Novato
- obtenerdorDePista : ObtenedorDePistasFaciles
- kmPorHora : int
- lectorMisiones : LectorMisiones
Detective
- obtenerdorDePista : ObtenedorDePistasMedias
- kmPorHora : int
- lectorMisiones : LectorMisiones
Investigador
- obtenerdorDePista : ObtenedorDePistasMedias
- kmPorHora : int
- lectorMisiones : LectorMisiones
Sargento
- obtenerdorDePista : ObtenedorDePistasDificiles
- kmPorHora : int
- lectorMisiones : LectorMisiones
<<interface>>
Rango
+pedirPista(edificio : Edificio, conMision : Mision) : String
+calculaTiempoViajeA(distancia : double) : double
+proximoRango() : Rango
+asignarMision(rutaArchivoCiudades : String, registroLadrones : RegistroLadrones) : Mision
1
<<interface>>
ObtenedorDePistas
+pedirPista(Edificio : Edificio, conMision : Mision) : String
ObtenedorDePistasFaciles
ObtenedorDePistasMedias
ObtenedorDePistasDificiles

4.3. Generador de Misiones



4.4. Ladron



5. Excepciones

Todas las ciudades deben tener un aeropeurto/puerto, banco/bolsa y biblioteca.

FileNotFoundException: Se levanta cuando al intentar leer un archivo este no se encuentra.

IOException: Se levanta cuando al intentar leer un archivo la entrada y salida es interrumpida.

ParseException: Se levanta cuando al intentar leer un archivo hay un error de parseo.

LadronNoEncontradoException: Se levanta cuando al intentar leer un archivo el ladron de la mision no estaba en el archivo de ladrones.

IllegalArgumentException: Se levanta cuando al intentar leer un archivo de misiones este tiene una sola mision.

