**FIUBA - 75.07**

**Algoritmos y programación III**

*Trabajo práctico 2: AlgoCity*

2do cuatrimestre, 2014

(trabajo grupal)

Alumnos:

|  |  |  |
| --- | --- | --- |
| Nombre | Padrón | Mail |
| Martin Queija | 96455 | tinqueija@gmail.com |
| Federico Axelrud | 94395 | fedeaxelrud@hotmail.com |

***Fecha de entrega final***: Jueves 04/12/2014

***Tutor***: Diego Marcet

***Nota Final***:

# 

Informe

**Supuestos**

Para realizar el trabajo práctico en tiempo y forma debimos tomar ciertas suposiciones. La principal suposición fue la que tomamos en base a decidir como incrementa la población. Para que el mapa incremente en población, el usuario debe contar si o si con viviendas residenciales en el mapa. Sumado a esto la validación que requieren las residencias para aumentar en población es la siguiente. Cada residencia debe cumplir las siguientes premisas:

1) Debe tener una ruta situada a no mas de 3 lotes de distancia.

2) Debe tener conexión de agua.

3) Debe tener conexión eléctrica.

4) Debe tener una industria cercana (al rededor de 14 lotes de distancia).

5) Debe tener un comercio cercano (al rededor de 6 lotes de distancia).

Las catástrofes solo pueden ocurrir solo una por turno. Aleatoriamente se elige si es Godzilla, un terremoto o nada.

El daño que ocasiona Godzilla a la estación de bomberos y al pozo de agua no estaba especificado, por ende se supuso que daña en un 100% a ambas estructuras.

Las líneas de tensión no ocupan un lote, por lo tanto puede haber una construcción y una línea de tensión a la vez en un mismo lote

Otra suposicion es que el mapa se crea estatico y es invariante a lo largo de todo el transcurso del juego. Descartando por completo otros arreglos de superficie y posibles catastrofes tales como inundaciones. Sin embargo, dada la abstraccion que impusimos, desarollar tales eventos no seria de gran dificultad pero si fuera del foco del trabajo práctico.

Las residencias se pueden construir aunque no se cumplan las condiciones para que se genere la poblacion dentro de ellas. Una vez que la residencia cumple con todas las condiciones se acredita la poblacion. Ante el faltante de cualquiera de las condiciones, la población desaparece automaticamente.

## Modelo de dominio

El principal componente del trabajo práctico son los Lotes. Estos se encuentran en un arreglo cuadrado dentro del Mapa. Cada lote puede contener hasta una construcción, una línea de tensión y una tubería. El mapa tiene grandes responsabilidades por sobre los lotes; se encarga de obtenerlos, dañarlos y mantenerlos actualizados con el ritmo de la ejecución.

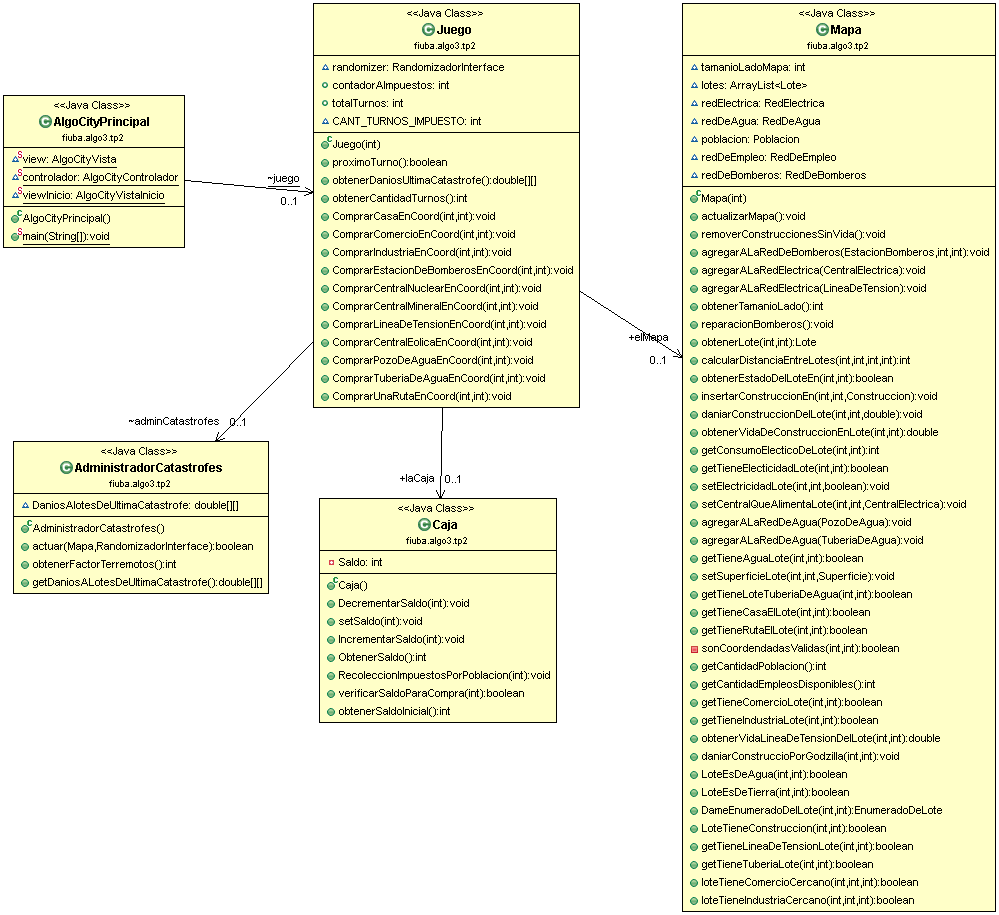
Por otro lado hay una clase Construcción de la cual heredan los 9 diferentes tipos de construcciones. Cada construcción posee una vida fijada en 100 unidades. Cada construcción sabe sobre que superficie puede construirse y cuanto cuesta. De este modo a la hora de construir se le pregunta a la construcción a insertar si puede construirse ahí y se le pregunta cuanto cuesta para restar correctamente el costo de la caja.