6. Diagramas de secuencia

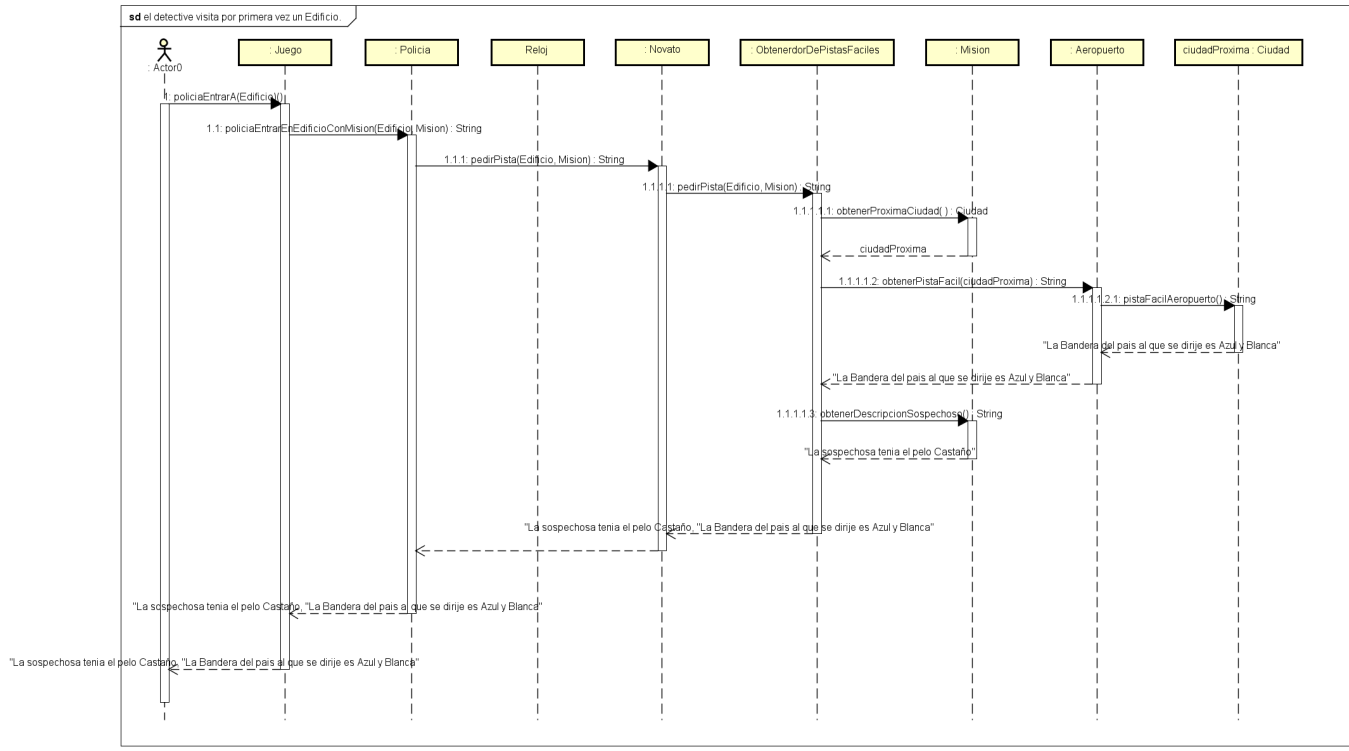
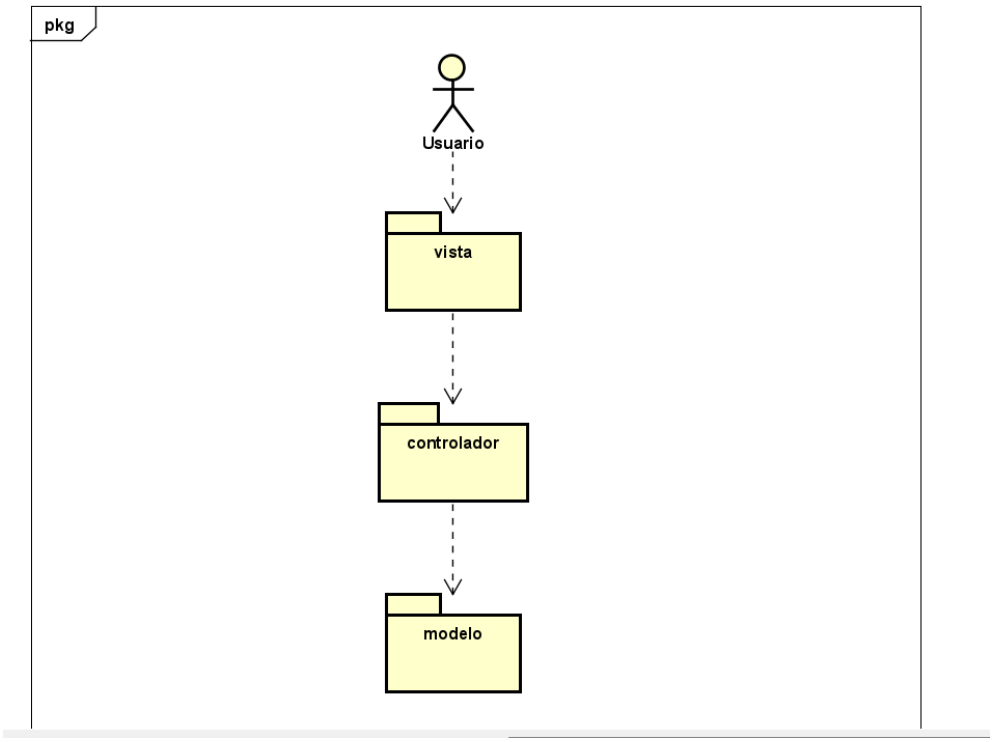


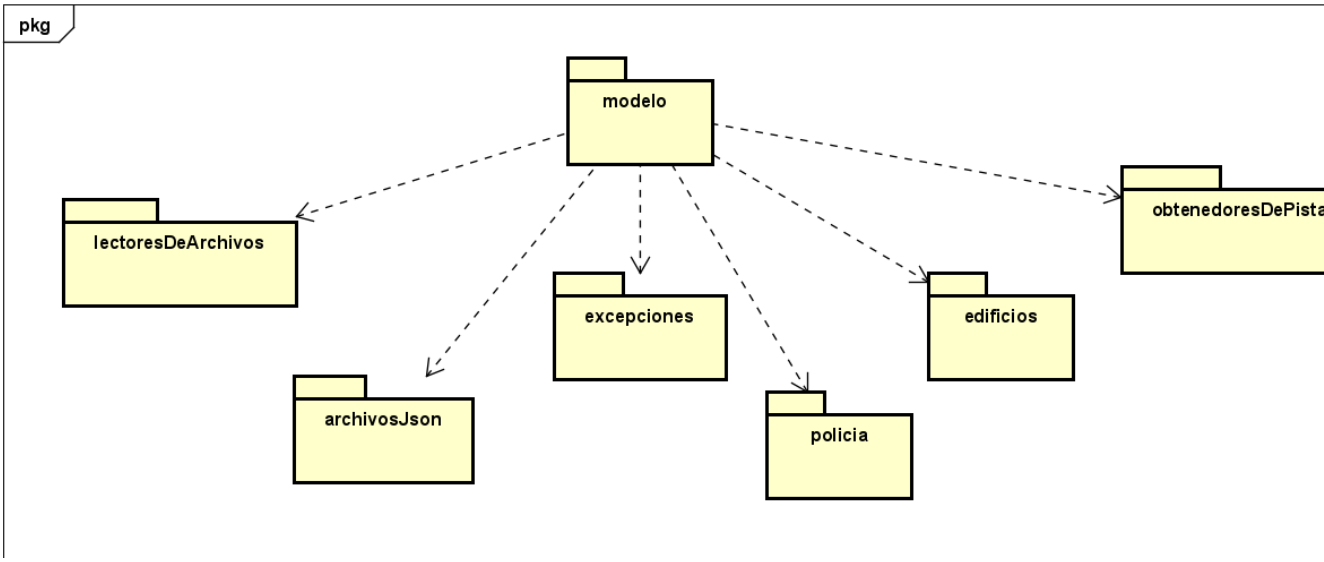
Figura 1: El detective visita por primera vez un edificio

7. Diagramas de Paquetes

El siguiente diagrama muestra la relación general de los paquetes del programa. El usuario interactúa completamente con las vistas y estas, junto con los controladores, se encargan del flujo general del modelo.

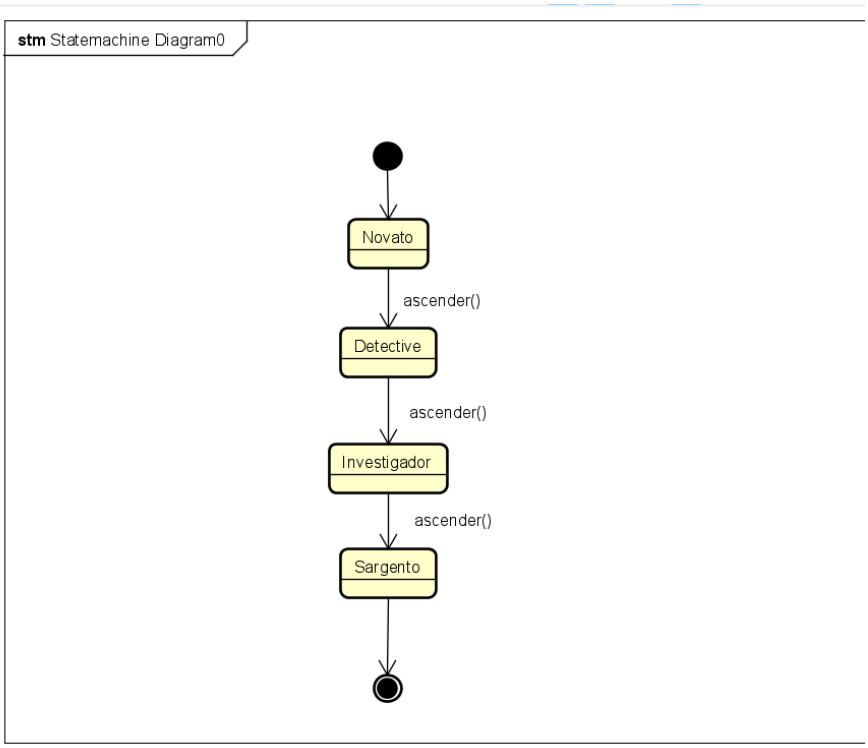


El siguiente diagrama muestra los paquetes en modelo.



8. Diagrama de Estados

El siguiente diagrama muestra los cambios de Rango del Policia. Comienza por el rango *Novato*, al recibir el método *ascender()* pasa al siuguiente estado que es *Detective*. De este último, pasa al rango *Investigador* para finalizar en *Sargento*.



https://github.com/jmdieguez/algothief.