Dentro de los diferentes tipos de construcciones están las viviendas residenciales. Estas viviendas pueden almacenar hasta 100 residentes, los cuales después de una determinada cantidad de turnos pagan un impuesto de $10 cada uno.

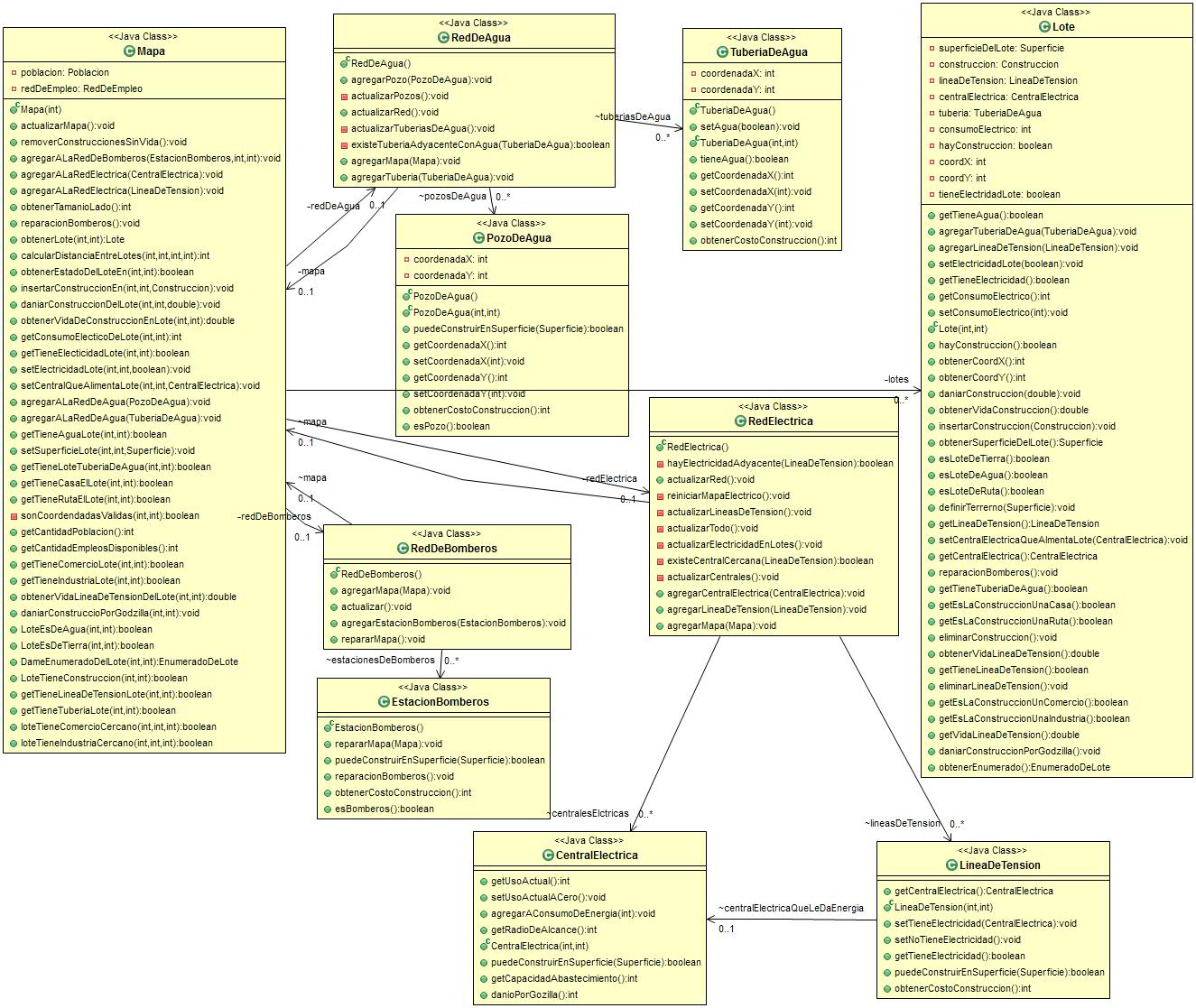
El juego funciona por turnos. Durante cada turno el jugador puede comprar todas las construcciones que el desee, siempre y cuando le alcance el saldo. Una vez que se avanza al siguiente turno, se llama a actuar al administrador de catástrofes, se actualiza el mapa para ver los daños y luego se llaman a actuar a los bomberos.

Diagramas de clases

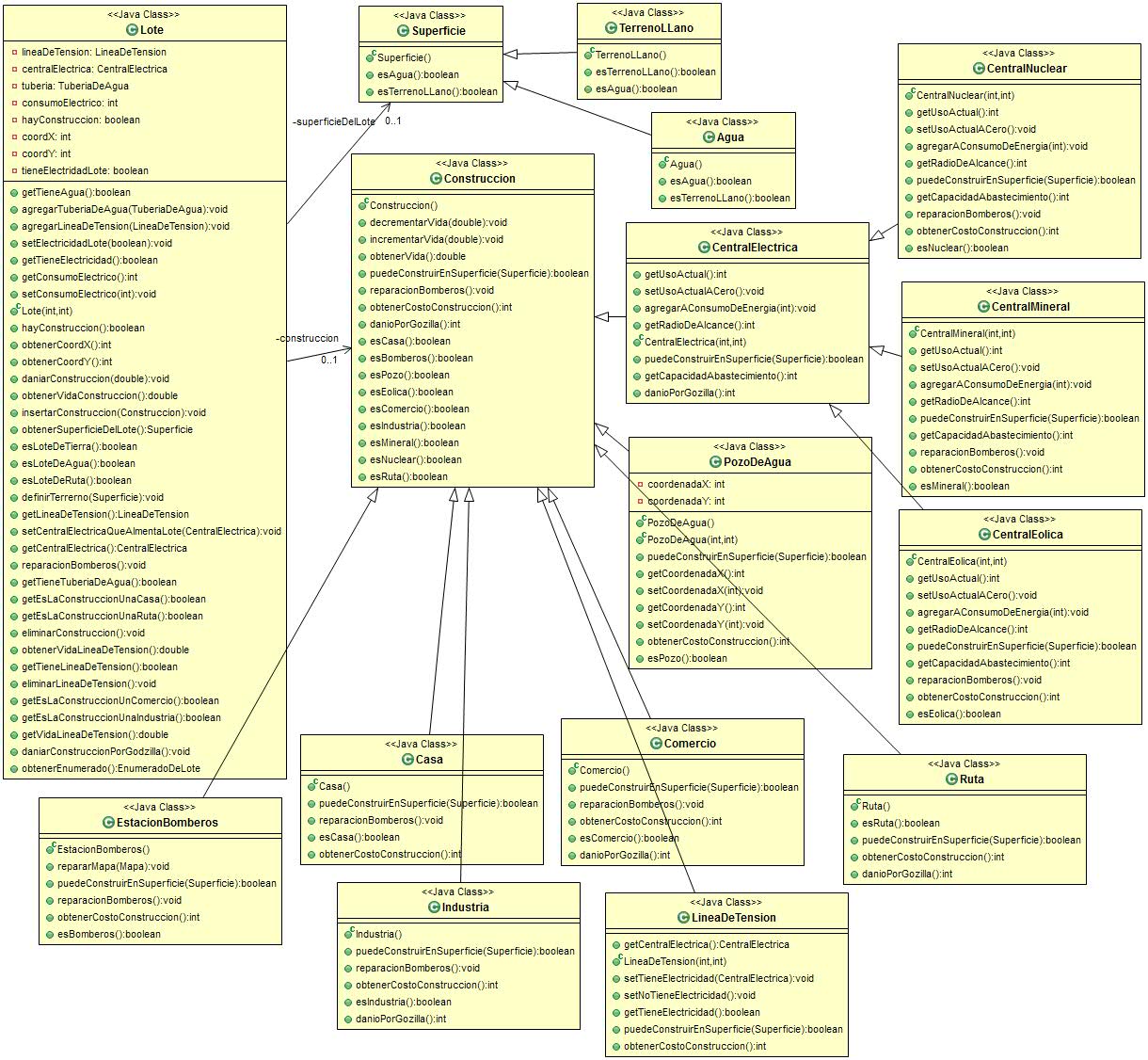
1)Diagrama general del modelo



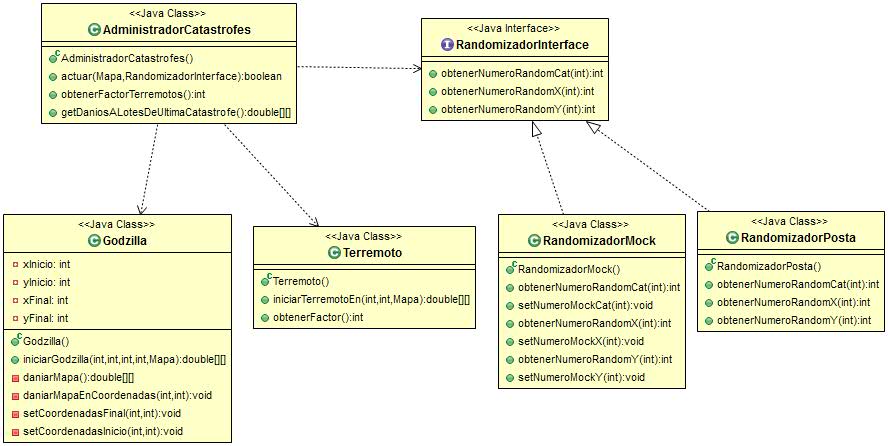
2)Diagrama de los componentes del mapa



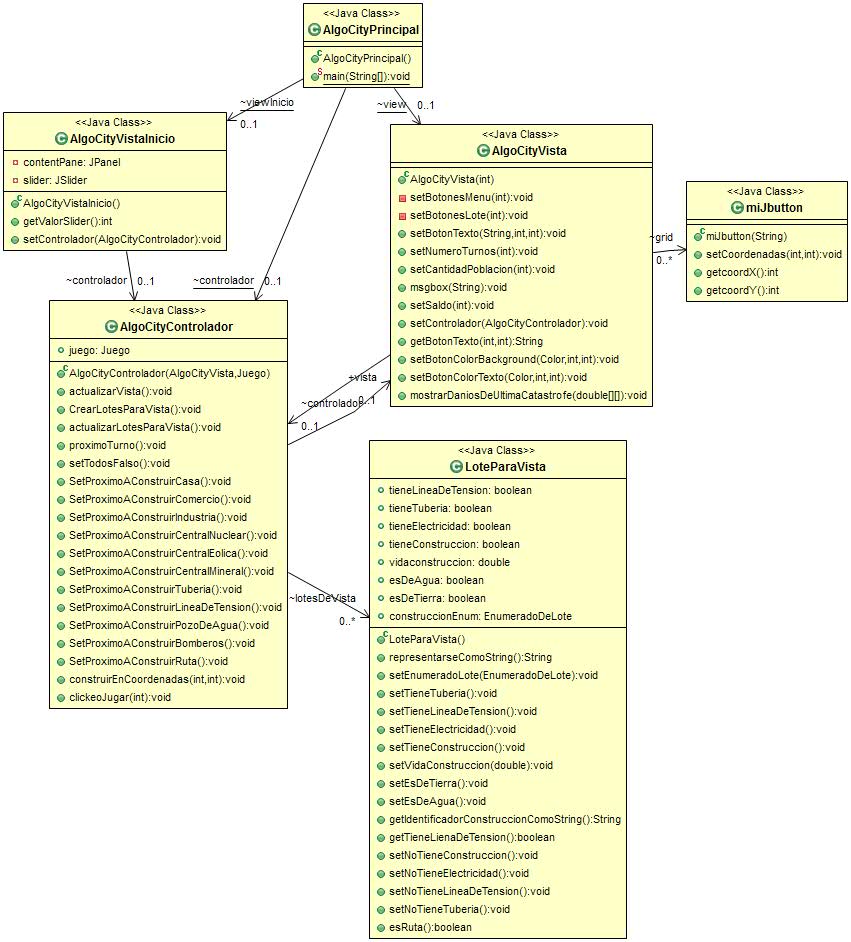
3)Diagrama de lote



4)Diagrama del administrador de catástrofes



5)Diagrama del controlador, de la vista y del modelo



Diagramas de secuencia

Diagrama de secuencia que representa al usuario haciendo click en Próximo Turno:

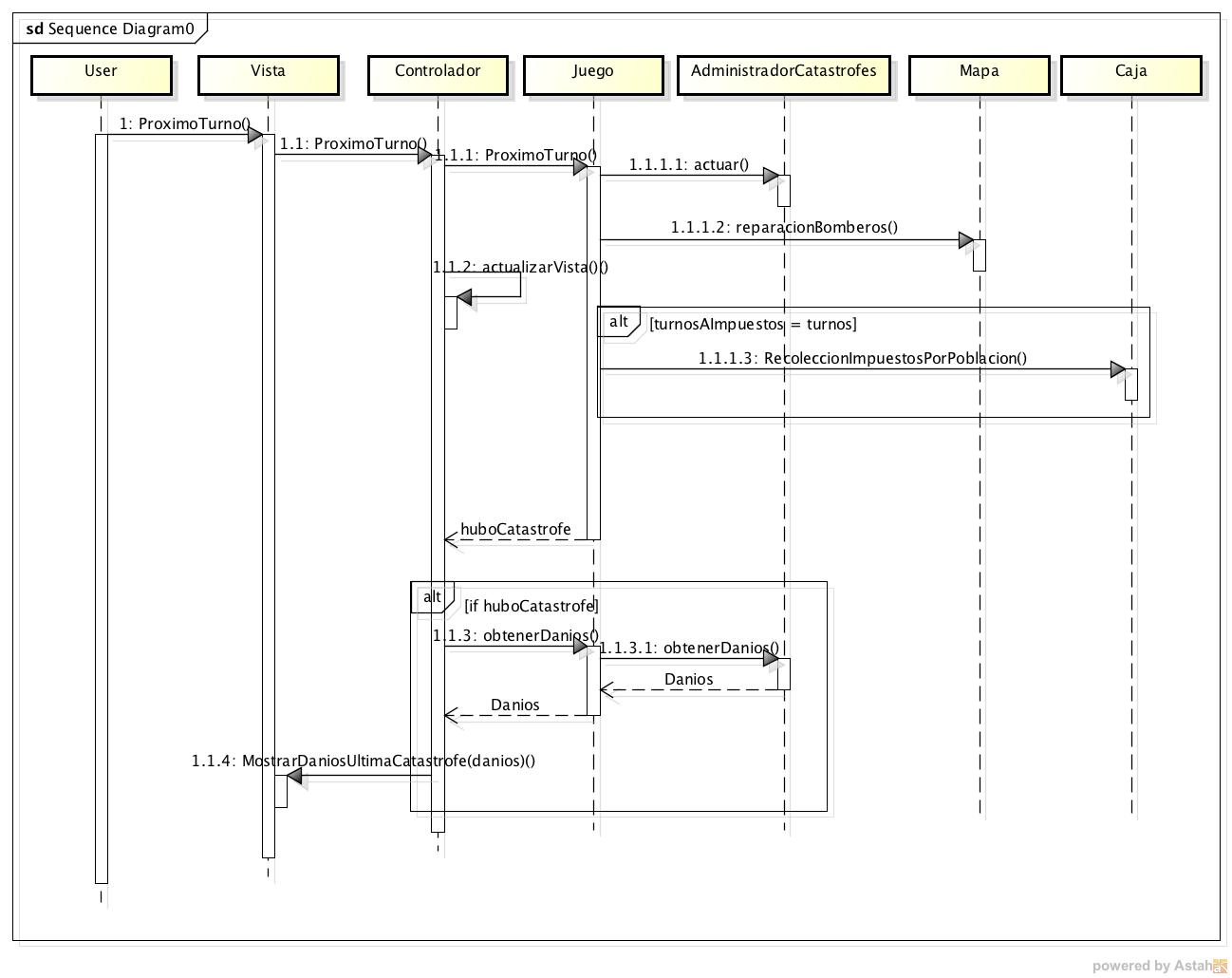


Diagrama de como el controlador actualiza la vista:

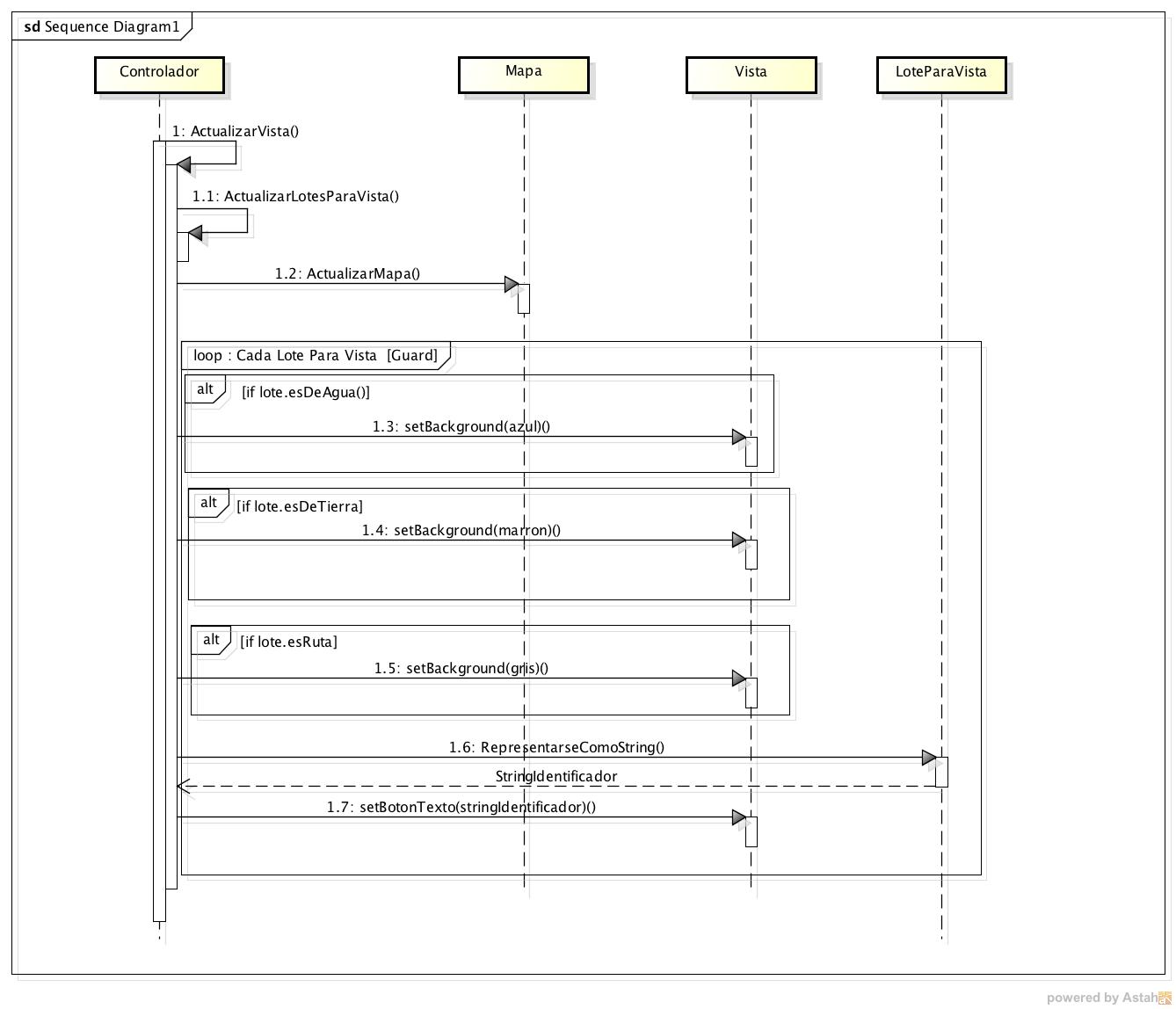


Diagrama de como se actualiza el mapa:

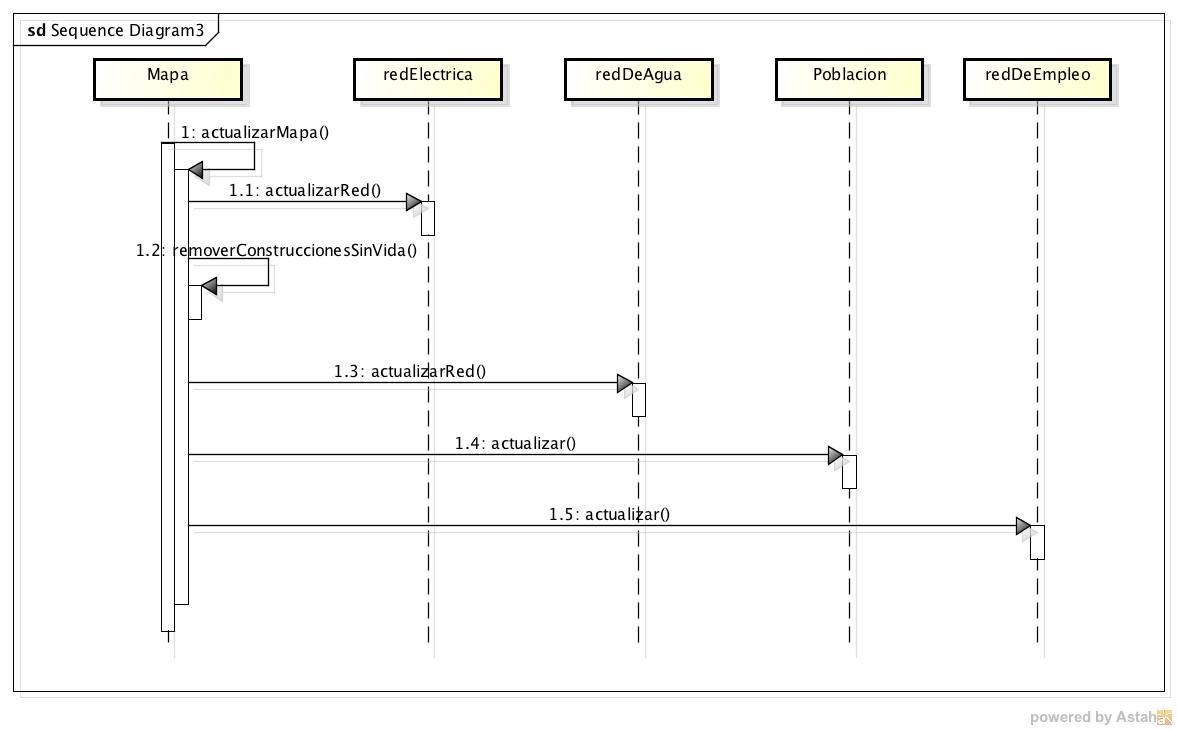


Diagrama que ejemplifica como se compra una construcción sobre cualquier lote:

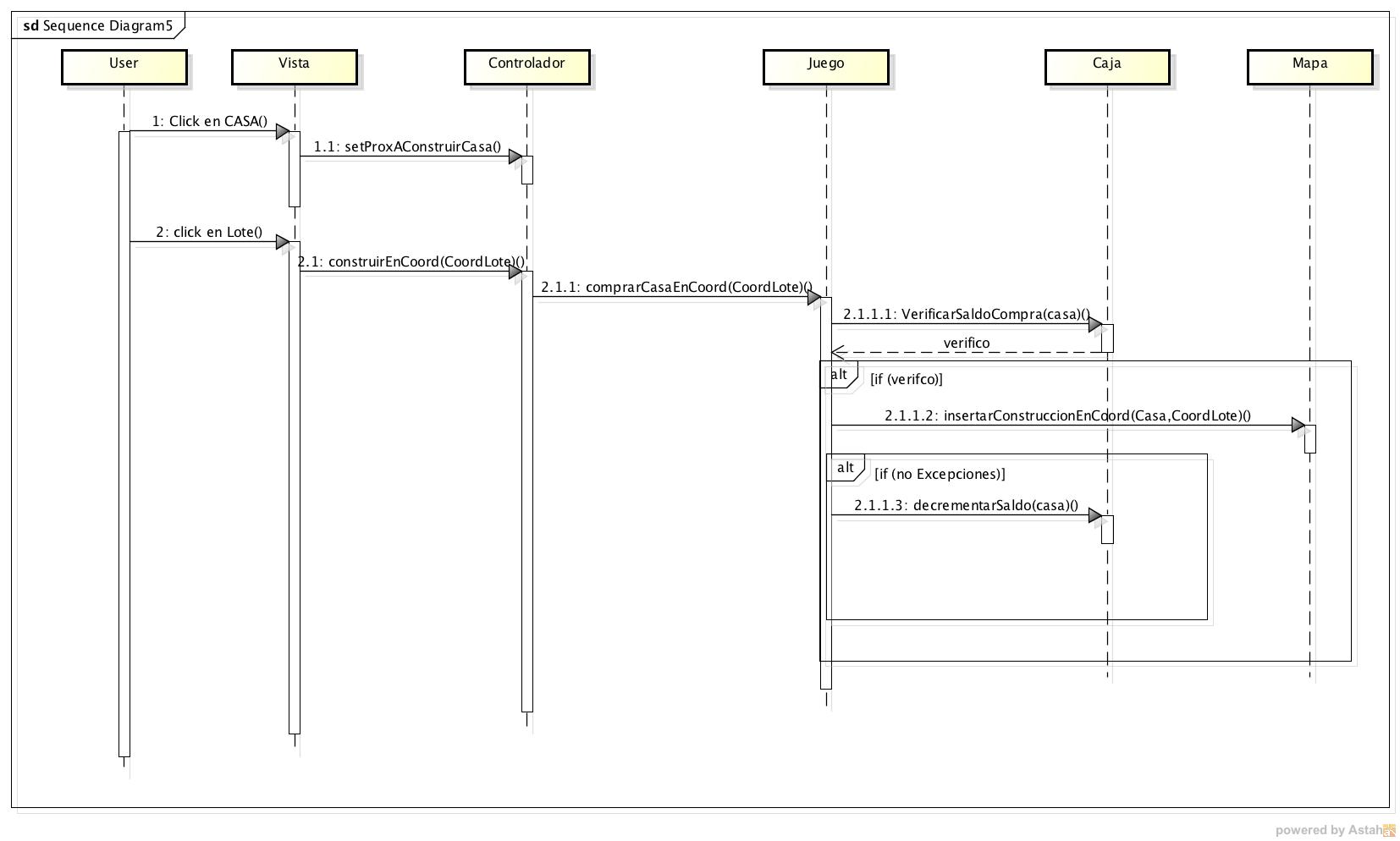
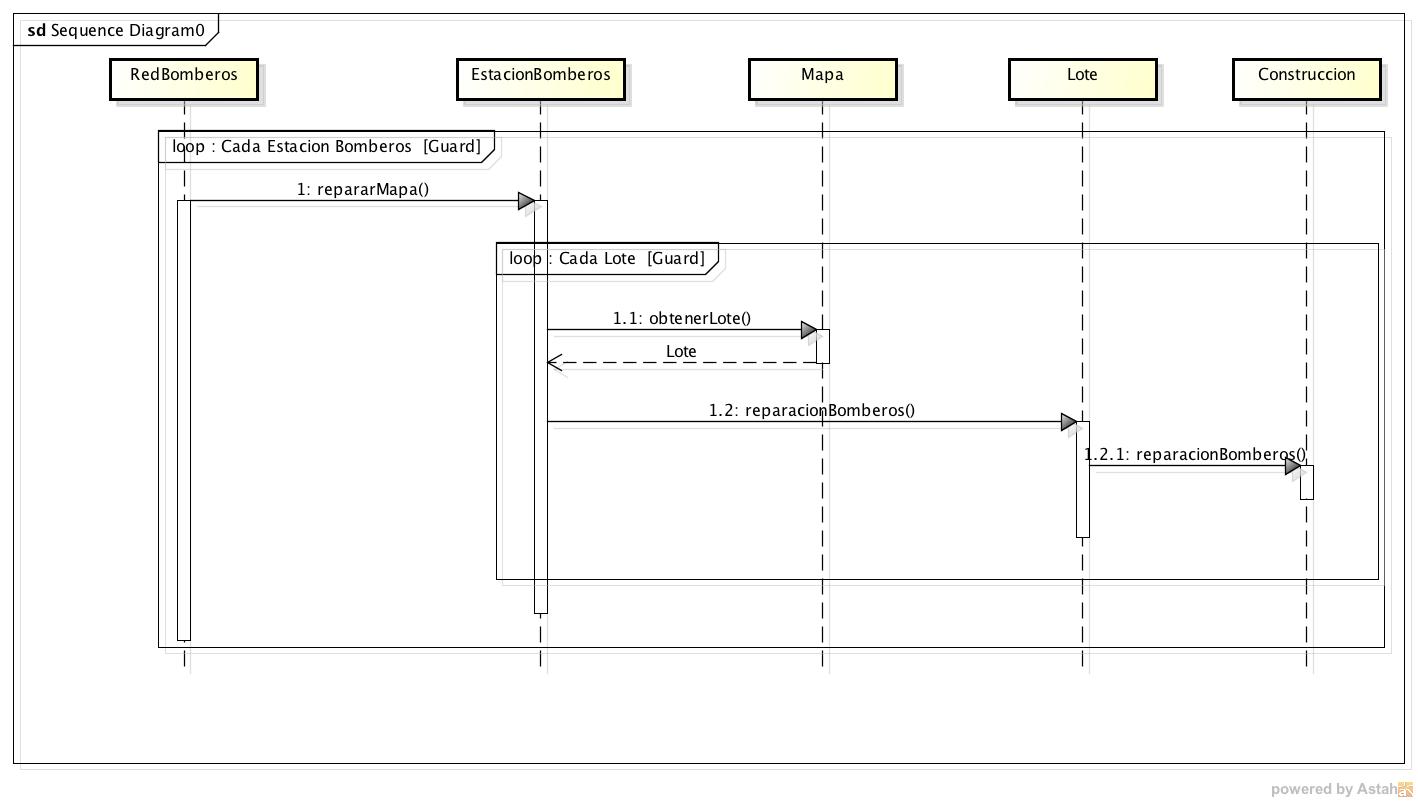


Diagrama de como los bomberos reparan el mapa:



Detalles de implementación

Con el fin de conseguir un buen trabajo practico y aplicar lo aprendido en clase desarollamos la aplicacion haciendo uso de las practicas de MVC. Esta practica, que divide la aplicacion en tres partes: Modelo, vista y controlador, requiere gran desacoplamiento entre la vista y el modelo. Para lograr esto, la vista tiene sus propios objetos Lote, que se dan en relacion 1 a 1 con los objetos Lote del modelo. El controlador se encarga de que esten siempre actualizados en los momentos necesarios y la vista se encarga de representar graficamente sus objetos Lote.

Terremoto: esta catastrofe define un epicentro dentro del mapa de manera aleatoria y daña a todos los lotes del mapa por un factor multiplicado inversamente por la distancia. De esta manera hace mayor y total daño en el epicentro y luego el daño cae de acuerdo al factor deseado.

Administrador de Catastrofes: Este se encarga de manejar al terremoto y a Godzilla. Cada turno crea un numero aleatorio y lo compara con otro. Dependiendo que numero sea dispara un terremoto o libera a godzilla. En ambos casos el administrador de catastrofes devuelve una grilla del tamaño del mapa con los daños hechos a cada lote, para que luego el controlador pueda representarlo en la vista.

Red Electrica:Es la clase que se encarga de manejar la electricidad de todos los lotes. Esta compuesta por un conjunto de centrales electricas y un conjunto de lineas de tension. Esta clase es la que le indica a cada lote si tienen luz. Para ello se fija que tengan un centrales electrica cercana con consumo disponible para alimentarla dependiendo de la construccion que posee, o que halla una linea electrica que una a una central con ese lote.

Red de Agua: Esta conpuesto por un conjunto de pozos de agua y por un conjunto de tuberias. Esta clase se ocupa de indicarle a cada lote si tiene o no agua, fijandose que halla un camino de tuberias que unen al lote con alguno de los posos de agua.

Excepciones

Las excepciones, tal como lo indica el nombre, fueron utilizadas para tratar situaciones excepcionales. En nuestro trabajo practico definimos para utilizar 6 clases de excepciones. Estas son:

* ExcepcionCoordenadasInvalidas.
  + Esta excepcion se utiliza siempre que se utilizen par de coordenadas en el codigo. Verifica siempre que las coordendas solocitadas (ya se para construir, destruir o consultar estado) sean validas dentro del mapa que estamos utilizando (el tamaño puede variar de acuerdo a las variables de inicializacion).
* ExcepcionLoteYaContieneConstruccion.
  + Esta excepcion es lanzada siempre que se quiera intentar construir una construccion sobre un lote que ya contenga una. Esta excepciones capturada a mas alto nivel para evitar que el usuario compre dos propiedades en el mismo lote.
* ExcepcionLoteYaContieneLineaDeTension.
  + Al igual que con construccion, esta exepcion se lanza cuando se intenta agregar a un lote una linea de tension cuando el lote ya tiene una linea de tension. Se captura para evitar el descuento de dinero sobre el saldo del jugador.
* ExcepcionLoteYaContieneTuberia.
  + Idem anterior.
* ExcepcionSinSaldo.
  + Como las mayoria de las excepciones, el nombre indica cuando se lanza. Se lanza cuando el monto a pagar es superior al saldo del jugador y por consiguiente no se realiza el descuento de dinero ni se acredita la construccion que se quiera comprar.
* ExcepcionSuperficieInvalida.
  + Excepcion que se lanza cuando el jugador intenta construir una construccion sobre una superficie no permitida, ya sea por ejemplo un pozo de agua sobre tierra firme o una residencia sobre agua.

# **Checklist de corrección**

Esta sección es para uso exclusivo de los docentes, por favor no modificar.

## Carpeta

**Generalidades**

* ¿Son correctos los supuestos y extensiones?
* ¿Es prolija la presentación? (hojas del mismo tamaño, numeradas y con tipografía uniforme)

**Modelo**

* ¿Está completo?¿Contempla la totalidad del problema?
* ¿Respeta encapsulamiento?
* ¿Hace un buen uso de excepciones?
* ¿Utiliza polimorfismo en las situaciones esperadas?

## Diagramas

**Diagrama de clases**

* ¿Está completo?
* ¿Está bien utilizada la notación?

**Diagramas de secuencia**

* ¿Está completo?
* ¿Es consistente con el diagrama de clases?
* ¿Está bien utilizada la notación?

**Diagrama de estados**

* ¿Está completo?
* ¿Está bien utilizada la notación?

# **Código**

**Generalidades**

* ¿Respeta estándares de codificación?
* ¿Está correctamente documentado?